Davy Til

February 22, 2023

IT FDN 100 A

Assignment 06

https://github.com/dtil-gh/IntroToProg-Python-Mod06

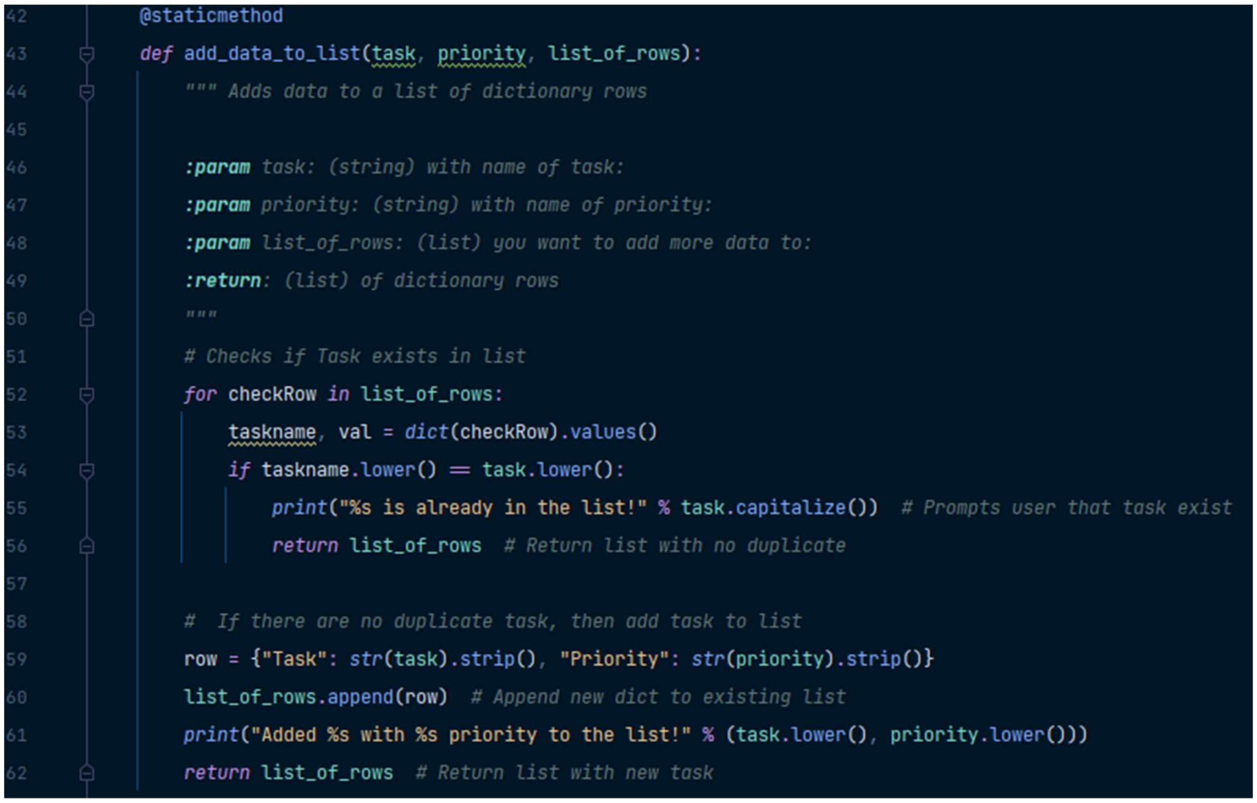
To Do List Python Script

Introduction

In this assignment, I will utilize the lessons learned about classes and functions to build upon a To Do List Python script 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 a premade starter script that notes the various steps to perform the intended operations. Within this script, it has 2 classes; one of which processes the data, and the other will handle the input and output of the data.

