Davy Til

February 15, 2023

IT FDN 100 A

Assignment 05

https://github.com/dtil-gh/IntroToProg-Python

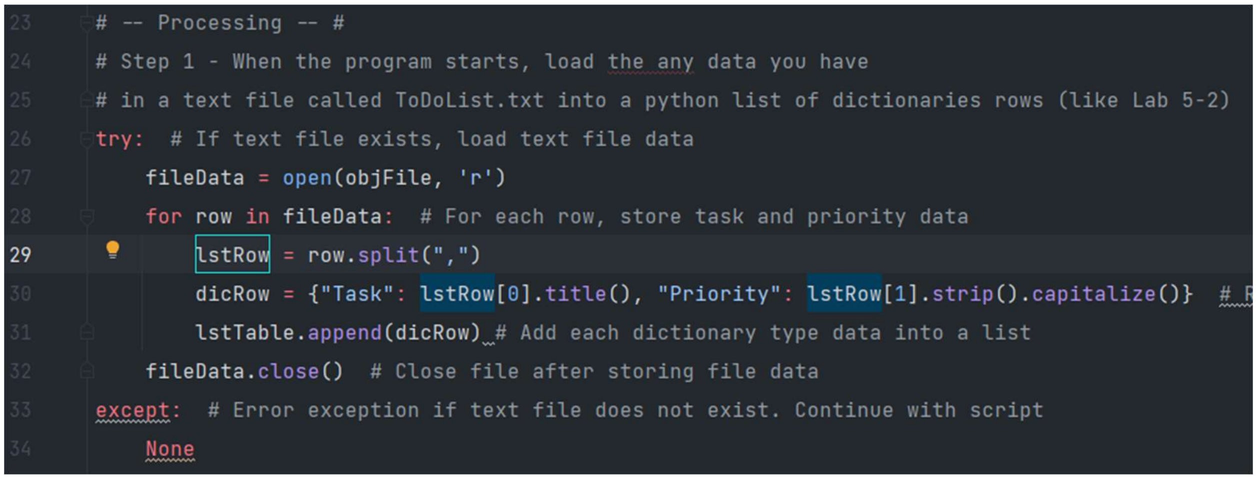
To Do List Priority

Introduction

In this assignment, I will utilize the lessons learned about lists and dictionaries to create a To Do List that lists a task along with its corresponding priority. This Python script will allow the user to either add or remove tasks to a saved To Do List text file. Also, I will be working off of a premade starter script that notes the various steps in order to perform the intended operations. This starter script separates the script into sections; starting with declaring variables at the top, followed by processing script, then lastly input/output script.

Processing Script

Since the variables are already declared, I will start with the processing section of the starter script. This section is to initialize the program immediately after running. I want to open a text file with the task and priorities information and assign it to a list of dictionaries. To plan for how I want my text file to look and how my script will handle the text file, I want each line (or row) of text in the text file to have the task name and assigned priority. To take the data and assign it to the declared variables, I will use a for loop to iterate through each line of text and assign the dictionary's key as Task and dictionary's value as Priority. Afterwards, I can append each dictionaries to the list that will hold all the information. (Figure 1) To combat the scenario where a text file does not exist, I will use error-handling to continue with the program to the Input/Output section. A text file does not need to exist for our script to run properly, and instead it will create one if chosen by the user as we will see later into the script in the Input/Output section.

