25UAMT303	OBJECT ORIENTED PROGRAMMING IN PYTHON	Category	L	T	P	R	Credit	Year of Introduction
		PBL	3	0	0	1	**4**	**2025**

Preamble

This course introduces the principles of Object-Oriented Programming using Python. The course focuses on designing reusable software components using classes and objects and implementing encapsulation, abstraction, inheritance, and polymorphism. It enables students to develop modular and scalable applications using Python.

Prerequisite: Nil

Course Outcomes: After the completion of the course, the student will be able to:

CO	Course Outcomes	Knowledge Level
CO1	Explain Python programming fundamentals including control structures and data structures.	K2 Understand
CO2	Design and implement classes and objects applying encapsulation and abstraction.	K3 Apply
CO3	Apply inheritance and polymorphism to develop reusable applications	K3 Apply
CO4	Develop modular object-oriented applications using file handling, exception handling and packages.	K6 Create

Mapping of course outcomes with program outcomes and program specific outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	2	–	–	1	–	–	–	–	–	–	3	1
CO2	3	3	3	1	2	–	–	–	1	–	1	3	2
CO3	3	3	3	1	2	–	–	–	1	–	–	3	3
CO4	3	3	3	2	3	–	–	–	2	1	1	3	3
Avg	3	2.6	3	1.3	2	–	–	–	1.3	1	1	3	2.3

Mark Distribution:

Total Marks	CIE Marks	ESE Marks	ESE Duration
100	**60**	**40**	**2.5 hrs**

Continuous Internal Evaluation Pattern:

Attendance	5
Continuous Evaluation	25
Mini Project	30
Total	60

SYLLABUS

Module 1 (13 hrs)

Introduction to Python- Features of Python, Applications of Python, Installation process, Python Interpreter, Integrated Development Environments (IDEs). Variables and Data Types- Numeric types (int, float, complex), Boolean type, Type conversion and casting. Operators- Arithmetic operators, Relational operators, Logical operators, Assignment operators, Bitwise operators, Membership operators, Identity operators. Input and Output Statements- input() function, print() function, Formatting output, f-strings. Conditional Statements- if, if-else, nested if, elif ladder. Looping Statements- for loop, while loop, break, continue, pass. Strings- String indexing, slicing, string methods. Data Structures- Lists, Tuples, Sets, Dictionaries, Built-in functions. Functions- Defining functions, parameters, return values, default arguments, keyword arguments, recursion, lambda functions.

Module 2 (14 hrs)

Introduction to OOP- Procedural programming vs Object Oriented Programming, Characteristics of OOP. Classes and Objects- Class definition, Object creation, self-keyword. Constructors and Destructor- __init__() method, __del__() method, Instance variables. Types of Methods- Instance methods, Class methods, Static methods. Encapsulation- Data hiding, Public, Private and Protected members. Abstraction- Abstract classes, abc module, Abstract methods. Practice Applications- Class-based applications (Student Management System / Bank Management System).

Module 3 (14 hrs)

Inheritance- Single inheritance, Multiple inheritance, Multilevel inheritance, Hierarchical inheritance. Method Overriding- Rules of overriding, super() function. Method Resolution Order (MRO)- Concept of MRO, C3 Linearization. Polymorphism- Duck typing, Operator

overloading. Magic Methods- __str__(), __repr__(), __add__(), __len__(). Exception Handling- try, except, else, finally, Raising exceptions. User Defined Exceptions- Creating custom exception classes.

Module 4 (15 hrs)

File Handling- File modes, Reading and Writing text files. Working with Files- Binary files, with statement. CSV and JSON Handling- Reading and writing CSV files, JSON parsing and serialization. Modules- Creating modules, import statement, from-import. Packages- Creating packages, __init__.py, Package structure. Iterators- Iterables, __iter__(), __next__(). Generators- yield keyword, Generator functions. Mini Project- Design and implementation of an OOP-based real-world and sustainable application.

Completion of an industrial visit or industrial training is desirable for this course.

Text Books

1. Reema Thareja, Python Programming: Using Problem Solving Approach, Oxford University Press, 3rd Edition (2025), ISBN-13: 978-9367257654.
2. Allen B. Downey, Think Python: How to Think Like a Computer Scientist, O'Reilly Media, 3rd Edition (2024), ISBN-13: 978-1098155438.
3. Paul Barry, Head First Python: A Brain-Friendly Guide, O'Reilly Media, 3rd Edition (2023), ISBN-13: 978-1491919530 .

Reference Books

1. Mark Lutz, Learning Python, O'Reilly Media, 6th Edition (2025).
2. Herbert Schildt, Python: The Complete Reference, McGraw Hill Education, 3/e (2018)
3. Wesley J. Chun, Core Python Applications Programming, Pearson Education, 3/e (2015) (ISBN-13: 978-9332555365)

PBL Course Elements

L: Lecture (3 Hrs.)	**R: Project (1 Hr.), 2 Faculty Members**		
	Tutorial	**Practical**	**Presentation**
Lecture delivery	Project identification	Simulation/ Laboratory Work/ Workshops	Presentation (Progress and Final Presentations)
Group discussion	Project Analysis	Data Collection	Evaluation
Question answer Sessions/ Brainstorming Sessions	Analytical thinking and self-learning	Testing	Project Milestone Reviews, Feedback, Project reformation (If required)
Guest Speakers (Industry Experts)	Case Study/ Field Survey Report	Prototyping	Poster Presentation/ Video Presentation: Students present their results in a 2 to 5 minutes video

