Semester: 2	
Programme: Data Science	
Course: Python Programming	
Paper code: C1DS250211P	**Credits: 4**
Hours/week: 5	
Category: Core	
Theory / Practical / Composite: Practical	
No of Module: 1	

Course Outcome:

1. **Remember** the fundamental syntax of Python, including various operators (arithmetic, relational, logical, bitwise), built-in keywords, and standard data types such as Numeric, String, List, Dictionary, Tuple, and Set.
2. **Understand** the mechanisms of program execution, function scopes, and the purpose of modules and packages in organizing and managing namespaces within a Python environment.
3. **Apply** sequential statements, conditional logic, loops (for and while), and file handling techniques to perform basic data input/output operations and manage file exceptions.
4. **Analyze** the performance and logic of functional programming tools—specifically lambda, map, filter, and reduce—to process complex data structures efficiently.
5. **Evaluate** the implementation of Object-Oriented Programming (OOP) principles, such as encapsulation, inheritance, and polymorphism, along with exception handling strategies to ensure code robustness and reusability.
6. **Create** modular applications by synthesizing custom functions, user-defined classes, and supporting libraries like NumPy and pandas to solve data-centric programming problems.

Prerequisites: ***Basic knowledge about any prior course***

SYLLABUS

Module/Unit	Content	Hours Or Number of Classes	CO Mapping	Cognitive Level
1	**Introduction to Python** Applications; technical strengths; Python programming environments; program execution in Python	2	CO2	K2
2	**Python Operators and Statements** Various operators in Python – assignment, arithmetic, relational, logical and bitwise; Sequential statements; importance of indentation; if statement and its variations; loops – for and while; keywords used with loops – break, continue, pass, loop else	12	CO1, CO3	K1, K3
3	**Python Types and Operations** Numeric types; String; List; Dictionary; Tuple; Set	14	CO1	K1
4	**Functions in Python** The def keyword; function scope; function arguments; recursive functions; anonymous	14	CO2, CO4, CO6	K2, K4, K6

	functions – the lambda keyword; functional programming tools – filter, map and reduce			
5	**Modules and Packages** Need for modules; working of imports; module creation; module usage; module namespaces; package imports; need for package imports	4	CO2	K2
6	**Object-Oriented Programming (OOP) in Python** OOP principles and their implementation in Python – data hiding, encapsulation, abstraction, polymorphism and inheritance; exception handling	14	CO5, CO6	K5, K6
7	**File Handling** Opening a file; file modes; reading a file; writing to a file; closing a file; file handling exceptions	6	CO3	K3
8	**Supporting Libraries in Python** Basic numpy and pandas processing of data	6	CO6	K6
Text Books				
1. Python for Everybody, Author: Charles R. Severance, Publisher: Shroff Publishers				
2. Learning Python, Author: Mark Lutz, Publisher: O'Reilly				
3. Python Programming: Using Problem Solving Approach, Author: Reema Thareja, Publisher: Oxford				
Evaluation	Continuous Practical Assessments			

Course outcomes (COs) and Cognitive Level Mapping

COs	CO Description	Cognitive levels
CO1	**Remember** the fundamental syntax of Python, including various operators (arithmetic, relational, logical, bitwise), built-in keywords, and standard data types such as Numeric, String, List, Dictionary, Tuple, and Set.	K1
CO2	**Understand** the mechanisms of program execution, function scopes, and the purpose of modules and packages in organizing and managing namespaces within a Python environment.	K2
CO3	**Apply** sequential statements, conditional logic, loops (for and while), and file handling techniques to perform basic data input/output operations and manage file exceptions.	K3
CO4	**Analyze** the performance and logic of functional programming tools—specifically lambda, map, filter, and reduce—to process complex data structures efficiently.	K4
CO5	**Evaluate** the implementation of Object-Oriented Programming (OOP) principles, such as encapsulation, inheritance, and polymorphism, along with exception handling strategies to ensure code robustness and reusability.	K5
CO6	**Create** modular applications by synthesizing custom functions, user-defined classes, and supporting libraries like NumPy and pandas to solve data-centric programming problems	K6