For Secondary Classes

Answer Key

Modular Ver. 2.0

1. Introduction to Python

Exercise

A. 1. d 2. d 3. b 4. a

B. 1. T 2. F 3. F 4. T

C. 1. Prompt 2. Guido van Rossum 3. Variables 4. print() 5. line by line

D. 1. The features of Python are:

- **Easy to code:** It is very easy to write programs in Python as compared to other high-level programming languages.
- **Open-source language:** Python is a free and open-source programming language. An open-source language is one which can be easily improved and distributed by anyone.
- **Object-oriented:** Python has an object-oriented approach. This means that the programs are designed using objects and classes that interact with each other.

2. On execution, a Python code is immediately converted into an intermediate form. This is known as byte code.

3. variable_name = value

E. 1. Four variable naming conventions are:

- A variable name must start with a letter or underscore character.
- A variable name cannot start with a number.
- A variable name can only contain alphanumeric characters (all the letters of the alphabet and numbers) and underscores (_).
- Variable names are case-sensitive.

2. There are two components of Python IDLE window:

- **Menu Bar:** The Menu Bar of Python IDLE window is similar to the Menu Bar of other programs. It has various menus such as File, Edit, Shell, Debug, Options, Window and Help.

- **Prompt:** You will see a blinking cursor after the symbol (>>>) in the window. This is known as the Prompt. The Prompt allows the user to enter commands directly into Python and get an output instantly by pressing the Enter key.

3. The input( ) function takes the user's input while a program executes.

   On the other hand, the print() function prints or sends the output to the standard output device, which is usually a monitor.

F. Aarav should write his codes in Script Mode.

IN THE LAB

1. `print("Hello, How are you? I am learning Python language.")`
2. `print("Education is the most powerful weapon which you can use to change the world.")`

2. Data Types and Operators in Python

EXERCISE

A. 1. b 2. a 3. d 4. d 5. d

B. 1. T 2. F 3. T 4. T 5. T

C. 1. Precedence 2. assignment 3. string 4. operators 5. false

D.
1. It is an assignment expression where the value 8 is assigned to the variable a.
2. Variables are memory reference points where we store values which can be accessed or changed later.
3. A data type specifies the type of value a variable can contains.
4. Precedence of operators determines the order in which the operators are executed.
5. Relational operators are used to compare the value of the two operands and returns True or False accordingly.

E. 1. The relational operators are described in the following table:

Operator	Name	Description	Example (x=8 and y=6)	Output
==	Equal	It checks if the values of two operands are equal or not. If yes, then the condition becomes true.	x == y	FALSE

Operator	Name	Description	Example (x=8 and y=6)	Output
!=	Not equal	It checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	x != y	TRUE
>	Greater than	It checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	x > y	TRUE
<	Less than	It checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	x < y	FALSE
>=	Greater than or equal to	It checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	x >= y	TRUE
<=	Less than or equal to	It checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	x <= y	FALSE

2. Comments in Python can be used to explain parts of the code. It can also be used to hide the code as well. Comments enable us to understand the way a program works. In python, any statement starting with # symbol is known as a comment. Python supports two types of comments: Single Line Comment and Multi-line Comment.

3. (i) Arithmetic operators are used to perform arithmetic operations between two operands.

   (ii) Variables are memory reference points where we store values which can be accessed or changed later.

   (iii) Logical operators are used to evaluate and decide.

4. (i) AND returns true, if both operands are true.
   Example (x=2): (x < 5) and (x < 10). Output: TRUE.
   OR returns true, if one of the operands is true.
   Example (x=2): (x < 5) or (x < 2). Output: TRUE.

   (ii) Modulus Operator divides left hand operand by right hand operand and returns remainder.

   Example (x=7 and y=3): x % y. Output: 1

   Division Operator divides left hand operand by right hand operand.

   Example (x=7 and y=3): x / y. Output: 2.33

(iii) In case, a user wants to specify a single line comment, then comment must start with the symbol **#**.

Example:

Program:

```
# printing a string
print("Hello world")
```

Output:

```
Hello World
```

Python does not have a syntax for multiline comments. To add a multiple line comment, you could insert a **#** for each line.

