Name: Mohammed Qadri
Date: 2023-08-15
Course: IT FDN 110 A
GitHub URL: https://github.com/mqadri22/IntroToProg-Python-Mod06
GitHub Site: https://mqadri22.github.io/IntroToProg-Python-Mod06/

Assignment 06 – Functions & Classes

Introduction

The objective of Assignment 06 is to gain familiarity with functions, classes of functions, and related aspects of programming in Python such as shadowing, etc. This would be accomplished through the creation of a script that builds a Task/Priority To Do list and modifies it based on user command and input. The responses to different user commands or inputs would each be encapsulated in separate functions that would be categorized in different classes.

Basic Code Structure

The basic structure of the script was already provided with the starter file in the assignment. It closely resembled the script written for Assignment 05, which served a similar purpose without the use of custom functions. Assignment 06, in contrast, focuses on the use of custom functions in different classes.

Similar to Assignment 05, the script begins with the declaration of variables that will be used throughout the script and often read in as parameters in the various functions. The starter file for this assignment provided two pre-defined classes for the functions that would be built in this assignment – as well as framework for said unfinished functions. These classes of functions are called Processor and IO (Input/Output), and categorize functions accordingly: Processor functions deal with the processing of data, whereas IO (Input/Output) functions deal with displaying menus and retrieving user input. Each class consists of multiple functions that are conditionally called from the main body of the script that follows.

The main body of the script begins with the execution of a Processor function to load data from an existing file. This is then followed by a while loop that prompts the user to select an option from the menu at each iteration and then calls functions from both the IO and Processor classes in accordance with the user input. This section of the script closely resembles the script from Assignment 05, the primary difference being the lines of code that were once nested within each if-elif-…-else are now contained within standalone functions. This significantly improves readability of the script.

Functions & Classes

Due to the similarity of this script and its uses to that of Assignment 05, there was not much novel in each individual line of code. Instead, the main learnings from this Assignment pertained to functions and classes. Though the author had made use of functions in previous assignments, certain characteristics and aspects of functions novel to them appeared in this assignment.

The first of these novel aspects of functions was named arguments. Prior to this assignment, it was assumed that Python only made use of positional arguments – similar to Matlab, the programming language with which the author is the most familiar. Though the named arguments did not need to be manipulated for the purposes of this assignment, they add a degree of robustness to the code when executing various functions in the main body of the script. Errors stemming from incorrect argument entry order could be headed off using this capability.

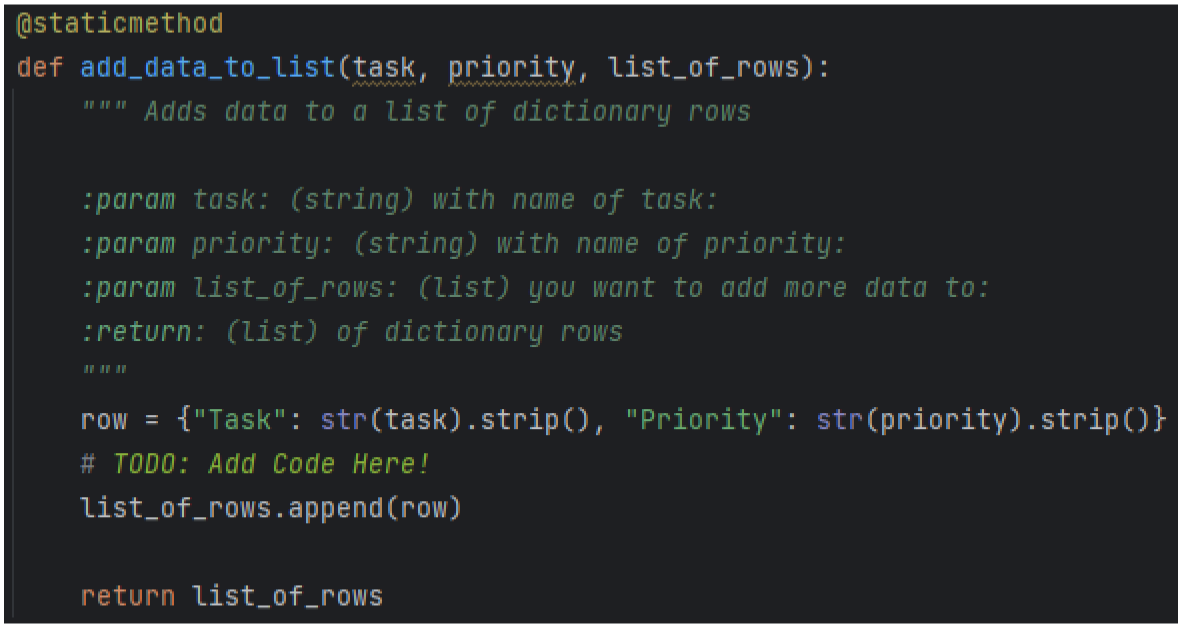
Next are function document headers, which are more of a quality of life characteristic. The standardization provided by use of function document headers helps significantly when attempting to understand how a function written by someone else works (as in this assignment) or when troubleshooting a script that makes use of a function.

Another lesson that was learned was in regard to shadowing. Though no variables were explicitly labelled global in this script, the issue of shadowing did arise when the functions were being modified. Using parameter or output names in functions similar to those used when executing the function in the script main body could potentially lead to errors or data mishandling. PyCharm noted as much in a suggestion to the author, which prompted the renaming of local variables in the functions that were being modified.

These lessons combined substantially augmented the author's working understanding of functions in Python and facilitated in completing the script, as described in the following section.

