Key Murray

05/22/2022

**IT FDN 110 A Sp 22: Foundations of Scripting: Python**

**Assignment 06**

[**https://github.com/KAM9010/IntroToProg-Python-Mod06**](https://github.com/KAM9010/IntroToProg-Python-Mod06)

Creating a To Do File Script

**Introduction**

In this document, I will explain the steps I took to modify the To Do File script that we were asked to add code to for assignment 6. I will go over the resources I used to help me complete the to do sections of the script. The functions portion of creating this script was challenging but a little easier as it was basically reformatting the script we created in our last assignment. The last assignment helped a lot with completing this task but still required me to think of how each portion of the script would fit into the three classes outlined in the script.

**Step 1: Processing**

**Read Data from a File**

The first portion of code that I started with was the processing data functions. The function for reading data to a file was already completed by Randal so there was no need to change anything there. That function starts with clearing current data in the list of rows/table of rows, opening the file, splitting the task and priority with a comma when returning the data, stripping the input data to ensure that there are no leading or trailing characters, then appending the row of data to the table (Figure 1).

Text

Description automatically generated

**Figure 1: Screen shot of the function to read data from a file.**

**Add Data to a List**

The next function in the processing class is to add data to the list/table of dictionary rows. For this function I copied part of the code I used in my last script for adding data. Matching the script up with the code I had in my previous assignment, it looked like the only code missing from the function was appending the dictionary row of data to the table. The code I added was “list\_of\_rows.append(row)” which will process the data provided by the user and append it to the dictionary row (Figure 2).

Text

Description automatically generated

***Figure 2: Screen shot of the function to add data to a file.***

**Remove Data from a List**

Removing data from the list was the next function that needed code added to it. The return statement and what it should return was the only line of code provided in this function. I referenced my assignment 5 script and added the code from the step to remove a task from the list. I did not include the line that asked the user for input data as that is not defined as processing data but presentation data. So I moved on to the next line that stated to use a the for- loop statement to execute the ”if” statement. The “if” statement declared that if the task row element was equal to the users input then to remove the task. The comparison occurs by using the comparison operator. The remove method is then used to remove that element from the list and the return statement is used to return the list once the element has been removed (Figure 3).

Text

Description automatically generated

***Figure 3: Screen shot of the function to remove data.***

**Write Data to a File**

The last function defined in the processor class is the function to write data to the file. I had to change a few of the variable names when copying my code as there were different variables declared in this script than what I used in my last script. The variable file\_obj is utilized to open the file. Instead of using the variable declared in the script for the exact file the positional parameter “file\_name” is used and later filled with arguments. It is followed by the mode in which it should open with using the “w” argument. Then the “for-loop” is used to write the data to the text file in the corresponding rows for each key the data should be paired to. The next line closes the file and returns the data that has been saved (Figure 4).

Text

Description automatically generated

***Figure 4: Screen shot of the function to write data to a file.***

**Step 2: Presentation (Input/Output)**

**Output Menu Tasks**

The function to output the menu options was already provided in the script so there was nothing that needed to be added to complete the function. The print function followed by triple quotes is used to print the menu on multiple lines as formatted in the script. Then the print function is used again with no argument so that it prints an empty line for looks (Figure 5).

Text

Description automatically generated

***Figure 5: Screen shot of the function to display a menu of choices.***

**Input Menu Option**

The next function is to get the menu choice from the user. This code was also provided and needed nothing added to it. The variable “choice”, prints out the question to the user, returns their input, and uses the strip method to remove any leading or trailing characters from the data received from the user. Then it prints a new line for looks and returns the choice to the user (Figure 6).

Text

Description automatically generated

***Figure 6: Screen shot of the function to get choice from user.***

**Output Current Task in List**

This function prints out the current list of tasks and priorities to the user. First using the print function to print the message “The current tasks to do are:” followed by the for loop to print out the task and priority rows from the table. Next it prints a line of \* and an empty line for looks (Figure 7).

