Title of Paper: Major Practical 3

Sr.No.	Heading	Particulars
1	**Description the course: Including but not limited to:**	This course offers a comprehensive exploration of advanced Python programming concepts, designed to equip students with the tools to tackle real-world problems efficiently. It covers a range of topics, including text processing with regular expressions to identify patterns and extract meaningful data, as well as file handling techniques for both text and binary files. Students will also gain expertise in manipulating and comparing dates using Python's built-in date and time modules, along with performing calendar-based operations. The course emphasizes performance optimization by teaching students how to measure and improve program execution time. Additionally, students will learn how to extract structured data, such as hyperlinks from HTML files, and apply these techniques in practical scenarios. By the end of the course, students will be adept at solving complex problems, optimizing their Python solutions, and utilizing advanced programming concepts to handle diverse data processing tasks.
2	**Vertical:**	Major
3	**Type:**	Practical
4	**Credits:**	2 credits (30 Hours of Practical work in a semester)
5	**Hours Allotted:**	30 Hours
6	**Marks Allotted:**	50 Marks
7	**Course Objectives (CO):** 1. Understand fundamental programming concepts in Python, including input/output operations, conditional statements, and loops. 2. Understand and apply array operations, indexing, slicing, and mathematical functions using NumPy. 3. Develop problem-solving skills by using functions, recursive logic, lambda expressions, and modular programming. 4. Use data structures like lists and dictionaries and perform file operations. 5. Work with text processing, file handling, date manipulation, and performance analysis using advanced Python programming concepts 6. To provide hands-on experience in implementing fundamental data structures like arrays, linked lists, stacks, queues, trees, and graphs. 7. To develop skills in algorithm design and analysis for solving computational problems using data structures. 8. To enable students to choose appropriate data structures for different applications and justify their choices. 9. To enhance understanding of dynamic memory allocation and efficient data management techniques. 10. To equip students with the ability to debug and optimize code for data structure operations.	
8	**Course Outcomes (OC):** . OC 1. Apply Python programming concepts like input/output, conditional statements, and loops, to solve fundamental problems effectively. . OC 2. Demonstrate proficiency in performing basic operations, indexing, slicing, and analyzing attributes of arrays using NumPy. . OC 3. Apply functions, recursion, and lambda expressions to solve computational problems, and implement modular programming for reusable and efficient code design.	

	. OC 4. Implement lists and dictionaries, perform file operations, and use functions to solve real-world problems effectively. . OC 5. Process text, extract information, handle dates, and measure execution time for solving complex real-world problems. . OC 6. Ability to implement and manipulate basic and advanced data structures to solve real-world problems. . OC7 Proficiency in writing efficient algorithms using suitable data structures for operations like searching, sorting, and traversal. . OC8 Capability to analyze the time and space complexity of algorithms for various data structures. . OC9 Enhanced problem-solving skills by applying data structures in different domains such as databases, networks, and operating systems	
9	**Module 1** 1. Write programs for the following: a. Write a program that asks the user to enter their name and their age. Print out a message addressed to them that tells them the year that they will turn 100 years old. b. Write a program to accept a number from the user and depending on whether the number is even or odd, print out an appropriate message to the user. c. Write a program to accept the SGPI from the user and print corresponding grade based on the following: d. SGPI Grade e. 9.00 – 10.00 O f. 8.00 – 8.99 A+ g. 7.00 – 7.99 A h. 6.00 – 6.99 B+ i. 5.50 – 5.99 B j. 5.00 – 5.49 C k. 4.00 – 4.99 P l. Below 4 F 2. Write programs for the following: a. d. Write a program to generate the Fibonacci series. b. e. Write a program to accept a number from the user display sum of its digits. 3. Write programs for the following: a. Write a program to perform basic operations, indexing and slicing on arrays. b. Write a program to implement mathematical functions on arrays. c. Write a program to perform array aliasing and copying. 4. Write programs for the following: a. Write a program to perform slicing, basic and advanced indexing on b. NumPy arrays. c. e.Write a program to analyze dimensions and attributes of arrays 5. Write programs for the following: a. Write a function to check the input value is Armstrong and also write the function for Palindrome. b. Write a recursive function to print the factorial for a given number. c. Write a lambda function that checks whether a given string starts with a specific character. 6. Write programs for the following: a. Write a program to compute number of characters and words in a string.	30 Hrs