Processing Class Script

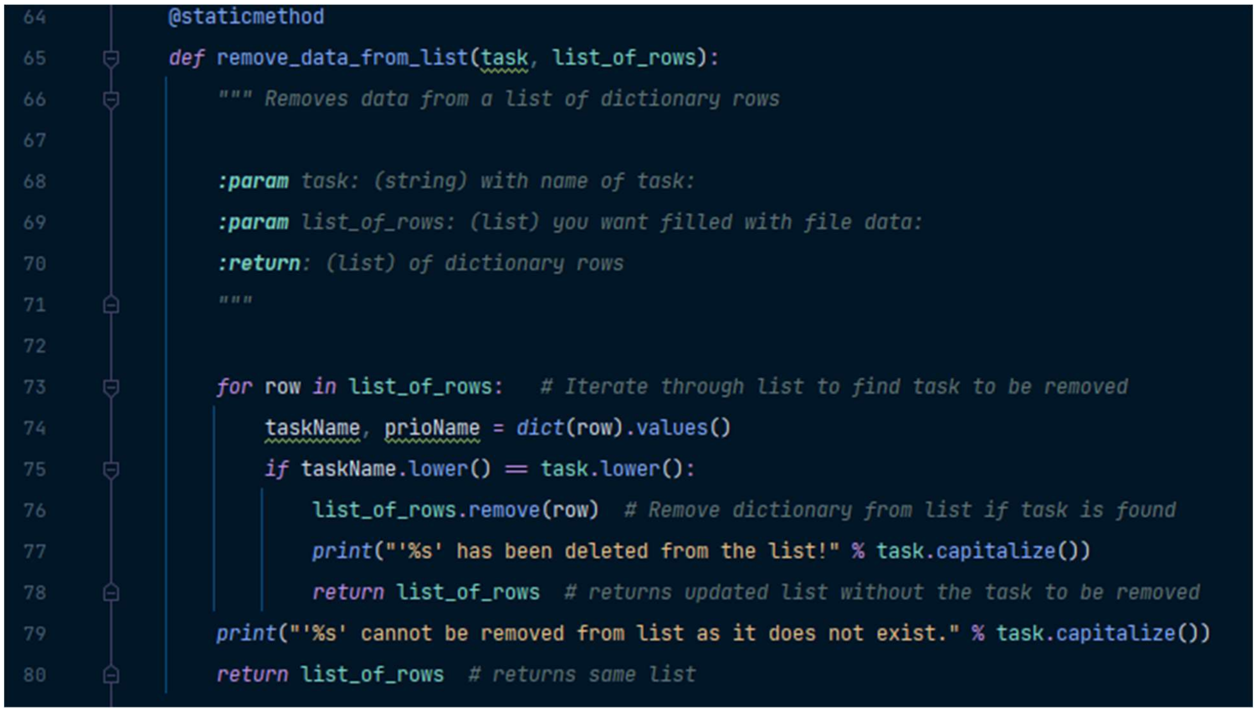
The first part of the script that needs to be updated from the starter script file is the Processing Class. One of the three functions to be updated is the 'add_data_to_list', which adds data to a list of dictionary. This function will be used in the 1st menu option. To prevent adding duplicate tasks to the list, I want to iterate through the list first to check if the task does exist. (Figure 1) If there is a duplicate, then the next if statement will prompt the user that there is a duplicate and will not add the data. Otherwise, the program will proceed to add the data to the list.



```python
    @staticmethod
    def add_data_to_list(task, priority, list_of_rows):
        """ Adds data to a list of dictionary rows

        :param task: (string) with name of task:
        :param priority: (string) with name of priority:
        :param list_of_rows: (list) you want to add more data to:
        :return: (list) of dictionary rows
        """
        # Checks if Task exists in list
        for checkRow in list_of_rows:
            taskname, val = dict(checkRow).values()
            if taskname.lower() == task.lower():
                print("%s is already in the list!" % task.capitalize())  # Prompts user that task exist
                return list_of_rows  # Return list with no duplicate

        #  If there are no duplicate task, then add task to list
        row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
        list_of_rows.append(row)  # Append new dict to existing list
        print("Added %s with %s priority to the list!" % (task.lower(), priority.lower()))
        return list_of_rows  # Return list with new task
```

