1) Arithmetic Operations Code Report

1. Code Summary

This code illustrates fundamental arithmetic operations in Python. It prompts the user to enter two integer values and performs the following operations:

* Addition: Sums the two numbers.
* Subtraction: Computes the difference by subtracting the second number from the first.
* Multiplication: Multiplies the two numbers together.
* Division: Divides the first number by the second.
* Modulus: Determines the remainder when the first number is divided by the second.
* Exponentiation: Raises the first number to the power of the second.

2. Approach

1. The user is asked to enter two numbers via the input() function.
2. These inputs are converted to integers using int() to facilitate arithmetic calculations.
3. Each arithmetic operation is performed and displayed using the print() function, ensuring that output is clear for each computation.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Report on Comparison Operators Code

1. Code Summary

This code demonstrates the implementation of comparison operators in Python, allowing the user to compare two integers obtained from user input. The program analyzes the relationship between the two numbers and provides the following comparisons:

* Checking Equality (==): Verifies whether both numbers are identical.
* Checking If Greater Than (>): Determines if the first number is greater than the second.
* Checking If Less Than (<): Assesses if the first number is less than the second, establishing that the two numbers are not the same.

2. Approach

The program follows this sequence:

1. User Input: The program prompts the user to input two numbers through the input() function.
2. Data Conversion: The inputs are converted to integers to facilitate numerical comparisons.
3. Comparison Logic: The program employs a series of if-else statements to evaluate the relationship between the two numbers:
   * Equality Check: If both numerical values are equal, the program outputs a message indicating that they are equal.
   * Greater Than Check: If the first number is larger than the second, it asserts that the first number is greater.
   * Less Than Check: In all other cases, the program concludes that the second number is larger.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3) String Manipulation in Computer Programming Tasks Report

1. Code in Short

The given code is quite simple and implements some string manipulation functions in the Python language. It helps to illustrate how to demonstrate the following actions:

Measure the number of characters making up a string.

Access the first and the last characters of the string.

In some appraisal, reverse sequence of the string.

Change to real world the case of a string both to upper and lower and vice versa.

1. Approach

The first string is taken as an input from the user as a value to a single prompt.

It displays size of the string object considering its properties which is measured by len ().

The first and The last characters of the string are reached by the specifying the index (a[0], a[len(a)-1]).

The sequence of the string is reversed by slicing (a[::– 1]) – , however this is not shown in the output.

To determine whether the string has been transformed into upper-case letters, you may use the command: The is upper () – decision is provided. If Yes, case lowers the string. If No, case converts the string to upper case.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4) Report on String Formatting Code

1. Code Overview

This code provides an example of how to take user input and format it into a message. For example, the program takes the name and age of the user and presents this information within a sentence format.

1. Approach

The user is asked to provide his or her name and age through the use of the input() command.

The name is kept as a string while age gets changed into an integer using int.

Name and age is formatted and displayed using print with a constructed sentence incorporating the given details.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5) Report on Client Input and String Organizing Code  
1. Code Overview  
This Python code illustrates how to assemble client input (title and age) and organize that input into a organized yield message. The objective of the program is to:  
  
Accept a user's title and age as input.  
Format and show these points of interest in a sentence.  
2. Approach  
The client is provoked to enter their title utilizing the input() function.  
Similarly, the client is inquired to enter their age, which is changed over from a string to an numbers utilizing the int() function.  
The program at that point groups and prints a sentence that incorporates the title and age utilizing the print() work, viably combining inactive content and user-provided factors.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6) Report on List Operations Code  
1. Code Overview  
This code handles fundamental list operations in Python, such as taking input to populate a list, calculating the whole of the list components, and recognizing the greatest and least values. The most operations include:  
  
Inputting values into a list.  
Sorting the list.  
Finding the most extreme and least values.  
Calculating the entirety of the list elements.  
2. Approach  
An purge list (li) is initialized beside a variable whole set to 0.  
A for circle runs six times, inquiring the client to input values that are added to the list utilizing append().  
The program computes the whole of the list components as they are inputted.  
After collecting all values, the list is sorted utilizing sort().  
The most extreme value is gotten to from the final record (li[5]), and the minimum esteem is from the primary list (li[0]).  
The last whole of all components within the list is printed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7) 7)Report on List Adjustment Code  
1. Code Overview  
This Python code permits the client to alter a predefined list of natural products (li = ["apple", "orange", "grape"]) through different operations:  
  
Change: Embed a unused thing at a particular position.  
Remove: Evacuate an thing from the list.  
Append: Include a modern thing to the conclusion of the list.  
The program presents a menu of choices and forms the user's choice to perform the comparing operation on the list.  
  