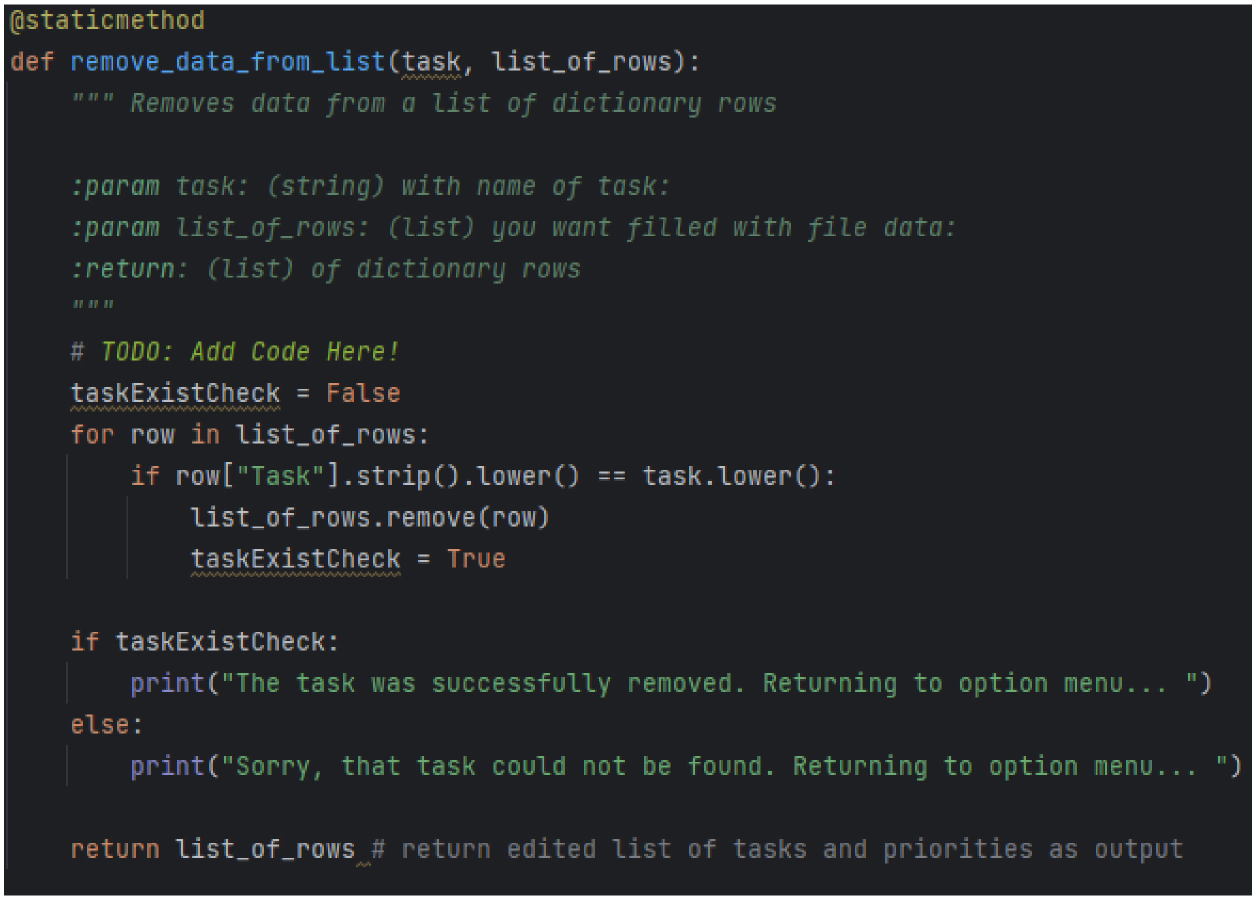
Script: Processing Class

The Processing class contains 3 functions that were modified by the author: a function to add data (a task and its priority) to a list, another to remove data (a user specified task) from a list, and another to save data to a file. As mentioned before, the contents of the individual functions were very similar to Assignment 05 so are not discussed in detail here. Figures 1 through 3 below show the Processing class functions.



