Table of Contents

Question No 1

Title:

Currency Denomination Breakdown Algorithm

Discussion of the Solution:

The problem at hand revolves around designing an algorithm that accepts a monetary input and breaks it down into the smallest possible number of currency notes. The currency notes we are considering are 100, 50, 20, 10, and 5 units.

The solution is a Python script that executes this task efficiently. It makes use of the integer division (//) and modulus (%) operations to determine the number of each type of note that should be dispensed.

The money_notes function loops through each available note denomination starting from the highest. It uses integer division to find out how many of each note can be given without exceeding the money_input. This number is stored in the Indexes_for_telling_strength_of_each_notes list. The remainder of the division operation (money_input %= available_note_in_notes_you_have) updates the money_input for the next smaller note denomination. This process continues until we exhaust all the note denominations.

Self-diagnosis and evaluation:

The algorithm is relatively straightforward, efficient, and accomplishes the task as expected. It uses a greedy approach, trying to use the largest notes possible before moving to smaller ones. However, the solution assumes the availability of all note denominations in unlimited quantities, which might not always be true in a real-world scenario.

Test evidence:

To validate the functionality, let's consider an example where the input money is 365 units. According to the algorithm, it will be broken down into:

3 notes of 100 units, 1 note of 50 units, 0 notes of 20 units, 1 note of 10 units, and 1 note of 5 units.

Therefore, the output will be [3, 1, 0, 1, 1].

```
------------------------------------------------------------------------
Enter your money :  365
------------------------------------------------------------------------
you have following available notes  :  [100, 50, 20, 10, 5]
------------------------------------------------------------------------
length of your available notes list is :  5
------------------------------------------------------------------------
available_notes_of_each_note :  [3, 1, 0, 1, 1]
------------------------------------------------------------------------
```

File names:

Python source code file: Task1.ipynb

Question No 2

Title:

Integer Comparison Function

Discussion of the Solution:

The provided Python script takes two integers as inputs from the user and checks for specific conditions. The checking_numbers function evaluates the following cases:

- If the two integers are equal, it returns True.
- If the sum of the two integers equals 9, it returns False.
- If the absolute difference between the two integers equals 9, it returns True.
- If none of the above conditions are met, it returns False.
- The function uses the built-in Python function abs() to calculate the absolute difference, ensuring that it will handle cases where integer_1 is less than integer_2 correctly.

Self-diagnosis and Evaluation:

The function seems to work well according to the given specification, correctly identifying and handling the required conditions. It should be noted that the function's behavior might be slightly confusing due to its return value when the sum of the two integers is 9 - this case returns False, which is counterintuitive when considering the other conditions.

Test evidence:

Given the integers 4 and 5:

They are not equal, the sum is not 9, and their difference is not 9, so the function returns False.

```
Enter First integer :  4
Enter Second integer :  5
1]: False
```

File names:
Python source code: Task2.ipynb

Question No 3

Title:

List Evaluation Function

Discussion of the Solution:

The provided Python script prompts the user to input integers, which are subsequently appended to a list. The script then checks whether the length of the list is 9 and if the fifth element in the list occurs three times.

The function checking_credentials() handles the core logic of the script. It uses Python's built-in len() function to check if the list's length is 9. Then, it uses the count() method to check if the fifth element (indexed as 4, as Python uses zero-based indexing) occurs three times in the list. If both conditions are satisfied, the function returns True; otherwise, it returns False.

Self-diagnosis and evaluation:

The function appears to work as expected based on the provided specification, successfully identifying the required conditions and responding accordingly. However, it should be noted that the function could be enhanced by making it more modular and flexible, for example, by allowing the list and the conditions to be parameters rather than hardcoding them into the function.