2. Approach  
The list li is initialized with three things: "apple", "orange", and "grape".  
The program prints the current list and gives a menu of choices:  
c for changing (embeddings a unused thing at a indicated position).  
r for expelling an item.  
a for adding a unused thing to the conclusion of the list.  
The client chooses an option:  
Change: The client inputs the position where they need to embed a modern natural product and the title of the natural product. The program embeds the unused natural product utilizing insert().  
Remove: The client inputs the thing they wish to evacuate, and the program employments expel() to erase it.  
Append: The client inputs the thing to include, and the program adds it to the conclusion of the list utilizing append().  
The list is overhauled and printed after each operation.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8) Report on Client Input and List Sorting Code  
1. Code Overview  
This Python code illustrates how to gather a arrangement of numbers values from client input, store them in a list, and sort that list in both climbing and plummeting arrange. The essential objective of the program is to:  
  
Accept six numbers values from the user.  
Format and show the first list, the sorted list in rising arrange, and the sorted list in slipping order.  
2. Approach  
List Initialization: The program starts by initializing an purge list to store client input.  
  
User Input:  
  
The client is provoked to enter numbers values utilizing the input() work interior a circle that runs six times.  
Each entered esteem is changed over to an numbers and added to the list.  
Sorting and Displaying:  
  
The program at that point sorts the list in rising arrange utilizing the sort() method.  
It prints the initial list and the sorted list in climbing order.  
Finally, the program inverts the list utilizing the turn around() strategy to show it in plummeting arrange.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

9) Report on Client Input and List Sorting Code  
1. Code Overview  
This Python code illustrates how to gather a arrangement of numbers values from client input, store them in a list, and sort that list in both climbing and plummeting arrange. The essential objective of the program is to:  
  
Accept six numbers values from the user.  
Format and show the first list, the sorted list in rising arrange, and the sorted list in slipping order.  
2. Approach  
List Initialization: The program starts by initializing an purge list to store client input.  
  
User Input:  
  
The client is provoked to enter numbers values utilizing the input() work interior a circle that runs six times.  
Each entered esteem is changed over to an numbers and added to the list.  
Sorting and Displaying:  
  
The program at that point sorts the list in rising arrange utilizing the sort() method.  
It prints the initial list and the sorted list in climbing order.  
Finally, the program inverts the list utilizing the turn around() strategy to show it in plummeting arrange.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

10) Report on List Control and Component Printing Code  
1. Code Overview  
This Python code illustrates how to control a list of integrability by printing the primary five components and after that switching the list to print the final five components. The key objectives of the program are to:  
  
Print and store the primary five components from a predefined list.  
Reverse the list and print the primary five components from the switched list.  
2. Approach  
List Initialization: The program starts with a predefined list of integrability from 1 to 10, put away within the variable numbers.  
  
Printing the Primary Five Elements:  
  
A for circle repeats through the primary five records (0 to 4) of the numbers list.  
Each element is printed on the same line, and at the same time, those components are added to a unused list li.  
After the circle, the program prints the substance of li, appearing the primary five elements.  
Reversing the List:  
  
The invert() strategy is called on the numbers list to invert its order.  
Printing the Primary Five Components of the Switched List:  
  
Another for circle iterates through the primary five lists of the now-reversed list.  
Again, each component is printed on the same line, and these components are added to li.  
Finally, the program prints the upgraded substance of li, which presently incorporates both the primary five and the final five components.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

11) Report on Understudy Score Collection and Normal Calculation Code  
1. Code Overview  
This Python code is outlined to gather the names and scores of different understudies, calculate the normal score for each understudy based on their scores in three subjects, and show the comes about. The essential objectives of the program are to:  
  
Accept a indicated number of understudy names.  
Gather scores for three subjects from each student.  
Calculate and show the normal score for each student.  
2. Approach  
User Input for Number of Students:  
  
The program starts by inciting the user to input the overall number of understudies (n).  
List Initialization:  
  
An purge list named understudy is initialized to store the names of the students.  
Student Data Collection:  
  
A for circle repeats from 1 to n (comprehensive) to gather each student's title and their scores.  
Inside this loop:  
The client is provoked to enter a student's name, which is added to the understudy list.  
A variable whole is initialized to for calculating the overall scores (adjusted from 03, as this may cause perplexity in Python).  
A settled circle repeats over three subjects (from 1 to 3), provoking the client to enter the scores for each subject.  
The entered score is included to sum.  
After collecting all three scores, the normal is calculated by isolating whole by 3.  
Output the Average:  
  
After calculating the normal score for each understudy, the program prints the normal score along side the understudy list.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_THE END\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_