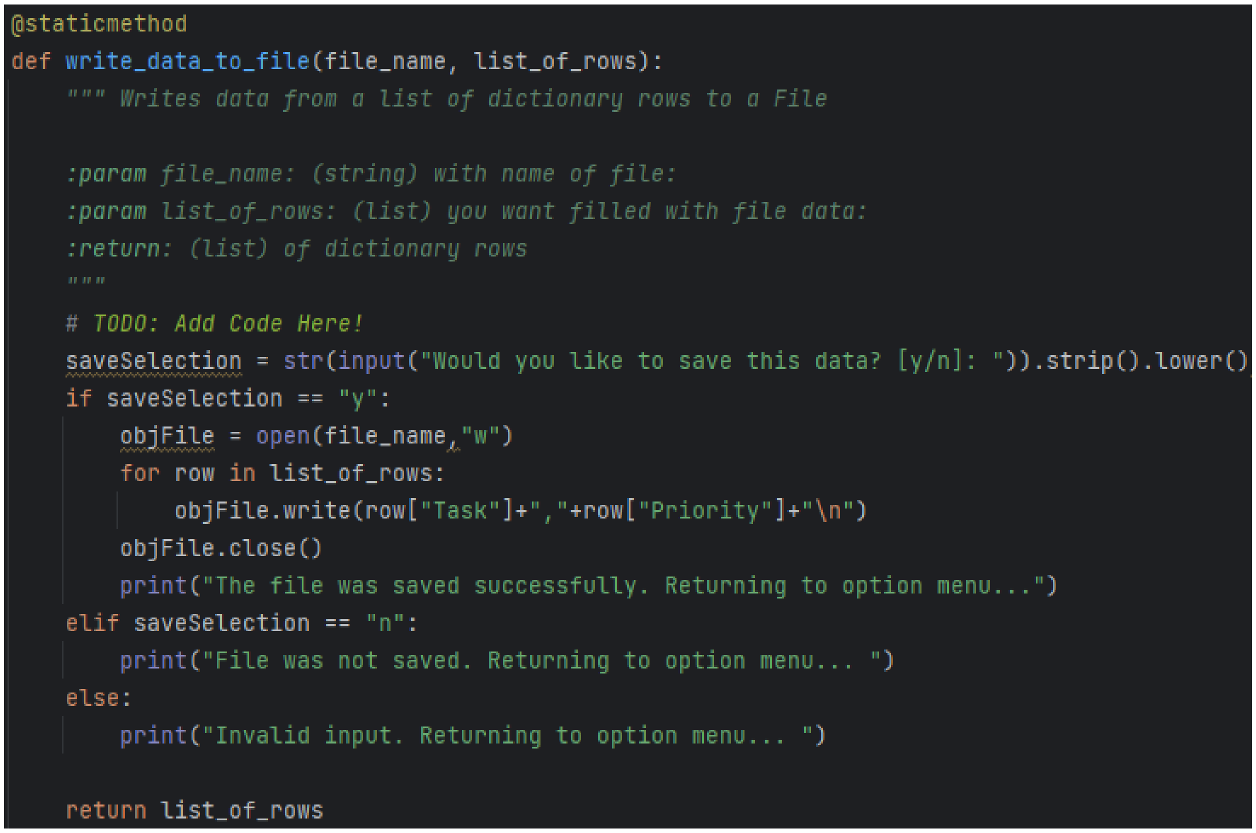
```python
@staticmethod
def add_data_to_list(task, priority, list_of_rows):
    """ Adds data to a list of dictionary rows

    :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
    """
    row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
    # TODO: Add Code Here!
    list_of_rows.append(row)

    return list_of_rows
```

Figure 1. Processing class function to add data to a list



```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
    """
    # TODO: Add Code Here!
    taskExistCheck = False
    for row in list_of_rows:
        if row["Task"].strip().lower() == task.lower():
            list_of_rows.remove(row)
            taskExistCheck = True

    if taskExistCheck:
        print("The task was successfully removed. Returning to option menu... ")
    else:
        print("Sorry, that task could not be found. Returning to option menu... ")

    return list_of_rows # return edited list of tasks and priorities as output
```

Figure 2. Processing class function to remove data from a list

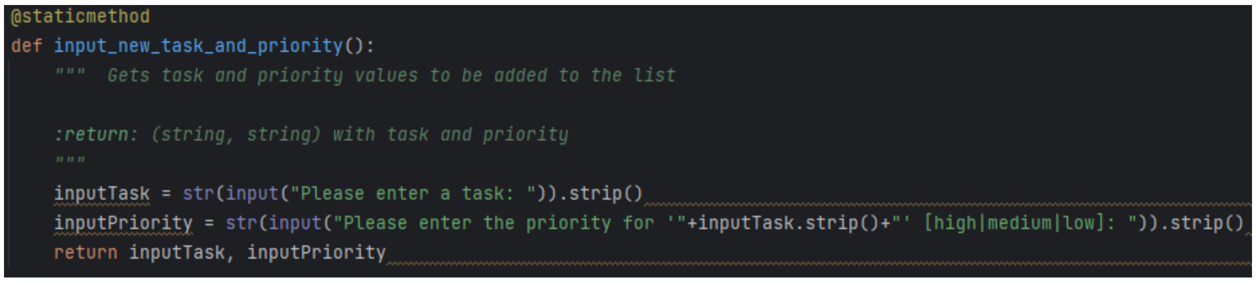


```
@staticmethod
def write_data_to_file(file_name, list_of_rows):
    """ Writes data from a list of dictionary rows to a File

    :param file_name: (string) with name of file:
    :param list_of_rows: (list) you want filled with file data:
    :return: (list) of dictionary rows
    """
    # TODO: Add Code Here!
    saveSelection = str(input("Would you like to save this data? [y/n]: ")).strip().lower()
    if saveSelection == "y":
        objFile = open(file_name,"w")
        for row in list_of_rows:
            objFile.write(row["Task"]+","+row["Priority"]+"\n")
        objFile.close()
        print("The file was saved successfully. Returning to option menu...")
    elif saveSelection == "n":
        print("File was not saved. Returning to option menu... ")
    else:
        print("Invalid input. Returning to option menu... ")

    return list_of_rows
```

