B.SC. (CBCS) (DATA SCIENCE) SYLLABUS

(With Mathematics & Statistics Combination only)

With effect from the Academic Year: 2024-25 (of First year)

Year	Sem	Paper code	Theory/ Practical	Paper Title	Credits	Work Load (Hours / Week)	Total Marks
I	I	BSDS-101-T	Paper-I	Problem solving and Python Programming	4	4	100
		BSDS-101-P	Practical -I	Problem solving and Python Programming	1	2	25
	II	BSDS-202-T	Paper-II	Data Structures and Algorithms	4	4	100
		BSDS-202-P	Practical -II	Data Structures and Algorithms Lab	1	2	25
II	III	BSDS-303-T	Paper-III	Data Engineering with Python	4	4	100
		BSDS-303-P	Practical-III	Data Engineering with Python (Lab)	1	2	25
		BSDS-303-SE	SEC-2	Information Technology	2	2	50
	IV	BSDS-404-T	Paper-IV	Programming in Java	4	4	100
		BSDS-404-P	Practical-IV	Programming in Java Lab	1	2	25
		BSDS-404-SE	SEC-4	Web Technology	2	2	50
III	V	BSDS-505-T	Paper-V	Machine Learning	4	4	100
		BS-505-P	Practical-5	Machine Learning (Lab)	1	2	25
		BS-505-GE	G E	Basic Statistics	4	4	100
	VI	BS-606-T	Paper-VI	Deep Learning	4	4	100
		BS-606-P	Paper-VI	Deep Learning Lab	1	2	25
		BS-606-O	Optional	Data Science Project	4	4	100

Note:

1. No of credits = No of theory hours for teaching = twice the no. of credits for practical teaching.
2. Skill Enhancement Courses (SEC) are offered for DS students in Sem-III & IV with continuation (offered for who are willing improve the practical skills in conducting of election forecasts / exit poll surveys.
3. Generic Elective (GE) course is offered for *other than this combination Course students.*
4. Optional paper course is offered if statistics students are willing to opt.

DATA SCIENCE SYLLABUS
B.SC. (Data Science) I-YEAR I-SEMESTER
BSDS-101T: Paper-I (Theory): Problem Solving and Python Programming
[4 HPW :: 4 Credits :: 100 Marks (External:80, Internal:20)]

Course Objectives:

The main objective is to teach Computational thinking using Python.

- To know the basics of Programming for solving a problem with writing an algorithm and flow charts.
- To convert an algorithm into a Python program
- To construct Python programs with control structures.
- To structure a Python Program as a set of functions
- To use Python data structures-lists, tuples, dictionaries.
- To do input/output with files in Python.
- To construct Python programs as a set of objects.

UNIT-I

Introduction to Computing and Problem Solving: Fundamentals of Computing – Computing Devices – Identification of Computational Problems – Pseudo Code and Flowcharts – Instructions – Algorithms – Building Blocks of Algorithms.

Introduction to Python Programming: Python Interpreter and Interactive Mode– Variables and Identifiers – Arithmetic Operators – Values and Types – Statements, Reading Input, Print Output, Type Conversions, The type() Function and Is Operator, Dynamic and Strongly Typed Language.

Control Flow Statements: The if, The if…else,The if…elif…else Decision Control Statements, Nested if Statement, The while Loop, The for Loop, The continue and break Statements.

UNIT-II

Functions: Built-In Functions, Commonly Used Modules, Function Definition and Calling the Function, The return Statement and void Function, Scope and Lifetime of Variables, Default Parameters, Keyword Arguments, *args and **kwargs, Command Line Arguments. **Strings**: Creating and Storing Strings, Basic String Operations, Accessing Characters in String by Index Number, String Slicing and Joining, String Methods, Formatting Strings.

UNIT-III

Lists: list operations, list slices, list methods, list loop, mutability, aliasing, cloning lists, list parameters; Tuples: tuple assignment, tuple as return value; Dictionaries: operations and methods; advanced list processing - list comprehension; Illustrative programs: selection sort, insertion sort, merge sort, histogram.

Files and exception: text files, reading and writing files, format operator; command line arguments, errors and exceptions, handling exceptions, modules, packages; Illustrative programs: word count, copy file.

UNIT-IV

Object-Oriented Programming: Classes and Objects, Creating Classes in Python, Creating Objects in Python, The Constructor Method, Classes with Multiple Objects, Class Attributes versus Data Attributes, Encapsulation, Inheritance, Polymorphism.

