TOUCHPAD

Modular Python (Ver. 2.0)

Teacher's Manual

Extended Support for Teachers

ORANGE
www.orangeeducation.in

Teacher's Time Table

Periods / Days	0	I	II	III	IV	BREAK	V	VI	VII	VIII
Monday										
Tuesday										
Wednesday										
Thursday										
Friday										
Saturday										

Development Milestones in a Child

Development milestones are a set of functional skills or age-specific tasks that most children can do at a certain age. These milestones help the teacher identify and understand how children differ in different age groups.

Age
5 - 8 Years

Physical	• First permanent tooth erupts • Shows mature throwing and catching patterns • Writing is now smaller and more readable • Drawings are now more detailed, organised and have a sense of depth
Cognitive	• Attention continues to improve, becomes more selective and adaptable • Recall, scripted memory, and auto-biographical memory improves • Counts on and counts down, engaging in simple addition and subtraction • Thoughts are now more logical
Language	• Vocabulary reaches about 10,000 words • Vocabulary increases rapidly throughout middle childhood
Emotional/ Social	• Ability to predict and interpret emotional reactions of others enhances • Relies more on language to express empathy • Self-conscious emotions of pride and guilt are governed by personal responsibility • Attends to facial and situational cues in interpreting another's feelings • Peer interaction is now more prosocial, and physical aggression declines

"If you cannot do great things, do small things in a great way."

Age 9 - 11 Years

Physical	• Motor skills develop resulting in enhanced reflexes
Cognitive	• Applies several memory strategies at once • Cognitive self-regulation is now improved
Language	• Ability to use complex grammatical constructions enhances • Conversational strategies are now more refined
Emotional/ Social	• Self-esteem tends to rise • Peer groups emerge

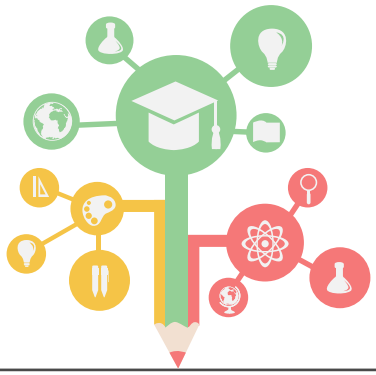
Age 11 - 20 Years

Physical	• If a girl, reaches peak of growth spurt • If a girl, motor performance gradually increases and then levels off • If a boy, reaches peak and then completes growth spurt • If a boy, motor performance increases dramatically
Cognitive	• Is now more self-conscious and self-focused • Becomes a better everyday planner and decision maker
Emotional/ Social	• May show increased gender stereotyping of attitudes and behaviour • May have a conventional moral orientation

Managing the children's learning needs according to their developmental milestones is the key to a successful teaching-learning transaction in the classroom.

> "Family is the most important thing in the world."

Teaching Pedagogies



Pedagogy is often described as the approach to teaching. It is the study of teaching methods including the aims of education and the ways in which such goals can be achieved.

Lesson Plans

A lesson plan is the instructor's road map which specifies what students need to learn and how it can be done effectively during the class time. A lesson plan helps teachers in the classroom by providing a detailed outline to follow in each class.

A lesson plan addresses and integrates three key components:

- Learning objectives
- Learning activities
- Assessment to check the student's understanding

A lesson plan provides an outline of the teaching goals:

Before the class

1. Identify the learning objectives.
2. Plan the lesson in an engaging and meaningful manner.
3. Plan to assess student's understanding.
4. Plan for a lesson closure.

During the class

Present the lesson plan.