Figure 3. Processing class function to save data to a file

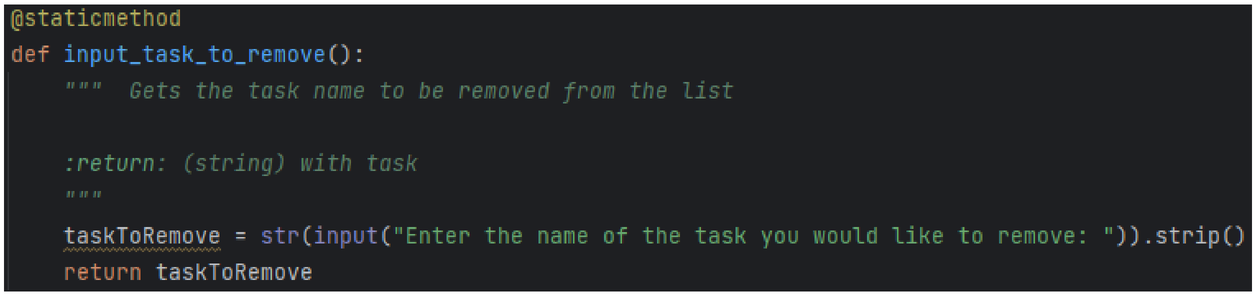
Script: IO Class

The remaining edits made to the script for this Assignment were within the IO (input/output) class. Two functions within the IO class were edited: one to retrieve new task and priority data for addition to a list, and another to retrieve the name of a task to be removed from that list. Again, the contents of these functions are similar to the script written for Assignment 05 and are thus not described in detail here. Figures 4 and 5 below show each modified function within the IO class.



```
@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
    """
    inputTask = str(input("Please enter a task: ")).strip()
    inputPriority = str(input("Please enter the priority for '"+inputTask.strip()+"' [high|medium|low]: ")).strip()
    return inputTask, inputPriority
```

Figure 4. IO class function for retrieving new tasks and priorities for addition to a list

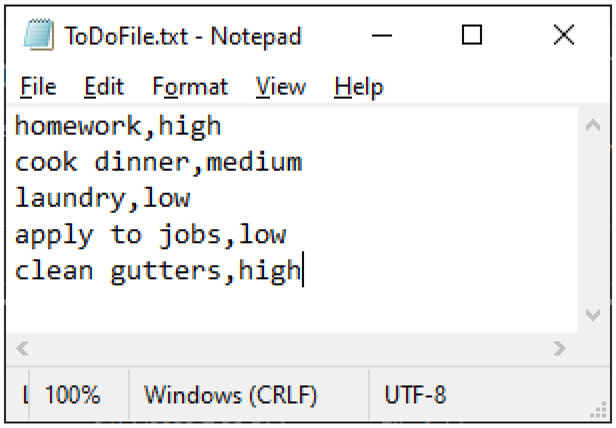


```
@staticmethod
def input_task_to_remove():
    """  Gets the task name to be removed from the list

    :return: (string) with task
    """
    taskToRemove = str(input("Enter the name of the task you would like to remove: ")).strip()
    return taskToRemove
```

Figure 5. IO class function for retrieving the name of a task for deletion from a list

Results

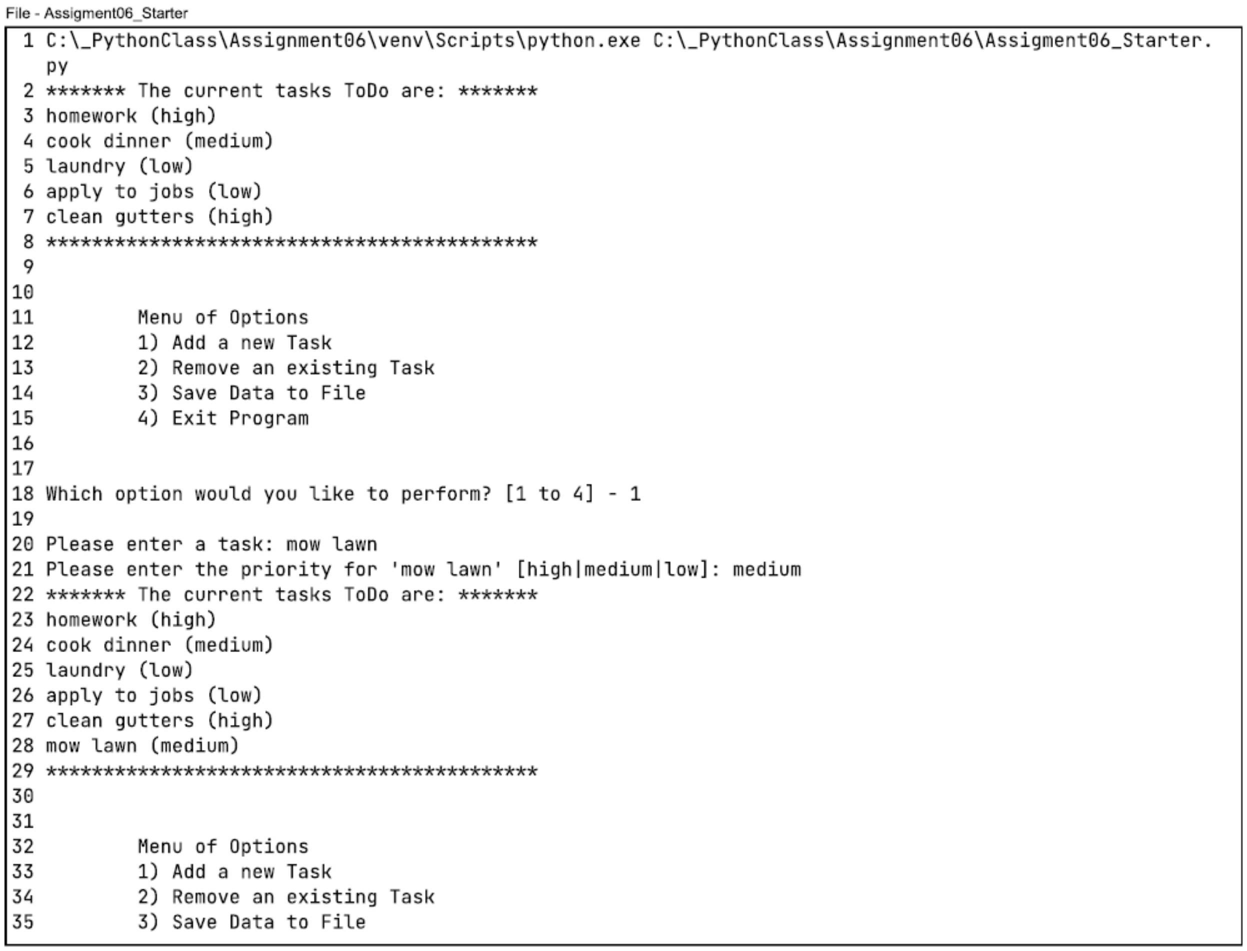
A baseline ToDoFile.txt file was created and populated with a list of tasks and their priorities, as shown in Figure 6.



```
homework,high
cook dinner,medium
laundry,low
apply to jobs,low
clean gutters,high
```

