Reflections

1) Which of the problems was the most challenging for you? What specific aspect puzzled you and how were you able to solve it (help from colleagues, notes, google search, office hours, generative AI, etc).

   The question 1-5) was one of the challenging ones. That question required application of various string methods. I had to refer to Geeks for Geeks and AI to learn all the required methods. These include:

   - isalnum() - checks if string contains only alphanumeric characters, eg. 44,

     WHAT, Ro33
   - isdigit() - checks if string contains only digits
   - isalpha() - checks if string contains only alphabetic characters
   - isupper() - checks if all alphabetic characters are uppercase
   - islower() - checks if all alphabetic characters are lowercase

   In my code for the same question I used a nested if statements to check for isalpha() -> isupper() and islower(). Reason being, in order for a string to be isupper/ islower it has to be a string first.

   I found question 6) to be challenging too. Since I used try and except logic to check for error handling. I had to refer to class notes and Geeks for Geeks to 1) raise error inside a condition using the raise function 2) how to use infinite loop with try and expect statements. This entire code can be rewritten using if and else statements too.

   Another difficult problem was the intermediate Ex.3. The most difficult part of that exercise was to reverse the maiden name, using slicing. It required me to understand the algorithm for reverse slicing, which can be found on the page below. I had to use AI in order to address my knowledge gaps. I have learned quite a lot about how to address slicing through -ve steps perspective. On top of it, I made it a point to generalize my code for maiden names which have a length greater than 5. This could have be avoided by hand coding the start, end and step variables. You can see the algorithm used for the same on next page.

Python - lab3

Intermediate Exercises

Q3 Task: Printing the last 3 letters of the family name in reverse order?

Index:	0	1	2	3	4
LNAME:	A	H	U	J	A
Index	-5	-4	-3	-2	-1

-5 → - len(lname)

Slicing Concept = List[start : end : step]

If step = 1 (forward)
→ start = 0
→ end = Something +ve, but excluded
eg. list[1: 3:1]
↳ HU

If step = -1 (backwards)
→ start = -1
→ end = something -ve, but excluded
eg. list[:-4:-1]
↳ AJU, Reversed
list[: :-1]
↳ AJUHA

Sol^n for Q3: Print last 3 alphabets from the family name.

lname[:-4:-1]

-4 → This is nothing but -len(lname)+1

Just to make the code more generic.

```
-5 -4 -3 -2 -1
A  H  U  J  A
```

A ↳ start = -1

H → Go till -4, but exclude that.

Note:

When taking step as negative, avoid using +ve Index for start & step.

list[:-4:-1] = list[:1:-1]

1 → Nothing but +ve Index counterpart of -4

It's much more intitutive to use -ve Indexes.

The most challenging problem was the Advanced Ex. 1. In this exercise, we had to make sure all the user input error have been handled correctly. Another point to note was that in the sample output for stock names, we can see that we just have to check that “name must start with a letter”. My interpretation of this was that we can accept alphanumeric but the first character should be a letter. Therefore, I used the if and else logic to check for the same (using name[0].isalpha())in a infinite while loop. For asset value we just had to make sure the input is digit, alphanumeric is not accepted. While solving for values the same way as names, I got an error. And after debugging the error I understood that the isdigit() works with string inputs. It wouldn’t work if we take the value as a floa t input. Therefore, I improvised my code and make sure the checking condition compares as the string and if true we convert the string datatype to float datatype and then append into the list. You can find detailed comments in my code, explaining the workings step by step.

2) Describe how you used string methods in each of the intermediate and advanced problems (if you didn't use any in a problem, please state that)?

Program	**Description**	**String Methods Used**	**Why**
Basic Ques1-5	Analyze user string for type (alphanumeric, digit, lowercase, etc.)	**.isalnum(), .isalpha(), .isdigit(), .isupper(), .islower()**	To test and classify the content of the input string.
Basic Ques8-9	Strip dollar sign, divide amount by 2	**.strip() or slicing [1:]**	To remove the dollar symbol ($) from the input string.
Basic Ques10	Add "Dr. " prefix to a name	**String concatenation ('Dr. ' + name)**	To build a new string by combining a fixed prefix with user input
Intermediate Ques3	Make strip lower() and capitalize() first letter	**.lower() and .capitalize()**	To make the input in lower case and capitalize to print string with capital letter.
Advanced Ques1	Check is input .isalpha() or .isdigit()	**.isalpha() and .isdigit()**	Check if the first letter of stock name is alphabet and stock value is digit.

I also used string function is question 6 and 7. The method used in those 2 question are a repetition of the above methods.

3) The advanced problems did not require the use of mainline functions with value returning functions. Describe how you would use such a structure in one of the advanced problems (no need to write code, just identify what the function would calculate and what it would return).

   I have already written a function for Advanced ques 1, its not value returning though. If I keep the same code, since I have multiple values that have to be returned in order to be printed in the mainline function. Using the return command in the helper function we can return as many variables as we want to use, but make sure that we assign those returned values to another variable in the in mainline.

   Lets take the example of Ques 1, We can change the same code to a mainline driver that gathers user input and validates the input. Post validation it prints results from a value-returning helper (portfolio_worth_and_avg) that does all the calculations. We can return the portfolio worth and average using this code "return portfolio_worth, avg". Like said, We have to make sure to save these in some other variable in the mainline code too. Use this code as an example, "prt_val, avg = portfolio_worth_and_avg(asset_values)".

   I guess we can also use the same for Ques 2 and 3, where the helper function returns the dataframe and we print it in the mainline function.