TOUCHPAD
Computer Textbook Series

12
CBSE

Computer Science with PYTHON

(Focus on Modularity & Documentation)

Naveen Kumar
Professor
Dept. of Computer Sc.
University of Delhi

Ankit Rajpal
Assistant Professor
Dept. of Computer Sc.
University of Delhi

Sheetal Rajpal
Assistant Professor
Dyal Singh College
Dept. of Computer Sc.
University of Delhi

Mohini Arora
HOD (Computer Sc.)
Air Force Golden Jubilee
Institute, New Delhi

ORANGE

Published by:

Orange Education

An ISO 9001: 2015 Company
9, Daryaganj
New Delhi-110002

Phone: 43776600
8588814859

Email: info@orangeeducation.in
Website: www.orangeeducation.in
IE Code: 0511063121

Branches:

• Chennai • Guwahati

Regional Offices:

• Ahmedabad • Bengaluru • Bhopal • Bhubaneswar
• Dehradun • Hyderabad • Jaipur • Kochi • Kolkata
• Lucknow • Mumbai • Patna • Raipur • Ranchi

Printed at:

© Publishers

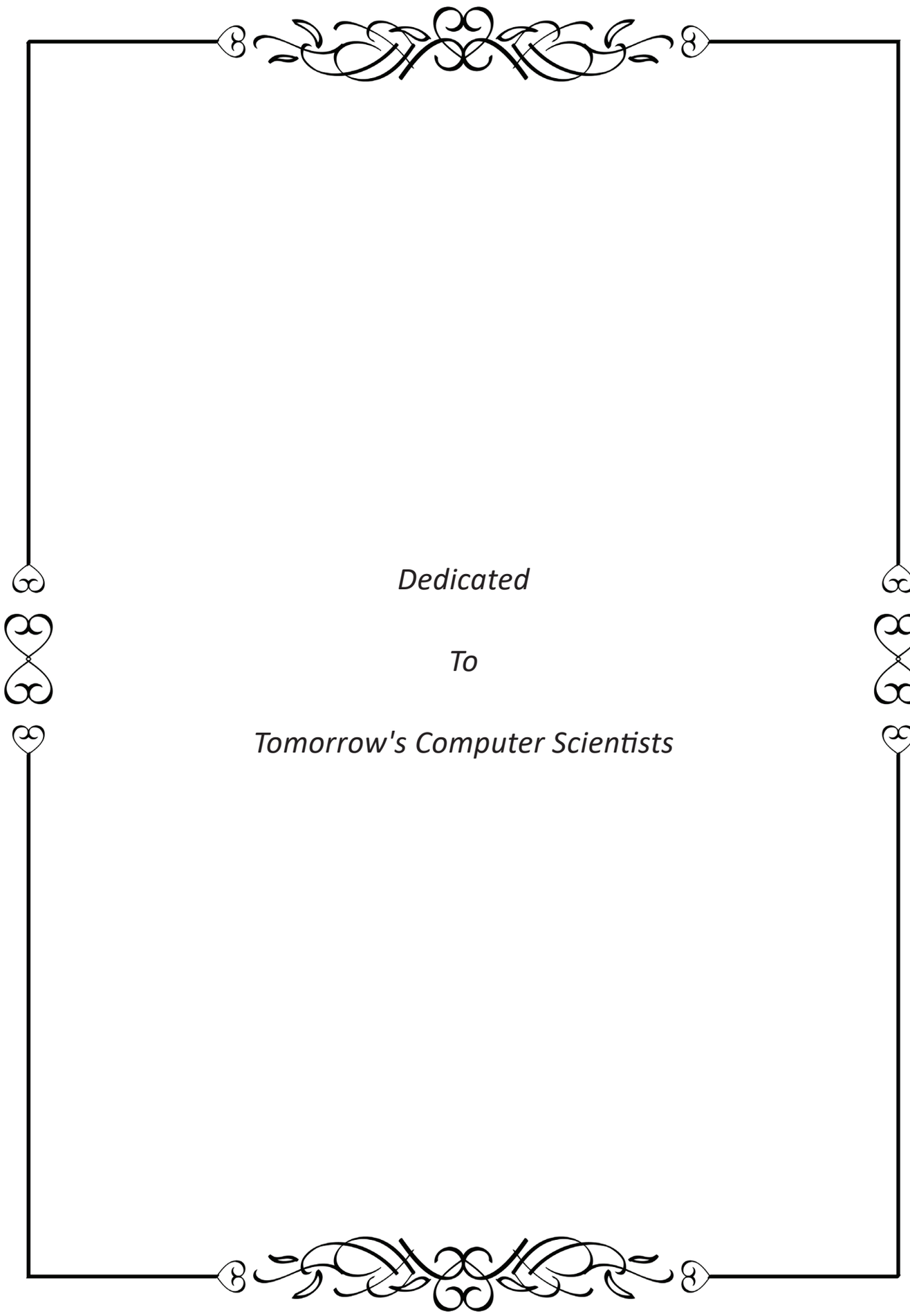
No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher.

Disclaimer

All product names, brand names and product logos mentioned or shown in this book are trademarks, registered trademarks, or trade names of their respective owners. The reproduction of these product names, brand names, and product logos and all instances of references of proprietary software in this book, are for educational purposes only.

Although every safety measure has been taken to verify the precision of the information contained herein, the author(s) and publisher assume no responsibility for any error or omission. No liability is assumed for damages that may result from the use of information contained within.

Printed and Distributed by Orange House Pvt Ltd



Dedicated

To

Tomorrow's Computer Scientists

PREFACE

In class XI, we learned the basics of computer systems and core concepts of programming with a focus on modularity and documentation and concluded with societal issues in computing. In class XII, we devote the first chapter (of the second volume) of the book to reviewing the programming concepts learnt in class XI. As developing modular code is a key aspect of successful IT projects, in this chapter, we include a detailed discussion on developing and using the functions. In the next chapter, we discuss some advanced concepts related to modularity, namely, local and global variables and their scope, including built-in and imported names. So far, we have stored all data in the computer's memory. However, while dealing with large volumes of data, we must keep the data and the results permanently in the files. In Chapter 3, we deal with file-handling concepts, including a discussion on how to work with text, binary, and CSV files. While executing a program, we often encounter exception events like division by zero or attempting to open a file which is already deleted. In Chapter 4, we give several examples that provide useful insights into exception handling. Structuring data for an application is a crucial aspect of programming. In Chapter 5, we introduce stack—a useful data structure and describe its application in evaluating an arithmetic expression.

Modern computing devices, such as smartphones, are seamlessly interconnected through local networks and the Internet. Encompassing Chapters 6 through 8, we embark on a journey to unravel the intricate realm of computer networking. This exploration spans a spectrum of essential topics, including transmission media, encompassing wired and wireless technologies. Along the way, we will encounter diverse network devices, explore various network topologies and protocols, and unveil the world of web services, including HTTP, FTP, and the robust framework of TCP/IP.

Real-life IT applications must necessarily deal with managing huge volumes of data. Database Management Systems (DBMS) enable the sharing of data amongst different users (database managers, application developers, and end users) while also addressing the security and privacy concerns of all users. DBMS being such a crucial component of IT applications, we devote four chapters (Chapters 9–12) to studying databases. We will begin by introducing key database concepts like the relational data model, keys, and constraints. We will learn Structured Query Language (SQL) to define the relevant constraints on data and store and retrieve it. Finally, we will learn how Python connects to SQL databases.

The book concludes with the following appendixes: a guide for installing Anaconda—the most popular environment for developing AI applications, a sample project, some solved and unsolved practical lab questions, some viva questions, and one solved and one unsolved question paper for practice. Finally, we provide a glossary of terms used in the book and an alphabetical list of topics and terms that appear in the book, along with the page numbers, in the form of an index.

ACKNOWLEDGEMENTS

We are grateful to Drashti Varshney and Nimish Kumar, undergraduate students at Dyal Singh College of the University of Delhi, Mr. P.K. Hazra, Dr. Manoj Agarwal and Dr. Neeraj Kumar Sharma, colleagues at the University of Delhi for reviewing different sections of the book.

We are grateful to the Orange Book team, especially Ms. Nidhi Gupta, who was readily available for help during the book's preparation. Last but not least, we would like to express our gratitude to our family members for their support and patience.

Naveen Kumar

Ankit Rajpal

Sheetal Rajpal

Mohini Arora

About the Authors



Naveen Kumar

Naveen Kumar is a Professor at Department of Computer Science, University of Delhi, Delhi, India. He holds a Ph.D. degree in Computer Science from IIT, Delhi. Prior to his doctoral studies, he earned his M.Tech. (Computer Science) and M.Sc. (Mathematics) degrees from IIT Delhi, and an honours degree in Mathematics from University of Delhi. He has taught a varied array of courses to undergraduate and postgraduate students and has teaching experience of about 40 years. He has co-authored a book with Sheetal Taneja titled *Python Programming, A Modular Approach,* published by Pearson Education. He has supervised many Ph.D. students in areas such as soft computing and social networks. His current areas of interest include explainable AI, machine learning, and deep learning. He has contributed several articles in national and international journals and conferences. He has been a member of the curriculum design committees of various universities and the CBSE. He has also been an organizing chair/program committee member of several workshops, conferences, and symposia.



Ankit Rajpal

Ankit Rajpal is an Assistant Professor in the Department of Computer Science at the University of Delhi since November 2020. He has successfully completed some research projects. He has a patent for a video watermarking tool developed for MPEG-4 AVC videos. He has a PhD in image and video watermarking from the Department of Computer Science, University of Delhi. His research interests include machine learning, explainable AI, watermarking, and data mining. He has published several papers in reputed international journals and conference proceedings. He obtained his M.Sc. and B.Sc. Honours degrees in Computer Science from the University of Delhi. He possesses around 11 years of teaching experience at the undergraduate level. He has been actively involved in curriculum design for undergraduate and postgraduate courses.



Sheetal Rajpal

Sheetal Rajpal is working as an Assistant Professor at Department of Computer Science, Dyal Singh College, Department of Computer Science, University of Delhi, Delhi, India. She earned her B.Sc. (Honours), M.Sc. and Ph.D. degrees in Computer Science from University of Delhi. Being the topper in the M.Sc. programme, she was awarded a gold medal by University of Delhi. Her research interests include machine learning, explainable AI, deep learning, and bioinformatics. She has authored several research articles in international journals. She possesses around 10 years of teaching experience at the undergraduate level at university of Delhi. She has been actively involved in curriculum design at University of Delhi.



Mohini Arora

Ms Mohini Arora is an experienced Computer Science Specialist with a demonstrated history of contributions in education, educational administration, strategic planning, and training. She is presently working as HOD (Computer Science) in Air Force Golden Jubilee Institute, Subroto Park, New Delhi. She has authored and reviewed several books, and conducted workshops in Computer Science, and Information and Communication Technology for CBSE, NIOS, and NCERT, for both academic and vocational courses.

SYLLABUS

CLASS-XII

COMPUTER SCIENCE [CODE-083]

1. Prerequisites

Computer Science – Class XI

2. Learning Outcomes

Student should be able to:

a. Apply the concept of function.
b. Explain and use the concept of file handling.
c. Use basic data structure: Stacks
d. Explain basics of computer networks.
e. Use Database concepts, SQL along with connectivity between Python and SQL.

3. Distribution of Marks:

Unit No.	Unit Name	Marks	Periods	
			Theory	Practical
I.	Computational Thinking and Programming–2	40	70	50
II.	Computer Networks	10	15	-
III.	Database Management	20	25	20
	Total	**70**	**110**	**70**

4. Unit wise Syllabus

Unit I: Computational Thinking and Programming–2

- Revision of Python topics covered in Class XI.
- Functions: types of function (built-in functions, functions defined in module, user defined functions), creating user defined function, arguments and parameters, default parameters, positional parameters, function returning value(s), flow of execution, scope of a variable (global scope, local scope)
- Exception Handling: Introduction, handling exceptions using try-except-finally blocks
- Introduction to files, types of files (Text file, Binary file, CSV file), relative and absolute paths
- Text file: opening a text file, text file open modes (r, r+, w, w+, a, a+), closing a text file, opening a file using with clause, writing/appending data to a text file using write() and writelines(), reading from a text file using read(), readline() and readlines(), seek and tell methods, manipulation of data in a text file.
- Binary file: basic operations on a binary file: open using file open modes (rb, rb+, wb, wb+, ab, ab+), close a binary file, import pickle module, dump() and load() method, read, write/create, search, append and update operations in a binary file
- CSV file: import csv module, open/close csv file, write into a csv file using writer(), writerow(), writerows() and read from a csv file using reader()
- Data Structure: Stack, operations on stack (push & pop), implementation of stack using list.

Unit II: Computer Networks

- Evolution of networking: introduction to computer networks, evolution of networking (ARPANET, NSFNET, INTERNET)
- Data communication terminologies: concept of communication, components of data communication (sender, receiver, message, communication media, protocols), measuring capacity of communication media (bandwidth, data transfer rate), IP address, switching techniques (Circuit switching, Packet switching)
- Transmission media: Wired communication media (Twisted pair cable, Co-axial cable, Fiber-optic cable), Wireless media (Radio waves, Micro waves, Infrared waves)
- Network devices (Modem, Ethernet card, RJ45, Repeater, Hub, Switch, Router, Gateway, WIFI card)
- Network topologies and Network types: types of networks (PAN, LAN, MAN, WAN), networking topologies (Bus, Star, Tree)
- Network protocol: HTTP, FTP, PPP, SMTP, TCP/IP, POP3, HTTPS, TELNET, VoIP
- Introduction to web services: WWW, Hyper Text Markup Language (HTML), Extensible Markup Language (XML), domain names, URL, website, web browser, web servers, web hosting

Unit III: Database Management

- Database concepts: introduction to database concepts and its need
- Relational data model: relation, attribute, tuple, domain, degree, cardinality, keys (candidate key, primary key, alternate key, foreign key)

- Structured Query Language: introduction, Data Definition Language and Data Manipulation Language, data type (char(n), varchar(n), int, float, date), constraints (not null, unique, primary key), create database, use database, show databases, drop database, show tables, create table, describe table, alter table (add and remove an attribute, add and remove primary key), drop table, insert, delete, select, operators (mathematical, relational and logical), aliasing, distinct clause, where clause, in, between, order by, meaning of null, is null, is not null, like, update command, delete command, aggregate functions (max, min, avg, sum, count), group by, having clause joins: cartesian product on two tables, equi-join and natural join
- Interface of python with an SQL database: connecting SQL with Python, performing insert, update, delete queries using cursor, display data by using connect(), cursor(), execute(), commit(), fetchone(), fetchall(), rowcount, creating database connectivity applications, use of %s format specifier or format() to perform queries.

5. Practical

S.No	Unit Name	Marks (Total=30)
1.	Lab Test: 1. Python program (60% logic + 20% documentation + 20% code quality)	8
	2. SQL Queries (4 queries based on one or two tables)	4
2.	Report file: • Minimum 15 Python programs. • SQL Queries—Minimum 5 sets using one table / two tables. • Minimum 4 programs based on Python-SQL connectivity	7
3.	Project (using concepts learnt in Classes 11 and 12)	8
4.	Viva voce	3

6. Suggested Practical List

Python Programming

- Read a text file line by line and display each word separated by a #.
- Read a text file and display the number of vowels/consonants/uppercase/lowercase characters in the file.
- Remove all the lines that contain the character 'a' in a file and write it to another file.
- Create a binary file with name and roll number. Search for a given roll number and display the name, if not found display appropriate message.
- Create a binary file with roll number, name and marks. Input a roll number and update the marks.
- Write a random number generator that generates random numbers between 1 and 6 (simulates a dice).
- Write a Python program to implement a stack using list.
- Create a CSV file by entering user-id and password, read and search the password for given userid.

Database Management

- Create a student table and insert data. Implement the following SQL commands on the student table:
  - ALTER table to add new attributes / modify data type / drop attribute
  - UPDATE table to modify data
  - ORDER By to display data in ascending / descending order
  - DELETE to remove tuple(s)
  - GROUP BY and find the min, max, sum, count and average
- Similar exercise may be framed for other cases.
- Integrate SQL with Python by importing suitable module.

7. Project

The aim of the class project is to create something that is tangible and useful using Python file handling/ Python-SQL connectivity. This should be done in groups of two to three students and should be started by students at least 6 months before the submission deadline. The aim here is to find a real world problem that is worthwhile to solve.

Students are encouraged to visit local businesses and ask them about the problems that they are facing. For example, if a business is finding it hard to create invoices for filing GST claims, then students can do a project that takes the raw data (list of transactions), groups the transactions by category, accounts for the GST tax rates, and creates invoices in the appropriate format. Students can be extremely creative here. They can use a wide variety of Python libraries to create user friendly applications such as games, software for their school, software for their disabled fellow students, and mobile applications, of course to do some of these projects, some additional learning is required; this should be encouraged. Students should know how to teach themselves.

The students should be sensitized to avoid plagiarism and violations of copyright issues while working on projects. Teachers should take necessary measures for this.

INSIDE THE SERIES

This is a quick shot to learn and remember the meaning of key terms.

This is note which provides additional information about the topic.

This is quick quiz to check what you have learnt.

This is coding task which contains a lab activity to use the concepts learnt and enhance practical skills.

Chapter Outline

Contains learning objectives of the chapter

Let's Summarise

Contains detailed summary of the chapter

Solved Exercises

Contains a variety of solved questions to enhance the intellectual and comprehension skills

Assessment

Contains a variety of unsolved questions to assess the intellectual and comprehension skills

Case-based Questions

Contains case-based questions to enhance creative and critical thinking capacities of students

Previous Years' Questions

Contains questions that were asked in previous years' board examinations to give an idea of the examination question paper

Assertion and Reasoning Based Questions

Contains assertion and reasoning based questions to develop logical reasoning skills of the students

NCERT Exercise Solutions

This section contains solutions of exercise given in the NCERT book.

CONTENTS

Unit I: Computational Thinking and Programming–2

1. Revision of Python **1**

1.1 Data Types
1.2 Mutable and Immutable Data Types
1.3 Variables and Assignment Statements
1.4 Keywords
1.5 Identifiers
1.6 Assigning Values to Several Variables
1.7 Operators
1.8 Functions
1.9 Built-in Functions
1.10 User-defined Functions
1.11 Types of Errors
1.12 Control Structures
1.13 Strings
1.14 Lists
1.15 Tuples
1.16 Dictionaries

2. Local and Global Scope **93**

2.1 Objects in Python
2.2 Updating Value of a Local Variable
2.3 Updating Value of a Global Variable
2.4 Introducing Global Variable in a Function
2.5 Built-in Names
2.6 Imported Names

3. Handling Files in Python **117**

3.1 Understanding Data Files
3.2 Opening and Closing Files
3.3 Reading and Writing Files
3.4 File Object Attributes
3.5 Standard Input/Output Devices as Files
3.6 Random Access Using seek() and tell()
3.7 Handling CSV Files in Python
3.8 File Processing Applications

4. Exception Handling **173**

4.1 Syntax Errors
4.2 Exceptions
4.3 Built-in Exceptions
4.4 Handling Exceptions using try...except
4.5 Handling Multiple Exceptions
4.6 Copying File Example
4.7 Finally Clause

5. Data Structures: Stacks **192**

5.1 Stacks
5.2 Applications of Stacks

Unit II: Computer Networks

6. Computer Networks **228**

6.1 Components of a Computer Network
6.2 Evolution of Networking
6.3 Switching Techniques
6.4 Data Communication Terminology
6.5 Data Transmission Modes
6.6 Transmission Media

6.7 Network Topologies
6.8 Network Types
6.9 Network Devices

7. Computer Networks: Protocols, Technologies, and Web Services **265**

7.1 TCP/IP
7.2 Mobile Telecommunication Technologies
7.3 Introduction to Web Services

8. Network Security **292**

8.1 Understanding Security Threats
8.2 Use of Cookies
8.3 Firewall
8.4 HTTPS
8.5 India IT Act
8.6 IPR Issues

Unit III: Database Management

9. Database Management System **308**

9.1 Introducing DBMS
9.2 Understanding Data Model

10. Structured Query Language (SQL) **331**

10.1 Structured Query Language (SQL)
10.2 Data Types
10.3 SQL Statements and Functions

11. SQL—Working with Multiple Tables **375**

11.1 Referential Integrity
11.2 Joins

12. Python Interface with MYSQL **410**

12.1 Connecting to MySQL Server from Python
12.2 A Complete Example
12.3 Suggestive Case Study
12.4 Database Update with Inputs from a CSV File
12.5 Verifying Database Updates

Installing Anaconda Python **453**

Practical **454**

Project **483**

Viva Voce Questions **488**

Practice Paper 1 **495**

Practice Paper 2 **505**

CBSE Sample Question Paper **512**

Glossary **520**

Index **525**

1 REVISION OF PYTHON

Chapter Outline

1.1 Data Types
1.2 Mutable and Immutable Data Types
1.3 Variables and Assignment Statements
1.4 Keywords
1.5 Identifiers
1.6 Assigning Values to Several Variables
1.7 Operators
1.8 Functions
1.9 Built-in Functions
1.10 User-defined Functions
1.11 Types of Errors
1.12 Control Structures
1.13 Strings
1.14 Lists
1.15 Tuples
1.16 Dictionaries

In class XI, we learnt the basic principles of Python programming using IDLE— Integrated Development and Learning Environment. We could do simple calculations using the IDLE shell and see the results quickly. Using the script mode of IDLE, we learnt to develop programs to solve simple problems. This chapter will review the Python programming concepts learnt in class XI. The following chapter will teach advanced concepts such as scope rules, data structures, file handling, and database connectivity.

Introduction

Python is an interactive and interpreted language. It was developed by Guido Van Rossum in 1991 at the National Research Institute for Mathematics and Computer Science in the Netherlands. Once you have the Python installed in your system, you may either use it via command line or Integrated Development Learning Environment (IDLE). IDLE has two main window types—the Shell window and the Editor window—giving us two modes to work – Interactive Mode (Fig 1.1) and Script Mode (Fig 1.2). In Fig 1.1, three greater-than symbols appearing adjacent to each other (>>>) form the IDLE prompt. A blinking cursor, following the IDLE prompt, indicates that the IDLE interpreter is ready to accept the commands from the user.

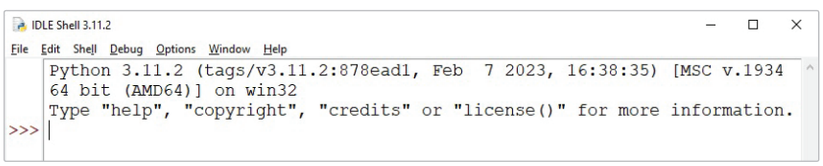


Fig 1.1: Showing Interactive Mode

To be able to save multiple instructions for future use in the form of a file (typically having a .py extension), we use the Python Editor. To use the IDLE editor, click the File → New File option from the Menu bar. A file containing Python instructions is called a Python script or a Python program, and working with Python scripts is called working in the script mode. After coding the .py file, you can simply execute it and see the result on screen.



Fig 1.2: Showing Script Mode

Python is a dynamically typed language. Variables, parameters, functions, and methods have no type declarations in the source code. When assigning a value to a variable in Python, we don't have to define the variable type. Let us revise this concept of dynamic typing using the following sequence of statements:

```
>>> a = 1
>>> print(a)
    1
>>> a = a + 4.0
>>> print(a)
    5.0
>>> a = int (a)
>>> print(a)
    5
>>> a = a * 'Hello'
>>> print(a)
    HelloHelloHelloHelloHello
```

The variable `a` is first initialized to `1` and `a` is dynamically typed as integer. On printing the value of `a`, it prints `1`. In the next expression, we add `4.0` to `a`, the resulting value is implicitly typecast to float, and `a` now gets `5.0` as the value. On printing the value of `a`, it prints `5.0`. This operation is referred to as implicit type conversion. In the next statement, the variable `a` is explicitly typecast to an integer, and the variable `a` is now assigned with its updated value, i.e. `5`. So, the value of `a` is now printed as `5`. This operation is known as explicit type conversion. Finally, when the variable `a` is multiplied with a string `'Hello'`, the variable is again implicitly typecast to a string comprising `'HelloHelloHelloHelloHello'` as its content and then assigned to the variable `a`.

1.1 Data Types

A data type refers to a type of data and the operations that can be performed on the objects of that type. For instance, the types `int, float,` and `str` correspond to integer, floating point, and string data. For example, `23, 12.75,` and `'Hello World'` are values of types `int, float,` and `str,` respectively. A data type in Python is also called a class. Thus, instead of saying types `int, float,` and `str,` we can also say classes `int, float,` and `str,` respectively. Thus, we would say that `23, 12.75, and 'Hello World'` are objects of (or instances of) the classes `int, float,` and `str,` respectively. Python allows different operations to be performed on different data types. We have already seen examples of operations associated with the abovementioned data types. Fig 1.3 shows some important data types available in Python.

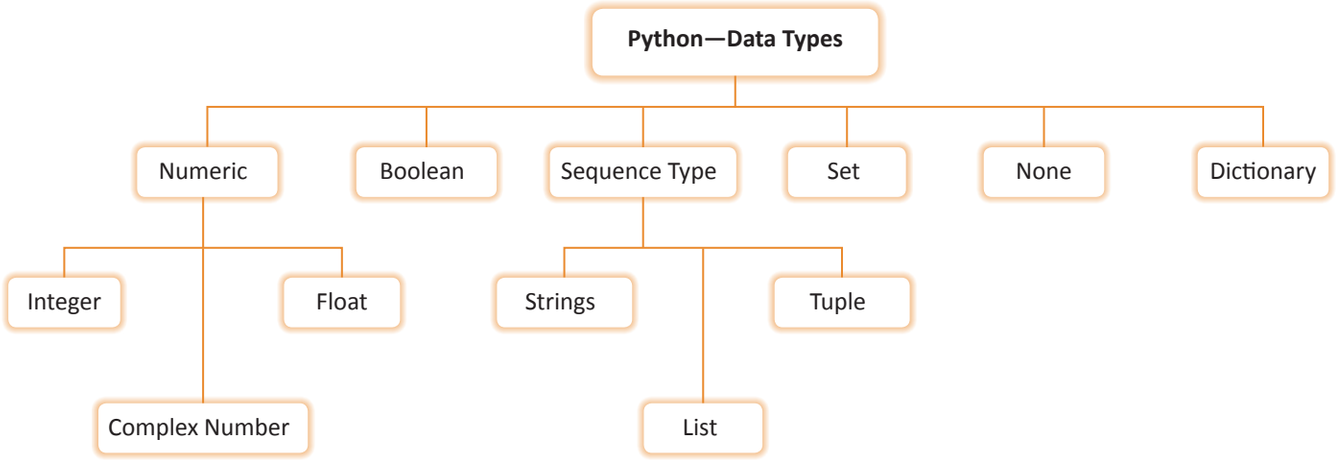


Fig 1.3: Displaying data types

1.1.1 Numeric

The numeric data types deal with numbers or numeric values. We can perform arithmetic operations on numeric types of data. The numeric types include integers, floating point numbers, and complex numbers.

`int` (integer)

The data type **`int`** includes integers. Literals of type **`int`** may be positive, negative, or zero. For example, `1`, `-29`, `78`, and `0` are values of the **`int`** data type. Interestingly, Python does not allow leading zeros. For example,

```
>>> 07
    SyntaxError: leading zeros in decimal integer literals are not permitted; use an 0o
    prefix for octal integers
```

However, Python does allow the use of octal and hexadecimal integers. An octal integer begins with 0o, followed by a sequence of digits. For example,

```
>>> 0o234
    156
>>> 0o1000
    512
>>> -0o234
    -156
>>> -(0o234)
    -156
```

A hexadecimal integer begins with 0x, followed by a sequence of digits. For example,

```
>>> 0x23
    35
>>> -0x128
    -296
```

`type()`

The function **`type()`** returns the type of an object.

Syntax

```
type(<name> | <value>)
```

Note that in the above description, a vertical bar denotes an option. Thus, we may like to fetch the type of object that a `name` refers to, or we may specify the object directly. For example,

```
>>> type(123)
    <class 'int'>
>>> type(0x123)
    <class 'int'>
```

```
>>> num = -56
    type(num)
    <class 'int'>
```

float (floating point numbers)

A **`float`** data type includes the floating point numbers. A floating point number typically (although not necessarily) includes a decimal point. For example, `10.5`, `-50.0`, `15.908` are floating-point numbers. Next, let us examine a floating point object `fNum`, constructed using a floating point number.

```
>>> fNum = 2*3.14*5
>>> type(fNum)
    <class 'float'>
```



- Python also allows scientific notation to express floating point numbers. In scientific notation, the symbol E denotes base 10 for the exponent. For example,

```
>>> 1.5E2
    150.0
```

- In the above example, 15.E2 denotes the number 1.5*102. Next, let us see some more examples,

```
>>> 1E2
    100.0
>>> -2.2E2
    -220.0
>>> -2.2E-2
    -0.022
```

Complex

A complex number is of the form `x+yj` and comprises a pair of floating point numbers. The first one `(x)` is called the real part, and the second one `(y)` is called the imaginary part. Some examples of complex numbers are `5+6j`, `3-2j`, `2+j`, and so on. The real and imaginary parts of a complex number can be retrieved as shown below:

```
>>> a = 5+6j
>>> a
    (5+6j)
>>> a.real
    5.0
>>> a.imag
    6.0
```

Let us now examine the type of a complex object and its parts,

```
>>> type(a)
    <class 'complex'>
>>> type(a.imag)
    <class 'float'>
>>> type(a.real)
    <class 'float'>
```

Note that when we write `a = 5+6j`, Python interprets `5` and `6` as floating point numbers.



In a complex number of the form `x+yj`, `j` denotes $\sqrt{-1}$.

1.1.2 bool (Boolean)

In Python, the Boolean data type is called **bool** and includes two values: **True** and **False**. An expression constructed using Boolean values is called a Boolean expression. Interestingly, in a Boolean expression, Python considers the numeric value 0 as **False** and a numeric value other than 0 as **True**. Now, let us examine the following examples,

```
>>> condition = True
>>> type(condition)
<class 'bool'>
```

So far, we have dealt with individual data elements. But, at times, one must deal with several data values, say, names of cities, marks of five subjects, details of items in a store, and so on. Sequence types available in Python are often used to deal with aggregates of objects.

1.1.3 Sequence Data Types

A Python sequence is an ordered collection of items indexed by the positions of the elements in the sequence. The first element in a sequence is at index value 0 (zero), the second element at index 1, the third element at index 2, and so on. Let us discuss three types of sequences, namely, strings, lists, and tuples.

Strings (str)

A string refers to a sequence of characters. These characters may include letters, digits, special characters, and spaces. A string may be enclosed within single quotation marks (for example, **'India'**) or double quotation marks (for example, **"India"**). Note that the quotation marks are not part of the string. They just mark the beginning and end of the string. Arithmetic operations cannot be performed on a string. Some examples of strings are as follows:

```
>>> country = "India"
>>> year = "1947"
>>> message = "I love my country"
```

When a string is displayed in IDLE, it is enclosed in single quotation marks so that leading and trailing spaces can be seen easily. As shown below, indexing is used to refer to an individual character of a string:

```
>>> country
'India'
>>> message
'I love my country'
>>> year
'1947'
>>> country[0]          ..... displays the first character
'I'
>>> message[7]          ..... Displays the eighth character
'm'
```

Lists (list)

A list is a sequence of items, separated by commas and enclosed within square brackets: []. The data items in a list may be of the same or different types. For example,

```
>>> lst = [1, 2, 3, 4]
>>> lst1 = [1, 2, 1, 1, 2, 3, 4, 3, 2]
>>> myList = ['A101',"Bread", 2, 20]
>>> print(myList)
['A101', 'Bread', 2, 20]
>>> lst
[1, 2, 3, 4]
>>> lst1
[1, 2, 1, 1, 2, 3, 4, 3, 2]
```

The individual elements of a list are accessed using the index as shown below:

```
>>> lst[2]              ..... displays the third element
    3
>>> myList[1]           ..... Displays the second element
    'Bread'
```

Indexes are also used to modify a list. For example,

```
>>> lst[0] = 10
>>> lst
    [10, 2, 3, 4]
```

Tuples

A tuple is a sequence of objects (not necessarily of the same type) enclosed in parenthesis: **()**. Just like lists, data items in a tuple are separated by commas (**,**). However, there is a striking difference between a list and a tuple. A tuple is an immutable object, i.e., once created, the values in a tuple cannot be changed using an assignment operator. In contrast, we have already seen that a list is a mutable object, i.e., the values in a list can be easily changed using an assignment operator.

```
>>> vowels
>>> vowels = ('a','e','i','o','u')
    ('a', 'e', 'i', 'o', 'u')
>>> weekDays = ('Monday', 'Tusday', 'Wednesday', 'Thursday', 'Friday')
>>> weekDays
    ('Monday', 'Tusday', 'Wednesday', 'Thursday', 'Friday')
```

On realising that you wrongly typed Tuesday, it would be tempting to modify **weekDays[1]** to **'Tuesday'**

```
>>> weekDays[1] =  'Tuesday'
    Traceback (most recent call last):
      File "<pyshell#101>", line 1, in <module>
        weekDays[1] = 'Tuesday'
    TypeError: 'tuple' object does not support item assignment
```

Note that an attempt to modify an element of a tuple has resulted in a type error because we have applied an illegal operation to a tuple.

1.1.4 Set

A set is an unordered and unindexed collection of data items separated by a comma and enclosed within curly brackets: **{}**. Unlike a list, a set cannot have duplicate values. For example,

```
>>> set1 = {1, 2, 3, 4}
>>> set1
    {1, 2, 3, 4}
>>> set2 = {1, 2, 3, 5, 1, 6, 2}
>>> set2
    {1, 2, 3, 5, 6}
```

Note that the duplicate entries in `set2` have been removed. A set may be modified using the operator `|=`. As follows:

```
>>> s = {1, 29, 4}
>>> id(s)
    2309188291712
>>> s = {54, 9}
>>> s
    {1, 4, 54, 9, 29}
>>> id(s)
    2309188291712
```

1.1.5 None

`NoneType` has a single value: **None.** It is used to signify the absence of value in a given statement. **None** supports no special operations. The value of **None** is neither **False** nor 0 (zero).

Consider the following examples:

```
>>> type(None)
<class 'NoneType'>
>>> print(None)
None
```

However, if you execute the command,

None

Nothing is printed. In a logical expression, the value of **None** is **False**.

1.1.6 Dictionary (Mapping)

A dictionary (`dict`) is an unordered set of key-value pairs enclosed in curly brackets: **{}**. It maps a set of keys to a set of values. The key is separated from its value using a colon(**:**) and key-value pairs are separated from each other by commas (**,**). A dictionary does not allow repeated keys.

```
>>> furniturePrice = {'chair':500, 'table': 2000, 'stool':1000}
>>> furniturePrice
{'chair': 500, 'table': 2000, 'stool': 1000}
>>> furniturePrice['table']
2000
```



For each of the following, identify its type.

1. `14.5`
2. `-203`
3. `14+7j`
4. `True`
5. `"Python"`
6. `[34,'Book', 25]`
7. `('Small','Medium','Large')`
8. `{'Small','Medium','Large'}`
9. `{'rollno':10,'name':'Joy'}`





1.2 Mutable and Immutable Data Types

Data types in Python are broadly categorised into two types—mutable and immutable. A modifiable object and its associated type are called mutable. For example, lists and dictionaries are mutable. Numeric data types (`int`, `float`, and `complex`), `bool`, `str`, and `tuple`, are immutable data types (see Fig 1.4). Next, we give examples of mutable and immutable data objects.

Fig 1.4: depicts the mutable and immutable data types in Python:

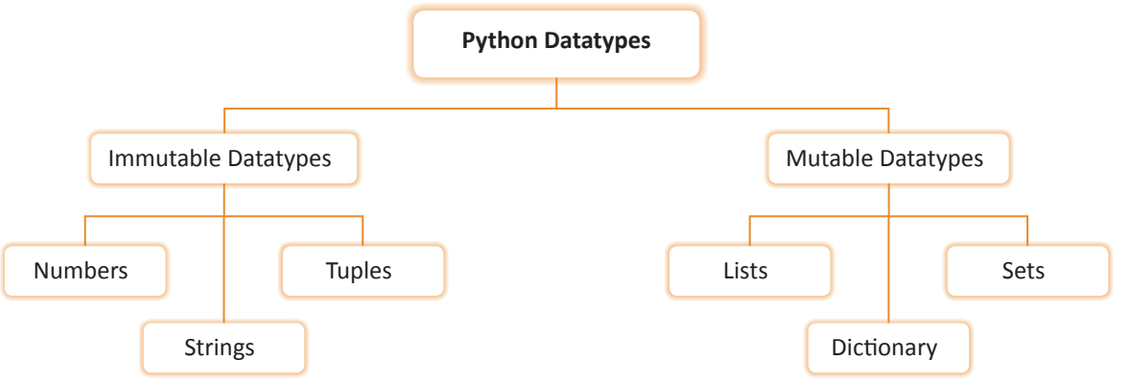


Fig 1.4: Mutable and immutable data types in Python

```
>>> num1 = 50
>>> id(num1)
    1734046084944
```

Recall that the function `id()` yields the object id of an object. Fig 1.4 shows object 50 being referenced by the variable `num1`.



Fig 1.5: num pointing to the data object, 50

Next, consider another assignment statement:

```
>>> num2 = num1
>>> id(num2)
    1734046084944
```

Note that each of the variables `num1`, and `num2` refers to the same object, 50 in the memory.

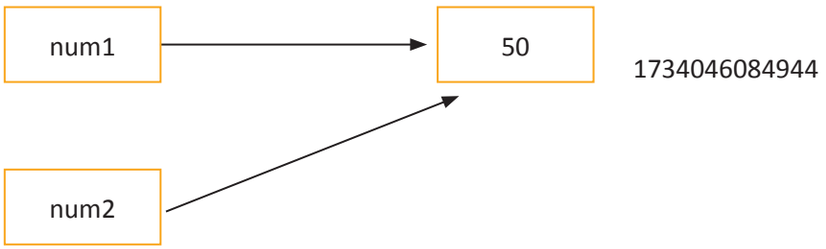


Fig 1.6: `num1` and `num2` pointing to the same object 50

Now, consider the following assignment statement:

```
>>> num1 = 100
```

It associates a new data object with the name `num1` as shown below:

```
>>> id(num1)
    1734046275024
>>> id(num2)
    1734046084944
```

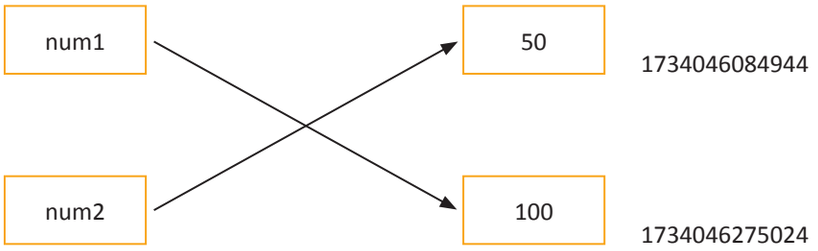


Fig 1.7: `num1` and `num2` refer to different data objects

Now let us examine how a mutable object such as a list may be modified.

```
>>> lst1 = [30, 60, 10, 20, 50]
>>> id(lst1)
    1933477279552
```

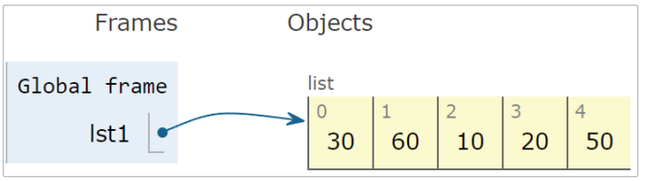


Fig 1.8: Variable `lst1` refers to a list object (object id 1933477279552)

```
>>> lst2 = lst1
>>> id(lst2)
1933477279552
```

Note that each of `lst1` and `lst2` refers to the same data object.

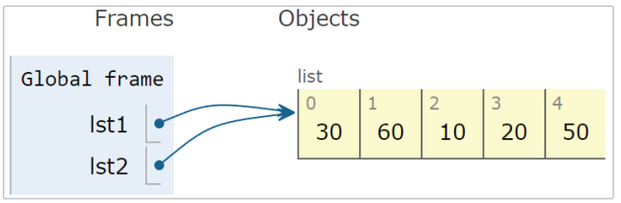


Fig 1.9: Variables `lst1` and `lst2` refer to the same list object (object id 1933477279552)

Next, consider the assignment:

```
>>> lst1[4] = 100
>>> lst1
[30, 60, 10, 20, 100]
>>> id(lst1)
1933477279552
```

Note that `lst1` and `lst2` still refer to the same object, as further illustrated below:

```
>>> lst2
[30, 60, 10, 20, 100]
>>> id(lst2)
1933477279552
```

Each of the variables `lst1` and `lst2` refers to the same modified list object.

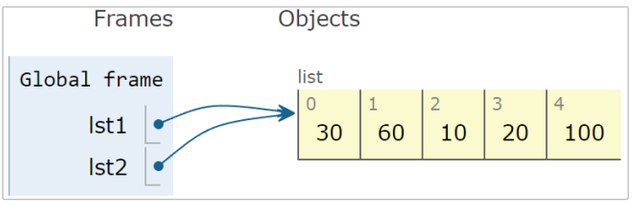


Fig 1.10: Variables `lst1` and `lst2` refer to the same list object (object id 1933477279552)

Finally, examine the following:

```
>>> lst1 = [3, 4, 2, 1]
>>> id(lst1)
2353324714816
>>> id(lst2)
1933477279552
```

Note that assigning the variable `lst1` to a new list object (indeed, it could as well be an object of some other type) changes its object id. As expected, the assignment of the list object `[3, 4, 2, 1]` to the variable `lst1`, did not impact the variable `lst2`, which continues to refer to the list `[30, 60, 10, 20, 100]` (object id `1933477279552`).

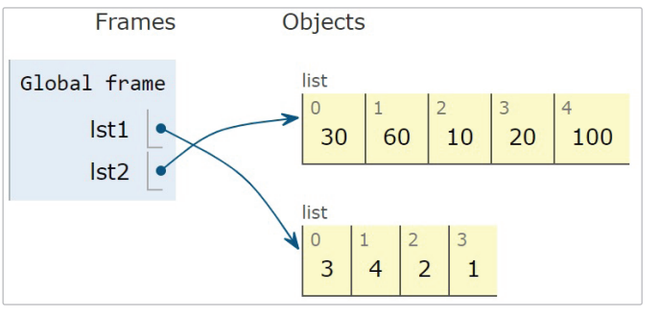


Fig 1.11: The variables `list1` and `list2` refer to different objects

1.3 Variables and Assignment Statements

In Python, data values like `234`, `56.54`, `2+3j`, and `'Hello world'` are called objects. Frequently, it is convenient to name the objects so they can be referred to and manipulated later. The name of an object is also called a *variable*, even though Python programmers prefer to use the term *name*. We will use the terms *name* and *variable* interchangeably. An assignment statement associates a value with a *name*; for example, the following assignment statement associates (assigns) the name `marks` with the integer object `56`:

```
>>> marks = 56
```

In other words, we say that the variable `marks` is assigned value `56`, or the variable `marks` refers to the value `56`, the variable `marks` is mapped to value `56`.

1.4 Keywords

Python reserves a set of words, called keywords, that cannot be used to name data objects, functions, etc. You can obtain the list of Python keywords by executing the following instructions:

```
>>> import keyword
>>> print(keyword.kwlist)
```

Output:

```
>>> ['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class',
    'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global',
    'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise',
    'return', 'try', 'while', 'with', 'yield']
```

1.5 Identifiers

A name in Python is also called an identifier as it uniquely identifies a data object, function, or class. However, in this book, we will restrict our attention to data objects and functions. An identifier must begin with a letter or an underscore character (_). An identifier may contain any number of letters, digits, or underscore characters, for example, `myMarks`, `salary`, `num_2022A`, and `lower_to_upper`, `binSearch`. However, an identifier cannot begin with a digit and must exclude special characters like : , `$`, `%`. Thus, `1digit`, `amount$`, and `marks%` are not valid Python identifiers.

1.6 Assigning Values to Several Variables

To enhance the readability of a program, several variables may be assigned values in a single statement as follows:

```
day, month, year = 1, 'April', 2023
```

The above statement assigns values `1`, `'April'`, and `2023` to variables `day`, `month`, and `year`, respectively.

1.7 Operators

Python provides several arithmetic, relational, and logical operators. In the following sections, we will briefly describe each type of operator and illustrate them using examples.

1.7.1 Arithmetic Operators

Arithmetic operators are used to build mathematical expressions involving numeric values. Python provides the following arithmetic operators:

- addition (+)
- subtraction (-)
- multiplication (*)

- division (/)
- floor division (//)
- modulus (%)
- exponentiation (**)

Below we illustrate the use of these arithmetic operators:

```
>>> 18 + 5
    23
>>> 18 - 5
    13
>>> 18 * 5
    90
>>> 18 / 5
    3.6
>>> 18 // 5
    3
>>> 18 % 5
    3
>>> 3**5
    243
```

1.7.2 Relational Operators

A relational operator is used for comparing two values. Execution of a relational operator yields a Boolean value **`True`** or **`False`**. The relational operators supported by Python are less than (<), greater than (>), less than or equal to (<=), greater than or equal to (>=), equals (==), and not equal (!=). Let us illustrate the use of relational operators in the following sequence of statements:

```
>>> 23 < 25
    True
>>> 23 > 25
    False
>>> 23 <= 23
    True
>>> 23 - 2.5 >= 5 * 4
    True
>>> 23 == 25
    False
>>> 23 != 25
    True
```

1.7.3 Logical Operators

The logical operators **`not, and,`** and **`or`** are applied to logical operands（also called Boolean operands or Boolean values）**`True`** and **`False`**, and yield a Boolean value. Below, we illustrate the use of logical operators:

```
>>> not True
    False
>>> 5 < 10 and 5 > 6
    False
>>> 5 < 4 or 6 > 5
    True
```

1.7.4 Identity Operators

Identity operators are typically used for checking the data type of an object. The identity operator returns **`True`** if the operands on either side of the operator are the same object or same type, and **`False`** otherwise.

Table 1.1 : Identity Operators

(Assume that `a, b, c` have values `-2, None, "Hello"`, respectively)

Operator	Explanation	Example

`is`	It yields **`True`** if the operands on either side of the operator are the same object or same type, and **`False`** otherwise.	`>>> type(a) is int`
		`True`
		`>>> type(c) is list`
		`False`
		`>>> a is b`
		`False`
		`>>> v = a`
		`>>> a is v`
		`True`

`is not`	It yields **`True`** if the operands on either side of the operator are different objects or different types, and **`False`** otherwise.	`>>> type(c) is list`
		`False`
		`>>> type(b) is not int`
		`True`
		`>>> type(c) is not list`
		`True`
		`>>> a is not b`
		`True`
		`>>> a = b`
		`>>> a is not b`
		`False`

1.7.5 Membership Operators

These operators are used to check whether a particular value is a member of a given sequence and return either **`True`** or **`False`**.

Table 1.2: Membership Operators

Operator	Explanation	Example

`in`	The expression `a` in `b` yields **`True`** if the object `a` is included in `b` and **`False`** otherwise.	`>>> myList = [0, -4, 5, 8, 10, 50, 4]`
		`>>> 5 in myList`
		`True`
		`>>> 100 in myList`
		`False`

`not in`	The expression `a not in b` yields **`True`** if the object `a` is not included in `b` and **`False`** otherwise.	`>>> myList = [0, -4, 5, 8, 10, 50, 4]`
		`>>> 5 not in myList`
		`False`
		`>>> 100 not in myList`
		`True`

1.7.6 Assignment Operators

Assignment operators are used to assign the value of the operand on the right-hand side of the operator to the operand on the left-hand side of the operator.

Table 1.3: Assignment Operators

(Assume the value of a, b, and c to be 10, 5, an −2 respectively for computations in the table)

Operator	Operation	Explanation	Examples

=	Assignment	The operator assigns the value on the RHS of the operator to the operand on the LHS of the operator.	>>> a = 10
			>>> b = a+b
			>>> b
			15

+=	Addition and assignment	The operator adds the value on the RHS of the operator to the operand on the left of the operator and assigns the result to the operand on the LHS of the operator. Note that the expression on the RHS is evaluated before applying +=.	>>> a += 20
			>>> a
			30
			>>> a += 3*4
			>>> a
			22

−+	Subtraction and assignment	The operator subtracts the value on the RHS of the operator from the value on the LHS of the operator and assigns the result to the operand on the LHS of the operator. Note that the expression on the RHS is evaluated before applying –=.	>>> b -= 2
			>>> b
			3
			>>> c -= 8//4
			>>> c
			-4

*=	Multiplication and assignment	The operator multiplies the value on the LHS of the operator by the value on the RHS of the operator and assigns the result to the operand on the LHS of the operator. Note that the expression on the RHS is evaluated before applying *=.	>>> b*=3
			>>> b
			15
			>>> c *= 5+3
			>>> c
			-16

/=	Division and assignments	The operator divides the value on the LHS of the operator by the value on the RHS of the operator and assigns the quotient (float) to the operand on the LHS of the operator. Note that the expression on the RHS is evaluated before applying /=.	>>> a /= 4
			>>> a
			2.5
			>>> a /= 2+3
			>>> a
			2.0

%=	Modulus and assignment	The operator divides the value on the LHS of the operator by the value on the RHS of the operator and assigns the remainder to the operand on its left. Note that the expression on the RHS is evaluated before applying %=.	>>> a %= 3
			>>> a
			1
			>>> a %= 2+4
			>>> a
			4

Operator	Operation	Explanation	Examples

//=	Integer division and assignment	The operator divides the value on the LHS of the operator by the value on the RHS of the operator and assigns the quotient (int) to the operand on the LHS of the operator. Note that the expression on the RHS is evaluated before applying //=.	>>> a //= 3
			>>> a
			3
			>>> c //= 2
			>>> c
			-1
			>>> a //= 1+3
			>>> a
			2

**=	Exponentiation and assignments	The operator computes (operand on the LHS of **=)** (operand on the RHS of **=) and assigns the result to the operand on the LHS of the operator. Note that the expression on the RHS is evaluated before applying **=.	>>> b **= 3
			>>> b
			125
			>>> c **= 2*3
			>>> c
			64

1.7.7 Precedence of Operators

When an expression contains multiple operators, Python resolves the order of execution according to the precedence of operators. An operator with higher precedence will be evaluated before the operator with lower precedence. The unary operators have higher precedence over binary operators. Table 1.4 shows the precedence of arithmetic operators from highest (first row) to lowest (last row):

Table 1.4: Precedence of Operators in Python

Order of Precedence	Operators	Description
1	**	Exponentiation
2	~, +, -	Compliment, unary plus, and unary minus
3	*, / , %, //	Multiply, divide, modulo, integer division
4	+, -	Addition, subtraction
5	<=, <, >, >=	Relational operators
6	==, !=	Equality operators
7	=, %=, /=, //=, -=, +=, *=, **=	Assignment operators
8	is, is not	Identity operators
9	in, not in	Membership operators
10	not, and, or	Logical operators

The parentheses `()` are used to override the precedence of operators, i.e., the expression within the `()` is evaluated first. Thus, `(12-(4-3))` evaluates to `11`. If two operators have the same precedence, then the expression is evaluated from left to right, except in the case of `**`, which is evaluated right to the left. Thus, while `12-4-3` evaluates to `5`, `2**2**3` evaluates to `256`.

1.8 Functions

Let us think of a car for a moment— a car is a complex entity with hundreds of parts which make it move with a little effort from the driver. Car manufacturing is a highly modular process. It involves manufacturing and testing various components/ parts of the car at different ancillary units. Subsequently, they are assembled at the main unit. Thus, the modular design of the car makes the process of fault diagnosis, servicing and maintenance easy to manage.



Similarly, as the complexity of a programming task increases, so does the length and complexity of the associated program, making it difficult to understand, manage, and debug. To manage a large program for solving a complex problem, we often divide the problem into subproblems and solve each sub-problem by a sub-program (small module). If a subproblem is still very complex, it may be further divided into subproblems. This process of dividing a complex problem into simpler manageable problems is called stepwise refinement. It saves a lot of time and effort. The process of dividing a computer program into separate blocks of code or sub-programs (each having a name and functionality) is called ***modular programming***. Each independent unit of code is called a ***module***. However, in the context of Python, a program file is called a module or a script. This chapter will discuss the modular approach to developing programs.

A function is a sub-program that manipulates data received as an input to solve a problem or sub-problem of a complex problem and often returns one or more results. Once a function has been defined, it may be called several times from different parts of the program. A function can often be tested independent of other functions. However, sometimes we may need to write a little extra code to test a function. Subsequently, the functions are integrated to solve the complex problem. Functions can be broadly classified into built-in functions and user-defined functions.

1.9 Built-in Functions

Python provides a large number of pre-defined built-in functions such as **`int()`**, **`input()`**, and **`print()`**. A function is invoked using the function's name followed by the input arguments within a pair of matching parentheses, separated by commas. The function arguments serve as inputs to the function for performing its computations. Some commonly used types of built-in functions are mentioned in Fig 1.12:

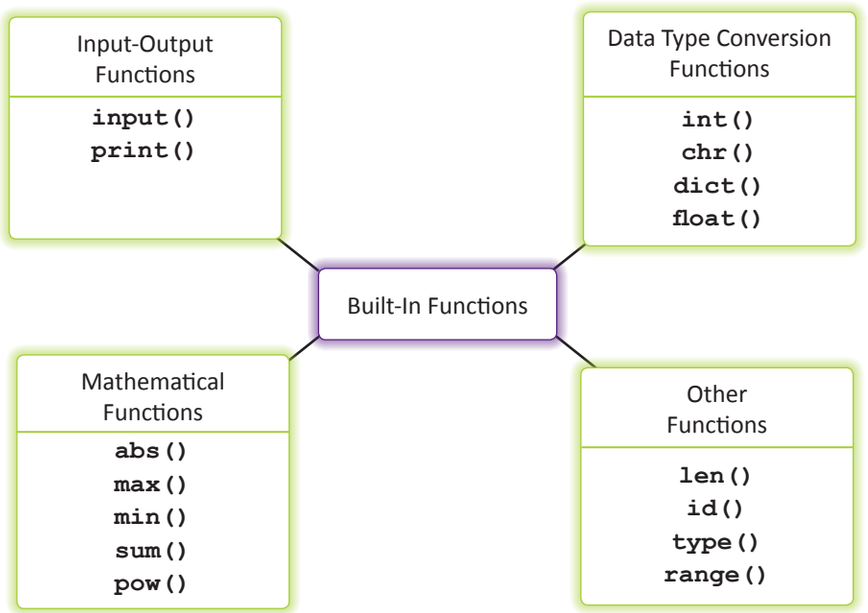


Fig 1.12: Some commonly used built-in functions

1.9.1 Input and Output

A computer program will often require inputs from the user, process them, and produce an output for the user. In Python, the `input()` and `print()` functions are used to perform the input and output operations, respectively.

input()

The input() function reads the text entered by a user until a newline is encountered. Suppose we wish to fetch the name of a programming language as an input from a user. For this purpose, the function `input()` may be invoked as follows:

```
>>> language = input()
    Python
```

On execution of the above statement, Python prompts the user for the name and waits for the input. When the user enters the string and hits the enter key, Python assigns the string Python to the variable language, as verified below:

```
>>> language
    'Python'
```

The above use of the `input()` function seems simple and devoid of any problems. If we write a small program that needs only one or two inputs and execute it soon after we are done with the program, then this approach works fine. But real-life programs may require several inputs and be executed days, months, or years after they are developed. In such scenarios, it is nearly impossible to remember the inputs required by the program. In fact, most of the software is used by people who had no role in developing it. Therefore, it is good practice to display a suitable message indicating the inputs required, as illustrated below:

```
>>> input('Enter name of a Programming language: ')
    Enter name of a Programming language: Python
    'Python'
>>> language = input('Enter name of a Programming language: ')
    Enter name of a Programming language: Python
>>> language
    'Python'
```

The following syntax describes the use of `input()` function to assign the user input to a variable.

```
<variable> = input([<message-to-prompt-user-to-enter-input>])
```

In the above syntax description, input is the keyword and `message-to-prompt-user-to-enter-input` is the message that is displayed on the screen, followed by the cursor. Note that `message-to-prompt-user-to-enter-input` is enclosed within square brackets, indicating that its use is optional. Indeed, we have already seen an example of the use of `input()` function where we did not specify any `message-to-prompt-user-to-enter-input`. In the above description, variable is to be replaced with a variable name. The function **`input()`** accepts user input from the keyboard. On seeing the cursor, the user enters the required inputs and hits the enter key to indicate the end of input. Once the user presses the enter key, the **`input()`** function yields the user input, which is assigned to the variable on the left-hand side of the assignment operator. Python considers user inputs as strings. The use of the `input()` function is illustrated below:

```
>>> rollNo = input('Enter your roll number: ')
    Enter your roll number: 12
>>> name = input('Enter your name : ')
    Enter your name: Kabir
>>> print(rollNo, name)
    12 Kabir
```

In the above example, the variable `rollNo` is assigned the value `12` entered by the user. Similarly, the variable `name` is assigned the value `Kabir`. As mentioned above, Python interprets the user inputs as strings. Thus, each of the values `12` and `Kabir`, entered (by the user) is an example of a string. Next, suppose we wish to calculate the perimeter of a rectangle based on the input `length` and `breadth` entered by a user.

```
>>> length = input('Enter length of rectangle: ')
    Enter length of rectangle: 6
>>> breadth = input('Enter breadth of rectangle: ')
    Enter breadth of rectangle: 4
>>> perimeter = (length + breadth)*2
>>> perimeter
    '6464'
```

Recall that Python considers the user inputs entered interactively as strings. So, the variables length and breadth were assigned the values '6' and '4' respectively. Therefore, `(length + breadth)` resulted in the string '64'. Finally, `(length + breadth)*2` resulted in '6464'. However, we were interested in the perimeter of the rectangle. So, we need to convert length and breadth to numeric values before applying the arithmetic operations. This can be done using the function **`int()`** as follows:

```
>>> perimeter = (int(length) + int(breadth))*2
>>> perimeter
    20
```

Alternatively, we could transform the string inputs to integer values while reading from the user as follows:

```
>>> length = int(input('Enter length of rectangle: '))
    Enter length of rectangle: 6
>>> breadth = int(input('Enter breadth of rectangle: '))
    Enter breadth of rectangle: 4
>>> perimeter = (length + breadth)*2
>>> perimeter
    20
```

print()

As the computer screen is the default device for displaying the results, the output produced on invoking the **`print()`** function is displayed on the screen.

```
>>> print('Hello')
    Hello
```

In the above example, `'Hello'` is the argument passed to the function **`print()`**. The function **`print()`** may be invoked with an arbitrary number of arguments. For example, the following function call makes use of two arguments:

```
>>> print('Hello!', 'How are you?')
    Hello! How are you?
```

Next, we make use of print() function to display the value of an arithmetic expression,

```
>>> print(2+14)
    16
```

In the above example, the **`print()`** function accepts the argument `2+14`, evaluates it, and displays the result of the evaluation (16) on the screen. More generally, we may invoke the **`print()`** function with several arguments separated by commas. For instance,

```
>>> print('Sum of',  4, 'and', 5, 'is', 4+5)
    Sum of 4 and 5 is 9
```

When no arguments are specified, the function **`print()`** does not output anything, as shown below:

```
>>> print()
```

Next, we would like to mention a refinement of the syntax for using the **`print()`** function as follows:

Syntax:

```
print(value, [,…,] [, sep = <separator>])
```



In the syntax description, the values/items enclosed in the square brackets [] are optional.

The syntax [,…,] indicates that while invoking the function **print()**, one can specify any number of values separated by commas. As the function **print()** may be invoked without any arguments also, above description of Python syntax may be revised as follows:

Syntax:

```
print([value1, value2, …, valueN] [, sep = <separator>])
```

Further, programmers can override the default separator space by specifying one of their choice. For example,

```
>>> print('Sum of',  4, 'and',  5, 'is', 4+5, sep ='   ')
    Sum of   4   and   5   is   9
```

Note that the values displayed by the **print()** function are now separated by three spaces instead of one.

Finally, we would like to mention a further refinement of the syntax for the **print()** function as follows:

```
print(value, [,…,] [ sep = <separator>, end = <endmarker>])
```

We already know that, by default, the output produced on invoking the **print()** function terminates the output line. Indeed, we have already seen that the output produced by invoking the **print()** function multiple times, appears on separate lines. Sometimes, we may like the output to appear with double line spacing. This may be achieved as follows:

```
>>> print('Hello', end = '\n\n')
    Hello

>>> print('Welcome to Python Programming', end = '\n\n')
    Welcome to Python Programming

```

Data Type Conversion Functions

Python provides several built-in functions for type conversion, for example, **int()**, **float()**, and **str()**. The data types in Python are also called classes. Thus, we may say `234` is an object of type **int**, or `234` is an object of class **int**, or `234` is an instance of class **int()**. Let us now illustrate their use:

```
>>> int('234')
    234
>>> str(234)
    '234'
>>> float('234.50')
    234.5
>>> int(234.50)
    234
```

Note that `'234'`, `234`, `'234.50'`, and `234.50` are arguments for the functions **int()**, **str()**, **float()**, and **int()**, respectively.

The help() Function

The `help()` function is used to display the documentation of the `functions`, `classes`, `modules`, etc. For instance, let us have look at the functionality of the in-built function `help()`:

```
>>> import math
>>> help(math.sin)
    Help on built-in function sin in module math:

    sin(x, /)
        Return the sine of x (measured in radians).
```

Built-in Modules

Python also provides several modules, each of which contains several useful functions. Oftentimes, it is convenient to use predefined functions available in Python modules.

However, to use these functions, we need to import the relevant module using the following syntax:

```
import <modulename>
```

Once a module is imported, we can use all the functions defined in the module by preceding function name by the name of the module and a dot(.) operator as shown below:

```
<modulename>.<functionname>()
```

For example, to use the **`sqrt()`** function, defined in the math module, we first import the math module and then use **`sqrt()`** function to find the square root of 25.

```
>>> import math
>>> math.sqrt(25)
    5
```

Alternatively, all the functions belonging to a module can be accessed directly without referring to the module name as follows:

```
>>> from math import *
>>> pi
    3.141592653589793
>>> e
    2.718281828459045
>>> ceil(2.3)      #smallest integer greater than or equal to 2.3
    3
>>> floor(2.8)     #largest integer not greater than 2.8
    2
>>> sqrt(25)       #outputs square root of 25
    5.0
>>> pow(2, 3)      #outputs 2^3
    8.0
>>> abs(-2)        #outputs absolute value of -2
    2
>>> fabs(-2)       #outputs floating point absolute of -2
    2.0
>>> sin(pi/2)
    1.0
>>> cos(0.0)
    1.0
>>> tan(pi/4)
    0.9999999999999999
```

Even though importing all the functions of a module appears to be simple, it is not usually wise to import all the functions of a module as some of the function names may conflict with the names of data objects or functions defined by the programmer. So, in the above example, we would prefer to write:

```
>>> from math import sqrt, pow
```

Python module **`random`** is another useful module that enables us to generate random numbers. The **`random()`** function of the **`random`** module enables us to generate a random number between 0 and 1.

```
>>>
>>> help(random.random)
    Help on built-in function random:

    random() method of random.Random instance
        random() -> x in the interval [0, 1).
>>> random.random()
    0.4988834047541426
```

The **`randint(a, b)`** function of the **`random`** module enables us to generate a random number between `a` and `b`, both inclusive.

```
>>> help(random.randint)
    Help on method randint in module random:

    randint(a, b) method of random.Random instance
        Return random integer in range [a, b], including both end points.
>>> random.randint(1,10)
    4
```

The **`randrange(start, stop=None, step=1)`** function of the **`random`** module enables us to generate a random number between `start` and `stop`, not including stop.

```
>>> help(random.randrange)
    Help on method randrange in module random:

    randrange(start, stop=None, step=1) method of random.Random instance
        Choose a random item from range(start, stop[, step]).

        This fixes the problem with randint() which includes the
        endpoint; in Python this is usually not what you want.
>>> random.randrange(1,11,2) # outputs any one odd number excluding 11.
    5
```

Python module **`statistics`** is another useful module that enables us to find the mean, median, and mode of numbers.

```
>>> import statistics
>>> statistics.mean([5, 3, 1, 2, 4, 7, 6, 8, 10, 9])
    5.5
>>> statistics.median([5, 3, 6, 8, 9, 12, 5])
    6
>>> statistics.mode([5, 3, 6, 8, 9, 12, 5])
    5
```

Some of the commonly used built-in functions are summarized in Table 1.5:

Table 1.5: Commonly Used Built-in Functions

Function	Description	Examples

`print(<data object>)`	Outputs the value of an expression passed to it as an input.	`>>> print('Hello')`
		`Hello`
		`>>> print(2+14)`
		`16`

`input(<display message>)`	Reads an input text from the user until a **`newline`** is encountered and returns it as a string without evaluating its value.	`>>> name = input('Enter name: ')`
		`Enter a name: Python`
		`>>> print(name)`
		`Python`

`int(value), float(value), str(value)`	Input argument converted to integer, floating point and string value respectively.	`>>> int('234')`
		`234`
		`>>> str(234)`
		`'234'`
		`>>> float('234.50')`
		`234.5`
		`>>> int(234.60)`
		`234`

Function	Description	Examples
`eval(<string>)`	Evaluates the input argument and returns the result of evaluation.	`>>> eval('15+10')` `25`

`max(<sequence>)`
`or`
`max(<val1,val2,…,valn>)`
	Returns the largest argument from a list, tuple, or sequence of arguments.	`>>> lst=[3, -2, 0, 78, 25]`
		`>>> max(lst)`
		`78`
		`>>> max(2,0,-4,19,36,8)`
		`36`
		`>>> max((6, 7, 4))`
		`7`
		`>>> max(59, 80, 95.6, 95.2)`
		`95.6`

`min(<sequence>)`
`or`
`min(<val1,val2,…,valn>)`
	Returns the smallest argument from a list, tuple, or sequence of arguments.	`>>> lst=[3, -2, 0, 78, 25]`
		`>>> min(lst)`
		`-2`
		`>>> min(2,0,-4,19,36,8)`
		`-4`
		`>>> min((6, 7, 4))`
		`4`
		`>>> min(59, 80, 95.6, 95.2)`
		`59`

`divmod(<num1,num2>)`	Returns a tuple comprising quotient and remainder when num1 is divided by num2.	`>>> divmod(17,4)`
		`(4, 1)`
		`>>> divmod(-21,5)`
		`(-5, 4)`

`sum(<sequence [,num2]>)`	Returns sum of all numeric values in the given input sequence. Optional input num2, when provided, is added to the sum of elements of the sequence.	`>>> lst =[1, 3, 5, 7]`
		`>>> sum(lst)`
		`16`
		`>>> sum(lst,5)`
		`21`
		`>>> sum((1, 3, 5, 7))`
		`16`

`abs(num)`	Returns absolute value of integer, floating-point, or complex number.	`>>> abs(-12)`
		`12`
		`>>> abs(-14.7)`
		`14.7`
		`>>> abs(21)`
		`21`
		`>>> abs(4+3j)`
		`5.0`

Function	Description	Examples

`pow(num1, num2[,num3])`	Returns `num1` raised to the power of `num2`. Optional input, `num3`, when provided, is used for computation as follows: $num1^{num2}\%num3$	`>>> pow(14, 3)`
		`2744`
		`>>> pow(3, 4, 5)`
		`1`
		`>>> pow(-2.3, 5)`
		`-64.36342999999998`
		`>>> pow(2, 5, 3)`
		`2`

`round(num[, digits])`	Rounds the given input number `num` to the nearest integer or up to the specified precision `(digits)` after the decimal point.	`>>> round(10.6)`
		`11`
		`>>> round(13.8907, 2)`
		`13.89`
		`>>> round(-12.0456, 3)`
		`-12.046`

range() function

Let's now learn about the built-in function **`range()`**, which will help us when we talk about the for statement. The function **`range()`** returns a sequence of numbers within the specified range. The **`range()`** function may be invoked using any of the following formats:

`normal running font with :`
`range(stop)`
`range(start, stop)`
`range(start, stop, step)`

The following syntax summarises the above three formats:

`range([start,] stop [, step])`

- In the above Syntax, the arguments start and stop is optional. The default values for start and stop are 0 and 1, respectively.
- When these arguments are not given, the function returns a list of integers starting at 0 and ending at the integer stop, but not including the integer stop. For example, range(6) returns the sequence of integers: 0, 1, 2,3, 4, 5 (excluding 6).
- When the range() function is invoked with two arguments, range(start, stop), it returns a sequence of integers beginning with start and going up to stop. For example, range(2,6) returns the sequence of integers: 2, 3, 4, 5. Similarly, range(-2, 6) returns the sequence of integers: -2, -1, 0, 1, 2, 3, 4, 5.
- When the third argument is also specified, the function call range(start, stop, step) returns a sequence of integers beginning with start and going up to stop, but stepping over in chunks of size step. For example, the function call range(-7, 20,4) returns the sequence of integers: -7, -3, 1, 5, 9, 13, 17. Similarly, the function call range(30, -10, -8) returns the sequence of integers: 30, 22, 14, 6, -2.
- Note that the function call range(30, 10) returns an empty sequence as we cannot count up from 30 to 10. Similarly, the function call range(10, 30, -2) returns an empty sequence because we cannot count down from 10 to 30 (in steps of 2).

In Table 1.6, we show more examples of sequences that are returned when the function **`range()`** is called with different arguments.

Table 1.6 Use of range()

Function call	Sequence	Explanation
`range(2, 10, 1)`	[2, 3, 4, 5, 6, 7, 8, 9]	start value is 2 last value is 9 (=10–1) step value is +1
`range(–3, 3)`	[–3, –2, –1, 0, 1, 2]	start value is -3 last value is 2 (=3–1) step value is +1 (default)
`range(10, 20, 2)`	[10, 12, 14, 16, 18]	start value is 10 last value is 18 (last in sequence before 20) step value is +2
`range(20, 14, -1)`	[20, 19, 18, 17, 16, 15]	start value is 20 last value is 15 (last in sequence before 14)
`range(20, 10)`	[]	start value is 20 Unable to count from 10 up to 10 (default step value is +1)
`range(7)`	[0, 1, 2, 3, 4, 5, 6]	start value is 0 (default) last value is 6 (=7–1), step up is +1.
`range(5, 10, 0)`	Value error	For counting, step value must be a non-zero integer.
`range(10, 30, 0.5)`	Value error	For counting, step value must be a non-zero integer.

1.10 User-defined Functions

We already know that the functions provide a methodical approach to problem-solving by dividing a complex problem into simpler sub-problems, developing solutions of each sub-problem, and combining the individual solutions of the sub-problems to solve the original problem. A Python program is a global frame in which functions and statements appear.

The syntax for defining a function is as follows:

```
def function_name (comma_separted_input_parameters):
    Sequence of statements - Body of the function
```

Function definition begins with the keyword **`def`**, followed by the name of the function, followed by a pair of parentheses that encloses a list of parameters (if any) separated by commas. The rules for naming a function are the same as those for naming an identifier. Recall that name of an identifier should not be a Python keyword. The first line of a function definition that ends with a colon is known as the function header. The sequence of statements followed by the colon is right indented and forms the function's body (also called function block). The sequence of statements included in a function's body is executed when it is invoked.

We wish to write a program that computes the simple interest for the given principal amount, rate of interest percent per annum, and time in years. A Python program is also called a Python script or simply a script.

To compute the simple interest, we develop a user-defined function interest() (see Program 1.1).

- Line 1 defines the function header for the function interest(). Note that the function definition begins with the keyword def, followed by the name of the function-interest, followed by the parameters (to be used for computation) within parenthesis, followed by a colon at the end. Lines 2-11 comprise a sequence of statements forming the function's body.
- Lines 2-9 span a multi-line string (called a docstring), enclosed within triple quotes, which describes the purpose of the function, input parameters for the function, and the result of the computation.
- Note that the multi-line string (lines 2-9) is a comment (non-executable code).
- The docstring is followed by the statement that computes simple interest (line 10).
- Finally, there is a return statement to return the result of the computation to the statement that invoked the function.

Program 1.1 Function to calculate the simple interest.

```
01 def interest(principal, rate, time):
02     '''
03     Objective: To determine simple interest
04     Input Parameters:
05      principal- numeric value denoting principal amount
06      rate - numeric value denoting rate of interest in % per annum
07      time - numeric values indicating time period in years
08     Return value: float-simple interest
09     '''
10     simpleInterest = (principal*rate*time)/100
11     return simpleInterest
```

1.10.1 Calling a Function

Now, we will make use of the above definition of the function `interest()` in Program 1.2 to compute simple interest for a specified amount, interest rate, and time. Remember that a function is executed only when it is called. Calling a function is also called invoking a function. A function definition is a provision to do something. So, just defining a function in a program does not achieve any result unless it is invoked.



A function is executed only when it is invoked. Defining a function only makes a provision to do something. Defining a function in a program does not achieve any result unless invoked.

Program 1.2 Computation of simple interest

```
01 def interest(principal, rate, time):
02     '''
03     Objective: To determine simple interest
04     Input Parameters:
05       principal- numeric value denoting principal amount
06       rate - numeric value denoting rate of interest in % per annum
07       time - numeric values indicating time period in years
08     Return value: float-simple interest
09     '''
10     simpleInterest = (principal*rate*time)/100
11     return simpleInterest
12
13 #main program segment
14 '''
15 Objective: To compute simple interest
16 User Interface:
17   1. User is asked to enter:
```

```
18      principal amount, rate of interest, time period of interest
19    2. Output to the user:
20       Simple interest computed
21 '''
22 p = int(input('Enter the principal amount: '))
23 r = int(input('Enter the rate of interest: '))
24 t = int(input('Enter the time period: '))
25 assert p>=0 and r>=0 and t>=0
26 simpleInterest = interest(p, r, t)
27 print('Simple Interest:', simpleInterest)
```

A Python program is a global frame in which functions and statements appear (see Fig 1.13). When the code in the Python module `interest.py` is executed, Python encounters the definition of function `interest()` (lines 1–11) in the global frame and makes a note of it, as shown in Fig 1.13:

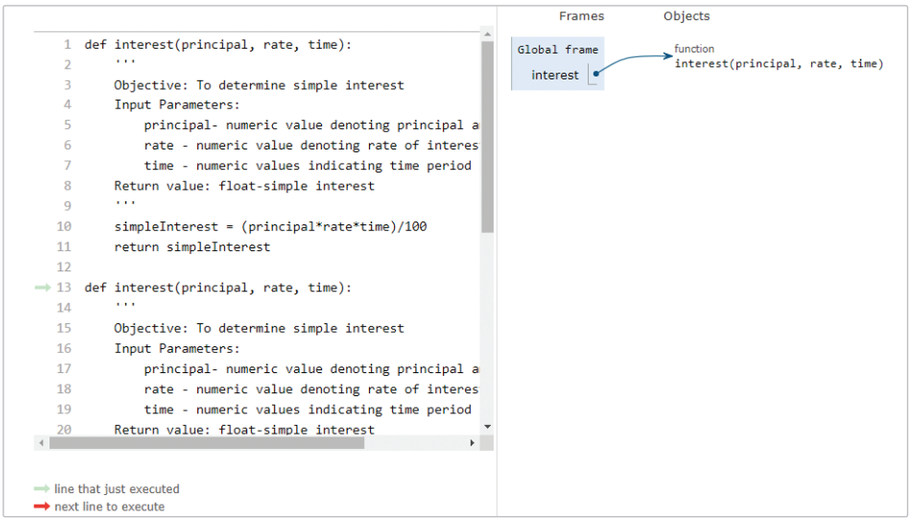


Fig 1.13: Python notes of the definition of the function `interest()`

- Lines 13–21 are for documentation purposes and are ignored by the Python interpreter.
- On execution of lines 22, 23 and 24, Python takes user inputs for the principal amount (p), rate of interest (r), and time period (t).
- Let us assume p, r, and t take values 1000, 5, and 2, respectively.
- Execution of line 25 validates the inputs.
- Line 26 is an assignment statement. The evaluation of the right-hand side expression of the assignment statement results in a call to the function interest().
- A call to a function is also known as invoking the function. So, instead of saying we make a call to the function interest(), we can say that we invoke the function interest(). Thus, the control is transferred to line 1 of Program 1.2 where the definition of the function interest() begins. The values 1000, 5, and 2 are passed as inputs to the function interest() and are assigned to the formal parameters (also known as dummy parameters) principal, rate, and time (see Fig1.14).
- As the function interest() is called from the global frame, we say that the global frame is the caller and the function interest() is the callee.

Note that a call to the function comprises the name of the function, followed by a pair of parentheses that includes (optionally) a comma-separated sequence of arguments:

```
function_name([comma_separated_sequence_of_arguments])
```

Further, note that the arguments in the function call appear in the same order in which formal parameters appear in the function definition.

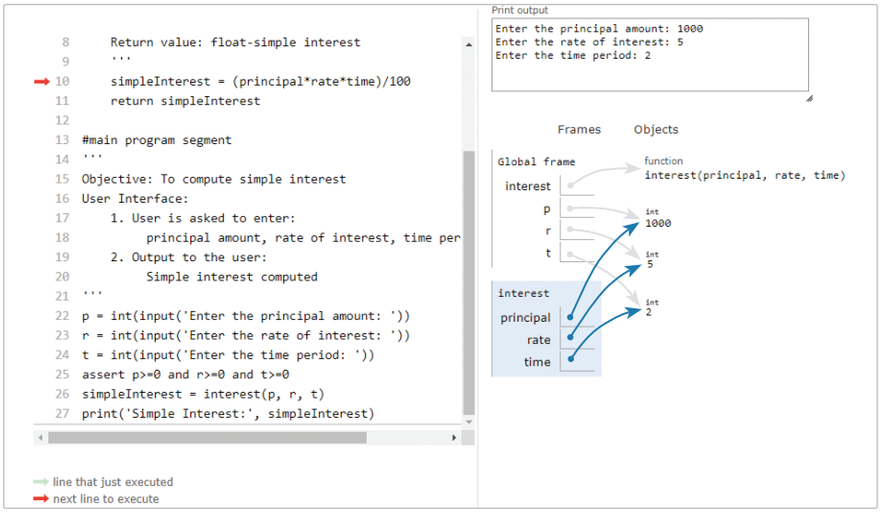


Fig 1.14: Call to function interest() with parameters: `1000, 5, 2`

- Lines 2–9 in the function interest() are for documentation purpose.
- Execution of line 10 computes simpleInterest. Note that the return value refers to simpleInterest (see Fig 1.15).

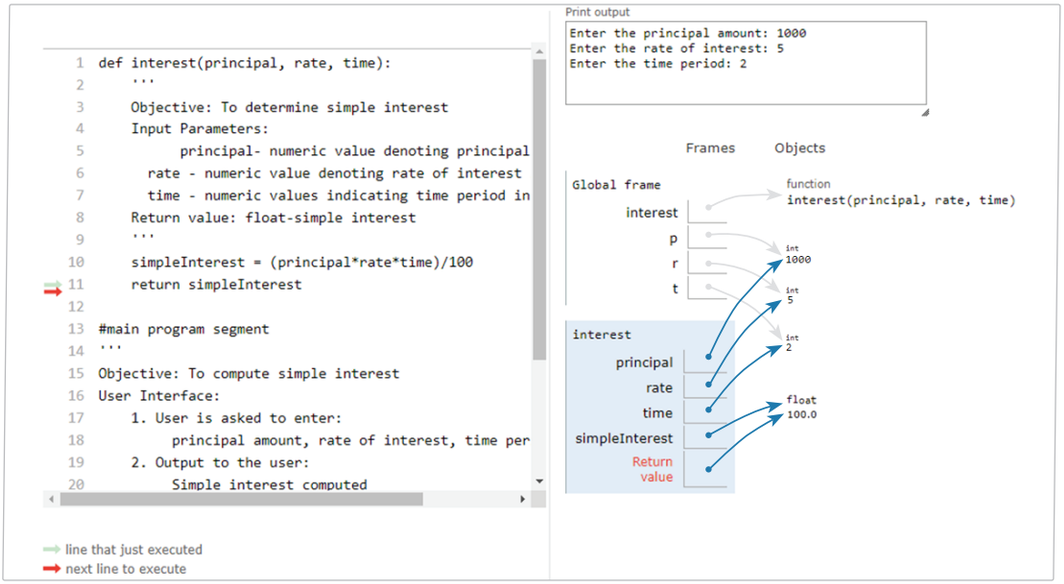


Fig 1.15: Return value (100.0): just before the return statement is executed

- When the return statement is executed, the control returns to line 26, and the execution of the assignment statement assigns the value 100.0 just returned by the function interest().
- Finally, the execution of line 27 prints the value of simpleInterest, and the execution of the program completes. (see Fig 1.16)

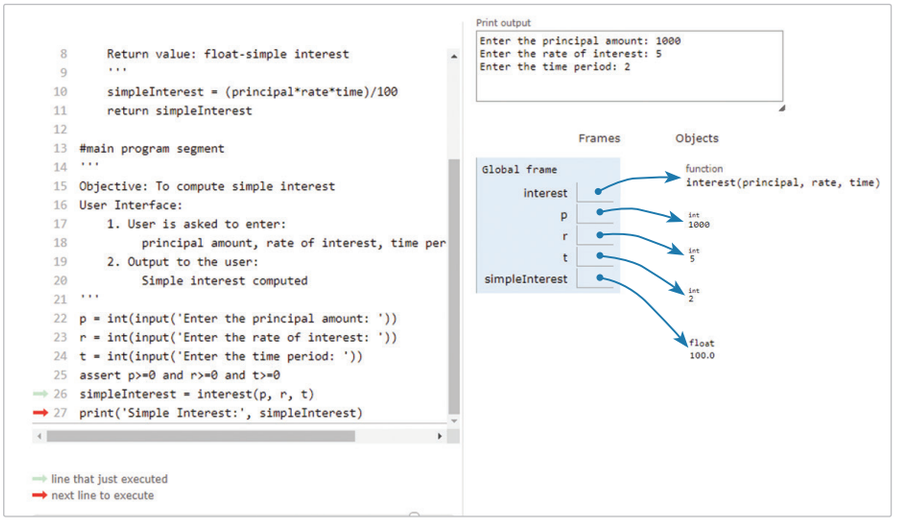


Fig 1.16: Execution of the assignment statement (line 26) completes as function `interest()` returns value 100.0 to the caller.

When Program 1.2 is executed in the Python's IDLE environment, the definition of the function `interest()` becomes available for use in the Python shell, and we can invoke it directly as shown below:

```
>>> interest(1000, 5, 2)
100.0
```

- The execution of the function begins by transferring the control to line 1, and the arguments 1000, 5, 2 are to be passed to the function interest().
- Skipping non-executable multi-line comment, execution of line 10 computes the simple interest, and line 11 returns the value of simpleInterest to the caller (Python shell in this case), which displays the value returned by the function interest().

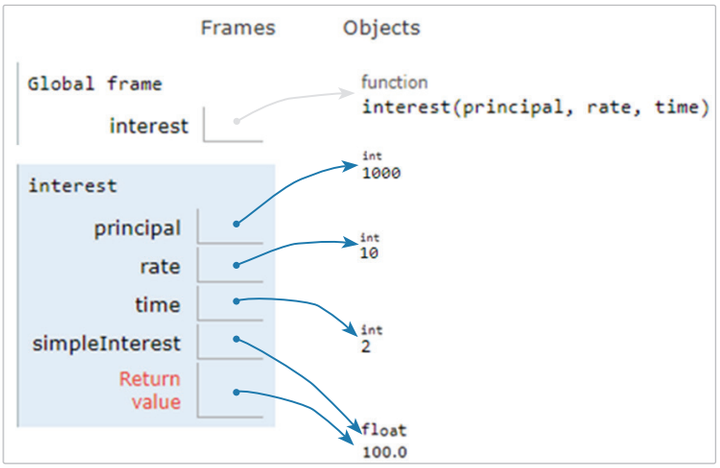


Fig 1.17: `interest(1000, 5, 2)` : call to function `interest()` from IDLE

Function Call: To invoke a function, using its name, followed by the arguments (if any) in parentheses
Input Arguments: Input values passed during the call to a function
Formal Parameters: Variables that receive the values during a call to the function

- The execution of the function begins by transferring the control to line 1, and the arguments 1000, 5, 2 are to be passed to the function interest().

- Skipping non-executable multi-line comment, execution of line 10 computes the simple interest, and line 11 returns the value of simpleInterest to the caller (Python shell in this case), which displays the value returned by the function interest().

Program 1.3 To display the triangle.

```
01 def printTriangle():
02     '''
03     Objective: To print the triangle
04     Input Parameters: None
05     Return value: None
06     '''
07     print('  *')
08     print(' ***')
09     print('*****')
10
11 printTriangle()
```

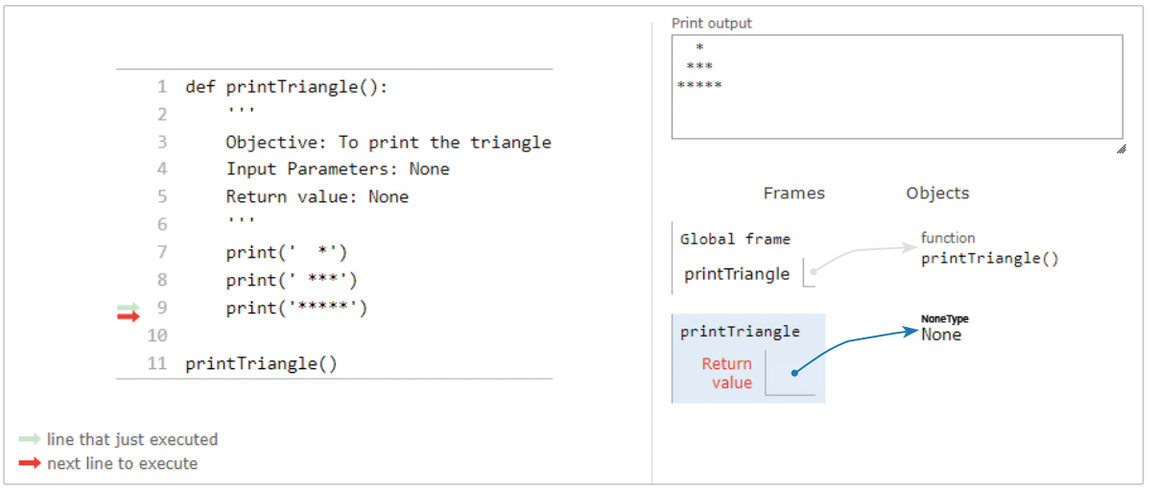


Fig 1.18: Return value **None** after the execution of `printTriangle()`

1.10.2 Help on a User-defined Function

When we invoke the `help()` function on a user-defined function, it displays the function name along with the parameter list and the first *docstring* (if present) in the function that appears before any executable statements, for example,

```
>>> help(interest)
Help on function interest in module __main__:
interest(principal, rate, time)
    Objective: To determine simple interest
    Input Parameters:
        principal- numeric value denoting principal amount
        rate - numeric value denoting rate of interest in % per annum
        time - numeric values indicating time period in years
    Return value: float-simple interest
```

Now, the function may be invoked either from within the program or Python shell. Suppose we wish to compute simple interest for the principal amount of `Rs. 1000`, rate `5%` and time `2` years.

1. **Positional arguments:** Input arguments which are provided in the function call in the same order in which the actual parameters are specified in the function definition.
2. **Input arguments** and **actual parameters** can have the same name.
3. **return** is a keyword used to return the result of computation from a function to the statement that invoked the function.
4. In absence of any return statement, a function returns the default value None.

Fig 1.14 shows how the input arguments in call to function `interest()` in line 26 are passed to the input parameters of the function (mapping of positional arguments and parameters) and how the value is returned back from the function to the main segment where the call to function is made. It may be noted that all the variables created in the main program segment are part of the global frame.

On executing Program 1.2, Python responded with the following output:

```
Enter the principal amount: 1000
Enter the rate of interest: 5
Enter the time period: 2
Simple Interest: 100.0
```

1.10.3 Default Parameters

So far, we have learnt that the function call must have the same number of arguments as defined in the function definition. Let us consider the function `interest()` again, but this time we imagine a scenario where the interest rate is typically 4% and the interest is usually computed for one year. In such a scenario, it would make sense to initialise the values of the formal parameters `rate` and `time` to `4` and `1`, respectively, while defining the function `interest()`. The formal parameters (`rate` and `time` in case of `interest()`) whose values are initialised as part of the definition of function are called default parameters. The values assigned to the formal parameters are called the default values of the parameters. To define the function `interest()` with the default parameters, we only need to modify the function header (line 1) to specify the default values of the parameters as shown below:

```
>>> def interest(principal, rate = 4, time = 1):
        '''
        Objective: To determine simple interest
        Input Parameters:
         principal- numeric value denoting principal amount
         rate - numeric value denoting rate of interest in % per annum
         time - numeric values indicating time period in years
        Return value: float-simple interest
        '''
        simpleInterest = (principal*rate*time)/100
        return simpleInterest
```

Now, to compute the simple interest for principal amount `Rs. 55,000`, for one year at `4%` interest, we only need to invoke the function `interest()` with the argument `55000` and leave the other two parameters unspecified as shown below:

```
>>> interest(55000)
2200.0
```

Of course, a user may still specify the values of the default parameters. When a user specifies the values of the default parameters, the default values are ignored. Suppose we wish to compute the simple interest for the principal amount of `Rs. 55000` for one year with an interest rate of `5%`. To carry out this computation, the following call to the function `interest()` takes default value `1` for the parameter `time`:

```
>>> interest(55000, 5)
2750.0
```

We have already seen above that a function, such as `interest()`, may have more than one default parameters. The unspecified parameters in the trailing sequence take the default values. So, if a parameter takes a default value, then all the other parameters to its right must also take default values. Skipping a default parameter and specifying a subsequent parameter will generate an error as shown below:

```
>>> interest(55000,,1)
    SyntaxError: invalid syntax
```

1. Default parameters take the default values provided in the function definition when these are not mentioned in a function call
2. All non-default parameters (if any) should precede the default parameters

1.10.4 Keyword Arguments

By default, the arguments in a function call are specified in the same sequence as in the formal parameter list. So, to compute the simple interest considering `55000` as the value of the principal, `4` as the rate of interest, and `2` as the time in years, we may invoke the `interest()` function as follows:

```
>>> interest(55000, 4, 2)
    4400.0
```

However, note that the interest rate 4 is already a default parameter. But, we had to specify it as we could not have specified the `time (2)`, skipping the `rate` parameter in the middle. Fortunately, Python allows us to specify the input arguments to a function in an arbitrary order by explicitly associating the names of the formal parameters with their values using the following syntax:

```
formal_paramter = argument value
```

For example, we can modify the above call to the function `interest()`, skipping the parameter `rate` as follows:

```
>>> interest(55000, time = 2)
    4400.0
```

Keyword arguments are often used when the default parameter list is quite long, and we wish to assign a user-specified value for only a few of them. For example, consider the following function `fun`:

```
>>> def fun(n1 = 10, n2= 20, n3 = 30, n4 = 40, n5 = 50, n6 = 60):
        return n1 + n2 + n3 + n4 + n5 + n6
>>> fun(n2 = 15, n4 = 35)
    200
```

1.10.5 Function Calls as Function Argument

A function argument may be an object of any Python type. For example, functions can also be passed as arguments as shown below:

```
>>> def applyFn(f, x):
...     '''
...     Objective: To demonstrate the use of a function as an argument
...     Input Parameters:
...      f: function being passed as argument
...      x: argument for invoking f
...     Return Value: f(x)
...     '''
...     return f(x)
```

Now, let us see some example of the use of above function:

```
>>> applyFn(abs,-4)
    4
>>> import math
>>> applyFn(math.sqrt,16)
```

```
    4.0
>>> applyFn(math.sin, math.pi/2)
    1.0
```

1.10.6 User-defined Modules

A Python module is nothing but a file comprising Python code. It can be imported into another program, just like other modules available in Python. In this section, we will build a module *geometry.py* that enables computation of the area and perimeter of rectangles and triangles. We encourage you to extend this module by incorporating the definitions of the functions for computing the area and perimeter of some more geometric figs. Currently, the module comprises the definition of the functions to compute the area and perimeter of rectangles and triangles in the file *geometry.py*. As the computations are self-explanatory, we do not describe these functions further in the text.

1. #Function to Compute Area of a Rectangle

```
>>> def areaRectangle(length, breadth):
...     '''
...     Objective: To compute the area of rectangle
...     Input Parameters: length, breadth – numeric value
...     Return Value: area - numeric value
...     '''
...     area = length * breadth
...     return area
```

2. #Function to Compute Perimeter of a Rectangle

```
>>> def perimeterRectangle(length, breadth):
...     '''
...     Objective: To compute the perimeter of rectangle
...     Input Parameters: length, breadth – numeric value
...     Return Value: perimeter - numeric value
...     '''
...     perimeter = (length + breadth) * 2
...     return perimeter
```

3. #Function to Compute Perimeter of a Triangle

```
>>> def perimeterTriangle(side1, side2, side3):
...     '''
...     Objective: To compute the perimeter of a triangle
...     Input Parameters: side1, side2, side3: – numeric values
...     Return Value: perimeter - numeric value
...     '''
...     perimeter = side1 + side2 + side3
...     return perimeter
```

4. #Function to Compute Area of a Triangle

```
>>> def areaTriangle(base, height):
...     '''
...     Objective: To compute the perimeter of a triangle
...     Input Parameters: base, height: numeric values
...     Return Value: area of triangle: numeric value
...     '''
...     #Approach: Formula used: area = height*base/2
...     area = (height*base)/2
...     return area
```

Suppose, the module *geometry.py* is stored in the folder *D:\classXII\Ch1*. Next, suppose we wish to use the module *geometry.py* for computing the areas of triangles and rectangles. For this purpose, we need to import these functions from the module *geometry* as follows:

```
import sys
sys.path.append('D:\classXII\ch1')
from geometry import areaTriangle, areaRectangle
```

Next, we demonstrate how the module *geometry* may be used from the Python shell:

```
>>> import sys
>>> sys.path.append('D:\classXII\ch1')
>>> from geometry import areaTriangle, areaRectangle
>>> areaTriangle(5, 6)
    15.0
>>> areaRectangle(5, 6)
    30
```

1.11 Types of Errors

Various types of errors may be generated while creating or executing a program. The process of identifying and removing such errors is known as **debugging**. The errors are called **bugs**. The errors in a program broadly fall in three categories:

- Syntax Error
- Logical Errors
- Runtime Errors

1.11.1 Syntax Errors

Every programming language is described by a set of rules, collectively called the syntax of the language. If a program violates a syntax rule, the interpreter/compiler will flash an error. Such errors are known as **syntax errors**. Thus, a Python interpreter will execute only those statements that are syntactically correct. When a syntax error is encountered, the execution of the program stops, and the error message is displayed. The execution of the program resumes only after the error is rectified. For example,

```
>>> print("hello)
...
    SyntaxError: incomplete input
```

in the statement `print("hello)`, the closing quotation marks are missing, leading to a syntax error. Python interpreter noted the beginning of string, but could not fing the closing quote marks. A syntax error must be removed before Python interpreter can proceed.



Python interpreter gives a brief description of each error, as seen in the above example.

1.11.2 Logical Errors

A logical error does not give the desired output, even though there is no syntax error in the program. Such errors are sometimes difficult to identify as all the statements in the program are executed successfully. When a logical error is encountered, often it helps to work backward by examining the output produced by the program and look for the cause of error.

For example, to calculate the percentage of marks obtained by a student, the expression should be:

```
percentage = marksObtained/maxMarks * 100
```

However, if the programmer writes the formula as:

```
percentage = maxMarks/marksObtained * 100
```

The statement being syntactically correct, will be executed but it will not give the correct output.

1.11.3 Runtime errors

As the name suggests, a runtime error occurs during the execution of the program. The statement is syntactically correct but cannot be executed by the interpreter. For example, while executing the statement:

```
result = numerator/denominator
```

if the value of denominator is 0 (zero), it will result in a runtime error (division by zero). Similarly, a runtime error will be generated, if the user enters a string instead of a number. The Program 1.4 below illustrates the runtime error.

Program 1.4 Illustrate the runtime error

```
01 # Objective: To compute quotient of two numbers
02 #           : To understand runtime errors.
03 #User inputs: numeric values for numerator and denominator
04 numerator = int(input("Enter the numerator : "))
05 denominator = int(input("Enter the denominator : "))
06 quotient = numerator/denominator
07 print("The quotient is : ", quotient)
```

Output 1: When the user enters denominator as 0

```
>>> Enter the numerator : 40
>>> Enter the denominator : 0
    Traceback (most recent call last):
      File "F:/PythonCode/class12/ch01/runTimeErr.py", line 4, in <module>
        quotient = numerator/denominator
    ZeroDivisionError: division by zero
```

Output 2 : When the user enters numerator as a string that cannot be interpreted as an int

```
>>> Enter the numerator : 23*23
    Traceback (most recent call last):
      File "F:/PythonCode/class11/ch07/7_2_runTimeErr.py", line 2, in <module>
        numerator = int(input("Enter the numerator : "))
    ValueError: invalid literal for int() with base 10: '23*23'
```

Output 3: When the user enters suitable values

```
>>> Enter the numerator : 20
>>> Enter the denominator : 5
    The quotient is :  4.0
```

Below, we give some more examples of runtime errors:

```
>>> print(x)
    Traceback (most recent call last):
      File "<pyshell#0>", line 1, in <module>
        print(x)
    NameError: name 'x' is not defined
```

In the above example, the variable (name), `x` is accessed for printing its value before it is assigned a value. So, a **`NameError`** is displayed because the name `x` is unknown to the interpreter. In the following example, the use of **`int`** and **`str`** objects is incompatible with the + operator, so the interpreter points out a **`TypeError`**.

```
>>> 10 + "Hello"
    Traceback (most recent call last):
      File "<pyshell#1>", line 1, in <module>
        10 + "Hello"
    TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

1.12 Control Structures

So far, we have developed Python code comprising functions in which all the statements are executed in a sequence, one after the other. However, while solving a real-world problem, a statement may or may not be executed depending on some condition. For example, while moderating the examination result, we may add one or two marks to the marks obtained by a student if they are failing by a margin of one or two marks but leave the marks unchanged in all other cases.

Further, we would like to repeat the computation of results for all the students of a particular class section (say, class 8 B) and also for each section of the class. We may have a different policy for computing the result of another class, say, class 9. In this section, we will study different Python statements that would enable us to control the flow of execution of the statements in a program/function. The statements used to control the execution flow are also called control structures. The statements used to control the flow of execution of statements in a program/ function are called control statements.

1.12.1 Control Statements

Python provides three control statements which are as follows:

* **`if`** statement * **`if-else`** statement * **`if-elif-else`** statement

Simple `if`

An **`if`** statement allows us to execute a statement only if a specific condition is satisfied. There is only one condition to check in a simple **`if`**. The flowchart of the simple **`if`** is shown in Fig 1.19:

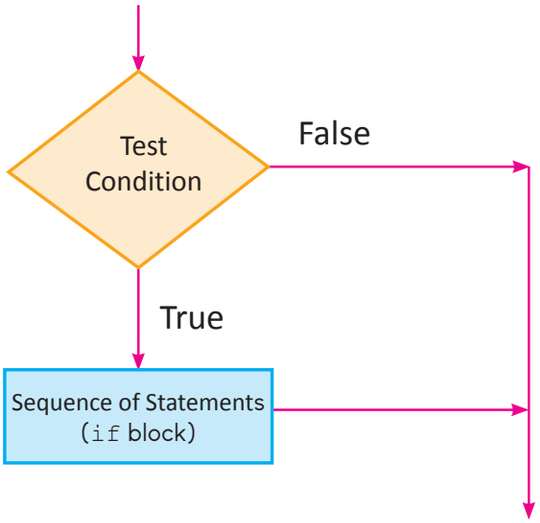


Fig 1.19: Simple **`if`** statement

Let us take the following example:

```
>>> marks = int (input ('Enter marks:'))
    Enter marks : 39
>>> if marks == 38 or marks == 39:
...     marks = 40
...
>>> marks
    40
```

Note that as the value of `marks` that was entered was `39`, the value of marks was changed to `40` on the execution of the **`if`** statement. However, in the following example, as the value of marks was too low, it remained unmodified:

```
>>> marks = int (input ('Enter marks:'))
    Enter marks : 23
>>> if marks == 38 or marks == 39:
... marks = 40
...
>>> marks 23
```

Let us consider another program segment:

```
01 n = int(input('Enter a number: '))
02 if n % 2 == 0:
03     print('Even Number')
```

In line 1, `n` is initialized with the value entered by the user. In the next line, there is an if statement which checks whether n leaves the remainder 0 on dividing by 2. If n indeed leaves the remainder 0 on dividing by 2, the following nested statement prints the string: `'Even Number'`.

if-else

An **`if-else`** statement evaluates the condition in the **`if`** clause and will execute the body of this clause only if the condition evaluates to **`True`**, but if the condition evaluates to **`False`**, then the body of the **`else`** clause is executed. The flowchart of the **`if-else`** statement is shown in Fig 1.20:

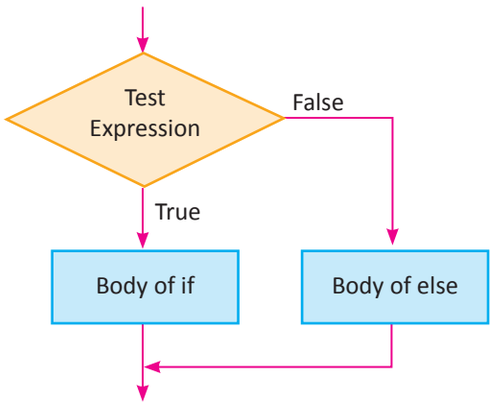


Fig 1.20: `if-else` statement

Let us take the following example:

```
01 n = int(input('Enter n: '))
02 if n % 2 == 0:
03     print('even number')
04 else:
05     print('odd number')
```

In line 1, `n` is initialised with the value entered by the user. In the next line, a condition checks whether the resulting value on evaluating an expression `n%2` against 0. If the `n` gets the value 7 from the user, the above code snippet outputs `Odd Number`, since 7 is an odd number. If the `n` gets the value 4 from the user, the above code snippet outputs `Even Number`.

if-elif-else

An **`elif`** clause checks multiple expressions for **`True`** and executes a block of code corresponding to an expression that evaluates **`True`**. Although there can be an arbitrary number of **`elif`** clauses following the **`if`** clause, an **`if-elif-else`** statement can include at most one **`else`** clause. The flowchart of the **`if-elif-else`** is shown in Fig 1.21. The general form of **`if-elif-else`** statement is as follows:

```
if < condition1 >:
    < Sequence S1 of statements to be executed >
elif < condition2 >:
    < Sequence S2 of statements to be executed >
elif < condition3 >:
    < Sequence S3 of statements to be executed >
...
else:
    < Sequence Sn of statements to be executed >
```

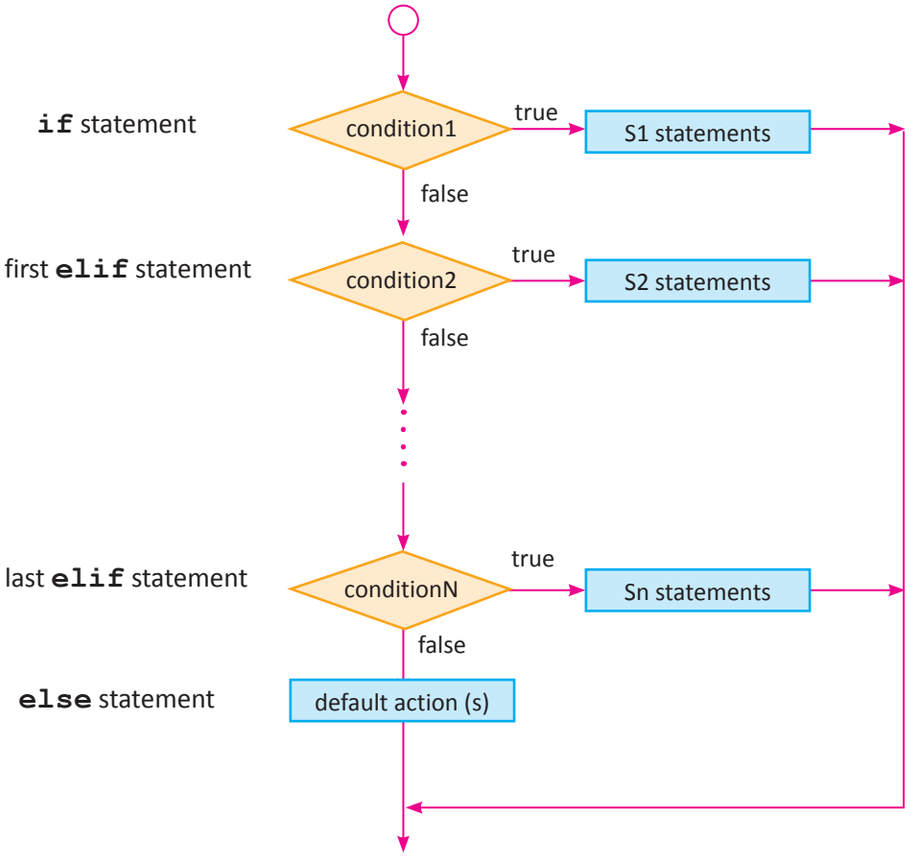


Fig 1.21: **if-elif-else** statement

```
num1 = int(input('Enter First Number: '))
num2 = int(input('Enter Second Number: '))
if num1 == num2:
    print (num1, 'is equal to', num2)
elif num1 < num2:
    print (num1, 'is less than', num2)
else:
    print (num1, 'is greater than', num2)
```

1.12.2 Statements for Looping

Often, a statement or a sequence of statements needs to be executed several times. Python enables us to achieve this by providing **for** and **while** statements. The repeated execution of a statement or a sequence of statements is called a loop. Hence **for** and **while** statements are also called **for** and **while** loops. The statement or a sequence of statements being executed in a loop is called the loop's body. An execution of the loop's body is called an iteration of the loop.

for statement

A **for** statement is used for repeated execution of the loop's body for each occurrence of an element in a sequence. Recall that a sequence may be a range of numbers, a list, a tuple, a dictionary, or a set. Syntax of **for** statement is as follows:

```
for item in sequence:
    <block S of statements>
```

The flowchart of the **for** loop is shown in Fig 1.22:

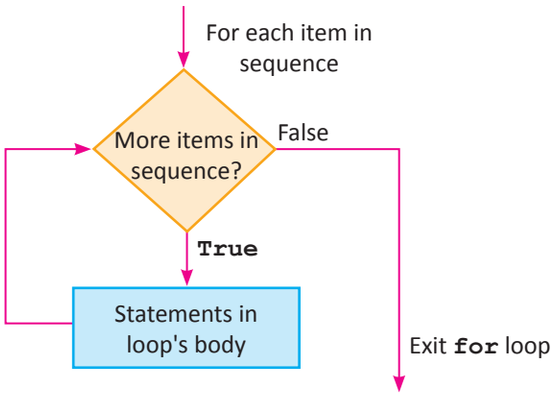


Fig 1.22: **for** loop

Program 1.5 Program to find the Factorial of a number.

```
01 def factorial(n):
02     '''
03     Objective: To compute factorial of a number
04     Input Parameter: n - numeric value
05     Return Value: factorial - numeric value
06     '''
07     fact = 1
08     assert n >= 0
09     for i in range(1, n + 1):
10         fact = fact * i
11     return fact
12
13 #main program segment
14 '''
15 Objective: To compute factorial of a number provided by user
16 Input Parameter: None
17 Return Value: None
18 '''
19 n = int(input('Enter the number: '))
20 fact = factorial(n)
21 print('Factorial of', n, 'is', fact)
```

while statement

The **while** statement is used to repeatedly execute a block of statements, called body of the **while** statement so long as the condition specified in the **while** statement is satisfied. The **while** statement has the following syntax:

```
while <test condition>:
    <Sequence of statements S>
```

As shown in the flowchart of the **while** statement (Fig 1.23), the execution of the **while** statement begins by evaluating a Boolean expression (also called conditional expression, test expression, or simply condition) in the **while** statement. If the evaluation of the Boolean expression yields **False**, the loop's body is skipped and the execution of the **while** statement ends. However, if the evaluation of the Boolean expression yields **True**, the body of the while statement is executed and the Boolean expression is evaluated again. On executing the loop's body, the Boolean expression is evaluated again and the loop's body is executed again if the evaluation of the Boolean expression yields **True**. This process of evaluating the Boolean expression and executing the body of the **while** statement is repeated until the Boolean expression becomes **False**.

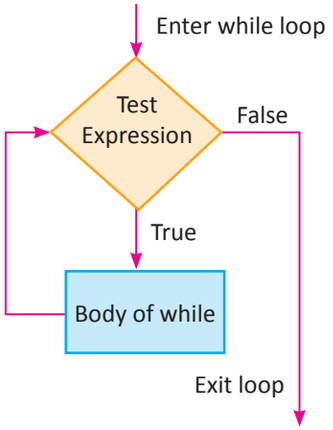


Fig 1.23: **while** statement

Now we give an application of the **while** statement. Suppose, we wish to find the sum of digits of a number. For this purpose, we write a function which accepts the number num as input parameter whose sum of digits we wish to find (see Program 1.6).

- As we are only interested in finding the sum of the digits of num, we ignore the sign of the number by takings its absolute value (line 13).
- Next, we initialise the sum of the digits (digitSum) as **zero**. Now, if the number whose sum of digits we want to find is **zero**, the condition num > 0 fails at the beginning of the execution of **while** statement and the execution of the **while** statement completes without executing the body of the while statement even once.
- However, if the condition num > 0 evaluates to **True** at the beginning of the **while** statement, we find the digits one by one beginning the unit's place. To find the digit at the unit's place, we compute the remainder obtained on dividing the number by 10 (num % 10) and add it to the digitsum accumulated so far (line 16).
- As the role of the digit at the unit's place is now over, we drop it from the number num by dividing it by 10 by setting num = num // 10 (line 17). Note that the number obtained on dividing the original number by 10, still contains all the relevant information, i.e. it contains all the digits except the one at the unit's place, which we have already accounted for.
- Further, the digit at 10's place in the original number now appears at the unit's place in the number obtained by dividing the original number by 10.
- So, we now need to find the sum of digits of the updated value of num and add it to the sum accumulated in **digitSum**.
- This process of finding the digit at the unit's place, adding it to the sum accumulated in **digitSum**, and updating the value of num as long as num>0 (line 15).
- When num becomes 0, the control exits the **while** statement and the computed value of the sum of digits **(digitSum)** is returned to the caller of the function.

Program 1.6 Find sum of digits of a number.

```
01 def sumOfDigits(num):
02     '''
03     Objective: To compute sum of digits of a number
04     Input Parameter: num - numeric
05     Return Value: numeric
06     '''
07     '''
08     Approach:
09     Ignore the sign of number. Initialize digit Sum to
```

```
    zero. Extract digits one by one beginning unit's
    place and keep on adding it to digit sum.
    '''
    num = abs(num)
    digitSum = 0
    while num > 0:
        digitSum += (num % 10)
        num = num // 10
    return digitSum

# main program segment
# Objective: To compute sum of digits of a user-provided input
num = int(input('Enter a number: '))
digitSum = sumOfDigits(num)
print('Sum of digits of', num, 'is', digitSum)
```

pass statement

A **pass** statement is ignored by the Python interpreter. It has the following simple syntax:

```
pass
```

It is often used as a stub when we want to leave some functionality to be defined in the future. For example, let us assume you don't want to say how much a test result should be changed for moderation. Instead, you want it to be determined statistically from the result of the examination. However, you may not yet have thought about how to do it. So, you may define a function that just contains a **pass** statement. For example,

```
def moderate(result):
    pass
```

Also, **pass** statement can also be used when no action is required on some conditions. For example,

```
>>> for letter in 'world':
...     if letter == 'l':
...         pass
...     else:
...         print(letter, end='')
...
...
word
```

It is important to remember that only the **pass** statement is ignored, other statements if any in the block are not ignored. For example,

```
>>> for letter in 'world':
...     if letter == 'l':
...         pass
...         print(letter, end ='')
...     else:
...         print(letter, end='')
...
...
world
```

CT 01

What will be the output produced on the execution of the following code snippet?

```
>>> for letter in 'world':
...     if letter == 'l':
...         pass
...         print(letter, end ='')
...     print(letter, end='')
```

1.12.3 Compound Statements

A compound statement such as **if, while, for** statement is used to control the execution of some statements. A compound statement comprises one or more *clauses.* A clause comprises a header and a **suite.** Each **clause header** has a **keyword** at the start and a **colon** (:) at the end. A sequence of indented statements defines a **suite.**

```
num1 = int(input('Enter First Number: '))
num2 = int(input('Enter Second Number: '))
if num1 == num2: #<--Header
    print (num1, 'is equal to', num2) #<--Suite
elif num1 < num2:
    print (num1, 'is less than', num2)
else:
    print (num1, 'is greater than', num2)
```

1.12.4 break and continue statements, and else Clauses on Loops

break statement

On execution of a **break** statement, Python transfers the control to the statement following the loop in which the break statement occurs. When a loop is terminated abruptly by execution of a **break** statement, the else clause (if included in the looping construct **while** or **for**) is not executed. The following loop that aims at checking the primality of the numbers between 2 and 10 illustrates the use of a **break** statement.

```
for i in range(2, 11):
    for j in range(2, i):
        if i % j == 0:
            print(i, '=', j, '*', i//j)
            break
    else:
        print(i, 'is a prime number')
```

Output:

```
2 is a prime number
3 is a prime number
4 = 2 * 2
5 is a prime number
6 = 2 * 3
7 is a prime number
8 = 2 * 4
9 = 3 * 3
10 = 2 * 5
```

In the above code, `i` ranges between `2` to `10` (outer for loop) and `j` ranges from `2` to `i` (inner **for** loop). For each value of `i`, the condition `i % j == 0` is evaluated, and if it holds the **break** statement gets executed and the control exits the inner **for** loop. The exit from the **for** loop on the execution of the **break** statement being abrupt, the **else** clause in the inner **for** loop does not get executed. However, if for a given value of `i`, `i%j==0` is not satisfied for any value of `j` in the range `2` to `i`, then the **else** part of the inner **for** loop is executed.

continue statement

The **continue** statement is used to skip the following statements in the body of the **for** loop (**while** loop), and the execution of the loop begins again for the next iteration. The following example illustrates the use of the **continue** statement:

```
for n in range(1, 11):
    if n % 2 != 0:
        print(n, "is an odd number")
        continue
    print(n, "is an even number")
```

Output:

```
1 is an odd number
2 is an even number
3 is an odd number
4 is an even number
5 is an odd number
6 is an even number
7 is an odd number
8 is an even number
9 is an odd number
10 is an even number
```

In the above code, `n` ranges from `1` to `10`. If `n%2 != 0` evaluates to **`True`**, the statements:

```
print(n, "is an odd number")
continue
```

When the **`continue`** statement is executed, the remaining statements in the current iteration of the **`for`** loop (line 5) are skipped and the control moves to the next iteration of the **`for`** loop.

1.13 Strings

Python provides a built-in class called **`str`** to deal with the strings in a Python program. A **`str`** instance is a string of text enclosed between single or double quotes. In this book, we prefer to use single quotes. Next, we give some examples of strings:

```
>>> 'abc123^$@ghi'
'abc123^$@ghi'
>>> 'Hello\nHow are you\nI am good, how are you'
'Hello\nHow are you\nI am good, how are you'
>>> print('Hello\nHow are you\nI am good, how are you')
Hello
How are you
I am good, how are you
```

Note that a string may include special characters. When a string includes the character sequence **`\n`**, the **`print()`** function transfers the print control to the next line for each occurrence of the character sequence **`\n`**, as shown above. When a string is enclosed between the single quotes, a double quote mark may be included as part of the string. However, to include a single quote (`'`), it should be preceded by a backslash (`\`) character.

```
>>> 'Hello"How are you"I am good, how are you'
'Hello"How are you"I am good, how are you'
>>> 'Hello\'How are you\'I am good, how are you'
"Hello'How are you'I am good, how are you"
```

Similarly, if a string is enclosed between the double quote marks, then to include the double quote mark (") in the sting, we precede it by a backslash symbol (\"), for example,

```
>>> "abc\"def\"ghi"
'abc"def"ghi'
>>> print('abc\"def\"ghi')
abc"def"ghi
```

A sequence of symbols, enclosed between double quotes can easily contain single quotes (e.g. 'I'm possible'), while single quoted strings can also contain double quotes.

A string literal can span multiple lines, for example,

```
>>> ''' abc
... def
```

```
... ghi
... '''
    ' abc\ndef\nghi\n'
```

However, if you do not wish to include end of lines in a multiline string, then terminate each line with a backslash character, as shown below:

```
>>> '''Hello, How are you? \
... I am good. \
... How are you?'''
'Hello, How are you? I am good. How are you?'
```

Position of a character in a string is called its index, for example, in the string `'hello'`, the characters `'h'`, `'e'`, `'l'`, `'l'`, and `'o'` are stored at index `0`, `1`, `2`, `3`, `4`, respectively.

```
>>> greet = 'hello'
>>> greet[0]
'h'
>>> greet[4]
'o'
```

Another way to access the characters in a string is to use the indices –1, –2, –3, ... from the other end, last character of the string being at index `-1`, for example,

```
>>> print(greet[-5], greet[-4], greet[-3], greet[-2], greet[-1])
h e l l o
```

Strings in Python are `immutable`, i.e., a string cannot be modified. An attempt to modify a string will yield an error. For example,

```
>>> name = 'Komal'
>>> name[1] = 'o'
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    name[1] = 'o'
TypeError: 'str' object does not support item assignment
```

However, it is perfectly fine to create new strings and assign them to existing variables; for example,

```
>>> name = 'Kamal'
>>> name
'Kamal'
>>> name = 'Komal'
>>> name
'Komal'
```

An index outside the valid range of indices is said to be out of bounds. An attempt to access a component of a string using an index outside of the valid range of indices will lead to an error, for example,

```
>>> greet = 'hello'
>>> greet[7]
Traceback (most recent call last):
  File "<pyshell#13>", line 1, in <module>
    greet[7]
IndexError: string index out of range
```

Length of a string: The built-in function **`len()`** returns the length of a string, for example,

```
>>> len('hello')
5
>>> len('hello\nhow are you')
17
>>> len('hello\n\nhow are you')
18
```

As the character sequence **`\n`** denotes a single end of line character, it is counted only once while computing the length of the string.

Concatenation of strings: The + operator is used to concatenate the strings, in the sequence (left to right) in which they appear, for example,

```
>>> 'Hello,' + 'How are you?' + 'I am good.' + 'How are you?'
'Hello,How are you?I am good.How are you?'
>>> 'Hello, ' + 'How are you? ' + 'I am good. ' + 'How are you?'
'Hello, How are you? I am good. How are you?'
```

Multiplication Operator *: The multiplication operator produces a string, concatenated with itself a specified number of times, for example,

```
>>> 'Hello'*5
'HelloHelloHelloHelloHello'
>>> 'Hello'*0
''
```

Reverse of a String: Now we are ready to write a function `reverse(str1)` that generates a string having characters in the reversed sequence of the original string, for example,

```
>>> reverse('excellent')
'tnellecxe'
>>> reverse('tnellecxe')
'excellent'
```

We begin by initialising the reverse string `reverseStr` as the null string and concatenate the characters in the string `str1` to the string `reverseStr` one by one as shown below:

```
>>> def reverse(str1):
    '''
    Objective: To reverse a string
    Input Parameter: str1 - string
    Return Value: reverse of str1 - string
    '''
    reverseStr = ''
    for i in range(len(str1)):
        reverseStr = str1[i] + reverseStr
    return reverseStr
```



1. Write a program that accepts a string from the user and prints its reverse using the above function.
2. Rewrite the function `reverse(str1)` using negative indices.



Numerical Values to Strings: A numerical value may be converted to a string using the **`str()`** function, for example,

```
>>> 'I want ' + str(2) + ' cups of ice cream.'
'I want 2 cups of ice cream.'
```

1.13.1 String Slices

A subsequence of characters of a string is called a slice, for example,

```
>>> 'excellent'[2:4]
'ce'
```

Note that a slice includes all characters in the string beginning the *start* index (2 in the above example) and up to the *finish* index (4 in the above example), including the character at the *start* index, but excluding the character at the *stop* index.

Considering the non-negative indices, when the finish index of a slice is lower than or equal to the *finish* index, Python yields the null string '', for example,

```
>>> 'excellent'[4:0]
''
>>> 'excellent'[3:3]
''
>>> 'excellent'[-5:0]
''
```

Note that the index –5 is the same as the index 4, so the slice `[-5:0]` is the same as the slice, `[4:0]` for which `4>0`.

Let us see some more examples of slicing:

```
>>> 'excellent'[-4:-1]
'len'
```

The slice `'excellent'[-4:-1]` includes all characters beginning the one at index `-4 (l)`, up to index `-1`, excluding the character at index `-1`,

```
>>> 'excellent'[-1:-4]
''
```

As we cannot count up from `-1` to `-4`. Python yields a null string for the slice `'excellent'[-1:-4]`. However, it is fine to mix negative and non-negative indices while defining a slice, for example,

```
>>> 'excellent'[3:-1]
'ellen'
>>> 'excellent'[3:8]
'ellen'
```

Note that the slice `[3:-1]` includes the characters at indices, `3, 4, 5, 6, 7` because the index `8` is the same as index `-1`. So, Python interprets the slice `[3:-1]` as the slice `[3:8]`. Sometimes, we leave the start index or finish index unspecified. Python substitutes `0` for the unspecified start index and length of the string for the unspecified finish index, for example,

```
>>> 'excellent'[-9:]
'excellent'
>>> 'excellent'[-7:]
'cellent'
>>> 'excellent'[2:]
'cellent'
>>> 'excellent'[:6]
'excell'
```

If the start index specified in a slice lies beyond the valid range of indices, it is replaced with the index 0, for example,

```
>>> 'excellent'[-12:6]
'excell'
```

Similarly, if the finish index specified in a slice is beyond the allowable range, it is replaced by the length of the string. less than the least valid index, it is replaced with the index 0, for example,

```
>>> 'excellent'[-12:26]
'excellent'
>>> 'excellent'[1:26]
'xcellent'
>>> 'excellent'[:]
'excellent'
```

Finally, for any string `s, s[:n] + s[n:]` always yields, `s`, irrespective of the value of `n`, for example,

```
>>> 'excellent'[:6] + 'excellent'[6:]
'excellent'
>>> 'excellent'[:-6] + 'excellent'[-6:]
'excellent'
```

```
>>> 'excellent'[:26] + 'excellent'[26:]
'excellent'
```

1.13.2 String Methods

Python provides several functions, also called methods for manipulating the strings. As the string is an immutable object, all the methods that operate on a string leave the original string unchanged. To invoke a method associated with a string object, the string object is followed by a dot, followed by the name of the method, followed by a pair of parentheses that encloses the arguments (if any) required for invoking the method. For example, given a string object, the method **`count`** returns the number of occurrences of the string passed as the argument in the given string; for example,

```
>>> 'excellent'.count('e')
3
>>> 'excellent'.count('excel')
1
>>> 'excellent'.count('excels')
0
```

Next, let us find the number of vowels in the string `'Encyclopedia'`.

```
>>> vowels = 'AEIOUaeiou'
>>> vowelCount = 0
>>> for ch in vowels:
...     vowelCount += 'Encyclopedia'.count(ch)
...
>>> print(vowelCount)
5
```

Next, we examine the functionality of some more methods for the string objects on a string object, say, `s`:

- **`s.lower()`, `s.upper()`** : The method **`lower()`** returns the lowercase version of the string. Similarly, the method **`upper()`** returns the uppercase version of the string, for example,

```
>>> emailId1 = 'orange@gmail.com'
>>> emailId2 = 'Orange@gmail.com'
>>> emailId1 == emailId2
False
>>> emailId1.lower() == emailId2.lower()
True
```

- **`s.title()`:** The method **`title()`** returns a string which has the first letter of every word of the original string converted to uppercase, while the remaining letters are converted to lowercase. For example,

```
>>> message = "Hello \'how are you?\' I AM GOOD, how are you?"
>>> message.title()
"Hello 'How Are You?' I Am Good, How Are You?"
```

- **`s.capitalize()`:** The method capitalize() returns a string which has all the letters in the original string converted to uppercase. For example,

```
>>> message = "Hello \'How are You?\'I am Good, How are You?"
>>> message.capitalize()
"Hello 'how are you?'i am good, how are you?"
```

- **`s.strip()`, `s.lstrip()`,** and **`s.rstrip()`:** The method strip() returns a string formed from the original string by removing the whitespace characters (space, tab, linefeed, return, form feed, and vertical tab) appearing at the beginning and end of the string. For example,

```
>>> '       Hello How are you!      '.strip()
'Hello How are you!'
```

Note that the method removes only the leading and trailing spaces, the white space appearing elsewhere is retained.

Let us examine another example,

```
>>> msg = '''    Hello How are you!
... I am Good,   Sir,
>>> Hello How are you    '''
>>> msg1 = msg.strip()
>>> msg1
'Hello How are you!\nI am Good,   Sir,\nHow are you'
>>> print(msg1)
Hello How are you!
I am Good,   Sir,
How are you
```

To remove the whitespace only from the beginning or end of the string, we use the Python string methods `lstrip()` and `rstrip()`, as illustrated below:

```
>>> '       Hello How are you!      '.lstrip()
'Hello How are you!      '
>>> '       Hello How are you!      '.rstrip()
'       Hello How are you!'
```

- **`s.isupper()`:** The method `isupper()` tests whether the string comprises only uppercase characters. For example,

```
>>> msg1 = 'SURAIYAJABIN'
>>> msg1.isupper()
True
>>> msg2 = 'SuraiyaJabin'
>>> msg2.isupper()
False
```

As the string msg1 comprises all uppercase letters, when the method `isupper()` is applied to it, the method returns **`True`**. However, as string msg2 comprises a mix of uppercase and lowercase characters, when the method `isupper()` is applied to it, the method returns **`False`**.

- **`s.islower()`** The method islower() tests whether the string comprises only lowercase characters, For example,

```
>>> msg1 = 'suraiyajabin'
>>> msg1.islower()
True
>>> msg2 = 'SuraiyaJabin'
>>> msg2.islower()
False
```

As the string msg1 comprises all lowercase letters, when the method is `lower()` is applied to it, the method returns **`True`**. However, since the string msg2 comprises a mix of uppercase and lowercase characters, when the method `islower()` is applied to it, the method returns **`False`**.

- **`s.isalpha()`:** The method isalpha() checks whether a string comprises only alphabets, for example,

```
>>> 'SuraiyaJabin'.isalpha()
True
>>> 'Suraiya Jabin'.isalpha()
False
```

In the first function call, the method `isalpha()` returned **`True`** as the string comprises only the alphabets of the English language. However, the second function call returned **`False`** because the string included a space.

- **`s.isdigit()`:** The method `isdigit()` checks whether a string comprises only digits. For example,

```
>>> '9898989898'.isdigit()
True
```

The above function call returns **`True`** since the string '9898989898' is composed of digits only. However, the following function call returns **`False`** as the string includes the symbols + and −, apart from the digits.

```
>>> '+91-9898989898'.isdigit()
False
```

- **s.isalnum():** A valid user name may be allowed to contain only alphabets and digits. The method `isalnum()` enables us to check whether the username entered by a user comprises only alphabets and digits. For example,

```
>>> userId = 'Prodyot123'
>>> userId.isalnum()
    True
>>> password = 'secret@123'
>>> password.isalnum()
    False
```

Note that the first function call `userId.isalnum()` returns **`True`** as the string `userId` comprises only alphabets and digits. However, `password.isalnum()` yields **`False`** as the string `password` includes a special character `@`. Note that space character is not part of the alphabet.

- **`s.isspace()`:** The method tests whether the string comprises only space, for example,

```
>>> ' '.isspace()
    True
>>> '      '.isspace()
    True
```

- **`s.startswith('other')`**: The method **`startswith('other')`** tests whether the string starts with the `other` string, passed as a parameter, for example,

```
>>> 'unethical'.startswith('un')
    True
>>> 'unethical'.startswith('ethical')
    False
```

- **`s.endswith('other')`:** The method **`endswith('other')`** tests whether the string ends with the `other` string, passed as a parameter, for example,

```
>>> 'ethically'.endswith('ly')
    True
>>> 'ethically'.endswith('yl')
    False
```

- **`s.find('other')`**: Finds the index where the first occurrence of the `other` string passed as a parameter begins in the string. **`find('other')`** searches for the given other string (not a regular expression) within `s`, and returns the first index where it begins or –1 if not found, for example,

```
>>> colors = 'green, red, blue, red, red, green'
>>> print('colors.find(\'red\'): ', colors.find('red'))
    colors.find('red'):  7
>>> print('colors.find(\'pink\'): ', colors.find('pink'))
    colors.find('pink'):  -1
```

- **`s.rfind(subStr)`**: The method `rfind()` returns the index of the last occurrence of a substring in the given string. For example,

```
>>> colors = 'green, red, blue, red, red, green'
>>> colors.rfind('red')
    23
```

- **`s.index(subStr)`:** The method `index(subStr)` returns the index of the first occurrence of the substring passed as an argument. But if the string specified as the argument is not found in the given string, Python raises an exception. Thus, if the argument `(subStr)` is a substring of the given string (s), the methods `find()` and `index` yield the same result. For example,

```
>>> colors.find('blue')
    12
>>> colors.index('blue')
```

```
    12
>>> colors.find('pink')
    -1
>>> colors.index('pink')
    Traceback (most recent call last):
      File "<pyshell#42>", line 1, in <module>
        colors.index('pink')
    ValueError: substring not found
```

- **`s.replace('old', 'new')`**: returns a string where all occurrences of `old` have been replaced by `new`, for example,

```
>>> message = 'Amey my friend, Amey my guide'
>>> message.replace('Amey', 'Vihan')
    'Vihan my friend, Vihan my guide'
```

- **`s.split(delimit)`**: The method **`split(delimit)`** returns a list of substrings, separated by the given delimiter: `delimit`. The delimiter must be a string. When no delimiter is specified, Python splits the string on the default delimiter, whitespace, i.e., any of the characters: **`' '`**, **`'\n'`**, **`'\t'`**, for example,

```
>>> '''Hello,\tHow are you?
... I am fine
... How are you'''.split()
    ['Hello,'How', 'are', 'you?', 'I', 'am', 'fine', 'How', 'are', 'you']
```

- **`s.partitition(delimit)`**: The method `partition(delimit)` splits the string into three parts, part of the string before the delimiter, the delimiter itself, and the part of the string after the delimiter, as illustrated below:

```
>>> txt = 'Amey my friend, Amey my guide'
>>> txt.partition(',')
    ('Amey my friend', ',', ' Amey my guide')
```

- **`s.join(list)`**: Joins the strings in the given list together using the string `s` as the delimiter.

```
>>> '>>'.join(['a', 'b', 'c', 'd'])
    'a>>b>>c>>d'
>>> ' '.join(['Hello,', '\tHow', 'are', 'you?\n', 'I', 'am', 'fine\n', 'How', 'are', 'you?'])
    'Hello, \tHow are you?\n I am fine\n How are you?'
>>> print(' '.join(['Hello,', '\tHow', 'are', 'you?\n', 'I', 'am', 'fine\n', 'How',
... 'are', 'you?']))
    Hello,    How are you?
    I am fine
    How are you?
```

1.14 Lists

Python supports another built-in list type called *list*. A list comprises a sequence of objects enclosed in square brackets []. As the characters in a string are indexed, so are the elements of a list. We can also apply the **`len()`** function to find the number of elements in a list. Slicing also applies to the lists just like strings.

```
>>> colors = ['red', 'green', 'blue', 'yellow', 'orange', 'white', 'black']
>>> colors[3]
    'yellow'
>>> colors[-2]
    'white'
>>> colors[3:6]
    ['yellow', 'orange', 'white']
>>> len(colors)
    7
>>> len(colors[3:6])
```

```
    3
>>> colors=colors[:3]
>>> colors
    ['red', 'green', 'blue']
>>> colors = ['red', 'green', 'blue', 'yellow', 'orange', 'white', 'black']
>>> colors[1:-1]
    ['green','blue','yellow','orange','white']
```

A pair of square brackets **[ ]** denotes an *empty list*. The operator + concatenates two lists, for example,

```
>>> [1, 2] + [3, 4, 5] + [6, 7]
    [1, 2, 3, 4, 5, 6, 7]
>>> [1, 2, 3] + []
    [1, 2, 3]
```

Multiplication Operator *: The multiplication operator produces a list, concatenated with itself a specified number of times, for example,

```
>>> [10, 20, 30]*3
    [10, 20, 30, 10, 20, 30, 10, 20, 30]
>>> [10, 20, 30]*0
    []
```

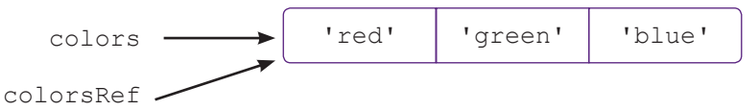
1.14.1 List Assignment



A list assignment does not create a new copy of the list. Instead, on the execution of the assignment statement, the variable on the left-hand side of the assignment statement refers to the list on the right-hand side of the assignment statement; for example,

```
>>> colors=['red', 'green', 'blue']
>>> colors
    ['red', 'green', 'blue']
>>> colorsRef = colors
>>> id(colors)
    1777056884928
>>> id(colorsRef)
    1777056884928
```

Note that the variables `colors` and `colorsRef` refer to the same list of objects having id: `1777056884928`.



Slicing

In the last chapter, we used slicing for accessing a subsequence of elements in a string. In this chapter, we will use slicing to access the subsequences of lists and tuples. Recall the following syntax for specifying a slice:

`<start>:<finish> [:<step>]>.`

Note that the `increment` is optional. Below we give some examples of slicing,

```
>>> colors = ['red', 'green', 'blue', 'yellow', 'orange', 'white', 'black']
>>> colors[3:6]
    ['yellow', 'orange', 'white']
```

The slice [3:6] yields a list comprising the elements `colors[3]`, `colors[4]`, and `colors[5]`, but not `colors[6]`.

The slice [:3] yields the list comprising the list elements up to index 3 (excluding the element at index 3, For example,

```
>>> colors [:3]
    ['red', 'green', 'blue']
```

Note that `colors[-1]` and `colors[6]` refer to the same position in list colors. So, the slice `[1, -1]` yields the list comprising the list elements `colors[1]`, `colors[2]`, …, `colors[5]` excluding `colors[6]`.

```
>>> colors[1:-1]
    ['green','blue','yellow','orange','white']
```

The slice [::2] yields a list, comprising the alternate elements of the list, because the step size is 2:

```
>>> colors[::2]
    ['red', 'blue', 'orange', 'black']
```

The slice `[1:7:3]` yields a list, comprising the elements beginning at index 1 and up to index 7 (excluding the element at index 7), in steps of size 3. Thus, the expression `colors[1:7:3]` yields a list comprising the elements `colors[1]` and `colors[4]`:

```
>>> colors[1:7:3]
    ['green', 'orange']
```

Nested List: A list that contains another list is called a nested list, for example,

```
>>> subjectCodes = [['Sanskrit', 78], ['English', 85], ['Maths', 88], ['Hindi', 90]]
>>> for code in subjectCodes:
...     print(code)
...
...
    ['Sanskrit', 78]
    ['English', 85]
    ['Maths', 88]
    ['Hindi', 90]
>>> for code in subjectCodes:
...     print(code[0])
...
...
    Sanskrit
    English
    Maths
    Hindi
```

Heterogeneous List: The items stored in a list can be of any type, such as numeric, string, boolean, list, or functions. This means that a list can contain any type of data and that we can iterate over it with any type of loop.

```
>>> mixedList = [1, [2, 3], 'three', 4.0]
```

Interestingly, the functions stored in a list can be invoked using the list index, for example,

```
>>> def f1():
...     return 'f1'
>>> def f2():
...     return 'f2'
...
>>> lst = [f1, f2]
>>> print(lst[0](), lst[1]())
    f1 f2
```

Functions on List

Let us apply some functions on the list:

```
>>> lst = ['Red', 'Green']
>>> list2 = [10, 20, 30]
>>> lst += ['Blue']
>>> lst
```

```
['Red', 'Green', 'Blue']
>>> len(lst)
3
>>> list2 * 2
[10, 20, 30, 10, 20, 30]
>>> min(list2)
10
>>> max(list2)
30
>>> sum(list2)
60
```

1.14.2 List Methods

Likewise, Python strings, methods are defined for list objects (say `lst`) also.

- **`lst.append(elem)`** : The method **`append()`** inserts the object `elem`, passed as an argument, at the end of the list. Note that we do not get the new list, it modifies the original list, for example,

```
>>> a = [10, 20, 30, 40]
>>> a.append(35)
>>> a
[10, 20, 30, 40, 35]
```

- **`lst.insert(index, elem)`** : The method **`insert()`** inserts the object `elem`, passed as an argument, at the specified index and shifts the existing objects to the right, for example,

```
>>> names = ['Aryan', 'Anthony', 'Sunpreet', 'Venkatesh']
>>> names.insert(2, 'Samantha')
>>> print(names)
['Aryan', 'Anthony', 'Samantha', 'Sunpreet', Venkatesh']
```

- **`lst.extend(lst)`** : The method **`extend()`** inserts the elements in the list `lst`, passed as an argument, at the end of the elements of the list `lst`. `list2` elements are added to the end of the list, for example,

```
>>> a = [10, 20, 30]
>>> a.extend([40,50])
>>> print(a)
[10, 20, 30, 40, 50]
```



If a = [1, 2, 3]

What is the difference between the operations:

`a.extend([40,50])` and `a + [40, 50]`





- **`lst.index(elem)`** : The method **`index()`** returns the index of the first occurrence of the element `elem` in the list. If `elem` does not appear in the list `lst`, it throws a `ValueError`, for example,

```
>>> names = ['Aryan', 'Anthony', 'Samantha', 'Sunpreet', 'Venkatesh']
>>> names.index('Samantha')
2
>>> names.index('Arjun')
Traceback (most recent call last):
  File '<pyshell#3>', line 1, in <module>
    names.index('Arjun')
ValueError: 'Arjun' is not in list
```

Note that to avoid **`ValueError`**, we can use the membership operator to check the membership of an element.

```
>>> if 'Arjun' in names:
...     names.index('Arjun')
```

```
... else:
...     print('Element doesn\'t exist in the list')
...
Element doesn't exist in the list
```

- **`lst.reverse()`**: The method **`reverse()`** reverses the order of the elements in the list, for example,

```
>>> names = ['Aryan', 'Anthony', 'Samantha', 'Sunpreet', 'Venkatesh']
>>> names.reverse()
>>> print(names)
['Venkatesh', 'Sunpreet', 'Samantha', 'Anthony', 'Aryan']
```

- **`lst.sort()`**: The method arranges the elements of the list in the ascending order, for example,

```
>>> names.sort()
>>> print(names)
['Anthony', 'Aryan', 'Samantha', 'Sunpreet', 'Venkatesh']
```

To sort the elements in a list, we set the argument `reverse = True`, as shown below:

```
>>> names = ['Aryan', 'Anthony', 'Samantha', 'Sunpreet', 'Venkatesh']
>>> names.sort(reverse = True)
>>> names
['Venkatesh', 'Sunpreet', 'Samantha', 'Aryan', 'Anthony']
```

When the argument `key=function` is specified, `function` is applied to each element of the list, and the values in the list are arranged according to the ascending order of the function values. For example, we define a function `f(x)` which returns `1/x`.

```
>>> def f(x):
...     return 1/x
...
>>> lst = [4, 16, 10, 2]
>>> lst.sort(key=f)
>>> lst
    [16, 10, 4, 2]
```

So, applying the function `f` on the elements 4, 16, 10, 2 of `lst` yields the numbers 1/4, 1/16, 1/10, 1/2. An ascending order listing of these numbers would be 1/16, 1/10, 1/4, 1/2. As this listing of numbers correspond to the numbers 16, 10, 4, 2, respectively, the list sorted according to the function values becomes [16, 10, 4, 2].

- **`lst.remove(element)`**: The method **`remove(element)`** searches for the first instance of the `element` in the list `lst` and removes it. If `element` is not found in lst, the method `remove` throws `ValueError`.

```
>>> a = [10, 20, 30, 10, 50, 20, 60, 20, 30, 55]
>>> a.remove(20)
>>> a
[10, 30, 10, 50, 20, 60, 20, 30, 55]
```

- **`lst.pop(index)`**: The method removes the element from the specified `index` and returns the element removed from the list. If the `index` is omitted, the rightmost element is returned, for example,

```
>>> a = [10, 20, 30, 10, 50, 20, 60, 20, 30, 55]
>>> a.pop(3) #a=[10, 20, 30, 10, 50, 20, 60, 20, 30, 55]
10
>>> a
[10, 20, 30, 10, 50, 20, 60, 20, 30, 55]
>>> a.pop()
55
```

Creating a Sorted List

The function `sorted` returns a sorted list comprising the elements of the list passed as argument, but without modifying it.

```
>>> lst = ['Physics', 'Chemistry', 'Maths', 'Computer Sc.']
>>> sorted(lst)
    ['Chemistry', 'Computer Sc.', 'Maths', 'Physics']
>>> lst
    ['Physics', 'Chemistry', 'Maths', 'Computer Sc.']
```

- Membership operation **`in`:** The operator **`in`** checks for the membership of a data object in the list and returns **`True`** or **`False`**, depending on whether the specified object is included in the list.

```
>>> 3 in [1, 2, 3, 4]
    True
>>> 5 in [1, 2, 3, 4]
    False
```

1.15 Tuples

A tuple is a sequence of elements, separated by commas and usually enclosed between parentheses, for example,

```
>>> bDate = (30, 'Septmber', 1987)
>>> bDate
    (30, 'September', 1987)
```

It is perfectly fine to drop the parentheses around the comma separated values, for example,

```
>>> bDate = 30, 'September', 1987
>>> bDate
    (30, 'September', 1987)
```

An empty pair of parentheses defines an empty tuple, for example,

```
>>> empty = ()
>>> empty
    ()
>>> type(empty)
    <class 'tuple'>
```

Interestingly, a single value enclosed in a string does not define a tuple as it is interpreted as an expression enclosed in a pair of parentheses, for example,

```
>>> republic = (1950)
>>> type(republic)
    <class 'int'>
>>> republic
    1950
```

To express a tuple comprising a single value, the value in the tuple is followed by a comma, for example,

```
>>> republic = (1950)
>>> republic
    1950
>>> republic = (1950,)
>>> republic
    (1950,)
```

A tuple assignment is often used to assign values to several variables in a single assignment statement, for example, any of the following would work equally well to assign values to `30, 'September', 1987` to the variables `bDay, bMonth, bYear:`

```
>>> bDay, bMonth, bYear = 30,'September',1987
>>> bDay, bMonth, bYear
    (30, 'September', 1987)
>>> bDay, bMonth, bYear = (30, 'September', 1987)
>>> bDay, bMonth, bYear
```

```
    (30, 'September', 1987)
>>> (bDay, bMonth, bYear) = (30, 'September', 1987)
>>> bDay, bMonth, bYear
    (30, 'September', 1987)
>>> (bDay, bMonth, bYear) = 30, 'September', 1987
>>> bDay, bMonth, bYear
    (30, 'September', 1987)
>>> bDay, bMonth, bYear = bDate
>>> bDay, bMonth, bYear
    (30, 'September', 1987)
```

Note that in each of the above examples, the tuple on the left hand side is assigned the tuple on the right hand side. Indexing and slicing works for tuples just like it works for strings and lists. The **`len()`** function also works on tuples and returns the number of elements in the tuple. A tuple is an immutable object and an attempt to substitute an element of a tuple by another value yields type error because the tuple does not support assignment, for example,

```
>>> bDate = (30,'September',1987)
>>> len(bDate)
    3
>>> bDate[:-1]
    (30, 'September')
>>> bDate[0] = 20
    Traceback (most recent call last):
      File "<pyshell#213>", line 1, in <module>
        bDate[0] = 20
    TypeError: 'tuple' object does not support item assignment
```

As seen above, a tuple is an immutable object. However, an element of a tuple such as a list or a dictionary may be mutable, for example,

```
>>> discipline = ('Science', ['Physics', 'Chemistry', 'Biology'])
>>> discipline[1].append('Environment Science')
>>> discipline
    ('Science', ['Physics', 'Chemistry', 'Biology', 'Environment Science'])
```

The *empty* tuple is simply a pair of empty parenthesis. The elements of a tuple can be accessed in the same way that a list can.

Next, we see some examples of tuple slices:

```
>>> myTuple = (22, 30, 'Hello', 2.4, 2, 'Python', (890,900), 45)
>>> myTuple[1:4] # yields 2nd to 4th element
    (30, 'Hello', 2.4)
>>> myTuple[:4]  # yields 1st to 4th element
    (22, 30, 'Hello', 2.4)
>>> myTuple[4:]  # yields 5th to the last element
    (2, 'Python', (890, 900), 45)
>>> myTuple[:]  # yields first to the last element
    (22, 30, 'Hello', 2.4, 2, 'Python', (890, 900), 45)
>>> myTuple[::2] # yields alternate elements, starts at 0
    (22, 'Hello', 2, (890, 900))
>>> myTuple[-4] # yields fourth element from the end
    2
>>> myTuple[::-1] # yields tuple in reverse
    (45, (890, 900), 'Python', 2, 2.4, 'Hello', 30, 22)
```

1.15.1 Converting a sequence to Tuple

A function **`tuple()`** is used to convert a sequence of elements to a tuple.

```
>>> vowels = 'aeiou'
>>> tuple(vowels)
```

```
    ('a', 'e', 'i', 'o', 'u')
>>> tuple(['a', 'e', 'i', 'o', 'u'])
    ('a', 'e', 'i', 'o', 'u')
```

1.15.2 Nested Tuple

As mentioned earlier, tuples can contain other compound objects, including lists, dictionaries, and other tuples. So, as a special case, tuples may be nested inside of other tuples.

```
>>> birthDates = ((8, 'August'), (1, 'October'), (30, 'September'))
```

1.15.3 Tuple Operations

Likewise, strings and lists, Python also supports a variety of operations on tuples. Below we illustrate some important operations on tuples, let us consider the following tuples.