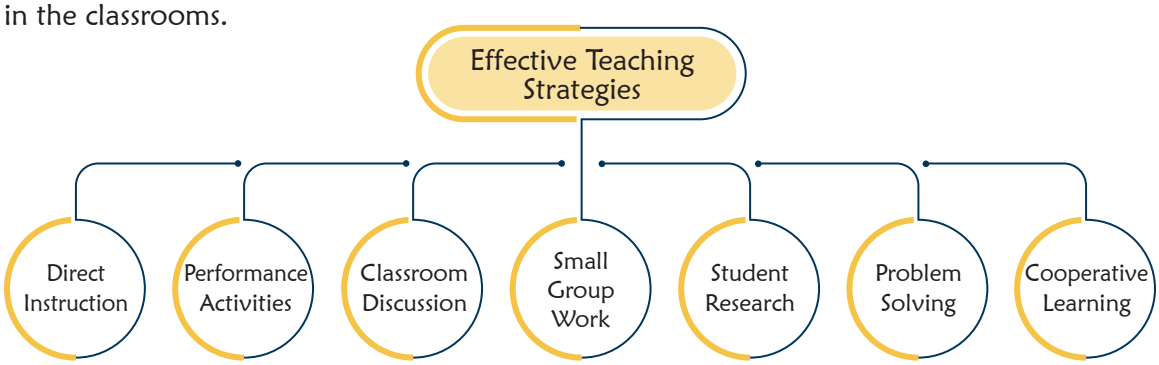
After the class

Reflect on what worked well and why. If needed, revise the lesson plan.

> "Knowing yourself is the beginning of all wisdom."

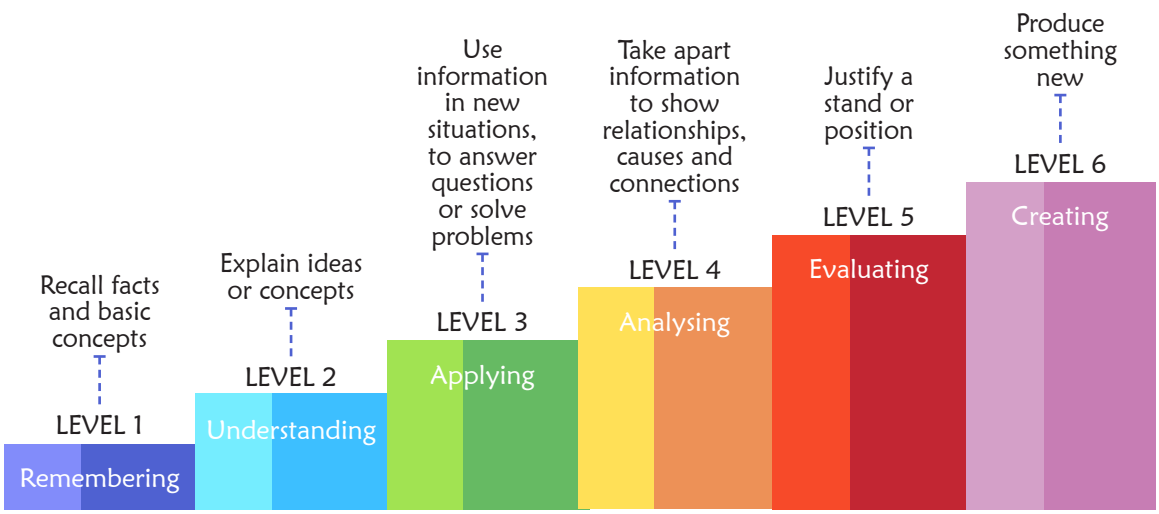
Teaching Strategies

Numerous strategies have evolved over the years to facilitate the teaching-learning process in the classrooms.



Bloom's Taxonomy

Bloom's Taxonomy was created by Dr Benjamin Bloom and several of his colleagues, to promote higher forms of thinking in education instead of rote learning. There are three domains of learning: cognitive (mental), affective (emotional), and psychomotor (physical). However, when we refer to Bloom's Taxonomy we speak of the cognitive domain. Bloom's Taxonomy is a list of cognitive skills that is used by teachers to determine the level of thinking their students have achieved. As a teacher, one should attempt to move students up the taxonomy as they progress in their knowledge.



Teachers should focus on helping students to remember information before expecting them to understand it, helping them understand it before expecting them to apply it to a new situation, and so on.

> "If you have no confidence in self, you are twice defeated in the race of life."