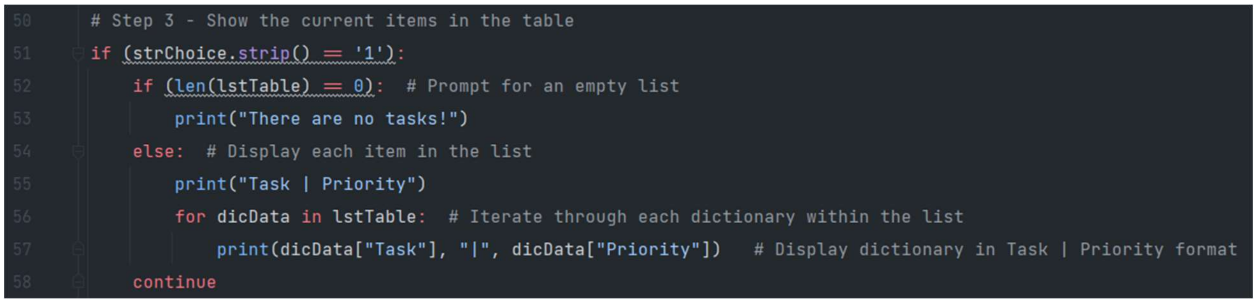


```python
# -- Processing -- #
# Step 1 - When the program starts, load the any data you have
# in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)
try:  # If text file exists, load text file data
    fileData = open(objFile, 'r')
    for row in fileData:  # For each row, store task and priority data
        lstRow = row.split(",")
        dicRow = {"Task": lstRow[0].title(), "Priority": lstRow[1].strip().capitalize()}  # R
        lstTable.append(dicRow) # Add each dictionary type data into a list
    fileData.close()  # Close file after storing file data
except:  # Error exception if text file does not exist. Continue with script
    None
```

Figure 1: Processing Script

Show Current Data Script

The menu options have already been done in this section from the starter script, so the next step is to write a script for each of the menu options listed starting with showing current data. To display the data in the list, it will need a for loop to iterate each dictionary object in the list and then print out the keys and value. (Figure 2) In case that the list does not have any kind of task, I implemented an if-else statement to give different prompts.

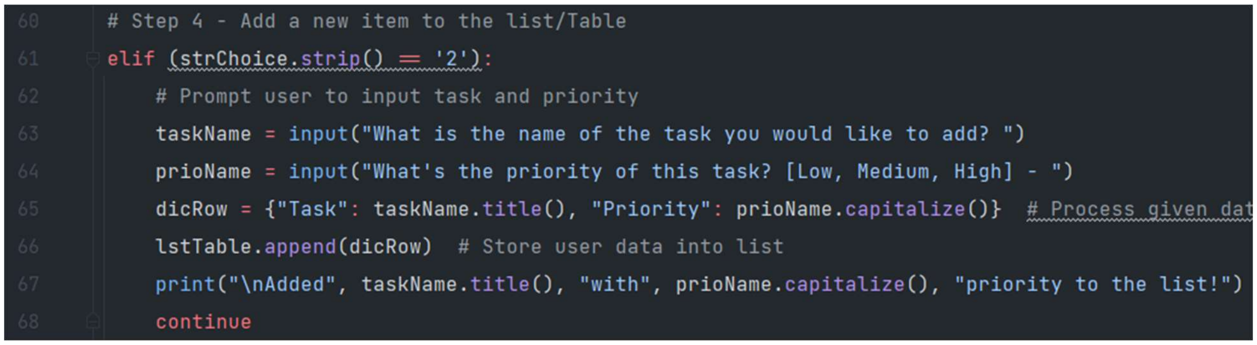


```python
# Step 3 - Show the current items in the table
if (strChoice.strip() == '1'):
    if (len(lstTable) == 0):  # Prompt for an empty list
        print("There are no tasks!")
    else:  # Display each item in the list
        print("Task | Priority")
        for dicData in lstTable:  # Iterate through each dictionary within the list
            print(dicData["Task"], "|", dicData["Priority"])   # Display dictionary in Task | Priority format
    continue
```

Figure 2: Show Data Option

Add New Item to List

Next menu option is to add an item to the list. In this portion, I will prompt the user to input two data values, one for the task name, and the other for the priority scale. I have set an option for priority scale to either be Low, Medium, or High and it is stated in the prompt such that the user knows what kind of information that the program is asking for. (Figure 3) From the inputted data, I assigned them to the "Task" key and "Priority" value of a dictionary variable to be appended to the overall list.