Assessment and Evaluation for Project Activity

Sl. No	**Evaluation for**	**Allotted Marks**
1	Project Planning and Proposal	5
2	Contribution in Progress Presentations and Question Answer Sessions	4
3	Involvement in the project work and Team Work	3
4	Execution and Implementation	10
5	Final Presentations	5

6	Project Quality, Innovation and Creativity	3
Total		**30**

1. Project Planning and Proposal (5 Marks)

- Clarity and feasibility of the project plan
- Research and background understanding
- Defined objectives and methodology

2. Contribution in Progress Presentation and Question Answer Sessions (4 Marks)

- Individual contribution to the presentation
- Effectiveness in answering questions and handling feedback

3. Involvement in the Project Work and Team Work (3 Marks)

- Active participation and individual contribution
- Teamwork and collaboration

4. Execution and Implementation (10 Marks)

- Adherence to the project timeline and milestones
- Application of theoretical knowledge and problem-solving
- Final Result

5. Final Presentation (5 Marks)

- Quality and clarity of the overall presentation
- Individual contribution to the presentation
- Effectiveness in answering questions

6. Project Quality, Innovation, and Creativity (3 Marks)

- Overall quality and technical excellence of the project
- Innovation and originality in the project
- Creativity in solutions and approaches

END SEMESTER EXAMINATION – MODEL QUESTION PAPER

Course Code	**25UAMT303**
Course Name	**OBJECT ORIENTED PROGRAMMING IN PYTHON**
Max. Marks: 40	**Duration: 2.5 Hours**

Part A
(Answer all questions. Each question carries 2 marks)

Qn No.	Question	Marks	CO	BL
1.	Explain the different data types in Python. Write a program demonstrating type conversion between int, float, and string.	2	CO1	L3
2.	Write a Python program using recursion to compute the nth Fibonacci number. Trace the execution for n=5.	2	CO1	L3
3.	Explain encapsulation in Python. Illustrate a class BankAccount with private attributes and methods to deposit, withdraw, and display balance.	2	CO2	L3
4.	Differentiate instance methods, class methods, and static methods with suitable examples.	2	CO2	L2
5.	Explain Method Resolution Order (MRO) in Python. Trace the MRO for a class D inheriting from B and C, where both inherit from A (diamond problem).	2	CO3	L2
6.	Explain operator overloading in Python. Write a class Complex to overload the + operator for complex number addition.	2	CO3	L3
7.	Write a Python program to read a CSV file containing student records and display students who scored more than 75%.	2	CO4	L3
8.	Explain Python generators. Write a generator function to produce Fibonacci numbers up to n terms.	2	CO4	L3

Part B
(Answer any one full question from each Module. Each carry 6 marks)

Module I

Qn No.	Question	Marks	CO	BL
9.1	Explain the built-in data structures of Python: List, Tuple, Set, and Dictionary. Compare their features and demonstrate at least three operations on each.	3	CO1	L2
9.2	Write a Python program using lambda functions and list comprehensions to filter even numbers from a list and compute their squares.	3	CO1	L3
OR				
10.1	Explain looping constructs in Python. Write a program to display a multiplication table for numbers 1 to 10 using nested loops.	3	CO1	L3
10.2	Write a Python function to sort a list of dictionaries by a specified key using default and keyword arguments.	3	CO1	L3
Module II				
Qn No.	**Question**	**Marks**	**CO**	**BL**
11.1	Illustrate a class hierarchy for a Student Management System using encapsulation and abstraction. Include attributes for name, roll number, marks, and methods for computing grade and displaying results.	3	CO2	L3
11.2	Explain abstract classes and abstract methods in Python using the abc module. Demonstrate with a class Shape having abstract methods area() and perimeter(), implemented by Rectangle and Circle.	3	CO2	L3
OR				
12.1	Explain constructors and destructors in Python. Write a program demonstrating __init__() and __del__() methods with a class Resource that manages a file resource.	3	CO2	L3
12.2	Demonstrate all types of methods (instance, class, and static) with a class Employee that tracks the employee count and computes annual salary.	3	CO2	L3
Module III				
Qn No.	**Question**	**Marks**	**CO**	**BL**

13.1	Explain all types of inheritance in Python with examples. Demonstrate multiple inheritance and resolve the diamond problem using MRO and super().	3	CO3	L3
13.2	Write a Python program demonstrating user-defined exceptions. Create a custom exception InvalidAgeError and use it in a function that validates age for voting eligibility.	3	CO3	L3
OR				
14.1	Explain duck typing and operator overloading in Python. Write a class Vector to overload +, -, and * (scalar multiplication) operators.	3	CO3	L3
14.2	Explain magic methods in Python. Demonstrate the use of __str__(), __repr__(), __add__(), and __len__() with a suitable class.	3	CO3	L3
Module IV				
Qn No.	**Question**	**Marks**	**CO**	**BL**
15.1	Illustrate a mini project for a Library Management System using OOP concepts in Python. The system should support adding books, issuing and returning books, and maintaining member records using file handling with JSON.	3	CO4	L3
15.2	Explain iterators and generators in Python. Write a generator that lazily yields prime numbers up to a given limit.	3	CO4	L3
OR				
16.1	Ilustrate Python packages and modules. Create a package mathutils with modules for arithmetic and statistical operations, and demonstrate its import and usage.	3	CO4	L3
16.2	Write a Python program to read and write binary files. Demonstrate the use of the with statement and explain its advantages over explicit file handling.	3	CO4	L3