CLASS 1

Lesson Plan

1 Introduction to Python

Teaching Objectives

Students will learn about

- Understand the basic concepts and features of the Python programming language.
- Learn how to install Python and use its IDLE interface.
- Differentiate between Interactive Mode and Script Mode in Python.
- Apply input() and print() functions to write basic Python programs.
- Understand variable declaration, assignment, and naming conventions in Python.

Number of Periods	
Theory	Practical
3	2

Teaching Plan

Introduction

Engagement Strategy: Think-Pair-Share

Question: "Have you ever interacted with a chatbot or calculator on your phone? What do you think powers them?" Let students reflect, pair up to discuss, and share with the class.

Lesson Delivery (Explanation & Demonstration)

1. **Introduction to Python**
   - Python is a high-level, open-source, object-oriented, and interpreted language.
   - Discuss its applications in games, websites, GUI programs, etc.
   - **Strategy: Fussing with Definitions:** Students write definitions for key terms like 'interpreted', 'object-oriented', 'open-source' and share with peers.
2. **Installing Python**
   - Steps to install Python from the official site and setting up environment.
   - **Activity: Guided Practice:** Perform installation on classroom systems together with students.
3. **Programming in Python**
   - Difference between Interactive Mode and Script Mode.
   - **Strategy: Peer Editing:** Students write simple programs in both modes and review each other's code.

4. **Using input() and print() Functions**
   - Explain syntax and demonstrate user input and output.
   - **Strategy: Application Cards:** Students create small programs using input() and print() in real-life scenarios.
5. **Variables and Assignments**
   - Variable rules, types of assignments, and naming conventions.
   - **Strategy: Round Robin:** Students list valid variable names and correct the invalid ones in small groups.

Extension (Further Exploration)

Discussion Questions:

- Why is Python considered easier than other programming languages?
- How do input() and print() functions simplify user interaction?
- What are the consequences of not following variable naming rules?

Creative Task:

- Design a Python program that asks for the user's name, age, and class, then prints a formatted introduction message.

Evaluation (Assessments & Review)

- MCQs on Python basics, IDLE, input/output, and variable rules.
- Short-answer questions from the exercise section of the textbook.
- **Lab Activity:** Create and run a Python program using both Interactive and Script Mode.
- **Project:** Students write and demonstrate a personal profile generator using input() and print().

Suggested Activity

Project: Python Introduction Poster

- Students design a visual infographic/poster highlighting Python features, modes, and basic syntax using digital tools or chart paper.

2 Data Types and Operators in Python

Teaching Objectives

Students will learn about

- Identify and understand various data types in Python including numbers, sequences, sets, dictionaries, and booleans.
- Differentiate between single-line and multi-line comments in Python.
- Understand and apply different types of operators: arithmetic, assignment, logical, and relational.
- Apply the concept of operator precedence in evaluating expressions.

- Write basic programs using operators and data types for real-world problems.

Number of Periods	
Theory	Practical
3	2

Teaching Plan

Introduction (Engagement)

Strategy Used: Brainstorming and Think-Pair-Share

- What do you think 'data type' means in programming?
- What types of values can a variable hold in Python?
- Have you ever seen or used operators like +, -, * in other contexts?

Lesson Delivery (Explanation & Demonstration)

Explain and demonstrate the following Python data types with examples:

- Numbers: Integer, Float, Complex
- Sequence: String, List, Tuple
- Set
- Dictionary
- Boolean

Discuss the use and types of comments:

- Single Line (starts with #)
- Multi-line (''' or """ triple quotes)

Discuss different operator types with examples:

- Arithmetic Operators
- Assignment Operators
- Logical Operators
- Relational Operators

Introduce the concept of operator precedence and explain its hierarchy.

Activity

Strategy Used: Peer Editing and QAXP

- Students will write a short program using arithmetic and relational operators.
- Swap codes with peers to check logical correctness and use of comments.
- **Group into QAXP roles:** One asks a question on precedence or operators, others respond collaboratively.

Extension (Further Exploration)

Strategy Used: Application Cards

- Create a Python program that calculates the surface area and volume of a cylinder.
- Write a code snippet that uses all types of operators discussed in the lesson.
- Design a quiz application using Boolean logic to check answers.