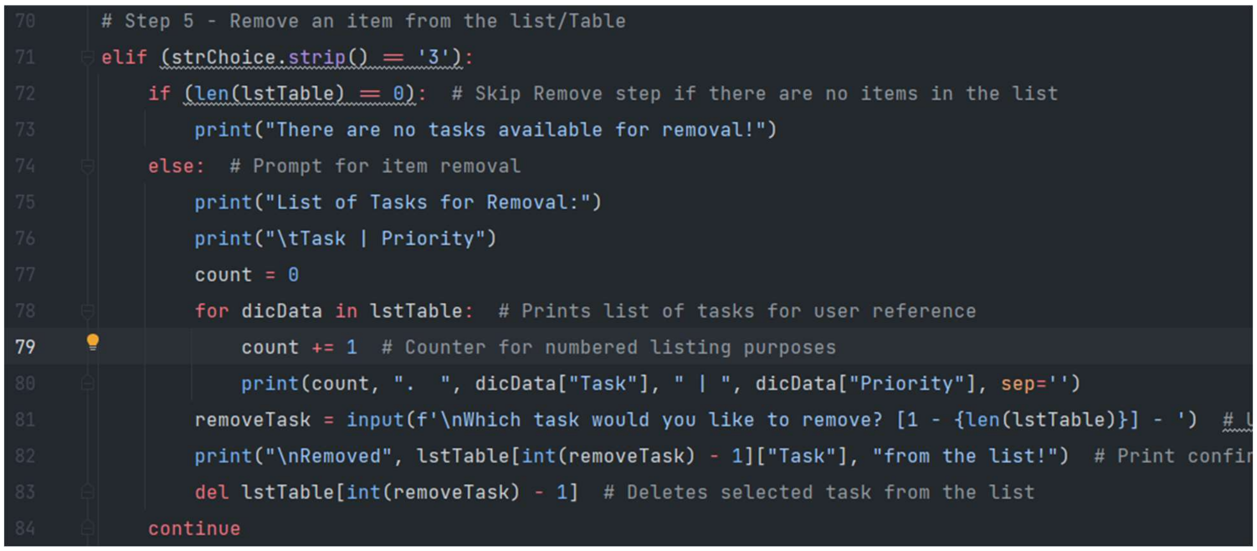


```
    # Step 4 - Add a new item to the list/Table
    elif (strChoice.strip() == '2'):
        # Prompt user to input task and priority
        taskName = input("What is the name of the task you would like to add? ")
        prioName = input("What's the priority of this task? [Low, Medium, High] - ")
        dicRow = {"Task": taskName.title(), "Priority": prioName.capitalize()}  # Process given dat
        lstTable.append(dicRow)  # Store user data into list
        print("\nAdded", taskName.title(), "with", prioName.capitalize(), "priority to the list!")
        continue
```

Figure 3: Adding New Item to List Option

Remove an Item from the List

Next menu option is to allow the user to remove a task from the list. Assuming that the user might not know what kind of tasks are in the list, I will print out the information with a for loop in a numbered list format from lines 77-80 in Figure 4. Then after listing the tasks, I will prompt the user to select a number from the numbered list of tasks that the user would like to remove. From there, I can remove the specific dictionary object from the list in line 83. In case that there are no tasks in the list, the program will let the user know first that there are no tasks in the list and will continue back to the menu options with an if-else statement.

