Richard Sauer

May 19, 2023

Foundations of Programming: Python

Assignment 06

GitHub: [richsau/IntroToProg-Python-Mod06 (github.com)]

Functions

Introduction

The goal of this assignment was to modify an existing program and add functionality to existing functions that edit data from a file called “ToDoFile.txt” in the form of Tasks and Priorities. As data is read in, it’s converted to a dictionary and added to a table in memory. A menu is then presented to the user allowing them to:

1) Add a new task
2) Remove an existing task
3) Save data to file
4) Exit program

The menu and related function calls are presented in a while(True) loop that only exist if the user selects the “Exit program” menu item. For each of the function calls, code needs to be added to complete each task. The functions are all contained in one of two classes. One called “Processor” that does the work of reading and writing to the file and the data structures, and one called “IO” that handles printing the menu items and getting input from the user.

Declaring Variables

The application came with several variables already defined. I added an additional one for keeping track of unsaved changes to the task list. (Figure 1)

```
12  # Data ---------------------------------------------------------------------- #
13  # Declare variables and constants
14  file_name_str = "ToDoFile.txt"  # The name of the data file
15  file_obj = None  # An object that represents a file
16  row_dic = {}  # A row of data separated into elements of a dictionary {Task,Priority}
17  table_lst = []  # A list that acts as a 'table' of rows
18  choice_str = ""  # Captures the user option selection
19  bolDirtyFlag = False  # Flag to keep track of unsaved changes
20
```

Figure 1. Pre-existing variables, along with one addition.

The Processor Class Functions

The first of the Processor class functions is ***"read_data_from_file()"***. This function takes in a file name, and the current list of rows of data and returns the modified list of rows. It first clears out the list of rows, then opens the data file and reads each row of data, creating a dictionary for each row before adding it to the list of rows. It then returns the list of rows. (Figure 2)

```
25      @staticmethod
26      def read_data_from_file(file_name, list_of_rows):
27          """ Reads data from a file into a list of dictionary rows
28
29          :param file_name: (string) with name of file:
30          :param list_of_rows: (list) you want filled with file data:
31          :return: (list) of dictionary rows
32          """
33          list_of_rows.clear()  # clear current data
34          file = open(file_name, "r")
35          for line in file:
36              task, priority = line.split(",")
37              row = {"Task": task.strip(), "Priority": priority.strip()}
38              list_of_rows.append(row)
39          file.close()
40          return list_of_rows
```

Figure 2. The read_data_from_file function.

Next is the function, "add_data_to_list". This function takes in a task, priority, and list of rows as input. It then creates a dictionary row and appends it to the list of rows before returning the list. (Figure 3)

```
    @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()}
        list_of_rows.append(row)
        return list_of_rows
```

Figure 3. The add_data_to_list function.

Next is the function, remove_data_from_list. This function takes in a task and a list of rows. It searches the list of rows for a matching task name. It removes it if found, otherwise it lets the user know it wasn't found. It returns the list of rows, along with a flag to indicate if there were changes to the list. (Figure 4)

```
    @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, (bool) dirty flag
        """
        bolFoundItem = False
        for row in list_of_rows:
            if (row["Task"].lower() == task.lower()):
                list_of_rows.remove(row)
                print("Task \"{0}\" has been removed.".format(task))
                bolFoundItem = True
        if not (bolFoundItem):
            print("Sorry, task \"{0}\" was not found.".format(task))
        return list_of_rows, bolFoundItem
```

Figure 4. The remove_data_from_list function.

Finally, there's the write_data_to_file function. This function takes as input the filename and the list of rows. It opens a file for writing, then writes each row of data to the file before closing it. It then lets the user know the file was saved and then returns the list of rows. (Figure 5)

```
73      @staticmethod
74      def write_data_to_file(file_name, list_of_rows):
75          """ Writes data from a list of dictionary rows to a File
76
77          :param file_name: (string) with name of file:
78          :param list_of_rows: (list) you want filled with file data:
79          :return: (list) of dictionary rows
80          """
81          objFile = open(file_name, "w")
82          for row in list_of_rows:
83              objFile.write(str(row["Task"]) + ',' + str(row["Priority"]) + '\n')
84          objFile.close()
85          print("Data saved in file \"{0}\".".format(file_name))
86          return list_of_rows
```

Figure 5. The write_data_to_file function.