Example:

Program:

```
# printing a string
#print five names
print("Hello World")
print("Rachna")
print("Sambhav")
print("Akshat")
print("Tushar")
```

Output:

```
Hello World
Rachna
Sambhav
Akshat
Tushar
```

F. 1. 2.0 2. 12 3. 4 4. 6

IN THE LAB

Do it yourself.

3. Conditional Statements in Python

EXERCISE

A. 1. c 2. a 3. d 4. b 5. d

B. 1. F 2. F 3. F 4. T 5. T

C. 1. if 2. true 3. false 4. else

D. 1. Decision making in Python is done by called conditional statements which decide the flow of program execution.

2. Syntax:

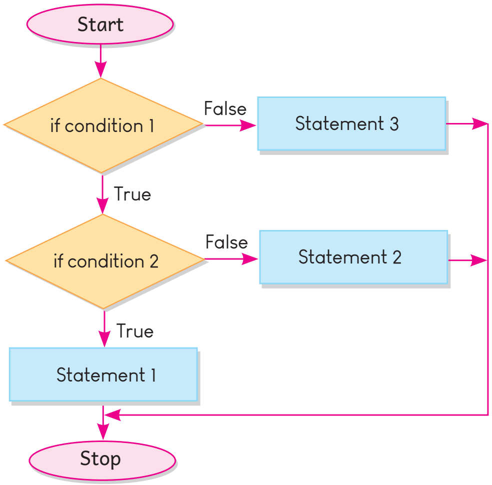
```
if (conditional expression 1):
     statement(s)
```

```
elif (conditional expression 2):
    statement(s)
elif (conditional expression 3):
    statement(s)
else:
    statement(s)
```

3. Syntax:

```
if (conditional expression):
    statement(s)
# if block ends here
```

E. 1.



2.

```
num = float(input("Enter the length in centimeter: "))
# 1 inch = 2.54 centimeters
inc = num / 2.54
print("Length in inch:", inc)
```

F. 1. Enter a First Number: 25

Enter a Second Number: 68

Second number is greater than first number

2. Positive number

3. Above ten

and also above 20!

4. b is not greater than a

G. Conditional Statements

In the Lab

Do it yourself.

Periodic Assessment-1

(Based on chapters 1 to 3)

A. Sum of 2 and 3 is 5

B.

```
i = 1
while i <= 25:
    if i % 2 == 0:
        print(i)
    i = i + 1
```

C. 1. It returns true, if both operands are true.

2. It checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.

4. Looping Statements in Python

Exercise

A.	1. b	2. a and b	3. a	4. c	
B.	1. T	2. F	3. T	4. T	5. T
C.	1. while	2. infinite	3. break, continue		

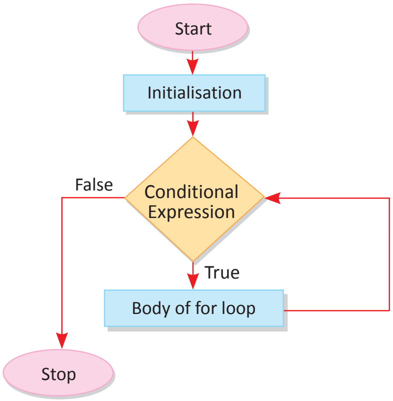
D. 1. Looping is the process of executing a set of instructions repeatedly until a specific condition is met. In Python, loops are created using the for and while statements.

2.

```
for <variable> in <iterator>:
    Statements
```

3. Sometimes, there is a situation when the control of the program needs to be transferred out of the loop body, even if all the values of the iterations of the loop have not been completed. For this purpose, jumping statements are used in Python.

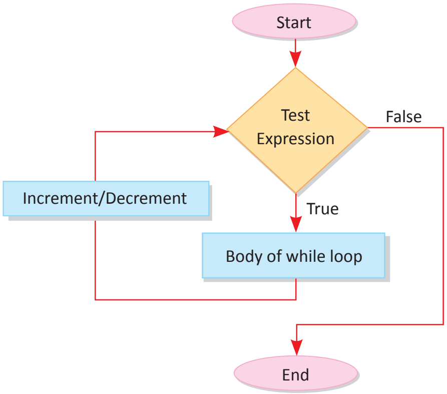
E. 1.