```
>>> greetings = ('Good Morning', 'Good Afternoon')
>>> marks = (78, 99, 34, 45)
>>> dateOfBirth = (1, 'October', 1990)
>>> greetings * 2 # Multiplication Operator *
    ('Good Morning', 'Good Afternoon', 'Good Morning', 'Good Afternoon')
>>> greetings = greetings + ('Good Night',) # concatenate +
>>> greetings
    ('Good Morning', 'Good Afternoon', 'Good Night')
>>> print(len(dateOfBirth),len(greetings)) # length of tuple
    3 3
>>> 'Good Afternoon' in greetings # Membership operation in
    True
```

1.15.4 Tuple Methods

Python provides several methods for manipulating the tuples. Like a string, a tuple is also an immutable object, All the methods that operate on a tuple leave the original tuple unchanged. To invoke a method associated with a tuple object, the tuple object is followed by a dot, followed by the method's name, followed by a pair of parentheses that enclose the arguments (if any) required for invoking the method. Next, we describe the functionality of some methods for the tuple objects on a tuple object, say, `t`.

- **`t.count(element)`:** The method returns the number of times the given element appears in a tuple, for example,

```
>>> (0, 1, 2, 3, 2, 3, 1, 3, 2).count(2)
    3
>>> age = (20, 18, 17, 19, 18, 18)
>>> age.count(18)
    3
```

- **`t.index(element)`:** The method returns the index of the first occurrence of an element in a tuple, for example,

```
>>> (0, 1, 2, 3, 2, 3, 1, 3, 2).index(3)
    3
```

- **`sorted(t)`:** The method sorts the tuple elements in ascending order by default. However, one can use the `reverse` parameter and sets its value to **`True`**. Note that the **`sorted()`** returns a list by default. So, we have to use **`tuple()`** to convert the list to a tuple, for example,

```
>>> myTuple = (5, 2, 24, 3, 1, 6, 7)
>>> sorted(myTuple)
    [1, 2, 3, 5, 6, 7, 24]
>>> myTuple
    (5, 2, 24, 3, 1, 6, 7)
```

Note that a tuple being an immutable object, when we sort a tuple, Python returns a list, leaving the original tuple unchanged.

However, if we are interested to obtain the result of sorting a tuple as a tuple, we can apply the tuple function to the result of sorting as shown below:

```
>>> tuple(sorted(myTuple))
    (1, 2, 3, 5, 6, 7, 24)
>>> tuple(sorted(myTuple, reverse=True))
    (24, 7, 6, 5, 3, 2, 1)
```

1.16 Dictionaries

A *dictionary* is an unordered set of `key:value` pairs. A dictionary may be specified by enclosing the `key:value` pairs in braces { }. An empty pair of braces denotes an empty dictionary. The keys in a dictionary are required to be unique. However, the same value may be associated with multiple keys. Objects of the immutable types, strings, numbers, and tuples are used as keys. However, the values associated with keys may be objects of any type. The syntax for defining and naming a dictionary is as follows:

```
>>> <identifier> = {key1:value1, key2:value2, ... }
>>> dictionaryName = {key1:value1, key2:value2, ... }
```

For Example,

```
>>> subjects = {'Sanskrit':78, 'English':85, 'Maths':88, 'Hindi':90}
>>> subjects
    {'Sanskrit': 78, 'English': 85, 'Maths': 88, 'Hindi': 90}
```

To access the value associated with a key, the dictionary object is followed by the key enclosed in square brackets, for example,

```
>>> subjects['English']
    85
```

An attempt to access a value associated with a non-existent key yields **`KeyError`**, for example,

```
>>> subject = 'history'
    subjects[subject]
    Traceback (most recent call last):
      File "<pyshell#19>", line 1, in <module>
        subjects[subject]
    KeyError: 'history'
```

To avoid **`KeyError`**, when in doubt, one may check for the existence of a key in the dictionary prior to accessing its value, for example,

```
>>> subject in subjects
    False
>>> if subject in subjects:
...     print(subjects['subject'])
... else:
...     print(subject, 'is not in the dictionary subjects')
...
    history is not in the dictionary subjects
```

Suppose, subject code for `'English'` changes to `185`, it can be modified using an assignment statement as shown below:

```
>>> subjects['English'] = 185
>>> subjects['English']
    185
>>> subjects
    {'Sanskrit': 78, 'English': 185, 'Maths': 88, 'Hindi': 90}
```

Next, we see some more examples of operations on dictionaries.

```
>>> myDict = {} # Create an empty dictionary
```

We can also use an assignment statement to add a key-value pair to a dictionary as illustrated below:

```
>>> myDict['a'] = 'alpha'
>>> myDict['b'] = 'beta'
>>> myDict['g'] = 'gamma'
>>> myDict
{'a': 'alpha', 'b': 'beta', 'g': 'gamma'}
```

As mentioned above, if we want to access the `value` corresponding to a `key 'b'`, the following can be used:

```
>>> myDict['b']
beta
```

To check the membership of a `key` in a dictionary, we may use `in` operator. It returns **`True`** when the `key` is present, **`False`** otherwise. For example, to check whether `'g'` is present in `myDict`, the following can be used:

```
>>> 'g' in myDict
True
```

In case, we try to access a `key` not present in the dictionary, we get a **`KeyError`**.

```
>>> print(myDict['o'])
Traceback (most recent call last):
  File '<pyshell#29>', line 1, in <module>
    print(myDict['o'])
KeyError: 'o'
```

1.16.1 Dictionary Methods

Python provides several methods for manipulating the dictionaries. Unlike strings and tuples, a dictionary is a mutable object. To invoke a method associated with a dictionary object, the dictionary object is followed by a dot, followed by the name of the method, followed by a pair of parentheses that encloses the arguments (if any) required for invoking the method. Next, we describe the functionality of some methods for the dictionaries on a dictionary object, say, `myDict`.

- **`myDict.keys()`:** The method **`keys()`** returns a `dict_keys` object comprising the set of keys included in the dictionary. We can check for membership of a key in a `dict_keys` and iterate over the keys in the `dict_keys` object. However, `dict_keys` does not support indexing, for example,

```
>>> subjects = {'Sanskrit': 78, 'English': 185, 'Maths': 88, 'Hindi': 90}
>>> subjects
{'Sanskrit': 78, 'English': 185, 'Maths': 88, 'Hindi': 90}
>>> subjects.keys()
dict_keys(['Sanskrit', 'English', 'Maths', 'Hindi'])
>>> 'English' in subjects.keys()
True
>>> for subject in subjects.keys():
...     print(subject, end = ' ')
...
Sanskrit English Maths Hindi
>>> subjects.keys()[0]
Traceback (most recent call last):
  File "<pyshell#38>", line 1, in <module>
    subjects.keys()[0]
TypeError: 'dict_keys' object is not subscriptable
```

- **`myDict.values()`:** The method **`values()`** returns a `dict_values` object comprising the set of values included in the dictionary. We can check for membership of a value in a `dict_values` and iterate over the values in the `dict_values` object. However, `dict_values` does not support indexing, for example,

```
>>> 78 in subjects.values()
True
```

```
>>> for subjectCode in subjects.values():
...     print(subjectCode, end=' ')
    78 185 88 90
```

- **myDict.items()** : The method **items()** returns a `dict_items` object comprising the set of items included in the dictionary. We can check for membership of an item in a `dict_items` and iterate over the values in the `dict_items` object. However, `dict_items` does not support indexing, for example,

```
>>> subjects = {'Sanskrit': 78, 'English': 185, 'Maths': 88, 'Hindi': 90}
>>> ('English', 185) in subjects.items()
    True
>>> for subject in subjects.items():
...     print(subject, end=' ')
    ('Sanskrit', 78) ('English', 185) ('Maths', 88) ('Hindi', 90)
```

Consider a dictionary, `myDict`. There are two ways of accessing all the keys of `myDict`.

```
>>> myDict = { 'b':'beta', 'g':'gamma', 'a':'alpha' }
```

First way:

```
>>> for key in myDict:
...     print(key,end=' ')
...
    b g a
```

Second way:

```
>>> print(myDict.keys())
    dict_keys(['b', 'g', 'a'])
```

- **myDict.update()**: **update()** method is used to add some key-value pairs in an existing dictionary, for example,

```
>>> loginDetails = {'Aryan':'a7@101', 'Sunpreet':'sun@102', 'Anthony':'ant@103' }
>>> moreUsers = {'Samantha':'sam@104','Venkatesh':'ven@105'}
>>> loginDetails.update(moreUsers)
>>> print(loginDetails)
    {'Aryan': 'a7@101', 'Sunpreet': 'sun@102', 'Anthony': 'ant@103', 'Samantha': 'sam@104',
    'Venkatesh': 'ven@105'}
```

- **myDict.pop(myKey)** : **pop()** method removes the item with the specified `key`, for example,

```
>>> Months = {3:'March', 11:'November', 12: 'December', 1:'January'}
>>> Months.pop(11)
    'November'
>>> Months
    {3: 'March', 12: 'December', 1: 'January'}
```

- **myDict.popitem()** : **popitem()** method removes the last inserted item from the dictionary, for example,

```
>>> Months = {3:'March', 11:'November', 12: 'December', 1:'January'}
>>> Months.popitem()
    (1, 'January')
>>> Months
    {3: 'March', 11: 'November', 12: 'December'}
```

Python has a keyword `del` to remove an item with the specified key. It can also be used to delete the entire dictionary, for example,

```
>>> Months = {3:'March', 11:'November', 12: 'December', 1:'January'}
>>> del Months[11]
>>> Months
    {3: 'March', 12: 'December', 1: 'January'}
>>> del Months #delete Months
```

- **myDict.clear()**: **clear()** method empties the dictionary, for example,

```
>>> Months = {3:'March', 11:'November', 12: 'December', 1:'January'}
>>> Months.clear()
>>> Months
    {}
```

- **myDict.sorted()**: **sorted()** method is used to sort by the `keys` or by the `values`. However, it sorts by default on keys, for example,

```
>>> myDict = { 'b':'beta', 'g':'gamma', 'a':'alpha' }
>>> sorted(myDict) # sorted(myDict.keys())
    ['a', 'b', 'g']
>>> sorted(myDict.values())
    ['alpha', 'beta', 'gamma']
```

Use **sorted()** method to print all key value pairs with respect to keys.

```
>>> for key in sorted(myDict.keys()):
...     print(key,myDict[key])
    a alpha
    b beta
    g gamma
```

- **myDict.get(myKey)**: similar to **myDict[myKey]**, **get()** method returns the value of the item with the specified key, for example,

```
>>> myDict = { 'b':'beta', 'g':'gamma', 'a':'alpha' }
>>> myDict.get('g')
    'gamma'
>>> myDict['g']
    'gamma'
```

- **myDict.copy()**: **copy()** method returns the copy of the dictionary, for example,

```
>>> myDict = { 'b':'beta', 'g':'gamma', 'a':'alpha' }
>>> myDictCopy = myDict.copy()
>>> myDictCopy
    {'b': 'beta', 'g': 'gamma', 'a': 'alpha'}
```

Note that the variables `myDict` and `myDictCopy` refer to two different objects having id `1222244876160` and `1222244887808`, respectively, for example,

```
>>> id(myDict)
    1222244876160
>>> id(myDictCopy)
    1222244887808
```

- **myDict.fromkeys(keys, value)**: **fromkeys()** method returns a dictionary with the specified keys and the specified value, for example,

```
>>> subjects = {}
>>> subjectTitle = {'Physics', 'Mathematics', 'Computer Sc.'}
>>> marks = 0
>>> subjects.fromkeys(subjectTitle, marks)
    {'Computer Sc.': 0, 'Mathematics': 0, 'Physics': 0}
```

- **myDict.setdefault(key, [value])**: **setdefault()** method returns the value of the item with the specified `key`. However, if the `key` does not exist, then the `key` is inserted in the dictionary and the key is associated with the value passed as the second parameter, for example,

```
>>> myDict = { 'b':'beta', 'g':'gamma', 'a':'alpha' }
>>> myDict.setdefault('a')
    'alpha'
```

```
>>> myDict.setdefault('d','delta')
    'delta'
>>> myDict
    {'b': 'beta', 'g': 'gamma', 'a': 'alpha', 'd': 'delta'}
```

Solved Programming Questions

1. Write a function `max3(n1, n2, n3)` that finds the maximum out of three numbers `n1, n2, and n3`.

```
def max3(n1, n2, n3):
    '''
    Objective: To find maximum of three numbers
    Input Parameters: n1, n2, n3 - numeric values
    Return Value: maximum of n1, n2, n3 - numeric value
    '''
    '''
    Approach:
    Compare each number with the other two numbers. Assign the maximum to the maximum
    among n1, n2, and n3.
    '''
    if (n1 >= n2) and (n1 >= n3):
        maximum = n1
    elif (n2 >= n1) and (n2 >= n3):
        maximum = n2
    else:
        maximum = n3
    return maximum
n1 = int(input('Enter first number '))
n2 = int(input('Enter second number '))
n3 = int(input('Enter third number '))
print('Maximum of ', n1, n2, n3, ':', max3(n1, n2, n3))
```

2. Write a function `max2(n1,n2)` that finds the maximum of two numbers and then make use of the function `max2(n1,n2)` to write the function `max3(n1, n2, n3)` to find the maximum of the numbers `n1, n2,` and `n3`.

```
def max2(n1, n2):
    '''
    Objective: To find maximum of two numbers
    Input Parameters: n1, n2 - numeric values
    Return Value: maximum of n1, n2 - numeric value
    '''
    '''
    Approach:
    Compare two numbers and return the maximum of two numbers.
    '''

    if n1 > n2:
        return n1
    else:
        return n2
def max3(n1, n2, n3):
    '''
    Objective: To find maximum of three numbers
    Input Parameters: n1, n2, n3 - numeric values
    Return Value: maximum of n1, n2, n3 - numeric value
    Approach: Find maximum of two numbers at a time
```

```
    '''
    return max2(max2(n1, n2), n3)
n1 = int(input('Enter first number :'))
n2 = int(input('Enter second number : '))
n3 = int(input('Enter third number : '))
print('Maximum of', n1, n2, n3, ':', max3(n1, n2, n3))
```

3. A string is a palindrome, if it reads the same from both the ends, for example, `'madam'`, `'rotator'`, `'level'`, `'abcde0edcba'`. Write a function `palindrome(string)` to check whether a given string is a palindrome. Write a program that takes a string as input from a user and makes use of the function **`palindrome()`** to and print an appropriate message depending on whether the input string is a palindrome.

```
def palindrome(string):
    '''
    Objective: To check whether the given string is a palindrome.
    Input Parameter: string – a Python string
    Return Value: True if string is a palindrome, False otherwise
    '''
    '''
    Approach:
    For each position i in string, check if elements at i and len(string)-(i-1) th
    locations are the same. If this holds true for all i in range [1, len(string)/2]
    return True else False.
    '''
    n = len(string)
    if n == 0: #null string
        return True
    else:
        j = n-1 #index of rightmost character
        for i in range(n//2+1):
            if string[i] != string[j]:
                return False #character mismatch
            j -= 1
        return True
'''
Objective: To check whether the given string is a palindrome
Input parameter: None
Return value: None
'''
'''
Approach: To use function palindrome
'''
myString = input('Enter a string to check:')
if palindrome(myString):
    print('String %s is a Palindrome' %(myString))
else:
    print('String %s is not a Palindrome' %(myString))
```

4. Define a function `wordLength(lst)` which takes a list of words and returns a list of tuples where each tuple comprises a word and its corresponding length. Make use of the function **`wordLength()`** to write a program that accepts a list of words from a user and prints a list of tuples where each tuple is word-length pair, that is, word and its corresponding length.

```
def wordLength(lst):
    '''
    Purpose: To return word-length pairs for the words in a list.
```

```
    Input Parameter: lst - list containing words
    Return Value: List of (word, length) tuples
    '''
    '''
    Approach:
    initialise an empty word-length list wordLenPairs
    for each word in lst:
    append the tuple (word, length) to wordLenPairs
    '''
    wordLenPairs = list()
    for word in lst:
        wordLenPairs.append((word, len(word)))
    return wordLenPairs
#Objective: Accept a word-list from a user and print list of (word.lenngth)tuples
lst = eval(input('Enter the list: '))
print('List of tuples:: ', wordLength(lst))
```

5. Write a function to remove the duplicate elements from a list of integers. You are not allowed to create a new list.

```
def removeDuplicates(lst):
    '''
    Objective: To return another list without any duplicates from the input list
    Input Parameter: lst- a list, possibly with duplicate elements
    Return Value: None
    Side Effect: The modified list has the duplicates removed
    '''
    '''
    Approach: for each element check for its count in the list and
    remove it if count>0.
    '''
    for element in lst:
        while lst.count(element)>1:
            lst.remove(element)
lst = eval(input("Enter integer List"))
removeDuplicates(lst)
print(lst)
```

6. Write a function that accepts a list as input parameters and returns a list which has the duplicates removed from the list. The original list should remain unchanged.

```
def removeDuplicates(lst1):
    '''
    Objective: To return a new which has duplicates removed from lst1.
    Input Parameter: lst1- list with duplicate elements
    Return Value: lst2 - list without duplicate elements
    '''
    lst2 = []
    for element in lst1:
        if element not in lst2:
            #append it to lst2
            lst2.append(element)
    return lst2
# Objective: To remove duplicate from a list
myList = eval(input('Enter the list='))
result = removeDuplicates(myList)
print('List with no duplicates:' , result)
```

7. Write a program to find the number of four-digit palindrome numbers divisible by 11. Assume that you already have a function `palindrome(string)` that returns **`True`** if the input string is a palindrome and **`False`** otherwise.

   We will repeatedly check all the four-digit numbers for its divisibility by 11. If the number is divisible by 11, we convert it to the string using `str` function. Finally we will use the function `palindrome(str(n))` to check whether the number is a palindrome or not. For each match, the count is incremented.

```
def div11Palindromes1():
    '''
    Objective: To find the number of Four digit palindrome divisible by 11
    Input Parameter: None
    Return Value: count
    '''
    count = 0
    start = 1001 # note that 1001 is divisible by 11
    finish = 10000
    for n in range(start, finish, 11):
        if palindrome(str(n)):
            count+=1
    return count
```

8. Write a program to find the number of four-digit palindrome numbers divisible by 11. You are not allowed to use a function `palindrome(string)` that returns **`True`** if the input string is a palindrome and **`False`** otherwise.

```
def div11Palindromes():
    '''
    Objective: To find the number of four-digit palindrome divisible by 11
    Input Parameter: None
    Return Value: count of palindromes divisible by 11
    '''
    count = 0
    start = 1001 #1001 is divisible by 11
    finish = 10000
    for n in range(start, finish, 11):
        #if unit digit equals thousand's digit and
        #  tens digit equals hundreds digit,
        #then n is a palindrome
        if ((n//1000)%10 == n%10) and ((n//100)%10 == (n//10)%10):
            count+=1
    return count
#Print the number of 4-digit palindromes divisible by 11
print('No of 4-digit palindromes divisible by 11 is ', div11Palindromes())
```

9. Given a list (say, list1) of numbers, its cumulative list is another list (say, list2) whose ith element is the sum of all elements till the ith element of lst1. If a list is empty then its cumulative list will also be an empty list. Write a program that accepts a list of integers and displays its cumulative list.

```
def cumulativeList(lst1):
    '''
    Objective: To return another list whose ith element is the sum of all elements till
    ith element of lst1
    Input Parameter: lst1- list
    Return value: cumLst - list
    '''
    if len(lst1) == 0 :
        return []
```

```
    else:
        cumLst = [lst1[0]] # lst1[0] is sum of 1-element list
        for i in range(1,len(lst1)):
            nextElement = cumLst[i-1] + lst1[i]
            cumLst.append(nextElement)
        return cumLst


# Objective: To find the cumulative sum of elements of an integer list
myList = eval(input('Enter a list of numbers: '))
result = cumulativeList(myList)
print('List of cumulative sums: ' ,result)
```

10. Write a function `wordCountDict(lst)` to find the number of occurrences of each word in a list of words in the form of a dictionary. Apply the function `wordCountDict(lst)` to a list entered by the user.

```
def wordCountDict(lst):
    '''
    Objective: To determine word count from words given in a list.
    Input Parameter: lst - list containing words
    Return Value: wordCount - Dictionary containing the words and its count
    '''
    '''
    Approach:
    For each word w in list
        if w is in dictionary, increment its count
        else add w to the dictionary with 1 as its count
    '''
    wordCount = dict()
    for w in lst:
        if w in wordCount:
            wordCount[w] += 1
        else:
            wordCount[w] = 1
    return wordCount
#Objective: To determine word count from words given in list.
lst = eval(input('Enter the list: '))
print('Dictionary of word count:: ', wordCountDict(lst))
```

11. Can you use the function `wordCountDict(lst)` that you developed in Q10 to count the number of occurrences of numbers in a list?

    Answer: Yes, because the function does not make use of the fact that the objects in the list are strings. So, it is applicable to any list of objects, including heterogeneous lists. However, the objects in the list should be immutable because we are using them as keys in the dictionary.

12. Define a function `divisibleDivisors(n, k)` which takes `n` and `k` (`n>=k`)as its arguments and returns the number of divisors of `n` which are divisible by `k`. Make use of the function to accept the values of `n` and `k` from the user and print the number of divisors of `n` which are divisible by `k`.

```
def divisibleDivisors(n, k):
    '''
    Objective: Given n>=k, find the count of divisors of n which are divisible by k
    Input Parameter: n,k - Integers
    Return Value: count of divisors of n which are divisible by k
    '''
```

```
    assert n>=k
    count = 0
    for i in range(1, n + 1):
        if (n % i == 0 and i % k == 0):
            count += 1
    return count
n = int(input('Enter a number n:'))
k = int(input('Enter k, n>=k:'))
count = divisibleDivisors(n, k)
print('number of divisors of', n, 'divisible by', k, '= ', count)
```

13. German mathematician Christian Goldbach gave the following conjecture, popularly known as Goldbach's conjecture: Every even natural number greater than two is the sum of two prime numbers. Write a function goldbach (number) which takes a number as an argument and returns a list of tuples having all possible pair of prime numbers that sums to number.

```
def isPrime(num):
    '''
    Objective: Given a number num, check whether it is prime.
    Input Parameter: num - integer
    Return Value: True/False
    '''
    #approach: check for divisibility upto sqrt(num)
    from math import sqrt
    for i in range(2, int(sqrt(num)+1)):
        if num%i == 0:
            return False
    return True
def goldbach(num):
    '''
    Objective: To return a list of pairs of primes that add to num.
    Input Parameter: num - integer
    Return Value: pairList - List of tuple(s)
    '''
    if number%2 !=0 or number <= 2:
        return 'Only even number>2 allowed'
    else:
        pairList = []
        for i in range(3, number+1):
            if isPrime(i):
                for j in range(i,number):
                    if isPrime(j):
                        if number == (i+j):
                            pairList.append((i,j))
    return pairList
# TO verify the Goldbach conjecture on a given even number.
number = int(input('Enter Even Number greater than 2:'))
print(goldbach(number))
```

Let's Summarise

- An assignment statement associates a value with a name.
- The reserved words of Python programming language are called keywords.
- The name of an object is also called an identifier, for example, `myMarks, salary, num_2022A`.
- Arithmetic operators are used to build mathematical expressions involving numeric values. Python provides following arithmetic operators:
  - addition (+),
  - subtraction (-),
  - multiplication (*),
  - division (/),
  - floor division (//),
  - modulus (%),
  - exponentiation (**)
- A relational operator compares two values and returns a Boolean value **`True`** or **`False`**. The relational operators supported by Python are as follows:
  - less than (<)
  - greater than (>)
  - less than or equal to (<=)
  - greater than or equal to (>=)
  - equals (==)
  - and not equal (!=)
- The logical operators **`not, and,`** and **`or`** are applied to logical operands (also called Boolean operands or Boolean values) **`True`** and **`False,`** and yield a Boolean value.
- The order in which operators are executed if an expression contains multiple operators is called operators precedence. An operator with higher precedence will be evaluated before the operator with lower precedence.
- The process of dividing a computer program into separate blocks of code or sub-programs (each having a name and functionality) is called modular programming.
- A function is a sub-program that manipulates data received as an input to solve a problem or sub-problem of a complex problem and often returns one or more results.
- Python provides a large number of pre-defined built-in functions such as **`int()`**, **`input()`**, **and `print()`**.
- The **`input()`** function reads the text entered by a user until a newline is encountered.
- The **`help()`** function is used to display the documentation of the `functions, classes, modules,` etc.
- Python also provides several built-in modules, each of which contains several useful functions. Syntax to import a module is as follows:

  ```
  import <modulename>
  ```
- Python also allows you to create your own functions according to your requirement which are called user-defined functions.
- A function is executed only when it is invoked. Defining a function ONLY makes a provision to do something. Defining a function in a program does not achieve any result unless invoked.
  - **Function Call:** To invoke a function, using its name, followed by the arguments (if any) in parentheses.
  - **Input Arguments:** Input values passed during the call to a function.
  - **Formal Parameters:** Variables that receive the values during a call to the function.
  - **Positional arguments:** Input arguments which are provided in the function call in the same order in which the actual parameters are specified in the function definition.

- **return** is a keyword used to return the result of computation from a function to the statement that invoked the function.
- **Default parameters:** Default parameters take the default values provided in the function definition when these are not mentioned in a function call.

➢ The statements used to control the execution flow are also called control structures.

➢ The statements used to control the flow of execution of statements in a program/ function are called control statements.
- if statement allows us to execute a statement only if a specific condition is satisfied.
- An **`if-else`** statement evaluates the condition in the **`if`** clause and will execute the body of this clause only if the condition evaluates to **`True`**, but if the condition evaluates to **`False`**, then the body of the **`else`** clause is executed.
- An **`elif`** clause checks multiple expressions for **`True`** and executes a block of code corresponding to an expression that evaluates **`True`**.

➢ The repeated execution of a statement or a sequence of statements is called a loop.
- A **`for`** statement is used for repeated execution of the loop's body for each occurrence of an element in a sequence. Syntax of **`for`** statement is as follows:

```
for item in sequence:
      <block S of statements>
```

- The **`while`** statement is used to repeatedly execute a block of statements, called body of the **`while`** statement so long as the condition specified in the **`while`** statement is satisfied. The **`while`** statement has the following syntax:

```
while <test condition>:
      <Sequence of statements S>
```

➢ A **`break`** statement transfers the control to the statement following the loop in which the break statement occurs.

➢ A **`continue`** statement skips the current iteration of the loop and the control moves to the next iteration of the loop.

➢ A string in Python is a sequence of characters enclosed between single or double quotes, for example, "How are you?"
- **`len()`** returns the length of a string
- **`+`** operator is used to concatenate the strings
- **`*`** operator produces a string, concatenated with itself a specified number of times
- **`str()`** function is used to convert a numerical value into string

➢ A subsequence of characters of a string is called a slice.

➢ String Methods: Python provides several functions, also called methods for manipulating the strings.
- **`lower()`**: returns the lowercase version of the string
- **`upper()`**: returns the uppercase version of the string
- **`strip()`**: returns a string with whitespace removed from the beginning and end of the string
- **`isalpha()`**: tests whether the string comprises only letters
- **`isdigit()`**: tests whether the string comprises only digits
- **`isspace()`**: tests whether the string comprises only space
- **`startswith('other')`**: tests whether the string starts with the `other` string, passed as a parameter
- **`endswith('other')`**: tests whether the string ends with the `other` string, passed as a parameter
- **`find('other')`**: finds the index where the first occurrence of the `other` string passed as a parameter begins in the string

- **replace('old', 'new')**: returns a string where all occurrences of `old` have been replaced by new
- **split(delimit)**: returns a list of substrings, separated by the given delimiter
- **join(list)**: joins the strings in the given list together using the string

- A list comprises a sequence of objects enclosed in square brackets [].
- A pair of square brackets **[ ]** denotes an empty list.
- The **len()** function returns the number of objects in a list.
- Multiplication * operator produces a list, concatenated with itself a specified number of times.
- List Methods: Likewise, Python strings, methods are defined for list objects also. These methods are as follows:
  - **append(elem)**: inserts the object `elem`, passed as an argument, at the end of the list
  - **insert(index, elem)**: inserts the object `elem`, passed as an argument, at the specified index and shifts the existing objects to the right
  - **extend(lst)**: inserts the elements in the list `lst`, passed as an argument, at the end of the elements of the list `lst`
  - **index(elem)**: returns the index of the first occurrence of the element elem in the list
  - **reverse()**: reverses the order of the elements in the list
  - **sort()**: arranges the elements of the list in the ascending order
  - **remove(element)**: searches for the first instance of the `element` in the list `lst` and removes it
  - **pop(index)**: removes the element from the specified `index` and returns the element removed from the list
- The operator **in** checks for the membership of a data object in the list and returns **True** or **False**, depending on whether the specified object is included in the list.
- A tuple is a sequence of elements, separated by commas and usually enclosed between parentheses.
- An empty pair of parentheses defines an empty tuple.
- A function **tuple()** is used to convert a sequence of elements to a tuple.
- Tuple Methods: Python provides several methods for manipulating the tuples.
  - **count(element)**: returns the number of times the given element appears in a tuple
  - **index(element)**: returns the index of the first occurrence of an element in a tuple
  - **sorted(t)**: sorts the tuple elements in ascending order by default
- A dictionary is an unordered set of `key:value` pairs. A dictionary may be specified by enclosing the `key:value` airs in braces { }.
- An empty pair of braces denotes an empty dictionary.
- Dictionary Methods: Python provides several methods for manipulating the dictionaries. These methods are as follows:
  - **keys()**: returns a `dict_keys` object comprising the set of keys included in the dictionary
  - **values()**: returns a `dict_values` object comprising the set of values included in the dictionary
  - **items()**: returns a `dict_items` object comprising the set of items included in the dictionary
  - **update()**: used to add some key-value pairs in an existing dictionary
  - **pop(myKey)**: removes the item with the specified `key`
  - **popitem()**: removes the last inserted item from the dictionary
  - **clear()**: method empties the dictionary
  - **sorted()**: used to sort by the `keys` or by the `values`
  - **get(myKey)**: returns the value of the item with the specified key
  - **copy()**: returns the copy of the dictionary
  - **fromkeys(keys, value)**: returns a dictionary with the specified keys and the specified value
  - **setdefault(key, [value])**: returns the value of the item with the specified `key`

Solved Exercises

A. Multiple Choice Questions

1. Which of the following is not a keyword in Python?
a. with b. as c. pass d. print

2. Which of the following is part of the character set of Python?
a. Alphabets b. Digits c. Special characters d. All of these

3. Which of the following is used to assign a value to a variable?
a. # b. ^ c. = d. !

4. An escape sequence starts with _____ symbol.
a. / b. \ c. # d. =

5. Which symbol is used to represent a comment line?
a. % b. \ c. @ d. #

6. The function `input()` accepts the data entered by the user as a _____.
a. String b. Number c. Either a string or number d. Float

7. Which of the following is a mutable data type?
a. string b. integer c. list d. tuple

8. Which of the following can be the value of a Boolean data type?
a. TRUE b. true c. True d. Truth

9. In a dictionary, which operator separates the key from its value?
a. : b. ; c. , d. '

10. Which of the following operators has the highest precedence?
a. * b. ** c. \ d. *

11. Which of the following is *not* contained in a header of a compound statement?
a. keyword b. test condition c. colon d. a pair of curly brackets

12. Which of the following is not a built-in function in Python?
a. **`print()`** b. **`input()`** c. **`int()`** d. **`sqrt()`**

13. Which of the following function headers will not generate a syntax error?
a. `def test(p = 0, q)` b. `def test(p = 0, q, r = 1):`
c. `def test(p, q, r = 1):` d. `def test(p = 0, q = 1, r):`

14. What will be the output on the execution of the following expression?
```
>>> eval('3 * 4')
```
a. 3 * 4 b. 12 c. 7 d. Error

15. Which of the following functions is used to convert an integer to a string?
a. **`str()`** b. **`string()`** c. **`int()`** d. **`integer()`**

16. What will be the output on the execution of the following instruction?
```
>>> divmod(28, 4)
```
a. 7 , 0 b. (7, 0) c. 7 d. 0

17. What will be the output on the execution of the following instruction?
```
>>> pow(2, 3, 3)
```
a. 512 b. 2 c. 729 d. 5

18. What will be the output produced on execution of the following instruction?
```
>>> round(-14.2990435981, 2)
```
a. –14.3 b. –14.299 c. –14.30 d. –14.4

19. Which of the following is the correct syntax for using the function `sqrt()` from `math` module to find the square root of 25?

a.
```
math import
sqrt(25)
```

b.
```
import math
sqrt(25)
```

c.
```
import math
math.sqrt(25)
```

d.
```
math import
math.sqrt(25)
```

20. Which of the following is returned to the function caller if there is no return statement in the function definition?

a. `0` b. `-1` c. `None` d. `Error`

B. State whether the following statements are True or False:

1. An escape sequence must be enclosed in quotes. _____

2. A function is a sub-program. _____

3. A function defined in a program can be called only once in the program. _____

4. The rules for naming a function are the same as those for naming an identifier. _____

5. The statements in the body of a function are executed when Python encounters the definition of the function. _____

6. Formal parameters in the definition of a function and actual parameters in a call to a function can have the same name. _____

7. Default parameters cannot be skipped in a function call. _____

8. Keyword arguments are the same as positional arguments. _____

9. By default, the statements in a program are executed sequentially. _____

10. `if-elif` statement is a compound statement. _____

C. Fill in the blanks.

1. A _____ statement is ignored by Python interpreter.

2. In the absence of any return statement, the function returns _____ to the calling function.

3. A function can be called _____ times(s) in a program.

4. In Python, the _____ of a loop is indented with respect the header.

5. The _____ block of the loop is executed when all the iterations of the loop have been executed.

6. If the test condition in a `while` loop never becomes False, it is called _____ loop.

7. The _____ function is used to add an element at the end of the existing list.

8. While a list is _____ (mutable/ immutable), a tuple is _____ (mutable/ immutable).

9. The _____ function returns a list comprising elements of a tuple in sorted order.

10. The explicit conversion of an operand to a specific data type is called _____.

D. Answer the following questions:

1. Which of the following cannot be used as identifiers in Python? Justify your answer.

`while, 1count, Your Class, hindi_movie, school-name`

Ans.

`while`	a keyword
`1count`	begins with a digit
`Your Class`	contains space
`school-name`	contains a hyphen(-)

2. What is the difference between a keyword and an identifier?

Ans. A keyword is a reserved word with a specific purpose in Python. An identifier is a user-defined name by which it is referenced in a program.

3. What is `None` in Python?

Ans. `None` is an object of type `NoneType`, which can be used for comparison; for example, `None == None` yields `True`, and `None == False` yields `False`.

4. What is `'None'` in Python?

Ans. It is an object of the `str`.

5. Name any two immutable data types in Python.

Ans. Tuple, String

6. Why is the Boolean data type considered a sub-type of integers?

Ans. This is because boolean values `False` and `True` map to 0 and 1, respectively. For example, `sum([True, True, False, True])` yields 3.

7. In each case, write the output that will be produced by Python interpreter (if any):

(i) `>>> '10' + '15'`

Ans. '1015'

(ii) `>>> '100' + 20`

Ans. TypeError

8. Write the escape sequences for the following:

(i) Horizontal tab (ii) **`New line`**

Ans. `\t, \n`

9. What will be the output produced on execution of each of the following:

(i) `>>> print('Python is \neasy to learn')`

(ii) `>>> print('You have \t an exam tomorrow')`

Ans. (i)
```
Python is
easy to learn
```

(ii)
```
You have    an exam tomorrow
```

10. What will be the output produced on execution of the following code snippet:

```
print('BIG', 'BIGGER', 'BIGGEST', sep = '*')
print('Jim', 'Jam', 'Jin', end='##')
print('First', 'Sem', 'End', sep='-', end = '#Festival#')
```

Ans.
```
BIG*BIGGER*BIGGEST
Jim Jam Jin##First-Sem-End#Festival#
```

11. What is the difference between mutable and immutable objects?

Ans. An object that can be modified after it is created is called mutable, for example, objects of type list, and set.

An object that cannot be modified after it is created is called immutable, for example, objects of type `int, float, tuple, str`.

12. For each of the following Python expressions, give the value that will be produced when it is evaluated. In case your answer is False, can you modify the expression by adding only parentheses so that it evaluates to True.

(i) `10 == 4 == 0`

(ii) `29 < 13 == 90 < 29`

(iii) `10 == 6 + 4`

(iv) `(10 == 6) + 4`

(v) `(10 != 6) + 4`

(vi) `False or 3 == 2 + 1`

Ans. (i) False, (10 == 4) == 0

(ii) False, (29 < 13) == (90 < 29)

(iii) True

(iv) 4

(v) 5

(vi) True

13. What is the index of the first element of a tuple?

Ans. 0

14. What is the difference between a list and a tuple

Ans. (i) While a list is mutable, a tuple is immutable

(ii) Elements of a list are separated by commas and enclosed in square brackets [] while elements of a tuple are separated by commas and enclosed in parentheses( ).

15. Enumerate different types of conditional statements in Python.

Ans. **`if, if-else, if-elif-else`**

16. Define modular programming. Give any one advantage of modular programming.

Ans. The process of dividing a computer program into separate blocks of code or sub-programs (each having a name and functionality) is called modular programming. Each independent unit of code is called a module.

Advantage: As the complex problem is divided into simpler sub-problems, It saves a lot of time and effort.

17. Is it mandatory for each function to have a return statement?

Ans. No

18. What will be the output on evaluating the following expressions?

(i) `19 * 3 / 4 + 1`

(ii) `14 * 22 / 7 * 2`

(iii) `5 ** 2 - 50 // 3`

(iv) `20 % 3 > 6 and 45 // 9`

(v) `(50 - 6 // 5) > 9 or 3 ** 2 > 11`

(vi) `not True or False and True`

(vii) `'ROSE' * 3`

(viii) `True and 45 // 9`

(ix) `True and False or True and not False and not True`

Ans. (i) `15.25`

(ii) `88.0`

(iii) `9`

(iv) `False`

(v) `True`

(vi) `False`

(vii) `'ROSEROSEROSE'`

(viii) `5`

(ix) `False`

19. Consider the following string declaration

```
>>> tag = 'UnCover and COVer'
```

Using slicing, write an expression that would yield the string `'COVer'`.

Ans. Any of the following expressions will do the needful:

```
tag[-5:], tag[12:17], tag[12:]
```

20. Consider the following assignment statement :

```
lst = list('TerM 1 EXam')
```

Match the following statement with the outputs that will be produced when they are executed.

(i)	`print(lst[1:12:3])`	a.	`[]`
(ii)	`print(lst[:-8])`	b.	`['e', ' ', 'E', 'm']`
(iii)	`print(lst[-6:0])`	c.	`['X', 'E', ' ', '1', ' ']`
(iv)	`print(lst[-3:-8:-1])`	d.	`['T', 'e', 'r']`

Ans. 1. b, 2. d, 3. a, 4. c

21. Consider the code given below:

```
import math
def practice(num):
    sq = math.sqrt(num)
    print('Square Root is ', sq)
p=49
practice(p)
```

Identify the following:

(i) Name of user defined function

(ii) Function header

(iii) Name of built-in function used to find square root

(iv) Argument(s)

(v) Formal parameter(s)

(vi) Function Body

(vii) Function Call Statement

Ans. (i) Name of user defined function : `practice`

(ii) Function header : `def practice(num):`

(iii) Name of built-in function used to find square root : `sqrt()`

(iv) Argument(s): `p`

(v) Formal parameter(s) : `num`

(vi) Function Body:

```
sq=math.sqrt(num)
print('Square Root is ',sq)
```

(vii) Function Call Statement: practice(p)

22. Differentiate between an argument and a formal parameter.

Ans. When a function is called, an argument replaces the corresponding formal parameter in the definition of a function.

23. Will the following code execute successfully? If yes, what will be the output produced on its execution. If not, identify the errors in the given code:

```
side = input('Enter side of a square: ')
perimeter = 4*side
print('The perimeter of square is ', perimeter)
```

Ans. The above code will execute successfully and will produce the following output on its execution.

```
Enter side of a square: 12
The perimeter of square is 12121212
```

24. The following code is intended to compute the perimeter of a square. Will it achieve the desired goal? If yes, what will be the output produced on its execution? If not, identify the errors in the given code:

```
side = input('Enter side of a square')
perimeter = 4*Side
PRINT('The perimeter is ', perimeter)
```

Ans.

```
side = int(input('Enter side of a square')) # Need to convert str to int
perimeter = 4*side # Identifiers are case sensitive
print('The perimeter is ', perimeter) # no function by the name PRINT. Identifiers
are case sensitive
```

25. Rewrite the following code after removing each error. Underline all the corrections made.

```
if x = 10:
    print(x)
elif
    print(x*x)
```

Ans.
```
if x == 10:
    print(x)
else:
    print(x*x)
```

26. What will be the output produced on execution of the following code?
```
def test(n1, n2):
    if n1 % 2 == 0 and n2 % 5 == 0:
        n1, n2 = n2, n1
    else:
        n1, n2 = n2 + 1, n1 + 1
    print('Inside function test n1, n2: ', n1, n2)
n1 = 24
n2 = 21
print('Before invoking test(), n1, n2: ', n1, n2)
test(n1, n2)
print('After invoking test(), n1, n2: ', n1, n2)
```

Ans.
```
Before invoking test(), n1, n2: 24 21
Inside function test n1, n2:  22 25
After invoking test(), n1, n2: 24 21
```

27. What will be the output produced on execution of the following code?
```
def search(myList, y = 5):
    count = 1
    for x in myList:
        if abs(x) == y:
            count += 1
    return count
lst = [4, 7, -3, 5, -5, 6, 10]
searchKey = 20
print('List to be searched:', lst)
cnt=search(lst, searchKey)
print('The number of occurrences of', searchKey, 'is: ', cnt)
cnt = search(lst)
print('The number of occurrences of default key 5 is: ', cnt)
```

Ans.
```
List to be searched: [4, 7, -3, 5, -5, 6, 10]
The number of occurrences of 20 is:  1
The number of occurrences of default key 5 is: 3
```

28. What will be the output produced on execution of the following code?
```
def myFunc(msg):
    print('Inside function, length of the string: ', len(msg))
print('String: Good Morning ')
length = myFunc('Good Morning')
print('After executing the function , length of the string:', length)
```

Ans.
```
String: Good Morning
Inside function, length of the string: 12
After executing the function , length of the string: None
```

29. Will the following code produce an error? If yes, point out the error, if not, what will be the output produced on execution of the code?
```
def process(num):
    a = a * num
    return
```

```
a = 100
b = 5
process(a)
print(b)
```

Ans. Yes, it will produce an error. This is because the variable a is not defined inside the function process. So, when a is referenced in an assignment statement (for multiplication operation), it does not have any value associated with it.

30. What will be the output produced on execution of the following code?

```
value = 10
number = 5
def modifyData():
    global number
    insideValue = value + 5
    number = pow(insideValue, 2)
    return insideValue
final = modifyData()
print(value, number, final)
```

Ans. `10 225 15`

31. What will be the output produced on execution of the following code?

```
def stringData(message):
    length = len(message)
    newStr = ''
    for i in range(0, length):
        if message[i].islower():
            newStr = newStr+message[i].upper()
        elif message[i].isdigit():
            newStr = eval(message[i])
            newStr = str(newStr)
        else:
            newStr = newStr + '#'
    print('New String : ', newStr)
stringData('Gr843?*$')
```

Ans. `New String :  3###`

32. What will be the output produced on execution of the following code?

```
def stringData(message):
    length = len(message)
    newStr = ''
    for i in range(0, length):
        if message[i].islower():
            newStr = newStr+message[i].upper()
        elif message[i].isdigit():
            str1 = eval(message[i])
            newStr = newStr + str(str1)
        else:
            newStr = newStr + '#'
    print('New String : ', newStr)
stringData('Hello123?*$')
stringData('Hello123Hello?*$')
```

Ans.
```
New String :  #ELLO123###
New String :  #ELLO123#ELLO###
```

33. Give all possible outputs that may be produced when the following code is executed.

```
import random
continents = ['Asia', 'Australia', 'South America', 'North America', 'Europe', 'Antarctica']
option1 = random.randint(2, 4)
option2 = random.randint(3, 5)
for x in range(option1, option2):
    print(continents[x], end = '#')
```

Ans.
```
Europe#
```
OR
```
North America#Europe#
```
OR
```
North America#
```
OR
```
South America#
```
OR
```
South America#North America#Europe#
```
OR

South America#North America#

34. What will be the output produced on execution of the following code?

```
limit = 0
for i in range(2, 9, 3):
    limit += pow(i,2)
    print(limit, end=' ')
```

Ans.
```
4 29 93
```

35. What will be the output produced on execution of the following code?

```
myString = 'God helps those who help themselves'
print(myString.split())
print(myString.split('my'))
print(myString.partition('my'))
```

Ans.
```
['God', 'helps', 'those', 'who', 'help', 'themselves']
['God helps those who help themselves']
('God helps those who help themselves', '', '')
```

36. Will the following code execute successfully? If yes, what will be the output produced on execution of the code?

```
myList = [1, 0, -1]
newList = myList * 3
newList.insert(-1, 5)
value = newList.index(4)
print(value)
```

Ans. No, the code will not execute as 4 is not present in the list `newList`, so `index()` will result in an error.

37. What will be the output produced on execution of the following code?

```
data =['Mother', 500, 'Father', 400, 'Aunt', 300, 'Brother', 200]
alpha = ' '
beta = 0
final = 0
for c in range(1, 7, 2):
    beta = beta + c
    alpha = alpha + data[c-1] + '!!'
    final = final + data[c]
print(beta, alpha, final, sep='#')
```

Ans.
```
9# Mother!!Father!!Aunt!!#1200
```

38. What will be the output produced on execution of the following code?

```
def test(num1, num2=-2):
    num1 = num1 // num2
    num2 += num1
    print(num1, num2, sep='@')
    return num1, num2
a1 = 25
a2 = test(a1)
print(a1, a2, sep = '@')
```

Ans.
```
-13@-15
25@(-13, -15)
```

39. What will be the output produced on execution of the following code?

```
import string
startString = '#NaTO5*'
finalString = ' '
pos = 0
value = 1
while pos < len(startString):
    if startString[pos] >= '0' and startString[pos] <= '9':
        value = int(startString[pos])
        value = value - 1
        finalString = finalString+startString[value]
    elif startString[pos].isupper():
        if startString[pos] not in 'AEIOU':
            finalString = finalString + '$'
        else:
            finalString = finalString + startString[pos + 1]
    else:
        finalString = finalString + '*'
    pos = pos + 1
print(finalString)
```

Ans.
```
*$*$5O*
```

40. Write a program that takes from a user temperature in Celsius and displays it in Fahrenheit.

Ans.
```
#Objective: Convert temperature in Celsius to fahrenheit.
celsius = int(input('Enter the temperature in degree Celsius : '))
fahrenheit = celsius * 9/5 + 32
print('The temperature in degree Fahrenheit is : ', fahrenheit)
```

41. Write a function that displays n rows of the following pattern for a given n and make use of it in a program (assume 0<=n<10)

1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

Ans.
```
def triangle(n):
    '''
    Objective: To print a triangular pattern having n rows
    Input Parameter: n - number of rows
    Return value: None
    Constraint: 0=<n<10
```

```
    '''
    assert 0<=n and n<10
    for i in range(1, n+1):
        for j in range(1, i+1):
            print(i, end = ' ')
        print()
nRows = int(input('Enter number of rows of output: '))
triangle(nRows)

#Alternative solution
def triangle(n):
    '''
    Objective: To print a triangular pattern having n rows
    Input Parameter: n - number of rows
    Return value: None
    Constraint: 0 = < n <10
    '''
    assert 0 <= n and n < 10
    for i in range(1, n+1):
        print((str(i)+ ' ')*i)
nRows = int(input('Enter number of rows of output: '))
triangle(nRows)
```

42. Write a program that prints the sum of following series:
1! + 2! + 3! + ...n!

Ans.
```
sum = 0
print('To find the sum of series 1! + 2! + 3! + ... n!')
limit = int(input('Enter no of terms in the series '))
for i in range(1, limit+1):
    fact = 1
    for j in range(1, i+1):
        fact = fact * j
    sum = sum + fact
print('Sum of Series is : ', sum)
```



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion(A): Consider the statement:

`Encrypt="Code1"+"Code2"`

After the above statement is executed, the value of **`Encrypt`** will be **`Code1Code2`**

Reasoning (R): The **`operator +'`** adds the numeric operands and appends the string operands.

2. Assertion(A): A variable preceded with the keyword, `global` inside a function is visible with changes made to it outside the function.

Reasoning (R): A variable declared outside all functions can be displayed by using **`print()`** inside the function.

3. Assertion (A): One can change the member of a list
   Reasoning (R): List is an immutable data type

4. Assertion(A): An argument is the value passed to the function at the time of the function call.
   Reasoning(R): A return statement is always present in the function as the control has to return to the calling function.

5. Assertion(A): If the modified value of a global variable is to be used outside the function, then the variable name should be preceded by the keyword 'global' inside the function.
   Reasoning(R): If both local and global variables are of the same name, the global variable is hidden inside the function.

Ans. 1. a 2. b 3. c 4. c 5. b



Case-based Questions

1. Shagun possesses exceptional numerical ability. Now, after learning Python, she wants to create a program in Python that displays the sum of the following series:
   ½–⅔+¾–⅘+.....n–1/n

   Help her complete the task.

Ans.
```
def sumOfSeries(num):
    """
    Objective: to find sum of series 1/2-2/3+3/4-4/5...n-1/n
    Input value: num - integer
    return value: None
    """
    sum=0
    for i in range(1,num+1):
        if (i & 1):
            sum=sum+i/(i+1)
        else:
            sum=sum-i/(i+1)
    print(sum)
n=5
sumOfSeries(n)
```

2. Girija is a student of class 9. She has the ability to draw beautiful patterns on paper. Her friend, Namita is good at programming. She saw Girija drawing the pattern given below:

```
######
#    #
#    #
#    #
#    #
######
```

   Namita decided to create a program in Python that will display the same pattern. Help her complete the task.

Ans.
```
size=6
for x in range(size):
    for y in range(size):
        if x==0 or x==size-1 or y==0 or y==size-1:
            print('#', end='')
        else:
            print(' ',end='')
    print()
```

Assessment

A. Multiple choice questions

1. Consider the following function definition.

```
def testFunc(a, b, c = 6):
    return str(a) + ' ' + str(b) + ' ' + str(c)
```

Which of the following is an incorrect call to the function `testFunc` defined above?

a. `testFunc('Correct')`
b. `testFunc(10, 'correct', 'value')`
c. `testFunc('correct', 'value')`
d. `testFunc(10, 50)`

2. Which of the following function calls will cause an error while invoking the function defined below?

```
def myFunc(a,b,c,d):
    return a + b + c + d
```

a. `myFunc(a = 4, 3, 2, 1)`
b. `myFunc(d = 4, c = 3, b = 2, a = 1)`
c. `myFunc(4, 3, 2, 1)`
d. `myFunc(4, b = 3, c = 2, d = 1)`

3. Consider the following code:

```
def testFunc(a, b, c = 6):
    print(a, b, c)
testFunc(c = 20,a = 5, b = 10)
```

Which of the following will be produced as output on execution of the above code?

a. 5 10 20
b. 20 5 10
c. 5 20 10
d. Error

4. Consider the code given below:

```
def testFunc(a1, a2, a3 = 6):
    return a1 + a2 + a3
testFunc(a3 = 20, a1 = 5, a2 = 10)      # Statement 1
testFunc(20, 5)                         # Statement 2
```

Which of the following statements is true?

a. Statement 1 will not execute as the arguments are not in the same order as parameters.
b. Statement 2 will not execute as third argument is missing
c. Statement 1 will execute as it is called with keyword arguments
d. Statement 2 will execute as it is called with keyword arguments

5. Which of the following is a valid keyword?

a. `IF`
b. `if`
c. `Elif`
d. `ELSE`

6. The expression **`range(10,2,-4)`** will include _____.

a. 10,6,2
b. 9,5,2
c. 10,6
d. 9,5

7. How many numbers will be printed by the following loop?

```
for z in range(3):
    print(z)
```

a. 1
b. 2
c. 3
d. 4

8. What is the output of the following code snippet?

```
for num in range(5):
    pass
print(num)
```

a. 3
b. 4
c. 5
d. 0

9. What will be the value of n on execution of the following loop?

```
n = 5
while(n <= 10):
    n = n+1
```

a. 12 b. 11 c. 10 d. 5

10. Which of the following is an example of a syntactically correct nested loop?

a.
```
for x in range(1,5):
    for y in range(x):
        print(x)
```

b.
```
for x in range(1,5):
    for y in range(x):
    print(x)
```

c.
```
for x in range(1,5):
    print(x)
  for y in range(x):
    print(y)
```

d.
```
for x in range(1,5):
  for y in range(x):
   print(x)
```

11. Which of the following yields the string having the first letter of every word in a string in uppercase and rest in lowercase?

a. `'Hello, How Are You'.Title()`

b. `'Hello, How Are You'.title()`

c. `'Hello, How Are You'.Sentence()`

d. `'Hello, How Are You'.sentence()`

12. Consider the following list:

```
lst = [1, 5, 0, 15, 30, 50]
```

What will be the output produced on execution of the following statement?

```
print(lst[1:2] + lst[3:6], lst)
```

a. [5, 15, 30, 50] [1, 5, 0, 15, 30, 50] b. [5, 15, 30, 50] [5, 15, 30, 50]

c. [1, 5, 0, 15, 30][1, 5, 0, 15, 30] d. [1, 5, 0, 15, 30] [1, 5, 0, 15, 30, 50]

13. Which of the following returns the element just removed from the list?

a. `del` b. `pop()` c. `remove()` d. `find()`

14. Which of the following statements is not true about `list()` function?

a. Given string as argument, it forms a new list comprising individual elements of the string.

b. It creates an empty list.

c. It converts a complex number into a list.

d. It converts a tuple into a list.

15. Which of the following functions will concatenate the two lists?

a. `extend()` b. `merge()` c. `append()` d. `join()`

16. Consider the tuple `tpl` defined below:

```
tpl = ('abc','efg','hij')
```

Which of the following is the correct statement for adding a new element to the tuple tpl?

a. `tpl + ('lmn')` b. `tpl + 'lmn'` c. `tpl + ('lmn',)` d. None of these

17. Which of the following statements will delete a key-value pair for the key `'tue'` in the following dictionary?
```
days = {'sun':1,'mon':2,'tue':3,'wed':4,'thur':5}
```
a. `delete days['tue']` b. `del.days['tue']` c. `del days['tue']` d. `days.del['tue']`

18. Which of the following is NOT true about tuples in Python?
   a. A tuple may comprise a string as one of its elements.
   b. A tuple is always enclosed in parenthesis.
   c. The method `del` works on tuples.
   d. A tuple may comprise a dictionary as one of its elements.

B. State whether the following statements are True or False:

1. The statements in a suite (block) are at the same indentation level. _____
2. The `else` clause in a `for` statement is always executed. _____
3. An expression such as 3 + 4 may be passed as an argument to a function. _____
4. The function `count()` counts the number of elements of a list. _____
5. A tuple is a mutable data type. _____
6. When the function `sorted()` is applied to a list, it modifies the list. _____
7. When the function `sorted()` is applied to a tuple, it modifies the tuple. _____
8. A dictionary is a sequence of objects over which we can iterate in a `for` statement. _____

C. Fill in the blanks.

1. _____ data type in Python signifies the absence of value in a given statement.
2. _____ character separates a key from a value in a dictionary.
3. While writing a program, if rules of programming language are not followed, then such type of error is _____ error.
4. The data type of **`num`** in the statement **`num=(5)`** is _____.
5. Strings, lists and tuples are _____ data types in Python.
6. The function that returns the number of characters in a string is _____.
7. The _____ operator joins two lists.
8. A string can be displayed multiple times by using the _____ operator.

D. Answer the following questions:

1. Give any two characteristics of the Python programming language that make it so popular.
2. What are two modes in which Python works? In which mode can we save the code?
3. Identify invalid identifiers from the following list:
```
not#You        false      gr8        FOR        1and2
```
4. Define an operator and an operand. Also mention different types of operators.
5. What is the importance of data types in Python?
6. Differentiate between implicit and explicit type conversion. Give examples to support your answer.
7. What is meant by indexing in the context of strings in Python. Illustrate the use of negative and non-negative indexing with the help of an example.
8. With the help of an example, illustrate two methods of adding an element to a list.
9. What is the difference between 10 and (10,)?
10. Is slicing applicable to dictionaries? Justify your answer.
11. Which type of objects can be used as keys in dictionaries?
12. Why is a dictionary termed as an unordered collection of key: value pairs?
13. Consider the following function header:
```
def testCall(num1, num2 = 10, num3 = 100):
```
Which of the following function call statements will be executed correctly? Justify your answer.

(i) `testCall(20)`

(ii) `testCall(num2 = 20)`

(iii) `testCall(num2 = 20,num1 = 30)`

(iv) `testCall(50,500,5000)`

(v) `testCall()`

(vi) `testCall(val,num3 = 50)  ( Assume val=500)`

(vii) `testCall(num3 = 50,num1 = 500)`

14. What will be the output produced on the execution of the following code?

```
myList = [1, 0, -1]
newList = myList * 3
newList.insert(-1, 5)
indx = newList.index(5)
print(indx)
```

15. What will be the output produced on the execution of the following code?

```
import math
number = 57
while number >= 10:
    localNumber = number // 10
    if localNumber % 2 == 0:
        break
    else:
        print(math.factorial(localNumber))
    number = number - 5
```

16. What will be the output produced on the execution of the following code?

```
import math
x = divmod(42, 4)
y = int(math.fmod(10, 3))
z = x + (y,)
print(z)
```

17. Write a program that accepts a number and prints its multiplication table.

18. Write a program that uses a user-defined function to convert pounds to kilograms.

19. Write a function that accepts an input date as an integer whose digits signify DDMMYYYY from left to right and returns a tuple of numbers having the values (DD, MM, YYYY). Use this function in a program to display the output in the following format:

```
Day : DD
Month: MM
Year : YYYY
```

20. Write a program that accepts a number, N and displays the sum of numbers from 1 to N if the number is positive and the sum of numbers from (2*N) to N if the number is negative. The number N is to be included in the sum.

21. Write a function that returns the number of seconds in the specified year.

22. Write a function that returns a tuple comprising the first five elements of the Fibonacci series.

23. Write a program that accepts n words and prints the longest word.

24. Write a program that accepts from a user the number of students in a class, followed by the names and marks of the students in the class, stores them in a dictionary as key-value pairs, and displays the name of the student who has scored maximum marks.

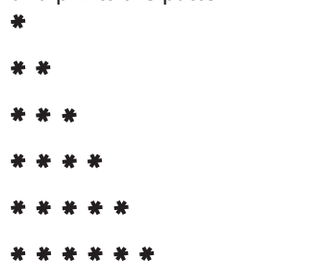
25. Write a program to do the following:

- Create a dictionary of month names and the number of days in a month.
- Displays the names of months having 31 days.
- Accepts a month name from a user and displays the corresponding number of days.
- Displays the names of months in alphabetical order.

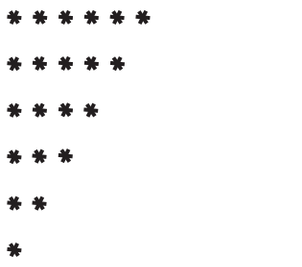
26. Write a function that takes a string as an argument and returns the number of words in it. Assume that a word is terminated by white space.

27. Write a program that displays the sum of the following series:
1/1! + 2/2! + 3/3!+.....n/n!

28. For each of the following patterns, write a function that accepts a parameter `nRows`—the number of rows in the pattern and prints the pattern.



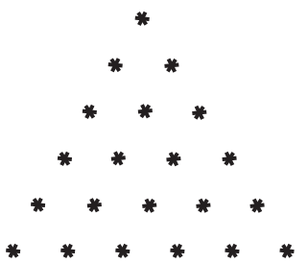
Half Pyramid



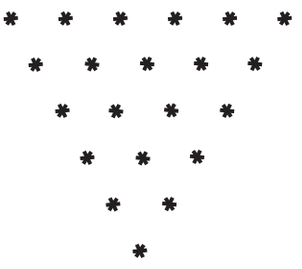
Inverted Half Pyramid



Hollow Inverted Half Pyramid



Full Pyramid



Inverted Full Pyramid



Hollow Full Pyramid

Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion(A): In Python, the **`return`** statement exits a function.
Reasoning (R): A **`return`** statement with no value is the same as return None.

2. Assertion(A): **`input()`** and **`print()`** are examples of built-in functions in Python
Reasoning (R): Built-in functions are predefined in the Python standard library.

3. Assertion(A): "Orange Education" is a valid string object in Python.
Reasoning (R): All strings in Python should be enclosed in single, double or triple quotation marks.

4. Assertion(A): break statement can be used without a loop in a program.
Reasoning (R): break statement moves the control flow out of the loop unconditionally.

5. Assertion(A): continue statement starts the next iteration of the loop
Reasoning (R): Continue is a keyword in Python

Case-based Questions

1. Ranveer knows about the Fibonacci series. He is also aware about various data types in Python. He wants to create a program that will store the first ten elements of the Fibonacci series in a tuple. Help him complete the task.
2. Saira is a student of language studies. She wants her friend, Mitu to create a program that will accept and store words and meanings in the form of a dictionary. Thereafter, she wants to print the meaning after the word is entered by the user. Help her complete the task.
3. Avi has just learned about random module in Python. He has created the following program using random module.

```
import random
Rivers=["Ganges","Yamuna","Teesta","Narmada", "Saraswati","Cavery"]
option1=random.randint(3,5)
option2=random.randint(2,4)
for x in range(option1,option2+2):
   print(Rivers[x], end="#")
```

What are the minimum and maximum values that variables option1 and option2 can have?

Previous Years' Questions

1. State True or False.

   "Identifiers are names used to identify a variable, function in a program". [2023]

Ans. True

2. Which of the following is a valid keyword in Python? [2023]

   a. false b. return c. non_local d. none

Ans. b. return

3. Given the following Tuple

```
Tup= (10, 20, 30, 50)
```

Which of the following statements will result in an error? [2023]

a. `print (Tup [0])`
b. `Tup.insert (2, 3)`
c. `print (Tup [1:2])`
d. `print (len (Tup))`

Ans. b. `Tup.insert (2, 3)`

4. Consider the given expression:

```
5<10 and 12>7 or not 7>4
```

Which of the following will be the correct output, if the given expression is evaluated? [2023]

a. True b. False c. NONE d. NULL

Ans. a. True

5. Select the correct output of the code:

```
S= "Amrit Mahotsav @ 75"
A=S.partition (" ")
print (A)
```

[2023]

a. `('Amrit Mahotsav', '@', '75')`
b. `['Amrit', 'Mahotsav', '@', '75']`
c. `('Amrit', 'Mahotsav @ 75')`
d. `('Amrit', '', Mahotsav @ 75')`

Ans. d. `('Amrit', '', Mahotsav @ 75')`

6. Fill in the blank.

_____ function is used to arrange the elements of a list in ascending order. [2023]

a. `sort()` b. `arrange()` c. `ascending()` d. `asort()`

Ans. a. `sort()`

7. Which of the following operators will return either True or False ? [2023]

a. += b. != c. = d. *=

Ans. b. !=

8. Which of the following statement(s) would give an error after executing the following code ?

```
Stud="Murugan" : 100, "Mithu" : 95}          # Statement 1
print (Stud[ 95])                            # Statement 2
Stud ["Murugan"] =99                         # Statement 3
print (stud.pop ())                          # Statement 4
print (Stud)                                 # Statement 5
```

[2023]

a. Statement 2 b. Statement 3 c. Statement 4 d. Statements 2 and 4

Ans. d. Statements 2 and 4

9. What will the following expression be evaluated to in Python?

```
print (4+3*5/3-5%2)
```

[2023]

a. `8.5` b. `8.0` c. `10.2` d. `10.0`

Ans. b. `8.0`

10. Atharva is a Python programmer working on a program to find and return the maximum value from the list. The code written below has syntactical errors. Rewrite the correct code and underline the corrections made.

```
def max_num (L):
    max=L(0)
    for a in L :
        if a > max
        max=a
    return max
```

[2023]

Ans. Colon is missing after the "if a > max" statement. The correct code is given as follows:

```
def max_num (L):
    max=L (0)
    for a in L :
        if a > max:
            max=a
    return max
```

11. Find the invalid identifier from the following: [2022]

a. name b. class c. section d. break

Ans. d. break

12. For a given declaration in Python as s= "WELCOME" [2022]

Which of the following will be the correct output of print (s [1: :2])?

a. WEL b. COME c. WLOE d. ECM

Ans. d. ECM

13. Which of the following is an incorrect Logical operator in Python? [2022]

a. not b. in c. or d. and

Ans. b. in

14. Given the Python declaration S1 = "Hello". Which of the following statements will give an error? [2022]

a. print (S1[4]) b. S2=S1 c. S1=S1 [4] d. S1[4]="Y"

Ans. d. S1[4]="Y"

15. Which of the following is the correct output for the execution of the following Python statement? [2022]

```
print (5 + 3 ** 2 /2)
```

a. 32 b. 8.0 c. 9.5 d. 32.0

Ans. c. 9.5

16. Which of the following in not a Tuple in Python? [2022]

a. (1, 2, 3) b. ("One", "Two", "Three")

c. (10 ,) d. ("One")

Ans. d. ("One")

17. Which of the following is not a valid Python string operation? [2022]

a. 'Welcome' + '10' b. 'Welcome' * 10

c. 'Welcome' * 10.0 d. "10" + 'Welcome'

Ans. c. 'Welcome' * 10.0

18. What will be the output for the following Python statements? [2022]

```
T= (10, 20, [30, 40, 50], 60, 70)
T[2] [1] = 100
print (T)
```

a. (10, 20, 100, 60, 70) b. (10, 20, [30, 100, 50], 60, 70)

c. (10, 20, [100, 40, 50], 60, 70) d. Error

Ans. b. (10, 20, [30, 100, 50], 60, 70)

19. What will be the output for the following Python statements? [2022]

```
L = [10, 20, 30, 40, 50]
L = L + 5
print (L)
```

a. [10, 20, 30, 40, 50, 5] b. [15, 25, 35, 45, 55]

c. [5, 10, 20, 30, 40, 50] d. Error

Ans. d. Error

20. Identify the output of the following Python statements: [2022]

```
D = {}
T=("ZEESHAN", "NISHANT", "GURMEET", "LISA")
for i in range (1, 5):
    D[i]=T[i-1]
print (D)
```

a. {"ZEESHAN", "NISHANT", "GURMEET", "LISA"}

b. "ZEESHAN", "NISHANT", "GURMEET", "LISA"

c. {[1, "ZEESHAN"], [2, "NISHANT"], [3, "GURMEET"], [4, "LISA"]}

d. {1: "ZEESHAN", 2: "NISHANT", 3: "GURMEET", 4: "LISA"}

Ans. d. {1: "ZEESHAN", 2: "NISHANT", 3: "GURMEET", 4: "LISA"}

21. What will be the output for the following Python statements? [2022]

```
D= {"AMIT":90, "RESHMA":96, "SUKHBIR":92, "JOHN":95}
print("JOHN" in D, 90 in D, sep= "#")
```

a. True#False b. True#True c. False#True d. False#False

Ans. a. True#False

22. Nitish has declared a tuple T in Python as following: [2022]

T = (10, 20, 30)

Now, he wants to insert an element 40 after these three elements of T so that the tuple may contain (10, 20, 30, 40).

Which of the following statements shall Nitish write to accomplish the above task?

a. T = T + 40

b. T = T + (40)

c. T = T + (40,)

d. Nitish cannot insert 40 into the tuple since Tuples are immutable

Ans. c. T = T + (40,)

23. Identify the output of the following Python statements: [2022]

```
S = "GOOD MORNING"
print (S. capitalize(),S.title(), end="!")
```

a. GOOD MORNING! Good morning

b. Good Morning! Good morning

c. Good morning! Good Morning!

d. Good morning Good Morning!

Ans. d. Good morning Good Morning!

24. Identify the output of the following Python statements: [2022]

```
L = [ ]
for i in range (4):
    L.append (2 * i + 1)
print (L[: : -1])
```

a. [7,5,3,1] b. [9,7,5,3] c. [4,3,2,1] d. [1,2,3,4]

Ans. a. [7,5,3,1]

25. Identify the output of the following Python statements: [2022]

```
L1, L2 = [10, 15, 20, 25], [ ]
for i in range (len (L1)):
    L2.insert( i, L1.pop())
print (L1, L2, sep= "&")
```

a. [ ] & [25, 20, 15, 10] b. [10, 15, 20, 25] & [25, 20, 15, 10]

c. [10, 15, 20, 25] & [10, 15, 20, 25] d. [25, 20, 15, 10] & [ ]

Ans. a. [ ] & [25, 20, 15, 10]

26. Identify the correct possible output for the following Python code: [2022]

```
import random
for N in range (2, 5, 2):
    print (random.randrange (1, N), end= "#")
```

a. 1#3#5# b. 2#3# c. 1#4# d. 1#3#

Ans. d. 1#3#

27. Which of the following option can be the output for the following Python code? [2022]

```
L1 = [10, 20, 30, 20, 10]
L2=[ ]
for i in L1:
    if i not in L2:
        L2.append (i)
```

```
print (L1, L2, sep= "&")
```

a. [10, 20, 30, 20,10] & [10, 20, 30, 20,10]
b. [10, 20, 30, 20,10] & [10, 20, 30, 20, 10]&
c. [10, 20, 30, 20, 10) & (30, 20, 10]
d. [10, 20, 30, 20, 10] & [10, 20, 30]

Ans. d. [10, 20, 30, 20, 10] & [10, 20, 30]

28. Identify the output of the following Python code: [2022]

```
D = {1: "One", 2: "Two", 3: "Three"}
L = [ ]
for K, V in D. items () :
    if V[0]=="T":
            L.append(K)
print (L)
```

a. [1, 2, 3]
b. ["One", "Two", "Three"]
c. [2, 3]
d. ["Two", "Three"]

Ans. c. [2, 3]

29. What will be the output of the following Python code? [2022]

```
L = [10, 20]
L1 = [30, 40]
L2 = [50, 60]
L.append(L1)
L.extend(L2)
print (L)
```

a. [60, 50, 40, 30, 20, 10]
b. [10, 20, 30, 40, 50, 60]
c. [10, 20, 30, 40, [50, 60]]
d. [10, 20, [30, 40], 50, 60]

Ans. d. [10, 20, [30, 40], 50, 60]

30. Which of the following is not correct in context of Positional and Default parameters in Python functions? [2022]

a. Default parameters must occur to the right of Positional parameters.
b. Positional parameters must occur to the right of Default parameters.
c. Positional parameters must occur to the left of Default parameters.
d. All parameters to the right of a Default parameter must also have Default values.

Ans. b. Positional parameters must occur to the right of Default parameters.

31. For a function header as follows: [2022]

```
def Calc (X, Y=20) :
```

Which of the following function calls will give an Error?

a. Calc(15,25)
b. Calc(X=15, Y=25)
c. Calc(Y=25)
d. Calc(X=25)

Ans. c. Calc(Y=25)

32. Which of the following is not correct in context of scope of variables? [2022]

a. global keyword is used to change value of a global variable in a local scope.
b. local keyword is used to change value of a local variable in a global scope.
c. global variables can be accessed without using the global keyword in a local scope.
d. local variables cannot be used outside its scope.

Ans. b. local keyword is used to change value of a local variable in a global scope.

33. Which of the following is not a function/method of the random module in Python? [2022]

a. rand**float()** b. rand**int()** c. random() d. randrange ()

Ans. a. rand**float()**

34. What will be the output of the following Python code? [2022]

```
S="WELCOME"
def Change (T):
 T="HELLO"
 print (T, end= '@')
Change (S)
print (S)
```

a. WELCOME@HELLO
b. HELLO@HELLO
c. HELLO@WELCOME
d. WELCOME@WELCOME

Ans. c. HELLO@WELCOME

35. What will be the output of the following Python code? [2022]

```
def FunStr (S):
    T = " "
    for i in S:
        if i.isdigit():
            T = T + i
    return T
X = "PYTHON 3.9"
Y = FunStr (X)
print (X, Y, sep="*")
```

a. PYTHON 3.9
b. PYTHON 3.9*3.9
c. PYTHON 3.9* 39
d. Error

Ans. c. PYTHON 3.9* 39

36. What will be the output of the following Python code? [2022]

```
V = 50
def Change(N):
   global V
   V, N = N, V
   print(V, N, sep="#", end="@")
Change(20)
print(V)
```

a. 20#50@20
b. 50#20@50
c. 50#50#50
d. 20@50#20

Ans. a. 20#50@20

37. What is the output of the following Python code? [2022]

```
def ListChange():
    for i in range (len (L)):
        if L[i]%2 == 0:
            L[i]=L[i]*2
        if L [i] % 3 == 0:
            L[i]=L[i]*3
        else:
            L[i]=L[i]*5
L = [2, 6, 9, 10]
ListChange ()
for i in L:
    print (i,end= "#")
```

a. 4#12#27#20#
b. 6#18#27#50#
c. 20#36#27#100#
d. Error

Ans. c. 20#36#27#100#

38. What will be the output of the following Python code? [2022]

```
V = 25
def Fun (Ch):
  V=50
  print (V, end=Ch)
  V *= 2
  print(V, end=Ch)
print(V, end="*")
Fun("!")
print (V)
```

a. 25*50!100!25 b. 50*100!100!100 c. 25*50!100!100 d. Error

Ans. a. 25*50!100!25

39. Write the definition of a user-defined function PushNV (N) which accepts a list of strings in the parameter N and pushes all strings which have no vowels present in it, into a list named NoVowel.

Write a program in Python to input 5 words and push them one by one into a list named All.

The program should then use the function PushNV () to create a stack of words in the list Novowel so that it stores only those words which do not have any vowel present in it, from the list All Thereafter, pop each word from the list NoVowel and display the popped word. When the stack is empty.. display.. the message "EmptyStack".

For example:

If the words accepted and pushed into the list All are

['DRY', 'LIKE', 'RHYTHM', 'WORK', 'GYM']

Then the stack NoVowel should store

['DRY', 'LIKE', 'RHYTHM', 'GYM']

And the output should be displayed as

GYM RHYTHM DRY EmptyStack [2022]

Ans.

```
NoVowel=[]
def PushNV(N):
    for i in range(len(N)):
        if 'A' in N[i] or 'E' in N[i] or 'I' in N[i] or 'O' in N[i] or 'U' in N[i]:
            pass
        else:
            NoVowel.append(N[i])
All=[]
for i in range(5):
    w=input("Enter Words:")
    All.append(w)
PushNV(All)
while True:
    if len(NoVowel)==0:
        print("EmptyStack")
        break
        else:
        print(NoVowel.pop()," ",end='')
```

OR

Write the definition of a user-defined function Push3_5(N) which accepts a list of integers in a parameter N and pushes all those integers which are divisible by 3 or divisible by 5 from the list N into a list named Only3_5.

Write a program in Python to input 5 integers into a list named NUM. The program should then use the function Push3_5() to create the stack of the list of Only3_5. Thereafter pop each integer from the list only3_5 and display the popped value. When the list is empty, display the message "StackEmpty".

For example:

If the integers input into the list NUM are:

[10, 6, 14, 18, 30]

Then the stack Only3 _5 should store

[10, 6, 18, 30]

And the output should be displayed as

30 18 6 10 StackEmpty

Ans.

```
Only3_5=[]
def Push3_5(N):
    for i in N:
        if i %3==0 or i%5==0:
            Only3_5.append(i)
NUM=[]
for i in range(5):
    n=int(input("Enter the number to add to list:"))
    NUM.append(n)
Push3_5(NUM)
print("List of Only 3-5:",Only3_5)
while True:
    if Only3_5==[]:
        print("StackEmpty")
        break
    else:
        print(Only3_5.pop()," ",end='')
```

Answers

Multiple choice questions.

1. (d)	2. (d)	3. (c)	4. (b)	5. (d)	6. (a)	7. (c)	8. (c)	9. (a)
10. (b)	11. (d)	12. (d)	13. (c)	14. (b)	15. (a)	16. (b)	17. (b)	18. (c)
19. (c)	20. (c)							

True or False

1. (T)	2. (T)	3. (F)	4. (T)	5. (F)	6. (T)	7. (F)	8. (F)	9. (T)
10. (T)								

Fill in the blanks.

1. pass	2. None	3. multiple	4. body/suite	5. else
6. endless / infinite	7. append()	8. mutable/ immutable	9. sorted()	10. typecasting

2 LOCAL AND GLOBAL SCOPE

Chapter Outline

2.1 Objects in Python
2.2 Updating Value of a Local Variable
2.3 Updating Value of a Global Variable
2.4 Introducing Global Variable in a Function
2.5 Built-in Names
2.6 Imported Names

Introduction

The part of the program where a name (also called an identifier) is accessible, is known as the *scope* of that name. While writing a program, sometimes we use unnamed objects like numbers such as `23`, `47.34`, and `3+4j`, and strings like `'Hello World'`. However, most of the objects used in a program of reasonable complexity are named objects. Names of the data objects are also called variables. While some names occur in the definition of a function as parameters, other names may be introduced within the function as required, and yet another type of name may appear in the program (module) outside of all functions. Part of the program where a named object is accessible is called its scope. In this chapter, we will study the scope of names in a Python program.

Scope: part of the program where a named object is accessible

2.1 Objects in Python

In Python, objects may or may not have names. For example, the object 5 has no name in the following example:

```
>>> print(5)
5
```

However, in the following example, the object 5 has the name n:

```
>>> n = 5
>>> print(n)
5
```

A named object is referenced in a program by its name, also called a variable. Thus, we may say, the name `n` refers to the object `5`. Equivalently, we may say that the variable `n` has the value `5`, or variable `n` refers to `5`.

2.1.1 Local Names (Local Variables)

All the names (variables) defined inside a function are accessible only within the body of a function (also called the scope of a function) and are known as local names or local variables.

Local Variables: variables that are defined inside a function and are accessible only within the body of a function (also called the scope of a function)

Access to a local variable outside the function will result in an error. A local variable defined in a function is said to be active during the execution of the function, (also called the lifetime of a variable). The input parameters of a function (also called formal parameters, or simply parameters) are also accessible within its scope like local variables. Next, we will examine several examples of the local scope of names.

Scope of local variable: execution of the function.

Modifying Data Locally within a Function

Suppose a company decides to give a yearly increment of 20,000 to an employee.

- For this purpose, we develop a function **`incrementSalary()`** (Program 2.1, lines 1–8) in the hope that it will update an employee's salary.
- When the function **`incrementSalary()`** is called (line 21), it updates the parameter **`salary1`** by adding default value of **`increment`** (line 8).

Program 2.1 Incrementing salary of an employee

```
01 def incrementSalary(salary1, increment = 20000):
02     """
03     Objective: To increment the salary of the employee
04     Input Parameters:
05     salary1 - numeric value denoting salary of an employee
06     increment - numeric value of increment
07     """
08     salary1 = salary1 + increment
09
10 #main program segment
11 """
12 Objective: To increment the current salary
13 User Interface:
14     1. The user is asked to enter:
15          Current salary
16     2. Output to the user:
17          Updated salary after increment
18 """
19 salary = int(input('Enter the Employee Salary: '))
20 assert salary>=0
21 incrementSalary(salary)
22 print('Updated Salary:', salary)
```

Let us now execute Program 2.1, and enter **`70000`** as the **`salary`** of an employee. On executing the program, Python responds as follows:

```
>>> Enter the Employee's Salary: 70000
    Updated Salary: 70000
```

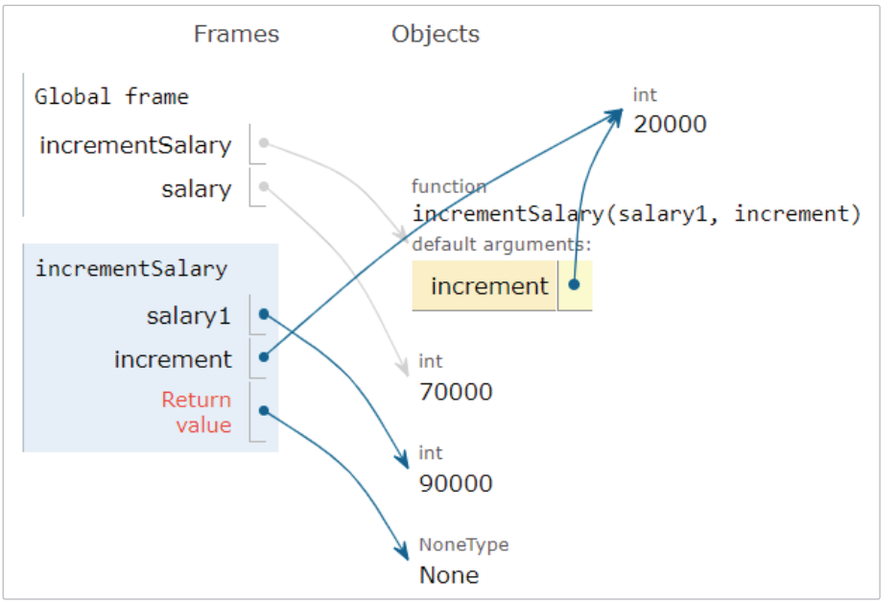


Fig 2.1: Variables on the execution of line 8 in the function `incrementSalary()`

Recall that the parameters are like local variables. Initially, a function parameter refers to the same object in memory as the argument with which the function was invoked. The program (or function) which calls a function is called the caller program (or caller function) and the called function is called the callee. The part of the program outside of all the functions in the program is called the global frame (or global environment). Note that initially, the name **`salary`** in the caller's global environment, and the name salary1 in the local environment of the function, refer to the same data object **`70000`**. However, when a parameter is assigned a new value during the execution of a function, a new object is created locally within the scope of the function. Thus, the corresponding variable in the scope of the caller remains unaffected.

- So, an update in **`salary1`** in the function is not reflected when we execute the print statement in the global frame (line 24).
- Indeed, the assignment statement in line 9 creates a different object **`salary1`** in memory for the updated value (90000) and the object **`salary`** in the global environment still refers to the original value. This is illustrated in Fig 2.1.

Program 2.2 Incrementing salary of an employee

```
01 def incrementSalary(salary1, increment = 20000):
02     """
03     Objective: To increment the salary of the employee
04     Input Parameters:
05         salary1 - numeric value denoting salary of the employee
06         increment - numeric value of increment
07     Return value: numeric value indicating updated salary
08     """
09     salary1 = salary1 + increment
10     return salary1
11 #main program segment
12 """
13 Objective: To increment the current salary
14 User Interface:
15     1. The user is asked to enter:
16          Current salary
```

```
    2. Output to the user:
         Updated salary after increment
"""
salary = int(input('Enter the salary: '))
assert salary>=0
salary = incrementSalary(salary)
print('Updated Salary:', salary)
```

Modification to an integer object passed to a function as an argument: not reflected in caller's environment

Modification to an mutable object passed to a function as an argument: reflected in caller's environment

In Program 2.1, we noted that when an integer object was passed to a function as an argument, the modifications to the argument in the function did not result in a corresponding change in the environment of the function caller. In contrast, suppose we pass a mutable object such as a list or dictionary to a function and modify it in the function (Program 2.3). In that case, the modifications in the object are reflected in the caller's environment because we are not modifying the original reference to the argument. For example, suppose salaries of three employees are stored in a list and we invoke the function **`incrementSalary()`** (see Fig 2.2(a)). It is important to understand that in the first example, the assignment statement inside the function **`incrementSalary()`** created an *entirely new object* **`70000`** to which the formal parameter **`salary1`** referred to, leaving the object salary in the caller's environment unchanged. However, in Program 2.3, the components of the object **`salary1`** (also being referred to by **`salary`** in the caller's environment) were modified, as shown in Fig 2.2(b). Therefore, the modifications were visible in the caller's environment (see Fig 2.2(c)).

Program 2.3 Incrementing salaries of several employees

```
def incrementSalary(salary1, increment = 20000):
    """
    Objective: To increment the salary of the employees
    Input Parameters:
    salary - list containing salaries of the employees
    increment - numeric value of increment
    Return value: None
    """
    for i in range(len(salary1)):
        salary1[i] = salary1[i] + increment

#main program segment
"""
Objective: To increment the current salary
User Interface:
    1. User is asked to enter:
         Current salary of n employees
    2. Output to the user:
         Updated salary after increment
"""
nEmployees = int(input('Enter the Number of Employees: '))
salary = []
for i in range(nEmployees):
    salary.append(int(input('Enter the salary of an employee: ')))
    assert salary[i]>=0
incrementSalary(salary)
print('Updated Salary:', salary)
```

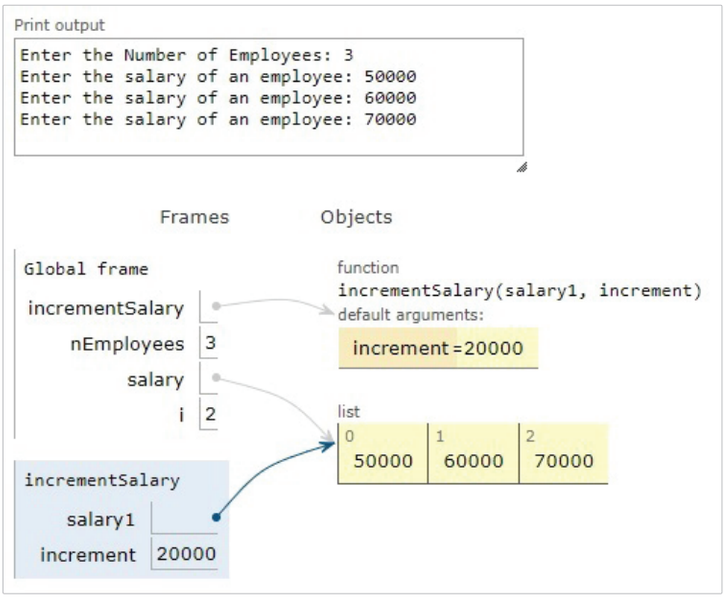


Fig 2.2(a): Call to the function incrementSalary

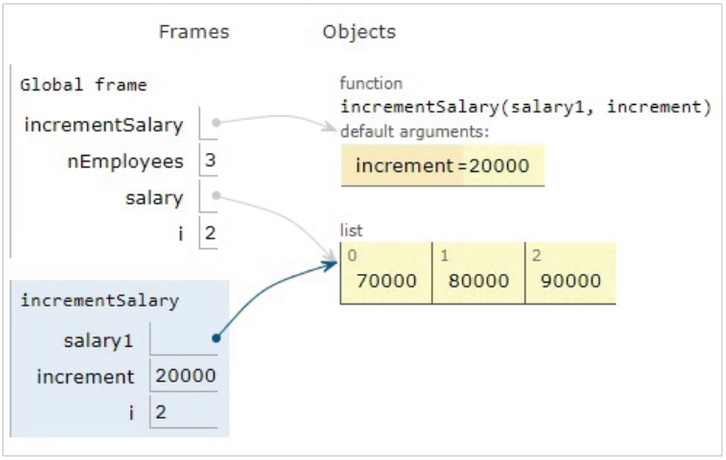


Fig 2.2(b) Object **`salary1`** has been updated in the function **`incrementSalary.`**
Note that the changes are also seen in the caller's environment.

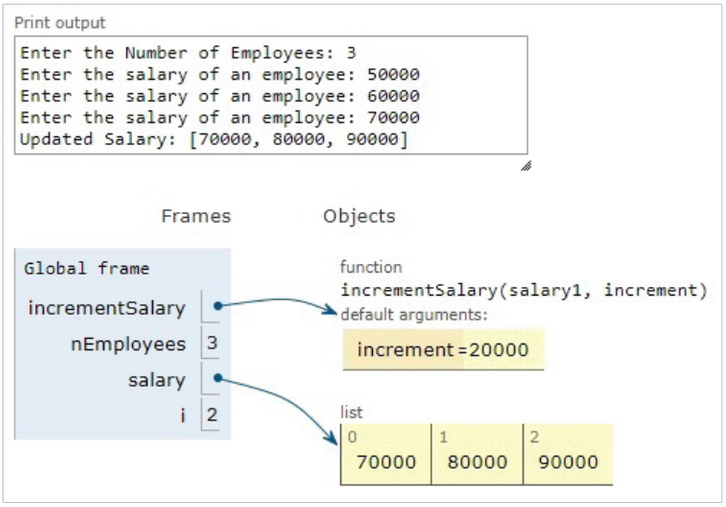


Fig 2.2(c): Objects in the environment of the global frame on exit from the function **`incrementSalary`**().

Fig 2.2: Data objects on execution of Program 2.3

Finally, let us consider one more example (Program 2.4) where we create a new object in the function **incrementSalary()** (Please see Fig 2.3(a) to 2.3(d)). For example, suppose the salaries of three employees are stored in a list salary and we invoke the function **incrementSalary()** (see Fig 2.3(a)).

- Line 9 creates a new object **localSalary**. Thus, any subsequent updates in the salary are in a local list as shown in Fig 2.3(b).
- Finally, in line 12, assignment to **salary1** creates a new reference for **salary1**, leaving the object **salary** untouched, as shown in Fig2.3(c).
- Therefore, on exit from the function **incrementSalary**, the list **salary** remains as before the call to the function **incrementSalary** (Fig2.3(d)).

Program 2.4 Incrementing salaries of several employees

```
01 def incrementSalary(salary1, increment = 20000):
02     """
03     Objective: To increment the salary of the employees
04     Input Parameters:
05       salary - numeric value denoting salary of the employee
06       increment - numeric value increment
07     Return value: None
08     """
09     localSalary = []
10     for i in range(len(salary1)):
11         localSalary.append(salary1[i] + increment)
12     salary1 = localSalary
13 #main program segment
14 """
15 Objective: To increment the current salary
16 User Interface:
17     1. User is asked to enter:
18          Current salary of n employees
19     2. Output to the user:
20          Updated salary after increment
21 """
22 nEmployees = int(input('Enter the Number of Employees: '))
23 salary = []
24 for i in range(nEmployees):
25     salary.append(int(input('Enter the salary of an employee: ')))
26     assert salary[i]>=0
27 incrementSalary(salary)
28 print('Updated Salary:', salary)
```

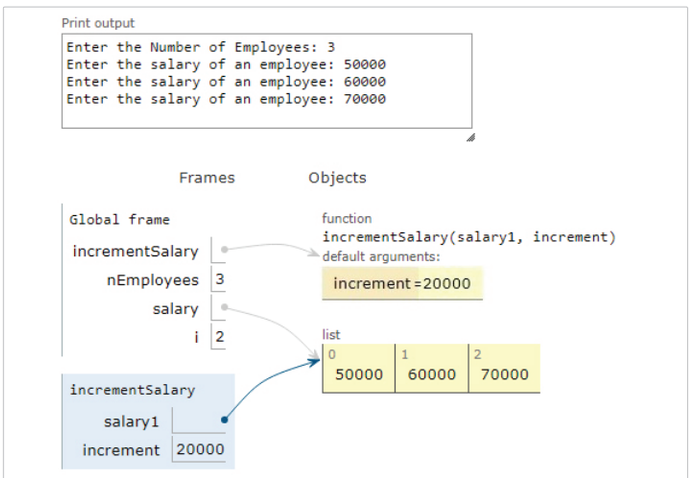


Fig 2.3 (a): At the point of call

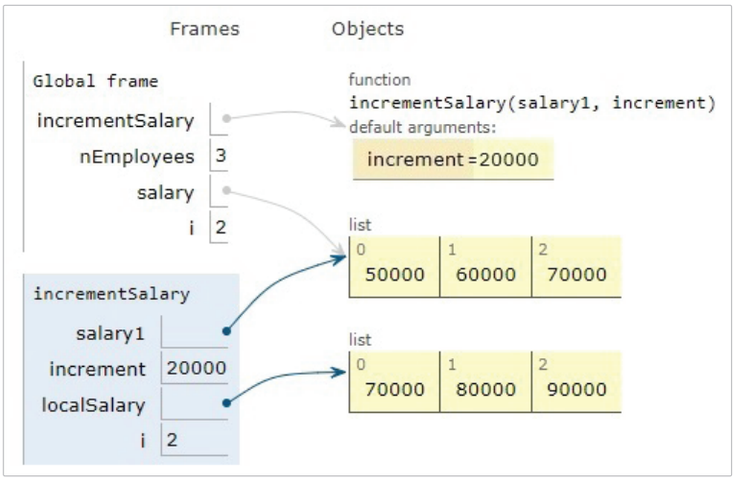


Fig 2.3 (b): Updates in the salary are in a local list

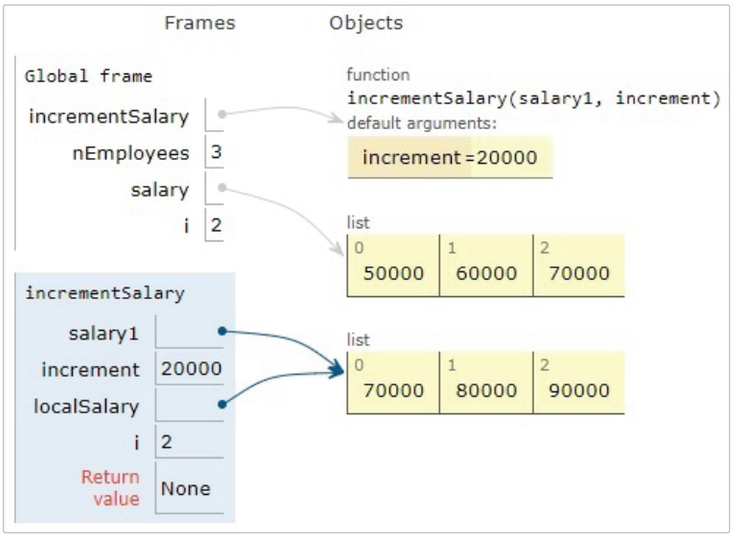


Fig 2.3 (c) Assignment to **`salary1`** creates a new reference for **`salary1`**, leaving the object salary untouched.

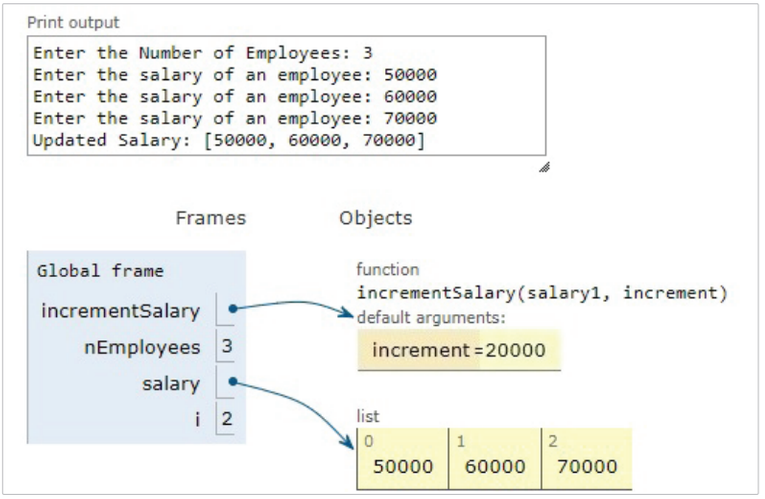


Fig 2.3 (d) On exit from the function **`incrementSalary()`**, the list **`salary`** remains as before the call to the function **`incrementSalary()`.**

Fig 2.3: Data objects during execution of Program 2.4

State whether the following statements are True or False.

1. A local variable can be edited anywhere in the program.
2. A local variable defined in a function is active only during the execution of the function.
3. The input parameters of a function have local scope.

2.1.2 Global Names (Global Variables)

A variable defined outside the functions is known as a global variable. It can be accessed inside as well as outside of functions. A global variable can be modified inside any function by preceding it with the keyword **`global`**.

Global variable: variable defined outside the functions and can be accessed inside as well as outside of functions. A global variable can be modified inside any function by preceding it with the keyword global.

Thus, modifications in the value of a global variable become permanent and therefore visible in any subsequent call to a function that makes use of it. If a global variable has the same name as a local variable inside a function, then the global variable is overridden or hidden.

If a global variable has the same name as a local variable inside a function, then the global variable is overridden.

The scope rules for accessing a name inside a function, say, **`func()`**, are as follows:

- The name is first searched in the scope local to the function **`func()`**.
- If the name is not found in the local scope of the function **`func()`**, Python searches for it outside the function (in global scope).
- If the name is not found in the local scope of the function **`func()`** as well as in the global frame, it is searched in the set of built-in names.

Scope Rule: search in local scope followed by global scope (outside the function), and thereafter in the set of built-in names.

We begin our study of the global scope with some simple examples. Let us examine Program 2.5.

Program 2.5 Using Global Scope

```
01 a = 10
02 def func():
03     print('Locally in function func(), a =', a)
04 func()
05 print('In global frame, a =', a)
```

On executing Program 2.5, Python responds with the following output:

```
Locally in function func(), a = 10
In global frame, a = 10
```

- On invoking the function **`func()`** (line 4), when Python encounters the name a (line 3), the name a is first searched in the local scope of function **`func()`**.
- As the search fails, name a is searched in the global scope, where it is found, and its value is printed (see Fig2.4). When the call to function **`func`** terminates, the control returns to the global frame.
- As the name **`a`** has a global scope, it is accessible from any function defined in the module as well as in the global frame. Hence, the value of **`a`** is printed on the execution of line 5.

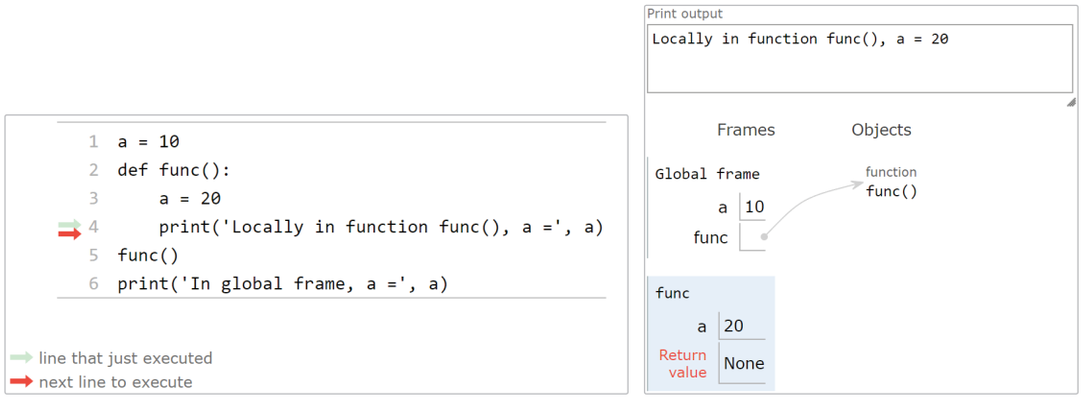


Fig 2.4: Data objects on the execution of line 3 in Program 2.5

Next, let us examine Program 2.6, which incorporates a minor change in Program 2.5. This program has a trivial error: the variable **a** is referenced when line 1 is executed before it is defined in the global frame.

Program 2.6 Using Global Scope

```
01 print('In global frame, a =', a)
02 a = 10
03 def func():
04     print('Locally in function func(), a =', a)
05 func()
   Traceback (most recent call last):
     File "F:\DOWNLOADS\scope1.py", line 1, in <module>
       print('In global frame, a =', a)
   NameError: name 'a' is not defined
```

- Now, let us examine a more interesting example. In Program 2.7, when Python encounters the definition of the function **`func()`**, it notes this.
- We already know that the statements in a function are executed only when it is invoked. So, a reference to the variable a in line 2 does not conflict with Python syntax.
- In line 3, the variable a is assigned value 10 in the global frame. Subsequently, when the function **`func()`** is invoked, and the variable a is referenced, its value is already known (see Fig2.5).

Program 2.7 Using Local Scope

```
01 def func():
02     print('Locally in function func(), a =', a)
03 a = 10
04 func()
```

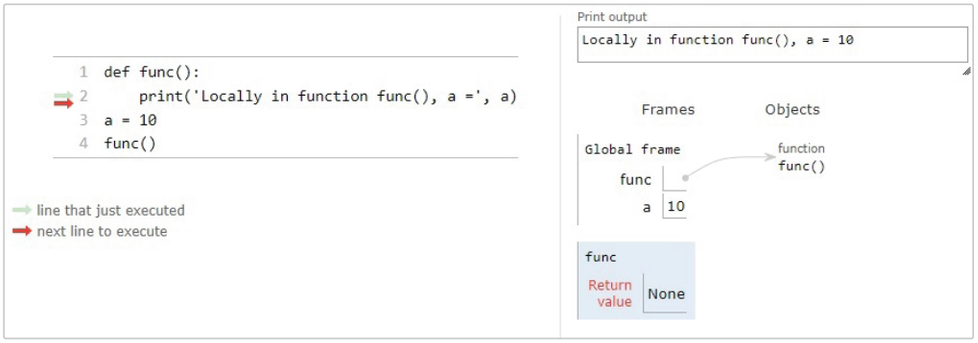


Fig 2.5: Data objects on execution of line 2 in Program 2.7

- Next, let us examine Program 2.8. Line 1 assigns the value 10 to the variable a in the global frame.
- When the function **`func()`** is invoked on the execution of line 5, line 3 assigns value 20 to variable **`a`**.
- Note that variable a in line 3 is local to the function. When access to the variable a is made in line 4, the name a is first searched in the local scope of the function **`func()`**.
- As the variable a is successfully found in the local scope of the function **`func()`**, the value 20 gets assigned to it on execution of line 3.
- Next, this value is printed on execution of line 4 and value 20 gets printed (see Fig 2.6).
- When the call to function **`func()`** terminates, the control returns to the global frame. In line 6, the value 10 of a in the global frame is printed (see Fig 2.6).

Program 2.8 Using Local Scope

```
01 a = 10
02 def func():
03     a = 20
04     print('Locally in function func(), a =', a)
05 func()
06 print('In global frame, a =', a)
```

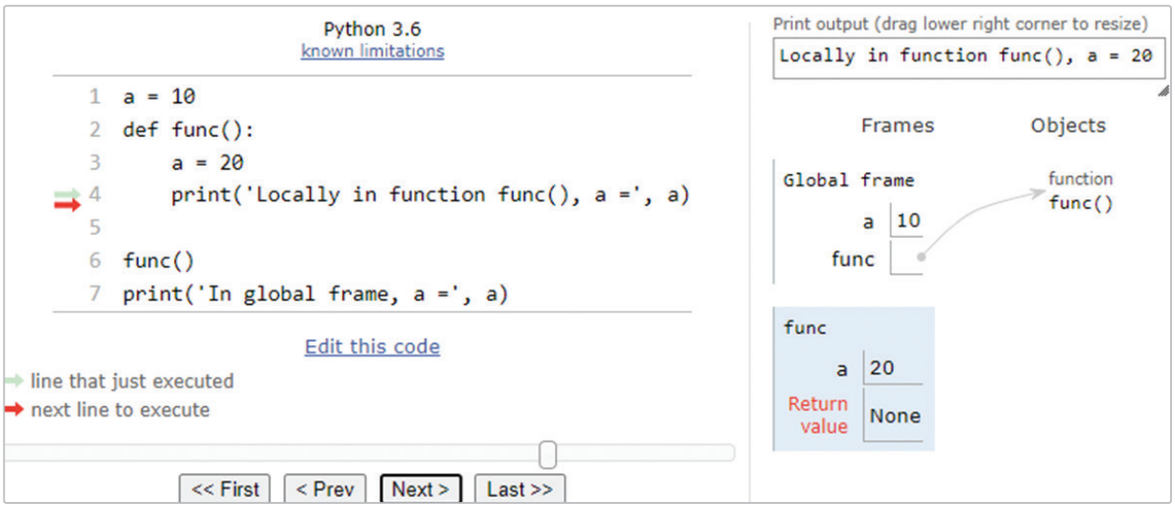


Fig 2.6: Data objects on execution of line 4 in Program 2.8

On executing Program 2.8, Python responds with the following output:

```
Locally in function func(), a = 20
In global frame, a = 10
```

2.2 Updating Value of a Local Variable

In Program 2.9, the execution of line 3 creates a local variable **`a`**, having value **`110`** which is subsequently updated to **`120`** on execution of line 4. However, the variable **`a`** in the global frame remains unaffected by this modification (see Fig 2.7).

Updating value of local variable: changes to a local variable in a function does not affect the variable with the same name in global scope

Program 2.9 Using Local and Global Variables

```
01 a = 90
02 def func():
```

```
03      a = 110
04      a += 10
05      print('Locally in function func(), a =', a)
06  func()
07  print('In global frame, a =', a)
```

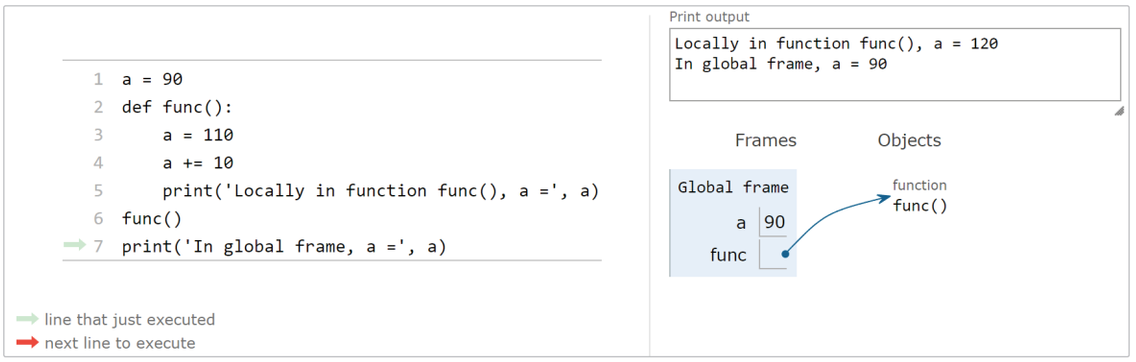


Fig 2.7: Data objects on execution of line 5 in Program 2.9

Finally, Program 2.9 produces the following output:

```
Locally in function func(), a = 120
In global frame, a = 90
```

2.3 Updating Value of a Global Variable

- In Program 2.10, we wish to update the value of a global variable, i.e., a variable in the global frame.
- Line 3 in Program 2.10 indicates that the variable **a** in the function **fun** refers to the variable a in the global frame (line 1).
- Hence, the execution of line 4 updates the value of variable a in the global frame (see Fig 2.8).

Program 2.10 Using Local and Global Variables

```
01  a = 90
02  def func():
03      global a
04      a += 10
05      print('Locally in function func(), a =', a)
06  func()
07  print('In global frame, a =', a)
```

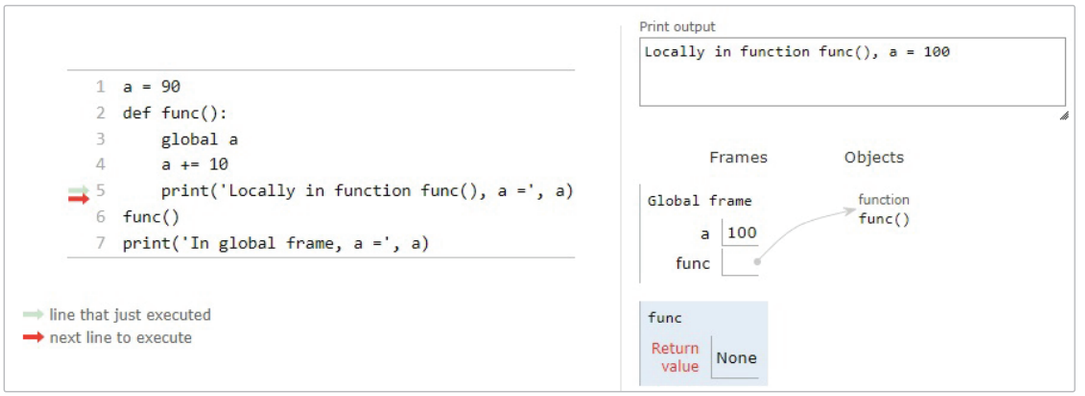


Fig 2.8: Data objects on the execution of line 5 in Program 2.10

On executing Program 2.10, Python responds with the following output:

```
Locally in function func(), a = 100
In global frame, a = 100
```

2.4 Introducing Global Variable in a Function

A global variable may also be introduced in a function, for example, see Program 2.11. In Fig 2.9, note that the variable **a** has been introduced in the global frame.



Introduce a global variable in a function: by preceding variable by a global keyword

Hence, it is available even after the execution of function **func()** terminates (see Fig 2.9).

Program 2.11 Using Local and Global Variables

```
01 def func():
02     global a
03     a = 10
04     print('Locally in function func(), a =', a)
05 func()
06 print('In global frame, a =', a)
```

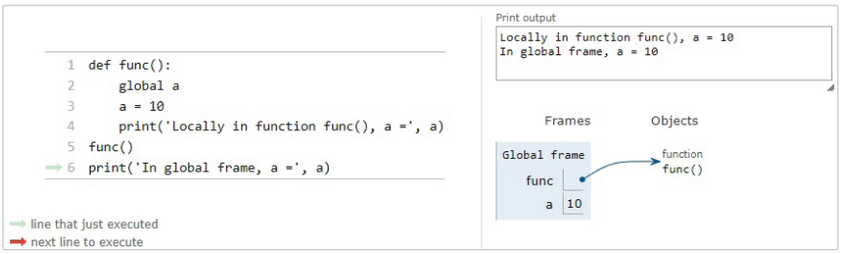


Fig 2.9: Data objects during the execution of Program 2.11

2.5 Built-in Names

When a name is referred in a Python program, but it is found neither in the local scope nor in the global scope, it is searched for in the built-in scope. For example, let us examine Program 2.12. When Program 2.12 is executed, it first encounters the definition of the function **func()**, followed by a call to the function **func()**. Inside the function **func()**, it encounters the function name **abs**. As the function **abs** is not available inside the body of the function **func()**, Python searches for it in the global environment but does not find it in the global frame also, so Python searches for it in the built-in environment, where it indeed finds the function **abs**.

Program 2.12 Reference to a name in built-in environment

```
01 def func():
02     print(abs(-4))
03
04 func()
```

On executing Program 2.12, Python responds with the following output:

```
>>> 4
```

2.6 Imported Names

Functions and variables that are imported in the global environment remain accessible in the global environment as well as in all the functions in the program. For example, in program 2.13, we import the variable **pi**, in the global frame but use it in the function **areaCircle()** (see Fig 2.10).



Imported names: accessible in global environment as well as in all functions in the program i.e. within the scope they have been imported.

Program 2.13 Using a global variable inside a function

```
01 from math import pi
02 def areaCircle(radius):
03     """
04     Objective: To compute the area of the circle
05     Input Parameter:
06         radius- numeric value denoting radius of circle
07     Return value: area of circle
08     """
09     area = pi * radius ** 2
10     return area
11
12 radius = 5
13 print('radius of circle = ', radius, 'area of circle = ', \ round(areaCircle(5),2))
14 print('value of pi = ', pi)
```

```
radius of circle =  5 area of circle =  78.54
value of pi =  3.141592653589793
```

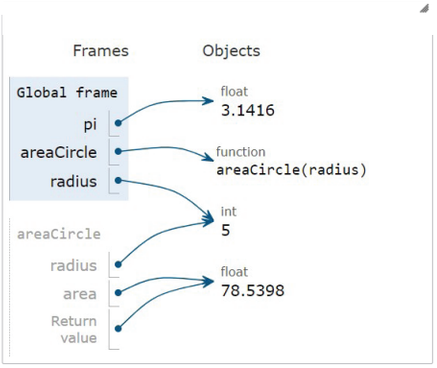


Fig 2.10: Using a global variable inside a function

2.6.1 Imported Names are Accessible Only within Applicable Scope

An imported name can only be accessed within its scope. For example, let us examine Program 2.14 with a minor modification.

Program 2.14 Using a global variable inside a function

```
01 def areaCircle(radius):
02     """
03     Objective: To compute the area of the circle
04     Input Parameters:
05         radius- numeric value denoting radius of circle
06     Return value: area of circle
07     """
08     from math import pi
09     area = pi * radius ** 2
10     return area
11
12 radius = 5
13 print('radius of circle = ', radius, 'area of circle = ', round(areaCircle(5),2) )
14 print('value of pi = ', pi)
```

In Program 2.14, as **`pi`** has been imported within the function **`areaCircle()`**, it is not accessible in the global frame. Hence, we get an error when an attempt is made to print the value of **`pi`** in the global frame.

On executing Program 2.14, Python responds as follows:

```
>>> radius of circle = 5 area of circle = 78.54
Traceback (most recent call last):
  File "D:/scope9.py", line 14, in <module>
    print('value of pi = ', pi)
NameError: name 'pi' is not defined
```

2.6.2 Built-in and Imported Names: Be Careful Not to Modify

Python permits us to modify the objects associated with built-in and imported functions and variables. So, we should be very careful while using the user-defined names as accidental modification of built-in and imported functions and variables may lead to a disaster. For example,

```
>>> abs = 6
>>> abs(-4)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    abs(-4)
TypeError: 'int' object is not callable
```

Note that Python does not recognise **abs** as a function because we made the mistake of redefining **abs** as an integer by setting:

```
abs = 6
```

Think twice before modifying imported and built-in names. Best, avoid such adventures unless required.

CT 01

Determine the output produced on the execution of following code:

```
num = 25
from math import sqrt
def practice(x):
    global num
    num = sqrt(num) + x
    x = x % 7
    return x
a = 30
final = practice(a)
print(final)
```

Match the following with respect to the above code.

1. Callee function	a. final
2. Global environment	b. x
3. Function parameter	c. globally imported
4. sqrt()	d. practice()

CT 02

Declare a variable **balance**, in global scope, initialised to value zero. Define a function **addAmount** that takes the input parameter **amount** and updates the variable **balance** by adding the **amount** to it.

However, one should avoid indiscriminate use of the global variables as it makes the programs unmanageable for debugging and maintenance.

A Practical Example: Updating Account Balance

Suppose you need to keep track of the current balance in your account. The balance should be updated whenever a transaction takes place. For example, when you deposit (add) money in your account, or when you withdraw money from your account.

- To keep track of balance in the account, initially we set **`balance`** to be **`0`** (line 2, Program 2.15).
- For updating the balance, we define functions **`credit()`** (lines 3–11) and **`debit()`** (lines 13–21) in the program. The function **`credit()`** takes the amount to be added to the balance as the input parameter. Similarly, the function **`debit()`** takes the amount to be debited from the balance as the input parameter.
- To ensure that the function does not create a local object **`balance`** and updates the global object **`balance`**, we explicitly specify that the variable balance refers to the global frame (line 10 and 20).

Program 2.15 Updating account balance

```
01 global balance
02 balance = 0 # current balance in account
03 def credit(amount):
04     """
05     Objective: To update the balance when amount is credited
06     Input Parameters:
07      amount - amount to be credited: numeric value
08     Return value: None
09     """
10     global balance
11     balance = balance + amount
12
13 def debit(amount):
14     """
15     Objective: To update the salary of the employee when amount is debited
16     Input Parameters:
17      amount - debit amount: numeric value
18     Return value: None
19     """
20     global balance
21     balance = balance - amount
22
23 #main program segment
24 """
25 Objective: To update balance in account on credit or debit
26 User Interface:
27     1. The user is asked to enter:
28          Transaction type- Credit/ Debit
29     2. Output to the user:
30          Updated balance
31 """
32 while True:
33     print("Choose an Option:")
34     print("1: Credit")
35     print("2: Debit")
36     print("3: Check bank balance")
37     print("0: Exit")
38     choice = int(input(""))
39     if choice == 1:
40         amount = float(input("Enter credit amount: "))
```

```
41          assert amount >= 0
42          credit(amount)
43      elif choice == 2:
44          amount = float(input("Enter debit amount: "))
45          assert amount >= 0
46          debit(amount)
47      elif choice == 3:
48          print('Current bank balance:', balance)
49      elif choice == 0:
50          break
51      else:
52          print("Wrong Input!")
```

In a sample execution of the Program 2.15, Python responded with the following output:

```
    Choose an Option:
    1: Credit
    2: Debit
    3: Check bank balance
    0: Exit
    1
>>> Enter credit amount: 5000
    Choose an Option:
    1: Credit
    2: Debit
    3: Check bank balance
    0: Exit
    3
    Current bank balance: 5000.0
    Choose an Option:
    1: Credit
    2: Debit
    3: Check bank balance
    0: Exit
    2
>>> Enter debit amount: 1500
    Choose an Option:
    1: Credit
    2: Debit
    3: Check bank balance
    0: Exit
    3
    Current bank balance: 3500.0
    Choose an Option:
    1: Credit
    2: Debit
    3: Check bank balance
    0: Exit
    1
>>> Enter credit amount: 5000
    Choose an Option:
    1: Credit
    2: Debit
    3: Check bank balance
    0: Exit
    3
    Current bank balance: 8500.0
    Choose an Option:
    1: Credit
    2: Debit
```

```
3: Check bank balance
0: Exit
0
```

It is best to avoid global variables to the extent possible as indiscriminate use of global variables makes the program unmanageable, especially when the code is being developed by a team of several programmers.

Let's Summarise

- The scope of a named object (variable) refers to the part of a program in which it is accessible.
- The variables defined inside a function are called local variables.
- The time period during which a variable is active (during execution of the program) is known as its lifetime.
- A variable defined outside the functions is known as a global variable.
- An imported name can only be accessed within its scope.
- We can modify a global variable inside any function by using the keyword **`global`**.

Solved Exercises

A. Multiple Choice Questions

1. Which of the following describes the lifetime of a variable?

a. Part of the program where the object is accessible.

b. Part of the program where the object is not accessible.

c. The time period during the execution of the program when the variable is active.

d. The time period during the execution of the program when the variable is not active.

2. Which of the following is true with respect to the scope of a variable?

a. A global variable introduced in a function cannot be accessed outside of it.

b. The input parameters passed to a function have local scope.

c. A name accessed in a function is first searched in the global frame and then locally within the definition of a function.

d. A global variable cannot be modified inside the function.

3. If a local variable is accessed outside the function in which it is defined, which of the following will be the possible outcomes?

a. An error will be generated.

b. The value of local variable will become 0.

c. It will be made global.

d. The value of local variable will become None.

4. Which of the following statements is False?

a. A variable referenced in a function is first searched within the function and then in the global frame.

b. A variable defined in a function is first searched in its global scope and then in the function.

c. A global variable can be modified inside any function by preceding it with the keyword **`global`**.

d. Access to local variables outside the enclosing function will result in an error.

5. What will be the output produced on execution of the following code?

```
x = 10
def test():
    global x
    x = x + 10
    print(x, end = " ")
test()
print(x)
```

a. 10 10 b. 10 20 c. 20 10 d. 20 20

B. State whether the following statements are True or False:

1. If a global variable has the same name as a local variable inside a function, then the global variable is overridden. _____
2. The lifetime of a global variable introduced in a function ends when the function is completely executed. _____
3. Variables defined inside a function have global scope. _____
4. If a variable is declared global in a function, such a declaration must precede any reference to it in the function. _____
5. An imported name can only be accessed within its scope. _____

C. Fill in the blanks.

1. Names of the data objects are also called _____ .
2. The part of the program where a variable is accessible is known as its _____ .
3. If a variable is not found locally within a function, it is searched in the _____ frame.
4. A global variable can be modified inside any function by preceding it with the keyword _____.
5. Access to a _____ variable outside the function will result in an error.

D. Answer the following questions:

1. Differentiate between local variables and global variables.

Ans. **Local Variable:** A local variable is declared within a function, without preceding the word global.

Global Variable: is accessible from any function within the module containing it, but only after it has been introduced. When a global variable is introduced or modified inside a function, it must be declared global by preceding it by the keyword global.

2. How can we edit a global variable inside a function?

Ans. Before editing any global variable inside a function, it must be declared global by using the keyword global.

3. What will be the output produced on execution of the following program?

```
01  nbooks = 5                                          # global variable
02  print('The global count of books is', nbooks)
03  def func1():
04      nbooks = 10                                      #local variable
05      print('The count of books inside the function func1 is ', nbooks)
06  def func2():
07      global nbooks    # accessing global variable to edit its value
08      nbooks = nbooks + 20
09      print('nbooks = ', nbooks)
10  func2()
11  func1()
12  print('The changed value globally is ', nbooks)
13  func2()
```

Ans.

```
The global count of books is 5
nbooks = 25
The count of books inside the function func1 is 10
The changed value globally is 25
nbooks = 45
```

4. Will the following code produce an error? Justify your answer.

```
a = 100
def localGlobal(x):
    global x
    a = a + x
    return
b = 10
```

```
localGlobal(b)
print(a, b, sep='*')
```

Ans. The given code will produce an error because x is passed as formal parameter, which behaves like a local variable in the function. So, it cannot be declared globally by preceding it with keyword **`global`**.

5. Write a program that declares maximum marks (maxMarks) as a global variable with value 80. Also define a function **calcPerc()**, that takes **marksObt** as argument and returns the percentage.

Ans.
```
maxMarks = 80
def calcPerc(marksObt):
    return marksObt / maxMarks * 100
marks = int(input("Enter Marks Out of 80"))
print('Marks =', marks, 'Max Marks =', maxMarks, 'Percentage =', calcPerc(marks))
```

6. What will be the output produced on execution of the following program? Will the modified value of global variable be available on exit from the function **`calcPerc()`**?

```
maxMarks = 80
def calcPerc(marksObt):
    global maxMarks
    maxMarks = 50
    return round(marksObt / maxMarks * 100)
marks = 43
print(calcPerc(marks))
```

Ans. The output will be 86

Yes, the value of global variable maxMarks will be available in the global frame.

7. What will be the output on execution of the following code:

```
def f1():
    global a
    a = 10
def f2():
    print('Locally inside function f2, a = ', a)
f1()
f2()
```

Ans. When the function **`f1`** is invoked, it introduces a global variable **`a`**, which remains active in the global frame when the function **`f1`** terminates. So, when the function f2 is introduced, the global variable **`a`** remains accessible in the function f2. Hence, the following output will be produced:

```
Locally inside function f2, a = 10
```

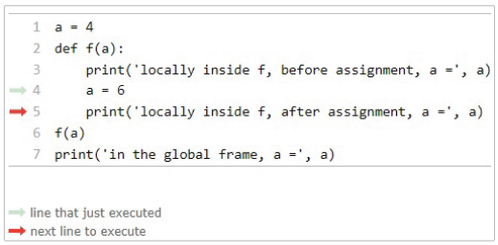
8. What will be the output produced on execution of the following program? Justify your answer.

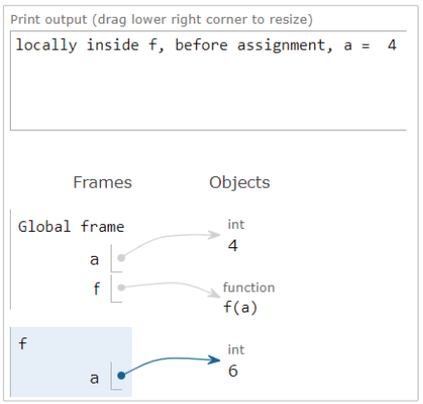
```
a = 4
def f(a):
    print('locally inside f, before assignment, a =', a)
    a = 6
    print('locally inside f, after assignment, a =', a)
f(a)
print('in the global frame, a =', a)
```

Ans. The following output will be produced:

```
locally inside f, before assignment, a = 4
locally inside f, after assignment, a = 6
in the global frame, a = 4
```

On entry to the function, the parameter **a** refers to the same integer object as the global variable **a**. Subsequently, when the assignment statement is executed, the parameter **a** refers to a new integer object (6), distinct from the global object. Therefore, the modification in the value of **a** inside the function is local to the function. Hence, when the function terminates, the value of the global variable **a** is printed as illustrated below:





9. What will be the output produced on execution of the following program? Justify your answer.

```
a = 4
def f1(a):
    print('locally inside f1, before assignment, a =', a)
    a = 6
    print('locally inside f1, after assignment, a =', a)
def f2():
    print('locally inside f2, before assignment, a =', a)
f1(a)
f2()
```

Ans. The above program will produce the following output:

```
locally inside f1, before assignment, a = 4
locally inside f1, after assignment, a = 6
locally inside f2, before assignment, a = 4
```

On entry to the function, the parameter **a** refers to the same integer object as the global variable **a**. Subsequently, when the assignment statement is executed, the parameter **a** refers to a new integer object (6), distinct from the global object. Therefore, the modification in the value of **a** inside the function is local to the function. Hence, when the function terminates, the value of the global variable **a** is still **4**. When the function **f2** is invoked, and the value of **a** is to be printed, Python looks for the name **a**, within the function **f2**, as the local search fails, Python looks for **a** in the global frame, where it finds the global variable **a**, and prints its value 4.

Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): A global variable is defined outside all user-defined functions of a program.
Reasoning (R): A global variable can be accessed anywhere in the program.

2. Assertion(A): Before modifying the value of a global variable inside a user-defined function, it must be declared global by using the keyword **`global`**.
   Reasoning (R): In the absence of the keyword global, modifying the value of a variable declared global in the program introduces a new local variable by the same name.

Ans. 1. d 2. a



Case-based Questions

1. Agamvir is a bit confused about using global variables in a local scope. He has written the following program:

```
finalMarks=50
def checkMarks():
    ut1Marks=15
    finalMarks+=ut1Marks
    print(finalMarks,ut1Marks)
checkMarks()
```

The program is giving an error. What could be the possible reason? Also, make the required correction in the program.

Ans. Agamvir should use the keyword global before editing the global variable, finalMarks. The correct code is as follows: '

```
finalMarks=50
def checkMarks():
    global finalMarks
    ut1Marks=15
    finalMarks+=ut1Marks
    print(finalMarks,ut1Marks)
checkMarks()
```

2. Shuchi is a student in class 11. Her computer science teacher has explained the concept of local and global variables in class. The teacher has asked the students to create a program in Python that calculates the percentage of marks. The teacher has given the following specifications for the program:
   - The variable maxMarks, which stores maximum marks should be defined in the global scope. The value of maximum marks is 50
   - A function, calcPercent() should accept marks obtained from the user and display the percentage.

   Help Shuchi complete the task.

Ans.

```
maxMarks=50
def calcPercent(marks):
    perc=(marks/maxMarks)*100
    print("Your percentage is :  ", perc)

marks = int(input("Enter the marks obtained :  "))
calcPercent(marks)
```



Assessment

A. Multiple Choice Questions

1. Which of the following refers to a variable that is defined inside a function?
   a. Global variable b. Local variable c. Both a and b d. Either a or b

2. What is the scope of a parameter given during function definition?
   a. Local b. Global c. None d. Null

3. What is the scope of the variable that is defined outside of every function?
   a. Local b. Global c. Varied d. None

4. Consider the following code:

```
num=10                    # Statement 1
def test(number):
    print(num)            # Statement 2
    print(number)         # Statement 3
print(num)                # Statement 4
```

   Which of the following is the correct statement?
   a. num has local scope
   b. number has local scope
   c. Statement 4 will generate an error
   d. Statement 2 will generate an error

5. Consider the following code:

```
num=10
def test(number):
    num=num+10                    # Statement 1
    number=number+10              # Statement 2
    print(number)                 # Statement 3
print(num)                        # Statement 4
```

   Which of the following statements will generate an error?
   a. Statement 1 b. Statement 2 c. Statement 3 d. Statement 4

B. State whether the following statements are True or False:

1. A global variable in Python can be accessed from anywhere within the program. _____
2. If a variable with the same name as a global variable is defined in a function, an error is generated in Python. _____
3. In order to use a global variable within a function having a local variable of the same name, we need to use the keyword "global". _____
4. The variables defined inside a function are called local variables. _____
5. A local variable defined in a function is said to be active during the execution of the function. _____

C. Fill in the blanks.

1. The part of the program where a named Python object is accessible is called its _____.
2. A _____variable is accessible within the function in which it is defined.
3. A _____ variable is accessible inside as well outside the functions in a program.
4. Access to a local variable outside the function will result in an _____.
5. The input parameters of a function are _____(accessible / not accessible) within a function's scope like local variables.

D. Answer the following questions.

1. Define scope and lifetime of a variable.
2. How is a global variable different from a local variable?
3. How can you access a global variable inside a function? Give an example.
4. Differentiate between local scope and global scope of a variable. Illustrate your answer with the help of an example.
5. Do the functions and variables that are imported in a function remain accessible in the global frame when the function terminates? Justify your answer.
6. In the following code, identify the local and global variables. Also write the output.

```
local = 10
global_1 = 20
def myFunc(num):
    local = 3
```

```
        local = local * 5
        global global_1
        global_1 = global_1 + 100
        print(local, global_1, sep='@@')
    myFunc(local)
    print(local, global_1, sep='@@')
```

7. Identify errors, if any, in the following program. If there are no errors, what will be the output on execution of the program? Justify your answer.

```
count = 0
def countSpaces(str1):
    for d in str1:
        if d == ' ':
            count += 1
    return count
sentence1 = 'How are you?'
print('No of spaces in "' + sentence1 + '" =', countSpaces(sentence1))
sentence2 = ' I am good, Thank you'
print('No of spaces in "' + sentence1 + sentence2 +  '" =', countSpaces(sentence2))
```

8. Identify errors, if any, in the following program. If there are no errors, what will be the output on execution of the program? Justify your answer.

```
count = 0
def countSpaces(str1):
    global count
    for d in str1:
        if d == ' ':
            count += 1
    return count
sentence1 = 'How are you?'
print('No of spaces in "' + sentence1 + '" =', countSpaces(sentence1))
sentence2 = ' I am good, Thank you'
print('No of spaces in "' + sentence1 + sentence2 +  '" =', countSpaces(sentence2))
```

9. What will be the output of the following code?

```
myList=[3, 9, -2, 0, 8, 6]
def leftShift(lst, n=1):
    length = len(lst)
    for x in range(0,n):
        first = lst[0]
        for y in range(0, length-1):
            lst[y] = lst[y + 1]
        lst[length-1] = first
print(myList)
leftShift(myList)
print(myList)
leftShift(myList, 2)
print(myList)
```

Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): A variable referenced in a function is first searched in the global frame and then within a function.
Reasoning (R): Access to local variables outside the enclosing function will result in an error.

2. Assertion(A): A variable declared with a global scope will be accessible in the entire program.
Reasoning (R): The time during which a variable is active is called the lifetime of a variable.

Case-based Questions

1. Write a program that accepts the name and marks of a student and prints the grades according to the percentage scored by the student. Make maxMarks (maximum marks) as a global variable and use user-defined functions to accept data and calculate grades. The output should look like:

```
Name: Veena Dua
Marks: 70
Grade:  B
```

Use the following table as a reference.

Percentage	Grade
>=90	A
>=60 and <90	B
>=40 and <60	C
<40	E

2. Sumedha wants to create a program to calculate the discounted sale price of items after deducting the discount from the sale price. She wants to have the global value of discount as 10% but for category of apparels and stationery, she wants the discount to be 12% and 5% respectively. Help her to develop a function **`discountedPrice()`** to compute the discounted price and subsequently write the complete program that invokes the function **`discountedPrice()`**. The program output should appear as follows:

Item Name: T Shirt

Category: Apparels

Sale price: 100

Discount: 12%

Discounted Sale Price: 88

Answers

Multiple Choice Questions

1. (c) 2. (b) 3. (a) 4. (b) 5. (d)

True or False

1. (T) 2. (F) 3. (F) 4. (T) 5. (T)

Fill in the blanks

1. variables 2. scope 3.global 4. global 5. local

3 HANDLING FILES IN PYTHON

Chapter Outline

3.1 Understanding Data Files
3.2 Opening and Closing Files
3.3 Reading and Writing Files
3.4 File Object Attributes
3.5 Standard Input/Output Devices as Files
3.6 Random Access Using seek () and tell ()
3.7 Handling CSV Files in Python
3.8 File Processing Applications

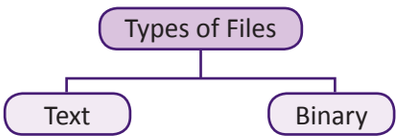
Introduction

We write programs to perform computation using several data objects. The data being processed is either included in the program itself, or is taken as input from the user. Several programs produce the results of computations in the form of data objects. These data objects remain in the main memory (RAM) and are accessible as long as we execute the program. Once the program terminates, these data objects are no longer accessible. In several real-life applications, we may need to store the user inputs or results of computation permanently. For example, during the registration at an e-commerce website, the login credentials provided by the user need to be stored permanently at the backend because they may be required subsequently during login for authenticating the user. Further, the output produced by a program may be used as input by another program. In Python, files provide a mechanism to store data for subsequent use. On a typical computer, input is taken from the standard keyboard results are displayed on the output console.

3.1 Understanding Data Files

A file is a stream of bytes comprising relevant data. It provides a source of interaction between the program and the user. Data objects in a file may be in the form of character sequences, or more complex objects such as lists, dictionaries, or objects of user-defined types. There are mainly two types of files used in Python: text files and binary files.

File: Stream of bytes. Provides a means to store data permanently.



Let us learn about them in detail.

3.1.1 Text Files

A text file is a human-readable file that comprises of a sequence of characters stored in the form of ASCII or UNICODE characters. Each line in a text file is terminated by an End of Line **(EOL)** character that may vary across operating systems. For example, UNIX uses the character **\n** to denote end of line whereas Microsoft's DOS uses a pair of characters **<CR>** and **<LF>** to transfer control to the beginning of the next line. A text file is recognized by its name having a *.txt* extension.



Text File: human-readable file. Line terminated by EOL.

A delimited text files makes use of a delimiter is to separate the contents in each line. For example, in a comma-separated file `(.csv extension)`, the data values in each line are separated by commas. Thus, commas act as delimiters in a `CSV` file. Similarly, in a tab-separated file `(.tsv extension)`, the data values in each line are separated by tabs. The data in a spreadsheet such as Excel, is often exported to `CSV` or `TSV` files. Consider a file `student.csv` that stores student details comprising the roll number, name, and marks obtained by the students in a course. The data in the `student.csv` file will appear as follows:

```
1,Pankaj,99.5
2,Vinay,98
3,Vaibhav,80
```

3.1.2 Binary Files

The data in a binary file is stored in machine-readable data objects. For example, in a text file the number–12345 will be stored as a sequence of six bytes, whereas in a binary file it may be stored as an integer object requiring 16, 32, or 64 bits, depending on the size of the integer object. The binary files are not in a human-readable form. Unlike text files, binary files do not require a comma, space, or end of line character.



Binary File: machine-readable file. No EOL needed.

Binary files are mainly used to store multimedia data like video, audio, and images.

3.2 Opening and Closing Files

We will carry out all file operations in a folder. You may set any folder of your choice as the current working folder. For example, you can make the folder **D:/CBSE_Python** as the current folder as follows:

```
>>> import os
    os.chdir('D:/CBSE_Python')
```

To read data from a file or write data into a file, it must be open for use. We also need to specify what operation we want to perform on the file. Finally, a file must be closed after the necessary operations on it have been carried out.

3.2.1 Opening Files

For opening a file, Python provides the built-in function **open()**. The **open()** takes the name of the file and the mode in which we need to access the file as an input. The syntax of the function is as follows:

```
f = open(fileName, accessMode)  # f is a user specified name
```



open(): to open a file

It is important to note that `fileName` is the name of the file by which the file is stored on a computer. So, `fileName` consistent with the syntax requirements of the operating system using which the file has been created. In contrast, `f` is a programmer specified name that will be used to refer to the file object throughout the program. As it is used in the development of program logic, it is also called logical file name. For example, a file containing student data may be stored on a hard disk by the name `student.txt`, but a Python program may use the name `student`. Recall that Python does not permit the use of `student.txt` as a valid identifier.

3.2.2 File Object/File Handle

The above call to **`open()`** function returns a file object **`(f)`**, also known as file handle. Access to a file is made using its handle, which enables us to perform different types of operations on the file such as writing into the file, reading from a file and appending data to the file.

3.2.3 File Access Mode

There are mainly three access modes to access the file: read **`(r)`**, write **`(w)`**, and append **`(a)`**. For reading an already existing file, read access **`(r)`** mode should be used as the second argument while invoking the function **`open()`**. For example, the following statement opens an existing file `Python.txt` in read mode:

```
>>> f = open('Python.txt','r')
```

In Python, read mode is the default access mode. So, we may also open the file `Python.txt` for reading as shown in the following statement:

```
>>> f = open('Python.txt')
```



Default mode for opening file: read

If an attempt is made to read a file that does not exist, the function fails to open the requested file, resulting in an error as shown below:

```
>>> f=open('File1.txt', 'r')
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    f=open('File1.txt', 'r')
FileNotFoundError: [Errno 2] No such file or directory: 'File1.txt'
```

To write the contents to a file, we use write mode **`(w)`** while opening the file. If the file being opened already exists in the folder, opening the file in write mode will destroy the existing data of the file. However, if the named file does not exist in the folder, a new file is created. For example, consider the following statement:

```
>>> f = open('news.txt', 'w')
```

Assuming that there is no file by the name *news.txt*, new file *news.txt* will be created. To write the new data at the end of an already existing or open file we use append. However, if the named file does not exist in the folder, a new file is created.

Sometimes, a program needs to read from a file as well as write into it. In such a case, we specify the access mode as **`r+`** or **`w+`**. Similarly, to read from a file as well as append new data to it, we use the access mode **`a+`**.



Modes for opening a file:
r: read mode
w: write mode
a: append mode to write at the end of the file.
r+, w+: both read and write mode
a+: both read and append mode

1. Which mode is used to write at the end of the file?
2. What is the use of **w+** mode?

3.2.4 Closing Files

Once the necessary operations on a file have been carried out, it should be closed using a call to the function **close()**. Invoking the function **close()** breaks the connection between file object and the file available on the hard disk. The following statement closes the file object **f**:

close(): to save and close a file

```
>>> f.close()
```

After closing a file, you cannot perform any operation like read, write, and append. However, having closed a file in the program, it may be opened again, possibly using a different access mode.

3.3 Reading and Writing Files

Python provides the function **read()** and **write()** for reading from and writing to files.

3.3.1 Reading From Files

A call to function **read()**, reads the entire contents of the file. For example, the contents of file `Python.txt` that has already been opened in read mode may be read using the following statement:

```
>>> f.read()
```

Specifically, suppose, file `Python.txt` contains the following text:

```
Python!
Python programming language is developed by Guido Van Rossum in 1991.
It is an interpreted high-level programming language.
```

It will be instructive to examine the execution of the following code:

```
>>> f = open('Python.txt','r')
>>> f.read()
    'Python!\nPython programming language is developed by Guido Van Rossum in 1991.\n
    It is an interpreted high-level programming language.\n'
>>> f.close()
```

Note that in the above example, the **read()** function returned the entire contents of the file `Python.txt`. Once the necessary operations on a file have been carried out, it should be closed. However, to read the first few characters from a file, we can specify the number of characters to be read as an argument to the function **read()** as shown in the following code:

```
>>> f = open(Python.txt', 'r')
>>> f.read(6)
    'Python'
>>> f.close()
```

read(): to read a file by retrieving its contents.

To read line by line from a file, **`readline()`** is used as shown in the following code:

```
>>> f = open('Python.txt', 'r')
>>> f.readline()
'Python!\n'
>>> f.readline()
'Python programming language is developed by Guido Van Rossum in 1991.\n'
>>> f.readline()
'It is an interpreted high-level programming language.\n'
>>> f.readline()
''
>>> f.readline()
''
>>> f.close()
```



Once control reaches the end of a file, further attempts to read from it return a null string.

Note that the first call to the **`readline()`** function returns the contents of the first line from the file. Subsequent calls to the **`readline()`** function return the contents of the following lines from the file. When all the lines have been read, subsequent read operations will yield an empty string as illustrated in the above code. Similar to **`readline()`**, the function **`readlines()`** returns the entire content of the file from the current position in the form of a list of lines as shown in the following code:

```
>>> f = open('D:/CBSE_Python/Python.txt','r')
>>> f.readline()
'Python!\n'
>>> f.readlines()
['Python programming language is developed by Guido Van Rossum in 1991.\n', 'It is
an interpreted high-level programming language.\n']
>>> f.close()
```

You may also iterate over the list of lines returned by **`readlines()`** function so as to show the content of the file, line by line, as shown in the following code:

```
>>> f = open('Python.txt','r')
>>> for line in f.readlines():
...     print(line)
...
...
Python!

Python programming language is developed by Guido Van Rossum in 1991.

It is an interpreted high-level programming language.

>>> f.close()
```

Note that Python has inserted a blank line after each line (except the last line) because the output of a **`print`** statement includes **`'\n'`**, but the strings in the list produced by the **`readlines()`** function already include **`'\n'`**. To avoid printing an extra line of output after each line, we can specify **`end=''`** as the keyword argument while invoking the **`print()`** function as shown in the following code:

```
>>> f = open('Python.txt','r')
>>> for line in f.readlines():
...     print(line, end='')
...
```

```
...
Python!
Python programming language is developed by Guido Van Rossum in 1991.
It is an interpreted high-level programming language.
>>> f.close()
```

Once a file is closed, it must be opened again to perform any operations on it, otherwise Python will yield an error as shown in the following code:

```
>>> f = open('Python.txt', 'r')
>>> f.readline()
'Python!\n'
>>> f.close()
>>> f.readline()
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    f.readline()
ValueError: I/O operation on closed file.
```

There is no need to close the file explicitly if a Python file is opened using the **`with`** clause as illustrated in the following code:

```
>>> with open('Python.txt','r') as f:
...     f.read()
...
...
'Python!\nPython programming language is developed by Guido Van Rossum in 1991.\nIt is
an interpreted high-level programming language.\n'
```



File opened using with clause need not be closed

Python files that are opened using the **`with`** clause are automatically closed when control exits the **`with`** clause. Now, we can reopen the file **`f`** and perform any operations on it.

3.3.2 Writing into Files

So far, we have learned how to read the contents from an already existing file. To write data in a file, we make use of the **`write()`** function as shown in the following code:

```
>>> f = open('Python.txt', 'w')
>>> f.write('Python!\nPython programming language is developed by Guido Van
Rossum in 1991.\nIt is an interpreted high-level programming language.\n')
132
>>> f.read()
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    f.read()
io.UnsupportedOperation: not readable
>>> f.close()
```



Opening a file in write mode will destroy existing data of the file
write(): to write contents to a file

In the preceding code, note that the **`write()`** function returns the number of bytes written to the file. Also, note that an attempt to read from the file **`f`** soon after writing 132 characters resulted in an error because the file was opened in write mode, so read operation on it was illegal. Hence, Python pointed out **`io.UnsupportedOperation: not readable`** error, indicating that **`io`** operation **`read`** is not supported on a file that has been opened in **`write`** mode.

Of course, once a file is opened in write mode, one can perform several write operations on it. Every subsequent write operation will write the new content at the end of the previously written content as shown in the following code:

```
>>> f = open('Python.txt', 'w')
>>> f.write('''Python!
... Python programming language is developed by Guido Van Rossum in 1991.\n''')
    78
>>> f.write('It is an interpreted high-level programming language.\n')
    54
>>> f.close()
>>> f=open('Python.txt', 'r')
>>> f.read()
    'Python!\nPython programming language is developed by Guido Van Rossum in 1991.\nIt
    is an interpreted high-level programming language.\n'
>>> f.close()
```



write(): returns the number of bytes written to a file
Attempt to read a file opened in write mode results in an error

As the **`readlines()`** function allows us to read several lines of input text, Python also allows us to write a sequence of lines to a file in one go using the **`writelines()`** function as shown in the following code. Unlike the **`write()`** function, **`writelines()`** function does not return the number of bytes written.

```
>>> f = open('Python.txt', 'w')
>>> f.writelines(['Python!\n', 'Python programming language is developed
    by Guido Van Rossum in 1991.\n', 'It is an interpreted high-level programming
    language.\n'])
>>> f.close()
```



writelines(): to write a sequence of lines to a file

As discussed earlier, having written some contents to a file (say, `Python.txt`), opening the file `Python.txt` again in **`write`** mode will destroy the existing data of the file. So, to append new data at the end of the file, the file should be opened in **`append`** mode as shown in the following code:

```
>>> f = open('Python.txt', 'r')
>>> f.read()
    'Python!\nPython programming language is developed by Guido Van Rossum in 1991.\n
    It is an interpreted high-level programming language.\n'
>>> f.close()
>>> f = open('Python.txt', 'a')
>>> f.write('Python makes coding easier!')
    27
>>> f.close()
>>> f = open('Python.txt', 'r')
>>> f.read()
    'Python!\nPython programming language is developed by Guido Van Rossum in 1991.\n
    It is an interpreted high-level programming language.\nPython makes coding easier!'
>>> f.close()
```

3.3.3 The flush ( ) Function

In general, the data written to a file is temporarily stored in a file buffer and transferred from buffer to file on disk only when the **`close()`** function is invoked. The **`flush()`** function can be used to forcefully write the content from Python's buffer to a file without waiting for the user to close the file. This makes the content in the buffer readily written to the file on the disk and available for use.

Also, it is recommended not to keep too much data in Python's buffer and flush the contents in a buffer on the disk on a regular basis to ensure that the data in the file is up to date. It will be instructive to examine the execution of the statements and the contents of file in the following code:

```
>>> f = open('Python.txt', 'w')                    #Python.txt File Content
>>> f.write('Python\n')                            #(Before Flush)
    7
>>> f.flush()                                      #(After Flush)
                                                   #Python
>>> f.write('Python programming language is        #(Before Flush)
    developed by Guido Van Rossum in 1991.\n')     #Python
    70
>>> f.flush()                                      #(After Flush)
                                                   #Python
                                                   #Python programming language is
                                                   #developed by Guido Van Rossum in 1991.
>>> f.close()
```



flush(): to forcefully write the content from Python's buffer to the file.

3.3.4 Removing Leading and Trailing Whitespaces in Lines of a Text File

Suppose we want to clean the contents of a text file (say, `poem.txt`) by removing the leading and trailing whitespaces from each line of the file. For this purpose, we will open a new file `(poem_formatted)` in write mode. Next, we will read from the input file line by line, and write the revised text to the output file. This is described in the Pseudocode 1:

```
01 Open the file poem.txt in read mode
02 Open the file poem_formatted.txt in write mode
03 For each line in poem.txt:
            Remove leading and trailing whitespaces from the line
            Write the updated line in file poem_formatted.txt
04 Close the files opened for reading or writing
```

Updated Lines without Whitespaces in Another File. The **`strip()`** function in Python allows us to remove all newlines, spaces, and tabs from the beginning and end of a string. In the code given below, we first begin by printing the contents of the file `poem.txt`.

```
>>> with open('poem.txt','r') as f:
        data = f.read()
        print(data)

    Twinkle, twinkle, little star,

    How I wonder what you are!
    Up above the world so high,

    Like a diamond in the sky.
```

Next, for every line in the file, we apply strip operation and store the updated line in the new file poemNew.txt:

```
>>> source = open('poem.txt','r')
>>> destination = open('poem_formatted.txt','w')
>>> for line in source.readlines():
>>>     newLine = line.strip()
>>>     destination.write(newLine)
    30
    0
    26
```

```
27
0
26
>>> source.close()
>>> destination.close()
```

Finally, let us read the poem from the modified file poem-formatted.txt.