The IO Class Functions

The first of these functions is output_menu_tasks. It does not take any input and returns nothing. This function simply displays the application's menu. (Figure 6)

```python
    @staticmethod
    def output_menu_tasks():
        """  Display a menu of choices to the user

        :return: nothing
        """
        print('''
        Menu of Options
        1) Add a new task
        2) Remove an existing task
        3) Save data to file
        4) Exit program
        ''')
        print()  # Add an extra line for looks

```

Figure 6. The output_menu_tasks function.

Next is the input_menu_choice function. It takes no input. The function asks the user to select a choice from the previously printed menu and then returns that value. (Figure 7)

```python
    @staticmethod
    def input_menu_choice():
        """ Gets the menu choice from a user

        :return: string
        """
        choice = str(input("Which option would you like to perform? [1 to 4] - ")).strip()
        print()  # Add an extra line for looks
        return choice

```

Figure 7. The input_menu_choice function.

output_current_tasks_in_list

This function takes the list of rows as input and returns nothing. It sets up a few variables that are used multiple times in the function. It then prints out a nice header for the list and then prints out each row of the list. (Figure 8)

```python
    @staticmethod
    def output_current_tasks_in_list(list_of_rows):
        """ Shows the current Tasks in the list of dictionaries rows

        :param list_of_rows: (list) of rows you want to display
        :return: nothing
        """
        strTaskTxt = "Task"
        strPriorityTxt = "Priority"
        seperatorText = "----------------"
        print("******* The current tasks ToDo are: *******")
        print("{:25} {:20}".format(strTaskTxt, strPriorityTxt))
        print("{:25} {:20}".format(seperatorText, seperatorText))
        for row in list_of_rows:
            print("{:25} {:20}".format(row[strTaskTxt], row[strPriorityTxt]))
        print("*****************************************")
        print()  # Add an extra line for looks
```

Figure 8. The output_current_tasks_in_list function.

input_new_task_and_priority

This function takes no input and returns a task and priority that comes from the user. (Figure 9)

```python
    @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
        """
        strTaskItem = input("Please add a new task: ")
        strTaskPriority = input("Please enter a priority for \"{0}\": ".format(strTaskItem))
        return strTaskItem, strTaskPriority
```

Figure 9. The input_new_task_and_priority function.

input_task_to_remove

This function takes no input and asks the user for a task name to remove. That name is then returned. (Figure 10)

```
148     @staticmethod
149     def input_task_to_remove():
150         """  Gets the task name to be removed from the list
151
152         :return: (string) with task
153         """
154         strTaskItem = input("Please enter the task to be removed: ")
155         return strTaskItem
```

Figure 10. The input_task_to_remove function.

Main Body of the Script

The main body of the script uses all of the functions described above to carry out the various menu items presented to the user. It starts by opening the file to read in the existing task list. It then has a while(True) loop that first prints out the task list, shows the menu, and then gets the menu choice from the user. From there, it uses a series of if/elif commands to call the appropriate functions depending on the user input choice.

Along the way, it sets the “data has changed" flag as appropriate and makes use of it when the user selects to exit the application. (Figure 11)

```python
# Main Body of Script  ------------------------------------------------------ #


# Step 1 - When the program starts, Load data from ToDoFile.txt.
Processor.read_data_from_file(file_name=file_name_str, list_of_rows=table_lst)  # read file data

# Step 2 - Display a menu of choices to the user
while (True):
    # Step 3 Show current data
    IO.output_current_tasks_in_list(list_of_rows=table_lst)  # Show current data in the list/table
    IO.output_menu_tasks()  # Shows menu
    choice_str = IO.input_menu_choice()  # Get menu option

    # Step 4 - Process user's menu choice
    if choice_str.strip() == '1':  # Add a new Task
        task, priority = IO.input_new_task_and_priority()
        table_lst = Processor.add_data_to_list(task=task, priority=priority, list_of_rows=table_lst)
        bolDirtyFlag = True
        continue  # to show the menu

    elif choice_str == '2':  # Remove an existing Task
        task = IO.input_task_to_remove()
        table_lst, bolDirtyFlag = Processor.remove_data_from_list(task=task, list_of_rows=table_lst)
        continue  # to show the menu

    elif choice_str == '3':  # Save Data to File
        table_lst = Processor .write_data_to_file(file_name=file_name_str, list_of_rows=table_lst)
        bolDirtyFlag = False
        continue  # to show the menu

    elif choice_str == '4':  # Exit Program
        if (bolDirtyFlag):
            strChoice = input("You have unsaved data. Are you sure you want to exit? (y/n) ")
            if strChoice.strip().lower() == "y":
                break  # exit the program
            else:
                continue  # abort exit and return to menu
        break  # and Exit the program
# end of Main Body
```

Figure 11. The main body of the script.