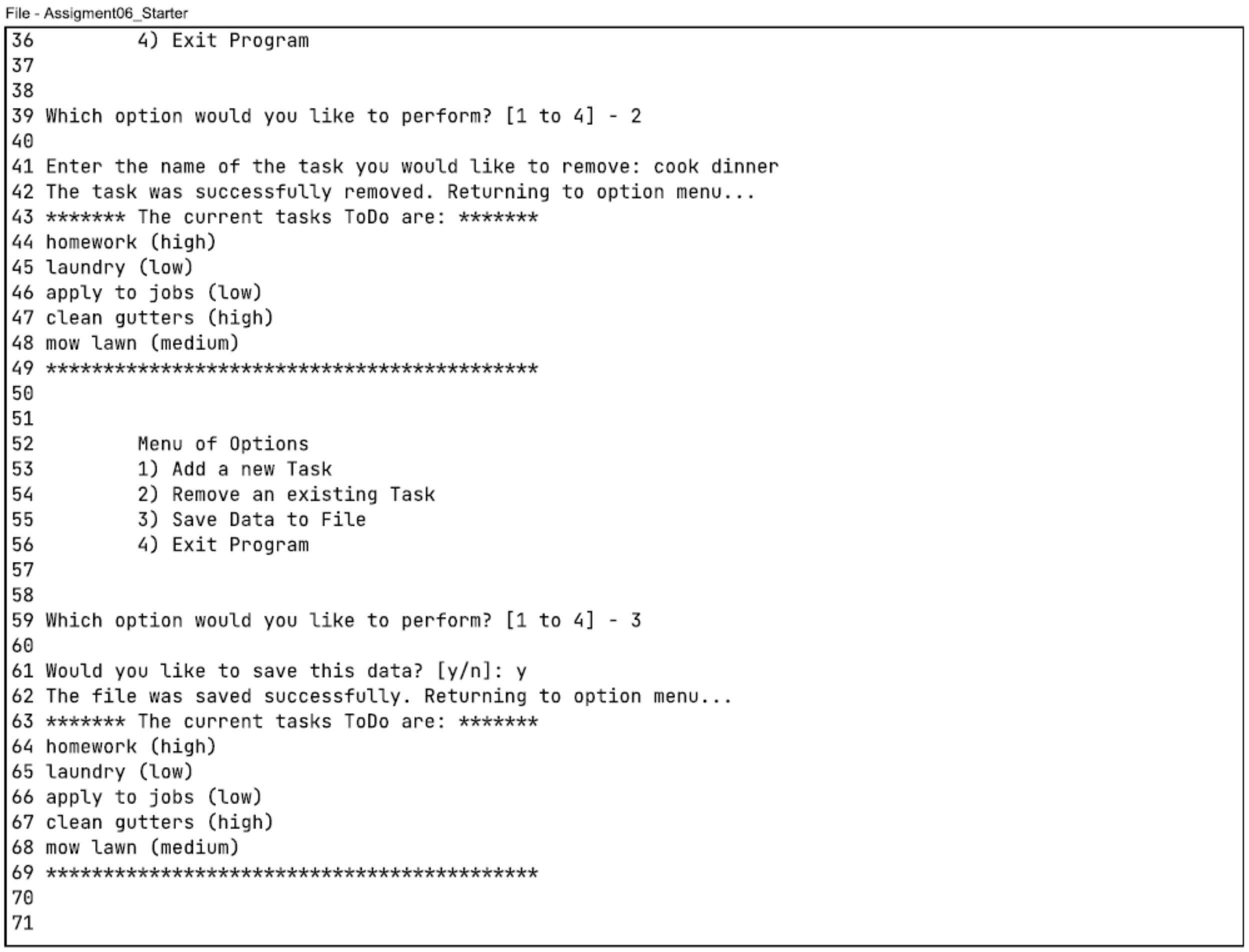
Figure 6. Original baseline ToDoFile.txt file

Figure 7.1 through 7.3 below show a print out of the PyCharm console as the script is run in PyCharm.