Figure 1: ‘add_data_to_list’ Function

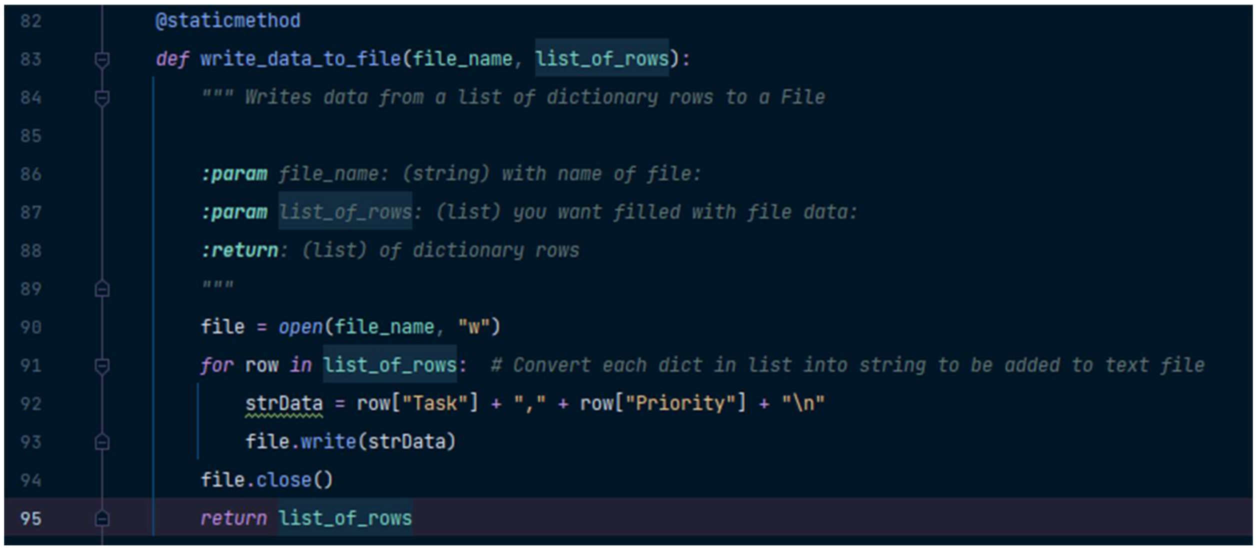
Second function to be updated is the ‘remove_data_from_list’ function, which is to remove a dictionary from the list of dictionaries. I will have the program iterate through the list of dictionaries and find the task to be removed. If the task name has been found, then the function will return the updated list of dictionaries in line 78. (Figure 2) If the task hasn’t been found, then the program will continue outside of the for loop to return the original list in line 80.



```python
    @staticmethod
    def remove_data_from_list(task, list_of_rows):
        """ Removes data from a list of dictionary rows

        :param task: (string) with name of task:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """

        for row in list_of_rows:   # Iterate through list to find task to be removed
            taskName, prioName = dict(row).values()
            if taskName.lower() == task.lower():
                list_of_rows.remove(row)  # Remove dictionary from list if task is found
                print("'%s' has been deleted from the list!" % task.capitalize())
                return list_of_rows  # returns updated list without the task to be removed
        print("'%s' cannot be removed from list as it does not exist." % task.capitalize())
        return list_of_rows  # returns same list
```