Evaluation (Assessment & Review)

Strategy Used: One-Minute Paper and Student Generated Test Questions

- **Quiz:** Multiple-choice questions based on operator precedence and data types.
- **One-minute paper:** What did you find most useful in today's lesson?
- Ask students to generate two test questions on today's topic and answer them in pairs.

Suggested Activity

Ask students to create a simple calculator program using different operators (arithmetic, relational, logical) and data types (integer, float, string). They can then swap programs with a partner to review the use of operators and data types.

3 Conditional Statements in Python

Teaching Objectives

Students will learn about

- Understand the concept of conditional statements in Python.
- Learn and apply if, if...else, nested if, and if...elif...else constructs.
- Implement decision making through simple Python programs.
- Use flowcharts to represent conditional logic.
- Develop and evaluate logic using practical applications of decision statements.

Number of Periods	
Theory	Practical
3	2

Teaching Plan

Introduction

Engagement Strategy: Think-Pair-Share

Pose the question: "How do you decide what to do on a rainy day?" Let students discuss decision-making logic in daily life and relate it to programming.

Lesson Delivery (Explanation & Demonstration)

1. **Introduction to Conditional Statements**
   - Explain the concept of decision making in programming.

- Introduce types of conditional statements: if, if...else, nested if, if...elif...else.
- **Strategy: Fussing with Definitions:** Students redefine terms like 'conditional', 'statement', 'block', and share with groups.

2. **The if Statement**
   - Syntax and logic flow with example to check positive number.
   - Hands-on: Students code a simple 'positive number check' using input().
3. **if...else Statement**
   - Syntax explanation and program to compare two numbers.
   - **Strategy: Application Cards**: Students write real-life if...else scenarios and convert them into Python code.
4. **Nested if Statement**
   - Explain multiple-level decision making.
   - Demonstrate selection logic for college admission using nested if.
   - **Strategy: Peer Editing:** Students write nested conditions and exchange for feedback.
5. **if...elif...else Ladder**
   - Show multi-condition testing structure.
   - Example: Checking number digit range.
   - **Strategy: Round Robin:** Groups contribute one possible range and its condition.
6. **Additional Examples**
   - **Programs:** Leap year check, score classification, lucky number detection.
   - **Strategy: Three-Step Interview:** Pairs create and discuss condition logic, then present partner's idea.

Extension (Further Exploration)

Discussion Questions:

- Why is indentation important in Python conditional statements?
- What are the key differences between nested if and if...elif...else?
- How can conditional logic be used to solve real-world problems?

Creative Task:

- Create a Python program to display a custom message based on the user's age and marks.

Evaluation (Assessments & Review)

- MCQs and True/False based on syntax and logic.
- **Short answers:** syntax of if, elif and flowchart drawing.
- **Lab:** Write Python programs to check leap year, grade allocation, and digit counting.
- **Project:** Create a student admission checker program with conditions for eligibility.

Suggested Activity

Project: Smart Decision System

- Students design a Python program that mimics a decision system (e.g., travel planner, simple chatbot, grading system) using all forms of conditional statements.

4 Looping Statements in Python

Teaching Objectives

Students will learn about

- Things to Do in a Computer Lab
- Understand the concept and purpose of looping in Python.
- Explain and apply 'for' and 'while' loops in Python programs.
- Use the range() function to control iterations in for loops.
- Demonstrate the use of jump statements (break, continue) within loops.
- Identify infinite loops and explain how to manage them.
- Develop Python programs using loop constructs for real-world problems.

Number of Periods	
Theory	Practical
3	2

Teaching Plan

Introduction (Engagement)

Strategy Used: Think-Pair-Share

- Have you ever repeated a task multiple times like drawing patterns or printing a list?
- What if we need to repeat a task 100 times—should we write the same line of code 100 times?
- Why do you think programming needs repetition control like loops?

Lesson Delivery (Explanation & Demonstration)

Explain the two primary types of loops in Python:

- 'for' loop – used for fixed iterations, often with range()
- 'while' loop – continues until a condition becomes false

Introduce the concept and syntax of range() in for loops with parameters (start, stop, step).

