|  |
| --- |
| 1. Highlight the features of Python and Compare it with 'C' and 'Java' |
| 1. What are Python libraries and how do they enhance the functionality of the Python programming language? Provide an overview of any 4 libraries available in Python. |
| 1. How is the print statement used in Python? Explain the usage of different attributes with the print statement and provide examples for each attribute. |
| 1. Explain the different types of conditional statements in Python. Provide examples for each type, including if, if-else, and if-elif-else statements. Additionally, discuss the importance of indentation in these statements. |
| 1. Explain different data types supported in Python |
| 1. Discuss the concept of conditional statements in Python and their role in controlling the flow of execution based on certain conditions. Provide an overview of the if, elif, and else statements, by giving examples |
| 1. Explain nested loops in Python and also show the usage of 'continue', 'break' and 'pass' in the loop |
| 1. Discuss the different types of loop statements in Python and provide examples for each type, including for and while loops. |
| 1. Discuss about any three Sequence Data types and also give example |
| 1. Explain Bitwise operators in Python with examples |
| 1. Explain the concepts of membership and identity operators in Python with examples. How do they differ, and in what scenarios would you use each? Give examples. |
| 1. Explain the usage of arithmetic, relational, and logical operators in Python with examples. How do these operators function and in what scenarios are they typically used? Provide illustrative code snippets. |
| 1. What is the purpose of itertools. combinations() and combinations\_with\_replacement()? How are they useful in combinatorial problems? |
| 1. Write notes on : i) Byte vs Bytearray ii) Set vs Frozenset iii) List vs Array |
| 1. Explain the concepts of slicing and indexing in Python. Provide examples to demonstrate how they work with lists, strings, and tuples. Additionally, discuss how negative indices and step parameters can be used in slicing. |
| 1. Briefly describe infinite iterators, shortest input sequence, and combinatoric generators from the itertools module with relevant examples. |
| **Unit - 2** |
| 1. Explain the use of map(), zip(), and filter() functions in Python. Write code examples to demonstrate their usage. |
| 1. Differentiate between an iterable and an iterator in Python. Explain the use of iter() and next() functions with a suitable example. |
| 1. What is dictionary unpacking in Python? Show how to pass a dictionary to a function using \*\*kwargs with an example. |
| 1. Discuss the concept of Dictionary unpacking. How do you unpack dictionary into variables and how do you merge dictionary using \*\*? Give example |
| 1. Explain the purpose and functionality of the range() function in Python. Provide examples demonstrating different ways to use the range() function, and discuss its significance in creating sequences of numbers for iteration and loop control in Python programs. |
| 1. Discuss the differences between lists and tuples in Python, highlighting their respective characteristics and typical use cases. Provide examples demonstrating the creation and manipulation of both lists and tuples. |
| 1. Explain the concept of user-defined functions in Python, their structure, and their importance in programming. Provide an example illustrating the definition and invocation of a user-defined function, and highlight the advantages of incorporating functions into Python code. |
| 1. Write Python user defined function to count the total number of vowels, consonants and blanks in a String (user input). |
| 1. Explain the usage and functionality of the following control statements and keywords in Python: pass, continue, break, assert, and return. Provide examples to illustrate how each of them works and discuss their importance in Python programming. |
| 1. Explain the functionalities of the following methods in Python dictionaries: update(), in operator, get(), pop(), and clear(). Provide explanations and examples demonstrating how each method is used for updating, checking for key existence, retrieving values, removing key-value pairs, and clearing the dictionary, respectively. |
| 1. Create an empty list. Input ‘n’ items. Count total number of Odd, Even, and zero items in the list and display the contents of new lists. |
| 1. Discuss the various methods available for working with dictionaries in Python, including methods for adding, accessing, modifying, and removing key-value pairs. Provide examples demonstrating the usage of these methods and discuss their importance in dictionary manipulation. |
| 1. What is name localization, and how does it affect variable access and performance within functions? Illustrate your answer with suitable examples. |
| 1. Explain the use of join() and split() string methods with examples. Describe why strings are immutable |
| 1. Define anonymous functions in Python. How do they differ from regular functions? Explain with examples how lambda functions can be used with higher-order functions like map(), filter(), and sorted(). |
| 1. Discuss the similarities and differences among Array, List and Tuples in Python and also highlight their advantages and disadvantages |
| 1. Explain the terms Explain the concepts of encapsulation, abstraction, and inheritance in object-oriented programming (OOP) with reference to Python. Provide definitions and examples illustrating how each concept is used in OOP and discuss their significance. |
| 1. Write a syntax for Handling Exceptions with example i)try ii)try-except iii)try-except-else iv)try-except-else-finally |
| 1. Explain the steps involved in Exception handling by giving examples |
| 1. What are exceptions in Python? How do you raise and handle exceptions? Write a program to demonstrate. Write a custom exception class NegativeNumberError and raise it when a user enters a negative number. |
| 1. Define what a constructor is in object-oriented programming and explain its purpose. Write a Python program demonstrating the creation of a Student class with a constructor that accepts more than one parameter. Provide an explanation of the program and discuss the significance of constructors in class instantiation. |
| 1. Explain the concepts of constructors, inheritance, and method overriding in object-oriented programming (OOP) with reference to Python. Provide definitions and examples illustrating how each concept is implemented in Python, and discuss their significance in building modular and reusable code. |
| 1. Explain the purpose of the \_\_init\_\_() method in Python classes and how it serves as a constructor. Provide examples demonstrating how the \_\_init\_\_() method is defined and used to initialize object attributes during object creation. |
| 1. Create Employee Class with relevant details (Eno, Name, Designation, Salary) and write 3 methods to Input values, Calculate Gross and Net salary and display the details. (You can assume appropriate percentage for DA, HRA) |
