Tyler Tidd
2/5/21
IT FDN 110 B Wi 21
Assignment03

To Do List

Introduction

This week's assignment was a more advanced version of the previous two assignments. We learned about professional coding practices and implemented them in this assignment. I had to create a "To-Do" list using a dictionary to hold the objects then use a list to serve as a table, holding the dictionaries as rows. I had to retrieve and save information in the table to and from a text file. I also had to add, remove, and view data in the table. Once the code was working I verified it by running the program in PyCharm and in the Terminal.

Body

The Set Up

This week we were given an outline script to start with. The objective for us was to fill in varies segments that made the code take certain actions. The code was organized using "Separation of Concerns" (SoC) a coding best practice that makes working with code easier as it becomes more complex. The three segments are: Data, Processing, and Presentation.

Writing the code

The first segment of code in the script was already completed in the "Data" section. Here is where all the variables were initialized.

The first code I need to write was in the "Processing" section. The objective was to retrieve the data, if any, from a text file and store it into a dictionary for each row then add each row into a list to serve as a table. To access and the data in the file I used Python's open() function with "r" mode. I learned that "r" mode enables Python to read data from files, but not write to the file. For the file name I used the pre-defined "objFile" provided in the Data section. Next I needed to convert the data in the file into objects in a list. I used the .split() string method with the "," separator to separate the elements coming from the first line in the file and put them into a list. I then sliced the list putting the elements into the first dictionary row. I added the .strip() method to remove any carriage returns or white space. Now that I had the first row of data, I used the .append() method to store this row into the "lstTable" list. I put these functions into a for loop to repeat these steps for every Task, Priority pair in the text file.

After I completed this segment I realized it would only work if there was a file available. Since there was no file yet I was receiving an error message. I wanted to allow the user to run the code even if there was no file. There weren't any instructions in the course work so I went to Google Search. I found I could use the path.isfile() function from the "os" module to check if the path to a file exists. I combined this with an if function to test if a file exists. If it does, it would run the code in the "Processing" section of the script. If not, it would skip to the "Presentation" or "Input/Output" section of the script.

The next section of code is called "Presentation" or "Input/Output" where we present a menu of options to the user, take their input, then process it accordingly. Normally in professional code we would have functions relay the processing part of the user input to the Processing section but since we have't learned custom functions yet we don't do that here. The outline of the code showing the menu and responses to the user's selections was already provided. Each section was part of an if statement. That statement first used the .strip() method to remove any spaces or carriage returns input by the user, then evaluated the input. Where appropriate, each area ended with a "continue" returning the user to the menu

The first code I had to write in this section was to show the current items in the table. To extract each row in the table I sliced the table using the dictionary keys Task and Priority. I formatted the results into a string with the "|" to separate the Task and Priority. I put this action inside a for loop to iterate through each dictionary row inside the table. However, this whole process wouldn't work if the table was empty. To solve for this possibility I contained the loop in an if statement that checked if the table was empty. If it is, it will skip the loop and print text letting the user know that before returning to the menu.

The next section was to enable the user to add a new item to the table. The first step here was to gather the "Task" and "Priority" from the user using the input() function and store their responses in variables. Next I took those variables and stored them as values in a dictionary by associating the "Task" and "Priority keys with the "Task" and "Priority" variables using the ":" between them. To add this dictionary row to the table I used the new list method I learned this week called .append().

Now that the user is able to add an item to the table they needed to be able to remove the item from the table. The first step was simple, I just asked the user which Task did they want to deleted and stored their response in a variable called "item." I struggled a little with the next step of actually removing the variable from the list. I couldn't find how to do this anywhere in the class notes so I went to Google. I found that I could use an if statement to evaluate if the "Task" key was equal to the item variable. If not, the statement was contained in a loop so it would check the next row. If so, I could then use I could use the new list method I learned called .remove() to remove that row from the list.

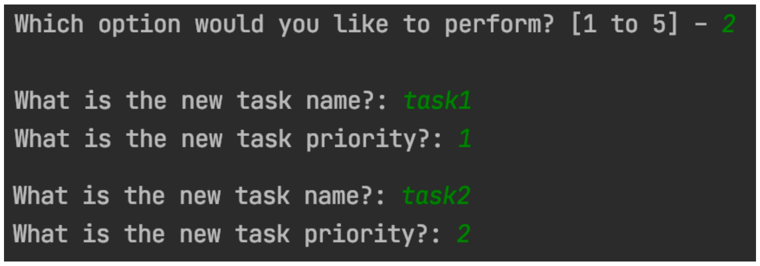
The next option the user could select was to save the current list into a text file. I started with the open() function again but this time set it to a new mode I learned this week called "w" The "w" enables Python to write to the file on your computer. Unlike the append mode, write mode will overwrite the existing contents of the file. I then used the new file method I learned this week called .write() to actually write the contents to the file. I extracted the contents of the list by slicing the Task and Priority keys from the dictionary and added formatting. This was contained in a for loop to iterate through each row. Once the loop was complete the contents of the table were written to the file

The last option was to exit the program. This was the easiest option since all I had to do was print some text letting the user know they had exited.