Demonstrate how break and continue statements influence loop control.

Activity

Strategy Used: Send-a-Problem and Peer Editing

- Form small groups. Each group writes a small program using loops (e.g., pattern printing, summation).
- Use 'Send-a-Problem' where groups pass their code to another group for reviewing and improving.
- Edit each other's programs to improve logic and correct loop structure.

Extension (Further Exploration)

Strategy Used: Application Cards

- Write a Python program to check if a number is an Armstrong number using loops.
- Generate a sequence like 8, 11, 14,... using a while loop.
- Create a loop that prints the first 20 odd numbers in reverse order.

Evaluation Evaluation (Assessment & Review)

Strategy Used: One Minute Paper and Student Generated Test Questions

- Quiz on loop types, syntax, and control flow statements like break and continue.
- Ask students to write one question on loop logic and answer it.
- **One-minute paper:** What did you find easiest or hardest about loops today?

Suggested Activity

Groups of students work together to create a program that prints patterns (e.g., triangles, diamonds) using loops. They should then compare their solutions and discuss the different approaches to loop control.

5 Functions in Python

Teaching Objectives

Students will learn about

- Understand the concept and structure of a function in Python.
- Differentiate between built-in and user-defined functions.
- Identify the components and advantages of functions.
- Learn the steps to create and call functions using parameters and return values.
- Apply different types of user-defined functions in practical programs.

Number of Periods	
Theory	Practical
3	2

Teaching Plan

Introduction

Engagement Strategy: Think-Pair-Share

Pose the question: "How would you solve a problem repeatedly without writing the same code again and again?" Students pair up, share responses, and relate to functions.

Lesson Delivery (Explanation & Demonstration)

1. **Introduction to Functions**
   - **Definition:** Block of reusable code that performs a specific task.
   - **Importance:** Modularity, reusability, and code efficiency.

- **Strategy: Fussing with Definitions:** Students rephrase the term 'function' and explain how it helps in writing better programs.

2. **Features and Components of Functions**
   - **Features:** Modularity, reusability, easier debugging, and memory efficiency.
   - **Components:** Name, arguments, statements (body), and return value.
   - **Strategy: Round Robin:** Groups list the importance of each function component.
3. **Types of Functions**
   - **Built-in Functions:** Example - print(), input(), range().
   - **User-defined Functions:** Type 1 (no parameter, no return), Type 2 (with parameter, no return), Type 3 (with parameter, with return).
   - **Activity: Application Cards:** Students identify and write example use-cases for each function type.
4. **Creating a Function**
   - **Steps:** Use def keyword, provide name, parameters, and define body.
   - Syntax demonstration with 'Hello' print function.
   - **Strategy: Peer Editing:** Students write their own function and peer-review each other's function definition for errors.
5. **Calling a Function**
   - Call function after definition, pass parameters as needed.
   - **Examples:** Sum and product programs.
   - **Strategy: Three-Step Interview:** Students explain how to call functions and present their partner's explanation to class.

Extension (Further Exploration)

Discussion Questions:

- How are user-defined functions better than writing repeated code blocks?
- What are the benefits of functions in large programs?
- What's the difference between parameters and return values?

Creative Task:

- Write a Python program that takes three numbers and returns their average using a user-defined function

Evaluation (Assessments & Review)

- MCQs and Fill-in-the-blanks from textbook exercises.
- Short answer questions on syntax and definition.
- **Lab Activity:** Implement all three types of user-defined functions.
- **Project:** Build a simple calculator using user-defined functions.

Suggested Activity

Project: Function-Based Utility App

- Students create a Python program with functions to calculate area of square, check even/odd, and greet user by name.

6 String Handling in Python

Teaching Objectives

Students will learn about

- Define strings and understand their characteristics in Python.
- Differentiate between single-line, multiline, and empty strings.
- Use escape sequences within strings.
- Apply various string operations such as concatenation, replication, and slicing.
- Utilise built-in string functions for string manipulation.
- Traverse strings using indexing and loop constructs.
- Distinguish between identity, membership, and comparison operators with strings.