File - Assigment06_Starter

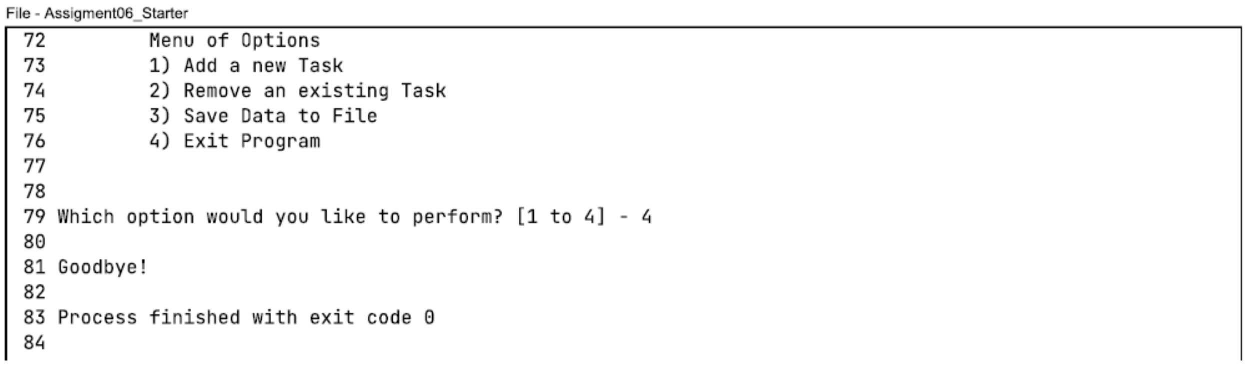
```
C:\_PythonClass\Assignment06\venv\Scripts\python.exe C:\_PythonClass\Assignment06\Assigment06_Starter.
py
******* The current tasks ToDo are: *******
homework (high)
cook dinner (medium)
laundry (low)
apply to jobs (low)
clean gutters (high)
*******************************************


        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

Please enter a task: mow lawn
Please enter the priority for 'mow lawn' [high|medium|low]: medium
******* The current tasks ToDo are: *******
homework (high)
cook dinner (medium)
laundry (low)
apply to jobs (low)
clean gutters (high)
mow lawn (medium)
*******************************************


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

Figure 7.1. Script running in PyCharm (1 of 3)



```
File - Assigment06_Starter
        4) Exit Program


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

Enter the name of the task you would like to remove: cook dinner
The task was successfully removed. Returning to option menu...
******* The current tasks ToDo are: *******
homework (high)
laundry (low)
apply to jobs (low)
clean gutters (high)
mow lawn (medium)
*******************************************


        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

Would you like to save this data? [y/n]: y
The file was saved successfully. Returning to option menu...
******* The current tasks ToDo are: *******
homework (high)
laundry (low)
apply to jobs (low)
clean gutters (high)
mow lawn (medium)
*******************************************


```

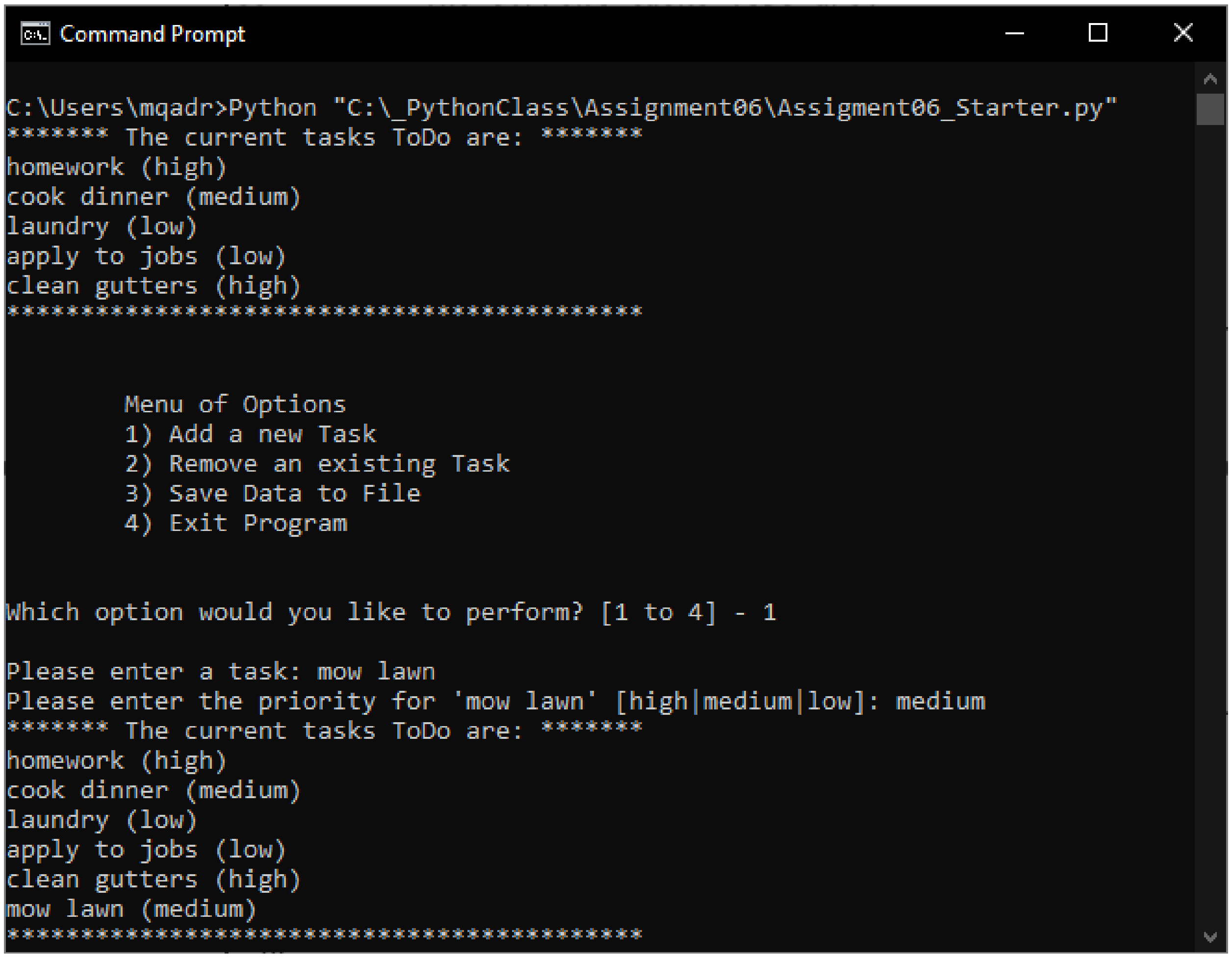
Figure 7.2. Script running in PyCharm (2 of 3)



```
File - Assigment06_Starter
        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] - 4

Goodbye!

