SEMESTER-IV

COURSE 10: PYTHON PROGRAMMING

Theory **Credits: 3** **3 hrs/week**

Course Objectives

1. **Introduce the foundational concepts** of Python programming including its syntax, IDEs, and control structures.
2. **Develop proficiency in modular programming** using functions, lambda expressions, recursion, and Python's built-in modules and packages.
3. **Explore core data structures** like strings, lists, tuples, and dictionaries for effective data manipulation.
4. **Teach exception handling mechanisms** and the use of regular expressions for pattern matching and text processing.
5. **Enable students to interact with files and databases** using Python to build real-world applications involving persistent storage and data retrieval.

Course Outcomes

At the end of the course, students will be able to:

1. **Write and execute structured Python programs** using variables, expressions, and flow control statements.
2. **Implement modular code** leveraging functions, argument types, recursion, and reusable libraries.
3. **Manipulate and organize data efficiently** using Python's string operations and complex data structures.
4. **Handle runtime errors and apply regular expressions** for robust and flexible program behaviour.
5. **Perform file operations and connect to databases** through Python scripts to store, retrieve, and manage data effectively.

Unit 1. Basics of Python Programming:

Features of python, history of python, Python IDEs, Writing and Executing Python Program, literal constants, variables and identifiers, Data types, input operation- comments, Reserved words, Indentation, Operators and Expressions: Expressions in python, Operations in Strings, Other Data types, Type conversion, Decision control Statements, Iterative Statements, Nested loops, break, Continue, Pass Statements, else statement used with loops.

Unit 2. Strings and Collections:

Strings: Built-in String Methods and Functions

Lists: Access values in List, Updating values in Lists, Nested lists, Basic list operations, List Methods.

Tuple: Creating, Accessing, Updating and Deleting Elements in a tuple,Nested tuples.

Dictionaries: Creating a dictionary, Accessing values, Modifying an Entry, Deleting items, Built-in Dictionary Functions and Methods

Unit 3. Functions and Modules:
Function Definition, Function Call, Variable Scope and lifetime, The return statement, Required Arguments, Keyword Arguments, Default Arguments and Variable Length Arguments, Lambda Functions, Recursive Functions.

Importing Libraries, Modules, Packages in python, Standard library modules- Globals(), Locals(), and Reload(), Function Redefinition.

Unit 4. Exception Handling:
Errors and Exceptions, Handling Exceptions, Multiple Except blocks, Multiple Exceptions in a single block, Except Block without Exception, The else clause, Built-in and user-defined Exceptions, The finally block, Re-raising Exception, Assertions in python

Unit 5. File Handling & Database Connectivity:
Types of files in Python, Opening and Closing files, Reading and Writing files: write() and writelines() methods- append() method, read() and readlines() methods, Splitting words, File Positions.

Database connectivity using Python, Executing SQL commands using python, Assimilating SQL command results using python.

Textbooks:

1. Python Programming using Problem Solving Approach Reema Thareja Oxford University Press 2020
2. Exploring Python, Budd T A, McGraw-Hill Education, 1st Edition, 2011.

Reference Book:

1. Python: The Complete Reference, Martin C. Brown, Mc Graw-Hill, 2018
2. Fundamentals of Python, Kenneth A. Lambert. (2019), First Programs,2nd Edition, CENGAGE Publication.

Activities:

Outcome: Write and execute structured Python programs using variables, expressions, and flow control statements.

Activity: Create a calculator program that uses variables, arithmetic expressions, and flow control (if-else) to perform basic operations (add, subtract, multiply, divide) based on user input.

Evaluation Method: Code walkthrough and output validation. Use a checklist to assess on a 10-point scale to check the:

- Correct use of variables and expressions
- Proper flow control logic
- Accurate results for different inputs

Outcome: Implement modular code leveraging functions, argument types, recursion, and reusable libraries.

Activity: Build a factorial calculator using both iterative and recursive functions. Include parameterized functions and import the math library for comparison.

Evaluation Method: Code review and oral explanation. Assess on 10-point scale based on:

- Function structure and argument usage
- Recursion logic
- Use of reusable libraries

Outcome: Manipulate and organize data efficiently using Python's string operations and complex data structures.

Activity: Develop a contact manager that stores names and phone numbers using dictionaries and lists. Include string formatting and search functionality.

Evaluation Method: Practical demo with test cases. Evaluate based on:

- Use of string methods (split, join, format)
- Data structure selection and manipulation
- Search and retrieval accuracy

Outcome: Handle runtime errors and apply regular expressions for robust and flexible program behaviour.

Activity: Create a form validator that checks email and phone number formats using regular expressions. Include try-except blocks to handle invalid inputs.

Evaluation Method: Scenario-based testing. Assess based on:

- Regex accuracy for pattern matching
- Robust error handling
- Program stability with edge cases

Outcome: Perform file operations and connect to databases through Python scripts to store, retrieve, and manage data effectively.

Activity: Build a student grade logger that reads from a CSV file, stores data in a SQLite database, and allows querying by student name.

Evaluation Method: Lab test with sample data. Evaluate on a 10-point scale:

- File read/write operations
- Database connection and query execution
- Data integrity and retrieval accuracy

SEMESTER-IV

COURSE 10: PYTHON PROGRAMMING

Practical **Credits: 1** **2 hrs/week**

List of Experiments

1. Display a welcome message using print().
2. Declare and use identifiers belonging to strings, integers, floats, and booleans.
3. Accept user input (name, age, height, student status) and display each value with its type using type().
4. Perform operations like .upper(), .find(), .replace() on strings.
5. Write a program to reverse the string, count vowels and words.
6. Write a program for slicing, sorting, and list comprehension.
7. Program to apply list methods: append(), extend(), insert(), remove(), pop(), sort().
8. Create tuples to store student (name, age, course) data and perform
    a. Accessing elements using indexing and slicing.
    b. Demonstrate immutability by attempting to modify a tuple.
    c. Create and navigate nested tuples.
9. Create a dictionary with student roll numbers as keys and names/marks as values.
    a. Accessing, adding, updating, and deleting key-value pairs.
    b. Iterating through keys, values, and items.
10. Write a program to demonstrate variable length arguments.
11. Write a program to illustrate lambda and recursive functions.
12. Write a program to demonstrate Globals(), Locals(), and Reload() functions.
13. Demonstrate exception handling and assertions in Python.
14. Write a Python program to copy the contents of one file into another in reverse order.
15. Write a program to connect to the database and retrieve the required information using SQL commands.