Number of Periods	
Theory	Practical
3	2

Teaching Plan

Introduction (Engagement)

Strategy Used: Brainstorming and Think-Pair-Share

- What do you understand by the term 'string' in real life?
- Can you name things that are made of a sequence of characters?
- Have you ever worked with text in any programming or typing tool?

Lesson Delivery (Explanation & Demonstration)

Explain the following concepts with examples:

- Creating strings using single and double quotes.
- Multiline and empty strings.
- **Escape sequences:** newline (\n), tab (\t), and quotes (\').
- String indexing using positive and negative values.
- **String operators:** + (concatenation), * (replication), in/not in, is/is not, ==, !=, etc.
- String slicing and traversal.
- **Built-in functions:** len(), lower(), upper(), capitalize(), count(), replace(), strip(), find(), isdigit(), isalpha(), isalnum()

Activity

Strategy Used: Peer Editing and QAXP

- Ask students to create a short biography using concatenation and multiline strings.

- Form QAXP groups where one asks, another answers, a third adds, and the fourth summarises based on a string program.
- Exchange string manipulation tasks with a peer and review for correct output and usage.

Extension (Further Exploration)

Strategy Used: Application Cards

- Write a program to count how many times a word appears in a paragraph.
- Check whether an input string is a palindrome using slicing.
- Create a Python program to convert lowercase input to uppercase using string functions.

Evaluation (Assessment & Review)

Strategy Used: One-Minute Paper and Student Generated Test Questions

- Conduct a quiz using MCQs and True/False based on string operators and functions.
- Ask students to generate two test questions on string slicing or built-in functions.
- **One-minute paper:** What is your favourite string function and why?

Suggested Activity

Group Task: Each group creates a Python program that performs a text analysis on a paragraph (e.g., word count, frequency of specific words, checking for palindromes). They will then present their results to the class and discuss the string methods they used.

7 List in Python

Teaching Objectives on it.

Students will learn about

- Understand the concept and structure of lists in Python.
- Differentiate between homogeneous and heterogeneous lists, nested lists, and empty lists.
- Perform list operations: indexing, slicing, traversing, and changing list elements.
- Use built-in list methods and Python functions like len(), max(), min().
- Apply practical list manipulation through coding exercises and real-life scenarios.

Number of Periods	
Theory	Practical
3	2

Teaching Plan

Introduction

Engagement Strategy: Think-Pair-Share

Ask students: "Can you think of a collection of items you use daily (like a shopping list or favourites list)? How would you store them using Python?"

Lesson Delivery (Explanation & Demonstration)

1. **Creating a List**
   - Define list as an ordered, mutable collection of elements enclosed in square brackets.
   - Show examples of homogeneous, heterogeneous, nested, and empty lists.
   - **Strategy: Round Robin:** Students list different types of items that can be part of a list.
2. **Traversing a List**
   - Explain indexing, negative indexing, and nested list access.
   - Examples to access and print values by index.
   - **Activity: Application Cards:** Write down index and values of given list items.
3. **List Operations**
   - Operators: + (join), * (repeat), : (slicing), == (comparison).
   - Hands-on practice for joining and slicing lists.
   - **Strategy: Fussing with Definitions:** Students define 'slicing' and 'replication' in their own words and share examples.
4. **Built-in List Methods**
   - Discuss append(), extend(), insert(), remove(), index(), count(), pop(), copy(), clear(), sort(), reverse().
   - **Strategy: Peer Editing:** Students use at least 5 methods and review each other's output.
5. **Python Functions on Lists**
   - Functions: len(), max(), min().
   - **Activity: Three-Step Interview:** Pairs explore function effects and explain each to the class.
6. **Slicing and Modifying Lists**
   - Demonstrate how to slice and change elements using index.
   - Add one or multiple items using append() and extend().

Extension (Further Exploration)

Discussion Questions:

- What's the advantage of using lists in Python over individual variables?
- How can slicing simplify data access in lists?
- When would you use a nested list instead of a simple list?