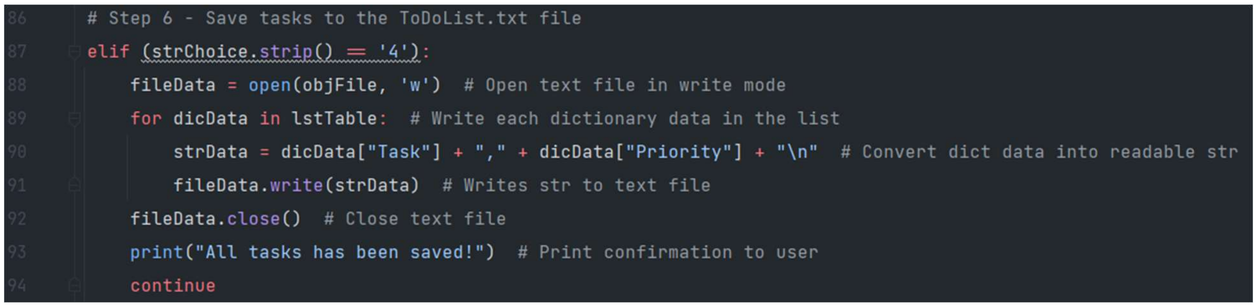


```
70      # Step 5 - Remove an item from the list/Table
71      elif (strChoice.strip() == '3'):
72          if (len(lstTable) == 0):  # Skip Remove step if there are no items in the list
73              print("There are no tasks available for removal!")
74          else:  # Prompt for item removal
75              print("List of Tasks for Removal:")
76              print("\tTask | Priority")
77              count = 0
78              for dicData in lstTable:  # Prints list of tasks for user reference
79                  count += 1  # Counter for numbered listing purposes
80                  print(count, ".  ", dicData["Task"], " | ", dicData["Priority"], sep='')
81              removeTask = input(f'\nWhich task would you like to remove? [1 - {len(lstTable)}] - ')  # U
82              print("\nRemoved", lstTable[int(removeTask) - 1]["Task"], "from the list!")  # Print confir
83              del lstTable[int(removeTask) - 1]  # Deletes selected task from the list
84          continue
```

Figure 4: Remove Item from List Option

Save Tasks to Text File

Next menu option is to save all the tasks to the text file, 'ToDoList.txt'. I can use the open function with write mode to perform this procedure. Having the text file in write mode will allow me to either create a text file if it doesn't exist or overwrite the existing information since all the information processed in this script will have the latest set of information. Since the write function only takes a single string argument, I will need to convert the each dictionary object in the list to a single variable. (Figure 5) Afterwards, the file can close which signals that the data is saved to the text file.

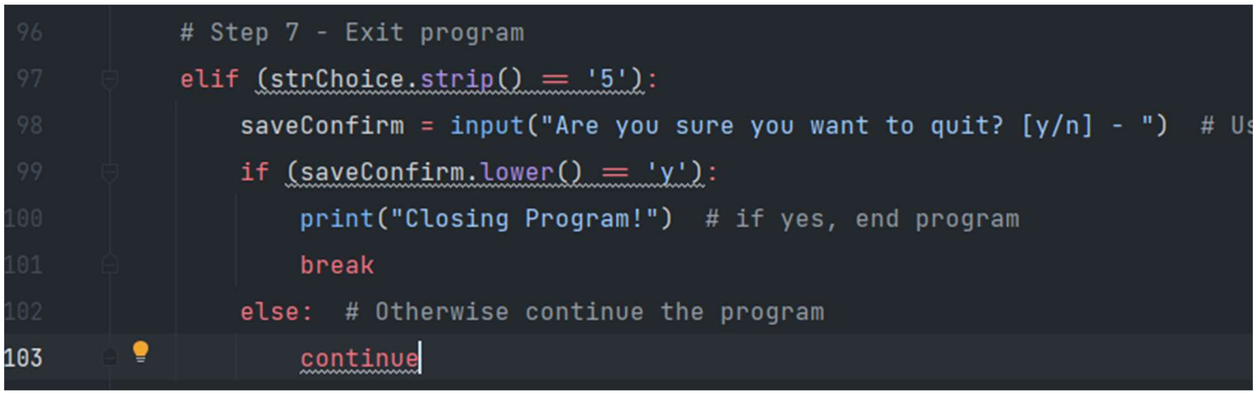


```python
        # Step 6 - Save tasks to the ToDoList.txt file
        elif (strChoice.strip() == '4'):
            fileData = open(objFile, 'w')  # Open text file in write mode
            for dicData in lstTable:  # Write each dictionary data in the list
                strData = dicData["Task"] + "," + dicData["Priority"] + "\n"  # Convert dict data into readable str
                fileData.write(strData)  # Writes str to text file
            fileData.close()  # Close text file
            print("All tasks has been saved!")  # Print confirmation to user
            continue
```

Figure 5: Save List of Tasks to Text File Option

Exiting the Program

Lastly, we need an option to let the program close once the user is finished. As seen in many applications, closing a program prompts the user with an "Are you sure?" prompt. I will do the same by implementing that feature with an input to ask the user if they want to quit and then using if-else statement.