2. The while statement executes a set of statements repeatedly, until the logical expression evaluates to true. When the condition becomes false, the control comes out of the loop.

Syntax:

```
while (test expression):
      Statements
      Increment/Decrement expression
```



3. The continue statement is used inside loops. When a continue statement is encountered inside a loop, control of the program jumps to the beginning of the loop for next iteration, skipping the execution of rest of the statements of the loop for the current iteration.

The break is a keyword in Python which is used for bringing the program control out of the loop.

F. 1. 55

2. apple
banana
cherry

3. 2
4

4. 0
1
2

G. Swara can use the while loop to repeat a block of statements until a given condition becomes false.

IN THE LAB

Do it yourself.

5. Functions in Python

EXERCISE

A. 1. a 2. d 3. d 4. c 5. c

B. 1. T 2. T 3. T 4. T 5. T

C. 1. return 2. built-in 3. User-defined 4. def 5. command

D. 1. A function is a block of organised and reusable code used to perform a single or related action.

2. The features of functions are:

- A program is divided into small modules and each module performs some specific task. Each module can be called as per the requirement.
- We can call a function as many times as required. This saves the programmer the time and effort to rewrite the same code again. Therefore, it also reduces the length of the program.

3. Following are the advantages of functions:

- You can write Python program in logically independent sections.
- Functions provide better modularity for your application and a high degree of code reusing.
- As the program grows larger, functions make it more organized and manageable.

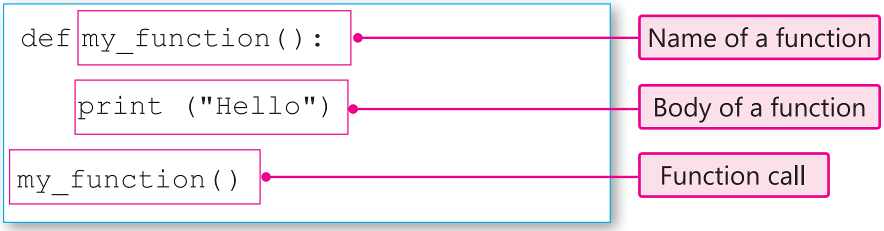
E. 1. A Python function consists of the following components:

- **Name of the function:** A function name should be unique and easy to correlate with the task it will perform. We can have functions of the same name with different parameters.
- **Arguments:** The input given to the functions are referred to as arguments. A function can or cannot have any arguments.
- **Statements:** The statements are the executable instructions that the function can perform.
- **Return Value:** A function may or may not return a value.

2. A function can be called anytime from other functions or from the command prompt after the definition. For calling a function, we type the function and pass the parameters.

For example:

To call a function.



```
def my_function():
    print ("Hello")
my_function()
```

3. Python functions can be categorised into built-in functions and user-defined functions. **Built-in functions** are the predefined functions of Python language. **User-defined functions** are created by the user according to the need of the program. Once the user defines a function, the user can call it in the same way as the built-in functions.

4. We can create a function in the following steps:
   - **Defining a Function:** We use the **def** keyword to begin the function definition.
   - **Naming a Function:** Provide a meaningful name to your function.
   - **Supply Parameters:** The parameters (separated by commas) are given in the parenthesis following the name of the function. These are basically the input values we pass to the function.
   - **Body of the function:** The body of the function contains Python statements that make our function perform the required task. Syntax of creating a function is:

```
def < name of the function > (list of parameters)
        <body>
```

F. 1. testing...

passing the value 4

the function returns 4

2. Enter number2

Raise to power3

2 raise to power 3 is 8

IN THE LAB

1.

```
def check_even_odd(num):
    if num % 2 == 0:
        print(num, "is even.")
    else:
        print(num, "is odd.")
number = int(input("Enter a number: "))
check_even_odd(number)
```

2.

```
def print_orange():
    print("orange")
print_orange()
```

Periodic Assessment-2

(Based on chapters 4 & 5)

A. 1.

```
Hello Touchpad
Hello Touchpad
Hello Touchpad
..........
```

Loop will run infinitely.