| 1. Explain the different types of Inheritance in Python. Demonstrate the usage of Single and multi level inheritance with the help of code |
| 1. Write a class Employee with attributes name, designation, department and salary, and a method to Calculate Gross salary and display details. Create instances and demonstrate object initialization using \_\_init\_\_(). |
| 1. Explain how to access base class members from a derived class using super(). Illustrate with an example involving method and constructor overriding. |
| 1. Explain inheritance in Python with an example. Differentiate between single and multiple inheritance. Write a program where a child class inherits from two parent classes and overrides a method. |
| 1. Define Test-Driven Development. What are the key steps involved in TDD? |
| 1. Explain the benefits and drawbacks of using TDD in software development. Write a simple TDD example in Python for a function that returns the square of a number |
| 1. What is Method Resolution Order (MRO) in Python? How does it help in multiple inheritance? Explain with an example. |
| 1. What are assertions in Python? How are they used for debugging and testing? Write a Python function that uses assertions to check if a given number is positive, and explain the output when the assertion fails. |
| 1. What is a masked array in NumPy? Explain its purpose with an example. |
| 1. Explain the concepts of Fancy Indexing and Broadcasting in NumPy. How does Fancy Indexing differ from Slicing? How does Broadcasting work when performing operations on arrays of different shapes? Illustrate both with suitable examples. |
| 1. Describe and demonstrate the key indexing techniques in NumPy arrays: basic indexing, slicing, Boolean indexing, and fancy indexing. How are they useful in data manipulation? |
| 1. Explain the different ways to create NumPy arrays. How do data types and shape attributes influence array behavior? Write code examples to demonstrate. |
| 1. Discuss how NumPy supports statistical computations. Write a Python program using NumPy to compute mean, median, standard deviation, and variance for a given dataset. |
| 1. Explain the concept of masked arrays in NumPy. Why are they useful, and how do they differ from normal arrays? Give an example to show how missing or invalid values are handled. |
| 1. Explain the different functions in NumPy used to generate random numbers. Compare rand(), randn(), randint(), and random(). Give examples for each. |
| 1. With reference to arrays explain the following methods: append(),insert(),pop(), reverse() |
| 1. Explain linspace(), arange(), eye() functions in Python by giving suitable examples |
| 1. Explain the concept of set operations in Python, including union, intersection, difference, and symmetric difference. Provide examples demonstrating how these operations are performed using built-in methods and operators. |
| 1. Discuss the concept of sets in Python and their significance in programming. Explain how sets differ from other data structures, such as lists and dictionaries, and highlight their key characteristics. |
| 1. Explain the usage of the following methods : linspace(), arange(), zeros() and ones() |
| 1. How can NumPy be used to compute basic descriptive statistics like mean, median, standard deviation, and variance? Write a Python program that computes these values for a given NumPy array and explain the output. |
| 1. Explain the basic array operations in NumPy with suitable examples. Your answer should include: i) creating 1 D and 2D arrays ii) Reshaping arrays iii) Transposing iv) Flattening |
| 1. Explain the functionalities of the find(), rfind(), index(), and rindex() methods in Python strings. Provide examples demonstrating their usage and discuss the differences between these methods, including how they handle substring searching and index retrieval. |
| 1. Discuss the functionalities of the following string methods in Python: count(), split(), join(), and replace(). Provide examples to illustrate how each method works and its significance in string manipulation. |
| 1. Explain the purpose and functionality of the seek() and tell() methods in file handling in Python. Provide examples illustrating how these methods are used to navigate within a file and determine the current file pointer position. Additionally, discuss the significance of these methods in file manipulation tasks. |
| 1. Discuss the functionalities of the following file handling methods in Python: read(), readlines(), write(), and writelines(). Provide explanations and examples demonstrating how each method is used for reading from and writing to files, and discuss their significance in file manipulation tasks. |
| 1. Write a python program to create a text file, insert n lines of text and display its contents in a neat format. Also count the number of lines, words and characters in the text file. |
| 1. Provide examples demonstrating how to open, read, and write to text files in Python using built-in functions and methods. Discuss common file modes and their implications in file handling operations. |
| 1. What is PyTable? Explain its purpose key features |
| 1. Explain the use case of PyTables and also show how to create PyTable, store data and read data |
| 1. How do you create Pandas Series and Data Frame? Explain with the help of examples |
| 1. With reference to Pandas explain i) info() ii) describe iii) head() and iv) columns() |
| 1. What are missing values in a dataset? How can you identify and fill missing values using Pandas in Python? Explain any four methods to handle missing values. |
| 1. Explain any four aggregate functions commonly used with the GROUP BY clause. Illustrate each with an example using a Python dictionary of sales data, where you group the data by salesman. |
| 1. What are the different types of Joins? Demonstrate the types of Joins by creating two dictionaries conating Employee data |
| 1. How do you achieve the following in Pandas i) Renaming columns, ii) dropping rows & coulmns iii) Adding new column iv) Sorting data frame. Illustrate with examples |
| 1. Describe the steps to visualize a DataFrame using pandas built-in plotting capabilities. How does it internally use matplotlib? Provide a use case showing line and bar plots. |
| 1. Define data visualization. Discuss the purpose of data visualization in data analysis and explain any three commonly used types of graphs (such as bar chart, line graph, scatter plot) with suitable examples. |
| 1. Write a pandas program to read a CSV file and filter data based on multiple conditions (e.g., Salesman = "Ravish" and Qty > 5, city Name like ). Display only selected columns. |
| 1. What is Matplotlib? Create a Python dictionary containing sales details of salesmen, convert it into a Pandas DataFrame, group the data by total sales per salesman, and then draw a bar graph and a pie chart to visualize the total sales of each salesman. |