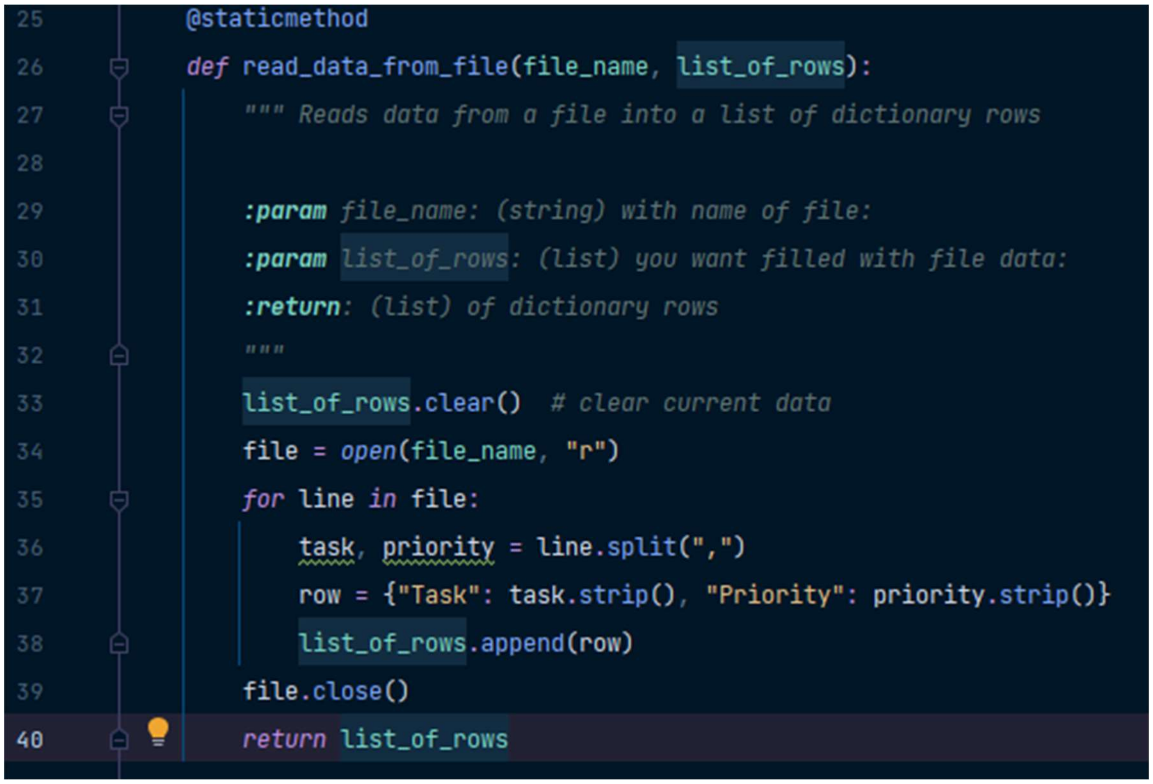
Figure 2: 'remove_data_from_list' Function

Lastly of the Processor class is the 'write_data_to_file' function that saves the list of dictionary data in the program onto the text file. I used the 'w' argument in the open function such that the list that exists in the dictionary will override all preexisting data in the text file. (Figure 3) I have converted the data from the list into a string format that can be read from the premade 'read_data_from_file' function as seen on Figure 4.



```python
    @staticmethod
    def write_data_to_file(file_name, list_of_rows):
        """ Writes data from a list of dictionary rows to a File

        :param file_name: (string) with name of file:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        file = open(file_name, "w")
        for row in list_of_rows:  # Convert each dict in list into string to be added to text file
            strData = row["Task"] + "," + row["Priority"] + "\n"
            file.write(strData)
        file.close()
        return list_of_rows
```