Functional Programming: Lambda. Iterators, Generators, List Comprehensions

References:

1. Introduction to Python Programming. Gowrishankar S., Veena A. CRC Press, Taylor & Francis Group, 2019
2. Allen B. Downey, ``Think Python: How to Think Like a Computer Scientist'', 2nd edition, Updated for Python 3, Shroff/O'Reilly Publishers, 2016 (http://greenteapress.com/wp/think- python/)
3. Learning To Program With Python. Richard L. Halterman. Copyright © 2011
4. Python for Everybody, Exploring Data Using Python 3. Dr. Charles R. Severance. 2016.

Course Outcomes:

- The main objective of this is to put into practice in computer lab with computational thinking and able to write, compile, run and debug Python programs with perfect usage of variables, conditionals statements, control structures, functions (both recursive and iterative), basic data types as well as compound data structures such as strings, lists, sets, tuples, dictionaries. object-oriented programming.
- able to develop algorithmic solutions to simple computational problems.
- able to Develop and execute simple Python programs.
- Structure a Python program into functions.
- Represent compound data using Python lists, tuples, dictionaries.
- Read and write data from/to files in Python Programs.

Practical Paper-I: Problem Solving and Python Programming (Lab)

[2 HPW :: 1 Credit :: 25 Marks]

List of Practicals:

I. Programs to demonstrate the usage of operators and conditional statements

1. Write a program that takes two integers as command line arguments and prints the sum of two integers.
2. Program to display the information: Your name, Full Address, Mobile Number, College Name, Course Subjects
3. Program to find the largest number among 'n' given numbers.
4. Program that reads the URL of a website as input and displays contents of a webpage.

II. Programs to demonstrate usage of control structures

1. Program to find the sum of all prime numbers between 1 and 1000.
2. Program that reads set of integers and displays first and second largest numbers.
3. Program to print the sum of first 'n' natural numbers.
4. Program to find the product of two matrices.
5. Program to find the roots of a quadratic equation.

III. Programs to demonstrate the usage of Functions and Recursion

6. Write both recursive and non-recursive functions for the following:
   a. To find GCD of two integers
   b. To find the factorial of positive integer
   c. To print Fibonacci Sequence up to given number 'n'
   d. To convert decimal number to Binary equivalent
7. Program with a function that accepts two arguments: a list and a number 'n'. It should display all the numbers in the list that are greater than the given number 'n'.
8. Program with a function to find how many numbers are divisible by 2, 3,4,5,6 and 7 between 1 to 1000.

IV. Programs to demonstrate the usage of String functions

9. Program that accept a string as an argument and return the number of vowels and consonants the string contains.
10. Program that accepts two strings S1, S2, and finds whether they are equal are not.
11. Program to count the number of occurrences of characters in a given string.
12. Program to find whether a given string is palindrome or not.

V. Programs to demonstrate the usage of lists, sets, dictionaries, tuples and files.

13. Program with a function that takes two lists L1 and L2 containing integer numbers as parameters. The return value is a single list containing the pair wise sums of the numbers in L1 and L2.
14. Program to read the lists of numbers as L1, print the lists in reverse order without using reverse function.
15. Write a program that combine lists L1 and L2 into a dictionary.
16. Program to find mean, median, mode for the given set of numbers in a list.
17. Program to find all duplicates in the list.
18. Program to o find all the unique elements of a list.
19. Program to find max and min of a given tuple of integers.
20. Program to find union, intersection, difference, symmetric difference of given two sets.
21. Program to display a list of all unique words in a text file
22. Program to read the content of a text file and display it on the screen line wise with a line number followed by a colon
23. Program to analyze the two text files using set operations
24. Write a program to print each line of a file in reverse order.

VI. Programs to demonstrate the usage of Object Oriented Programming

25. Program to implement the inheritance
26. Program to implement the polymorphism.

VII. Programs to search and sort the numbers

27. Programs to implement Linear search and Binary search
28. Programs to implement Selection sort, Insertion sort

DATA SCIENCE SYLLABUS
B.SC. I YEAR II SEMESTER (CBCS)
PAPER – II : DATA STRUCTURES (Theory)

UNIT – I

Fundamental Concepts: Introduction to Data Structures, Types of Data Structures, Introduction to Algorithm, Pseudo-code, Flow Chart, Analysis of Algorithms. Linear Data Structure Using Arrays: 1-D Arrays, 2-D Arrays, N-D Arrays, Memory Representation and Address Calculation of 1-D, 2-D, N-D Arrays, Concept of Ordered List, String Manipulation, Pros and Cons of Arrays. Stacks: Concept, Primitive Operations, Abstract Data Type, Representation Stacks Using Arrays, Prefix, Infix, Postfix Notations for Arithmetic Expression, Applications of Stacks– Converting Infix Expression to Postfix Expression, Evaluating the Postfix Expression, Checking Well-formed (Nested) Parenthesis, Processing of Function Calls, Reversing a String.