```
>>> with open('poem-formatted.txt','r') as f:
        data = f.read()
        print(data)

Twinkle, twinkle, little star,How I wonder what you are!Up above the world
so high,Like a diamond in the sky.
```

3.3.5 Counting the Number of Characters, Lines, and Words in a File

Suppose you wish to display descriptive statistics of a file. Specifically, we wish to find the count of characters, lines, and words in a file. Consider the file `Python.txt` whose contents are shown as output of the following code snippet:

```
>>> f = open('Python.txt','r')
>>> data= f.read()
>>> print(data)
Python!
Python programming language is developed by Guido Van Rossum in 1991. It is an
interpreted high-level programming language.
Python makes coding easier!
>>> f.close()
```

To find the descriptive statistics of a file (say, Python.txt), we write a function **`descriptiveStat()`**. The function initializes the variable **`numChars`**, **`numWords`**, and **`numLines`** to zero. Next, the source file is read as a list of lines. The count function applied to this list yields the number of lines (numLines) in the file. For each line, we find the number of characters and words in the line and update the **`numChars`** and **`numWords`**. This is described in the Pseudocode 2: Finding the descriptive statistics of a text file.

```
01 Open the source file in read mode
02 Initialize numChars, numWords, numLines to 0
03 Read the source file as a list of lines (namely, listLines)
04 numLines <- count(lines)
05 For each line in listLines:
06 # update the numChars and numWords
07 numChars <- numChars + length(line)
08 words <- splitBySpace(line) # words in current line
09 numWords <- numWords + count(words)
10 Close the source file
```

The complete function **`descriptiveStat()`** is described in Program 3.1.

- The descriptiveStat() function opens the user-specified file in read mode (line 7).
- Next, the function initializes the variable **`numChars`**, **`numWords`**, and **`numLines`** to zero (line 8).
- In line 9, the function reads the entire content of the file in a list of **`lines(lines)`**.
- Line 10 finds the number of lines using built-in function **`len()`**.
- For every line in the file (line 11), we apply the function **`len()`** to count the number of characters and update the value of numChars in line 12.
- Next, the line is split into words using **`split()`** which returns a list of **`words(words)`**.

- In line 14, the word count of the current line is computed using the function **`len()`** on the list **`words`** and **`numWords`** is also updated.
- Finally, line 15 closes the file and lines 16, 17, and 18 display the number of characters, words, and lines in the file.

Program 3.1 Counting the number of characters, lines, and words in a file.

```
01 def descriptiveStat(fileName):
02     """
03     Objective: To count the number of characters, words, and lines in a file.
04     Input Parameter: fileName-string
05     Return Value: None
06     """
07     f1 = open(fileName,'r')
08     numChars, numWords, numLines = 0, 0, 0
09     lines = f1.readlines()
10     numLines = len(lines)
11     for line in lines:
12         numChars += len(line)
13         words = line.split(' ')
14         numWords += len(words)
15     f1.close()
16     print("Number of Characters:", numChars)
17     print("Number of Words:", numWords)
18     print("Number of Lines:", numLines)
19
20 ##### Main program segment starts here! #####
21 #Objective: To count the number of characters, words, and lines in a file.
22 file = input("Enter the name of the file: ")
23 descriptiveStat(file)
```

In Line 22, Program 3.1 takes the name of the file *(file)* as user input. In Line 23, invokes the function **`descriptiveStat()`** with the file name entered by the user.

On execution of Program 3.1, Python responds with the following output:

```
>>> Enter the name of the file: D:/CBSE_Python/Python.txt
    Number of Characters: 159
    Number of Words: 23
    Number of Lines: 4
```

3.3.6 Binary File Operations

So far, we have been dealing with text data while performing read/write operations on files. However, often we need to deal with objects such as lists and dictionaries. Such objects are written to a file in binary mode **`(b)`**. Python module **`pickle`** is used for reading and writing data objects to a binary file. Pickling is the process of transforming a Python object into a byte stream for writing the object to a file. While reading a binary file, reverse process of unpickling is applied to obtain the data back in the object form. Access mode **`'b'`** indicates that the file is opened in binary mode. For example, access modes in binary files are of the form **`'rb'`**, **`'rb+'`**, **`'wb'`**, **`'wb+'`**, **`'ab'`**, and **`'ab+'`**. Usually, the binary files are stored with the extension **`.dat`**.



pickle module: comprises functions for reading and writing data objects to a binary file

The methods **`load()`** and **`dump()`** of the **`pickle`** module are used for reading and writing the objects from and to the binary files respectively. The **`dump()`** method takes the file object as an input argument and writes it to the binary file. The **`load()`** method reads a an object from a binary file.

Syntax of **load()** and **dump()** methods of the pickle module are mentioned as follows:

```
pickle.dump(sourceObject, fileObject)
pickle.load(fileObject)
```



dump(): Writes an object to a binary file

One should be careful about unpickling data in a file unless the file has been received from an authentic source as a pickled byte stream may be the source of viruses, worms, etc.



load(): Reads stream of bytes from a binary file

Consider a list of lists (`StudentDetails`) that stores student details. Each component of this list is a list comprising roll number, name, and percentage of a student. We construct a list `StudentDetails` comprising details of three students. Next, we open a file for writing in binary mode and dump the list corresponding to each of three students into the binary file `Student`, one by one. Subsequently, we open the file for reading (in the binary mode), read the details of each student as a list, reconstruct the list comprising all the three students and print it, as shown in the following code:

```
>>> import pickle
>>> StudentDetails = [[1, 'Vijay', 99.5], [2, 'Vidya', 98], [3, 'Sakeena', 80]]
>>> f = open('Student.dat', 'wb')
>>> for stud in StudentDetails:
...     pickle.dump(stud, f)
...
...
>>> f.close()
>>> f = open('Student.dat', 'rb')
>>> details1 = pickle.load(f)
>>> details2 = pickle.load(f)
>>> details3 = pickle.load(f)
>>> details = [details1, details2, details3]
>>> print(details)
[[1, 'Vijay', 99.5], [2, 'Vidya', 98], [3, 'Sakeena', 80]]
>>> f.close()
```

Note that once the last object is read from the binary file , a further attempt to read data from the file will raise an **EOFError** as shown in the following code:

```
>>> import pickle
>>> StudentDetails = [[1, 'Vijay', 99.5], [2, 'Vidya', 98], [3, 'Sakeena', 80]]
>>> f = open('Student.dat', 'wb')
>>> for stud in StudentDetails:
...     pickle.dump(stud, f)
...
...
>>> f.close()
>>> f = open('Student.dat', 'rb')
>>> details1 = pickle.load(f)
>>> details2 = pickle.load(f)
>>> details3 = pickle.load(f)
>>> details4 = pickle.load(f)
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    details4 = pickle.load(f)
EOFError: Ran out of input
```

Later in this chapter, we will learn about what we mean by exceptions and how to handle them.

CT 01

Write a Python program that accepts the student details from a user and writes the details in a file of list objects. Subsequently, the program should read the binary data from the binary file just created and print the list corresponding to each student one by one.

3.4 File Object Attributes

Recall that a call to the function **`open()`** returns a file object *f*, also known as file handle. There are several data members and methods associated with the file object that can be used to perform operations on the file or to show the details of the file. The **`dir()`** function returns a complete list of these objects. For example, data members **`name`** and **`mode`** can be used to retrieve the name of the file and the mode of access to the file, respectively. The data member **`closed`** returns **`True`** if the file is closed and **`False`** otherwise. For the sake of completeness, in the following code, we provide a complete list of the objects associated with the file *f* that has been opened in read mode.

```
>>> f=open('D:/CBSE_Python/Python.txt', 'r')
>>> f.read()
'Python\nPython programming language is developed by Guido Van Rossum in 1991.\n'
>>> f.close()
>>> dir(f)
['_CHUNK_SIZE', '__class__', '__del__', '__delattr__', '__dict__', '__dir__', '__
doc__', '__enter__', '__eq__', '__exit__', '__format__', '__ge__', '__
getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__',
'__le__', '__lt__', '__ne__', '__new__', '__next__', '__reduce__', '__reduce_ex__',
'__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '_
checkClosed', '_checkReadable', '_checkSeekable', '_checkWritable', '_finalizing',
'buffer', 'close', 'closed', 'detach', 'encoding', 'errors', 'fileno', 'flush',
'isatty', 'line_buffering', 'mode', 'name', 'newlines', 'read', 'readable',
'readline', 'readlines', 'reconFig', 'seek', 'seekable', 'tell', 'truncate',
'writable', 'write', 'write_through', 'writelines']
>>> f.name
'D:/CBSE_Python/Python.txt'
>>> f.mode
'r'
>>> f.closed
True
```

Data members of file object
name: retrieves name of file
mode: retrives access mode of file
closed: returns True if file is closed, and False otherwise

3.5 Standard Input/Output Devices as Files

In Python, default reading takes place from the keyboard and the outputs are displayed on the console. Standard input **`stdin`** is a file object that corresponds to the input stream of characters. Similarly, standard output **`stdout`** is a file object that receives outputs. The error messages are received in the file object **`stderr`**.

3.5.1 Absolute and Relative Paths

Consider the following Python statement that opens the file `'Python.txt'` in read mode:

```
>>> f = open('Python.txt','r')
```

Relative path: location of file w.r.t current folder
Absolute path: complete path from root to the file

Here, the filename `'Python.txt'` refers to the file in the current working directory. Alternatively, we may use the path `'./Python.txt'` for referring to file `Python.txt` in current working directory as follows:

```
>>> f = open('./Python.txt','r')
```

The path `'./Python.txt'` is a relative path as it specifies file location with respect to the current folder. To refer to a file in an arbitrary folder, we need to specify the complete path from root to the file (also called absolute path) as follows:

```
>>> f = open('D:\CBSE_Python\Python.txt','r')
```

3.6 Random Access Using seek() and tell()

Sometimes, we seek to read or write the contents of a file from a particular position (also known as offset from the beginning). By default, when the file is opened, the file object refers to the offset 0 i.e., the beginning of the file. Using the method **`seek()`**, the file object can point to a particular position in the file. Suppose we wish to convert the contents of a text file to title case i.e. we need to capitalize the first letter of each word. Since we need to read and update the contents of the same file, we open the file in **`'r+'`** mode. Using the **`read()`** function, we first read the entire content of the file into **`data`**. However, this brings the file object to the end of the file. So, after transforming the text in the file to title case using the function **`title ()`**, we need to execute a seek operation to bring the file pointer to location 0 in the file, i.e., beginning of the file. The code given below illustrates these operations.

```
>>> f1 = open('D:/CBSE_Python/Python.txt','r+')
>>> data = f1.read()
>>> data
'Python!\nPython programming language is developed by Guido Van Rossum in 1991.\n
It is an interpreted high-level programming language.\nPython makes coding easier!'
>>> f1.seek(0)
0
>>> f1.write(data.title())
159
>>> f1.flush()
>>> f1.seek(0)
0
>>> f1.read()
'Python!\nPython Programming Language Is Developed By Guido Van Rossum In 1991.\n
It Is An Interpreted High-Level Programming Language.\nPython Makes Coding Easier!'
>>> f1.close()
```

seek(): to point file object to a particular position

The function **`tell()`** returns the current position of the file object as illustrated in the following code:

```
>>> f = open('D:/CBSE_Python/Python.txt','r+')
>>> f.tell()
0
>>> f.read()
'Python!\nPython programming language is developed by Guido Van Rossum in 1991.\n
It is an interpreted high-level programming language.\nPython makes coding easier!'
>>> f.seek(0)
0
>>> f.read(8)
```

```
        'Python!\n'
>>> f.tell()
        9
>>> f.read(18)
        'Python programming'
>>> f.tell()
        27
>>> f1.close()
```

tell(): returns current position of file object

Note that although Python reads 8 characters **`'Python!\n', f.tell()`** indicates that the file object is at position 9 because Windows stores the end-of-line as a sequence of two characters: line feed, followed by the carriage return.

3.7 Handling CSV Files in Python

CSV (Comma Separated Values) is a comma-delimited file. CSV is the most popular format for import and export operations on spreadsheets and databases. Often, we deal with tabular data comprising several records. In a table, each record has a fixed structure. For example, consider data about employees of an organization. Suppose, we have information about an employee's ID, first name, last name, date of birth, salary, and department (see Fig 3.1). Often such data is stored and analyzed using spreadsheets or databases. Python provides a **`csv`** module which includes classes for reading and writing tabular data to and from CSV files.

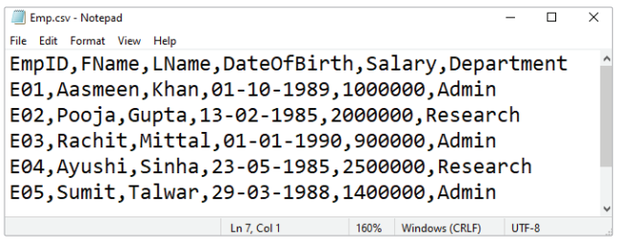


Fig 3.1: The text in file Emp.csv

CSV: comma-delimited file with comma-separated values
CSV: module for reading from csv file

3.7.1 Writing to a CSV File

To facilitate writing data into csv files, Python module **`csv`** provides the writer class that takes the file object as an input argument and returns the **`writer`** object that is used for writing to a CSV file. The writer class enables us to write one row at a time using **`writerow()`**. Multiple rows can be written using **`writerows()`**.

csv.writer class: facilitates writing to a csv file Methods—
1. writerow(): write one row at a time
2. writewrows(): write multiple rows at a time

Let us now write a program to create a file named employees.csv.

- In this file, we will store details about the employees of an organization. The employee details include ID, first name, last name, date of birth, salary, and department (see Program 3.2).
- In line 2, we import the **`csv`** module.

- In line 3, we create the list columns of column names (also called field names) for the CSV file.
- In lines 4–10, we create the list **empDetails** of employee details to be written to the csv file.
- In line 11, the open function opens the file employees.csv in write mode and returns the file object **employee**. To avoid writing a blank row after each record, keyword argument newline='' is added in the call to open() function.
- In line 12, we create the object writeEmployee that enables us to write to the CSV file.
- In line 13, we write to the csv file, the header information from the list columns.
- In line 14, the details corresponding to all the employees are written to the CSV file from the list **empDetails**.
- Finally, line 15 closes the file **employee**.

Program 3.2 Writing to a CSV File

```
01 #Objective: To write the content onto a csv file.
02 import csv
03 columns = ['EmpID', 'FName', 'LName', 'DateOfBirth', 'Salary', 'Department']
04 empDetails = [ ['E01', 'Aasmeen', 'Khan', '01-10-1989', 1000000, 'Admin'],
05      ['E02', 'Pooja', 'Gupta', '13-02-1985', 2000000, 'Research'],
06      ['E03', 'Rachit', 'Mittal', '01-01-1990', 900000, 'Admin'],
07      ['E04', 'Ayushi', 'Sinha', '23-05-1985', 2500000, 'Research'],
08      ['E05', 'Sumit', 'Talwar', '29-03-1988', 1400000, 'Admin'],
09      ['E06', 'Rahul', 'Roy', '22-05-1986', 1200000, 'Admin'],
10      ['E07', 'Diksha', 'Dewan', '12-12-1979', 3200000, 'Research']]
11 employee = open('D:\CBSE_Python\employees.csv', 'w', newline='')
12 writeEmployee = csv.writer(employee)
13 writeEmployee.writerow(columns)
14 writeEmployee.writerows(empDetails)
15 employee.close()
```

3.7.2 Reading From a CSV File

To facilitate reading the data from already existing CSV files, Python module **csv** provides the **reader** class that takes file object as an input argument and returns a reader object that is used for reading data from the CSV file. In this section, we will write a program (see Program 3.3) that reads and displays data from a CSV file `(employees.csv)` containing details about the employees of an organization.



csv.reader class: facilitates reading from a csv file

- In Line 2, we import the **csv** module.
- In Line 3, the **open()** function opens the file employees.csv in read mode and returns the file object employee.
- In Line 4, we create the object readEmployee that enables us to read from the CSV file.
- In Line 5, we declare a variable **lineNum** to keep track of line number so that during printing, we may treat field names (the first row) differently as compared to records (remaining rows).
- In Line 6, For every row in the CSV file, if it is the row with column header (line 7), tab-separated field names are displayed followed by a line comprising stars.
- Thereafter, for the rows comprising records (**else** clause in line 11), tab-separated values of fields are printed.
- Finally, Line 16 closes the file opened for reading using the **employee**.

Program 3.3 Reading from the csv file

```
#Objective: To read and display employee details from CSV file employees.csv
import csv
employee = open('D:\CBSE_Python\employees.csv', 'r')
readEmployee = csv.reader(employee)
lineNum = 0
for row in readEmployee:
    if lineNum == 0:
        for value in row:
            print(value, end="\t")
        print("\n"+"*"*60)
    else:
        print()
        for value in row:
            print(value, end="\t")
    lineNum+=1
employee.close()
```

On execution of the program in Program 3.3, we get the following output:

```
EmpID     FName       LName       DateOfBirth     Salary          Department
************************************************************
E01       Aasmeen     Khan        01-10-1989      1000000         Admin
E02       Pooja       Gupta       13-02-1985      2000000         Research
E03       Rachit      Mittal      01-01-1990      900000          Admin
E04       Ayushi      Sinha       23-05-1985      2500000         Research
E05       Sumit       Talwar      29-03-1988      1400000         Admin
E06       Rahul       Roy         22-05-1986      1200000         Admin
E07       Diksha      Dewan       12-12-1979      3200000         Research
```

Alternatively, the contents of `employees.csv` can also be read using the **`next()`** method (Program 3.4) which reads and returns the current row and also positions the iterator to next line in sequence.

Program 3.4 Reading from the csv file

```
import csv
#Objective: To read the content from a csv file.
employee = open('D:\CBSE_Python\employees.csv', 'r')
readEmployee = csv.reader(employee)
lineNum=0
while True:
    row = next(readEmployee)
    if lineNum == 0:
        for value in row:
            print(value, end="\t")
        print("\n"+"*"*60)
    else:
        print()
        for value in row:
            print(value, end="\t")
    lineNum+=1
employee.close()
```



next(): iterator that reads and returns the current row and positions the iterator to next line

On execution of the program in Program 3.4, Python responds with the following output:

```
EmpID     FName       LName       DateOfBirth      Salary          Department
*************************************************************************
E01       Aasmeen     Khan        01-10-1989       1000000         Admin
E02       Pooja       Gupta       13-02-1985       2000000         Research
E03       Rachit      Mittal      01-01-1990       900000          Admin
E04       Ayushi      Sinha       23-05-1985       2500000         Research
E05       Sumit       Talwar      29-03-1988       1400000         Admin
E06       Rahul       Roy         22-05-1986       1200000         Admin
E07       Diksha      Dewan       12-12-1979       3200000         Research
Traceback (most recent call last):
  File "D:/CBSE_Python/employee2.py", line 23, in <module>
    readCSV('D:\CBSE_Python\emp1.csv')
  File "D:/CBSE_Python/employee2.py", line 12, in readCSV
    row = next(readerMarks)
StopIteration
```

Note that once the entire content of the file has been read, the program terminates abruptly as the system raises the **`StopIteration`** exception. This is handled later.

3.7.3 Reading and Writing a Dictionary to and from a CSV File

Python also allows us to read and write comma-separated files using dictionary objects. Each dictionary-based record comprises several key-value pairs. Similar to **`csv.reader`** and **`csv.writer`** classes, that facilitate reading from and writing lists to CSV files, Python module **`csv`** provides us **`csv.DictReader`** and **`csv.DictWriter`** classes for reading and writing dictionaries to CSV files.

- **`csv.Dictwriter`** class takes the file handler and a list of column names as an input arguments, and returns a writer object that is used for writing to a CSV file. The **`csv.Dictwriter`** class enables us to write one row at a time using the method **`writerow()`**. Multiple rows can be written using the method **`writerows()`**.



csv.DictWriter class: facilitates writing dictionaries to CSV file

- **`csv.Dictreader`** class takes file handler as an input argument and returns a reader object that is used for reading data from the CSV file.



csv.DictReader: facilitates reading dictionaries from a CSV file

- In Program 3.5, we will store details about the employees of an organization in the dictionary based records (**`empDetails`** in line 3).
- The employee details include ID, first name, last name, date of birth, salary, and department.
- The program writes these records to the file **`employees.csv`** (lines 12–17).
- Note that while writing dictionary-based records using methods of **`csv.DictWriter`** class, header row is written to the file using **`writeheader()`**.
- Subsequently, the data is read and displayed from a CSV file **`(employees.csv)`** containing details about the employees (lines 20–33).

Program 3.5 Reading and writing dictionary based records to and from the csv file.

```
#Objective: To write the content onto a csv file.
import csv
empDetails = \
[{'EmpID':'E01', 'FName':'Aasmeen', 'LName':'Khan', 'DateOfBirth': '01-10-1989',
'Salary':1000000, 'Department':'Admin'},
{'EmpID':'E02', 'FName':'Pooja', 'LName':'Gupta', 'DateOfBirth': '13-02-1985',
'Salary':2000000, 'Department':'Research'},
{'EmpID':'E03', 'FName':'Rachit', 'LName':'Mittal', 'DateOfBirth': '01-01-1990',
'Salary':900000, 'Department':'Admin'},
{'EmpID':'E04', 'FName':'Ayushi', 'LName':'Sinha', 'DateOfBirth': '23-05-1985',
'Salary':2500000, 'Department':'Research'},
{'EmpID':'E05', 'FName':'Sumit', 'LName':'Talwar', 'DateOfBirth': '29-03-1988',
'Salary':1400000, 'Department':'Admin'},
{'EmpID':'E06', 'FName':'Rahul', 'LName':'Roy', 'DateOfBirth': '22-05-1986',
'Salary':1200000, 'Department':'Admin'},
{'EmpID':'E07', 'FName':'Diksha', 'LName':'Dewan', 'DateOfBirth': '12-12-1979',
 'Salary':3200000, 'Department':'Research'}]

employee = open('D:\CBSE_Python\employees.csv', 'w', newline='')
columns = list(empDetails[0].keys())
writeEmployee = csv.DictWriter(employee, fieldnames=columns)
writeEmployee.writeheader()
writeEmployee.writerows(empDetails)
employee.close()

#Objective: To read and display employee details from CSV file employees.csv
employee = open('D:\CBSE_Python\employees.csv', 'r')
readEmployee = csv.reader(employee)
lineNum = 0
for row in readEmployee:
    if lineNum == 0:
        for value in row:
            print(value, end="\t")
        print("\n"+"*"*60)
    else:
        print()
        for value in row:
            print(value, end="\t")
    lineNum+=1
employee .close()
```

On execution of the program in Program 3.5, Python responds with the following output:

```
EmpID     FName       LName       DateOfBirth      Salary          Department
************************************************************
E01       Aasmeen     Khan        01-10-1989       1000000         Admin
E02       Pooja       Gupta       13-02-1985       2000000         Research
E03       Rachit      Mittal      01-01-1990       900000          Admin
E04       Ayushi      Sinha       23-05-1985       2500000         Research
E05       Sumit       Talwar      29-03-1988       1400000         Admin
E06       Rahul       Roy         22-05-1986       1200000         Admin
E07       Diksha      Dewan       12-12-1979       3200000         Research
```

3.8 File Processing Applications

3.8.1 Moderation of Examination Results

In this section, we will write a program to moderate marks obtained by the students in an examination. We have been provided a comma-separated data file `examResults.csv`. The file comprises the following information about the students: roll number, name, and marks. The information for each student appears on a different line. The following is an example of a CSV file:

```
RollNum,Name,Marks
1201,Shashank Sen,92
1202,Ridhi Dogra,74
1203,Sumit Jain,79
1204,Aaradhya Pandey,55
1205,Mehul Kapoor,66
```

Note that the first line of the file comprises the names of the columns (also called fields or attributes) of the CSV file. To compute the moderated marks, the marks of each student are moderated upwards by a certain number, called **`moderation`**. To perform the moderation, we take **`moderation`** as an input from the user. After moderation, the moderated marks are written to **`another`** file, namely, `moderatedResults.csv`. We follow Pseudocode 3 for producing file `moderatedResults.csv`:

```
01 Open the file examResults.csv in read mode
02 Open the file moderatedResults.csv in write mode
03 For each record in examResults.csv:
   i.   Calculate moderated marks
   ii.  Write the student's details with the moderated marks
        in the file moderatedResults.csv.
04 Close the files opened for reading or writing.
```

To ensure that the program runs smoothly and terminates with a suitable message if an IO error is encountered, Python provides a **`try-except`** mechanism. Exceptions are run-time errors encountered by Python while executing a program. For example, an attempt to open a file which does not exist results in an **`IOError`** exception. Similarly, an attempt to write to a file which is opened in read mode (and vice-versa) results in an **`IOError`** exception. We will use **`try-except`** blocks to handle the exceptions. A **`try`** block contains statements that may lead to an exception, the **`except`** block specifies what action should be performed if an exception is raised. Typically, an **`except`** block includes a sequence of exceptions and the corresponding action to be performed for each exception.



exceptions: run-time errors
try block: statements that may lead to an exception
except block: action to be performed if exception is raised

The Program 3.6 is intended to moderate the examination results of the students.

- The **`main`** function (lines 39–43) takes as input, the name of the file containing exam marks (source file) and name of the output file for moderated marks (lines 40–41).
- Next, it takes the amount of **`moderation`** for computing updated marks as an input from the user (line 42).
- Finally, it invokes the moderate() (line 43) that accepts as input the names of the source file, output file, and moderation to compute the moderated result for each student and write the moderated result to the output file.

- In Lines 15–16,The function moderate() opens source file and destination file, being referred to by the file handlers **studentIn** and studentOut respectively.
- In Line 14, As these lines may generate **IOError** exceptions, they are enclosed in a try block which is followed by an **except** block in lines 17–18 which handles the exceptions.
- In Line 20, creates a **reader** object **(studentIn)**.
- In Line 21, creates a **writer** object **(studentOut)**.
- In Line 25, The column header row **(lineNum 0)** is written as it is to the destination file using the **writerow()** method.
- In Lines 27–30, For other rows, marks in a row (record) are moderated.
- In Line 31,the updated record is written to the destination file.

Program 3.6 Program to compute moderated marks (moderatedMarks.py)

```
01 import csv
02 def moderate(file1, file2, moderation):
03     """
04     Objective: To moderate exam marks
05     Input Parameter:
06         file1-string : input file comprising original marks
07         file2-string : output file for storing moderated
08                        marks
09         moderation-numeric : moderation marks
10     Return Value: None
11     """
12     #Approach: Add moderation to marks, but do not exceed maxMarks
13     maxMarks = 100
14     try:
15         studentIn = open(file1, 'r')
16         studentOut = open(file2, 'w', newline='')
17     except IOError:
18         print("Error in opening File")
19     try:
20         readerMarks=csv.reader(studentIn)
21         writerMarks=csv.writer(studentOut)
22         lineNum=0
23         for row in readerMarks:
24             if lineNum == 0:
25                 writerMarks.writerow(row)# copy header information
26             else:  # moderate marks available in row[2]
27                 updatedMarks = row[2] + moderation
28                 if updatedMarks > maxMarks:
29                     updatedMarks = maxMarks
30                 row[2]= updatedMarks
31                 writerMarks.writerow(row)
32             lineNum += 1
33     except IOError:
34         print("Error in Reading/Writing the file")
35     studentIn.close()
36     studentOut.close()
37
38 ##### Main program segment starts here! #####
39 #Objective: To moderate exam marks
40 file1 = input('Enter file name for comprising exam results:')
41 file2 = input('Enter file name for storing moderated marks:')
42 moderation = int(input('Enter marks for moderation:'))
43 moderate(file1, file2, moderation)
```

On execution of Program 3.6 with `examResults.csv` as the file containing examination data name and 10 as the moderation, the system outputs the contents of the file `moderatedResults.csv` as follows:

```
RollNum,Name,Marks
1201,Shashank Sen,100
1202,Ridhi Dogra,84.0
1203,Sumit Jain,89.0
1204,Aaradhya Pandey,65.0
1205,Mehul Kapoor,76.0
```

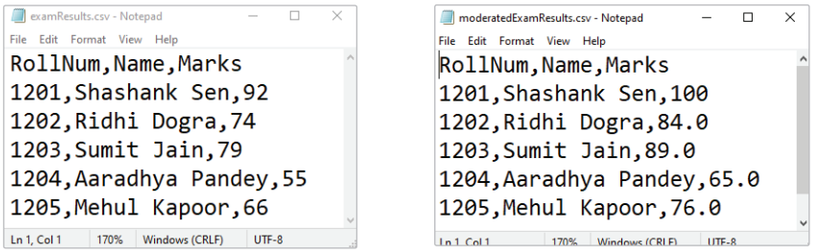


Fig 3.2: Showing the files examResult.csv and *moderatedResults.csv*

Consider a scenario where the exam results i.e. roll number, name, and marks are to be taken as an input from the user in an interactive manner. Also, the records are to be stored in the form of a dictionary in a binary file `(students.dat)`.

- This is achieved using the function **`writeStudRecords()`** defined (see line 21–45 in Program 3.7).
- To display the records of students entered by the user, we define the function **`readStudRecords`** (lines 3–19).
- Note that the method pickle.load (line 14) will yield an **`EOFError`** when an attempt is made to read beyond the file contents. So, a try-except block is used to catch and handle the exception by closing the file opened for reading. Subsequently, the file Students.dat, the file may be opened in append mode.
- In Lines 21–45,using the function **`writeStudRecords()`** defined in Program 3.7. To display the records of students entered by the user.
- In Lines 03–19,we define the function **`readStudRecords`**.
- Note that the method pickle.load() in line 14 will yield an **`EOFError`** when an attempt is made to read beyond the file contents. So, a try-except block is used to catch and handle the exception by closing the file opened for reading. Subsequently, the file Students.dat, the file may be opened in append mode.

Program 3.7 Program to compute moderated marks

```
01 import pickle
02
03 def readStudRecords(fileName):
04     """
05     Objective: To take student's data as an input from the user
06                 and write to a file
07     Input Parameter:
08         fileName-string: file name for storing students' records
09     Return Value: None
10     """
11     try:
12         studFile = open(fileName, 'rb')
13         while True:
14             student = pickle.load(studFile)
15             print(student)
16     except EOFError:
17         studFile.close()
```

```
    except IOError:
        print("Error in reading/writing file")

def writeStudRecords(fileName):
    """
    Objective: To take student's data as an input from the user
                and write to a file
    Input Parameter:
        fileName-string: file name for storing students' records
    Return Value: None
    """
    try:
        studFile = open(fileName, 'wb')
        student = {}
        choice = "y"
        while choice=="y" or choice=="Y":
            print("Enter Student's Details:")
            rollNum = input("Roll Number: ")
            name = input("Name: ")
            marks = int(input("Marks: "))
            student['RollNum'] = rollNum
            student['Name'] = name
            student['Marks'] = marks
            pickle.dump(student, studFile)
            choice = input("Do you want to enter more records: (y/n)")
        studFile.close()
    except IOError:
        print("Error in reading/writing file")

##### Main program segment starts here! #####
#Objective: To read and write student records
fileName = input('Enter file name for storing student records:')
print("Writing Records")
writeStudRecords(fileName)
print("\nReadingRecords")
readStudRecords(fileName)
```

On execution of Program 3.7 with **`students.dat`** as the file name, the system outputs the following:

```
>>> Enter file name for storing student records:students.dat
    Writing Records
>>> Enter Student's Details:
    Roll Number: 1
    Name: Raman
    Marks: 95
    Do you want to enter more records: (y/n)y
>>> Enter Student's Details:
    Roll Number: 2
    Name: Sriram
    Marks: 38
    Do you want to enter more records: (y/n)y
>>> Enter Student's Details:
    Roll Number: 3
    Name: Fatema
    Marks: 28
    Do you want to enter more records: (y/n)y
>>> Enter Student's Details:
    Roll Number: 4
    Name: Rajan
    Marks: 85
```

```
Do you want to enter more records: (y/n)n
ReadingRecords
{'RollNum': '1', 'Name': 'Raman', 'Marks': 95}
{'RollNum': '2', 'Name': 'Sriram', 'Marks': 38}
{'RollNum': '3', 'Name': 'Fatema', 'Marks': 28}
{'RollNum': '4', 'Name': 'Rajan', 'Marks': 85}
```

In this section, we will develop two programs:

1. Given a roll number the program searches the file for a record with the given roll number.
2. Update student records.

3.8.2 Searching a Given Roll number in a File of Student Records

We wish to display the details of a student having a given roll number.

- In Lines 3–25, we define the function **`searchRecord()`** in Program 3.8.
- In Line 20,The search procedure compares the roll number in every record with the roll number to be searched **`(rollNum)`**.
- If a student's record with the given **`rollNum`** is found in the file, it is returned in Line 32, and Lines 33–36 will be executed.
- Otherwise, the function displays **`Student Record Not Found`** in Line 38.

Program 3.8 Program to compute moderated marks

```
01 import pickle
02
03 def searchRecord(fileName, rollNum):
04     """
05     Objective: To take student's data as an input from the user
06                and write to a file
07     Input Parameter:
08         fileName-string: file name for storing students' records
09     Return Value:
10                student record, if rollNum search is successful
11                None: if rollNum search fails
12     """
13
14     #Approach: sequential search is carried out
15
16     try:
17         studFile = open(fileName, 'rb')
18         while True:
19             student = pickle.load(studFile)
20             if student['RollNum'] == rollNum:
21                 return student
22     except EOFError:
23         studFile.close()
24     except IOError:
25         print("Error in reading/writing file")
26
27 ##### Main program segment starts here! #####
28 #Objective: To search for a student record
29 fileName = input('Enter file name:')
30 print("\nSearching Records")
31 rollNum = input('Enter the roll number of student to be searched: ')
32 found = searchRecord(fileName, rollNum)
33 if found:
34      print("Student Record Found")
```

```
35      print("Details are as follows:")
36      print(found)
37  else:
38      print("Student Record Not Found")
```

On execution of Program 3.8 with `students.dat` as the file name, the system outputs the following:

Execution 1:

```
>>> Enter file name:students.dat

    Searching Records
>>> Enter the roll number of student to be searched: 3
    Student Record Found
    Details are as follows:
    {'RollNum': '3', 'Name': 'Fatema', 'Marks': 28}
```

Execution 2:

```
>>> Enter file name:students.dat

    Searching Records
>>> Enter the roll number of student to be searched: 5
    Student Record Not Found
```

3.8.3 Updating Student Records

We wish to update the marks of a selected set of students. Considering pass marks to be 40, you want to update the marks to 40 if a student has scored marks between 30 and 39.

- The **`function updateRecord`** (Program 3.9) achieves this by searching and thereafter, updating the record, if found. The updated record will replace the previous record in the file.
- If the desired record is found, we save its **`position`** using the **`tell`** method (line 15).
- The modified record is written back to the same location by positioning the file handler at **`position`** using **`seek`** method (line 20) and invoking the method **`dump`** (line 21).

Program 3.9 Program to compute moderated marks

```
01  import pickle
02
03  def updateRecord(fileName):
04      """
05      Objective: To update student's data update the marks to 40
06                 if a student has scored marks between 30 and 39.
07      Input Parameter:
08          fileName-string: file name for storing students' records
09      Return Value: None
10      """
11      flag = False #Record not found
12      try:
13          studFile = open(fileName, 'rb+')
14          while True:
15              position = studFile.tell()
16              student = pickle.load(studFile)
17              if float(student['Marks']) >=30 and float(student['Marks']) <40:
18                  flag = True #Record found
19                  student['Marks'] = 40
20                  studFile.seek(position)
21                  pickle.dump(student, studFile)
22      except EOFError:
23          studFile.close()
```

```
    except IOError:
        print("Error in reading/writing file")
    if flag == False:
        print("No Record Found!")
    else:
        print("All records updated successfully!")

def readStudRecords(fileName):
    """
    Objective: To read students data
    Input Parameter:
        fileName-string: file name for storing students' records
    Return Value: None
    """
    try:
        studFile = open(fileName, 'rb')
        while True:
            student = pickle.load(studFile)
            print(student)
    except EOFError:
        studFile.close()
    except IOError:
        print("Error in reading/writing file")
##### Main program segment starts here! #####
#Objective: To update student records
fileName = input('Enter file name: ')
print("\nUpdating Records")
updateRecord(fileName)
print("\nReadingRecords")
readStudRecords(fileName)
```

On execution of Program 3.9 with `Students.dat` as the file name, the system outputs the following:

```
>>> Enter file name: students.dat
    Updating Records
    All records updated successfully!
    ReadingRecords
    {'RollNum': '1', 'Name': 'Raman', 'Marks': 95}
    {'RollNum': '2', 'Name': 'Sriram', 'Marks': 40}
    {'RollNum': '3', 'Name': 'Fatema', 'Marks': 28}
    {'RollNum': '4', 'Name': 'Rajan', 'Marks': 85}
```

3.8.4 Computing Monthly Salary

In this example, we wish to compute the monthly salary of the employees of a company, based on hourly rate and the number of hours worked. The first file `(empMain.csv)` comprises comma-separated data about the employees: ID, name, and hourly rate. Assume that the file is sorted with respect to the employee's ID.

The file include data about five employees:

```
empID,name,hourlyRate
9001,Rohanpreet Kumar,40
9002,Ramit Sen,34
9003,Sidhi Gupta,35
9004,Vinay Sen,28
9005,Raman Singh,30
```

The second file `(empHoursWorked.csv)` includes the following comma separated information about each employee: ID and number of hours worked. Assume that the file is sorted w.r.t. employee's ID.

The `empHoursWorked.csv` file includes the following data:

```
tEmpId,numHours
9001,250
9002,225
9003,267
9004,248
9005,215
```

To calculate the monthly pay of all employees, we must read the files `empMain.csv` and `empHoursWorked.csv` and generate a third file `empMonthlySalary.csv` that contains the monthly salaries of all the employees. To achieve this task, we follow the Pseudocode 4:

```
01 Open file empMain.csv in read mode
02 Open the file empHoursWorked.csv in read mode
03 Open the file empMonthlySalary.csv in write mode
04 For each file1Record in empMain.csv:
   Read next record (file2Record) from empHoursWorked.csv.
   Calculate monthly salary by multiplying hourlyRate with numHours.
   Write empID along with computed total monthly salary in the monthlySalary.csv file
05 Close the files opened for reading or writing.
```

The Program 3.10 computes monthly salary.

- The **`main`** function takes as input the names of the two source files as well as the output file that stores the computed monthly salary (lines 39–40).
- In line 42, it invokes the function **`calculateMonthlySalary`** to compute the salaries for the employees. The function **`calculateMonthlySalary`** opens the input data files and the output file (lines 12–14). The input files are referred to by file handles **`employeeIn1`**, **`employeeIn2`**. The output file is referred to by file handle **`employeeOut`**.
- Note that the lines 12–14 are enclosed in a **`try`** block as the open file operations may generate **`IOError`** exceptions. The **`except`** block following the **`try`** block ensures smooth termination of the program in case an exception is raised.
- Lines 18 and 19 create two reader objects for file handlers employeeIn1 and employeeIn2.
- Line 20 creates a writer object for the file handler **`employeeOut`**. Line 22 writes the header row to the destination file.
- Line 23 and 24 cause skipping of the header rows of source files. For every record being read using the reader object of source file 1, the corresponding record is read from the source file 2, monthly salary is computed using **`hourlyRate`** and **`numHours`**.
- Finally employee ID and monthly salary are written to the destination file along with employee ID in line 30.

Program 3.10 Program to compute total monthly salary of employees (monthlySalary.py)

```
01 import csv
02 def calculateMonthlySalary(file1, file2, file3):
03     """
04     Objective: To compute monthly salary
05     Input Parameter:
06         file1-string : input file comprising hourly rate
07         file2-string : input file comprising hours worked
08         file3-string : output file for storing total monthly salary
09     Return Value: None
10     """
11     try:
12         employeeIn1 = open(file1, 'r')
13         employeeIn2 = open(file2, 'r')
14         employeeOut = open(file3, 'w', newline='')
15     except IOError:
```

```
        print("Error in opening File")
    try:
        readerEmployee1 = csv.reader(employeeIn1)
        readerEmployee2 = csv.reader(employeeIn2)
        writerEmployee = csv.writer(employeeOut)
        fieldHeader=['empID','monthySalary']
        writerEmployee.writerow(fieldHeader)
        next(readerEmployee1) # skip reading header row
        next(readerEmployee2) # skip reading header row
        for file1Record in readerEmployee1:
                file2Record=next(readerEmployee2)
                hourlyRate = file1Record[2]
                numHours = file2Record[1]
                line=[file1Record[0],float(hourlyRate)*float (numHours) ]
                writerEmployee.writerow(line)
    except IOError:
        print("Error in Reading/Writing the file")
    employeeIn1.close()
    employeeIn2.close()
    employeeOut.close()

##### Main program segment starts here! #####
#Objective: To compute monthly salary of employees
file1 = input('Enter file name having hourly rate:')
file2 = input('Enter file name having hours worked:')
file3 = input('Enter file name for monthly salary:')
calculateMonthlySalary(file1, file2, file3) # Compute employees' salaries
```

On execution of Program 3.10 with `empMain.csv, empHoursWorked.csv,` and `empMonthlySalary.csv` as the input for **`file1`**, **`file2`**, and **`file3`**, the system outputs the contents of the file `mempMonthlySalary.csv` as follows:

```
empID,monthySalary
9001,10000.0
9002,7650.0
9003,9345.0
9004,6944.0
9005,6450.0
```

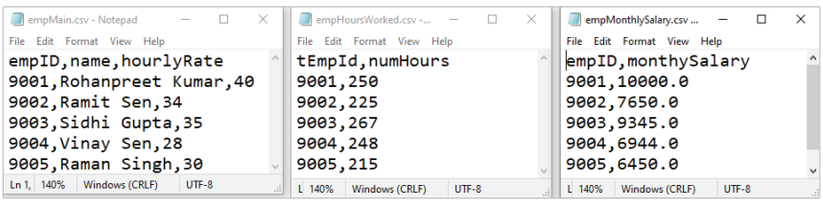


Fig 3.3: Showing the files `empMain.csv, empHoursWorked.csv` *and* `empMonthlySalary.csv`

Next, assume that there may be some employees who may not have worked at all in a month. So, the file `empHoursWorked.csv` does not have any record for such employees. A sample file `empHoursWorked.csv` is shown below:

```
EmpId,numHours
9001,250
9004,248
```

As before, to calculate the monthly pay of all employees in this case, we must read the files `empMain.csv` and `empHoursWorked.csv` and generate a third file `empMonthlySalary.csv` that contains the monthly salaries of all the employees. Salary is shown as 0 for the employees who did not work at all in the whole month.

Now, go through the Pseudocode 5 and write a Python program to do the needful:

```
01 Open the file empMain.csv in read mode
02 Open the file empHoursWorked.csv in read mode
03 Open the file empMonthlySalary.csv in write mode
04 Read first record (file2Record) from empHoursWorked.csv
05 For each file1Record in empMain.csv:
   i.  If empID of file1Record and tEmpID of file2Record matches.
       a. Calculate monthly salary by multiplying hourlyRate with numHours.
       b. Write empID along with computed total monthly salary in the monthlySalary.
          csv file
       c. Read next record (file2Record) from empHoursWorked.csv
   ii. Else
       a.  Calculate monthly salary by multiplying hourlyRate with numHours (assumed
           to be zero).
       b. Write empID along with computed total monthly salary in the monthlySalary.
          csv file
06 Close the files opened for reading or writing.
```



Write a program to calculate the monthly pay of all employees. In this case, we must read the files `empMain.csv` and `empHoursWorked.csv` and generate a third file `empMonthlySalary.csv` that contains the monthly salaries of all the employees assuming number of hours worked as zero of there is no entry in second file.



3.8.5 Sorting Examination Data

In the next application, let us revisit the first example where the results board exams of class XII are stored in comma-separated data file `examResults.csv`. The file comprises information about roll number, name, and marks of the students. The information for each student appears on a different line. The following is an example of data from the file:

```
RollNum,Name,Marks
1201,Shashank Sen,92
1202,Ridhi Dogra,74
1203,Sumit Jain,79
1204,Aaradhya Pandey,55
1205,Mehul Kapoor,66
```

We would like to sort the records of the students based on their marks and rewrite the sorted result back to another file, namely, `sortedExamResults.csv`. The Pseudocode 6 for producing a sorted file of the student data is:

```
01 Open the file examResults.csv in read mode
02 Open the file sortedExamResults.csv in write mode
03 Read the entire content of examResults.csv in a list of lists
04 Sort the sublists comprising students details w.r.t. marks
05 Store the sorted list to the file sortedExamResults.csv
06 Close the files opened for reading or writing.
```

The program for sorting records of the students on the basis of their marks is given in Program 3.11.

- The **`main`** function takes the names of the source file as input, comprising board exam results of class XII (line 44) and the output file name for storing the sorted results (line 45).
- In line 46, the **`main`** function invokes **`sortRecords`** function to sort exam results with respect to marks.
- The **`sortRecords`** function opens the source file (line 25) and the output file (line 26). The source file and the output file are referred to by file handlers **`studentIn`**, and **`studentOut`** respectively.

- Note that these lines are enclosed in a **try** block as the open file operation may generate **IOError** exceptions. The **except** block following the **try** block ensures smooth termination of the program in case an exception is raised.
- Lines 30 and 31 create reader object **readerMarks** and writer object **writerMarks** for file handlers **studentIn** and **studentOut** respectively.
- Line 32 reads the header row from the source file which is written to the destination file (line 33). Subsequently, line 34 reads the remaining list of records in the file referred.
- Thereafter, line 35 invokes the method **sort** (lines 2–13) which sorts these records using bubble sort.
- Note that sorting has been done on the marks obtained by the students i.e. the second component of the list (line 11).
- Finally, the sorted list of records returned by **sort** function is written to the destination file (line 36) and the source and destination files are closed (lines 39–40).

Program 3.11 Program to sort the exam results with respect to marks

```
01 import csv
02 def sort(records):
03     """
04     Objective: To sort list of lists w.r.t. marks(3rd field)
05     Input Parameter:
06         records-list
07     Return Value: None
08     """
09     for i in range(len(records)):
10         for j in range(0,len(records)-1-i):
11             if records[j][2]<records[j+1][2]:
12                 records[j], records[j+1] = records[j+1], records[j]
13     return records
14
15 def sortRecords(file1, file2):
16     """
17     Objective: To moderate exam marks
18     Input Parameter:
19         file1-string : input file comprising board exam results
20         file2-string : output file for storing results sorted w.r.t.
21                        marks
22     Return Value: None
23     """
24     try:
25         studentIn=open(file1, 'r')
26         studentOut=open(file2, 'w', newline='')
27     except IOError:
28         print("Error in opening File")
29     try:
30         readerMarks=csv.reader(studentIn)
31         writerMarks=csv.writer(studentOut)
32         header=next(readerMarks)
33         writerMarks.writerow(header)
34         records=list(readerMarks)
35         sortedRecords=sort(records)
36         writerMarks.writerows(sortedRecords)
37     except IOError:
38         print("Error in Reading/Writing the file")
39     studentIn.close()
40     studentOut.close()
41
```

```
42 ##### Main program segment starts here! #####
43 #Objective: To determine moderated marks of students
44 file1 = input('Enter the file name comprising exam results:')
45 file2 = input('Enter the file name for storing sorted records:')
46 sortRecords(file1, file2)
```

On execution of Program 3.11 with `examResults.csv` and `sortedExamResults.csv` as the input for **`file1`** and **`file2`**, the system outputs the contents of the file `sortedExamResults.csv` as follows:

```
RollNum,Name,Marks
1201,Shashank Sen,92
1203,Sumit Jain,79
1202,Ridhi Dogra,74
1205,Mehul Kapoor,66
1204,Aaradhya Pandey,55
```

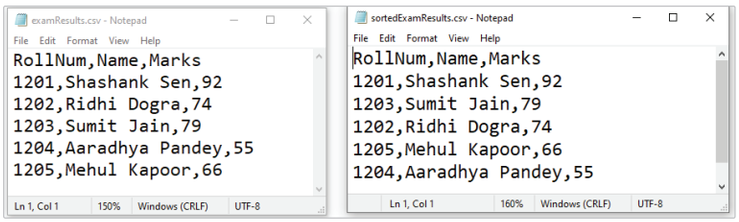


Fig 3.4: Showing the files *examResults.csv* and `sortedExamResults.csv`

In Program 3.12, we write the above program differently. Instead of defining our own function for sorting the records, we make use of Python's in-built function **`sorted`** (line 25). The function takes the records and the key (keyword argument) for sorting the records. Here, we define a function **`criteria`** that returns the key on which we wish to sort the records. Further, the keyword argument **`reverse`** is set to **`True`** so that the records are sorted in decreasing order of marks.

Program 3.12 Program to sort the exam results w.r.t. marks

```
01 import csv
02 def sort(records):
03     """
04     Objective: To sort list of lists w.r.t. marks(3rd field)
05     Input Parameter:
06         records-list
07     Return Value: None
08     """
09     for i in range(len(records)):
10         for j in range(0,len(records)-1-i):
11             if records[j][2]<records[j+1][2]:
12                 records[j], records[j+1] = records[j+1], records[j]
13     return records
14
15 def sortRecords(file1, file2):
17 try:
18         readerObj=csv.reader(f1)
19         writerObj=csv.writer(f2)
20         header=next(readerObj)
21         writerObj.writerow(header)
22         records=list(readerObj)
23         def criteria(record):
24             return record[2]
25         sortedRecords=sorted(records, key=criteria, reverse=True)
26         writerObj.writerows(sortedRecords)
27     except IOError:
```

```
28          print("Error in Reading/Writing the file")
29      f1.close()
30      f2.close()
31
32  ##### Main program segment starts here! #####
33  #Objective: To determine moderated marks of students
34  file1 = input('Enter the file name comprising exam results:')
35  file2 = input('Enter the file name for storing sorted records:')
36  sortRecords(file1, file2)
```

Let's Summarise

- File is a stream of bytes.
- A text file is a human-readable file that comprises a sequence of characters stored in the form of ASCII or UNICODE characters. Each line in a text file is terminated by an End of Line (**EOL**) character sequence that may vary from operating system to operating system.
- The data in a binary file is stored as the machine-readable data objects.
- Python provides the built-in function `open` for opening the file. The function takes the name of the file and the mode in which we need to access the file as an input. The syntax of the function is as follows:

  f = open(*fileName, accessMode*) # f is a user specified name

- The following access modes may be used as required:
  - mode (**r**): For reading an already existing file
  - mode (**w**): For writing to a file. If the file being opened already exists in the folder, opening the file in write mode will destroy the existing data of the file.
  - mode (**a**): For writing the new data at the end of an already existing file. If the named file does not exist in the folder, a new file is created.
  - mode (**r+**) or (**w+**): For reading from a file as well as writing into it.
  - mode(**a+**): For reading from a file as well as appending new data to it.
- There are several data members and methods associated with the file object that can be used to perform operations on the file or to show the details of the file. The **dir()** function returns a complete list of these objects. For example, members **name** and **mode** can be used to retrieve the name of the file and the mode of access to the file, respectively. The data member **closed** returns **True** if the file is closed and **False** otherwise.
- Function **close()** is used to close a file, once the necessary operations on a file have been carried out.
- Function **read()** is used to read the entire contents of the file.
- Function **readline()** is used to read line by line from a file.
- Function **readlines()** returns the entire content of the file from the current position in the form of a list of lines
- Function **write()** is used to write the data to the file. The function returns the number of bytes written to the file.
- Function **writelines()** allows us to write a sequence of lines to a file in one go.
- Python files that are opened using the with clause are automatically closed when control exits the with clause.
- The **flush()** method can be used to forcefully write the content from Python's buffer to the file without waiting for the user to close the file.
- Python module **pickle** is used for reading and writing data objects to a file:
  - Pickling is the process of transforming a Python object into a byte stream for writing the object to a file.

- While reading a binary file, reverse process of unpickling is applied to obtain the data back in the object form.
- The methods **load()** and **dump()** of the **pickle** module are used for reading and writing the objects from and to the files respectively. Syntax for using the **load()** and **dump()** methods of the **pickle** module are mentioned as follows:

```
pickle.dump(sourceObject, fileObject)
pickle.load(fileObject)
```

- In Python, default reading takes place from the keyboard and the outputs are displayed on the console. Standard Input **stdin** is a file object that corresponds to the input stream of characters. Similarly, standard output **stdout** is a file object that receives outputs. The error messages are received in the file object **stderr**.
- Function **seek()** is used to point the file handler to a particular position in the file.
- The function **tell()** returns the current position of the file handler .
- CSV (Comma Separated Values) is a comma-delimited file.
- To facilitate writing the data to CSV files, Python module **csv** provides the **writer** class that takes the file object as an input argument and returns the writer object **csv.writer** that is used for writing to a CSV file:
  - The **writer** class enables us to write one row at a time using the method **writerow()**. Multiple rows can be written using the method **writerows()**. To facilitate reading the data from the already existing CSV files,
  - Python module **csv** provides the **reader** class that takes file object as an input argument and returns a reader object (**csv.reader**) that is used for reading data from the CSV file.
- Exceptions are the errors generated by Python while executing a program.
- The **try-except** blocks are used to handle the exceptions. A **try** block contains statements that may lead to an exception, the **except** block specifies what action should be performed if an exception is raised.
  - Typically, an **except** block includes a sequence of exceptions and the corresponding action to be performed for each exception.
- Opening a file which does not exist or writing to a file which is opened in read mode (or vice-versa) results in **IOError** exception.

Solved Exercises

A. Multiple Choice Questions

1. While writing to a CSV file, we want to avoid having a blank row after writing each record. Name the keyword argument that should be included while invoking **open()**?

a. delimiter b. newline c. buffering d. encoding

2. Identify the valid syntax to write an object onto a binary file opened in write mode:

a. pickle.dump **(<object to be written>, <file handler>)**

b. dump.pickle **(<file handler>,<object to be written>)**

c. dump.pickle **(<object to be written>, <file handler>)**

d. pickle.dump **(<file handler>,<object to be written>)**

3. When you read a CSV file using **csv.reader()**, it returns the values in a _____ object.

a. Dictionary b. List c. String d. Integer

4. Which built-in function returns the entire file content in the form of a list of lines?

a. read() b. readline() c. readlines() d. write()

5. Which of the following is NOT an example of a text file?

a. .txt b. .csv c. .py d. .bin

6. Which file open mode overwrites the contents of an already existing file?

a. a b. r+ c. w d. r

7. What will be the output produced on execution of the following code?

```
f = open("File1.txt","wb")
print(f.mode)
f.close()
```

a. wb b. w c. b d. Runtime Error

8. Which function is used to read all characters in a text file?

a. readline() b. read() c. readall() d. readlines()

9. Which of the following statements is not True?

a. **`tell()`** returns an integer b. **`seek()`** positions the file pointer in a data file

c. **`seek()`** always takes 2 arguments d. **`tell()`** never takes any argument

10. The process of converting a byte stream to Python object is called _____.

a. pickling b. unpickling c. objectifying d. unobjectifying

11. Observe the following code carefully and then choose the correct option to complete the **`with`** statement:

```
lst = ["I01","Pencil",10,5]
with _________ as f:
  rowWriter = csv.writer(f, delimiter = ',')
  rowWriter.writerow(lst)
```

a. open("abc.txt",'r') b. open("abc.csv",'r') c. open("abc.txt",'w') d. open("abc.csv",'w')

12. Suppose the content of text file, '*goodManners.txt*' is: (all sentences on a new line)

```
Wish all your elders
always be polite
Be punctual
```

What will be the output of the following code?

```
f = open("goodManners.txt","r")
data = f.readlines()
print(len(data))
f.close()
```

a. 2 b. 3 c. 4 d. 5

13. Consider the following code:

```
def count(key):
    f = open("countFile.txt","r")
    num=0
    data = f.readlines()
    for line in data:
        if line[0] == key:
            num+=2
    return num
ch='K'
print(print("The characters K occurs", count(ch), "times")
```

If the contents of countFile.txt is as given in the image, what will be the output of the given program?

```
Do you know about kites?
Kites are seen in multiple colours and shapes.
I love to fly kites.
Kites are very easy to make.
```

a. 4 b. 2 c. 8 d. Program will flash an error.

14. Consider the following code:

```
import pickle
_____                                  # Statement 1
data = ("T1010", "Sunny", "Chandigarh", "Nagpur", 15000)
pickle.dump(data,f)
f.close()
```

Which of the following statements (Statement 1) will open the file, result.dat, to add the new record at the end of the file?
Options:

a. f.open("result.dat","ab")
b. pickle.open("result.dat","ab")
c. open.pickle("result.dat","ab")
d. f = open("result.dat","ab")

15. To print the 4th line from the text file *try.txt*, which of the following is correct? Assume that the file object **f** currently points to the beginning of the file.

a. lst = f.readlines(); print(lst[3])
b. lst = f.read(4); print(lst[3])
c. lst = f.readline(4); print(lst[3])
d. lst = readlines(); print(lst[4])

16. Which of the following is not correct? [CBSE Sample Paper 2021–22]

a. if we try to read a text file that does not exist, an error occurs.
b. if we try to read a text file that does not exist, the file gets created.
c. if we try to write on a text file that does not exist, no error occurs.
d. if we try to write on a text file that does not exist, the file gets created.

17. Which of the following should be used to read the first line of the text file myFile.txt? [CBSE Sample Paper 2021–22]

a. f = open('myFile.txt'); f.read()
b. f = open('myFile.txt','r'); f.read(n)
c. f = open('myFile.txt'); f.readline()
d. f = open('myFile.txt'); f.readlines()

18. Which of the following should be used to read the whole text file *myFile.txt* as a list of strings that appear line by line?

a. f = open('myFile.txt'); f.read()
b. f = open('myFile.txt','r'); f.read(n)
c. f = open('myFile.txt'); f.readline()
d. f = open('myFile.txt'); f.readlines()

19. Which of the following should be used to retrieve the string comprising the first 3 characters of the file *myFile.txt*?

a. f = open('myFile.txt'); f.read(); f.read();f.read()
b. f = open('myFile.txt','r'); f.read(3)
c. f = open('myFile.txt'); f.read()[3]
d. f = open('myFile.txt'); f.readlines(3)

20. Which of the following should be used to retrieve the third line of the text file *myFile.txt*?

a. f = open('myFile.txt'); f.read(2)
b. f = open('myFile.txt','r'); f.readline(2)
c. f = open('myFile.txt'); f.readlines()[3]
d. f = open('myFile.txt'); f.readlines()[2]

21. Which of the following options can be used to open the text file student.txt, stored on a hard disk? [CBSE Sample Paper 2021–22]

(i) myfile = open('student.txt','rb')
(ii) myfile = open('student.txt','w')
(iii) myfile = open('student.txt','r')
(iv) myfile = open('student.txt')

a. only (i)
b. both (i) and (iv)
c. both (iii) and (iv)
d. (i), (iii), and (iv)

B. State whether the following statements are True or False:

1. Test.bin is a human-readable file. _____

2. If a file object is created using r+ file mode, an exception will be raised if the file does not exist. _____

3. The pickle module contains function definitions to read and write data from a text file. _____

4. A file object has to be created before writing data to a text file. _____

5. When you open a file for writing and the file does not exist, an error is generated. _____

C. Fill in the blanks.

1. _____ function is used to close a data file in Python.
2. _____ is a process where a byte stream from a binary file is converted to a Python object.
3. In a case of a text file, a _____ character signifies the end of a line.
4. The default delimiter for a CSV file is _____.
5. To read and write into a binary file, _____ module has to be imported.
6. The default file mode to open a file is _____.
7. To write a list of strings into a text file, _____ method is used.
8. The file mode _____ does not erase the previously stored records in a binary file.
9. The file mode to open a binary file for writing, as well as reading, is _____.
10. The _____ method of pickle module reads data from a binary file.

D. Answer the following questions:

1. What is the difference between `open()` and `close()`?

Ans. An `open()` is used to open a data file to read and write data from and to the data file.

`close()` closes the data file. After this command, no further reading or writing operation can take place.

2. Rasna wants to add data to the text file but she wants the old data to be retained. In which mode should she open the file? Give the command if the file handler is H and the file to be opened is *quotes.txt*

Ans. Rasna should open the file in append mode. The command will be:
H=open('quotes.txt','a')

3. Give any two characteristics of a binary file.

Ans. (i) The data in binary files is stored in the form of 0s and 1s.

(ii) The binary files are not human readable.

4. What is the advantage of opening a file using **`with`** clause?

Ans. If the file is opened using **`with`** clause, it may not be closed explicitly.

5. Write a function to count the number of sentences beginning with an uppercase alphabet in a text file that is passed as an argument to the function.

Ans.
```
def countLines(fileName):
    f=open(fileName,"r")
    l=f.readlines()
    for x in l:
        if x[0].isupper():
            w=w+1
    print('THE NUMBER OF SENTENCES BEGINNING WITH UPPERCASE ALPHABET IS:',w)
```

6. Define unpickling with respect to binary files. Which function from the pickle module is used for unpickling of data?

Ans. Unpickling is the process of transforming the byte stream back to the Python object. load() is used for unpickling of data.

7. Identify the error(s) in the following code that writes data on to a CSV file.

```
import pickle
L1=[123,"Pencil",45]
L2=[123,"Pen",50]
print("Writing data to a csv file")
with open("ABC.csv",'w',newline='') as C1:
     wr=csv.writer(C1, delimiter='comma')
wr.writerows(L1,L2)
```

Ans. Following program has the errors striked off and the correction done alongside:

```
import csv     # csv module should be imported
L1=[123,"Pencil",45]
L2=[123,"Pen",50]
print("Writing data to a csv file")
```

```
with open("ABC.csv",'w',newline='') as C1:
    wr=csv.writer(C1, delimiter=',') # use ',' operator
    wr.writerows(L1)  # only one sequence argument
    wr.writerow(L2)
```



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): Data can be stored permanently in data files.
Reasoning (R): Data is stored temporarily in Python variables.

2. Assertion(A): A csv file format is one which looks like a text file.
Reasoning (R): The data is organized with one record on each line and each field value of the record is separated by a space.

3. Assertion(A): A binary file in Python is used to store data objects such as lists and dictionaries that can be later retrieved in their original form using the pickle module.
Reasoning (R): A binary file can be read using any text editor.

4. Assertion(A): By default, the writerow() creates a header row in a csv file.
Reasoning (R): The writerow() of csv module allows to write a single record into each row of csv file.

5. Assertion(A): A binary file stores data in the same way as is stored in the memory.
Reasoning (R): A binary file does not have an EOL (End of Line) character.

Ans. 1. b 2. c 3. c 4. d 5. b



Case-based Questions

1. Garvit has created a text file named friends.txt and typed the following sentences in it:
Friends are fun to be with

They make you happy

We are nothing without friends. They are our lifeline

His friend is learning various functions to read data from a text file. He has written the following program using the text file created by Garvit.

```
def COUNTLINES() :
  file = open ('friends.txt', 'r')
    line1 = file.readline()
    line2=file.readline(10)
    line3=file.read(3)
    print(line1,line2,line3,sep='*')
    print(file.readline())
    file.close()
```

What will be the output produced on the execution of the given code?

Ans. Friends are fun to be with

*They make *you

Happy

2. Sudha is an entrepreneur. She wants to store the data of all her clients in a binary file, client.dat. She has approached you to write a function that stores the following details of her clients:
clientId- integer

cName- string

city - string

contactNo- string

Ans.
```
import pickle
def write_bin():
    bin_file=open("client.dat","wb")
    while True:
        clientId=int(input("enter client id : "))
        cName=input("enter client name : ")
        city=input("enter city : ")
        contactNo=input("enter contact no : ")
        cDetail=[clientId,cName,city,contactNo]
        pickle.dump(cDetail,bin_file)
        ans=input("Do you wish to enter more records y/n")
        if ans.lower()=='n':
            break
    bin_file.close()
write_bin()
```

3. Sudha has already saved the records in a binary file, client.dat. Now, she wants to retrieve the records of the clients who stay in Delhi. Write a user-defined function, read_bin() that will perform the desired task.

Ans.
```
import pickle
def read_bin():
    bin_file=open("client.dat","rb")
    try:
        while True:
            c_detail=pickle.load(bin_file)
            #print(c_detail)
            if c_detail[2]=="delhi":
                print(c_detail)
    except:
        print(" No more records")
    finally:
        bin_file.close()
```

4. Umang is doing a research on population of various countries. For his research, he wants to create a Python program that will store the following list object into a csv file named country.csv:
countryData=[country,capital,population]

Write the following user defined functions for Umang.

(i) writeData(): This function that takes a list object as an argument and writes it (along with headings) into a csv file.

(ii) readData(): This function reads data from the csv file and displays those records that have population greater than 50.

Ans.
```
def writeData(cdata):
    headings=["Country","Capital","Population"]
    f=open('country.csv','w',newline='')
    csvwriter=csv.writer(f)
```

```
    csvwriter.writerow(headings)
    csvwriter.writerow(cdata)
    f.close()

def readData():
    f=open('country.csv','r')
    csvreader=csv.reader(f, delimiter=',')
    head=list(csvreader)
    print(head[0])
    for x in head:
        if x[0]=="Country":
            continue
        if int(x[2])>50:
            print(x)
    f.close()
```

5. Rahul has successfully completed partial code to add data to a binary file, EMP.DAT. But due to lack of complete knowledge of Python File Handling, he has left a certain portion of the code. You as an expert of Python have to help Rahul by providing the missing portions and also by answering other related queries based on the following code written by Rahul.

```
import pickle
def NEWEMPS():
    Fw=_____________________ #1To create binary file to store data
    for i in range(5):
    EMPID = int(input("EMP ID :"))
    NAME = input("NAME : ")
    DEPT = input("DEPT :"))
    EMPD = [EMPID, NAME, DEPT]
    _____________ #2 To write the list EMPD into the file
    __________ #3 To close the file
```

Ans.
```
Statement 1: Fw=open("EMP.DAT",'ab')
Statement 2: pickle.dump(EMPD,Fw)
Statement 3: Fw.close()
```



Assessment

A. Multiple Choice Questions

1. Which of the following is the file mode that allows adding new content at the bottom of the existing file and creates a new file if it does not exist?

a. r+ b. w+ c. a+ d. w

2. A text file **"info.txt"** contains 20 lines of text. Consider the code given below:

```
F=open('info.txt')
message=F.readline()
```

What will be the data type of message?

a. Tuple b. List c. String d. integer

3. Assume that the position of the file pointer is at the beginning of 4th line in a text file . Which line displayed when the following code is executed?

```
s = f.readlines()
print(s[1])
```

a. 1st line b. 2nd line c. 4th line d. 5th line

4. A text file, message.txt contains the following text:
You should love yourself.

Which of the following code will produce the following output?

You Should Love Yourself.

a.
```
f=open("message.txt")
print(f.read().titlecase())
```

b.
```
f=open("message.txt")
print(f.read().upper())
```

c.
```
f=open("message.txt")
print(f.read().title())
```

d.
```
f=open("message.txt")
print(f.read(1).title())
```

5. Choose the most appropriate option out of (a), (b), (c) and (d) to fill in the blank space in the following code:

```
import csv
data = [ [1, "Ram"], [2,"Balram"] ]
with open("myth.csv","w") as FB:
        csv.____(data, FB)
```

a. `write` b. `dump` c. `writelines` d. `writerows`

6. What would be the output of the following code, if the file `sample.txt` contains the following text
Welcome

```
f=open("sample.txt","a")
print(f.tell())
```

a. 8 b. 7 c. 0 d. –1

7. What is the description of the file mode bit `rb+`?

a. Opens a binary file for reading and writing with the file pointer at the beginning of the file
b. Opens a binary file for reading and writing with the file pointer at the end of the file
c. Opens a binary file for reading only
d. Opens a text and binary file for reading only

8. Which of the following methods is used to read data (unpickling) from a binary file?

a. load() b. dump() c. seek() d. tell()

9. A csv file, `data.csv` contains the following data:

Laptop	DELL
OS	Windows
HDD	1 TB

Which of the following option contains the correct code to add `["RAM", "4 GB"]` to the csv file?

a.
```
import csv
F=open("data.csv","w",newline='')
F.writerow(["RAM", "4 GB"] )
F.close()
```

b.
```
import csv
F=open("data.csv","a",newline='')
F.writerow(["RAM", "4 GB"] )
F.close()
```

c.
```
import csv
F=open("data.csv","w",newline='')
F.writer(["RAM", "4 GB"] )
F.close()
```

d.
```
import csv
F=open("data.csv","a",newline='')
F.writer(["RAM", "4 GB"] )
F.close()
```

10. Which of the following parameter needs to be used with `open()` to avoid inserting a blank row after each record in a CSV file?

a. delimiter b. newline c. file object d. writer

11. Which of the following is the correct output for the code given below, if the text file, `myfile.txt` contains the content–CHECK?

```
def printCode():
    try:
        f=open("myfile.txt")
        S=f.read()
        NS=""
        for C in range(len(S)):
            NS += S[C]
            print(NS,end="@")
        f.close()
    except:
        print("File not found")
printCode()
```

a. `C@H@E@C@K@`

b. `C@CH@CHE@CHEC@CHECK@`

c.
```
C@
C@H@
C@H@E@
C@H@E@C@
C@H@E@C@K@
```

d.
```
C@
CH@
CHE@
CHEC@
CHECK@
```

12. A binary file, `Student.dat` contains the following student records
[roll no, name, marks].

The function `copystudents()` has been written to copy the records of students with marks greater than 75 from `student.dat` to `excel.dat`. Complete statements 1 and 2 to accomplish this task.

```
def CopyStudents():
    G=open("EXCEL.DAT", "wb")
    with open("STUDENT.DAT","rb") as F:
        while True:
            try:
```

```
            r=___________ # statement 1
            if r[2]>75:
                ___________# statement 2

        except:
            break
G.close()
```

a. `pickle.load(F)`
   `pickle.dump(r,G)`

b. `pickle.load(F)`
   `pickle.dump(G,r)`

c. `pickle.dump(r,G)`
   `pickle.load(F)`

d. `pickle.dump(G,r)`
   `pickle.load(F)`

13. What will be the output of the given code if file, `Emp.csv` contains the following data:

Header →	**Empno**	**Empname**	**Salary**
Row1 →	101	Anuj	50000
Row2 →	102	Arijita	40000
Row3 →	103	Hanika	30000
Row4 →	104	Firoz	60000
Row5 →	105	Vijaylakshmi	40000

```
import csv
with open("Emp.csv") as F:
    Re=csv.reader(F)
    next(Re)
    Lst=[]
    for I in Re:
        x = int(I[0])
        if x>=102 and x<=104:
            continue
        Lst.extend(I)
print(Lst)
```

a. `[ ['101', 'Anuj', '50000'], ['102','Arijita','40000'] ]`

b. `['101', 'Anuj', '50000', '102', 'Arijita','40000']`

c. `[['101', 'Anuj', '50000'], ['105','Vijaylakshmi','40000']]`

d. `['101', 'Anuj', '50000', '105', 'Vijaylakshmi','40000']`

14. Which of the following statements is used to read each record from the binary file EMP.DAT? Assume that file object if F and data object is `EMPID`.

a. `EMPD = pickle.load(F)`

b. `F.readrow(EMPD)`

c. `EMPD= pickle.read(F)`

d. `pickle.load(EMPD,F)`

15. Which of the following statements is used to close a file? Assume that file object is `F`.

a. `close(F)`

b. `pickle.close()`

c. `F.close()`

d. `pickle.close(F)`

16. Which of the following is the most appropriate statement to open the file `EMP.DAT` for writing data at the end of a binary file?

a. `F=open("EMP.DAT",'wb')`
b. `F=open("EMP.DAT",'ab')`
c. `F=open("EMP.DAT",'wb+')`
d. `F=open("EMP.DAT",'rb')`

17. Which of the following statements is used to write the list `EMPD` into the binary file, `EMP.DAT`? Assume the file object is Fw.

a. `Fw.writerow(EMPD)`
b. `pickle.write(Fw, EMPD)`
c. `pickle.dump(EMPD,Fw)`
d. `F=pickle.dump(EMPD)`

B. State whether the following statements are True or False:

1. The relative path of a data file is relative to the current working directory. _____
2. The `load()` of the pickle module performs the pickling process. _____
3. The functions, `readline()` and `readlines()` return a string object. _____
4. The csv files are also essentially text files. _____
5. A binary file can only have an extension .dat. _____

C. Fill in the blanks.

1. A _____ file stores data in the form of ASCII characters.
2. A _____ file stores data in the form of 0s and 1s.
3. The file mode that opens the text file for writing, retaining the previous records is ____.
4. The default file mode is _.
5. A csv file is a type of _____ file.
6. The data files that are human readable are _____ file and _____ file.
7. A _____ file can be opened in text editor as well as MS Excel.
8. The function used to close a binary file is _____.
9. Each line of a text file is terminated by _____ character.
10. In case of _____ files, the data values of each record are separated by _____.

D. Answer the following questions:

1. Distinguish between the following:
   - readline and readlines
   - write and writelines
   - write mode and append mode
   - load and dump method
   - try and except
   - csv.reader and csv.writer
   - pickling and unpickling
   - seek and tell

2. Determine the output on executing the following sequence of statements.

```
f = open('file1.txt', 'r')
f.tell()
f.read(2)
f.tell()
```

Assume that file *file1.txt* contains 'XYZ' as the content.

3. Determine the output on the execution of following code snippet:

(i)
```
stud = open('student.txt', 'r')
length = len(stud.read())
print('Number of characters read from file student',length)
data = stud.read(5)
print(data)
```

(ii)
```
file1 = open('meditation.txt', 'w')
file1.write('Meditation is the food to the soul!')
file1.write('It takes care of the health of mind.\n')
file1.write('Meditation has been practiced for thousands of years to help people
become more aware of the present moment.')
```

```
file1.close()
file1 = open('meditation.txt', 'r')
data = file1.readline()
print(data)
file1.close()
```

(iii)
```
file1 = open('meditation.txt', 'w')
file1.write('Meditation is the food to the soul!')
file1.write('It takes care of the health of mind.\n')
file1.write('Meditation has been practiced for thousands of years to help people
become more aware of the present moment.')
file1.close()
file1 = open('meditation.txt', 'r')
data = file1.read(60)
print(data)
file1.close()
```

(iv)
```
file1 = open('meditation.txt', 'w')
file1.write('Meditation is the food to the soul!')
file1.write('It takes care of the health of mind.\n')
file1.write('Meditation has been practiced for thousands of years to help people
become more aware of the present moment.')
file1.close()
file1 = open('meditation.txt', 'r')
data1 = file1.readline()
data2 = file1.read(60)
print(data1, data2)
file1.close()
```

4. Find the error in the following code snippet:

```
empDetails = [  ['E01', 'Aasmeen', 'Khan', '01-10-1989', 1000000, 'Admin'],
     ['E02', 'Pooja', 'Gupta', '13-02-1985', 2000000, 'Research'],
     ['E03', 'Rachit', 'Mittal', '01-01-1990', 900000, 'Admin'],
     ['E04', 'Ayushi', 'Sinha', '23-05-1985', 2500000, 'Research'],
     ['E05', 'Sumit', 'Talwar', '29-03-1988', 1400000, 'Admin'],
     ['E06', 'Rahul', 'Roy', '22-05-1986', 1200000, 'Admin'],
     ['E07', 'Diksha', 'Dewan', '12-12-1979', 3200000, 'Research']]
employee = open('employees.csv', 'w', newline='')
writeEmployee = csv.writer(employee)
writeEmployee.writerow(empDetails)
```

5. Consider the file `'employees.csv'` created in the text in the chapter. Determine the output on the execution of the following code snippet.

```
employee = open('D:\CBSE_Python\employees.csv', 'r')
readEmployee = csv.reader(employee)
lineNum=0
for row in readEmployee:
    if lineNum == 0:
        for value in row:
            print(value, end="\t")
            print("\n"+"*"*60)
        else:
            print()
            print(value[0], end="\t")
            print(value[1], end="\t")
            print(value[-1], end="\t")
    lineNum+=1
employee.close()
```

6. Consider a file `'message.txt'` with the content `'Good Morning!'`. Determine the output of the file after the execution of the following code snippet:

```
file1 = open('message.txt', 'w')
file1.write('Welcome to the class.')
file1.close()
```

7. Write a function that takes two file names, file1 and file2 as input. The function should read the contents of the file file1 and should write them to another file file2 after adding a newline at the end of each line.
8. Write a function that takes data line by line as input from the user in an interactive manner until the user provides a null string as the input. Each line read should be capitalized and written to a file named *data.txt*.
9. Write a function that takes two file names, namely, file1 and file2 as input parameter from the user. The function should read the line-wise contents of the first file and write only the alternate odd-numbered lines to the second file.
10. Write a function that reads the contents of the *Poem.txt* file and counts the number of times the words "my" and "rainbow" appear in it.
11. Write a function that reads the contents of the swap file and list the frequency count of every word in the file. Also, it should determine and display the most frequently occurring word in the file.
12. Write a program that defines two functions. First function should take the contents of file *Story.txt* as input from the user. Write another function that reads the contents from the file and display the lines that begin with the article 'A'.
13. Write a program that defines two functions. The first function saveText should take some text as input from the user and store it in a newly created file *Story.txt*. The second function descFile should read the contents from a file and compute the descriptive statistics by computing the number of characters, words, and sentences. The function should return its output as a tuple. The program should use the function descFile to compute and display the descriptive statistics of the file *Story.txt*.
14. Write a function that accepts as input the name of a file and computes the number of uppercase and lowercase characters in it, and the number of words which start with a vowel. The function should return its output as a list.
15. A CSV file contains information about several items. It contains one line for each item that contains comma-separated values of item number and the price per unit. For every item stored in the first file, the second file contains eaxctly one line comprising comma-separated values of item number and the quantity purchased. Items in the two files appear in the same order of item number. Write a function that takes two CSV file names as input parameters from the user and creates a third CSV file which contains one line for each item having comma separated values of item number and total price of the item.
16. For the above problem, consider that the first file contains an additional field mentioning weight of the item. Rewrite the function so that the third CSV file contains one more column storing price per unit weight.
17. Write a program that takes two dictionaries containing details of all students of the class XII A and XII B respectively. For each student, the dictionary should have an entry for roll number as the key and the name of the student as the value part. Write a function in the program that takes these two dictionaries as an input and writes it to a file '*classXIIstudents.dat*'. Write another function that reads these two dictionaries from the file and display the details of each student in separate lines.
18. Write a program that takes a dictionary of the following form from a user:

```
{'RollNum':12001, 'Name':'Priyanka', 'Section':'A', 'Marks':85}
```

The value of variable n should be taken as input from the user. The program should n dictionaries in the file'classXIIstudents.dat'. Thereafter, it should read the contents of the file and should update the marks by providing an increment of 5 while ensuring that the marks scored does not exceed maximum marks of 100.

19. The program should writes the key: value pairs of DICT objects in file and create another comma-separated file with two fields, namely, username and domain_name. For every record (email id) in the original file, the new file should contain the separate entry under two heads. For example, for the entry 'mrinali@gmail.com', split the string into two parts such that mrinali should be displayed under the username and gmail.com should be identified as the domain_name.
20. Rewrite the program to moderate the student results (section 3.10.1) using the following moderation criteria:

```
marks<=27, moderation = 7
27<marks<=40, moderation =6
40<marks<=50, moderation =5
50<marks<=60, moderation =4
60<marks<=70, moderation =3
70<marks<=80, moderation = 2
80<marks<=90, moderation =1
90<marks, moderation =0
```



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): dump() from pickle module is used to perform the pickling process with binary files.
Reasoning (R): Pickling is the process by which a Python object is converted to byte stream.

2. Assertion(A): The file modes 'r','w', and 'a' can be used to open both text files and csv files.
Reasoning (R): A csv file is a type of text file with comma(,) as a delimiter.

3. Assertion(A): Pickling is the process of converting a Python object into byte stream.
Reasoning (R): The process of pickling is performed while reading data from a binary file.

4. Assertion(A): A read or write operation in a data file takes place at the current position of file pointer.
Reasoning (R): The position of the file pointer will change after every read or write operation.



Case-based Questions

1. Rohit, a student of class 12th, is learning CSV File Module in Python. During examination, he has been assigned an incomplete python code (shown below) to create a CSV File '*Student.csv*' (content shown below). Help him in completing the code which creates the desired CSV File.

CSV File

1,AKSHAY,XII,A
2,ABHISHEK,XII,A
3,ARVIND,XII,A
4,RAVI,XII,A
5,ASHISH,XII,A

Incomplete Code

```
import_____ #Statement-1
fh = open(_____, _____, newline='') #Statement-2
stuwriter = csv._____ #Statement-3
data = []
header = ['ROLL_NO', 'NAME', 'CLASS', 'SECTION']
data.append(header)
for i in range(5):
    roll_no = int(input("Enter Roll Number : "))
    name = input("Enter Name : ")
    Class = input("Enter Class : ")
    section = input("Enter Section : ")
    rec = [_____] #Statement-4
    data.append(rec)
stuwriter. _____ (data) #Statement-5
fh.close()
```

(i) Identify the suitable code for blank space in line marked as Statement-1.

a. csv file b. CSV c. csv d. Csv

(ii) Identify the missing code for blank space in line marked as Statement-2?

a. `"School.csv","w"` b. `"Student.csv","w"`

c. `"Student.csv","r"` d. `"School.csv","r"`

(iii) Choose the function name (with argument) that should be used in the blank space of line marked as Statement-3.

a. **reader(fh)** b. **reader(MyFile)** c. **writer(fh)** d. **writer(MyFile)**

(iv) Identify the suitable code for blank space in line marked as Statement-4.

a. **'ROLL_NO', 'NAME', 'CLASS', 'SECTION'**

b. **ROLL_NO, NAME, CLASS, SECTION**

c. **'roll_no','name','Class','section'**

d. **roll_no,name,Class,section c) co.connect()**

(v) Choose the function name that should be used in the blank space of line marked as Statement-5 to create the desired CSV File?

a. **dump()** b. **load()** c. **writerows()** d. **writerow()**

2. Amritya Seth is a programmer, who has recently been given a task to write a python code to perform the following binary file operations with the help of two user defined functions/modules:

(I) **AddStudents()** to create a binary file called *STUDENT.DAT* containing student information—roll number, name and marks (out of 100) of each student.

(II) **GetStudents()** to display the name and percentage of those students who have a percentage greater than 75. In case there is no student having **percentage > 75** the function displays an appropriate message. The function should also display the average percent. He has succeeded in writing partial code and has missed out certain statements, so he has left certain queries in comment lines. You as an expert of Python have to provide the missing statements and other related queries based on the following code of Amritya.

Answer any four questions (out of five) from the below mentioned questions.

```
import pickle
def AddStudents():
    Choice="y"
    ____________ #1 statement to open the binary file to write data
    while Choice=="y":
        Rno = int(input("Rno :"))
        Name = input("Name : ")
        Percent = float(input("Percent :"))
        L = [Rno, Name, Percent]
        ____________ #2 statement to write the list L into the file
    Choice = input("enter more (y/n): ")
    if Choice in "nN":
        break
    F.close()
def GetStudents():
   Total=0
   Countrec=0
   Countabove75=0
   with open("STUDENT.DAT","rb") as F:
   while True:
      try:
   ____________ #3 statement to read from the file
   Countrec+=1
   Total+=R[2]
   if R[2] > 75:
        print(R[1], " has percent = ",R[2])
        Countabove75+=1
   except:
        break
```

```
        if Countabove75==0:
            print("There is no student who has percentage more than 75")
            average=Total/Countrec
            print("average percent of class = ",average)
    AddStudents()
    GetStudents()
```

(i) Which of the following commands is used to open the file "*STUDENT.DAT*" for writing only in binary format? (marked as #1 in the Python code)

a. `F= open("STUDENT.DAT",'wb')`
b. `F= open("STUDENT.DAT",'w')`
c. `F= open("STUDENT.DAT",'wb+')`
d. `F= open("STUDENT.DAT",'w+')`

(ii) Which of the following commands is used to write the list L into the binary file, STUDENT.DAT? (marked as #2 in the Python code)

a. `pickle.write(L,f)`
b. `pickle.write(f, L)`
c. `pickle.dump(L,F)`
d. `f=pickle.dump(L)`

(iii) Which of the following commands is used to read each record from the binary file STUDENT.DAT? (marked as #3 in the Python code)

a. `R = pickle.load(F)`
b. `pickle.read(r,f)`
c. `r= pickle.read(f)`
d. `pickle.load(r,f)`

(iv) Which of the following statement(s) are correct regarding the file access modes?

a. 'r+' opens a file for both reading and writing. File object points to its beginning.
b. 'w+' opens a file for both writing and reading. Adds at the end of the existing file if it exists and creates a new one if it does not exist.
c. 'wb' opens a file for reading and writing in binary format. Overwrites the file if it exists and creates a new one if it does not exist.
d. 'a' opens a file for appending. The file pointer is at the start of the file if the file exists.

(v) Which of the following statements correctly explain the function of seek() method?

a. tells the current position within the file.
b. determines if you can move the file position or not.
c. indicates that the next read or write occurs from that position in a file.
d. moves the current file position to a given specified position



Previous Years' Questions

1. The syntax of `seek( )` is :
`file_object.seek(offset[,reference_point])`

What is the default value of `reference_point`? [2023]

a. 0 b. 1 c. 2 d. 3

2. a. Write the definition of a Python function named `LongLines()` which reads the contents of a text file named `LINES. TXT'` and displays those lines from the file which have at least 10 words in it. For example, if the content of `'LINES. TXT'` is as follows:

Once upon a time, there was a woodcutter
He lived in a little house in a beautiful, green wood.
One day, he was merrily chopping some wood.
He saw a little girl skipping through the woods, whistling happily.
The girl was followed by a big gray wolf.

Then the function should display output as:

He lived in a little house in a beautiful, green wood.
He saw a little girl skipping through the woods, whistling happily.

OR

b. Write a function `count_Dwords()` in Python to count the words ending with a digit in a text file `"Details. txt"`.

Example:

If the file content is as follows :

On seat2 VIP1 will sit and

On seat1 VVIP2 will be sitting

Output will be:

Number of words ending with a digit are 4 [2023]

Ans. a.
```
def LongLines():
  f=open("lines.txt")
  dt=f.readlines()
  for i in dt:
    if len(i.split())>10:
      print(i)
  f.close()
LongLines()
```

b.
```
def Dwords():
  f=open("details.txt")
  dt=f.read()
  w=dt.split()
  c=0
  for i in w:
    if i[-1].isdigit():
      c+=1
  print("Number of words ending with digit are:",c)
Dwords()
```

3. Which of the following statement is not correct? [2022]

a. We can write content into a text file opened using 'w' mode

b. We can write content into a text file opened using 'w+' mode

c. We can write content into a text file opened using 'r' mode

d. We can write content into a text file opened using 'r+' mode

Ans. c. We can write content into a text file opened using 'r' mode

4. Which of the following option is the correct Python statement to read and display the first 10 characters of a text file "Notes.txt"? [2022]

a. `F = open ('Notes.txt'); print (F.load (10))`

b. `F = open('Notes.txt'); print (F.dump (10))`

c. `F = open('Notes.txt'); print (F.read(10))`

d. `F = open('Notes.txt'); print (F.write(10))`

Ans. c. `F = open('Notes.txt'); print (F.read(10))`

5. Which of the following is not a correct Python statement to open a text file "Notes.txt" to write content into it? [2022]

a. `F = open('Notes.txt', 'w')`

b. `F = open('Notes.txt', 'a')`

c. `F = open('Notes.txt', 'A')`

d. `F = open('Notes txt', 'w+')`

Ans. c. `F = open('Notes.txt', 'A')`

6. A text file opened using the following statement: `MyFile = open ('Notes.txt')` [2022]
Which of the following is the correct Python statement to close it?

a. `MyFile=close('Notes.txt')`
b. `MyFile.close('Notes.txt')`
c. `close.MyFile()`
d. `MyFile.close()`

Ans. d. `MyFile.close()`

7. Which of the following option is the correct usage for the tell( ) of a file object? [2022]

a. It places the file pointer at a desired offset in a file
b. It returns the entire content of a file
c. It returns the byte position of the file pointer as an integer
d. It tells the details about the file

Ans. c. It returns the byte position of the file pointer as an integer

8. Which of the following statement is incorrect in the context of pickled binary files? [2022]

a. csv module is used for reading and writing objects in binary files.
b. pickle module is used for reading and writing objects in binary files.
c. load() of the pickle module is used to read objects.
d. dump () of the pickle module is used to write objects.

Ans. a. csv module is used for reading and writing objects in binary files.

9. What is the significance of the seek() method? [2022]

a. It seeks the absolute path of the file.
b. It tells the current byte position of the file pointer within the file.
c. It places the file pointer at a desired offset within the file.
d. It seeks the entire content of the file.

Ans. c. It places the file pointer at a desired offset within the file.

10. Which of the following is the correct expansion of csv? [2022]

a. Comma Separated Values.
b. Centrally Secured Values.
c. Computerised Secured Values.
d. Comme Secured Values.

Ans. a. Comma Separated Values.

11. If the following statement is used to read the contents of a testfile object F: x=F. readline () [2022]
Which of the following the correct data type of x?

a. string
b. list
c. tuple
d. dictionary

Ans. b. list

12. Which of the following is the default character for the newline parameter for a csv file object opened in write mode in Python IDLE? [2022]

a. \n
b. \t
c. ,
d. ;

Ans. a. \n

13. Suppose the content of a text file Notes.txt is: [2022]
"The way to get started is to quit talking and begin doing"
What will be the output of the following Python code?

```
F = open ("Notes.txt")
F = seek (29)
S = F.read()
print (S)
```

a. The way to get started is to
b. quit talking and begin doing
c. The way to get started is to quit talking and begin doing
d. gniod nigeb dna gniklat tiug ot si detrats teg ot yaw ehT

Ans. b. quit talking and begin doing

14. Which of the following Python modules is imported to store and retrieve objects using the process of serialization and deserialization? [2022]

a. csv b. binary c. math d. pickle

Ans. d. pickle

15. Which of the following function is used with the csv module in Python to read the contents of a csv file into an object? [2022]

a. readrow() b. readrows() c. reader() d. load()

Ans. c. reader()

16. Suppose the content of a text file "Rhymes.txt" is as follows: [2022]

```
Jack & Jill
went up the hill
```

What will be the output of the following Python code?

```
F = open("Rhymes.txt")
L = F.readlines ()
for i in L:
  S=i.split()
  print (len (S), end="#")
```

a. 2#4# b. 3#4# c. 2# d. 7#

Ans. b. 3#4#

17. Suppose the content of "Rhymes.txt" is: [2022]

```
One, two, three, four, five
Once I caught a fish alive.
```

What will be the output of the following Python code?

```
F = open("Rhymes.txt")
S = F.read()
print (S.count('e' ,20))
```

a. 20 b. 1 c. 3 d. 6

Ans. b. 1

18. Suppose the content of "Rhymes.txt" is: [2022]

```
Baa baa black sheep,
have you any wool?
```

What will be the output of the following Python code?

```
F = open("Rhymes.txt")
S = F.read()
L=S.split ()
for i in L :
    if len(i) %3 != 0 :
        print (i, end= " ")
```

a. Baa baa you any b. black have wool?

c. black sheep, have wool? d. Error

Ans. c. black sheep, have wool?

19. Suppose the content of "Rhymes.txt" is [2022]

```
Good Morning Madam
```

What will be the output of the following Python code?

```
F = open("Rhymes.txt")
L = F.read().split ()
```

```
for W in L:
    if W.lower () == W[:: -1].lower():
        print(W)
```

a. Good b. Morning c. Madam d. Error

Ans. c. Madam

20. Suppose the content of "Rhymes.txt" is [2022]

```
Hickory Dickory Dock
The mouse went up the clock
```

What will be the output of the following Python code?

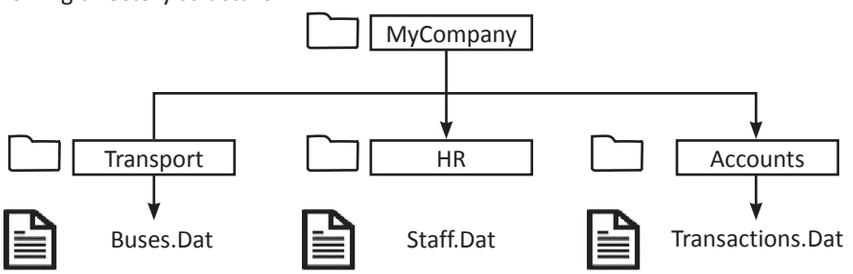
```
F = open("Rhymes.txt")
L = F.readlines()
X = ["the", "ock"]
for i in L:
    for W in i.split() :
        if W in X:
            print(W,end = "*")
```

a. the* b. Dock*The*the*clock*

c. Dock*the*clock* d. Error

Ans. d. Error

21. Consider the following directory structure [2022]



Suppose the present working directory is MyCompany. What will be the relative path of the file Transactions.Dat?

a. MyCompany/Transactions.Dat b. MyCompany/Accounts/Transactions.Dat

c. Accounts/Transactions.Dat d. . . /Transactions.Dat

Ans. b. MyCompany/Accounts/Transactions.Dat

Case Study:

22. Nisha, an intern in ABC Pvt. Ltd., is developing a project using the csv module in Python. She has partially developed the code as follows leaving out statements about which she is not very confident. The code also contains errors in certain statements. Help her in completing the code to read the desired CSV File named "Employee.csv" [2022]

```
#CSV File Content
ENO, NAME, DEPARTMENT
E1, ROSHAN SHARMA, ACCOUNTS
E2, AASMA KHALID, PRODUCTION
E3, AMRIK GILL, MARKETING
E4, SARAH WILLIAMS, HUMAN RESOURCE
#incomplete Code with Errors
import CSV   #Statement-1
with open (___, ___, newlines= ' ') as File:     #Statement-2
ER = csv.____#Statement-3
for R in range (ER) :    #Statement-4
```

```
if _____ == "ACCOUNTS" :    #Statement-5
    print ( ____, ____)     #Statement-6
```

1. Nisha gets an Error for the module name used in Statement-1. What should she write in place of CSV to import the correct module?

a. file b. csv c. Csv d. pickle

Ans. b. csv

2. Identify the missing code for blank spaces in the line marked as Statement-2 to open the mentioned file.

a. "Employee.csv", "r" b. "Employee.csv", "w" c. "Empolyee.csv", "rb" d. "Employee.csv", "wb"

Ans. a. "Employee.csv", "r"

3. Choose the function name (with parameter) that should be used in the line marked as Statement-3.

a. reader (File) b. readrows (File) c. writer (File) d. writerows (File)

Ans. a. reader (File)

4. Nisha gets an Error in Statement-4. What should she write to correct the statement?

a. for R in ER: b. while R in range (ER): c. for R= ER: d. while R == ER :

Ans. a. for R in ER:

5. Identify the suitable code for blank space in Statement-5 to match every row's 3rd property with "ACCOUNTS".

a. ER[3] b. R[2] c. ER[2] d. R[3]

Ans. b. R[2]

6. Identify the suitable code for blank space in Statement-6 to display every Employee's Name and corresponding Department?

a. ER[1], R [2] b. R[1], ER[2] c. R[1], R[2] d. ER[1], ER[2]

Ans. c. R[1], R[2]

23. Write a statement in Python to open a text file MARKER.TXT so that existing content can be read from it. [2020]

Ans.
```
file= open("MARKER.TXT","r")
OR
file= open("MARKER.TXT","r+")
```

24. Write a statement in Python to open a text file DATA.TXT so that new contents can be written in it. [2020]

Ans.
```
file= open("DATA.TXT","w")
OR
file= open("DATA.TXT","w+")
```

25. Write a method/function ABLINES() in python to read contents from a text file LINES.TXT, to display those lines, which are either starting with an alphabet 'A' or starting with alphabet 'B'. [2020]

For example:

If the content of the file is

A BOY IS PLAYING OUTSIDE

THE PLAYGROUND IS BIG

BANYAN TREE IS IN THE GROUND

The method/function should display

A BOY IS PLAYING OUTSIDE

BANYAN TREE IS IN THE GROUND

Ans.
```
def ABLINES():
  file=open('LINES.TXT','r')
  lines = file.readlines()
  for w in lines:
    if w[0]=="A" or w[0]=="B":
```

```
        print(w)
    file.close()
```

26. Write a method/function shortWords() in python to read lines from a text file WORDBANK.TXT, and display those words, which are lesser than 5 characters. [2020]

For example:

If the content of the file is

HAPPY JOY WELCOME KITE

LOVELY POSITIVE FUN

The method/function should display

JOY

KITE

FUN

Ans.
```
def shortWords():
    c=0
    file=open('WORKBANK.TXT','r')
    line = file.read()
    word = line.split()
    for w in word:
      if len(w)<5:
        print(w)
    file.close()
```

27. Which of the following is not correct? [2019]

a. if we try to read a text file that does not exist, an error occurs.
b. if we try to read a text file that does not exist, the file gets created.
c. if we try to write on a text file that does not exist, no error occurs.
d. if we try to write on a text file that does not exist, the file gets created.

Ans. b. if we try to read a text file that does not exist, the file gets created.

NCERT Exercise Solutions

1. Differentiate between:

a. text file and binary file
b. `readline()` and `readlines()`
c. `write()` and `writelines()`

Ans. a. A text file is a human-readable file that comprises of a sequence of characters stored in the form of ASCII or UNICODE characters. Each line in a text file is terminated by an End of Line (EOL) character that may vary across operating systems.

A binary file is a machine-readable data object. For example, in a text file the number–12345 will be stored as a sequence of six bytes, whereas in a binary file it may be stored as an integer object requiring 16, 32, or 64 bits, depending on the size of the integer object.

b. The `readline()` method is used to read line by line from a file. Each line is terminated with a newline (\n).

The `readlines()` method returns the entire content of the file from the current position in the form of a list of lines.

c. The method `write()` takes a string as an argument and writes the data to the file. The function also returns the number of bytes written to the file.

The `writelines()` method allows us to write a sequence of lines to a file in one go. We need to pass iterable objects such as lists, tuple, etc., containing strings to the `writelines()` method. Unlike method `write()`, the `writelines()` method does not return the number of characters written in the file.

2. Write the use and syntax for the following methods:

a. open()

b. read()

c. seek()

d. dump()

Ans. a. Refer to 3.2.1: Opening Files

b. **`read(n)`** **:** The read function reads n bytes of data from a file into a variable/object. If n is not specified or is a negative number, it reads the contents of an entire file. The syntax is:

```
file_object.read(n)
```

c. Refer to 3.6 Random Access Using seek() and tal()

d. Refer to 3.3.6 Binary File Operation

3. Write the file mode that will be used for opening the following files. Also, write the Python statements to open the following files:

a. a text file "example.txt" in both read and write mode

b. a binary file "bfile.dat" in write mode

c. a text file "try.txt" in append and read mode

d. a binary file "btry.dat" in read only mode.

Ans. a. `'r+'`

```
file_object=open("example.txt", 'r+')
```

b. `'wb'`

```
file_object = open("bfile.dat", 'wb')
```

c. `'a+'`

```
file_object = open("try.txt", 'a+')
```

d. `'rb'`

```
file_object = open("btry.dat", 'rb')
```

4. Why is it advised to close a file after we are done with the read and write operations? What will happen if we do not close it? Will some error message be flashed?

Ans. Closing a file after use is a good programming practice as it helps free all the resources allocated to it. When we close a file, Python makes sure that any unwritten or unsaved data is written to the file before it is closed. No error message is generated.

5. What is the difference between the following set of statements (a) and (b):

a.
```
P = open("practice.txt","r")
x = P.read(10)
```

b.
```
with open("practice.txt", "r") as P:
        x = P.read()
```

Ans. Differences between statement (a) and (b) are given the following:

a. In statement **a**, call to `read()` function using file object P reads 10 bytes from the file "practice.txt", whereas, in statement **b**, it reads the entire contents of the file.

b. In statement a, `close()` method is not invoked to close the file, whereas in statement b, the file is automatically closed when the control moves out of the **`with`** block.

6. Write a command(s) to write the following lines to the text file named `hello.txt`. Assume that the file is opened in append mode.

" Welcome my class"

"It is a fun place"

"You will learn and play"

Ans.
```
F = open('hello.txt', 'a')
f.write(" Welcome my class")
f.write("It is a fun place")
f.write("You will learn and play")
f.close()
```

7. Write a Python program to open the file `hello.txt` used in question no 6 in read mode to display its contents. What will be the difference if the file was opened in write mode instead of append mode?

Ans.
```
f = open('hello.txt','r')
print(f.read())
```

Opening the file in write mode will overwrite the existing data of the file. However, when the file is opened in append mode, the new data is written at the end of already existing content.

8. Write a program to accept string/sentences from the user till the user enters "END" to. Save the data in a text file and then display only those sentences which begin with an uppercase alphabet.

Ans.
```
f = open("sentences.txt", 'w')
sentence=input("enter a line of text('END' to quit)")
while sentence!= 'END':
    f.write(sentence + "\n")
    sentence=input("enter a line of text('END' to quit)")
f.close()

f = open("sentences.txt",'r')
linesList = f.readlines()
for line in linesList:
    if line[0].isupper():
        print(line)
f.close()
```

9. Define pickling in Python. Explain serialization and deserialization of Python object.

Ans. Pickling is the process of transforming a Python object into a byte stream for writing the object to a file. In Python, we use the pickle module to perform operations on Binary files. Process of pickling is also known as serialization of Python object.

While reading a binary file, the reverse process of unpickling is applied to obtain the data back in the object form, which is also known as deserialization.

10. Write a program to enter the following records in a binary file:
Item No integer

Item_Name string

Qty integer

Price float

Number of records to be entered should be accepted from the user. Read the file to display the records in the following format:

Item No:

Item Name:

Quantity:

Price per item:

Amount: (to be calculated as Price * Qty)

Ans.
```
import pickle
item={}
f = open('itemDetails.dat','wb')
num = int(input("Enter number of records to be stored"))
for i in range(num):
    itemNum = input("Enter the Item No")
    itemName = input("Enter the item name")
    qty = int(input("enter quantity"))
    price = float(input("Enter the item price"))
    item['Item No'] = itemNum
    item['Item Name'] = itemName
    item['Qty'] = qty
    item['Price'] = price
    pickle.dump(item, f)
f.close()

f = open('itemDetails.dat','rb')
try:
    while True:
        item = pickle.load(f)
        print("Item no: ", item['Item No'])
        print("Item name: ", item['Item Name'])
        print("Quantity", item['Qty'])
        print("Price", item['Price'])
        print("Amount", item['Qty']*item['Price'])
except EOFError:
    f.close()
```

Answers

Multiple Choice Questions

1. (b)	2. (a)	3. (b)	4. (c)	5. (d)	6. (c)	7. (a)	8. (b)	9. (c)
10. (b)	11. (d)	12. (b)	13. (a)	14. (d)	15. (a)	16. (b)	17. (c)	18. (d)
19. (b)	20. (d)	21. (d)						

True and False

1. (F)	2. (T)	3. (F)	4. (T)	5. (F)

Fill in the blanks

1. close()	2. Unpickling	3. newline	4. comma	5. pickle
6. r	7. writelines()	8. a / a+	9. w+/ r+	10. load()

4 EXCEPTION HANDLING

Chapter Outline

4.1 Syntax Errors
4.2 Exceptions
4.3 Built-in Exceptions
4.4 Handling Exceptions using try...except
4.5 Handling Multiple Exceptions
4.6 Copying File Example
4.7 Finally Clause

Introduction

Sometimes a program terminates abruptly with error messages incomprehensible to the end user. This kind of behaviour is entirely unexpected for a professional program being used in practice. Exception handling is a feature of Python that allows developers to manage and respond to errors or exceptional situations gracefully. When a program encounters an error or an unexpected event, known as an exception, it disrupts the normal flow of execution. Exception handling enables programmers to catch these exceptions, handle them appropriately, and prevent the program from terminating abruptly.

In this chapter, we will study some techniques for handling exceptions in Python programs. By effectively handling exceptions, we can make the programs enhance the user experience by providing informative error messages and enabling graceful recovery from exceptional situations.

Exceptions are errors or exceptional conditions that occur during the execution of a program, disrupting the normal flow of code.

4.1 Syntax Errors

We have already discussed syntax errors, also known as parsing errors, which occur when there is a violation of the rules (grammar) of the programming language (Python in our case). In the event of a syntax error in a Python program, the Python interpreter promptly detects and reports it, along with an explanation of the error and suggestions for resolution. Thus, when a syntax error is encountered, the program execution is stopped by the interpreter. So, to proceed further, we need to correct the errors, save the modifications, and run the program again.

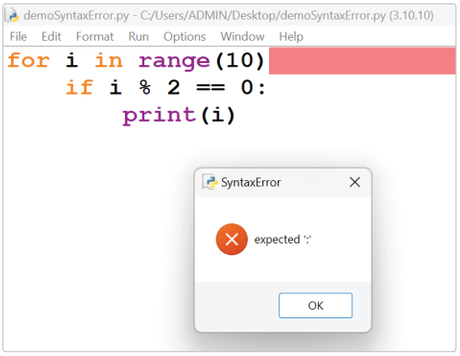
Below, we show an example of a syntax error encountered while executing a for-loop in shell mode:

An example of a syntax Error

```
>>> print('Hello) #missing quote mark at the end of the string
...
    SyntaxError: incomplete input
>>> for i in range (0, 10)   # missing colon in the for statement
```

```
...     print(i)
...
SyntaxError: expected ':'
```

In script mode, when a syntax error is encountered during program execution, a dialog box appears, showcasing the error name along with a concise description of the issue.



Give two examples of `syntaxErrors`.

4.2 Exceptions

Exceptions are erroneous events that arise during the execution of a program, causing a deviation from the normal flow of code. Exception handling allows programmers to anticipate and respond to potential errors or exceptional situations gracefully. By incorporating exception handling mechanisms, such as try-except blocks, Python programs gain resilience even in the face of unexpected events. Instead of causing a complete program failure, exceptions can be caught, analyzed, and appropriate actions can be taken to handle them.

This approach provides several advantages. First, it helps debug and diagnose issues by providing valuable information about the cause and location of errors. The exception messages generated by Python are informative and aid developers in understanding the nature of the problem. Second, exception handling enables graceful recovery from errors, allowing programs to continue executing and taking alternative paths. Thus, by effectively handling exceptions, we can improve user experience. So, it is essential for developers to have a good understanding of exception handling techniques to build robust and error-tolerant software systems. When an exception occurs, it is automatically raised by the Python interpreter. Subsequently, the program execution is transferred to an exception handling block, where the exception can be captured and managed effectively, ensuring that the program does not terminate abruptly.

4.3 Built-in Exceptions

Built-in exceptions are commonly encountered exceptions that are pre-defined by the compiler or interpreter. Python ' s standard library includes a comprehensive set of built-in exceptions that cater to various common errors and provide standardized solutions for effectively handling them. When a built-in exception is encountered, Python provides the name of the raised exception along with the accompanying explanation.

Built-in exceptions in Python are predefined error situations that address common issues and provide standardized solutions.

It is the responsibility of the programmer to implement appropriate actions to handle the exception and ensure the program's smooth execution. Below, we describe several frequently encountered built-in exceptions in Python and illustrate with examples how they may be handled.

1. **NameError**: This exception occurs whenever a name that appears in a statement is not found globally. Below, we give an example of a **NameError**:

```
>>> marks = Input('Enter your marks')

Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    marks = Input('Enter your marks')
NameError: name 'Input' is not defined. Did you mean: 'input'?
```

Note that in the above example, Python raises the `NameError` exception. It not only tells that the identifier `'Input' is not defined`, it also suggests, `Did you mean: 'input'?`

so that it is easy to apply the correction. We have already discussed that identifiers (names) in Python are case-sensitive. So, the name **input** and **Input** are considered different variables.

2. **TypeError**: This exception occurs when an operation or function is applied to an object of inappropriate type. For example, let us examine the following code snippet:

```
>>> 'sum of 2 and 3 is ' + 5
...
Taackback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    'sum of 2 and 3 ' + 5
TypeError: can only concatenate str (not "int") to str
```

Note that Python is complaining of a TypeError because we are trying to concatenate a string `'sum of 2 and 3 is'` with the integer `5` using the `+` operator.

3. **ValueError**: This exception occurs whenever an inappropriate argument value, even though of the correct type, is used in a function call. For example, if you invoke the function **int()** with the string **'23,684'**, Python raises the **ValueError** exception:

```
>>> int('23,684')
Taackback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    int('23,684')
ValueError: invalid literal for int() with base 10: '23,684'
```

Although the **int()** function is used to convert a value to an integer, for the conversion to succeed, the string argument must represent a valid integer literal. However, in the above example, the string **'23,684'** does not represent a valid integer (as the use of the comma is not allowed), so the conversion fails. Therefore, executing the expression **int('23,684')**, it raises a **ValueError** with the message **invalid literal for int() with base 10: '23,684'**. This error indicates that the string **'23,684'** cannot be interpreted as an integer in base **10**.

4. **ZeroDivisionError:** This exception occurs when we try to perform a numeric division in which the denominator happens to be zero. For example, consider

```
>>> 78/(2+3-5)
...
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    78/(2+3-5)
ZeroDivisionError: division by zero
```

The problem with the expression **78/(2+3-5)** is that the denominator **(2+3-5)** evaluates to **zero (0)**, leading to **ZeroDivisionError** exception. In Python (and most programming languages), division by zero is not allowed because it results in undefined behaviour in mathematics.

5. **IndexError:** This exception occurs whenever we try to access an index that is out of a valid range. For example, let us examine the following code snippet:

```
>>> colors = ['red', 'green', 'blue']
>>> colors[3]
Traceback (most recent call last):
  File "<pyshell#1>, line 1, in <module>
    color[3]
IndexError: list index out of range
```

We already know that in Python, indexing begins with index **0**, so the first element is at index **0**, the second element is at index **1**, and so on. In the above code, the list **colors** has three elements, which means the valid indices are **0**, **1**, and **2**. On attempting to access **colors[3]**, it results in an **IndexError** because index **3** is beyond the bounds of the valid indexes of the list **colors**.

1. Which built-in exception is raised when division by zero is performed?
2. What is the purpose of the IndexError exception in Python?

6. **EOFError:** It is raised when an input function, such as **input()**, reaches the end of a file while trying to read input from the user. Suppose, a program is reading the user input interactively. The user accidentally ends the input by pressing the end-of-file character In the interactive mode, the end of input is indicated by entering Ctrl+d. Suppose a user enters Ctrl+d, by mistake before providing the expected input. For example, consider the following code snippet:

```
>>> num = int(input('Enter a number: '))
Enter a number:
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    num - int(input('Enter a number: '))
EOFError: EOF when reading a line
```

As the user entered Ctrl+d, Python raised an EOFError exception, as it could not find the user input.

4.4 Handling Exceptions using try...except

In order to avoid the sudden termination of a program when an exception is raised, it is necessary to handle the exception by catching it and implementing the appropriate actions using the try...except clause. The try block consists of statements that can potentially raise an exception. These statements are enclosed within the try block because they are likely to raise an exception during execution. The except block defines the actions to be performed when an exception is raised within the try block. Thus, it provides a handling mechanism for the identified exception. Thus, we can deal with runtime errors in a controlled manner and avoid abrupt program terminations.

The general syntax of a try...except block in Python is as follows:

```
try:
    # statements that may raise an exception
except ExceptionType:
    # actions to be performed when ExceptionType is raised
```

In a program, you can replace ExceptionType with the specific exceptions you want to catch and handle. For example, in the above code, we could have enclosed the statement:

```
num = int(input('Enter a number: '))
```

within try...except block, as shown below:

```
try:
    num = int(input('Enter a number: '))
    # Rest of the code that uses num
except EOFError:
    print('End of file reached. Please provide some input.')
    num = int(input('Enter a number: '))
```

In the above code, if the user enters the end-of-file character instead of providing a number, an `EOFError` will still be raised, but it would be managed smoothly by asking the user (one more time) to enter the input. Thus, the Python code has now become more user-friendly.

7. **`FileNotFoundError:`** It is raised when an attempt is made to open or access a file that does not exist. For example,

```
>>> file = open("example.txt", "r")
    Traceback (most recent call last):
       File "<pyshell#2>", line 1, in <module>
          file = open("example.txt", "r")
    FileNotFoundError: [Errno 2] No such file or directory: 'example.txt'
```

Note that as we tried to open a file named **`example.txt`** in read mode, which did not exist in the working directory, a **`FileNotFoundError`** exception is raised. Below we illustrate how we can handle the **`FileNotFoundError`** exception smoothly:

```
>>> try:
...         file = open("example.txt", "r")
...         # Perform operations on the file
...         file.close()
... except FileNotFoundError:
...         print("File missing in working directory")
```

In the above code, when we try to open a file named `example.txt` in read mode, and the file doesn't exist in the working directory, a `FileNotFoundError` is still raised, but it would be managed smoothly. If the exception occurs, the program jumps to the except block that contains the handling mechanism for the exception `FileNotFoundError`. The user now gets a suitable message.

8. **`PermissionError:`** It is raised when the program doesn't have the necessary permissions to read from or write to a file. For, example,

```
>>> file = open("example.txt", "w")
    Traceback (most recent call last):
      File "C:\User\ADMIN\Desktop\hhh.py", line 1, in <module>
        file = open("example.txt", "w")
    PermissionError: [Error 13] Permission denied: 'example.txt'
```

Note that as we tried to open a file named **`example.txt`** in write mode, which doesn't have the write permission.

```
>>> try:
...     file = open("example.txt", "w")
...     # Perform operations on the file
...     file.close()
... except PermissionError:
...     print("Permission denied to access the file.")
```

In the above example, we attempt to open a file named **`example.txt`** in write mode. If the program doesn't have the required permissions to access the file, a **`PermissionError`** is raised. If the exception occurs, the program jumps to the except block that contains the handling mechanism for the exception **`PermissionError`**. The exception is caught using the except block, and the message "Permission denied to access the file." is printed.



What is the purpose of the `FileNotFoundError` exception?



CT 01

Rewrite the following code segment so that likely division by zero exception is handled smoothly.

```
num1 = int(input("Enter a number: "))
num2 = int(input("Enter another number: "))
result = num1 / num2
print("Result:", result)
```

The try block contains the code that may raise an exception, and the `except` block specifies how to handle specific exception types.

4.5 Handling Multiple Exceptions

When a code segment involves the possibility of several exception situations, the **`try`** block is followed by a sequence of except blocks. We can also include multiple **`except`** blocks to handle different exceptions separately. The general syntax of a **`try...except`** block in case of multiple exceptions is as follows:

```
try:
    # Code that may raise an exception
    # ...
except ExceptionType1:
    # Code to handle ExceptionType1
    # ...
except ExceptionType2:
    # Code to handle ExceptionType2
    # ...
else:
    # Optional code that executes if no exception is raised
    # ...
```

Now, let's delve into the individual components of the syntax provided above.

- The **`try`** block contains the code that may raise an exception.
- Multiple `except` blocks are used to handle different types of exceptions. Each `except` block can handle a specific exception type.
- When an exception of occurs within the **`try`** block, the corresponding **`except`** block is executed.
- The **`else`** block is *optional* and is executed if no exceptions are raised within the `try` block.

In the following code snippet, we consider the possibility of one of the following exception: `ValueError`, `ZeroDivisionError`, occurring in the `try` block:

```
>>> num1 = 24
>>> try:
...     num2 = int(input("Enter the second number: "))
...     result = num1 / num2
...     print("The result is:", result)
... except ValueError:
...     print("Invalid input. Please enter a valid integer.")
... except ZeroDivisionError:
...     print("Error: Cannot divide by zero.")
... else:
...     print("No exception occurred.")
```

In the above example, the code inside the `try` block attempts to read a string and convert it to a number. Suppose, by mistake, you entered the string `24.5`, expecting that Python will typecast it to an integer; an exception will be raised because Python expects the string to be converted to an int to be comprised of only digits, possibly preceded by a **+** or **–** sign. divide two numbers provided by the user. If a `ValueError` occurs (e.g., the user enters a non-integer for `num2`), the first `except` block is executed. Similarly, if the user enters 0 as the input, a `ZeroDivisionError` occurs. If no exceptions occur, the `else` block is executed.

To catch all possible exceptions in Python, we can use a generic `except` block without specifying a particular exception type. This allows us to catch any exception that occurs during the execution of the code.

Let us understand it with the help of an example:

```
try:
    # Code that may raise an exception
    # ...
except:
    # Code to handle any exception
    # ...
```

The `except` block without any specified exception type acts as a catch-all for all exceptions. It will handle any exception that occurs within the `try` block and display the type of exception. In the following example, the code snippet is used to calculate the price per unit weight of an item based on user input. To begin with, we import the `sys` module, which is used later to access information about exceptions. On execution of the code, it prompts the user to enter the cost of the item purchased and assigns the input value to the variable `cost`. Similarly, it prompts the user to enter the weight of the item purchased and assigns the input value to the variable `weight`. Finally, the price per unit weight is computed. As the except block does not include any specific exceptions, all exceptions are handled uniformly. When an exception occurs, Python uses `sys.exc_info()` to retrieve information about the exception. It produces a tuple that includes the exception type, exception value, and traceback object. This tuple can be utilized to extract further particulars regarding the exception and manage it in a suitable manner. If no exceptions occur in the `try` block, the `else` block is executed. In the `else` block, it prints the **`price.`**

```
01 import sys
02 cost = input('Enter cost of item purchased: ')
03 weight = input('Enter weight of item purchased: ')
04
05 try:
06     cost = float(cost)
07     weight = float(weight)
08     price = cost / weight
09
10 except:
11     print('Invalid input provided by user \n' + str(sys.exc_info()))
12 else:
13     print('Price per unit weight: ', price)
```

Sample Outputs:

```
>>>
    Enter cost of item purchased:
    Enter weight of item purchased: 60
    Invalid input provided by user
    (<class 'ValueError'>, ValueError("could not convert to float: '' "), <traceback
    object at 0x0000029F250A8400>)
>>>
    Enter cost of item purchased: 100
    Enter weight of item purchased:
    Invalid input provided by user
```

```
(<class 'ZeroDivisionError'>, ZeroDevisionError('float division by zero'), <traceback
object at 0x000001D4A7AAC400>)
>>>
Enter cost of item purchased: 100
Enter weight of item purchased: 50
Price per unit weight: 2.0
```

4.6 Copying File Example

In the following example, we intend to make a copy of an existing file available in the working directory.

The execution of the code may run into difficulty for several reasons, such as

1. The source file does not exist in the working directory.
2. The program is not authorised to write into the destination file, possibly because the destination file has been marked *read only*.

   We incorporate these two situations in the form of exceptions in the following code snippet:

```
01 import sys
02 try:
03     # Open the source file in read mode
04     source_file = open('source.txt', 'r')
05
06     # Read the contents of the source file
07     content = source_file.read()
08
09     # Close the source file
10     source_file.close()
11
12     try:
13         # Open the destination file in write mode
14         destination_file = open('destination.txt', 'w')
15
16         # Write the contents to the destination file
17         destination_file.write(content)
18
19         # Close the destination file
20         destination_file.close()
21
22         print('File copied successfully.')
23
24     except PermissionError:
25         print('Permission denied to write to the destination file.')
26
27     except Exception as e:
28         print('An error occurred while copying the file:', str(e))
29
30      except FileNotFoundError:
31     print('Source file not found.')
32
33 except Exception as e:
34     print('An error occurred while accessing the source file:', str(e))
```

In the above example, we attempt to copy the contents of a source file named `source.txt` to a destination file named `destination.txt`. First, we open the source file in read mode and read its contents. Then, we close the source file. Inside the `try` block, we open the destination file in write mode and write the contents from source file to it. Finally, we close the destination file and print a *success* message if the file copying process completes without any exceptions. If an error occurs, such as the source file not being found or permission denied to write to the destination file, the corresponding exception is caught and an appropriate error message is displayed.

4.7 Finally Clause

finally clause is used to define a block of code that will always be executed, regardless of whether an exception occurred or not. The **finally block** is placed after all the **except blocks** (if any) and is optional.

Here is the general syntax of a **try...except...finally** block:

```
try:
    # Code that may raise an exception
    # ...
except ExceptionType1:
    # Code to handle ExceptionType1
    # ...
except ExceptionType2:
    # Code to handle ExceptionType2
    # ...
finally:
    # Code that always executes, regardless of exceptions
    # ...
```

As usual, the **try** block contains the code that may raise an exception and the **except** block(s) specify the type of exception(s) you want to catch and handle. Each **except** block can handle a specific exception type. In case, an event corresponding to a specified exception type occurs, the corresponding **except** block(s) is executed. Next, the **finally** block is executed regardless of whether an exception occurred or not. The **finally** block is typically used for cleanup tasks. It ensures that certain actions are always performed, regardless of how the **try** block is exited (either through an exception or normal execution).

The following code defines a function named `divide` that divides the first operand by the second operand. The `divide` function takes two arguments: `numerator` and `denominator`.

```
01 def divide(numerator, denominator):
02     try:
03         result = numerator / denominator
04         return result
05     except:
06         print('Error: Division by zero')
07     finally:
08         print('Function ends smoothly')
09 print(divide(6,0))
```

On execution of the above code, Python produces the following output:

```
Error: Division by zero
Function ends smoothly
None
```

Note that as division by zero is attempted in the above code, Python performs the following tasks:

1. As the **ZeroDivisionError** exception is raised, Python executes the corresponding except block and displays the message:

   Error: Division by zero

2. It executes the `Finally` block and displays the following message:

   Function ends smoothly

3. Since **ZeroDivisionError** exception is raised, the **return** statement within the function **divide** is not executed. Instead, the function comes to an end on the execution of the **finally** block. So, in the absence of any **return** statement that returns a particular value, Python returns **None,** which gets displayed.

   In the above code, if we replace the statement:

   print(divide(6,0))

by the statement:

print(divide(6,''))

we get the following output:

```
Function ends smoothly
Traceback (most recent call last):
  File "F:/DOWNLOAD/Python/try.py", line 11, in <module>
    print(divide(6,''))
  File "F:/DOWNLOAD/Python/try.py", line 4, in divide
    result = a / b
TypeError: unsupported operand type(s) for /: 'int' and 'str'
```

Note that even though division by a string leads to a **TypeError** exception, `finally` clause is still executed. However, due to the abrupt termination of the function, the statement

print(divide(6,''))

is not executed.

In the above code, if we replace the statement:

print(divide(6,0))

with the statement:

print(divide(6,2))

we get the following output because the program executes smoothly:

```
Function ends smoothly
3.0
```



The `finally` block is optional and is executed regardless of whether an exception occurred or not.

```
Function ends smoothly
3.0
```

In each case, the **finally block** is executed regardless of whether the exception occurred or not.

Let's Summarise

- `Exceptions` are errors or exceptional conditions that occur during the execution of a Python program, disrupting the normal flow of code.
- During the program execution, Python automatically raises exceptions, but that terminates abruptly.
- Syntax errors, also known as parsing errors, occur when there is a violation of Python grammar rules.
- Built-in exceptions in Python are predefined errors that address common issues and provide standardized solutions.
- Examples of built-in exceptions include `NameError`, `TypeError`, `ValueError`, `ZeroDivisionError`, `IndexError`, `EOFError`, `FileNotFoundError`, and `PermissionError`.
- Exception handling in Python is done using the **try...except** block. The try block contains the code that may raise an exception, and the **except** block specifies how to handle specific exception types.
- Multiple **except** blocks are used to handle different types of exceptions, each allowing for specific error handling.
- An optional **else** block can be included in the code after all the `except` block(s) are specified. The **else** block is executed if and only if no exceptions occur in the try block.
- A generic **except** block without specifying an exception type can be used to catch all possible exceptions.
- The **finally** block is optional and is executed regardless of whether an exception occurred or not.

Solved Exercises

A. Multiple Choice Questions

1. What are exceptions in Python?

a. Unusual conditions in code execution
b. Syntax errors
c. Logical errors
d. All of these

2. When does a syntax error occur?

a. During runtime
b. Due to logical errors
c. When violating Python grammar rules
d. When handling exceptions

3. Which exception occurs when a name is not found globally?

a. `ValueError` b. `TypeError` c. `NameError` d. `SyntaxError`

4. Which exception occurs when an operation is applied to an inappropriate object type?

a. `ValueError` b. `TypeError` c. `ZeroDivisionError` d. `IndexError`

5. What does the `try` block contain?

a. Code that may raise an exception
b. Code that always executes
c. Code to handle exceptions
d. Code that prevents exceptions

6. Which block in a `try...except` statement if present, is always executed?

a. `try` block b. `except` block c. `else` block d. `finally` block

7. How can we catch all possible exceptions in Python?

a. Use multiple `except` blocks for each exception type
b. Use a generic `except` block without specifying an exception type
c. Use a `finally` block
d. Use the `raise` keyword

8. Which exception occurs when we try to access an index beyond the valid range?

a. `ValueError`
b. `TypeError`
c. `ZeroDivisionError`
d. `IndexError`

9. What is the purpose of the `finally` block in exception handling?

a. To execute some code regardless of exceptions
b. To handle specific exception types
c. To prevent program termination
d. To catch all possible exceptions

10. When does an **`EOFError`** typically occur?

a. When an input function encounters a syntax error
b. When there is an error in the end-of-file character
c. When a file is not found during file handling
d. When the user prematurely ends input before providing expected input

11. What is the purpose of handling an **`EOFError`** in the code snippet?

a. To handle syntax errors in the code
b. To catch any error that occurs during file handling
c. To handle situations when a user ends input prematurely
d. To handle permission-related errors during file operations

12. What does a **`FileNotFoundError`** exception indicate?

a. A syntax error occurred while reading input
b. The program does not have the necessary permissions for file access
c. An attempt to open or access a non-existent file was made
d. The user prematurely ended the input before providing the expected input

13. When does a **`PermissionError`** occur?
    a. When there is an error in the end-of-file character
    b. When the user prematurely ends input before providing expected input
    c. When a file is not found while opening it.
    d. When the program does not have necessary permissions for file access

B. State whether the following statements are True or False:

1. Exceptions in Python are raised during program execution. _____
2. Syntax errors in Python are also known as runtime errors. _____
3. A `NameError` occurs when a variable is not found globally in Python. _____
4. A `TypeError` occurs when an operation or function is applied to an object of an inappropriate type in Python. _____
5. The `try` block in a `try...except` statement contains code that may raise an exception. _____
6. The `else` block in a `try...except` statement is executed if any exception occurs within the try block. _____
7. A generic `except` block without specifying an exception type can handle any exception that occurs within the `try` block. _____
8. An `IndexError` occurs when trying to access an index that is within the valid range of a list. _____
9. The `finally` block in exception handling is always executed regardless of whether an exception occurred or not. _____
10. The `finally` block is typically used for cleanup tasks or releasing resources in Python. _____
11. The `EOFError` occurs when the user presses the end-of-file character before providing the expected input. _____
12. The `FileNotFoundError` is raised when an attempt is made to open or access a file that doesn't exist. _____
13. The `PermissionError` is raised when there is a permission issue while accessing a file or performing a file operation. _____

C. Fill in the blanks.

1. Syntax errors in Python occur when there is a violation of _____ rules.
2. _____ represent errors or exceptional conditions that need to be handled.
3. _____ occurs when a name is not found globally.
4. _____ occurs when an operation or function is applied to an inappropriate object type.
5. The _____ block contains the code that may raise an exception.
6. The _____ block is executed if no exceptions are raised within the try block.
7. To catch all possible exceptions in Python, we can use a generic _____ block.
8. _____ occurs when we try to perform a numeric division with a zero denominator.
9. _____ occurs when we try to access an index that is out of a valid range.
10. The _____ block is used to define a block of code that will always be executed.
11. The _____ occurs when `input()` is invoked in a loop to repeatedly receive user input, and the user accidentally ends the input by pressing the end-of-file character (usually `ctrl+d`) before providing the expected input.
12. The _____ is raised when an attempt is made to open or access a file that does not exist.
13. The _____ typically occurs when the program doesn't have the necessary permission to read from or write to a file.

D. Answer the following questions:

1. What are exceptions in Python?

Ans. Exceptions in Python are events that occur during program execution, leading to the disruption of the normal flow of code. They represent errors or exceptional conditions that need to be handled appropriately.

2. What are syntax errors and when do they occur?

Ans. Syntax errors, also known as parsing errors, occur when there is a violation of Python grammar rules. They cause the interpreter to halt program execution unless the errors are fixed.

3. What is a `NameError`? Give an example.

Ans. `NameError` occurs when a name used in a statement is not found globally. For example:

```
length = LEN('Enter your marks')
```

In the above code, the name **`LEN`** is not defined in Python. The correct function name is **`len`**. Hence, it raises a **`NameError.`**

4. What is a `TypeError`? Give an example.

Ans. `TypeError` occurs when an operation or function is applied to an object of an inappropriate type. For example:

```
len(1234)
```

In the above code, the **`len`** function being applied to an integer object, leading to a `TypeError.` However, **`len('1234')`** would have been fine.

5. How can exceptions be handled in Python using the `try...except` block?

Ans. The `try...except` block allows us to catch and handle specific exceptions that may occur during the execution of code. The code that may raise an exception is placed within the `try` block, and the corresponding exception handling code is written in the `except` block.

6. What is `finally` clause in exception handling?

Ans. The `finally` clause defines a block of code that will always be executed, regardless of whether an exception occurred or not.

7. Explain the `IndexError` and provide an example.

Ans. `IndexError` occurs when an attempt is made to access an index that is out of the valid range. For example:

```
colors = ['red', 'green', 'blue', 'orange']
colors[4]
```

In the above code, the list `colors` has three elements, so the valid indices are 0, 1, 2, and 3. Accessing `colors[4]` results in an `IndexError` because there is no element at that index.

8. How can a generic `except` block be used to catch all exceptions?

Ans. A generic `except` block without specifying a particular exception type can be used to catch all exceptions. It acts as a catch-all for any exception that occurs within the try block.

9. What is the purpose of the else block in a `try...except` statement?

Ans. The `else` block in a `try...except` statement is optional and is executed if no exceptions are raised within the `try` block. It allows the execution of specific code when no exceptions occur.

10. What is the role of the `finally` block in exception handling?

Ans. The `finally` block is executed regardless of whether an exception occurred or not. It ensures that certain actions are always performed, regardless of how the `try` block is exited, whether through an exception or normal execution.



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion: Exception handling in Python allows developers to manage and respond to errors gracefully.

   Reasoning: When an exception occurs, Python transfers the program execution to an exception handling block, where the exception can be captured and managed effectively, ensuring that the program does not terminate abruptly.

2. Assertion: Syntax errors in Python are also known as parsing errors.

   Reasoning: When a syntax error is encountered in a Python program, the Python interpreter promptly detects and reports it, along with an explanation of the error and suggestions for resolution.

3. Assertion: The `finally` block is optional and is executed regardless of whether an exception occurred or not.

   Reasoning: The `finally` block is typically used for cleanup tasks and ensures that certain actions are always performed, regardless of how the try block is exited.

Ans. 1. a 2. a 3. a



Assessment

A. Multiple Choice questions

1. What is the primary purpose of exception handling in Python?
   a. To intentionally terminate the program
   b. To hide error messages from the user
   c. To gracefully manage and respond to errors
   d. To increase the complexity of code

2. Which of the following is an example of a syntax error?
   a. `ValueError`
   b. `IndexError`
   c. `IndentationError`
   d. `KeyError`

3. What type of exception is raised when trying to access an index that is out of the valid range in a list?
   a. `NameError` b. `IndexError` c. `ValueError` d. `TypeError`

4. Which built-in exception is raised when attempting to perform a numeric division with a denominator of zero?
   a. `TypeError`
   b. `ValueError`
   c. `ZeroDivisionError`
   d. `IndexError`

5. What is the primary purpose of the `finally` block in a `try…except…finally` construct?
   a. To catch and handle exceptions
   b. To define cleanup tasks that always execute
   c. To specify the type of exception to catch
   d. To terminate the program abruptly

6. When handling multiple exceptions, which block is executed if no exceptions occur within the try block?
   a. `except` block b. `else` block c. `finally` block d. `try` block

7. What does the `try` block contain in a `try…except` construct?
   a. Cleanup code
   b. Code that may raise an exception
   c. Code that always executes
   d. Code to specify the exception type

8. Which exception is raised when a program reaches the end of a file while trying to read input from the user?
   a. `FileNotFoundError`
   b. `PermissionError`
   c. `EOFError`
   d. `ValueError`

9. Which of the following is NOT a benefit of exception handling in Python?
   a. Providing informative error messages
   b. Preventing the program from terminating abruptly
   c. Increasing program complexity
   d. Allowing graceful recovery from errors

10. What is the primary advantage of using `try…except` blocks for handling exceptions?
    a. To make the code more readable
    b. To ensure that no exceptions ever occur
    c. To catch and handle exceptions gracefully
    d. To create complex error messages

B. State whether the following statements are True or False:

1. Exception handling is a feature of Python that allows developers to manage and respond to errors or exceptional situations gracefully. _____
2. Syntax errors in Python are runtime errors that can be caught and handled using exception-handling techniques. _____
3. Built-in exceptions in Python are predefined error situations that address common issues and provide standardized solutions. _____
4. The `IndexError` exception is raised when division by zero is performed. _____
5. The purpose of the `FileNotFoundError` exception is to handle cases when a file is not found during file operations. _____

6. The `try` block contains the code that may raise an exception, and the `except` block specifies how to handle specific exception types. _____
7. The `finally` block is optional and is executed only if an exception occurs in the `try` block. _____
8. The `finally` block is used for cleanup tasks and is executed regardless of whether an exception occurred or not. _____
9. The `except` block, without specifying an exception type, acts as a catch-all for all exceptions that may occur. _____
10. Exception handling in Python helps improve user experience by providing informative error messages and enabling graceful recovery from exceptional situations. _____

C. Fill in the blanks.

1. Missing parentheses and _____ are examples of syntax errors in Python.
2. The _____ exception is raised when an operation or function is applied to an object of inappropriate type.
3. The _____ exception occurs when we try to perform a numeric division in which the denominator is zero.
4. The _____ exception is raised when we try to access an index that is out of a valid range in a sequence, such as a list or a string.
5. The _____ exception is raised when an input function, such as `input()`, reaches the end of a file while trying to read input from the user.
6. When handling multiple exceptions in a `try...except` block, the _____ block is optional and is executed if no exceptions are raised within the try block.
7. The _____ block is typically used for cleanup tasks and ensures that certain actions are always performed, regardless of how the try block is exited.

D. Answer the following questions:

1. What are exceptions in Python, and what is their purpose?
2. What are the advantages of incorporating exception handling mechanisms in programs.
3. What are built-in exceptions in Python? How do they help in handling common errors?
4. Describe the purpose of the following exceptions: `NameError`, `TypeError`, `ValueError`, `ZeroDivisionError`, and `IndexError`.
5. How does the `try...except` block work in Python? Illustrate with an example.
6. What is the purpose of the `finally` clause in a `try…except…finally` block? How does it ensure code execution?
7. Write a code snippet that raises a `NameError` exception and explain how you would handle it.
8. Give an example of a `try…except` block to handle a `TypeError` exception when concatenating a string with an integer.
9. Write a program that prompts the user to enter a number and handles a `ValueError` exception if the input is not a valid integer.
10. Write a code snippet that performs division between two numbers and handles a `ZeroDivisionError` exception.
11. Develop a program that opens a file and handles both `FileNotFoundError` and `PermissionError` exceptions.
12. Write a function that takes a numerator and a denominator as arguments and handles any exceptions that may occur during division.



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion: `ZeroDivisionError` occurs when division by zero is attempted in Python.
   Reasoning: Division by zero is not allowed because it results in undefined behavior in mathematics.

2. Assertion: The `try` block contains the code that may raise an exception, and the except block specifies how to handle specific exception types.

   Reasoning: Exception handling allows for controlled management of runtime errors, avoiding abrupt program terminations.

3. Assertion: Exception handling provides valuable information about the cause and location of errors.

   Reasoning: Exception messages generated by Python are informative and aid developers in understanding the nature of the problem.



Previous Years' Questions

Why it is important to close a file before exiting?

Write a program in Python that defines and calls the following user defined functions :

(i) Add_Book(): Takes the details of the books and adds them to a csv file 'Book.csv'. Each record consists of a list with field elements as book_ID, B_ name and pub to store book ID, book name and publisher respectively.

(ii) Search_Book() : Takes publisher name as input and counts and displays number of books published by them. [2023]

Ans. Files are a limited resource managed by the OS, so it is important to close them to avoid reaching the open limit and impacting program performance. Files left unintentionally open become vulnerable to loss of data, and changes to files cannot be readable until closed.

```
import csv
def Add_Book():
  f=open("book.csv","a",newline='')
  wo=csv.writer(f)
  book_id=input("Enter Book ID:")
  b_name=input("Enter Book Name:")
  pub=input("Enter Publisher:")
  wo.writerow([book_id,b_name,pub])
  f.close()

def Search_Book():
  f=open("book.csv",'r')
  ro=csv.reader(f)
  pn=input("Enter Publisher:")
  cnt=0
  for i in ro:
    if i[2]==pn:
      cnt+=1
      print("Book ID:",i[0])
      print("Book:",i[1])
      print("Publisher:",i[2])
  print("Total Books Published by ",pn," are:",cnt)
Add_Book()
Search_Book()
```



NCERT Exercise Solutions

1. "Every syntax error is an exception, but every exception cannot be a syntax error." Justify the statement.

Ans: Syntax errors occur when there is a violation of the rules (grammar) of the programming language (Python, in our case). In the event of a syntax error in a Python program, the Python interpreter promptly detects and reports it, along with an explanation of the error and suggestions for resolution. However, exceptions cover a wider range, including runtime errors unrelated to syntax. Thus, while syntax errors are a subset of exceptions, not all exceptions arise from syntax errors; they can stem from various runtime issues.

2. When are the following built-in exceptions raised? Give examples to support your answers.
   a. ImportError
   b. IOError
   c. NameError
   d. ZeroDivisionError

Ans. a. `ImportError` exception is commonly encountered when attempting to import a module that is not installed or if there are problems with the module's code. To handle `ImportError` in Python, you can use a try-except block. For example, importing `sklearn` without installing the `scikit-learn` will result in `ModuleNotFoundError`.

```
>>> import sklearn
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    import sklearn
ModuleNotFoundError: No module named 'sklearn'
```

In Python 3.6 and later versions, the `ImportError` exception was narrowed down, and the more specific `ModuleNotFoundError` was introduced. This means that when a module cannot be found during an import, Python raises a `ModuleNotFoundError` rather than a generic `ImportError`.

b. `IOError` is an exception that is raised when an input/output operation (I/O) fails or encounters an issue. For better clarity and precision in handling various I/O-related issues, Python 3 has replaced the `IOError` with more specific exceptions such as `OSError`, `FileNotFoundError`, and `PermissionError`. For example, an attempt to open a non-existent file, `abc.txt`:

```
>>> f = open ("abc. txt")
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    f = open ("abc. txt")
FileNotFoundError: [Errno 2] No such file or directory: 'abc. txt'
```

c. Refer to section 4.3: Built-in Exception

d. Refer to section 4.3: Built-in Exception

3. What is the use of a raise statement? Write a code to accept two numbers and display the quotient. An appropriate exception should be raised if the user enters the second number (denominator) as zero (0).

Ans. The raise statement is used to raise an exception. You can deal with bugs or uncommon situations in your code by using exceptions. You can raise exceptions when something unexpected or incorrect occurs and you wish to handle the situation in a specified way.

The following program takes two numbers as input, displays the quotient, and raises the appropriate exception if the second number (denominator) is given as zero:

```
try:
    numerator = int(input("Enter the numerator: "))
    denominator = int(input("Enter the denominator: "))
    if denominator == 0:
          raise ZeroDivisionError("Denominator cannot be zero. ")
    quotient = numerator // denominator
    print("Quotient: ", quotient)
except ZeroDivisionError as e:
    print("Error:", e)
```

4. Use assert statement in Question No. 3 to test the division expression in the program.

Ans. The `assert` statement allows you to check a program condition and raise an `AssertionError` if it is not satisfied. You may confirm that the division expression in this situation yields the appropriate outcome using the assert statement. For example, the `assert` statement may be used to validate that the `denominator` is non-zero. But if the denominator is zero, "`Invalid Operation`" should be printed.

```
numerator = int(input("Enter the numerator: "))
denominator = int(input("Enter the denominator: "))
```

```
assert denominator != 0, "Invalid Operation"
print(numerator//denominator)
```

5. Define the following:

a. Exception Handling

b. Throwing an exception

c. Catching an exception

Ans. a. Refer to Section 4.4: Handling Exceptions using try...except

b. Throwing an exception, also known as raising one, is the intentional generation of an exception to demonstrate that an erroneous or unexpected event has happened within a program. When an exceptional state is discovered, the application creates an exception object, which contains information about the error or uncommon event. After that, the program is "notified" of the error by being "thrown" or "raised" using this exception object.

c. When an exception occurs inside a `try` block, the program looks for an appropriate except block that can handle that specific type of exception. If a match is found, the code in the `except` block is executed, allowing you to take the appropriate action, such as reporting the problem by displaying an appropriate message or fixing the error.

6. Explain catching exceptions using try and except block.

Ans. Refer to Section 4.4: Handling Exceptions using try...except

7. Consider the code given below and fill in the blanks.

```
print (" Learning Exceptions...")
try:
    num1= int(input ("Enter the first number"))
    num2=int(input("Enter the second number"))
    quotient=(num1/num2)
    print ("Both the numbers entered were correct")
except ___________ : # to enter only integers
    print (" Please enter only numbers")
except ___________ : # Denominator should not be zero
    print(" Number 2 should not be zero")
else:
    print(" Great .. you are a good programmer")
___________ : # to be executed at the end
    print(" JOB OVER... GO GET SOME REST")
```

Ans.

```
print (" Learning Exceptions...")
try:
    num1= int(input ("Enter the first number"))
    num2=int(input("Enter the second number"))
    quotient=(num1/num2)
    print ("Both the numbers entered were correct")
except ValueError: # to enter only integers
    print (" Please enter only numbers")
except ZeroDivisionError: # Denominator should not be zero
    print(" Number 2 should not be zero")
else:
    print(" Great .. you are a good programmer")
finally: # to be executed at the end
    print(" JOB OVER... GO GET SOME REST")
```

8. You have learnt how to use math module in Class XI. Write a code where you use the wrong number of arguments for a method (say `sqrt()` or `pow()`). Use the exception handling process to catch the `ValueError` exception.

Ans. Python ' s ValueError exception can be used to notice errors when you use the incorrect number of parameters for functions like `sqrt()` or `pow()`.

Here is an example that demonstrates how to do this:

```
import math
try:
    res = math.sqrt(9, 2)
except ValueError as e:
    print(f"Caught a ValueError: {e}")
except Exception as e:
    print(f"Caught an exception: {e}")
else:
    print(f"Result: {res}")
```

9. What is the use of finally clause? Use finally clause in the problem given in Question No. 7.

Ans. When handling exceptions, finally clause is used to specify a block of code that must be executed whether or not an exception is thrown. It is generally used for actions that must be taken as part of cleanup or finalization, such as freeing up resources, closing files, or guaranteeing that certain actions are taken regardless of what occurs in the `try` and except blocks that come before them.

