Week 8 Assignment

Jason Cameron

American Military University

This week we completed the lesson on incorporating files into our application code. The following were the requirements: one, add menu items to write to a file and to read from a file; two, test the application; three, sample output; and use the same output from last assignment. This paper will reflect the types of test that were conducted to ensure the application worked according to the assignment requirements.

To begin, the application opens with new menu items prompting the user to select one. The user can select the item by selecting a number or by typing the word or phrase. It does not matter what case the user types in as everything is reduced to a lower case by the code. Notice, that if the use selects either the “SCALC,” or “Save” at the beginning of the program, a prompt will inform them that they cannot do so at this time and will need to restart the application. If the user selects the “Read File” option, they may type in the file name that they want to read. This is done by typing the name without the “.txt” extension (If I had time, I would have written code to check if an extension was found. If it was found the code would manage a way to look up the file accordingly). If a file is not found, and error exception handling function is called and the user is informed “no file found.”

During our testing we choose to complete all the operations on the numbers that will be selected afterward. This was done by selecting the number five. The user could type in “add” or even the letter “a” to get the same result. This was tested in several different scenarios. Text

Description automatically generated

Next the users selected the low number between 100 and -100 per the requirements from the original assignment. It was unsure if this still needed to be incorporated in the application so I kept it, along with the error exception handling codes for all the number requests. The code that requests the “high,” “low,” “first,” and “second” numbers reside in the MyLib.py file as part of the “Calc” class. All numbers are stored in the created object, “objCalc,” are used in the “allInOne” method. This method combines the previous “rangeDetection” function that ensures or “first” and “second” numbers remain between the “high” and “low” numbers. In truth, this check isn’t really needed because the numbers are checked when the user is asked to provide them at the beginning.

There are a couple notes about the “allInOne” method. First, it pulls separate calc methods (add, subtract, multiply, and divide) and stores the results in a list. I believe all the methods could be combined into a complete “All In One.” Second, the list is then fed to a “res” library because we had a requirement in a past assignment and there is a requirement to “use same output as last assignment.” This library can be accessed when the user selects the “SCALC” menu option.

The user selected -10 and 10 for the “low” and “high” numbers. Then the user selects the first and second numbers as 2 and 5 respectfully. Afterward, we see the full string results of their calculations. The user is presented with the menu again. This time the user can select the “SCALC” and “write” function. The user selects “Save Last Answer to File.” Afterwards the user is asked to name the file. The file types “mycalcfile.” The string that is submitted is converted into all lowercase and a “.txt” extension is added to the string. This could cause confusion if someone types “thefile.txt” as “thefile.txt.txt” would be the result. However this also ensures the file isn’t saved as the wrong file type such as “thefile.exe.”

Text

Description automatically generated

The result is a file with the name “mycalcfile.txt” appears in the current folder. The application also informs the user that file is created, and the last calculations performed were save in the file. The data provided in this notice is the exact data that is written in the file. This was done to ensure the “Sample output” requirement was completed.

Text

Description automatically generated

Finally, the user can read the file by selecting the eighth option. The user is asked to type the name of the file that they want to see. This done in the same manner as the when a file is written. The user types “mycalcfile” and the file is read in the terminal. Text

Description automatically generated

Here is an example of what happens when the user types in a file that does not exist:

Text

Description automatically generated

In addition to the requirements, I reviewed the rubric to ensure everything had been completed. The following were completed with the project: assignment details were included in the “flowerpot” portion of the code. Also, comments explaining new features or changes in accordance to Week 08’s requirements are made within the code. These are marked as “Week 08:” while older code is marked as “Old...”

A read and write function were added to the existing application and has been demonstrated here in the paper. Comments were included to show where this code is implemented. Finally, I tested the code multiple times. It is functional, but admittedly clunky.