Text

Description automatically generated

***Figure 7: A screen shot of the function to output the current list of tasks and priorities.***

**Input New Task and Priority**

The function to ask the user which task and priority they would like to add to the list was something we had done in our last assignment, so I copied only the input portion of the code. The processing portion of the code had already been used so it was easy to spot what needed to be added. The first line prints “Please enter a ‘Task’ and ‘Priority’.” then the variable “task” is used to gather the task from the user and the same for the variable for “priority”. The return statement then returns the list of tasks and priority with the new task and priority included (Figure 8).

Text

Description automatically generated

***Figure 8: Screen shot of the function to get input of what tasks to add from the user.***

**Input Task to Remove**

The last function in the class IO is to ask the user which task they would like to remove from the current list. Unlike the option to add task and priorities, the remove option only requires that the user specify which task they would like to remove and then removes the row and any corresponding data in the row (Figure 9).

Text

Description automatically generated

***Figure 9: Screen shot of the function to remove the task the user input.***

**Main Body of the Script**

**Step 1**

The first line specifies the processor class to call the function to read the file. The empty parameter “file\_name” is assigned the variable for the text file to be read (Figure 10).

***Text

Description automatically generated***

***Figure 10: Screen shot of the read data function being used in the main body of the script.***

**Step 2/Step3**

Step 2 uses the “while loop” to display a menu of choices to the user by calling the function from the Class IO to output the current list of tasks while the statement is true. Then step 3 outputs the menu and gets a choice from the user using the functions to output menu task and input menu choices. Next it uses the “continue” statement to loop back to the while true statement (Figure 11).

Text

Description automatically generated

***Figure 11: Screen shot of the output current task, output menu, and input menu functions being used in the main body of the script.***

**Step 4**

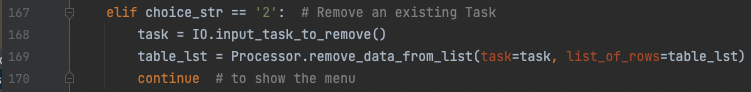
Step 4 is the continuation of step 2 and 3 using the if and “elif” statements. The first “if” statement uses a choice variable to define when option 1 is selected to proceed with using the class IO function to get input from the user. Next the processor class function is called to add data to the list, followed by the continue statement to loop back to the while true statement (Figure 12).

Text

Description automatically generated

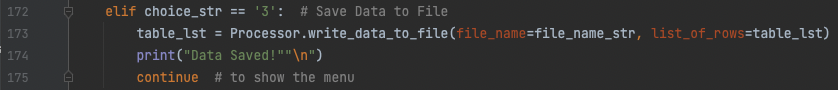
***Figure 12: Screen shot of the input new task and add data to list functions being used in the main body of the script.***

The next “elif” statement is for option 2 to remove data from the list. The IO class function to remove a task is used to ask the user what task they want to remove and next the processor class function is used to remove the task from the list. Lastly the continue statement is used again to loop back to the while true statement (Figure 13).



***Figure 13: Screen shot of the input task to remove and remove data from list functions being used in the main body of the script.***

Another ”elif” statement is used for option 3 which writes the data to the text file. The processor class function to write data to the file is used, followed by the print function to display to the user “Data Saved!” (Figure 14).



***Figure 14: Screen shot of the write data to file function being used in the main body of the script.***

The last “elif” statement is used for choice 4 which uses no class functions but does use the print function to display “Goodbye!” to the user. Then it’s followed with the break statement to break from the while loop and script (Figure 15).

A screenshot of a computer

Description automatically generated with medium confidence

***Figure 15: Screen shot of the code to break from the while loop.***

**Summary**

When I first looked at the script it was a little overwhelming but once I started to break it down into sections I noticed I would be able to utilize code that I had already created. Using functions is a more sophisticated way of organizing code and now when I look at the script, I understand what is happening at each layer. This assignment was fun but challenging and I learned a lot.

**Script running in PyCharm:**

**Option 1:**

**Text

Description automatically generated**

**Option 2:**

**Text

Description automatically generated**

**Option 3:**

**Text

Description automatically generated**

**Option 4:**

**Text

Description automatically generated**

**Script running in Terminal:**

**Option 1:**

**Text

Description automatically generated**

**Option 2:**

**Text

Description automatically generated**

**Option 3:**

**Text

Description automatically generated**

**Option 4:**

**Text

Description automatically generated**