CSR101:PYTHON PROGRAMMING

L:3 T:0 P:2 Credits:4

Course Outcomes: Through this course students should be able to

CO1 :: establish Python environment and demonstrate basic programming using core syntax.

CO2 :: execute control structures, functions, recursion, lambdas, and string operations.

CO3 :: apply data structures, OOP concepts like inheritance, and functional programming.

CO4 :: develop modular, maintainable code using modules, packages, and exception handling.

CO5 :: perform efficient numerical and matrix operations using NumPy and SciPy.

CO6 :: analyze and visualize data using Pandas, Matplotlib, and Seaborn.

Unit I

Python Environment Setup and Basics : Installing Python and IDEs (Anaconda, Jupyter, VS Code), Python syntax, variables, data types, operators, expressions, input/output, Module basics, Math module, Representing text, String basics, List, Common data types summary, Type conversions, Binary numbers, String formatting, Using Python shell as a calculator and running simple scripts, Introduction to software development best practices, PEP 8, code readability, If-else statement, Equality and relational operators, Boolean operators and expressions, Order of evaluation, Membership and identity operators

Unit II

Control Flow, Functions, and Problem-Solving : Conditional statements (if-else), loops (for, while), nested loops, Break and continue, Counting, range() function, Function definitions, arguments, return values, recursion basics, lambda function, String slicing, Advanced string formatting, String methods, Splitting and joining strings, String format method, Regular expressions, Algorithmic thinking, writing Python code for common problem-solving patterns

Unit III

Data Structures, Classes, and Inheritance : Lists, list comprehension, tuples, dictionaries, sets: creation, manipulation, mutability, map, filter, reduce, Cartesian product, Itertools, eval, exec, enumerate, zip, copy, Introduction to classes, constructors, methods, attributes, instances, Inheritance concepts, base and derived classes, method overriding, multiple inheritance, Practical OOP examples and use of decorators and generators

Unit IV

Modules and Exception Handling : Handling exceptions using try and except, Multiple exception handlers, Raising exceptions, Exceptions with functions, Using finally to clean up, Custom exception types, Modules, Finding modules, Importing specific names from a module, Executing modules as scripts, Reloading modules, Packages and Standard library, datetime, Writing modular, maintainable, error-resilient code

Unit V

Matrices : Random number generation, Numpy, Indexing and Slicing, Attributes of a Numpy array, Basic mathematical operations, Array Manipulation functions, Matrix decomposition, Sparse Matrices and associated Data Structures, Scipy and vectorization of Code

Unit VI

File Handling, Data Loading, and Visualization : Reading Files, Writing Files, Interacting with file systems, Binary data, Command-line arguments and files, 'with' statement, Comma separated values files, Getting files from the Internet, loading data, selecting, filtering, Visualizing data with Matplotlib and Seaborn, line plots, bar charts, histograms, scatter plots

List of Practicals / Experiments:

List of Practicals

- Let's say I give you a list saved in a variable: a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]. Write one line of Python that takes this list "a" and makes a new list that has only the even elements of this list in it.
- Write a program that asks the user for a long string containing multiple words. Print back to the user the same string, except with the words in backwards order.
- Create a program that will play the "cows and bulls" game with the user.

- Write a program that creates a student class that initializes attributes name, age, and grade while creating an object.
- Write a program that creates a teacher class that initializes parameters name and age while creating an object and inherits the attributes, from its parent class Staff, the attributes' role, department, and salary.
- Write a program that creates a cohort class consisting of an array of students and provides methods for adding students to a cohort and removing students from a cohort.
- Write a program that creates an overload of the + and > operators of the cohort class so that two cohorts can be merged and we can compare if one cohort is larger than the other.
- Create a method in the cohort that returns students of a particular age.
- Write a Python program to calculate the geometric sum of n-1.
- Write a Python program to find the greatest common divisor (gcd) of two integers.
- Finding the factorial of a number.
- Make a two-player Rock-Paper-Scissors game. (Hint: Ask for player plays (using input), compare them, print out a message of congratulations to the winner, and ask if the players want to start a new game)Remember the rules: i.Rock beats scissors ii.Scissors beat paper iii.Paper beats rock
- Develop a program that generates random passwords based on user-specified criteria (length, inclusion of uppercase/lowercase letters, digits, special characters) and evaluates password strength.
- Write a Python script that simulates a basic bank account system, allowing users to deposit, withdraw, check balance, and view transaction history, with input validation and error handling.

Text Books:

1. PROGRAMMING AND PROBLEM SOLVING WITH PYTHON by ASHOK KAMTHANE, AMIT ASHOK KAMTHANE, Tata McGraw Hill, India

References:

1. PYTHON CRASH COURSE: A HANDS-ON, PROJECT-BASED INTRODUCTION TO PROGRAMMING by ERIC MATTHES, NO STARCH PRESS

2. BEGINNING PROGRAMMING WITH PYTHON FOR DUMMIES, 2ND EDITION by JOHN PAUL MUELLER, WILEY

3. FLUENT PYTHON by LUCIANO RAMALHO, O'reilly Media