Creative Task:

- Create a nested list with your weekly timetable and access specific day/time slots using index operators.

Evaluation (Assessments & Review)

- MCQs and True/False based on list syntax, functions, and methods.
- Short answer questions from textbook exercises.

- **Lab Activity:** Create and manipulate lists using slicing, pop(), append(), and sort().
- **Project:** Create a grocery list manager in Python using various list operations.

Suggested Activity

Project: Student List Manager

- Students build a small program to maintain a class list using insert(), remove(), and sort(). It should allow adding new students and displaying names alphabetically.

8 Tuple in Python

Teaching Objectives on it.

Students will learn about

- Understand what a tuple is and how it differs from a list.
- Learn to create tuples with different types of elements including nested structures.
- Access tuple elements using indexing, slicing, and unpacking.
- Apply tuple operations like concatenation, repetition, and slicing.
- Use built-in methods (count, index) and functions (len, max, min, sorted) on tuples.
- Demonstrate how tuples maintain immutability and how list elements inside them can be changed.
- Understand how to delete tuples and differentiate between tuples and lists..

Number of Periods	
Theory	Practical
3	2

Teaching Plan

Introduction (Engagement)

Strategy Used: Think-Pair-Share

- Have you ever stored multiple items in one variable in Python?
- Do you think there is a way to store data that doesn't change after it's created?
- What are some use-cases where immutability might be helpful?

Lesson Delivery (Explanation & Demonstration)

Explain the following tuple concepts with live coding examples:

- Creating tuples with homogeneous, heterogeneous, and nested elements.
- Accessing elements using indexing and negative indexing.
- Unpacking tuples to assign values to multiple variables.
- Tuple slicing to extract ranges of elements.
- Using built-in methods: count(), index().
- Using built-in functions: len(), max(), min(), sum(), sorted(), enumerate(), all().
- Tuple operations: concatenation (+), replication (*), slicing.

- Changing list elements inside a tuple and deleting an entire tuple.

Activity

Strategy Used: Send-a-Problem and Peer Editing

- Students create tuples with various types of data including a nested tuple.
- Exchange tuples with peers and perform indexing, slicing, and methods on each other's tuples.
- Identify which elements in a tuple can be changed and which cannot, and correct errors in sample code.

Extension (Further Exploration)

Strategy Used: Application Cards

- Write a program that uses max() function to get the maximum value in a tuple.
- Write a program to convert a tuple into a list, modify it, and convert it back to tuple.
- Use slicing and conditions to display tuple values except specific characters like 'a' and 'e'.

Evaluation (Assessments & Review)

- Strategy Used: One-Minute Paper and Student Generated Test Questions
- Quiz on tuple creation, functions, and indexing.
- One-minute paper: What makes a tuple different from a list?
- Ask students to write and answer one question related to tuple slicing or unpacking

Suggested Activity

Students create a Python program that demonstrates tuple slicing, unpacking, and element modification within nested lists. They will then share their results with a partner to explain the behavior of tuples versus lists.

9 Dictionary in Python

Teaching Objectives on it.

Students will learn about

- Understand the concept of a dictionary as a collection of key-value pairs.
- Learn how to create and access dictionaries using keys and the get() method.
- Use built-in dictionary methods such as pop(), popitem(), clear(), and update().
- Perform operations on dictionaries using built-in functions like len(), sorted(), all(), and any().
- Update, remove, or delete dictionary elements effectively.
- Convert dictionaries to list of tuples and use them in loops.
- Differentiate between dictionary and other data types like lists and tuples.

Number of Periods	
Theory	Practical
3	2

Teaching Plan

Introduction (Engagement)

Strategy Used: Think-Pair-Share

- What comes to your mind when you hear the word 'dictionary'?
- How would you store multiple related data like student name, roll number, and marks?
- Have you used maps or tables where each entry has a name and value?