Test evidence:

Given the list of integers [1, 2, 3, 4, 5, 5, 5, 8, 9]:
The length of the list is 9, and the fifth element (5) occurs three times, so the function returns True.

```
Enter integers to convert into list :  1
Enter integers to convert into list :  2
Enter integers to convert into list :  3
Enter integers to convert into list :  4
Enter integers to convert into list :  5
Enter integers to convert into list :  5
Enter integers to convert into list :  5
Enter integers to convert into list :  8
Enter integers to convert into list :  9
--------------------------------------------------------------------------------
LIST_OF_INTEGERS :  ['1', '2', '3', '4', '5', '5', '5', '8', '9']
--------------------------------------------------------------------------------
LENGTH_OF_LIST_OF_INTEGERS :  9
--------------------------------------------------------------------------------
]:  True
```

File names:

Python source code: Task3.ipynb

Question No 4

Title:

Basic Calculator Using Tkinter

Discussion of the Solution:

This Python script uses Tkinter, a standard Python interface to the Tk GUI toolkit, to create a simple calculator GUI. The calculator can perform basic arithmetic operations: addition, subtraction, multiplication, and division.

The script defines four functions addnumbers(), subtractnumbers(), multiplynumbers(), and dividenumbers(). These functions are associated with their respective buttons on the GUI. When the user clicks a button, the corresponding function is triggered. The function retrieves the inputs from the two input fields, performs the requested operation, and displays the result on the screen.

The functions handle invalid input using a try/except block. If the input is not a valid integer, a messagebox pops up to inform the user of the error.

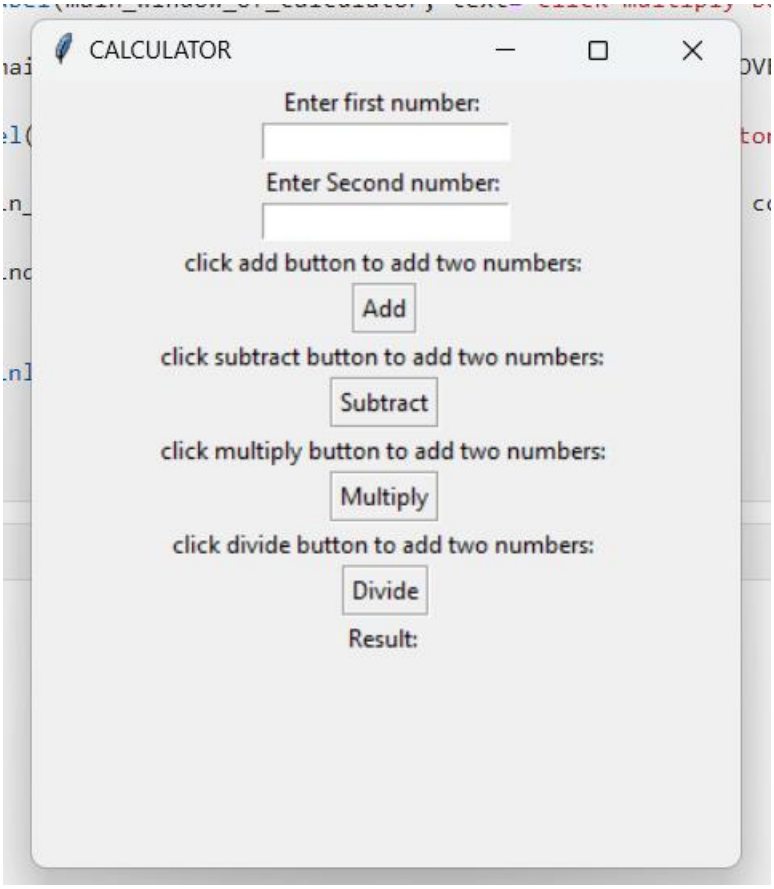
In the Tkinter GUI, Labels, Entry fields, and Buttons are created for the user to interact with. Each button is associated with a function that executes the corresponding mathematical operation.

Self-diagnosis and evaluation:

The script appears to perform well according to the requirements provided, correctly identifying and handling the various arithmetic operations. However, there is room for improvement, as it currently doesn't handle the case of division by zero, which would lead to a ZeroDivisionError.

Test evidence:

Testing this program involves running the script and interacting with the GUI. By inputting various pairs of integers and selecting different operations, we can observe whether the program correctly performs the operations and returns the expected results. An invalid input, such as a non-numeric string, should cause the program to display an error message.



File names:

Python source code: Task4.ipynb