Testing the Results

In testing the application, it worked as expected using sample input data. I first ran the application from within the PyCharm IDE using each menu item. (Figures 9 - 12)

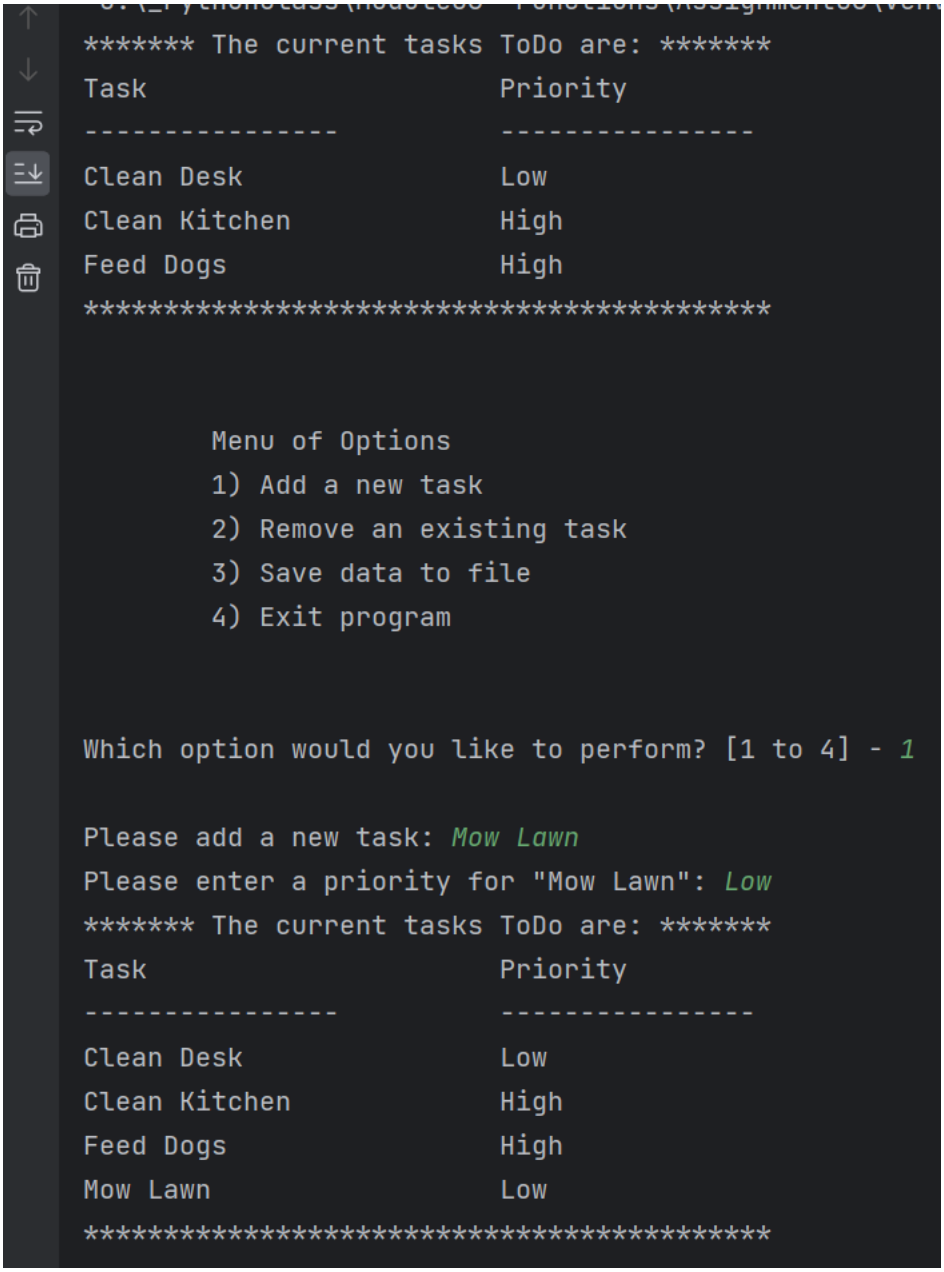

Figure 12. Add a new task.

```
******* The current tasks ToDo are: *******
Task                    Priority
---------------         ---------------
Clean Desk              Low
Clean Kitchen           High
Feed Dogs               High
Mow Lawn                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

Please enter the task to be removed: clean desk
Task "clean desk" has been removed.
******* The current tasks ToDo are: *******
Task                    Priority
---------------         ---------------
Clean Kitchen           High
Feed Dogs               High
Mow Lawn                Low
*****************************************
```