Figure 3: 'write_data_to_file' Function

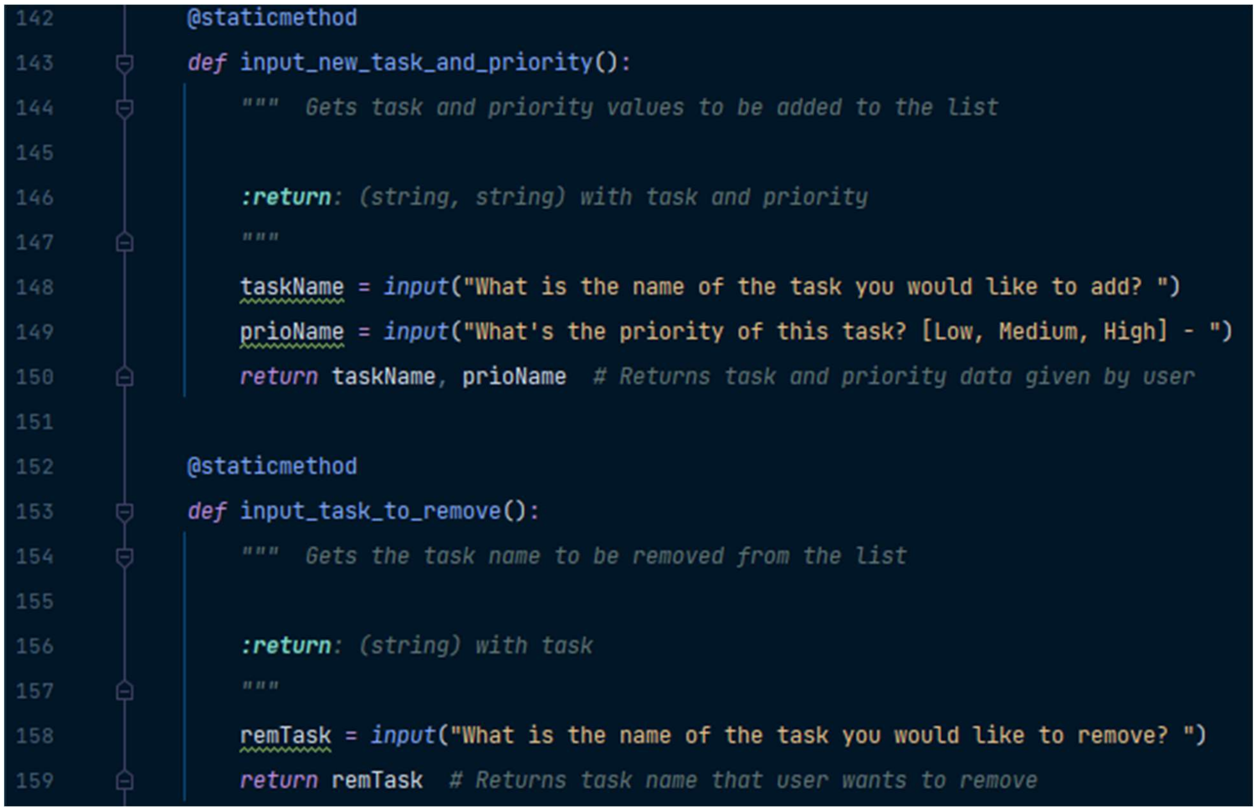


```python
    @staticmethod
    def read_data_from_file(file_name, list_of_rows):
        """ Reads data from a file into a list of dictionary rows

        :param file_name: (string) with name of file:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        list_of_rows.clear()  # clear current data
        file = open(file_name, "r")
        for line in file:
            task, priority = line.split(",")
            row = {"Task": task.strip(), "Priority": priority.strip()}
            list_of_rows.append(row)
        file.close()
        return list_of_rows
```

Figure 4: ‘read_data_from_file’ Function Existed from Starter File

Input/Output Class Script

In this class, the only areas to add code from the starter file is the inputting data to add to the list and remove from the list. Working with the starter script and looking at the ‘add_data_to_list’ function, two of the 3 arguments are task and priority data. From this I can let the ‘input_new_task_and_priority’ function return two data values. (Figure 5) Similarly with ‘input_task_to_remove’ function will return one data value based on the argument needed from ‘remove_data_from_list’ function.