2.

```
Name:  Taarush
Age:  21
```

B.

```
sum = 0
i = 1
while i <= 5:
    sum += i
    i += 1
print("The sum of first five natural numbers is", sum)
```

C. 1. Parameters are the input values passed to a function when it is called.

2. The pre-defined functions of Python are called built-in function.

3. The loop which never terminates is called infinite loop.

4. The range() function is a built-in function which generates a sequence of numbers.

Test Sheet-1

(Based on chapters 1 to 5)

Section A

A.	1. d	2. a	3. d	4. d	5. d
	6. c	7. a and b	8. c	9. a	10. c
B.	1. F	2. T	3. T	4. T	5. T
C.	1. Variables	2. Precedence	3. false	4. infinite	5. def

Section B

A. 1. Features of Python are:

- **Easy to code:** It is very easy to write programs in Python as compared to other high-level programming languages.

- **Open-source language:** Python is a free and open-source programming language. An open-source language is one which can be easily improved and distributed by anyone.
- **Object-oriented:** Python has an object-oriented approach. This means that the programs are designed using objects and classes that interact with each other.

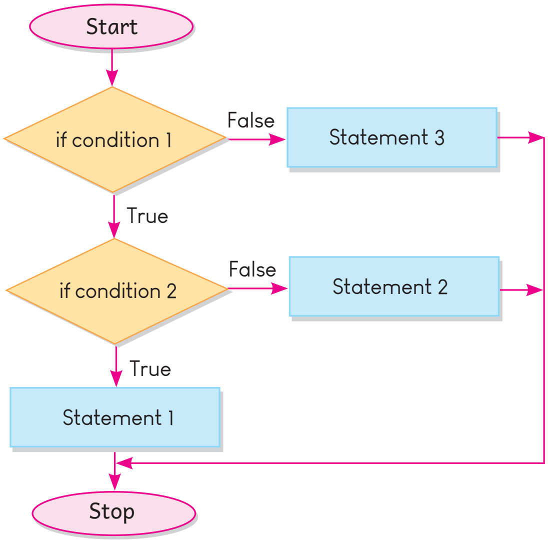
2. Variables are memory reference points where we store values which can be accessed or changed later.
3. Decision making in Python is done by called conditional statements which decide the flow of program execution.
4. Sometimes, there is a situation when the control of the program needs to be transferred out of the loop body, even if all the values of the iterations of the loop have not been completed. For this purpose, jumping statements are used in Python.
5. A function is a block of organised and reusable code used to perform a single or related action.

B. 1. Four variable naming conventions are:

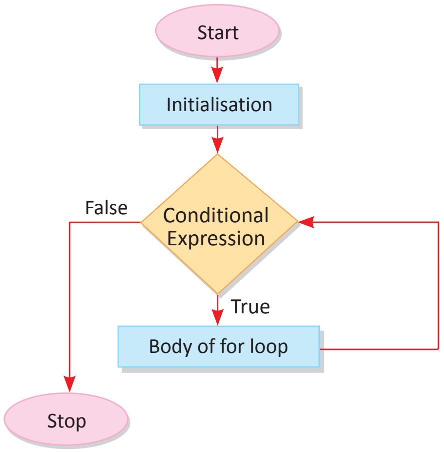
- A variable name must start with a letter or underscore character.
- A variable name cannot start with a number.
- A variable name can only contain alphanumeric characters (all the letters of the alphabet and numbers) and underscores (_).
- Variable names are case-sensitive.

2. Comments in Python can be used to explain parts of the code. It can also be used to hide the code as well. Comments enable us to understand the way a program works. In python, any statement starting with # symbol is known as a comment. Python supports two types of comments: Single Line Comment and Multi-line Comment.

3.

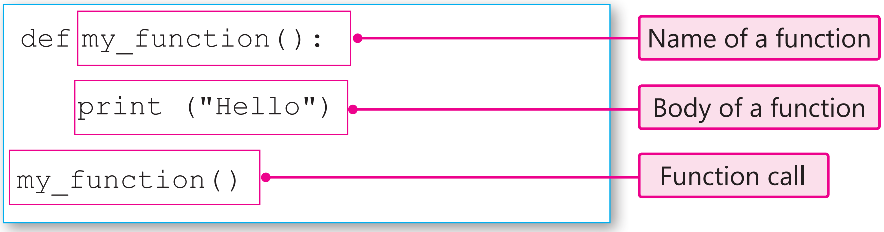


4.