Process finished with exit code 0

```

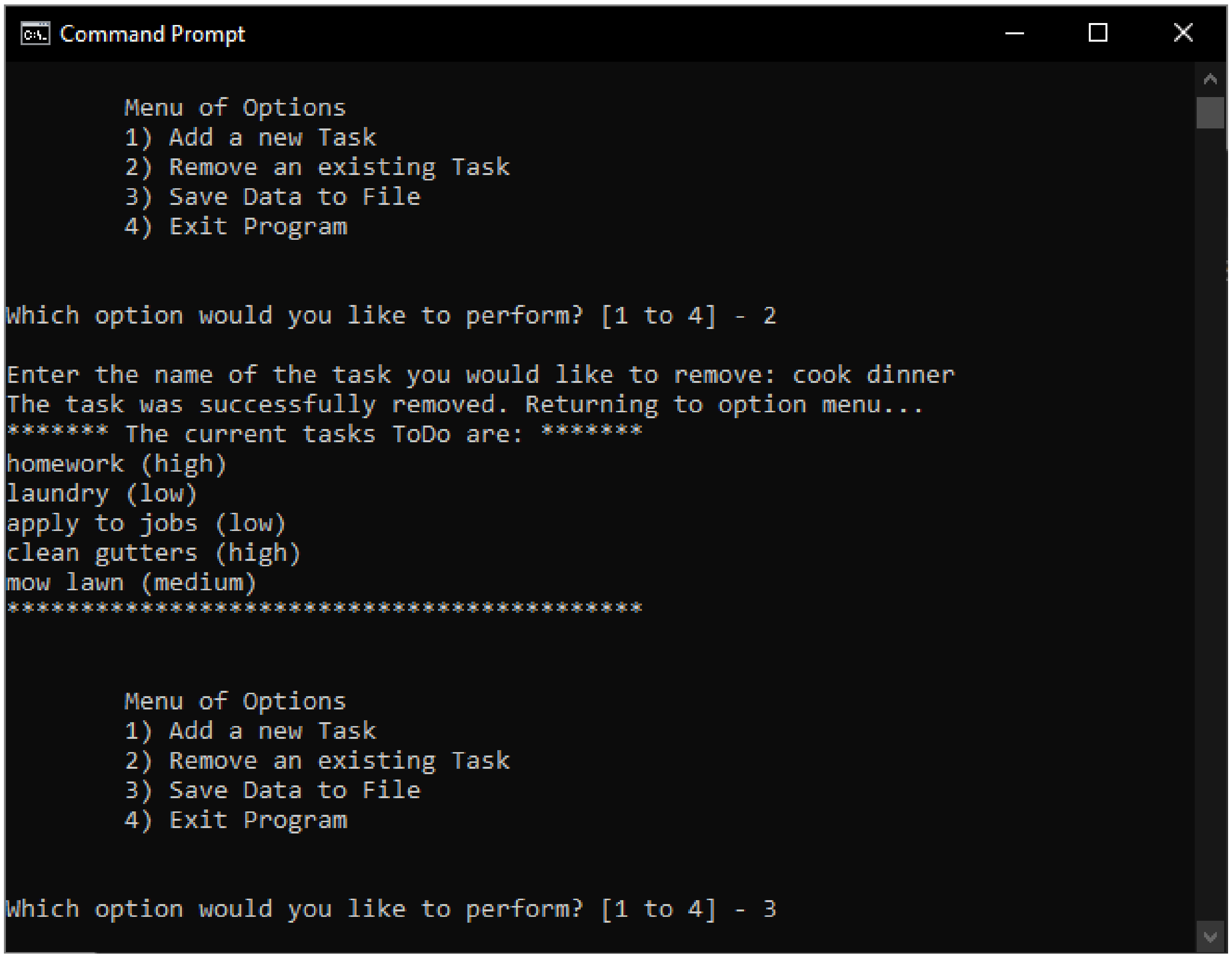
Figure 7.3. Script running in PyCharm (3 of 3)

Figures 8.1 through 8.3 below show the script running in the command window, using the same baseline file and inputs as when run in PyCharm.



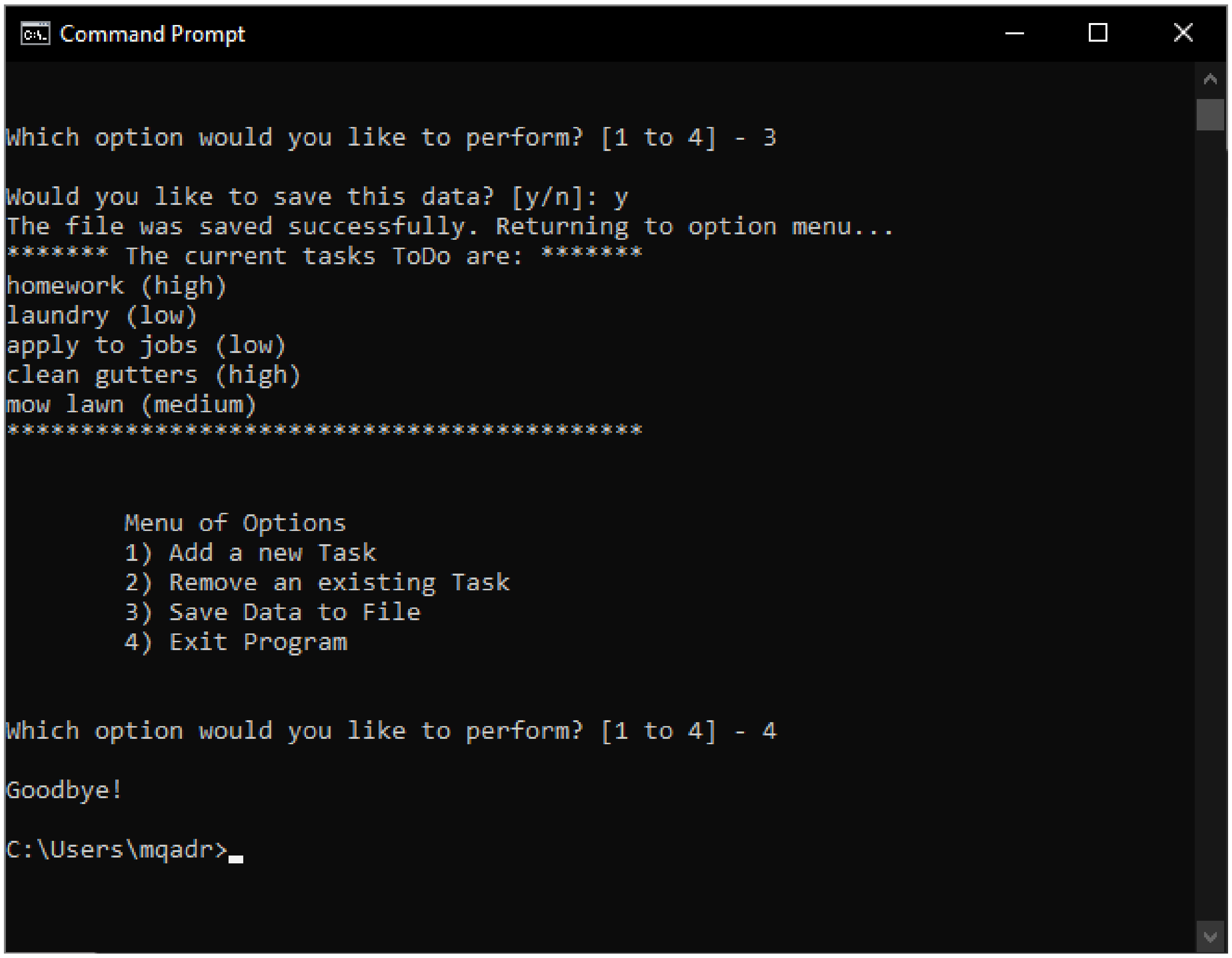
```
C:\Users\mqadr>Python "C:\_PythonClass\Assignment06\Assigment06_Starter.py"
******* The current tasks ToDo are: *******
homework (high)
cook dinner (medium)
laundry (low)
apply to jobs (low)
clean gutters (high)
*******************************************