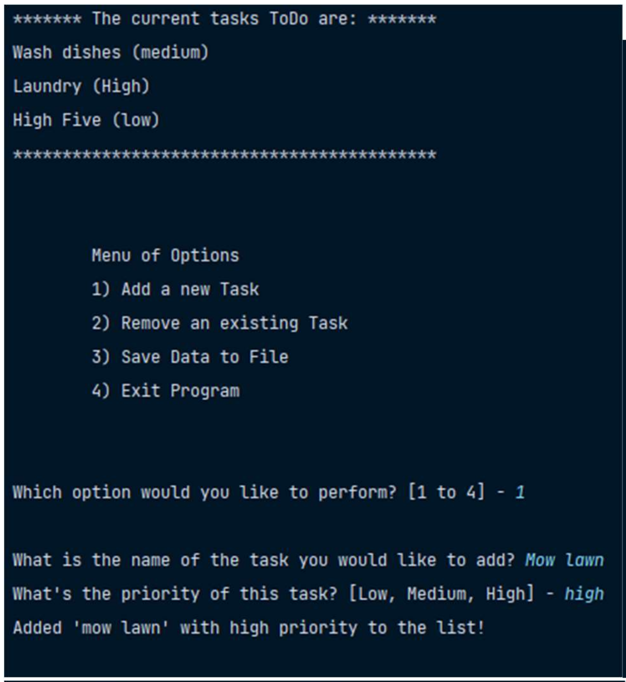


```
    @staticmethod
    def input_new_task_and_priority():
        """  Gets task and priority values to be added to the list

        :return: (string, string) with 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] - ")
        return taskName, prioName  # Returns task and priority data given by user

    @staticmethod
    def input_task_to_remove():
        """  Gets the task name to be removed from the list

        :return: (string) with task
        """
        remTask = input("What is the name of the task you would like to remove? ")
        return remTask  # Returns task name that user wants to remove
```

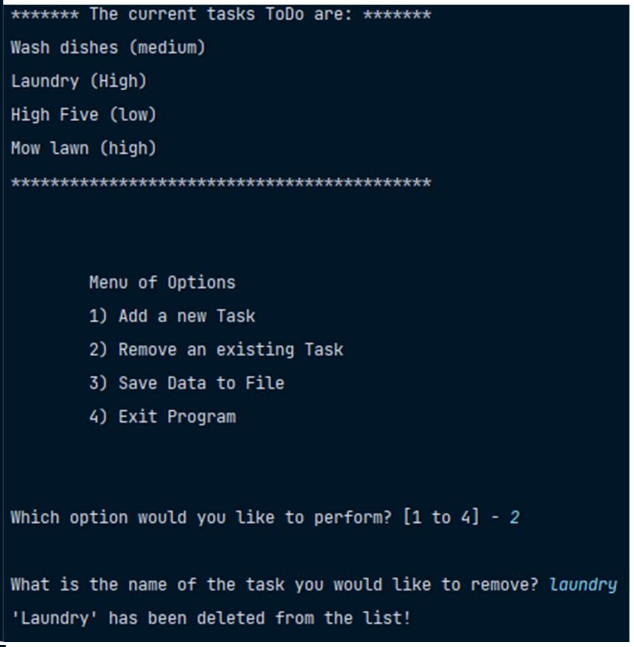
Figure 5: Input Functions to Add Data and Remove Data