5. A function can be called anytime from other functions or from the command prompt after the definition. For calling a function, we type the function and pass the parameters. For example:

   To call a function.



```
def my_function():
    print ("Hello")
my_function()
```

6. String Handling in Python

Exercise

A. 1. a 2. b 3. d 4. d 5. a

B. 1. F 2. T 3. T 4. F 5. T

C. 1. right 2. escape 3. concatenation 4. lowercase 5. capitalize()

D. 1. A sequence of characters which is enclosed or surrounded by single (' ') or double (" ") quotes is known as a string.

2. String concatenation operator(+) joins two or more strings into one, whereas the replication operator(*) is used to repeat the string for a given number of times.

3. Traversing means visiting each element and processing it as required by the program. We can access the characters of a string one at a time using indexing.

E. 1. There are two ways of indexing a list:

- **Index with positive integers:** Index from left starts with 0.
- **Index with negative integers:** Index from right starts with -1.

  Syntax for traversing a string:

  ```
  <name of the string> [index]
  ```

2. Two built-in functions to manipulate strings are:

- **len():** The len() function calculates and returns the length of a string supplied as an argument.

  Syntax of using len() function is:

  ```
   len(string_name)
  ```

- **lower():** The lower() function converts all uppercase letters to lowercase.

  Syntax of using lower() function is:

  ```
  string_name.lower()
  ```

3. An escape sequence is a sequence of characters which does not get displayed in output when used inside a character or a string. It is typically used to specify actions such as carriage returns and tab movements. The backslash (\) is a special character and is also known as the escape character in Python. It is used to represent white space characters, for example, '\t' for tab, '\n' for new line.

F. The original string is : Good Morning

The resultant string : GOOD MORNING

G. Taarush can convert his name to uppercase by using the upper() function.

IN THE LAB

1.
```
string = "PYTHON"
for char in string:
    print(char)
```

2.
```
string = "orange education"
uppercase_string = string.upper()
print(uppercase_string)
```

7. List in Python

Exercise

A. 1. b 2. a 3. b 4. b 5. a

B. 1. T 2. F 3. T 4. F 5. T

C. 1. sequence 2. item 3. indexing 4. mutable 5. +

D.

1. A list can be defined as a sequence of objects arranged in an organised form. In a list, each element or value is called an item.
2. In case if the user tries to access an element from a list beyond the defined range of the list, then it will give an IndexError.
3. The index of elements of a list starts from 0; which means if a list contains 10 elements then its index (it is always an integer number) is from 0 to 9.
4. reverse()
5. Negative indexing means the index of -1 refers to the last elements of the list, the index of -2 refers to the second last element.

E.

1. The index of elements of a list starts from 0; which means if a list contains 10 elements then its index (it is always an integer number) is from 0 to 9, whereas List slicing refers to a part of list. In python list slicing is done by using the Slicing operator(:).
2. The replication operator * is used to repeat a list a given number of times.
3. **remove():** Remove first element from the list.

   index(): Returns the index of first element of the list.

   count(): Returns the occurrence (number of times) of a particular element in the list.
4.
```
list = [13, 25, 41, 63, 82]
list.extend([12, 2, 34, 65])
print(list)  # Output: [13, 25, 41, 63, 82, 12, 2, 34, 65]
```
5. **max(list):** Returns the largest element from the given list.

   min(list): Returns the smallest element from the given list.

F.