Verifying the code

The final part of creating this script was to verify the code runs in PyCharm and the Terminal.

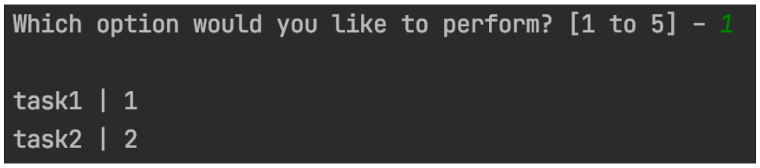
I first ran the code in PyCharm. I stared with an empty table so I stared by adding two tasks. (Figure 1)



```
Which option would you like to perform? [1 to 5] - 2

What is the new task name?: task1
What is the new task priority?: 1

What is the new task name?: task2
What is the new task priority?: 2
```

(Figure 1: Adding tasks)

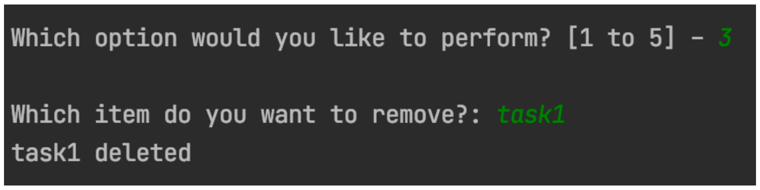
Next I needed to check and see if the tasks were actually added so I used the "Show current data" option. (Figure 2)



```
Which option would you like to perform? [1 to 5] - 1

task1 | 1
task2 | 2
```

(Figure 2: Show current data)

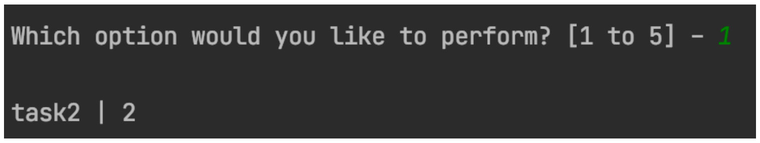
Now that I have data in the table I checked to see if the delete data option worked. (Figure 3)



```
Which option would you like to perform? [1 to 5] - 3

Which item do you want to remove?: task1
task1 deleted
```

(Figure 3: Delete a task)

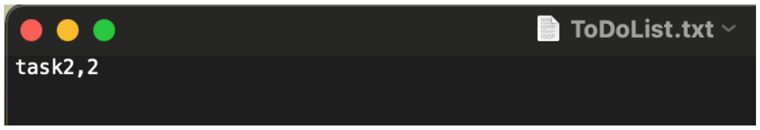
Next I confirmed the task was deleted by showing the current data again (Figure 4)



```
Which option would you like to perform? [1 to 5] - 1

task2 | 2
```

(Figure 4: Show data after task deleted)

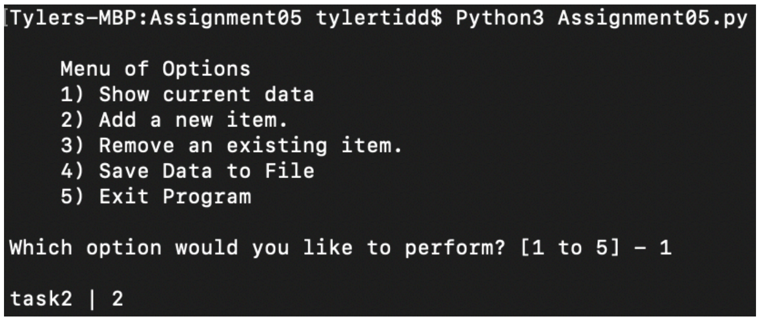
The last options where to save the data to a file and exit the program. Since it only prints out text and doesn't show much I decided not to include screenshots. Lastly I needed to verify the data was saved to a file. (Figure 5)


ToDoList.txt

```
task2,2
```

(Figure 5: Show data in file)