```
print ("Learning Exceptions...")
try:
    num1 = int(input ("Enter the first number"))
    num2 = int(input("Enter the second number"))
    quotient = (num1 / num2)
    print ("Both the numbers entered were correct")
except ValueError: # to enter only integers
    print (" Please enter only numbers")
except ZeroDivisionError: # Denominator should not be zero
    print(" Number 2 should not be zero")
else:
    print(" Great .. you are a good programmer")
finally: # to be executed at the end
    print(" JOB OVER... GO GET SOME REST")
```

Answers

Multiple Choice Questions

1. (a)	2. (c)	3. (c)	4. (b)	5. (a)	6. (c)	7. (b)	8. (d)
9. (a)	10. (d)	11. (c)	12. (c)	13. (d)			

True or False

1. (T)	2. (F)	3. (T)	4. (T)	5. (T)	6.(F)	7. (T)	8. (F)
9. (T)	10.(T)	11.(T)	12. (T)	13. (T)			

Fill in the blanks

1. grammar	2. Exceptions	3. `NameError`	4. `TypeError`	5. `try`
6. `else`	7. `except`	8. `ZeroDivisionError`	9. `IndexError`	10. `finally`
11. `EOFError`	12. `FileNotFoundError`	13. `PermissionError`		

5 DATA STRUCTURES: STACKS

Chapter Outline

5.1 Stacks

5.2 Applications of Stacks

Introduction

The phrase *data structures* comprises two words, *data* and *structures*. The term data refers to facts and statistics relating to something of interest. Similarly, the word *structure* refers to a formation, an organisation, an arrangement, a collection, an entity, etc. The Oxford dictionary defines *'structure'* as *the arrangement of and relations between the parts or elements of something complex.* Thus, when we talk about a data structure, we are interested in knowing how data is organised (or arranged) as a collection or an entity. Different organisations of data enable different operations on it. Thus, a data structure is an arrangement of data along with the associated operations. A specific organisation of data (such as a stack, to be discussed in this chapter) leads to operations specific to that organisation. In this chapter, we will discuss the data structure called 'stacks' and their implementation in Python using lists.

5.1 Stacks

All of us are familiar with stacks. When we go to a party, we find a stack of plates. Every guest picks up the top plate from the stack for eating, and the stack size keeps reducing. Again, when guests finish eating, they stack the used plates in a basket. Subsequently, a cleaner picks up the plates from the stack of used plates one by one, cleans each plate, and makes another stack of clean plates. We note that a discarded plate is always put on top of an existing stack of used plates (initially empty when no one has yet discarded a plate after eating), and the cleaner always picks up the top plate from the stack of used plates for cleaning. Similarly, we note that the cleaner always puts a cleaned plate on top of a stack (initially empty) of cleaned plates. Further, a guest also picks up the top plate from the stack of cleaned plates for eating. The key point is that the plates are picked up from a stack in reverse order of their placement on the stack. Next, let us consider the example of a library. When the library users finish reading books, they stack them on a table before leaving the library. Later, a library staff member picks up books from the top of the stack one by one and puts them on the shelves in the right place. Thus, the books are picked up from the stack in reverse order of their placement on the stack. Thus, a stack is a *Last In, First Out* (LIFO) arrangement. Now we understand that stacks find multiple uses in our daily life.



Fig 5.1 : Stack of Plates and Books

Stack is a Last In First Out (LIFO) arrangement.

What is a stack?

5.1.1 Uses of Stacks

Organising data as stacks is also useful in several computer science applications. For example,

- **Editing a Document:** While editing a document, *undo* and *redo* operations are applied in a stacked manner. For example, when we perform a sequence of edit operations, they are stacked on one another. Subsequently, when we apply an *undo* operation, the last edit operation is picked up from the stack, and it is undone. The sequence of *undo* operations is also stacked. Subsequently, if we realise that the *undo* operations were carried out erroneously, we can recover from the error by performing a sequence of *redo* operations.
- **Browsing on the Internet:** While browsing the web pages on Internet, maintaining the navigation history as a stack enables users to move back and forth between the recently visited web pages. For example, suppose a user visited web pages P1, followed by P2, and is currently viewing page P3. These page references (excluding the current page) are stored on the top of one another (P2→P1), allowing a user to go to the previous page by clicking on the BACK button in the browser, which retrieves P2 stored at the top of the stack.
- **Parenthesis Matching:** For expression evaluation, the compiler uses a stack to ensure that the expression is correctly parenthesised, i.e. for every opening parentheses, there is a matching closing parenthesis.
- **A Sequence of Function Calls:** When a function calls another function (or itself), the sequence of calls is maintained as a stack. When a function call terminates, the system can retrieve the position from where the execution is to begin.

Give any three applications of stacks.

5.1.2 Stack Operations

As discussed above, a stack is a Last In, First Out arrangement of objects. So, given a stack, we may perform the following operations:

- **Push:** *Push* operation places an element on the top of the stack. Sometimes, *push* operation is also called the insertion of an element on top of a stack.
- **Pop:** *Pop* operation removes the top element from the stack. Pop operation is also called the deletion of the top element of the stack.

Push: Insertion at the top of the stack.
Pop: Deletion from the top of the stack.

Note that to perform push and pop operations in a stack, we only need to access the top of the stack. Indeed, at any point in time, only the `TOP` element of the stack is accessible. The *push* and *pop* operations on the stack may result in the following scenarios:

- **Stack Overflow Condition:** A stack is full when no more elements can be added to it. In Python, a stack is often implemented using a list. So, Python will complain that the stack is full when no system memory is available to append elements to the list. An attempt to push an element when the stack is already full results in a stack overflow condition. For simplicity, we will assume that the system has sufficient memory for all the elements that need to be pushed on the stack.
- **Stack Underflow Condition:** A stack having no elements is called an empty stack. An attempt to pop an element from the empty stack results in a stack underflow condition.

Stack Overflow: Attempt to push on a stack that is already full.
Stack Underflow: Attempt to pop from empty stack.

Consider an empty stack and the following sequence of operations:

1. PUSH B1
2. PUSH B2
3. PUSH B3
4. POP
5. PUSH B4
6. POP
7. POP
8. POP
9. POP

Fig 5.2 shows the contents of the stack after each of the operations mentioned above. The stack becomes empty on executing the POP operation (Sr. No. 8). So, the next pop operation (Sr. No. 9) results in the stack underflow condition.

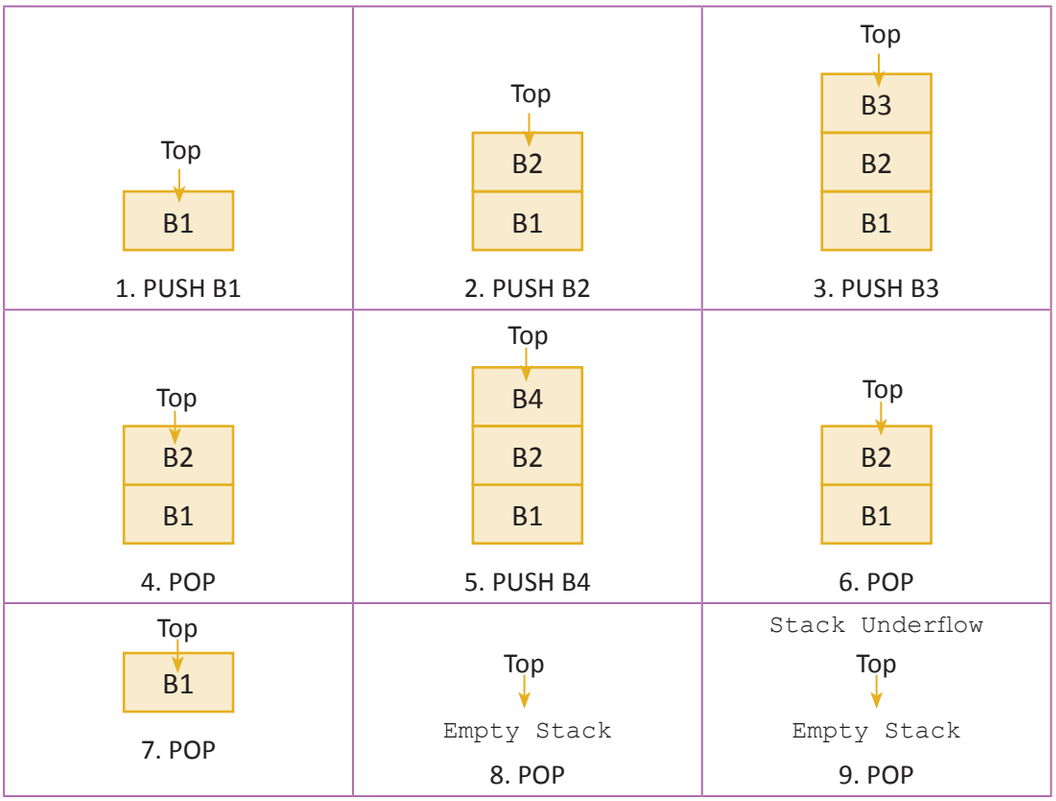


Fig 5.2: Stack Operations

When we are done with all the operations on a stack, we should delete it so that the memory occupied by it is released. Before executing a `popStack()` operation on a stack, we must ensure that the stack is not empty. The `isEmpty()` function achieves precisely this by returning `True` when the stack is empty and `False` otherwise. In this section, we will study an implementation of stacks, which includes the following user-defined functions:

- **`createStack()`:** to create an empty stack
- **`isEmpty()`:** to check whether the stack is empty or not
- **`pushStack()`:** to push an element on top of the stack
- **`popStack()`:** to delete the top element of the stack
- **`top()`:** to view the top element of the stack
- **`size()`:** to count the number of elements in the stack
- **`displayStack()`:** to display the elements of the stack

1. createStack()

While performing operations on a stack, we assumed that a stack already existed. However, the stack creation operation **`createStack()`** must precede all operations on a stack. As the stack is to be implemented as a list, the creation of a stack only requires an empty list to be created as shown below:

```
>>> def createStack():
...     '''
...     Objective: To create an empty stack
...     Input Parameter: None
...     Return Value: An empty list denoting empty stack
...     '''
...     stack = list()
...     return stack
```

To create a stack, we invoke the function `createStack()`:

```
>>> stack = createStack()
```

2. isEmpty()

The function `isEmpty()` takes a stack object as an input parameter and checks whether it is empty. As an empty stack has zero elements, to test whether a stack is empty, we check whether the length of the list denoting the stack is zero, i.e. whether `len(stack) == 0`. If the condition evaluates to `True`, the function returns `True`, indicating the stack is empty, and `False` otherwise, as shown below:

```
>>> def isEmpty(stack):
...     if len(stack) ==0:
...         return True
...     else:
...         return False
```



Can you rewrite the above code without using if-else statement?



3. pushStack()

The function `pushStack()` accepts two arguments: a stack object and an element to be pushed on the stack. It appends the new element (to be pushed on the stack) to the list representing the stack as shown below:

```
>>> def pushStack(stack, element):
...     stack.append(element)
```

4. popStack()

The function `popStack()` takes a stack object as an argument, for example, `popStack(stack)`. To execute a pop operation, we must ensure that the stack is not empty. So, we use the function call `isEmpty(stack)` to check whether the stack is empty. If the stack is non-empty, the function call `stack.pop()` returns the element appended last to the list `stack` and removes it from the list. Otherwise, the function displays the message 'Stack Underflow' and returns `None`.

The following code statement achieves a pop operation on the stack.

```
>>> def popStack(stack):
...     if not(isEmpty(stack)):
...         return stack.pop()
...     else:
...         print('Stack Underflow')
...         return None
```



What is the effect of replacing `stack.pop()` in the above code with `stack.pop(-1)`?



5. top()

Sometimes, we need to examine the top element of the stack without removing it from the stack. The function `top()` yields the top element of the stack without deleting it. The function `top()` takes a stack object as an input argument and invokes the `isEmpty()` function to check whether the stack is empty. If the stack is not empty, it returns the top element of the stack by accessing index –1 of the list. Otherwise, the function prints the 'Stack Empty' message and returns `None`:

```
>>> def top(stack):
...     if not(isEmpty(stack)):
...         return stack[-1]
...     else:
...         print('Stack Empty')
...         return None
```

6. size()

Sometimes, one may be interested to know the current size of the stack. The function `size()` returns the size of the stack. It takes the `stack` object as an input argument. It returns the number of elements in the stack by invoking the `len` function on the associated list as shown below:

```
>>> def size(stack):
...     return len(stack)
```

7. displayStack()

As we always access the top element of the stack, displaying the entire stack is not a standard stack operation. However, you may like to display the entire stack for checking whether your stack implementation is working properly. Once you are sure of your implementation, you will not require it for the application programs that use a stack. The function `displayStack()` takes a stack object as an input argument and displays the contents of the stack starting from the TOP of the stack.

```
>>> def displayStack(stack):
...     for element in reversed(stack):
...         print(element)
```

5.1.3 Deleting a Stack

To delete a stack, we only need to invoke the `del` operator as shown below:

```
>>> del stack
```

Now, we are ready to present a complete program for implementing stacks (see Program 5.1, (`myStack.py`). We have not provided a function to check whether a stack is full because the append function for Python lists never fails as long as enough storage is available to the application program. In the Program 5.1, we define the above-mentioned stack operations using inbuilt list operations `append` and `pop`.

Program 5.1 Stack Operations (myStack.py)

```
01 def createStack():
02   '''
03   Objective: To create an empty stack
04   Input Parameter: None
05   Return Value: An empty list denoting empty stack
06   '''
07   stack = list()
08   return stack
09
10 def pushStack(stack, element):
11   '''
12   Objective: To push an element at the top of the stack,
13   Input Parameter:
14       stack - input stack
15       element - value to be inserted
16   Return Value: None
17   '''
18   stack.append(element) # The list denoting the stack is updated.
19
20 def isEmpty(stack):
21   '''
22   Objective: To determine if the stack is empty
23   Input Parameter:
24       stack - input stack
25   Return Value: True if the stack is empty else False
26   '''
27   if len(stack) == 0:
28       return True
29   else:
30       return False
31
32 def popStack(stack):
33   '''
34   Objective: To remove an element from the top of stack
35   Input Parameter:
36       stack - stack data structure
37   Return Value: Top element of the stack if stack is not empty
38                 else None
39   '''
40   if not(isEmpty(stack)):
41       return stack.pop() #The list denoting the stack is updated.
42   else:
43       print('Stack Underflow')
44       return None
45
46 def top(stack):
47   '''
```

```
  Objective: To return topmost element of the stack
  Input Parameter:
      stack - stack data structure
  Return Value: Top element of the stack if stack is not empty
                else None
  '''
  if not(isEmpty(stack)):
      return stack[-1]
  else:
      print('Stack Empty')
      return None

def size(stack):
  '''
  Objective: To return the number of elements in the stack
  Input Parameter:
      stack - stack data structure
  Return Value: number of elements in stack
  '''
  return len(stack)

def displayStack(stack):
  '''
  Objective: To print contents of the stack
  Input Parameter: stack - stack data structure
  Return Value: None
  '''
  for element in reversed(stack):
      print(element)

#main program segment
"""
Objective: To provide stack functionality
User Interface:
    1. User is asked to enter:
         stack operation to be performed
    2. This is repeated so long as the
       user wants to continue.
"""

while True:
      print('\nChoose the stack operation to be performed')
      print('1: Create stack')
      print('2: Delete stack')
      print('3: Push')
      print('4: Pop')
      print('5: Print Stack Data')
      print('6: Top element')
      print('7: No. of elements')
      print('0: Exit')
      choice = int(input('Enter Choice: '))
      if choice == 1:
          stack = createStack()
          print('Stack Created')
      elif choice == 2:
          del stack
          print('Stack Deleted')
      elif choice == 3:
          element = int(input('Enter element to be inserted: '))
```

```
            pushStack(stack, element)
        elif choice == 4:
            print('Element poped:',popStack(stack))
        elif choice == 5:
            displayStack(stack)
        elif choice == 6:
            print('Topmost element',top(stack))
        elif choice == 7:
            print('No. of elements', size(stack))
        elif choice == 0:
            break
        else:
            print("Invalid Choice!")
```

On executing the Program 5.1, Python responded with the following output:

```
Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 1
Stack Created
```




```
Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 3
Enter element to be inserted: 10
```




```
Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 3
Enter element to be inserted: 20
```




```
Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
```



```
0: Exit
Enter Choice: 3
Enter element to be inserted: 30

Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 7
No. of elements 3
Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 6
Topmost element 30

Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 5
30
20
10

Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 4
Element poped: 30

Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
```



```
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 6
Topmost element 20

Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 3
Enter element to be inserted: 40
```




```
Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 5
40
20
10

Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 4
Element poped: 40

Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 4
Element poped: 20
```





```
Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 4
Element poped: 10
```




```
Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 4
Stack Underflow
Element poped: None
```

```
Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 5
```

```
Choose the stack operation to be performed
1: Create stack
2: Delete stack
3: Push
4: Pop
5: Print Stack Data
6: Top element
7: No. of elements
0: Exit
Enter Choice: 0
```

5.2 Applications of Stacks

There are several applications of stacks. Let us study about some of them.

5.2.1 Evaluating Arithmetic Expressions

The most common notation used for representing an expression is the infix notation. An infix expression comprises a sequence of operands separated by operators. Pairs of parentheses are used to enforce the desired order of evaluation. For example, consider the following expression:

```
a + (b - c) * (d + (e + f) * g)
```

Fig 5.3 shows the order of evaluation for the above-mentioned expression

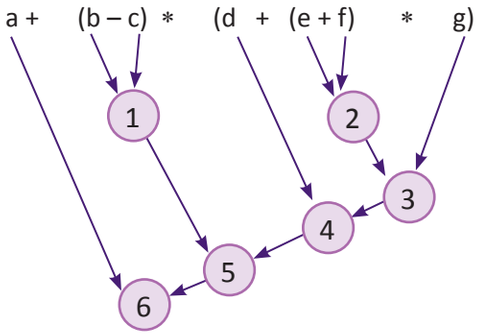


Fig 5.3: Evaluation of expression a + (b - c) * (d + (e + f) * g)

To correctly evaluate the expression, we must keep track of the precedence of operators and matching parenthesis. Often, we need to scan parts of an expression numerous times while evaluating it. The postfix notation (also called postfix form) is another way of expressing arithmetic expressions while eliminating the need to scan parts of an arithmetic expression repeatedly while evaluating it.

Infix Notation: Operands separated by operators
Postfix Notations: Operators appear after the operands

Postfix Notation

In the postfix notation, operators appear after the operands in the expression. For example, consider the infix expression: **`a + b`**. In the postfix form, this expression would be written as **`a b +`**. Next, let us consider another expression: **`(a + b) * c`**. It requires **`(a + b)`** to be multiplied by **`c`**. So, to write the postfix form of **`(a + b) * c`**, we perform the following steps:

1. write the postfix form of **`(a + b)`**, i.e., **`a b +`**
2. write the next operand, i.e., **`c`**
3. write the infix operator between **`(a + b)`** and **`c`** operator, i.e., **`*`**

Thus, the postfix of the expression **`(a + b) * c`** is **`a b + c *`**. Before delving into the specifics of converting expressions from infix form to postfix form, we will learn to evaluate the expressions in postfix. We will assume that we are dealing with binary operators only, even though the case of the unary operator is not significantly difficult. A postfix expression is evaluated using a stack while scanning it from left to right:

1. Initialise an empty stack.
2. On encountering an operand, push it on the stack.
3. On encountering an operator, the two operands are popped off the stack, one by one. The operator is applied on the operands (just popped off), and the result of computation is pushed on the stack as shown below:

```
operand2 = popStack(stack)
operand1 = popStack(stack)
result = operand1 <operator> operand2
pushStack(stack, result)
```

Why don't we call the operand which is popped first **`operand1`**, and the one which is popped next **`operand2`**?

Let us now consider the evaluation of postfix expression: **`a b + c *`**, where the variables **`a`**, **`b`**, and **`c`** take values **`2`**, **`3`**, and **`4`**, respectively. Fig 5.4 illustrates how the contents of the stack change as the expression is scanned and evaluated left to right.

When the entire expression has been scanned left to right, the stack contains only one operand, i.e., the result of evaluating the postfix expression, which is finally popped off from the stack.

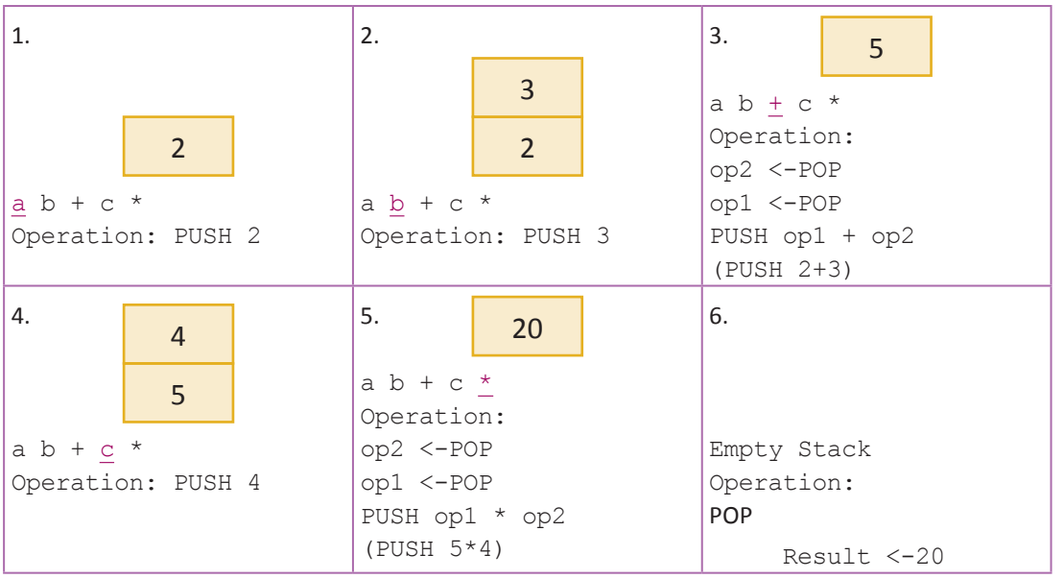


Fig 5.4: Evaluation of postfix expression: `a b + c *`

While evaluating postfix expression, on encountering operator, operands from top of the stack are popped and operator is applied on them.

Question: Evaluate the following postfix expression showing stack status at every step.

`10, 20, +, 15, 5, /, 50, -, +`

Program 5.2 Postfix Evaluation (postfixEval.py)

```
01 def evalPostfixExp(expr):
02     '''
03     Objective: To evaluate a postfix expression
04     Input Parameter: expr - string (Postfix Expression)
05     Return Value: result - integer (Result of Evaluation)
06     Assumption: Expression is correctly formed
07                 Allowed operators: binary
08                 Allowed operands: single digit
09     '''
10     stack = createStack()
11     operators = ['+', '-', '*', '/']
12     for symbol in expr:
13         if symbol in operators:
14             operand2 = popStack(stack)
15             operand1 = popStack(stack)
16             if symbol == '+':
17                 result= operand1 + operand2
18             elif symbol == '-':
19                 result= operand1 - operand2
20             elif symbol == '*':
21                 result= operand1 * operand2
22             elif symbol == '/':
23                 result= operand1 / operand2
24             pushStack(stack, result)
25         else:   #Symbol is an operand
26             pushStack(stack, int(symbol))
```

```
    result = popStack(stack)
    return result

#main program segment
"""
Objective: To evaluate postfix expression
User Interface:
    1. User is asked to enter:
         postfix expression
    2. User is presented with the result of evaluation
"""

expr = input('Enter a postfix expression: ')
result = evalPostfixExp(expr)
print("Result of evaluation of expression:",result)
```

On executing the Program 5.2, Python responded with the following output:

```
>>> Enter a postfix expression: 23+4*
>>> Result of evaluation of expression: 20
```

Conversion of an Infix Expression to Postfix Form

Having seen the advantage of the postfix form of an arithmetic expression over the infix form in evaluating the expression, it is time to discuss how to convert an infix expression to postfix form. We will use a stack for this purpose. Let us first consider an example infix expression: **a + b / c**. As division has higher precedence than addition, the expression may be written as **a b c / +** in postfix notation. Recall that for evaluating a postfix, it is scanned left to right and the operators appear in a postfix expression in the order in which they are to be applied. Thus, the expression **a b c / +** will get evaluated as **a + (b / c)**. Note that the **/** operator appears before the **+** operator in the expression **a b c / +**, so the **/** operator is applied before the **+** operator division when it is evaluated using a stack. It is easy to note that the expression **a b + c /** will not be a correct postfix representation since it will be evaluated as **(a + b) / c**. For implementing the infix to postfix conversion, as we scan the expression from left to right, operands are appended to the result string and operators are dealt with using a stack. Next, we provide the rules for processing the current symbol for converting an infix expression to its postfix form:

```
1. if symbol is an operand, append the symbol to resultString
2. Otherwise, if symbol  is an operator
   a. if precedence(symbol) > precedence(top(stack)), Push Symbol onto the Stack
   b. Otherwise
      b. 1 While notEmpty(stack) and precedence(symbol) <= precedence(top(stack) ,
           Pop operator from top of the stack and append the operator to resultString
      b. 2 Push Symbol onto the Stack
3. if stack is not empty, pop operator from top of the stack one by one and append the popped
   operator to resultString
```

OR

```
resultString = ''
for each symbol in the scanned expression:
if symbol == operand
Append the symbol to resultString
Otherwise, if symbol == operator
if precedence(symbol) > precedence(top(stack))
Push Symbol onto the Stack
Otherwise
While precedence(symbol) <= precedence(top(stack)
Pop operator from top of the stack
Append the operator to resultString
Push Symbol onto the Stack
if isEmpty(stack) != True
Pop operator from top of the stack
Append the operator to resultString
```

Pseudocode: Infix to Postfix Conversion

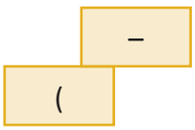
Note that we have not considered parentheses yet while dealing with infix expressions. As the parenthesis have the highest precedence, the sub-expression within the opening '('and closing ')' parenthesis should be evaluated first. Thus, as soon as the opening parenthesis '(' is found while scanning the infix expression, it should be added to the top of the stack. Similarly, when closing parenthesis is encountered, all the operators on the stack should be popped till the first opening parenthesis '(' and should be appended to the result string. The left parenthesis is no more required now and is therefore popped off the stack. We present the above-mentioned rule in the form of pseudocode:

```
1. if symbol == operand, Append the symbol to resultString
2. Otherwise, if symbol is '(', Push symbol onto the stack
2. Otherwise, if symbol is operator
   a. if precedence(symbol) > precedence(top(stack)), Push Symbol onto the Stack
   b. Otherwise
      b.1 While notEmpty(stack) and
          precedence(symbol) <= precedence(top(stack),
          Pop operator from top of the stack and append the operator to resultString
      b.2 Push Symbol onto the Stack
3. Otherwise if symbol == ')',
   While top(stack) != '(',
   Pop operator from top of the stack and append the operator to resultString
   Pop operator from top of the stack
4. if stack is not empty, pop operator from top of the stack one by one and append the popped
   operator to resultString
```

Let us now consider the conversion of postfix expression: **(7–5)*(6+4)** using stack (see Fig 5.5). At every step, Fig 5.5 illustrates the input symbol being scanned, operation performed, and the postfix expression corresponding to the infix expression scanned so far.

<table>
<tr>
<td> 

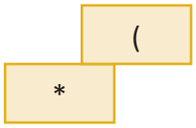
1. (7–5)*(6+4)
Symbol: (
Operation: PUSH (
Postfix expression:</td>
<td> 

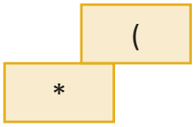
2. (7–5)*(6+4)
Symbol: 7
Operation: Append to Result
Postfix expression: 7</td>
<td> 

3. (7–5)*(6+4)
Symbol: –
Operation: PUSH –
Postfix expression: 7</td>
</tr>
<tr>
<td> 

4. (7–5)*(6+4)
Symbol: 5
Operation: Append to Result
Postfix expression: 75</td>
<td> 

5. (7–5)*(6+4)
Symbol:)
Operation: Pop and Append to Result
Postfix expression: 75-</td>
<td>Empty Stack
6. (7–5)*(6+4)
Symbol:)
Operation: Pop
Postfix expression: 75-</td>
</tr>
<tr>
<td> 

7. (7–5)*(6+4)
Symbol: *
Operation: PUSH *
Postfix expression: 75-</td>
<td> 

8. (7–5)*(6+4)
Symbol: (
Operation: PUSH (
Postfix expression: 75-</td>
<td> 

9. (7–5)*(6+4)
Symbol: 6
Operation: Append to Result
Postfix expression: 75-6</td>
</tr>
</table>

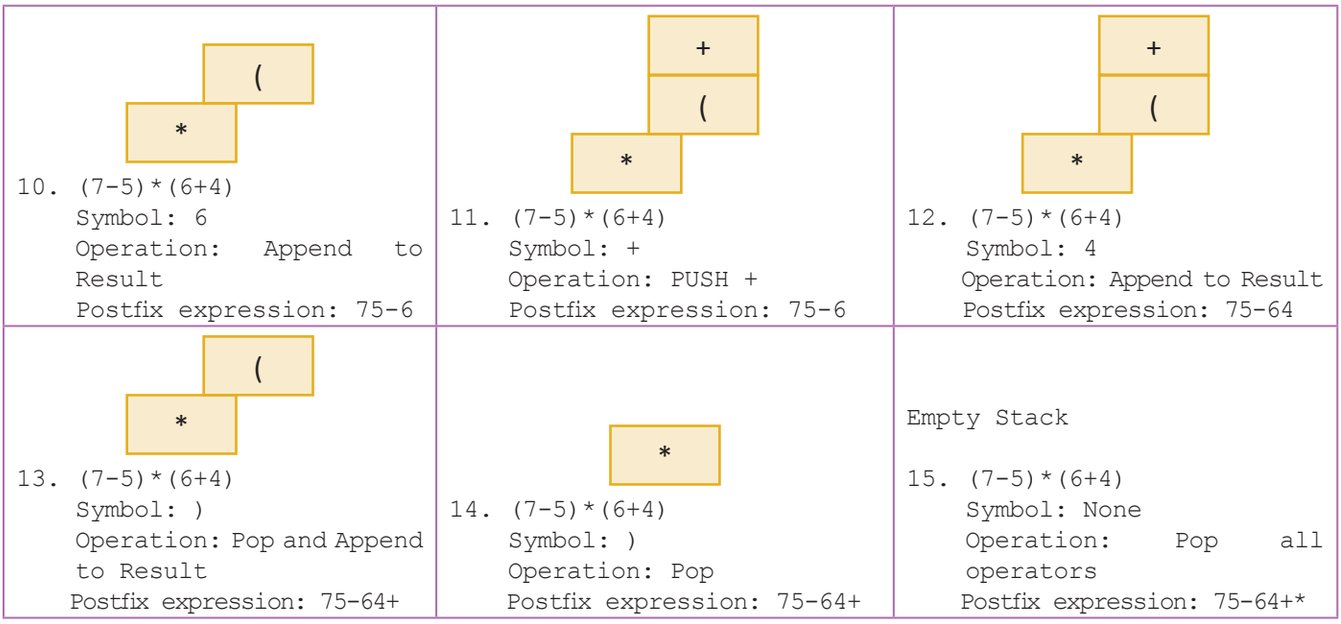


Fig 5.5 Conversion of Infix Expression to Postfix Expression: (7–5)*(6+4)

Assuming that **a**, **b**, **c**, and **d** have values **2**, **3**, **4**, and **5**, respectively, what will be the value of the postfix expression: **(i) abcd*++ (ii) ab+cd*+ (iii) abc- (iv) ab-c-**

We, summarise the above discussion in the Table given below:

Infix Expression: **(7–5)*(6+4)**

Symbol	Action Taken	Stack Status	Postfix Expression
(	PUSH (	(	
7	APPEND 7	(	7
–	PUSH –	(–	7
5	APPEND 5	(–	7 5
)	POP –		7 5 –
	APPEND –		
	POP (		
*	PUSH *	*	7 5 –
(	PUSH (	* (	7 5 –
6	APPEND 6	* (	7 5 – 6
+	PUSH +	* (+	7 5 – 6
4	APPEND 4	* (+	7 5 – 6 4
)	POP +	*	7 5 – 6 4 +
	APPEND +		
	POP (		
	POP	EMPTY STACK	7 5 – 6 4 + *

The equivalent postfix expression is `7 5 – 6 4 + *`

- Based, on the above discussion, we present a complete program for converting infix expression to postfix expression (Program 5.3: `infixToPostfix.py`).
- We develop a function `infixToPostfix()` that accepts an infix expression as an input parameter (lines 1–32, Program 5.3) and returns the resultant postfix expression.
- In this program, we assume that the expression is correctly formed, operands are single digit values, and the operators are binary operators.

- The function `infixToPostfix()` first defines the precedence of all the operators in line 10 followed by an operand list in line 11.
- We create the `result` string in line 13. The function `infixToPostfix()` then invokes the function `createStack()` to create a stack in line 14.
- For each symbol in the expression (line 15), depending upon the appropriate type (operand, operator, opening parenthesis, or closing parenthesis), appropriate action is performed in line 16–32 and the string `result` is updated.
- Finally, the function returns the string `result` comprising the equivalent expression in postfix notation.

Program 5.3 Infix to Postfix Conversion (**`infixToPostfix.py`**)

```
01 def infixToPostfix(expr):
02     '''
03     Objective: To convert infix expression to postfix expression
04     Input Parameter: expr - string (Infix Expression)
05     Return Value: result - string (Postfix Expression)
06     Assumption: Expression is correctly formed
07                 Allowed operators: binary
08                 Allowed operands: single digit
09     '''
10     OprPrecedence = {'+':1, '-':1, '*':2, '/':2}
11     operand = '0123456789'  #string of valid operands
12
13     result = ''
14     stack = createStack()
15     for symbol in expr:
16         if symbol in operand:
17             result += symbol
18         elif symbol == '(':
19             pushStack(stack, symbol)
20         elif symbol == ')':
21             value = popStack(stack)
22             while value != '(':
23                 result += value
24                 value = popStack(stack)
25         elif symbol in OprPrecedence:
26             while not(isEmpty(stack)) and top(stack) != '(' \
27                   and OprPrecedence[top(stack)] >= OprPrecedence[symbol]:
28                 result += popStack(stack)
29             pushStack(stack, symbol)
30     while not(isEmpty(stack)):
31         result += popStack(stack)
32     return result
33
34 #main program segment
35 """
36 Objective: To map infix expression to postfix expression
37 User Interface:
38     1. User is asked to enter:
39          infix expression
40     2. User is presented with the equivalent postfix expression
41 """
42
43 expr = input('Enter an infix expression: ')
44 result = infixToPostfix(expr)
   print("Equivalent postfix expression:",result)
```

On executing the Program 5.3, Python responded with the following output:

```
>>> Enter an infix expression: 2+3*4
>>> Equivalent postfix expression: 234*+
>>> Enter an infix expression: (7-5)*(6+4)
>>> Equivalent postfix expression: 75-64+*
```

Converting an Infix Expression to Postfix Form

Informally, the method of converting an infix expression to postfix form described above, looks at the order of evaluation in the infix expression and the conversion is applied to different subexpressions, as illustrated in the following examples:

1. Convert the following expression to postfix:

   (i) `a-b/(c+d)`

   The operations will be applied in the following order:

   a. `c+d`, we construct the postfix expression for `c+d` as `cd+`

   b. `b/(c+d)`, as we have known that the postfix expression for `c+d` is `cd+`, the postfix expression for `b/(c+d)` will be: `bcd+/`

   c. `a-b/(c+d)`, as we already know the postfix form of `b/(c+d)` is `bcd+/`, the postfix form of `a-b/(c+d)` will be `abcd+/-`.

   (ii) `True and False or not True`

   a. `True and False`: postfix form: `True False and`

   b. `not True`: postfix form: `True not`

   c. `True and False or not True ≡(True and False) or (not True)`

So, the postfix form of `True and False or not True` is:

`True False and True not or`

2. Convert the following infix expression to postfix showing stack status at every step.

   ((A*((B+C)/D))–E)

Symbol	Action Taken	Stack Status	Resultant Postfix Expression
(	PUSH (	(	
(	PUSH (	((	
A	APPEND A	((	A
*	PUSH *	((*	A
(	PUSH (	((*(	A
(	PUSH (	((*((	A
B	APPEND B	((*((	AB
+	PUSH +	((*((+	AB
C	APPEND C	((*((+	ABC
)	POP + APPEND + POP (	((*(	ABC+
/	PUSH /	((*(/	ABC+
D	APPEND D	((*(/	ABC+D
)	POP / APPEND / POP (	((*	ABC+D/
)	POP * APPEND * POP (	(	ABC+D/*
–	PUSH –	(-	ABC+D/*
E	APPEND E	(-	ABC+D/*E
)	POP - APPEND - POP (	EMPTY STACK	ABC+D/*E-

Equivalent Postfix expression is ABC+D/*E-

The beauty of Python is that the implementation of a data structure such as a stack is entirely independent of the type of data being pushed. Indeed, one can push different types of objects on the same stack. So, exactly the same code as given in the text will work for different types of data. Next, consider the example given below.



Write a program to implement a stack for account details (accountNo, balance) of customers. Assume that each element of a stack consists of one record. The program should push, pop and display elements of the stack.



```
def pushRecord(accountStack, element):
  '''
  Objective: To push a record at the top of the stack,
  Input Parameter:
      accountStack - input stack of records
      element - record to be inserted
  Return Value: None
  '''
  accountStack.append(element)
def popRecord(accountStack):
  '''
  Objective: To remove a record  from the top of stack
  Input Parameter:
      accountStack - stack of records
  Return Value: None if there are no records, otherwise the Record being deleted '''
  if len(accountStack) == 0:
      print("NO RECORDS")
      return None
  else:
      return accountStack.pop()
def displayRecord(accountStack):
  '''
  Objective: To print contents of the stack
  Input Parameter: accountStack - stack data structure
  Return Value: None
  '''
  if len(accountStack) == 0:
      print("NO RECORDS")
  else:
      for element in reversed(accountStack):
          print(element)
#main program segment
"""
Objective: To display menu, accept data and execute menu options
User Interface:
    1. User is asked to enter choice of options and account details
    2. This is repeated so long as the user wants to continue.
"""
accountStack=[]
while True:
      print('\nChoose the stack operation to be performed')
      print('1: Push Record')
      print('2: Pop Record')
      print('3: Print Records')
      print('0: Exit')
```

```
        choice = int(input('Enter Choice: '))
        if choice == 1:
            accNo = int(input('Enter Account Number :  '))
            balance=int(input('Enter Balance amount :  '))
            record=[accNo,balance]
            pushRecord(accountStack, record)
        elif choice == 2:
            print('The deleted record is :',popRecord(accountStack))
        elif choice == 3:
            print(" The records are : ")
            displayRecord(accountStack)
        elif choice == 0:
            break
        else:
            print("Invalid Choice!")
```

Let's Summarise

- Stack is a Last In First Out (LIFO) arrangement.
- Push operation inserts an element on the top of the stack.
- Pop operation removes the top element from the stack.
- An attempt to push an element when the stack is already full results in a stack overflow condition.
- An attempt to pop an element from the empty stack results in a stack underflow condition.
- The del operator is used to delete a stack.
- An infix expression comprises a sequence of operands and operators in which operands are separated by operators.
- The postfix notation (also called postfix form) is another way of expressing the arithmetic expressions, eliminating the need to scan parts of an arithmetic expression repeatedly while evaluating it.

Solved Exercises

A. Multiple Choice Questions

1. Which of the following describes the arrangement of objects in a stack correctly?

a. Last In Last Out b. First In First Out c. Last In First Out d. None of these

2. Elements in a stack are pushed in the following sequence (leftmost first):

`4, 15, 8, 10, 3, 1`

Which of the following elements will be popped first?

a. 4 b. 1 c. 10 d. 3

3. Adil is browsing through the web pages on his browser. Suppose, having browsed pages B9, B1, B12, and B3, he moved to page B15. At present, he is on page B15 and clicks the BACK button. To which of the following pages will he navigate?

a. B3 b. B9 c. B12 d. B1

4. Which of the following is NOT the application of a stack?

a. Undo / Redo the tasks in a document b. Printer Scheduling

c. Navigating through the web pages d. Parenthesis Matching

5. Which of the following data types is used to implement a stack in Python?

a. Dictionary b. List c. Set d. None of these

6. Consider an empty stack and the following sequence of operations:

1. **PUSH 50** 2. **PUSH 10** 3. **POP** 4. **PUSH 25**
5. **PUSH 75** 6. **POP** 7. **POP**

Which of the following elements is present on top of the stack?

a. 50 b. 10 c. 75 d. Empty Stack

7. Which of the following is the correct postfix notation for the expression, **p* (q/a) +b** ?

a. `pqa/*b+` b. `pq*a/b+` c. `pqab/*+` d. `pqa*/b+`

B. State whether the following statements are True or False:

1. The sequence of *undo* operations in Microsoft Office is stacked. _____

2. A stack is a Last in First out arrangement of objects. _____

3. In Python, a stack is often implemented using a list. _____

4. An attempt to pop an element from an empty stack results in a stack overflow condition. _____

5. In the postfix notation, operators appear after the operands in the expression. _____

6. A postfix expression is evaluated using a stack while scanning it from right to left. _____

C. Fill in the blanks.

1. _____ operation places an element on the top of the stack.

2. _____ operation is also called deletion of the top element of the stack.

3. A _____ is used to enable a user to move back and forth between the recently visited web pages.

4. At any point in time, only the _____ element of the stack is accessible.

5. When no more elements can be added to the stack, it is called _____ condition.

6. A stack having no elements is called an _____ stack.

7. In an _____ expression, operands are separated by operators.

D. Answer the following questions:

1. What is a Data Structure?

Ans. An arrangement of data along with the associated operations is called a data structure.

2. Give any two characteristics of stacks. [CBSE Sample Paper 2021–22]

Ans. (i) It is a Last In First Out Data Structure

(ii) Pushing and popping of elements can happen at the same end of the stack.

3. List any two applications of stacks.

Ans. (i) Navigating through web pages in the browser using FORWARD and BACK buttons.

(ii) Using Undo / Redo operations while editing a document.

4. What role does a stack perform while functions are invoked in a program?

Ans. When a function calls another function (or itself), the sequence of calls is maintained as a stack. When a function call terminates, the system can retrieve the position from where the execution is to begin.

5. Name any two operations that can be performed on a stack.

Ans. Push and Pop

6. How is Push operation different from Pop operation with respect to stack?

Ans. While *Push* operation places an element on top of the stack, *Pop* operation removes the top element from the stack.

7. What is a Stack Overflow condition?

Ans. If an attempt is made to push an element when the stack is already full, it results in a stack overflow condition.

8. Convert the following infix notations to postfix showing stack status at every step.

(i) `a - b - c`

(ii) `P+Q * (C-D)`

(iii) `W*Y - (A/B ^C)` ^ denotes exponentiation

(iv) `True and False or False`

Ans. (i) a - b- c

Operand/Operator	Action Taken	Stack Status	Postfix expression constructed so far
a	APPEND a		a
-	PUSH -	-	
b	APPEND b	-	a b
-	POP -	EMPTY STACK	
	APPEND -		a b -
	PUSH -	-	
c	APPEND c	-	a b - c
	POP -	EMPTY STACK	
	APPEND -		a b - c -

Thus, equivalent postfix expression is a b - c -

(ii) **P+Q * (C–D)**

The conversion of the given expression will be as follows:

Operand/Operator	Action Taken	Stack Status	Postfix expression constructed so far
P	APPEND P		P
+	PUSH +	+	
Q	APPEND Q	+	P Q
*	PUSH *	+ *	P Q
(	PUSH (	+ * (	P Q
C	APPEND C	+ * (	P Q C
-	PUSH -	+ * (-	P Q C
D	APPEND D	+ * (-	P Q C D
)	POP -	+ * (	
	APPEND -		P Q C D -
	POP (	*	
	POP *	+	
	APPEND *		P Q C D - *
	POP +	EMPTY STACK	
	APPEND +		P Q C D - * +

Thus, equivalent postfix expression is P Q C D - * +

(iii) ((W*Y) - (A/(B ^ C)))

Symbol	Action Taken	Stack Status	Postfix Expression Constructed So far
(	PUSH (	(	
(	PUSH (	((	
W	APPEND W	((	W
*	PUSH *	((*	W
Y	APPEND Y	((*	WY
)	POP *	(	WY*
	APPEND *		
	POP (		
-	PUSH -	(-	WY*
(	PUSH (	(-(	W Y *
A	APPEND A	(-(	W Y * A
/	PUSH /	(-(/	W Y * A
(	PUSH (	(-(/(	W Y * A
B	APPEND B	(-(/(	W Y * A B
^	PUSH ^	(-(/(^	W Y * A B
C	APPEND C	(-(/(^	W Y * A B C
)	POP ^	(-(/	
	APPEND ^		W Y * A B C ^
	POP (		
)	POP /	(-	
	APPEND /		W Y * A B C ^ /
	POP (		
)	POP -	EMPTY STACK	
	APPEND -		W Y * A B C ^ / -
	POP (		

The equivalent postfix expression is `W Y * A B C ^ / -`

(iv) `True and False or False`

Symbol	Action Taken	Stack Status	Postfix Expression Constructed So Far
True	APPEND True		True
and	PUSH and	and	True
False	APPEND False	and	True False
or	POP and	EMPTY STACK	
	APPEND and		True False and
	PUSH or	or	
False	APPEND False	or	True False and False
	POP or	EMPTY STACK	
	APPEND or		True False and False or

The equivalent postfix expression is `True False and False or`

9. Assume that **a, b, c, and d** have values **12, 6, 3,** and **2** respectively. What will be the value of the following postfix expressions (assume that / is an integer division operator)? You should show the contents of the stack at each step.

(i) **a b c d + * ***

(ii) **a b + c d * -**

(iii) **a b c d * / +**

Ans. (i) **a b c d + * ***

Operand/Operator	Action Taken	Stack Status	Output
a	PUSH a	12	
b	PUSH b	12 6	
c	PUSH c	12 6 3	
d	PUSH d	12 6 3 2	
+	POP 2	12 6 3	
	POP 3	12 6	
	PUSH 5 (=2+3)	12 6 5	
*	POP 5	12 6	
	POP 6	12	
	PUSH 30(=6*5)	12 30	
*	POP 30	12	
	POP 12	EMPTY STACK	
	PUSH 360(=12*30)	360	
	POP 360	EMPTY STACK	360

Ans. The value of the given expression is 360.

(ii) **a b + c d * -**

Operand/Operator	Action Taken	Stack Status	Output
a	PUSH a	12	
b	PUSH b	12 6	
+	POP 6	12	
	POP 12	EMPTY STACK	
	PUSH 18(=12+6)	18	
c	PUSH c	18 3	
d	PUSH d	18 3 2	
*	POP 2	18 3	
	POP 3	18	
	PUSH 6 (=3*2)	18 6	
–	POP 6	18	
	POP 18	EMPTY STACK	
	PUSH 12 (=18–6)	12	
	POP 12	EMPTY STACK	12

Ans. The value of the given expression is 12.

(iii) **a b c d * / +**

Operand/Operator	Action Taken	Stack Status	Output
a	PUSH a	12	
b	PUSH b	12 6	
c	PUSH c	12 6 3	
d	PUSH d	12 6 3 2	
*	POP 2	12 6 6	
	POP 3		
	PUSH 6(=3*2)		
/	POP 6	12 1	
	POP 6		
	PUSH 1(=6/6)		
+	POP 1	13	
	POP 12		
	PUSH 13(=12+1)		
	POP 13	EMPTY STACK	13

Ans. The value of the given expression is 13.

10. Convert each of the following infix expressions to the postfix form. You should show the contents of the stack at each step:

(i) **a + b * c + d**

(ii) **a/(b-c)*d**

(iii) **a/((b-c)*d)**

Ans. (i) **a + b * c + d**

Symbol	Action Taken	Stack Status	Postfix Expression Constructed So Far
a	APPEND a		a
+	PUSH +	+	a
b	APPEND b	+	a b
*	PUSH *	+ *	a b
c	APPEND c	+ *	a b c
+	POP *	+	
	APPPEND *		a b c *
	POP +	EMPTY STACK	
	APPPEND +		a b c * +
	PUSH +	+	
d	APPEND d	+	a b c * + d
	POP +	EMPTY STACK	a b c * + d +
	APPEND +		

The equivalent postfix expression is a b c * + d +

(ii) **a/ (b-c) *d**

Symbol	Action Taken	Stack Status	Postfix Expression Constructed So Far
a	APPEND a		a
/	PUSH /	/	a
(	PUSH (	/ (	a
b	APPEND b	/ (	a b
-	PUSH -	/ (-	a b
c	APPEND c	/ (-	a b c
)	POP -	/(	
	APPPEND -		a b c -
	POP (	(	
*	POP /	EMPTY STACK	
	APPEND /		a b c - /
	PUSH *	*	
d	APPEND d	*	a b c - / d
	POP *	EMPTY STACK	
	APPEND *		a b c - / d *

The equivalent postfix expression is a b c - / d *

(iii) **a/ ((b-c) *d)**

Symbol	Action Taken	Stack Status	Postfix Expression Constructed So Far
a	APPEND a		a
/	PUSH /	/	a
(	PUSH (	/ (	a
(	PUSH (	/ ((	a
b	APPEND b	/ ((	a b
-	PUSH -	/ ((-	a b
c	APPEND c	/ ((-	a b c
)	POP -	/ ((	
	APPPEND -		a b c -
	POP (		
*	PUSH *	/ (*	a b c -
d	APPEND d	/ (*	a b c - d
)	POP *	/	
	APPPEND *		a b c - d *
	POP (		
	POP /	EMPTY STACK	
	APPEND /		a b c - d * /

The equivalent postfix expression is a b c - d * /

11. Evaluate the following postfix expressions, where / denotes integer division.

(i) `10 5 2 ^ 8 3 // - +`

(ii) `True False True not False or and or`

Ans. (i) `33` (ii) `True`

12. Evaluate the following postfix expression if the value of x, y and z is 2, 3 and 5 respectively, where / denotes integer division.

`z x x ^ y / -`

Ans. 4

13. Write the equivalent postfix notation for the following:

`num + var* 5/var1`

Ans. `num var 5 * var1 / +`

14. Consider the following stack implemented as a list.

`P, Q , A, B, G` (top)

What will be the status of the stack as the following operations take place in a sequence?

(i) POP

(ii) PUSH B

(iii) POP

(iv) POP

(v) PUSH C

Ans. (i) `P, Q, A, B`

(ii) `P, Q, A, B, B`

(iii) `P, Q, A, B`

(iv) `P, Q, A`

(v) `P, Q, A, C`

15. Consider the following stack implemented as a list that can store a maximum of eight elements.

`M, H, A, R, S, T`

(i) How many POP operations will cause the Stack Underflow condition to occur?

(ii) How many PUSH operations will cause the Stack Overflow condition to occur?

Ans. (i) 7

(ii) 3

16. Consider the code given below and predict the output.

```
def pushStack(stack,num):
    stack.append(num)
def popStack(stack):
    return stack.pop()
stk = []
pushStack(stk, 4)
pushStack(stk, 6)
del stk
print(popStack(stk))
```

Ans. name 'stk' is not defined

17. Julie has created a dictionary containing names and marks as **`name:marks`** pairs of six students. Write a program, with the following user defined functions to perform *push* and *pop* operations:

- push the keys (**`name`** of the student) appearing in the dictionary to a stack, *only* for those students having **`marks > 75`**.
- display the contents of the stack, popping the names one by one.

For example,

If the sample content of the dictionary is as follows:

```
R = {"OM":76, "JAI":45, "BOB":89, "ALI":65, "ANU":90, "TOM":82}
```

The output from the program should be:

```
TOM ANU BOB OM
```

[2022]

Ans.

```
R={"OM":76, "JAI":45, "BOB":89, "ALI":65, "ANU":90, "TOM":82}
def PUSH(stack,name):
    '''
    Objective: To push an element on top of the stack,
    Input Parameter:
          stack - list of names
          name - the name to be pushed
    Return Value: None
    '''
    stack.append(name)
def POP(stack):
    '''
    Objective: To pop an element from the stack
    Input Parameter:
          stack - list of names
    Return Value: Popped element of the stack
    '''
    if stack != []:
         return stack.pop()
    else:
          return None
# main Program
nameStack = []
for name in R:
  if R[name] >= 75:
    PUSH(nameStack, name)
while True:
  if nameStack != []:
    print(POP(nameStack), end = " ")
  else:
    break
```

18. Alam has a list containing 10 integers. You need to help him create a program with separate user defined functions to perform the following operations based on this list.

- Traverse the content of the list and push the even numbers into a stack.
- Pop and display the content of the stack.

For Example:

If the sample Content of the list is as follows:

```
N = [12, 13, 34, 56, 21, 79, 98, 22, 35, 38]
```

Sample output of the code should be:

```
38 22 98 56 34 12
```

CBSE Sample Paper 2022

Ans. We first initialize the stack as as a global list in the Python program, carry out the stack operations, and finally display the stack contents by popping off the numbers from the stack one by one. .

```
N = [12, 13, 34, 56, 21, 79, 98, 22, 35, 38]
def PUSH(stack, num):
    '''
    Objective: To push the even number at the top of the stack,
    Input Parameter:
```

```
            stack - list of even numbers
            num - number to be pushed
        Return Value: None
        '''
        stack.append(num)

    def POP(stack):
        '''
        Objective: To pop an element from the stack
        Input Parameter:
            stack - list of even numbers
        Return Value: Top element of the stack
        '''
         if stack!=[]:
             return stack.pop()
        else:
             return None
    # Main program
    stackEvens = []
    for number in N:
      if number%2==0:
        PUSH(stackEvens,number)
    print(stackEvens)
    while True:
      if stackEvens != []:
        print(POP(stackEvens),end=" ")
      else:
        break
```



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): A stack is a LIFO data structure
Reasoning (R): In case of a stack, the element that is inserted last is deleted first.

2. Assertion (A): A stack is always implemented using a list object.
Reasoning (R): `append()` is used to push an element into a stack.

3. Assertion (A): Inserting element into the stack is known as PUSH operation.
Reasoning (R): One can access only the top element of a stack.

4. Assertion (A): A stack is a FIFO data structure
Reasoning (R): A stack is used to store browsing history.

Ans. 1. a 2. d 3. b 4. d



Case-based Questions

1. Aadil wants to create the following functions in Python using a stack named package:

- makePush(package): to accept a package name (string) and push it onto the stack package
- makePop(package): to delete an item from the top of the stack, package.

Ans.
```
def makePush(package):
    pname=input(" Enter package name:   ")
    package.append(pname)

def makePop(package):
    if (package==[]):
        print( "Stack empty")
    else:
        print ("Deleted element: ", package.pop())
```

2. Rakshit wants to store a list containing the details of employees on to the stack. The tuple object, empData contains the details of the employees as follows:
`empID(int), empName(string), and salary(float).`

Write the following user-defined functions for Rakshit:

- pushData(): to accept data of employees from the user and add it on top of the stack
- popData(): to delete the tuple object stored on the top of the stack.

Assume that the list stack named empStack is declared in the global scope.

Ans.
```
empStack=[]
def pushData():
    empId=int(input("Enter Id of employee: "))
    empName=input("Enter Name of employee : ")
    salary= float(input("Enter Salary of employee :  "))
    empData=(empId, empName ,salary)
    empStack.append(empData)
def popData():
    print(empStack)
    if (empStack==[]):
       print("Stack empty")
    else:
       print("Deleted element:",empStack.pop())
```



Assessment

A. Multiple Choice Questions

1. A list stack can store five elements. An attempt is made to push 6th element on the stack. This is called _____.

a. Stack underflow b. Stack overflow c. Stack garbage d. Stack empty

2. Which of the following element(s) of a stack are accessible?

a. The element that is inserted last b. The element that is inserted first

c. Any element d. Middle element

3. Which of the following functions returns a true if the stack is empty?

a. isNULL() b. isNone() c. isempty() d. isTrue

4. In which of the following order are the elements of a stack displayed?

a. No specific order b. Ascending order

c. Descending order d. First element to last element

5. Which of the following is the application of a stack?

a. Adding data to a file b. Undo-redo operations while editing a document

c. Viewing unused storage on a device d. Sending mails

B. State whether the following statements are True or False:

1. An underflow condition occurs when an attempt is made to push on a stack that is full. _____
2. Infinite number of elements can be inserted into a stack. _____
3. A stack should always contain at least one element. _____
4. A stack is a LIFO structure. _____
5. An element can be added to a list stack using append(). _____

C. Fill in the blanks.

1. A _____ is the arrangement of data along with its associated operations.
2. The process of inserting an element on top of the stack is called _____operation.
3. The process of deleting an element from the stack is called _____ operation.
4. If an attempt is made to remove an element from an empty stack, it is called _____.

D. Answer the following questions:

1. Why are data structures so called?
2. Define a stack.
3. What is the *Stack Underflow* condition?
4. List all operations that can be performed on a stack.
5. Assume that **a, b, c, d, e,** and **f** have values **12, 6, 3, 2, 6,** and **9,** respectively. What will be the value of the following postfix expressions (assume that / is an integer division operator)? You should show the contents of the stack at each step.:
   (i) `a b c d + * *`
   (ii) `a b c d * + *`
   (iii) `a b c d * * +`
   (iv) `a b c d + + *`
   (v) `a b c d + * +`
   (vi) `a b c d * + +`
   (vii) `a b + c * d *`
   (viii) `a b c + * d *`
   (ix) `a b - c * d *`
   (x) `a b c - * d *`
   (xi) `a b - c d * /`
   (xii) `a b c d / + +`
   (xiii) `a b + c * d /`
   (xiv) `a b + c / d *`
   (xv) `a b c / * d *`
   (xvi) `a b + c d / *`
   (xvii) `a b - c / d *`
   (xviii) `a b - c * d /`
   (xix) `a b c - / d *`
   (xx) `a b - c d / *`
   (xxi) `a b c d e f + * + + +`
6. Convert each of the following infix expressions to the postfix form. Show the contents of the stack at each step:
   (i) `a - b - c - d`
   (ii) `(a - b) - (c - d)`
   (iii) `(a-b)-c-d`
   (iv) `a/b+c-d`
   (v) `a/b/c/d`
   (vi) `(a/b)/(c/d)`

(vii) `(a/b)/c/d`
(viii) `a/(b/c)/d`
(ix) `a/((b/c)/d)`
(x) `a/b/(c/d)`
(xi) `a-b*c/d`
(xii) `a/b-c/d`
(xiii) `a/b-(c/d)`
(xiv) `(a-b)/(c/d)`
(xv) `(a-b)*c/d`
(xvi) `a/(b-c)/d`
(xvii) `a/(b+c/d)`
(xviii) `a/b/(c-d)`
(xix) `a+b/(c-d)`
(xx) `(a - b) * (c - d) * e`
(xxi) `(a-b)*c+(d+e)*f`
(xxii) `(a-b)*c*d+e`
(xxiii) `a*(b+(c+d)*e)`

7. The trains numbered 1, 2, and 3 arrive on the railway track one after the other in the sequence 1, 2, 3. Thus, train number 1 is at the front end and train number 3 is at the rear end. The station master moves the train out of the station according to priority using a shunting track that acts as a stack.



Which of the following exit sequences are possible? At each step, show the position of the trains in the two tracks?

(i) 1 2 3
(ii) 1 3 2
(iii) 2 1 3
(iv) 2 3 1
(v) 3 1 2
(vi) 3 2 1

8. The trains numbered 1, 2, 3, and 4 arrive on railway track one after the other in the sequence 1, 2, 3, 4. Thus, train number 1 is at the front end and the train number 4 is at the rear end. and wait in in a queue for moving out of the station. The station master moves the train out of the station according to priority using a shunting track that acts like a stack.



Which of the following exit sequences are possible? At each step, show the position of the trains in the two tracks?

(i)	1, 2, 3, 4	(ii)	1, 2, 4, 3	(iii)	1, 3, 2, 4	(iv)	1, 3, 4, 2	(v)	1,4,2,3
(vi)	1,4,3,2	(vii)	2,1,3,4	(viii)	2,1,4,3	(ix)	2,3,1,4	(x)	2,3,4,1
(xi)	2,4,1,3	(xii)	2,4,3,1	(xiii)	3,1,2,4	(xiv)	3,1,4,2	(xv)	3,2,1,4
(xvi)	3,2,4,1	(xvii)	3,4,1,2	(xviii)	3,4,2,1	(xix)	4,1,2,3	(xx)	4,1,3,2
(xxi)	4,2,1,3	(xxii)	4,2,3,1	(xxiii)	4,3,1,2	(xxiv)	4,3,2,1		

9. Write a function in Python that takes a list as an argument and pushes only even multiples of 5 to the stack. Thereafter, display the members and return the size of the newly created stack.

10. Kabir wants to accept a set of characters from the user and store its contents on to a stack. The characters are thereafter printed in reverse order and each character is printed twice. Write a program in Python for Kabir to complete his task. (Assume maximum 30 characters can be entered by the user.)

For example,

If the user enters My Country, the output will be

yyrrttnnuuooCC yyMM

11. Julie wants to accept 10 numbers and store them into a list. Thereafter, she wants to store only even numbers onto the stack. WAP to display ONLY the largest element from the stack.
For example:

If the user enters the following numbers:

1, 8, 5, 2, 90, 4, –4, –10, 40, 99,

The output will be :

The maximum value of stack is 90

12. Write a menu driven program to perform PUSH, POP and DISPLAY operations on a stack with each element comprising a list with travel details as follows:

travelId	integer
travelDate	String
Destination	String

13. Write a program to accept 10 numbers as a list and push only prime numbers into a stack.

14. Given a string str1 involving only three symbols {a, b, c}. The symbol b occurs exactly once in the string as the middle character of the string. Write a function that uses stack to test whether the string str1 is a palindrome.

15. Consider an expression which has been parenthesized for readability and ease in evaluation. Write a program that makes use of stack and check whether the parenthesis in the given expression are evenly balanced or not.

16. Kulwinder has successfully created the program to navigate the web pages in backward direction by storing the URLs of web pages in a stack. He now wants to create a similar program to navigate in forward direction. Help him to complete this task by creating another stack in the program. The program will now have the following menu options:

1. BACK
2. FORWARD
3. DISPLAY BACK PAGES
4. DISPLAY FORWARD PAGES
5. EXIT

17. You have studied how to convert an infix expression to postfix using multiple passes in stacks. Can we evaluate an infix expression in a single pass using two stacks? The algorithm to do so is given below. Write a program for the same.

Algorithm:

1. Create an empty operator stack
2. Create an empty operand stack
3. for each token in the input string
   a. Get the next token in the infix string
   b. If next token is an operand, place it on the operand stack
   c. If next token is an operator, evaluate the operator (next op)

4. While operator stack is not empty, pop operator and operands (num2 and num1), evaluate num1 operator num2 and push result onto operand stack
5. Pop result from operator stack

Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): If a stack is implemented using a list, append() is used to push an element into the stack.
Reasoning (R): `pop()` is used to delete the bottom most element of the stack.

2. Assertion (A): Any new element pushed into the stack always has an index one more than the index of last element of the stack.
Reasoning (R): A stack is a FIFO structure.

Case-based Questions

1. Write user-defined functions, `pushOn(Book)` and `popFrom(Book)`, in Python to add a new Book name and delete a book name respectively from a list stack named `Book`.

2. Ruhil has created a dictionary containing the names and ages of 5 students. Write a program, with separate user-defined functions to perform the following operations:

a. `push15()`–to push names of those students onto the stack whose age is greater than 15

b. `pop15()`–to display and delete the elements of the stack in LIFO order.

For example: if the dictionary contains the following data:

```
D={"SHANTI":14, "Joe":20, "Prabhjot":12, "Tehzeeb":16}
```

The output will be:

Tehzeeb

Joe

Previous Years' Questions

1. a. A list contains following record of customer: [2023]

```
[Customer_name, Room Type]
```

Write the following user defined functions to perform given operations on the stack named `'Hotel'`:

(i) `Push_Cust ()`–To Push customers' names of those customers who are staying in Delux' Room Type.

(ii) `Pop_Cust ()`–To Pop the names of customers from the stack and display them. Also, display "Underflow" when there are no customers in the stack.

For example:

If the lists with customer details are as follows:

```
["siddarth", "Delux"]
[Rahul", "Standard"]
"Jerry", "Delux"]
```

The stack should contain

Jerry

Siddharth

The output should be:

Jerry

Siddharth

Underflow

OR

b. Write a function in Python, `Push (Vehicle)` where, Vehicle is a dictionary containing details of vehicles - `{Car_Name: Maker}`.

The function should push the name of car manufactured by 'TATA' (including all the possible cases like Tata, TaTa, etc.) to the stack.

For example:

If the dictionary contains the following data:

```
Vehicle={"Santro" : "Hyundai", "Nexon": "TATA", "Safari" : "Tata"}
```

The stack should contain

Safari

Nexon

Ans. a
```
customer=[["Siddarth", "Delux"], ["Rahul", "Standard"], ["Jerry", "Delux"]]
hotel=[]
def push_cust():
  for i in customer:
    if i[1]=='Delux':
      hotel.append(i[0])
  return hotel

def pop_cust():
  if hotel==[]:
    return "Underflow"
  else:
    return hotel.pop()

push_cust()
while True:
  if hotel==[]:
    print(pop_cust())
    break
  else:
    print(pop_cust())
```

b.
```
Vehicle={"Santro" : "Hyundai", "Nexon": "TATA", "Safari" : "Tata"}
stk=[]
def push(vehicle):
  for i in vehicle:
    if vehicle[i].lower()=='tata':
      stk.append(i)
  return stk

push(Vehicle)
```

```
for i in range(-1,-len(stk)-1,-1):
  print(stk[i])
```

2. "Stack is a linear data structure which follows a particular order in which the operations are performed." What is the order in which the operations are performed in a Stack?

Name the List method/function available in Python which is used to remove the last element from a list implemented stack.

Also write an example using Python statements for removing the last element of the list. [2022]

Ans. The operations are performed in a stack follows Last In First Out (LIFO) order.

The `pop()` method/function available in Python which is used to remove the last element from a list implemented stack. `stk.pop()`

NCERT Exercise Solutions

1. State TRUE or FALSE for the following cases:

a. Stack is a linear data structure.

b. Stack does not follow LIFO rule.

c. PUSH operation may result into underflow condition.

d. In POSTFIX notation for expression, operators are placed after operands.

Ans. a. True

b. False

c. False

d. True

2. Find the output of the following code:

a.
```
result =0
numberList = [10,20,30]
numberList.append(40)
result = result + numberList.pop()
result = result + numberList.pop()
print("Result=",result)
```

a.
```
answer = []; output=''
answer.append('T')
answer.append('A')
answer.append('M')
ch = answer.pop()
output = output+ch
ch = answer.pop()
output = output + ch
ch = answer.pop()
output = output + ch
print("Result=",output)
```

Ans. a. `Result = 70`

b. `Result = MAT`

3. Write a program to reverse a string using stack.

Ans.
```
def reverseString(inputString):
    stack = []
    # Push each character onto the stack
```

```
    for char in inputString:
        stack.append(char)
    reversedString = ""
    # Pop characters from the stack to reverse the string
    while stack:
        reversedString += stack.pop()
    return reversedString


inputStr = input("Enter a string to reverse: ")
reversedStr = reverseString(inputStr)
print(f"Reversed string: {reversedStr}")
```

4. Write a program to create a Stack for storing only odd numbers out of all the numbers entered by the user. Display the content of the Stack along with the largest odd number in the Stack. (Hint. Keep popping out the elements from stack and maintain the largest element retrieved so far in a variable. Repeat till Stack is empty).

Ans.

```
# user input
n = int(input("Enter the number of values: "))
# initialize empty stack
stack = []
# take n numbers from the user
for i in range(n):
    num = int(input("Enter number : "))
    if num % 2 != 0: # if only the number is odd push it
        stack stack.append(num)
# assuming the first number out of stack is the largest
largestNum = stack.pop()
# loop till stack is empty
while(len(stack)>0):
    num = stack.pop()
# if the number popped out is greater than the largest number found until now
# then make the this number the largest number
    if num > largestNum:
        largestNum = num
# print the largest number
print("The largest number found is: ",largestNum)
```

Answers

Multiple Choice Questions

1. (c) 2. (b) 3. (a) 4. (b) 5. (b) 6. (a) 7. (a)

True or False

1. (T) 2. (T) 3. (T) 4. (T) 5. (T) 6. (T)

Fill in the blanks

1. push 2. Pop 3.stack 4. top 5. Stack overflow 6. empty 7. infix

6 COMPUTER NETWORKS

Chapter Outline

6.1 Components of a Computer Network
6.2 Evolution of Networking
6.3 Switching Techniques
6.4 Data Communication Terminology
6.5 Data Transmission Modes
6.6 Transmission Media
6.7 Network Topologies
6.8 Network Types
6.9 Network Devices

Social networks, such as those made up of relatives, classmates, and members of a Facebook group, allow friends to communicate with one another. A computer network is a group of related computers that are linked together. It gives computers in a network a way to communicate and share hardware and software resources. Friends communicate with each other on social networks, for example, relatives, classmates, and people in a Facebook group form social networks. A computer network is a collection of interconnected computers in the same spirit. It provides a means of communication amongst the computers in a network to share hardware and software resources. The Internet is the largest computer network. Computers connected to the Internet can share information amongst themselves. There are several applications of computer networks such as collaborative learning, scientific research, video-conferencing, online gaming, e-shopping, e-governance, telehealth, and chatting. This chapter will discuss what constitutes a computer network, its categorisations, functionalities, and the technologies that make it work and briefly mention the Internet.

Computer Networks: Set of devices connected via links.

6.1 Components of a Computer Network

The main components of a network are entities participating in the network, the medium through which communication takes place, and the content to be communicated. Children in a kindergarten class communicate with the teacher and themselves without being aware of any communication support. Of course, we know that communication is made possible by the air present in the room that serves as a medium (also called a channel) through which sound waves travel from one person to another. In this example, the students and the teacher form the entities, the air is the medium, and the conversation in a natural language is the content. Similarly, the computers communicate with each other by exchanging (sending and receiving data) the content in the form of binary data.

The participating computers agree on the norms of communication. Transfer of data between computers (also called nodes or terminals) via some channel is called data communication. Fig 6.1 shows a communication system comprising the following components:

1. Sender—the node that sends the data.
2. Receiver—the node that receives the data.
3. Message—information being communicated.
4. Channel—the medium through which data is transmitted.

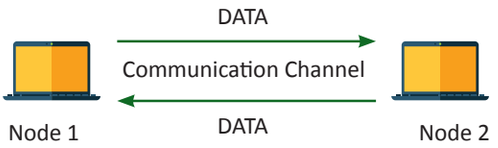


Fig 6.1 Communication System

Fig 6.2 shows a collection of interconnected nodes that communicate through some channel that forms a computer network. Each participating node in a network has a unique media access control address (MAC address). For example, if you are using WINDOWS® you can find your computer's MAC address (physical address) by executing the following command from the command prompt:

```
ipconfig /all
```



What is the MAC address of your computer?

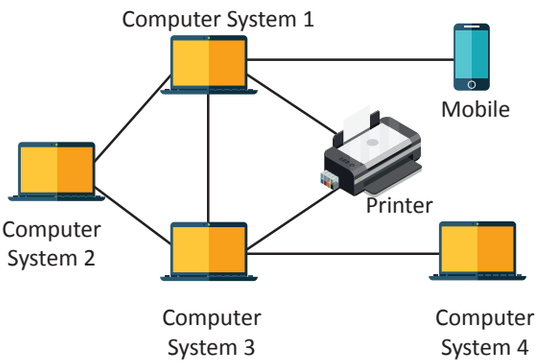


Fig 6.2 Computer Network

One or more nodes in a computer network may provide resources and services to other nodes. An application program that provides a service in a computer network is called a server. For example, an *FTP* server installed on a computer enables the transfer of files to various nodes in the network. In informal use, the node that hosts a service is sometimes called a sever, even though it does not confirm to the technical meaning of the term.

6.2 Evolution of Networking

In 1876, Bell developed a communication paradigm based on telephone lines. Initially, each telephone was connected by a dedicated wire to another phone it wished to communicate with. However, the network soon became a mess of cables. In 1877, his Bell Telephone Company established the Public Switched Telephone Network (PSTN), allowing several home phones to connect to a dedicated office that would establish a dedicated connection between the interested parties. This form of end to end communication remained the primary mode of communication for the next hundred years. Establishing a dedicated connection (also called path or circuit) between the two end-users often requires passing through several intermediate switches en route. So, setting up a telephone circuit is called **circuit switching**. Fig 6.3 shows a telephone network.

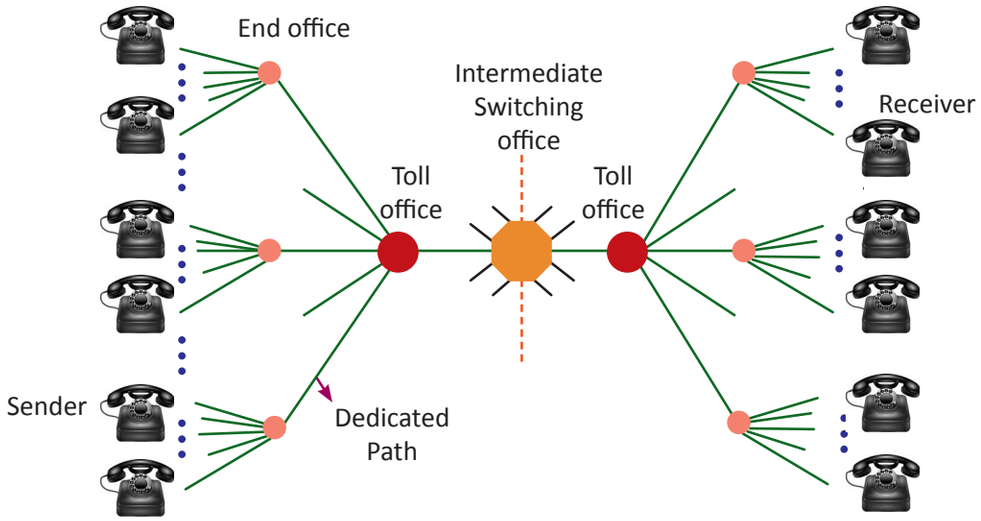


Fig 6.3 Example of a telephone circuit in a telephone network

As telephone circuits were susceptible to single point failure, during the 1960s, Paul Baran, along with Donald Davies and Len Kleinrock, came forward with the idea of fault resistant network based on digital packet switching. Unlike circuit switching, which requires a dedicated circuit for the entire session (duration of communication), **packet switching** works on the principle of sharing the links in the network. Messages to be transmitted are divided into small chunks called packets. A packet carries the following information: destination address, data, and a sequence number for arranging the packets in the correct sequence at the receiver end. The different packets in a session may take different paths from a sender to the receiver.

ARPANET: First Public Packet-switched Computer network.

Advanced Research Project Agency (ARPA), set up by US Department of Defence (DoD), realised the need to connect geographically separated research computers to form a network. Consequently, Advanced Research Project Agency Network (ARPANET) evolved with the networking of computers of four host institutes (UCLA, Stanford, UCSB and the University of Utah) and later grew to 100's of hosts. In Fig 6.4, we show ARPANET introduced in the late 1960s.

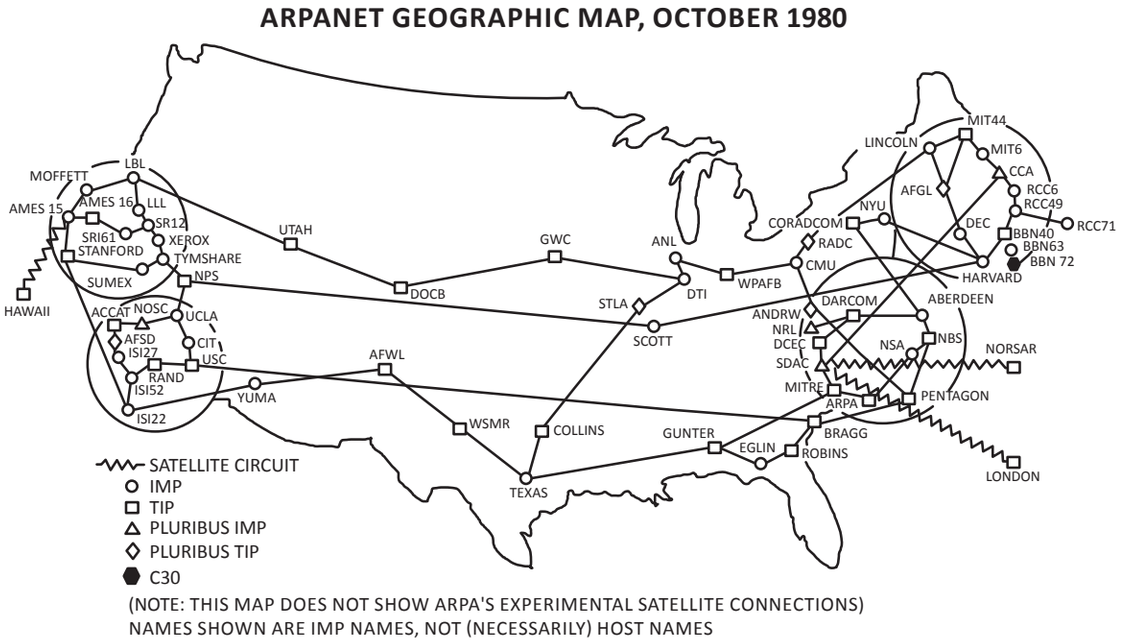


Fig 6.4 ARPANET

Initially, the scope of ARPANET was restricted to military and research. Subsequently, its use extended to education by supporting various educational institutes. In the 1970s, other networks such as SatNet (Satellite Network), Packet Radio, and Ethernet LAN came into being. Different networks deployed different technologies. So, a unifying mechanism was required to enable communication between the networks with heterogeneous technologies. Towards this end, Cerf and Kahn developed TCP/IP (Transmission Control Protocol/Internet Protocol) in 1970. TCP/IP became the de facto standard for communication during the early eighties. It facilitated interconnectivity amongst heterogeneous networks. In late 1984, National Science Foundation (NSF) created NSFNET—a network with six supercomputers that aimed to promote advanced research and education. Several regional networks of universities could access these supercomputers via a computer network. Subsequently, in the 1990s, several other countries and regions developed their networks for scientific and commercial purposes. When ARPANET and NSFNET were interconnected, the network growth increased tremendously. This vast network formed the Internet (network of networks), and its growth connected the whole world.

In the mid-1990s, the number of hosts connected through the Internet grew exponentially. Several organisations in government and private sectors (collectively called Internet Service Providers (ISPs)) joined hands to provide Internet connectivity. The Internet made it possible to exchange information and communicate with remote hosts. The Internet provides several services such as

1. **Email:** Sending and receiving electronic messages.
2. **File Transfer:** Files can be copied from one system to another.
3. **Newsgroup**: A Newsgroup is a discussion forum where users of related interest can post information and reply to others' posts.
4. **Remote login:** Remote login enables users on one system to log into a remote system and use hardware and software available at the remote computer.
5. **Chat:** Chat provides real-time communication between two or more users. It allows the connected users to exchange messages in a chat window. Examples of chat applications include WhatsApp, Facebook messenger, and Telegram.
6. **Video Conferencing:** Video Conferencing allows multiple users to exchange video and voice messages in real-time. Examples: Video calls using applications like Facebook and WhatsApp, or more sophisticated tools like Google Meet, Zoom Meetings, Cisco Webex.
7. **Cloud Computing:** Cloud Computing enables storage and computing services over the Internet. For example, Google Cloud Platform (GCP), Amazon Web Services (AWS), and Microsoft Azure enable collaborative computing environments.
8. **Interspace:** Interspace is a client/server software tool that allows users to connect online in dynamic 3D worlds using real-time audio, video, and text chat.



Interspace is a client/server software program that allows multiple users to communicate.



1. Rimit and Azaad are connected with each other through video and voice in real time. Which internet service are they using?
2. Abha has created a document in Google Drive and shared it with her friends. Is she using a Cloud Computing service?



6.3 Switching Techniques

A direct point to point link enables the exchange of information from source to destination. However, setting up a connection between each pair of source and destination would lead to an unmanageably large number of point-to-point connections. Switching techniques such as circuit switching and packet switching are used to deliver messages from a sender to a receiver efficiently.

A subnet is part of a more extensive network comprising a grouping of network devices connecting various transmission lines (see Fig 6.5). The data travels from source to destination through the subnet. Routers are responsible for forwarding incoming packets to the appropriate transmission lines so that the messages reach their respective destinations.

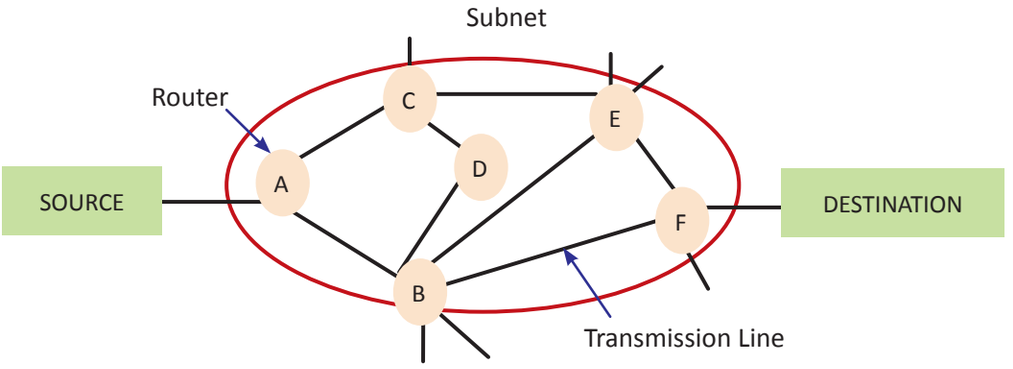


Fig 6.5 Subnet

6.3.1 Circuit Switching

Circuit switching requires a dedicated connection between the source and the destination. A circuit defines a fixed path that data would follow for the entire transmission duration. Suppose fixed circuit connecting routers A, B, and F (see Fig 6.6) is established for transmitting data between nodes A and F. When the data transfer is complete, the connection between A and F is terminated. There are two significant problems with circuit switching:

1. The set-up time of the order of several seconds is required to establish a connection. High set-up time may be undesirable for several real-time applications such as streaming videos.
2. Failure of any intermediate node on the dedicated path will halt the entire transmission.

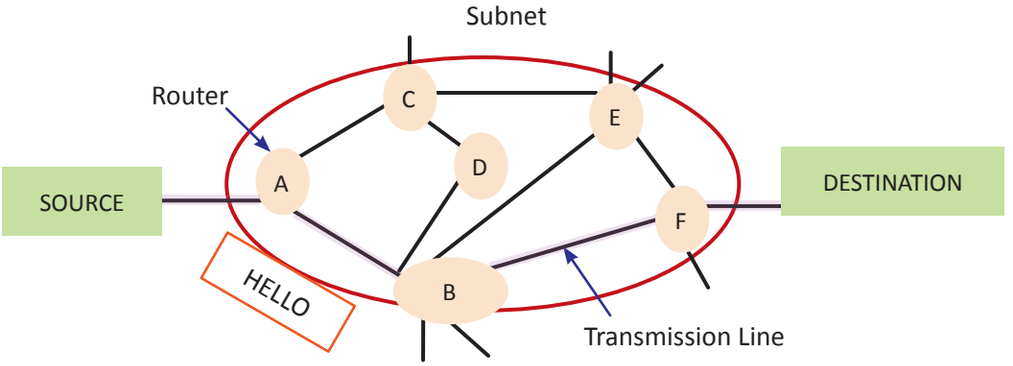


Fig 6.6 Circuit Switching

6.3.2 Packet Switching

Packet switching eliminates the problem of network congestion by dividing a message into smaller units known as packets. Smaller packets are not required to be stored at intermediate switching nodes. Each packet contains the destination address, sequence number denoting its position in the original message and the data to be transmitted. Packet switching is categorised as datagram packet switching and virtual circuit packet switching. No prior connection is needed between source and destination in datagram packet switching. Each packet is routed to the destination, independently of other packets in the network. Hence, the route followed by each packet may differ. Fig 6.7 shows the transmission of a message 'HELLO' from source (A) to the destination (F). Assume each letter of the message corresponds to a packet and takes a different route to F. On receiving all the packets of a message, the destination station will assemble them according to the sequence number. On the contrary, in virtual circuit packet switching, a virtual connection is established first between the source and the destination. All the packets of a given message follow the same dedicated virtual path. Virtual circuits are more reliable than datagrams, but datagrams provide greater efficiency.

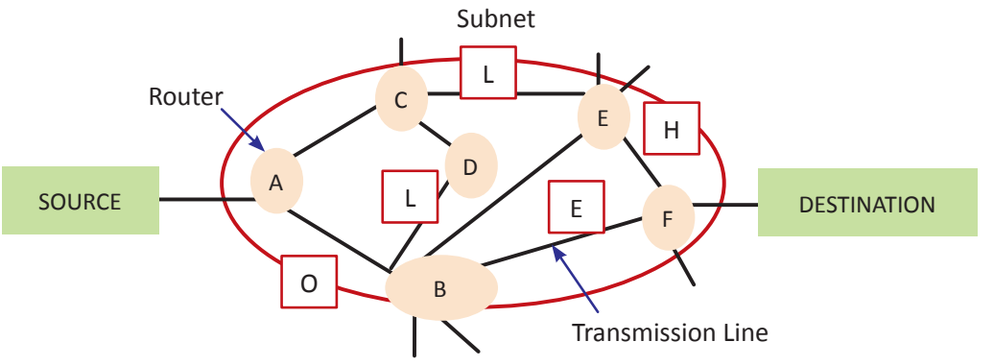
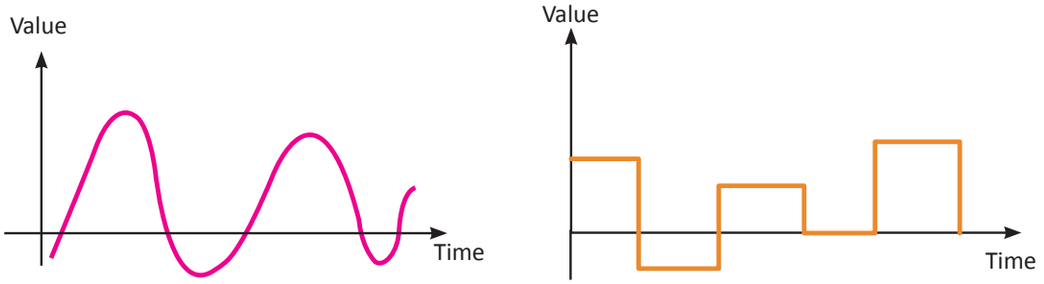


Fig 6.7 Datagram Packet Switching

Circuit Switching: circuit needs to be established.
Packet Switching: packet goes through a dynamic route.

6.4 Data Communication Terminology

Data communication refers to exchanging information between two devices. The data is transmitted between two devices in the form of electrical signals that may be analog or digital. The value of an analog signal changes continuously with respect to time (see 6.8 (a)), so its value lies in a continuous range. Digital signal being discrete, its value changes at discrete points in time. It takes only a fixed set of values, say 0 or 1 (see 6.8 (b)). When data communication occurs between two devices within a small geographical area, it is called local communication. Similarly, data communication between remotely located devices is called remote communication.



(a) Analog Signal (b) Digital Signal

Fig 6.8 Analog and Digital Signals

Bit Rate or Data Transfer Rate

Bit Rate or Data Transfer Rate is the number of bits transmitted per second. It is measured in bps (bits per second) and Bps (Bytes per second). Thus, KBps, MBps, GBps, TBps refer to transfer rates of Kilobytes per second, Megabytes per second, Gigabytes per second, and Terabytes per second, respectively. Suppose, a channel transmits data at a rate of 72 Mbps. Transmitting a .... movie of size 720 MB via this.... channel will require 80 seconds (=720*8/72). While transfer rate is often specified in the decimal system (1 Kbps = 10^3 bits per second), the file size is specified in powers of 2 (1Kb = 2^{10} bytes). The minor difference in time due to mismatch between the number systems (1000 bits in decimal system vs 1024 bits in binary system) is often ignored in computations for estimating the communication time.

Baud Rate: The term baud rate is named in honour of Emile Baudot—a French engineer who invented a 5-bit teletype code for coding each character. Baud rate refers to the number of signal units transmitted in a second. Baud rate is computed by dividing bit rate by the number of bits per signal unit. For example, if the bit rate is 120 kbps, using the 8-bit coding scheme, i.e. 8 bits per character, baud rate would be 15 (=120/8) kcps (kilo characters per second),

Cycle: A cycle in a signal refers to a recurring pattern, for example, see Fig 6.9.

Frequency: The number of cycles per second defines the frequency of a signal. It is expressed in Hertz (Hz). In Fig 6.9, we observe that the signal completes three cycles in a second. So, the frequency of this signal is 3Hz.

Bandwidth: A channel's bandwidth (BW) denotes the frequencies that travel through it (Unit: Hz). Thus, it is the difference between the channel's highest frequency and the lowest frequency. It is measured in Hz (Hertz), KHz (KiloHertz), MHz (MegaHertz), etc. Similarly, the bandwidth (BW) of a signal is defined as the range of frequencies it comprises. So, if a signal consists of several component signals with frequency 120 Hz, 400Hz, 550 Hz, 700 Hz and 990 Hz, its bandwidth will be 870Hz (990 Hz–120 Hz).

Data Rate: Data rate is the amount of data that goes through a medium per unit time (unit: bits/sec).

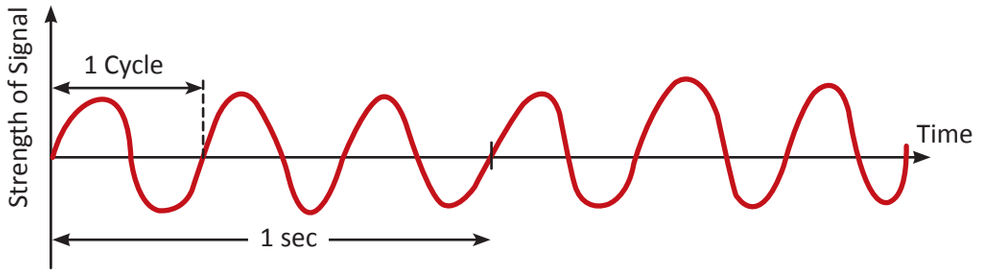


Fig 6.9 Cycle in a Signal

Data Rate: Amount of data passed over a given period.

Bandwidth: Range of frequencies.

1. This type of switching technique forms a dedicated connection between the source and destination. What is the name of that switching technique?
2. What is the number of bits transmitted per second on a network?
3. If a signal consists of multiple signals with frequency 100 Hz, 250Hz, 370 Hz, 510 Hz and 750 Hz, what will be its bandwidth?

6.5 Data Transmission Modes

The transmission of data over a communication channel occurs in the form of signals that travel from source (sender) to destination (receiver). The transmission of data may occur in one of the following modes:

- Simplex Transmission
- Half-Duplex Transmission
- Full-Duplex Transmission

6.5.1 Simplex Transmission

It is the simplest transmission mode in which the data travels in one direction only. As shown in Fig 6.10. The unidirectional flow of data is from sender to receiver. For example, monitors and keyboards use simplex mode of transmission. The keyboard can only transmit the data for processing and the monitor can only receive data to be displayed as output.

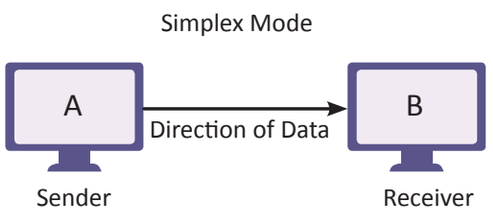


Fig 6.10: Simplex Transmission

6.5.2 Half-Duplex Transmission

In half-duplex transmission, the data transmission can happen in both directions. However, when one device sends the data, others can only receive it and vice versa. When a device wants to transmit data, its transmitter is switched on and the receiver is switched off. Once the transmission is complete, the roles may be switched, so that the receiver can switch on its transmitter to send data. Thus, the communication channel is shared for transmitting data in both directions. For example, while using a walkie-talkie, one person's switches on the transmitter to talk while the other device just receives the message (Fig 6.11). The direction of communication can be reversed at any point of time. The half-duplex transmission is faster than simplex but slower than full-duplex transmission.

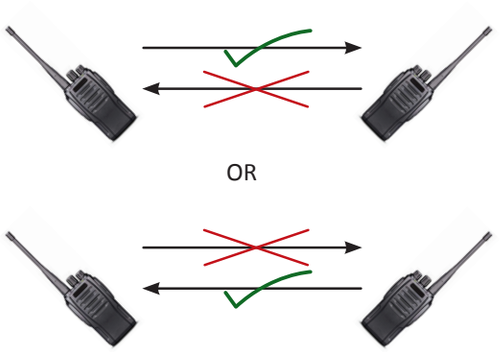


Fig 6.11: Half Duplex Transmission

6.5.3 Full-Duplex Transmission

In full-duplex transmission mode, the data can be transmitted in both directions simultaneously (Fig 6.12). The communication in mobile phones or landline phones is an example of full-duplex transmission as the individuals can talk and listen at the same time. In full-duplex transmission, the signals share the channel capacity in one of the following ways:

- There are two physically separate transmission paths (simplex lines)–one for sending and one for receiving.
- The capacity of the channel is divided between signals travelling in either direction.

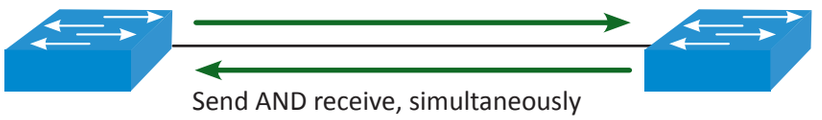


Fig 6.12: Full Duplex Transmission

6.6 Transmission Media

A transmission medium (also called a channel or a link) carries signals (also called data or messages) from one node to another in a network. As already mentioned, when we talk, we use air as the medium of communication. A transmission medium may be wired or wireless as described below:

1. **Guided Medium:** The physical medium comprises a wired link, for example, twisted pair, coaxial cable, and fibre optics cable.
2. **Unguided media:** Unguided medium comprising space (including vacuum) is used for wireless communication. Electromagnetic waves such as Bluetooth, infrared, radio, microwave are transmitted using unguided media. Whereas an infrared link is used for short-range communication, radio links that use satellites are deployed for long-range communication, possibly across the continents.

We describe below some important transmission media and their advantages and disadvantages.

6.6.1 Twisted pair cable

Twisted-pair cables were the first guided transmission media. A twisted pair comprises two copper wires twisted around each other. The copper conductors in these wires are covered with insulating shielding material. The twisting of wires reduces electromagnetic interference. The twisted wires are approximately 1mm thick. They can be used for communication up to a maximum length of 100m. A signal is transmitted in the form of an electric current. Telephone companies widely use these cables for connecting telephones. In Fig 6.13, we show a twisted pair cable.



Fig 6.13 Twisted Pair Cable

A twisted-pair cable may be unshielded (UTP) or shielded (STP). STP cable is provided with an additional shield that prevents electromagnetic interference. However, STP cable is more expensive than UTP.

Advantages of Twisted-Pair Cable

- Twisted-Pair cables are easy to install and connect.
- Twisting of wires in twisted-pair cables leads to a reduction in interference.
- Installation and connection costs of twisted pair cables are low.
- Twisted-Pair cables are capable of transmitting both analog and digital signals.
- Twisted-Pair cables can run over several kilometres and support data rates of up to 100Mbps.

A disadvantage of Twisted-Pair Cable

- Twisted pair cables are not suitable for long-distance transmissions as signal strength reduces after travelling a distance of a few kilometres, and some mechanism is required for regenerating the signal strength.

6.6.2 Coaxial cable

Coaxial cables have better insulation than twisted-pair cables. A coaxial cable comprises a copper core covered by an insulating material that is further surrounded by an outer conductor (typically in the form of a copper mesh). Again, an insulating protective material surrounds the outer conductor. Like twisted pair cables, signals in a coaxial cable are transmitted in the form of electric current. Due to better shielding, these wires are noise-resistant and can carry signals over long distances. The coaxial cables have been traditionally used in television cables. Fig 6.14 shows a coaxial cable.

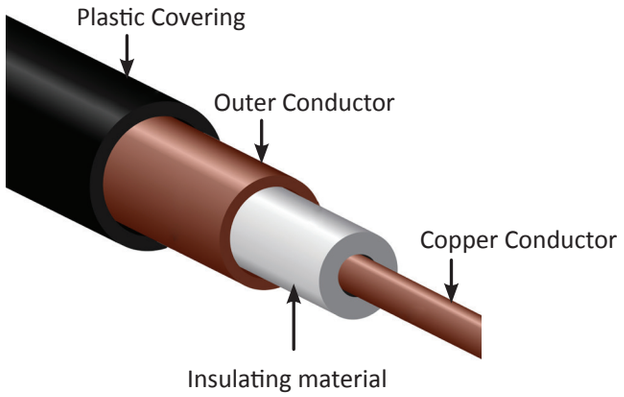


Fig 6.14 Coaxial Cable

Advantages of Coaxial Cable

- Coaxial cables have higher bandwidth (approximately 1GHz) than twisted-pair cables.
- Coaxial cables have high noise immunity because of better shielding.
- Coaxial cables can carry data over longer distances without much signal loss than twisted-pair cables.

A Disadvantage of Coaxial Cable

- Coaxial cables are more expensive than twisted-pair cables.

6.6.3 Fibre-Optic Cable

Fibre-optic cables (also known as optical fibre cables) transmit signals in the form of light. Electrical signals are converted to light signals at the source and converted back to electrical signals at the destination. An optical transmission system comprises three components: transmission medium, source of light, and light detector. For example, a Light Emitting Diode (LED) may serve as a light source at one end of a fibre, and a photo-diode may serve as a light detector at the other end. As optical fibres transmit data only in one direction, two cables are needed for a duplex connection.

Fibre optic transmission works on the principle of total internal reflection. When a light ray travels through the fibre and falls on the boundary of the medium at an angle of incidence greater than the critical angle, it gets reflected back in the medium. It can be trapped for long distances without any loss. When the light falls on a detector, it converts the light pulses into electrical signals. In Fig 5.15, we show an optical fibre. Optical fibre comprises cladding—a covering through which light reflects into the core and a protective cover.

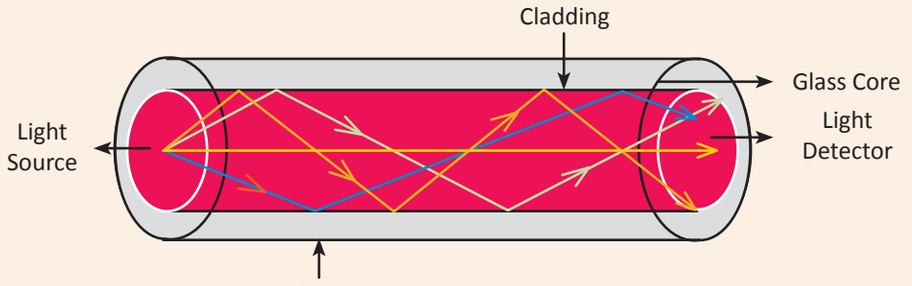


Fig 5.15 Fibre Optic

Optical fibres are categorised as multi-mode and single-mode fibres. In the case of multi-mode fibre, when light rays are refracted back in the medium, they bounce in the medium at different angles and exhibit different modes. Multi-mode fibres can have a maximum segment length of 2 Km. However, in the case of single-mode fibre, the light in the fibre is unidirectional. Being more expensive, single-mode fibres are used for communication over longer distances. They can have a maximum segment length of 100 Km.

Advantages of Fibre Optic

- Compared to copper wires, optical fibres support larger bandwidth.
- Optical fibres have minimal loss in signal strength when light travels through longer distances.
- Compared to copper wires, optical fibres are lighter in weight.
- Optical fibres are noise-resistant.

Disadvantages of Fibre Optic

- Optical fibres get easily damaged if they are bent.
- Optical fibres are more expensive than copper cables. They also have high maintenance costs.

6.6.4 Radio Waves

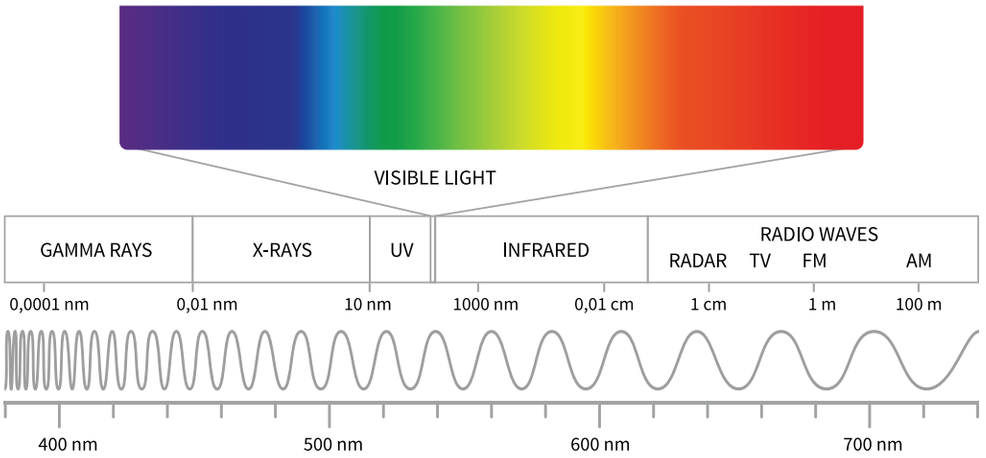
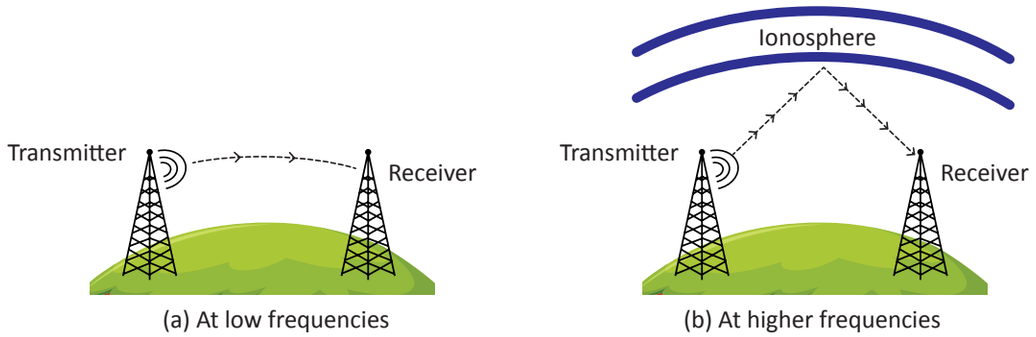


Fig 6.16: Electromagnetic waves spectrum

Radio waves are well suited for long-distance communication. Unlike infrared waves, these waves are capable of penetrating through solid objects. These waves can travel in all directions from the source. So, they are called omnidirectional. The frequency for the radio waves lies in the range of 3KHz–300GHz. This frequency band comprises LF (Low Frequency), MF (Medium Frequency), HF (High Frequency) and VHF (Very High Frequency) bands. At the low frequencies, the radio waves can pass through obstacles (see Fig 6.17(a)). At higher frequencies (HF), radio waves strike free electrons in the ionosphere, and the ionosphere radiates the energy back to earth at the same frequency (see Fig 6.17(b)).



(a) At low frequencies (b) At higher frequencies

Fig 6.17 Radio Waves

Advantages of Radio Waves

- Radio waves can travel longer distances.
- Being omnidirectional, radio waves do not require alignment between transmitter and receiver.

A Disadvantage of Radio Waves

- Electrical equipment may interfere with radio wave frequencies.

6.6.5 Microwave Link

Microwaves are unidirectional and follow a straight path. They impose the constraint that the transmitter and the receiver antennas be aligned. Microwaves cannot pass through solid objects. Therefore, they often require tall transmitter and receiver antennas for proper alignment and line-of-sight transmission. Repeaters are often needed for long-distance communication. Microwave communication is illustrated in Fig 6.18.

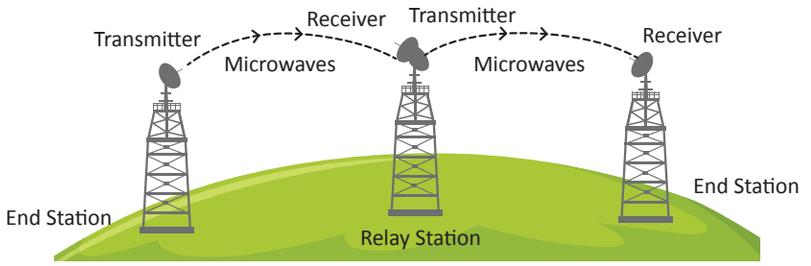


Fig 6.18 Microwave Communication

Since microwaves are unidirectional, two-way communication requires two frequency waves to be transmitted from source to destination and vice versa. Hence, setting up two pairs of aligned antennas is required, i.e., two transmitters and two receivers. The frequency for the microwave lies in the range 1GHz–300GHz.

Advantages of Microwave Communication

- As the microwaves follow a line of sight communication, there is less noise and interference.
- Microwaves can travel long distances.

Disadvantages of Microwaves Communication

- Microwaves cannot pass well through buildings, unlike radio waves of low frequency.
- If the microwaves get out of sync, they may cancel signals. This phenomenon is called multipath fading.

6.6.6 Infrared Waves

Infrared light has frequencies lower than visible red light. Infrared waves find applications in short-range communications such as remote control devices for televisions or stereos. As these waves cannot penetrate solid objects, we cannot use them for communication from one room to another. However, for the same reason, these waves are well suited for security purposes. As they are unidirectional, they require proper alignment between transmitter and receiver. The frequency for the infrared waves lies in the range of 300 GHz to 400 THz.

Advantages of Infrared Waves

- Infrared Waves are suitable for security. For example, one cannot control your television sitting in another room.
- Infrared waves provide fairly substantial bandwidth at relatively low Cost.

A Disadvantage of Infrared Waves

- Infrared waves are not suited for long-distance communications.

6.6.7 Satellite Link

In satellite-based communication, signals are transmitted in a straight path. Frequency for the satellite communication lies in the range 1 GHz–40 GHz. The satellite acts as a tall antenna. It comprises several transponders. Each of them is responsible for listening to some part of the frequency spectrum. On receiving the incoming signal, the satellite amplifies it and broadcasts it at another frequency to avoid interference with the incoming signal. The downward beam sent by satellite can span a few hundred kilometres. Signals follow line-of-sight transmission. Fig 6.19 illustrates satellite communication.

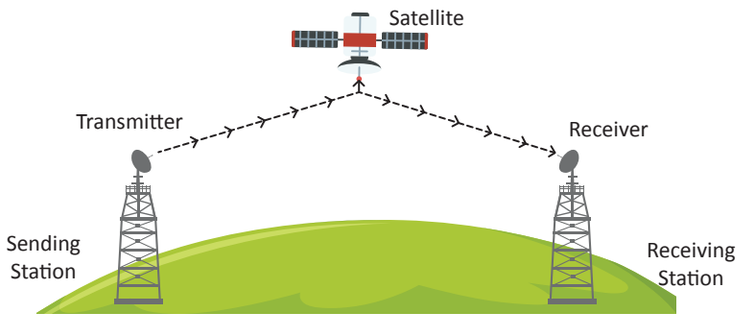


Fig 6.19 Satellite Communication

Advantages of Satellite communication

- Remote areas on the earth can easily communicate with each other.
- Satellite communication does not suffer significantly from noise interference.

Disadvantages of Satellite communication

- Setting up satellites is expensive.
- Since signals transmitted through satellites can be captured by anyone having a receiver, encryption is often required to provide security.

1. Which is the first guided transmission media?
2. Which guided transmission medium works on the principle of light?
3. Which wireless medium is used by your TV remote?
4. What is the frequency range for satellite communication?

6.7 Network Topologies

We already know that a network connects computers and devices (collectively called nodes). The arrangement of nodes in a network is called topology. It describes the placement of various network devices and physical connections between them. We will discuss various topologies of networks in this section.

6.7.1 Bus Topology

In bus topology, various nodes are connected to a long cable, called backbone cable. The nodes are connected to the backbone cable via connectors, also called taps. In Fig 6.20, we show a network that uses bus topology. Messages in a bus topology are broadcast to all nodes. So, the network is called a broadcasting network. Although every node receives the message broadcast by a node, a node knows whether it is the intended recipient by examining the message's destination address. When one device communicates a message, other devices wait for the transmission to complete.

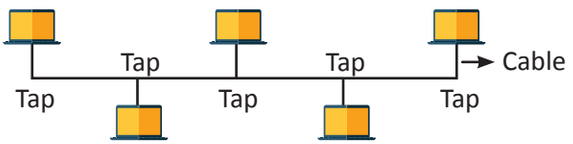


Fig 6.20 Bus Topology

Advantages of Bus Topology

- Setting up bus topology is also easy.
- Bus topology requires less cabling than other topologies such as star and tree (discussed later in this section).
- It is easy to extend the bus network since it requires connecting a new device to the cable through a tap.

Disadvantages of Bus Topology

- When a device transmits a message, a part of the signal's energy gets transformed into heat as it travels through the cable, resulting in a weaker signal.
- If the backbone cable is damaged at a single point, it will disrupt the entire network.
- Detecting the point of fault and its isolation is problematic.

6.7.2 Ring Topology

In the ring topology, all devices are attached through a cable in the form of a ring (in circular fashion). Each device connects to precisely two devices, one on either side. To avoid data collisions, the message to be communicated is transmitted in one direction device by device. Each node on the path examines whether it is the intended recipient.

If not, it intensifies the strength of the incoming signal and forwards (also called transmits or relays) it to the next node, and so on, until the message reaches the destination device. For this purpose, each intermediate node includes a repeater (discussed in the following sections) responsible for regenerating the incoming message. In Fig 6.21, we show a network that uses the ring topology.

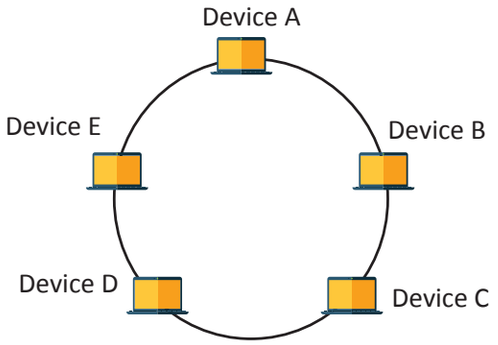


Fig 6.21 Ring Topology

Advantages of Ring Topology

- Adding or removing a device is easy, requiring updating only two connections.
- Faulty devices can be easily located and isolated.
- Since communication is unidirectional, there is no data loss due to collision.

Disadvantages of Ring Topology

- Since data moves only in one direction, a message sent to a device may take more time. For example, consider five devices A, B, C, D and E laid in a ring topology. Sending a message from node A to node E, would require four hops when the data flow is in a clockwise direction. However, the same data transfer would require only one hop in the anti-clockwise direction.
- Breakdown in any part of the ring will disrupt the entire network communication. If a single node fails to function, the entire network will fail to work.

6.7.3 Star Topology

The star topology has a central controller known as a hub. All the nodes are connected to the hub. So, in a network with one hub device, and n nodes, there will be n connections, each connecting the hub to a node. Communication between any two devices must pass through the hub. So, no two devices can communicate directly in a star network. A hub may be of one of the two types: a broadcasting hub or a unicast hub. On receiving a message, a broadcasting hub will broadcast it to all nodes. In contrast, on receiving a message, a unicast device forwards it to the destination node on an appropriate outgoing line. In Fig 6.22, we show a network laid using star topology.

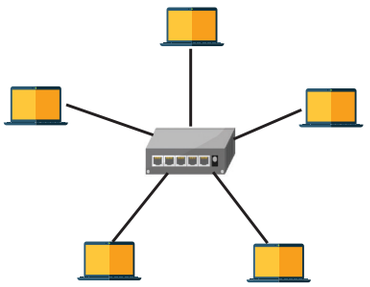


Fig 6.22 Star Topology

Advantages of Star Topology

- Since each device in star topology requires only one link point (I/O port) for connecting it to a hub, it can be easily installed and reconFigd.

- Addition and removal of devices in an existing network are easy.
- Failure of a link will only affect communication with one node. The rest of the network will continue to function as before.
- A faulty node can be easily detected and isolated from the network.

Disadvantages of Star Topology

- There is no point to point communication between devices. Any communication must pass through the hub.
- Failure of the hub will disrupt the functioning of the entire network.

6.7.4 Tree Topology

The tree topology exhibits a hierarchical structure. The central hub at the root of the tree may be connected to other nodes or secondary hubs. Devices in tree topology are connected in a hierarchical structure with one root and several intermediate nodes (devices). The central hub is called an active hub because it includes a repeater that regenerates the signal. However, the secondary hubs simply forward a message to the appropriate outgoing line. Fig 6.23 shows a network that uses the tree topology. When a node wants to communicate with another node, the messages are relayed via central and secondary hubs, as required. For example, in Fig 6.23, to send a message from node A to node C, the secondary hub (Hub2) will forward it directly to node C without involving the central hub (Hub1). However, to send a message from node A to node F, node A will send the message to the secondary hub (Hub2), which will forward it to the central hub (Hub1). Then the central hub will deliver it to the secondary hub (Hub3), which will forward it to node F.

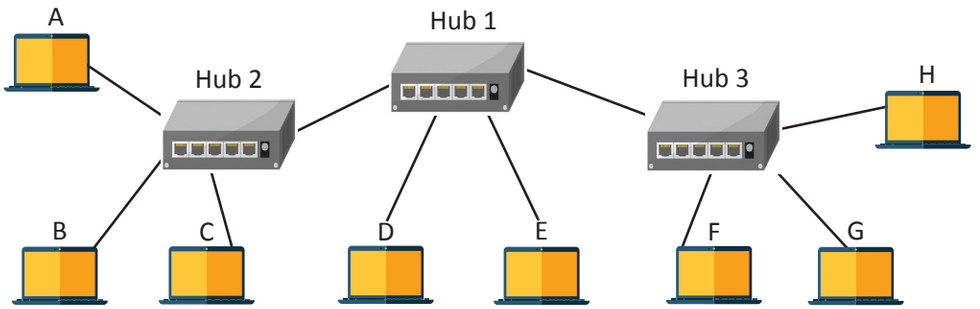


Fig 6.23 Tree Topology

Advantages of Tree Topology

- Tree topology is easy to install and reconFig.
- A faulty node can be easily detected and isolated from the network.
- Being a hierarchical network, it is easily scalable.
- Isolation of a network connected to a secondary hub is easy.

Disadvantages of Tree Topology

- This topology requires more cabling than bus and ring topology.
- There is no point to point communication between nodes.
- If the central hub fails to function, the entire network fails.

1. In which type of network topology failure of any one node brings down the entire network?
2. Which type of topology has Central Node Dependency?
3. Which type of topology exhibits a hierarchical layout?

6.8 Network Types

Networks differ based on their geographical span, communication channels, and purpose. Based on these differentiating parameters, networks may be classified as follows:

6.8.1 PAN

A Personal Area Network (PAN) is a computer network that links computers and gadgets within a person's range. PANs have a short-range, often up to 10 metres (33 feet). For example, a person travelling with a laptop, a Personal Digital Assistant (PDA), and a portable printer may use a PAN to connect them all. A PAN may be further connected to the Internet or other networks. Fig 6.24 shows a PAN.

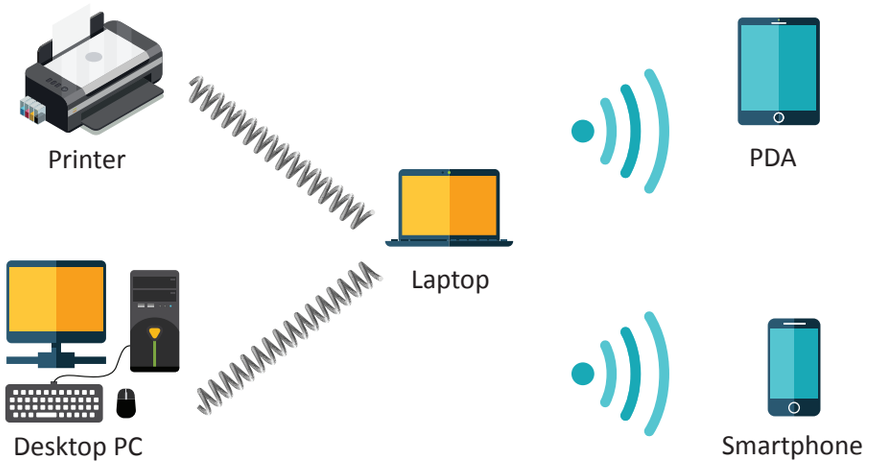


Fig 6.24 Personal Area Network

6.8.2 LAN

A Local Area Network (LAN) is typically owned privately by an organisation. It can span an area of about 1 Km. A LAN is usually established within a building or campus to facilitate resource sharing amongst its users. These resources may be hardware resources such as printers or software resources such as application programs or data. One of the computers often acts as a server and responds to requests from other computers. The computers requesting the server are called clients. For example, suppose you have a printer and a few computers. If all computers need a printer, they can access it via a LAN.

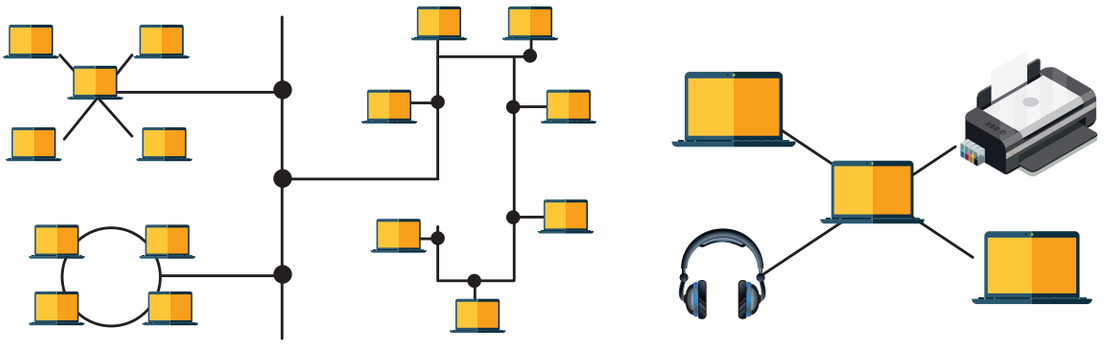


Fig 6.25 Examples of LAN

The most commonly used topologies in LANs are bus, ring and star. The transmission media used in LANs are twisted-pair, coaxial cable, and fibre optic. Fig 6.25 (a) shows a LAN within a building. Fig 6.25 (b) shows a LAN within a campus. LANs typically operate at speeds in the range of 10 Mbps to 1 Gbps.

6.8.3 MAN

A Metropolitan Area Network (MAN) can span a radius of up to 50 kilometres. A MAN typically connects various local area networks within a city. A single organisation may own a MAN or it may be a public network that many individuals or organisations use. MANs usually operate at higher speeds than LANs. Typical transmission media for a MAN include Ethernet cable, microwave, radio wave, and fibre optic. Fig 6.26 shows a MAN that connects server nodes of multiple LANs.

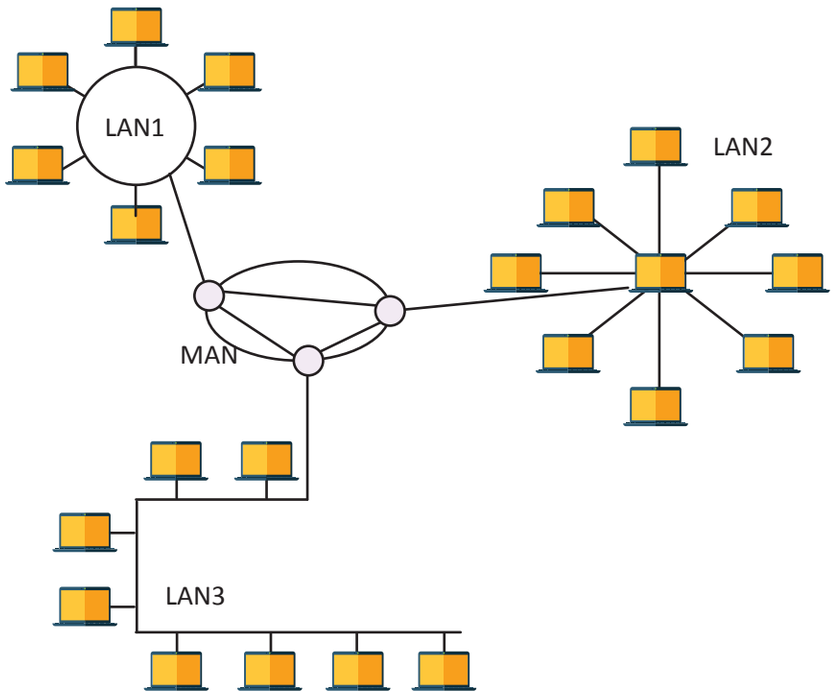
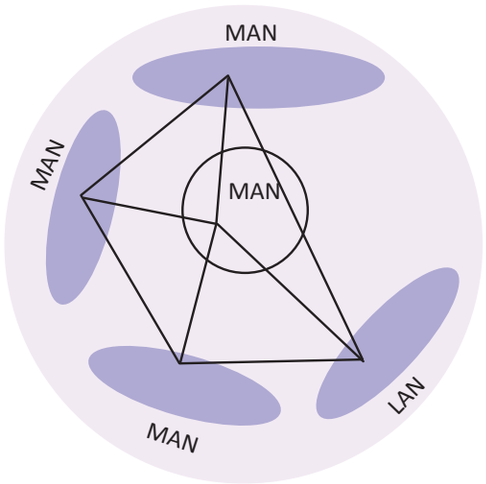


Fig 6.26 MAN

6.8.4 WAN

A Wide Area Network (WAN) can span a radius of up to 1000 km. Therefore, a WAN can connect across countries and continents. WANs are used for long-distance communication and are well-suited to connect remote areas. A WAN connects various local and metropolitan area networks using network devices such as bridges and routers. The Internet is a common example of a WAN. A WAN is typically managed and owned by a single organisation. Setting up a WAN is costlier than setting up a LAN. WANs typically use leased lines and satellite links as transmission media. Fig 6.27 shows a WAN that connects various LANs and MANs.



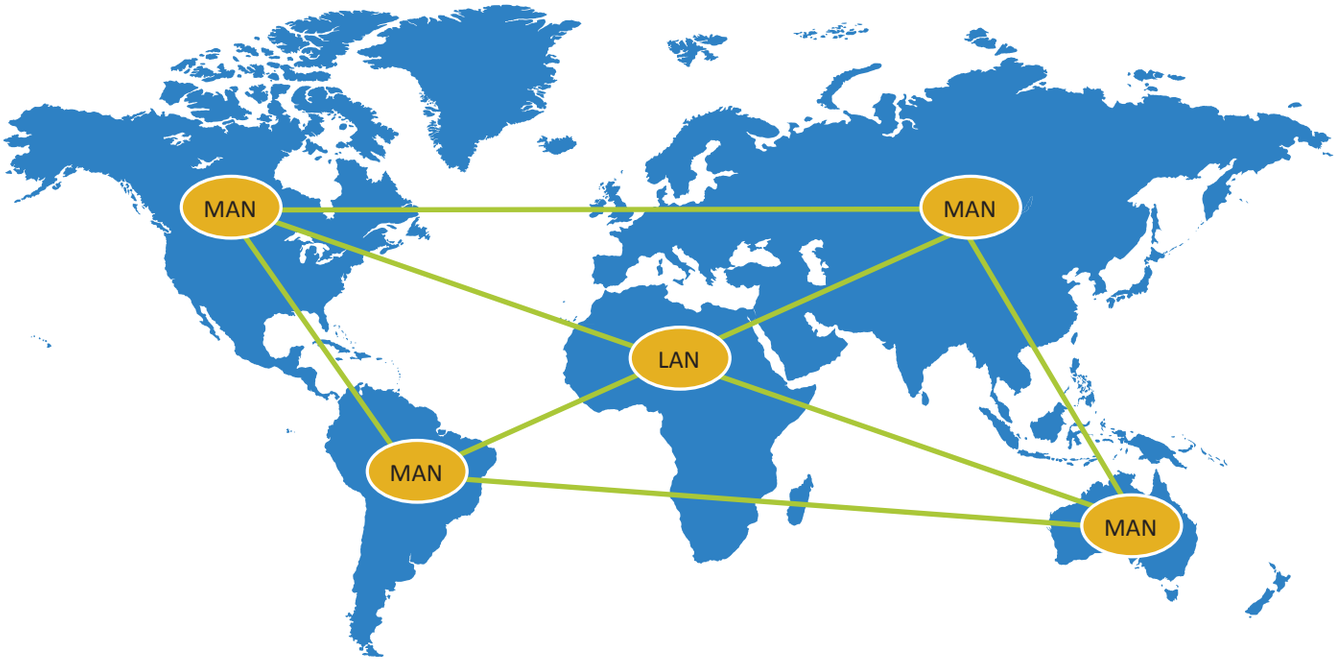


Fig 6.27 WAN

6.9 Network Devices

Creating a network requires several network devices, each playing a specific role. In the following paragraphs, we discuss some important network devices.

6.9.1 RJ-45 Connector

RJ-45 (*RJ* denotes *Registered Jack*, and *45* refers to an interface standard) connector is used to connect the devices to the network cables. An RJ-45 connector has eight pins connected to eight wires of the incoming cable. An RJ-45 connector (plugs) fits into RJ-45 jacks, as shown in Fig 6.28. RJ-45 connectors are attached to Ethernet cables for connecting computers to the local area networks.

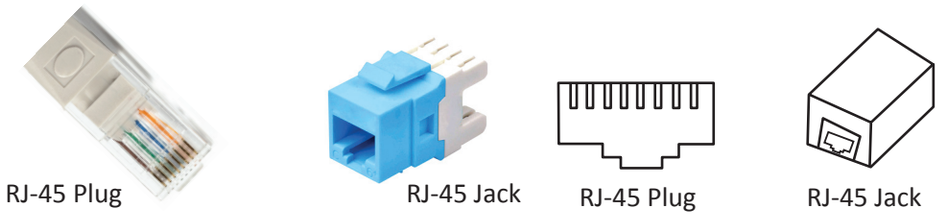


Fig 6.28 RJ45 plug and jack

6.9.2 Network Interface Card (NIC)

A network interface card is a piece of hardware (also known as a network adapter, network interface controller, or LAN adapter) that allows a computer to connect to a network. It is a circuit board installed in a Peripheral Component Interconnect (PCI) expansion slot on a computer's motherboard. NIC supports both wired and wireless communications. Following are the two types of NICs:

1. Wired (Ethernet)
2. WiFi Card

Ethernet Card

Ethernet cards, also known as Ethernet adaptors, connect networks using cables. They provide high-speed network connections in the range 10 Mbps to 10 Gbps. An Ethernet card makes it possible for two systems on a network to communicate with each other. The most common connector used with Ethernet cards are RJ-45. Fig 6.29 depicts the Ethernet card that will be installed internally to the motherboard. The RJ-45 connector will go inside the RJ-45 LAN port shown in the Fig and the activity LEDs are lit when the device is communicating with the Ethernet network connected with it.

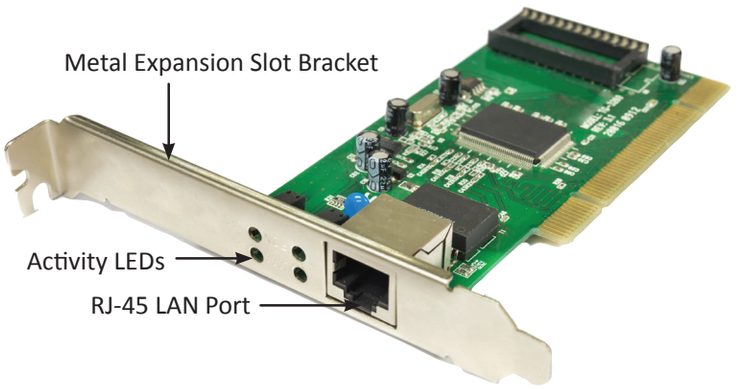


Fig 6.29 Ethernet Card

WiFi Card

A WiFi card provides wireless connectivity and is used to form wireless LANs. Systems with WiFi cards can connect wirelessly to networks. Fig 6.30 depicts two types of WiFi cards: (a) Internal (installed internally on the motherboard) and (b) External (usually inserted in the Universal Serial Bus (USB) port). Communication in a wireless network occurs via radio waves. A router with one or more antennae is required for a wireless network to function. A PCI-Express WiFi card is often installed on the motherboard of a desktop computer.



(a) Internal WiFi card



(b) External WiFi Card

Fig 6.30 WiFi Card

6.9.3 Repeater

With an increase in distance, the strength of the signal decreases. A repeater intensifies the strength of the incoming signal before transferring it further. As a repeater restores the strength of a signal, it is also called a regenerator.

A repeater is required when the length of a network cable exceeds a particular distance. For example, if the medium is a twisted pair cable, a repeater is required after every 100 metres (328 feet). Suppose a signal can travel only 100 metres, so connecting three cables with repeaters will enable the signal to cover 300 metres. In Fig 6.31, we show the use of the repeater.

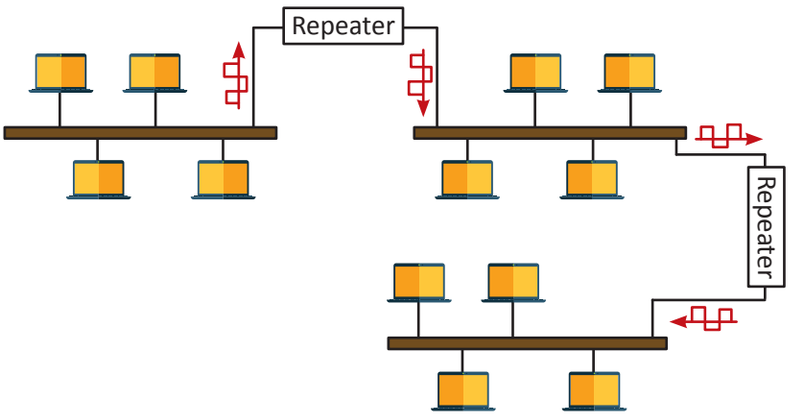


Fig 6.31 Repeater

6.9.4 Hub

A hub comprises several ports, each of which connects to a single cable. A hub transmits the data arriving on any incoming line to all the connected lines. A hub is used for communication up to 200 meters. A hub may be categorised as passive or active. A passive hub copies a signal on the outgoing lines as it is received. An active hub intensifies the incoming signals before forwarding them. A hub may support 8, 12, or 24 RJ-45 connections. If the two nodes connected to a hub attempt to send a message simultaneously, a collision is said to occur. However, it is easy to resolve the problem of collisions. Each node waits for a randomly chosen amount of time and communicates again so that a collision is improbable to follow another. Fig 6.32 illustrates the use of a hub in a network that uses star topology. Node D sends a message (addressed to node B) to the hub. On receiving the message from node D, the hub forwards it to all the connected nodes. As the message is intended for node B, node B accepts it, while all other nodes discard it.

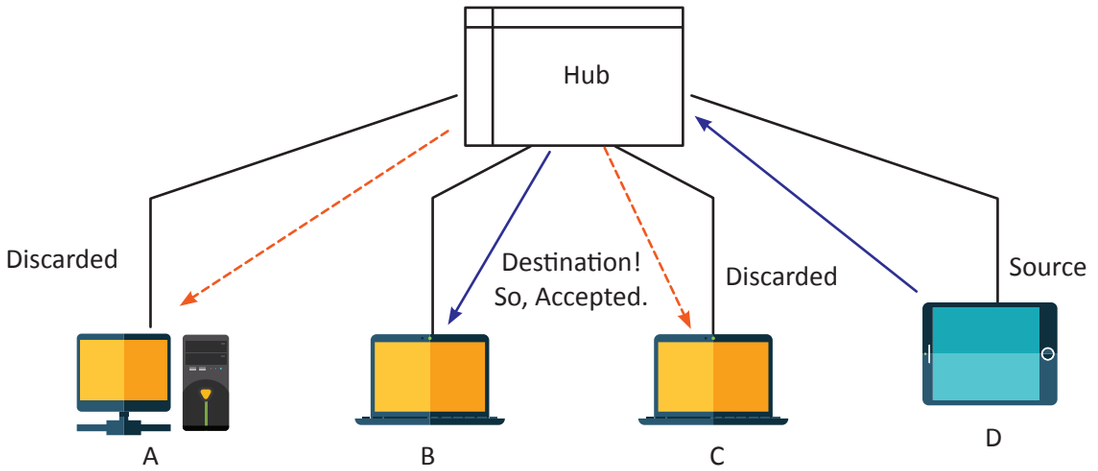


Fig 6.32 Hub

6.9.5 Bridge

A bridge is a multiport device that connects two or more local area networks (LANs) to form an extensive network. Thus, the devices on one LAN segment can communicate with the devices on another segment. Unlike a hub, a bridge is an intelligent device that exercises discretion while forwarding data. When a bridge receives a frame from a

LAN segment, it examines the destination address of the incoming frame and forwards it to the appropriate LAN segment. Before transmitting a frame to the outgoing line, it regenerates the signal. Different LAN segments may operate at different speeds.

Fig 6.33 illustrates the use of the bridge in a network. Suppose node A on LAN segment 1 wishes to send a packet to node G on segment 3. On receiving this packet from node A, the bridge will examine its destination address and forward it only to LAN segment 3.

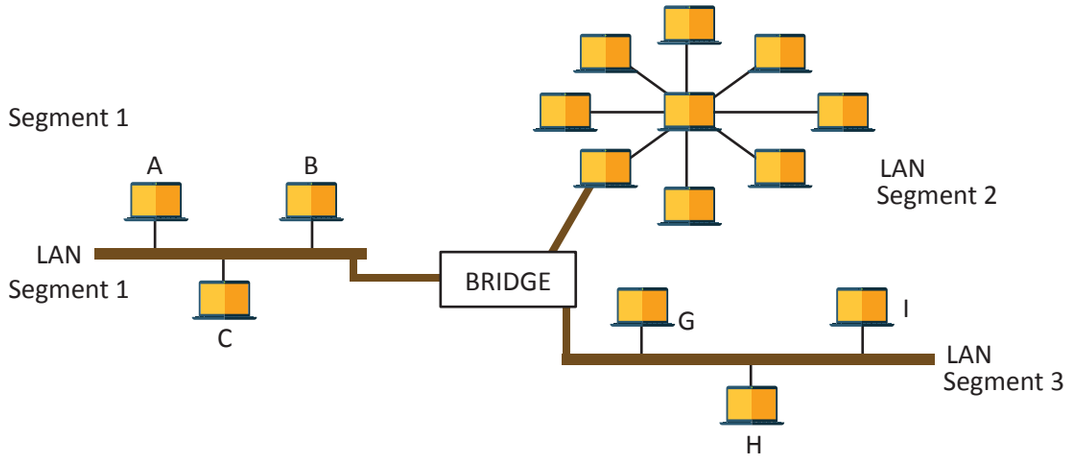


Fig 6.33 Bridge

6.9.6 Switch

Like hubs, switches also connect individual computers. Each computer is connected to a unique port on a switch. However, it does not transmit an incoming *frame* to all the computers attached to it. When a switch receives a frame from a LAN segment, it examines the incoming frame's destination address and forwards it to the appropriate node. However, on receiving a corrupted packet, the switch asks the sender to transmit it again. A store and forward switch would wait until it receives a complete frame and then transmit it to the destination. Alternatively, a cut-through switch initiates the process of forwarding the data as soon as it receives the frame's destination address. Fig 6.34 shows the use of a switch in a network. If node A wishes to send a message to node G, it will pass the message to the switch, which will transmit it on the line that connects to node G.

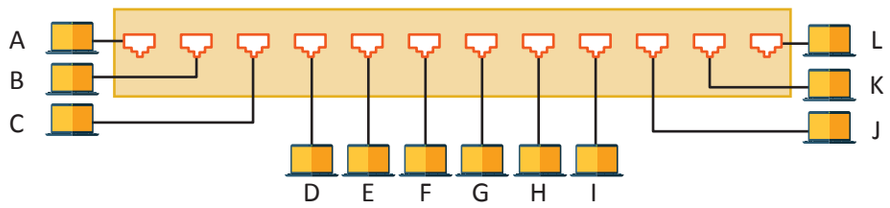


Fig 6.34 Switch

6.9.7 Modem

Whereas computers store and process data digitally, traditional telephone lines deal with analog signals. A modem connects a computer to the Internet using telephone lines. It combines the modulation and demodulation functions by transforming a digital signal to an analog signal and vice-versa. Digital data to be sent from a computer is modulated for transmission over telephone lines. Similarly, at the receiving end, analog data is demodulated to digital form. A modem's speed is expressed in bits per second or in terms of baud rate.

A modem may be placed internally in an expansion slot of a computer, or it may be connected externally to a port. Nowadays, wireless modems are also in use that do not require a physical connection with a computer. Fig 6.35 explains the use of modems in a network. Label two computers on Left (A) and Right (B). Digital data sent by node

A passes through the modem, which converts it into an analog signal. The analog signal generated is transmitted via the telephone network. At the other end, a modem converts the signal back into digital form, which is sent to node B. A modem also compresses the data before sending and uncompresses it at the receiving end.

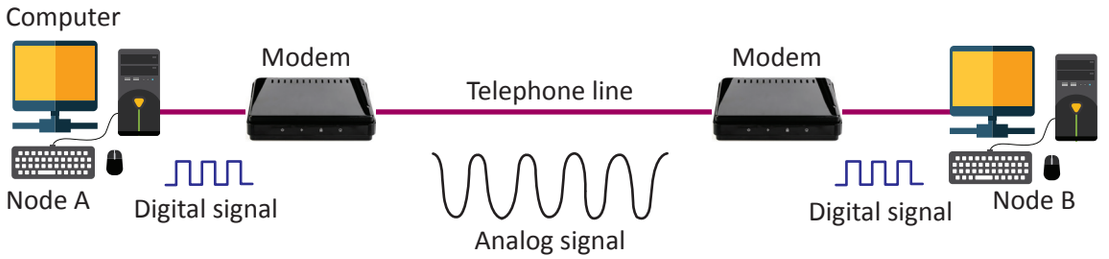


Fig 6.35 MODEM

6.9.8 Router

Routers are used for connecting various networks. A router may be used to interconnect several networks, possibly having different topologies. Also, a router may be used to connect several nodes (computers, mobile phones, tablets, etc.) to the internet. For example, a WiFi router is often used in homes to connect to the Internet. Each router has a routing table that it consults for forwarding an incoming packet to an outgoing line. It examines the destination address (IP address) of a received packet and sends the packet to the appropriate network, keeping in view the current information about the network. Fig 6.36 shows a router that is connected to two switches. In Fig 6.37, we see several nodes, connected to a router. On receiving a packet from a node, it passes it on to the Internet.

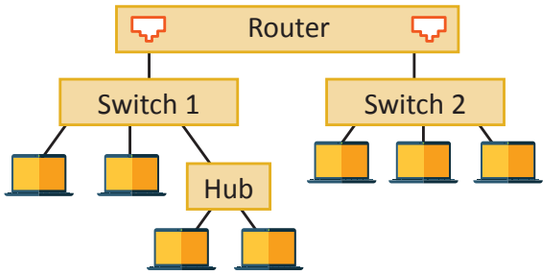


Fig 6.36 Router

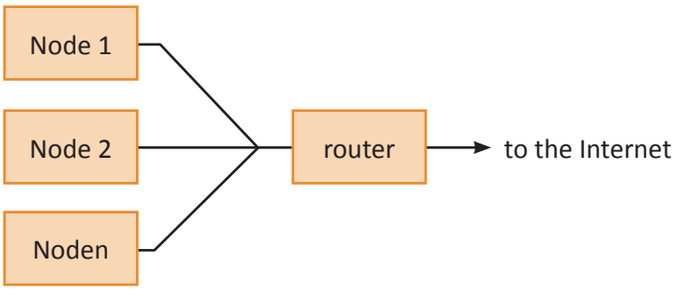


Fig 6.37 Use of Router

6.9.9 Gateway

A gateway is a hardware device that operates as a "gate" between networks. A gateway connects the networks that use different protocols. Thus, a gateway may be thought of as a router with protocol conversion software. The data from a network that uses a specific protocol is transformed to make it compatible with the outgoing line's protocol. Fig 6.38 shows a switch that acts as a gateway for the data flowing from the server to any node (A–E) or vice versa. Also, data from a node (say D) to a node (say A) will flow via the gateway.

Fig 6.39 shows two different networks connected via a router. Here, the router acts as another gateway. If node B wants to send data to node G, the data will flow from node B to its server (gateway) passing through the router (gateway). Subsequently, the router will pass the data to the server of the second network, which will pass it on to node G.

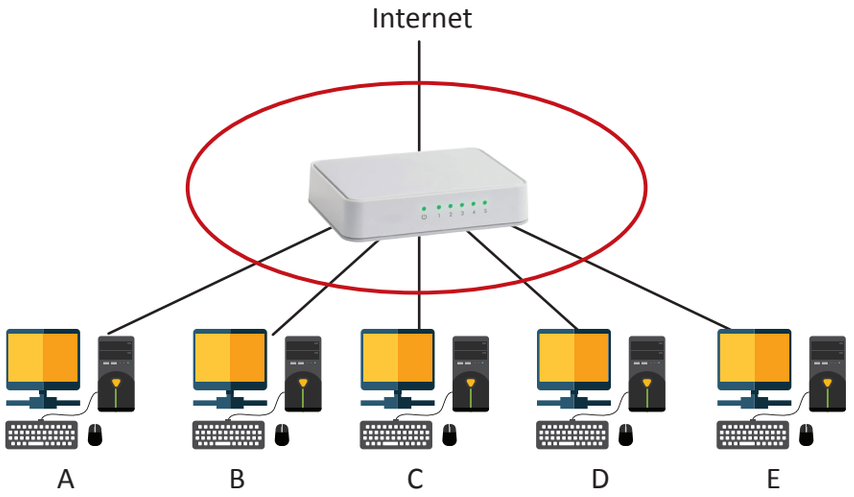


Fig 6.38 Switch as Gateway

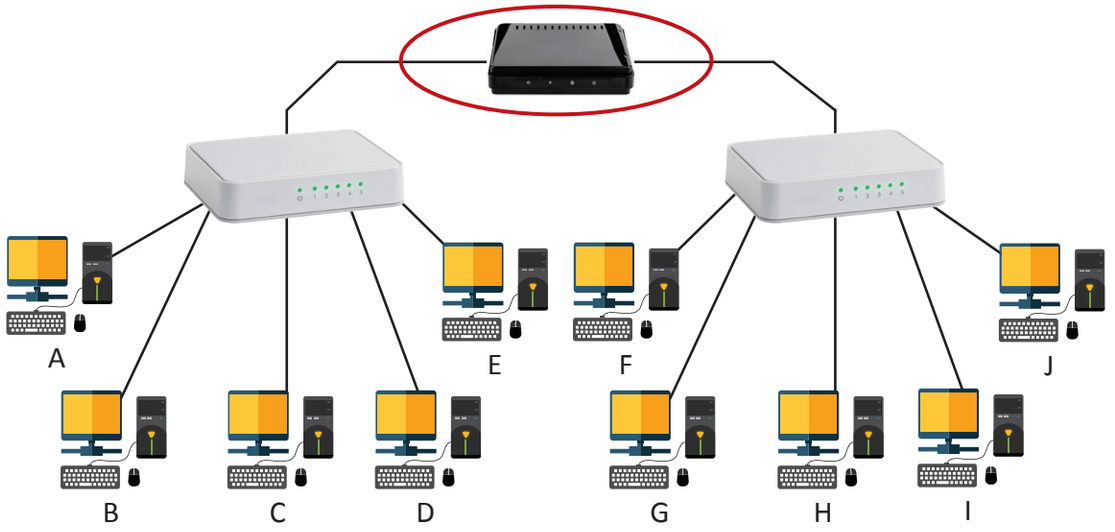


Fig 6.39 Router as Gateway

1. Which network (LAN / MAN/PAN/WAN) covers the geographical area of a university campus?
2. Which is the largest WAN existing on Earth?
3. Name the device that connects two networks with different protocols.
4. Can a router be replaced with a bridge?
5. Which network device is used to connect computers to the local area networks?

Let's Summarise

- A computer network is defined as a collection of interconnected computers that provide a means of communication amongst the computers to share hardware and software resources.
- The Internet is the world's largest computer network.
- The main components of a network are the entities participating in the network, the medium through which communication takes place, and the content to be communicated.
- A sender is the node that sends the data.
- A receiver is the node that receives the data.
- A message is the information being communicated.
- A channel is the medium through which data is transmitted.
- The unique address of each participating node in a computer network is known as a Media Access Control address (MAC address).
- An application program that provides a service of managing the devices and resources on the network is called a server.
- Setting up a telephone circuit is an example of circuit switching.
- In case of packet switching, a message to be transmitted is divided into small chunks called packets.
- ARPANET was the First Public Packet-switched computer network.
- The government and private organisations that provide Internet connectivity are called Internet Service Providers (ISPs).
- A newsgroup is a discussion forum where users of similar interests can post information and reply to others' posts.
- Email is a form of communication in which we can send and receive messages electronically.
- Remote login is a way of accessing and using hardware and software available on a remote computer from any other computer.
- Chat refers to a means of real-time communication between two or more users .
- Video conferencing allows multiple users to exchange video and voice messages in real-time.
- Cloud computing enables storage and computing services over the Internet.
- Interspace is a client/server software tool that allows users to connect online in dynamic 3D worlds using real-time audio, video, and text chat.
- A subnet is part of a more extensive network that comprises a grouping of network devices connecting various transmission lines.
- Data communication refers to exchanging information between two devices.
- Bit rate or data transfer rate, is the number of bits transmitted per second.
- Baud rate refers to the number of signal units transmitted in a second.
- A cycle in a signal refers to a recurring pattern.
- The frequency of a signal is defined as the number of cycles per second.
- The bandwidth of a signal is defined as the range of frequencies it comprises.
- A data rate is the amount of data that goes through a medium per unit of time.
- Simplex transmission is a mode of transmission in which the data travels in one direction only.

- Half-duplex transmission is a mode of transmission in which the data travels in both directions but not at the same time.
- Full-duplex transmission is a transmission mode in which the data can be transmitted in both directions simultaneously.
- A twisted pair cable comprises two copper wires twisted around each other.
- A coaxial cable comprises a copper core covered by an insulating material that is further surrounded by an outer conductor.
- Fibre-optic cables transmit signals in the form of light.
- Radio waves are well suited for long-distance communication and are capable of penetrating through solid objects.
- Microwaves are unidirectional and follow a straight path and they impose the constraint that the transmitter and the receiver antennas be aligned.
- Infrared light has frequencies lower than visible red light and it is used in short-range communications such as remote control devices for televisions or stereos.
- In satellite-based communication, signals are transmitted in a straight path.
- The arrangement of nodes in a network is called topology.
- In bus topology, various nodes are connected to a backbone cable via connectors.
- In the ring topology, all devices are attached through a cable in the form of a ring.
- The star topology has a central controller known as a hub and all the nodes are connected to the hub.
- Devices in tree topology are connected in a hierarchical structure with one root (central hub) and several intermediate nodes (devices).
- A PAN is a computer network that links computers and gadgets within a person's range i.e., often up to 10 metres (33 feet).
- A LAN is a computer network that is usually established within a building or campus to facilitate resource sharing amongst its users and, can span an area of about 1 Km.
- A MAN is a computer network that typically connects various local area networks within a city, and it can span a radius of up to 50 kilometres.
- A WAN connects various local and metropolitan area networks using network devices such as bridges and routers within a radius of up to 1000 km.
- The RJ45 connector has eight pins to connect the devices to the eight wires of the network cables.
- A network interface card is a piece of hardware that allows a computer to connect to a network.
- Ethernet cards connect networks using cables and they provide high-speed network connections in the range 10 Mbps to 10 Gbps.
- A WiFi card provides wireless connectivity and is used to form wireless LANs.
- A repeater intensifies the strength of the incoming signal before transferring it further.
- A hub transmits the data arriving on any incoming line to all the connected lines up to 200 meters.
- A bridge is a multiport device that connects two or more local area networks (LANs) to form an extensive network.
- A switch connects individual computers to a unique port on it.
- A modem connects a computer to the Internet using telephone lines and combines the modulation and demodulation functions by transforming a digital signal to an analog signal and vice-versa.

- A router may be used to interconnect several networks, possibly having different topologies.
- A gateway is a hardware device that operates as a "gate" between networks and connects the networks that use different protocols.

Solved Exercises

A. Multiple Choice Questions

1. The devices on the network are connected to each other via _____.
a. Transmission Media b. keyboard c. Computer d. Mouse

2. Which device is also known as an intelligent hub?
a. Switch b. Hub c. Router d. Gateway

3. Which of the following can be a *node* in a network?
a. RJ-45 b. Printer c. Data d. Fibre optic cable

4. Which of the following is a measure of the capacity of a communication channel?
a. Bandwidth b. Bit capacity c. Baud rate d. Data flow

5. How many hertz are there in 10 MegaHertz?
a. 10^5 b. 10^6 c. 10^8 d. 10^7

6. When a binary signal is being used, data transfer rate is equal to _____.
a. One bit b. Four bits c. Baud Rate d. 20 bits

7. Which of the following is required for the transmission of data between two computers?
a. Communication hardware b. Transmission medium c. Protocol d. All of these

8. Which of the following transmission mediums has the highest data transfer speed?
a. Twisted Pair Cable b. Coaxial Cable c. Optical Fibre Cable d. Parallel cables

9. Which of the following metals is used as a conductor in a coaxial cable?
a. Copper b. Silver c. Gold d. Aluminium

10. Which type of cable is typically used in a LAN?
a. Twisted pair cable b. Fibre optic cable c. Coaxial cable d. A pair of parallel cables

11. Which network device broadcasts the message received to all nodes in the network?
a. Router b. Hub c. Repeater d. Switch

12. Which of the following can be used for communication through a standard modem?
a. Cable lines b. Telephone lines c. Both A and B d. None of these

13. In a WAN, a router can be _____.
a. Only wired b. Only wireless
c. Either wired or wireless d. None of these as a router is used only in a LAN

14. Out of the following, which is the fastest wireless transmission medium of communication?
a. Coaxial Cable b. Infrared c. Ethernet Cable d. Microwave

15. Which of the following transmission waves can penetrate through solid objects or walls?
a. Radio Waves b. Microwaves c. Infrared waves d. None of these

16. In which of the following topologies the entire network load is on a single central cable?
a. Star b. Bus c. Tree d. Mesh

17. In which of the following network topologies are all nodes connected to a single central node?
a. Bus b. Tree c. Star d. Ring

18. In which topology failure of any one node brings down the entire network?

a. Bus b. Ring c. Star d. Tree

B. State whether the following statements are True or False:

1. Each participating node in a network has a unique MAC address. _____

2. The data rate is measured in hertz. _____

3. An optical fibre transmits data in one direction only. _____

4. Microwaves can pass through solid objects. _____

5. A bus topology is easy to expand. _____

C. Fill in the blanks.

1. Communication media can be broadly categorised as _____ and _____.

2. In case of _____ switching, all data packets follow the same path from sender to receiver.

3. In _____ topology, all nodes are connected to a central device.

4. A collection of interconnected nodes that communicate via some channel is a _____.

5. A _____ is a discussion forum where users of related interest can post information and reply to others' posts.

6. The number of signal units transmitted in a second is known as _____.

7. The frequency of a signal on a transmission medium is expressed in _____.

8. Devices in tree topology are connected in a _____ structure.

9. A _____ card provides wireless connectivity and is used to form wireless LANs.

10. A _____ can alternatively be defined as a router with protocol conversion software.

D. Answer the following questions:

1. Give any two applications of computer networks.

Ans. collaborative learning and research, video-conferencing, online gaming, e-shopping, e-governance.

2. What is data communication?

Ans. Data communication is the exchange of data between two or more networked devices.

3. Name the four components of a communication system.

Ans. Sender, Receiver, Message and Channel

4. Write one advantage of Coaxial cable.

Ans. Less susceptible to electromagnetic interference.

5. Give one advantage and one disadvantage of twisted-pair cables.

Ans. Disadvantage: Due to high attenuation, signals cannot be transported over long distances without repeaters.

Advantage: It is easy to install and inexpensive.

6. Give any two ways to connect devices wirelessly.

Ans. Bluetooth, Wi-fi

7. Among LAN, MAN and WAN, which can cover the maximum area?

Ans. WAN

8. Give three examples of communication media.

Ans. Ethernet cable, Microwaves, satellite link.

9. What is Baud Rate?

Ans. It is a measure of the number of signals transferred per second over a communication channel.

10. Define the term topology.

Ans. The physical arrangement of the computer systems in a computer network connected via communication medium is called topology.

11. What are the disadvantages of bus topology?

Ans. i. Locating a fault is difficult

ii. It becomes slow with an increase in the number of nodes

12. Give three examples of the physical medium of communication.

Ans. Telephone line, coaxial cable and optical fibre.

13. A user wants to upload a text document at the rate of 15 pages/20 seconds. What will be the required data rate of the channel? (Assume 1 page contains 800 characters and each character is equal to 8 bits)

Ans. Required data rate= (15*800*8)/20

= 4800 bps

= 4.8 Kbps

14. Write functions of a modem.

Ans. Functions of modem in computer networking—

- Data compression
- Modulation/demodulation of signals

15. Define a router.

Ans. A router is a network device that can receive the data, examine its destination address, and transmit it to other networks. It connects a local area network to the internet.

16. Differentiate between Optical Fibre and Ethernet Cable.

Ans. Optical fibres: These cables are made up of many thin strands of glass that transmit data using light signals.

Ethernet Cable: Typically twisted copper wires are used as Ethernet cable. Although coaxial cables and fibre optic cables are also used as Ethernet cables occasionally.

17. Differentiate between a router and a switch.

Ans.

Router	Switch
1. The main objective of a router is to connect to various networks.	**1.** The main objective of a switch is to connect various devices in a local area network.
2. A router is used in all types of networks	**2.** A switch is typically used in LANs.

18. Differentiate between a router and gateway.

Ans.

Router	gateway
1. It is used to manage and forward data packets to computer networks.	**1.** It is a hardware device that acts as a gate between the networks.
2. Works by installing routing details for multiple networks and routes the traffic based upon the destination address.	**2.** Ensures compatibility between the protocols of the incoming and outgoing traffic.

19. Write a short note on Radio wave transmission.

Ans. When two terminals are connected using radio frequencies, the communication is known as radio wave transmission. Radio wave transmission has:

- Transmitters: Devices that transmit signals.
- Receivers: Devices that receive signals.

20. ABC University of Uttarakhand is setting up a secured network for its campus at Rishikesh for operating its day-to-day office & web-based activities. They are planning to have network connectivity between four buildings. Answer the question (a) to (d) after going through the building positions in the campus and other details given below:



Distance between various buildings:

Buildings	Distance (in m)
Professional Studies to Admin	50
Admin to Sports Centre	60
Sports Centrre to Academic	70
Academic to Professional Studies	80
Admin to Academic	160
Professional Studies to Sports Centre	160

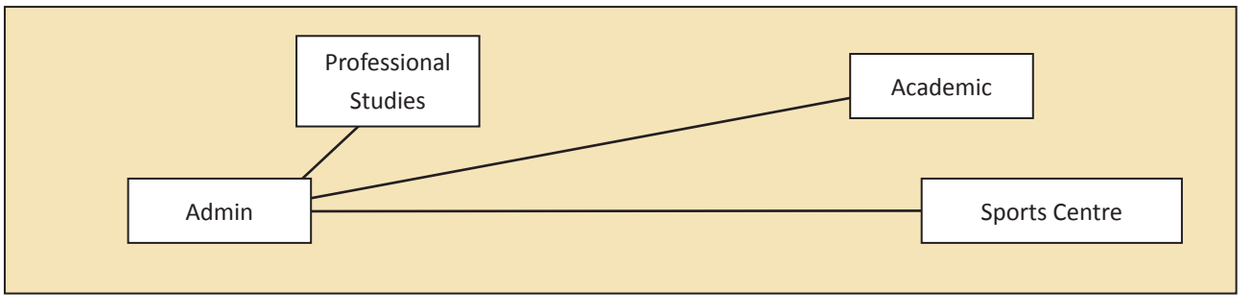
Number of computers in each building:

Buildings	No of Computers
Admin	140
Professional Studies	70
Sports Centre	54
Academic	81

As a network expert, you are required to give best possible solutions for the given queries of the university administration:-

(i) Suggest cable layout for the connections between the various buildings,

Ans.



(ii) Suggest the most suitable building to house the server of the network of the university,

Ans. Admin building, as it has a maximum number of computers.

(iii) Suggest the placement of the following devices with justification:

a. Switch/Hub

b. Repeater

Ans. Switch will be placed in each building to form a LAN, Repeater will be placed between Admin and Academic buildings as the distance is more than 100 m

(iv) Which of the following technologies should be used for fast access to the Internet?

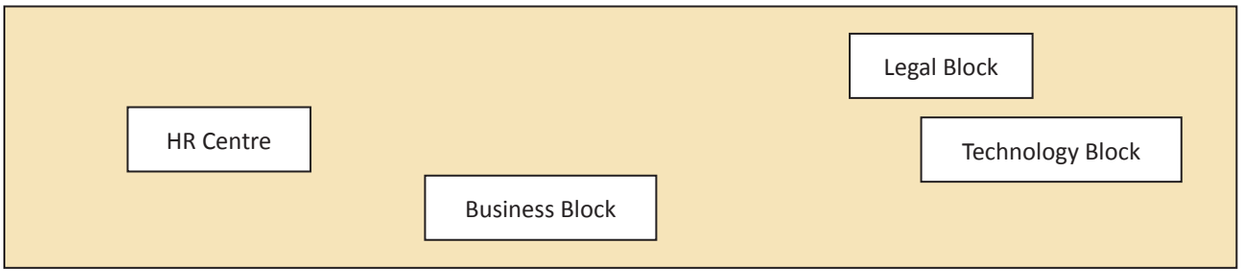
a. Optical Fibre

b. Coaxial cable

c. Ethernet Cable

Ans. Optical Fibre

21. SkyHigh Worldwide Ltd. is setting up its corporate office in Delhi. It has has four blocks including Human Resource Centre as shown in the diagram below:



Block to Block distances between various blocks/centre is as follows:

Block	Distance (in m)
HR Centre to Business Block	30
Business Block to Technology Block	95
Technology Block to Legal Block	50
Legal Block to HR Centre	110
HR Centre to Technology Block	190
Legal Block to Business Block	140

No of computers in each block:

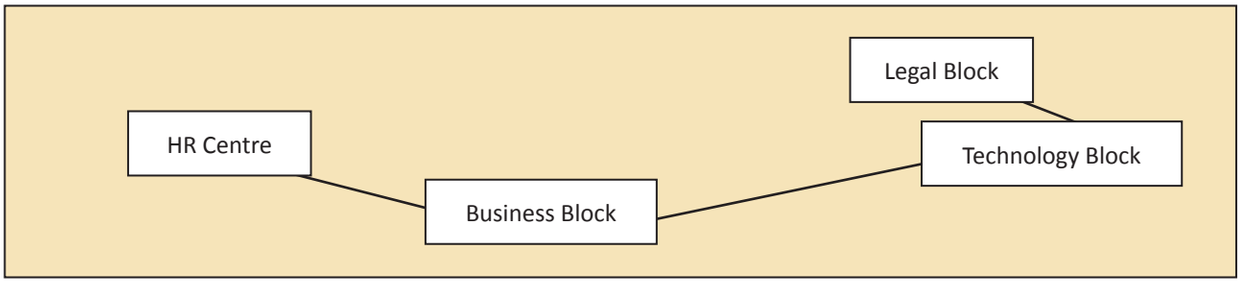
Block	No Of Computers
HR Centre	60
Business Block	71
Legal Block	33
Technology Block	104

(i) Suggest the most suitable place (i.e., Block/ Center) to install the server of this University with a suitable reason.

Ans. Technology Block, as it has most number of computers

(ii) Suggest an ideal layout for connecting these blocks/centres for a wired connectivity.

Ans.



(iii) Which device will you suggest to be placed/installed in each of these blocks/centres to efficiently connect all the computers within these blocks/centres?

Ans. Hub / Switch

(iv) Suggest the placement of a repeater in the network justification.

Ans. A repeater should be placed between Business Block and Technology Block, as the distance is 95 metres. There are chances that the signal will lose its strength.

(v) A university located in Delhi is planning to connect its admission office in Mumbai, which is more than 11150 km from Delhi. Which of LAN, MAN, or WAN networks will you recommend? Justify your answer.

Ans. WAN, as the given distance is more than the range of LAN and MAN.

Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): A modem is a device used to connect to Internet.
Reasoning (R): Modem stand for Modulator Demodulator

2. Assertion (A): When your mobile is connected to your laptop via bluetooth, a PAN is formed.
Reasoning (R): A PAN stands for Personal Area Network

3. Assertion (A): Router and gateway both are similar network devices.
Reasoning (R): Multiple LANs and WANs are joined through network devices to form a WAN.

4. Assertion (A): A network is formed when multiple devices (called nodes) are connected together.
Reasoning (R): A node is connected to a network through an RJ-45 connector.

Ans. 1. b 2. a 3. d 4. a

Assessment

A. Multiple Choice Questions

1. What is the size of IPv4 address?
a. 32 bits b. 64 bits c. 34 bytes d. 64 bytes

2. Which of the following network devices is responsible for forwarding data from one device to another on a network?
a. NIC b. RJ45 c. Router d. Repeater

3. Which of the following is not a communication medium?
a. Satellite b. Microwave c. Radiowave d. Wi-Fi

4. Which of the following is not an example of unguided media?
a. Microwave b. Optical fibre c. Radiowave d. Infrared waves

5. In which type of switching technique, first a connection is established between sender and receiver and then data is transferred?
a. Packet switching b. Circuit switching c. Message switching d. Parallel switching

6. Which of the following cable consists of an inner copper core and a conducting outer sheath?
a. Optical fibre
b. Unshielded Twisted pair cable
c. Coaxial cable
d. Shielded twister pair cable

7. Which of the following transmission media has the highest transmission speed?
a. Co-axial cable b. Fibre optic c. Twisted pair cable d. Series Cable

8. Which of the following transmission media is NOT susceptible to external interferences?
a. Co-axial cable b. Twisted pair cable c. Series Cable d. Fibre optic

9. Name the wired transmission medium that uses the principle of total internal reflection of light for the transmission of data.
a. Twisted Pair Cable b. Optical Fibre c. Co-axial Cable d. Electromagnetic cable

10. Which of the following devices is an eight pin connector that is used to connect each computer to a local area network?
a. RJ 45 b. Router c. Gateway d. Modem

11. Rita wants to transfer data to another office in the same city. Which type of network is formed?
a. LAN b. MAN c. WAN d. PAN

12. Which of the following devices will you suggest for connecting all computers within a building?
a. Modem b. Repeater c. Switch/hub d. Gateway

13. Which of the following is NOT true regarding IP address?
a. It can be used to trace the actual physical location of a device in a network.
b. A printer connected to a network will also have IP address.
c. Two devices on a network can have same IP address.
d. An IP address is a set of numbers separated by a dot.

14. Which of the following type of topology connects all computers with a single cable called backbone?
a. Star b. Bus c. Tree d. Mesh

B. State whether the following statements are True or False:

1. WAN covers smaller geographical area than MAN. _____
2. A network that consists of both LAN and MAN is called a WAN. _____
3. If the computer of a school library is connected to Internet, then the type of network will be WAN. _____
4. Communication is unidirectional in case of ring topology. _____
5. Infrared light has frequencies greater than visible light. _____
6. Bandwidth is the difference between lowest and highest frequencies. _____
7. In case of simplex transmission, data transfers in both directions on a network. _____
8. Guided medium is the physical medium of data transfer. _____
9. Optical fibres are noise resistant. _____
10. An active hub intensifies the signal before forwarding it further. _____

C. Fill in the blanks.

1. _____ refers to Wireless Fidelity and it enables us to connect to the iSP without any cable.
2. The technique of switching in which data is divided into small packets is _____ switching.
3. The speed at which data transfer takes place over a network is _____.
4. In the case of fibre optic transmission, data transfers in the form of _____.
5. A local telephone network is an example of _____ switching.
6. Fault detection and isolation is easy in case of _____ topology.
7. _____ is the number of bits transmitted per second over a network.
8. _____ of a channel denotes the frequencies that travel through it.

D. Answer the following questions:

1. Give full form of the following:
MAC, NIC, ARPANET, LAN, MAN, WAN
2. Name four types of networks.
3. Give two examples of MAN.
4. Define the following terms:
(i) Transmission medium
(ii) Topology
(iii) Bandwidth
(iv) Communication channel

5. Differentiate between the following:
   (i) Packet switching and circuit switching
   (ii) Star topology and tree topology
   (iii) Bus topology and ring topology
   (iv) LAN and MAN
   (v) MAN and WAN
   (vi) Hub and switch
   (vii) Modem and router
   (viii) Guided and unguided transmission media
6. Ramya wants to take the printout of a document that she has saved on her mobile phone. She connects her mobile phone to her laptop using Bluetooth and connects her printer with the laptop using a wired medium. Which network PAN/ LAN/ WAN/ MAN) is she using?
7. What is the data transfer rate?
8. How is bit rate different from baud rate? Illustrate with an example.
9. Ramachandran wants to upload a text document at the rate of 12 pages/15 seconds. What should be the minimum required data rate of the channel? (Assume one page contains 600 characters and each character is equal to 8 bits)
10. Name three types of cables used in computer networks.
11. What are the different ways in which the devices may be connected in a LAN?
12. Give an example of wired PAN.
13. Give two characteristics of a hub.
14. What is a modem? Categorise a modem on the basis of its placement.
15. Write a short note on modems.
16. Which device should be used to provide wireless internet access to all smartphone/laptop users in a campus?
17. Write the following in ascending order of the frequencies at which they operate:
    Microwaves, infrared and radio waves.
18. Which transmission media is best suitable for connecting to hilly areas and why?
19. Differentiate between sender and receiver.
20. What is the use of MAC address?
21. Write a short note on *microwaves*.



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

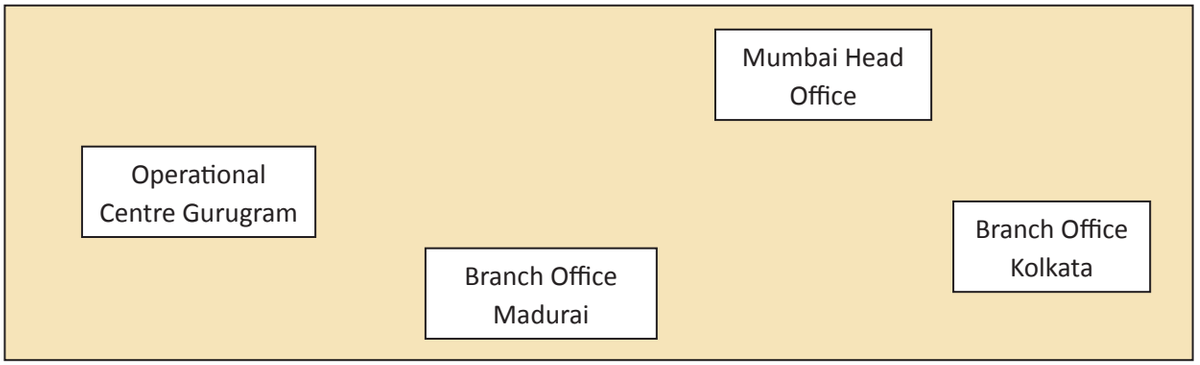
c. A is true but R is false

d. A is false but R is true

1. Assertion (A): Fault diagnosis and correction is easy in case of star topology.
   Reasoning (R): In star topology, multiple nodes are directly connected to the central server.
2. Assertion (A): Extension of network is easy in case of bus topology.
   Reasoning (R): In star topology, all nodes are connected to a central cable, called backbone.
3. Assertion (A): Large business, educational institutions connects their different branches in different locations across the world through WAN.
   Reasoning (R): WAN connects computers and other LANs and MANs spread across the world.

Case-based Questions

1. CareTaker is an NGO with its head office at Mumbai and branches located at Gurugram, Kolkata and Madurai. It is planned to have an Operational Centre at Gurugram that will liaison with the head office and all the branch offices. The NGO has received a grant from the central government for setting up the network. The physical distances between the branch offices and the Operational Centre and the number of computers to be installed in each of these branch offices and the head office are given below. As a network expert, suggest the best possible solutions for the queries raised by the NGO (questions (a) to (d).



Distance between various locations in kilometres:

Location	Approximate Distance (in kms)
Mumbai HO to Gurugram Operational Centre	1450
Mumbai HO to Kolkata Branch Office	1690
Mumbai HO to Madurai Branch Office	2690
Gurugram Operational Centre to Kolkata Branch Office	1450
Gurugram Operational Centre to Madurai Branch Office	1810
Madurai Branch Office to Kolkata Branch Office	1720

Number of computers installed at various locations are as follows:

Mumbai HO	2000
Gurugram Operational Office	1200
Kolkata Branch Office	590
Madurai Branch Office	1100

a. Suggest by drawing the best cable layout for effective network connectivity of all the Branches and the Head Office for communicating data.

b. Suggest the most suitable location to install the main server of this NGO to communicate data with all the offices.

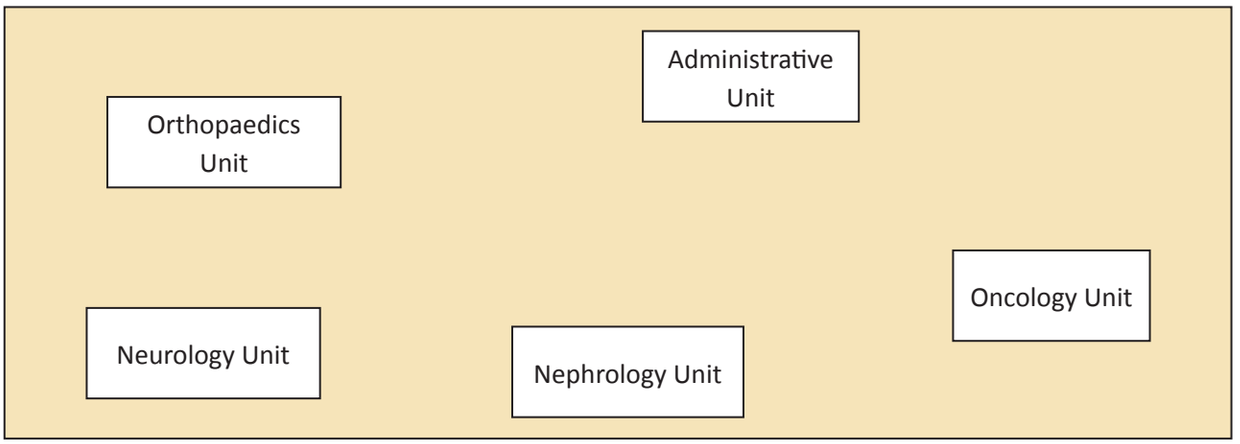
c. Write the name of the type of network out of the following, which will be formed by connecting all the computer systems across the network:

(A) WAN (B) MAN (C) LAN (D) PAN

d. Suggest the most suitable medium for connecting the computers installed across the network out of the following:

(A) Optical Fibre (B) Telephone wires (C) Radio Waves (D) Ethernet cables

2. Welfare Medcare Institute is setting up its centre in Bengaluru with four specialised Units for Oncology, Nephrology, Orthopaedics and Neurology and an administrative office in separate buildings.



The physical distances between various buildings (in metres) are as follows:

Administrative Unit to Oncology Unit	80
Oncology Unit to Nephrology Unit	40
Nephrology Unit to Neurology Unit	50
Neurology Unit to Orthopaedic Unit	75
Orthopaedic Unit to Administrative Unit	60
Administrative Unit to Nephrology Unit	110
Orthopaedic Unit to Oncology Unit	140
Orthopaedic Unit to Nephrology Unit	130

- Numbers of Computers installed at the various locations are as follows:

Oncology Unit	45
Administrative Unit	120
Neurology Unit	49
Orthopaedics Unit	80
Nephrology Unit	25

You as a network expert have to answer the queries as raised by them in (a) to (d).

a. Suggest the most suitable location to install the main server of this institution to get efficient connectivity.

b. Suggest/ Draw the best cable layout for effective network connectivity of the building having a server with all the other buildings.

c. Suggest the devices to be installed in each of these buildings for connecting computers installed within the building out of the following:

- Gateway
- Modem
- Switch

d. Suggest the topology of the network and network cable for efficiently connecting each computer installed in each of the buildings out of the nothing given to choose from.

Previous Years' Questions

1. Fill in the blank:

_____ is used for point-to-point communication or unicast communication such as radar and satellite. [2023]

a. INFRARED WAVES
b. BLUETOOTH
c. MICROWAVES
d. RADIOWAVES

Ans. c. MICROWAVES

2. Differentiate between wired and wireless transmission. [2023]

Ans.

Wired	Wireless Transmission
Data transmission is done using physical media (wires or cables).	Data transmission is done using waves and air.
Antenna is not required.	It requires antennas at both ends.
Information sent in the form of electronic signals.	Information sent in the form of waves.
Wired Transmission is mostly suitable for short distances.	Wireless Transmission is mostly suitable for long distances.
Wired Transmission is quite easy to maintain and configure.	Wireless Transmission is quite complex to maintain and configure.

3. Riya wants to transfer pictures from her mobile phone to her laptop. She uses Bluetooth technology to connect two devices. Which type of network (PAN/LAN/MAN/WAN) will be formed in this case? [2022]

Ans. PAN

NCERT Exercise Solutions

1. What is data communication? What are the main components of data communication?

Ans. Data communication refers to the process of exchanging information between two or more devices through a medium, such as cables, optical fibers, or wireless channels. It involves the transmission and reception of data in the form of signals, enabling communication and information exchange between connected devices.

The main components of data communication are:

a. Sender—the node that sends the data.
b. Receiver—the node that receives the data.
c. Message—information being communicated.
d. Channel—the medium through which data is transmitted.

2. Which communication mode allows communication in both directions simultaneously?

Ans. The communication mode that allows communication in both directions simultaneously is called "Full-Duplex" mode. In Full-Duplex communication, data can be transmitted and received simultaneously, enabling two-way communication between the sender and receiver without the need for switching or alternate modes.

3. Among LAN, MAN, and WAN, which has the highest speed and which one can cover the largest area?

Ans. LAN has the fastest speed because it is used in small areas like offices where cables are short and hence faster speeds.

WAN has the largest area as it spans over the globe.

4. What are three categories of wired media? Explain them.

Ans. Refer to 6.6.1 Twisted pair cable, 6.6.2 Coaxial cable and 6.6.3 Fibre-Optic Cable

5. Compare wired and wireless media.

Ans. In guided transmission, there is a physical link made of wire or cable through which data in terms of signals is propagated between the nodes. These are usually metallic cables, fiber-optic cables, etc. They are also known as wired media whereas, in wireless media, the electromagnetic spectrum of light is used to transfer data. Based on the frequencies, these can be categorized into a number of waves such as radio wave, microwave etc.

6. Which transmission media carries signals in the form of light?

Ans. Refer to 6.6.3 Fibre-Optic Cable

7. List out the advantages and disadvantages of optical fiber cable.

Ans. Refer to 6.6.3 Fibre-Optic Cable

8. What is the range of frequency for radio waves?

Ans. Refer to 6.6.4 Radio Waves

9. 18 Gbps is equal to how many Bits per second?

Ans. 18 Gigabit per second equal to 18×10^9 bits per second

10. HTTP stands for?

Ans. HTTP stands for HyperText Transfer Protocol.

11. Write short note on the following:

a. HTTP

b. Bandwidth

c. Bluetooth

d. DNS

e. Data transfer rate

Ans. a. Refer to 7.3.2 HTTP

b. Refer to 6.4 Data Communication Terminology, Bandwidth

c. Bluetooth: Bluetooth is a short-range wireless technology that can be used to connect mobile-phones, mouse, headphones, keyboards, computers, etc. wirelessly over a short distance.

d. DNS: Domain name system (DNS) maps domain names to numerical IP addresses.

e. Data transfer rate: Data travels in the form of signals over a channel. One signal carries one or more bits over the channel. Data transfer rate is the number of bits transmitted between source and destination in one second. It is also known as bit rate. It is measured in terms of bits per second (bps).

12. What is protocol in data communication? Explain with an example.

Ans. A network protocol is a set of rules that govern data transfer over a network. These rules need to be followed by all the sender, receiver and all devices that are connected.

For example, Ethernet and HTTP. The Hypertext Transfer Protocol facilitates access to hypertext from the World Wide Web by defining how information is formatted and transmitted and how the Web servers and browsers should respond to various commands. Refer Section 7.1 for more details.

13. A composite signal contains frequencies between 500 MHz and 1GHz. What is the bandwidth of a signal?

Ans. Since 1GHz = 1000MHz

Bandwidth = 1000MHz – 500MHz = 500MHz

Bandwidth: Bandwidth of a channel is the range of frequencies available for transmission of data through that channel. Higher the bandwidth, higher the data transfer rate. Normally, bandwidth is the difference of maximum and minimum frequency contained in the composite signals. Bandwidth is measured in Hertz (Hz).

Answers

Multiple Choice Questions

1. (a)	2. (a)	3. (b)	4. (a)	5. (d)	6. (c)	7. (d)	8. (c)	9. (a)
10. (c)	11. (b)	12. (c)	13. (c)	14. (d)	15. (a)	16. (b)	17. (c)	18. (b)

True or False

1. (T)	2. (F)	3. (T)	4. (F)	5. (T)

Fill in the blanks

1. Guided/Wired media and Unguided/Wireless media

2. circuit	3.star	4. network	5. newsgroup	
6. Baud rate	7. Hertz	8. hierarchical	9. WiFi	10. gateway

7

COMPUTER NETWORKS: PROTOCOLS, TECHNOLOGIES, AND WEB SERVICES

Chapter Outline

7.1 TCP/IP

7.2 Mobile Telecommunication Technologies

7.3 Introduction to Web Services

Introduction

While discussing a topic in a room, people follow some rules (also called protocols) to communicate effectively with each other. A protocol that is typically observed in a discussion is that one person will speak at a time while others will listen to the speaker. Of course, it can happen accidentally that two persons start talking simultaneously. If that happens, one of them will become quiet so that the discussion can proceed. Similarly, in a computer network, a protocol is a set of rules that must be observed when two or more devices interact. It specifies a format for data to be communicated and how and when communication should occur.

In this chapter, we will study some popular protocols, TCP/IP: a protocol that combines protocols for transmission control and internetworking, FTP: a protocol for file transfer, PPP: a protocol for point to point communication, Telnet: a protocol for logging on to a remote system, SMTP and POP3: protocols for email transfer, IRC: a protocol for chat, VoIP: a protocol for the transmission of voice over internet, Wireless Fidelity (WiFi) and Worldwide Interoperability for Microwave Access (WiMax): protocols for wireless communication, and HTTP and HTTPS: protocols for transfer of hypertext documents. We will briefly discuss various technologies such as GSM, GPRS, and DLL for mobile communication and mention 1G, 2G, 3G, 4G, and 5G—different generations of mobile communication. Finally, we will briefly mention domain names, web pages, web browsers and how they work, websites and their URLs, web hosting, and web scripting. In this context, we will introduce scripting languages HTML—a language for designing hypertext documents for the web and XML—a language designed to deal with arbitrary data. We will also discuss how a web browser works and briefly mention about HT9.

7.1 TCP/IP

Initially designed by Cerf and Kahn in 1974, TCP/IP enables seamless connectivity between multiple networks, enabling end-to-end data delivery. While TCP is responsible for functions such as segmentation, reassembly, and error detection, IP handles the routing of datagrams. UC Berkeley upgraded the UNIX operating system to integrate TCP/IP in 1981. The availability of network software with the widely used UNIX operating system contributed significantly to the rise of internetworking.

TCP/IP is a layered protocol. To understand the role of layers in communication, let us examine postal communication between two friends, *A* and *B*. Fig 7.1 shows a hierarchy of tasks involving a sender, receiver, and the carrier that supports the transmission of the letter. The following activities are performed one after the other at the sender ' s side:

- **Higher layer:** The sender composes a letter, puts it in an envelope, puts the sender and recipient addresses on it, and drops it in a mailbox.
- **Middle layer:** A postman picks up the mail to the post office.
- **Lower Layer:** The letter is sorted at the post office and passed on to a transporter who gives it to the recipient ' s post office.

At the other end, the transporter delivers the letter to the post office (lowest layer), which gives the letter to the postman (middle layer). Finally, the letter is delivered to the receiver, who opens it and reads it. Subsequently, the recipient may act as a sender and initiate a letter for the other person. As the role of the sender and receiver are now reversed, each layer must be able to perform the task required for sending and receiving the letter.

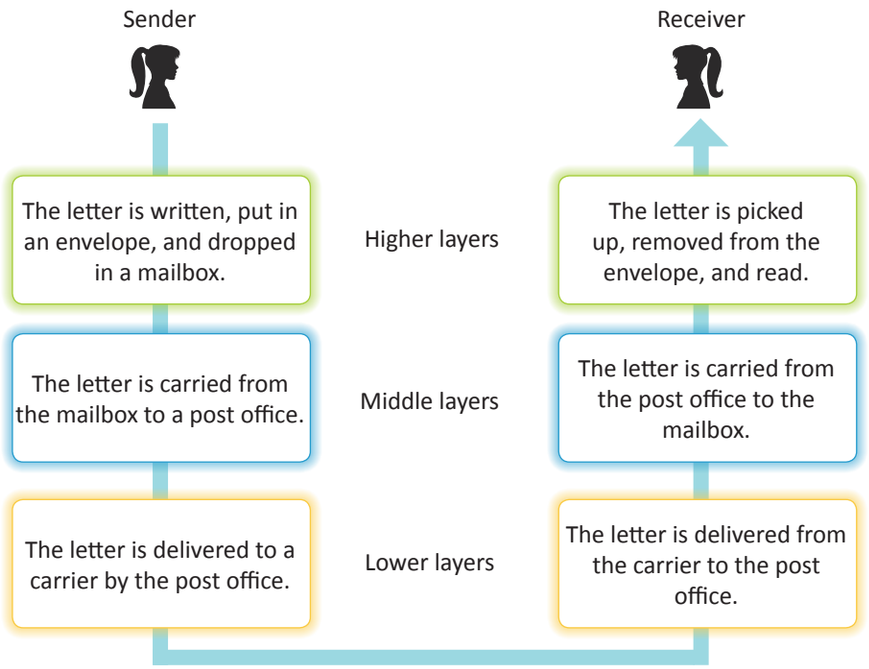


Fig 7.1: Layered communication of letter from sender to receiver.

As the name suggests, TCP/IP comprises TCP (Transmission Control Protocol) and IP (Internet Protocol). We briefly describe these protocols in the following sub sections.

7.1.1 TCP

Transmission Control Protocol (TCP) is a connection-oriented protocol. It requires that the communicating devices must establish a connection between themselves before initiating an exchange of information. The maximum amount of data that can be transmitted in one chunk is called the maximum transfer unit (MTU). On the transmission side, if the data size is greater than the MTU, TCP divides the data/ message received from the application into smaller units. The process of dividing large chunks of data into smaller packets is called fragmentation. Each of these data packets is given a unique sequence number. The original data/ message can be reconstructed from the fragments at the receiving end using the sequence numbers. The process of rebuilding a message from the data packets is called re-assembly. Each packet that is received correctly by the receiver is acknowledged. If a packet is damaged or lost on the way from sender to receiver, it is retransmitted to the destination. TCP manages flow control by ensuring that the speed at which the sender sends messages balances the speed at which the receiver processes messages.

7.1.2 IP

Internet Protocol (IP) is a connectionless protocol since no connection is required to be established between source and destination before communication. It does not generate feedback or acknowledgement. Data packets are called datagrams. The IP protocol deals with route selection and is responsible for delivering datagrams to the destination machine. Under IP protocol, datagrams travel by hopping from one intermediate device (called a router) to another until it reaches the router connected to the destination network. Often there are several paths that a packet may take to travel from source to destination. So, different datagrams of the same message may follow different paths. IP does not keep track of damaged and lost frames, nor provides any provision for sequencing of datagrams, thereby generating no acknowledgement or feedback for the sender.

Transmission Control Protocol / Internet Protocol (TCP/IP)
TCP is responsible for functions such as segmentation, reassembly, and error detection of data packets.
IP handles the routing of data packets.

7.1.3 File Transfer Protocol (FTP)

File Transfer Protocol (FTP) is used for transferring files from one machine to another. FTP is based on the client-server model. The client initiates a file transfer request, and the server fulfils the request by transferring the file to the client. A client request typically originates locally, and the server accomplishes the request from a remote machine. The client and server are also called localhost and remote host, respectively. Moving a file from a remote host to the local host is called a download operation. The opposite operation of moving a file from a localhost to a remote host is called an upload operation. Using FTP, the localhost can create and view directories, download (upload) files to (from) a remote host and create (delete) files at the remote host. An FTP session typically begins with a control connection that authenticates the user via a username and password mechanism. This connection exists for the complete duration of the FTP session. Another connection called a data connection is established to transfer files from the server. Fig 7.2 shows an FTP session. Using anonymous FTP, files available at the remote host for public access may be accessed without authentication.

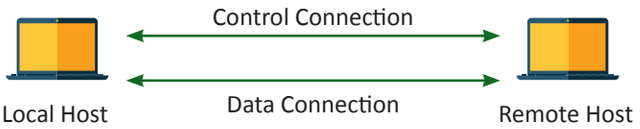


Fig 7.2 Communication System

File Transfer Protocol (FTP) is used for transferring files from one device to another.

7.1.4 PPP

The Point-to-Point Protocol (PPP) establishes a direct connection between two communicating devices. It defines rules for establishing the connection, authentication, data transfer, and connection termination when the connection is no more required. For example, router to router communication between two network segments often takes place using PPP. Home Internet users may use PPP to connect to their service provider (ISP). Also, branches of a business organisation may connect to their head office using PPP.



Fig 7.3

Point to Point Protocol: (P2P) is used for setting up a direct connection between two devices.

7.1.5 Remote Login (TELNET)

TELNET stands for TErminaL NETwork. It operates in client-server mode and enables a user working on one system to access a remote system. It requires a connection to be established between the two systems. For initiating a remote log in, the client user provides the address of the remote system. Although some servers may not require authentication, the server often requires a username and password. Once authenticated, the client can access the remote system's hardware/ software resources. Nowadays, several application tools such as TeamViewer, AnyDesk, Window® Remote Desktop Connection, and Chrome Remote Desktop Connection enable us to log on to a remote computer.

TErminal NETwork (TELNET) enables access to a remote system.

1. Is TCP a layered protocol?
2. Fill in the blanks.
   a. The machine that initiates an FTP request is called a _____.
   b. The machine that fulfills an FTP request is called _____.
   c. _____ protocol establishes a direct connection between two communicating devices.
   d. The full form of the acronym TELNET is _____.

7.1.6 Electronic Mail Protocols

Simple Mail Transfer Protocol (SMTP) and Post Office Protocol 3 (POP3) are popular protocols for exchanging emails over the Internet.

SMTP

An email usually has some text and may optionally include text or multimedia attachments. Also, an email may be sent to multiple recipients. SMTP is a TCP/IP based protocol. It involves a User Agent (UA) and a mail transfer agent (MTA). UA is responsible for packing the message into an envelope on the client side and unpacking the message from the envelope on the receiving side. MTA is responsible for the transmission of messages between the mail servers. As MTA on the receiver side may use a protocol different from the sender side, a gateway can establish compatibility by making the necessary transformation. The SMTP protocol requires the email server at the recipient end to be online. However, if the email server is not on, repeated efforts will be made to deliver the email. If the email server remains inaccessible for several days, the email message will be discarded and an intimation of this effect will be sent to the sender. Fig 7.4 shows the SMTP components. On receiving an email, the SMTP server places it in the recepient's mailbox. Subsequently, the recipient can access the email from his/her mailbox.

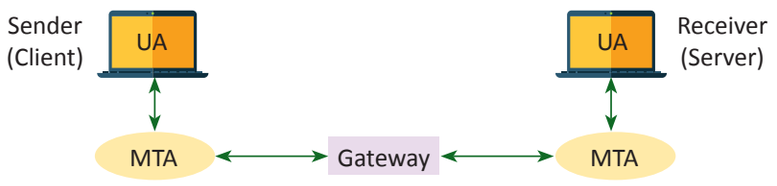


Fig 7.4 SMTP Components

POP3

Post Office Protocol version 3 (POP3) is an email protocol for receiving emails from a remote server and sending them to a local email client. Thus, POP3 allows us to save email messages to our local device (computer, tablet, mobile phone, etc.) and read them. Once a message is delivered to an email client, Internet is not required to view it.

In POP3, the workstation intended to receive mail need not be online all the time. An SMTP server receives emails on behalf of all organisations' users and provides a mail drop facility. Subsequently, the users can retrieve the messages from the server. Fig 7.5 shows email transfer using POP3.

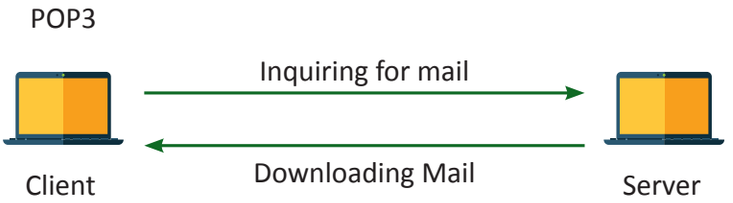


Fig 7.5 POP3

Simple Mail Transfer Protocol (SMTP) and **Post Office Protocol Version 3 (POP3)** are used for exchange of emails over the Internet.

7.1.7 Protocols for Chat and VoIP

Several protocols define rules for communication for chatting with someone, i.e., exchanging text, audio, and video messages. Examples of chat protocols include IRC and VoIP.

Internet Relay Chat (IRC)

IRC is the most commonly used text-based chat protocol. It allows two or more users to form a (logical) chat room for communicating with each other. When a node (source) wishes to send a message to the destination node(s), the source sends the message to the IRC server, which relays it to the destination node(s).

Voice over Internet Protocol (VoIP)

VoIP allows us to make calls using an Internet connection. You may use an application program such as Skype from your computer or smartphone to make a call. You can also use a traditional landline phone for making a VoIP call, provided you have a VoIP adapter. A VoIP call converts an analog voice signal to a digital signal before sending it to the desired destination (IP address) over the internet.

Internet Relay Chat (IRC)/ Voice Over Internet Protocol (VOIP) Protocols are used for exchanging text, audio and video messages over the Internet.

7.1.8 Wireless Protocols

Wireless Fidelity (WiFi) protocol describes rules for communication for transmitting data without any guided medium. A WiFi router serves as an access point (also known as a hotspot) for Internet connectivity. Radio waves connect devices to the Internet via a WiFi hotspot. To ensure the security of WiFi communication, a strong password should be used. Apart from connecting to the Internet, a WiFi access point can also be used for sharing resources such as a printer in a wireless LAN. Fig 7.6 shows a WiFi hotspot that provides Internet connectivity to various devices. The devices within a short-range (say, a 10-metre radius) can connect to the hotspot. The hotspot is unlikely to be accessible to devices located farther than 10 metres as the signal strength drops significantly.

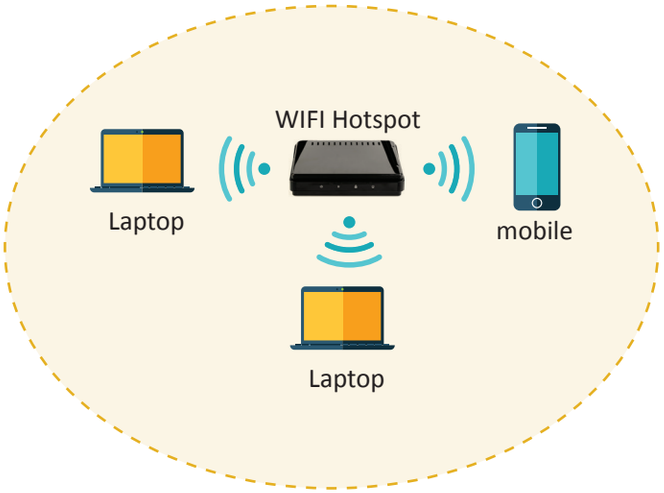


Fig 7.6 WiFi

Worldwide Interoperability for Microwave Access (WiMAX) is another standard for wireless communication. End-users can connect to the Internet via microwave link through WiMAX devices (also called WiMAX stations).

A WiFi router serves as an access point (hotspot) for Internet connectivity.

7.1.9 Wireless Communication

In this section, we will discuss various protocols for wireless communication.

GSM

European Telecommunications Standards Institute (ETSI) developed the Global System for Mobile communications (GSM) for transmission over a mobile network. GSM replaced first-generation analog mobile communication with digital communication in the second generation. GSM used a Subscriber Identity Module (SIM) detachable from the cell phone. It stored the subscriber's identity and phone book. For making a call, a mobile would search for a nearby cell station for connecting to the cellular network. It used time division and frequency division multiplexing to allow several users to communicate simultaneously in real-time. In frequency division multiplexing, a portion of the available frequency spectrum is allocated for communication between two users. In time-division multiplexing, the available bandwidth is allocated periodically to the users for a few milliseconds.

GPRS

General Packet Radio Service (GPRS) was launched during the second generation of mobile telephony. It extends GSM service and facilitates multimedia messaging and wireless data services such as e-mail and Internet access. GPRS enables mobile devices to communicate with each other by exchanging IP packets in a cell. It provided Point-to-multipoint (P2M) service for making group calls.

WLL

Wireless in Local Loop (WLL) provides first mile/ last mile connectivity to the phone and Internet users in a wireless manner. WLL establishes a wireless connection (local loop) between the telephone station and the subscriber's device, such as a mobile handset. It finds widespread use in locations where it is difficult to set up wired connections for landline communication or the number of mobile phone subscribers is very high. It offers the advantages of low set-up cost and large capacity. Several technologies, such as Code-division multiple access (CDMA), Time-division multiple access (TDMA), and Digital Enhanced Cordless Telecommunications (DECT), are used to provide WLL service.

1. Ruchi and Joy are exchanging text messages with each other. Name the protocol that might be used to send and receive messages?
2. Jamila wants to have a voice call with her cousin in Canada. Name a protocol that may be used to make the call using an Internet connection?

7.2 Mobile Telecommunication Technologies

Various generations of mobile technology have evolved during the last three decades. In the following paragraphs, we describe some of these technologies.

7.2.1 First-Generation (1G)

First-generation (1G) mobile technology used analog radio signals for communication. 1G mobile technology was launched by Nippon Telegraph and Telephone in 1979 and remained in use for a decade. Advanced Mobile Phone System (AMPS) was introduced in this generation. The geographical region was divided into cells where each cell used a specific set of frequencies disjointed from adjacent cells. The mobile service provider would set up enough base stations so that a mobile phone user could connect to a nearby base station in its reach. The base stations were connected to the telephone network of the service provider. It supported a data rate of 2Kbps.

Major Issues with 1G mobile technology:

- Owing to electrical interferences, noisy communication was a significant issue with the 1G mobile phones.
- 1G mobile service did not provide encryption for secure communication.

First Generation (1G) mobile technology supported data rate of upto 2Kbps.

7.2.2 Second-Generation (2G)

A set of protocols, collectively called Global System for Mobile Communication (GSM), paved the way for the second generation (2G) of mobile communication. The GSM standard, developed by European Telecommunications Standards Institute (ETSI), laid the foundation for subsequent wireless standards. Launched in 1991, the second-generation mobiles used digital radio signals. Digital encryption made mobile communication secure. The 2G mobile technology supports short message service (SMS) and Multimedia Message Service (MMS). In the mid-1990s, Qualcomm introduced its Code Division Multiple Access (CDMA), outperforming GSM in spectral efficiency. It also achieved a higher data rate of 14.4 to 64 Kilobits per second (kbps).

Major Issues with 2G mobile technology:

- Strong digital signals were required to make mobile phones work.
- 2G mobile service was not suitable to communicate videos.

Second Generation (2G) mobile technology supported data rate of 14.4 to 64 Kbps.

7.2.3 Third-Generation (2G)

NTT DoCoMo introduced the third-generation (3G) mobile technology in 2001. The 3G technology uses circuit switching and packet switching technologies. The 3G of mobiles included technologies such as General Packet Radio Service (GPRS), Universal Mobile Telecommunication System (UMTS) and Enhanced Data Rates for GSM Evolution (EDGE).

UMTS ensures consistency of mobile services as a user moves across the world. As the name suggests Enhanced Data Rates for GSM Evolution (EDGE) technology ensured higher peak data transfer rate of up to 1 Mbps.

Smartphones were introduced in the 3G era. Multimedia messaging service (MMS), higher speed web browsing, video conferencing, and 3D gaming are key features of 3G mobile technology. The 3G provides a digital broadband connection having data transfer rates up to 2 Mbps.

Issues with 3G mobile technology:

- High infrastructure costs
- Required 3G compatible handsets that were more expensive than the phones of the 2G era.



Third Generation (3G) mobile technology supported data rate of upto 2 Mbps.

7.2.4 Fourth-Generation (4G)

Fourth generation (4G) mobile technology was introduced in 2009. It used Multiple Input Multiple Output (MIMO) and Orthogonal Frequency Division Multiplexing (OFDM) technologies. It deployed the Long-Term Evolution (LTE) standard. LTE signifies progression towards true 4G from 3G. As the name suggests, it included a series of updates to the UMTS technology. WiMAX—the other standard popular in the 4G era, is no more prevalent. 4G is based on packet switching. It enabled the data transfer speed of 100 Mbps to 1Gbps. In this generation, technologies such as Voice over Internet Protocol (VoIP), Wireless Fidelity (WiFi), and Worldwide Interoperability for Microwave Access (WiMAX) were introduced. 4G handsets are typically backwards compatible, i.e., they can use 3G and 2G networks if the 4G network is not available somewhere.

Major Issues with 4G mobile technology:

- Significant infrastructure costs as voice call needs circuit-switching.
- 4G compatible high-end mobile devices are required, which is costly.



Fourth Generation (4G) mobile technology supported data transfer speed of 100 Mbps.

7.2.5 Fifth-Generation (5G)

Fifth Generation (5G) mobile technology has made businesses more efficient by giving consumers faster access to information. Like 4G mobile technology, 5G technology is also based on packet switching. 5G supports the data transfer speed of approximately 10 Gbps. Thus, it enables low latency, i.e., significantly reduced data transfer time. As of now, 5G Ultra-Wideband is available in several cities of India.



Fifth Generation (5G) mobile technology supports data rate of approximately 10 Gbps.

7.3 Introduction to Web Services

In this section, we will explore the basic parts of the internet. We will learn about web servers, which help websites work, and the HTTP/ HTTPS protocols that make communication possible. We will also discover domain names, webpages, websites, and URLs, as well as how web browsers like Chrome and Firefox help us view content. We will look at the languages of the web, like HTML and XML. As we go on, we will see how the internet evolved from Web 2.0 to newer versions, and how web hosting and web scripting make websites interactive and functional.

7.3.1 WWW and Web server

Sir Timothy John Berners-Lee (also known as Tim BL) introduced the World Wide Web (WWW). He developed a browser, a server, and the first website. WWW is a collection of documents and resources (for example, music and movies). Resources are identified using unique Uniform Resource Identifiers (URIs), also known as Uniform Resource Locators (URLs). The resources are hosted on computers spread worldwide and are linked by hyperlinks and URIs on the Internet. Web resources are typically accessed using Hypertext Transfer Protocol (HTTP) or Hypertext Transfer Protocol Secure (HTTPS), which adds security functionality to HTTP.

The World Wide Web Consortium (WWWC) develops standards for the smooth growth of the world wide web. Fig 7.7 provides an overview of the working of WWW. The Internet connects you to WWW. When you wish to search for information on the web, the request is relayed to the appropriate server (via the Internet), which uses the Internet to send the reply to your system (host).

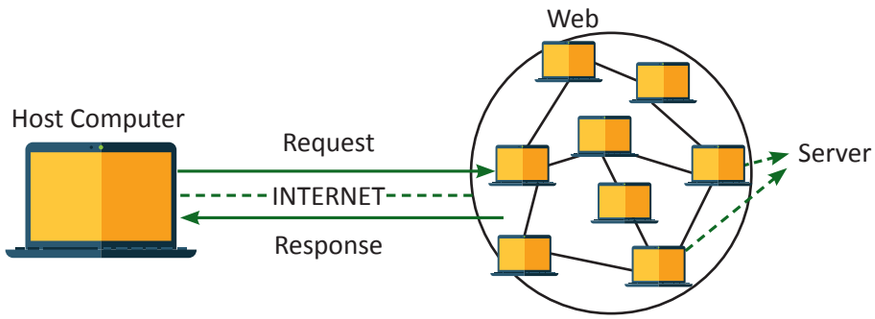


Fig 7.7 www and Internet

WWW is a collection of resources that can be accessed via Internet. Each resource is identified using unique Uniform Resource Identifiers (URIs)

7.3.2 HTTP

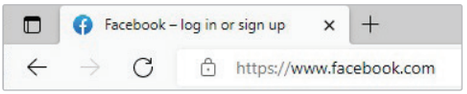
Hypertext Transfer Protocol (HTTP) enables smooth transmission of hypermedia documents, such as HTML The content on WWW is available in the form of hypertext documents containing links from document to document. A hypertext document typically includes several links that connect its content to other documents on the web. A request sent by a client to a web server is called an HTTP request. A response returned by the server is called an HTTP response.

HTTP and HTTPS

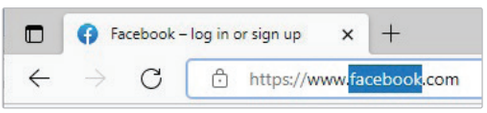
Nowadays, Hypertext Transfer Protocol Secure (HTTPS) protocol is used instead of HTTP, ensuring secure communication over the Internet. It enables data to be transferred in an encrypted format. For example, when we need to submit login credentials to a bank, HTTPS protocol ensures the security of personal information. As another example, you may connect to Facebook by entering the URL:

facebook.com

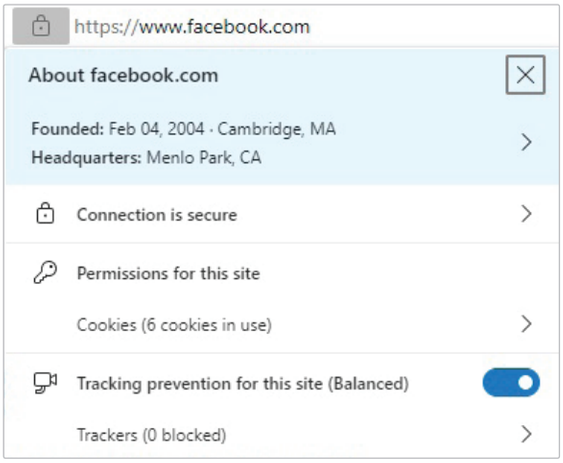
On hitting the enter key, the URL bar reads as follows:



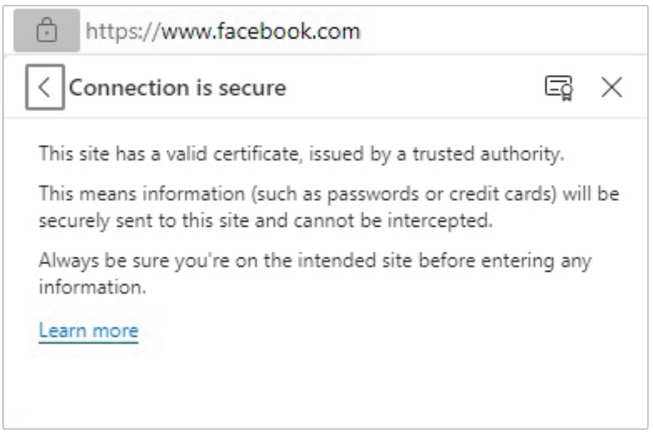
On double-clicking on the URL bar (Facebook.com), the URL appears as follows:



Next, clicking on the lock symbol yields the following information:



Further clicking on the symbol ▶ will yield the following information about facebook.com



Hypertext Transfer Protocol (HTTP) enables smooth transmission of hypermedia documents, such as HTML.
Hypertext Transfer Protocol Secure (HTTPS) protocol ensures secure communication over the Internet.

1. What symbol shows that your connection is secure through HTTPS?
   a. Tick Mark b. Lock
   c. Earth d. Star
2. Which protocol helps a recipient who is currently offline to retrieve the mails?
3. Give full form of the acronym SMTP.

7.3.4 Domain Names

A device on the Internet is identified by its unique IP address, such as 142.250.206.99, comprising four numbers separated by dots. As it is difficult to remember the numerical addresses, so the IP addresses' textual representation (called domain name) is often used. For example, *google.co.in* is much easier to remember than a numerical IP address. Further, an organisation may move its web pages from one machine to a different IP address. In such a scenario, all that needs to be done is map the domain name to the new IP address. Thus, the website users need not be told the new IP address as they will use the same domain name. The Domain name system (DNS) maps domain names to numerical IP addresses.

Domain names are organised hierarchically. Two popular top-level domains are *edu* for educational institutions and *com* for commercial organisations. Examples of country domains include the domains *in* (India) and *us* (the United States).

The **Domain name system (DNS)** maps domain names to numerical IP addresses.

7.3.5 Web Page, Website, and URL

A web page is a document that displays multimedia content such as text, images, audio, and video. To view a web page, you require a web browser such as Google Chrome, Mozilla Firefox, or Microsoft Edge (formerly Internet Explorer). An organisation typically has its content spread over several inter-linked web pages. Such a collection of web pages is called a website. Each website has a *home page* that includes some important information about the organisation and links to other web pages on the website. Using the link on the *home page*, one can navigate to other pages on the website, which may also have links to other website pages. Each web page on the world wide web is identified by a unique address known as Uniform Resource Locator (URL).

A web page is a document that displays multimedia content such as text, images, audio, and video.
A collection of interlinked web pages is called a website.

A URL comprises three parts, protocol name, domain name and name of the file. It identifies the webserver and the location (specific folder) of a particular web page. For example, consider the following URL:

https://www.orangebooks.in/services/book-editing-and-proofreading

Protocol Name Domain Name Path Name of Resource

In the above URL, the domain name is *orangebooks,* and the webpage *book-editing-and-proofreading* resides in the folder *services*.

A URL may be absolute or relative. An absolute URL contains the complete address which includes protocol, domain name and path name of the resource. On the other hand, a relative URL may contain the domain and the path name of the resource or just the name of the resource. It starts with a forward slash and it is assumed that the browser will stay on the current website. For example, *https://www.orangebooks.in/services/book-editing-and-proofreading is an absolute URL while /services/book-editing-and-proofreading* is a relative URL.

7.3.6 Web Browser

A web browser is a software that enables us to view web pages. A web browser may be invoked from a desktop computer, a mobile, a smart TV, a car navigation system, etc. Google Chrome, Mozilla Firefox and Edge (formerly Internet Explorer) are the most commonly used web browsers. A web browser sends a request to the server for the requested web page. At the other end, the web server responds with a web page in HTML format, and the browser interprets the HTML page received from the server and displays its content in the proper format. The functionality of a web browser may be enhanced by adding to it specialised software in the form of plugins and extensions. For example, plugins are required for displaying pdf files, videos, and animations.

How does a Web Browser Work?

Suppose you want to access a webpage identified by the URL: https://docs.python.org/3/tutorial/datastructures.html on your web browser. For this purpose, All you need to do is type https://docs.python.org/3/tutorial/datastructures.html on the Uniform Resource Locator (URL) bar of your web browser and hit the enter key. Fig 7.8 shows the current version of this webpage:

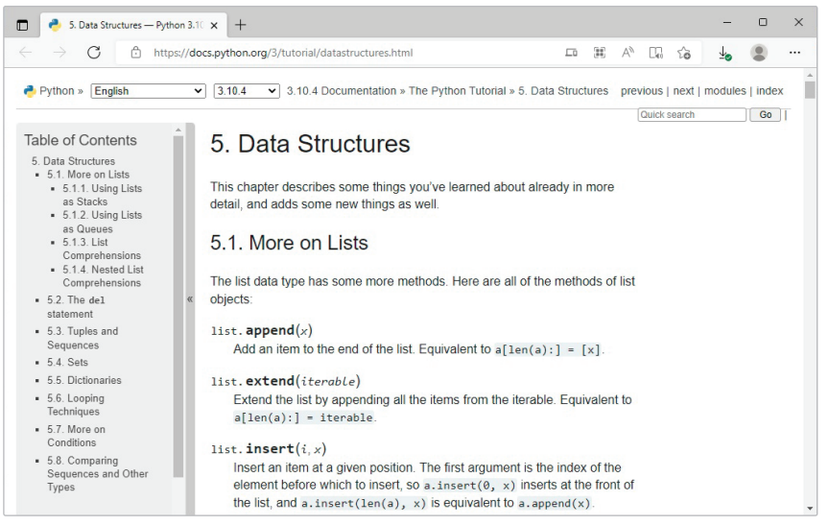


Fig 7.8: Webpage comprising the python data structures using lists.

The uniform resource locator (URL) has the following structure:

protocol://domain:port/path?query_string#fragment_id

- **protocol:** such as HTTP, HTTPS
- **domain:** domain name of the server
- **port:** Typically 443 for secure pages and 80 for the unsecured ones.
- **path:** the location of the resource
- **query string (optional):** any related term searched on the URL. For example, if you search for the term *"using lists as queues",* the URL https://docs.python.org/3/tutorial/datastructures.html changes to

  https://docs.python.org/3/search.html?q=Using+lists+as+queues&check_keywords=yes&area=default

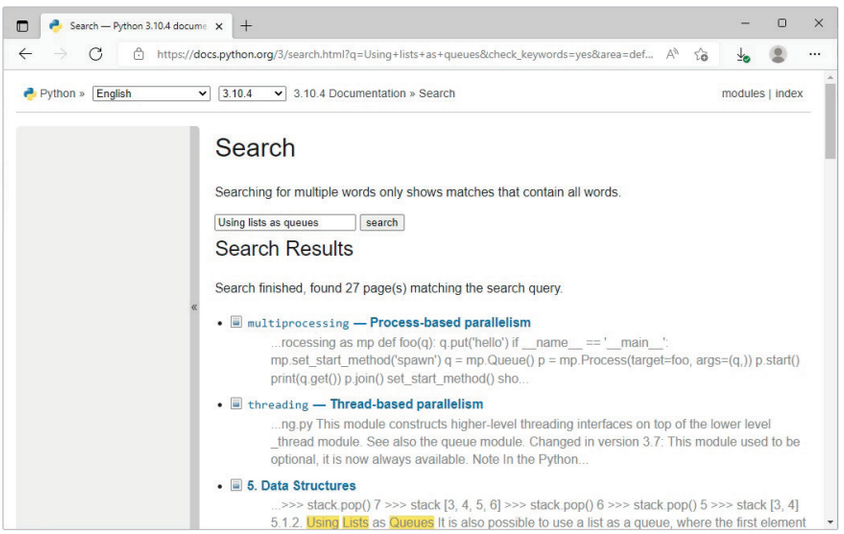


Fig 7.9: Result of query string "Using lists as queues".

Fragment ID (Optional): A fragment identifier is used to identify a portion of the document. For example, if you click on a hyperlink in the left panel of the webpage, the corresponding content is displayed (see Fig 7.10).

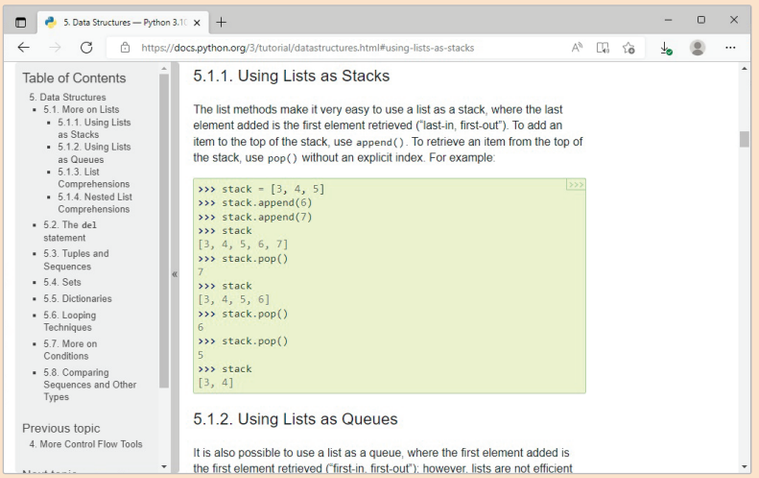
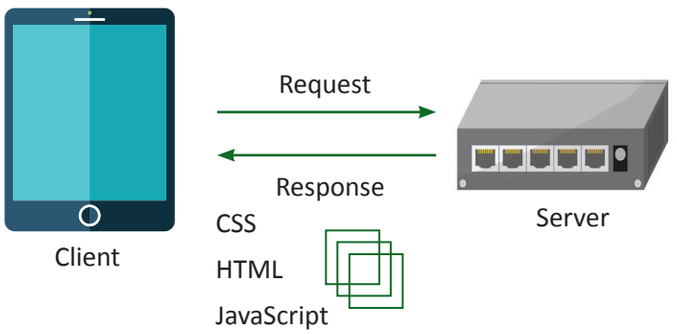


Fig 7.10: Result of clicking the portion on the left panel

When you request a resource from the Internet, your browser acts as a client. An application program called resolver (invoked by the operating system) looks for the web page in the local cache of your machine. If not found, your Internet service provider ' s resolver (ISP) looks for it in its local cache. If the required IP address is not found in the local cache of the ISP, the resolver will search for it in the public DNS (Domain Name System). Once the IP address of the docs.python.org is resolved (199.232.20.223), the HTTP request for the requested web page from the client goes to the server (IP Address: 199.232.20.223).

- The browser function that examines the HTML file is called a parser. The parser interprets the HTML comprising several pieces: the content, structure, and URLs. The structure of an HTML file is described in section 6.3.7.
- The browser **fetches the linked resources** such as media content (images and videos), cascading style sheets (CSS) and determines how the web page appears on the device (font, position, etc.). A javascript executing on the browser determines the web page ' s behaviour (action performed on click of a button) when loaded.
- The browser then **builds and renders** (makes visible) the document on the browser screen.



7.3.7 HTML and XML

HyperText Markup Language (HTML) is used to design web pages. Developed in the year 1990, HTML is used for specifying the content and layout of web pages. It is case insensitive language. It makes use of a set of predefined tags. Earlier, web pages developed using HTML language were static.

Extensible Markup Language (XML) was developed in 1996. It is a case sensitive language and unlike HTML, the tags of XML and not pre-defined. It provides a framework for storing and transmitting arbitrary data.

CT 01

Consider a task to create a simple web page using HTML that contains information about *Turing Award*. So, let the Turing *Award* be the title of the web page.

<HTML> Code </HTML>: the lines of code are enclosed in this tag.

<HEAD> Heading </HEAD>: heading information for the page.

<TITLE> Title of webpage </TITLE>: title inside these tags will appear as the title of the webpage.

<BODY>body of the webpage</BODY>: the desired material to be displayed on the webpage.

<H1> Text </H1>: Heading of some text

<P> Text </P>: anything enclosed within <P> and </P> is the paragraph text.

<A>: used to create a hyperlink, which links pages to each other. The most important attribute of the <A> tag is HREF, which indicates the link ' s destination.

HTML Code:

```
<HTML>
<HEAD>
<TITLE> Turing Award </TITLE>
</HEAD>
<BODY>
<H1> ACM A. M. Turing Award </H1>
<P> The <A HREF="https://www.acm.org/">ACM</A> A. M. Turing Award is an annual prize given by the Association for Computing Machinery (ACM) for contributions "of lasting and major technical importance to the computer field". It is considered the highest distinction in computer science. Sometimes, it is referred to as the "Nobel Prize of Computing".</P>
<P>The award is named after <A HREF="https://en.wikipedia.org/wiki/Alan_Turing">Alan Turing</A>, who was a British mathematician and reader in mathematics at the University of Manchester. Turing is often credited as a key founder of theoretical computer science and artificial intelligence.</P>
</BODY>
</HTML>
```

When opened in the Browser, it appears as shown in Fig 7.11:

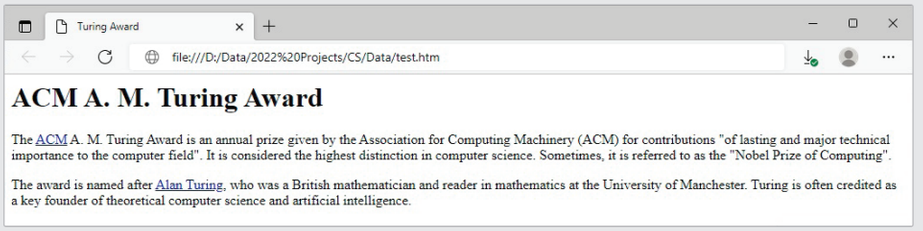


Fig 7.11: Webpage created using HTML

A web browser is a software that enables us to view web pages.

7.3.8 Web 2.0 and Subsequent Versions

The second generation of the World Wide Web is called Web 2.0. A web page whose contents remain unchanged is called a static web page. The first-generation web used static web pages. In contrast, the content of Web 2.0 introduced the concept of dynamic web pages whose content changes with time. For example, a marketing website may vary the price of a commodity in response to the demand for the commodity. Such a change may apply uniformly to all customers or to a handful of customers making repeated purchases of the same item.

As another example of dynamic web pages, the information about the COVID cases in India may be updated dynamically on a website by picking the latest update from the government of India website https://www.mygov.in/covid-19.

Web 2.0 facilitates applications like podcasting, blogging, tagging, social networking, and web content voting. Using artificial intelligence, 3D graphics, and Web 3.0 significantly enhanced web capabilities. The semantic web that focused on understanding the web content (for example, user reviews of products and services) was a vital feature of this era. Internet connectivity included the Internet of Things (IoT), and blockchain technology and cryptocurrency also became popular.

7.3.9 Web Hosting

Web Hosting is a service for hosting websites for the users (clients) on the Internet. There are several options for hosting a website depending on the requirements and the budget. Typically, a webhosting service may belong to one of the following types:

- **Shared Hosting:** A single server (freely accessible or chargeable) may host several domains corresponding to different organisations. Some service providers provide free space for hosting a website comprising a few web pages. Shared hosting is an excellent choice for the beginners. Free hosting usually suffices for the personal websites. Typically, a domain address (for a user website) provided by a free web hosting platforms includes the service provider's name in the URL, for example, As a business organization may not like another organization's name appearing in its website address (URL), usually free hosting does not suit business organizations.
- **Virtual Private Server:** A Virtual Private Server (VPS) offers dedicated resources for each user. VPS is the common choice for small business organizations.
- **Own Private Servers:** An organization may prefer to use its own server for hosting its website.
- **Cloud Hosting:** Nowadays cloud hosting of websites has become very popular. VPS are typically available in the cloud environment.

A web hosting service typically provides facilities for creating and maintaining websites. It provides a web server running on the host. The space is provided on the server for the files that constitute the website, including the databases that the website may require. Domain Name System configuration defines a name for a website and associates it to the hosting server(s). The owner of the website is provided login credentials such as user name and password to maintain the website. Once a website is functional, it can be accessed on the World Wide Web by the users of the Internet.



Web Hosting is a service for hosting websites.

7.3.10 Web Scripting

A web script is a program embedded within a web page. Depending on how it is used, a web script may be a client-side script or a server-side script. The former script is downloaded and then executed by the client's browser to help the client to interact with the web page. However, the server side script is responsible for password authentication and responding to client's requests. The client-side scripts are often written in a language such as HTML, JavaScript, and cascading style sheets (CSS), while the server-side scripts are typically written in languages like Perl, Python, PHP, Active Server Pages (ASP), Java Server Pages (JSP) and Ruby. Let us take the example of Amazon Prime Video.

On the client's side, the HTML, CSS, and JavaScript helps the user to interact with the homepage to select the desired movies to be viewed. The user moves the mouse pointer over the thumbnails to select the desired movie. The client side script captures the user's response and initiates an action itself. The server side script will be executed to send the selected movie to the client's browser for viewing.

A web script is a program embedded within a web page.

1. Name the following:
   a. A software used to access web pages
   b. The unique address that identifies a web page.
   c. A web page whose content remains fixed until explicitly changed by a developer of the web page.
2. Fill in the blanks:
   a. A _____ is a program embedded within a web page.
   b. A web service that uploads a website on the Internet is called _____
3. Match each mobile technology with its key feature.

a. 1G	(i) GSM
b. 2G	(ii) GPRS
c. 3G	(iii) Wi-Max
d. 4G	(iv) Analog signals

Let's Summarise

- In a computer network, a protocol is a set of rules that specifies a format for data to be communicated and how and when communication should occur.
- TCP/IP is a protocol that enables seamless connectivity between multiple networks and it is responsible for functions such as segmentation, reassembly, and error detection, IP handles the routing of datagrams.
- FTP is a protocol used for transferring files from one machine to another.
- PPP is a protocol used for point to point communication and it defines rules for establishing the connection, authentication, data transfer, and connection termination when the connection is no more required.
- Telnet is a protocol operates in client-server mode and enables a user working on one system to access a remote system.
- SMTP is a TCP/IP based protocol that involves a User Agent (UA) for packing/unpacking the message to and from an envelope, and a mail transfer agent (MTA) for the transmission of messages between the mail servers.
- Post Office Protocol version 3 (POP3) is an email protocol for receiving emails from a remote server and sending them to a local email client.
- IRC is the most commonly used text-based chat protocol that allows two or more users to form a (logical) chat room for communicating with each other.
- VoIP is a protocol for the transmission of voice over internet.
- Wireless Fidelity (WiFi) protocol describes rules for communication for transmitting data without any guided medium.
- WWW is a collection of documents and resources that are hosted on computers spread worldwide and are linked by hyperlinks and URIs on the Internet.
- HTTP enables smooth transmission of hypermedia documents, such as HTML.

- A hypertext document typically includes several links that connect its content to other documents on the web.
- HTTPS protocol is used instead of HTTP, ensuring secure communication over the Internet i.e., enables data to be transferred in an encrypted format.
- The textual representation of an IP address is called a domain name.
- The Domain Name System (DNS) maps domain names to numerical IP addresses.
- A Web page is a document that displays multimedia content such as text, images, audio, and video.
- A website is a collection of inter-linked web pages.
- A home page is the main page of a website that includes some important information about the organisation and links to other web pages on the website.
- The unique address of a web page on the WWW is called its Uniform Resource Locator (URL).
- A web browser is a software that enables us to view web pages.
- HTML is used for specifying the content and layout of web pages.
- XML is a case sensitive language for storing and transmitting arbitrary data.
- Web 2.0 introduced the concept of dynamic web pages whose content changes with time.
- Web Hosting is a service for hosting websites for the users (clients) on the Internet.
- A web script is a program embedded within a web page.

Solved Exercises

A. Multiple Choice Questions

1. TCP ensures that the message or data is broken into smaller chunks called _____.
a. Data Packets b. Packets c. Datagram d. None of these

2. TCP guarantees the delivery of packets to the designated _____ on the network.
a. MAC Address b. IP Address c. Network Server d. None of these

3. Which protocol helps us transfer files to and from a remote computer?
a. FTP b. PPP c. HTTP d. POP3

4. FTP is based-on _____ architecture.
a. Client-server b. P2P c. Data centric d. Service-orientated

5. Which of the following is true about PPP protocol?
a. Establishes a direct connection between two devices on the network
b. Used for making an internet connection over a phone line
c. Used for sharing bandwidth
d. Cannot be used to establish a connection between two routers

6. Which symbol is used to separate the domain name and the resource description in a URL?
a. . b. / c. - d. \

7. Which of the following is not a part of a URL?
a. Network Protocol Name b. Name of ISP
c. Domain Name d. Location and name of file/resource

8. In URL, https://www.cbse.gov.in/cbsenew/cbse.html, which is the domain name?
a. www.cbse.gov.in b. https c. /cbse.html d. cbsenew/cbse.html

9. Which of the following protocols is used to create a logical communication link between the user machine (client) and the webserver?
   a. SMTP b. HTTPS c. PPP d. POP3

10. Which of the following allows the recipient to access the mails, even if he/she is offline?
   a. SMTP b. HTTP c. PPP d. POP3

11. Which of the following is a correct sequence of HTML tags for creating a web page?
   a. <HEAD>, <TITLE>,< HTML>,<BODY>
   b. < HTML>,<BODY>,<HEAD>, <TITLE>
   c. < HTML>,<HEAD>,<TITLE>,<BODY>
   d. < HTML>,<TITLE>,<HEAD>,<BODY>

12. Which of the following is a correct description of a website?
   a. Collection of interlinked web pages
   b. Collection of web pages that may or may not be interlinked
   c. A software to view web pages
   d. A repository of web pages

13. Which software is used to view web pages?
   a. TeamViewer b. Browser c. Hyperlink d. MS Word

14. Which language is used to create a web page?
   a. HTML b. HTTPS c. FTP d. HTTP

15. Which of the following is the correct expansion of URL?
   a. Universal Resource Locator
   b. Universal Resource Location
   c. Uniform Resource Location
   d. Uniform Resource Locator

16. Which of the following is not correct?
   a. TELNET is an application development tool
   b. TELNET lets users access an application on a remote computer
   c. It is not necessary to install a TELNET client application on the remote machine
   d. TELNET stands for Terminal Network

17. Which of the following is not the characteristic of XML?
   a. It is a case sensitive language
   b. The XML tags are not pre-defined
   c. It provides a framework for storing and transmitting arbitrary data
   d. It is static language

18. Which of the following is not a distinct feature of Web 2.0?
   a. Blogging b. Artificial Intelligence c. Podcasting d. Social Networking

19. Which of the following is used for server-side scripting?
   a. CSS b. HTML c. PHP d. DHTML

B. State whether the following statements are True or False:

1. TCP/IP is a layered protocol. _____
2. IP does not keep track of damaged and lost frames during transmission of data packets on the network. _____
3. TELNET operates on Client-Server model. _____
4. A WiFi access point cannot be used for sharing a printer in a wireless LAN. _____
5. The web page that counts the number of visitors is an example of a static web page. _____
6. An email can be sent to multiple recipients. _____

7. The control connection of FTP session exists for the entire duration of the FTP session. _____
8. SMTP is not a TCP/IP based protocol. _____
9. POP3 requires the recipient to be online to receive mails all the time. _____
10. First generation (1G) of mobile technology made use of analog radio signals for communication. _____
11. Web Hosting allows the websites to be viewed on the Internet. _____
12. The content of a dynamic web page does not change with time. _____

C. Fill in the blanks.

1. A _____ is a set of rules that must be observed when two or more devices communicate with each other.
2. The process of dividing large chunks of data into smaller packets is called _____.
3. Moving a file from a remote host to the local host is called _____ operation.
4. Using _____ FTP, files available at the remote host for public access may be accessed without authentication.
5. The _____ is used to establish a direct connection between two communicating devices.
6. _____ allows us to make calls using an Internet connection.
7. _____ is a collection of documents and resources on the Internet.
8. _____ Protocol is used for transferring files from one machine to another.
9. Resources on the Internet are identified using unique _____.
10. A textual representation of an IP address is known as _____.

D. Answer the following questions:

1. Expand the following:

   TCP, SMTP, VoIP, PPP, SIM, GPRS, XML

Ans. TCP—Transmission Control Protocol

SMTP—Simple Mail Transfer Protocol

VoIP—Voice Over Internet Protocol

PPP—Point to Point Protocol

SIM—Subscribers Identity Module

GPRS—General Packet Radio Service

XML—Extensible Markup Language

2. Name any one application that uses TELNET.

Ans. AnyDesk

3. Name any two protocols used for exchanging mails over the Internet.

Ans. Simple Mail Transfer Protocol (SMTP) and Post Office Protocol 3 (POP3)

4. What is the disadvantage of SMTP with respect to sending and receiving emails?

Ans. The disadvantage of SMTP is that it requires the receiver to be online.

5. Expand IRC. Also, state its functions.

Ans. IRC stands for Internet Relay Chat. It allows two or more users to form a (logical) chat room for communicating with each other.

6. Name any one application that uses VoIP for communication over the Internet.

Ans. Skype

7. List three distinctive features of IP.

Ans.
- Internet Protocol (IP) is a connectionless protocol
- It does not generate feedback or acknowledgement
- It deals with route selection and is responsible for delivering datagrams to the destination machine.

8. Enumerate the drawbacks of 3G and 4G mobile technologies.

3G

- High infrastructure costs.
- Required 3G compatible handsets that were more expensive than the phones of the 2G era.

4G

- Significant infrastructure costs
- 4G compatible high end mobile devices are required, which are expensive.

9. Why is it advised to use HTTPS instead of HTTP for data communication over the Internet?

Ans. Hypertext Transfer Protocol Secure (HTTPS) protocol ensures secure communication over the Internet. It enables data to be transferred in an encrypted format.

10. Give the names of any two web browsers.

Ans. Google Chrome, Mozilla FireFox

11. Name two languages used for client side scripting.

Ans. HTML, CSS

12. Differentiate between dynamic webpage and static webpage?

Ans.

Static Webpage	Dynamic Webpage
In static web pages, information will remain the same until someone changes it manually.	In dynamic web pages, content of pages may vary over time and may also be different for different visitors.
Static Web Page takes less time for loading than dynamic web page.	Dynamic web page takes more time for loading.

13. Describe the structure of the URL.

Ans. The uniform resource locator (URL) has the following structure:

protocol://domain:port/path?query_string#fragment_id

protocol: such as HTTP, HTTPS

domain: domain name of the server

port: Typically 443 for secure pages and 80 for the unsecured ones.

path: the location of the resource

query string (optional): any related term searched on the URL.

For example, https://sia.mic.gov.in/CBSE

14. Differentiate between a URL and a domain name.

Ans.

URL	Domain Name
1. URL is known as Uniform Resource Locator. It is a string that represents the complete web address of any web page.	1. The domain name is the textual representation of an IP address which is relatively more user friendly.
2. Example: https://www.google.com/	2. Example: google.com

15. Differentiate between HTML and XML.

Ans.

XML	HTML
XML tags are not predefined.	HTML tags are predefined.
XML is case-sensitive language.	HTM is not case-sensitive laguage.
XML tags describe the form of data.	HTML tags describe how data will be displayed.

16. What is Web Hosting?

Ans. Web hosting is posting or uploading a website or a web page on the Internet.

17. What do you mean by shared web hosting and free web hosting?

Ans. In shared hosting, a single server hosts several domains corresponding to different organizations. Some servers use shared hosting to provide free space for hosting a few web pages. This is called free web hosting.



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): IP is a connection less protocol while TCP is a connection oriented protocol.
Reasoning (R): IP protocol deals with route selection while TCP deals with fragmentation of data.

2. Assertion(A): TELNET enables a user working on one system to access a remote system.
Reasoning (R): TELNET does not operate in client-server mode.

3. Assertion(A): VoIP allows us to make calls using Internet connection.
Reasoning (R): A VoIP call converts an analog voice signal to digital signal before sending it to the desired IP address.

Ans. 1. b 2. c 3. a



Case-based Questions

1. Sultana knows that protocol is a set of rules that are followed while transferring data over a network. But she is not clear about various types of protocols and their functions. Help her by naming the following protocols:

a. A layered protocol that is responsible for segmentation, reassembly, and error detection.

b. A protocol used for transferring files from one machine to another.

c. A protocol used for router-to-router communication between two segments on a network.

d. Anydesk application uses this protocol to log in to a remote computer.

e. Protocol used to make voice calls using Internet connection

Ans. a. TCP/IP

b. FTP

c. PPP

d. TELNET

e. VoIP

2. Sushant visits a web page that has the following URL:
http://abc.com/folder1/first.html

His friend, Rupam tells him that this URL does not ensure secure transmission of data. Why did Rupam say so?

Ans. In the given URL, http is used which does not ensure secure transmission of data over the Internet. Instead, https should be used as it enables data to be transferred in encrypted format.

Assessment

A. Multiple Choice Questions

1. Which of the following protocols is used to receive mails over the Internet?
a. SMTP b. POP3 c. PPP d. VoIP

2. Which of the following protocols is used to receive emails over Internet?
a. SMTP b. POP3 c. PPP d. VoIP

3. Which of the following protocols breaks the data into small packets at the sender ' s end and re-assembles them back at the receiver ' s end?
a. TCP b. IP c. SMTP d. PPP

4. Which of the following statements is true about PPP?
a. It is designed for simple links which transport packets between two nodes on a network.
b. It is used to login to a remote device
c. It is used for sharing bandwidth
d. It is used for voice communication

5. What symbol in the browser ' s address bar shows that your connection is secure through HTTPS?
a. Secure Lock b. Tick mark c. Star d. Globe

6. In URL, https://www.mygov.in/task/design-logo-project-lion/do.html, which component identifies the name of the web page?
a. www.mygov.in b. /task c. do.html d. https

7. In URL, https://www.mygov.in/task/design-logo-project-lion/do.html, which component identifies the domain name?
a. www.mygov.in b. /task c. do.html d. https

8. Which of the following mobile generations is associated with MIMO(Multiple Input Multiple Output) technology?
a. 1G b. 2G c. 3G d. 4G

9. What is the approximate data rate supported by 5G mobile technology?
a. 10 Gbps b. 10 Kbps c. 100 Mbps d. 10 Mbps

10. Which of the following is the correct expansion of EDGE?
a. Enhanced Data rates for Global Evolution b. Extended Data rates for Global Evolution
c. Extended Data rates for GSM Evolution d. Enhance Data rates for GSM Evolution

11. Which of the following is used to identify a web page on the Internet?
a. IP Address b. URL c. MAC Address d. NIC

12. Which of the following is true about shared web hosting?
a. It is always chargeable
b. A single server hosts domain for one organization
c. It is always available for free.
d. A user ' s website will always have the domain service provider's name.

13. Which of the following is a client side scripting language?
a. Perl b. JSP c. CSS d. ASP

14. Which of the following is text-based chat protocol?
a. IRC b. VOIP c. TELNET d. SMTP

15. Which of the following statements is false?
a. POP3 allows to save email messages on a local device.
b. SMTP is a TCP/IP based protocol.
c. TeamViewer application helps us to login to a remote computer.
d. A connection between two communicating devices exists for during login period.

B. State whether the following statements are True or False:

1. While transmissting data, if the data packet is lost on the way from sender to receiver, it is retransmitted to the destination. _____
2. 5G technology is based on circuit switching. _____
3. A URL may be absolute or relative. _____
4. Plug-ins help to enhance the functionality of a web browser. _____
5. An organization cannot have its own server for hosting its website. _____
6. TELNET operates in client server mode. _____
7. Domain names are organized hierarchically. _____
8. A website can be accessed on the Internet only after it is published. _____
9. XML is case insensitive. _____
10. A home has links to navigate to other pages of a website. _____

C. Fill in the blanks.

1. _____ is a communication methodology designed to deliver both voice and multimedia communications over Internet Protocol.
2. _____ protocol allows access to the command line interface on a remote computer.
3. Smartphones were introduced in _____ generation of mobile technology.
4. The _____ develops standards for the smooth growth of the World Wide Web.
5. A device on the Internet is identified by its _____.
6. A web page whose contents remain unchanged is known as a _____ web page.
7. A _____ is a program embedded in a web page.
8. _____ protocol describes rules for transmitting data without any guided medium.
9. Domain name is the textual representation of _____.
10. IRC and VoIP are examples of _____.

D. Answer the following questions:

1. Expand the following acronyms:
   DNS, POP3, TELNET, CDMA, MMS,
2. "TCP is a layered protocol." comment.
3. What is the function of transmission control protocol?
4. Describe the working of an FTP session.
5. A router connects to the ISP for internet connectivity. Which protocol is used to establish connection between the router and the ISP?
6. Write a short note on TELNET.
7. Distinguish between SMTP and POP3 electronic mail protocols.
8. SMTP is used for exchanging emails over the Internet. Name the two agents involved in the process.
9. Give two examples of Chat protocols.
10. What is the function of VoIP?
11. Which transmission medium is used to connect a device to the Internet via a WiFi hotspot?
12. Give one difference between WiFi and WiMax.
13. Give two distinct features of 5G mobile technology.
14. Define the following:
    (i) Web Server (ii) Web Browser (iii) URL (iv) Web Hosting (v) Web Scripting
15. What is the role of W3C?
16. What is the function of DNS?
17. How does a web browser work?

18. Differentiate between the functionality of HTML and XML.
19. Name three internet applications facilitated by Web 2.0
20. Differentiate between client-side scripting and server-side scripting.

Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): A WiFi router works as an access point for Internet connectivity.
   Reasoning (R): A WiFi access point can be used for sharing a printer on a LAN.
2. Assertion(A): http ensures secure communication of web pages on the Internet.
   Reasoning (R): http protocol is used for transmission of hypertext documents.
3. Assertion(A): A web browser enables to view web pages.
   Reasoning (R): A web server responds to the request of a web browser in HTML format.

Case-based Questions

Santushti has just learnt about URLs. Her teacher has given her the following URL and asked her to identify the protocol, domain name, path and name of the HTML document:

https://www.incredibleindia.org/content/incredible-india-v2/en/experiences/adventure.html

Previous Years' Questions

1. Differentiate between URL and domain name with the help of an appropriate example. [2023]

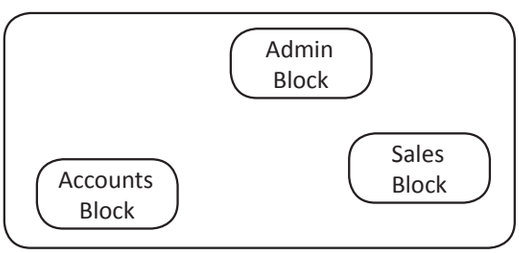
Ans.

URL	Domain Name
A URL is a complete web address for a website.	A domain name is like a brand name for a web site.
The URL contains the complete domain name.	The domain name is part of the URL.
A URL is made up of the protocol, domain name, port, and path of the file.	The domain name is made up of the name and top-level domain of a particular website.
Example: https://www.thetouchcode.com/	Example: thetouchcode.com

2. Galaxy Provider Ltd. is planning to connect its office in Texas, USA with its branch Mumbai. The Mumbai branch has 3 Offices in three blocks located at distance from each other for different operations—ADMIN, SALES and ACCOUNTS.

   As a network consultant, you have to Suggest the best network-related solutions for the issues/problems raised in (i) to (iv), keeping in mind the distances between various locations and other given parameters.

   Layout of the Offices in the Mumbai branch: [2022]



Shortest distances between various locations:

Admin Block to Sales Block	**300** m
Sales Block to Accounts Block	**175** m
Admin Block to Accounts Block	**350** m
Mumbai Branch to Texas head Office	**14000** km

Number of Computers installed at various locations are as follows:

ADMIN Block	**225**
ACCOUNTS Block	**75**
SALES Block	**30**
TEXAS Block	**90**

(i) It is observed that there is a huge data loss during the process of data transfer from one block to another. Suggest the most appropriate networking device out of the following, which needs to be placed along the path of the wire connecting one block office with another to refresh the signal and forward it ahead.

a. MODEM b. ETHERNET CARD c. REPEATER d. HUB

Ans. c. REPEATER

(ii) Which hardware networking device out of the following, will you suggest connecting all the computers within each block?

a. SWITCH b. MODEM c. REPEATER d. ROUTER

Ans. a. SWITCH

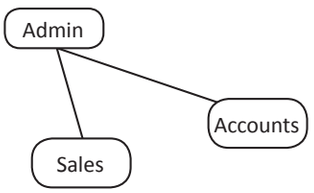
(iii) Which service/protocol out of the following will be most helpful to conducting live interactions of employees from Mumbai Branch and their counterparts in Texas ?

a. FTP b. PPP c. SMTP d. VoIP

Ans. d. VoIP

(iv) Draw the cable layout (block to block) to efficiently connect the three offices of the Mumbai branch.

Ans.



NCERT Exercise Solutions

1. Expand the following:

a. ARPANET

b. MAC

c. ISP

d. URI

Ans. a. Advanced Research Project Agency Network

b. Media Access Control

c. Internet Service Provider

d. Uniform Resource Identifier

2. What do you understand by the term network?

Ans. A connectivity between two or more computers or computing devices is known as a computer network. A network is any collection of independent computers that communicate with one another over a shared network medium.

3. Mention any two main advantages of using a network of computing devices.

Ans. a. Resource Sharing

b. Communication and Collaboration

4. Differentiate between LAN and WAN.

Ans. Refer to 6.8.2 LAN and 6.8.4 WAN

5. Write down the names of a few commonly used networking devices.

Ans. Modem, Repeater, Routers, Gateway, Hubs, Switches, etc.

6. Two universities in different states want to transfer information. Which type of network do they need to use for this?

Ans. Refer to 6.8.4 WAN

7. Define the term topology. What are the popular network topologies?

Ans. The arrangement of nodes in a network is called topology. It describes the placement of various network devices and the physical connections between them. Popular network topologies include ring topology, bus topology, star topology, and tree topology.

8. How is tree topology different from bus topology?

Ans. Tree topology differs from bus topology in that a tree topology combines multiple bus or star topologies into a hierarchical structure. In a tree topology, a central root node connects to other nodes, forming branches. Each branch can function as an independent bus network or a star network. This hierarchical arrangement allows for better organization, easier expansion, and improved fault isolation compared to the linear structure of a bus topology. Refer to sections 6.7.1: Bus Topology and 6.7.4: Tree Topology for more details.

9. Identify the type of topology from the following:

a. Each node is connected with the help of a single cable.

b. Each node is connected to central switching through independent cables.

Ans. a. Bus topology

b. Star topology

10. What do you mean by a modem? Why is it used?

Ans. Refer to 6.9.7: Modem

11. Explain the following devices:

a. Switch

b. Repeater

c. Router

d. Gateway

e. NIC

Ans. a. Refer to 6.9.6 Switch

b. Refer to 6.9.3 Repeater

c. Refer to 6.9.8 Router

d. Refer to 6.9.9 Gateway

e. Refer to 6.9.2 Network Interface Card (NIC)

12. Draw a network layout of star topology and bus topology connecting five computers.

Ans. Refer to 6.7.3 Star Topology and 6.7.1 Bus Topology

13. What is the significance of the MAC address?

Ans. The Network Interface Controller (NIC) manufacturer typically assigns the MAC (Media Access Control) address, which is kept in the hardware. This address is significant because it uniquely identifies network interfaces.

This address ensures the precise delivery of data to the intended device within a local network. It is crucial for network communication, enabling routers and switches to direct data to the correct destination, and plays a key role in device authentication on a network.

14. How is an IP address different from a MAC address? Discuss briefly.

Ans. An IP address is a string of numbers that identifies a device on a network. It enables the devices on the network to communicate with each other, whereas a media access control address (MAC address) is a unique identifier assigned to a network interface controller (NIC) for use as a network address in communications within a network segment.

15. What is DNS? What is a DNS server?

Ans. DNS is a system that translates human-readable domain names into numerical IP addresses, facilitating communication between devices on a network. A DNS server is a specialized server that manages the process of domain name resolution. It holds a database mapping domain names to IP addresses and responds to DNS queries by providing the requested information.

16. Sahil, a class X student, has just started understanding the basics of Internet and web technologies. He is a bit confused between the terms "World Wide Web" and "Internet". Help him understand both terms with the help of suitable examples of each.

Ans. The Internet is a global network that connects millions of devices worldwide. It is the infrastructure that enables communication between computers. For example, when Sahil sends an email, the information travels through the Internet to reach its destination. The World Wide Web (WWW) is a system of interconnected documents and resources accessed via the Internet using web browsers. It's like a subset of the Internet focused on content and services. When Sahil visits a website, like searching for educational resources, he is using the World Wide Web over the Internet.

Answers

Multiple Choice Questions

1. (c)	2. (b)	3. (a)	4. (a)	5. (a)	6. (b)	7. (b)	8. (a)	9. (c)
10. (d)	11. (c)	12. (a)	13. (b)	14. (a)	15. (d)	16. (a)	17. (d)	18. (b)
19. (c)								

True or False

1. (T)	2. (T)	3. (T)	4. (F)	5. (F)	6. (T)	7. (T)	8. (F)	9. (F)
10. (T)	11. (T)	12. (F)						

Fill in the blanks

1. protocol	2. fragmentation	3.download	4. anonymous	5. PPP
6. VoIP	7. WWW	8. File Transfer	9. Uniform Resource Locator (URL)	
10. domain name				

8 NETWORK SECURITY

Chapter Outline

8.1 Understanding Security Threats
8.2 Use of Cookies
8.3 Firewall
8.4 HTTPS
8.5 India IT Act
8.6 IPR Issues

As the data typically moves over unsecure public data channels, it is prone to various risks. For example, an intruder might access the data being communicated over a channel, thus violating the privacy and/ or security of data. Even worse, an intruder may alter the data and an innocent recipient may believe it to be genuine. So, it is important to ensure security and integrity of data over the networks, including the Internet. For example, if a user of a bank wants to make an online transaction, the banking system must guarantee the security of the passworded information over the Internet. In the previous two chapters, we discussed the basics of computer networks and the services, protocols, and devices. In this chapter, we will discuss a variety of security threats, the ways to prevent these security threats, and conclude the chapter with a brief discussion of the Indian IT Act that addresses the legal issues related to intellectual property rights (IPRs) and cybercrimes.

Introduction

Security of an IT system may be compromised in several ways. Essentially, there are three important goals of security, viz. confidentiality, integrity, and availability. We have already mentioned the need to maintain confidentiality, for example, while logging on to a bank portal, username and password must not fall into the hands of an unscrupulous person or agency. Similarly, someone may alter an e-mail sent by your bank, thus violating the integrity of the message being communicated. Finally, someone may access the website of a company excessively, thus, making it crash or become extremely slow for other users. Sometimes, when an IT resource ceases to be available to the intended users, it is called denial of service (DoS) attack. Malicious software such as viruses, worms, trojan horses, etc., harm the computer system or computer network in some or the other manner.

Key Security concerns: confidentiality, integrity, and availability

Information security aims to combat a wide range of threats. As most of the security violations take place when data moves over a network, it is extremely important to securitize the network (see Fig 8.1). Network security protocols

protect data transferred via information networks from being tampered with or damaged. The network administrators are responsible for enforcing robust security policies to keep an organization's IT infrastructure safe. They install, config, and maintain the hardware and software to detect security threats.



Fig 8.1 Network Security

8.1 Understanding Security Threats

An attempt to access an IT resource illegitimately is called an attack. The possibility of the occurrence of an attack is called a threat. The attacker may exploit one or more vulnerabilities (weaknesses) that can become a threat to the system. In this section, we will discuss some of the threats to a computer system.

8.1.1 Virus

A virus is a software that may harm a computer system by overwriting or corrupting the files. Computer viruses are similar to viruses in our body that replicate themselves and affect body cells. A typical virus gets embedded in an executable program file. When an infected code is executed, the virus may get loaded into the computer and affect other files too, thus generating multiple copies of the virus. When an infected computer comes in contact with another computer via network or while using a file/ device like a flash drive from an infected computer, the infection can spread from one computer to another, Fig 8.2:



Fig 8.2 Virus

A virus is a software that may harm a computer system by overwriting or corrupting the files. Like biological viruses, a computer virus can replicate itself.

A computer virus makes several copies of itself by inserting its code onto the system programs, files, or boot sector of hard drives, thereby corrupting them. As a result, the system may slow down or even stop functioning. The viruses are mainly categorized as boot sector virus, file infector virus, and macro virus.

- Boot sector viruses, such as Michelangelo virus, either affects the boot sector of the hard disk or the master boot record (MBR) of the hard disk. The virus will be loaded the next time a user tries to boot their computer.

- File Infectors viruses such as Romeo and Juliet attach themselves to executable files either by overwriting a part of their code or appending their code to the files.
- Macro viruses such as Melissa Virus embed themselves into the documents, where macro refers to a specific set of instructions within an application. A macro virus is an executable file that is often received as an e-mail attachment. When an infected attachment is opened, the virus may start functioning by affecting the system programs (deleting, creating or overwriting other files). Melissa Virus spread through a Microsoft word document sent as an attachment. When the word file was opened, the virus got executed automatically. The attachment got forwarded to the first fifty recipients in the address book in the Microsoft outlook.

8.1.2 Worms

Unlike viruses which come into action when humans unknowingly perform some action, a worm may flood a hard disk by automatically creating several copies of itself. No host application is required for worms to replicate themselves. Also, when a worm is received through an e-mail attachment, it may get automatically forwarded to several e-mail ids leading to network congestion (see Fig 8.3). Thus, a worm may crash not only a computer system, but the entire network. Example: Code Red Worm, which makes more than 2,50,000 copies of itself in approximately 9 hours.

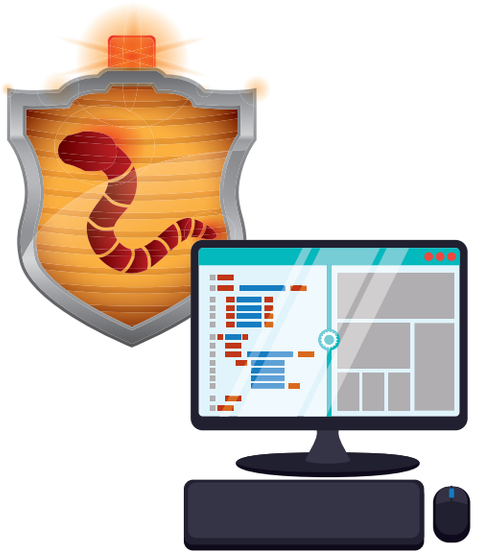


Fig 8.3 Worm

A worm typically enters the system via the network, most commonly via the Internet.

8.1.3 Trojan Horse

Trojan Horse is a code that appears to be desirable and useful but ends up harming the system. A Trojan horse typically attaches itself with a safe application. For example, it may be attached to a game downloaded over the Internet. When executed, such an application creates a backdoor in the system (as shown in Fig 8.4) through which hackers can access the system. Hackers can monitor the activities performed by the users on the system. They can also control your system and harm the system's resources.

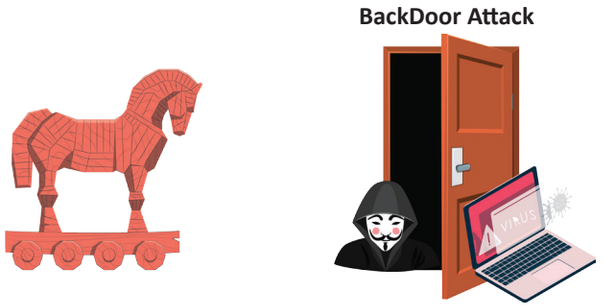


Fig 8.4 Trojan Horse

Trojan Horse: An apparently useful code that harms the computer.

8.1.4 Spams

Spams are unwanted electronic mails generally sent in bulk over the Internet to recipients. Such undesirable mails are generally commercial e-mails sent for advertisement purposes. However, they may contain a link to phishing sites that attempt to steal user information or may have a link to sites that contain malware or infected files. E-mail service providers such as Google provide Spam filters (Fig 8.5) that help in separating spam mails from genuine e-mails.

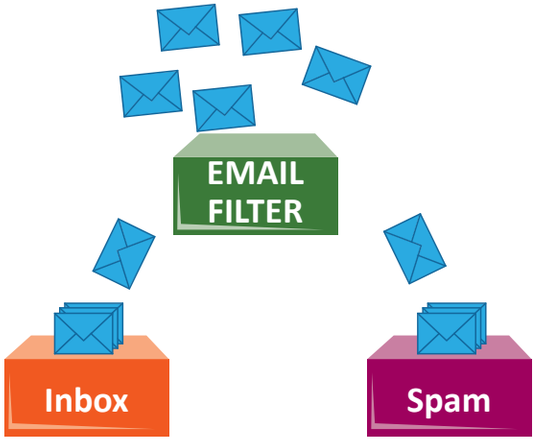


Fig 8.5 Spam Filtering

Spam: An unwanted e-mail, often sent by an advertiser.

Give one example of each of the following:

a. Boot sector virus

b. Worm

c. File Virus

8.1.5 Hacking

Hacking refers to having unauthorized access to someone's computer or computer network for stealing resources such as passwords or confidential files or causing harm to the network or system. The hacker identifies the vulnerabilities of the system to achieve this. Hackers may indulge in doing so for their interest, as a means of fun, or protest (Fig 8.6).

Hackers are also categorized as a good hacker and bad hacker. Bad hacker hacks the system with bad intentions, whereas a good hacker tries to hack the system to identify its weaknesses so that they can be isolated. The bad (unethical) hackers are called crackers, whereas the term hacker is generally used to refer to the good (ethical) hackers.



Fig 8.6

8.2 Use of Cookies

When a user browses the Internet, a web server may send some information in the form of encrypted files, called browser cookies and web cookies. Cookies sent by a website aim at enhancing a user's experience on that website, based on the activity log of the user and are stored in the browser's directory. For example, when you visit a commercial website, you may see the advertisements of the products related to your searches on the website earlier.

Cookies: When a user visits a website via his/her browser, the website attempts to enhance the user's experience on the website. For this purpose, it sends to the user's browser, some pieces of text called cookies.

8.3 Firewall

A firewall is a hardware device or a software mechanism or a combination of both and is placed between two or more different networks (see Fig 8.7). A firewall inspects the network traffic and controls which incoming and outgoing data should be allowed to pass through it, thus ensuring the network's security. For example, when we receive data over the Internet, it passes through our system's firewall before getting transmitted to us. Similarly, a firewall in an organization can check whether a user is permitted to send the data outside the organization. Once the security checks are cleared, data (packets) are routed from (to) the Internet, otherwise the flow of data is blocked.



Fig 8.7 Firewall

Firewall: A firewall provides security to a computer network. It is a hardware device or a software mechanism or a combination of both and is placed between two or more different networks

8.4 HTTPS

The content stored on WWW is available in the form of hypertext containing links from one document to another. HTTP stands for Hypertext Transfer Protocol. It is a set of rules that governs how data should be communicated between client and server. It controls how data is accessed from WWW and how a particular server on WWW sends back the reply. A request sent by a client or host system to the web server is called HTTP request, and a response returned by the server is called HTTP response. TCP is used to establish a connection between client and server before initiating the transfer of data, ensuring reliable and ordered delivery of data.

Nowadays, HTTPS (Hypertext Transfer Protocol Secure) protocol is preferred over HTTP protocol because HTTPS ensures secure communication of data. HTTPS enables data to be sent in an encrypted format. For example, when we submit login credentials to our bank, we usually use the HTTPS protocol. HTTPS employs the Transport Layer Security (TLS) encryption technology, often known as the Secure Sockets Layer (SSL).

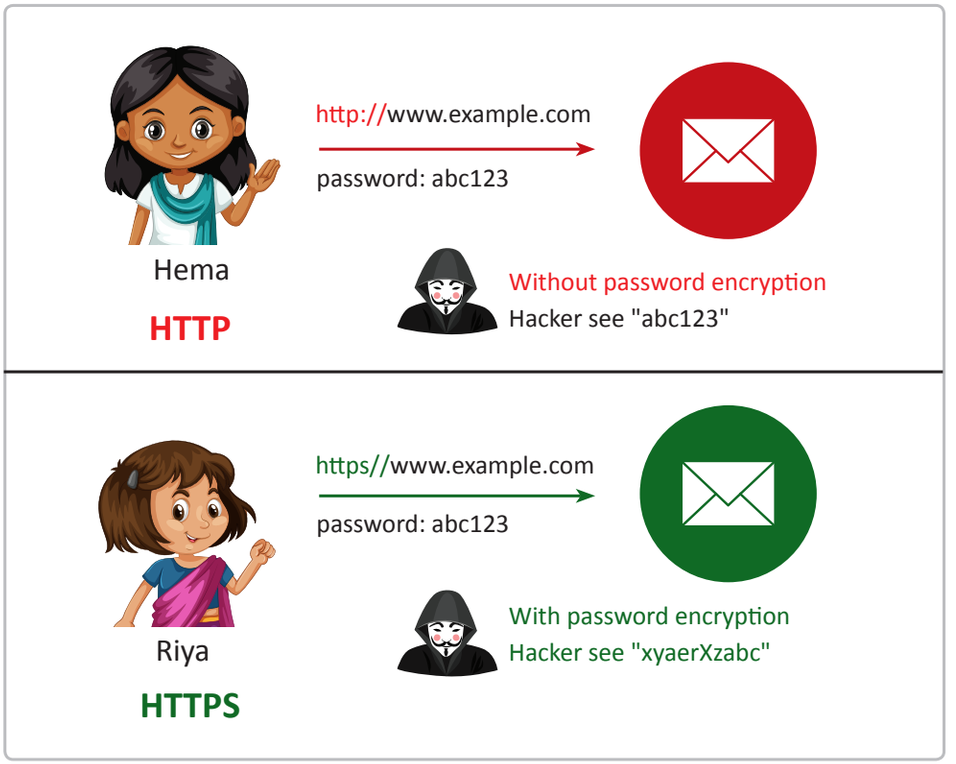


Fig 8.8 HTTP vs HTTPS

HTTP: Hypertext Transfer Protocol.
HTTPS: Hypertext Transfer Protocol Secure.

Give one word answer for the following:

1. A piece of text that helps to enhance a user's browsing experience.
2. A security mechanism that controls the incoming and outgoing data to/from a network.
3. A request sent by a client or host system to the web server.
4. A protocol that is used to establish a connection between client and server before initiating the transfer of data.

8.5 India IT Act

The advent of the Internet has brought significant changes in our lives. People from all walks of life are increasingly using computers to create, communicate, and store information in electronic form rather than in the form of traditional paper format. Electronically stored information offers numerous advantages, like cost savings and ease of storage and retrieval. Despite many benefits of the online availability of information, it has been abused by many people for personal gain or to damage someone's documents. Unfortunately, the faster access to information from anywhere in the world has been accompanied by an increase in cybercrimes, necessitating the creation of laws to safeguard the interests of the people. Considering the need to protect the interests of the users of Information Technology, the United Nations Commission on International Trade Law (UNCITRAL) has proposed a model law to deal with the menace of cybercrimes. The governments have taken precautions and enacted legislation governing the use of Internet resources. Based on the UNCITRAL model law, the Indian Parliament passed the Information Technology Act 2000 to deal with IT related security threats. The Information Technology act 2000 came into action on 17 October 2000. The major goal of this act is to streamline online digital business operations as well as to make provisions for prevention of cybercrime. Thus, it describes the norms for smooth availability of IT services, and protects the interests of various entities, such as businesses and their clients, involved in the use of IT services. Thus, IT Act 2000 provides a legal infrastructure for e-commerce in India. In this direction, the act also included a provision for digital signatures for authenticating (confirming this source and content) the digital information. It also identifies various undesirable activities as cybercrimes and prescribes punishment for them.



IT Act 2000 aims at defining a legal framework for using WWW. It defines what constitutes cybercrimes and prescribes punishment for them.

8.5.1 Cyber Laws

Cyber laws are the statutory provisions against various forms of cybercrimes. Cyber laws define various crimes and prescribe penalties/ punishments imposed against persons committing such crimes. Spread over defaming someone using the Internet.

There are 13 chapters and 90 sections in the IT Act that defines offences and penalties for cybercrimes. An amendment in IT Act 2000 named Information Technology Amendment Act, 2008 was also introduced for cyber security. It prescribed an agency, namely, Indian Computer Emergency Response Team (CERT-In) for administration of IT act. Cyber police are responsible for investigating the crimes and taking the necessary measures against them under the IT Act.

8.5.2 Cybercrimes

As mentioned above, cybercrimes are crimes related to the misuse of computers or the Internet, such as theft, fraud, and forgery. Thus, a cybercrime may be defined as an unlawful act wherein a computer or an IT device such as a mobile phone is either a tool or a target or both. The following activities (amongst others) are considered cybercrimes:

- Sending spam e-mails to uninterested recipients.
- Hacking someone's account or system.
- Stealing someone's personal information through phishing.
- Hosting a site carrying lots of malware or being a source for spreading them.
- Harassing someone through e-mails, messages or social networking sites.
- Posting offensive content on any site or sending it to anyone.

Section	Offence	Penalty
65	Tampering with the computer source documents	Imprisonment up to three years, or/and with a fine up to ₹2,00,000
66	Hacking with computer system	Imprisonment up to three years, or/and with a fine up to ₹5,00,000
67	Publishing of information which is obscene in electronic form	Imprisonment up to five years, or/and with a fine up to ₹10,00,000
68	Power of Controller to give directions	Imprisonment up to five years, or/and with a fine up to ₹100,000
69	Directions of Controller to a subscriber to extend facilities to decrypt information	Imprisonment up to seven years, and can also be subjected to a fine up to ₹10,00,000
70	Securing access or attempting to secure access to a protected system	Imprisonment up to ten years, or/and with fine
71	Misrepresentation	Imprisonment up to 2 years, or/and with a fine up to ₹1,00,000
72	Breach of confidentiality and privacy	Imprisonment up to 2 years, or/and with a fine up to ₹1,00,000
73	Publishing Digital Signature Certificate false in certain particulars	Imprisonment up to 2 years, or/and with a fine up to ₹1,00,000
74	Publication for fraudulent purpose	Imprisonment up to 2 years, or/and with a fine up to ₹1,00,000

8.6 IPR Issues

Intellectual Property Rights (IPR) relate to the rights held by a person for their Intellectual Property, where Intellectual property refers to the work done by a person using the mind and his creativity. This Intellectual property comprises intangible assets such as literary work, artistic work, music-related work, etc. Commonly defined Intellectual Rights include patents, copyright, industrial design rights, trademarks, trade dress, and trade secrets. The following is the brief description of the IPR:

- **Patent:** A patent is a right of ownership conferred on an inventor (individual or a group or an institution) for a set period of time. Patents are issued for new and useful machines, manufactured goods, and industrial processes, as well as significant improvements to existing ones. Anyone intending to use a patented item must seek prior approval of the owner of the patent.
- **Copyright**: A "copyright" (or "author's right") refers to the legal rights that creators have over their literary and artistic works. Books, music, paintings, sculpture, and films are all covered by copyright, as are computer programs, databases, advertisements, maps, and technical drawings.
- **Industrial Design Right:** The Designs Act of 2000 protects industrial design intellectual property rights in India. The proprietor gains 'copyright' in the design as a result of registration, i.e. the exclusive right to apply a design to an article belonging to the class in which it is registered.
- **Trademark:** A trademark is a type of intellectual property that consists of a recognisable sign, design, or expression that identifies and distinguishes products or services from a specific source. An individual, a business organisation, or any other legal entity can own a trademark. A trademark can be found on a package, a label, a voucher, or the actual product.

- **Trade Dress:** A trade dress refers to a product's visual appearance, such as packaging, shape, and colour combination, that can be registered or protected by competitors in terms of their business and services. It aids consumers in recognising the product and distinguishing it from others. It also aids an illiterate consumer in distinguishing between products based on their packaging.
- **Trade Secret:** A trade secret is any company practice or process that is not widely known outside of the company. Trade secrets are often the result of internal research and development and provide the company with a competitive advantage over its competitors. A trade secret, unlike a patent, is not known to the general public.

There are various issues concerned with these rights, such as piracy of software, plagiarism (presenting the literary work done by someone as your work), trademark violations, patent violations, copyright violations, etc.

Match the following:

1. IT Amendment Act 2008	a. Protecting intangible assets
2. IPR	b. copying text from a published book in your book
3. Patent	c. Unethical hacker
4. Copyright	d. Government issued right to inventor
5. Trademark	e. Author's Right
6. Plagiarism	f. Company Logo
7. Cracker	g. Cyber Security

Let's Summarise

- A security threat is the possibility of the occurrence of an attack.
- Some of the security threats to a computer system are:
  - **Virus:** a software that may harm a computer system by overwriting or corrupting the files
  - **Worm:** a self-replicating software that duplicates itself without human intervention
  - **Trojan Horse:** a code that appears to be desirable and useful but ends up harming the system by attaching itself with a safe application
  - **Spams:** An unwanted e-mail, often sent by an advertiser.
- A process of having unauthorized access to someone's computer or computer network for stealing resources such as passwords or confidential files or causing harm to the network or system is called hacking.
- A firewall is a hardware device or a software mechanism or a combination of both and provides security to a computer network.
- Hypertext Transfer Protocol (HTTP) is a set of rules that governs how data should be communicated between client and server.
- IT Act 2000 aims at defining a legal framework for using WWW. It defines what constitutes cybercrimes and prescribes punishment for them.
- Cyber laws are the statutory provisions against various forms of cybercrimes.
- Cybercrimes are crimes related to the misuse of computers or the Internet, such as theft, fraud, and forgery.
- Intellectual Property Rights (IPR) relate to the rights held by a person for their intellectual property, where intellectual property refers to the work done by a person using the mind and his creativity.
  - **Patent:** It is a right of ownership conferred on an inventor (individual or a group or an institution) for a set period of time.

- **Copyright:** It refers to the legal rights that creators have over their literary and artistic works.
- **Industrial Design Right:** It protects industrial design intellectual property rights in India.
- **Trademark:** It is a type of intellectual property that consists of a recognisable sign, design, or expression that identifies and distinguishes products or services from a specific source.
- **Trade Dress:** It refers to a product's visual appearance, such as packaging, shape, and colour combination, that can be registered or protected by competitors in terms of their business and services.
- **Trade Secret:** It is any company practice or process that is not widely known outside of the company.

Solved Exercises

A. Multiple Choice Questions

1. Which of the following is not a network security goal?

a. Integrity b. Availability c. Sharing d. Confidentiality

2. Which of the following amounts to an attack on a computer network?

a. An attempt by an employee of an organization to use his designated computer
b. An attempt by an employee of an organization to log into the user account of his/her boss.
c. An attempt to add a printer to a network as desired by the manager.
d. An attempt to add an authenticated user id and password on the network server.

3. Which of the following helped in the spread of a computer virus?

a. Flash drives b. Shared Printer
c. A sick person using a computer on a network d. A student learning a computer

4. Which of the following is an example of a file virus?

a. Michelangelo b. Romeo and Juliet c. Melisa d. Worm

5. Which of the following types of viruses spread through a Microsoft Word document sent as an email attachment to the first fifty recipients in Microsoft Outlook?

a. Michelangelo b. Romeo and Juliet c. Melisa d. Worm

6. Which of the following do not require a host computer for replication?

a. Worm b. File virus c. Boot Sector virus d. Macro Virus

7. Which of the following is NOT the characteristic of a spam?

a. Emails sent in bulk b. Unwanted emails
c. Commercial emails for advertisement purposes d. Replicate themselves

8. Which of the following is True about a web cookie?

a. It is the text sent by the user's browser to the server
b. It is stored in web browser's directory on the host computer
c. It aims to enhance the web developer's experience while creating a web page.
d. It depends on the specific Internet Service Provider

9. Which of the following helps to keep control over the data that is being shared by the employees of an organization over the Internet?

a. Cookies b. Trojan Horse c. Firewall d. Modem

10. Which protocol is used to transfer hypertext documents over the internet?

a. HTTS b. HTST c. HTTCP d. HTTPS

11. Which of teh following does not come under the purview of IT Act 2000?

a. Hacking b. Taking backup of one's own data

c. Breach of Privacy d. Publishing false digital certifcate

12. Which of the following is NOT considered as cyber crime according to IT Act 2000?

a. Sending spam e-mails to uninterested recipients.

b. Phishing

c. Hosting a web site

d. Posting offensive content on any site or sending it to anyone.

13. Which of the following works will be protected by IPR?

a. Your result stored in a CD

b. A mobile phone purchased by you

c. A piece of music created by you

d. Video recording of a public function shared freely on social media

14. Which of the following is the legal right given to the creator of a literary or art work?

a. Patent b. Copyright c. Trademark d. Trade Secret

15. Which of the following is not an act of ethical hacking?

a. Exploring a web site to expose its weak points

b. Accessing the password of your neighbour's router without his/her permission

c. Accessing an instagram account by the cyber police for investigation

d. Using your father's WhatsApp account by to send a message as requested by him

B. State whether the following statements are True or False:

1. User id and password ensure authenticated access to a computer user. _____

2. The boot sector virus attaches itself to the executable files. _____

3. Viruses and Worms work in the same manner. _____

4. A spam email is an unwanted email. _____

5. A cookie is stored on the web server. _____

6. Meeting someone online is NOT cyber crime. _____

7. A patent is a right given to an inventor. _____

8. Plagiarism is not illegal according to IT Act 2000. _____

9. Crackers are ethical hackers. _____

C. Fill in the blanks.

1. When an IT resource ceases to be available to the intended users, it is called a _____ attack.

2. An attempt to access an IT resource illegitimately is called an _____.

3. A _____ floods a hard disk by automatically creating several copies of itself.

4. A _____ is a code that appears to be desirable and useful but ends up harming the system.

5. The data sent by the web server to a user' browser in the form of encrypted files is called a _____.

6. _____ is used to establish a connection between a client and a server before initiating the transfer of data.

7. _____ right relates to a creative work produced by a person.

8. _____ are the statutory provisions against various forms of cybercrimes.

9. _____ refers to the legal rights owned by the creators of literary and artistic works.

10. An unauthorized access to someone's computer for stealing resources such as passwords is called _____.

D. Answer the following questions:

1. Why is it important to secure a network?

Ans. As most of the security violations take place when data moves over a network, it is extremely important to securitize the network.

2. What is the job of a network administrator in the context of information security?

Ans. The network administrators are responsible for enforcing a robust security policies to keep an organization's IT infrastructure safe. They install, config, and maintain the hardware and software to detect security threats.

3. Give any two characteristics of a firewall.

Ans.
- Provides security to a computer network
- It may be a hardware device, a piece of software, or a combination fo both

4. Differentiate between File virus and Macro Virus.

Ans. File Virus:
- attaches itself to executable files
- overwrites or appends the malicious code
- Example : Romeo and Juliet

Macro Virus:
- executable file that is often received as an email attachment
- affects the system programs as the attachment is opened
- Example: Melissa virus

5. Expand HTTPS, TCP, SSL

Ans. HTTPS: Hypertext Transfer Protocol Secure

TCP: Transmission Control Protocol

SSL: Secure Socket Layer

6. Why is HTTPS preferred over HTTP?

Ans. HTTPS ensures secure communication of data as data is sent in an encrypted format.

7. What is the purpose of IT Act 2000?

Ans. IT Act 2000 aims at defining a legal framework for using Information Technology. It defines what constitutes IT related crimes and prescribes punishment for them.

8. Identify the relevant section of IT Act 2008 for each of the following offences:

a. Breach of confidentiality and privacy

b. Tampering with computer source documents

Ans. a. Section 72

b. Section 65

9. Name any four activities that are considered cyber crimes according to IT Act 2000.

Ans.
- Sending spam e-mails to uninterested recipients.
- Hacking someone's account or system.
- Stealing someone's personal information through phishing
- Hosting a site carrying lots of malware or being a source for spreading them.
- Harassing someone through e-mails, messages or social networking sites.
- Posting offensive content on any site or sending it to someone.
- Defaming someone using the Internet.

10. What do you mean by the term hacking? What are the two types of hackers?

Ans. Hacking refers to having access to someone's computer or computer network.

When unauthorised access is acquired for stealing resources such as passwords or confidential files or causing harm to the network or system, the hacker is called a bad hacker or a cracker. However, a good hacker explores a system to expose its vulnerability to attacks so that corrective action may be taken.

11. Raza wanted to purchase sunglasses for himself. He browsed the web to have an idea about the cost of various sunglasses. Thereafter, he observed that he automatically started getting advertisements and suggestions on his browser, even though he was not searching for them any more. Why did this happen?

Ans. This happened because of web cookies. Raza's browsing history, according to his activity log, was stored in the browser's directory.



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): A virus can spread through email attachments.
Reasoning (R): A virus that is spread through email attachments does not require a host application to replicate itself.

2. Assertion(A): A cookie spoils user's browsing experience.
Reasoning (R): A cookie is a text file storing one's browsing history.

3. Assertion(A): A firewall inspects the incoming and outgoing data to/from network.
Reasoning (R): A firewall is a device or software or a combination of both.

4. Assertion(A): https ensures safe browsing on the Internet.
Reasoning (R): https employs Secure Socket Layer(SSL) that enables data to be transmitted in encrypted format.

Ans. 1. c 2. d 3. b 4. a



Case-based Questions

1. Ruhi has just created her account on social media. She has accepted friend requests of some of her classmates and shares her pictures with them. One day, she finds her morphed picture with obscene comments on social media. Is Ruhi a victim of cyber crime? Which act can help her seek legal remedy?

Ans. yes, Ruhi is a victim of cyber crime (cyber bullying).

IT Act 2000 can help her seek legal remedy.

2. Sarabjeet is very active on social media. One day, his friends receive messages from his account asking for money as he has lost his wallet. But, Sarabjeet has not sent these messages. What do you think has happened? What should he do?

Ans. Social media account of Sarabjeet has been hacked. He should immediately change his password and report the matter to Cyber Crime Cell/cyber police.

3. Nimisha is the network administrator of an IT company. She has been asked by her manager to make sure that the IT infrastructure of the company is safe from any type of unauthorized access. Which device/software should Nimisha use?

Ans. Firewall

4. Hussain works in an IT company. For any type of official work, he has to log in to the organization's network. One day he receives an email from an unknown source and accidentally downloads its email attachment. Soon after that, the entire network crashed. What could be the possible reason for the same.

Ans. The email attachment might contain malware (most probably a worm) that has replicated itself and caused the network to slow down and eventually crash.

Assessment

A. Multiple Choice Questions

1. Which of the following is not an essential goal of network security?
a. Confidentiality b. Availability c. Integrity d. Profit generation

2. Which of the following is NOT a malicious software?
a. Virus b. Worm c. Backup d. Trojan Horse

3. Which of the following statements is true about computer viruses?
a. It is a hardware.
b. It can never embed itself in an executable program file.
c. A flash drive cannot spread viruses.
d. It may overwrite the files of the infected computer.

4. Romeo and Juliet is a type of _____.
a. File virus b. Macro virus c. Boot sector virus d. worm

5. Files of which of the following applications were used to spread Melissa virus?
a. Microsoft Excel b. Microsoft Word c. Microsoft PowerPoint d. Microsoft Outlook

6. Which of the following is true about worms?
a. It is a malware.
b. It replicates itself and flood the hard disk of the computer.
c. Romeo and Juliet is an example of a worm.
d. It does not require any host application.

7. Which of the following are bulk mails generally sent for advertisement purposes?
a. Worm b. Spam c. Firewall d. Plug-ins

8. Which of the following is true about cookies?
a. It is used to enhance user's experience on a website.
b. It is stored in server's directory.
c. It is based on the browsing activity log of the user.
d. It is in the form of encrypted files.

9. What is the full form of SSL with respect to secure transmission of web pages on the Internet?
a. Safe Socket Layer b. Secure Software Layer c. Secure Socket Layer d. Safe Software Layer

10. When did the Information Technology Act 2000 come into action?
a. 17 October 2000 b. 15 October 2000 c. 17 November 2000 d. 15 November 2000

11. Which of the following is NOT a cyber crime?
a. Unethical hacking b. Phishing c. Bank Burglary d. Spamming

12. Which of the following refers to the right of ownership given to the inventor?
a. Copyright b. Trademark c. Licence d. Patent

B. State whether the following statements are True or False:

1. A worm replicates itself and floods the hard disk. _____

2. A trojan horse is hardware. _____

3. Spams are bulk emails. _____

4. An ethical hacker is also known as a cracker. _____

5. A cookie is a piece of text stored on the server. _____

6. A firewall is a piece of software that provides security to the network. _____

7. An invention is patented for a set period of time. _____

8. An artist's painting is protected through patents. _____

C. Fill in the blanks.

1. The possibility of occurrence of attack is known as _____.
2. Melissa virus is an example of _____ virus.
3. A macro virus is an executable file that is generally received as an _____.
4. A _____ tries to hack the system to identify its weaknesses.
5. A _____ monitors incoming and outgoing data on a network.
6. The Intellectual Property Right(IPR) that protects books, music, and films is _____.

D. Answer the following questions:

1. What are the key security concerns?
2. What is meant by an attack on an information system? How are attack and vulnerability related?
3. Clearly distinguish between an attack and vulnerability with the help of relevant examples.
4. Is there any similarity between computer virus and human virus?
5. How does a virus harm our computer system?
6. Differentiate between the following:
   (i) Virus and Worm
   (ii) HTTP and HTTPS
   (iii) Trademark and Trade Secret
   (iv) Hacker and Cracker
7. Consider the following scenarios and identify each of them as virus, worms, trojan horse, or spam. Justify your answer in each case.
   (i) A code containing some instructions was downloaded when Alex opened an e-mail attachment, and it got attached to a target program and took control of the program. It then replicated and attached itself to another target program and gained control over it, and thus, it spread to several target files.
   (ii) Alex downloaded an e-mail attachment containing "Happy New Year.exe". On clicking, a video file started playing on his system. While the video was playing, a code got downloaded to his computer and it started attaching the same code with every e-mail that Alex sent using his computer.
   (iii) As soon as Bob has downloaded an MP3 file, he started getting multiple unsolicited commercial e-mails every day.
   (iv) A malicious code started by taking over the victim's Gmail program. It then used a macro to hijack their e-mail system and send messages to the addresses in their mailing lists. This malicious code spread so rapidly that e-mail systems were overwhelmed, and the entire network got congested.
8. Identify the relevant section of IT Act 2000 and state the penalty which is liable for each of the following offences:
   (i) Writing abusive messages on someone else's profile on a social networking website.
   (ii) Post materials on a social networking website with the intent of inciting communal riots/ disharmony.
9. An internet e-mail user received an e-mail with an attachment named "Happy2021". When the user clicked on it, a video with the message "Happy 2021" was played. While this was happening, a code got downloaded on the user's computer and it attached the same code to all the mails that the user sent. What type of software attack was it?



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): Ethical hacking is a cybercrime.
   Reasoning (R): IT Act 2000 lays down statutory provisions against various forms of cybercrime.
2. Assertion(A): Nestle owns the patent to manufacture Maggi masala.
   Reasoning (R): A patent is the right of ownership conferred to the inventor.

3. Assertion(A): A trademark can be any sign that distinguishes services from a specific source.
Reasoning (R): A poem written by a poet is protected under IPR.



Case-based Questions

1. Roy is a script writer for a theatre company. He often gives his script to actors so that they prepare for their respective roles. One day, he finds that a movie has been made using his scripts, and the film's producer-director has neither informed him nor given any credit to him for his script. On further investigation, he finds that one of his theatre actors had passed on the script to the producer-director of the movie. Which crime has been committed by the producer-director of the movie? Can he seek legal protection? If yes, how?
2. Surabhi is working in a pharmaceutical company. She has recently invented a drug to cure breast cancer. What should she do to protect her invention from rival companies?

Answers

Multiple Choice Questions

1. (c)	2. (b)	3. (a)	4. (b)	5. (c)	6. (a)	7. (d)	8. (b)	9. (c)
10. (d)	11. (b)	12. (c)	13. (c)	14. (b)	15. (b)			

True or False

1. (T)	2. (F)	3. (F)	4. (T)	5. (F)	6. (T)	7. (T)	8. (F)	9. (F)

Fill in the blanks

1. (denial of service (DoS)	2. attack	3. worm	4. Trojan Horse
5. cookie	6. TCP	7. Intellectual property	8. Cyber laws
9. Copyright	10. Hacking		

9 DATABASE MANAGEMENT SYSTEM

Chapter Outline

9.1 Introducing DBMS

9.2 Understanding Data Model

Introduction

We have been dealing with data in every aspects of life, much before computers' advent. Railways, hospitals, schools, universities, defence, and various businesses deal with vast amounts of data that need to be stored and managed efficiently. Since, organisational decisions heavily depend on the timely availability of information, the companies felt the need for a system that could handle the data efficiently. Database Management System (DBMS) helps store related data and manage it through application programs. Structured Query Language (SQL) is the most popular language for database creation and management. In this chapter, we shall study the basic concepts of DBMS.

9.1 Introducing DBMS

Data are raw facts. We often need to extract useful information from the data that forms the basis of decision-making. For example, consider a class of 50 students who have indicated their preference for the game they want to play out of the four games available: basketball, badminton, cricket, and table tennis. While individual choices are important, the school administration would like to know how many students want to play each game. Thus, summary information that cricket, badminton, basketball, and table tennis are the choices of 23, 10, 9, and 8 students, respectively, will be helpful to the school for planning purposes. Further, the preceding information concludes that cricket is the most popular game. Thus, we have obtained beneficial information by processing the raw data about individual student preferences. Figure 9.1 illustrates the relationship between data and information:

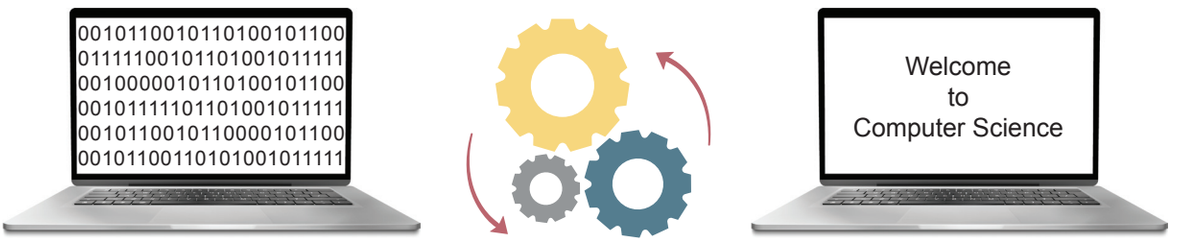


Fig 9.1 Relationship between data and information

To retrieve valuable information from data, we need to organise the data in a form that makes it easy to retrieve and process. Often, it is convenient to manage the data in a tabular format. Indeed, data has been stored in manual registers in the form of tables since times immemorial. Table 9.1 shows sample data about individual student preferences of the games they want to play.

Roll_No	Name	Game
1	Pushkar	Cricket
2	Aadhira	Badminton
3	Jayaprakash	Cricket
4	Priyabrata	Table-tennis
5	Farook	Cricket
6	Amisha	Cricket
7	Aayam	basketball

Table 9.1 Game preferences

Each row in Table 9.1 has data about one student's preference for a game. Column headings describe the information stored in the columns. Thus, for each student, `RollNo`, `Name`, and `Game` are stored in the first, second, and third columns respectively. In addition to the user data, DBMS also keeps a description of the stored data, called metadata (also called data about data). For example, in the above example, DBMS may store the table's name, names of columns, and the type of data that columns contain.

Metadata: description of data like names of tables, names and types of attributes.

9.1.1 DBMS

For creating and managing databases, several application programs are required. Database Management System (DBMS) is a collection of programs that help define, construct, and manipulate the database. Database definition includes specifying the structure of the database, type of data, and restrictions imposed on data. Database construction deals with storing the actual data in the database. Finally, database manipulation deals with the modification or retrieval of data. Figure 9.2 represents a database system that comprises DBMS software and the database:

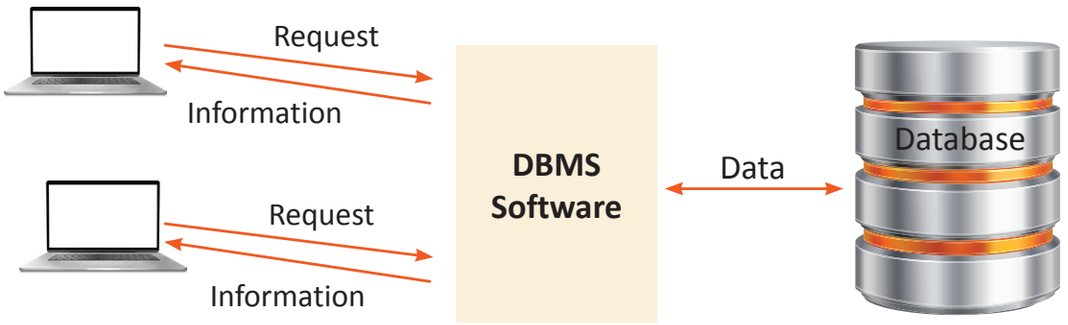


Fig 9.2 User interface with DBMS software

Note that the DBMS acts as an interface between end-users and the database. As every access to data in a database must go through DBMS, it can control access to data by checking against unauthorised access to data. Since, several end-users may access the same database, DBMS facilitates sharing of data.

Users of DBMS may belong to one of the following five types:

- **End Users:** Users who use the DBMS for retrieving or modifying the data.

- **Database Administrator (DBA):** Person who manages the entire database by controlling access and acquiring resources (both hardware and software).



DBA: responsible for defining data to be stored and access rights of various types of users.

- **Database Designers:** People who analyse the requirements of the end-users and design the structure of the database.
- **Systems Analysts:** People who analyse the data requirements of the end-users and arrive at the objectives and descriptions of the database.
- **Application Programmers:** People who develop programs to meet the objectives identified by the systems analysts.

9.1.2 Need for DBMS

In the olden days, the data was managed manually in registers or as papers in files. Related files were physically placed in folders, and folders were organised in cupboards or wardrobes. If we wish to manage all the information about a company manually, we would need separate file folders to maintain data about employees, departments, projects, etc. The data about an employee would be typically fragmented in several files. Further, there may be multiple copies of the same data in different files. For example, an employee's leave record may be maintained by the local office and the head office. Similarly, the human resource (HR) department and the accounts department may maintain their copies of the salary payable to the employees. Multiple copies of the same data will lead to the possibility of inconsistency in data as it may not be possible to apply updates in all copies of the data simultaneously. Thus, as data increases, ensuring consistency of data becomes cumbersome.

Over time, manual file systems were replaced by computerised file systems. Searching or modifying data became much faster in these systems. Based on the computerised file systems, various information systems were developed to meet the requirements of businesses. However, these systems were also plagued with the problem of data redundancy because they still contained multiple copies of the same data. Data redundancy leads to the following problems:

- **Wastage of Storage Space:** Same data appears at various places.
- **Increased Data Processing Effort:** An update operation involving a data item that appears in several files requires updating all those files. For example, if an employee's salary is updated, all the files containing information about the employee's salary will have to be updated.
- **Inconsistent Data:** While updating data, it may happen that one file correctly reflects updated data whereas the other file still contains old data or incorrectly updated data. For example, suppose an employee's salary needs to be increased by 3500. Using a conventional file system, a programmer may carry out the update correctly in the file relating to the accounts department. But the salary mentioned in a file of the HR department may not have been updated. There could be another situation that the salary in the HR department's file might be increased by only 3000 because an incorrect version of the data file was used for the update. This kind of anomaly is quite likely to happen if different individuals are responsible for updating the same data at multiple places.

Apart from data redundancy, computerised file systems have several other disadvantages as mentioned below:

- **Structural Data Dependence:** A change in the structure of a file may require changes in the applications programs that use this file. For example, the inclusion of a new attribute email-id in a file containing *employee* data will lead to changes in the programs that use the *employee* file. Similarly, a change in data characteristics, such as an increase in the length of the address line, will affect other programs that use the *employee* file. Thus, application programs become dependent on the file structure.
- **Lack of Access Control Mechanism:** Typically, organisations enforce a hierarchical system for defining access rights of different categories of users. For example, while the head of the EDP department may access all or most of the information available in an organisation, the head of the HR department, staff of the HR department, head of the accounts department, and accounts clerk will have restricted access to data.

It is hard to enforce rules for authorising legitimate access to data in applications developed using conventional file systems.

- **Lack of Data Sharing:** Since each department is dealing with its copy of data files and the structure of these files may vary significantly across the departments, sharing data becomes a challenge, often an impossible task.
- **Lack of Design:** As each department manages its data, it may not have access to expert advice may be needed how to organise the data into files in a manner that enables easy processing.

The problems mentioned above are handled smoothly by DBMS, as discussed in the next section.

9.1.3 Advantages and Disadvantages of DBMS

The database management systems offer several advantages over conventional file systems as mentioned below:

- **Sharing Data:** As DBMS maintains a single data repository, all users have shared access to the same data.
- **Reduces Data Redundancy:** Storing the data centrally in a single repository avoids the problem of data redundancy.



Centralized storage and control of data leads to: reduced redundancy, integrity of data, security, data independence.

- **Data Independence:** Changes in the data structure, such as adding a new column in a table, do not affect existing application programs. Similarly, a change in the data type of an attribute of interest, say, `UserID` from an *integer* to a *long integer,* does not affect existing application programs because users access data through the DBMS interface.



Data Independence: columns may be added or deleted, data types may change, but applications programs do not change.

- **Database Authorisation:** DBMS authenticates access to the database based on user name and password. Indeed, DBMS can also enforce authorisation based on data values stored in the database; for example, a user may be allowed to access details of employees whose salary does not exceed ₹ 75000.
- **Backup and Recovery:** DBMS provides a measure of safety by providing support for backup and recovery of data. In case of some user error, hardware failure, or any catastrophic event, the DBMS can recover the database from the point it was failed. Further, DBMS maintains a historical record of all changes made to the database and can recover the database from any point in time.
- **Integrity:** DBMS ensures the correctness of data stored by introducing various integrity constraints. For example, each employee must have a unique id. Every employee's date of birth should lie in a reasonable range, and no employee should be allocated to a department that does not exist in the organisation. DBMS can also enforce constraints like the date of birth of an employee cannot be left unspecified. These restrictions provide a way to ensure the integrity of data.



Database Integrity: consistency and correctness of data.

Having all these advantages, DBMS also have a few disadvantages as follows:

- **Initial Cost:** Developing applications based on DBMS involves hardware and software costs. Further, there may be a need to train employees.
- **Complexity:** As DBMS offers various functionalities for a wide variety of users (like the sharability and integrity of data, enforcement of access rights to data, and support for data manipulation), it becomes a large and complex software.
- **Database Failure:** System crash, hard disk failure and application software error may cause database failure. In that case, all the data stored in database may lost and causes organisation collapse.

9.2 Understanding Data Model

To build a house, we require a house blueprint that serves as the basis for constructing the house. Similarly, a data model is a representation (often graphical) of the structure of real-world objects. Developing a database system requires a design (model) that will be the basis of communication between designers and programmers. A data model comprises the following components:

- **Entity**: An entity is a real-world object about which data is stored in the database. For example, a company's database may have *employee* and *department* entities.
- **Attribute**: An attribute defines the characteristics of an entity. For example, employee id, date of birth, address, and salary form attributes of an employee.
- **Relationship**: A relationship specifies how two entities are linked (or related) to each other. For example, in a database that deals with employees of a company who work in different departments, the *employee* entity is related to the *department* entity since an *employee works in a department.*
- **Constraint**: Constraint specifies the restrictions imposed on the data stored. For example, each employee should have a unique identification number.

9.2.1 Relational Data Model

The relational model was founded in 1969 by English computer scientist Edgar F. Codd and represents all data in terms of tuples, which are grouped into relations. Each relation is a table comprising rows and columns. We will use the terms, relation and table interchangeably.

We consider a company having several employees, who work in different departments. Further, an employee may work on several projects in the company. Let us call the database `COMPANY`. We organise information about the company into four tables, `EMPLOYEE`, `DEPARTMENT`, `PROJECT`, and `WORKS_ON`, as shown in Figure 9.3:

Table: EMPLOYEE

ID	FName	LName	Gender	Address	City	Pin_Code	DOB	Salary	Dept_No
10001	Raj	Reddy	M	West Godavari	Andhra Pradesh	534197	1980-06-13	100000	2
10002	Dhiraj	Bora	M	Dispur, Kamrup, Assam	Guwahati	781005	1975-09-30	85000	1
10003	Muskan	Taneja	F	8/33, Geeta Colony	Delhi	110031	1990-01-25	100000	2
10004	Hiten	Oberoi	M	15, Dimna Road, Mango	Jamshedpur	831018	1985-06-24	100000	4
10005	Anshul	Verma	M	House 10, Sector 16, Gautum Budh Nagar	Noida	201304	1990-01-01	100000	1
10006	Rajit	Gadh	F	12, Beldih Triangle, Bistupur	Jamshedpur	831001	1960-05-07	60000	4
10007	Taran	Adarsh	M	B-76, CST Road, Kalina, Santacruz East	Mumbai	400098	1965-01-13	70000	5
10008	Naval	Dhingra	M	E-14 Vivek Vihar	Delhi	110095	1975-08-04	70000	2
10009	Naveen	Basra	F	28, Aambagan Road, Sakechi	Jamshedpur	831001	1980-09-24	60000	4
10010	Savita	Ambedkar	F	C-49, G-Block, Bandra Kurla, Bandra East	Mumbai	400051	1987-07-11	50000	5

Table: DEPARTMENT

Dept_No	Dept_Name	Location	Mgr_Id
1	Accounts	Noida	10002
2	Administration	Delhi	10005
3	Home Goods	Mumbai	NULL
4	Automobile	Jamshedpur	10007
5	Textile	Mumbai	10004

Table: PROJECT

Proj_No	Proj_Name	Dept_Num
1	ProjectA	2
2	ProjectB	1
3	ProjectA	1
4	ProjectC	5
5	ProjectD	4

Table: WORKS_ON

Proj_No	Emp_ID	Hours
1	10001	23
2	10005	18
3	10002	4
2	10002	4
4	10010	30
5	10009	10
4	10007	8
1	10003	17
1	10008	21
2	10006	25

Fig 9.3 COMPANY Database

A relational database management system (RDBMS) stores data in the form of relations, also called tables. Let us study about some of the elements of a table.

Attribute

Each column of a table represents a unique attribute. For example, the table `EMPLOYEE` comprises ten columns representing `ID`, `FName`, `LName`, `Gender`, `Address`, `City`, `Pin_Code`, `DOB`, `Salary`, and `Dept_No`. Similarly, the table `DEPARTMENT` comprises attributes `Dept_No`, `Dept_Name`, `Location`, and `Mgr_Id`, the table `PROJECT` comprises the attributes `Proj_No`, `Proj_Name`, and `Dept_Num`, and the table `WORKS_ON` comprises attributes namely `Proj_No`, `Emp_Id`, and `Hours`.

One row of a table stores information about all attributes of an entity. For example, the following row of table `EMPLOYEE` stores information about an employee having employee id `10005`:

ID	FName	LName	Gender	Address	City	Pin_Code	DOB	Salary	Dept_No
10005	Anshul	Verma	M	House 10, Sector 16, Gautum Budh Nagar	Noida	201304	1990-01-01	100000	1

Attribute: a column in a table.
Attribute value: a value inserted in the column corresponding to an attribute must confirm to the attribute type.

Relational Schema

Relational schema describes the structure of a relation. It includes the names of attributes of a table, types of attributes, and the constraints on a table. Consider the relation schema of `EMPLOYEE` relation:

`EMPLOYEE(ID, FName, LName, Gender, Address, City, Pin_Code, DOB, Salary, Dept_No)`

The names of the relations and the attributes must relate to the attribute's real-world meaning. For example, `FName` and `LName` denote the first name and last name in the `EMPLOYEE` relation, respectively. The rest of the attribute names are self-explanatory.

Relational schema: attribute names, types, constraints.

Degree

The number of attributes in a relation is known as the degree of relation. For example, relation `EMPLOYEE` has degree 10, relation `DEPARTMENT` has degree 4, and each of the relations `PROJECT` and `WORKS_ON` has degree 3.

Degree: number of attributes in a table.

Tuple

In a relation, a row represents an entity's instance (also called occurrence) and is known as a **tuple**.

For example, the tuple in the EMPLOYEE table for the employee with 10001 as ID is shown below:

```
+-------+--------+----------+--------+-------------------+----------------+----------+------------+--------+---------+
| 10001 | Raj    | Reddy    | M      | West Godavari     | Andhra Pradesh | 534197   | 1980-06-13 | 100000 |       2 |
+-------+--------+----------+--------+-------------------+----------------+----------+------------+--------+---------+
```

Tuple: a row in a table.

Entity Set

The set of all tuples in a relation is known as the *entity set*. The entity set is also called the state of the relation at that time. For example, in the EMPLOYEE table, all the ten rows collectively constitute the entity set of relation EMPLOYEE.

Entity set: Set of all tuples at any point of time.

We express a tuple of a table by enclosing the comma-separated values of the attributes within angular brackets. For example, the relation state of relation DEPARTMENT is given below:

```
{
<1, "Accounts", "Noida", 10002>,
<2, "Administration", "Delhi", 10005>,
<3, "Home Good", "Mumbai", 10003>,
<4, "Automobile", "Jamshedpur", 10007>,
<5, "Textile", "Mumbai", 10004>
}
```

Cardinality

The number of rows in the entity set of a relation defines the cardinality of the relation. For example, the entity set of relation EMPLOYEE has cardinality 10.

Domain: a domain in relational database terminology defines the permissible range of values for an entity's attribute. For example, in the DEPT table, the attribute Dept_No can take one of the possible values: 1, 2, 3, 4, or 5.

Constraint

Each attribute is of a specific type, such as numeric or string. All the values in the column corresponding to an attribute must conform to its type. For example, attribute ID and Salary are of numeric types and the attributes FName and LName are of string type. The set of values that an attribute can take defines its domain. For example, the attribute Gender of relation EMPLOYEE may have the domain {'M', 'F', 'O'}, even though the current state of the table EMPLOYEE includes only two values, 'M' and 'F'. The domain of an employee's Salary may be the set of numeric values. We may restrict the domain of Salary by specifying a certain range of values, say, between 10000 and 100000. In this case, the DBMS will reject any attempt to add a tuple in the table EMPLOYEE where Salary does not satisfy the constraint 10000 <= Salary <= 100000.

Order within a Table

The order in which names of the attributes appear in a relation schema is not relevant to the DBMS for its operations. Similarly, DBMS ignores the ordering of tuples in a relation.

Answer the following questions:

1. Can a table have two identical tuples?
2. Consider the table MOVIE given below:

MOVIEID	TITLE	LANGUAGE	RATING	PLATFORM
M001	Pushpa: The Rise	Telugu	7.9	Amazon Prime Video
M004	The Mauritanian	Spanish	7.5	Showtime
M010	Jai Bheem	Tamil	9.1	Disney plus Hotstar
M011	Spider-Man: No Way Home	Hindi	8.8	NA
M015	Thalaivi	French	6.1	Netflix
M018	Zack Snyder's Justice League	English	8.1	HBO Max

What is the degree of the table MOVIE? What is its cardinality?

Order within a table: rows in a table are considered unordered.

NULL Value

If the value for an attribute is unknown for a tuple, DBMS uses the **NULL** value. An attribute value may be **NULL** in the following situations:

- The value exists but is currently unknown (for example, DOB).
- We do not know whether the value exists (for example, landline number).
- No value is applicable (for example, spouse name for an unmarried employee).

Note that in `DEPARTMENT` relation, `Mgr_Id` corresponding to the department having `Dept_No` 3 is **NULL** as a manager has not been assigned to the department.

NULL: denotes an unknown attribute value.

Key

Each row in a table is uniquely identified by an attribute or a combination of attributes. Such an attribute or a combination of attributes is called a key of the relation. For example, the value of attribute `ID` of relation `EMPLOYEE` is unique for each row. So, `ID` is a key of the relation `EMPLOYEE`.

As mentioned above, a key identifies a tuple in a table uniquely. Attributes that are part of the key are known as key attributes. When a key has more than one attribute, it is called a composite key. For example, in the relation `WORKS_ON`, neither `Proj_No` nor `Emp_Id` can act as the key of the table `WORKS_ON` (why?). However, attributes `Proj_No` and `Emp_Id` together form a composite key of the relation `WORKS_ON`. Let us learn about some keys that are used in the database.

Candidate Key

A set of attributes that uniquely identify each tuple in a relation is called its candidate key. The set of attributes that form a candidate key is minimal in the sense that no subset of a candidate key qualifies to be a candidate key.

For example, let us consider a scenario where no two projects have the same project number Proj_No. Further, the project names are unique within a department. However, different departments may have projects with the same name.

Candidate key: minimal set of attributes that identifies a tuple.

Let us consider candidate keys for the relation PROJECT in this scenario. As no two projects have the same Proj_No, the attribute Proj_No will serve as a candidate key. As Proj_No identifies the tuples in the table PROJECT uniquely, it is evident that the following combinations of the attributes will also pinpoint the tuples in the table PROJECT uniquely:

Proj_No

Proj_No, Proj_Name

Proj_No, Dept_Num

Proj_No, Proj_Name, Dept_Num

Proj_Name, Dept_Num

Note that in the combination of attributes: {Proj_No, Proj_Name}, the attribute Proj_Name is superfluous as the attribute Proj_No alone qualifies to be a candidate key. So, the combination of attributes {Proj_No, Proj_Name} does not qualify to be a candidate key. Similarly, none of the combinations of attributes qualifies to be a candidate key.

{Proj_No, Dept_Num}, {Proj_No, Proj_Name, Dept_Num}

1. Does Proj_Name alone qualify to be a candidate key of the relation PROJECT? Why?
2. Does Dept_Num alone qualify to be a candidate key of the relation PROJECT? Why?

As mentioned above, since two departments may have projects with the same name, Proj_Name alone will not form a candidate key. However, project names being unique within a department, Proj_Name, and Dept_Num, the combination of attributes {Proj_Name, Dept_Num} will also form a candidate key. In light of the above discussion, the table PROJECT has the following candidate keys:

{Proj_No}

{Proj_Name, Dept_Num}

Primary Key

One of the candidate keys is chosen as the primary key. Generally, the candidate key with the minimum number of attributes is chosen as the primary key. So, we choose Proj_No as the primary key for the relation PROJECT.

Primary Key: One of the candidate keys is designated as primary key.

1. ID is a candidate key for the table EMPLOYEE. Why?
2. Is there a candidate key other than ID for the table EMPLOYEE? Why?
3. Dept_No is a candidate key for the table DEPARTMENT. Why?
4. Is there a candidate key other than Dept_No for the table DEPARTMENT? Why?

Alternate Keys

A candidate key that is not chosen as the primary key is called an alternate key. For example, {Proj_Name, Dept_Num} will be an alternate key for the relation PROJECT.



Alternate Key: a candidate key(s) other than primary key.

Foreign Key

An attribute or a set of attributes in one table that refer to the primary key of another table is known as a foreign key. For example, in the table PROJECT, attribute Dept_Num refers to the attribute Dept_No of the table DEPARTMENT. So, Dept_No being the primary key of the table DEPARTMENT, attribute Dept_Num of relation PROJECT becomes a foreign key of the table PROJECT.



Foreign Key: An attribute of table R1 that is part of primary key of table R2 is a foreign key of table R1.



1. Dept_No of EMPLOYEE table is a foreign key referencing to Dept_No of DEPARTMENT table. How?
2. Mgr_Id of DEPARTMENT table is a foreign key referencing to ID of EMPLOYEE table. How?



9.2.2 Relational Model Constraints

The relational model allows us to define several constraints to ensure correct and consistent data in the database. These constraints are called integrity constraints. Let us learn about some integrity constraints for the relational model.

Entity Integrity Constraint

The DBMS ensures the integrity of data in the database. A tuple stored in the database is uniquely identified by its primary key. So, no primary key attribute can have a **NULL** value. This is called entity integrity constraint. Further, note that the database cannot have two tuples with an identical primary key value because the DBMS distinguishes between the two tuples based on primary key only. For example, since the Dept_No is the primary key of the DEPARTMENT table, therefore the following tuple cannot be added in DEPARTMENT.