1. [13, 50, 41, 45, 82]
2. [13, 25, 41, 63, 82, 19]
3. [13, 302, 25, 41, 63, 82]
4. 5

   82

   13

In the Lab

1.
```
list1 = [4, 9, 10, 28]
print(list1)
```

2.
```
nested_list = [[4, 9, 10, 28], [7, 11, 13, 48]]
print(nested_list)
```

3.
```
list3 = [4, 9, 10, 28]
print(list3[-1])  # Last element
print(list3[-2]) # Second last element
print(list3[-3])  # Third last element
print(list3[-4])  # First element
```

4.
```
list4 = [4, 9, 10, 28, 87, 98]
list4[1] = 45
list4[4] = 55
print(list4)
```

Periodic Assessment-3

(Based on chapters 6 & 7)

A.
```
def count_vowels(string):
    vowels = "aeiouAEIOU"
    count = 0
    for char in string:
        if char in vowels:
            count += 1
    return count
string = "Mango"
print("Number of vowels:", count_vowels(string))
```

B.
```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for num in numbers.copy():
    if num % 2 != 0:
        numbers.remove(num)
print(numbers)  # Output: [2, 4, 6, 8, 10]
```

C. 1. Escape sequences or characters are used to insert special characters that are invalid in Python. Quotes are special characters and to insert them in a string we must escape them with a character in front of them.

2. Traversing means accessing or visiting the elements of a list. Various ways to traverse a list are indexing, negative indexing and slicing.

3. Python provides several built-in functions to manipulate strings:

- **len():** Returns the length of the string.
- **lower():** Converts all characters in the string to lowercase.
- **upper():** Converts all characters in the string to uppercase.

8. Tuple in Python

Exercise

A.	1. d	2. a	3. c	4. c	5. a
B.	1. T	2. T	3. T	4. F	
C.	1. comma	2. positive	3. sorted()	4. max()	5. slicing

D. 1. A tuple is a sequence of values enclosed in parentheses and its indices start with 0.

2. The count(a) method returns number of times ‘a’ occurs in the tuple.

3. To create an empty tuple: `tup= tuple()`

To create a tuple with single element: `tup= (1,)`

E. 1. The index or subscript operator [] is used to access the elements of a tuple. The index value must be an integer as it will result into TypeError.

For example: If a tuple contains 5 elements, then the index value of tuple will varies from 0 to 4.

2.

List	Tuple
A list is a sequence of multiple values.	A tuple is a collection of objects.
List is enclosed in square brackets [].	Tuples are enclosed in parenthesis ( ).
Elements of the list can be changed.	Elements of the tuple cannot be changed.
List has the variable length.	Tuple has the fixed length.
List is mutable.	Tuple is immutable.

3. **len( ):** Returns the total length of the tuple.

max( ): Returns the largest element of the tuple.

min(): Returns the smallest element of the tuple.

sum(): Returns the sum of all the elements of the tuple.

F. 1. (3, 4, 6, 7)

2. tup1[0]: sst

tup2[1:5]: (2, 3, 4, 5)

3. (10, 20, 30, 10, 20, 30, 10, 20, 30)

4. Sanjay

Ajay

7020

400

In the Lab

Do it yourself.

9. Dictionary in Python

Exercise

A.	1. a	2. a	3. c	4. c
B.	1. F	2. T	3. T	4. T
C.	1. pop()	2. clear()	3. popitem()	4. del

D. 1. A dictionary in Python is another data type which contains a collection of values in the form of key value pairs.

2. To access the elements of a dictionary, you need the key defined in the key:value pairs. You can also use the get( ) method for fetching the value of a particular key of a dictionary.

Syntax to access a dictionary:

```
<dictionary-name>[<key>]
```

3. You can easily add or modify the elements of a dictionary. If the item is present in the dictionary, then you can change its value by providing new value to its key term.

E. 1. To create a dictionary in Python use the following syntax:

```
d = {key1: value1, key2 : value2, ... }
```

Example:

```
d2 = {'width': 8.5, 'height': 11}
d3 = {1: 'RED', 2: 'GREEN', 3: 'BLUE', }
```

2. The pop() method is used to remove an element whose key is given. On the other hand, clear() method is used to remove all the elements from a dictionary.

3. The get() method returns the value of the item with the given key.

F. 1. {'name': 'A', 'class': 'XII', 'year': 2022}

{'name': 'A', 'class': 'XII', 'year': 2022, 'stream': 'Science'}

2. {'Name': 'A', 'Rollno': 1}

{'Name': 'A', 'Rollno': 2, 'Marks': [15, 47, 54]}

IN THE LAB

Do it yourself.