Lesson Delivery (Explanation & Demonstration)

Explain the following tuple concepts with live coding:

- Creating dictionaries using {}, dict() constructor, and from tuples.
- Accessing values using keys and get() method.
- Dictionary methods: pop(), popitem(), clear(), update(), items(), keys(), copy().
- Built-in functions: len(), sorted(), any(), all().
- Updating and deleting dictionary items.
- Traversing dictionary using for loops and converting to list of tuples.
- Difference between mutable and immutable structures like list, tuple, and dictionary

Activity

Strategy Used: Send-a-Problem and Peer Editing

- Students will create a dictionary with at least 5 key-value pairs.
- Use Send-a-Problem: each student passes their dictionary to another for modification (update or delete).
- Review each other's use of get(), pop(), and update() for correctness.

Extension (Further Exploration)

Strategy Used: Application Cards

- Write a program to create a dictionary with 15 keys (0 to 14) using a loop.
- Traverse a dictionary using a for loop and print each key and its value.
- Build a small dictionary-based menu system to simulate a student record app.

Evaluation (Assessments & Review)

Strategy Used: One-Minute Paper and Student Generated Test Questions

- Conduct a quiz on dictionary creation, access methods, and difference between methods like pop() and clear().
- Ask students to write one question based on dictionary traversal or manipulation.
- **One-minute paper:** Why are dictionaries useful and how are they different from lists?

Suggested Activity

Students work in pairs to create a simple address book application using a dictionary to store names and contact information. They will implement functions to add, search, and delete contacts.

10 App Development

Teaching Objectives on it.

Students will learn about

- Understand the concept and purpose of apps.
- Differentiate between desktop, mobile, and web apps.
- Explain the major types of mobile apps: Native, Web, and Hybrid.
- Identify popular categories of mobile apps and their uses.
- Learn how to download, install, and develop apps using App Inventor.

Number of Periods	
Theory	Practical
2	2

Teaching Plan

Introduction

Engagement Strategy: Think-Pair-Share

Ask: "What apps do you use most often on your mobile phone? What makes them useful?" Let students discuss and share their favourites.

Lesson Delivery (Explanation & Demonstration)

1. **Introduction to Apps**
   - Define an app as software developed for smart devices.
   - Discuss the difference between desktop, mobile, and web apps.
   - **Strategy: Round Robin:** Students list different apps they've used and classify them into types.
2. **Operating Systems for Mobile Apps**
   - Discuss Android and iOS platforms and their significance.
   - Show interfaces and features of both platforms.
3. **Types of Mobile Apps**
   - Native Apps – platform specific, downloaded from app stores.
   - Web Apps – run through browser, need internet connection.
   - Hybrid Apps – combine features of native and web apps.
   - **Strategy: Application Cards:** Students write use cases for each app type.
4. **Categories of Apps**
   - Discuss Gaming, Productivity, Entertainment, Utility, Educational, Social Networking, Communication, E-commerce apps.

- **Strategy: Three-Step Interview:** Students explain features of any one category app to a partner and then share with class.

5. **Downloading and Installing Apps**
   - Explain how to install an app from Google Play Store using step-by-step demonstration.
   - **Activity: Guided Practice:** Simulate the process using images/screenshots.
6. **Developing an App Using App Inventor**
   - Introduce App Inventor as a drag-and-drop tool similar to Scratch.
   - Explain Design View and Block Editor View.
   - Steps to create, design, and test a talking app.
   - **Strategy: Peer Editing:** Students pair up to follow the setup and check each other's app design and logic..

Extension (Further Exploration)

Discussion Questions:

- What is the importance of using educational apps in today's learning environment?
- What are the advantages of using App Inventor for mobile app development?
- Why are hybrid apps becoming more popular than native or web apps?

Creative Task:

- Create a concept note for an app idea that solves a common school or classroom problem.

Evaluation (Assessments & Review)

- MCQs and Fill in the Blanks from textbook exercise.
- Short answers on types and categories of apps.
- **Lab Activity:** Students use App Inventor to build a simple talking app.
- **Project:** Design a basic app interface using Design and Block Editor views.

Suggested Activity

Project: My First Educational App

- Students create an idea, draw layout, and develop an app using App Inventor focused on a learning task (e.g., quiz app, flashcard app, dictionary).