UNIT – II

Recursion: Introduction, Recurrence, Use of Stack in Recursion, Variants of Recursion, Execution of Recursive Calls, Recursive Functions, Iteration versus Recursion. Queues: Concept, Primitive Operations, Abstract Data Type, Representation Queues Using Arrays, Circular Queue, Double-Ended Queue, Applications of Queues. Linked Lists: Introduction, Concept, Terminology, Primitive Operations-creating, inserting, deleting, traversing, Representation of Linked Lists, Linked List Abstract Data Type, Linked List Variants - Singly Linked List, Doubly Linked List, Linear and Circular Linked List, Representation Stacks and Queues Using Linked Singly Lists, Application of Linked List–Garbage Collection.

UNIT – III

Trees: Introduction, Representation of a General Tree, Binary Tree Introduction, Binary Tree Abstract Data Type, Implementation of Binary Trees, Binary Tree Traversals – Preorder, Inorder, Postorder Traversals, Applications of Binary Trees Briefly. Graphs: Introduction, Graph Abstract Data Type, Representation of Graphs, Graph Traversal – Depth-First Search, Breadth-First Search, Spanning Tree – Prim's Algorithm, Kruskal's Algorithm. Hashing: Introduction, Hash Functions, Collision Resolution Strategies.

UNIT – IV

Searching and Sorting: Sequential (Linear) Search, Binary Search, Bubble Sort, Insertion Sort, Selection Sort, Quick Sort, Merge Sort, and Comparison of Sorting Techniques. Heaps: Concept, Implementation, Abstract Data Type, Heap Sort.

References

1. "Computer Algorithms" - Ellis Horowitz, Sartaj Sahni and S. Rajasekaran
2. "Data Structure and Algorithmic Thinking with Python" - Narasimha Karumanchi
3. "Data Structures and Algorithms in Python"- Roberto Tamassia, M. H. Goldwasser, M.T. Goodrich.
4. "Problem Solving in Data Structures & Algorithms Using Python"- Hemant Jain

Practical Paper-II: Data Structures Using Python (Lab)

[2 HPW :: 1 Credit :: 25 Marks]

1. Write programs to implement the following using an array: a) Stack ADT b) Queue ADT.
2. Write a program to convert the given infix expression to postfix expression using stack.
3. Write a program to evaluate a postfix expression using stack.
4. Write a program to ensure the parentheses are nested correctly in an arithmetic expression.
5. Write a program to find following using Recursion a) Factorial of +ve Integer b) n^{th} term of the Fibonacci Sequence (c) GCD of two positive integers
6. Write a program to create a single linked list and write functions to implement the following operations. a) Insert an element at a specified position b) Delete a specified element in the list c) Search for an element and find its position in the list d) Sort the elements in the list ascending order
7. Write a program to create a double linked list and write functions to implement the following operations. a) Insert an element at a specified position b) Delete a specified element in the list c) Search for an element and find its position in the list d) Sort the elements in the list ascending order
8. Write a program to create singular circular linked lists and function to implement the following operations. a) Insert an element at a specified position b) Delete a specified element in the list c) Search for an element and find its position in the list
9. Write programs to implement the following using a single linked list: a) Stack ADT b) Queue ADT. 10 Write a program to implement Binary search technique using Iterative method and Recursive methods.
10. Write a program for sorting the given list numbers in ascending order using the following technique: Bubble sort and Selection sort
11. Write a program for sorting the given list numbers in ascending order using the following technique: Insertion sort and Quicksort
12. Write a program for sorting the given list numbers in ascending order using the following technique: Merge sort and Heapsort
13. Write a program to traverse a binary tree in following way. a) Pre-order b) In-order c)Post-order 15 Write a program to the implementation graph traversals – BFS and DFS.
14. Write a program to find the minimum spanning tree for a weighted graph using a) Prim's Algorithm b) Kruskal's Algorithm.

Note: Write the Pseudo Code, flowcharts and Python program code for the above problems/methods/ algorithms with different possibilities like with and without oops, functions, etc. is mandatory.