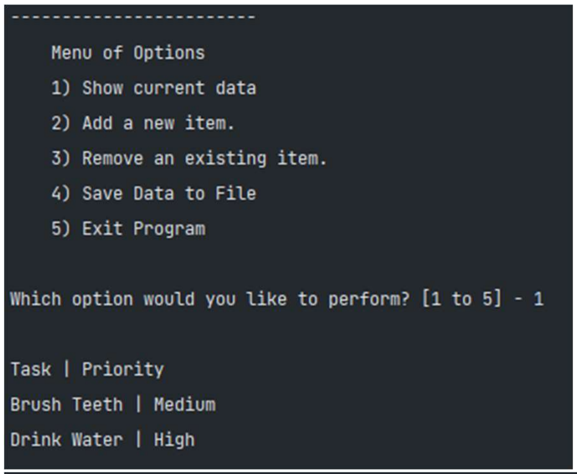


```python
        # Step 7 - Exit program
        elif (strChoice.strip() == '5'):
            saveConfirm = input("Are you sure you want to quit? [y/n] - ")  # Us
            if (saveConfirm.lower() == 'y'):
                print("Closing Program!")  # if yes, end program
                break
            else:  # Otherwise continue the program
                continue
```

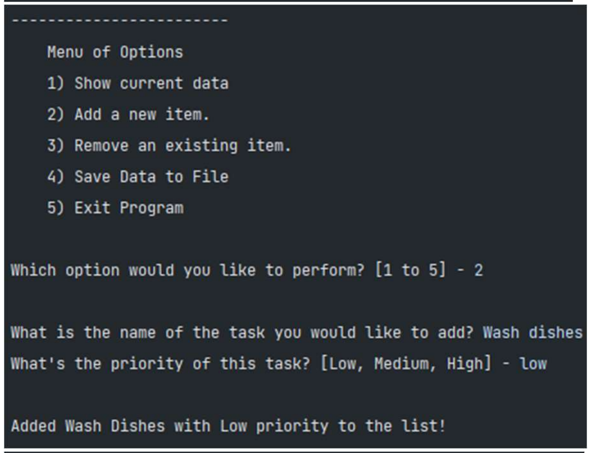
Figure 6: Prompt to Exit the Program

Testing and Verification

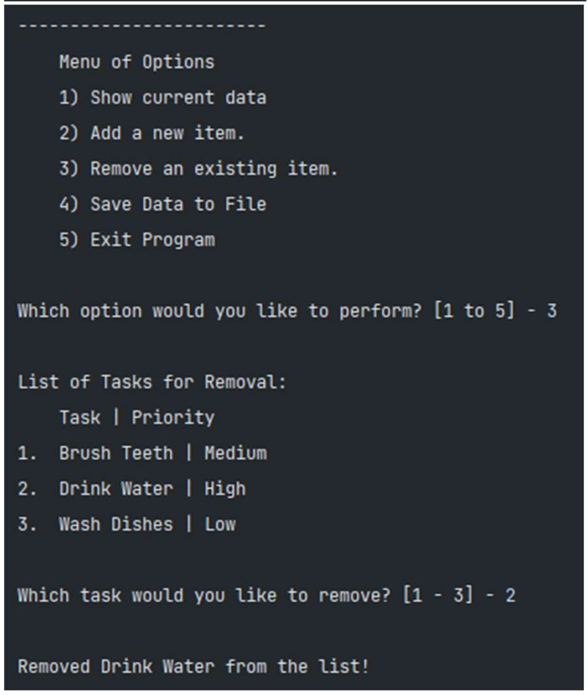
Now that the script is finished, we can perform tests and validate that the program runs as intended. I will test the program in both PyCharm and OS Shell starting with PyCharm. (Figure 7a) The text file started with two task and after running the script, we can see that the text file works as intended from PyCharm. (Figure 7b)



