filoger

فناپ کمپس
FANAP CAMPUS

Silver Exercises

Python

6620

CV - 1402

OPERATORS

1. **Arithmetic Operators: (80)**

   Write a program that takes two numbers as input and calculates their sum, difference, product, and quotient.

2. **Comparison Operators: (80)**

   Write a program that takes two numbers as input and determines whether the first number is greater than, less than, or equal to the second number.

OPERATORS

3. Assignment Operators: (80)

Write a program that takes an initial value and applies different assignment operators (e.g., +=, -=, *=) to modify the value and print the result.

4. Comparison Operators: (80)

Write a program that takes a person's age as input and determines whether they are eligible to vote (consider the legal voting age is 18).

CONTROL FLOW AND ITERATION

5. If-else Statements: (80)

Write a program that takes a number as input and checks whether it is positive, negative, or zero.

6. For Loops: (80)

Write a program that takes a string as input and prints each character on a separate line using a for loop.

CONTROL FLOW AND ITERATION

7. While Loops: (80)

Write a program that takes a number as input and prints all the even numbers up to that number using a while loop.

8. Control Flow with Lists and Conditions: (130)

Write a program that takes a list of words and prints the length of each word that starts with the letter 'a'.

LIST

9. Basic List Operations: (80)

Create two lists of numbers and concatenate them into a single list.

10. Accessing List Elements: (90)

Create a list of numbers and print the sum of the first and last elements.

LIST

11. Modifying Lists: (80)

Create a list of numbers and change the value of the third element.

12. Slicing and Sublists: (80)

Create a list of strings and print a sublist containing the last two elements.

TUPLE

13. Basic Tuple Operations: (100)

Create a tuple of strings and print the elements in reverse order.

14. Accessing Tuple Elements: (100)

Create a tuple of numbers and calculate the sum of the first and last elements.

TUPLE

15. Modifying Tuples: (100)

Create a tuple of numbers and change the value of the third element (Note: Tuples are immutable, so you'll need to convert it to a list, modify it, and then convert it back to a tuple).

16. Slicing and Subtuples: (100)

Create a tuple of strings and print a subtuple containing the last two elements.

DICTIONAY

17. Basic Dictionary Operations: (100)

Create a dictionary representing a product with keys for "name", "price", and "quantity". Print the dictionary.

18. Accessing Dictionary Elements: (100)

Create a dictionary representing a person and print the value associated with the "name" key.

DICTIONAY

19. Modifying Dictionaries: (100)

Create a dictionary representing a product and update the value associated with the "quantity" key.

20. Adding and Removing Dictionary Entries: (100)

Create an empty dictionary and add key-value pairs for "name" and "age" representing a person.

SET

21. Creating and Modifying Sets: (100)

Create a dictionary representing a product and update the value associated with the "quantity" key.

22. Set Operations: (100)

Create two sets of numbers and perform the union operation to combine them into a single set.

SET

23. Removing Set Elements: (200)

a) Create a set of numbers and remove a specific element using the remove() method.

b) Create a set of strings and use the discard() method to remove an element that may or may not be present in the set.

STRING

24. String Operations: (100)

Create a string and repeat it multiple times using the multiplication operator.

25. Accessing String Characters: (100)

Create a string and print the last character.

STRING

26. String Length and Slicing: (100)

Create a string and print a substring using slicing.

27. Modifying Strings: (100)

Create a string and convert it to uppercase using the upper() method.

FUNCTIONS

28. Function Basics: (100)

Write a function called calculate_area() that takes the length and width of a rectangle as arguments and returns its area.

29. Function Parameters and Arguments: (150)

Write a function called add_numbers() that takes two numbers as arguments and returns their sum.

FUNCTIONS

30. Default Parameters: (150)

Write a function called calculate_power() that takes a number and an exponent as arguments and calculates the power of the number. If no exponent is provided, it should default to 2.

31. Lambda Functions: (150)

Write a lambda function to check if a number is divisible by 2.

ITERATION

32. Sum of List: (150)

Write a function that takes a list of numbers as input and returns the sum of all the numbers using a loop.

33. Factorial: (200)

Write a function that takes a number as input and calculates its factorial using a loop. For example, the factorial of 5 (5!) is 5 * 4 * 3 * 2 * 1 = 120.

ITERATION

34. Counting Vowels: (150)

Write a function that takes a string as input and counts the number of vowels (a, e, i, o, u) in the string using a loop.

35. Fibonacci Sequence: (200)

Write a function that takes a number as input and prints the Fibonacci sequence up to that number using a loop. The Fibonacci sequence is a series of numbers where each number is the sum of the two preceding ones (e.g., 0, 1, 1, 2, 3, 5, 8, ...).

RECURSION

36. Factorial: (150)

Write a recursive function that takes a number as input and calculates its factorial. Hint: The factorial of a number n can be calculated as n * factorial(n-1).

37. Fibonacci Sequence: (200)

Write a recursive function that takes a number as input and returns the nth number in the Fibonacci sequence. Hint: The nth number can be calculated as fibonacci(n-1) + fibonacci(n-2).

EXCEPTION HANDLING

38. Divide by Zero: (150)

Write a program that takes two numbers as input and divides the first number by the second number. Handle the ZeroDivisionError exception if the second number is 0 and print an error message.

39. File Reading: (Search!)(150)

Write a program that reads the contents of a file specified by the user. Handle the FileNotFoundError exception and print an error message if the file does not exist.

EXCEPTION HANDLING

40. Input Validation: (150)

Write a program that asks the user to enter a positive integer. Use a while loop and exception handling to ensure that the user enters a valid integer. Handle the ValueError exception and prompt the user to enter a valid input.

41. Index Error: (150)

Write a program that creates a list of numbers. Handle the IndexError exception and print an error message if the user tries to access an index that is out of range.

NUMPY

42. Array Creation: (100)

Create a 2-dimensional NumPy array with shape (3, 3) filled with zeros

43. Array Operations: (100)

Create two NumPy arrays of the same shape and perform element-wise addition.

NUMPY

44. Array Indexing and Slicing: (100)

Create a 2-dimensional NumPy array and print a 2x2 subarray.

45. Array Shape and Reshaping: (100)

Create a 1-dimensional NumPy array and reshape it into a 2x3 array.

NUMPY

46. Array Aggregation: (150)

Create a 2-dimensional NumPy array and calculate the mean of each column.

47. Array Broadcasting: (150)

Create a NumPy array and multiply it element-wise with a 1-dimensional array.

PANDAS (USE DESIRED DATASET)

48. DataFrame Creation:(200)

Create a pandas DataFrame from a dictionary, where the keys represent column names and the values represent the data.

49. Data Exploration: (200)

Load a dataset into a desired DataFrame and print the first few rows using the head() method.

PANDAS (USE DESIRED DATASET)

50. Data Selection and Indexing: (150)

Select a specific column from the DataFrame using square brackets notation.

51. Data Manipulation: (150)

Sort the DataFrame based on a specific column

PANDAS (USE DESIRED DATASET)

52. Missing Data Handling: (150)

Check for missing values in the DataFrame using the isnull() method.

53. Grouping and Aggregation: (200)

Group the DataFrame by a specific column and calculate the mean, sum, or count of another column.