        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

Please enter a task: mow lawn
Please enter the priority for 'mow lawn' [high|medium|low]: medium
******* The current tasks ToDo are: *******
homework (high)
cook dinner (medium)
laundry (low)
apply to jobs (low)
clean gutters (high)
mow lawn (medium)
*******************************************
```

Figure 8.1. Script running in command window (1 of 3)



```
        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

Enter the name of the task you would like to remove: cook dinner
The task was successfully removed. Returning to option menu...
******* The current tasks ToDo are: *******
homework (high)
laundry (low)
apply to jobs (low)
clean gutters (high)
mow lawn (medium)
*******************************************



        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
```

Figure 8.2. Script running in command window (2 of 3)

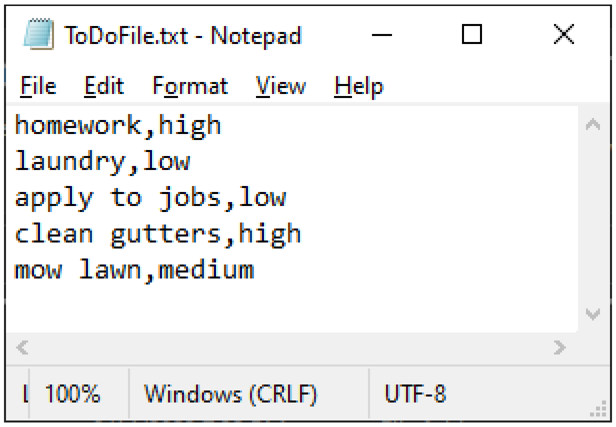


Command Prompt

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

Would you like to save this data? [y/n]: y
The file was saved successfully. Returning to option menu...
******* The current tasks ToDo are: *******
homework (high)
laundry (low)
apply to jobs (low)
clean gutters (high)
mow lawn (medium)
*******************************************


        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] - 4

Goodbye!

C:\Users\mqadr>_
```

Figure 8.3. Script running in command window (3 of 3)

Figure 9 below shows the ToDoFile.txt file that is output as a result of running the script as shown above using the baseline file shown in Figure 6. As can be seen in Figures 6 through 9, the output for each user-input command as well as the script overall are as expected.



```
homework,high
laundry,low
apply to jobs,low
clean gutters,high
mow lawn,medium
```

Figure 9. Output ToDoFile.txt file

Conclusion

The objectives of this assignment were achieved. The script was successfully written with all required functions modified as needed to generate the correct outputs. Familiarity was gained with functions, classes of functions, and their nuances. Further familiarity was gained with GitHub repositories and websites that will help with future assignments as well as personal use. Overall, the assignment was a good learning experience and provided further example of useful rudimentary functions that can be written in Python.