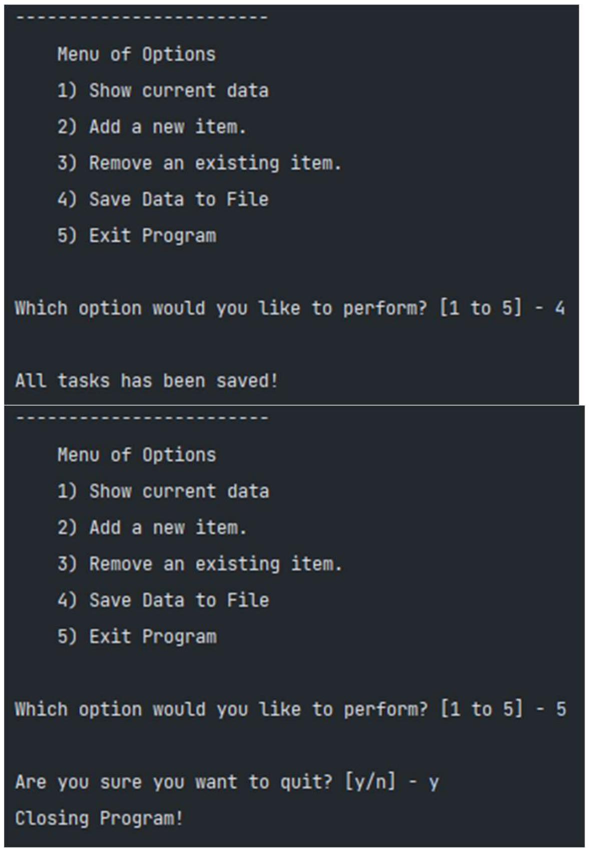
```
------------------------
    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program

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

Task | Priority
Brush Teeth | Medium
Drink Water | High
```



```
------------------------
    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] - 2

What is the name of the task you would like to add? Wash dishes
What's the priority of this task? [Low, Medium, High] - low

Added Wash Dishes with Low priority to the list!
```

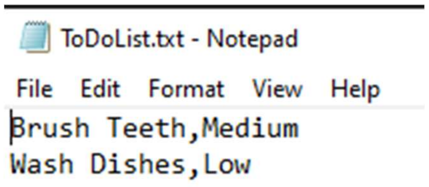


```
------------------------
    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program

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

List of Tasks for Removal:
    Task | Priority
1.  Brush Teeth | Medium
2.  Drink Water | High
3.  Wash Dishes | Low

Which task would you like to remove? [1 - 3] - 2

Removed Drink Water from the list!
```



```
------------------------
    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] - 4

All tasks has been 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

Are you sure you want to quit? [y/n] - y
Closing Program!
```

