Semester: I	
Programme: M.Sc. Data Science	
Course: Data Structures using Python	
Paper code: MDTS4113	**Credits: 6**
Hours/week: 7 (4Th + 3Pr)	
Category: Core/MDC/SEC/VAC: Core	
Theory / Practical / Composite: Composite	
No of Modules: 1	

Course Overview:

Course Outcome:

1. Remember the fundamental concepts of data structures, Abstract Data Types (ADT), and the essential syntax of Python, including data types, conditional statements, and iterations.
2. Understand the structural characteristics and operational logic of linear data structures (arrays, strings, linked lists, stacks, queues) and non-linear data structures (binary trees, BST, and AVL trees).
3. Apply searching and sorting algorithms—such as binary search, quick sort, and heap sort—along with collision resolution techniques in hashing to manage and organize data effectively.
4. Analyse the efficiency of various algorithmic paradigms and data structures through formal algorithm analysis to determine their performance impact.
5. Evaluate the suitability of different data structures and sorting techniques for specific computational problems to optimize resource usage and solve complex data challenges.
6. Create robust Python-based implementations of complex data structures and custom ADTs to solve real-world problems by synthesizing theoretical principles with practical programming.

SYLLABUS

UNIT/Module	CONTENT	HOURS or NUMBER OF CLASSES	CO Mapping	COGNITIVE LEVEL
I.	**Fundamentals:** Introduction to Data Structures; Abstract Data Type (ADT); Types of Data Structures. Algorithm Analysis; Introduction to Advanced Algorithmic Paradigm.	**7**	CO1, CO4	KO1, KO4
II.	**Essentials of Python:** Data Types, Conditional Statement, Iterations	**7**	CO1	KO1
III.	**Linear Data Structures:** Array; Strings; Linked List; Stack; Queue.	**12**	CO2, CO5, CO6	KO2, KO5, KO6
IV.	**Searching and Sorting algorithms:** Linear Search; Binary Search; Bubble Sort; Selection Sort Insertion Sort, Quick Sort; Merge Sort; Heap Sort,	**10**	CO3, CO5	KO3, KO5
V.	**Non-linear Data Structures:** Binary Tree; Binary Search Tree (BST); AVL Tree,	**10**	CO2, CO5, CO6	KO2, KO5, KO6
VI.	**Hashing:** Hash tables; Hash functions; Collision resolution techniques.	**6**	CO3, CO5	KO3, KO5
VII.	**Practical:**	**39**		

	Introduction to Python Programming; Data Structures using Python			

Text Books

1. **Goodrich, M. T., Tamassia, R., & Goldwasser, M. H.** (2013). *Data Structures and Algorithms in Python*. John Wiley & Sons
2. **Sheehy, D. R.** (2022). *A First Course on Data Structures in Python*. (Published independently / Open-Source Edition)
3. **Samanta, D.** (2009). *Classic Data Structures* (2nd ed.). PHI Learning.
4. **Thareja, R.** (2025). *Data Structures and Algorithms*. Oxford University Press.
5. **Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C.** (2022). *Introduction to Algorithms*
6. **Horowitz, E., Sahni, S., & Rajasekaran, S.** (2008). *Fundamentals of Computer Algorithms* (2nd ed.). Universities Press (Orient Blackswan)

Evaluation

Paper Structure for Theory Semester Exam Module:

Marks	**Theory** **CIA:** 10 **End Sem Exam:** 50 **Total:** 60	**Practical** **Continuous Assessment:** 40
Paper Structure for Theory Semester Exam	Short questions: 5 marks each 2 out of 4	Long Questions: 10 Marks each 4 out of 6

Course outcomes (COs) and Cognitive Level Mapping

COs	CO Description	Cognitive levels
CO1	Remember the fundamental concepts of data structures, Abstract Data Types (ADT), and the essential syntax of Python, including data types, conditional statements, and iterations.	KO1
CO2	Understand the structural characteristics and operational logic of linear data structures (arrays, strings, linked lists, stacks, queues) and non-linear data structures (binary trees, BST, and AVL trees).	KO2
CO3	Apply searching and sorting algorithms—such as binary search, quick sort, and heap sort—along with collision resolution techniques in hashing to manage and organize data effectively.	KO3
CO4	Analyse the efficiency of various algorithmic paradigms and data structures through formal algorithm analysis to determine their performance impact.	KO4
CO5	Evaluate the suitability of different data structures and sorting techniques for specific computational problems to optimize resource usage and solve complex data challenges.	KO5
CO6	Create robust Python-based implementations of complex data structures and custom ADTs to solve real-world problems by synthesizing theoretical principles with practical programming.	KO6