	b. Create a file geometry.py to calculate base areas for shapes square and circle. In another file, write a function pointyShapeVolume(x, y, squareBase) that calculates the volume of a square pyramid if squareBase is True and of a right circular cone if squareBase is False. x is the length of an edge on a square if squareBase is True and the radius of a circle when squareBase is False. y is the height of the object. First use squareBase to distinguish the cases. Use the circleArea and squareArea from the geometry module to calculate the base areas. 7. Write programs for the following: a. Write a program that takes two lists and returns True if they have at least one common member. b. Write a Python script to sort (ascending and descending) a dictionary by value. 8. Write programs for the following: a. Write a program to accept and pass radius to a function that returns area and circumference (using tuple). b. Write a program to perform basic file operations on text files and binary files. c. Write a Python program to read last n lines of a file. 9. Write programs for the following: a. a.Write a program to count the occurrences of a specific word in a file using regular expressions. b. b.Write a program to extract all hyperlinks (<a href="...">) from an HTML file. 10. Write programs for the following: a. Write a program that compares two dates (in DD/MM/YYYY format) and prints which one is earlier. b. Write a program to measure program execution time. c. Write a program using the calendar module to print the weekday of the first day of a given month and year.	
	Module 2	30 Hrs
	1. Array Operations: Write a program to implement basic array operations: a. Insert an element at a specific position in an array. b. Delete an element from a specific position in an array. c. Search for an element in an array (linear search). 2. Linked List Manipulation: Write a program to: a. Create a singly linked list. b. Insert a node at the beginning, end, and at a given position in a linked list. c. Delete a node from a given position in a linked list. 3. Stack Application: Write a program to: a. Implement a stack using an array. b. Convert an infix expression to postfix notation using a stack. 4. Queue Application: Write a program to: a. Implement a queue using an array. b. Simulate a simple queuing system (e.g., customer service queue). 5.Binary Search Tree: Write a program to: a. Create a binary search tree. b. Insert nodes into a binary search tree. c. Search for a node in a binary search tree. 6. Tree Traversal: Write a program to: a. Implement pre-order, b. in-order,	

	c. Post-order traversal of a binary tree. 7.Hash Table: Write a program to: a. Implement a hash table with separate chaining for collision handling. b. Store and retrieve data from the hash table. 8.Sorting Algorithms: Write programs to implement and compare the following sorting algorithms: a. Bubble sort b. Insertion sort c. Selection sort 9.Searching Algorithms: Write programs to implement and compare: a. Linear search b. Binary search (on a sorted array) 10.Combined Application a. Design a simple program that uses multiple data structures .	
10	Text Books: 1. Learning Python, Fourth Edition by Mark Lutz Copyright © 2009 Mark Lutz. Published by O'Reilly Media, Inc. 2. Python Basics: A Practical Introduction to Python 3 Revised and Updated 4th Edition David Amos, Dan Bader, Joanna Jablonski, Fletcher Heisler 3. Data Structures and Algorithms made Easy: Data Structures and Algorithmic Puzzles, Narasimha Karumanchi ,5th Edition 2017	
11	Reference Books: 1. Let Us Python, Yashwant. B. Kanetkar, BPB Publication, 2019 2. Python: The Complete Reference, Martin C. Brown, McGraw Hill, 2018 3. Beginning Python: From Novice to Professional, Magnus Lie Hetland, Apress, 2017 4. A Simplified Approach to Data Structures, Lalit Goyal, Vishal Goyal, Pawan Kumar SPD,1st 2014 5. Problem Solving in Data Structures & Algorithms Using C by Hemant Jain ,1st Edition, BPB Publications, 2018 6. Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, 4th Edition, MIT Press,2022	
12	**Internal Continuous Assessment: 40%**	**Semester End Examination: 60%**
13	**Continuous Evaluation through:** Students are expected to attend each practical and submit the written practical of the previous session. Performing Practical and writeup submission will be continuous internal evaluation. 2.5 marks can be awarded for each practical performance and writeup submission totaling to 50 marks and can be converted to 20 marks.	30 marks practical exam of 2 hours duration
14	**Format of Question Paper: Duration 2 hours. Certified copy of Journal is compulsory to appear for the practical examination** Practical Slip: Q1. From Module 1 13 marks Q2. From Module 2 12marks Q3. Journal and Viva 05 marks	