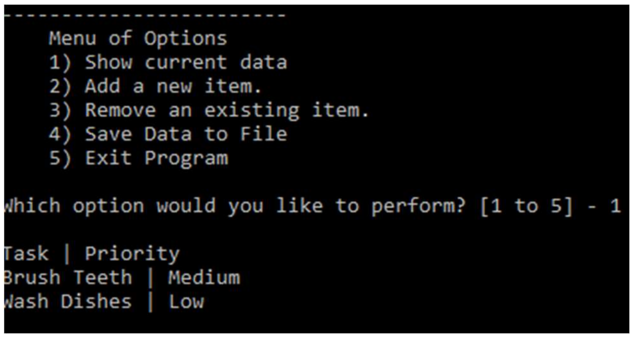
Figure 7a: Testing Script in PyCharm



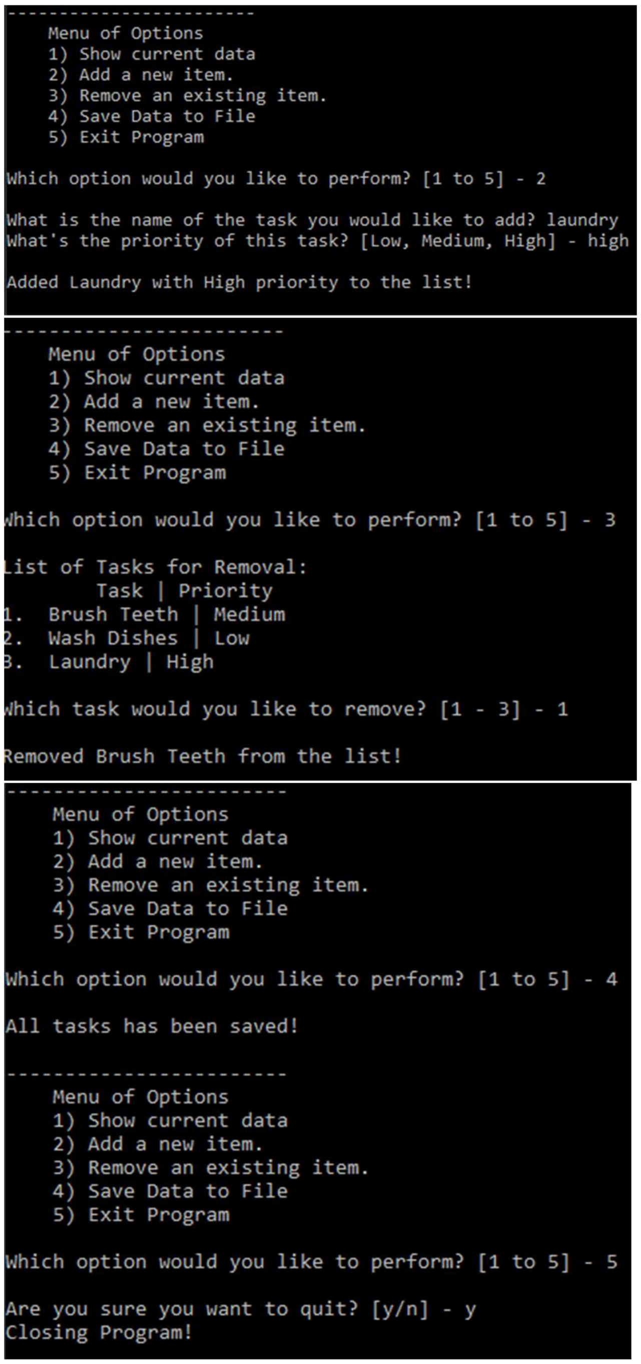
```
Brush Teeth,Medium
Wash Dishes,Low
```

Figure 7b: ToDoList.txt PyCharm Result

Now we can test in OS Shell. (Figure 8a) And the program works as intended for that as well since we have added “Laundry” task and removed “Brush Teeth”. (Figure 8b)



```
------------------------
   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

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

Task | Priority
rush Teeth | Medium
ash Dishes | Low
```

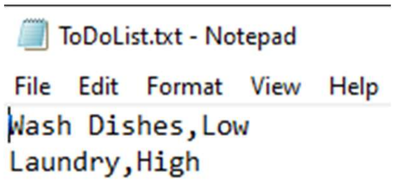


```
-----------------------
    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program

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

What is the name of the task you would like to add? laundry
What's the priority of this task? [Low, Medium, High] - high

Added Laundry with High priority to the list!
```

```
-----------------------
    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] - 3

List of Tasks for Removal:
        Task | Priority
1.  Brush Teeth | Medium
2.  Wash Dishes | Low
3.  Laundry | High

Which task would you like to remove? [1 - 3] - 1

Removed Brush Teeth from the list!
```

```
-----------------------
    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] - 4

All tasks has been 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

Are you sure you want to quit? [y/n] - y
Closing Program!
```

Figure 8a: Testing Script in OS Shell



```
Wash Dishes,Low
Laundry,High
```

Figure 8b: ToDoList.txt OS Shell Result

Summary

In this module, I learned how to add code to an existing script and utilize both lists and dictionaries as loops and if-else statements. The biggest challenge was to add code to an existing script because that forces you to use the parameters given instead of using your own. It can also be an ease of mind since the script might even lay out the proper steps to take to finish the script as seen from the premade starter script file.