10. App Development

EXERCISE

A. 1. a 2. a 3. a 4. a 5. c

B. 1. F 2. T 3. F 4. T 5. F

C. 1. hybrid 2. Mobile 3. Android 4. gaming 5. install

D. 1. An app is a software program primarily developed for hand-held smart devices such as mobile and tablet.

2. Web apps are actually web applications which give a user with experience similar to native apps. These apps are not deployed on the app store. Hence, you need an extra app called browser to access these apps on your mobile device.

3. Native apps are the type of Mobile apps. These are platform dependent which means that these apps are primarily developed for a specific platform.

4. Built-in Blocks, Component Blocks and Workspace.

E. 1. a. **Gaming Apps:** Today's most popular category of mobile apps is gaming apps which shared more than 24% area of the app store. The most commonly used gaming apps are Clash of Clans, Candy Crush Saga, and Angry Birds.

b. **Productivity Apps:** Productivity apps, also known as business apps used by businessmen to perform several complex tasks. The most commonly used productive apps are Google Calendar, Evernote and Dropbox.

c. **Entertainment Apps:** Entertainment apps are developed to entertain the people. The most commonly used entertainment apps are Netflix, Talking Tom and YouTube.

2. To change the display name of button, follow the given steps:

   Step 1: Click on the button in the View pane.

   Step 2: Type new name for button in the Text box.

3. Web apps are different from websites. The major difference is that a web app can be a small part of a website which provides a particular functionality. On the other hand, a website can contain many web apps.

4. Educational apps provide a platform for children to learn from anywhere and anytime. These apps use advance methodologies and new concepts to make the learning easier. The most commonly used educational apps are Khan Academy, Vedantu, and Grammar EN.

IN THE LAB

Do it yourself.

Periodic Assessment-4

(Based on chapters 8 to 10)

A. 1. App 2. Play Store 3. iOS 4. Web App
5. App Store 6. Hybrid App 7. Gaming Apps 8. Educational Apps
9. Social Networking Apps 10. Web App

B. 1. T 2. F 3. T 4. F 5. T

C. 1. (10, 20, 30, 40, 50, 60, 70)

2. (10, 20, 30, 10, 20, 30)
(60, 70, 60, 70, 60, 70)

Test Sheet-2

(Based on chapters 6 to 10)

Section A

A. 1. b 2. a 3. b 4. a 5. a
6. c 7. a 8. c 9. a 10. c

B. 1. F 2. F 3. T 4. T 5. F 6. F
7. F 8. T

C. 1. escape 2. item 3. positive 4. clear() 5. Android 6. capitalize()
7. + 8. max()

Section B

A. 1. A sequence of characters which is enclosed or surrounded by single (' ') or double (" ") quotes is known as a string.

2. In case if the user tries to access an element from a list beyond the defined range of the list, then it will give an IndexError.

3. A tuple is a sequence of values enclosed in parentheses and its indices start with 0.

4. A dictionary in Python is another data type which contains a collection of values in the form of key value pairs.

5. Built-in Blocks, Component Blocks and Workspace.

6. String concatenation operator(+) joins two or more strings into one, whereas the replication operator(*) is used to repeat the string for a given number of times.

7. reverse()

B. 1. There are two ways of indexing a list:

- **Index with positive integers:** Index from left starts with 0.
- **Index with negative integers:** Index from right starts with -1.

Syntax for traversing a string:

```
<name of the string> [index]
```

2. **remove( ):** Remove first element from the list.

index( ): Returns the index of first element of the list.

count( ): Returns the occurrence (number of times) of a particular element in the list.

3. The index or subscript operator [] is used to access the elements of a tuple. The index value must be an integer as it will result into TypeError.

For example: If a tuple contains 5 elements, then the index value of tuple will varies from 0 to 4.

4. The pop() method is used to remove an element whose key is given. On the other hand, clear() method is used to remove all the elements from a dictionary.

5. Educational apps provide a platform for children to learn from anywhere and anytime. These apps use advance methodologies and new concepts to make the learning easier. The most commonly used educational apps are Khan Academy, Vedantu, and Grammar EN.

6. Repeating lists using the * operator is useful when you want to duplicate the contents of a list multiple times.

7. To access values in a tuple, use the square brackets along with the index number for reference.

Example:

tup = (1, 2, 3)

print(tup[0]) # Output: 1