FACULTY OF ENGINEERING & TECHNOLOGY

Effective from Academic Batch: 2025-26

Programme: **Bachelor of Technology (CSD, AIDS AND IOT SEM 3)**

Semester: **III**

Course Code: **102040306**

Course Title: **Introduction to Python Programming**

Course Group: **Professional Core Course**

Course Objectives:

The objective of this course is to introduce students to Python as a simple, powerful, and versatile programming language. The course aims to develop strong programming fundamentals, logical thinking, and problem-solving skills. Students will learn core concepts such as variables, control structures, functions, data structures, file handling, and basic object-oriented programming. The course also prepares students for advanced computing courses and real-world programming applications by strengthening algorithmic thinking using Python.

Teaching & Examination Scheme:

Contact hours per week			Course Credits	Examination Marks (Maximum / Passing)				
				Theory		J/V/P*		
Lecture	Tutorial	Practical		Internal	External	Internal	External	Total
2	0	2	3	50 / 18	50 / 18	25 / 9	25 / 9	150 / 54

* **J**: Jury; **V**: Viva; **P**: Practical

Detailed Syllabus:

Sr.	Contents	Hours
1	**Introduction to Python and Programming Fundamentals:** Introduction to programming, History and features of Python, Applications of Python, Installing Python, Python interpreter and IDEs, Writing first Python program, Python syntax rules, Indentation, Keywords, Comments.	3
2	**Data Types, Variables, and Operators:** Identifiers and variables, Built-in data types, Type conversion, Input and output functions, Operators and expressions.	4

3	**Control Flow Statements:** Decision making: if, if-else, nested if, elif, Looping statements: for loop, while loop, range function, break, continue, pass statements, Simple problem-solving using control structures.	**5**
4	**Functions and Modular Programming:** Built-in functions, User-defined functions, Function arguments, return values, Scope of variables, Introduction to lambda functions, Introduction to recursion, Introduction to modules, and importing standard modules.	**4**
5	**Python Data Structures and File Handling:** Lists, Tuples, Sets, Dictionaries, and their operations, String creation and operations, String methods and formatting, Traversing data structures, Built-in methods, comprehensions, and practical use cases of data structures, Introduction to file handling, File modes, reading from and writing to text files, Basic file operations.	**5**
6	**Exception Handling and Introduction to Object-Oriented Programming** Types of errors, Exception handling using try, except, else, finally, Introduction to object-oriented programming, Classes and objects, Constructors, Simple class-based examples.	**5**
	Total:	**26**

List of Practicals / Tutorials:

1	A. Write a Python program to accept two numbers from the user and perform addition, subtraction, multiplication, and division. B. Write a Python program to calculate the area and circumference of a circle by accepting the radius from the user. C. Write a Python program to convert temperature from Celsius to Fahrenheit and Fahrenheit to Celsius based on user choice.
2	A. Write a Python program to check whether a given number is positive, negative, or zero. B. Write a Python program to find the largest of three numbers entered by the user using nested if–else statements. C. Write a Python program to calculate the grade of a student based on marks entered by the user.
3	A. Write a Python program to generate the Fibonacci series up to n terms entered by the user. B. Write a Python program to find the factorial of a given number using a while loop. C. Write a Python program to display all prime numbers between two given numbers.

4	A. Write a Python program to define a function that checks whether a number is a palindrome. B. Write a Python program to define a function that calculates simple interest using parameters and return values. C. Write a Python program to define a function that returns the maximum and minimum values from a list. D. Write a Python program to compute the factorial of a given number using a recursive function.
5	A. Write a Python program to create a list of n integers and perform insertion and deletion operations on the list. B. Write a Python program to store elements in a tuple and access the tuple elements using indexing and looping statements. C. Write a Python program to create a dictionary with key–value pairs and perform basic operations such as insertion, deletion, and retrieval of values.
6	A. Write a Python program to perform string operations such as concatenation, length calculation, case conversion, and substring extraction. B. Write a Python program to read the contents of a text file using appropriate file handling modes and display them on the screen. C. Write a Python program to write data into a text file and append additional data to the same file using file handling operations.
7	A. Write a Python program to handle division by zero using try and except blocks. B. Write a Python program to handle multiple exceptions in a single program. C. Write a Python program to handle runtime errors using try, except, else, and finally blocks.
8	A. Write a Python program to create a class and an object and access class attributes and methods. B. Write a Python program to initialize class data using a constructor. C. Write a Python program to implement a simple class-based application using object-oriented concepts.
9	A. Write a Python program to generate five random integers between 1 and 100 using the random module and calculate their square roots using the math module. B. Write a Python program to define a lambda function that accepts two numbers and returns the greater number. C. Write a Python program to create a list of integers and use built-in functions len(), max(), min(), and sum() to display the total number of elements, maximum value, minimum value, and sum of elements in the list.

10	A. Write a Python program to implement a stack using a list with operations such as push, pop, and display. B. Write a Python program to develop a small game where the computer generates a random number and the user guesses the number within a limited number of attempts. C. Write a Python program to create a simple contact management system that allows adding new contacts, saving contact details to a text file, and retrieving and displaying the saved contacts.
11	Design and implement a mini-project using Python concepts covered in the course.

Reference Books:

1	Reema Thareja, *Python Programming Using Problem Solving Approach*, Oxford University Press.
2	Mark Lutz, Learning Python, 5th Edition, O'Reilly Media.
3	Charles R. Severance, Python for Everybody: Exploring Data Using Python 3, Pearson Education.
4	Guido van Rossum and Fred L. Drake Jr., An Introduction to Python, Network Theory Ltd.

Supplementary learning material:	
1	https://www.python.org/
2	https://docs.python.org/3/
3	https://www.w3schools.com/python/
4	https://www.tutorialspoint.com/python/index.htm
5	https://realpython.com/

Pedagogy:

- Direct classroom teaching
- Audio Visual presentations/demonstrations
- Assignments/Quiz
- Continuous assessment
- Interactive methods
- Seminar/Poster Presentation
- Industrial/ Field visits
- Course Projects

Suggested Specification table with Marks (Theory) (Revised Bloom's Taxonomy):

Distribution of Theory Marks in %					
R	**U**	**A**	**N**	**E**	**C**
12%	18%	32%	18%	8%	12%

R: Remembering; **U**: Understanding; **A**: Applying; **N**: Analyzing; **E**: Evaluating; **C**: Creating

Note: This specification table shall be treated as a general guideline for students and teachers. The actual distribution of marks in the question paper may vary slightly from above table.

Student Learning Outcomes (SLOs):

Sr.	Course Outcome Statements	%weightage
SLO-1	Explain the fundamentals of programming and basic Python syntax to write simple Python programs.	**15%**
SLO-2	Apply appropriate Python data types, variables, operators, and input–output functions to solve basic computational problems.	**15%**
SLO-3	Analyze problem statements and implement decision-making and looping constructs to develop logical Python programs.	**20%**
SLO-4	Design and implement modular Python programs using user-defined functions, recursion, and standard library modules.	**20%**
SLO-5	Develop Python programs using data structures, string manipulation, and file handling techniques for efficient data management.	**20%**
SLO-6	Create simple Python applications using exception handling and basic object-oriented programming principles.	**10%**

Curriculum Revision:	
Version:	3.0
Drafted on (Month-Year):	March-2026
Last Reviewed on (Month-Year):	-
Next Review on (Month-Year):	April-2029