Semester: II	
Programme: M.Sc. Data Science	
Course: Advanced Python Programming	
Paper code: M2C4DS26021P	**Credits: 6**
Hours/week: 8 (8Pr)	
Category: Core	
Theory / Practical / Composite: Practical	
No of Modules: 1	

Course Outcome:

1. **Remember** the fundamental syntax and structural elements of Python, including operators, built-in data types (Numeric, String, List, Dictionary, Tuple, Set), file modes, and module import mechanisms.
2. **Understand** the mechanics of program execution, function scopes, module namespaces, Object-Oriented Programming (OOP) principles, and the memory structures underlying high-performance frameworks like Apache Arrow.
3. **Apply** advanced functional programming features—such as decorators, generators, closures, comprehensions, and iterators—alongside file handling and exception management to write modular and reusable code.
4. **Analyze** numerical computing, data manipulation, and visualization workflows using supporting libraries (NumPy, Pandas, and Matplotlib) to process and clean complex datasets efficiently.
5. **Evaluate** high-performance data processing technique (e.g. including Polars lazy evaluation, expression trees, and query optimization against Pandas limitations—to determine optimal data execution strategies.)
6. **Create** parallel and distributed computing pipelines using multiprocessing shared memory to solve data-intensive and computationally demanding problems.

Prerequisites: ***Basic knowledge about Python***

SYLLABUS

UNIT/Module	CONTENT	NUMBER OF CLASSES	CO Mapping	COGNITIVE LEVEL
I	Revision of Python Fundamentals; Python programming environments	4	**C01, C02**	**K1, K2**
II	**Data Types and collections:** Numeric types; String; List; Dictionary; Tuple; Set; Mutability/immutability; references/aliasing; shallow/deep copy; garbage collection.	6	**C01**	**K1**

III	**Functions in Python:** function definition, function scope and arguments; anonymous functions /Lambda; map/filter/reduce; closures; decorators; generators; iterators; itertools; lazy evaluation.	8	**C01**	**K1**
IV	**Modules and Packages:** Module creation; module usage; module namespaces; package imports; dependency/environment management.	7	**C02, C03**	**K2,K3**
V	**Object-Oriented Programming (OOP) in Python:** OOP principles and their implementation in Python.	7	**C01,C03**	**K1,K3**
VI	**File Handling and Structured Data:** File modes and operations; context managers; handling text and binary files; CSV and JSON processing; serialization and deserialization; pickle; regular expressions; incremental file processing; streaming file processing.	20	**C04**	**K4**
VII	**Linear Algebra using NumPy :** Various operations on NumPy array; Element-wise operations; inverse, determinant and rank ; solving systems of linear equations; eigenvalues and eigenvectors; Singular Value Decomposition (SVD); matrix decompositions	12	**C02,C05**	**K2,K5**
VIII	**Pandas and Data Manipulation:** Series and DataFrames; import from various file formats; various operations on pandas DataFrames; Data Cleaning;	10	**C04**	**K4**
IX	**Data Visualisation using Matplotlib:** visualization of different types of data; saving and exporting visualisations;	10	**C05**	**K5**
X	**High-Performance Data Processing:** Limitations of Pandas; vectorization vs loops; memory efficiency; Pandas performance tuning; Apache Arrow; Polars; lazy evaluation; expression trees;	10	**C06**	**K6**
XI	**Parallel and Distributed Computing:** Multiprocessing for CPU-bound pipelines, shared memory, and introduction to distributed DataFrames .	10	**C06**	**K6**

Text Books

1. Learning Python by Mark Lutz, O'Reilly Media.
2. Programming Python by Mark Lutz, O'Reilly Media.
3. Python for Data Analysis (3rd Edition) by Wes McKinney (O'Reilly)
4. Data Processing with Dask by Michael Thomas (Manning Publications)
5. High-Performance Python (2nd Edition) by Micha Gorelick and Ian Ozsvald (O'Reilly)

Web Resources
1. The Joy of Computing using Python, IIT Ropar, Prof. Sudarshan Iyengar
2. Python for Data Science, IIT Madras, Prof. Ragunathan Rengasamy

Course outcomes (COs) and Cognitive Level Mapping

COs	CO Description	Cognitive levels
CO1	**Remember** the fundamental syntax and structural elements of Python, including operators, built-in data types (Numeric, String, List, Dictionary, Tuple, Set), file modes, and module import mechanisms.	K1
CO2	**Understand** the mechanics of program execution, function scopes, module namespaces, Object-Oriented Programming (OOP) principles, and the memory structures underlying high-performance frameworks like Apache Arrow.	K2
CO3	**Apply** advanced functional programming features—such as decorators, generators, closures, comprehensions, and iterators—alongside file handling and exception management to write modular and reusable code.	K3
CO4	**Analyze** numerical computing, data manipulation, and visualization workflows using supporting libraries (NumPy, Pandas, and Matplotlib) to process and clean complex datasets efficiently.	K4
CO5	**Evaluate** high-performance data processing technique (e.g. including Polars lazy evaluation, expression trees, and query optimization against Pandas limitations—to determine optimal data execution strategies.)	K5
CO6	**Create** parallel and distributed computing pipelines using multiprocessing shared memory to solve data-intensive and computationally demanding problems.	K6