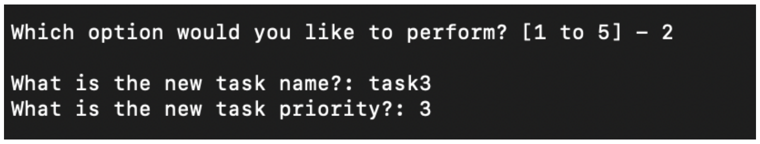
Next I did the same process with the Terminal. I checked for current data and found Task2 that I had created while running the script in PyCharm. (Figure 6)



```
[Tylers-MBP:Assignment05 tylertidd$ Python3 Assignment05.py

    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program

Which option would you like to perform? [1 to 5] - 1

task2 | 2
```

(Figure 6: Terminal start - current data)

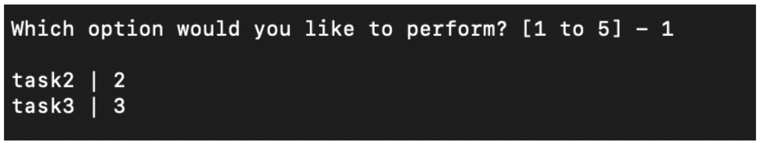
I then added a new task and priority and called them “task3” and “3”. (Figure 7)



```
Which option would you like to perform? [1 to 5] - 2

What is the new task name?: task3
What is the new task priority?: 3
```

(Figure 7: Add task in Terminal)

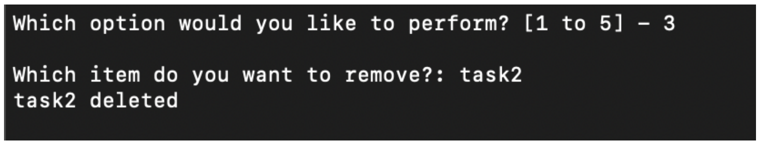
To check to see if the task added I viewed current data again (Figure 8)



```
Which option would you like to perform? [1 to 5] - 1

task2 | 2
task3 | 3
```

(Figure 8: View data after task added)

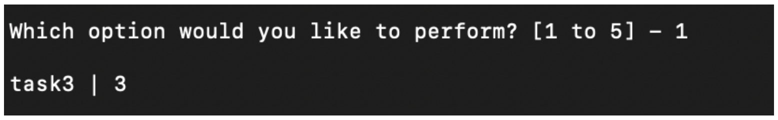
Next I had to delete a task. (Figure 9)



```
Which option would you like to perform? [1 to 5] - 3

Which item do you want to remove?: task2
task2 deleted
```

(Figure 9: Delete task in Terminal)

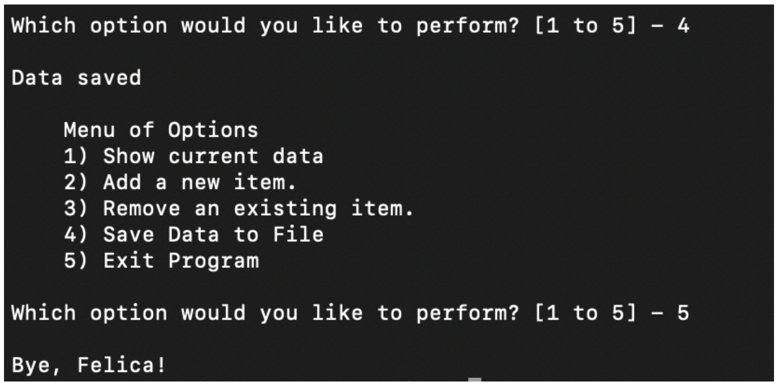
Then I checked to see if the task was actually deleted by viewing the data again. (Figure 10)



```
Which option would you like to perform? [1 to 5] - 1

task3 | 3
```

(Figure 10: Confirm deleted task in Terminal)

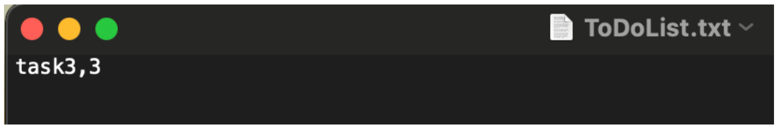
Next I saved the data and exited the program (Figure 11)



```
Which option would you like to perform? [1 to 5] - 4

Data saved

    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program

Which option would you like to perform? [1 to 5] - 5

Bye, Felica!
```

(Figure 11: Save and Exit)

Finally I verified the new task was saved in the text file. (Figure 12)



```
task3,3
```

(Figure 12: Terminal saved to file)

Conclusion

In this assignment I used lists and dictionaries to perform several actions. I extracted and saved data to/from files and added and removed data to/from a table. I used new functions I learned .split() and .strip() to manipulate the data during transit so it would be formatted to be usable. I also used a standard coding practice called Separation of Concerns to organize my data to be more usable by other programmers.

Lastly I ran the code in PyCharm and Terminal using all the menu options to verify the code worked. I also check the text firm to confirm the data was saved.