Testing the Script

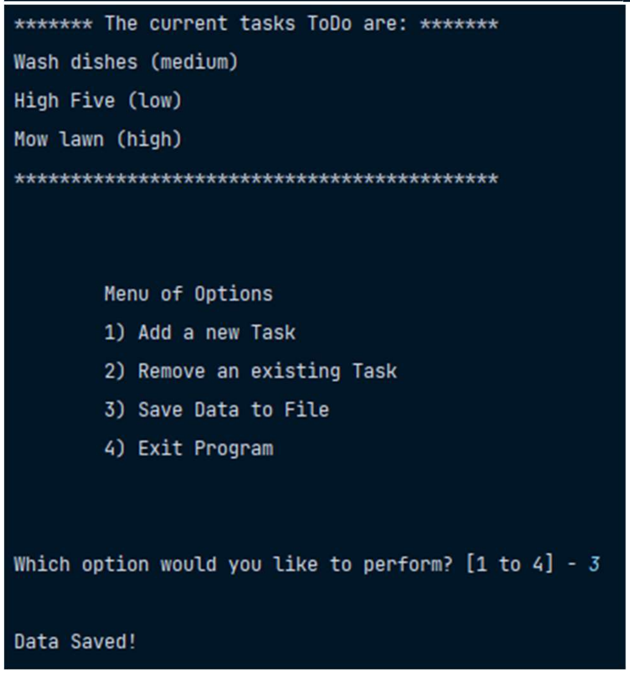
Now that the entire starter script has been fully updated, I will test menu options 1-3 that I have written code for using an existing ToDoList.txt file in both PyCharm (Figure 6 & 7) and Command Prompt (Figure 8 & 9).



```
******* The current tasks ToDo are: *******
Wash dishes (medium)
Laundry (High)
High Five (low)
*******************************************



        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program



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

What is the name of the task you would like to add? Mow lawn
What's the priority of this task? [Low, Medium, High] - high
Added 'mow lawn' with high priority to the list!
```

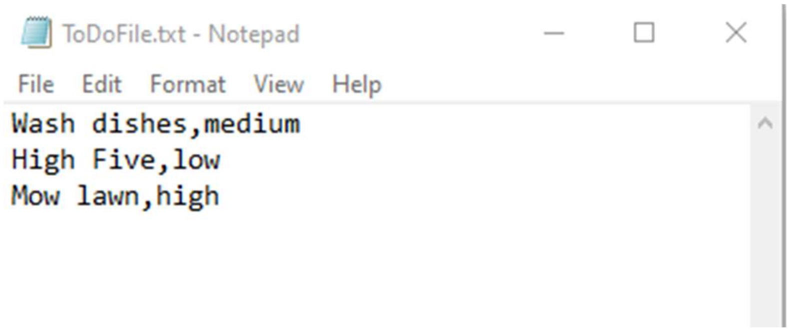


```
******* The current tasks ToDo are: *******
Wash dishes (medium)
Laundry (High)
High Five (low)
Mow lawn (high)
*******************************************



        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program



Which option would you like to perform? [1 to 4] - 2

What is the name of the task you would like to remove? laundry
'Laundry' has been deleted from the list!
```