```
<NULL, "Sales", "Delhi", 10008>
```



Entity integrity constraint: no primary key attribute can have a NULL value.

Referential Integrity Constraint

If a table has a foreign key that refers to a UNIQUE attribute (typically primary key) of another table, then the foreign key value should refer to an existing primary key value or should have a **NULL** value. For example, attribute Mgr_Id of relation DEPARTMENT refers to the attribute ID of relation EMPLOYEE. As ID is a primary key of relation EMPLOYEE, Mgr_Id becomes a foreign key for relation DEPARTMENT. For example, in the DEPARTMENT table, departments having Dept_No 1, 2, 3, 4, and 5 have Mgr_Id 10002, 10004, 10005, and 10007 respectively.

So, the referential integrity constraint requires that the EMPLOYEE table must have employees having ID values 10002, 10004, 10005, and 10007. Indeed, the table EMPLOYEE does have employees having the aforementioned IDs, So the foreign key constraint is satisfied. For Home Goods department (Dept_No = 3), Mgr_Id value is **NULL**.

This is in conformity with the referential integrity constraints. However, as of now, `Mgr_Id` value cannot be `10014` because there is no employee in the `EMPLOYEE` table having this value of attribute `ID`. We will discuss more about referential integrity in the next chapter.

NOT NULL Constraint

`NOT NULL` constraint can be applied to any attribute of a relation. According to this constraint, attribute on which it is applied should not have **`NULL`** value. For example, we may define this constraint on the `Salary` attribute of `EMPLOYEE` relation indicating that salary of all employees must be specified.

`NOT NULL` Constraint: ensure that an attribute on which it is applied never contains NULL value

UNIQUE Constraint

UNIQUE constraint when applied on any attribute, will ensure that no two tuples in the relation should have the same attribute value. For example, you may enforce each department to have a unique manager by applying `UNIQUE` constraint on attribute `Mgr_Id` of relation `DEPARTMENT`. This would imply that two departments cannot have the same manager.

Unique Constraint: Applies to an attribute. No two rows can have same value of the specified attribute.

Domain Constraint

Domain constraint allows you to specify a set of values that a particular attribute may take. For example, attribute `Gender` has a domain constraint ensuring that it may take only two values, either `M` or `F`. Similarly, we may constraint attribute `Salary` to take values within a reasonable range of numbers, say, between `8000` and `100000`.

Let's Summarise

- Database Management System (DBMS) is a collection of programs that help define, construct, and manipulate the database.
- Database administrator (DBA) manages the entire database by acquiring resources (both hardware and software) and controlling access.
- Disadvantages of conventional file-based information systems:
  - **Structural Data Dependence:** A change in the structure of a file may require changes in the applications programs that use this file.
  - **Lack of Access Control Mechanisms:** It is hard to enforce rules for authorising legitimate access to users at different levels of hierarchy.
  - **Lack of Data Sharing:** As different application programmers deal with common data but in different format, sharing data becomes a challenge.
- The Data Base Management Systems offer several advantages over conventional file systems:
  - **Sharing Data:** As DBMS maintains a single data repository, all users have shared access to the same data.
  - **Lack of Data Redundancy:** Storing the data centrally in a single repository avoids the problem of data redundancy.

- **Data Independence:** Columns may be added or deleted, data types may change, but applications programs do not change.
- **Database Authorization:** DBMS authenticates access to the database based on user name and password, or value of an attribute. for example, a user may be allowed to access details of employees whose salary does not exceed Rs 75000.
- **Backup and Recovery**: In case of some user error, hardware failure, or any catastrophic event, the DBMS can recover the database from the point it was failed.
- **Integrity**: DBMS ensures the correctness of data stored by introducing various integrity constraints. For example, DBMS will ensure that no two employees have the same Aadhaar number.

➢ Disadvantages of DBMS:
- **Initial Cost:** Applications based on DBMS involve higher costs of hardware and DBMS software.
- **Complexity:** As DBMS offers various functionalities, it becomes a large and complex software, requiring training of application developers.

➢ **Data Model:** A data model is a representation of the structure of real-world objects. A data model comprises the following components:
- **Entity:** An entity is a real-world object (such as employee and department in a company) about which data is stored in the database.
- **Attribute**: An attribute defines the characteristics of an entity. For example, employee id, date of birth, address, and salary form attributes of an employee.
- **Relationship**: A relationship specifies how two entities are related to each other. For example, in a ccompany's database the *employee* entity is related to the *department* entity since an employee works in a department.
- **Constraint**: A constraint describes the restrictions imposed on the data stored. For example, each employee should have a unique Aadhaar number.

➢ **Relational Data Model:** A relational database management system (RDBMS) stores data in the form of *relations*, also called *tables.*

➢ A table has the following characteristics:
- Each column of a table represents a unique *attribute*.
- An attribute has a *type* denoting the type of data in the column.
- Value in a column must conform to the attribute type.
- Relational schema describes the structure of a relation. It includes the names of attributes of a table, types of attributes, and the constraints on a table.
- **Degree:** No of attributes in the column is called degree of the relation.
- **Tuple:** A row in a table is called a tuple.
- **Entity Set:** The set of all tuples at a specific time in a relation is known as the entity set.
- **Relation state, Relation instance:** Other names for the entity set.
- **Cardinality:** Numbers of tuples in the entity set.

➢ **NULL**: When value of an attribute is unknown, it is denoted by **NULL.**

➢ **Candidate Key:** The minimal set of attributes that identifies a tuple in a table is called its candidate key. Minimality means that no attribute is superfluous.

➢ **Primary Key:** One of the candidate keys is chosen as primary key.

- **Entity Integrity Constraint:** No primary key attribute can have a **NULL** value
- **Alternate Key:** A candidate key that is not chosen as primary key.
- **Foreign Key:** A non-key attribute of table R1 that is part of primary key of table R2 is a foreign key of table R1.
- **Foreign Key Constraint:** If a table R1 has a foreign key (say, A) which is part of the primary key of another table R2, then
  - the value of foreign key A maybe be **NULL.**
  - the value of foreign key A in table R1 maybe be non- **NULL** (say, val). In this case. there must be a tuple in table R2 having val as the value of attribute B.
- **NOT NULL Constraint:** No row in the table can have a **NULL** value of the specified attribute.
- **Unique Constraint:** No two rows can have same value of the specified attribute.
- **Domain Constraint:** Domain Constraint allows you to restrict the set of values that a particular attribute may take.

Solved Exercises

A. Multiple Choice Questions

1. Which of the following is a feature of a Primary Key attribute?
a. Uniqueness b. NOT NULL c. No repetition d. All of these

2. Which of the following describes the degree of a relation?
a. Number of rows present in that relation b. Number of columns present in that relation
c. Average number of rows in that relation d. Count of numeric fields in a relation

3. Which non-primary attributes of a table refers to the primary key of another table?
a. Alternate Key b. Foreign Key c. Candidate Key d. NOT NULL key

4. What does a **`NOT NULL`** constraint on an attribute in a table ensure?
a. That the attribute cannot have a `NOT NULL` value.
b. That the attribute cannot have a `NULL` value.
c. That the attribute cannot have zero value.
d. That the attribute cannot have non-zero value.

5. The term Relation in an RDBMS denotes a _____.
a. Database b. Query c. File d. Table

6. A relation can have only one _____ key and multiple _____ keys.
a. primary, candidate b. candidate, alternate c. candidate, primary d. alternate, candidate

7. All attribute combinations inside a relation that can serve as a primary key are _____ keys.
a. candidate b. foreign c. alternate d. primary

8. A table has initially 4 columns and 5 rows. Consider the following sequence of operations performed on the table:
(i) 6 rows added (ii) 3 columns added (iii) 2 rows deleted (iv) 1 column deleted
What will be the cardinality and degree of the table at the end of given operations?
a. 8,7 b. 7,8 c. 9,6 d. 6,9

9. Which of the following is the correct explanation of Entity Integrity Constraint?
a. No primary key attribute can have NULL value b. A table must have a foreign key
c. No attribute of a relation can have a NULL value d. Every attribute of a relation must have a unique value.

10. In a relation, which constraint states that no two rows of a specified attribute can have the same value?

a. Unique constraint
b. Not NULL constraint
c. Entity Integrity constraint
d. Referential Integrity constraint

11. Which attribute key can accept multiple NULL values?

a. Foreign Key
b. Candidate Key
c. Primary Key
d. None of these

12. Which of the following is the correct statement with respect to RDBMS?

a. A primary key can contain a NULL value in any one row.
b. A primary key can have "Empty String" as one of the values.
c. A table that has a primary key must have an alternate key.
d. A table can have only one primary key.

B. State whether the following statements are True or False:

1. The names of the columns in a table must be unique.
2. Tuples in a table always appear in a specific order.
3. No value of primary key can repeat in different tuples in a table.
4. An entity set is the state of a relation at a particular time.
5. Any tuple can be deleted from any table without affecting the consistency of the database.
6. Deleting an attribute from a table will not change the database state.
7. In a relational data model, the terms table and relations can be used interchangeably.

C. Fill in the blanks.

1. _____ is an attribute whose values are derived from the primary key of some other table.
2. In a table, an alternate key is a _____ which is not the primary key.
3. Raw data is processed to get _____.
4. _____ acts as an interface between end users and database.
5. _____ refers to multiple copies of the same data in a database.
6. An _____ is a real-world object about which data has to be stored in a database.
7. The Relational Data model defines a database as a collection of _____.

D. Answer the following questions:

1. Consider the following table and answer the questions that follow:

Table: MEMBER

Mno	Name	Qty	PurchaseDate
201	Shirt	67	2019-07-14
202	Skirt	15	2020-12-12
203	Shoes	102	2020-09-10
204	Saree	59	2021-08-31
201	Shoes	50	2021-09-30

(i) In the above table, can we take Mno as Primary Key only? Justify your answer with a valid reason.
(ii) What is the degree and the cardinality of the above table?

Ans. (i) No: If we assume many transactions can be performed on day by the same member, even though it is not clearly evident from the table because the table shows only an instance of the relation and more rows may be added to it.
(ii) degree: 4, cardinality: 5

2. List two advantages and two disadvantages of DBMS.

Ans. The advantages of DBMS are:

- Sharing of Data
- Reduction in Data Redundancy

The disadvantages of DBMS are:

- High initial cost
- Complex software

3. Define an entity and an attribute with respect to RDBMS.

Ans. (i) An entity is a real-world object whose data is stored in the database.
(ii) An attribute defines the characteristics of an entity.

4. Describe Referential Integrity Constraint.

Ans. If a table R1 has a foreign key (i.e., a non-key attribute), say, A, that refers to a UNIQUE attribute, say, B (typically primary key) of another table R2, then

(i) the value of foreign key A maybe be **NULL**.

(ii) the value of foreign key A in table R1 maybe be non-NULL (say, val), then there must be a tuple in table `R2` having `val` as the value of attribute B. For example, suppose attribute `Mgr_Id` of relation `DEPARTMENT` refers to the attribute `ID` of relation `EMPLOYEE`. As `ID` is a primary key of relation `EMPLOYEE`, `Mgr_Id` becomes a foreign key for relation `DEPARTMENT`. So, in the `DEPARTMENT` table, if the department having `Dept_No` value 1 has `Mgr_ID` value `10005`, then the referential integrity constraint requires that the `EMPLOYEE` table must have an employee having `ID` value `10005`. Alternately, `Mgr_ID` value should be **NULL** for the `DEPARTMENT` having `Dept_No` value `1`.

5. Consider the following database tables:

CLASS (Primary key: ClassId, Section)

ClassId	Section	Stream	ClassTeacher
11	A	Science	Sona Sahu
12	B	Commerce	Anirban Bose
11	C	Arts	Vijaya Ahuja
12	B	Commerce	Sushmita Sen
12	C	Arts	Amisha Patel
11	B	Commerce	S. Jayanthi

STUDENT (Primary key: S_ID)

S_ID	SName	Class	Section	Marks
17	Mukesh Agrawal	11	C	88
21	Sanjoy	11	A	76
34	Meena	12	B	67
41	Latha	12	A	40
22	Niranjan	11	C	91

A command is executed to insert each of the following tuples in the table `CLASS`. For each operation, indicate whether it will be executed successfully. Justify your answer. If not, mention which DBMS constraint does the insert operation violate? Consider each operation independent of others.

(i) `<11, "A", "Science", "Pooja Sahu">`

(ii) `<11, "D", "Science", "Sara Nayak">`

(iii) `<11, NULL, "Science", "Ananya Kasaravall">`

(iv) `<NULL, "D", "Science", "Vinayak Damodar">`

(v) `<11, "D", "Science", "Sona Sahu">`

(vi) `<11, "D", NULL, NULL>`

(vii) `<11, "d", "Commerce","Ananya Kasaravall">`

Ans. (i) `<11, "A", "Science", "Pooja Sahu">`

Error: there is already a tuple in the CLASS table with the primary key `<11, "A">`.

(ii) `<11, "D", "Science", "Sara Nayak">`

Insertion will be successful as there is no tuple in the CLASS table with the primary key `<11, "D">`.

(iii) `<11, NULL, "Science", "Ananya Kasaravall">`

Error: Violation of entity integrity constraint. Section is part of the primary key, so cannot be **NULL**.

(iv) `<NULL, "D", "Science", "Vinayak Damodar">`

Error: Violation of entity integrity constraint. `ClassId` is part of the primary key, so cannot be **NULL**.

(v) `<11, "D", "Science", "Sona Sahu">`

Insertion will be successful as there is no problem in repeating `Stream` and `ClassTeacher`. The same teacher can be a class teacher for more than one class. But if this is not allowed, we will have to declare that `ClassTeacher` is `UNIQUE`.

(vi) `<11, "D", NULL, NULL>`

Insertion will be successful as non-key attributes can be **NULL** unless otherwise mentioned.

(vii) `<11, "d", "Commerce","Ananya Kasaravall">`

Insertion will be successful as Section "d" is different from already existing section "C" with class code.



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): DBMS stores both data and metadata.
Reasoning (R): Metadata is description of the stored data.

2. Assertion (A): DBMS acts as an interface between end-users and database.
Reasoning (R): An end user analyses the requirements of a database.

3. Assertion (A): DBMS reduces data redundancy.
Reasoning (R): Avoiding multiple copies of the same data reduces data inconsistency in a database.

4. Assertion (A): Sharing of data is facilitated using a DBMS.
Reasoning (R): A DBMS maintains a single repository of data that each user can access.

Ans. 1. b 2. c 3. a 4. a



Case-based Questions

1. Nidhi works in a publishing house. She wants to store names of all her marketing representatives along with their contact number and area of operation. The relational scheme of the Marketing table is as follows:
`Marketing(MRId, MRName, Area, ContactNo)`

Name the attributes of the relation Marketing. Also, according to you, which attribute should be made the primary key and why?

Ans. Attributes of relation `Marketing` are `MRId`, `MRName`, `Area`, and `ContactNo`. The attribute, `MRId` should be made the primary key as it uniquely identifies a record in the relation.

2. Amrit is a database administrator of an ecommerce company. He wants to store the details of all transactions in the table, `Transact` as given below:
Table: Transact

T005	Jay	1234567890	Cap	UPI	250
T006	Ginisha	8888899999	Hand Bag	Credit Card	4500
T009	Gurpreet	1111122222	TShirt	Debit Card	1200
T010	Hari Prasad	5656565656	Shoes	Net Banking	3250
T111	Gracy	6767676767	Shoes	Credit Card	2499
T115	Shlok	1234543210	Body Mist	UPI	440

Amrit is not very familiar with various terms used in a relational data model. Help her understand by answering the questions given below:

a. What is the relational shema of the table `Transact`.

b. Name the attributes of table `Transact`.

c. What is the degree and cardinality of the table `Transact`.

d. Name the candidate key(s), primary key and alternate key(s) of the table `Transact`.

Ans. a. Relational Schema of table Transact is :

`Transact(TID, CustName, ContactNo, ItemPuchased, ModeofPayment, Amount)`

b. Attributes: `TID, CustName, ContactNo, ItemPuchased, ModeofPayment, Amount`

c. Degree-6, Cardinality- 7

d. Candiadte keys- `TID, ContactNo`

Primary key- `TID`

Alternate key- `ContactNo`



Assessment

A. Multiple Choice Questions

1. Which of the following deals with modification and retrieval of data?

a. Database definition
b. Database Construction
c. Database manipulation
d. Database deletion

2. Who, among the following, analyses the data requirements of the end users and designs the structure of the database?

a. Database Administrator b. Systems Analyst c. Database Designers d. Application Programmers

3. Which of the following is the advantage of DBMS?

a. Increases data redundancy
b. Ensures shared access and integrity of data
c. Backup and recovery of data is not possible
d. Any person can access data from a database.

4. Which of the following describes the characteristics of an entity

a. Attribute b. Degree c. Cardinality d. constraint

5. Which of the following specifies the restrictions imposed on the data stored in a database?

a. Attribute b. Degree c. Cardinality d. constraint

6. Which of the following is NOT included in the relational schema?

a. Attributes of table
b. Queries used to retrieve data
c. Data types of attributes
d. Constraints on the attributes

7. Which of the following represents an entity's instance?

a. Attribute b. Degree c. Tuple d. cardinality

8. Which of the following defines the cardinality of a relation?

a. Number of rows b. Number of columns c. Name of the relation d. Constraints in a relation

9. Which of the following is used if the value for an attribute in a relation is not known?

a. None b. Unknown c. N-A d. NULL

10. A key that is a set of one or more columns and can identify a record uniquely is called _____.

a. Natural Key b. Candidate Key c. Alternate Key d. Primary Key

11. Which key accepts multiple null values?

a. Foreign Key b. Candidate Key c. Primary Key d. None of these

12. What is the full form of DBMS

a. Data Backup Management System
b. Data of Binary Management System
c. Database Management System
d. Database Management Service

13. Which of the following is the correct statement about DBMS?

a. DBMS is a collection of queries
b. DBMS is a high-level programming language
c. DBMS is a web development platform
d. DBMS is an application that stores, modifies, and retrieves records

14. In which of the following formats data is stored in the database management system?

a. Image b. Text c. Table d. Graph

15. Which of the following statements are true?
    a. There can be only one key with unique values in table
    b. There can be only one primary key in a table.
    c. There can be only one foreign key in a table.
    d. A table must have a primary key.

B. State whether the following statements are True or False:

1. One row of a table stores information about all attributes of an entity. _____
2. A database administrator is a person who manages the entire database by controlling access of hardware and software resources. _____
3. A key always refers to a single attribute that uniquely identifies a tuple. _____
4. A failed insert operation does not change the database state. _____
5. An end-user decides which components of the database he/she will access. _____
6. DBMS may allow a user to access the salary of the employees in the EMPLOYEE table, but may not allow access to the Addresses of the employees. _____
7. DBMS does not ignore the ordering of tuples in a relation. _____

C. Fill in the blanks.

1. Each column of a table represents a unique _____.
2. _____ describes the structure of a relation.
3. The number of attributes in a relation is known as _____ of a relation.
4. A set of all tuples in a relation is known as _____.
5. When a key in a table has got more than one attribute, it is known as _____.
6. A _____ constraint allows you to specify a set of values that a particular attribute may take.

D. Answer the following questions:

1. Which of the following statements are true?
    (i) Names of the columns in a table must be unique.
    (ii) Tuples in a table do not appear in a specific order.
    (iii) No component of the primary key can repeat in different tuples in a table.
    (iv) DBA decides which tables should be included in the database.
    (v) Any tuple can be deleted from any table without affecting the consistency of the database.
    (vi) Delete operation changes the database state.
    (vii) A failed insert operation does not change the database state.
    (viii) End-users have no role in the design of the database.
    (ix) An end-user decides which components of the database he/she will access.
    (x) DBMS may allow a user to access the `EMPLOYEE` table to a user, but may not allow access to the data relating to employees `SALARY>100000`.
    (xi) DBMS may allow a user to access the salary of the employees in the `EMPLOYEE` table, but may not allow access to the `Address` of the employees.
2. Consider a table that stores the data of an item using the following attributes:
   `ItemCode, ItemName, Price`
   Suggest an appropriate domain for each attribute.
3. Can a relation have two identical tuples? Why / Why not?
4. Define the following with respect to databases:
   (i) Domain (ii) Constraint (iii) Candidate Key (iv) Alternate Key
5. A company has several employees who work in different departments. However, each employee is assigned to a unique department. Each employee is assigned a unique employee id. Although department names are unique in the organization, they have also been allotted unique department ids. Consider the following tables:

Employee

Emp_id	Name	Dept_no
E001	Tarun Sharma	10
E002	Kajal Chugh	10
E003	Jatin Chawla	20
E004	Rahul Khanna	30

Department

Dept_no	Dept_name
10	Accounting
20	Sales
30	Marketing

(i) Identify a suitable primary key for the Employee and Department tables.

(ii) Give an example of each of the following operations (consider each part independent of others):

a. An insert operation that will be consistent with the current state of the Employee table.

b. An insert operation that will be consistent with the current state of the Department table.

c. An insert operation that will be inconsistent with the current state of the Employee table, but would be fine if the Employee table were empty.

d. An insert operation that will be inconsistent with the current state of the Department table, but would be fine if the Department table were empty.

e. An insert operation that will be invalid on the empty table Employee.

f. An insert operation that will be invalid on the empty table Department.

g. Will an operation to delete the tuple having Dept_no 20 be consistent with the current state of the Employee and Department tables? Justify your answer.

6. A company has several employees and has undertaken several projects. An employee may work on several projects. The database for the company has the following tables. Attribute names in the table clearly indicate their meaning.

```
Employee (E_id, Ename, City, Salary, Department,
  YearofJoining)
Project (P_no, PName, City, DeptName, StartYear)
WorksOn (P_no, E_id)
```

Identify a suitable primary key for each table. Also identify suitable foreign keys. If required, you may make suitable assumptions, state them, and justify your answer in light of the assumptions.

(i) Give an example of each of the following (consider each part independent of others):

a. A tuple insertion of which will be invalid on the empty table EMPLOYEE.

b. A tuple insertion of which will be invalid on the empty table PROJECT.

c. A tuple insertion of which will be invalid on the empty table WORKSON.

d. A tuple insertion of which will be valid on the empty table EMPLOYEE.

e. A tuple insertion of which will be valid on the empty table PROJECT.

f. A tuple insertion of which will be valid on the empty table WORKSON.

7. A company deals with several projects. The company has on its panel a number of suppliers which supply different parts. Although a part may be used in several projects, a shipment always relates to a specific part for a specific project being supplied by a supplier. The supplier, part, and project have been abbreviated as S, P, and J respectively in the description of the tables given below:

```
Suppliers (SNo, SName, Status, SCity)
Parts (PNo, PName, Colour, Weight, City)
Project (JNo, Jname, JCity)
Shipment (SNo, PNo, JNo, Quantity)
```

Identify primary key for each table. Also identify the foreign key constraints and foreign keys.

8. For the tables mentioned in Q7, and keeping in mind the primary keys identified by you, given an example of a tuple such that its insertion:

(i) on the empty table SUPPLIERS will generate an error.

(ii) on the empty table PARTS will generate an error.

(iii) on the empty table SUPPLIERS will be successful.

(iv) on the empty table PARTS will be successful.

(v) on the empty table PROJECT will be successful.

(vi) on the empty table SHIPMENT that will execute successfully.

9. Consider the tables EMPLOYEES, DEPARTMENT and RESOURCES which store the details of the employees, the department they belong to and the resources each department has.

```
Employee (employee_id, name, date_of_birth, age, department_id, address)
Department (department_id, name, number_of_people)
Resources (resource_id, department_id, resource_name, quantity)
```

Identify the primary and foreign keys for all the three tables. Determine which table should be created first so that the foreign keys can be referred correctly.

10. Differentiate between entity integrity and referential integrity. Support your answer with suitable examples.

11. A command is executed to insert each of the following tuples in the table STUDENT.

STUDENT (Primary key: S_ID)

S_ID	SName	Class	Section	Marks
17	Mukesh Agrawal	11	C	88
21	Sanjoy	11	A	76
34	Meena	12	B	67
41	Latha	12	A	40
22	Niranjan	11	C	91

For each operation, indicate whether it will be executed successfully. Justify your answer. If not, mention which DBMS constraint does the insert operation violate? Consider each operation independent of others.

(i) <18, "Mukesh Agrawal", 11, "C", 88>

(ii) <21, "Sanjay", 11, "A", 76>

(iii) <NULL, "Phule Bai", 11, "A", 88>

(iv) <20, NULL, 11, "A", 88>

(v) <20, NULL, 11, NULL, NULL>

(vi) <20, NULL, NULL, NULL, NULL>

12. Consider a table PERSON described as follows:

Field	Type	Null
AadhaarNo	decimal(12,0)	NO
last_name	varchar(20)	NO
first_name	varchar(20)	YES
birth_date	date	YES
PhoneNo	decimal(10,0)	YES

PERSON

AadhaarNo	last_name	first_nmae	birth_date	PhoneNo
672034567812	Sharma	Arjun	1990-08-31	8345123987
712349049911	Wadhwa	Sumit	1992-09-12	9812476543

AadhaarNo and last_name obey the **NOT NULL** constraint. AadhaarNo also obeys the uniqueness constraint UNIQUE. Check whether the following tuples can be added to the PERSON table? Justify your answer.

(i) <799575933699, NULL, "Pooja", "1990-10-01", 9776626565>

(ii) <712349049911, "Singh", "Gagan", "1990-11-11", 9812476543>

Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): Each attribute in a database must have at least one constraint specified with it.
   Reasoning (R): A constraint specifies the restrictions imposed on the stored data.
2. Assertion (A): If any row or column is added to a relation, its relational schema changes.
   Reasoning (R): A relational schema describes the structure of a relation.
3. Assertion (A): The set of tuples in a relation is known as the entity set.
   Reasoning (R): Deleting a record modifies the entity set of a relation.
4. Assertion (A): The value of spouse name for an unmarried candidate will be NULL in a database.
   Reasoning (R): If a value for an attribute is not known for a tuple, the DBMS uses a NULL value.

Case-based Questions

Consider a `Food` and `Beverage` department having `Dept_No` 6 located in `Pune`. Also, consider the following tuple for an employee with id `E0011` who is the manager of this department:

```
+-------+---------+-----------+----+-----------------------------------------------+------+--------+------------+-------+-------+
| E0011 | Rashmi | Singhania | F  | 7, First Floor, MSFC Building, Shivajinagar | Pune | 411016 | 1990-10-08 | 90000 |     1 |
+-------+---------+-----------+----+-----------------------------------------------+------+--------+------------+-------+-------+
```

You need to insert a tuple both for a new employee and a new manager. Now if you attempt to insert a tuple for a new department first, it will be rejected by DBMS since it violates referential integrity constraint as there is no employee with id `E0011`. Similarly, adding employee entry working in department 6 will also be rejected since there is no department with this number. Propose a solution for this problem that does not violate referential integrity constraints.

1. Lakshmi studies in class 10. Her teacher has just introduced her to databases. She has asked them to create a table, `myClass` to store details of their classmates such as name, father's name, contact number, birth date, hobbies, etc. Design a relational schema for the table. Also, specify the primary key.
2. Joseph has just come to know about database applications that help to organize data for efficient storage and retrieval of data. His school is organizing an Annual Athletics Meet. He wants to design a database for maintaining records of players participating in various events. His database will have two tables:
   Table1: to store details of various events such as Event Id, Event Name, and Event Date.

   Table 2: To store the details of players such as Player Id, Player Name, Age, Event in which he is participating, and result.

   Design a relational schema for both tables, giving appropriate names ot tables and attributes. Also, specify the primary key and foreign key in each table.
3. Abhishek has been assigned a project by his teacher on Railway Management. He has created the following two tables:
   Table: `Trains`

TrainID	TrainName	Source	Destination
3457	Rajdhani	Delhi	Kolkata
1274	Jammu Express	Jammu	Delhi
3891	Shatabdi	Lucknow	Mumbai
6192	Jhelum Express	Amritsar	Mumbai
4072	Vande Bharat	Delhi	Trivandrum

Table: Passengers

RefNo	TrainID	PassengerName	DOJ
R112	3457	Surya	2021-03-12
R981	1274	Manoj	2020-12-09
R402	4072	NULL	2021-07-07
R505	3457	Morjina	2020-01-23

a. From the tables given above, identify the following:

(i) Primary key from both tables

(ii) Foreign key in `Trains` table

(iii) Degree and Cardinality of both tables

b. Can the attribute `TrainID` in the table `Passengers` have the value `9999`? Why/ why not?



Previous Years' Questions

1. Fill in the blank.

_____ is a number of tuples in a relation. [2023]

a. `Attribute`

b. `Degree`

c. `Domain`

d. `Cardinality`

Ans. d. `Cardinality`

2. Explain the concept of "Alternate Key" in a Relational Database Management System with an appropriate example. [2023]

Ans. A candidate key for a table that is not its primary key is called its alternate key.

For example:

Consider the table as follows:

Field	Key
Rollno	Primary Key
Name	–
Email	Alternate Key
Contact_no	Alternate Key

The above table has three candidate keys (Rollno, Email, and Contact_no) as they may have unique values. So these columns are candidate keys. Now, the rollno field is assigned as the primary key, so the left candidate keys become alternate keys, i.e., Email and Contact_no.

3. The school has asked their estate manager Mr. Rahul to maintain the data of all the labs in a table LAB. Rahul has created a table and entered data of 5 labs. [2023]

LABNO	LAB _NAME	INCHARGE	CAPACITY	FLOOR
LO01	CHEMISTRY	Daisy	20	I
LO02	BIOLOGY	Venky	20	II
LO03	MATH	Preeti	15	I
L004	LANGUAGE	Daisy	36	III
LO05	COMPUTER	Mary Kom	37	II

Based on the data given above answer the following questions:

(i) Identify the columns which can be considered as Candidate keys.

(ii) Write the degree and cardinality of the table.

(iii) Write the statements to:

a. Insert a new row with appropriate data.

b. Increase the capacity of all the labs by 10 students which are on 'I' Floor.

OR

(Option for part (iii) only)

(iii) Write the statements to:

a. Add a constraint PRIMARY KEY to the column LABNO in the table.

b. Delete the table LAB.

Ans. (i) LABNO and LABNAME columns can be considered as candidate keys because they can be the primary key of the table. The reason behind this is they have unique records.

(ii) Degree:5, Cardinality:5

(iii) Write Statements to:

```
insert into LAB values('L006','PHYSICS','Kane',40,'I');
update LAB set capacity=capacity+10 where floor='I';
```

OR

a. alter table LAB add primary key (LABNO);

b. Drop table LAB;

4. Differentiate between the terms Attribute and Domain in the context of Relational Data Model. [2022]

Ans. The vertical data set of a relation is known as attribute. It is also known as field or property. It represents one entity. For example customerid and customername are attributes for customer relation. The domain refers to the pool of values from which a field of a table derives its values. Example customerid field derives its values from set of integers, customername derives from the names which are set of characters.

5. Differentiate between Candidate Key and Primary Key in the context Relational Database Model. [2022]

Ans. Candidate Key: The columns which can serve as primary key of a table is known as candidate keys. There can be multiple candidate for a relation.

Primary Key: Primary key is a field name which identifies rows uniquely. There can be only one primary key for a relation.

OR

Consider the following table PLAYER

Table : PLAYER

PNO	NAME	SCORE
P1	RIHSABH	52
P2	HUSSAIN	45
P3	ARNOLD	23
P4	ARNAV	18
P5	GURSHARAN	42

(i) Identify and write the name of the most appropriate column from the given table player that can be used as a Primary Key.

Ans. PNO

(ii) Define the term degree in the relational data model. What is the degree of the given table Palyer?

Ans. Degree refers to the number of attributes or columns of the relation. The degree of table player is: 3

Answers

Multiple Choice Questions

1. (d) 2. (b) 3. (b) 4. (b) 5. (d) 6. (a) 7. (a) 8. (c) 9. (a)
10. (a) 11. (a) 12. (d)

True and False

1. (T) 2. (F) 3. (T) 4. (T) 5. (F) 6. (F) 7. (T)

Fill in the blanks

1. foreign key 2. candidate key 3. information 4. Database Management System (DBMS)
5. Data redundancy 6. entity 7. relations

10 STRUCTURED QUERY LANGUAGE (SQL)

Chapter Outline

10.1 Structured Query Language (SQL)

10.2 Data Types

10.3 SQL Statements and Functions

Introduction

Structured Query Language (SQL)—the standard language for managing relational databases and performing various operations on the data in the tables. Structured Query Language (SQL) would enable us to store, retrieve, and manipulate data in the form of tables.

10.1 Structured Query Language (SQL)

Structured Query Language (SQL) is used to create and manipulate relational databases. It is a high-level interactive language that allows users to specify what is required to be done in the form of queries. Unlike other high-level languages, we do not need to specify how the SQL operations are implemented. Thus, SQL becomes an easy to learn database language, so it is widely used. It efficiently manages large databases, provides fast responses to the queries operating on data and facilitates an authentication mechanism that ensures the security of data.

Based on their functionality, SQL operations are organised in two categories, DDL (Data Definition Language) and DML (Data Manipulation Language). Let us study about them.

SQL: most popular language for creating and manipulating relational databases

10.1.1 Data Definition Language (DDL)

Data Definition Language (DDL) is used for describing database structure. It is used to create and delete databases and tables. The attributes of already existing tables can also be added, deleted or modified using DDL statements, also known as DDL commands. Some of the examples of DDL statements are **`CREATE TABLE`**, **`ALTER TABLE`**, and **`DROP TABLE`**. In this chapter, we will discuss these in detail.

DDL: used for defining database attributes, their types, and constraints on attribute values

10.1.2 Data Manipulation Language (DML)

Data Manipulation Language (DML) is used to retrieve and modify data. It includes statements that enable us to store and retrieve information from database tables and modify the already stored data in the database tables. SQL statements such as **`INSERT`**, **`SELECT`**, **`DELETE`**, and **`UPDATE`** are used for this purpose.



SQL also facilitates the creation of views containing the results produced by an SQL statement. Such saved results can be used later.

10.2 Data Types

Attributes of a table describe the data that will be stored in the table. The data type indicates the domain of values for an attribute. A data type may be of one of the several permissible types. Let us learn about some important data types.

10.2.1 Numeric

The numeric data types may be used to specify that an attribute can contain numeric data. SQL provides **`INT`** or **`INTEGER`**, **`SMALLINT`**, **`DECIMAL(L, D)`**, and **`FLOAT(p)`** numeric data types. Let us discuss about them.

INT or INTEGER

The data type **`INT`** or **`INTEGER`** is used for specifying 32-bit integers. The value that can be stored in the **`INTEGER`** data type ranges from **`-2147483648`** to **`2147483647`**. For example, the attribute `Salary` of an employee may be of type **`INTEGER`**. Some valid and invalid examples of type **`INT`** are as follows:

Valid

`0`

`-0`

`1234`

`1234567890`

`-1234567890`

`123.45` (digits to the right of the decimal point are truncated)

Invalid

`1,23,456` (punctuation characters not allowed)

`29-01-22` (punctuation characters not allowed)

`-2147483649` (beyond valid range)

`2147483648` (beyond valid range)

`98765432100.1234` (beyond valid range)



`INT or INTEGER:` 32-bit integer

SMALLINT

The data type **`SMALLINT`** is used to store small integers (typically 16-bit integers). The value that can be stored in the **`SMALLINT`** data type ranges from `-32768` to `32767`.

For example, the attribute **Dept_No** of the department may be of type **SMALLINT**. Some valid and invalid examples of type **SMALLINT** are as follows:

Valid

0
-0
1234
-32768
32767
123.45 (digits to the right of the decimal point are truncated)

Invalid

1,23,456 (punctuation characters not allowed)
29-01-22 (punctuation characters not allowed)
-32769 (beyond valid range)
32768 (beyond valid range)
987654321.1234 (beyond valid range)



SMALLINT: typically 16-bit integer

DECIMAL(L, D)

The data type **DECIMAL(L, D)** is used to denote a fixed-point decimal value. The arguments **L** and **D** denote the maximum number of digits and number of digits to the right of the decimal point, respectively. For example, attribute type **DECIMAL(6,2)** indicates that the attribute values may lie in the interval [-9999.99, 9999.99]. When the value of an attribute of **DECIMAL** type lies outside its limits, the effect is system defined. For example, for the attribute type **DECIMAL(6,2),** the value 1234.56789 may be rounded to 1234.57 or truncated to 1234.56. If an attribute of **DECIMAL** type is assigned a value whose integer part has more digits than the permitted, the system may assign the maximum (or minimum) allowed value. For example, given the **DECIMAL(6,2)** specification, the number -12345 may be stored as -9999.99. However, MySQL will raise an error and reject the out-of-range values if strict SQL mode is enabled. The value that can be stored in the **DECIMAL** data type ranges from -3.402823466E+38 to -1.175494351E-38. Some valid and invalid examples of type **DECIMAL(6,2)** are:

Valid

1234
1234.56
1234.56789 (rounded to 1234.57)
-1234.56789 (rounded to –1234.57)

Invalid

		Possible Value
12345	(more than four digits in the integer part)	9999.99
12345.12	(more than four digits in the integer part)	9999.99
12345678.1	(more than four digits in the integer part)	9999.99
-12345.1234	(more than four digits in the integer part)	-9999.99
-12345678.12	(more than four digits in the integer part)	-9999.99



DECIMAL(L, D) : Fixed point decimal, with total **L** digits and **D** digits after decimal.

FLOAT (p)

The data type **FLOAT (p)** denotes the floating-point numbers. The argument **p** denotes precision. If the value of **p** is in the range 0 to 23, it denotes a single-precision (4-byte) floating-point value. If the value of **p** is in the range 24 to 53, it denotes a double-precision (8-byte) floating-point value. Some of the valid examples of type **FLOAT (8)** are mentioned below:

```
1234
-123456
1234.56
1234.56789              (rounded to 1234.57)
-1234.56789             (rounded to -1234.57)
```

FLOAT (p) : floating-point number

10.2.2 Character

To store string type data, SQL provides **CHAR (L)** or **CHARACTER (L)** and **VARCHAR (L)** data types. Let us learn about them.

CHAR(L) or CHARACTER(L)

The data type **CHAR (L)** denotes fixed-length character strings. The argument **L** denotes the length of the string and the unused spaces are filled with blanks. For example, if we declare **Emp_Name** of an employee to be of type **CHAR (15)**, then the name string **'SMITA KALA'** will be expanded to **'SMITA KALA'**. If the length of a string of type **CHAR** is not specified, the default length is 1. If the length of a string is more than the specified length, the trailing characters will be truncated. The CHAR data type can store any value from 0 to 255. Next, we give some valid and invalid examples of type **CHAR (10)** :

Valid Value	CHAR(10)	No of bytes required
'IIIT Delhi'	'IIIT Delhi'	10
'IIT Delhi'	'IIT Delhi '	10
'IIT'	'IIT '	10
'110016'	'110016 '	10
'29-01-2022'	'29-01-2022'	10
'IIT Kharagpur'	'IIT Kharag'	10

Invalid	
110016	not a string
29-01-2022	not a string

CHAR (L) : fixed-length character strings of length **L**

VARCHAR (L)

The data type **VARCHAR (L)** is used for specifying variable-length character strings where **L** specifies the maximum length. Unlike **CHAR**, **VARCHAR** must include a positive integer specifying the maximum allowable length of the string as it does not use a default value. For example, declaring **Emp_Name** to be of type **VARCHAR (25)** will use one byte more than the length of the string being stored. Thus, the string **'IIT'** will require four bytes of memory.

A string longer than the defined maximum length will result in the truncation of the extra trailing characters. If any truncated characters are not blank, SQL will raise an error. The VARCHAR data type can store any value from 0 to 65,535. Some valid and invalid examples of type **VARCHAR(10)** are:

Valid

Value	**VARCHAR(10)** string	**No of bytes required**
'IIIT Delhi'	'IIIT Delhi'	11
'IIT Delhi'	'IIT Delhi'	10
'IIT'	'IIT'	4
'110016'	'110016'	7
'29-01-2022'	'29-01-2022'	11
'IIT Kharagpur'	'IIT Kharag'	11

Invalid

110016	not a string
29-01-2022	not a string



VARCHAR(L): variable-length character strings of length upto **L.**

10.2.3 DATE

The data type **DATE** denotes a date. The standard date format is **YYYY-MM-DD**. However, SQL also permits other formats like **DD_MM_YYYY** for storing and retrieving dates.



The standard date format is **YYYY-MM-DD.**



Time: The data type **TIME** denotes time. It supports various time formats such as **HH:MM: SS**, **HH-MM-SS**, **HH.MM AM/PM**.

10.3 SQL Statements and Functions

This section will discuss several SQL statements and functions for defining and manipulating data. Each SQL statement terminates with a semicolon.

10.3.1 CREATE DATABASE Statement

Creating a database is the first step in database management. **CREATE DATABASE** statement creates a database. The syntax for the **CREATE DATABASE** statement is as follows:

CREATE DATABASE database_name;

In the above description of **CREATE DATABASE** statement, **CREATE** is an SQL keyword and **database_name** is the name of the database to be specified by the user.



While describing the SQL statements, we shall write the SQL keywords in **courier** font uppercase letters in boldface. We will write the user-defined names and user inputs in **courier** font but not boldface.

Now, we are ready to create the **COMPANY** database (described in previous chapter) using the **CREATE DATABASE** statement:

```
CREATE DATABASE COMPANY;
```

To display the list of databases that have been created so far, we use the **SHOW DATABASES** statement:

```
SHOW DATABASES;
```

Before we perform any operations on a database, it has to be opened for use. SQL statement **USE <database_name>** serves this purpose:

```
USE database_name;
```



SHOW DATABASES: displays names of all the existing databases

To select the database **COMPANY** for creating and manipulating tables in the **COMPANY** database, we execute the statement:

```
USE COMPANY;
```



***USE* database_name:** makes a database active for use

10.3.2 CREATE TABLE Statement

Next, let us create tables corresponding to different entities (EMPLOYEE, DEPARTMENT, PROJECT, and WORKS_ON). **CREATE TABLE** statement creates tables (relations). The syntax for the **CREATE TABLE** statement is:

```
CREATE TABLE table_name(
attribute1 data_type [constraint]
[, attribute2 data_type [constraint]]
⋮
[, attributeN data_type [constraint]]
table_constraints]
);
```

In the description of an SQL statement, a pair of square brackets (**[]**) denotes optional parts. Thus, a **CREATE TABLE** statement may include any number of attributes or constraints. Note that the attributes and constraints are specified within parenthesis separated by commas.

Next, we give an example of a statement that creates an EMPLOYEE table with the following attributes: ID, FName, LName, Gender, Address, City, Pin_Code, DOB, Salary and DeptNo. Also note that attribute ID is the primary key of the table **EMPLOYEE.** When the primary key comprises a single attribute, often we use the keyword **PRIMARY KEY**, following the type of the attribute to denote that it is the primary key of the table as shown below in the **CREATE TABLE** statement:

```
CREATE TABLE EMPLOYEE (
ID INT PRIMARY KEY,
FName VARCHAR(20) NOT NULL,
LName VARCHAR(20) NOT NULL,
Gender CHAR(1) NOT NULL,
Address VARCHAR(50),
```

```
City VARCHAR(20),
Pin_Code CHAR(6),
DOB DATE,
Salary INT NOT NULL,
Dept_No SMALLINT
); #EMPLOYEE
```

The primary key can also be specified after all attributes have been defined, as shown in the **CREATE TABLE** statement below:

```
CREATE TABLE EMPLOYEE (
ID INT NOT NULL UNIQUE,
FName VARCHAR(20) NOT NULL,
LName VARCHAR(20) NOT NULL,
Gender CHAR(1) NOT NULL,
Address VARCHAR(50),
City VARCHAR(20),
Pin_Code CHAR(6),
DOB DATE,
Salary INT NOT NULL,
Dept_No SMALLINT,
PRIMARY KEY(ID)
); #EMPLOYEE
```

As **ID** is the primary key for the table EMPLOYEE, no two employees can have the same **ID**, i.e., **ID** must be unique across the EMPLOYEE table. Further, as **ID** identifies an employee uniquely, this attribute cannot be **NULL**. So, column constraints **UNIQUE** and **NOT NULL** are automatically applied to the attribute **ID**.



CREATE TABLE: used to specify
Table name
Attribute names, default values
Attribute types
Primary key
Constraints: **NOT NULL**, foreign key, values allowed

A table may have more than one attribute as the primary key, collectively known as composite key. In such case, the primary key is defined after the attributes of the table have been defined, as shown in the example below:

```
CREATE TABLE WORKS_ON (
Proj_No SMALLINT NOT NULL,
Emp_Id INT NOT NULL,
Hours SMALLINT,
PRIMARY KEY (Proj_No, Emp_Id)
); #WORKS_ON
```

The statement given above creates the WORKS_ON table that has a composite primary key consisting of attributes, Proj_No and Emp_Id.

While writing SQL statements, the symbol # marks the beginning of a comment that extends to the end of the line. A C-style comment is enclosed between /* and */. A comment may also begin with--(double hyphen), followed by a whitespace or control character such as space, tab, or end of line (EOL), and extends to the end of the line.

The primary key can be defined as a constraint as shown below:

```
CONSTRAINT EMPLOYEE_Key PRIMARY KEY(ID)
```

Syntax for the constraint statement is given below:

```
CONSTRAINT ConstraintName PRIMARY KEY(attribute_1, [attribute_2],...)
```

Note that the primary key constraint includes the name of at least one attribute included in the primary key. In general, several attributes may be included in the primary key constraint.

```
CREATE TABLE EMPLOYEE (
ID INT NOT NULL UNIQUE,
FName VARCHAR(20) NOT NULL,
LName VARCHAR(20) NOT NULL,
Gender CHAR(1) NOT NULL,
Address VARCHAR(50),
City VARCHAR(20),
Pin_Code CHAR(6),
DOB DATE,
Salary INT NOT NULL,
Dept_No SMALLINT,
CONSTRAINT EMPLOYEE_Key PRIMARY KEY(ID)
); #EMPLOYEE
```

Note that the `EMPLOYEE_Key` is the name of the primary key constraint on `ID` attribute. A named constraint may be modified or dropped later by referring to its name.

Next, we discuss some more SQL constraints.

10.3.3 DEFAULT Constraint

A default value for an attribute may be specified using the **DEFAULT** constraint. For example, if most of the employees have a salary of 10000, it would be wise to set it as the default salary for an employee as follows:

```
Salary INT NOT NULL DEFAULT 10000
```

The revised **CREATE TABLE** statement for the table `EMPLOYEE` would appear as follows:

```
CREATE TABLE EMPLOYEE (
ID INT PRIMARY KEY,
FName VARCHAR(20) NOT NULL,
LName VARCHAR(20) NOT NULL,
Gender CHAR(1) NOT NULL,
Address VARCHAR(50),
City VARCHAR(20),
Pin_Code CHAR(6),
DOB DATE,
Salary INT NOT NULL DEFAULT 10000,
Dept_No SMALLINT
); #EMPLOYEE
```

While inserting a tuple into the table `EMPLOYEE`, if the user does not provide any value for the attribute **Salary**, then it will be set as `10000`. The default value of the character type must be enclosed within single quotes or double-quotes.

10.3.4 CHECK Constraint

The **CHECK** constraint restricts the values that an attribute can take. For example, it would be reasonable to limit the values of **Gender** to 'M', 'F', and 'O', indicating male, female, and other. A **CHECK** constraint to achieve this may be included as part of the statement specified as follows:

```
CHECK(Gender = 'M' OR Gender = 'F' OR Gender = 'O')
```

The revised **CREATE TABLE** statement for the table EMPLOYEE would appear as follows:

```
CREATE TABLE EMPLOYEE (
ID INT PRIMARY KEY,
FName VARCHAR(20) NOT NULL,
LName VARCHAR(20) NOT NULL,
Gender CHAR(1) NOT NULL
CHECK(Gender = 'M' OR Gender = 'F' OR Gender = 'O'),
Address VARCHAR(50),
City VARCHAR(20),
Pin_Code CHAR(6),
DOB DATE,
Salary INT NOT NULL DEFAULT 10000,
Dept_No SMALLINT
); #EMPLOYEE
```

The **CHECK** constraint may also be used to specify that an attribute would take one of the several specified values or a numeric value in a specified range. For example, the constraint to perform a check on valid values of **Gender** may also be written as follows:

```
CHECK(Gender IN ('M', 'F', 'O'))
```

As another example, Salary may be restricted to lie between 8000 and 100000 as follows:

```
CHECK(Salary BETWEEN 8000 AND 100000)
```

Alternatively, **CHECK** constraints may be may be described separately as a table-level constraints, after all, attributes are described, for example:

```
CREATE TABLE EMPLOYEE (
ID INT PRIMARY KEY,
FName VARCHAR(20) NOT NULL,
LName VARCHAR(20) NOT NULL,
Gender CHAR(1) NOT NULL,
Address VARCHAR(50),
City VARCHAR(20),
Pin_Code CHAR(6),
DOB DATE,
Salary INT NOT NULL,
Dept_No SMALLINT,
   CHECK(Gender = 'M' OR Gender = 'F' OR Gender='O'),
CHECK(Salary BETWEEN 8000 AND 100000)
);#EMPLOYEE
```

10.3.5 DESCRIBE TABLE Statement

The SQL statement **DESCRIBE** (or **DESC** in short) is used to show the structure of a database table. The syntax of the **DESCRIBE** statement is as follows:

```
DESCRIBE table_name;
```

or

```
DESC table_name;
```

On executing the statement:

```
DESCRIBE EMPLOYEE;
```

Execution of the above statement will produce the output shown in Table 10.1:

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	
FName	varchar(20)	NO		NULL	
LName	varchar(20)	NO		NULL	
Gender	char(1)	NO		NULL	
Address	varchar(50)	YES		NULL	
City	varchar(20)	YES		NULL	
Pin_Code	char(6)	YES		NULL	
DOB	date	YES		NULL	
Salary	int	NO		NULL	
Dept_No	smallint	YES		NULL	

Table 10.1: Structure of EMPLOYEE table

10.3.6 SHOW TABLES Statement

Having created all the tables, we can use the following SQL statement to show the list of tables created in the database **COMPANY** as follows:

```
SHOW TABLES;
```

On executing the above statement, SQL will display the list of tables, as shown in Table 10.2:

Tables_in_company
department
employee
project
works_on

Table 10.2: Result of SHOW TABLES statement

10.3.7 INSERT Statement

Once the database tables are created, we can store the data in the database. Storing the available data in the database is also called loading the database or populating the database. The SQL statement **INSERT** is used for this purpose. The syntax for a simple **INSERT** statement is as follows:

```
INSERT INTO table_name
VALUES (value1, value2, …, valueN);
```

INSERT statement allows us to insert a tuple by specifying table name and comma-separated attribute values within parenthesis. The values of type **CHAR, VARCHAR,** and **DATE** are enclosed within single-quotes or double-quotes. The order of values in the **INSERT** statement should correspond to the order of attributes. If the value of an attribute is not known, we use the **NULL** value. However, values of the primary key attributes cannot be **NULL.**

INSERT: to insert a row in a table

Now let us insert into the **EMPLOYEE** table, a tuple for **Raj Reddy** whose **ID** is **10001** and lives in **Andhra Pradesh** as follows:

```
INSERT INTO EMPLOYEE
VALUES (10001,'Raj','Reddy','M','WestGodavari','AndhraPradsh',534197,'1980-06-13',
100000,2);
```

Note that in the above example, the date has been provided in YYYY–MM–DD format. If you prefer, you can enter date in MM–DD–YYYY format as follows:

```
INSERT INTO EMPLOYEE
VALUES (10001,'Raj','Reddy','M','West Godavari','Andhra Pradesh', 534197,
STR_TO_DATE("June 13 1980", "%M %d %Y");
```

Alternatively, SQL allows us to specify the attributes whose values are known by enumerating them in a comma-separated list. In the SQL statement below, we have only listed five attributes (**Salary, LName, FName, Gender, ID**) of the **EMPLOYEE** table while inserting a tuple. Moreover, as the attributes are named, we can list them in any order. Note that there should be one to one correspondence between the attribute names and their values.

```
INSERT INTO EMPLOYEE(Salary, LName, FName, Gender, ID)
VALUES (100000, 'Reddy', 'Raj', 'M', 10001);
```

General syntax for the above statement is given below:

```
INSERT INTO table_name (Attribute1, Attribute2, ..., AttributeN)
VALUES (value1, value2, ..., valueN);
```

Next, suppose the values of attributes Address, City, Pin_Code, DOB and Dept_No are not known for Raj Reddy whose ID is 10001. The following **INSERT** statement inserts such a tuple into the table EMPLOYEE:

```
INSERT INTO EMPLOYEE
VALUES (10001, 'Raj', 'Reddy', 'M', NULL, NULL, NULL, NULL, 100000, NULL);
```

Write SQL statements to insert more rows into EMPLOYEE table so that it appears, as shown in Table 10.3:

Table 10.3 Showing the EMPLOYEE table

ID	FName	LName	Gender	Address	City	Pin_Code	DOB	Salary	Dept_No
10001	Raj	Reddy	M	West Godavari	Andhra Pradesh	534197	1980-06-13	100000	2
10002	Dhiraj	Bora	M	Dispur, Kamrup, Assam	Guwahati	781005	1975-09-30	85000	1
10003	Muskan	Taneja	F	8/33, Geeta Colony	Delhi	110031	1990-01-25	100000	2
10004	Hiten	Oberoi	M	15, Dimna Road, Mango	Jamshedpur	831018	1985-06-24	100000	4
10005	Anshul	Verma	M	House 10, Sector 16, Gautum Budh Nagar	Noida	201304	1990-01-01	100000	1
10006	Rajit	Gadh	F	12, Beldih Triangle, Bistupur	Jamshedpur	831001	1960-05-07	60000	4
10007	Taran	Adarsh	M	B-76, CST Road, Kalina, Santacruz East	Mumbai	400098	1965-01-13	70000	5
10008	Naval	Dhingra	M	E-14 Vivek Vihar	Delhi	110095	1975-08-04	70000	2
10009	Naveen	Basra	F	28, Aambagan Road, Sakechi	Jamshedpur	831001	1980-09-24	60000	4
10010	Savita	Ambedkar	F	C-49, G-Block, Bandra Kurla, Bandra East	Mumbai	400051	1987-07-11	50000	5

Table 10.3: EMPLOYEE table after inserting some tuples

In Table 10.3, we have shown the tuples in the order in which they were inserted in the table. However, a relational DBMS does not ensure any specific order of the tuples. Placing the tuples in a particular sequence does not impact the database operations.

10.3.8 UPDATE-SET Statement

Suppose the salary of an employee having ID value 10003 has been revised recently. SQL provides an **UPDATE** statement for making changes in the tuples already stored in a table. The syntax of the update statement is as follows:

```
UPDATE table_name
SET Attribute_name = value
[WHERE criterion_for_selecting_specific_tuple];
```

The following **UPDATE** statement finds a tuple in the EMPLOYEE table having an **ID** equal to **10003** and sets the value of its Salary attribute to 90000.

```
UPDATE EMPLOYEE
SET Salary = 90000
WHERE ID = 10003;
```

On executing the above statement, SQL will update the EMPLOYEE tuple for the employee whose ID is 10003, as shown in Table 10.4:

ID	FName	LName	Gender	Address	City	Pin_Code	DOB	Salary	Dept_No
10003	Muskan	Taneja	F	8/33, Geeta Colony	Delhi	110031	1990-01-25	90000	2

Table 10.4: Result of UPDATE statement on EMPLOYEE table

Next, suppose we want to assign a manager (ID = 10002) to the department having Dept_No 1. The following **UPDATE** statement finds a tuple in the DEPARTMENT table having Dept_No 1 and sets the value of its Mgr_Id attribute to 10002.

```
UPDATE DEPARTMENT
SET Mgr_Id = 10002
WHERE Dept_No = 1;
```

On execution of the above SQL statement, DEPARTMENT table appears, as shown in Table 10.5:

Dept_No	Dept_Name	Location	Mgr_Id
1	Accounts	Noida	10002
2	Administration	Delhi	10005
3	Home Goods	Mumbai	10003
4	Automobile	Jamshedpur	10007
5	Textile	Mumbai	10004

Table 10.5: Result of UPDATE statement on DEPARTMENT table

10.3.9 DELETE Statement

To delete one or more tuples (satisfying a particular criterion) from a table, we use the **DELETE** statement. The syntax of the delete statement is as follows:

```
DELETE FROM table_name
[WHERE criterion-for-selecting-specific-tuple];
```

For example, to delete the record of the employee from the **EMPLOYEE** table whose **ID** is **10007**, we execute the following SQL statement:

```
DELETE FROM EMPLOYEE
WHERE ID = 10007;
```

DELETE: to delete rows from a table

Let us now execute the statement that shows the current state of the table from which the tuple corresponding to the employee having `ID 10007` has been deleted, as shown in Table 10.6:

ID	FName	LName	Gender	Address	City	Pin_Code	DOB	Salary	Dept_No
10001	Raj	Reddy	M	West Godavari	Andhra Pradesh	534197	1980-06-13	100000	2
10002	Dhiraj	Bora	M	Dispur, Kamrup, Assam	Guwahati	781005	1975-09-30	85000	1
10003	Muskan	Taneja	F	8/33, Geeta Colony	Delhi	110031	1990-01-25	100000	2
10004	Hiten	Oberoi	M	15, Dimna Road, Mango	Jamshedpur	831018	1985-06-24	100000	4
10005	Anshul	Verma	M	House 10, Sector 16, Gautum Budh Nagar	Noida	201304	1990-01-01	100000	1
10006	Rajit	Gadh	F	12, Beldih Triangle, Bistupur	Jamshedpur	831001	1960-05-07	60000	4
10008	Naval	Dhingra	M	E-14 Vivek Vihar	Delhi	110095	1975-08-04	70000	2
10009	Naveen	Basra	F	28, Aambagan Road, Sakechi	Jamshedpur	831001	1980-09-24	60000	4
10010	Savita	Ambedkar	F	C-49, G-Block, Bandra Kurla, Bandra East	Mumbai	400051	1987-07-11	50000	5

Table 10.6: Result of `DELETE` statement on `EMPLOYEE` table

The execution of the following statement will delete all rows of the table **EMPLOYEE**:

```
DELETE FROM EMPLOYEE;
```

However, the table `EMPLOYEE` remains part of the database and can be populated again.

10.3.10 DROP TABLE Statement

The **DROP TABLE** statement deletes an existing table in a database. The syntax of the **DROP TABLE** statement is as follows:

```
DROP TABLE table-name;
```

For example, to drop the table **EMPLOYEE** from the database **COMPANY**, we can use the following SQL statement:

```
DROP TABLE EMPLOYEE;
```



DROP TABLE: to remove a table and its data

10.3.11 ALTER TABLE Statement

ALTER TABLE statement allows us to add, change, and delete attributes of a table. It can also be used to create and remove SQL constraints from tables.



ALTER TABLE: to add, modify, and delete attributes in a table

Add Attributes

To add attributes to an existing table; we use the **ALTER TABLE** statement with the **ADD** keyword. The syntax of the **ALTER TABLE-ADD** statement is as follows:

```
ALTER TABLE table_name ADD attribute data_type [constraint];
```

For example, to add a new attribute **Phone** in the **EMPLOYEE** table, the SQL statement is as follows:

```
ALTER TABLE EMPLOYEE
ADD Phone DECIMAL(10,0);
```

On executing the above statement, the tuples that already exists in the EMPLOYEE table will be assigned **NULL** value for Phone attribute, as shown in Table 10.7:

ID	FName	LName	Gender	Address	City	Pin_Code	DOB	Salary	Dept_No	Phone
10001	Raj	Reddy	M	West Godavari	Andhra Pradesh	534197	1980-06-13	100000	2	NULL
10002	Dhiraj	Bora	M	Dispur, Kamrup, Assam	Guwahati	781005	1975-09-30	85000	1	NULL
10003	Muskan	Taneja	F	8/33, Geeta Colony	Delhi	110031	1990-01-25	100000	2	NULL
10004	Hiten	Oberoi	M	15, Dimna Road, Mango	Jamshedpur	831018	1985-06-24	100000	4	
10005	Anshul	Verma	M	House 10, Sector 16, Gautum Budh Nagar	Noida	201304	1990-01-01	100000	1	NULL
10006	Rajit	Gadh	F	12, Beldih Triangle, Bistupur	Jamshedpur	831001	1960-05-07	60000	4	NULL
10007	Taran	Adarsh	M	B-76, CST Road, Kalina, Santacruz East	Mumbai	400098	1965-01-13	70000	5	NULL
10008	Naval	Dhingra	M	E-14 Vivek Vihar	Delhi	110095	1975-08-04	70000	2	NULL
10009	Naveen	Basra	F	28, Aambagan Road, Sakechi	Jamshedpur	831001	1980-09-24	60000	4	NULL
10010	Savita	Ambedkar	F	C-49, G-Block, Bandra Kurla, Bandra East	Mumbai	400051	1987-07-11	50000	5	NULL

Table 10.7: Phone attribute is added to the table EMPLOYEE

To add more than one attribute to a table, we can execute several **ALTER TABLE** statements. Alternatively, we can include several attributes in a single **ALTER TABLE** statement using the following syntax:

```
ALTER TABLE table_name
ADD (attribute1 data_type [constraint],
attribute2 data_type [constraint],
...
attributeN data_type [constraint]);
```

For example, execution of the following SQL statement will add two attributes Mobile and Email to the EMPLOYEE table:

```
ALTER TABLE EMPLOYEE ADD (
Mobile DECIMAL(10,0) DEFAULT '9999999999',
Email VARCHAR(50) NOT NULL);
```

We can examine the change in the EMPLOYEE table on the execution of above SQL statement by executing the following **DESC** statement:

```
DESC EMPLOYEE;
```

Execution of the above statement will produce the output as shown in Table 10.8:

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	
FName	varchar(20)	NO		NULL	
LName	varchar(20)	NO		NULL	
Gender	char(1)	NO		NULL	
Address	varchar(50)	YES		NULL	
City	varchar(20)	YES		NULL	
Pin_Code	char(6)	YES		NULL	
DOB	date	YES		NULL	
Salary	int	NO		NULL	
Dept_No	smallint	YES		NULL	
Phone	decimal(10,0)	YES		NULL	
Mobile	decimal(10,0)	YES		9999999999	
Email	varchar(50)	NO		NULL	

Table 10.8: Structure of EMPLOYEE table after adding Mobile and Email Attributes

Modify Attributes

To modify the definition of attributes of an existing table; we use the **ALTER TABLE** statement with the **MODIFY** keyword. The syntax of the **ALTER TABLE-MODIFY** statement is as follows:

```
ALTER TABLE table_name MODIFY attribute data_type [constraint];
```

For example, to modify the definition of an existing attribute **Dept_No** in the **EMPLOYEE** table, we can execute the following SQL statement:

```
ALTER TABLE EMPLOYEE MODIFY Dept_No INT;
```

We can examine the change in the EMPLOYEE table on the execution of above SQL statement by executing a **DESC** statement as follows:

```
DESC EMPLOYEE;
```

On executing the above statement, we get the result as shown in Table 10.9:

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	
FName	varchar(20)	NO		NULL	
LName	varchar(20)	NO		NULL	
Gender	char(1)	NO		NULL	
Address	varchar(50)	YES		NULL	
City	varchar(20)	YES		NULL	
Pin_Code	char(6)	YES		NULL	
DOB	date	YES		NULL	
Salary	int	NO		NULL	
Dept_No	int	YES		NULL	
Phone	decimal(10,0)	YES		NULL	
Mobile	decimal(10,0)	YES		9999999999	
Email	varchar(50)	NO		NULL	

Table 10.9: Structure of EMPLOYEE table after modifying Dept_No

Drop Attributes

To drop an attribute from a table, we use the **ALTER TABLE** statement with the **DROP** keyword. The syntax of the **ALTER TABLE-DROP** statement is as follows:

```
ALTER TABLE table_name DROP attribute;
```

For example, to drop the attributes Phone, Mobile, and Email from the EMPLOYEE table, we can execute the following SQL statements:

```
ALTER TABLE EMPLOYEE DROP Phone;
ALTER TABLE EMPLOYEE DROP Mobile;
ALTER TABLE EMPLOYEE DROP Email;
```

As before, we describe the modified table EMPLOYEE as follows:

```
DESC EMPLOYEE;
```

On executing the above statement, we get the result as shown in Table 10.10:

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	
FName	varchar(20)	NO		NULL	
LName	varchar(20)	NO		NULL	
Gender	char(1)	NO		NULL	
Address	varchar(50)	YES		NULL	
City	varchar(20)	YES		NULL	
Pin_Code	char(6)	YES		NULL	
DOB	date	YES		NULL	
Salary	int	NO		NULL	
Dept_No	int	YES		NULL	

Table 10.10: Structure of EMPLOYEE table after dropping the Phone, Mobile, and Email attributes

Add and Remove Primary Key Constraint

Using an **ALTER TABLE** statement, we can also add and/or remove the primary key constraints. The primary key constraint can be enforced on an attribute using the following **ALTER TABLE** statement:

```
ALTER TABLE EMPLOYEE ADD PRIMARY KEY(ID);
```

To drop the primary key constraint from an attribute, the following **ALTER TABLE** statement can be used:

```
ALTER TABLE EMPLOYEE DROP PRIMARY KEY(ID);
```

Extra Info

Rename Attributes

To rename an attribute of an existing table, we use the **ALTER TABLE** statement with the **RENAME COLUMN** keyword. The syntax of the **ALTER TABLE RENAME COLUMN** statement is as follows:

```
ALTER TABLE table_name RENAME COLUMN old_name TO new_name;
```

For example, to modify the definition of an existing attribute `Pin_Code` in the `EMPLOYEE` table, to `PinCode`, we execute the statement:

```
ALTER TABLE EMPLOYEE RENAME COLUMN Pin_Code TO PinCode;
```

Next, we examine the modified table `EMPLOYEE`, by executing the following statement:

```
DESC EMPLOYEE;
```

On executing the above statement, we get the result as given in Table 10.11:

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	
FName	varchar(20)	NO		NULL	
LName	varchar(20)	NO		NULL	
Gender	char(1)	NO		NULL	
Address	varchar(50)	YES		NULL	
City	varchar(20)	YES		NULL	
PinCode	char(6)	YES		NULL	
DOB	date	YES		NULL	
Salary	int	NO		NULL	
Dept_No	int	YES		NULL	

Table 10.11: Structure of `EMPLOYEE` table after renaming Pin_Code to PinCode

10.3.12 SELECT Statement

The **SELECT** statement is used to retrieve information from database tables. Such statements are also known as SQL queries. Syntax of a simple **SELECT** statement is given below:

```
SELECT attribute_list
FROM table_name;
```

`attribute_list` is a comma-separated list of names of the attributes whose value is to be retrieved. For example, the following SQL statement retrieves department number, name and location for all the departments:

```
SELECT Dept_No, Dept_Name, Location
FROM DEPARTMENT;
```

In the above statement, the first line is the **SELECT** clause that comprises the keyword **SELECT** and a list of attributes `Dept_No, Dept_Name, Location`. The second line is called **FROM** clause. Execution of the above statement will produce the output shown in Table 10.12:

Dept_No	Dept_Name	Location
1	Accounts	Noida
2	Administration	Delhi
3	Home Goods	Mumbai
4	Automobile	Jamshedpur
5	Textile	Mumbai

Table 10.12: `Dept_No, Dept_Name, Location` from the `DEPARTMENT` table

Similarly, the following SQL statement yields information about all employees working in the company from the **EMPLOYEE** table:

```
SELECT ID, FName, LName, Gender, Address, City, Pin_Code, DOB, Salary, Dept_No
FROM EMPLOYEE;
```

When we need to display all attributes in a table, we use the wildcard character ***** (asterisk) as a placeholder for all the attributes of the table specified in the **FROM** clause, as shown below:

```
SELECT *
FROM EMPLOYEE;
```

Executing the above statement will produce output shown in Table 10.13.

Table: EMPLOYEE

ID	FName	LName	Gender	Address	City	Pin_Code	DOB	Salary	Dept_No
10001	Raj	Reddy	M	West Godavari	Andhra Pradesh	534197	1980-06-13	100000	2
10002	Dhiraj	Bora	M	Dispur, Kamrup, Assam	Guwahati	781005	1975-09-30	85000	1
10003	Muskan	Taneja	F	8/33, Geeta Colony	Delhi	110031	1990-01-25	100000	2
10004	Hiten	Oberoi	M	15, Dimna Road, Mango	Jamshedpur	831018	1985-06-24	100000	4
10005	Anshul	Verma	M	House 10, Sector 16, Gautum Budh Nagar	Noida	201304	1990-01-01	100000	1
10006	Rajit	Gadh	F	12, Beldih Triangle, Bistupur	Jamshedpur	831001	1960-05-07	60000	4
10007	Taran	Adarsh	M	B-76, CST Road, Kalina, Santacruz East	Mumbai	400098	1965-01-13	70000	5
10008	Naval	Dhingra	M	E-14 Vivek Vihar	Delhi	110095	1975-08-04	70000	2
10009	Naveen	Basra	F	28, Aambagan Road, Sakechi	Jamshedpur	831001	1980-09-24	60000	4
10010	Savita	Ambedkar	F	C-49, G-Block, Bandra Kurla, Bandra East	Mumbai	400051	1987-07-11	50000	5

Table 10.13: All details of employees in EMPLOYEE table

SQL provides a **WHERE** clause that facilitates retrieval of only those tuples that satisfy the condition specified in the **WHERE** clause. Syntax of a **SELECT** statement with **WHERE** clause is given below:

```
SELECT attribute_list
FROM table_name
WHERE condition;
```

Here, **condition** may be a simple condition or may involve several simple conditions and use of logical operators: **AND**, **OR**, and **NOT**. For example, the following SQL statement retrieves the names of all male employees:



SELECT: retrieves the attribute values from the table

```
SELECT FName, LName
FROM EMPLOYEE
WHERE Gender = 'M';
```

Executing the above statement will produce output shown in Table 10.14:

FName	LName
Raj	Reddy
Dhiraj	Bora
Hiten	Oberoi
Anshul	Verma
Taran	Adarsh
Naval	Dhingra

Table 10.14: Names of male employees

Similarly, we can use any of the relational and logical operators to specify conditions in the `WHERE` clause. Consider the following examples:

Query: To display first name and the city of employees whose salary is greater than 70000 and who do not live in Delhi.

Solution:

```
SELECT FName,City FROM EMPLOYEE
WHERE City != 'Delhi' AND Salary > 70000;
```

Executing the above statement will produce output shown in Table 10.15:

FName	City
Raj	Andhra Pradesh
Dhiraj	Guwahati
Hiten	Jamshedpur
Anshul	Noida

Table 10.15: First name and the city of employees whose salary is greater than 70000 and who do not live in Delhi

Query: To display the records of employees who either live in Delhi or in Noida.

```
SELECT *
FROM EMPLOYEE
WHERE City = 'Delhi' OR City = 'Noida';
```

On executing the above statement, SQL will produce output shown in Table 10.16:

ID	FName	LName	Gender	Address	City	Pin_Code	DOB	Salary	Dept_No
10003	Muskan	Taneja	F	8/33, Geeta Colony	Delhi	110031	1990-01-25	100000	2
10005	Anshul	Verma	M	House 10, Sector 16, Gautum Budh Nagar	Noida	201304	1990-01-01	100000	1
10008	Naval	Dhingra	M	E-14 Vivek Vihar	Delhi	110095	1975-08-04	70000	2

Table 10.16: Details of employees who either live in Delhi or in Noida

Query: To display the department numbers of all employees.

```
SELECT Dept_No
FROM EMPLOYEE;
```

On executing the above statement, SQL will produce output shown in Table 10.17:

Dept_No
2
1
2
4
1
4
5
2
4
5

Table 10.17: Department numbers of all employees

10.3.13 DISTINCT Clause

We see in Table 10.17 that the execution of the above queries yields the department numbers, with multiple values. Table 10.17 will look more elegant if we remove the repeating tuples. For this purpose, the keyword **DISTINCT** is used in the **SELECT** clause to retrieve unique values of an attribute. The resulting SQL statement would be:

```
SELECT DISTINCT Dept_No
FROM EMPLOYEE;
```

Execution of the above **SELECT** statement will produce the output shown in Table 10.18:

Dept_No
2
1
4
5

Table 10.18: Unique department numbers from EMPLOYEE table

10.3.14 ORDER BY Clause

The SQL **ORDER BY** clause sorts data on one or more columns in ascending or descending order. By default, SQL sorts the data in ascending order of the specified attribute.

To display ID and the first name of all employees arranged alphabetically according to their first names, we use the statement:

```
SELECT ID, FName
FROM EMPLOYEE
ORDER BY FName;
```

On executing the above statement, SQL will produce output shown in Table 10.19. Note the **ORDER BY** clause's default setting (ascending order) for the attribute FName:



ORDER BY:
sorts data based on specified attribute(s)
Default order: ascending, or specify **DESC**

ID	FName
10005	Anshul
10002	Dhiraj
10004	Hiten
10003	Muskan
10008	Naval
10009	Naveen
10001	Raj
10006	Rajit
10010	Savita
10007	Taran

Table 10.19: ID, FName for employees in ascending order of FName

To display ID, first name, and the salary of all employees ordered by their salary in descending order, the following query may be used:

```
SELECT ID, FName, Salary
FROM EMPLOYEE
ORDER BY Salary DESC;
```

On executing the above statement, SQL will produce output shown in Table 10.20:

ID	FName	Salary
10001	Raj	100000
10004	Hiten	100000
10005	Anshul	100000
10003	Muskan	100000
10002	Dhiraj	85000
10007	Taran	70000
10008	Naval	70000
10006	Rajit	60000
10009	Naveen	60000
10010	Savita	50000

Table 10.20: ID, FName, Salary in descending order of Salary

We can also display the records arranged according to the values of two or more attributes in a table. For example, to display ID, first name, last name, and the salary of all employees ordered by their salary in descending order. Further, for the two employees having the same salary, the tuples must appear in the ascending order of their last names.

```
SELECT ID, FName, Lname, Salary
FROM EMPLOYEE
ORDER BY Salary DESC, LName;
```

Executing the above statement will produce output shown in Table 10.21. Note that the table appears in descending order of `Salary`. However, due to the **ORDER BY** clause's default setting (ascending order), the employees having the same salary are arranged in ascending order of their last name (`LNAME`).

ID	FName	Lname	Salary
10004	Hiten	Oberoi	100000
10001	Raj	Reddy	100000
10005	Anshul	Verma	100000
10003	Muskan	Taneja	100000
10002	Dhiraj	Bora	85000
10007	Taran	Adarsh	70000
10008	Naval	Dhingra	70000
10009	Naveen	Basra	60000
10006	Rajit	Gadh	60000
10010	Savita	Ambedkar	50000

Table 10.21: `Salary` in descending order, `LName` in ascending order for the employees having the same salary

10.3.15 BETWEEN Operator

The **BETWEEN** operator checks whether a value is within a specified range (inclusive of both lower limit and upper limit values). It may be applied to numbers, text, and dates. Syntax of an SQL query using **BETWEEN** operator is given below:

```
SELECT attribute1, [attribute2, …]
FROM table_name
WHERE attributeJ [NOT] BETWEEN value1 AND value2;
```

The above **WHERE** clause serves as an abbreviation for the following clause:

```
WHERE [NOT](attributeJ >= value1 AND attributeJ <= value2)
```

To display `ID`, `FName`, and `Salary` of those employees whose salary is between 85000 to 100000, both inclusive, we may use the following query:

```
SELECT ID, FName, Salary
FROM EMPLOYEE
WHERE Salary BETWEEN 85000 AND 100000;
```

Table 10.22 displays the output of the given query:

ID	FName	Salary
10001	Raj	100000
10002	Dhiraj	85000
10003	Muskan	100000
10004	Hiten	100000
10005	Anshul	100000

Table 10.22: Employees having salary between 85000 and 100000

Note that Table 10.22 includes salary 85000 and 100000, specified for the **BETWEEN** operator. Thus, **BETWEEN** is an inclusive operator as it includes the *begin* and *end* values of the specified range.

To display the first name, last name, and date of birth of all employees born between the years 1985 to 1990, we can use the following SQL statement:

```
SELECT FName, LName, DOB
FROM EMPLOYEE
WHERE DOB BETWEEN '1985-01-01' AND '1990-12-31';
```

Execution of the above statement will produce the output shown in Table 10.23:

FName	LName	DOB
Muskan	Taneja	1990-01-25
Hiten	Oberoi	1985-06-24
Anshul	Verma	1990-01-01
Savita	Ambedkar	1987-07-11

Table 10.23: Employees whose date of birth is between '1985–01–01' and '1990–12–31'

Note that in the above example, the date appears in **YYYY-MM-DD** format. We can display a date in **DD/MM/YYYY** format as follows:

```
SELECT FName, LName, DATE_FORMAT(DOB,'%d/%m/%Y') AS DOB
FROM EMPLOYEE
WHERE DOB BETWEEN '1985-01-01' AND '1990-12-31';
```

Execution of the above statement will produce the output shown in Table 10.24:

FName	LName	DOB
Muskan	Taneja	25/01/1990
Hiten	Oberoi	24/06/1985
Anshul	Verma	01/01/1990
Savita	Ambedkar	11/07/1987

Table 10.24: Employees whose date of birth is between '1985–01–01' and '1990–12–31'

Query: Display the first name, last name, and date of birth of all employees, except those who were born between the years other than `1985` to `1990`.

To answer the above query, we use **NOT** operator before **BETWEEN** operator as follows:

```
SELECT FName, LName, DOB
FROM EMPLOYEE
WHERE DOB NOT BETWEEN '1985-01-01' AND '1990-12-31';
```

Execution of the above statement will produce the output shown in Table 10.25:

FName	LName	DOB
Raj	Reddy	1980-06-13
Dhiraj	Bora	1975-09-30
Rajit	Gadh	1960-05-07
Taran	Adarsh	1965-01-13
Naval	Dhingra	1975-08-04
Naveen	Basra	1980-09-24

Table 10.25: Employees whose date of birth is outside the interval ['1985-01-01','1990-12-31']

10.3.16 IN Operator

The **IN** operator checks whether a value exists in the given list of values. Thus, the **IN** operator provides an alternative to using multiple **OR** operators. If we wish to exclude specific values, **IN** may be preceded by **NOT**.

An SQL statement involving **IN** operator has the following syntax:

```
SELECT attribute1, attribute2, ...
FROM table_name
WHERE attributeJ [NOT] IN (value1, value2, ...);
```

For example, to display the first name and the last name of those employees who work in the department number 1, 2, or 5, we can use the following SQL statement:

```
SELECT FName, LName, Dept_No
FROM EMPLOYEE
WHERE Dept_No IN (1,2,5);
```

Execution of the above SQL statement will display the output shown in Table 10.26:

FName	LName	Dept_No
Raj	Reddy	2
Dhiraj	Bora	1
Muskan	Taneja	2
Anshul	Verma	1
Taran	Adarsh	5
Naval	Dhingra	2
Savita	Ambedkar	5

Table 10.26: Employees who work in department number 1, 2, or 5

Next, to display those employees who work for the departments other than department number 1, 2, or 5, we use **NOT** operator in SQL statement as follows:

```
SELECT FName, LName, Dept_No
FROM EMPLOYEE
WHERE Dept_No NOT IN (1,2,5);
```

Executing the above statement will produce the output shown in Table 10.27:

FName	LName	Dept_No
Hiten	Oberoi	4
Rajit	Gadh	4
Naveen	Basra	4

Table 10.27: Employees who do not work in department number 1, 2, or 5

Query: Display the department names and their locations for the departments located in Mumbai or Delhi.

```
SELECT Dept_Name, Location
FROM DEPARTMENT
WHERE Location IN ('Mumbai', 'Delhi');
```

Executing the above statement will produce the output shown in Table 10.28:

Dept_Name	Location
Administration	Delhi
Home Goods	Mumbai
Textile	Mumbai

Table 10.28: Departments located in Mumbai or Delhi

10.3.17 IS Operator

Recall from the previous chapter that when the value of an attribute is not known, we use either the default value (if specified) or the **NULL** value. We can check whether a value is **NULL** using **IS NULL** or **IS NOT NULL**.

Using the following syntax:

```
SELECT attribute1, attribute2, ...
FROM table_name
WHERE attributeJ IS [NOT] NULL;
```

The operator **IS NULL** returns **TRUE** if the specified attribute (attributeJ in the above description) has a **NULL** value and **FALSE** otherwise.

The operator **IS NOT NULL** returns **TRUE** if the specified attribute (attributeJ in the above description) has a non-**NULL** value and **FALSE** otherwise.

For example, to display the department name and the manager ID from the **DEPARTMENT** table for the departments having a manager:

```
SELECT Dept_Name, Mgr_Id
FROM DEPARTMENT
WHERE Mgr_Id IS NOT NULL;
```

Executing the above statement will produce the output shown in Table 10.29:

Dept_Name	Mgr_Id
Accounts	10002
Administration	10005
Home Goods	10003
Automobile	10007
Textile	10004

Table 10.29: Dept_Name, Mgr_Id for the departments with a manager

10.3.18 LIKE Operator

The SQL **LIKE** operator uses wildcard operators % and _ to define a pattern. The percent (%) operator denotes zero, one, or more occurrences of arbitrary characters. The underscore (_) operator represents exactly one occurrence of an arbitrary character. If required, we may use a combination of % and _ operators. The syntax for the **LIKE** operator is given below:

```
SELECT attribute1, attribute2, ...
FROM table_name
WHERE attributeJ LIKE 'pattern';
```

For example, to display ID, first name, last name, and the salary of all employees whose first name starts with **'R'**, use the statement given below:

```
SELECT ID, FName, LName, Salary
FROM EMPLOYEE
WHERE FName LIKE 'R%';
```

Execution of the above query will produce the output shown in Table 10.30:

ID	FName	LName	Salary
10001	Raj	Reddy	100000
10006	Rajit	Gadh	60000

Table 10.30: Employees whose first name begins with 'R%'

Note that there are two tuples having the first name that matches the pattern **'R%'** (**FName** starting with **R**).

Query: Display ID, first name, last name, and the salary of employees whose first name starts with **'R'** and salary in the sixty thousand range, i.e., a five-digit number beginning 6.

```
SELECT ID, FName, LName, Salary
```

```
FROM EMPLOYEE
WHERE FName LIKE 'R%' AND Salary LIKE '6____';
```

Execution of the above query will produce the output shown in Table 10.31:

ID	FName	LName	Salary
10006	Rajit	Gadh	60000

Table 10.31: Employees whose first name begins with 'R%' and have a salary in the range of sixty thousand

Query: Display the employee ID and first name of all employees whose first name contains two **a**'s or two **e**'s.

```
SELECT ID, FName
FROM EMPLOYEE
WHERE FName LIKE '%a%a%' OR FName LIKE '%e%e%';
```

Execution of the above query will produce the output shown in Table 10.32:

ID	FName
10007	Taran
10008	Naval
10009	Naveen
10010	Savita

Table 10.32: Employees whose first name contains two a's or two e's

Query: Display the first name, last name, and date of birth of all employees who were born in 1980.

```
SELECT FName, LName, DOB
FROM EMPLOYEE
WHERE DOB LIKE '1980______';
```

OR

```
SELECT FName, LName, DOB
FROM EMPLOYEE
WHERE DOB LIKE '1980%';
```

Execution of either of the above two queries, will produce the output as shown in Table 10.33:

FName	LName	DOB
Raj	Reddy	1980-06-13
Naveen	Basra	1980-09-24

Table 10.33: Employees born in 1980

10.3.19 Aggregate Functions

SQL provides several functions where the selection criterion applies to an aggregate of tuples instead of a single tuple. Let us learn about some of the aggregate functions.

COUNT() Function

The function **COUNT()** is used to display the count of the tuples satisfying a specified criterion. For example, to find the number of employees earning 75000 or more, **COUNT()** function can be used as follows:

Aggregate Functions: apply to an aggregate of tuples instead of a single tuple.
COUNT() : Returns the number of tuples satisfying a specified condition.

```
SELECT COUNT(*)
FROM EMPLOYEE
WHERE Salary >= 75000;
```

Execution of the above query will produce the output shown in Table 10.34:

COUNT(*)
5

Table 10.34: Count of employees earning 75000 or more

To execute the above statement, the function **COUNT()** finds the number of tuples in the EMPLOYEE table whose Salary attribute has value 75000 or more.

Note that the above output would look much nicer if the SQL output column were named **EmployeeCount** instead of **COUNT(*)**. For this purpose, SQL provides an **AS** clause which renames the output column. The following SQL statement achieves this:



AS: renames the output column for ease of display/ referencing

```
SELECT COUNT(*) AS EmployeeCount
FROM EMPLOYEE
WHERE Salary >= 75000;
```

The above SQL statement introduces a name (EmployeeCount) for the output column. A name defined using the **AS** clause is known as an alias. Execution of the above statement will produce the output shown in Table 10.35:

EmployeeCount
5

Table 10.35: Count of employees as EmployeeCount

SUM() Function

The function **SUM()** is used to compute the sum of attribute values (based on the selection criterion mentioned in **WHERE** clause, if specified). For example, to display the total salary of all employees, the following query may be used:



SUM() : **r**eturns sum of the attribute values satisfying condition(s) in **WHERE** clause

```
SELECT SUM(Salary) AS Total_Salary
FROM EMPLOYEE;
```

Execution of the above statement will produce the output shown in Table 10.36:

Total_Salary
795000

Table 10.36: Total salary of all employees

MAX() Function

The **MAX()** function finds the maximum of a set of values. For example, to display the maximum salary from the EMPLOYEE table, following SQL statement may be used:

```
SELECT MAX(Salary) AS Maximum_Salary
FROM EMPLOYEE;
```

Execution of the above statement will produce the output shown in Table 10.37:

Maximum_Salary
100000

Table 10.37: Maximum salary amongst all employees

For example, to find the employees' first name and last name having a maximum salary, it may be tempting to use the following SQL statement:

```
SELECT FName, LName, MAX(Salary) AS Maximum_Salary
FROM EMPLOYEE;
```

However, on the execution of the above statement, displays only one row for an employee who has the maximum salary, as shown in Table 10.38:

FName	LName	Maximum_Salary
Raj	Reddy	100000

Table 10.38: First name and last name of employee having a maximum salary

Let us first find the maximum salary by executing the sub-query (often called a nested query):

```
SELECT MAX(Salary)
FROM EMPLOYEE;
```

Next, we look for the employees who have the maximum salary (found on the execution of the above query) as follows:

```
SELECT FName, LName, Salary
FROM EMPLOYEE
WHERE Salary = (SELECT MAX(Salary) FROM EMPLOYEE);
```

Execution of the above query will produce the output shown in Table 10.39:

FName	LName	Salary
Raj	Reddy	100000
Muskan	Taneja	100000
Hiten	Oberoi	100000
Anshul	Verma	100000

Table 10.39: Employees getting maximum salary

When a query encloses another query, the inner query is called a nested query. In the above example, the query is an example of a nested query:

MIN() Function

The function **MIN()** yields the minimum value of the specified attribute. For example, to find the minimum salary from the EMPLOYEE table, the SQL query will be:

```
SELECT MIN(Salary) AS Minimum_Salary
FROM EMPLOYEE;
```

Execution of the above query will produce the output as given in Table 10.40:

Minimum_Salary
50000

Table 10.40: Least salary amongst all employees

AVG() Function

The function **AVG()** yields the average value of the specified attribute. To find the average salary of all employees, the query will be:

```
SELECT AVG(Salary) AS Average_Salary
FROM EMPLOYEE;
```

Execution of the above query will produce the output shown in Table 10.41:

Average_Salary
79500.0000

Table 10.41: Average salary of employees rounded to 4 decimal places

Note that the results of applying the **AVG()** function will look more elegant if the average salary is shown up to two places of decimal. For this purpose, SQL provides a **ROUND()** function. Which rounds a number to a specified number of decimal places. In the following query, **ROUND()** is used to round the average salary in each department to two decimal places:

```
SELECT ROUND(AVG(Salary),2) AS Average_Salary
FROM EMPLOYEE;
```

Execution of the above query will produce the output shown in Table 10.42:

Average_Salary
79500.00

Table 10.42: Average salary of employees rounded to two decimal places

10.3.20 GROUP BY Clause

The **GROUP BY** clause returns summary data based on one or more groupings. The grouping may involve one or more attributes of the table. For example, we can use the **GROUP BY** clause to count the number of employees in each department of the **EMPLOYEE** table, or to find the total salary of each department in the **EMPLOYEE** table. The **GROUP BY** clause can also be used with aggregate functions to generate database summary reports. The syntax for **GROUP BY** clause is given below:



GROUP BY: groups the tuples based on value(s) of the specified attribute(s)

```
SELECT attribute1, attribute2, ..., aggregate_function (attributeN)
FROM table_name
[WHERE conditions]
GROUP BY attribute1, attribute2, ...
[ORDER BY attribute [ ASC | DESC ]];
```

To display the minimum, maximum, and average salary (rounded to two decimal places) of each department group by Dept_No from the EMPLOYEE table, the query will be:

```
SELECT Dept_No, MIN(Salary) AS MinSalary,
MAX(Salary) AS MaxSalary,
ROUND(AVG(Salary),2) AS AverageSalary
FROM EMPLOYEE
GROUP BY Dept_No;
```

In the above statement, the **ROUND()** is used to round the average salary in each department to two decimal places.

Execution of the above query will produce the output as given in Table 10.43:

Dept_No	MinSalary	MaxSalary	AverageSalary
2	70000	100000	90000.00
1	85000	100000	92500.00
4	60000	100000	73333.33
5	50000	70000	60000.00

Table 10.43: Minimum, maximum, and average salary (rounded) in each department

10.3.21 HAVING CLAUSE

The **HAVING** clause is used to specify conditions for choosing the groups that will appear in the SQL output. The conditions in the **WHERE** clause apply to the column attributes for selecting the rows. The conditions in the **HAVING** clause apply to the groups that the **GROUP BY** clause has generated. The following syntax of **SELECT** statement includes **FROM, WHERE, GROUP BY, HAVING, ORDER BY** clauses is as follows:



Condition in the **HAVING** clause applies to a group as a whole.

```
SELECT attribute1, attribute2, ..., aggregate_function (attributeJ)
FROM table_name
[WHERE conditions]
GROUP BY attribute1, attribute2, ...
[HAVING conditions]
[ORDER BY attribute [ ASC | DESC ]];
```

To display the minimum, maximum, and average salary (rounded to two decimal places) of each department from the **EMPLOYEE** table having the average salary greater than 70000, the query will be:

```
SELECT Dept_No, MIN(Salary) AS MinSalary, MAX(Salary) AS MaxSalary, ROUND(AVG(Salary),2)
AS AverageSalary
FROM EMPLOYEE
GROUP BY Dept_No
HAVING AVG(Salary)>70000;
```

On executing the above statement, SQL will display the output shown in Table 10.44:

Dept_No	MinSalary	MaxSalary	AverageSalary
2	70000	100000	90000.00
1	85000	100000	92500.00
4	60000	100000	73333.33

Table 10.44: Minimum, maximum, and average salary (rounded to two decimal places) of the departments having the average salary greater than 70000

To display the range of salary for employees residing in the same city having the range of salary greater than equal to 30000, the SQL statement given below may be used:

```
SELECT City, (MAX(Salary) - MIN(Salary)) AS RangeSalary
FROM EMPLOYEE
GROUP BY City
HAVING (MAX(Salary) - MIN(Salary)) >= 30000;
```

On executing the above statement, SQL will display the output shown in Table 10.45:

City	RangeSalary
Delhi	30000
Jamshedpur	40000

Table 10.45 City and the range of salary for employees residing in the same city having the range of salary greater than or equal to 30000

Let's Summarise

- **SQL:** Most popular language for creating and manipulating relational databases.
- **DDL:** Used for defining database attributes, their types, and constraints on attribute values.
- **DML:** Used to retrieve and modify data.
- **Data Types:** Indicates the domain of values for an attribute. For example:
  - **INT** or **INTEGER:** 32-bit integer.
  - **SMALLINT:** typically 16-bit integer.
  - **DECIMAL(L, D):** Fixed point decimal, comprising **L** digits, **D** digits after the decimal point.
  - **FLOAT(p):** Floating point number.
  - **CHAR(L)** or **CHARACTER(L):** Fixed-length character strings of length **L.**
  - **VARCHAR(L):** Variable-length character strings of length up to **L.**
  - **DATE:** The standard date format is **YYYY-MM-DD.**
- **CREATE DATABASE**: Creates a database and assigns it a name.
- **SHOW DATABASES**: Displays names of all the existing databases.
- **USE database_name:** Makes the database active for use.
- **CREATE TABLE:** Creates a table in the database which is in use. It defines:
  - Attribute names, default values
  - Attribute types
  - Primary key
  - **Constraints: NOT NULL**, foreign key, values allowed
- **Constraint:** restricts the values an attribute can take.
- **DESCRIBE**: Gives the structure of the table—attribute names, their data type and constraints.
- **INSERT**: Used to insert a row in a table.
- **UPDATE**: Used to update an attribute value in a table for the tuple(s), typically based on a criterion specified in a **WHERE** clause.
- **DELETE**: Used to delete tuples from a table, typically based on a criterion specified in a **WHERE** clause.
- **DROP TABLE**: Deletes an existing table in a database. The data stored in the table is lost.

- **ALTER TABLE**: Allows us to add, modify, rename, and drop attributes of an existing table. It can also be used to create and remove SQL constraints from tables.
- **SELECT**: Used to retrieve attribute values from the table, typically based on a criterion specified in a **WHERE** clause.
- **ORDER BY** clause: Sorts data based on the specified attribute(s). By default, sorting is done in ascending order. If required **DESC** should be specified to sort the data in descending order.
- Aggregate Functions: Apply to an aggregate of tuples instead of a single tuple, for example:
  - **COUNT() :** Displays the count of the tuples, typically based on a criterion specified in a **WHERE** clause.
  - **MAX()**, **MIN()**, **SUM()**, **AVG()** : Displays the maximum, minimum, sum, or average value of an attribute from the set of tuples.
- **Nested Query**: When an SQL query encloses another query, the outer query may use the result of the inner query.
- **BETWEEN operator**: Checks whether a value is within a specified range. It may be applied to numbers, text, and dates.
- **IN operator**: Checks whether a value is within a set.
- **LIKE** operator: Uses wildcard operators % and _ to define a pattern.
- **GROUP BY clause**: Used to output groupwise summary data.
- **HAVING clause**: Used to group the data so that the group as a whole satisfies a condition.

Solved Exercises

A. Multiple Choice Questions

1. Which of the following is used to ensure that an attribute is not left unspecified?
a. BLANK b. NOT NULL c. DISTINCT d. UNIQUE

2. Which of the following is the keyword used in **SELECT** query to avoid duplicate records?
a. UNIQUE b. DISTINCT c. PRIMARY KEY d. DIFFERENT

3. The **HAVING** keyword is used with _____ clause.
a. ORDER BY b. SORT BY c. GROUP BY d. UPDATE

4. Which of the following functions returns the average value of an SQL expression?
a. AVG() b. AVERAGE() c. MEAN() d. AVERAGEVALUE()

5. Which SQL data type is most suitable for storing date of birth values of the students?
a. CHAR b. VARCHAR c. DATE d. INTEGER

6. Which of the following is a DML statement?
a. ALTER b. CREATE c. DELETE d. None of these

7. What is the correct expansion of DDL?
a. Dual Data Language b. Data Decision Language
c. Data Definition Language d. Data Development Language

8. Which of the following statements is NOT true with respect to databases?
a. DML is used for retrieval of data. b. DML is used for modification of data.
c. DDL is used for specifying constraints. d. DDL is used to delete a row from a database.

9. Which of the following is not a VALID integer value?
a. –5656 b. –5,65,656 c. –0 d. –5656.65

10. While creating a table, Rachita does not want the field value of attribute **Hobby** to be left blank. Instead, she wants the value "NOT AVAILABLE" to be stored. Which of the following constraints should she use while defining the Hobby field during the creation of the table?

a. **CHECK** b. **DEFAULT** c. **UNIQUE** d. **NOT NULL**

11. Which of the following statements will display the structure of the table, PRACTICE?

a. **DESC PRACTICE** b. **STRUCTURE PRACTICE**

c. **SHOW PRACTICE** d. **DISPLAY PRACTICE**

12. If an attribute **Title** is created with the following specifications:

Title **CHAR(10) NOT NULL**

Which of the following statements is NOT correct?

a. Title will be allocated 10 bytes of memory in all cases.

b. Title will be allocated memory according to the length of the field value, but maximum will be 10 bytes.

c. If an attempt is made to insert a string of length more than 10 characters, an error will be generated.

d. If the field value is less than 10 characters, the remaining characters will be filled by blanks.

13. Which operator is used to check whether an attribute is **NULL** values in SQL?

a. **IS** b. **=** c. **EQUAL** d. **IN**

14. Which of the following is the correct order of clauses in a **SELECT** statement?

a. **SELECT, WHERE, ORDER BY, GROUP BY**

b. **SELECT, WHERE, GROUP BY, HAVING**

c. **SELECT, WHERE, ORDER BY, HAVING**

d. **SELECT, WHERE, HAVING, GROUP BY**

15. Which of the following enforces a database constraint?

a. **PRIMARY KEY** b. **UNIQUE** c. **FOREIGN KEY** d. All of these

16. Which statement can be used to select employees from the **EMPLOYEE** table whose name begins with **K?**

a. **SELECT * FROM** EMPLOYEE **WHERE name LIKE** '%K';

b. **SELECT * FROM** EMPLOYEE **WHERE name LIKE** '%K%';

c. **SELECT * FROM** EMPLOYEE **WHERE name LIKE** '_K%';

d. **SELECT * FROM** EMPLOYEE **WHERE name LIKE** 'K%';

17. Which statement can be used to select employees from the **EMPLOYEE** table whose name ends with **Narayanan?**

a. **SELECT * FROM** EMPLOYEE **WHERE name LIKE** '%Narayanan';

b. **SELECT * FROM** EMPLOYEE **WHERE name LIKE** '%Narayanan%';

c. **SELECT * FROM** EMPLOYEE **WHERE name LIKE** '%_Narayanan%';

d. **SELECT * FROM** EMPLOYEE **WHERE name LIKE** 'Narayanan';

18. Which statement can be used to select employees from the **EMPLOYEE** table whose name contains **Kumar?**

a. **SELECT * FROM** EMPLOYEE **WHERE name LIKE** '%Narayanan';

b. **SELECT * FROM** EMPLOYEE **WHERE name LIKE** '%Kumar%';

c. **SELECT * FROM** EMPLOYEE **WHERE name LIKE** '_K%';

d. **SELECT * FROM** EMPLOYEE **WHERE name LIKE** 'K%';

19. Which SQL statement is used to display the result of a query in ascending order of a particular attribute?

a. **ORDER BY** b. **SORT BY** c. **ORDER** d. **SORT**

20. Which of the following aggregate functions ignores **NULL** values?

a. **COUNT()** b. **SUM()** c. **MIN()** d. All of these

B. State whether the following statements are True or False:

1. A DDL command can be used to modify the constraints of an attribute in a table. _____

2. A DML command can only be used to retrieve data. _____

3. A string longer than the defined maximum length will result in the truncation of extra trailing characters. _____

4. A **VARCHAR** data type can have a default value. _____

5. A table may have more than one attribute as a primary key. _____

6. **SELECT** is a DDL command. _____

7. **UNIQUE** constraint can be applied to multiple columns in a table. _____

C. Fill in the blanks.

1. Data Manipulation language (DML) includes statements that modify the _____ of a database.
2. The data type indicates the _____ of values for an attribute.
3. If the length of a field of type **CHAR** is not specified, the default length is _____.
4. Each SQL statement terminates with a _____.
5. _____ command is used to delete a table from a database.
6. The _____ command is used to add, delete, or modify columns in an existing table.

D. Answer the following questions:

1. Which of the following are correct values for the given data types?

	Type	Value
(i)	char(20)	a. 'R Vasudevan'
(ii)	float	b. 1394.78
(iii)	integer	c. 218,790
(iv)	date	d. '1998–5–12'
(v)	integer	e. 4150987

Ans. Correct field values: (a), (b), (d) and (e)

2. Name any two aggregate functions used in SQL.

Ans. **min(), max()**

3. Write an SQL statement to open a database named **MyDB**.

Ans. **USE MyDB;**

4. Differentiate between **ALTER TABLE** and **UPDATE…SET** statements.

Ans. While **ALTER TABLE** is used to modify the structure of the table, **UPDATE … SET** statement is used to modify the values of the attributes in the table.

5. Categorise the following statements into DDL and DML.

SELECT, DELETE, CREATE TABLE, UPDATE…SET, DROP TABLE

Ans. DDL - **CREATE TABLE, DROP TABLE**

DML - **SELECT, DELETE, UPDATE…SET**

6. Which clause is used to display a column value only once?

Ans. **DISTINCT**

7. Which clause is used to display the results of a query in ascending or descending order?

Ans. **ORDER BY**

8. Consider the following query:

SELECT SName FROM Subjects WHERE SName like ???

Which string should replace ??? to display the names of the subjects that end with **'CE'**?

Ans. **'%CE'**

9. Consider the table **TRAVEL** given below and write the SQL queries (i) to (viii).

Table: TRAVEL

CNo	CName	TravelDate	Distance	NPersons	Destination	CAT
101	Shaloo	2022-03-10	340	4	Srinagar	Hill Station
102	Sonia	2022-03-10	224	4	Jaipur	Historical
103	Abhimanyu	2022-01-01	510	2	Goa	Beach
104	Senthil	2022-10-05	125	5	Chandigarh	NULL
105	Gurvinder	2022-07-18	460	1	Goa	Beach
106	Penny	2022-08-08	0	3	Null	Null

(i) Display names of customers who want to travel to a hill station.

(ii) Display records of those customers that have more than 2 persons travelling.

(iii) Display the names and dates of travel of customers in ascending order of **TRAVELDATE**

(iv) Display the number of customers travelling to each destination.

(v) Display the **CNAME** and **TRAVELDATE** of employees whose travelling distance is between 200 and 500 kms.

(vi) Change the destination to **GANGTOK** for customer whose **CID** is 106.

(vii) Display the names of customers whose names begin with 'S'.

(viii) Delete the records of customers whose destination is not known.

Ans. (i)
```
SELECT CNAME FROM TRAVEL WHERE CAT="HILL STATION";
```

(ii)
```
SELECT * FROM TRAVEL WHERE NPERSONS>2;
```

(iii)
```
SELECT CNAME, TRAVELDATE FROM TRAVEL
ORDER BY TRAVELDATE;
```

(iv)
```
SELECT DESTINATION, COUNT(*) FROM TRAVEL
GROUP BY DESTINATION;
```

(v)
```
SELECT CNAME, TRAVELDATE FROM TRAVEL
WHERE DISTANCE BETWEEN 200 AND 500;
```

(vi)
```
UPDATE TRAVEL
SET DESTINATION="GANGTOK"
WHERE CNO=106;
```

(vii)
```
SELECT CNAME FROM TRAVEL
WHERE CNAME LIKE "S%";
```

(viii)
```
DELETE FROM TRAVEL
WHERE DESTINATION IS NULL;
```

10. Refer to table given in question no. 9 and write the output for the following queries:

(i)
```
SELECT MIN(TRAVELDATE) FROM TRAVEL;
```

(ii)
```
SELECT CNAME, TRAVELDATE,DESTINATION FROM TRAVEL
WHERE NPERSONS<3 AND CAT IS NULL;
```

(iii)
```
SELECT AVG(DISTANCE) FROM TRAVEL
WHERE TRAVELDATE BETWEEN "2022-01-01" AND "2022-06-30";
```

(iv)
```
SELECT CAT,SUM(DISTANCE) FROM TRAVEL
GROUP BY CAT
HAVING CAT !="NULL"
```

(v)
```
SELECT CNAME, TRAVELDATE FROM TRAVEL
ORDER BY TRAVELDATE;
```

(vi) `SELECT DESTINATION, COUNT(*) FROM TRAVEL GROUP BY DESTINATION;`

(vii) `SELECT CNAME, TRAVELDATE FROM TRAVEL WHERE DISTANCE BETWEEN 200 AND 500;`

(viii) `SELECT CNAME FROM TRAVEL WHERE CNAME LIKE "S%";`

Ans. (i)

```
+------------------+
| MIN(TRAVELDATE)  |
+------------------+
| 2022-01-01       |
+------------------+
```

(ii) Empty Set

(iii)

```
+---------------+
| AVG(DISTANCE) |
+---------------+
|      358.0000 |
+---------------+
```

(iv)

```
+--------------+---------------+
| CAT          | SUM(DISTANCE) |
+--------------+---------------+
| HILL STATION |           340 |
| HISTORICAL   |           224 |
| BEACH        |           970 |
+--------------+---------------+
```

(v)

```
+------------+------------+
| CNAME      | TRAVELDATE |
+------------+------------+
| ABHIMANYU  | 2022-01-01 |
| SHALOO     | 2022-03-10 |
| SONIA      | 2022-03-10 |
| GURVINDER  | 2022-07-18 |
| PENNY      | 2022-08-08 |
| SENTHIL    | 2022-10-05 |
+------------+------------+
```

(vi)

```
+-------------+----------+
| DESTINATION | COUNT(*) |
+-------------+----------+
| SRINAGAR    |        1 |
| JAIPUR      |        1 |
| GOA         |        2 |
| CHANDIGARH  |        1 |
| NULL        |        1 |
+-------------+----------+
```

(vii)

```
+-----------+------------+
| CNAME     | TRAVELDATE |
+-----------+------------+
| SHALOO    | 2022-03-10 |
| SONIA     | 2022-03-10 |
| GURVINDER | 2022-07-18 |
+-----------+------------+
```

(viii)

```
+---------+
| CNAME   |
+---------+
| SHALOO  |
| SONIA   |
| SENTHIL |
+---------+
```

Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion(A): DDL commands are used to describe the structure of a database.
Reasoning(R): The attributes of tables that are already existing can be added, deleted, or modified in a database.

2. Assertion(A): `SELECT` is a DML command.
Reasoning(R): A DML command is used to retrieve and modify data from a table in a database.

3. Assertion(A): A table will either have a primary key or a composite key.
Reasoning(R): If a table has more than one attribute as the primary key, it is collectively known as a composite key.

Ans. 1. b 2. a 3. b

Case-based Questions

1. Tia is in class 12 and is learning SQL. As the school is about to begin with the Annual Theatre Festival, her teacher has asked her to create the following table in MySQL that will store the schedule of all plays that will be enacted during the festival:

Table: THEATRE

TCODE	TITLE	AUDI	LANGUAGE	SHOWDATE	TICKET PRICE
T001	Akbar The Great Nahi Rahe	2	Hinglish	2023–05–12	150
T002	Mughle-E-Azam	1	Hindi	2023–05–08	200
T003	Romeo and Juliet	2	English	2023–05–09	125
T004	The Lost Love	2	English	2023–05–10	100
T005	Salaam Noni Appa	1	Hindi	2023–05–12	250

(i) Write the command to create the given table with TCODE as primary key. None of the fields should be left blank.

(ii) Write the following SQL queries:

a. To display the names of plays that will be staged in Audi 1.

b. To display the records of those plays that have language as Hindi.

c. To display the names of plays that have ticket price of more than 200.

d. To display the count of plays that will be screened Hindi and English languages.

e. To display the names of plays in alphabetical order.

f. To add an attribute TROUPE to the table. The attribute is of type string OF SIZE 40 and cannot be left blank.

g. To delete the records of those plays that are staged in Audi 1.

Ans. (i) Command to create table:

```
-> CREATE TABLE THEATRE
-> (
-> TCODE CHAR(5) PRIMARY KEY,
-> TITLE CHAR(20) NOT NULL,
-> AUDI INT,
-> LANGUAGE CHAR(15),
-> SHOWDATE DATE,
-> TICKETPRICE INT
-> );
```

(ii) Queries:

a. `SELECT TITLE FROM THEATRE WHERE AUDI=1;`

b. `SELECT * FROM THEATRE WHERE LANGUAGE="Hindi";`

c. `SELECT TITLE FROM THEATRE WHERE TICKETPRICE>200;`

d.
```
SELECT LANGUAGE, COUNT(*) FROM THEATRE
GROUP BY LANGUAGE
HAVING LANGUAGE IN ("HINDI","ENGLISH");
```

e.
```
SELECT TITLE FROM THEATRE
ORDER BY TITLE;
```

f.
```
ALTER TABLE THEATRE
ADD TROUPE CHAR(40) NOT NULL;
```

g. `DELETE FROM THEATRE WHERE AUDI=1;`

2. Ganesh is a whole sale trader of stationeries. He has many dealers working under him. As his business is expanding, it is becoming difficult for him to maintain the records of each dealer. Therefore, he requests you to create a table in MySQL that stores the details of dealers. You have created the table as given below:

Table: Dealer

DealerCode	DName	City	ContactNo	TurnOver
D001	Verma Stationers	Amritsar	9898989898	9834.80
D002	Sindhi Brothers	Surat	1111199999	123.45
D003	Perfect Stationers	Delhi	1234567890	4092.00
D004	Roy and Sons	Kolkata		7164
D005	India Stationers	Delhi	9876987600	567.81

Write the following queries to help Ganesh retrieve the desired records from the table:

(i) To display the records in alphabetical order of names of dealers.

(ii) To display the name of the dealers along with their turnover.

(iii) To display the name and contact number of dealers who are not from Delhi.

(iv) To display the records of dealers whose name begins with an `'R'`.

(v) To display the average turnover of all dealers.

(vi) To display the names of dealers whose contact number is not known.

Ans. (i)
```
SELECT * FROM Dealer
ORDER BY DName;
```
(ii)
```
SELECT Dname, TurnOver FROM Dealer;
```
(iii)
```
SELECT DName, ContactNo FROM Dealer
WHERE City!="Delhi";
```
(iv)
```
SELECT * FROM Dealer
WHERE DName LIKE "R%";
```
(v)
```
SELECT AVG(TurnOver) FROM Dealer;
```
(vi)
```
SELECT DName FROM Dealer
WHERE ContactNo IS NULL;
```



Assessment

A. Multiple Choice questions

1. Which of the following is a DDL command?

a. `ALTER` b. `DELETE` c. `UPDATE` d. `SELECT`

2. In MySQL, what is the default order for the `ORDER BY` clause to sort the records?

a. Descending b. Ascending

c. There is no `ORDER BY` clause in MySQL d. Both a and b

3. Which of the following is a DML command?

a. `DELETE` b. `ALTER` c. `CREATE TABLE` d. `ORDER BY`

4. Which of the following is a valid integer value that can be stored in a MySQL table?

a. 12,345 b. 123–45

c. 12345 d. Twelve thousand three hundred and forty-five

5. Which of the following is the default DATE format in MySQL?

a. yyyy–mm–dd b. dd–mm–yyyy c. dd/mm/yyyy d. yyyy/mm/dd

6. Which of the following constraints is used to restrict the values that an attribute can take?

a. `DEFAULT` b. `CHECK` c. `DISTINCT` d. `WHERE`

7. Which of the following commands will remove the entire table from the MySQL database?
   a. DELETE TABLE b. DROP TABLE c. REMOVE TABLE d. CLEAR TABLE

8. Which of the following SQL statements will display all tables in a database?
   a. SHOW ALL TABLES b. SHOW TABLES
   c. SELECT * FROM DATABASE d. DESCRIBE TABLES

9. Consider a table PICNIC. Which of the following statements will not generate any error?
   a. ALTER TABLE PICNIC SET PAX=5 b. UPDATE TABLE PICNIC MODIFY VENUE="JAIPUR"
   c. DROP FROM TABLE PICNIC d. DELETE FROM PICNIC

10. Which of the following is not a valid key in MySQL?
    a. Primary Key b. Foreign Key c. Alternate Key d. Secondary Key

11. Which of the following DML statements is used to retrieve records from a database?
    a. QUERY b. SELECT c. INSERT d. RETRIEVE

12. Which of the following functions is used to display the total number of records from a table?
    a. TOTAL() b. SUM() c. COUNT(*) d. COUNTRECORDS()

13. Which of the following keywords is used, along with ALTER TABLE command to delete an attribute from a table?
    a. DROP b. REMOVE c. DELETE d. DEL

14. Which of the following is NOT a valid operator and so cannot be used in SQL queries?
    a. IN b. IS c. LIKE d. SAME AS

15. Which of the following wild card characters are used with the LIKE operator to define a pattern of records to be retrieved?
    a. % and ? b. _ (underscore) and % c. _ (unserdcore) and ? d. % and #

16. Which of the following clauses can be used to generate summary reports in a database?
    a. ORDER BY b. REPORT BY c. GROUP BY d. SUMMARY

B. State whether the following statements are True or False:

1. UNIQUE and PRIMARY KEY constraints are same. _____
2. Attributes cannot be added to an existing table in a database. _____
3. Using UPDATE-SET command, we can perform conditional updates in a table in MySQL. _____
4. An underscore(_) character along with LIKE operator represents exactly one occurrence of an arbitrary character. _____
5. A name of a column defined using AS clause in a SELECT statement is known as an alias. _____
6. A primary key attribute can have a NULL value. _____

C. Fill in the blanks.

1. The SELECT statement, combined with _____ clause, retrieves the records from a MySQL table and displays them without any repetition.
2. An attribute in a relation is a foreign key if it is a _____ key in some other table.
3. A _____ restricts the values an attribute can take.
4. _____ constraint ensures that no two rows have the same value in the specified columns.
5. _____ clause is used with the SELECT statement to specify conditions on groups of data to be displayed.
6. In case of a nested query, the _____ query may use the result of an _____ query.
7. Along with the SELECT statement, the _____ clause is used to display the result of an SQL query in ascending or descending order with respect to specified attribute values.

D. Answer the following questions:

1. For each of the following, point out the SQL statement that will enable us to perform the specific operation on the database:
   (i) To view the list of databases.
   (ii) To start using the database named TEST.

(iii) To view the structure of table PRACTICE.

(iv) To display all the records from table PRACTICE.

2. Differentiate between **ORDER BY** and **GROUP BY** clauses of SQL.

3. What are the two wildcard characters used for pattern matching while using the keyword **LIKE**?

4. Consider the table TEACHER given below and answer the questions that follow:

Table: TEACHER

ID	First_Name	Last_Name	Dept	Contact_Num	Salary	Email_ID
P0001	Hema	Priya	Political Science	9953799955	75000	hemapriya@gmail.com
P0002	Allu	Arjun	English	8010235773	35000	arjunallu@gmail.com
P0003	Naishadh	Kumar	History	9965789799	75000	naishadh.k@yahoo.com
P0004	Sameer	Reddy	Computer Science	9899417155	90000	sameer@hotmail.com
P0005	Sathvik	Bal	Political Science	9653367876	85000	NULL
P0006	Tenzin	Wangdi	Economics	8023456780	65000	tenzin@yahoo.com
P0007	Krishan	Kumar	Economics	9977885566	70000	krishnakumar@gmail.com

(i) Write a **CREATE TABLE** statement for table **TEACHER**.

(ii) Write a **SELECT** statement for each of the following:

a. List the salary of those teachers whose name start with 'S'.

b. List the first name and last name of the teachers who have salary more than 70000.

c. List the count of number of teachers of each department.

d. List the first name and contact number of teachers whose mail id is not known.

e. Display the rows from the table **TEACHER** in descending order of salary.

f. Add an attribute **Subject** to the table **TEACHER**.

g. Drop attribute **Email_ID** from the table **TEACHER**.

(iii) For each of the following SQL queries, write the output that will be produced on its execution:

a.
```
SELECT AVG(Salary) FROM TEACHER
WHERE Dept='Economics';
```

b.
```
SELECT First_Name, Last_Name,Contact_Num
FROM TEACHER
WHERE Salary BETWEEN 40000 AND 80000;
```

c.
```
SELECT DISTINCT Dept FROM TEACHER;
```

d.
```
SELECT MAX(Salary) FROM TEACHER
GROUP BY Dept
HAVING Dept='Political Science';
```

5. Consider the table STUDENT given below and answer the questions that follow:

Table: STUDENT

Roll_Num	Student_Name	Course_Name	Duration(months)	Fee	Batch_Prefer
1	Bhaskar Dhyani	Web Development	3	20000	Morning
2	Dakshinesh	Machine Learning	4	25000	Evening
3	Saumit Raj	Office Tools	NULL	50000	Evening
4	Lovepreet Singh	Mobile App	3	18000	Morning
5	Aditya Jayant	Python Programming	6	45000	Evening
6	Jenny Fernandis	Office Tools	4	40000	Evening

(i) Write a **CREATE TABLE** statement for table **STUDENT**.

(ii) Write the **SELECT** statements to retrieve records from the **STUDENT** table for the following:

a. List the names of students in each course.

b. Display the course name along with their duration for the courses whose fee is more than **30000**.

c. Display the names of students in alphabetical order.

d. Display the count of students who prefer the **Evening** batch.

e. Display the average duration of courses from the table.

f. Increase the fee by 10% for **Mobile App** course.

g. Delete rows from table **STUDENT** where duration of the course is not known.

(iii) For each of the following SQL queries, write the output that will be produced on its execution:

a.
```
SELECT * FROM STUDENT
WHERE Student_Name LIKE '%h' AND Fee < 25000;
```

b.
```
SELECT MIN(Fee) FROM STUDENT
GROUP BY Batch_Prefer;
```

c.
```
SELECT Roll_Num, Course_Name, Duration(months) FROM STUDENT
ORDER BY Duration DESC;
```

6. Consider the table SALESPERSON given below and answer the questions that follow:

Table: SALESPERSON

Code	Name	Salary	Product	City
1001	SANDEEP JHA	60000	Stationary	New Delhi
1002	YOGRAJ SINHA	70000	Footwear	Gurugram
1003	TENZIN JACK	45000	Books	Noida
1004	TARANA SEN	55000	Toys	Gurugram
1005	ANOKHI RAJ	45000	Footwear	Noida

(i) Write the **SELECT** statement to retrieve records from **SALESPERSON** table for the following:

a. List the names of salespersons who are dealing with `Books` in `Noida`.

b. Display the names of cities without any repetition.

c. Display the count of salespersons in each city.

d. Display the name and salary of salespersons in descending order of `Salary`.

e. Delete the sales person whose salary is more than `60000`.

f. Add a field `Contact_No` in the table `SALESPERSON` to store the mobile number of salespersons.

(ii) For each of the following SQL queries, write the output that will be produced on its execution:

a.
```
SELECT Product, SUM(Salary) FROM SALESPERSON
GROUP BY Product;
```

b.
```
SELECT Name, Salary FROM SALESPERSON
WHERE Product IN ('Toys','Footwear');
```

c.
```
SELECT Code, Name FROM SALESPERSON
WHERE Salary>50000 and Name LIKE '%E%';
```

d.
```
SELECT AVG(Salary) FROM SALESPERSON
WHERE CITY='Gurugram';
```

7. Consider the table `MOBILE` that stores information about the company manufacturing a mobile, its names, price, and launch date. Thereafter answer the questions that follow:

Table: MOBILE

M_Id	M_Company	M_Name	M_Price	Launch_Date
MB001	Samsung	Galaxy	4500	2013-02-12
MB003	Nokia	N1100	2250	2011-04-15
MB004	Micromax	Unite3	4500	2016-10-17
MB005	Sony	XperiaM	7500	2017-11-20
MB006	Oppo	SefieEx	8500	2010-08-21

(i) Write SQL queries for the following:

a. To insert a new tuple in the **MOBILE** table whose **M_Price** is not yet known.

b. Display the mobile name and name of the company of the mobile phones whose price is greater than **5000**.

c. List the name of the mobile phones along with their price that were launched in the year **2017**.

d. Display **M_Company, M_Name** and **M_Price** in descending order of their launch date.

e. List the details of a mobile whose name starts with "S" or ends with "a".

f. Display the names of the mobile companies having prices between **3000** and **5000**

(ii) What will be the output produced on the execution of the following SQL queries:

a. **SELECT MAX(**Launch_Date**), MIN(**Launch_Date**) FROM** MOBILE;

b. **SELECT AVG(**M_Price**) FROM** MOBILE;

c. **SELECT** M_Name, Launch_Date **FROM** MOBILE
WHERE M_Company='Nokia' **OR** M_Price **IS NULL;**

8. Consider the table **TRAINS** and answer the questions that follow:

Table: TRAINS

TNO	TName	Start	End	Journey_Time
1651	Pune Hbj Special	Pune Junction	Habibganj	6
11096	Ahimsa Express	Pune Junction	Ahmedabad Junction	12
12002	Bhopal Shatabdi	New Delhi	Habibganj	28
12015	Ajmer Shatabdi	New Delhi	Ajmer Junction	4
12030	Swarna Shatabdi	Amritsar Junction	New Delhi	6
12314	Sealdah Rajdhani	New Delhi	Sealdah	37
12417	Prayag Raj Express	Allahabad Junction	New Delhi	14
12451	Shram Shakti Express	Kanpur Central	New Delhi	5
12498	Shan-e-Punjab	Amritsar Junction	New Delhi	6
13005	Amritsar Mail	Howrah Junction	Amritsar Junction	40
14673	Shaheed Express	Jaynagar	Amritsar Junction	36

(i) Write SQL queries for the following:

a. To display details of all trains which start from Pune Junction.

b. To display details of trains that have duration of more than 15 hours.

c. To display the names of trains that end with the word "Shatabdi".

d. To delete the record of trains that start from Jaynagar.

e. To change the duration of Swarna Shatabdi to 7 hours.

f. To display the count of trains that end at **New Delhi**.

(ii) Write the output for the following SQL queries:

a. **SELECT** START, **COUNT(*) FROM** TRAINS
GROUP BY START
HAVING COUNT(*)>1;

b. **SELECT** TNO, TName **FROM** TRAINS
WHERE Journey_Time
ORDER BY TNO;

c. **SELECT** Start, End **FROM** TRAINS
WHERE Journey_Time **AND COUNT(*)=**2;

9. Consider the following table STUDENT that stores information about the students in a school'. Thereafter, answer the questions that follow: [CBSE Sample Paper, 2020]

Table: STUDENT

RollNO	Name	Class	DOB	Gender	City	Marks
1	Nanda	X	06-06-1995	M	Agra	551
2	Saurabh	XII	07-05-1993	M	Mumbai	462
3	Sanal	XI	06-05-1994	F	Delhi	400
4	Trisla	XII	08-08-1995	F	Mumbai	450
5	Store	XII	08-10-1995	M	Delhi	369
6	Marisla	XI	12-12-1994	F	Dubai	250
7	Neha	X	08-12-1995	F	Moscow	377
8	Nishant	X	12-06-1995	M	Moscow	489

(i) Write the command to create the table **STUDENT**.

(ii) Write an SQL command to insert a new tuple in the table **STUDENT** whose values are:

Roll No-10, Name-Aziz, Class-XI, DOB: 15-7-1996, Gender-M, City-Delhi, and marks are not yet known.

(iii) Write SQL queries for the following based on the table STUDENT:

a. To display the average marks in each class.

b. To display the average marks in each class, gender-wise.

b. To display the number of students in each city.

c. To display the records from table STUDENT in alphabetical order of the names of the students.

d. To display RollNO, Class, DOB and City for students whose marks are between 450 and 551.

e. To increase marks of all students by 20 who are in class XII.

(iv) What will be the output produced on the execution of each of the following queries:

a. **SELECT COUNT(*),** City **FROM** STUDENT
GROUP BY CITY **HAVING COUNT(*)>1;**

b. **SELECT MAX(**DOB**), MIN(**DOB**) FROM** STUDENT;

c. **SELECT NAME, GENDER FROM** STUDENT **WHERE CITY=**'Delhi';

10. Consider the following database table ACCOUNTS that deals with the accounts of the customers in a bank. Thereafter answer the questions that follow. [2018]

Table: ACCOUNTS

ANo	AName	Branch	Contact_Number	Balance_Amount	Account_Type
101	Nirja Singh	Bangalore	9812356093	90870.6	Savings
102	Rohan Gupta	Chennai	NULL	129065	Current
103	Ali Reza	Hyderabad	8877091567	45000.7	Savings
104	Rishabh Jain	Chennai	9971092568	89002.8	Savings
105	Simran Kaur	Chandigarh	7680145320	150001	Current

(i) Write SQL statements to create the table ACCOUNTS.

(ii) Write the command to insert a new tuple in the ACCOUNTS table with the following values:

Ano:106
Name: Joe Mathews
Branch: Goa
Contact Number: Not Known
Balance Amount: 76392.90
Type of account: Savings

(iii) Write SQL queries for the following:

a. To display details of all account holders who have a savings account from table ACCOUNTS.

b. To display the Name and Contact Number of all account holders Branch wise.

c. To display the names of the account holders whose balance is less than 20000.

d. To display Name and Branch of account holders whose contact number is not known.

e. To display the names of Branch only once.

(iv) What will be the output produced on execution of the following SQL queries:

a.
```
SELECT ANO, ANAME FROM ACCOUNTS
WHERE BRANCH NOT IN ('CHENNAI', 'BANGALORE');
```

b.
```
SELECT DISTINCT ANO FROM ACCOUNTS;
```

c.
```
SELECT ANO, COUNT(*), BALANCE_AMOUNT FROM ACCOUNTS GROUP BY ANO
HAVING COUNT(*)> 1;
```



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion(A): UPDATE-SET is a DDL command.

Reasoning(R): A DDL command modifies the structure of a table in a database.

2. Assertion(A): COUNT() is an example of an aggregate function.

Reasoning(R): Aggregate functions apply to a single tuple in a relation.

3. Assertion(A): GROUP BY clause along with aggregate functions can be used to generate summary reports in a database.

Reasoning(R): WHERE clause is used to specify conditions on groups that will be displayed as output.



Case-based Questions

1. Sarfaraz runs a stationary shop. He wants to store the details of stationary so that he can retrieve the desired data as and when required. The database designer has created the following table:

Table: Stationary

SCODE	SNAME	PRICE	QTY	BRAND
S001	Gel Pen	55	10	Reynolds
S002	Permanent Marker	60	5	Luxor
S003	White Board Marker	45	10	Luxor
S004	Paper Ream	310	4	Sando
S006	Pencil		15	Natraj
S009	Colouring Book	70	3	Artist
S101	Eraser	15	5	Natraj
S102	Sharpener	6	12	Natraj

Sarfaraz wants to retrieve some data from the table. Help him by writing the queries for the following:

(i) To display the structure of the table.

(ii) To display the records whose quantity is less than 5.

(iii) To display the name and price of those stationary items whose names end with `'Marker'`.

(iv) To display the price of the most expensive item.

(v) To display the average price of all items.

(vi) To display the brand name and count of items of each brand.

(vii) To display the total number of records in a table.

(viii) To display the records in descending order of quantity.

(ix) To display the records whose price is not known.

(x) To display the name of each brand only once.

2. Tanmay is very fond of watches, just like his father. His father has a very good collection of watches. As a student of class 12, he learnt to create tables and perform queries in a MySQL database. He has created the table, Watches as given below:

Table: Watches

WatchId	WName	Price	Type	Brand	YOP
T001	Apple iWatch	56000	Smart watch	Apple	2020–12–12
T002	Titan Raga	25673	Designer	Titan	2015–04–15
T003	NeoSplash	34908	Sports Watch	Neo Sports	2018–11–09
T005	G-Shock	13892	Sports Watch	Casio	2021–03–18
T006	Max Pro	15673	Designer	Maxima	2014–06–03

Write the queries for the following:

(i) To display the name and price of each watch.

(ii) To change the price of Apple iWatch to 59350.

(iii) To display the records of all Sports watches.

(iv) To display the names of all brands, without repetition.

(v) To display the count of designer watches.

(vi) To display the count of watches of each brand.

(vii) To display the name and brand of watches that have been purchased in the year 2020.



Previous Years' Questions

1. Fill in the blank: [2023]

_____ clause is used with SELECT statement to display data in a sorted form with respect to a specified column.

a. `WHERE` b. `ORDER BY` c. `HAVING` d. `DISTINCT`

2. Which function returns the sum of all elements of a list? [2023]

a. `count()` b. `sum()` c. `total()` d. `add()`

Ans. b. `sum()`

3. a. Differentiate between CHAR and VARCHAR data types in SQL with appropriate example.

OR

b. Name any two DDL and any two DML commands. [2023]

Ans. a. char() is a fixed-length data type, whereas varchar() is a variable-length data type.

For example:

Consider the field: name char (20)

If we store "Prerna" in it, the name will occupy 20 bytes. Now if we store the same name in a field, name varchar(20), it will occupy 6 bytes, which is the length of the word "Prerna".

b. DDL Commands: `CREATE TABLE, ALTER TABLE`

DML Commands: `INSERT INTO, UPDATE`

4. Write output of SQL Queries (a) to (d) based on the table VACCINATION_DATA given below: [2022]

VID	NAME	AGE	DOSE1	DOSE2	CITY
101	Jenny	27	2021–12–25	2022–01–31	Delhi
102	Harjot	55	2021–07–14	2021–10–14	Mumbai
103	Srikanth	43	2021–04–18	2021–07–20	Delhi
104	Gazala	75	2021–07–31	NULL	Kolkata
105	Shiksha	32	2022–01–01	NULL	Mumbai

(i) `SELECT NAME, AGE FROM VACCINATION_DATA WHERE Dose2 IS NOT NULL AND Age > 40;`

Ans.

NAME	AGE
Harjot	55
Srikanth	43

(ii) `SELECT CITY, COUNT(*) FROM VACCINATION_DATA GROUP BY City;`

Ans.

City	Count
Delhi	2
Mumbai	2
Kolkata	1

(iii) `SELECT DISTINCT CITY FROM VACCINATION_DATA;`

Ans.

DISTINCT CITY
Delhi
Mumbai
Kolkata

(iv) `SELECT MAX (DOSE1), MIN(DOSE2) FROM VACCINATION DATA;`

Ans.

max(dose1)	min(dose2)
2022–01–01	2021–07–20

Answers

Multiple Choice Questions

1. (b)	2. (b)	3. (c)	4. (a)	5. (c)	6. (c)	7. (c)	8. (d)	9. (b)
10. (b)	11. (a)	12. (b)	13. (a)	14. (b)	15. (d)	16. (d)	17. (a)	18. (b)
19. (a)	20. (d)							

True or False

1. (T)	2. (F)	3. (T)	4. (F)	5. (T)	6. (F)	7. (T)

Fill in the blanks

1. structure	2. domain	3. 1	4. ; (semicolon)
5. DROP TABLE	6. ALTER TABLE		

11 SQL—WORKING WITH MULTIPLE TABLES

Chapter Outline

11.1 Referential Integrity
11.2 Joins

Introduction

In the previous chapter, we discussed creating tables, inserting records, and retrieving information from tables in a database. However, one query would retrieve tuples from one table only. This chapter will use Structured Query Language (SQL) to retrieve tuples from multiple tables. Such tables have a common attribute that creates a relationship between two or more tables. While inserting the data into such tables, we must ensure that the foreign key constraints are not violated.

11.1 Referential Integrity

While working with two or more related tables in a database, SQL enforces the referential integrity constraints. Figure 11.1 shows the relationship between entities `EMPLOYEE`, `DEPARTMENT`, and `PROJECT`. Since an `EMPLOYEE` works in a `DEPARTMENT`, these two entities are related via `Dept_No` of the table `EMPLOYEE`, which refers to the primary key `Dept_No` of `DEPARTMENT`. You will recall that the attribute `Dept_No` of the table `EMPLOYEE` is called a foreign key as it is the primary key of the table `DEPARTMENT`. Since `DEPARTMENT` is managed by a manager who is also an `EMPLOYEE`, these two entities are also related through foreign key `Mgr_Id` of `DEPARTMENT`, which refers to the primary key `ID` of `EMPLOYEE`. Further, since each `PROJECT` is handled by a `DEPARTMENT`, these two entities are related to each other through foreign key `Dept_Num` of `PROJECT`, which refers to the primary key `Dept_No` of `DEPARTMENT`.

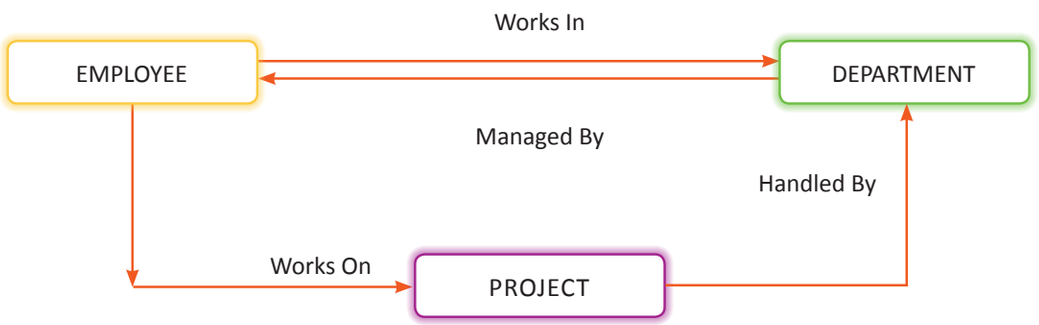


Figure 11.1 Relationship between entities of COMPANY Database

DBMS uses foreign keys to enforce the integrity of the database. For example, DBMS will disallow an attempt to insert a tuple in the EMPLOYEE table having a Dept_No that is not present in the DEPARTMENT table. Using SQL, we can specify Dept_No as a foreign key in the table EMPLOYEE and indicate that it references the primary key Dept_No of the table DEPARTMENT as follows:

```
FOREIGN KEY (Dept_No) REFERENCES DEPARTMENT (Dept_No)
```

The revised **CREATE TABLE** statement for the table EMPLOYEE would appear as follows:

```
CREATE TABLE EMPLOYEE
(
    ID INT PRIMARY KEY,
    FName VARCHAR(20) NOT NULL,
    LName VARCHAR(20) NOT NULL,
    Gender CHAR(1) NOT NULL,
    Address VARCHAR(50),
    City VARCHAR(20),
    Pin_Code CHAR(6),
    DOB DATE,
    Salary INT NOT NULL,
    Dept_No SMALLINT,
    FOREIGN KEY(Dept_No) REFERENCES DEPARTMENT(Dept_No)
);
```

Recall that a foreign key should either refer to existing **UNIQUE** attribute (typically primary key) value of referenced table or should be **NULL**. An update in the **UNIQUE** attribute (typically primary key) will lead to a corresponding update in the table where this attribute is a foreign key. For example, suppose Hiten Oberoi—an employee who recently joined the company as a department manager, complains that his Aadhar number has been shown wrongly in his payslip. In that case, the company will have to correct his Aadhar number by updating the EMPLOYEE table. Such an update in the EMPLOYEE table will trigger a correction in the DEPARTMENT table for the department he manages. We can use the following **CREATE TABLE** statement for creating the DEPARTMENT table:

```
CREATE TABLE DEPARTMENT (
Dept_No SMALLINT PRIMARY KEY,
Dept_Name VARCHAR(25) NOT NULL,
Location VARCHAR(30),
Mgr_Id INT UNIQUE
);#DEPARTMENT
```

Now, we are faced with the following situation:

We want to specify in the description of the EMPLOYEE table that Dept_No of EMPLOYEE references Dept_No of DEPARTMENT. But for this purpose, the DEPARTMENT table must already exist in the database; else SQL will generate an error.

We also want to specify in the description of the DEPARTMENT table that Mgr_Id of DEPARTMENT references the ID of EMPLOYEE, But for this purpose, the EMPLOYEE table must already exist in the database.

To resolve the above conflict, we first create the tables without specifying foreign key constraints and then alter them by including the foreign key constraints. as shown below:

```
ALTER TABLE EMPLOYEE
ADD FOREIGN KEY (Dept_No) REFERENCES DEPARTMENT (Dept_No);
ALTER TABLE DEPARTMENT
ADD FOREIGN KEY (Mgr_Id) REFERENCES EMPLOYEE(ID);
```

11.1.1 Handling Referential Integrity Constraints

Since Mgr_Id of DEPARTMENT referencing the ID of EMPLOYEE and Dept_No of EMPLOYEE referencing Dept_No of DEPARTMENT, there are some restrictions in inserting the tuples in EMPLOYEE and DEPARTMENT tables. For example, suppose we wish to insert information about the Accounts department having Dept_No 1, located in Noida. Assume that the manager of the department has not yet been identified. We can still insert this tuple using the following **INSERT** statement:

```
INSERT INTO DEPARTMENT
VALUES (1, 'Accounts', 'Noida', NULL);
```

Suppose, we wish to insert information about the Automobile department having Dept_No 5, located in Jamshedpur, and having a manager whose ID is 10007. To insert a tuple into the DEPARTMENT table for the Automobile department, let us execute the following **INSERT** statement:

```
INSERT INTO DEPARTMENT
VALUES (4, 'Automobile', 'Jamshedpur', 10007);
```

The DBMS will reject such an attempt to insert data in the DEPARTMENT table as it violates the foreign key constraint. Recall that the foreign key constraint dictates that before insertion of tuple with **VALUES** (4, 'Automobile', 'Jamshedpur', 10007), a tuple must exist in the EMPLOYEE table having ID value 10007. However, at this point, the EMPLOYEE table does not have a tuple having ID value 10007. So, while inserting a tuple for the Accounts department, we must specify the value of attribute Mgr_Id equal to NULL as shown below:

```
INSERT INTO DEPARTMENT
VALUES (4, 'Automobile', 'Jamshedpur', NULL);
```

Likewise, we can insert some more tuples in the DEPARTMENT table. Finally, the DEPARTMENT table appears as Table 11.1.

Dept_No	Dept_Name	Location	Mgr_Id
1	Accounts	Noida	NULL
2	Administration	Delhi	NULL
3	Home Goods	Mumbai	NULL
4	Automobile	Jamshedpur	NULL
5	Textile	Mumbai	NULL

Table 11.1: DEPARTMENT table after inserting some tuples with Mgr_Id set to NULL

Now, let us insert into the EMPLOYEE table, a tuple for Taran Adarsh whose ID is 10007 and belongs to Automobile department as follows:

```
INSERT INTO EMPLOYEE
VALUES (10007, 'Taran', 'Adarsh', 'M', 'B-76, CST Road, Kalina, Santacruz East',
'Mumbai', 400098, '1965-01-13', 70000, 4);
```

Once a tuple is stored in the EMPLOYEE table having ID value 10007, we can update the table DEPARTMENT to set Mgr_Id equal to 10007 using the following UPDATE statement:

```
UPDATE DEPARTMENT
SET Mgr_Id = 10007
WHERE Dept_No = 4;
```

The above UPDATE statement finds a tuple in the DEPARTMENT table having Dept_No equal to 4 and sets the value of its Mgr_Id attribute to 10007. Table 11.2 shows the updated DEPARTMENT table:

Dept_No	Dept_Name	Location	Mgr_Id
1	Accounts	Noida	NULL
2	Administration	Delhi	NULL
3	Home Goods	Mumbai	NULL
4	Automobile	Jamshedpur	10007
5	Textile	Mumbai	NULL

Table 11.2: DEPARTMENT table after updating the Mgr_Id to 10007 for Dept_No = 5

On inserting the required data, the current state of the EMPLOYEE and DEPARTMENT tables are shown in Tables 11.3 and 11.4, respectively:

ID	FName	LName	Gender	Address	City	Pin_Code	DOB	Salary	Dept_No
10001	Raj	Reddy	M	West Godavari	Andhra Pradesh	534197	1980-06-13	100000	2
10002	Dhiraj	Bora	M	Dispur, Kamrup, Assam	Guwahati	781005	1975-09-30	85000	1
10003	Muskan	Taneja	F	8/33, Geeta Colony	Delhi	110031	1990-01-25	100000	2
10004	Hiten	Oberoi	M	15, Dimna Road, Mango	Jamshedpur	831018	1985-06-24	100000	4
10005	Anshul	Verma	M	House 10, Sector 16, Gautum Budh Nagar	Noida	201304	1990-01-01	100000	1
10006	Rajit	Gadh	F	12, Beldih Triangle, Bistupur	Jamshedpur	831001	1960-05-07	60000	4
10007	Taran	Adarsh	M	B-76, CST Road, Kalina, Santacruz East	Mumbai	400098	1965-01-13	70000	5
10008	Naval	Dhingra	M	E-14 Vivek Vihar	Delhi	110095	1975-08-04	70000	2
10009	Naveen	Basra	F	28, Aambagan Road, Sakechi	Jamshedpur	831001	1980-09-24	60000	4
10010	Savita	Ambedkar	F	C-49, G-Block, Bandra Kurla, Bandra East	Mumbai	400051	1987-07-11	50000	5

Table 11.3: EMPLOYEE table

Dept_No	Dept_Name	Location	Mgr_Id
1	Accounts	Noida	10002
2	Administration	Delhi	10005
3	Home Goods	Mumbai	10003
4	Automobile	Jamshedpur	10007
5	Textile	Mumbai	10004

Table 11.4: DEPARTMENT table

11.1.2 SELECT Statement Involving Multiple Tables

When a **SELECT** statement is used to retrieve data from multiple tables, names of all the tables are specified in the **FROM** clause in the form of a table_list. Syntax of a **SELECT** statement involving multiple tables is given below:

```
SELECT attribute [, attribute] ...
FROM table_list
WHERE criterion_for_selecting_specific_tuple
```

To execute the above query, the **FROM** clause constructs the cross product of the specified tables, then the attributes mentioned in the **SELECT** clause are selected from those tuples of the cross product which satisfy the criterion specified in **WHERE** clause. To obtain a cross product of tables in SQL, we only need to list all the tables whose cross product is required in the **FROM** clause. For example, the following **SELECT** statement yields cross product of tables EMPLOYEE and DEPARTMENT:

```
Select*
FROM EMPLOYEE, DEPARTMENT;
```

Execution of the above SQL statement yields Table 11.5:

ID	FName	LName	...	DOB	Salary	Dept_No	Dept_No	...	Mgr_Id
10001	Raj	Reddy	...	1980-06-13	100000	2	5	...	10004
10001	Raj	Reddy	...	1980-06-13	100000	2	4	...	10007
10001	Raj	Reddy	...	1980-06-13	100000	2	3	...	10003
10001	Raj	Reddy	...	1980-06-13	100000	2	2	...	10005
10001	Raj	Reddy	...	1980-06-13	100000	2	1	...	10002
10002	Dhiraj	Bora	...	1975-09-30	85000	1	5	...	10004
10002	Dhiraj	Bora	...	1975-09-30	85000	1	4	...	10007
10002	Dhiraj	Bora	...	1975-09-30	85000	1	3	...	10003
10002	Dhiraj	Bora	...	1975-09-30	85000	1	2	...	10005
10002	Dhiraj	Bora	...	1975-09-30	85000	1	1	...	10002
.									
.			...					...	
.									
10009	Naveen	Basra	...	1980-09-24	60000	4	5	...	10004
10009	Naveen	Basra	...	1980-09-24	60000	4	4	...	10007
10009	Naveen	Basra	...	1980-09-24	60000	4	3	...	10003
10009	Naveen	Basra	...	1980-09-24	60000	4	2	...	10005
10009	Naveen	Basra	...	1980-09-24	60000	4	1	...	10002
10010	Savita	Ambedkar	...	1987-07-11	50000	5	5	...	10004
10010	Savita	Ambedkar	...	1987-07-11	50000	5	4	...	10007
10010	Savita	Ambedkar	...	1987-07-11	50000	5	3	...	10003
10010	Savita	Ambedkar	...	1987-07-11	50000	5	2	...	10005
10010	Savita	Ambedkar	...	1987-07-11	50000	5	1	...	10002

Table 11.5: Cross Product of EMPLOYEE and DEPARTMENT

Next, suppose we wish to retrieve complete details about employees along with the details of their departments. To do this, we need to consider only those tuples from Table 11.5 which have the same value of the Dept_No attribute for the tables EMPLOYEE **and** DEPARTMENT. The following SQL query achieves this:

```
SELECT *
FROM EMPLOYEE, DEPARTMENT
WHERE EMPLOYEE.Dept_No = DEPARTMENT.Dept_No;
```

Note that the attribute Dept_No is common in EMPLOYEE and DEPARTMENT tables. SQL provides a simple mechanism of distinguishing between the attributes of different tables having the same name. It allows us to prefix an attribute name by the table's name followed by a period to distinguish between common attribute names in different tables. Thus EMPLOYEE.Dept_No denotes attribute Dept_No of table EMPLOYEE and DEPARTMENT.Dept_No denotes attribute Dept_No of table DEPARTMENT. You will recall that this is the usual method of specifying an attribute of an object.

Suppose we wish to find the ID, first name, last name, department numbers, department name, and salary for employees with a salary greater than 80000. The following **SELECT** statement achieves this:

```
SELECT ID, FName, LName, EMPLOYEE.Dept_No, Dept_Name, Salary
FROM EMPLOYEE, DEPARTMENT
WHERE EMPLOYEE.Dept_No = DEPARTMENT.Dept_No AND Salary > 80000;
```

Execution of the above statement will yield Table 11.6:

ID	FName	LName	Dept_No	Dept_Name	Salary
10001	Raj	Reddy	2	Administration	100000
10002	Dhiraj	Bora	1	Accounts	85000
10003	Muskan	Taneja	2	Administration	100000
10004	Hiten	Oberoi	4	Automobile	100000
10005	Anshul	Verma	1	Accounts	100000

Table 11.6: Details from EMPLOYEE and DEPARTMENT tables for the employees with Salary > 80000

Sometimes, it is convenient to define an alias (another name) for a table using the keyword **AS.** Thus above query may be written as follows:

```
SELECT ID, FName, LName, E.Dept_No, Dept_Name, Salary
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE E.Dept_No = D.Dept_No AND E.Salary > 80000;
```

In the above query, **E** and **D** are aliases for the tables **EMPLOYEE** and **DEPARTMENT** respectively. The result of the above query (Table 11.7) is the same as in Table 11.6.

ID	FName	LName	Dept_No	Dept_Name	Salary
10001	Raj	Reddy	2	Administration	100000
10002	Dhiraj	Bora	1	Accounts	85000
10003	Muskan	Taneja	2	Administration	100000
10004	Hiten	Oberoi	4	Automobile	100000
10005	Anshul	Verma	1	Accounts	100000

Table 11.7: Details from EMPLOYEE and DEPARTMENT tables for the employees with salary > 80000

Now, let us answer some more queries using SQL statements.

Query: Arrange the employees' details (ID, FName, Dept_Name, and Salary) in ascending order of their department names.

Solution:

```
SELECT ID, FName, Dept_Name, Salary
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE E.Dept_No = D.Dept_No
ORDER BY Dept_Name ASC;
```

The result of executing the above query is shown in Table 11.8:

ID	FName	Dept_Name	Salary
10002	Dhiraj	Accounts	85000
10005	Anshul	Accounts	100000
10001	Raj	Administration	100000
10003	Muskan	Administration	100000
10008	Naval	Administration	70000
10004	Hiten	Automobile	100000
10006	Rajit	Automobile	60000
10009	Naveen	Automobile	60000
10007	Taran	Textile	70000
10010	Savita	Textile	50000

Table 11.8: Employee details in ascending order of department names

Query: Arrange the employees' details (ID, FName, Dept_Name, and Salary) in ascending order of their department names. However, within a department, arrange the employees in descending order of their Salary.

Solution:

```
SELECT ID, FName, Dept_Name, Salary
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE E.Dept_No = D.Dept_No
ORDER BY Dept_Name ASC, Salary DESC;
```

The result of executing the above query is shown in Table 11.9:

ID	FName	Dept_Name	Salary
10005	Anshul	Accounts	100000
10002	Dhiraj	Accounts	85000
10001	Raj	Administration	100000
10003	Muskan	Administration	100000
10008	Naval	Administration	70000
10004	Hiten	Automobile	100000
10006	Rajit	Automobile	60000
10009	Naveen	Automobile	60000
10007	Taran	Textile	70000
10010	Savita	Textile	50000

Table 11.9: Employee details in ascending order of department names and descending order salary of employees within a department

Query: Display salary of all employees in the `Accounts` department.

Solution:

```
SELECT Salary, Dept_Name
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE E.Dept_No = D.Dept_No AND Dept_Name = 'Accounts';
```

Table 11.10 shows the result on executing the above SELECT statement:

Salary	Dept_Name
85000	Accounts
100000	Accounts

Table 11.10: Details from `EMPLOYEE` and `DEPARTMENT` tables for the `Dept_Name = 'Accounts'`

Query: Compute the average salary of all employees in the Accounts department.

Solution: To answer this query, we apply the **AVG** function to the salary of employees in the `Accounts` department:

```
SELECT AVG(Salary) AS Average_Salary, Dept_Name
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE E.Dept_No = D.Dept_No AND Dept_Name = 'Accounts';
```

Table 11.11 shows the result on executing the above SELECT statement:

Average_Salary	Dept_Name
92500.0000	Accounts

Table 11.11: Average salary of employees for `Dept_Name = 'Accounts'`

11.1.3 UPDATE Statement

As discussed in the previous chapter, an **UPDATE** statement is used to update the tuples, typically based on one or more conditions. Now we will demonstrate the use of **UPDATE** operation based on a condition involving more than one table. For example,

Query: Increase the salary of employees who belong to the Accounts department by `Rs. 1000.`

Solution:

```
UPDATE EMPLOYEE AS E, DEPARTMENT AS D
SET SALARY = SALARY + 1000
WHERE E.Dept_No = D.Dept_No AND Dept_Name = 'Accounts';
```

UNION, INTERSECTION AND DIFFERENCE IN SQL

You may combine the results returned by two different **SELECT** clauses using union, intersection and set-difference operators. The tables involved in these operations must be union compatible. Let us consider the problem discussed before. A company wishes to give bonuses to all employees working in the Administration department or working for more than 20 hours on any project. We can answer this query by writing two queries and taking the union of the results obtained on executing them. The first query finds the ID of employees working in the Administration department, and the second query finds the ID of employees working for more than 20 hours on any project. Thus, we may use the following SQL statement to answer the original query:

```
(SELECT E.ID
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE E.Dept_No = D.Dept_No AND D.Dept_Name = 'Administration')
UNION
(SELECT DISTINCT Emp_Id
FROM WORKS_ON
WHERE Hours > 20);
```

Result after executing the above statement is shown in Table 11.12:

ID
10001
10003
10008
10010
10006

Table 11.12: UNION for displaying the employees who are either working in the Administration department or work for more than 20 hours on any project

Similarly, you may use operators **INTERSECT** and **EXCEPT** for intersection and set difference, respectively, depending upon the problem statement.

Examples

Let us now practice some SQL queries that retrieve data from multiple tables.

Table: EMPLOYEE

ID	FName	LName	Gender	Address	City	Pin_Code	DOB	Salary	Dept_No
10001	Raj	Reddy	M	West Godavari	Andhra Pradesh	534197	1980-06-13	100000	2
10002	Dhiraj	Bora	M	Dispur, Kamrup, Assam	Guwahati	781005	1975-09-30	85000	1
10003	Muskan	Taneja	F	8/33, Geeta Colony	Delhi	110031	1990-01-25	100000	2
10004	Hiten	Oberoi	M	15, Dimna Road, Mango	Jamshedpur	831018	1985-06-24	100000	4
10005	Anshul	Verma	M	House 10, Sector 16, Gautum Budh Nagar	Noida	201304	1990-01-01	100000	1
10006	Rajit	Gadh	F	12, Beldih Triangle, Bistupur	Jamshedpur	831001	1960-05-07	60000	4
10007	Taran	Adarsh	M	B-76, CST Road, Kalina, Santacruz East	Mumbai	400098	1965-01-13	70000	5
10008	Naval	Dhingra	M	E-14 Vivek Vihar	Delhi	110095	1975-08-04	70000	2
10009	Naveen	Basra	F	28, Aambagan Road, Sakechi	Jamshedpur	831001	1980-09-24	60000	4
10010	Savita	Ambedkar	F	C-49, G-Block, Bandra Kurla, Bandra East	Mumbai	400051	1987-07-11	50000	5

Table: DEPARTMENT

Dept_No	Dept_Name	Location	Mgr_Id
1	Accounts	Noida	10002
2	Administration	Delhi	10005
3	Home Goods	Mumbai	10003
4	Automobile	Jamshedpur	10007
5	Textile	Mumbai	10004

Query: Retrieve the first name, last name, and the city of all employees who work in the `Administration` department.

Solution:

```
SELECT FName, LName, City
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE E.Dept_No = D.Dept_No AND Dept_Name = 'Administration';
```

Preceding query will generate the output shown in Table 11.13:

FName	LName	City
Raj	Reddy	Andhra Pradesh
Muskan	Taneja	Delhi
Naval	Dhingra	Delhi

Table 11.13: First name, last name, and the city of all employees who work in the Administration department

Query: Find the sum of the salaries of all employees of the `Accounts` department, as well as the maximum salary, the minimum salary, and the average salary in this department.

Solution:

```
SELECT SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE E.Dept_No = D.Dept_No AND Dept_Name = 'Accounts';
```

Preceding query will generate the output shown in Table 11.14:

SUM(Salary)	MAX(Salary)	MIN(Salary)	AVG(Salary)
185000	100000	85000	92500.0000

Table 11.14: Sum of the salaries, maximum salary, minimum salary, and the average salary of all employees of the Accounts department

Query: Find the number of employees in the `Administration` department.

Solution:

```
SELECT COUNT(*) AS NumOfEmployees
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE E.Dept_No = D.Dept_No AND Dept_Name = 'Administration';
```

Preceding query will generate the output shown in Table 11.15:

NumOfEmployees
3

Table 11.15: Number of employees in the Administration department.

Query: Retrieve the names of all the departments and their managers' names.

Solution:

```
SELECT Dept_Name, Fname, LName
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE E.ID = D.Mgr_Id;
```

Preceding query will generate the output shown in Table 11.16:

Dept_Name	Fname	LName
Accounts	Dhiraj	Bora
Administration	Anshul	Verma
Home Goods	Muskan	Taneja
Automobile	Taran	Adarsh
Textile	Hiten	Oberoi

Table 11.16: Names of all the departments and their manager's name

Query: Retrieve the names of all employees who work in the department located in the same city where they live.

Solution:

```
SELECT FName, LName
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE (E.Dept_No = D.Dept_No) AND (E.City = D.Location);
```

Preceding query will generate the output shown in Table 11.17:

FName	LName
Muskan	Taneja
Hiten	Oberoi
Anshul	Verma
Rajit	Gadh
Taran	Adarsh
Naval	Dhingra
Naveen	Basra
Savita	Ambedkar

Table 11.17: Names of all employees who work in the department located in the same city in which they live.

Query: Retrieve the first and last names of the employees working in the same department as the employee with `Employee ID` = 10001.

Answer:

```
SELECT E1.FName AS EmpFName, E1.LName AS EmpLName, E2.FName AS ColleagueFName,
E2.LName AS CollagueLName
FROM EMPLOYEE AS E1, EMPLOYEE AS E2
WHERE (E1.Dept_No = E2.Dept_No) AND (E1.ID = 10001) AND (E2.ID != 10001);
```

Preceding query will generate the output shown in Table 11.18:

EmpFName	EmpLName	ColleagueFName	CollagueLName
Raj	Reddy	Muskan	Taneja
Raj	Reddy	Naval	Dhingra

Table 11.18: First name, last name of the employees along with the employees working in the same department of Employee ID = 10001.

Query: Retrieve the first name, last name, department name, and date of birth of the employees born after the employee named "Hiten Oberoi".

Solution:

```
SELECT FName, LName, Dept_Name, DOB
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE (E.Dept_No = D.Dept_No) AND DOB >
(SELECT DOB FROM EMPLOYEE
WHERE FName = "Hiten" AND LName = "Oberoi");
```

Preceding query will generate the output shown in Table 11.19:

```
+--------+----------+----------------+------------+
| FName  | LName    | Dept_Name      | DOB        |
+--------+----------+----------------+------------+
| Muskan | Taneja   | Administration | 1990-01-25 |
| Anshul | Verma    | Accounts       | 1990-01-01 |
| Savita | Ambedkar | Textile        | 1987-07-11 |
+--------+----------+----------------+------------+
```

Table 11.19: First name, last name, department name, and date of birth of the employees who are born after the employee named "Hiten Oberoi"

Query: Retrieve the department name, location, number of employees, and the average salary for all employees in that department.

Solution:

```
SELECT Dept_Name, Location, COUNT(ID) AS "NoofEmployees", AVG(Salary) AS
"AverageSalary"
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE (E.Dept_No = D.Dept_No)
GROUP BY Dept_Name;
```

Preceding query will generate the output shown in Table 11.20:

```
+----------------+------------+---------------+---------------+
| Dept_Name      | Location   | NoofEmployees | AverageSalary |
+----------------+------------+---------------+---------------+
| Administration | Delhi      |             3 |    90000.0000 |
| Accounts       | Noida      |             2 |    92500.0000 |
| Automobile     | Jamshedpur |             3 |    73333.3333 |
| Textile        | Mumbai     |             2 |    60000.0000 |
+----------------+------------+---------------+---------------+
```

Table 11.20: Department name, location, number of employees, and the average salary for all employees in that department

11.2 Joins

11.2.1 EQUI JOIN

Suppose we wish to retrieve the details of all the employees along with their department details. The following SQL query joins the tables EMPLOYEE and DEPARTMENT to achieve this:

```
SELECT *
FROM DEPARTMENT AS D, EMPLOYEE AS E
WHERE E.DEPT_NO = D.DEPT_NO;
```

Note that the result of executing the above query yields the column Dept_No twice, once for each table (Table 11.21). As the above query joins only those tuples which have identical values of Dept_No, such a join is called *equijoin*.

Dept_No	Dept_Name	Location	Mgr_Id	ID	FName	...	City	PinCode	...	Salary	Dept_No
2	Administration	Delhi	10005	10001	Raj	...	Andhra Pradesh	534197	...	100000	2
1	Accounts	Noida	10002	10002	Dhiraj	...	Guwahati	781005	...	85000	1
2	Administration	Delhi	10005	10003	Muskan	...	Delhi	110031	...	100000	2
4	Automobile	Jamshedpur	10007	10004	Hiten	...	Jamshedpur	831018	...	100000	4
1	Accounts	Noida	10002	10005	Anshul	...	Noida	201304	...	100000	1
4	Automobile	Jamshedpur	10007	10006	Rajit	...	Jamshedpur	831001	...	60000	4
5	Textile	Mumbai	10004	10007	Taran	...	Mumbai	400098	...	70000	5
2	Administration	Delhi	10005	10008	Naval	...	Delhi	110095	...	70000	2
4	Automobile	Jamshedpur	10007	10009	Naveen	...	Jamshedpur	831001	...	60000	4
5	Textile	Mumbai	10004	10010	Savita	...	Mumbai	400051	...	50000	5

Table 11.21: Equijoin of EMPLOYEE and DEPARTMENT

TABLE: Employee

ID	FName	...	Dept_No
10001	Raj		2
10002	Dhiraj		1
10003	Muskan		2
10004	Hiten		4
10005	Anshul		1
10006	Rajit		4
10007	Taran		5
10008	Naval		2
10009	Naveen		4
10010	Savita		5

TABLE: Department

Dept_No	Dept_Name	Location	Mgr_Id
1	Accounts	Noida	10002
2	Administration	Delhi	10005
3	Home Goods	Mumbai	10003
4	Automobile	Jamshedpur	10007
5	Textile	Mumbai	10004

Query:
```
SELECT FNAME, Dept_Name
FROM Employee E, Department D
WHERE E.Dept_No=D.Dept_No;
```

FName	Dept_Name	Dept_No
Raj	Administration	2
Dhiraj	Accounts	1
Muskan	Administration	2
Hiten	Automobile	4
Anshul	Accounts	1
Rajit	Automobile	4
Taran	Textile	5
Naval	Administration	2
Naveen	Automobile	4
Savita	Textile	5

Figure 11.2: Equi-Join

Let us see some more examples of equijoin.

Query: To display the first name, last name and department name of all employees.

Solution:

```
SELECT FName, LName, Dept_Name
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE E.Dept_No = D.Dept_No;
```

Preceding query will generate the output shown in Table 11.22:

FName	LName	Dept_Name
Raj	Reddy	Administration
Dhiraj	Bora	Accounts
Muskan	Taneja	Administration
Hiten	Oberoi	Automobile
Anshul	Verma	Accounts
Rajit	Gadh	Automobile
Taran	Adarsh	Textile
Naval	Dhingra	Administration
Naveen	Basra	Automobile
Savita	Ambedkar	Textile

Table 11.22: First name, last name, and department name of all employees

To display the name of the departments located in Mumbai and first name of the employees working in it.

Solution:

```
SELECT FName, Dept_Name
FROM EMPLOYEE AS E, DEPARTMENT AS D
WHERE E.Dept_No = D.Dept_No AND Location = "Mumbai";
```

Preceding query will generate the output shown in Table 11.23:

FName	Dept_Name
Taran	Textile
Savita	Textile

Table 11.23:First name and department name of departments located in Mumbai

11.2.2 NATURAL JOIN

SQL supports a **NATURAL JOIN** operator that removes the duplicate column common to both the tables and positions the common attribute as the first column in the result. A **NATURAL JOIN** query to get the details of all the managers of all the departments may be formulated as follows:

```
SELECT *
FROM EMPLOYEE
NATURAL JOIN DEPARTMENT;
```

The execution of the above query will yield Table 11.24. Note that Dept_No is the first column in Table 11.24:

Dept_No	ID	FName	LName	City	PinCode	DOB	Salary	Dept_Name	Location	Mgr_Id
2	10001	Raj	Reddy	Andhra Pradesh	534197	1980-06-13	100000	Administration	Delhi	10005
1	10002	Dhiraj	Bora	Guwahati	781005	1975-09-30	85000	Accounts	Noida	10002
2	10003	Muskan	Taneja	Delhi	110031	1990-01-25	100000	Administration	Delhi	10005
4	10004	Hiten	Oberoi	Jamshedpur	831018	1985-06-24	100000	Automobile	Jamshedpur	10007
1	10005	Anshul	Verma	Noida	201304	1990-01-01	100000	Accounts	Noida	10002
4	10006	Rajit	Gadh	Jamshedpur	831001	1960-05-07	60000	Automobile	Jamshedpur	10007
5	10007	Taran	Adarsh	Mumbai	400098	1965-01-13	70000	Textile	Mumbai	10004
2	10008	Naval	Dhingra	Delhi	110095	1975-08-04	70000	Administration	Delhi	10005
4	10009	Naveen	Basra	Jamshedpur	831001	1980-09-24	60000	Automobile	Jamshedpur	10007
5	10010	Savita	Ambedkar	Mumbai	400051	1987-07-11	50000	Textile	Mumbai	10004

Table 11.24: Natural Join of EMPLOYEE and DEPARTMENT

Of course, we can achieve the same effect by dropping the duplicate column while specifying the list of columns to be displayed in the **SELECT** clause as shown below:

```
SELECT EMPLOYEE.Dept_No, ID, FName, LName, Gender, Address, City, Pin_Code, DOB,
Salary, Dept_Name, Location, Mgr_Id
```

```
FROM EMPLOYEE, DEPARTMENT
WHERE EMPLOYEE.Dept_No = DEPARTMENT.Dept_No;
```

Table 11.25 shows the result of executing the above **SELECT** statement:

Dept_No	ID	FName	LName	City	Pin_Code	DOB	Salary	Dept_Name	Location	Mgr_Id
2	10001	Raj	Reddy	Andhra Pradesh	534197	1980-06-13	100000	Administration	Delhi	10005
1	10002	Dhiraj	Bora	Guwahati	781005	1975-09-30	85000	Accounts	Noida	10002
2	10003	Muskan	Taneja	Delhi	110031	1990-01-25	100000	Administration	Delhi	10005
4	10004	Hiten	Oberoi	Jamshedpur	831018	1985-06-24	100000	Automobile	Jamshedpur	10007
1	10005	Anshul	Verma	Noida	201304	1990-01-01	100000	Accounts	Noida	10002
4	10006	Rajit	Gadh	Jamshedpur	831001	1960-05-07	60000	Automobile	Jamshedpur	10007
5	10007	Taran	Adarsh	Mumbai	400098	1965-01-13	70000	Textile	Mumbai	10004
2	10008	Naval	Dhingra	Delhi	110095	1975-08-04	70000	Administration	Delhi	10005
4	10009	Naveen	Basra	Jamshedpur	831001	1980-09-24	60000	Automobile	Jamshedpur	10007
5	10010	Savita	Ambedkar	Mumbai	400051	1987-07-11	50000	Textile	Mumbai	10004

Table 11.25: Equijoin of EMPLOYEE and DEPARTMENT with department number appearing once

Let's Summarise

- To retrieve information from multiple tables, the tables are joined on one or more common attributes.
- Foreign keys are used to enforce the integrity of the database across the tables.
- In a relation, the value of a foreign key may be NULL, or it may be the same as the value of a **UNIQUE** attribute (generally primary key) value of the referenced table.
- When a **SELECT** statement is used to retrieve data from multiple tables, names of the relevant tables are specified in the **FROM** clause.
- If cardinality of Table1 is m and cardinality of Table2 is n, the cross product of Table1 and Table2 will have cardinality m * n.
- Equijoin condition tests for the equality of value of the attribute common to both tables.
- In the case of Natural join, the two tables are joined on two attributes with the same name and same domain.

Solved Exercises

A. Multiple Choice Questions

1. Which of the following is NOT true with respect to working on multiple tables in a MySQL database?

a. The related tables must have a common attribute.

b. While entering data into related tables, foreign key constraints must not be violated.

c. All tables should be from the same database.

d. The common attribute should not be a primary key in any of the tables.

2. Tables T1 and T2 have 6 and 3 tuples, respectively. What will be the degree and cardinality of the cross product of T1 and T2?

a. 9 and 18, respectively

b. 18 and 9, respectively

c. 18 and 18, respectively

d. 9 and 9, respectively

3. Which of the following is NOT true about Natural Join?

a. The attribute common to both tables has the same name in both tables.

b. The attribute common to both tables has the same domain in both tables.

c. Natural join removes the duplicate column from the cross product of the two tables.

d. Natural join positions the common attribute as the last column of the result.

Answer questions numbers 4–9 based on the tables WORKER and PAYLEVEL tables given below:

Table: WORKER

W_code	Name	Desig	DOJ	DOB	P_Level
11	Ghanshyam	Supervisor	2006-10-15	1981-12-12	P001
12	Gonsalvis	Operator	2010-03-24	1990-07-10	P003
13	Azam	Mechanic	2014-06-24	1995-10-09	P001
14	Reena	Clerk	2015-08-10	1980-01-15	P002
15	Ritesh	Supervisor	2005-11-06	1979-03-31	P001

Table: PAYLEVEL

P_Level	Pay	Allowance
P001	45000	14000
P002	39000	10000
P003	35000	8000

4. What will be the degree and cardinality of the table produced on the execution of the following query?

```
SELECT * FROM WORKER NATURAL JOIN PAYLEVEL;
```

a. Degree: 8 Cardinality: 5
b. Degree: 8 Cardinality: 15
c. Degree: 9 Cardinality: 5
d. Degree: 9 Cardinality:15

5. Which attribute will enforce referential integrity between the two tables—WORKER and PAYLEVEL?

a. WCode b. Name c. P_Level d. Pay

6. Which of the following queries is an example of an equi-join?

a. `SELECT Name, Pay FROM WORKER, PAYLEVEL;`
b. `SELECT Name, Pay FROM WORKER, PAYLEVEL WHERE Pay=40000;`
c. `SELECT Name, Pay FROM WORKER, PAYLEVEL WHERE WORKER.P_Level = PAYLEVEL.P_Level;`
d. `SELECT Name, Pay FROM WORKER, PAYLEVEL WHERE P_Level=P_Level;`

7. What will be the output produced on execution of the following query?

```
SELECT COUNT(*) FROM WORKER W, PAYLEVEL P WHERE W.P_Level=P.P_Level;
```

a. 5 b. 15 c. 6 d. 9

8. What will be the output produced on execution of the following query?

```
SELECT MAX(Allowance) FROM WORKER W , PAYLEVEL P
WHERE W.P_Level=P.P_Level;
```

a. 10000 b. 8000 c. 14000 d. 24000

9. What will be the degree and cardinality of the table produced on the execution of the following query?

```
SELECT * FROM WORKER,PAYLEVEL;
```

a. Degree: 15 Cardinality: 9
b. Degree: 8 Cardinality: 9
c. Degree: 9 Cardinality: 8
d. Degree: 9 Cardinality:15

10. Tables T1 and T2 have 6 and 3 tuples, respectively. What will be the cardinality of the natural join of T1 and T2?

a. 18
b. 3
c. 6
d. Information is not sufficient to determine the cardinality of the join of T1 and T2.

B. State whether the following statements are True or False:

1. It is not compulsory to have a common attribute to perform an equi-join operation. _____
2. DBMS uses alternate keys to enforce the integrity of the database. _____
3. To distinguish between the attributes of different tables having the same name, the table's name is followed by a period and the attribute name. _____
4. In the case of Natural join, the attributes that join the two tables should have the same domain. _____
5. While inserting data into related tables, the foreign key constraints can be violated. _____

C. Fill in the blanks.

1. _____ keys are used to enforce the integrity of the database across the tables.
2. If cardinality of TableA is x and cardinality of Table B is y, the cross product of TableA and TableB will have cardinality _____.

3. While working with two or more related tables in a database, SQL enforces the _____ constraints.
4. When a SELECT statement is used to retrieve data from multiple tables, names of all the tables are specified in the _____ clause.
5. A _____ operator that removes the duplicate column common to both the tables in an SQL query.

D. Answer the following questions:

1. Describe the notion of referential integrity constraints.

Ans. If a table has a foreign key that refers to a UNIQUE attribute (typically primary key) of another table, then the foreign key value should refer to an existing primary key value or should have a NULL value.

2. Differentiate between Natural Join and Equi-Join.

Ans. **Natural Join:** It is based on the common attribute in the two tables being joined. In the resultant query, the duplicate column is removed and the common column is the first column.

Equi-Join: It is also based on the common attribute between the two tables. The values of this common attribute are compared using the equality operator (=). No other comparison operator can be used in case of an equi-join.

E. Application Based Questions:

1. In addition to EMPLOYEE and DEPARTMENT tables, consider the PROJECT and WORKS_ON tables given below to answer the queries that follow:

CREATE TABLE statements for the tables PROJECT and WORKS_ON are given below:

```
CREATE TABLE PROJECT
(
      Proj_No SMALLINT PRIMARY KEY,
      Proj_Name VARCHAR(25) NOT NULL,
      Dept_Num SMALLINT,
      FOREIGN KEY(Dept_Num) REFERENCES DEPARTMENT (Dept_No)
);
CREATE TABLE WORKS_ON
(
Proj_No SMALLINT NOT NULL,
Emp_Id INT NOT NULL,
Hours SMALLINT,
PRIMARY KEY(Proj_No, Emp_Id),
FOREIGN KEY(Proj_No) REFERENCES PROJECT (Proj_No),
FOREIGN KEY(Emp_Id) REFERENCES EMPLOYEE (ID)
);
```

Table: PROJECT

Proj_No	Proj_Name	Dept_Num
1	Victory	2
2	Wisdom	1
3	Sputnik	1
4	Wizards	5
5	Performer	4

Table: WORKS_ON

Proj_No	Emp_ID	Hours
1	10001	23
2	10005	18
3	10002	4
2	10002	4
4	10010	30
5	10009	10
4	10007	8
1	10003	17
1	10008	21
2	10006	25

a. For each project, retrieve project number, project name, and the number of employees who work on that project.

Ans.
```
SELECT P.Proj_No, Proj_Name, COUNT(*) AS "NoOfEmployee"
FROM PROJECT AS P, WORKS_ON AS W
WHERE P.Proj_No = W.Proj_No
GROUP BY P.Proj_No;
```

b. For each project on which more than one employee works, retrieve the project number, the project name, and the number of employees who work on the project.

Ans.
```
SELECT P.Proj_No, Proj_Name, COUNT(*) AS "NoOfEmployee"
FROM PROJECT AS P, WORKS_ON AS W
WHERE P.Proj_No = W.Proj_No
GROUP BY P.Proj_No
HAVING COUNT(*) > 1;
```

c. To display complete details of all the departments' managers.

```
SELECT EMPLOYEE.*
FROM EMPLOYEE, DEPARTMENT
WHERE MGR_ID=ID;
```

Ans.

ID	FName	LName	Gender	Address	City	Pin_Code	DOB	Salary	Dept_No
10002	Dhiraj	Bora	M	Dispur, Kamrup, Assam	Guwahati	781005	1975–09–30	86000	1
10003	Muskan	Taneja	F	8/33, Geeta Colony	Delhi	110031	1990–01–25	100000	2
10004	Hiten	Oberoi	M	15, Dimna Road, Mango	Jamshedpur	831018	1985–06–24	100000	4
10005	Anshul	Verma	M	House 10,Sector 16,Gautum Budh Nagar	Noida	201304	1990–01–01	101000	1
10007	Taran	Adarsh	M	B-76, CST Road, Kalina, Santacruz East	Mumbai	400098	1965–01–13	70000	5

2. Consider the COLLEGE database for a school, comprising the following tables: TEACHER, DEPARTMENT, SUBJECT, and STUDENT. Each department has a head of the department, although departments may not have been assigned head of the department yet. A department controls several subjects, for example, Science department controls Mathematics, Computer Science, Physics, and Chemistry as shown in the SUBJECT table. A student may enroll for one subject only.

Table: TEACHER

ID	First_Name	Last_Name	Subject_No	Contact_Num	Salary	Email_ID
10001	Hema	Priya	1	9953799955	75000	hemapriya@gmail.com
10002	Allu	Arjun	2	8010235773	35000	arjunallu@gmail.com
10003	Naishadh	Kumar	4	9965789799	75000	naishadh.k@yahoo.com
10004	Sameer	Reddy	1	9899417155	90000	sameer@hotmail.com
10005	Sathvik	Bal	8	9653367876	85000	sathvik@gmail.com
10006	Tenzin	Wangdi	3	8023456780	65000	tenzin@yahoo.com
10007	Krishan	Kumar	2	9977885566	70000	krishnakumar@gmail.com
10008	Ankuj	Pandey	5	9990825566	80000	ankujp@gmail.com
10009	Sangeetha	NULL	2	9890235262	65000	sangeetha@gmail.com
10010	Charu	Srivastava	7	9555235321	95000	charus@gmail.com

Table: DEPARTMENT

Dept_No	Dept_Name	HOD_Id
1	Arts	10005
2	Commerce	10007
3	Science	10003
4	Physical Education	NULL

Table: SUBJECT

Subject_No	Subject_Name	Dept_No
1	English	1
2	Chemistry	3
3	Physics	3
4	Business Studies	2
5	Economics	1
6	Accounts	2
7	Mathematics	3
8	History	1
9	Geography	1
10	Computer Science	3

Table: STUDENT

Roll_Num	Student_Name	Subject_No
1	Bhaskar Dhyani	1
2	Dakshinesh	2
3	Saumit Raj	1
4	Lovepreet Singh	2
5	Aditya jayant	4
6	Venkata Ramana	10
7	Mohd. Abdullah	3
8	Ram Prasad	1
9	Rohith P	2
10	Manoj Kundu	8

Using the tables TEACHER and SUBJECT, write a **SELECT** statement for each of the following :

(i) For each teacher, list his/her first and last name and the name of the subject he/she teaches.

Ans.
```
SELECT First_Name, Last_Name, Subject_Name
FROM SUBJECT S, TEACHER T
WHERE T.Subject_No = S.Subject_No;
```

(ii) List subject number and subject name for the subjects for which teachers have been currently hired. You should avoid duplicate rows.

Ans.
```
SELECT DISTINCT S.Subject_No, Subject_Name
FROM SUBJECT S, TEACHER T
WHERE T.Subject_No = S.Subject_No;
```

(iii) List the name of the subjects for which teachers have been currently hired.

Ans.
```
SELECT DISTINCT Subject_Name
FROM SUBJECT S, TEACHER T
WHERE T.Subject_No = S.Subject_No;
```

(iv) For each subject, list the subject number, subject name and the first and last names of teachers teaching that subject. The output of the query should be arranged subject-wise.

Ans.
```
SELECT DISTINCT S.Subject_No, Subject_Name, First_Name, Last_Name
FROM SUBJECT S, TEACHER T
WHERE T.Subject_No = S.Subject_No
ORDER BY S.Subject_No;
```

(v) For each subject, list the subject number, subject name and the number of teachers teaching that subject.

Ans.
```
SELECT S.Subject_No, Subject_Name, COUNT(*) AS "NoOfTeachers"
FROM SUBJECT S, TEACHER T
WHERE T.Subject_No = S.Subject_No
GROUP BY Subject_Name;
```

(vi) For each subject, list the subject number and average salary of the teachers teaching that subject.

Ans.
```
SELECT S.Subject_No, AVG(Salary) AS "AverageSalary"
FROM SUBJECT S, TEACHER T
WHERE T.Subject_No = S.Subject_No
GROUP BY S.Subject_No;
```

(vii) For each subject, list the subject ID and the maximum salary amongst the teachers teaching that subject.

Ans.
```
SELECT S.Subject_No, MAX(Salary) AS "MaximumSalary"
FROM SUBJECT S, TEACHER T
WHERE T.Subject_No = S.Subject_No
GROUP BY S.Subject_No;
```

(viii) For each subject, list the subject name and the maximum salary amongst the teachers teaching that subject.

Ans.
```
SELECT Subject_Name, MAX(Salary) AS "MaximumSalary"
FROM SUBJECT S, TEACHER T
WHERE T.Subject_No = S.Subject_No
GROUP BY Subject_Name;
```

(ix) For each subject, list the subject name, first and last names of teachers drawing salary more than 70000. The output should be arranged alphabetically subject-wise and within a subject; teachers' names should appear alphabetically according to their first names.

Ans.
```
SELECT Subject_Name, First_Name, Last_Name, Salary
FROM SUBJECT S, TEACHER T
WHERE T.Subject_No = S.Subject_No AND Salary > 70000
ORDER BY Subject_Name, First_Name;
```

(x) Find the number of subjects for which teachers have been currently hired.

Ans.
```
SELECT COUNT(DISTINCT S.Subject_No)
```

```
FROM SUBJECT S, TEACHER T
WHERE S.Subject_No = T. Subject_No;
```

3. Consider the COLLEGE database and write the output that will be produced on the execution of following queries:

a.
```
SELECT Dept_Name, First_Name AS HODFirstName, Last_Name AS HODLastName
FROM TEACHER T, DEPARTMENT D
WHERE T.ID = D.HOD_Id;
```

Ans.

Dept_Name	HODFirstName	HODLastName
Arts	Sathvik	Bal
Commerce	Krishan	Kumar
Science	Naishadh	Kumar

b.
```
SELECT Subject_Name, Dept_Name
FROM SUBJECT AS S, DEPARTMENT AS D
WHERE S.Dept_No = D.Dept_No;
```

Ans.

Subject_Name	Dept_Name
English	Arts
Economics	Arts
History	Arts
Geography	Arts
Business Studies	Commerce
Accounts	Commerce
Chemistry	Science
Physics	Science
Mathematics	Science
Computer Science	Science

c.
```
SELECT Student_Name, Subject_Name
FROM SUBJECT AS J, STUDENT AS S
WHERE J.Subject_No = S.Subject_No;
```

Ans.

Student_Name	Subject_Name
Bhaskar Dhyani	English
Dakshinesh	Chemistry
Saumit Raj	English
Lovepreet Singh	Chemistry
Aditya jayant	Business Studies
Venkata Ramana	Computer Science
Mohd. Abdullah	Physics
Ram Prasad	English
Rohith P	Chemistry
Manoj Kundu	History



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): In the case of an equi join, values from joining tables are retrieved only if the condition in the WHERE clause is satisfied.

   Reasoning (R): An equi join retrieves data from two or more tables.

2. Assertion (A): Equi join and Natural join are the same.

   Reasoning(R): A join helps to retrieve data from two or more tables.

Ans. 1. b 2. d

Case-based Questions

1. Kirti studies in class 11 and is a sports enthusiast. To promote co-curricular activities and sports, her school conducts various competitions and events throughout the year. There are five houses, whose names have been inspired from the rivers of India—Ganga, Jamuna, Saraswati, Narmada, and Jhelum. Each student is allocated one house. In the upcoming Annual Athletic Meet in her school, various inter-house sports events will be held and she will be representing her house-Saraswati. Kirti's favourite subject is Computer Science and as she has just studied databases, she wants to maintain data of all athletes of her school in a table named `Athletes`. In another table, named `House`, she wants to store the house details.

She has created both the tables and has inserted records as shown below:

Table: `Athletes`

AthleteId	Name	Gender	Category	Event1	Event2	HouseId
A002	Anushka	Girl	Sub Junior	100m		H01
A003	Prakhar	Boy	Senior	High Jump Boys	200m	H02
A004	Piyush	Boy	Senior	Relay	100m	H02
A005	Nyasa	Girl	Junior	Long Jump	Relay	H03
A007	Hamdaan	Boy	Sub Junior	100m	400m	H01
A008	Gurpurab	Boy	Junior	Long Jump Boys		H04
A009	Manasvi	Girl	Junior	Relay		H05

Table : `House`

HouseID	HouseName	HouseMaster	HouseCaptain
H01	Ganga	Manju	Aahana
H02	Saraswati	Onkar	Kabir
H03	Jamuna	Kusum	Manasa
H04	Narmada	Sunil	Prabhjot
H05	Jhelum	Pratul	Soumya

Help Kirti to write the queries to perform the following tasks:

a. To display Athlete name and House name for all athletes.

b. To display the house name and house master's name for all athletes of category, Seniors.

c. To count the number of athletes for each category.

d. To display the Category, house name and house captain of athletes that are participating in 100m.

e. To display gender wise count of athletes.

f. To display cartesian product of the two tables.

Ans. a.
```
SELECT Name, HouseName
FROM Athlete A, House H
WHERE A.HouseId=H.HouseId;
```

b.
```
SELECT HouseName, HouseMaster
FROM Athlete A, House H
WHERE A.HouseId=H.HouseId
AND Category="Senior";
```

c.
```
SELECT Category, COUNT(*)
FROM Athlete
GROUP BY Category;
```

d. SELECT Category, HouseName, HouseCaptain
FROM Athlete A, House H
WHERE A.HouseId=H.HouseId
AND (Event1="100m" OR Event2="100m");

e. SELECT GENDER, COUNT(*)
FROM Athlete
GROUP BY GENDER;

f. SELECT * FROM Athlete, House;

2. Qala is a watch dealer and runs a store named 'Timepiece'. She wants to keep a record of all watches that are for sale in her shop. At the same time, she also wants to keep a record of all her customers. As a database expert, you have created two tables–Watches and Customers and have also inserted records as per the data provided by Qala.

Table: Watches

WatchId	WName	Price	Type	Brand	YOP
T001	Apple iWatch	56000	Smart watch	Apple	2020–12–12
T002	Titan Raga	25673	Designer	Titan	2015–04–15
T003	NeoSplash	34908	Sports Watch	Neo Sports	2018–11–09
T005	G-Shock	13892	Sports Watch	Casio	2021–03–18
T006	Max Pro	15673	Designer	Maxima	2014–06–03
T007	Zoop	8987	Kids	Titan	2019–03–18

Table: Timepiece_Customers

CID	CName	ConatctNo	WatchId
101	Sulekha	9090909090	T002
103	Jonathan	8888899999	T006
104	Gurvinder		T001
105	Pradipto	7676767634	T005

Qala wants to retrieve specific data from the database. As per the given tables, write queries for each of the following:

a. To display the Customer name and the watch name for all customers.

b. To display names of all customers who have purchased a watch of brand Apple.

c. To display the names of the brand in reverse alphabetical order.

d. To display the minimum price and latest purchase date of each brand.

e. To increase the price of each watch by 10%.

f. To delete the record of customer whose contact no. is not known.

Ans. a. SELECT CName, WName FROM Timepiece_Customers T, Watches W
-> WHERE T.WatchId=W.WatchId;

b. SELECT CName, Brand FROM Timepiece_Customers T, watches W
WHERE T.WatchId=W.WatchId
AND Brand="Apple";

c. SELECT Brand FROM Watches
ORDER BY Brand DESC;

d. SELECT MIN(Price),MAX(YOP) FROM Watches
GROUP BY Brand;

e. UPDATE Watches
SET Price=Price+(10/100*Price);

f. DELETE FROM Timepiece_Customers
WHERE ContactNo IS NULL;

3. Refer to question no 2. If the following queries are given, what will be the output?

a. SELECT WName, Price , CName FROM Timepiece_Customers T, watches W
WHERE T.WatchId=W.WatchId
AND price <50000;

b. SELECT CName, Price FROM Timepiece_Customers T, watches W
WHERE T.WatchId=W.WatchId
AND ContactNo IS NULL;

c. SELECT DISTINCT WName, CName FROM Timepiece_Customers T, watches W
WHERE T.WatchId=W.WatchId;

d. SELECT Type,MAX(Price) FROM watches
GROUP BY Type
HAVING TYPE!="Kids";

e. SELECT COUNT(*) FROM Timepiece_Customers;

Ans. a.

Wname	Price	CName
Titan Raga	25416	Sulekha
Max Pro	15516	Jonathan
G-Shock	13753	Pradipto

b.

Cname	Brand
Gurvinder	Apple

c.

Wname	CName
Titan Raga	Sulekha
Max Pro	Jonathan
Apple iWatch	Gurvinder
G-Shock	Pradipto

d.

Type	MAX(Price)
Smart Watch	58757
Designer	25416
Sports Watch	34559

e.

COUNT(*)
4

Assessment

A. Multiple Choice Questions

1. Which of the following is used to enforce integrity of a database?

a. Candidate key b. Alternate key c. Foreign key d. Primary key

2. Which of the following statements is true with respect to foreign key?

a. It should refer to primary key of another table b. It can be NULL

c. Both a and b d. Neither a nor b

3. While retrieving data from two tables, which of the following types of join removes the duplicate column that is common to both tables?

a. Equi join b. Natural join c. Same join d. Unique join

4. Which of the following statements is NOT true about equi joins?

a. The columns joining the tables must have the same name in both tables.

b. The common attribute values appear only once in the output.

c. Specific columns from joining tables are checked for equality.

d. Values from joining tables are retrieved if the condition in the WHERE clause is satisfied.

B. State whether the following statements are True or False:

1. A DBMS will reject any attempt to insert data in a table that violates the foreign key constraint. _____
2. A SELECT statement can be used to retrieve data from multiple tables. _____
3. Using a SELECT statement with multiple tables displays the cross-product of specified tables. _____
4. It is mandatory to use aliases with multiple tables. _____
5. In case of equi join, the column that is common to both tables is displayed only once. _____

C. Fill in the blanks.

1. To retrieve information from two or more tables, the tables are joined on one or more common _____.
2. Equi join condition checks for _____ of the value of the attribute common to both tables.
3. A _____ key constraint is used to retrieve data from multiple tables.
4. A foreign key linking multiple tables is a _____ key in at least one of the linked tables.

D. Answer the following questions:

1. Describe the notion of referential integrity constraints with the help of an example., clearing showing two tables and the foreign key involved. Write SQL statements to define the foreign key constraints.
2. What do you mean by natural join? Illustrate with the help of an example.
3. Give an example of an equi-join condition involving two tables.

E. Application Based Questions:

1. Using the tables `STUDENT` and `SUBJECT` from the `COLLEGE` database, write a **`SELECT`** statement for each of the following:
   a. For each student, retrieve his/her roll number and the name of the subject that the student is studying.
   b. For each student, retrieve his/her roll number, name, and department number of the department to which the student belongs.
   c. For each student, retrieve his/her roll number and student name for all the students who are studying Chemistry.
   d. List the names of students who are studying **`Physics`**.
   e. For each subject, retrieve the subject name and the number of students who have taken that subject.
   f. List subject name, student roll number, and student name for all the students and subjects.
2. Using the tables `DEPARTMENT` and `SUBJECT` from the `COLLEGE` database, write a **`SELECT`** statement for each of the following:
   a. For each department, list the department name and subjects offered by that department.
   b. For each department, list the department name and the number of subjects offered by that department.
   c. List names of the subjects and the department number of the associated department for the subjects for which teachers have been currently hired.
3. Write SQL queries for (a) to (d) and write outputs for SQL queries (i) and (ii) of part (e) which are based on the tables given below: (2019)

TRAINS

TNO	TNAME	START	END
11096	Ahimsa Express	Pune Junction	Ahmedabad Junction
12015	Ajmer Shatabdi	New Delhi	Ajmer Junction
1651	Pune Hbj Special	Pune Junction	Habibganj
13005	Amritsar Mail	Howrah Junction	Amritsar Junction
12002	Bhopal Shatabdi	New Delhi	Habibganj
12417	Prayag Raj Express	Allahabad Junction	New Delhi
14673	Shaheed Express	Jaynagar	Amritsar Junction
12314	Sealdah Rajdhani	New Delhi	Sealdah
12498	Shan-e-Punjab	Amritsar Junction	New Delhi
12451	Shram Shakti Express	Kanpur Central	New Delhi
12030	Swarna Shatabdi	Amritsar Junction	New Delhi

PASSENGERS

PNR	TNO	PNAME	GENDER	AGE	TRAVELDATE
P001	13005	R N AGRAWAL	MALE	45	2018-12-25
P002	12015	P TIWARY	MALE	28	2018-11-10
P003	12015	S TIWARY	FEMALE	22	2018-11-10
P004	12030	S K SAXENA	MALE	42	2018-10-12
P005	12030	S SAXENA	FEMALE	35	2018-10-12
P006	12030	P SAXENA	FEMALE	12	2018-10-12
P007	13005	N S SINGH	MALE	52	2018-05-09
P008	12030	J K SHARMA	MALE	65	2018-05-09
P009	12030	R SHARMA	FEMALE	58	2018-05-09

NOTE : All Dates are given in `'YYYY-MM-DD'` format.

a. To display details of all passengers travelling in train whose `TNO` is `12030`

b. To count the number of FEMALE passengers travelling in train whose `TNO` is `12030`

c. To display the names of the passengers along with the train names in which they are travelling.

d. To display the date of travel for the train, 'Ajmer Shatabdi'.

e. Write the output of the following queries:

(i)
```
SELECT TNAME, PNAME
FROM TRAINS T, PASSENGERS P
WHERE (T.TNO = P.TNO) AND (AGE BETWEEN 40 AND 50);
```

(ii)
```
SELECT P.TNO, TNAME, COUNT(*) AS "NoOfBookings"
FROM TRAINS T, PASSENGERS P
WHERE (T.TNO = P.TNO)
GROUP BY P.TNO, TNAME
HAVING COUNT(*)>1;
```

4. Consider a database having three tables: `SUPPLIER`, `PRODUCT`, and `SHIPMENT`. A supplier has the following attributes: supplier number (`SNo`), supplier name (`Sname`), and the location (`SCity`) of the supplier. Each product has a product name (`Pname`). Products of different brands are distinguished from each other and are assigned different product numbers (`PNo`). It is assumed that a product of a particular brand (`Brand`) is manufactured only in one city (`Pcity`). An order is identified by (`OrderNo`) and includes a certain quantity (`Quantity`) of only one product supplied by a particular supplier at a specific price (`Price`).

SUPPLIER

SNo	Sname	SCity
101	Aradhaya Pvt. L	Mumbai
102	XYZ Enterprises	Bangalore
103	Komal Enterpris	Delhi
104	Cloudtail	Patna
105	PQR Enterprises	Uttrakhand
106	Tech Solutions	Bangalore
107	XYZ Enterprises	Mumbai

PRODUCT

PNo	Pname	Brand	City
201	Refrigerator	Whirlpool	Chennai
202	Washing Machine	Samsung	Kolkata
203	Television	Sony	Mumbai
204	Television	Samsung	Kolkata
205	Washing Machine	Whirlpool	Chennai
206	Refrigerator	Whirlpool	Delhi
207	Microwave Oven	IFB	Mumbai
208	Microwave	Samsung	Mumbai
209	Air Fryer	IFB	Chandigarh
210	Air Fryer	Philips	Lucknow

SHIPMENT

OrderNo	SNo	PNo	Price	Quantity
1000	101	201	50000.00	50
1001	101	202	70000.00	30
1002	102	202	90000.00	75
1003	104	205	25000.00	20
1004	105	205	55000.00	30
1005	102	207	48000.00	50
1006	105	210	23000.00	30
1007	104	209	60000.00	75
1008	102	208	65000.00	20
1009	103	207	81000.00	30
1010	106	204	56000.00	30
1011	107	210	38000.00	50
1012	101	201	72000.00	50

Write a **SELECT** statement for the following queries:

(i) Queries involving tables **PRODUCT** and **SHIPMENT**

a. Retrieve the order number and name of the product supplied in each shipment.

b. Retrieve the order number of the products of the `Whirlpool` brand.

c. Retrieve the product names sold at a price greater than 70000.

d. For each product, retrieve the minimum price and the supplier number of the supplier supplying that product.

e. For each city, find the total quantity shipped from that city.

(ii) Queries involving tables **SUPPLIER** and **SHIPMENT**

a. Retrieve the supplier names for suppliers who supply the product with PNo `201`.

b. Retrieve the names of suppliers who have supplied more than two orders.

c. Retrieve the number of orders supplied by each supplier.

d. Retrieve the supplier names for suppliers who supply from `Mumbai`, and the order exceeds `50000`.

e. Retrieve the supplier names for suppliers who supply from `Mumbai` and supply quantity exceeding `40`.

f. Retrieve the details of shipments supplied by suppliers from `Karnataka`.



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion(A): A Cartesian Product is an unrestricted join.

Reasoning(R)**:** For cartesian product AxB, each row of Table A is joined with all rows of Table B.

2. Assertion(A): The result of an equi join depends on the condition of equality between common fields of two tables.

Reasoning(R): Without checking for the condition of equality, we will not get the desired output.



Case-based Questions

Aarushi owns a Performing Arts Studio in Delhi. In its two auditoriums, various theatre troupes stage their play/acts. She wants to maintain a database to store data of all the stage shows in the Theatre table given below.

Table: Theatre

TCODE	TITLE	AUDI	LANGUAGE	SHOWDATE	TICKET PRICE	TROUPID
T001	Akbar The Great Nahi Rahe	2	Hinglish	2023–05–12	150	R003
T002	Mughal-E-Azam	1	Hindi	2023–05–08	200	R001
T003	Romeo and Juliet	2	English	2023–05–09	125	R004
T004	The Lost Love	2	English	2023–05–10	100	R001
T005	Salaam Noni Appa	1	Hindi	2023–05–12	250	R002
T006	Gandhi The Untold Story	1	Hinglish	2023–07–13	375	R006

The details of each troupe is stored in the Troupe Table given below.

Table: `Troupe`

TROUPID	TNAME	CITY
R001	Nattak Mela	Nagpur
R002	Dhara	Delhi
R003	OnStage	Delhi
R004	Reflection	Mumbai
R005	Beyond the Skies	Jaipur
R006	Parivartan	Mumbai

As a DBA, help Aarushi to retrieve the following data from the given tables in a MySQL database.

a. To view Audi no. and Troupe Name from tables Theatre and Troupe.

b. To view troupe name, title of play and ticket price of the shows staged in Audi 1.

c. To view count of plays in each language.

d. To view the names of troupes in alphabetical order.

e. To view the names of the plays and the troupes that are from Delhi, in chronological order.

f. Aarushi wants to add another column to the table TROUPE with the following details:

Name of column—CONTACTNO

Data Type—string of size 10

Null Values allowed

Write the SQL statement to complete the task.



Previous Years' Questions

1. fetchall() method fetches all rows in a result set and returns a: [2023]

a. `Tuple of lists`

b. `List of tuples`

c. `List of strings`

d. `Tuple of strings`

Ans. b. `List of tuples`

2. a. Write the outputs of the SQL queries (i) to (iv) based on the relations COMPUTER and SALES given below: [2023]

Table: COMPUTER

PROD_ID	PROD_NAME	PRICE	COMPANY	TYPE
P001	MOUSE	200	LOGITECH	INPUT
P002	LASER PRINTER	4000	CANON	OUTPUT
P003	KEYBOARD	500	LOGITECH	INPUT
P004	JOYSTICK	1000	IBALL	INPUT
P005	SPEAKER	1200	CREATIVE	OUTPUT
P006	DESKJET PRINTER	4300	CANON	OUTPUT

Table : SALES

PROD_ID	QTY_SOLD	QUARTER
P002	4	1
P003	2	2
P001	3	2
P004	2	1

(i) `SELECT MIN (PRICE), MAX (PRICE) FROM COMPUTER;`

Ans.

MIN(PRICE)	MAX(PRICE)
200	4300

(ii) `SELECT COMPANY, COUNT (*) FROM COMPUTER GROUP BY COMPANY HAVING COUNT (COMPANY) > 1;`

Ans.

COMPANY	COUNT(*)
LOGITECH	2
CANON	2

(iii) `SELECT PROD_NAME, QTY_SOLD FROM COMPUTER C, SALES S WHERE C.PROD_ID=S.PROD_ID AND TYPE = 'INPUT';`

Ans.

PROD_NAME	QTY_SOLD
KEYBOARD	2
MOUSE	3
JOYSTICK	2

(iv) `SELECT PROD_NAME, COMPANY, QUARTER FROM COMPUTER C, SALES S WHERE C.PROD_ID=S.PROD_ID;`

Ans.

PROD_NAME	COMPANY	QUARTER
LASER PRINTER	CANON	1
KEYBOARD	LOGITECH	2
MOUSE	LOGITECH	2
JOYSTICK	IBALL	1

b. Write the command to view all databases.

Ans. show databases;

3. Write the output of SQL queries (a) and (b) based on the following two tables DOCTOR and PATIENT belonging to the same database: [2022]

Table: Doctor

DNO	DNAME	FEES
D1	Amitabh	1500
D2	Aniket	1000
D3	Nikhil	1500
D4	Anjana	1500

Table: Patient

PNO	PNAME	ADMDATE	DNO
P1	Noor	2021–12–25	D1
P2	Annie	2021–11–20	D2
P3	Prakash	2020–12–10	NULL
P4	Harmeet	2019–12–20	D1

(i) `SELECT DNAME, PNAME FROM DOCTOR NATURAL JOIN PATIENT ;`

Ans.

DNAME	PNAME
AMITABH	NOOR
ANIKET	ANNIE
AMITABH	HARMEET

(ii) `SELECT PNAME, ADMDATE, FEES FROM PATIENT P, DOCTOR D WHERE D. DNO = P.DNO AND FEES >1000;`

Ans.

PNAME	ADMDATE	FEES
NOOR	2021–12–25	1500
HARMEET	2019–12–20	1500

4. Write SQL queries for (a) to (d) based on the tables PASSENGER and FLIGHT given below: [2022]

Table: Passenger

PNO	PNAME	ADMDATE	FNO
1001	Suresh	Male	F101
1002	Anita	Female	F104
1003	Harjas	Male	F102
1004	Nita	Female	F103

Table: Flight

FNO	START	END	F_DATE	FARE
F101	MUMBAI	CHENNAI	2021–12–25	4500
F102	Mumbai	BENGALURU	2021–11–20	4000
F103	Delhi	Chennai	2021–12–10	5500
F104	Kolkata	MUMBAI	2021–12–20	4500
F105	DELHI	BENGALURU	2021–01–15	5000

(i) Write a query to change the fare to 6000 of the flight whose FNO is F104.

Ans.
```
UPDATE Flight SET FARE=6000 WHERE FNO='F104';
```

(ii) Write a query to display the total number of MALE and FEMALE PASSENGERS.

Ans.
```
SELECT GENDER, COUNT(*) FROM Passenger GROUP BY GENDER;
```

(iii) Write a query to display the NAME, corresponding FARE and F_DATE of all PASSENGERS who have a flight to START from DELHI.

Ans.
```
SELECT NAME, FARE, F_DATE FROM Passenger p, Flight f WHERE P.FNO=F.FNO AND START='DELHI';
```

(iv) Write a query to delete the records of flights which end at Mumbai.

Ans.
```
DELETE FROM Flight WHERE END='MUMBAI';
```

NCERT Exercise Solutions

1. Answer the following questions:

a. Define RDBMS. Name any two RDBMS software.

b. What is the purpose of the following clauses in a select statement?

i. ORDER BY

ii. HAVING

c. Site any two differences between Single_row functions and Aggregate functions.

d. What do you understand by Cartesian Product?

e. Write the name of the functions to perform the following operations:

i. To display the day like "Monday", "Tuesday", from the date when India got independence.

ii. To display the specified number of characters from a particular position of the given string.

iii. To display the name of the month in which you were born.

iv. To display your name in capital letters.

Ans. a. A DBMS that is used to manage Relational Databases is called RDBMS (Relational Database Management System). Some popular RDBMS software available are: Oracle, MySQL.

b. i. ORDER BY clause in SQL is used to sort the result set of a query based on one or more columns.

ii. The HAVING clause is used in SQL queries to filter the results of a GROUP BY clause based on a specified condition.

c. i. Single row functions operate on a single row from the table at a time whereas aggregate functions operate on a set of rows at a time.

ii. Single row function display output for single row and display number rows whereas aggregate functions display a single value from a set of rows.

d. In a relational database management system (RDBMS), the Cartesian Product is a mathematical operation that combines every row from one table with every row from another table. The result is a new table where each row represents a combination of rows from the original tables. The size of the Cartesian Product is the product of the number of rows in each table. It is denoted by the symbol "×" or by using the CROSS JOIN operation in SQL.

e. i.
```
SELECT DAYNAME('1947/08/15');
```
```
+-----------------------+
| DAYNAME('1947/08/15') |
+-----------------------+
| Friday                |
+-----------------------+
```

ii.
```
SELECT SUBSTRING('Hello, World!', 1, 5) AS result;
```
```
+--------+
| result |
+--------+
| Hello  |
+--------+
```

iii.
```
SELECT MONTHNAME('2019-09-24') AS birth_month;
```
```
+-------------+
| birth_month |
+-------------+
| September   |
+-------------+
```

iv.
```
SELECT UPPER('ankit') as nameUpperCase;
```
```
+---------------+
| nameUpperCase |
+---------------+
| ANKIT         |
+---------------+
```

2. Write the output produced by the following SQL commands:

a. `SELECT POW(2,3);`

b. `SELECT ROUND(123.2345, 2), ROUND(342.9234,-1);`

c. `SELECT LENGTH("Informatics Practices");`

d. `SELECT YEAR("1979/11/26"), MONTH("1979/11/26"), DAY("1979/11/26"), MONTHNAME("1979/11/26");`

e. `SELECT LEFT("INDIA",3), RIGHT("Computer Science",4);`

f. `SELECT MID("Informatics",3,4),SUBSTR("Practices",3);`

Ans. a. 8

b. 123.23, 340

c. 21

d. 1979 11 26 November

e. IND, ence

f. form, actices

3. Consider the following MOVIE table and write the SQL queries based on it.

MovieID	MovieName	Category	Release Date	ProductionCost	BusinessCost
001	Hindi_Movie	Musical	2018-04-23	124500	130000
002	Tamil_Movie	Action	2016-05-17	112000	118000
003	English_Movie	Horror	2017-08-06	245000	360000
004	Bengali_Movie	Adventure	2017-01-04	72000	100000
005	Telugu_Movie	Action	–	100000	–
006	Punjabi_Movie	Comedy	–	30500	–

a. Display all the information from the Movie table.

b. List business done by the movies showing only MovieID, MovieName and Total_Earning. Total_ Earning to be calculated as the sum of ProductionCost and BusinessCost.

c. List the different categories of movies.

d. Find the net profit of each movie showing its MovieID, MovieName and NetProfit. Net Profit is to be calculated as the difference between Business Cost and Production Cost.

e. List MovieID, MovieName and Cost for all movies with ProductionCost greater than 10,000 and less than 1,00,000.

f. List details of all movies which fall in the category of comedy or action.

g. List details of all movies which have not been released yet.

Ans. a.
```
SELECT * FROM movie;
```

b.
```
SELECT MovieID, MovieName, ProductionCost+BussinessCost AS 'Total_ Earnings' FROM movie;
```

c.
```
SELECT DISTINCT(Category) FROM movie;
```

d.
```
SELECT MovieID, MovieName, (BusinessCost – ProductionCost) AS "Netprofit" FROM movie;
```

Note: If movie is not released than our query return negative values .

e.
```
SELECT MovieID, MovieName, ProductionCost
FROM movie
WHERE ProductionCost > 10000 AND ProductionCost < 100000;
```

f.
```
SELECT * FROM movie WHERE category IN ("Comedy","Action");
```

g.
```
SELECT * FROM movie WHERE ReleasedDate IS NULL;
```

4. Suppose your school management has decided to conduct cricket matches between students of Class XI and Class XII. Students of each class are asked to join any one of the four teams—Team Titan, Team Rockers, Team Magnet and Team Hurricane. During summer vacations, various matches will be conducted between these teams. Help your sports teacher to do the following:

a. Create a database "Sports".

b. Create a table "TEAM" with following considerations:

i. It should have a column TeamID for storing an integer value between 1 to 9, which refers to unique identification of a team.

ii. Each TeamID should have its associated name (TeamName), which should be a string of length not less than 10 characters.

c. Using table level constraint, make TeamID as the primary key.

d. Show the structure of the table TEAM using a SQL statement.

e. As per the preferences of the students four teams were formed as given below. Insert these four rows in TEAM table:

Row 1: (1, Team Titan)

Row 2: (2, Team Rockers)

Row 3: (3, Team Magnet)

Row 3: (4, Team Hurricane)

f. Show the contents of the table TEAM using a DML statement.

g. Now create another table MATCH_DETAILS and insert data as shown below. Choose appropriate data types and constraints for each attribute.

Table: Match_Details

MatchID	MatchDate	FirstTeamID	SecondTeamID	FirstTeamScore	SecondTeamScore
M1	2018-07-17	1	2	90	86
M2	2018-07-18	3	4	45	48
M3	2018-07-19	1	3	78	56
M4	2018-07-19	2	4	56	67
M5	2018-07-18	1	4	32	87
M6	2018-07-17	2	3	67	51

Ans. a. `CREATE DATABASE Sports ;`

b. `CREATE TABLE Team (TeamID INT CHECK (TeamID BETWEEN 1 AND 9), TeamName VARCHAR(30) CHECK (LENGTH(TeamName)>=10));`

c. `CREATE TABLE Team (TeamID INT CHECK (TeamID BETWEEN 1 AND 9), TeamName VARCHAR(30) CHECK (LENGTH(TeamName)>=10), PRIMARY KEY(TeamID));`

d. `DESC Team ;`

e. `INSERT INTO Team VALUES (1, 'Team Titan') ; INSERT INTO Team VALUES (2, 'Team Rockers') ; INSERT INTO Team VALUES (3, 'Team Magnet') ; INSERT INTO Team VALUES (4, 'Team Hurricane') ;`

f. `SELECT * FROM Team ;`

g.
```
CREATE TABLE Match_Details
(MatchID CHAR(2) NOT NULL PRIMARY KEY,
MatchDate DATE,
FirstTeamID INT,
SecondTeamID INT,
FirstTeamScore INT,
SecondTeamScore INT) ;

INSERT INTO Match_details VALUES("M1", '2018-07-17', 1, 2, 90, 86);
INSERT INTO Match_details VALUES("M2", '2018-07-18', 3, 4, 45, 48);
INSERT INTO Match_details VALUES("M3", '2018-07-19', 1, 3, 78, 56);
INSERT INTO Match_details VALUES("M4", '2018-07-19', 2, 4, 56, 67);
INSERT INTO Match_details VALUES("M5", '2018-07-20', 1, 4, 32, 87);
insert into Match_details values("M6", '2018-07-21', 2, 3, 67, 51);

ALTER TABLE Match_details
ADD FOREIGN KEY (FirstTeamID) REFERENCES Team(TeamID);
ALTER TABLE Match_details
ADD FOREIGN KEY (SecondTeamID) REFERENCES Team(TeamID);
```

5. Using the sports database containing two relations (TEAM, MATCH_DETAILS) and write the queries for the following:

a. Display the MatchID of all those matches where both the teams have scored more than 70.

b. Display the MatchID of all those matches where FirstTeam has scored less than 70 but SecondTeam has scored more than 70.

c. Display the MatchID and date of matches played by Team 1 and won by it.

d. Display the MatchID of matches played by Team 2 and not won by it.

e. Change the name of the relation TEAM to T_DATA. Also change the attributes TeamID and TeamName to T_ID and T_NAME respectively.

Ans. a. `SELECT MatchID FROM MATCH_DETAILS WHERE FirstTeamScore > 70 and SecondTeamScore > 70;`

b. `SELECT MatchID FROM MATCH_DETAILS WHERE FirstTeamScore < 70 and SecondTeamScore > 70;`

c. `SELECT MatchID , MatchDate FROM MATCH_DETAILS WHERE FirstTeamID = 1 and FirstTeamScore > SecondTeamScore;`

d. `SELECT MatchID , MatchDate FROM MATCH_DETAILS WHERE FirstTeamID = 2 and FirstTeamScore < SecondTeamScore;`

OR

SELECT MatchID , MatchDate FROM MATCH_DETAILS WHERE SecondTeamID = 2 and FirstTeamScore > SecondTeamScore;

e.
```
ALTER TABLE Team RENAME TO T_DATA;
ALTER TABLE T_DATA CHANGE COLUMN TeamID T_ID;
ALTER TABLE T_DATA CHANGE COLUMN TeamName T_Name;
```

6. Consider the following table named "Product", showing details of products being sold in a grocery shop.

PCode	PName	UPrice	Manufacture
P01	Washing Powder	120	Surf
P02	Tooth Paste	54	Colgate
P03	Soap	25	Lux
P04	Tooth Paste	65	Pepsodent
P05	Soap	38	Dove
P06	Shampoo	245	Dove

Write SQL queries for the following:

a. Create the table Product with appropriate data types and constraints.

b. Identify the primary key in Product.

c. List the Product Code, Product name and price in descending order of their product name. If PName is the same then display the data in ascending order of price.

d. Add a new column Discount to the table Product.

e. Calculate the value of the discount in the table Product as 10 per cent of the UPrice for all those products where the UPrice is more than 100, otherwise, the discount will be 0.

f. Increase the price by 12 per cent for all the products manufactured by Dove.

g. Display the total number of products manufactured by each manufacturer.

h. Write the output(s) produced by executing the following queries on the basis of the information given above in the table Product:

i. `SELECT PName, AVG(UPrice) FROM Product GROUP BY Pname;`

ii. `SELECT DISTINCT Manufacturer FROM Product;`

iii. `SELECT COUNT(DISTINCT PName) FROM Notes Product;`

iv. `SELECT PName, MAX(UPrice), MIN(UPrice) FROM Product GROUP BY PName;`

Ans. a.
```
CREATE TABLE product
(PCode  CHAR(3) PRIMARY KEY,
Pname  VARCHAR(25)  NOT NULL,
Uprice DECIMAL(10,2),
Manufacturer VARCHAR(30));
```

b. PCode

c. `SELECT pcode,pname,uprice FROM product ORDER BY pname DESC,uprice ASC;`

d. `ALTER TABLE product ADD discount DECIMAL(8,2);`

e. `UPDATE product SET discount=0.10*uprice WHERE uprice >100;`

f. `UPDATE product SET uprice = uprice+0.12*uprice WHERE manufacturer="Dove";`

g. `SELECT manufacturer, COUNT(*) FROM product GROUP BY manufacturer;`

i.
```
+------------------+---------------+
|  Manufacturer    |  Avg(UPrice)  |
+------------------+---------------+
|  Washing Powder  |  120.000000   |
|  Tooth Paste     |   59.500000   |
|  Soap            |   33.780000   |
|  Shampoo         |  274.400000   |
+------------------+---------------+
```

ii.
```
+-----------------------+
|  Manufacturer         |
+-----------------------+
|  Surf                 |
|  Colgate              |
|  Lux                  |
|  Pepsodant            |
|  Dove                 |
+-----------------------+
```

iii.
```
+-----------------------+
|  COUNT(DISTINCT PName)|
+-----------------------+
|  4                    |
+-----------------------+
```

iv.
```
+------------------+------------------+------------------+
|  PName           |  MAX(UPrice)     |  MIN(UPrice)     |
+------------------+------------------+------------------+
|  Washing Powder  |  120             |  120             |
|  Tooth Paste     |  65              |  54              |
|  Soap            |  43              |  25              |
|  Shampoo         |  274             |  274             |
+------------------+------------------+------------------+
```

7. A shop called Wonderful Garments who sells school uniforms maintains a database SCHOOLUNIFORM as shown below. It consisted of two relations—UNIFORM and COST. They made UniformCode as the primary key for UNIFORM relations. Further, they used UniformCode and Size to be composite keys for COST relation. By analysing the database schema and database state, specify SQL queries to rectify the following anomalies.

a. M/S Wonderful Garments also keeps handkerchiefs of red colour, medium size of Rs. 100 each.

b. INSERT INTO COST (UCode, Size, Price) values (7, 'M',100);

When the above query is used to insert data, the values for the handkerchief without entering its details in the UNIFORM relation is entered. Make a provision so that the data can be entered in the COST table only if it is already there in the UNIFORM table.

c. Further, they should be able to assign a new UCode to an item only if it has a valid UName. Write a query to add appropriate constraints to the SCHOOLUNIFORM database.

d. Add the constraint so that the price of an item is always greater than zero.

Ans. a. `INSERT INTO UNIFORM (UCode, UName, Color, Size) VALUES(7, 'HANDKERCHIEF', 'RED', 'M');`

This query will insert a new row in the UNIFORM table for Red HANDKERCHIEF.

b. `ALTER TABLE COST ADD FOREIGN KEY(UCode) REFERENCES UNIFORM(UCode);`

This query will add foreign key constraint to the COST table which is referencing the UNIFORM table, so if someone tries to add a uniform's cost without having information for that uniform in the UNIFORM table will fail.

c. `ALTER TABLE UNIFORM FOREIGN KEY (UName) REFERENCES COST(UName);`

This constraint ensures that values in UName column of UNIFORM must exist in the UName column of COST.

d. `ALTER TABLE COST ADD CHECK (Price > 0);`

The CHECK constraint will make sure that the price of an item is always greater than zero.

8. Using the CARSHOWROOM database given below, write the SQL queries for the following:

Table: INVENTORY

CarId	CarName	Price	Model	YearManufacture	Fueltype
D001	Dzire	582613.00	LXI	2017	Petrol
D002	Dzire	673112.00	VXI	2018	Petrol
B001	Baleno	567031.00	Sigma1.2	2019	Petrol
B002	Baleno	647858.00	Delta1.2	2018	Petrol
E001	EECO	355205.00	5 STR STD	2017	CNG
E002	EECO	654914.00	CARE	2018	CNG
S001	SWIFT	514000.00	LXI	2017	Petrol
S002	SWIFT	614000.00	VXI	2018	Petrol

Table: CUSTOMER

CustId	CustName	CustAdd	Phone	Email
C0001	Amit Saha	L-10, Pitampura	4564587852	amitsaha2@gmail.com
C0002	Rehnuma	J-12, SAKET	5527688761	rehnuma@hotmail.com
C0003	Charvi Nayyar	10/9, FF, Rohini	6811635425	charvi123@yahoo.com
C0004	Gurpreet	A-10/2, SF, Mayur Vihar	3511056125	gur_singh@yahoo.com

Table: SALE

InvoiceNo	CarId	CustId	SaleDate	PaymentMode	EmpID	SalePrice
100001	D001	C0001	2019-01-24	Credit Card	E004	613248.00
100002	S001	C0002	2018-12-12	Online	E001	590321.00
100003	S002	C0004	2019-01-25	Cheque	E010	604000.00
100004	D002	C0001	2018-10-15	Bank Finance	E007	659982.00
100005	E001	C0003	2018-12-20	Credit Card	E002	369310.00
100006	S002	C0002	2019-01-30	Bank Finance	E007	620214.00

Table: EMPLOYEE

EmpID	EmpName	DOB	DOJ	Designation	Salary
E001	Rushil	1994-07-10	2017-12-12	Salesman	25550
E002	Sanjay	1990-03-12	2016-06-05	Salesman	33100
E003	Zohar	1975-08-30	2016-06-05	Peon	20000
E004	Arpit	1989-06-06	2010-12-02	Salesman	39100
E006	Sanjucta	1985-11-03	2012-07-01	Receptionist	27350
E007	Mayank	1993-04-03	2017-01-01	Salesman	27352
E010	Rajkumar	1987-02-26	2013-10-23	Salesman	31111

a. Add a new column Discount in the INVENTORY table.

b. Set appropriate discount values for all cars keeping in mind the following:

   i. No discount is available on the LXI model.

   ii. VXI model gives a 10% discount.

   iii. A 12% discount is given on cars other than LXI model and VXI model.

c. Display the name of the costliest car with fuel type "Petrol".

d. Calculate the average discount and total discount available on Car4.

e. List the total number of cars having no discount.

Ans. a. `ALTER TABLE Inventory ADD Discount DECIMAL(10,2);`

b. i. `UPDATE Inventory SET Discount=0 WHERE model="LXI";`

ii. `UPDATE Inventory SET Discount=0.10*price WHERE model="VXI";`

iii. `UPDATE Inventory SET Discount=0.12*price WHERE model NOT IN ("LXI","VXI");`

c.
```
SELECT CarName FROM Inventory WHERE Price = (SELECT MAX(Price)FROM Inventory
WHERE FuelType="petrol";
```

d. `SELECT AVG(Discount),SUM(Discount) FROM Inventory WHERE carname="car4";`

e. `SELECT COUNT(*) FROM Inventory WHERE Discount=0;`

Answers

Multiple Choice Questions

1. (d)	2. (a)	3. (d)	4. (a)	5. (c)	6. (c)	7. (a)	8. (c)
9. (d)	10. (d)						

True or False

1. (F)	2. (F)	3. (T)	4. (T)	5. (F)

Fill in the blanks

1. Foreign	2. x * y	3. referential integrity	4. FROM	5. NATURAL JOIN

12 PYTHON INTERFACE WITH MYSQL

Chapter Outline

12.1 Connecting to MySQL Server from Python
12.2 A Complete Example
12.3 Suggestive Case Study
12.4 Database Update with Inputs from a CSV File
12.5 Verifying Database Updates

Introduction

In the previous chapters, we studied how to create a database schema. We also learnt how to store, update and retrieve information in a database. We used SQL to carry out these operations interactively. This approach works fine when we are dealing with small databases. Now suppose a company has ten thousand employees and we need to update the salary of all the employees. We will have to execute ten thousand database operations to accomplish this task. Interactively executing ten thousand database operations will be monotonous, error-prone and wasteful of human resources. In this chapter, we will study how Python interfaces with SQL to carry out repetitive operations automatically using a Python program. For this purpose, we will establish the Python program's connection with the database server in MySQL.

12.1 Connecting to MySQL Server from Python

To access the MySQL database from Python, we need to ensure that the MySQL server runs at the backend. For connecting to MySQL server, we will use Python MySQL client library **`pymysql`**. It can be downloaded by executing the following command from Windows® command prompt:

```
python3 -m pip install PyMySQL
```

Once `pymysql` is installed, we import the `pymysql` module that provides Python interface to a MySQL database.

Another Python library that can be used for connectivity with databases is mysql.connector

`pymysql`: Python MySQL client for connecting to MYSQL server from Python.

12.1.1 Establishing Connection with a Database

Next, we use **connect()** method of the library to create a connection with an already existing database. The **connect()** method takes four arguments and returns a connection object. The syntax of the connect method is given below:

<connectionObject>**=pymysql.connect(host=**<hostName>**,user=**<userName>**,**
password=<password>**, db=**<databaseName>**)**

<hostName>: MySQL server hostname or the IP address
<userName>: User name
<password>: password associated with the user
<databaseName>: Name of the MySQL database with which connection is sought



connect(): Establishes connection with an existing database.

The following code snippet can be used for connecting to the **COMPANY** database.

```
>>> import pymysql
>>> cnxn = pymysql.connect(host="localhost", user="root", password="", db= "COMPANY")
```

If you try to access a database that does not exist, Python will flag a `pymysql` operational error with the message **Unknown database**.

12.1.2 Creating a connection object

Once a connection object (**cnxn**) is created, the class **cursor() method** of the connection object is used to create a handler/cursor object (**company** in this case) as follows:

```
>>> company = cnxn.cursor()
```



connect.cursor(): creates a handler of connection object for performing operations on the database.

12.1.3 Closing Database Connection

Once all the required database operations have been carried out, access to the database is no longer required. So, we close the database connection using method **close()** associated with the connection object `cnxn`, for example,

```
>>> cnxn.close()
```



connect.close(): closes the connection.

12.1.4 Executing SQL Queries

Method **execute()** of the cursor object is used to execute the SQL operations. Thus, to execute SQL operations on **COMPANY** database, we will use the method **company.execute()**. The **execute()** method typically accepts an SQL command string as an argument.

- In program 12.1, we intend to create EMPLOYEE table and populate it by inserting three rows.
- Thus, we first create a table EMPLOYEE in the database and execute the query using **execute()** method of cursor object company (see Program 12.1, lines 5–14).
- Note that the structure of the table EMPLOYEE is different from the one used in the previous chapter. We have done this intentionally to keep the illustrations simple.
- Next, we insert details about three employees in the table using the `INSERT` query (lines 16–18).

cursor.execute() : executes the query.

12.1.5 Making the Changes in a Database Permanent

- Although we have inserted three tuples into the table in lines 16–18, these changes are not yet permanent.
- To reflect the changes in the database, we need to commit the changes using method `commit()` of the object `cnxn` (line 21). This ensures that the database updates become permanent. Updates are now visible to all users.
- Finally, the connection is closed (line 23).

The state of the table `EMPLOYEE` after inserting three tuples in it and executing the **`commit()`** operation is shown in Table 12.1.

Aadhaar	Name	Gender	Salary	Dept_No
123456789012	Raj	M	10000	2
234567890123	Dhiraj	M	85000	1
345678901234	Muskan	F	100000	2

Table 12.1: State of table `EMPLOYEE` after inserting the rows

connect.commit(): commits changes to the database.

Program 12.1 Creating table `EMPLOYEE` and inserting the rows.

```
01 import pymysql
02 cnxn = pymysql.connect(host="localhost",user="root",password="",db= "COMPANY")
03 company = cnxn.cursor()
04 #Creating tables
05 company.execute("""CREATE TABLE EMPLOYEE (
06 Aadhaar CHAR(12) PRIMARY KEY,
07 Name VARCHAR(20) NOT NULL,
08 Gender CHAR(1) NOT NULL,
09 Salary INT NOT NULL,
10 Dept_No SMALLINT,
11 # Validation constraints
12 CHECK(Gender = 'M' OR Gender = 'F' OR Gender = 'O'),
13 CHECK(Salary BETWEEN 8000 AND 100000)
14 );""")
15 #Inserting data into table
16 company.execute("INSERT INTO EMPLOYEE VALUES('123456789012', 'Raj','M',10000,2);")
17 company.execute("INSERT INTO EMPLOYEE VALUES('234567890123','Dhiraj','M',85000,1);")
18 company.execute("INSERT INTO EMPLOYEE VALUES ('345678901234','Muskan','F',100000,2);")
19
20 #To commit changes to the database
21 cnxn.commit()
22 #Closing connection
23 cnxn.close()
```

12.1.6 Using Placeholder for Attribute Values

For reasons of simplicity, we have hardcoded the values of row attributes in the above example. But this defeats the very purpose of using a Python interface to SQL because if we were to hardcode the attribute values for each tuple, we might as well have used SQL statements interactively.

So, we replace the statement:

company.execute("INSERT INTO EMPLOYEE VALUES('123456789012', 'Raj','M',10000,2);")

by the following code segment:

```
#Using placeholder inserting user-provided attribute values into the row
print("Enter the Employee details:")
aadhaar = input("ID: ")
name = input("Name: ")
gender = input("Gender: ")
salary = int(input("Salary: "))
dno = int(input("Department Number: "))
company.execute("INSERT INTO EMPLOYEE VALUES(%s, %s, %s, %s, %s);",\
(aadhaar, name, gender, salary, dno))
```

We take the value of attributes for the row to be inserted as an input from the user. For this purpose, we use the placeholder **%s** for each of the attribute values in the tuple. To insert a row for an employee, we first take the `Aadhaar`, `Name`, `Gender`, `Salary`, and `Dept_No` as an input from the user. Next, we execute the **INSERT** operation. Note that **INSERT** operation requires a sequence of placeholders: **(%s, %s, %s, %s, %s)** and the list of data objects **(aadhaar, name, gender, salary, dno)**. Here, **%s** denotes that the user input format is a stream of bytes. Note that the attribute names **aadhaar, name, gender, salary, dno** do not conflict with the corresponding attribute names defined in the table **EMPLOYEE**, so there is no need to invent new data names.



%s, %{}s: placeholder for attribute values



Specifying Attribute Values in Arbitrary Order

We may specify the attributes in any order using a dictionary of *key:value* pairs as shown in the following code segment:

```
>>> #Using placeholder for inserting user-provided attribute values into the row
>>> print("Enter the Employee details:")
>>> empAadhaar = input("ID: ")
>>> empName = input("Name: ")
>>> empGender = input("Gender: ")
>>> empSalary = int(input("Salary: "))
>>> empDno = int(input("Department Number: "))
>>> company.execute("INSERT INTO EMPLOYEE VALUES(%(aadhaar)s, %(name)s,\
>>> %(gender)s, %(salary)s, %(dno)s);",\
>>> { 'name':empName,  'aadhaar':empAadhaar, 'salary':empSalary ,\
>>> 'dno':empDno,  'gender':empGender  })
```

Although we have used the identifiers `empAadhaar`, `empName`, `empGender`, `empSalaray`, and `empDno`, we could have used the same identifiers as the keys used in the dictionary. Thus, it would have been perfectly fine to use the following code instead of the one provided above:

12.1.7 Constructing an SQL Query String Using `format()` Method

We may also use the **format()** method to provide attribute values in a tuple. The substitutions in the string are specified using curly braces `'{'` and `'}'`, and the corresponding values to be substituted are passed as the comma-separated list of arguments. The resultant string also referred as formatted string can then be used as a query to be executed by **execute()** method as shown in following code snippet:

```
#Using placeholder for inserting user-provided attribute values into the row
print("Enter the Employee details:")
aadhaar = input("ID: ")
name = input("Name: ")
gender = input("Gender: ")
salary = int(input("Salary: "))
dno = int(input("Department Number: "))
```

```
query = "INSERT INTO EMPLOYEE VALUES('{}', '{}', '{}', {}, {});"\
.format(aadhaar, name, gender, salary, dno)
company.execute(query)
```

On providing **456789012345**, **Savita**, **F**, **80000**, and 1 as the input values for variables **aadhaar**, **name**, **gender**, **salary**, and **dno**, the **format()** method returns the following string object, named **query:**

```
INSERT INTO EMPLOYEE VALUES('456789012345', 'Savita', 'F', 80000, 1);
```

1. The default username for MySQL database is _____.
2. _____ is the server name on which MySQL runs.

12.1.8 Fetching Rows from the Result of Executing an SQL Query

Consider the current state of table EMPLOYEE as shown in Table 12.2:

Aadhaar	Name	Gender	Salary	Dept_No
123456789012	Raj	M	10000	2
234567890123	Dhiraj	M	85000	1
345678901234	Muskan	F	100000	2
456789012345	Savita	F	80000	1
567890123456	Taran	M	99000	2

Table 11:2: Current state of table EMPLOYEE

Suppose we wish to display all the rows in the table EMPLOYEE using the following **SELECT** query:

company.execute("Select * from EMPLOYEE ")

We may access the set of rows returned by the above query in any of the following manners: all rows at a time, a few (say, n) rows at a time, or one row at a time using one of the following methods supported by cursor class:

1. fetchall(): Retrieves all the rows from the result of execution of an SQL query, in the form of a tuple of tuples.
2. fetchmany(n): Retrieves n rows from the result of execution of an SQL query, in the form of a tuple of tuples.
3. fetchone(): Retrieves single row from the result of execution of an SQL query, in the form of a tuple.

Fetching all Rows

- In the program 12.2, we retrieve all the rows from the result of executing an SQL query in the form of a tuple of tuples (also known as the result set) using fetchall() method of cursor object company (line 8).
- For accessing the database, we first open the connection and once a connection object (cnxn) is created, the class cursor()method of the connection object is used to create a cursor object (company in this case) in lines 2–3.
- The data member rowcount (line 6) of cursor object (company in this case) yields the number of rows retrieved from the company database.

Program 12.2 Retrieving all rows from the table EMPLOYEE.

```
01 import pymysql
02 cnxn = pymysql.connect(host="localhost",user="root",password="",db= "COMPANY")
03 company = cnxn.cursor()
04 #Retrieving data from database
05 company.execute("Select * from EMPLOYEE ")
06 print("Number of rows retrieved: ", company.rowcount)
07 print("All Rows retrieved")
08 tuples = company.fetchall()
09 for row in tuples:
```

```
10      print(row)
11  #Closing connection
12  cnxn.close()
```



cursor.rowcount: yields the number of rows retrieved on executing an SQL query.
cursor.fetchall(): retrieves all the rows from the result of executing a SQL query in the form of a tuple of tuples.

On executing the above code-snippet, Python responds with the following output:

```
Number of rows retrieved:  5
All Rows retrieved
('123456789012', 'Raj', 'M', 10000, 2)
('234567890123', 'Dhiraj', 'M', 85000, 1)
('345678901234', 'Muskan', 'F', 100000, 2)
('456789012345', 'Savita', 'F', 80000, 1)
('567890123456', 'Taran', 'M', 99000, 2)
```

Note that every row of the table is returned as a tuple with comma-separated attribute values.

Fetching Selected Number of Rows

Suppose you now wish to retrieve the required number of **n** rows from the result of executing a query in the form of a tuple of tuples. The **cursor** object supports the method **fetchmany(n),** taking n as the input argument. To fetch two rows from the table EMPLOYEE, we replace lines 4–10 of Program 12.2 with the following code snippet. We have deliberately avoided saying the *first two rows* as SQL does not guarantee any specific ordering of tuples unless **ORDER BY** clause is applied on a set of rows:

```
#Retrieving data from database
company.execute("Select * from EMPLOYEE ")
print("Two rows retrieved")
tuples = company.fetchmany(2)
print("\nTuples:", tuples)
```



cursor.fetchmany(n): retrieves n rows from the result of executing a SQL query in the form of a tuple.

On executing the above code-snippet, Python responds with the following output:

```
Two rows retrieved
Tuples: (('123456789012', 'Raj', 'M', 10000, 2), ('234567890123', 'Dhiraj', 'M',
85000, 1))
```

Fetching a Single Row

The set of rows returned by a **SELECT** query may also be accessed one at a time. In the following code snippet, we invoke **fetchone()** method twice to retrieve two rows from the set of rows returned on executing the following query:

Select * from EMPLOYEE

```
#Retrieving data from database
company.execute("Select * from EMPLOYEE ")
print("Two rows retrieved")
row = company.fetchone()
print(row)
row = company.fetchone()
print(row)
```



cursor.fetchone(): retrieves single row from the result of executing a SQL query in the form of a tuple

On executing the above code segment, Python responded with the following output:

```
Two rows retrieved
('123456789012', 'Raj', 'M', 10000, 2)
('234567890123', 'Dhiraj', 'M', 85000, 1)
```

Suppose you wish to display those rows from table EMPLOYEE which satisfy a certain test condition, say all male or female employees. In this regard, we may take gender as an input from the user and substitute it in the query using placeholder notation as shown in the following code snippet:

```
#Retrieving data from database
gender = input("""Input M if you want to retrieve information about male employees
else F for female employees: """)
company.execute("Select * from EMPLOYEE WHERE Gender=%s;",gender)
print("Details of employees based on gender provided as input")
tuples = company.fetchall()
print("fetchall", type(tuples))
for row in tuples:
    print(row)
```

On executing the above code-snippet, Python responds with the following output:

```
Input M if you want to retrieve information about male employees
else F for female employees: F
Details of employees based on gender provided as input
('345678901234', 'Muskan', 'F', 100000, 2)
('456789012345', 'Savita', 'F', 80000, 1)
```

Updating the Details of Selected Employees

Suppose, the current state of the EMPLOYEE table is as shown in Table 12.3

Aadhaar	Name	Gender	Salary	Dept_No
123456789012	Raj	M	10000	2
234567890123	Dhiraj	M	85000	1
345678901234	Muskan	F	100000	2
456789012345	Savita	F	80000	1
567890123456	Taran	M	99000	2

Table 12.3: State of EMPLOYEE table before carrying out update operation

Next, suppose we want to update the department number of an employee based on his/her Aadhaar ID. The following code snippet takes an employee's aadhaar and new department number as input and updates his/her details in the database:

```
#Updating Department Number
aadhaar = input("Enter ID: ")
dno = int(input("Enter Department Number: "))
company.execute("UPDATE EMPLOYEE SET Dept_No=%s WHERE Aadhaar=%s;", (dno, aadhaar))
```

On executing the above code-snippet, Python prompts the user to enter ID and department number:

```
>>> Enter ID: 567890123456
>>> Enter Department Number: 1
```

The updated EMPLOYEE table is shown in Table 12.4:

Aadhaar	Name	Gender	Salary	Dept_No
123456789012	Raj	M	10000	2
234567890123	Dhiraj	M	85000	1
345678901234	Muskan	F	100000	2
456789012345	Savita	F	80000	1
567890123456	Taran	M	99000	1

Table 12.4: State of EMPLOYEE table after carrying out update operation

Deleting the details of a Selected Employee

Consider the current state of the EMPLOYEE table shown in Table 12.5:

Aadhaar	Name	Gender	Salary	Dept_No
123456789012	Raj	M	10000	2
234567890123	Dhiraj	M	85000	1
345678901234	Muskan	F	100000	2
456789012345	Savita	F	80000	1
567890123456	Taran	M	99000	2

Table 12.5: State of EMPLOYEE table before carrying out delete operation

Next, suppose we want to delete the details of a particular employee based on his/her Aadhaar ID. The following code snippet takes the aadhaar ID of an employee as input and deletes his/her details in the database.

```
#Delete the particular record of the employee.
aadhaar = input("Enter ID: ")
company.execute("DELETE FROM EMPLOYEE WHERE Aadhaar=%s;", aadhaar)
```

On executing the above code-snippet, Python asks for the following input:

```
>>> Enter ID: 567890123456
```

and the EMPLOYEE table gets updated as shown in Table 12.6:

Aadhaar	Name	Gender	Salary	Dept_No
123456789012	Raj	M	10000	2
234567890123	Dhiraj	M	85000	1
345678901234	Muskan	F	100000	2
456789012345	Savita	F	80000	1

Table 12.6: State of EMPLOYEE table after carrying out delete operation

A. Fill in the blanks:

1. To retrieve multiple rows from a table _____method is used.
2. _____ method is used to finalize changes in a database.

B. True or False.

1. We can execute a query even after the cursor object is closed.

12.2 A Complete Example

So far, we have learnt how to: make a connection with a database, create an EMPLOYEE table, insert, update, delete, and retrieve rows from the table. Now, we are ready to write a complete menu-driven program. Program 12.3 creates a connection with the database and creates a table EMPLOYEE. Thereafter, it presents users with the choice to insert, retrieve, delete, or update the EMPLOYEE information based on the user choice.

Program 12.3 Manipulating EMPLOYEE table's data.

```
#main program segment
"""
Objective: To create an EMPLOYEE table, and insert, update, delete, and retrieve rows from the table
User Interface:
    1.User is asked to enter:
       option choice for inserting, retrieving, updating, or deleting rows.
    2. The user is asked to enter attribute values depending upon option selected.
    This is repeated so long as the user wants to continue.
"""

```

```
11
12 import pymysql
13 cnxn= pymysql.connect(host="localhost",user="root",password="",db= "COMPANY")
14 company = cnxn.cursor()
15
16
17 #Creating tables
18 company.execute("""CREATE TABLE EMPLOYEE (
19       Aadhaar CHAR(12) PRIMARY KEY,
20       Name VARCHAR(20) NOT NULL,
21       Gender CHAR(1) NOT NULL,
22       Salary INT NOT NULL,
23       Dept_No SMALLINT,
24       # Validation constraints
25       CHECK(Gender = 'M' OR Gender = 'F' OR Gender = 'O'),
26       CHECK(Salary BETWEEN 8000 AND 100000)
27 );""")
28
29 while True:
30     print("\nChoose the option:")
31     print("1: Insert a row for a new employee\n"+\
32           "2: Retrieve complete information about all employees\n"+\
33          "3: Given Aadhaar No, retrieve complete information about an employees\n"+\
34           "4: Update department number\n"+\
35           "5: Delete employee data\n"+\
36           "6: EXIT\n")
37     choice = int(input("Enter a choice: "))
38     if choice == 1:  #INSERT
39         print("Enter the employee details:")
40         aadhaar = input("ID: ")
41         name = input("Name: ")
42         gender = input("Gender: ")
43         salary = int(input("Salary: "))
44         dno = int(input("Department Number: "))
45         company.execute("INSERT INTO EMPLOYEE VALUES(%s, %s, %s, %s, %s);",\
46             (aadhaar, name, gender, salary, dno))
47     elif choice == 2: #SELECT
48         company.execute("Select * from EMPLOYEE ")
49         print("Details of all employees:")
50         tuples = company.fetchall()
51         for row in tuples:
52             print(row)
53     elif choice == 3:
54         aadhaar = input("Enter the employee's ID whose details you wish to retrieve: ")
55         company.execute("Select * from EMPLOYEE WHERE Aadhaar=%s", aadhaar)
56         print("Details of selected employee:")
57         row = company.fetchone()
58         print(row)
59     elif choice == 4:
60        aadhaar = input("Enter the employee's ID whose department you wish to update: ")
61         dno = int(input("Updated Department Number: "))
62         company.execute("UPDATE EMPLOYEE SET Dept_No=%s WHERE Aadhaar=%s;", (dno, aadhaar))
63     elif choice == 5:
64        aadhaar = input("Enter the employee's ID whose details you wish to delete: ")
65         company.execute("DELETE FROM EMPLOYEE WHERE Aadhaar=%s;", aadhaar)
66     elif choice == 6:
67         break
68     else:
69         print("Invalid Choice")
```

```
70      # making changes permanent
71      cnxn.commit()
72
73  # closing the connection
74  cnxn.close()
```

```
Choose the option:
1: Insert a row for a new employee
2: Retrieve complete information about all employees
3: Given Aadhaar No, retrieve complete information about an employees
4: Update department number
5: Delete employee data
6: EXIT
Enter a choice: 1
Enter the employee details:
ID: 123456789012
Name: Raj
Gender: M
Salary: 10000
Department Number: 2
Choose the option:
1: Insert a row for a new employee
2: Retrieve complete information about all employees
3: Given Aadhaar No, retrieve complete information about an employees
4: Update department number
5: Delete employee data
6: EXIT
Enter a choice: 1
Enter the employee details:
ID: 234567890123
Name: Dhiraj
Gender: M
Salary: 85000
Department Number: 1
Choose the option:
1: Insert a row for a new employee
2: Retrieve complete information about all employees
3: Given Aadhaar No, retrieve complete information about an employees
4: Update department number
5: Delete employee data
6: EXIT
Enter a choice: 2
Details of all employees:
('123456789012', 'Raj', 'M', 10000, 2)
('234567890123', 'Dhiraj', 'M', 85000, 1)
Choose the option:
1: Insert a row for a new employee
2: Retrieve complete information about all employees
3: Given Aadhaar No, retrieve complete information about an employees
4: Update department number
5: Delete employee data
6: EXIT
Enter a choice: 3
Enter the employee's ID whose details you wish to retrieve: 123456789012
Details of selected employee:
('123456789012', 'Raj', 'M', 10000, 2)
Choose the option:
1: Insert a row for a new employee
2: Retrieve complete information about all employees
```

```
3: Given Aadhaar No, retrieve complete information about an employees
4: Update department number
5: Delete employee data
6: EXIT
Enter a choice: 4
Enter the employee's ID whose department you wish to update: 234567890123
Updated Department Number: 2
Choose the option:
1: Insert a row for a new employee
2: Retrieve complete information about all employees
3: Given Aadhaar No, retrieve complete information about an employees
4: Update department number
5: Delete employee data
6: EXIT
Enter a choice: 2
Details of all employees:
('123456789012', 'Raj', 'M', 10000, 2)
('234567890123', 'Dhiraj', 'M', 85000, 2)
Choose the option:
1: Insert a row for a new employee
2: Retrieve complete information about all employees
3: Given Aadhaar No, retrieve complete information about an employees
4: Update department number
5: Delete employee data
6: EXIT
Enter a choice: 5
Enter the employee's ID whose details you wish to delete: 123456789012
Choose the option:
1: Insert a row for a new employee
2: Retrieve complete information about all employees
3: Given Aadhaar No, retrieve complete information about an employees
4: Update department number
5: Delete employee data
6: EXIT
Enter a choice: 2
Details of all employees:
('234567890123', 'Dhiraj', 'M', 85000, 2)
Choose the option:
1: Insert a row for a new employee
2: Retrieve complete information about all employees
3: Given Aadhaar No, retrieve complete information about an employees
4: Update department number
5: Delete employee data
6: EXIT
Enter a choice: 6
```

CT 01

Extend program 12.3 to include the following queries:

a. To display the name of employees whose salary is greater than 70000.

b. To display the Department numbers of all employees.

c. To display those employees who work for the departments other than department number 1, 2, or 5.

d. To display Aadhaar ID, name, and the salary of employees whose first name starts with 'R' and salary in the sixty thousand range, i.e., a five-digit number beginning 6.

e. To display the Aadhaar ID and name of employees whose first name contains two a's or two e's.

f. To display each department's minimum, maximum, and average salary (rounded to two decimal places) from the EMPLOYEE table.

12.3 Suggestive Case Study

When insert and update operations are required to be carried out on a huge volume of data, the required data is typically available in a **txt** or **csv** file. In this section, we will read the data from a CSV file to populate a database table. Although a practical database for a company will involve several tables, we will assume that the COMPANY database for a company involves only two tables: an EMPLOYEE table that contains personal information about the employees and a DEPARTMENT table that contains information about each department. First, we will write a program to create the EMPLOYEE table and the DEPARTMENT tables. Next, we will write a program to populate these tables using data from **CSV** files. The program will also generate a report for the rows that fail to satisfy certain constraints and thus, are not inserted into the database. After that, we will write a program to retrieve information from the database as a user requires. Finally, we will write a program to update employees' salaries, taking the updated salary from a **CSV** file. We will also generate a list of the updates for manual verification.

12.3.1 Creation of EMPLOYEE and DEPARTMENT Tables

The program in 12.4 aims at creating EMPLOYEE and DEPARTMENT tables.

In lines 11–12, we set up a connection to the database COMPANY and create the cursor object **company**. We will use the cursor object **company** for facilitating the execution of queries.

- Lines 17–21 describe the **CREATE TABLE** query for DEPARTMENT.
- Similarly, lines 29–39 describe the **CREATE TABLE** query for EMPLOYEE.
- In line 36, we create a named constraint **employee_gender** for validating an employee's gender.
- Similarly, in line 37, we create another constraint **employee_salary** for validating an employee's salary. The names are inspired from the default name employee_ibfk_1, Python will be associating with the foreign key constraint of EMPLOYEE table.
- Note that lines 14–24 are enclosed in a **try-except** block as the table creation may generate **pymysql** error.
- If the DEPARTMENT table is created successfully, execution of line 22 will display the message DEPARTMENT table created successfully!, otherwise an error will be pointed out.
- Similarly, if the EMPLOYEE table is created successfully, execution of line 40 will display the message EMPLOYEE table created successfully!. otherwise, an error will be pointed out.
- Lines 47–57 attempt to create an existing table EMPLOYEE.

Program 12.4 Creation of EMPLOYEE and DEPARTMENT tables.

```
01 #main program segment
02 """
03 Objective: To create an EMPLOYEE and DEPARTMENT tables
04 User Interface: None
05 """
06
07 import pymysql
08 import csv
09
10 # Set up database connection
11 cnxn = pymysql.connect(host="localhost",user="root",password="",db= "COMPANY")
12 company = cnxn.cursor()
13
14 try:
15     # Create table DEPARTMENT
16     print("\nCreating DEPARTMENT table")
17     company.execute("""CREATE TABLE DEPARTMENT(
18                         Dept_No SMALLINT PRIMARY KEY,
```

```
                        Dept_Name VARCHAR(25) NOT NULL,
                        Location VARCHAR(30)
                    );""")
    print("DEPARTMENT table created successfully!")
except pymysql.Error as err:
    print(err.args[1])

try:
    #create table EMPLOYEE
    print("\nCreating EMPLOYEE table")
    company.execute("""CREATE TABLE EMPLOYEE (
        Aadhaar CHAR(12) PRIMARY KEY,
        Name VARCHAR(20) NOT NULL,
        Gender CHAR(1) NOT NULL,
        Salary INT NOT NULL,
        Dept_No SMALLINT,
        # Validation constraints
        CONSTRAINT employee_gender CHECK(Gender = 'M' OR Gender = 'F' OR Gender = 'O'),
        CONSTRAINT employee_salary CHECK(Salary BETWEEN 8000 AND 100000),
        FOREIGN KEY(Dept_No) REFERENCES DEPARTMENT(Dept_No)
        );""")
    print("EMPLOYEE table created successfully!")
except pymysql.Error as err:
    print(err.args[1])

try:
    #Try creating table EMPLOYEE one more time
    print("\nCreating EMPLOYEE table")
    company.execute("""CREATE TABLE EMPLOYEE (
        Aadhaar CHAR(12) PRIMARY KEY,
        Name VARCHAR(20) NOT NULL,
        Gender CHAR(1) NOT NULL,
        Salary INT NOT NULL,
        Dept_No SMALLINT,
        # Validation constraints
        CONSTRAINT employee_gender CHECK(Gender = 'M' OR Gender = 'F' OR Gender = 'O'),
        CONSTRAINT employee_salary CHECK(Salary BETWEEN 8000 AND 100000),
        FOREIGN KEY(Dept_No) REFERENCES DEPARTMENT(Dept_No)
        );""")
    print("EMPLOYEE table created successfully!")
except pymysql.Error as err:
    print(err.args[1])

# making changes permanent
cnxn.commit()

# Close the connection
cnxn.close()
```

On execution of the program 12.4, Python generates the following report on the console:

```
Creating DEPARTMENT table
DEPARTMENT table created successfully!
Creating EMPLOYEE table
EMPLOYEE table created successfully!
Creating EMPLOYEE table
Table 'employee' already exists
```

12.3.2 Populating Database Tables from CSV Files

To populate the database tables, we will write a program (Program 12.5) that operates as follows:

1. reads the name of the CSV file containing details of each department (line 73).
2. invokes the method **populateDept** to populate details of each department to the DEPARTMENT table. (line 76)
3. reads the name of the CSV file containing details of each employee. (line 78)
4. invokes the method **populateEmployee** to populate employee details to the EMPLOYEE. (line 81)

The method **populateDept** (lines 4–33) works as follows:

1. accepts one input argument: name of the CSV file containing details of the departments (line 4)
2. opens a connection to the company database (line 12).
3. creates the cursor object **company** for accessing the database COMPANY (line 15).
4. reads the CSV file containing details of the departments as the object **deptDetails** (line 18).
5. creates a reader object **readDepartment** for file handler **deptDetails** (line 22).
6. skip the header row (line 23).
7. for each of the remaining rows, insert the data into the DEPARTMENT table and print an error message if the insert operation fails because of a violation of some database constraint (lines 24–29).
8. finally, commit updates to the database (line 30).
9. closes the database connection (line 33).

The method **populateEmployee** works in a similar manner to the method **populateDept** (lines 35–64).

Program 12.5 Insertion of data from csv files into EMPLOYEE and DEPARTMENT tables.

```
01 import pymysql
02 import csv
03
04 def populateDept(deptFile):
05     """
06     Objective: To populate Department table from a csv files
07     Input Parameter:
08         deptFile : csv file comprising details of departments
09     Return Value: None
10     """
11     # open connection to COMPANY database
12    cnxn = pymysql.connect(host="localhost", user="root", password="", db= "COMPANY")
13
14     # Set up cursor for company database.
15     company = cnxn.cursor()
16     # open department file
17     try:
18         deptDetails = open('D:\CBSE_Python\\'+deptFile+'.csv', 'r')
19     except IOError:
20             print("Error in opening File")
21
22     readDepartment = csv.reader(deptDetails)
23     header = next(readDepartment) # Skip header row
24     for row in readDepartment:
25         try:
26                 company.execute("INSERT INTO DEPARTMENT VALUES(%s, %s, %s);",\
27                         (int(row[0]), row[1], row[2]))
28         except pymysql.Error as err:
29             print("\n","Row: "+str(row),"\n",err.args[1])
```

```
    cnxn.commit() #Make database populate permanent

    # close connection to COMPANY database
    cnxn.close()

def populateEmployee(empFile):
    """
    Objective: To populate Employee table from a csv files
    Input Parameter:
        empFile : csv file comprising employees' details
    Return Value: None
    """
     # open connection to COMPANY database
    cnxn = pymysql.connect(host="localhost", user="root", password="", db= "COMPANY")

    # Set up cursor for company database.
    company = cnxn.cursor()
    # open employee file
    try:
        empDetails = open('D:\CBSE_Python\\'+empFile+'.csv', 'r')
    except IOError:
            print("Error in opening File")

    readEmployee = csv.reader(empDetails )
    header = next(readEmployee) #To skip reading header row
    for row in readEmployee:
        try:
                company.execute("INSERT INTO EMPLOYEE VALUES(%s, %s, %s, %s, %s);",\
                (row[0], row[1], row[2], int(row[3]), int(row[4])))
        except pymysql.Error as err:
            print("\n","Row: "+str(row),"\n",err.args[1])
    cnxn.commit() #Make database populate permanent

    # close connection to COMPANY database
    cnxn.close()

#main program segment
"""
Objective: To populate EMPLOYEE and DEPARTMENT tables
User Interface: None
"""

#Read name of the CSV file containing department data.
deptFile = input("Enter name of file containing department details: ")
#Upload CSV data to DEPARTMENT table.
print("Inserting rows in DEPARTMENT Table")
populateDept(deptFile)
#Read name of the CSV file containing employee data.
empFile = input("\n\nEnter name of file with employee's details: ")
#Upload CSV data to EMPLOYEE table.
print("Inserting rows in EMPLOYEE Table")
populateEmployee(empFile)
```

On execution of the Program 12.5, Python generates the following report on the console:

```
Enter name of file containing department details: departmentDb
Inserting rows in DEPARTMENT Table
Row: ['1', 'Research', 'Delhi']
Duplicate entry '1' for key 'PRIMARY'
```

```
Enter name of file with employee's details: employeeDb
Inserting rows in EMPLOYEE Table
Row: ['345678901234', 'Muskan', 'F', '100000', '3']
Cannot add or update a child row: a foreign key constraint fails ('company'.'employee',
CONSTRAINT 'employee_ibfk_1' FOREIGN KEY ('Dept_No') REFERENCES 'department' ('Dept_No'))
Row: ['567890123456', 'Taran', 'M', '99000000', '3']
CONSTRAINT 'employee_salary' failed for 'company'.'employee'
```

12.3.3 Information Retrieval from Database Tables (Display on console)

Having populated the database tables successfully, now we are ready to retrieve and update the information stored in the database as required by a user. We will write a program (Program 12.6) that invokes the method **`infoRetrieval`** to retrieve information from the **`COMPANY`** database (line 97).

The method **`infoRetrieval`** (lines 4–87) works as follows:

1 opens a connection to the **`COMPANY`** database (line 11).

2 creates the cursor object **`company`** for accessing the database `COMPANY` (line 13).

3 Display the current state of the `EMPLOYEE` table (lines 16–20).

4 Display the current state of the `DEPARTMENT` table (lines 22–26)

5 Display the name and gender of all employees who work in a department, specified by the user. (lines 28–36)

6 Display the sum, minimum, maximum, and average of the salaries of the employees who work in a department, specified by the user (lines 38–49).

7 Display the number of employees in a department specified by the user (lines 51–58).

8 Display the department name, location, number of employees, and the average salary for all employees in a department(60–70).

9 Update the salary of all employees in a department. specified by the user by 1000 (lines 72–84).

10 closes the database connection (line 87).

Program 12.6 Information retrieval from `EMPLOYEE` and `DEPARTMENT` tables.

```
01 import pymysql
02 import csv
03
04 def infoRetrieval():
05     """
06     Objective: To retrieve information from the database
07     Input Parameter: None
08     Return Value: None
09     """
10     # open connection to COMPANY database
11    cnxn = pymysql.connect(host="localhost", user="root", password="", db= "COMPANY")
12
13     company = cnxn.cursor()
14     print("\nInformation Retrieval from COMPANY database:\n")
15
16     print("\n1. Current state of EMPLOYEE table:\n")
17     company.execute("Select * from EMPLOYEE;")
18     tuples = company.fetchall()
19     for row in tuples:
20         print(row)
21
22     print("\n2. Current state of DEPARTMENT table:\n")
23     company.execute("Select * from DEPARTMENT;")
24     tuples = company.fetchall()
```

```
    for row in tuples:
        print(row)

    print("\n3: Retrieve the name and gender\
    of all employees who work in the department.")
    dept = input("Enter the name of department: ")
    company.execute(""" SELECT Name, Gender
    FROM EMPLOYEE AS E, DEPARTMENT AS D
    WHERE E.Dept_No = D.Dept_No AND Dept_Name = %s;""",dept)
    tuples = company.fetchall()
    for row in tuples:
        print(row)

    print("\n4: Find the sum of the salaries of all\
    employees of the department, as well as the maximum salary,\
    the minimum salary, and the average salary in this department.")
    dept = input("Enter the name of department: ")
    company.execute(""" SELECT SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)
    FROM EMPLOYEE AS E, DEPARTMENT AS D
    WHERE E.Dept_No = D.Dept_No AND Dept_Name = %s;""",dept)
    row = company.fetchone()
    print("Total Salary: ", row[0])
    print("Max Salary: ", row[1])
    print("Min Salary: ", row[2])
    print("Average Salary: ", row[3])

    print("\n5: Retrieve the number of employees in the \
    department.")
    dept = input("Enter the name of department: ")
    company.execute(""" SELECT COUNT(*)
    FROM EMPLOYEE AS E, DEPARTMENT AS D
    WHERE E.Dept_No = D.Dept_No AND Dept_Name = %s;""",dept)
    row = company.fetchone()
    print("Number of Employees in Department "+dept+":", row[0])

    print("\n6: Retrieve the department name, location,\
    number of employees, and the average salary for all\
    employees in that department..")
    company.execute(""" SELECT Dept_Name, Location, COUNT(Aadhaar) AS "No of Employees", \
    AVG(Salary) AS "AverageSalary"
    FROM EMPLOYEE AS E, DEPARTMENT AS D
    WHERE (E.Dept_No = D.Dept_No)
    GROUP BY Dept_Name, Location;""")
    tuples = company.fetchall()
    for row in tuples:
        print(row)

    print("\n7: Update the salary of all employees\
    working in department by 1000$")
    dept = input("Enter the name of department: ")
    company.execute("""UPDATE EMPLOYEE AS E, DEPARTMENT AS D
    SET SALARY = SALARY + 1000
    WHERE E.Dept_No = D.Dept_No AND Dept_Name = %s;""",dept)
    company.execute("""SELECT Dept_Name, Name, Salary
  FROM EMPLOYEE AS E, DEPARTMENT AS D
  WHERE E.Dept_No = D.Dept_No AND Dept_Name = %s;""",dept)
    print("Updated Salaries")
    tuples = company.fetchall()
```

```
    for row in tuples:
        print(row)

    # close connection to COMPANY database
    cnxn.close()

#main program segment
"""
Objective: To retrieve information from EMPLOYEE and DEPARTMENT tables
User Interface:
    1.User is asked to enter department name for queries 3, 4, 5, and 7.
"""

#Retrieve information from Database COMPANY
infoRetrieval(cnxn)
```

On execution of the program in 12.6, Python generates the following report on the console:

```
Information Retrieval from COMPANY database:
1. Current state of EMPLOYEE table:
('123456789012', 'Raj', 'M', 10000, 1)
('234567890123', 'Dhiraj', 'M', 85000, 1)
('456789012345', 'Savita', 'F', 80000, 2)
('678901234567', 'Naval', 'M', 98760, 2)
('789012345678', 'Anshul', 'M', 10000, 2)
2. Current state of DEPARTMENT table:
(1, 'Accounts', 'Noida')
(2, 'Administration', 'Gurgaon')
3: Retrieve the name and gender of all employees who work in the department.
Enter the name of department: Administration
('Savita', 'F')
('Naval', 'M')
('Anshul', 'M')
4: Find the sum of the salaries of all employees of the department, as well as the
maximum salary, the minimum salary, and the average salary in this department.
Enter the name of department: Accounts
Total Salary:  95000
Max Salary:  85000
Min Salary:  10000
Average Salary:  47500.0000
5: Retrieve the number of employees in the department.
Enter the name of department: Accounts
Number of Employees in Department Accounts: 2
6: Retrieve the department name, location, number of employees, and the average salary
for all employees in that department.
('Accounts', 'Noida', 2, Decimal('47500.0000'))
('Administration', 'Gurgaon', 3, Decimal('62920.0000'))
7: Update the salary of all employees working in department by 1000$
Enter the name of department: Administration
Updated Salaries
('Administration', 'Savita', 81000)
('Administration', 'Naval', 99760)
('Administration', 'Anshul', 11000)
```

12.4 Database Update with Inputs from a CSV File

As the part of salary appraisal, suppose the company decides to enhance the salaries of some employees in the company. The updated salaries of the employees are stored in a CSV file.

For this purpose, we will write a program (Program 12.7) that performs the following tasks:

1. reads the name of the CSV file containing salary updates to be carried out (line 52).
2. invokes the method **updateSalary** to update the salaries of the employees of the COMPANY table. (line 54).

The method **updateSalary** (lines 4–42) works as follows:

1. accepts one input argument: name of the CSV file containing details of the salary updates to be carried out (line 4).
2. opens a connection to the **COMPANY** database (line 13).
3. creates the cursor object **company** for accessing the database COMPANY (line 15).
4. reads the entire contents of the CSV file as the object **updationDetails.** The file contents are read (line 19).
5. creates a reader object **readUpdate** for file handler **updationDetails** (line 24).
6. skips the header row (line 25).
7. for each of the remaining rows, updates the salary of the employee whose Aadhaar matches with the Aadhaar of the employee in the row being read. (lines 26–34).
8. commits the changes to database (line 37)
9. closes the opened file (line 39).
10. closes the database connection (line 42).

Program 12.7 Updating salaries of employees.

```
01 import pymysql
02 import csv
03
04 def updateSalary(salUpdateFile):
05     """
06     Objective: To update employees' salaries by reading updation details
07                 from file salUpdateFile
08     Input Parameter:
09         salUpdateFile: CSV file containing details of the salary update to be carried out
10     Return Value: None
11     """
12     # open connection to COMPANY database
13    cnxn = pymysql.connect(host="localhost", user="root", password="", db= "COMPANY")
14
15     company = cnxn.cursor()
16
17     # open the file comprising salary updates to be carried out
18     try:
19         updationDetails = open('D:\CBSE_Python\\'+salUpdateFile+'.csv', 'r')
20     except IOError:
21             print("Error in opening File")
22
23     #update the salary
24     readUpdate = csv.reader(updationDetails )
25     header = next(readUpdate) # Skip header row
26     for row in readUpdate:
27         aadhaar = row[0]
28         update = int(row[1])
29         try:
30              company.execute("""UPDATE EMPLOYEE
31                         SET SALARY = SALARY + %s
32                         WHERE Aadhaar=%s;""",(update , aadhaar))
33         except pymysql.Error as err:
34             print("\n",err.args[1])
35
```

```
36      # Commit the changes
37      cnxn.commit()
38      # Closing the opened file
39      updationDetails.close()
40
41      # close connection to COMPANY database
42      cnxn.close()
43
44  #main program segment
45  """
46  Objective: To update salaries of employees
47  User Interface:
48      1.User is asked to enter the name of csv file comprising salary updates
49      2. User is asked to enter the department name for queries 3, 4, 5, and 7.
50  """
51  #Read name of the CSV file containing salary updates.
52  salUpdateFile = input("Enter name of file containing salary updates: ")
53  #Update employees' salary
54  updateSalary(salUpdateFile)
```

On execution of the program in 12.7, Python generates the following report on the console:

```
Enter name of file containing salary updates: salaryUpdates
Sample data of file salaryUpdates:
Aadhaar,Updation
456789012345,2000
123456789012,1000
789012345678,2000
```

12.5 Verifying Database Updates

Once the salaries are updated in the database, we will retrieve the updated salaries (of those employees whose salaries have been updated) and save them to a CSV file for manual verification to ensure that the updates have been applied correctly. For this purpose, we will write a program (Program 12.8) that performs the following tasks:

1. reads the name of the CSV file containing salary updates to be carried out (line 59).
2. reads the name of the CSV file for storing updated salaries (line 60)
3. invokes the method **`retrieveUpdates`** to save updated employees' salaries to another file for later manual verification (line 62)

The method **`retrieveUpdates`** (lines 4–49) works as follows:

1. accepts two input arguments: name of the CSV file containing details of the salary updates to be carried out and name of the CSV file where updated salaries (of those employees are to be stored (line 4).
2. opens a connection to the **`COMPANY`** database (line 15).
3. creates the cursor object **`company`** for accessing the database `COMPANY` (line 17).
4. opens the CSV file where the updated salaries are to be written as the object **`empSalary`** (line 20).
5. creates a writer object **`writeEmpSal`** for file handler **`empSalary`** (line 22).
6. writes the header row in CSV file using writer object **`writeEmpSal`** (line 23).
7. reads the CSV file containing details of the salary update to be carried out as the object **`updationDetails`** (line 27).
8. creates a reader object **`readUpdate`** for file handler **`updationDetails`** (line 32).
9. skip the header row (line 33).
10. for each of the remaining rows, read the updated salary of the employee whose Aadhaar matches with the Aadhaar of the employee in the row being read and write it to the output file. (lines 34–42).

11. closes all files (lines 45–46).

12. closes the database connection (line 49).

Program 12.8 Verifying updated salaries of employees.

```
01 import pymysql
02 import csv
03
04 def retrieveUpdates(salUpdateFile, outFile):
05     """
06     Objective: To save updated employees' salaries to outFile
07                for later manual verification
08     Input Parameter:
09         salUpdateFile: CSV file containing details of the salary update to be carried out
10         outFile: CSV file where updated salaries (of those employees whose salaries have
11                  been updated) are to be stored
12     Return Value: None
13     """
14     #open connection to COMPANY database
15     cnxn = pymysql.connect(host="localhost", user="root", password="", db= "COMPANY")
16
17     company = cnxn.cursor()
18
19     #open the file where updated salaries are to be written
20     empSalary = open('D:\CBSE_Python\\'+outFile+'.csv', 'w', newline='')
21     columns = list(['Aadhaar', 'Salary'])
22     writeEmpSal = csv.writer(empSalary)
23     writeEmpSal.writerow(columns)
24
25     # open the file comprising salary updates to be carried out
26     try:
27         updationDetails = open('D:\CBSE_Python\\'+salUpdateFile+'.csv', 'r')
28     except IOError:
29             print("Error in opening File")
30
31     #read the updated salary and write it to another file
32     readUpdate = csv.reader(updationDetails)
33     header = next(readUpdate) # Skip header row
34     for row in readUpdate:
35         aadhaar = row[0]
36         try:
37              company.execute("""SELECT Aadhaar, Salary FROM EMPLOYEE
38                           WHERE Aadhaar = %s;""",(aadhaar))
39              tuple = company.fetchone() # Fetch updated salary
40              writeEmpSal.writerow(tuple) # Write updated salary
41         except pymysql.Error as err:
42             print("\n",err.args[1])
43
44     # Closing the opened files
45     updationDetails.close()
46     empSalary.close()
47
48     # close connection to COMPANY database
49     cnxn.close()
50
51 #main program segment
52 """
53 Objective: To update salaries of employees
54 User Interface:
```

```
    1.User is asked to enter the name of csv file comprising salary updates
    2.User is asked to enter the name of csv file for storing updated salaries
"""
#Read name of the CSV file containing salary updates.
salUpdateFile = input("Enter name of file containing salary updates: ")
outFile = input("Enter name of file for storing updated salaries: ")
#Update employees' salary
retrieveUpdates(salUpdateFile, outFile)
```

On execution of the program in 12.8, Python generates the following report on the console:

```
Enter name of file containing salary updates: salaryUpdates
Enter name of file for storing updated salaries: outputFile
Sample data of file outputFile:
Aadhaar,Salary
456789012345,82000
123456789012,11000
789012345678,12000
```

12.5.1 Some More Programs

1. Write a Python Code that create table (TRAVEL) in the database and enter at least 5 records in the table. The program should also give the option to display data from table TRAVEL.

   Specifications for TRAVEL table:

```
T_ID          of type string
Package       of type string (Luxury / Deluxe/ Economy)
Destination   of type string
NumDays       of type int
Pax       of type int
```

```
#main program segment
"""
Objective: To create an TRAVEL table, and insert and retrieve rows from the table
User Interface:
    1.User is asked to enter:
       option choice for inserting and retrieving rows.
    2. The user is asked to enter values of attribute of TRAVEL table for choice 1.
    This is repeated so long as the user wants to continue.
"""

import pymysql
cnxn= pymysql.connect(host="localhost",user="root",password="",db= "PAYROLL")
payroll= cnxn.cursor()

#Creating tables
payroll.execute("""CREATE TABLE TRAVEL (
    T_ID CHAR(12) PRIMARY KEY,
    Package VARCHAR(20) NOT NULL,
    Destination VARCHAR(20) NOT NULL,
    NumDays INT NOT NULL,
    Pax INT NOT NULL,
    # Validation constraint
    CONSTRAINT travel_package CHECK(Package = 'Luxury' OR Package = 'Deluxe' OR
    Package = 'Economy')
);""")

while True:
    print("\nChoose the option:")
    print("1: Insert a row for a new travel package\n"+\
```

```
          "2: Retrieve complete information about all travel packages\n"+\
          "3: EXIT\n")
    choice = int(input("Enter a choice: "))
    if choice == 1:  #INSERT
        print("Enter the travel package details:")
        tid = input("Travel Id: ")
        package = input("Package Name: ")
        dest = input("Destination: ")
        numDays = int(input("Number of days: "))
        pax = int(input("Pax: "))
        try:
            payroll.execute("INSERT INTO TRAVEL VALUES(%s, %s, %s, %s, %s);",\
            (tid, package, dest, numDays, pax))
        except pymysql.Error as err:
            print("\n","Row: "+str(row),"\n",err.args[1])
    elif choice == 2: #SELECT
        payroll.execute("Select * from TRAVEL ")
        print("Details of all travel packages:")
        tuples = payroll.fetchall()
        for row in tuples:
            print(row)
    elif choice == 3:
        break
    else:
        #making changes permanent
        cnxn.commit()

#closing the connection
cnxn.close()
```

2. Assume that the following table MOVIE is existing in the database.

Movie_ID	Movie_Name	Category	Release_Date
M001	Sherlock Homes-II	Thriller	2010-11-14
M002	Do Dooni Char	Comedy	2014-01-20
M003	Ranjhana	Romance	2013-10-12
M004	A Wednesday	Thriller	2008-04-25
M006	Hera Pheri	Comedy	2015-02-28

Assume that the database is created and the connection with the database is also established. Write a function that takes a list object (consisting of Movie_ID,Movie_Name, Category and Release_Date) as arguments and saves them in the table MOVIE.

3. Write a Python code to create the following tables, insert 4–5 records in these tables and then display the following data:
   - Product(PID, PName, SID, QTY, Price)
   - Shipment(SID, Date, Destination, Cost)
   - Queries to be executed from the program:
   - Display Product Id, Product Name, Shipment Date and destination for all products.
   - Display the date of shipment for each product.
   - Display the products which will be delivered to London
   - Display the average cost of shipments.

   #close connection also!!!

Let's Summarise

- Python MySQL client library **pymysql** can be used for connecting to MySQL database server from Python.
- method **connect()** of the **pymysql** module is used to create a connection with an already existing database.
- method **cursor**() of **connect** instance is used to create a handler of the connection object.
- method **close()** of the **connect** module is used to close the connection with the database once all the required database operations have been carried out.
- To reflect the changes in the database, we need to commit the changes using method **commit()** of the **connect** object.
- Method **execute()** of **cursor** object is used to execute the SQL query from Python.
- Placeholders **%s** or **%{}s** are used to provide user-defined values in the SQL queries to be executed.
- Method **fetchall()** of **cursor** object is used to retrieve all the rows from the result of executing an SQL query in the form of a tuple of tuples.
- Method **fetchmany(n)** of **cursor** object is used to retrieve n rows from the result of executing a SQL query in the form of a tuple of tuples.
- Method **fetchone()** of **cursor** object is used to retrieve single row from the result of executing a SQL query.

Solved Exercises

A. Multiple Choice Questions

1. Which of the following method of cursor class can be used to retrieve the single row from the result of executing a SQL query?
i: `fetchone()` ii: `fetchall()` iii: `fetchmany()`
a. i and iii b. i, ii, and iii c. ii and iii d. None of these

2. Which of the following method of cursor class can be used to retrieve multiple rows from the result of executing a SQL query?
i: `fetchone()` ii: `fetchall()` iii: `fetchmany()`
a. i and iii b. i, ii, and iii c. ii and iii d. None of these

3. Which of the following methods from the `pymysql` module is used to execute an SQL query from Python?
a. `connect()` b. `close()` c. `execute()` d. `fetchall()`

4. Which of the following methods can be used to commit the changes permanently to the database?
a. `commit()` b. `close()` c. `execute()` d. `save()`

5. Which of the following is referred to as the result set returned as the result of executing a SQL query?
a. `tuple of tuples` b. `list of tuples` c. `tuple of lists` d. `list of lists`

6. Which of the following methods of **cursor** class can be used to retrieve the number of rows in the result of executing SQL query?
a. `cursor.rowcount` b. `cursor.fetchall` c. `cursor.fetchone` d. None of these

B. State whether the following statements are True or False:

1. By default, the **PyMySQL** library is installed in the Python application. _____

2. **connect()** is used to establish a connection between a database and the Python interface. _____

3. Once a record is inserted in a table using **`execute()`**, it is permanently stored. _____

4. One row from the result of executing an SQL query can be retrieved using **`fetchone().`** _____

5. We cannot count the number of rows being retrieved from the table on the execution of an SQL query. _____

C. Fill in the blanks.

1. _____ method is used to create Python program's connection with an already existing database.
2. The SQL queries in the Python program are executed using _____ method.
3. The _____ method ensures that the updates made in the database are permanent.
4. In the `execute()` method, the placeholder _____ is used for each value in the tuple.
5. The _____ method retrieves all rows from the result of execution of an SQL query.

D. Answer the following questions:

1. Consider the following statement to extract data from the resultset retrieved from the database. [CBSE Sample Paper, 2022]

```
Mydata = cursor.fetchone()
```

(i) How many records will be returned by **`fetchone()?`**

(ii) What will be the datatype of `Mydata` object after the given command is executed?

Ans. (i) One (ii) tuple

2. Why do we need to establish connectivity between a Python program and MYSQL database?

Ans: We need to establish connectivity between a Python program and MYSQL database so that we can we access the MySQL database from Python.

3. Differentiate between connect() and cursor().

Ans: connect() : connect() method is used to create a connection with an already existing database. It takes four arguments and returns a connection object. The syntax of the connect method is given below:

<connectionObject> = pymysql.connect(host=<hostName>, user=<userName>, password=<password>,

db=<databaseName>)

<hostName>: MySQL server hostname or the IP address

<userName>: User name

<password>: password associated with the user

<databaseName>: Name of the MySQL database with which connection is sought

cursor(): cursor() method creates a handler of connection object for performing operations on the database.

E.g : if cnxn is a connection object then cursor object can be created as:

company = cnxn.cursor()

4. What is the use of commit()?

Ans: The changes we make in the database using function execute() are not permanent. To reflect these changes in the database, we need to commit the changes using the method commit() of the connection object.

E.g : if cnxn is a connection object then cnxn.commit() will commit the changes to a database.

5. Differentiate between fetchone(), fetchmany(n) and fetchall() methods.

Ans: **fetchone():** retrieves single row from the result of executing a SQL query in the form of a tuple

fetchmany(n): retrieves n rows from the result of executing a SQL query in the form of a tuple of tuples

fetchall(): retrieves all the rows from the result of executing a SQL query in the form of a tuple of tuples

6. Consider the SCHOOL database given below.

TEACHER

ID	First_Name	Last_Name	Subject_No	Contact_Num	Salary	Email_ID
10001	Hema	Priya	1	9953799955	75000	hemapriya@gmail.com
10002	Allu	Arjun	2	8010235773	35000	arjunallu@gmail.com
10003	Naishadh	Kumar	4	9965789799	75000	naishadh.k@yahoo.com
10004	Sameer	Reddy	1	9899417155	90000	sameer@hotmail.com
10005	Sathvik	Bal	8	9653367876	85000	sathvik@gmail.com
10006	Tenzin	Wangdi	3	8023456780	65000	tenzin@yahoo.com
10007	Krishan	Kumar	2	9977885566	70000	krishnakumar@gmail.com
10008	Ankuj	Pandey	5	9990825566	80000	ankujp@gmail.com
10009	Sangeetha	NULL	2	9890235262	65000	sangeetha@gmail.com
10010	Charu	Srivastava	7	9555235321	95000	charus@gmail.com

SUBJECT

Subject_No	Subject_Name	Dept_No
1	English	1
2	Chemistry	3
3	Physics	3
4	Business Studies	2
5	Economics	1
6	Accounts	2
7	Mathematics	3
8	History	1
9	Geography	1
10	Computer Science	3

Write Python statements to do the following:

(i) Assume that the required library is imported, the username is root, and the password for the database is "#practice123". Write the statement to establish a connection between Python and the database SCHOOL.

(ii) Create a cursor object, school using the connection object created in part a.

Use the connection and cursor objects created above to answer the questions that follow:

(iii) Write the commands to create the tables TEACHER and SUBJECT teacher using **execute**() of Python.

(iv) Write a function enterTeacherDetails() that accepts the data from the user and then inserts them in table TEACHER of student database using format().

(v) Write a function enterSubjectDetails() that takes Subject_No, Subject_Name and Dept_No as arguments and inserts them into the table SUBJECT of STUDENT database.

(vi) Write the execute() to do the following: (Also, use **commit**() wherever required)

a. Display all rows from the tables TEACHER and SUBJECT

b. For each subject, display subject name first name, and last name of all the teachers who have a salary more than 20000.

c. List the name of the subjects for which teachers have been currently hired.

d. Update the salary by providing an increment of 1000 to teachers teaching Computer Science.

```
#objective:
#User interface:


def enterSubjectDetails(school):
    print("Enter the subject details:")
    subjNum = int(input("Subject Number: "))
    subjName = input("Subject Name: ")
    deptNum = int(input("Department Number: "))
    try:
        query = "INSERT INTO SUBJECT VALUES({}, '{}', {});"\
                .format(subjNum, subjName, deptNum)
        school.execute(query)
```

```
    except pymysql.Error as err:
        print("\n","Row: "+str(row),"\n",err.args[1])

def enterTeacherDetails(school):
    print("Enter the teacher details:")
    tid = input("ID: ")
    fName = input("First Name: ")
    lName = input("Last Name: ")
    subNo = int(input("Subject Number: "))
    contactNum = int(input("Contact Number: "))
    salary = int(input("Salary: "))
    emailId = input("Email ID: ")
    try:
       query = "INSERT INTO TEACHER VALUES('{}', '{}', '{}', {}, {}, {}, '{}');"\
                .format(tid, fName, lName, subNo, contactNum, salary, emailId )
        school.execute(query)
    except pymysql.Error as err:
        print("\n","Row: "+str(row),"\n",err.args[1])


#main program segment
"""
Objective: To create TEACHER and SUBJECT tables, and insert, update, delete, and
retrieve rows from the  table
User Interface:
    1.User is asked to enter:
       option choice for inserting, retrieving, or updating rows.
     2. The user is asked to enter attribute values depending upon the option
selected.
    This is repeated so long as the user wants to continue.
"""

import pymysql
cnxn= pymysql.connect(host="localhost",user="root",password="",db= "SCHOOL")
school = cnxn.cursor()

#Creating tables
school.execute("""CREATE TABLE SUBJECT
           (Subject_No SMALLINT PRIMARY KEY,
           Subject_Name VARCHAR(30),
           Dept_No SMALLINT);""")

school.execute("""CREATE TABLE TEACHER(
           ID VARCHAR(5) PRIMARY KEY,
           First_Name VARCHAR(20),
           Last_Name VARCHAR(20),
           Subject_No SMALLINT,
           Contact_Num DECIMAL(10,0),
           Salary INT,
           Email_ID VARCHAR(50),
```

```
            Foreign Key(Subject_No) references SUBJECT(Subject_No));""")

while True:
    print("\nChoose the option:")
    print("1: Insert a row for a new subject\n"+\
          "2: Insert a row for a new teacher\n"+\
         "3: Retrieve complete information about all subjects and teachers\n"+\
           "4: For each subject, retrieve subject name first name, and last name of all
the"+\
             " teachers who have a salary more than 20000\n"+\
            "5: List the name of the subjects for which teachers have been currently
hired\n"+\
           "6: Update the salary by providing an increment of 1000 to teachers
teaching
Computer Science\n"+\
          "7: EXIT\n")
    choice = int(input("Enter a choice: "))
    if choice == 1:  #Insert SUBJECT details
        enterSubjectDetails(school)
        cnxn.commit()
    elif choice == 2: #Insert TEACHER details
        enterTeacherDetails(school)
        cnxn.commit()
    elif choice == 3:
        #Retrieve SUBJECT details
        school.execute("Select * from SUBJECT ")
        print("Details of all subjects:")
        tuples = school.fetchall()
        for row in tuples:
            print(row)
        #Retrieve TEACHER details
        school.execute("Select * from TEACHER ")
        print("Details of all teachers:")
        tuples = school.fetchall()
        for row in tuples:
            print(row)
    elif choice == 4:
        school.execute("""SELECT Subject_Name, First_Name, Last_Name, Salary
                        FROM SUBJECT S, TEACHER T
                        WHERE T.Subject_No = S.Subject_No AND Salary > 20000
                        ORDER BY Subject_Name, First_Name;""")
        tuples = school.fetchall()
        for row in tuples:
            print(row)
    elif choice == 5:
        school.execute("""UPDATE SUBJECT S, TEACHER T
            SET SALARY = SALARY +1000
            WHERE S.Subject_Name = "Computer Science" AND T.Subject_No = S.Subject_
            No;""")
        tuples = school.fetchall()
```

```
            for row in tuples:
                print(row)
        elif choice == 6:
            school.execute("""UPDATE SUBJECT S, TEACHER T
                      SET SALARY = SALARY +1000
                    WHERE T.Subject_No = S.Subject_No AND S.Subject_Name = "Computer  Science";""")
            cnxn.commit()
            tuples = school.fetchall()
            for row in tuples:
                print(row)
        elif choice == 7:
            break
        else:
            print("Invalid Choice")
    cnxn.close()
```



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

Assertion (A): The library mysql.connector has to be installed before establishing connection between Python and MySQL database.

Reasoning (R): While establishing the connection between Python and MySQL, the MySQL server may not run at the backend.

Ans. c



Case-based Questions

Aarushi owns a Performing Arts Studio in Delhi. In its two auditoriums, various theatre troupes stage their play/acts. She wants to maintain a database to store data of all the stage shows in the Theatre table given below.

TCODE	TITLE	AUDI	LANGUAGE	SHOWDATE	TICKET PRICE	TROUPID
T001	Akbar The Great Nahi Rahe	2	Hinglish	2023–05–12	150	R003
T002	Mughal-E-Azam	1	Hindi	2023–05–08	200	R001
T003	Romeo and Juliet	2	English	2023–05–09	125	R004
T004	The Lost Love	2	English	2023–05–10		R001
T005	Salaam Noni Appa	1	Hindi	2023–05–12	250	R002
T006	Gandhi The Untold Story	1	Hinglish	2023–07–13	375	R006

Create the table, Theatre, in MySQL database named ORANGE. Thereafter insert the records as given in the table above. Write the functions in Python to perform the following tasks.

a. Display all records of table Theatre.

b. To update the Audi no of play with Title 'Mughal-E-Azam' to Audi no 2.

c. To display the title and ticket price of plays that are in hindi language and whose ticket price is known.

d. Accept TCODE from user and display the corresponding record.

For establishing connection with the database, assume that the host is "localHost", user id is "root" and password is "hello123".

Ans. a.
```
def display():
    import mysql.connector
    M1 = mysql.connector.connect(host='localhost', user='root', passwd='hello123')
    C1 = M1.cursor()
    C1.execute("USE ORANGE")
    C1.execute("SELECT * FROM Theatre")
    data=C1.fetchall()
    for i in data:
        print(i)
display()
```
b.
```
def updateAudi():
    import mysql.connector
   M1 = mysql.connector.connect(host='localhost', user='root', passwd='hello123')
    C1 = M1.cursor()
    C1.execute("USE ORANGE")
    C1.execute("UPDATE Theatre SET Audi=2 WHERE Title=('%s')" %("Mughle Azam"))
    M1.commit()
    data=C1.fetchall()
    for i in data:
        print(i)
updateAudi()
```
c.
```
def hindiPlays():
    import mysql.connector
    M1 = mysql.connector.connect(host='localhost', user='root', passwd='hello123')
    C1 = M1.cursor()
    C1.execute("USE orange")
    C1.execute("SELECT TITLE, TICKETPRICE FROM Theatre WHERE LANGUAGE=('%s') AND
    TICKETPRICE IS NOT NULL" % ('Hindi'))
    data=C1.fetchall()
    for i in data:
        print(i)
hindiPlays()
```
d.
```
def displayRecord():
    import mysql.connector
    M1 = mysql.connector.connect(host='localhost', user='root', passwd='tiger')
    C1 = M1.cursor()
    C1.execute("USE orange")
    code=input("Enter the TCODE :  ")
    C1.execute("SELECT * FROM Theatre WHERE TCODE='{}'".format(code,))
    data=C1.fetchall()
    for i in data:
        print(i)

displayRecord()
```

Assessment

A. Multiple Choice Questions

1. Which of the following libraries is used to access MySQL database from Python?

a. mysqlPython b. pymysql c. PythonMySQL d. PSQL

2. Which of the following commands is used to download Python MySQL client library, pymysql?

a. python3-m pip install pymysql
b. mysql3-m pip install python
c. python3 pip install pymysql
d. python3-m install pymysql

3. Which of the following methods is used to connect Python with an already existing database in MySQL?

a. `databaseConnect()` b. `MySQLconnect()` c. `connect()` d. `connectDatabase()`

4. Which of the following is an optional argument for connect()?

a. MySQL host name
b. User Name
c. Password associated with the user
d. Name of SQL database with which connection is sought.

5. Which of the following is the correct statement to connect Python with a database named "TRIAL"? Assume that the host is localhost, user name is "SchoolLab" and password is "orange".

a. `CO=pysql.connect(host="localhost",user="SchoolLab",password="orange",db="TRIAL")`
b. `CO=pysql.connect(host="SchoolLab",user="localhost",password="orange",db="TRIAL")`
c. `CO=connect(host="localhost",user="SchoolLab",password="orange",db="TRIAL")`
d. `CO=connect(host="SchoolLab",user="localhost",password="orange",db="TRIAL")`

6. Which of the following methods ensures that the database updates become permanent?

a. `execute()` b. `connect()` c. `cursor()` d. `commit()`

7. Consider the statement given below:

```
sal=1000
C1.execute("SELECT * FROM Employee WHERE Salary>%s;"),\sal)
```

Which of the following statements is NOT true?

a. %s is a placeholder for value 1000
b. %s will be replaced by 1000 when the query is executed
c. The statement will generate an error
d. The statement will fetch all records whose Salary is greater than 1000.

8. Which of the following methods retrieve teh result of an SQL query in the form of a tuple?

a. `fetchone()` b. `fetchmany()` c. `fetchall()` d. `fetch()`

B. State whether the following statements are True or False:

1. `connect()` is used to establish connection with a non-existing database in MySQL. _____

2. `close()` disconnects the database connection with Python. _____

3. `execute()` ensures that records are inserted permanently in the database. _____

4. While executing a query, the `format()` has to be used to provide attribute values to the tuple. _____

5. `fetchall()` retrieves records from the database in the form of a nested tuple. _____

C. Fill in the blanks.

1. The _____ method establishes connection between Python and an existing database in MySQL.

2. The _____ method of connection object is used to create a handler for performing operations on a database.

3. `execute()` is used to execute _____ through a Python program.

4. While executing an SQL query using `execute()`, the _____ method is used to provide attribute values to the tuple.

5. _____ yields the number of rows retrieved on executing an SQL query.

D. Answer the following questions:

1. Explain the two methods that can be used to pass values of attributes to the SQL query in the execute()?
2. Explain the use of method format() to pass attribute values for the execution of the SQL query.
3. Consider a table `STUDENT` having three attributes Roll_No, Name, and Marks. Are the following two statements for inserting a row in `STUDENT` table equivalent ?

   (i) >>> school.execute("INSERT INTO STUDENT VALUES(%(roll_no)s, %(name)s,\ %(marks)s);",\
   >>> { 'roll_no':studRollNo, 'name':studName, 'marks':studMarks })

   (ii) >>> school.execute("INSERT INTO STUDENT VALUES(%(roll_no)s, %(name)s,\ %(marks)s);",\
   >>> {'name':studName, 'marks':studMarks, 'roll_no':studRollNo})
4. Give the syntax of functions for reading and writing to a csv file.
5. Concerning the table TEACHER in the SCHOOL database given in question no 1 , write a menu-driven program with the following menus:

   (i) ENTER Data

   (ii) DISPLAY Data

   (iii) UPDATE Data

   (iv) DELETE Data

   (v) EXIT

   - The option ENTER Data should accept data from the user and store them in the database.
   - The option DISPLAY Data should display the rows from the table.
   - The option UPDATE Data should accept ID and contact number from the TEACHER and then update it in the table.
   - The option DELETE Data should delete the teacher's record whose ID is entered by the user.

   Note: Create separate user-defined functions for each menu item.
6. Consider the following table SALE in the database INVENTORY.

Table: SALE

Bill_ID	Cust_Name	Item_Purchased	Price	CGST
B0101	Saleem	Watch	4980	18
B011	Ashwini	Book	598	5
B012	Joydeep	Watch	1439	12
B014	Rashmi	Saree	23899	14
B015	Samantha	Phone	47350	12

   Write a menu driven program in Python to do the following:

   - Accept data
   - Display entire data along with total price which is sum of Price+ CGST/100*Price
   - Update the price after accepting the Bill_ID
   - Delete the row that has value of Item_Purchased as Saree.
7. Consider that an empty database with the name `SHIPMENTDB` already exists at MYSQL server. The database should contain three tables—SUPPLIER, PRODUCT AND SHIPMENT.

Table: SUPPLIER

SNo	Sname	SCity
101	Aradhaya Pvt. L	Mumbai
102	XYZ Enterprises	Bangalore
103	Komal Enterpris	Delhi
104	Cloudtail	Patna
105	PQR Enterprises	Uttrakhand
106	Tech Solutions	Bangalore
107	XYZ Enterprises	Mumbai

Table: PRODUCT

PNo	Pname	Brand	City
201	Refrigerator	Whirlpool	Chennai
202	Washing Machine	Samsung	Kolkata
203	Television	Sony	Mumbai
204	Television	Samsung	Kolkata
205	Washing Machine	Whirlpool	Chennai
206	Refrigerator	Whirlpool	Delhi
207	Microwave Oven	IFB	Mumbai
208	Microwave	Samsung	Mumbai
209	Air Fryer	IFB	Chandigarh
210	Air Fryer	Philips	Lucknow

Table: SHIPMENT

OrderNo	SNo	PNo	Price	Quantity
1000	101	201	50000.00	50
1001	101	202	70000.00	30
1002	102	202	90000.00	75
1003	104	205	25000.00	20
1004	105	205	55000.00	30
1005	102	207	48000.00	50
1006	105	210	23000.00	30
1007	104	209	60000.00	75
1008	102	208	65000.00	20
1009	103	207	81000.00	30
1010	106	204	56000.00	30
1011	107	210	38000.00	50
1012	101	201	72000.00	50

Write a Python script that achieves the following:

(i) Make a connection with the database. assume that the required library is imported, the username is `root`, and the password for the database is "`abc#123`".

(ii) Create the tables `SUPPLIER`, `PRODUCT`, and `SHIPMENT` through the Python program.

(iii) Accept the data from the user and store the records in three tables.

(iv) Retrieve all the information from the database by displaying the current state of `SUPPLIER`, `PRODUCT`, and `SHIPMENT` tables.

(v) Generate a report to display the following:

a. Order number and name of the product supplied in each shipment.

b. Order number of the products of `Whirlpool` brand.

c. Names of suppliers who have supplied more than two orders.

d. Number of orders supplied by each supplier.



Assertion and Reasoning Based Questions

The following questions are Assertion(A) and Reasoning(R) based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation of A

b. Both A and R are true and R is not the correct explanation of A

c. A is true but R is false

d. A is false but R is true

1. Assertion (A): A global variable is defined outside all user-defined functions of a program.
Reasoning (R): A global variable can be accessed anywhere in the program.

2. Assertion(A): Before modifying the value of a global variable inside a user-defined function, it must be declared global by using the keyword global.
Reasoning (R): In the absence of the keyword global, modifying the value of a variable declared global in the program introduces a new local variable by the same name.



Case-based Questions

1. Sudha is an entrepreneur. She wants to store the data of all her clients in a MySQL database. The host name is "HOST1", user name is "ORANGE" and password is "education@1". The table Client contains the following fields.
clientId-integer-primary key
cName-string-not null
city-string-not null
contactNo-string
She has approached you to write a program to accept data from the user and save it in the table Client. The program should also display the records of the clients who are in Delhi.

2. Sukriti has created a table Troupe in the database named Adventure. The host name is "HOST1", user name is "ORANGE" and password is "education@1". Thereafter the table was populated with following records:
Table: Troupe

TROUPID	TNAME	CITY
R001	Nattak Mela	Nagpur
R002	Dhara	Delhi
R003	OnStage	Delhi
R004	Reflection	Mumbai
R005	Beyond the Skies	Jaipur
R006	Parivartan	Mumbai

Post covid, many troupes have moved to new cities. Sukrit wants you to write a program in Python that will accept the troupe id and city. Thereafter, the table is updated with new value of city for the given troupe id.



Previous Years' Questions

Consider the following SQL table MEMBER in a SQL Database CLUB: [2022]

M_ID	NAME	ACTIVITY
M1001	Amina	GYM
M1002	Pratik	GYM
M1003	Simon	SWIMMING
M1004	Rakesh	GYM
M1005	Avneet	SWIMMING

Assume that the required library for establishing the connection between Python and MYSQL is already imported in the given Python code. Also assume that DB is the name of the database connection for table MEMBER stored in the database CLUB. Predict the output of the following code:

```
MYCUR =  DB.cursor()
MYCUR. execute ("USE CLUB")
MYCUR.execute ("SELECT * FROM MEMBER WHERE ACTIVITY= 'GYM';")
R=MYCUR.fetchone ()
for i in range (2):
    R=MYCUR. fetchone ()
    print (R[0], R[1], sep = "#")
```

Ans. M1002#Pratik

M1004#Rakesh



NCERT Exercise Solutions

1. Give the terms for each of the following:

a. Collection of logically related records.

b. DBMS creates a file that contains a description of the data stored in the database.

c. Attribute that can uniquely identify the tuples in a relation.

d. a special value that is stored when the actual data value is unknown for an attribute.

e. An attribute that can uniquely identify tuples of the table but is not defined as the primary key of the table.

f. Software that is used to create, manipulate, and maintain a relational database.

Ans. a. Database

b. Data Dictionary

c. Primary Key

d. Null

e. Candidate key

f. Relational Database Management System (RDBMS)

2. Why are foreign keys allowed to have NULL values? Explain with an example.

Ans. Foreign keys are referenced values from the table having similar attributes as a primary key from referenced relations. As a result, some values in the relation are unknown or do not match the referenced relation in the database. Thus, this foreign key can have a null value.

Example:

Consider two tables, `Order` and `Customer`. The `Order` table has a foreign key, `customerID`, which references the primary key `customerID` in the `Customer` table. If a customer places an order, the `customerID` in the `Order` table is set to the customer's ID. However, if a guest or unregistered user places an order, the `customerID` in the Orders table may be `NULL`, indicating that no particular customer is associated with the order.

3. Differentiate between:

a. Database state and database schema

b. Primary key and foreign key

c. Degree and cardinality of a relation

Ans. a. **Database State (also known as an extension)** refers to the actual data stored in a database at a specific moment. It represents the current snapshot of the information contained in the database.

Database Schema (also known as intension) defines the structure of the database, including tables, fields, relationships, and constraints. It does not contain the actual data but provides a blueprint for organizing and accessing the data.

b. **Primary Key:** A unique identifier for a record in a database table. It ensures that each record in the table is distinct.

Foreign Key: A field in a database table that refers to the primary key in another table. It establishes a link between tables, enforcing referential integrity and supporting relationships between them.

Refer to 9.2.1: Relational Data Model, Primary Key and 9.2.1: Relational Data Model, Foreign Key for more details.

c. **Degree of a Relation:** Refers to the number of attributes or columns in a relation (table). For example, a relation with attributes A, B, and C has a degree of 3.

Cardinality of a Relation: Describes the number of tuples or rows in a relation. It indicates the size or cardinality of the set of relationships. For instance, a relation with five rows has a cardinality of 5.

4. Compared to a file system, how does a database management system avoid redundancy in data through a database?

Ans. A Database Management System (DBMS) avoids redundancy in data by centralizing and organizing data in a structured manner. Unlike file systems, a DBMS uses a relational model where data is stored in tables with relationships defined between them. This normalization process minimizes redundancy by storing data in a way that avoids duplication. Data integrity is maintained through foreign key relationships, and changes are made in one place, reducing the risk of inconsistencies that can occur in file systems. This ensures efficient data storage, retrieval, and maintenance within a database.

5. What are the limitations of file systems that a relational DBMS can overcome?

Ans. a. Difficulty in Access

b. Data Redundancy

c. Data Inconsistency

d. Data Isolation

e. Data Dependence

f. Controlled Data Sharing

6. A school has a rule that each student must participate in a sports activity. So each one should give only one preference for sports activity. Suppose there are five students in a class, each having a unique roll number. The class representative has prepared a list of sports preferences, as shown below. Answer the following:

Table: Sports Preferences

Roll_no	Preference
9	Cricket
13	Football
17	Badminton
17	Football
21	Hockey
24	NULL
Null	Kabaddi

a. Roll No. 24 may not be interested in sports. Can a NULL value be assigned to that student ' s preference field?

b. Roll No. 17 has given two preferences for sports. Which property of relational DBMS is violated here? Can we use any constraint or key in the relational DBMS to check against such violations, if any?

c. Kabaddi was not chosen by any student. Is it possible to have this tuple in the Sports Preferences relation?

Ans. a. It cannot be assigned a NULL value because, according to school rules, every student is required to participate in a sports activity.

b. Because it violates the primary key constraint. Roll no. 17 cannot give two preferences. Every student has a unique roll number. We can use the primary key constraint to check for such violations.

c. This is not possible because the roll number field cannot be NULL.

In another class having two sections, the two respective class representatives have prepared two separate Sports Preferences tables, as shown below:

Sports preference of section 1 (arranged in the roll number column)

Table: Sports Preferences

Roll_no	Sports
9	Cricket
13	Football
17	Badminton
21	Hockey
24	Cricket

Sports preference of section 2 (arranged in Sports name column, and column order is also different)

Table: Sports Preferences

Sports	Roll_no
Badminton	17
Cricket	9
Cricket	24
Football	13
Hockey	21

Are the states of both tables equivalent? Justify.

Ans. The states of both tables are equivalent despite the differences in their organization. In the relational model, the order of rows and columns does not affect the equivalence of tables. What matters is the content of the tables and whether they represent the same information. Despite the differences in column order and row arrangement, the information represented in both tables is the same—each table conveys the sports preferences of students in their respective sections. Therefore, the states of both tables are equivalent.

7. The school canteen wants to maintain records of items available in the school canteen and generate bills when students purchase any item from the canteen. The school wants to create a canteen database to keep track of items in the canteen and the items purchased by students. Design a database by answering the following questions:

a. To store each item name along with its price, what relation should be used? Decide on appropriate attribute names along with their data type. Each item and its price should be stored only once. What restriction should be used while defining the relation?

b. In order to generate the bill, we should know the quantity of an item purchased. Should this information be in a new relation or a part of the previous relation? If a new relation is required, decide the appropriate name and data type for attributes. Also, identify the appropriate primary key and foreign key so that the following two restrictions are satisfied:

   i. The same bill cannot be generated for different orders.

   ii. Bills can be generated only for available items in the canteen.

c. The school wants to find out how many calories students consume when they order an item. In which relation should the attribute 'calories' be stored?

Ans. a. `Item` relation, which stores information about the food item's name and price.

```
Item (ItemId, ItemName, Price)
```

The `ItemId` column is the primary key of the relation. `ItemName` is the name of the food item, and the `price` is its price per quantity.

To restrict duplicate values, ItemId is made the primary key of the relation.

b. A new relation is required to store the item's purchase. It would be called `Orders`, and it should have the following attributes:

   i. `OrderId`: a unique ID for every order. It is the primary key. This constraint satisfies the condition (i).

   ii. `Quantity`: stores the quantity of items purchased by the student.

   iii. `ItemId`: foreign key that references the `Item` relation to make sure that item is available in the canteen. This constraint satisfies condition (ii).

c. The attribute 'calories' should be stored in the `Item` relation. Since every item has its own calorie value, it would be stored with every item.

8. An organisation wants to create a database called `EMPDEPENDENT` to maintain the following details about its employees and their dependents.

```
EMPLOYEE(AadharNumber, Name, Address, Department, EmployeeID)
DEPENDENT(EmployeeID, DependentName, Relationship)
```

a. Name the attributes of `EMPLOYEE`, which can be used as candidate keys.

b. The company wants to retrieve details about the dependents of a particular employee. Name the tables and the keys that are required to retrieve this detail.

c. What is the degree of EMPLOYEE and DEPENDENT relation?

Ans. a. `AadharNumber, EmployeeID`

b. From `DEPENDENT` table EmployeeID, DependentName, and from the `EMPLOYEE` table AdhaarNumber, Name would be required to retrieve details about the dependents of a particular employee.

c. The degree of `EMPLOYEE` table is 5 and the degree of `DEPENDENT` table is 3.

9. School uniforms are available at M/s Sheetal Private Limited. They have maintained `SCHOOL_UNIFORM` Database with two relations viz. `UNIFORM` and `COST`. The following figure shows the database schema and its state.

`SCHOOL_UNIFORM` Database

Attributes and Constraints

Table: UNIFORM

Roll_no	Sports	Sports	Sports
Constraints	Primary Key	Not Null	–

Table: COST

Attribute	UCode	Size	Price
Constraints	Composite Primary Key		>0

Table: COST

UCode	Size	COST Price
1	M	500
1	L	580
1	XL	620
2	M	810
2	L	890
2	XL	940
3	M	770
3	L	830
3	XL	910
4	S	150
4	L	170
5	S	180
5	L	210
6	M	110
6	L	140
6	XL	160

Table: UNIFORM

UCode	UName	UColor
1	Shirt	White
2	Pant	Grey
3	Skirt	Grey
4	Tie	Blue
5	Socks	Blue
6	Belt	Blue

a. Can they insert the following tuples into the `UNIFORM` relation? Give reasons in support of your answer.
   i. 7, Handkerchief, NULL
   ii. 4, Ribbon, Red
   iii. 8, NULL, White

b. Can they insert the following tuples into the `COST` relation? Give reasons in support of your answer.
   i. 7, S, 0
   ii. 9, XL, 100

Ans. a. i. Yes, because `Ucode` 7
   ii. No, because `Ucode` 4 is already assigned to Tie and `Ucode` can't have duplicate values being a primary key.
   iii. No, because Uname is following the Not Null constraint.

b. i. No, as it is violating check constraints
   ii. Yes

10. In a multiplex, movies are screened in different auditoriums. One movie can be shown in more than one auditorium. In order to maintain the record of movies, the multiplex maintains a relational database consisting of two relations viz. MOVIE and AUDI respectively as shown below:

```
Movie(Movie_ID, MovieName, ReleaseDate)
Audi(AudiNo, Movie_ID, Seats, ScreenType, TicketPrice)
```

a. Is it correct to assign Movie_ID as the primary key in the MOVIE relation? If no, then suggest an appropriate primary key.

b. Is it correct to assign AudiNo as the primary key in the AUDI relation? If no, then suggest appropriate primary key.

c. Is there any foreign key in any of these relations?

Ans. a. Yes, we can use `Movie_ID` as the primary key because each movie can have unique values.

b. No, `AudiNo` along with `Movie_ID` should be made as the primary key. That is, the relation should have a composite primary key as `(AudiNo, Modie_ID)`

c. Yes, `Movie_ID` is a foreign key in the `AUDI` table, which refers to the `Movie_ID` column of `MOVIE` table.

11. For the above given database STUDENT-PROJECT, answer the following:

Student-Project Database

Table: STUDENT

Roll No	Name	Class	Section	Registration_ID
11	Mohan	XI	1	IP-101-15
12	Sohan	XI	2	IP-104-15
21	John	XII	1	CS-103-14
22	Meena	XII	2	CS-101-14
23	Juhi	XII	2	CS-101-10

Table: PROJECT

ProjectNo	PName	SubmissionDate
101	Airline Database	12/01/2018
102	Library Database	12/01/2018
103	Employee Database	15/01/2018
104	Student Database	12/01/2018
105	Inventory Database	15/01/2018
106	Railway Database	15/01/2018

Table: PROJECT ASSIGNED

Registration_ID	ProjectNo
IP-101-15	101
IP-104-15	103
CS-103-14	102
CS-101-14	105
CS-101-10	104

a. Name primary key of each table.

b. Find foreign key(s) in table PROJECT-ASSIGNED.

c. Is there any alternate key in table STUDENT? Give justification for your answer.

d. Can a user assign duplicate value to the field RollNo of STUDENT table? Jusify.

Ans. a. The name of the primary key of each table is as follows:

Table	Primary Key
`Student`	`RollNo`
`Project`	`ProjectNo`
`ProjectAssigned`	`{Registration_ID, ProjectNo}`

Note that: `Registration_ID` of `STUDENT` is an alternate key.

b. The `ProjectNo` is the foreign key for `PROJECT-ASSIGNED` table.

c. Yes, `Name` and `Registration_ID` can be an alternate keys in the `STUDENT` table as it sores unique values for each rows.

d. No, as we have already defined `RollNo` as a primary key. So user cannot assign duplicate value to the field `RollNo`.

12. For the above given database STUDENT-PROJECT, can we perform the following operations?

a. Insert a student record with missing roll number value.

b. Insert a student record with missing registration number value.

c. Insert a project detail without submission-date.

d. Insert a record with registration ID IP-101-19 and ProjectNo 206 in table PROJECT-ASSIGNED.

Ans. a. No, because `RollNo` is a primary key, so it can not be NULL.

b. Yes, because `Registration_ID` is a foreign key in the `STUDENT` table, so the record with missing `Registration_ID` can be inserted.

c. Yes, detail of project can be inserted provided a null constraint is not set for `SubmissionDate`.

d. No, because `Registration_ID` with value `IP-101-19` does not exist in `STUDENT`. Also, `ProjectNo` with value `206` does not exist in `PROJECT`.

Answers

Multiple Choice Questions

1. (a) 2. (c) 3. (c) 4. (a) 5. (a) 6. (a)

True or False

1. (F) 2. (F) 3. (F) 4. (T) 5. (F)

Fill in the blanks

1. connect() 2. execute() 3. commit() 4. %s 5. fetchall()

APPENDICES

Installing Anaconda Python

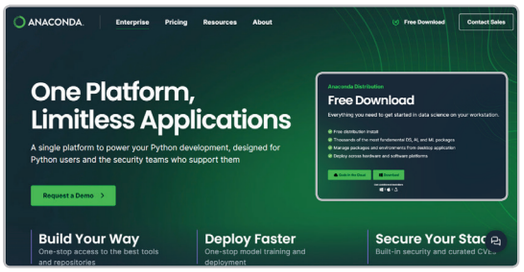
Anaconda is developed and maintained by Peter Wang and Travis Oliphant founders of Anaconda, Inc., in 2012. It is a popular, free and open-source distribution of Python and R for Windows, Linux, and macOS. It has a Python interpreter and a large collection of packages used for Data Science, Machine Learning, Deep Learning that simplifies the process of package management and deployment. It gives you the facility of working on all domains of AI at one place by installing many packages with a single installation process.

Anaconda Prompt is an Anaconda's Command Line Interface used for Python programming by creating different virtual environments and using the installed packages. You can open this Anaconda Prompt by writing "Anaconda Prompt" in the windows search bar.

Anaconda Navigator is an alternative to the Anaconda Prompt. It is a desktop graphical user interface included in Anaconda that allows you to configure, install, and launch tools such as Jupyter Notebook. Jupyter notebook can easily be accessed using the Anaconda Prompt with the help of a local host.

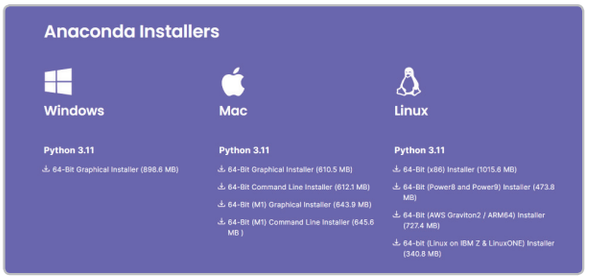
Follow the given steps to install Anaconda for Windows.

1. Visit the web site: https://www.anaconda.com/distribution

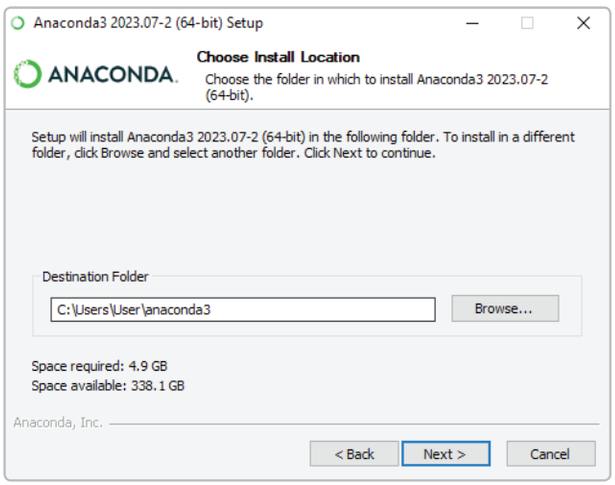


2. Click on Get Additional Installers → Windows.

   It displays the given screen: In category: Windows → Python 3.11 → Select the right option according to the configuration of your pc(32-bit/64-bit). This will start the download process on your computer. (Go to Windows Start → Settings → system → About to see the configuration of your computer)



3. Select Start → Search → type anaconda. The given screen will be displayed.



Practical

Solved

Python

Program 1: Write a program that reads a text file line by line and displays each word separated by a #.

Ans.
```
def wordSep():
    '''
    Objective: To read a text file line by line and display each word separated by a #
    Input Parameter: None
    Return Value: None
    '''
    f = open('lines.txt', 'r')
    sentenceList = f.readlines()
    for sentence in sentenceList:
       words = sentence.split()
       for y in words:
         print(y, end='#')
       print()
    f.close()
    wordSep()
```

Program 2: Write a program that counts vowels, consonants, uppercase, and lowercase characters in a file.

Ans.
```
def fileCount(filename):
    '''
    Objective : To display the count of vowels, consonants, uppercase and
    lowercase characters
     Input Parameter : Filename - string
     Return Value : None
     '''
     f = open(filename, 'r')
     content = f.read()
     countVowels = 0
     countConsonants = 0
     countUppercase = 0
     countLowercase = 0
```

```
        print(content)
        for ch in content:
          if ch.isalpha():
            if ch in "aeiouAEIOU":
              countVowels += 1
            else:
              countConsonants += 1
            if ch.islower():
              countLowercase += 1
            else:
              countUppercase += 1
        print("The number of vowels are : ", countVowels)
        print("The number of consonants are : ", countConsonants)
        print("The number of uppercase characters are : ", countUppercase)
        print("The number of lowercase characters are : ", countLowercase)
        f.close()

    filename = input('Enter the filename: ')
    fileCount(filename)
```

Program 3: Write a program that removes all the lines that contain the character 'a' from a text file and writes it to another file.

Ans.
```
def aLines():
    '''
    Objective : To remove all the lines that contain a character 'a' in a text file write
    it to another file
    Input Parameter : None
    Return Value : None
    '''
    f = open('lines.txt', 'r')
    f1 = open('a.txt', 'w')
    sentenceList = f.readlines()
    for sentence in sentenceList:
       words = sentence.split()
       for y in words:
         if y.find('a') == -1:
          f1.write(sentence)
```

```
f.close()
f1.close()
f1 = open('a.txt', 'r')
print(f1.read())
aLines()
```

Program 4: Write a program that creates a binary file with name and rollno. It should also allow you to search for a given roll number and display the name; if not found, display the appropriate message.

Ans.
```
import pickle
def enterData():
'''
Objective : To input student data
Input Parameter : None
Return Value : None
'''
ch = 'y'
f = open("student.dat","ab")
while ch == 'y':
   name = input("Enter name of student : ")
   rollno = int(input("Enter roll number of student : "))
   data =[name, rollno]
   pickle.dump(data,f)
   ch = input("Do you wish to enter more : ")
   if ch != 'y':
    break
f.close()
enterData()
def searchData():
'''
Objective : To search for a given roll number and display the name
Input Parameter : None
Return Value : None
'''
print(" Searching Data ")
rollno = int(input("Enter the roll number to be searched"))
f = open("student.dat","rb")
L1 = []
```

```
cnt = 0
try:
   while True:
     L1 = pickle.load(f)
     if L1[1] == rollno:
       print("The name for roll no ", rollno, "is : ",L1[0])
cnt = 1
except EOFError:
   print()
if cnt == 0:
   print("Record not found")
f.close()
searchData()
```

Program 5: Write a program to create a binary file with roll no, name and marks. It should also allow you to input a roll number and update the marks.

Ans.
```
import pickle
def enterData():
'''
Objective : To input student data
Input Parameter : None
Return Value : None
'''
ch = 'y'
f = open("student.dat","ab")
while ch == 'y':
    rollNo = int(input("Enter Roll No of student : "))
    name = input("Enter name of student : ")
    marks = int(input("Enter Marks : "))
    data = [rollNo, name, marks]
    pickle.dump(data,f)
    ch = input("Do you wish to enter more : ")
    if ch != 'y':
    break
f.close()
enterData()
def updateData():
```

```
'''
Objective : To update the marks
Input Parameter : None
Return Value : None
'''
print("Updating Data")
rollno = int(input("Enter the roll number : "))
f = open("student.dat","rb+")
L1 = []
cnt = 0
try:
while True:
    pos = f.tell()
    if L1[0] == rollno:
    cnt = 1
    print("The data of roll no ", rollno, " is ", L1)
    updatedMarks = int(input("Enter marks to be updated : "))
    L1[2] = updatedMarks
    f.seek(pos)
    pickle.dump(L1,f)
except EOFError:
    print()
if cnt == 0:
    print("Record not found")
f.seek(0)
print()
print()
print("Records After updation")
try:
    while True:
    L1=pickle.load(f)
    L1 = pickle.load(f)
    print(L1)
except EOFError:
    print()
f.close()
updateData()
```

Program 6: Write a random number generator that generates a random number between 1 and 6

Ans.
```
import random
def randomGenerator():
'''
Objective : To generate a random number between 1 and 6
Input Parameter : None
Return Value : None
'''
print("The random number generated between 1 and 6 are : ")
num = random.randint(1,6)
print(num)
randomGenerator()
```

Program 7: Write a program to implement a stack using list.

Ans.
```
def PUSH(numberList):
'''
Objective : To implement a stack
Input Parameter : None
Return Value : None
'''
stack = []
for num in numberList:
    if num % 5 == 0:
    stack.append(num)
if stack:
    print("Stack contains numbers divisible by 5:", stack)
else:
    print("No numbers divisible by 5 found in the list.")
numbers = [10, 25, 7, 15, 30, 12]
PUSH(numbers)
```

Program 8: Write a program to create a csv file by adding user id and password. It should also allow user to read and search the password for a user id.

Ans.
```
import csv
def enterData():
'''
Objective : To accept and store data in csv file
Input Parameter : None
```

```
Return Value : None
'''
ch = 'y'
f = open("user.csv","a",newline="")
csvwriter = csv.writer(f)
while ch == 'y':
    userId = input("Enter User Id : ")
    passwrd = input("Enter password : ")
    data = [userId,passwrd]
    csvwriter.writerow(data)
    ch = input("Do you wish to enter more : ")
    if ch != 'y':
    break
f.close()
enterData()
def searchData():
'''
Objective : To search for a given user id
Input Parameter : None
Return Value : None
'''
print(" Searching Data ")
userId = input("Enter the user id to be searched : ")
f = open("user.csv","r",newline="")
cnt = 0
data1 = csv.reader(f)
for r in data1:
    if r[0] == userId:
    print("The password for user id ", r[0], "is : ",r[1])
    cnt = 1
    break
if cnt == 0:
    print("Record not found")
f.close()
searchData()
```

Program 9: A binary file stores data of items in the form of dictionary objects as shown below:

Data = {itemCode:[name,price]}

Write a menu driven program to add and display data to the binary file.

Ans.
```
import pickle
    def menu():
    '''
    Objective : To print a menu to select a choice
    Input Parameter : None
    Return Value : None
    '''
    print(" 1. Add Data\n","2. Display Data\n","3. Exit\n")
    ch = int(input("Enter Your Choice : "))
    if ch == 1:
        enterData()
    elif ch == 2:
        displayData()
    elif ch == 3:
        return
    else:
       print("Wrong Choice")
    def enterData():
    '''
    Objective : To input item data
    Input Parameter : None
    Return Value : None
    '''
    ch = 'y'
    f = open("items.dat","ab")
    data = {}
        while ch == 'y':
        code = input("Enter Item Code : ")
        name = input("Enter Name : ")
        price = int(input("Enter Price : "))
        data[code] =[name, price]
        print(data)
        pickle.dump(data,f)
```

```
        ch = input("Do you wish to enter more : ")
        if ch != 'y':
        break
    f.close()
    menu()
    def displayData():
    '''
    Objective : To display items
    Input Parameter : None
    Return Value : None
    '''
    f=open("items.dat","rb")
    D1 = {}
    cnt = 1
    try:
        while True:
         D1 = pickle.load(f)
         print("Record No : ", cnt)
         for r in D1:
           print("Item Code: ", r)
           print("Item Name: ", D1[r][0])
           print("Item price: ",D1[r][1])
         cnt = cnt + 1
         print()
    except EOFError:
        print()
    f.close()
    menu()
    menu() # displaying menu
```

Program 10: Write a program that takes a year as an input from the user and checks whether the given year is a leap year or not.

Ans.
```
def leapYear(year):
    '''
    Objective : To check whether the given year is a leap year or not
    Input Parameter : year – numeric value
    Return Value : None
```

```
    '''
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        print(f"{year} is a leap year.")
    else:
        print(f"{year} is not a leap year.")

    year = int(input("Enter a year: "))
    leapYear(year)
```

Program 11: Write a program that accepts a digit as an input from the user. If the digit is a positive number (1–9), then display it in words; otherwise, display the message, "Error: A positive digit is expected and less than 10".

Ans.
```
def positiveDigit(digit):
    '''
    Objective : To display digit in words
    Input Parameter : digit – numeric value
    Return Value : None
    '''
    if digit >= 0:
        digitWords = ["Zero", "One", "Two", "Three", "Four", "Five", "Six",
        "Seven", "Eight", "Nine"]
        print(f" digit {digit} in words is: { digitWords [digit]}")

    else:
        print("Error: A positive digit is expected or less than 10.")

    digit = int(input("Enter a digit: "))
    positiveDigit(digit)
```

Program 12: Write a program that accepts three colours from the user. If the user enters orange, red and green, respectively, the message "Indian Flag" is displayed, otherwise, the message "Not an Indian Flag" is displayed.

Ans.
```
def flag(color1, color2, color3):
    '''
    Objective : To accept 3 colors and check whether it is Indian flag or not
    Input Parameter : color1, color2, color3 - strings
    Return Value : None
    '''
    if color1 == "orange" and color2 == "white" and color3 == "green":
```

```
        print("Indian Flag")
    else:
        print("Not an Indian Flag")

    color1 = input("Enter the first color: ").strip().lower()
    color2 = input("Enter the second color: ").strip().lower()
    color3 = input("Enter the third color: ").strip().lower()
    flag(color1,color2,color3)
```

Program 13: Write a program to accept two numbers and print all prime numbers between those two numbers.

Ans.
```
def isPrime(num):
    '''
    Objective : To accept two numbers and print all prime numbers between them
    Input Parameter : num – numeric value
    Return Value : bool value
    '''
    if num <= 1:
       return False
     if num <= 3:
       return True
     if num % 2 == 0 or num % 3 == 0:
       return False
     i = 5
     while i * i <= num:
       if num % i == 0 or num % (i + 2) == 0:
         return False
       i += 6
     return True
    start = int(input("Enter the starting number: "))
    end = int(input("Enter the ending number: "))
    if start >= end:
    print("Invalid input. The starting number should be less than the ending
    number.")
    else:
    print(f"Prime numbers between {start} and {end}:")
    for num in range(start, end + 1):
        if isPrime(num):
         print(num)
```

Program 14: Write a program to accept a number and print sum of following series:

1/1! + 2/2! + 3/3! + ... n/n!

Ans.
```
def factorial(num):
    '''
    Objective : To find factorial of a number
    Input Parameter : num – numeric value
    Return Value : numeric value
    '''
    if num == 0:
        return 1
    else:
        return num * factorial(num - 1)
    n = int(input("Enter a number (n): "))
    seriesSum = 0
    for i in range(1, n + 1):
    term = i / factorial(i)
    seriesSum += term
    print(f"The sum of the series is: {seriesSum:.4f}")
```

Program 15: Write a program to accept a positive integer (n) and a floating point number (x) print sum append: (Assume n <= 10.)

$x/1! + x^2/2! + x^3/3! + ... + x^n/n!$

Ans.
```
def factorial(num):
        if num == 0:
            return 1
        else:
            return num * factorial(num - 1)
    n = int(input("Enter a positive integer (n): "))
    x = float(input("Enter a floating-point number (x): "))
    if n <= 0:
        print("Invalid input. n must be a positive integer.")
    else:
        series_sum = 0
        for i in range(1, n + 1):
            term = (x ** i) / factorial(i)
            series_sum += term
          print(f"The sum of the series is: {series_sum:.4f}")
```

Program 16: Write a program to check whether a number is a palindrome or not.

Ans.
```
def palindrome(num):
    '''
    Objective : To check whether a number is a palindrome
    Input Parameter : num - numeric value
    Return Value : 1 - if number is palindrome, 0 otherwise
    '''
    n = num   #storing the number for later usage
    reverseNum = 0
    remainder = 0
    while num > 0:
        remainder = num %10
        reverseNum = reverseNum * 10 + remainder
        num //= 10
    if n == reverseNum:
        return 1
    else:
        return 0
num = int(input("Enter a number: "))
result = palindrome(num)
if result == 1:
    print(num, 'is palindrome')
  else:
    print(num, 'is not palindrome')
```

Program 17: Write a program that accepts a number and displays sum of squares of all even numbers till that number.

Ans.
```
def sumSquare(num):
    '''
    Objective : To accept a number and display sum of squares of all even numbers
    till that number
    Input Parameter : num – numeric value
    Return Value : None
    '''
    sum = 0
    for a in range(0,num + 1,2):
        sum = sum + a * a
    print("The sum is ", sum)
    number = int(input("Enter a number : "))
    sumSquare(number)
```

Program 18: Write a program to accept a string and output a tuple comprising the length of each word of the string. Finally, the program should display the tuple elements in ascending order. For example, if the string entered by the user is `'Hello what do you want from me today'`, then the program should produce the following output:

```
(5 4 2 3 4 4 2 5)
2 2 3 4 4 4 5 5
```

Ans.

```
def ascendingLength(line):
    '''
    Objective : To print length of words in line in ascending order
    Input Parameter : line - string
    Return Value : None
    '''
    T = ()
    L = []
    words = line.split()
    for x in words:
    p = len(x)
    T = T + (p,)
    print("The tuple is : ", T)
    L = sorted(T)
    print("The lengths in sorted order is : ")
    for x in L:
        print(x, end=' ')
    line = input("Enter a string : ")
    ascendingLength(line)
```

Program 19: Write a program that accepts a number, n and displays the following pattern.

```
    *
   * *
  * * *
 * * * *
* * * * *
```

Ans.

```
def triangle(n):
    '''
    Objective : To display the triangle pattern
    Input Parameter : n – numeric value
    Return Value : None
    '''
    spaces = n - 1
```

```
    for i in range(0, n):
        for j in range(0, spaces):
          print(end=" ")
        spaces = spaces - 1
        for j in range(0, i+1):
          print("* ", end="")
        print()
    num = int(input("Enter a number : "))
    triangle(num)
```

Program 20: Write a program that accepts 10 numbers and adds them to a list. Thereafter, it displays the following menu:

1. **Sort in Ascending order**
2. **Sort in Descending order**
3. **Largest Number**
4. **Smallest Number**
5. **Exit**

Ans.
```
L = []
def accept():
'''
Objective : To input a list
Input Parameter : None
Return Value : None
'''
for x in range(10):
    p = int(input("Enter number "))
    L.append(p)

def menu():
'''
Objective : To select one of the choices
Input Parameter : None
Return Value : None
'''
print(" 1. Sort in ascending order\n","2. Sort in Descending order\n","3.
Largest Number\n",
"4. Smallest Number\n","5. Exit")
```

```
ch = int(input("Enter Your Choice : "))
if ch == 1:
    L.sort()
    print(L)
elif ch ==2:
    L.sort(reverse = True)
    print(L)
elif ch ==3:
    print("The largest number is : ",max(L))

elif ch ==4:
    print("The smallest number is : ",min(L))
elif ch == 5:
    return
else:
    print("Wrong Choice")
accept()
menu()
```

Program 21: Write a program to accept a positive integer (n) and a floating point number (x) print sum of following series:

$x/1! + x^2/2! + x^3/3! + ... + x^n/n!$

The program should keep adding the terms until the absolute difference between the two consecutive terms becomes less than 0.001.

Ans.
```
def factorial(num):
    '''
    Objective : To find factorial of a number
    Input Parameter : num – numeric value
    Return Value : numeric value
    '''
        if num == 0:
        return 1
    else:
        return num * factorial(num - 1)
    n = int(input("Enter a positive integer (n): "))
    x = float(input("Enter a floating-point number (x): "))
    if n <= 0:
    print("Invalid input. n must be a positive integer.")
```

```
else:
seriesSum = 0
term = x
i = 1
while abs(term) >= 0.001:
    seriesSum += term
    i += 1
    term = (x ** i) / factorial(i)
print(f"The sum of the series is: {seriesSum:.4f}")
```

Program 22: Write a program to accept a decimal number and display its binary equivalent.

Ans.
```
def decToBin(decNum):
'''
Objective : To calculate binary equivalent of decimal number
Input Parameter : decNum – numeric value
Return Value : None
'''
if decNum >= 1:
   decToBin(decNum // 2)
   print(decNum % 2,end="")
print()
dNum = int(input("Enter the decimal number : "))
print("The binary equivalent of ",dNum, "is : ",end = "")
decToBin(dNum)
```

Program 23: Write a program that reads the test file myFile.txt and counts the number of words that end with a period('.')

Ans.
```
def wordEnd():
'''
Objective : To count the number of words ending with a period('.')
Input Parameter : None
Return Value : None
'''
f = open('myFile.txt', 'r')
cnt = 0
wordList = f.read()
words = wordList.split()
for x in words:
```

```
        if x.endswith('.'):
        cnt = cnt + 1
    print("The number of words that end with a period are : ", cnt)
    f.close()
    wordEnd()
```

Program 24: Write a program that accepts a string, and forms a new string that contains only three lettered words, with space between each word. Display both strings.

For example, if the string entered is You are very good and sweet person

The new string will have You are and

Ans.
```
def threeLetterWord(inputString ,words):
    '''
    Objective : To form a string containing only 3 letter words
    Input Parameter : inputString, words – string
    Return Value : None
    '''
    threeLetterWords = []
    for word in words:
        if len(word) == 3:
        threeLetterWords.append(word)
    newString = " ".join(threeLetterWords)
    print("Original string:", inputString)
    print("New string:", newString)
    inputString = input("Enter a string: ")
    words = inputString.split()
    threeLetterWord(inputString ,words)
```

Program 25: Write a function in Python that takes a list as an argument and displays the square of even numbers and square roots of odd numbers separately. For example, if the list is.

```
L = [4,49,10,2,25]
```

The output should be:

Ans.
```
import math
    def processNumbers(numbers):
    '''
    Objective : To calculate square of even numbers and square root of odd numbers
    Input Parameter : numbers – numeric value
    Return Value : None
    '''
    evenSquares = []
    oddSquareRoots = []
```

```
for num in numbers:
    if num % 2 == 0:
     evenSquares.append(num ** 2)
    else:
     oddSquareRoots.append(math.sqrt(num))
print("Square of even numbers are:", end=" ")
for square in evenSquares:
    print(square, end="\t")
print("\nSquare root of odd numbers are:", end=" ")
for root in oddSquareRoots:
    print(root, end="\t")
L = [4, 49, 10, 2, 25]
processNumbers(L)
```

Program 26: Write a user-defined function that accepts a character as an argument, and returns 0 if it is a digit, 1 if it is an alphabet and 2 if it is a special character. Display appropriate messages according to the values returned by the function.

Ans.
```
def charType(char):
    '''
    Objective : To tell the type of character
    Input Parameter : char – numeric value
    Return Value : 0,1,2 – numeric values
    '''
    if char.isdigit():
        return 0 # 0 for a digit
     elif char.isalpha():
        return 1 # 1 for an alphabet
    else:
        return 2 # 2 for a special character
    inputChar = input("Enter a character: ")
    result = charType(inputChar)
    if result == 0:
    print("The character is a digit.")
    elif result == 1:
    print("The character is an alphabet.")
    else:
    print("The character is a special character.")
```

Program 27: Write a function that reads a csv file, `userData.csv`, that stores user id and password of students in the form of list objects. The function should display only those records whose user id ends with a digit.

Ans.
```
import csv
def displayRecordsWithDigitEnd(filename):
'''
Objective : To display only those records whose user id ends with a digit
Input Parameter : filename - file compromising details
Return Value : None
'''
try:
   with open(filename, mode='r', newline='') as file:
     reader = csv.reader(file)
     matchingRecords = []
     for row in reader:
       userId = row[0]
       if userId[-1].isdigit():
         matchingRecords.append(row)
     if matchingRecords:
       print("Matching records (User IDs ending with a digit):")
       for record in matchingRecords:
         print("User ID:", record[0], "Password:", record[1])
     else:
       print("No matching records found.")
except FileNotFoundError:
    print("The file", filename, "was not found.")
csvFilename = "userData.csv" displayRecordsWithDigitEnd (csvFilename)
```

Program 28: Write a menu driven program to save, display, and search records using a csv file, `furniture.csv` that uses a list object to store following data:

`furnitureId`	int
`fNname`	string
`price`	float

The program should also have an option to search for a record based on `furnitureId`.

Ans.
```
import csv
def saveRecord(filename, record):
'''
Objective : To save a record to file
Input Parameter : filename – file compromising furniture details, record
Return Value : None
'''
with open(filename, mode='a', newline='') as file:
```

```
    writer = csv.writer(file)
    writer.writerow(record)
def displayRecords(filename):
'''
Objective : To display records from file
Input Parameter : filename - file compromising furniture details
Return Value : None
'''
try:
    with open(filename, mode='r', newline='') as file:
     reader = csv.reader(file)
     print("Furniture Records:")
     for row in reader:
       print("Furniture ID:", row[0])
       print("Furniture Name:", row[1])
       print("Price:", row[2])
       print()
except FileNotFoundError:
    print("The file", filename, "was not found.")
def searchRecord(filename, furnitureId):
'''
Objective : To search for records in a file
Input Parameter : filename - file compromising furniture details, furnitureId
- numeric value
Return Value : None
'''
try:
   with open(filename, mode='r', newline='') as file:
     reader = csv.reader(file)
     for row in reader:
       if row[0] == furnitureId:
         print("Furniture Record Found:")
         print("Furniture ID:", row[0])
         print("Furniture Name:", row[1])
         print("Price:", row[2])
         return
     print("Furniture ID not found.")
 except FileNotFoundError:
   print("The file", filename, "was not found.")
def mainMenu():
 '''
```

```
    Objective : To print a menu to select choice from
    Input Parameter : None
    Return Value : None
    '''
    while True:
        print("\nMenu:")
        print("1. Save Record")
        print("2. Display Records")
        print("3. Search Record by Furniture ID")
        print("4. Quit")
        choice = input("Enter your choice: ")
        if choice == '1':
        furnitureId = input("Enter Furniture ID: ")
        furnitureName = input("Enter Furniture Name: ")
        price = input("Enter Price: ")
        record = [furnitureId, furnitureName, price]
        saveRecord("student.csv", record)
        print("Record saved successfully.")
        elif choice == '2':
         displayRecords("student.csv")
        elif choice == '3':
         furnitureId = input("Enter Furniture ID to search: ")
         searchRecord("student.csv", furnitureId)
        elif choice == '4':
         break
        else:
        print("Invalid choice. Please select a valid option.")
    if __name__ == "__main__":
        mainMenu()
```

SQL

Program 29: Consider the table Schedule given below:

Table: Schedule

EventId	EName	EDate	ETime	Etype
1001	Appointment with Maple	2023–11–10	11 AM	Official
1002	Birthday Rashmi	2023–10–01	7 AM	Personal
1003	Play–The Hamlet	2023–09–30	7 PM	Personal
1004	Board Meeting	2023–11–18	2 PM	Official
1005	Flight–Mumbai	2023–07–11	10:30 AM	Official
1006	Submit Tender	2023–08–03	12 AM	Official

Write SQL statements to perform the following tasks:

(i) Display the records in ascending order of event date.

Ans.
```
mysql> SELECT * FROM Schedule
    -> ORDER BY EDate;
```

(ii) Display name, date and time of personal events.

Ans.
```
mysql> SELECT EName, EDate, ETime
    -> FROM Schedule
    -> WHERE Etype = 'Personal';
```

(iii) Change the time of event named 'Birthday Rashmi' to 12 AM

Ans.
```
mysql> UPDATE Schedule
    -> SET ETime = '12 AM'
    -> WHERE EName = 'Birthday Rashmi';
```

(iv) Add another attribute in the table with following specifications:

Attribute Name : Remarks

Data Type : varchar(50)

Ans.
```
ALTER TABLE Schedule
ADD Remarks varchar(50);
```

(v) Add the value of Remark as 'Collect Tickets' for the event id 1005.

Ans.
```
UPDATE Schedule
    -> SET Remarks = 'Collect Tickets'
    -> WHERE EventId = 1005;
```

(vi) Delete the events that are scheduled before November 2023.

Ans.
```
mysql> DELETE FROM Schedule
    -> WHERE EDate < '2023-11-01';
```

(vii) Remove the attribute Remarks from the table.

Ans.
```
ALTER TABLE Schedule
DROP Remarks;
```

Program 30: Consider the table Student and write queries for performing the following tasks:

Table: Student

Roll_Num	Student_Name	Course_Name	Duration	Fee	Batch_Prefer
1	Bhaskar Dhyani	Web Development	3	20000	Morning
2	Dakshinesh	Machine Learning	4	25000	Evening
3	Saumit Raj	Office Tools	NULL	50000	Evening
4	Lovepreet Singh	Mobile App	3	18000	Morning
5	Aditya Jayant	Python Programming	6	45000	Evening
6	Jenny Fernandis	Office Tools	4	40000	Evening

(i) Display the total number of records in the table.

Ans.
```
SELECT COUNT(*) FROM student;
```

(ii) Display the average fee given by students.

Ans.
```
SELECT AVG(Fee) FROM student;
```

(iii) Display the total duration of all courses.

Ans.
```
SELECT SUM(Duration) FROM Student;
```

(iv) Display the count of records for each batch preference.

Ans.
```
SELECT COUNT(*), Batch_prefer FROM student
    GROUP BY Batch_Prefer;
```

(v) Display the minimum and maximum fee paid by the student.

Ans.
```
SELECT MIN(Fee), MAX(Fee) FROM student;
```

(vi) Display the average fee of records whose duration is known.

Ans.
```
SELECT AVG(Fee) FROM student
   -> WHERE Duration IS NOT NULL;
```

Program 31: Consider the tables Passengers and Trains given below:

Table: Passengers

PNR	TNO	PNAME	GENDER	AGE	TRAVELDATE
P001	112300	AVIRAJ	M	21	2023–10–10
P002	918273	PRAGYA	F	49	2023–09–30
P003	650912	RAMESH	M	75	2023–09–20
P004	301582	SOUMIL	M	25	2023–11–07
P005	403822	SNEHA	F	33	2023–12–25
P006	135790	HEMA	F	NULL	2024–01–18

Table: Trains

TNO	TName	Departure	Arrival
112300	Centenary Express	Delhi	Mumbai
135799	Passenger	Delhi	Alwar
918273	Jammu Mail	Amritsar	Jammu
403822	Sushant Mail	Ranchi	Delhi

Write SQL statements to perform the following tasks:

(i) Display the names of trains that depart from Delhi

Ans.
```
SELECT TName FROM Trains
    WHERE Departure = "Delhi";
```

(ii) Display the Passenger Names, date of travel, and corresponding Train Names for all passengers.

Ans.
```
SELECT PName, TravelDate, TName FROM Passengers P, Trains T
    WHERE P.TNO = T.TNO;
```

(iii) Display the count of male and female passengers.

Ans.
```
SELECT Gender, COUNT(*) FROM Passengers
    GROUP BY Gender
    HAVING Gender IN ('M','F');
```

(iv) Display the name of the passenger whose age is not known.

Ans.
```
SELECT PNAME FROM Passengers
     WHERE AGE IS NULL;
```

(v) Change the Arrival station of Sushant Mail to Patna

Ans.
```
UPDATE Trains
     SET Arrival = "Patna"
     WHERE TName = "Sushant Mail";
```

(vi) Delete the record of passengers who have travelled in the year 2023.

Ans.
```
DELETE FROM Passengers
     WHERE TRAVELDATE LIKE "2023%";
```

Program 32: Create a database Orange in MYSQL and thereafter, create a table Employee with the following records:

ENO	NAME	DOJ	DOB	GENDER	DEPT
1001	Reena	2013–09–02	1991–09–01	F	Production
1002	Sahir	2013–02–03	1987–03–20	M	NULL
1003	Morjina	2012–12–03	1986–11–14	F	Admin
1004	Zubair	2014–06–09	1985–06–23	M	Sales
1005	Seema	2013–11–18	1987–03–31	F	Admin

Thereafter, write a program that performs the following tasks:

- Display all data from the table
- Display the record of employee whose ENO is accepted from the user
- Change the department of employee, Morjina to HR
- Delete the records of employees who have joined in the year 2012

Ans.
```
# Establishing Connectivity
import mysql.connector
M1 = mysql.connector.connect(host = 'localhost', user = 'root', passwd = 'tiger')
C1 = M1.cursor()
C1.execute("USE Orange")

# Display all data from the table
def displayData():
    C1.execute("SELECT * FROM Employee")
    data = C1.fetchall()
    print(" DATA from table Employee")
```

```
    for i in data:
        print(i)
# Display the record of employee whose ENO is accepted from the user
def searchData():
    eno = int(input("Enter Employee Number"))
    C1.execute("SELECT * FROM Employee where ENO = {}".format(eno,))
    data = C1.fetchall()
    for i in data:
        print(i)
# Change the department of employee, Morjina to HR
def updateData(eno):
    print("Updating Data")
  C1.execute("UPDATE Employee SET dept = '{}' WHERE ENO = {}".format("HR",eno))
    M1.commit()
    displayData()

# Delete the records of employees who have joined in the year 2012
def deleteData():
    joinDate = "2012%"
    print("Deleting Data")
    C1.execute("DELETE FROM Employee WHERE DOJ LIKE('%s')" %(joinDate))
    M1.commit()
    displayData()

# Calling functions ...
displayData()
print()
searchData()

print()
e = int(input("Enter the Employee Number"))
updateData(e)
print()
deleteData()
```

Unsolved

1. Write a user-defined function, named *volumeSphere()*, that takes the value of the radius and returns the volume of a sphere. Make use of the function *volumeSphere()* to write a program that accepts the radius of a sphere as an input from the user and produces the volume of the sphere.
2. Ruma has measured the distance between her house and her friend's house in feet. She wants to write a program in Python that accepts distance in feet and displays it in yards and miles. Help her complete the task.
3. Write a program that functions like a simple calculator. It accepts two numbers (say, num1 and num2) and displays the following menu:

   num1 / num2

   num2 / num1

   num1 // num2

   num2 // num1

   num1 % num2

   num2 %num1

Display the result as per the choice entered by the user.

4. Write a program that accepts your name and percentile scored in an entrance examination and displays the allocated course and institute as per the table given below:

Percentile Scored	Course	Institute
>= 99	B. Tech. Computer Science	IIT Delhi
>= 97 and < 99	B. Tech. IT	IIT Kanpur
<97	B. Tech. Computer Science	NSUT Delhi

5. Write a program to accept a number and check whether it is an Armstrong number or not.

   Note: An Armstrong number is a number that is equal to the sum of the cubes of its digits. For example, 153 is an Armstrong number because $153 = 1^3 + 5^3 + 3^3 = 153$.

6. Write a program to accept a positive integer (n) and a floating point number (x) print sum of following series:

   $x/1! - x^2/2! + x^3/3! - ...$

   The program should keep adding the terms until the absolute value of a term becomes less than 0.001.

7. Write a program to accept a number (say, n) and print n terms the following sequence of numbers:

   1, 2, 4 , 7, 11, 16, 22 (if the value of n is 7)

8. Write a program to display the sum of squares of each digit of a given number.
9. Write programs to display the following patterns:

   a.

```
      *
    *   *
  *       *
*           *
*  *  *  *  *  *
```

b. **********
   *******__*******
   ******____******
   *****______*****
   ****________****
   ____
   ____
   ____

10. Write a program to accept a list of integers, reverse the order of elements in the list, and display the reversed list. Do NOT use any built-in function or apply slicing in your program. For example, for the given list `[1, 4, 2, 6]`, the program should output the list `[6, 2, 4, 1]`
11. Write a program that reads the content of file, `tagore.txt`, and displays only those lines that begin with 'W'.
12. Write a menu-driven program that reads the contents of the file, `tagore.txt` and computes and displays the following:
    a. Number of sentences (assume that a semicolon terminates a sentence)
    b. Number of words
    c. Average word size
13. Write a user-defined function that takes file name as argument and displays the words of the file in such a manner that first word appears in uppercase, second in lowercase, third in uppercase and so on. Develop a program to invoke the function and view the data in the desired format.

    For example, if the file contains:

    This is my book. It is a very good read. You must try it.

    The output will be:

    THIS is MY book. IT is A very GOOD read. YOU must TRY it.
14. Write a program in Python to write records to the binary file, items.dat. The structure of record is as follows:

    ```
    itemData = [iCode, iName, price, qty]
    ```

    The program should accept data from the user and append it to the binary file for n number of records, where n should also be taken as user input.
15. Refer to the binary file, `items.dat` created in ques 14. Write a program to read the records, calculate total amount (price * qty) for each record and display data in following format:

    Item Code:

    Item Name:

    Price:

    Quantity:

    Total Amount:
16. Chanu has created a binary file named, `trips.dat`. She wants to write a function to display only those records from the binary file, that have the value of category as 'Adventure'. Help her complete the task. The data in the binary file has been written using the following list object:

    ```
    trip = [tripId, destination, price, category]
    ```

    To test your program, you must have the fille `trips.dat` available with you.

17. Data in the binary file, `customer.dat` is stored using the dictionary object:

    ```
    custData:
    custData = {accountNo : Name}
    ```

    Write user-defined functions to perform the following tasks:

    (i) To accept data from the user and append it to the binary file. Two examples of user data are given below:

    ```
    {10851465211 : 'Abhijita'}
    {10858546792 : 'Madhurima'}
    ```

    (ii) To read data and display in the following format:

Sr No	Account No	Name
*****	*********	****
1	10851465211	Abhijita
2	10858546792	Madhurima

    Write a menu-driven program to invoke both functions.

18. Abeer has developed a Python program that stores name and date of birth of his friends in a csv file, `friends.csv`. The data in the file is stored in the form of dictionary objects with name as key and date of birth as value. Now, he wants to write a function that will take name as argument and search for it in the csv file to display the respective date of birth. Display appropriate message if the record is not found in the file.

19. Write a function to count the number of records in a csv file, `friends.csv`, created in ques 18.

20. Ranganathan wants to accept 10 numbers and store them into a list. Next, he wants to store only even numbers onto the stack. Write a function that takes the list as an argument, creates the stack as described above, pops and displays all elements of the stack one by one. Also, display the total number of elements that were pushed in the stack.

    For example:

    If the user enters the following numbers:

    1, 8, 5, 2, 90, 4, –4, –10, 40, 99,

    The output will be:

    [40, –10, –4, 4 , 90, 2, 8]

    Total number of elements pushed in to the stack : 7

21. Consider the table Athlete given below:

    Table: Athlete

AthleteId	Name	Gender	Category	Event1	Event2
A002	Anushka	Girl	Sub Junior	100m	NULL
A003	Prakhar	Boy	Senior	High Jump Boys	200m
A004	Piyush	Boy	Senior	Relay	100m
A005	Nyasa	Girl	Junior	Long Jump	Relay
A007	Hamdaan	Boy	Sub Junior	100m	400m
A008	Gurpurab	Boy	Junior	Long Jump	NULL
A009	Manasvi	Girl	Junior	Relay	NULL

Write a program in Python to perform the following tasks

- Display the records of category Sub Junior
- Change the category of Piyush to Junior
- Display the records in alphabetical order of names.
- Delete the records of students who are not participating in Event2.

Project

APPLE STORE MANAGEMENT

Introduction

Management of products/stock for a store of the Multinational tech company like Apple Inc. is definitely not a trivial task in itself.

The program thus concerned is developed in a way so as to provide a smooth access and interface to the store manager and the stock information being dealt with.

The program allows the user to interpret product data which includes addition, modification, deletion and other necessary information of stock records.

Modules & Built-in Function

Modules:

- csv
- os

Built-in Functions:

- csv.reader()
- csv.writer()
- os.remove()
- os.rename()
- csv.writerow()

HARDWARE & SOFTWARE REQUIREMENTS

Hardware requirements:

- Desktop computer
- USB drive (for import)

Software requirements:

- IDLE(python)
- MS Excel
- MS Word

SOURCE CODE:

```
import os
import csv

def aboutUs():
    print("Hey there! Thank you for using AppleUs! We are one of the branches
    of stores established by the Multi-national technology company Apple Inc.
    \n\n\t\t\t\t\tHOW WE STARTED\n\nOur store was exclusively started for the
    sales of Apple products in XXXX city in 20XX. Since then, we've been working
    diligently to provide our best consumer service.\n\n\t\t\t\t\tSERVICE &
    QUALITY\n\nOur goal is to make Apple's Technology bring out a positive impact
    in your lives\n\nThank you for signing in!")

def addRecord():
    print("="*180)
    print("Add records:")
    print("\n")
    file = open("Applestore records.csv","a",newline="\r\n")
```

```
    wrt = csv.writer(file)
    sno = int(input("Enter serial no. of product:"))
    print("\n")
    name = str(input("Enter name of product:"))
    print("\n")
    qty = int(input("Enter quantity of arrived product:"))
    print("\n")
    costPrice = int(input("Enter product's cost price:"))
    print("\n")
    sellingPrice = int(input("enter product's market price:"))
    print("\n")
    record = [sno, name, qty, costPrice, sellingPrice]
    wrt.writerow(record)
    file.close()
    print("Record saved successfully!")
    input("Press enter to continue...")
def modifyRecord():
    print("="*180)
    print("Modify a record")
    print("\n")
    file1 = open("Applestore records.csv","r",newline="\r\n")
    file2 = open("temp.csv","a",newline="\r\n")
    sno = input("Enter serial no. of the product you wish to modify:")
    recordRead = csv.reader(file1)
    recordWrite = csv.writer(file2)
    for record in recordRead:
        if record[0] == sno:
            print("Serial no.=",record[0],"\n")
            print("Product name=", record[1],"\n")
            print("Stock quantity=",record[2],"\n")
            print("Cost Price=",record[3],"\n")
            print("Market price=", record[4],"\n")
            ch = input("Do you wish to modify the given record? (y/n):")
            if ch == "y" or ch == "Y":
                sno = int(input("Enter new serial no.:"))
                name = input("Enter changed name:")
                qty = int(input("Enter new quantity:"))
                costPrice = int(input("Enter new cost price:"))
                sellingPrice = int(input("Enter new market price:"))
                record = [sno,name,qty,costPrice,sellingPrice]
                recordWrite.writerow(record)
                print("Record modified successfully!")
                print("="*81)
```

```
            else:
                recordWrite.writerow(record)
        else:
            recordWrite.writerow(record)
    file1.close()
    file2.close()
    os.remove("Applestore records.csv")
    os.rename("temp.csv","Applestore records.csv")
    input("Press enter to continue...")
def deleteRecord():
    print("="*180)
    print("DELETE RECORD")
    print("\n")
    file1 = open("Applestore records.csv","r",newline='\r\n')
    file2 = open("temp.csv","a",newline="\r\n")
    sno = input("Enter serial no. of the product you wish to delete:")
    recordRead = csv.reader(file1)
    recordWrite = csv.writer(file2)
    for record in recordRead:
        if record[0]== sno:
            print("Serial no.=",record[0],"\n")
            print("Product name=", record[1],"\n")
            print("Stock quantity=",record[2],"\n")
            print("Cost Price=",record[3],"\n")
            print("Market price=", record[4],"\n")
            ch = input("Do you wish to delete this record?(y/n):")
            if ch=="y" or ch=="Y":
                pass
                print("Record deleted successfully!")
                print("="*170)
            else:
                recordWrite.writerow(record)
        else:
            recordWrite.writerow(record)
    file1.close()
    file2.close()
    os.remove("Applestore records.csv")
    os.rename("temp.csv","Applestore records.csv")
    input("Press enter to continue...")

def viewRecord():
    print("="*180)
    print("LIST OF ALL RECORDS:")
```

```
    print("\n")
    file = open("Applestore records.csv","r",newline="\r\n")
    recordRead = csv.reader(file)

    for record in recordRead:
        print("*"*160)
        print("Serial No.","\t\t\t\t",record[0])
        print("Product name:","\t\t\t",record[1])
        print("Stock quantity:","\t\t",record[2])
        print("Debit/cost price:","\t\t",record[3])
        print("Selling Price:","\t\t\t",record[4])
    file.close()
    input("\nPress enter to continue...")

def search():
    print("="*180)
    print("SEARCH RECORD")
    print("\n")
    file = open("Applestore records.csv","r",newline="\r\n")
    sno = input("Enter serial no. of product you wish to search for:")
    searchSNo = csv.reader(file)
    for record in searchSNo:
        if record[0]== sno:
            print("Serial no.=",record[0],'\n')
            print("Product name=", record[1],'\n')
            print("Stock quantity=",record[2],'\n')
            print("Cost Price=",record[3],'\n')
            print("Market price=", record[4],'\n')
    file.close()
    input("Press enter to continue...")

def clear():
    print("="*180)
    print("DELETE ALL RECORDS")
    os.remove("Applestore records.csv")
    print("\nFILE CLEARED!")

def mainMenu():
    choice=0
    while choice!=8:
        print("\n")

print("================================================================================
MAIN MENU ======================================================================")
```

```
        print("1. AppleUs (About Us)\n")
        print("2. Add new product details\n")
        print("3. Modify product details\n")
        print("4. Delete product details\n")
        print("5. Delete/clear out all records of products\n")
        print("6. Search for product details\n")
        print("7. Display all product details\n")
        print("8. Exit\n")

        choice=int(input("CHOOSE DESIRED OPTION:"))
        if choice==1:
            aboutUs()
        elif choice==2:
            addRecord()
        elif choice==3:
            modifyRecord()
        elif choice==4:
            deleteRecord()
        elif choice==5:
            clear()
        elif choice==6:
            search()
        elif choice==7:
            viewRecord()
        elif choice==8:
            print("THANK YOU FOR SIGNING IN TO APPLE US :)")
            print("="*180)
            break
        else:
            print("Wrong Choice Entered!!")

print("------------------------------------------------------------------------
A.P.P.L.E S.T.O.R.E XXX    ------------------------------------------------
---------------\n")
print("########################################################################
RECORD MANAGEMENT SOFTWARE ############################################################
########################")
mainMenu()
```

Viva Voce Questions

1. Consider the following dictionary:

   ```
   stateCapitals = {"Punjab": "Chandigarh", "Uttar Pradesh": "Lucknow", "Tamil Nadu": "Chennai"}
   ```

   Write code to:

   a. Add the state as "Karnataka" and capital as " Bengaluru" in the above dictionary.

   b. Display a list of all the states in `stateCapitals`.

Ans. a. `Caps['Karnataka'] = 'Bengaluru'`

b. `sorted(Caps.items( ))`

2. What will be the output produced on execution of the following code?

   ```
   lst = [1, 3, 6, 82, 5, 7, 11, 92]
   print(lst[2:5])
   ```

Ans. [6, 82, 5]

3. What are the key features of Python programming?

Ans. Python is a versatile and widely-used programming language known for its simplicity and readability. It has many key features that make it popular among developers for a variety of applications. Here are some of the key features of Python programming:

i. Extensive Standard Library

ii. Dynamic Typing

iii. Open Source

iv. Object-Oriented

4. Write a statement in Python to create a dictionary whose keys are 1, 2, 3 and values are Monday, Tuesday and Wednesday respectively.

Ans. `day = {1: 'Monday', 2: 'Tuesday', 3: 'Wednesday'}`

5. A tuple is created as

   ```
   t = (2,5,6,9,8)
   ```

   What will be the value of `sum(t)`?

Ans. 30

6. What are the difference between a list and a tuple?

Ans.

List	Tuples
A list is mutable.	A tuple is immutable.
Lists are enclosed in brackets[] and their elements and size can be changed.	Tuples are enclosed in parethesis() and can not be updated.
Example: L = [10,11,12,13]	Example: S = ("Orange", "education") or S = ("orange", "education")

7. Name the built-in mathematical function / method that is used to return an absolute value of a number.

Ans. `abs( )`

8. What will be the output produced on execution of the following code?

```
>>> name = "Computer Science with Python"
>>> print(name[9:17])
```

Ans. Science

9. What are the built-in types that Python provides?

Ans. Python provides built-in types of two types: mutable and immutable. Mutable types:

- List
- Sets
- Dictionaries

Immutable built-in types:

- Strings
- Tuples
- Numbers

10. Evaluate the following expressions:

a. `6 * 3 + 4**2 // 5 - 8`

b. `10 > 5 and 7 > 12 or not 18 > 3`

Ans. a. 13

a. False

11. What do you understand by the term iteration?

Ans. Repeatation of statement/s finite number of times is known as iteration.

12. What is lambda expression used for in Python?

Ans. It is a single expression, anonymous function often used as inline function.

13. What is the difference between write( ) and writelines( ) methods used for processing text files?

Ans. The write( ) method is be used to write a string to a text file. It does not write a newline even if it appears in the input string. The writelines( ) method is used to write a sequence of strings into a file. When included as part of a string, the writelines( ) method outputs a newline character.

14. What is the purpose of pass statement in Python?

Ans. Pass means no-operation Python statement in Python, the pass statement is ignored by the interpreter. Thus, it is a dummy statement. Often, it is used to leave a slot for the code to be filled in later. Sometimes, it is also used to simplify program logic when no action is required for a particular condition.

15. What is the difference between text files and binary files?

Ans.

Text Files	Binary Files
Data stored as a sequence of characters	Data stored in the form of 0's and 1's
Human readable	Only machine readable
Can be created using any text editor	Can only be created using special methods of a programming language

16. Write a Python code to read the first line of a file "Pyramid.txt" and display the contents.

Ans.

```
pyramid = open('Pyramid.txt', 'r')
inData = pyramid.readline( )
print(inData)
X = File.readline( )
print(X)
```

17. What is the purpose of negative indexes in Python?

Ans. Python sequences can be indexed using both the positive and negative numbers. For positive indexes, 0 is the index of the first element, 1 is the index of the second element, and so on. For negative indexes, –1 is the index of the last element, –2 is the index of the element at the last but one position, and so on. Negative indexes may be used when we do not know the number of elements in a sequence.

18. What is the purpose of load( ) and dump( ) functions used for processing binary files?

Ans. The dump( ) function serializes or pickles a Python object like a list or a dictionary into a stream of bytes for storing it in a binary file.

The load( ) function is used to read an object like a list or a dictionary from a binary file. Thus, the load() function unpickles or un-serializes data from binary file back to the desired object like a list or a dictionary.

19. Write a statement in Python to open a text file MARKER.TXT so that existing contents can be read from it.

Ans.
```
file = open("MARKER.TXT","r")
```
OR
```
file = open("MARKER.TXT","r+")
```

20. What are the rules for local and global variables in Python?

Ans. **Local variables:** variables that are defined inside a function and are accessible only within the body of a function (also called the scope of a function).

Global variables: variable defined outside the functions and can be accessed inside as well as outside of functions. A global variable can be modified inside any function by preceding it with the keyword global.

21. Write a statement in Python to open the text file DATA.TXT so that new contents can be written in it.

Ans.
```
file = open("DATA.TXT","w")
```
OR
```
file = open("DATA.TXT","w+")
```

22. What is a stack?

Ans. Characteristics of Stacks:

- It is a LIFO data structure
- The insertion and deletion happens at one end i.e. from the top of the stack

23. Which function can be used to delete a file in a Python program?

Ans. A file can be deleted by using os.remove (filename) or os.unlink(filename).

24. Write any one advantage and disadvantage of coaxial cable.

Ans. Advantage: It offers high bandwidth thus huge amounts of data can be transferred in parallel.

Disadvantage: It is expensive to install for longer distances.

25. Differentiate between HTTP and HTTPS.

Ans. HTTP does not encrypt the data when transferring whereas HTTPS uses data encryption to transfer data in encrypted mode. As HTTPS provides SSL or TLS Digital Certificate to secure the communication between the server and the client, HTTPS is far more secure than HTTP.

26. What is the use of // operator in Python?

Ans. It is a divide and truncate the quotient operator; for example, 15//4 --> 3, 15//4.0 -->

3.0, 15.0//4.0 --> 3.0. Note that if one (both) of the operands is (are) floating point numbers, it yields a floating point number.

27. Write any two features of star topology.

Ans. a. Each node(device) in the star topology is connected to a central hub.

b. If a node stops working it does not affect the entire network.

28. What do you understand by the term bandwidth in the context of networking?

Ans. The bandwidth is the maximum amount of data that can be transmitted over the Internet connection in a given period of time.

29. What are literals in Python?

Ans. A literal is a fixed value, examples include string literals, numeric literals, and booleanliterals.

30. Name the protocol that is used to send emails.

Ans. SMTP

31. Name The transmission media best suitable for connecting in hilly areas.

Ans. Microwave/radio wave

32. What is a file mode in Python, and why is it important when opening a file?

Ans: A file mode is a string that specifies the purpose or mode of file access (e.g., reading, writing, appending) when opening a file. It is crucial because it determines what operations are allowed on the file.

33. What is the difference between the "seek" and "tell" methods when working with files?

Ans: The "seek" method is used to change the file pointer's position, allowing you to move to a specific location within the file. The "tell" method returns the current position of the file pointer.

34. When should you use the "a" mode when opening a file? What does this mode do to an existing file?

Ans: The "a" mode should be used when you want to append data to an existing file without overwriting its content. It adds data to the end of the file.

35. What is the purpose of the "x" mode in file opening? How does it differ from the "w" mode?

Ans: The "x" mode is used for exclusive creation, meaning it creates a new file but raises an error if the file already exists. It differs from "w" in that it won't overwrite an existing file.

36. Differentiate between Viruses and Worms in context of networking and data communication threats.

Ans. A virus is a piece of software code that can damage your system by replacing or messing up the system files. Computer viruses are like viruses that live in our bodies, copy themselves, and damage cells. A typical virus attaches itself to a program and can spread to another piece of code or file that has not been infected. A worm is a malicious program that replicates itself. So, a networked device could have several copies of a worm infecting many files. Like viruses, to spread a worm for the first time, an email or an instant chat is sent with attachments from seemingly trustworthy senders.

37. Rearrange the following terms in increasing order of data transfer rates.
    Gbps, Mbps, Tbps, Kbps, bps

Ans. Bps, Kbps, Mbps, Gbps, Tbps

38. Write the expanded form of Wi-Fi.

Ans. Wireless Fidelity

39. Give the full forms of the following:
    a. HTTP b. FTP c. VoIP d. SSH

Ans. a. Hyper Text Transfer Protocol
    b. File Transfer Protocol
    c. Voice Over Internet Protocol
    d. Secure Shell

40. What is a Dictionary in Python?

Ans. Dictionary is an important built-in data type in Python. It defines one-to-one relationship between keys and values. Dictionaries contain a pair of keys and their corresponding values.

Dictionaries are indexed by keys.

41. What is DNS?

Ans. The Domain name system (DNS) maps domain names to numerical IP addresses.

42. Which communication mode allows communication in both directions simultaneously?

Ans. Full-duplex Communication allows communication in both directions simultaneously.

43. How will you reverse a list?

Ans. list.reverse()—Reverses items of list in place.

44. How many bits per second are there in 18 Gbps?

Ans. 18,000,000,000 bits per second or 1.8 e+10 bits per second.

45. HTTP stands for?

Ans. HTTP stands for HyperText Transfer Protocol.

46. What is the purpose of `return` statement in Python?

Ans. The purpose of the return statement in Python is to provide a function's result as output when the function is called.

47. Differentiate between a candidate key and a primary key for a relation in SQL.

Ans. A candidate key is any column or combination of columns that could be used as a primary key, whereas a primary key is the chosen candidate key that uniquely identifies rows and enforces entity integrity in the table.

48. What is meant by a NULL value? What are the reasons that lead to their occurrence in database relations?

Ans. The absence of a data item is represented by a special value called the null value. There are three situations which may require the use of null values

(i) When a particular attribute does not apply to an entity,
(ii) Value of an attribute is unknown, although it exists;
(iii) When the value is unknown because it does not exist

49. Differentiate between the count(column) and count(*) function in SQL.

Ans. COUNT(column) counts non-null values in the specified column, while COUNT(*) counts all rows, regardless of whether they contain NULL values or not.

50. What is the difference between the % and _ wildcard characters in SQL LIKE clause?

Ans. % matches zero or more characters, while _ matches exactly one character in SQL's LIKE clause for pattern matching.

	Command	Output	Explanation
Statement 1	select name from scores where name like '%e%'	Ritesh Brijesh Rakesh Byomkesh	All names will b chosen as the like clause specifies the presence of the letter e with any number of characters on either side.
Statement 2	select name from scores where name like '_ _ _ e%'	Ritesh Rakesh	Since the like clause has three _ characters followed by e and then a %, it specifies a word at least 4 characters long with an e.

51. How can we display the contents of text file in reverse order?

Ans. (i) Convert the given file into a list by reading the file in read mode and using the list() method.
(ii) Reverse the list generated in step (a) using the reversed() method.
(iii) print the reversed list element-wise.

```
reversedList = reversed(list(open("filename", "r")))
for line in reversedList:
    print(line)
```

52. How will you remove records having no products from the table "Suppliers" having fields Sno, name, area, products?

Ans. `DELETE FROM Suppliers WHERE products = NULL;`

53. Your friend Ranjana complaints that somebody has created a fake profile on Facebook and defaming her character with abusive comments and pictures.

    Identify the type of cybercrime for these situations.

Ans. Cyber Bullying

54. Differentiate between the append() and extend() methods used for the list.

Ans. **`append()`:** This method adds its argument as a single element to the end of the list. In other words, it appends the entire object as a new item within the list.

`extend()`: The extend() method is used to append each element from an iterable (e.g., a list, tuple, or string) to the end of the list. It effectively combines the two lists, adding all the elements of the iterable to the original list.

55. Name the clause of SQL used to arrange the rows in ascending order based on an attribute.

Ans. `ORDER BY`

56. List the aggregate functions used in SQL.

Ans. SUM, AVG, COUNT, MAX, and MIN

57. What do you understand by Python modules?

Ans. Python provides several modules, each of which contains several useful functions. Often, it is convenient to use predefined functions available in Python modules. However, to use these functions, we need to import the relevant module using the following syntax:

```
import <modulename>
e.g. import math
```

Once a module is imported, we can use all the functions defined in the module by preceding function name by the name of the module and a dot(.) operator as shown below:

```
<modulename>.<functionname>()
e.g. math.pi()
```

58. What is primary key?

Ans. Primary key is a set of one or more attributes that uniquely identifies a tuple in a relation.

59. What is foreign key?

Ans. An attribute or a set of attributes in one table that refer to the primary key of another table is known as a foreign key.

60. What is MySQL?

Ans. MySQL is an open source RDBMS that relies on SQL for processing the data in the database. The database is available for free under the terms of General Public License (GPL).

61. What is RDBMS?

Ans. Relational Database Management System (RDBMS) facilitates access, security and integrity of data, and eliminates data redundancy. For example, MySQL, Oracle, Microsoft Sql Server, etc.

62. How can we remove a table from a SQL database?

Ans. We can remove a table using the DROP statement.

63. How can we add a new column in a table?

Ans. A new column can be added using ALTER TABLE statement.

64. Who is the developer of the Python programming language? In which year was Python initially developed?

Ans. Python was developed by Guido Van Rossum. Python was first developed in the year 1991.

65. Identify and provide the module name to which the following functions belong:

    a. ceil() b. floor c. randint

    d. dump() e. sqrt()

Ans. a. math module b. math module b. random module

    d. pickle module e. math module

66. What is the purpose of the `split()` method in Python strings?

Ans. The purpose of the `split()` method in Python strings is to separate and tokenize a string into a list of substrings based on a specified delimiter.

67. What is the purpose of `pickle` module?

Ans. Python module `pickle` is used for reading and writing data objects to a binary file. Pickling is the process of transforming a Python object into a byte stream for writing the object to a file.

68. What do you understand by ARPANET?

Ans. ARPANET(Advanced Research Projects Agency Network) was the world's first wide area network funded by the US defence department in the 1960s to link various defense institutes. It formed the basis of the modern Internet.

69. Define the term topology. What are the popular network topologies?

Ans. Topology is the pattern of interconnection of nodes in a local area network (LAN). Popular network topologies include Mesh, Ring, Bus, Star and Tree.

70. What do you mean by a modem? Why is it used?

Ans. A modem (Modulator—Demodulator) is a peripheral device that enables a computer to transmit data over, telephone or cable lines.

It combines the modulation and demodulation functions by transforming a digital signal into an analog signal and vice versa. Digital data to be sent from a computer is modulated for transmission over telephone lines. Similarly, at the receiving end, analog data is demodulated into digital form.

71. How is IP address different from MAC address? Discuss briefly.

Ans. A device on the Internet is identified by its unique IP address, such as 142.250.206.99, comprising four numbers separated by dots. It enables the devices on the network to communicate with each other, whereas a media access control address (MAC address) is a unique identifier assigned to a network interface controller (NIC) for use as a network address in communications within a network. A MAC address is a 48-bit identifier usually represented as 12 hexadecimal digits. An example of a MAC address is 00–B0–D0–63–C2–26.

Practice Paper 1

Time: 3 Hrs. ***Max. Marks: 70***

General instructions

1. This question paper contains five sections, Section A to E.
2. All questions are compulsory.
3. Section A have 18 questions carrying 01 mark each.
4. Section B has 07 Very Short Answer type questions carrying 02 marks each.
5. Section C has 05 Short Answer type questions carrying 03 marks each.
6. Section D has 03 Long Answer type questions carrying 05 marks each.
7. Section E has 02 questions carrying 04 marks each. One internal choice is given in Q34 against part (iii) only.
8. All programming questions are to be answered using Python Language only.

SECTION A

1. What will be the last value printed when the following code is executed?

```
for a in range(20,500,20):
    print(a)
```

a. 480 b. 460
c. 500 d. 440

2. What will be the output on executing the following Python statement?

```
"Python">"PYTHON"
```

a. `True` b. `False`
c. `Error` d. `0`

3. What will be the output on executing the following code segment?

```
sample = [2,3,9,12,4]
sample.insert(4,17)
sample.insert(2,23)
print(sample[-4])
```

a. 4 b. 9
c. 12 d. 23

4. Given `A = 2, B = 1, C = 3`, what will be the output on executing the following statement?

```
print((A>B) and (B>C) or(C>A))
```

a. `False` b. `True`
c. `None` d. `Null`

5. What will be the output of the execution of the following code segment?

```
T=('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun')
print(max(T), min(T))
```

a. Sun Sat b. Mon Sun
c. Wed Fri d. Tue Thu

6. Which of the following statement should you use to read four characters from a file object "sample"?

   a. `sample.readlines(4)`
   b. `sample.readline(3)`
   c. `sample.read(4)`
   d. `sample.read(5)`

7. A relational database is a collection of …… .

   a. Attributes
   b. Records
   c. Fields
   d. Tables

8. Which of the following functions gives the total number of rows in a given table?

   a. SUM( )
   b. COUNT ( )
   c. TOTAL ( )
   d. Both a and b

9. Which of the following statement(s) would give an error after executing the following code?

```
STRING = 'WELCOME"                  # Statement 1
SAMPLE =" "                         # Statement 2
for s in range(0,7):
    print(STRING[s])                # Statement 3
print(SAMPLE + STRING)              # Statement 4
```

   a. Statement 1
   b. Statement 2
   c. Statement 3
   d. Statement 4

10. Which of the following types of table constraints will not prevent NULL entries in a table?

    a. UNIQUE
    b. NOT NULL
    c. PRIMARY KEY
    d. None of these

11. Which of the following is the correct syntax of the open() function to read the data from a file named 'file.txt'?

    a. `open('file.txt', 'a')`
    b. `open('file.txt', 'r')`
    c. `open('file.txt', 'w')`
    d. `open('file.txt', 'b')`

12. Which of the following in a table represents a logical relationship among a set of values?

    a. Column
    b. Key
    c. Row
    d. Attribute

13. Which of the following transmission media has the highest bandwidth?

    a. Coaxial cable
    b. Fiber optic cable
    c. Twisted pair cable
    d. None of these

14. Find the output: `print(10,20,30,sep='!',end=",")`

    a. 10,20,30!
    b. 10!20!30!
    c. 10,20,30
    d. 10!20!30,

15. While retrieving records from a MYSQL table in a Python program, which of the following functions is used to get all records …… .

    a. `cursor.fetchmany()`
    b. `cursor.fetchall( )`
    c. `select * from <table name>;`
    d. `cursor.execute( )`

16. Which of the following functions is used to execute a SQL in Python?

    a. `fetch()`
    b. `exec()`
    c. `resultset()`
    d. `execute()`

Q17 and 18 are ASSERTION (A) AND REASONING (R) based questions. Mark the correct choice as:

a. Both A and R are true and R is the correct explanation for A
b. Both A and R are true and R is not the correct explanation for A
c. A is True but R is False
d. A is False but R is True

17 **Assertion (A):** Text files store information in ASCII or Unicode characters.

Reasoning (R): In a text file, there is no delimeter used as EOL character

18. **Assertion (A):** `fetchall()` method returns the records one by one.

Reasoning (R): `fetchone()` method retrieves one record from the result set.

SECTION B

19. Anusri has written a code to read a number and, if the number is even, print half the number; otherwise, print the next number. But her code has errors. Rewrite the correct code and underline the corrections made.

```
number = int(input("Enter a number : ")
if number%2==0
  print("The half number is", number/3)
else:
  print("The next number is:" number+1)
```

20. Give two differences between packet switching and message switching.

OR

Distinguish between web browser and web server.

21. What will be the output of the execution of the following code segments?

```
(a)
t1 = (2,3,4,5,6)
print(t1.index(4))
(b)
t = [1,2,3,4,2]
print(t.count(2))
print(len(t))
print(t.index(2))
if 3 in t:
     print("Bye")
```

22. Explain the concept of candidate keys with the help of an example.

23. a. Write the full forms of the following:

(i) HTTP (ii) FTP

b. Explain the purpose of gateway.

24. What will be the output of the execution of the following code segment?

```
Values=[10,20,30,40]
for Val in Values:
    for I in range(1, Val%9):
        print(I,"* ",end="")
    print()
```

OR

What will be the output of the execution of the following code segment?

```
def increment(n):
    n.append([49])
```

```
    return n[0], n[1], n[2], n[3]
L = [23, 35, 47]
m1, m2, m3, m4 = increment(L)
print(L)
print(m1, m2, m3, m4)
print(L[3]== m4)
```

25. Mention any two differences between an equi-join and a natural join.

OR

Give the syntax of any two DML statements.

SECTION C

26. a. Consider the table STUDENT given below. What will be the output produced on executing the following SQL statements?

Table: STUDENT

ROLL NO	NAME	CLASS	DOB	GENDER	CITY	MARKS
1	Nanda	10	1995–06–06	M	Agra	551
2	Saurabh	12	1993–05–07	M	Mumbai	462
3	Sonia	11	1994–05–06	F	Delhi	400
4	Trisha	12	1995–08–08	F	Mumbai	450
5	Prakash	12	1995–10–07	M	Delhi	369
6	Pooja	11	1994–12–12	F	Delhi	250
7	Neha	10	1995–12–08	F	Mumbai	377
8	Nishant	10	1995–06–12	M	Mumbai	489

(i) `SELECT NAME FROM STUDENT WHERE CITY = 'Delhi';`

(ii) `SELECT CLASS, DOB, CITY FROM STUDENT WHERE MARKS BETWEEN 490 AND 551;`

(iii)
```
SELECT NAME, CLASS, MARKS, COUNT(*) FROM STUDENT GROUP BY CLASS HAVING
MARKS > 450;
```

(iv) `SELECT ROLLNO FROM STUDENT WHERE GENDER = 'F' AND CITY = 'MUMBAI';`

b. Consider the following tables:

Table: STUDENT

ID	NAME	CLASS	CITY
1	Deepak	11	Delhi
2	Mohan	12	Mumbai
3	Sonali	11	Delhi

Table: RECORD

ID	MARKS	SUBCODE
1	89	101
2	90	102
3	80	103

What will be the output produced on executing the following SQL statement?

```
SELECT STUDENT.ID, STUDENT.NAME, RECORD.MARKS, RECORD.SUBCODE
FROM STUDENT
```

```
JOIN RECORD
ON STUDENT.ID = RECORD.ID;
```

27. A text file `QUOTES.TXT` has the following data written in it:

Living a life you can be proud of

Doing your best

Spending your time with people and activities that are important to you

Standing up for things that are right even when it's hard

Becoming the best version of you

Write a user defined function `COUNTWORDS()` to display the total number of words present in the file.

OR

Write a method/function `ISTOUPCOUNT()` in Python to read contents from a text file `WRITER.TXT`, to count and display the occurrence of the word "IS" or "TO" or "UP".

For example:

If the content of the file is:

IT IS UP TO US TO TAKE CARE OF OUR SURROUNDING. IT IS NOT POSSIBLE ONLY FOR THE GOVERNMENT TO TAKE RESPONSIBILITY

The method/function should display Count of IS TO and UP is 6.

28. a. Consider two tables COMPANY and CUSTOMER given below:

Table: COMPANY

CID	CNAME	CITY	PRODUCTNAME
111	SONY	DELHI	TV
222	NOKIA	MUMBAI	MOBILE
333	ONIDA	DELHI	TV
444	SONY	MUMBAI	MOBILE
555	BLACKBERRY	CHENNAI	MOBILE
666	DELL	DELHI	LAPTOP

Table: CUSTOMER

CUSTID	NAME	PRICE	QTY	CID
101	ROHAN	70000	20	222
102	DEEPAK	50000	10	666
103	MOHAN	30000	5	111
104	SAHIL	35000	3	333
105	NEHA	25000	7	444
106	SONAL	20000	5	333
107	ARUN	50000	15	666

Write SQL commands to perform the following operations:

(i) To display those company names which have price more than 35000

(ii) To display the name of the companies in reverse alphabetical order

(iii) To increase the price by 1000 for those customers whose name starts with 'S'

(iv) To display the product name and city of all the products from the COMPANY table

(b) Write the SQL statement to view the structure of a table named 'MOBILE'.

29. Write a function SWAP2BEST(lst, Size) in Python to modify the content of the list in such a way that the elements, which are multiples of 10, swap with the value present in the very next position in the list.

    For example, if lst = [10, 12, 20, 16]

    Output: [12, 10, 16, 20]

30. Consider a list given below.

    `lst = [10, 13, 15, 16, 20]`

    Define a function `push(lst)` that takes a list `lst` of integer items as an input, pushes only those items onto a stack which are divisible by `five`, and display the stack.

    OR

    Define a function `push(lst)` that takes a list `lst` of numbers as an input, pushes even and odd numbers in two seperate stacks and subsequently concatenates both the stacks to form a new stack. The function should finally display the concatenated stack. If the concatenated stack is empty, then display the appropriate message.

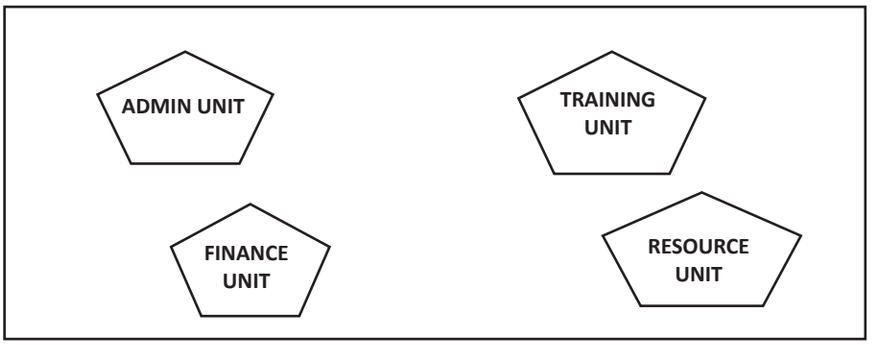
    For example:

    Input: L = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

    Output: [2, 4, 6, 8, 10, 1, 3, 5, 7, 9]

SECTION D

31. "ASAZ" is planning to setup its new campus at Salem for its educational activities. The campus has four units as shown below:



Shortest distances between various buildings are as follows:

ADMIN to TRAINING	65m
ADMIN to RESOURCE	120m
ADMIN to FINANCE	100m
FINANCE to TRAINING	60m
FINANCE to RESOURCE	40m
TRAINING to RESOURCE	50m

Number of Computers installed at various buildings are as follows:

ADMIN	150
FINANCE	25
TRAINING	90
RESOURCE	75

(i) Suggest an ideal cable layout for connecting the above units.

(ii) Suggest the most suitable place i.e. UNIT to install the server for campus.

(iii) Which network device is used to connect the computers in all units?

(iv) Suggest the placement of Repeater in the units of above network.

(v) Suggest the name of the protocol that shall be needed to send email from ADMIN to RESOURCE.

32. a. What will be the output on executing the following code segment?

```
def fun():
    a=10
    y=globals()['a']
    print("inside", a)
    print("y=", y)
a=5
fun()
print("outside", a)
```

b. The function given below updates the following record in the table Employee:

EmpNo – integer

EmpName – string

Designation – string

Salary – integer

Note the following to establish connectivity between Python and MySQL:

- Username is root
- Password is root@123
- The table exists in a MySQL database named record.
- The details (`EmpNo, Designation` and `Salary`) are to be accepted from the user.

Write the following missing statements to complete the code:

Statement 1 – to import the module

Statement 2 – to form the cursor object

Statement 3- to execute the command that updates the record in the table Employee.

```
import _____ #Statement 1
def sql_update():
connection=mysql.connector.connect(host="localhost", user="root",
password="root@123", database="record")
    mycursor=_____ #Statement 2
    EmpNo=int(input("Enter employee number whose record you want to update:
    "))
    Designation=input("Enter designation: ")
    Salary=int(input("Enter salary: "))
    sql_update_query="UPDATE EMPLOYEE SET Designation='{0}', Salary={1} WHERE
    EmpNo={2}".format(Designation, Salary, EmpNo)
    mycursor._____ # Statement 3
    connection.commit()
    print("Record updated successfully")
```

OR

a. What will be the output on executing the following code segment?

```
digit = 6
def mydigit():
    global digit
    digit = 8
    return digit
print(digit)
print(mydigit())
print(digit)
```

b. The code given below deletes the record of the employee from the table Employee whose EmpNo is given:

EmpNo – integer

Note the following to establish connectivity between Python and MySQL:

- Username is root
- Password is root@123
- The table exists in a MySQL database named record.
- The `EmpNo` is to be accepted from the user.

Write the following missing statements to complete the code:

Statement 1 – to connect to the database

Statement 2 – to form the cursor object

Statement 3- to delete the record from the table permanently

```
import mysql.connector
conn=_________________   (host="localhost",user="root",   password="root@123",
database="record") #Statement 1
___________________ #Statement 2
EmpNo=int(input("Enter employee number whose record you want to delete: "))
delete_query="DELETE FROM EMPLOYEE WHERE EmpNo={}".format(EmpNo)
mycursor.execute(delete_query)
___________________    # Statement 3
print("Record deleted successfully")
```

33. Write a function SCOUNT( ) to read the content of binary file "NAMES.DAT" and display number of records (each name occupies 20 bytes in file) where name begins from "S". The content of the binary file is as follows:

SACHIN
AMIT
AMAN
SUSHIL
DEEPAK
HARI SHANKER

OR

Consider the following CSV file (emp.csv):

Eno,name,salary
1,Peter,3500
2,Scott,4000

3,Harry,5000
4,Michael,2500
5,Sam,4200

Define a function `SalaryAbove4000()` to read the content of file emp.csv and display only those records where salary is 4000 or above.

SECTION E

34. Consider the table EMPLOYEE given below:

Table: EMPLOYEE

ENO	ENAME	DEPARTMENT	DESIGNATION	DATEOFJOIN	SALARY
1	Jack	Sales	MGR	2012–09–12	89000
2	Priya	Accts	MGR	2005–04–22	56000
3	Ria	Pers	Clerk	2000–01–09	25000
4	Anil	Pers	Officer	1994–04–03	67000
5	Sumit	Sales	Officer	NULL	19000
6	Akash	Sales	Officer	NULL	20000

Write SQL statements to perform the following:

(i) Display name and department of employees whose name begins with "S".

(ii) Display details of employees whose designation ends with "r".

(iii) Display details of employees whose name has 1st letter "P" and 3rd letter "i".

OR(Option for part (iii) only)

(iii) Display name, department, salary of employees whose department name ends with "S".

(iv) Delete the employee record whose salary is 20000.

35. Anuj Kumar of class 12 is writing a program to create a CSV file "user.csv" which will contain user name and password for some entries. He has written the following code. As a programmer, help him to successfully execute the given task.

```
import _____ # Statement 1
def addCsvFile(UserName,PassWord):
        # to write / add data into the CSV file
        f=open('user.csv', 'a')
        newFileWriter = csv.writer(f)
        newFileWriter.writerow([UserName,PassWord])
        f.close() #csv file reading code
def readCsvFile(): # to read data from CSV file
        usr = open('user.csv', 'r')
        readUser = csv._____(usr) # Statement 2
        lineNum = 0
        for row in readUser:
            if lineNum == 0:
                for value in row:
```

```
                print(value, end="\t")
            else:
                print()
                for value in row:
                    print(value, end="\t")
            lineNum+=1
        usr._____  #Statement 3
addCsvFile("Arjun", "123@456")
addCsvFile("Arunima", "aru@nima")
addCsvFile("Frieda","myname@FRD")
readCsvFile()            # Statement 4
```

(i) Name the module he should import in Statement 1.

(ii) Fill in the blank in Statement 2 to read the data from a csv file.

(iii) Fill in the blank in Statement 3 to close the file.

(iv) Write the output he will obtain while executing Statement 4.

Practice Paper 2

Time: 3 Hrs. ***Max. Marks: 70***

- Please check that this question paper contains 11 printed pages.
- Q.P. Code given on the right hand side of the question paper should be written on the title page of the answer-book by the candidate.
- Please check that this question paper contains 35 questions.
- Please right down the serial number of the question in the answer-book before attempting it.
- 15 minute time has been allotted to read this question paper. The question paper will be distributed at 10.15 a.m. From 10.15 a.m. to 10.30 a.m., the candidate will read the question paper only and will not write any answer on the answer-book during the period.

General instructions

1 This question paper contains five sections, Section A to E.
2. All questions are compulsory.
3. Section A has 18 questions carrying 01 mark each.
4. Section B has 07 Very Short Answer type questions carrying 02 marks each.
5. Section C has 05 Short Answer type questions carrying 03 marks each.
6. Section D has 02 questions carrying 04 marks each.
7. Section E has 03 questions carrying 05 marks each.
8. All programming questions are to be answered using Python Language only.

Section-A

1. State True or False:

   "In Python, there are three control statements—break, continue and pass."

2. Which one of the following symbol may be filled in the blank in the following SQL query?

   ```
   SELECT name FROM stu WHERE subject LIKE '_______ Computer Science';
   ```

   a. $ b. _

   c. || d. %

3. What will be the output produced by executing the following statements?

   ```
   >>>l1=[1,2,3]
   >>>l1.append([5,6])
   >>>l1
   ```

   a. [1,2,3,4,5] b. [1,4,5]]

   c. [[4,5]] d. [1, 2, 3, [5, 6]]

4. What will be the output produced by executing the following statement?

   ```
   print(3-2**2**3+99/11)
   ```

   a. –244.0 b. –245.0

   c. 244.0 d. 247.0

5. Consider the following query:

   ```
   SELECT pname FROM product WHERE profit _____ NULL;
   ```

   Which comparison operator may be used to fill the blank space in above query?

   a. `if` b. `IS/IS Not`

   c. = d. `LIKE`

6. What will be the output produced by executing the following code segment?

```
name = "Augmented Reality"
print(name.replace('Augmented', 'Virtual'))
```

a. Virtual Reality
b. Reality Augmented
c. Augmented Virtual
d. Augmented Reality

7. Bluetooth is an example of:

a. Wide area network
b. Mteropolitan Area Network
c. Local area network
d. Personal area network

8. What will be the output produced by executing the following statements?

```
>>> d1={"Aryan":25,"Tiya":23}
>>> print("Tiya" in d1)
```

a. `True`
b. `False`
c. No output
d. `Error`

9. What will be the output produced by executing the following code segment?

```
Str= "Computers"
print(Str.rstrip("ters"))
```

a. Computer
b. Compu
c. Compute
d. compute

10. What will be the output produced by executing the following statements?

```
>>> tup1=(5,6,8,7)
>>> tup2=(5,6,7,8)
>>> print(tup1>tup2)
```

a. `True`
b. `False`
c. `Error`
d. None of these

11. _____ allows a visited website to store its own information about a user on the user's computer.

a. Cookies
b. Spam
c. Malware
d. Adware

12. What will be the output produced by executing the following code segment?

```
def say(message, times = 1):
    print(message * times , end =' ')
say('Hello and')
say('World', 5)
```

a. Hello and WorldWorldWorldWorldWorld
b. Hello and World 5
c. Hello and World,World,World,World,World
d. Hello and HelloHelloHelloHelloHello

13. _____ is a python object that represents an error.

a. Interpreter
b. Compiler
c. Exception
d. Module

14. Which one of the following sublanguages of SQL is used to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database?

a. DDL (Data Definition Language)
b. DML (Data Manipulation Language)
c. Query
d. Relational Schema

15. NIC Stands for—

a. Network identity card
b. Network interface code.
c. National interface card
d. Network interface card

16. Which one of the following command is used to open a file "c:\temp.txt" in writemode only?
    a. `outfile = open("c:\temp.txt", "w")`
    b. `outfile = open("c:\\temp.txt", "w")`
    c. `outfile = open(file = "c:\temp.txt", "w+")`
    d. `outfile = open(file = "c:\\temp.txt", "w+")`

Q17 and 18 are ASSERTION AND REASONING based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation for A
b. Both A and R are true and R is not the correct explanation for A
c. A is True but R is False
d. A is false but R is True

17. **Assertion (A):** Keys in a Python dictionary should be unique.
    Reasoning (R): Only immutable data types can be used as keys.
18. **Assertion (A):** Digital footprint is the trail of data we leave behind when we visit any website (or use any online application or portal) to fill-in data or perform any transaction.
    Reasoning (R) : While online, all of us need to be aware of how to conduct ourselves, how best to relate with others and what ethics, morals and values to maintain.

Section-B

19. (i) Expand the following:
        FTP, HTTP
    (ii) Give one example of each—Guided media and unguided media
20. Write a function that takes two strings as input and returns `True` if two inputted strings are of same length, `False`, otherwise.

OR

Write a function that takes two numbers `num1` and `num2` as input and generates a random number in the range (`num1`,`num2`). Invoke this function to display three random numbers.

21. Identify and correct the errors in the following code snippet.

```
for Name in [Amar, Shveta, Parag]
    IF Name[0] = 'S' :
        print(Name)
```

22. What will be the output produced by executing the following code segment?

```
arr = [7, 8, 9, 10, 11, 12]
for i in range(1, 6) :
    arr [i - 2] = arr [i]
for i in range(0, 6) :
    print(arr [ i ], end = " ")
```

23. Consider the string `str = '  Orange Books'`. Write python statements using inbuilt functions that performs the following:
    (i) To remove all the leading white spaces from the left of the string.
    (ii) To convert uppercase alphabets to lowercase.

24. Write SQL statements to perform the following:
    (i) Create the table `Product` with appropriate data types and constraints.
    (ii) Insert two records in the `Product` table.
25. What will be the output produced by executing the following code segment?

```
def Call (P = 30, Q = 10) :
    P = P + Q
    Q = P - Q
    print(P, '@', Q)
    return P

R = 300
S = 200
R = Call(R, S)
print (R, '@',S)
S = Call(S)
print (R, '@',S)
```

Section-C

26. What will be the output produced by executing the following code segment?

```
my_dict = {}
my_dict[(1,2,4)] = 8
my_dict[(4,2,1)] = 10
my_dict[(1,2)]= 12
sum = 0
for k in my_dict :
    sum += my_dict[k]
print (sum)
print(my_dict)
```

27. Consider the table given below. What will be the output produced on the execution of following SQL statements?

Table: Employee

Empid	Firstname	Lastname	Designation	City	Salary
010	Ravi	Kumar	Manager	GZB	75000
105	Harry	Waltor	Manager	GZB	65000
152	Sam	Tones	Director	Paris	80000
215	Sarah	Ackerman	Manager	Upton	75000
244	Manila	Sengupa	Clerk	New Delhi	50000
300	Robert	Samuel	Clerk	Washington	45000
400	Rachel	Lee	Salesman	New York	32000
441	Peter	Thompson	Salesman	Paris	28000

    (i) `Select Designation , count(*) from Employee Group by Designation Having count(*)>=3;`
    (ii) `Select Max (salary), Min(Salary) from Employee Where City in ('GZB', 'Paris');`
    (iii) `Select Firstname, Lastname from Employee where Firstname like 'R%';`
28. Write a program to count the number of uppercase alphabets present in a text file `Article.txt`.

OR

Write a program that accepts the names of two files `File1` and `File2` from the user and display the content of `FIle1` followed by the content of `File2`.

29. Consider the table GRADUATE given below:

Table: GRADUATE

S.No.	Name	Stipend	Subject	Average	Div
1	Karan	400	Physics	68	1
2	Divakar	450	Computer Sc.	68	1
3	Divya	300	Chemistry	62	2
4	Arun	350	Physics	63	1
5	Sabina	599	Mathematics	70	1
6	John	400	Chemisry	55	2
7	Robert	250	Physics	64	1
8	Rubina	450	Mathematics	68	1
9	Vikas	500	Computer Sc.	1	
10	Mohan	300	Mathematics	57	2

Write SQL statements to perform the following:

(a) List the names of those students who have obtained DIV 1 in ascending order of name.

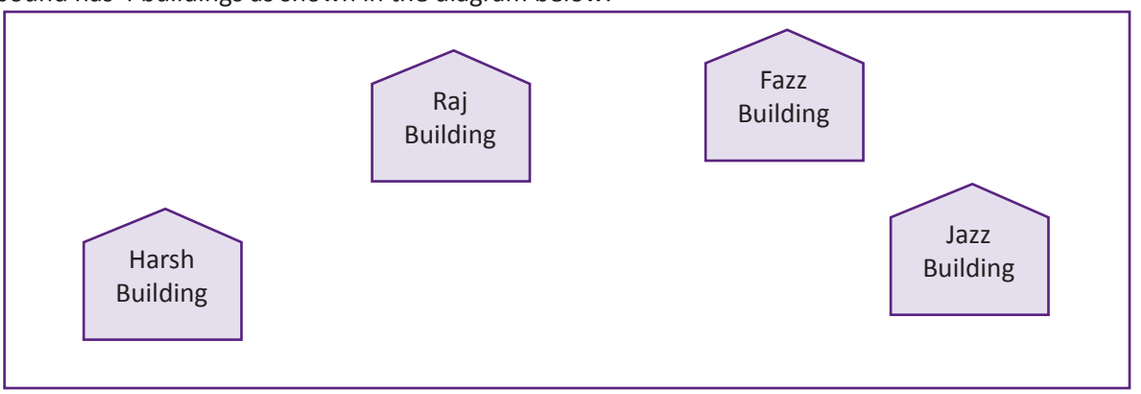
(b) Display a report, listing NAME, STIPEND, SUBJECT and amount of stipend received in a year assuming that the STIPEND is paid every month.

(c) To count the number of students who are either PHYSICS or COMPUTER SC graduates.

30. Write a function `addCustomer(customerNames)` to add customers from the list `customerNames` using push operation of Stack data structure.

Section-D

31. Ravya Industries has set up its new center at Kaka Nagar for its office and web based activities. The company compound has 4 buildings as shown in the diagram below:



Center to center distances between various buildings is as follows:

Harsh Building to Raj Building	50 m
Raj Building to Fazz Building	60 m
Fazz Building to Jazz Building	25 m
Jazz Building to Harsh Building	170 m
Harsh Building to Fazz Building	125 m
Raj Building to Jazz Building	90 m

Number of Computers in each of the buildings is as follows:

Harsh Building	15
Raj Building	150
Fazz Building	15
Jazz Building	25

(a) Suggest a cable layout of connections between the buildings.

(b) Suggest the most suitable place (i.e. building) to house the server of this organization with a suitable reason.

(c) Suggest the placement of the following devices with justification:

(i) Internet Connecting Device/Modem

(ii) Switch

(d) The organization is planning to link its sale counter situated in various parts of the same city. Which type of network out of LAN, MAN or WAN will be formed? Justify your answer.

(e) Suggest a device/software to be installed to take care of data security.

32. (i) Differentiate between r and w file modes in Python.

(ii) Write a Python program to display the size of a file after removing EOL characters, leading and trailing white spaces and blank lines.

OR

(Option for part (ii) only)

(iii) Write a function `myCount()` in Python to read the text file "DATA.TXT" and count the number of times "my" occurs in the file.

33. (i) Define the term Foreign key. What is the purpose of foreign key in relational model?

(ii) ABC Infotech Pvt. Ltd. needs to store, retrieve and delete the records of its employees. Develop an interface that provides front-end interaction through Python, and stores and updates records using MySQL.

The operations on MySQL table "emp" involve reading, searching, updating and deleting the records of employees.

Write a program to read and fetch all the records from EMP table having salary more than 70000.

Section-E

34. Consider two tables `INTERIORS` and `NEWONES` given below:

Table: INTERIORS

No.	ItemName	Type	DateofStock	Price	Discount
1	Red rose	Double bed	23/02/02	32000	15
2	Soft touch	Baby cot	20/01/02	10	
3	Jerry's home	Baby cot	19/02/02	8500	10
4	Rough wood	Office Table	01/01/02	20000	20
5	Comfort zone	Double bed	12/01/02	15000	20
6	Jerry look	Baby cot	24/02/02	7000	19
7	Lion king	office Table	20/02/02	16000	20
8	Royal tiger	Sofa	22/02/02	30000	25
9	Park sitting	Sofa	13/12/01	9000	15
10	Dine Paradise	Dining Table	19/02/02	11000	15

Table: NEWONES

No.	ItemName	Type	DateofStock	Price	Discount
11	White wood	Double Bed	23/03/03	20000	20
12	James 007	Sofa	20/02/03	15000	15
13	Tom look	Baby cot	21/02/13	7000	10

a. Write SQL statements to perform the following:

b. To list the ITEMNAME which are priced at more than 10,000 from the INTERIORS table.

c. To list ITEMNAME and TYPE of those items, in which DATEOFSTOCK is before 22/01/02 from the INTERIERS table in the descending order of ITEMNAME.

d. To display ITEMNAME and DATEOFSTOCK of those items, in which the discount Percentage is more than 15 from INTERIORS table.

35. Create a CSV file "Groceries" to store information of different items existing in a shop. The information is to be stored w.r.t. each item code, name, price, qty. Write a program to accept the data from user and store it permanently in CSV file.

CBSE Sample Question Paper

Class: XII Session: 2023-24

COMPUTER SCIENCE

Code 083

Maximum Marks: 70 **Time: 3 hours**

General instructions

- Please check this question paper contains 35 questions.
- The paper is divided into 5 Sections- A, B, C, D and E.
- Section A, consists of 18 questions (1 to 18). Each question carries 1 Mark.
- Section B, consists of 7 questions (19 to 25). Each question carries 2 Marks.
- Section C, consists of 5 questions (26 to 30). Each question carries 3 Marks.
- Section D, consists of 3 questions (31 to 33). Each question carries 5 Marks.
- Section E, consists of 2 questions (34 to 35). Each question carries 4 Marks.
- All programming questions are to be answered using Python Language only.

Section-A

1. State True or False:

   "In a Python program, if a `break` statement is given in a nested loop, it terminates the execution of all loops in one go." [1]

2. In a table in MYSQL database, an attribute `A` of datatype `varchar(20)` has the value `"Keshav"`. The attribute B of datatype `char(20)` has value `"Meenakshi"`. How many characters are occupied by attribute `A` and attribute `B`? [1]

   a. 20,6 b. 6,20
   c. 9,6 d. 6,9

3. What will be the output of the following statement: [1]

```
print(3-2**2**3+99/11)
```

   a. 244 b. 244.0
   c. –244.0 d. Error

4. Select the correct output of the code: [1]

```
s = "Python is fun"
l = s.split()
s_new = "-".join([l[0].upper(), l[1], l[2].capitalize()])
print (s_new)
```

   Options:

   a. `PYTHON-IS-Fun` b. `PYTHON-is-Fun`
   c. `Python-is-fun` d. `PYTHON-Is-Fun`

5. In MYSQL database, if a table, **Alpha** has degree 5 and cardinality 3, and another table, **Beta** has degree 3 and cardinality 5, what will be the degree and cardinality of the Cartesian product of **Alpha** and **Beta**? [1]

   a. 5,3 b. 8,15
   c. 3,5 d. 15,8

6. Riya wants to transfer pictures from her mobile phone to her laptop. She uses Bluetooth Technology to connect two devices. Which type of network will be formed in this case? [1]

   a. PAN b. LAN
   c. MAN d. WAN

7. Which of the following will delete key-value pair for key = "Red" from a dictionary D1? [1]

   a. `delete D1("Red")`
   b. `del D1["Red"]`
   c. `del.D1["Red"]`
   d. `D1.del["Red"]`

8. Consider the statements given below and then choose the correct output from the given options: [1]

```
pride="#G20 Presidency"
    print(pride[-2:2:-2])
```

   a. ndsr
   b. ceieP0
   c. ceieP
   d. yndsr

9. Which of the following statement(s) would give an error during execution of the following code? [1]

```
tup = (20,30,40,50,80,79)
print(tup) #Statement 1
print(tup[3]+50) #Statement 2
print(max(tup)) #Statement 3
tup[4]=80 #Statement 4
```

   a. Statement 1
   b. Statement 2
   c. Statement 3
   d. Statement 4

10. What possible outputs(s) will be obtained when the following code is executed? [1]

```
import random
myNumber=random.randint (0,3)
COLOR=["YELLOW", "WHITE", "BLACK", "RED"]
for I in COLOR:
      for J in range (1, myNumber):
        print (I, end = "*")
      print ()
```

   a. RED*
      WHITE*
      BLACK*
      RED*
   b. YELLOW*
      WHITE*
      BLACK*
      RED*
   c. WHITE* WHITE*
      YELLOW* YELLOW*
      BLACK* BLACK*
      RED* RED*
   d. YELLOW*
      WHITE*WHITE*
      BLACK* BLACK* BLACK*
      RED* RED* RED* RED* RED*

11. Fill in the blank: [1]

    The modem at the sender's computer end acts as a _____.

   a. Model
   b. Modulator
   c. Demodulator
   d. Convertor

12. Consider the code given below: [1]

```
b = 100
def test (a):
    _____ # missing statement
    b = b + a
    print (a,b)
test (10)
print (b)
```

    Which of the following statements should be given in the blank for #Missing Statement, if the output produced is 110?

   a. `global a`
   b. `global b=100`
   c. `global b`
   d. `global a=100`

13. State whether the following statement is True or False: [1]

An exception may be raised even if the program is syntactically correct.

14. Which of the following statements is FALSE about keys in a relational database? [1]

a. Any candidate key is eligible to become a primary key.

b. A primary key uniquely identifies the tuples in a relation.

c. A candidate key that is not a primary key is a foreign key.

d. A foreign key is an attribute whose value is derived from the primary key of another relation.

15. Fill in the blank: [1]

In case of _____ switching, before a communication starts, a dedicated path is identified between the sender and the receiver.

16. Which of the following functions changes the position of file pointer and returns its new position? [1]

a. `flush()` b. `tell()`

c. `seek()` d. `offset()`

Q17 and 18 are ASSERTION AND REASONING based questions. Mark the correct choice as

a. Both A and R are true and R is the correct explanation for A

b. Both A and R are true and R is not the correct explanation for A

c. A is True but R is False

d. A is false but R is True

17. Assertion(A): List is an immutable data type [1]

Reasoning(R): When an attempt is made to update the value of an immutable variable, the old variable is destroyed and a new variable is created by the same name in memory.

18. Assertion(A): Python standard library consists of number of modules. [1]

Reasoning(R): A function in a module is used to simplify the code and avoids repetition.

Section B

19. (i) Expand the following terms: [1+1=2]

POP3 , URL

(ii) Give one difference between XML and HTML.

20. The code given below accepts a number as an argument and returns the reverse number. Observe the following code carefully and rewrite it after removing all syntax and logical errors. Underline all the corrections made. [2]

```
define revNumber (num):
   rev = 0
   rem = 0
   While num > 0:
       rem == num % 10
       rev = rev*10 + rem
       num = num//10
       return rev
print(revNumber(1234))
```

21. Write a function `countNow(PLACES)` in Python, that takes the dictionary, places as an argument and displays the names (in uppercase)of the places whose names are longer than 5 characters. [2]

For example, Consider the following dictionary

```
PLACES = {1:"Delhi",2:"London",3:"Paris",4:"New York",5:"Doha"}
```

The output should be:

LONDON

NEW YORK

OR

Write a function, `lenWords(STRING)`, that takes a string as an argument and returns a tuple containing length of each word of a string.

For example, if the string is "`Come let us have some fun`", the tuple will have (`4`, `3`, `2`, `4`, `4`, `3`)

22. Predict the output of the following code: [2]

```
S = "LOST"
L = [10,21,33,4]
D = { }
for I in range(len (S)):
  if I%2==0:
     D[L.pop()] = S[I]
  else:
     D[L.pop()] = I+3
for K,V in D.items():
   print (K,V, sep="*")
```

23. Write the Python statement for each of the following tasks using BUILT-IN functions/methods only: [1+1=2]

(i) To insert an element `200` at the third position, in the list `L1`.

(ii) To check whether a string named, `message` ends with a full stop / period or not.

24. Ms. Shalini has just created a table named "`Employee`" containing columns `Ename`, `Department` and `Salary`. [2]

After creating the table, she realized that she has forgotten to add a primary key column in the table. Help her in writing an SQL command to add a primary key column `EmpId` of integer type to the table `Employee`.

Thereafter, write the command to insert the following record in the table:

```
EmpId-999
Ename-Shweta
Department: Production
Salary: 26900
```

25. Predict the output of the following code: [2]

```
def Changer (P, Q = 10) :
    P = P / Q
    Q = P % Q
    return P
A = 200
B = 20
A = Changer(A,B)
print(A,B, sep='$')
B = Changer(B)
print (A,B, sep='$', end='###')
```

Section C

26. Predict the output of the Python code given below: [3]

```
Text1 = "IND-23"
Text2 = ""
I = 0
while I < len (Text1):
   if Text1[I] >= "0" and Text1[I] <= "9":
```

```
        Val = int(Text1[I])
        Val = Val + 1
        Text2 = Text2 + str(Val)
    elif Text1[I] >= "A" and Text1[I] <= "Z":
        Text2 = Text2 + (Text1[I+1])
    else:
        Text2 = Text2 + "*"
    I += 1
print(Text2)
```

27. Consider the table CLUB given below and write the output of the SQL queries that follow. [1×3 = 3]

CID	CNAME	AGE	GENDER	SPORTS	PAY	DOAPP
5246	AMRITA	35	FEMALE	CHESS	900	2006–03–27
4687	SHYAM	37	MALE	CRICKET	1300	2004–04–15
1245	MEENA	23	FEMALE	VOLLEYBALL	1000	2007–06–18
1622	AMRIT	28	MALE	KARATE	1000	2007–09–05
1256	AMINA	36	FEMALE	CHESS	1100	2003–18–05
1720	MANJU	33	FEMALE	KARATE	1250	2004–04–10
2321	VIRAT	35	MALE	CRICKET	1050	2005–04–30

(i) `SELECT COUNT(DISTINCT SPORTS) FROM CLUB;`

(ii) `SELECT CNAME, SPORTS FROM CLUB WHERE DOAPP<"2006-04-30" AND CNAME LIKE "%NA";`

(iii) `SELECT CNAME, AGE, PAY FROM CLUB WHERE GENDER = "MALE" AND PAY BETWEEN 1000 AND 1200;`

28. Write a function in Python to read a text file, `Alpha.txt` and displays those lines which begin with the word '`You`'. [3]

OR

Write a function, `vowelCount()` in Python that counts and displays the number of vowels in the text file named `Poem.txt`.

29. Consider the table `Personal` given below: [1*3 = 3]

`Table: Personal`

P_ID	Name	Desig	Salary	Allowance
P01	Rohit	Manager	89000	4800
P02	Kashish	Clerk	NULL	1600
P03	Mahesh	Superviser	48000	NULL
P04	Salil	Clerk	31000	1900
P05	Ravina	Superviser	NULL	2100

Based on the given table, write SQL queries for the following:

(i) Increase the salary by 5% of personals whose allowance is known.

(ii) Display Name and Total Salary (sum of Salary and Allowance) of all personals. The column heading 'Total Salary' should also be displayed.

(iii) Delete the record of Supervisors who have salary greater than 25000

30. A list, NList contains following record as list elements: [3]

`[City, Country, distance from Delhi]`

Each of these records are nested together to form a nested list. Write the following user defined functions in Python to perform the specified operations on the stack named `travel`.

(i) **`Push_element(NList)`**: It takes the nested list as an argument and pushes a list object containing name of the city and country, which are not in India and distance is less than 3500 km from Delhi.

(ii) **`Pop_element()`**: It pops the objects from the stack and displays them. Also, the function should display "Stack Empty" when there are no elements in the stack.

For example: If the nested list contains the following data:

```
NList=[["New York", "U.S.A.", 11734],
["Naypyidaw", "Myanmar", 3219],
["Dubai", "UAE", 2194],
["London", "England", 6693],
["Gangtok", "India", 1580],
["Columbo", "Sri Lanka", 3405]]
```

The stack should contain:

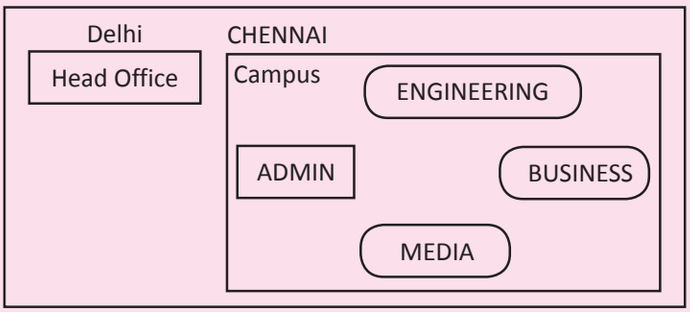
```
['Naypyidaw', 'Myanmar'],
['Dubai', 'UAE'],
['Columbo', 'Sri Lanka']
```

The output should be:

```
['Columbo', 'Sri Lanka']
['Dubai', 'UAE']
['Naypyidaw', 'Myanmar']
Stack Empty
```

Section D

31. Meticulous EduServe is an educational organization. It is planning to setup its India campus at Chennai with its head office at Delhi. The Chennai campus has 4 main buildings – ADMIN, ENGINEERING, BUSINESS and MEDIA [1×5=5]



Block to Block distances (in Mtrs.)

From	To	Distance
ADMIN	ENGINEERING	55 m
ADMIN	BUSINESS	90 m
ADMIN	MEDIA	50 m
ENGINEERING	BUSINESS	55 m
ENGINEERING	MEDIA	50 m
BUSINESS	MEDIA	45 m
DELHI HEAD OFFICE	CHENNAI CAMPUS	2175 km

Number of computers in each of the blocks/Center is as follows:

ADMIN	110
ENGINEERING	75
BUSINESS	40
MEDIA	12
DELHI HEAD OFFICE	20

a. Suggest and draw the cable layout to efficiently connect various blocks of buildings within the CHENNAI campus for connecting the digital devices.

b. Which network device will be used to connect computers in each block to form a local area network?

c. Which block, in Chennai Campus should be made the server? Justify your answer.

d. Which fast and very effective wireless transmission medium should preferably be used to connect the head office at DELHI with the campus in CHENNAI?

e. Suggest a device/software to be installed in the CHENNAI Campus to take care of data security.

32. (i) Differentiate between r+ and w+ file modes in Python. [2+3=5]

(ii) Consider a file, `SPORT.DAT`, containing records of the following structure:

`[SportName, TeamName, No_Players]`

Write a function, `copyData()`, that reads contents from the file `SPORT.DAT` and copies the records with Sport name as **`"Basket Ball"`** to the file named `BASKET.DAT`. The function should return the total number of records copied to the file `BASKET.DAT`.

OR

(Option for part (ii) only)

A Binary file, `CINEMA.DAT` has the following structure:

`{MNO:[MNAME, MTYPE]}`

Where

`MNO`–Movie Number

`MNAME`–Movie Name

`MTYPE` is Movie Type

Write a user defined function, `findType(mtype)`, that accepts `mtype` as parameter and displays all the records from the binary file `CINEMA.DAT`, that have the value of Movie Type as `mtype`.

33. (i) Define the term Domain with respect to RDBMS. Give one example to support your answer. [1+4=5]

(ii) Kabir wants to write a program in Python to insert the following record in the table named `Student` in MYSQL database, `SCHOOL`:

- `rno`(Roll number)-integer
- `name`(Name)-string
- `DOB` (Date of birth)–Date
- `Fee`–float

Note the following to establish connectivity between Python and MySQL:

- Username-`root`
- Password-`tiger`
- Host-`localhost`

The values of fields `rno`, `name`, `DOB` and `fee` has to be accepted from the user.

Help Kabir to write the program in Python.

Section E

34. Consider the tables PRODUCT and BRAND given below: [1×4=4]

Table: PRODUCT

PCode	PName	UPrice	Rating	BID
P01	Shampoo	120	6	M03
P02	Toothpaste	54	8	M02
P03	Soap	25	7	M03
P04	Toothpaste	65	4	M04
P05	Soap	65	5	M05
P06	Shampoo	245	6	M05

Table: BRAND

BID	BName
M02	Dant Kanti
M03	Medimix
M04	Pepsodent
M05	Dove

Write SQL queries for the following:

(i) Display product name and brand name from the tables PRODUCT and BRAND.

(ii) Display the structure of the table PRODUCT.

(iii) Display the average rating of Medimix and Dove brands

(iv) Display the name, price, and rating of products in descending order of rating.

35. Vedansh is a Python programmer working in a school. For the Annual Sports Event, he has created a csv file named `Result.csv`, to store the results of students in different sports events. The structure of `Result.csv` is : [4]

`[St_Id, St_Name, Game_Name, Result]`

Where

`St_Id` is Student ID (integer)

`ST_name` is Student Name (string)

`Game_Name` is name of game in which student is participating(string)

Result is result of the game whose value can be either '`Won`', '`Lost`' or '`Tie`'

For efficiently maintaining data of the event, Vedansh wants to write the following user defined functions:

`Accept()`–to accept a record from the user and add it to the file `Result.csv`. The column headings should also be added on top of the csv file.

`wonCount()`–to count the number of students who have won any event.

As a Python expert, help him complete the task.

Scan for the answers of CBSE Sample Question Paper.

Glossary

- **Aggregate Functions:** SQL provides several functions where the selection criterion applies to an aggregate of tuples instead of a single tuple.
- **ALTER TABLE Statement:** ALTER TABLE statement allows us to add, change, and delete attributes of a table.
- **Alternate Keys:** A candidate key that is not chosen as the primary key is called an alternate key.
- **Arithmetic Operators:** Arithmetic operators are used to build mathematical expressions involving numeric values.
- **Attribute:** An attribute defines the characteristics of an entity. Each column of a table represents a unique attribute.
- **Bandwidth:** A channel's bandwidth (BW) denotes the frequencies that travel through it (Unit: Hz). Thus, it is the difference between the channel's highest frequency and the lowest frequency.
- **Baud rate:** Baud rate refers to the number of signal units transmitted in a second. Baud rate is computed by dividing bit rate by the number of bits per signal unit.
- **Bit Rate or Data Transfer Rate:** Bit Rate or Data Transfer Rate is the number of bits transmitted per second.
- **`break` statement:** On execution of a break statement, Python transfers the control to the statement following the loop in which the break statement occurs.
- **Bus Topology:** In bus topology, various nodes are connected to a long cable, called backbone cable.
- **Candidate Key:** A set of attributes that uniquely identify each tuple in a relation is called its candidate key.
- **Cardinality:** The number of rows in the entity set of a relation defines the cardinality of the relation.
- **Circuit Switching:** Circuit switching requires a dedicated connection between the source and the destination.
- **Complex:** A complex number is of the form x+yj and comprises a pair of floating point numbers.
- **Computer Network:** A computer network is a group of related computers that are linked together.
- **Connecting to MySQL Server from Python:** To access the MySQL database from Python, we need to ensure that the MySQL server runs at the backend. For connecting to MySQL server, we will use Python MySQL client library `pymysql`.
- **Constraint:** Constraint specifies the restrictions imposed on the data stored.
- **continue statement:** The continue statement is used to skip the following statements in the body of the for loop (while loop), and the execution of the loop begins again for the next iteration.
- **Cookies:** When a user visits a website via his/her browser, the website attempts to enhance the user's experience on the website. For this purpose, it sends to the user's browser, some pieces of text called cookies.
- **Cyber Laws:** Cyber laws are the statutory provisions against various forms of cybercrimes.
- **Data Definition Language (DDL):** Data Definition Language (DDL) is used for describing database structure.
- **Data Manipulation Language (DML):** Data Manipulation Language (DML) is used to retrieve and modify data.
- **Data Types:** A data type refers to a type of data and the operations that can be performed on the objects of that type.
- **Database Administrator (DBA):** Person who manages the entire database by controlling access and acquiring resources (both hardware and software).
- **DBMS:** Database Management System (DBMS) is a collection of programs that help define, construct, and manipulate the database.
- **Degree:** The number of attributes in a relation is known as the degree of relation.
- **Dictionaries:** A dictionary is an unordered set of `key:value` pairs.

- **Dictionary (Mapping):** A dictionary `(dict)` is an unordered set of key-value pairs enclosed in curly brackets: {}.
- **Domain Names:** A device on the Internet is identified by its unique IP address, such as 142.250.206.99, comprising four numbers separated by dots.
- **DROP TABLE Statement:** The DROP TABLE statement deletes an existing table in a database.
- **`dump()`:** Writes an object to a binary file.
- **Entity Set:** The set of all tuples in a relation is known as the entity set.
- **Entity:** An entity is a real-world object about which data is stored in the database.
- **Exceptions:** Exceptions are erroneous events that arise during the execution of a program, causing a deviation from the normal flow of code.
- **`fetchall()`:** Retrieves all the rows from the result of execution of an SQL query.
- **`fetchmany(n)`:** Retrieves n rows from the result of execution of an SQL query.
- **`fetchone()`:** Retrieves single row from the result of execution of an SQL query.
- **Fibre optic transmission:** Fibre optic transmission works on the principle of total internal reflection. When a light ray travels through the fibre and falls on the boundary of the medium at an angle of incidence greater than the critical angle, it gets reflected back in the medium. It can be trapped for long distances without any loss. When the light falls on a detector, it converts the light pulses into electrical signals.
- **File Transfer Protocol (FTP):** File Transfer Protocol (FTP) is used for transferring files from one machine to another. FTP is based on the client-server model.
- **FileNotFoundError:** It is raised when an attempt is made to open or access a file that does not exist.
- **Finally Clause: `Finally clause`** is used to define a block of code that will always be executed, regardless of whether an exception occurred or not.
- **Firewall:** A firewall is a hardware device or a software mechanism or a combination of both and is placed between two or more different networks.
- **First Out (LIFO):** a stack is a Last In, First Out (LIFO) arrangement.
- **The `flush()` Function:** The `flush()` function can be used to forcefully write the content from Python's buffer to a file without waiting for the user to close the file.
- **`for` statement:** A for statement is used for repeated execution of the loop's body for each occurrence of an element in a sequence.
- **Frequency:** The number of cycles per second defines the frequency of a signal.
- **Full-Duplex Transmission:** In full-duplex transmission mode, the data can be transmitted in both directions simultaneously.
- **`Function close()`:** Function `close()` is used to close a file, once the necessary operations on a file have been carried out.
- **Gateway:** A gateway is a hardware device that operates as a "gate" between networks. A gateway connects the networks that use different protocols.
- **Global Names (Global Variables):** A variable defined outside the functions is known as a global variable.
- **Hacking:** Hacking refers to having unauthorized access to someone's computer or computer network for stealing resources such as passwords or confidential files or causing harm to the network or system. Bad hacker hacks the system with bad intentions, whereas a good hacker tries to hack the system to identify its weaknesses so that they can be isolated.
- **Half-Duplex Transmission:** In half-duplex transmission, the data transmission can happen in both directions. However, when one device sends the data, others can only receive it and vice versa.
- **HTML and XML:** HyperText Markup Language (HTML) is used to design web pages.
- **Identifiers:** A name in Python is also called an identifier as it uniquely identifies a data object, function, or class.

- **Identity Operators:** Identity operators are typically used for checking the data type of an object. The identity operator returns **`True`** if the operands on either side of the operator are the same object or same type, and **`False`** otherwise.
- **`if-elif-else:`** An `elif` clause checks multiple expressions for True and executes a block of code corresponding to an expression that evaluates True. Although there can be an arbitrary number of `elif` clauses following the if clause, an `if-elif-else` statement can include at most one else clause.
- **`if-else:`** An **`if-else`** statement evaluates the condition in the **`if`** clause and will execute the body of this clause only if the condition evaluates to **`True`**, but if the condition evaluates to **`False`**, then the body of the **`else`** clause is executed.
- **IndexError:** This exception occurs whenever we try to access an index that is out of a valid range.
- **`input():`** The `input()` function reads the text entered by a user until a newline is encountered.
- **IP:** Internet Protocol (IP) is a connectionless protocol since no connection is required to be established between source and destination before communication.
- **LAN:** A Local Area Network (LAN) is typically owned privately by an organisation. It can span an area of about 1 Km.
- **LIKE Operator:** The SQL LIKE operator uses wildcard operators % and _ to define a pattern.
- **Lists (list):** A list is a sequence of items, separated by commas and enclosed within square brackets [].
- **Lists:** list comprises a sequence of objects enclosed in square brackets [].
- **`load():`** Reads stream of bytes from a binary file.
- **Local Names (Local Variables):** All the names (variables) defined inside a function are accessible only within the body of a function (also called the scope of a function) and are known as local names or local variables.
- **Local variables:** The variables defined inside a function are called local variables. Variables that are defined inside a function and are accessible only within the body of a function (also called the scope of a function).
- **Logical Operators:** The logical operators **`not`**, **`and`**, and **`or`** are applied to logical operands (also called Boolean operands or Boolean values) **`True`** and **`False`**, and yield a Boolean value.
- **MAN:** A Metropolitan Area Network (MAN) can span a radius of up to 50 kilometres.
- **Microwave Link:** Microwaves are unidirectional and follow a straight path.
- **modular Programming:** The process of dividing a computer program into separate blocks of code or sub-programs (each having a name and functionality) is called modular programming.
- **Mutable Data Types:** A modifiable object and its associated type are called mutable.
- **NameError:** This exception occurs whenever a name that appears in a statement is not found globally.
- **NATURAL JOIN:** SQL supports a NATURAL JOIN operator that removes the duplicate column common to both the tables and positions the common attribute as the first column in the result.
- **Nested List:** A list that contains another list is called a nested list.
- **Network Interface Card (NIC):** A network interface card is a piece of hardware (also known as a network adapter, network interface controller, or LAN adapter) that allows a computer to connect to a network.
- **Network Topologies:** The arrangement of nodes in a network is called topology.
- **Opening Files:** For opening a file, Python provides the built-in function **`open()`**.
- **Packet Switching:** Packet switching eliminates the problem of network congestion by dividing a message into smaller units known as packets.
- **PAN:** A Personal Area Network (PAN) is a computer network that links computers and gadgets within a person's range.
- **`pass` statement:** A pass statement is ignored by the Python interpreter.
- **PermissionError:** It is raised when the program doesn't have the necessary permissions to read from or write to a file.
- **Pop:** Pop operation removes the top element from the stack.

- **POP3:** Post Office Protocol version 3 (POP3) is an email protocol for receiving emails from a remote server and sending them to a local email client.
- **Postfix Notation:** In the postfix notation, operators appear after the operands in the expression.
- **PPP:** The Point-to-Point Protocol (PPP) establishes a direct connection between two communicating devices.
- **Primary Key:** One of the candidate keys is chosen as the primary key.
- **`print()`:** As the computer screen is the default device for displaying the results, the output produced on invoking the **`print()`**.
- **Push:** Push operation places an element on the top of the stack.
- **Python:** Python is an interactive and interpreted language.
- **Reading From Files:** A call to function **`read()`**, reads the entire contents of the file.
- **`readline()`: `readline()`** function returns the contents of the first line from the file.
- **`readlines()`: `readlines()`** returns the entire content of the file from the current position in the form of a list of lines.
- **Referential Integrity:** While working with two or more related tables in a database, SQL enforces the referential integrity constraints.
- **Relational Operators:** A relational operator is used for comparing two values. Execution of a relational operator yields a Boolean value **`True`** or **`False`**.
- **Relationship:** A relationship specifies how two entities are linked (or related) to each other.
- **Remote Login (TELNET):** TELNET stands for TErminaL NETwork. It operates in client-server mode and enables a user working on one system to access a remote system.
- **Repeater:** With an increase in distance, the strength of the signal decreases. A repeater intensifies the strength of the incoming signal before transferring it further.
- **Ring Topology:** In the ring topology, all devices are attached through a cable in the form of a ring (in circular fashion).
- **Runtime errors:** a runtime error occurs during the execution of the program.
- **`seek()`:** Using the method `seek()`, the file object can point to a particular position in the file.
- **SELECT Statement:** The SELECT statement is used to retrieve information from database tables.
- **Sequence Data Types:** Sequence is an ordered collection of items indexed by the positions of the elements.
- **Set:** A set is an unordered and unindexed collection of data items separated by a comma and enclosed within curly brackets: {}.
- **Simple `if`:** if statement allows us to execute a statement only if a specific condition is satisfied.
- **Simplex Transmission:** It is the simplest transmission mode in which the data travels in one direction only.
- **Spams:** Spams are unwanted electronic mails generally sent in bulk over the Internet to recipients.
- **Stack Overflow Condition:** A stack is full when no more elements can be added to it.
- **Stack Underflow Condition:** A stack having no elements is called an empty stack.
- **Star Topology:** The star topology has a central controller known as a hub. All the nodes are connected to the hub.
- **String Slices:** A subsequence of characters of a string is called a slice.
- **Strings (str):** A string refers to a sequence of characters.
- **Structured Query Language (SQL):** Structured Query Language (SQL) is used to create and manipulate relational databases.
- **TCP:** Transmission Control Protocol (TCP) is a connection-oriented protocol. It requires that the communicating devices must establish a connection between themselves before initiating an exchange of information.
- **`tell()`:** The function `tell()` returns the current position of the file object.

- **Text Files:** A text file is a human-readable file that comprises of a sequence of characters stored in the form of ASCII or UNICODE characters.
- **The `help()` Function:** The `help()` function is used to display the documentation of the functions, classes, modules, etc.
- **Router:** A router may be used to interconnect several networks, possibly having different topologies.
- **Transmission Media:** A transmission medium (also called a channel or a link) carries signals (also called data or messages) from one node to another in a network.
- **Trojan Horse:** Trojan Horse is a code that appears to be desirable and useful but ends up harming the system. When executed, such an application creates a backdoor.
- **Try block:** The try block consists of statements that can potentially raise an exception.
- **Tuple:** a row represents an entity's instance (also called occurrence) and is known as a tuple.
- **Tuples:** A tuple is a sequence of elements, separated by commas and usually enclosed between parentheses.
- **Tuples:** A tuple is a sequence of objects (not necessarily of the same type) enclosed in parenthesis: ().
- **TypeError:** This exception occurs when an operation or function is applied to an object of inappropriate type.
- **Types of Errors:** Various types of errors may be generated while creating or executing a program. The process of identifying and removing such errors is known as debugging. The errors are called bugs.
- **Understanding Data Files:** A file is a stream of bytes comprising relevant data.
- **ValueError:** This exception occurs whenever an inappropriate argument value, even though of the correct type, is used in a function call.
- **Virus:** A virus is a software that may harm a computer system by overwriting or corrupting the files.
- **VoIP:** VoIP allows us to make calls using an Internet connection.
- **WAN:** A Wide Area Network (WAN) can span a radius of up to 1000 km. Therefore, a WAN can connect across countries and continents.
- **Web Hosting:** Web Hosting is a service for hosting websites for the users (clients) on the Internet.
- **Web Page, Website, and URL:** A web page is a document that displays multimedia content such as text, images, audio, and video.
- **web pages:** A web browser is a software that enables us to view web pages.
- **Web Scripting:** A web script is a program embedded within a web page.
- **website:** A collection of interlinked web pages is called a website.
- **`while` statement:** The while statement is used to repeatedly execute a block of statements, called body of the while statement so long as the condition specified in the while statement is satisfied.
- **ZeroDivisionError:** This exception occurs when we try to perform a numeric division in which the denominator happens to be zero.

INDEX

A

Absolute and Relative Paths 128
Aggregate Functions 354
ALTER TABLE 343, 344, 346
Alternate Keys 317
Arithmetic Operators 10, 14
ARPANET 230, 231
Assignment Operators 12, 14
Attribute 312, 313
AVG() Function 357

B

Bandwidth 234
Baud Rate 233
BETWEEN Operator 350
Binary Files 118, 126
Bit Rate or Data Transfer Rate 233
break statement 40
Bridge 247
Built-in Exceptions 174
Built-in Functions 15
Built-in Modules 18
Built-in Names 104
Bus Topology 240

C

Calling a Function 24
Candidate Key 315, 316
capitalize() 45
Cardinality 314
CHAR(L) or CHARACTER(L) 334
CHECK Constraint 339
Circuit Switching 229, 232, 233
Cladding 237
clear() 59
close() 120, 123, 411
Closing Files 120
Coaxial cable 236, 237
comment 24, 27, 28
Complex 4
Computer Networks 228
Confidentiality, integrity, and availability 292
Constraint 312, 314
continue statement 40, 41
Control Statements 34
Cookies 296
copy() 59
Copying File Example 180
COUNT() Function 354
count(element) 55
CREATE DATABASE Statement 335
CREATE TABLE Statement 336
CSV File 118, 130, 131
cursor() 411
Cyber Laws 298
Cybercrimes 298
Cycle 233

D

Data Definition Language (DDL) 331
Data Rate 234
Data Transmission Modes 234
Data Types 2, 7, 18, 332
Database Administrator (DBA) 310
DATE 335
DBMS 39, 310, 311
DECIMAL(L, D) 333
DEFAULT Constraint 338
Default Parameters 29
Degree 313, 314
DELETE 332
DESCRIBE TABLE 340
Dictionaries 56
Dictionary (Mapping) 7
DISTINCT Clause 348
docstring 24, 28
Domain Constraint 318
Domain Names 274, 275
Drop Attributes 345
DROP TABLE 343
dump() 126, 127

E

Electronic Mail Protocols 268
endswith('other') 47
Entity Integrity Constraint 317
Entity Set 314
EOFError 127, 137, 176, 177
EQUI JOIN 385
Ethernet Card 246
Exceptions 135, 173, 174
execute() 411, 413

F

fetchall() 414
fetchmany(n) 414
fetchone() 414, 415
Fibre-Optic Cable 237
Fifth-Generation (5G) 272
File Access Mode 119
File Object Attributes 128
File Transfer Protocol (FTP) 267
FileNotFoundError 177
Files 118
Finally Clause 181
find('other') 47
Firewall 296
First-Generation (1G) 271
float (floating point numbers) 4
FLOAT(p) 334
flush() Function 123
for statement 36, 40
Foreign Key 317, 375
Formal Parameters 25, 27, 29
format() 413, 414
Fourth-Generation (4G) 272
Frequency 233
fromkeys(keys, value) 59
Full-Duplex Transmission 235
Function Call 17, 22, 27, 29
Function Calls as Function Argument 30
function header 23, 24

function's body 23
Functions on List 50
Functions 14, 15, 50

G

Gateway 249
get(myKey) 59
Global Names (Global Variables) 99
Global variable 99, 102, 103
GROUP BY Clause 357
Guided Medium 235

H

Hacking 295
Half-Duplex Transmission 235
Handling Multiple Exceptions 179
HAVING CLAUSE 358
help() Function 18, 28
help() 18, 28
Heterogeneous List 50
HTML and XML 272, 277
HTTPS 272, 297
Hub 241, 242, 247

I

Identifiers 10
Identity Operators 11, 14
if-elif-else 35
if-else 35
IN Operator 351
index(element) 55
index(subStr) 47
IndexError 176
India IT Act 298
Infrared Waves 239
Input Arguments 27, 29
input() 16
INSERT 340, 341, 377
int (integer) 3
INT or INTEGER 332
Integrated Development Learning Environment (IDLE) 1
internal reflection 237
Internet Relay Chat (IRC) 269
IOError 135, 136
IP 231, 266, 267
IPR Issues 299
IS Operator 352
isalnum() 47
isalpha() 46
isdigit() 46
islower() 46
isspace() 47
isupper() 46
items() 58

J

join(list) 48

K

Key 315
KeyError 56, 57
keys() 57
Keyword Arguments 30
Keywords 10

L

LAN 243
LIKE Operator 353
List Assignment 49
Lists (list) 5
Lists 48
Load() 126, 127
Local Names (Local Variables) 93
Local Variables 93, 94
Logical Errors 32
Logical Operators 11
lower() 45, 46
lst.append(elem) 51
lst.extend(lst) 51
lst.index(elem) 51
lst.insert(index, elem) 51
lstrip() 46

M

MAC 229
MAN 244
MAX() Function 356
Membership Operators 12
Microwave Link 238, 270
MIN() Function 356
Mobile Telecommunication Technologies 271
Modem 248, 249
Modular programming 15
module 15
multi-line string 24
Multiplication Operator * 43, 49
Mutable and Immutable Data Types 7
MySQL Server 410

N

NameError 175
NATURAL JOIN 387
Nested List 50
Nested Tuple 55
Network Interface Card (NIC) 245
Network Topologies 240
Network Types 243
None 6
NOT NULL Constraint 318
NULL Value 315, 317, 318
Numeric 3, 332

O

Objects in Python 93
open() 118, 119, 128
Opening and Closing Files 118
ORDER BY Clause 349
Order within a Table 315

P

Packet Switching 230, 231, 232, 233
PAN 243
partitition(delimit) 48
pass statement 39
PermissionError 177
pickle 126, 127
Pop 193, 194
pop(index) 52
pop(myKey) 58
POP3 269
popitem() 58
Postfix Notation 203, 205
PPP 265, 267
Precedence of Operators 14, 203
Primary Key 316, 317
print() 17
Push 194, 195
pymysql 410
Python Interface 410, 412

R

Radio Waves 238, 239
randint(a, b) 20
random() 19
randrange(start, stop=None, step=1) 20
range() function 24
read() 120
Reading and Writing Files 120
readline() 121
readlines() 121, 123
Referential Integrity Constraint 317
Referential Integrity 375
Relational Data Model 312
Relational Operators 11
Relational Schema 313
Remote Login (TELNET) 268
remove(element) 52
Repeater 246, 247
replace('old', 'new') 48
reverse() 52
rfind(subStr) 47
Ring Topology 240, 241
RJ-45 Connector 245, 246
Router 249
rstrip() 45

S

Satellite Link 239
Scope Rule 100
Second-Generation (2G) 271
Security Threats 292, 293
seek() 129
SELECT 332, 337, 342
Set 6
setdefault(key, [value]) 59
SHOW TABLES 340
Simple if 34
Simplex Transmission 234
Slicing 44, 48, 49
SMALLINT 332
SMTP 265, 269
sort() 52
sorted() 55, 59
sorted(t) 68
Spams 295
split(delimit) 48
Stack Overflow Condition 194
Stack Underflow Condition 194
Stacks 192, 195, 203
Star Topology 241, 252
startswith('other') 47
Statistics 20, 125
String Slices 43
Strings (str) 5
Strings 41
strip() 45, 124
Structured Query Language (SQL) 308, 331
SUM() Function 355
Switch 248
Switching Techniques 231
Syntax Errors 32, 173

T

TCP 265, 266, 267
TCP/IP 231, 265, 266, 267, 268
tell() 129
Text Files 118
Third-Generation (2G) 271
title() 47
Tree Topology 242
Trojan Horse 294, 300
try...except 186, 177, 178, 181
try-except 135
Tuple Operations 55
Tuple 314
tuple() 54, 55
Tuples 6, 53
Twisted pair cable 236
type() 3, 15
TypeError 175, 182

U

Understanding Data Files 117
Unguided media 235
UNIQUE Constraint 318
UPDATE Statement 342, 377
update() 58
UPDATE-SET 342
Updating Value of a Global Variable 103
Updating Value of a Local Variable 102
upper() 45
User-defined Functions 23
User-defined Modules 31

V

ValueError 51, 52, 175, 179
values() 57
VARCHAR(L) 334, 335
Virus 293
Voice over Internet Protocol (VoIP) 269

W

WAN 244
Web Browser 275, 276
Web Hosting 279
Web Page, Website, and URL 275
Web Scripting 279
while statement 36, 37
WiFi Card 246
with clause 122
Worms 294
write() 122, 123
writelines() 123
writerow() 130, 133
writerows() 130, 133

Z

ZeroDivisionError 175, 179, 182

From The Experts

Examination Hall Tips

01 Make sure that you keep your identity card, and other important documents in your bag the night before your exam.

02 If there are people around in the exam hall who are panicking, avoid them.

03 Too much last-minute cramming can send your brain into a spin.

04 Sit comfortably and spread out your arms and lean back on the chair. This releases a type of hormone which makes you more confident.

05 Fill up your Roll No, Name and other details carefully.

06 Take 5 minutes time to read the question paper carefully before you start attempting the questions.

07 Start answering the questions that you feel most confident about. There is no need to answer the questions in order as long as you label them correctly.

08 Check your handwriting.

09 If your brain freezes or you feel blank out, take 3–4 deep breaths. Simply skip that question and move on to something easier.

10 Keep last 15 minutes before the final bell to cross check your answers. Don't leave early.

Good Luck! ☺





Orange Education wishes you the Best of Luck for your exams!