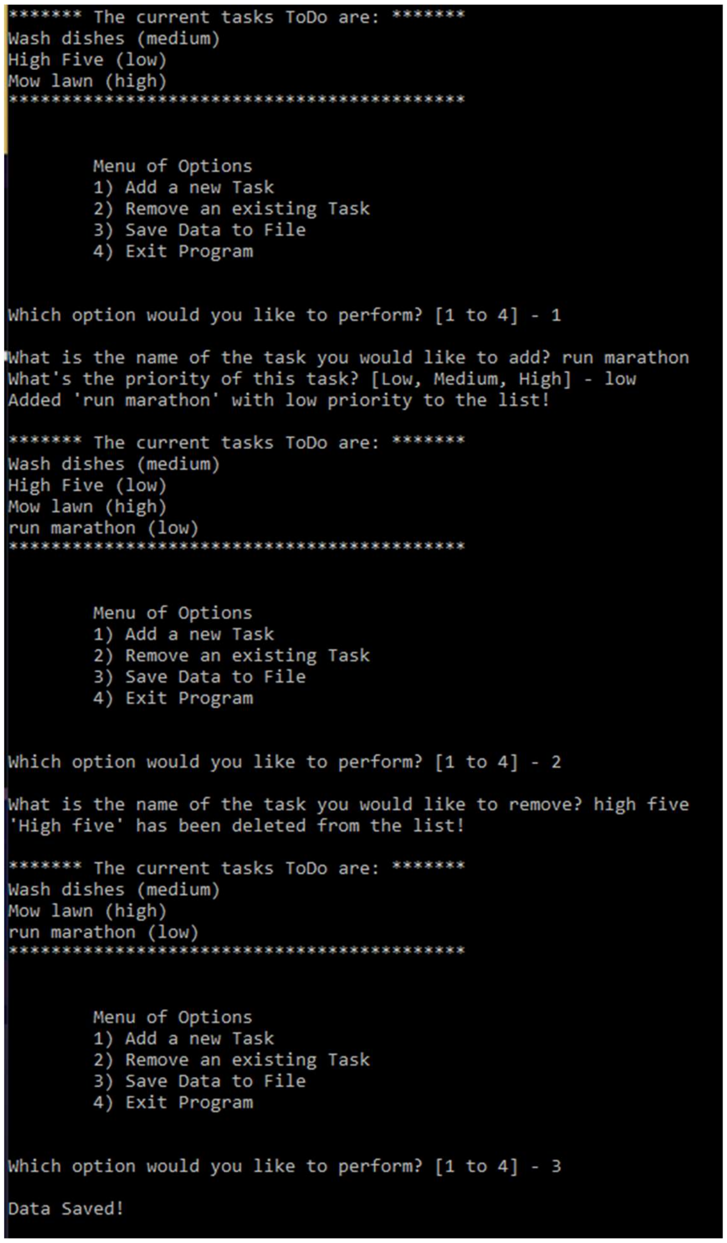
```
******* The current tasks ToDo are: *******
Wash dishes (medium)
High Five (low)
Mow lawn (high)
*******************************************



        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program



Which option would you like to perform? [1 to 4] - 3

Data Saved!
```

Figure 6: Running the Finalized Script in PyCharm



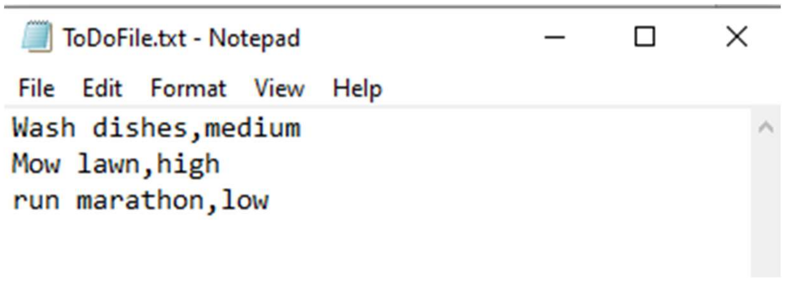
```
Wash dishes,medium
High Five,low
Mow lawn,high
```

Figure 7: PyCharm Program Text File Output



```
******* The current tasks ToDo are: *******
Wash dishes (medium)
High Five (low)
Mow lawn (high)
*******************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


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

What is the name of the task you would like to add? run marathon
What's the priority of this task? [Low, Medium, High] - low
Added 'run marathon' with low priority to the list!

******* The current tasks ToDo are: *******
Wash dishes (medium)
High Five (low)
Mow lawn (high)
run marathon (low)
*******************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 2

What is the name of the task you would like to remove? high five
'High five' has been deleted from the list!

******* The current tasks ToDo are: *******
Wash dishes (medium)
Mow lawn (high)
run marathon (low)
*******************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 3

Data Saved!
```

Figure 8: Running the Finalized Script in Command Prompt



```
Wash dishes,medium
Mow lawn,high
run marathon,low
```

Figure 9: Command Prompt Text File Output

Summary

In this module, I learned how to further add code to an existing script and utilize functions and classes. Functions and classes can be helpful as it organizes the script. Also, I have used the debugging tool and it is easier to work with when dealing with functions as the issues that I've run across are at least within the function itself and not outside of the function.