Figure 13. Remove an existing task. Note that case doesn't matter here.

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

You have unsaved data. Are you sure you want to exit? (y/n) n
******* The current tasks ToDo are: *******
Task                    Priority
---------------         ---------------
Clean Kitchen           High
Feed Dogs               High
Mow Lawn                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 in file "ToDoFile.txt".
******* The current tasks ToDo are: *******
Task                    Priority
---------------         ---------------
Clean Kitchen           High
Feed Dogs               High
Mow Lawn                Low
*********************************************
```

Figure 14. Trying to exit before saving and then saving the current data.

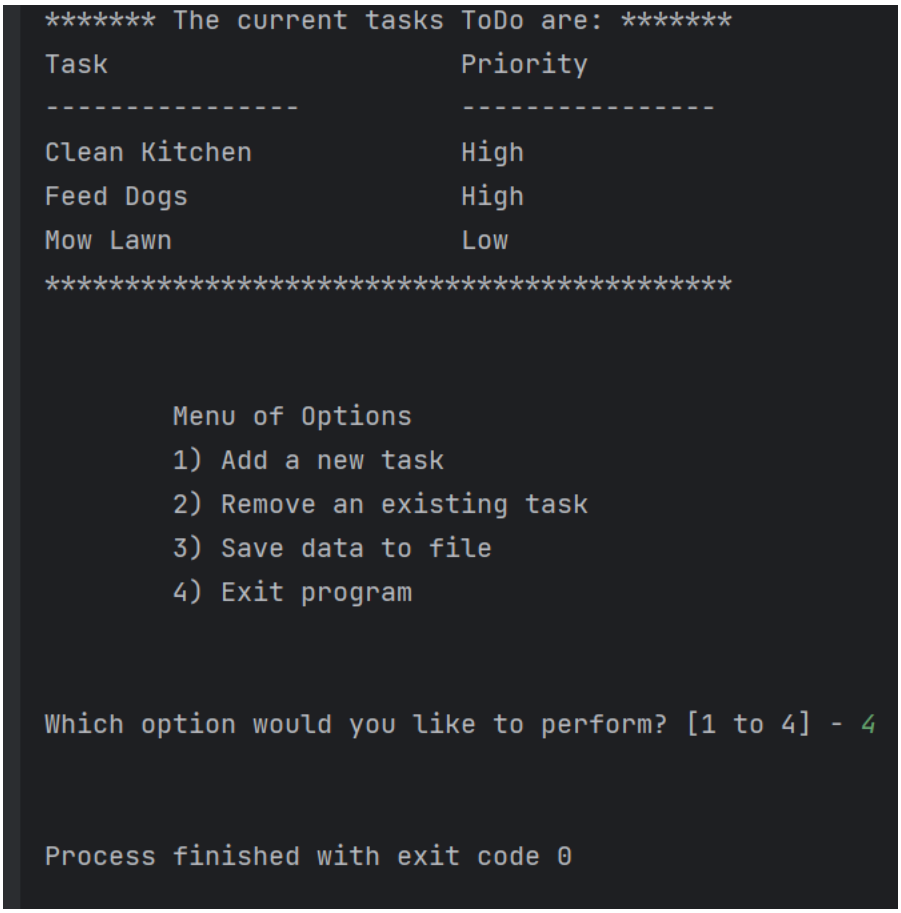

Figure 15. Exiting the program after saving the data.

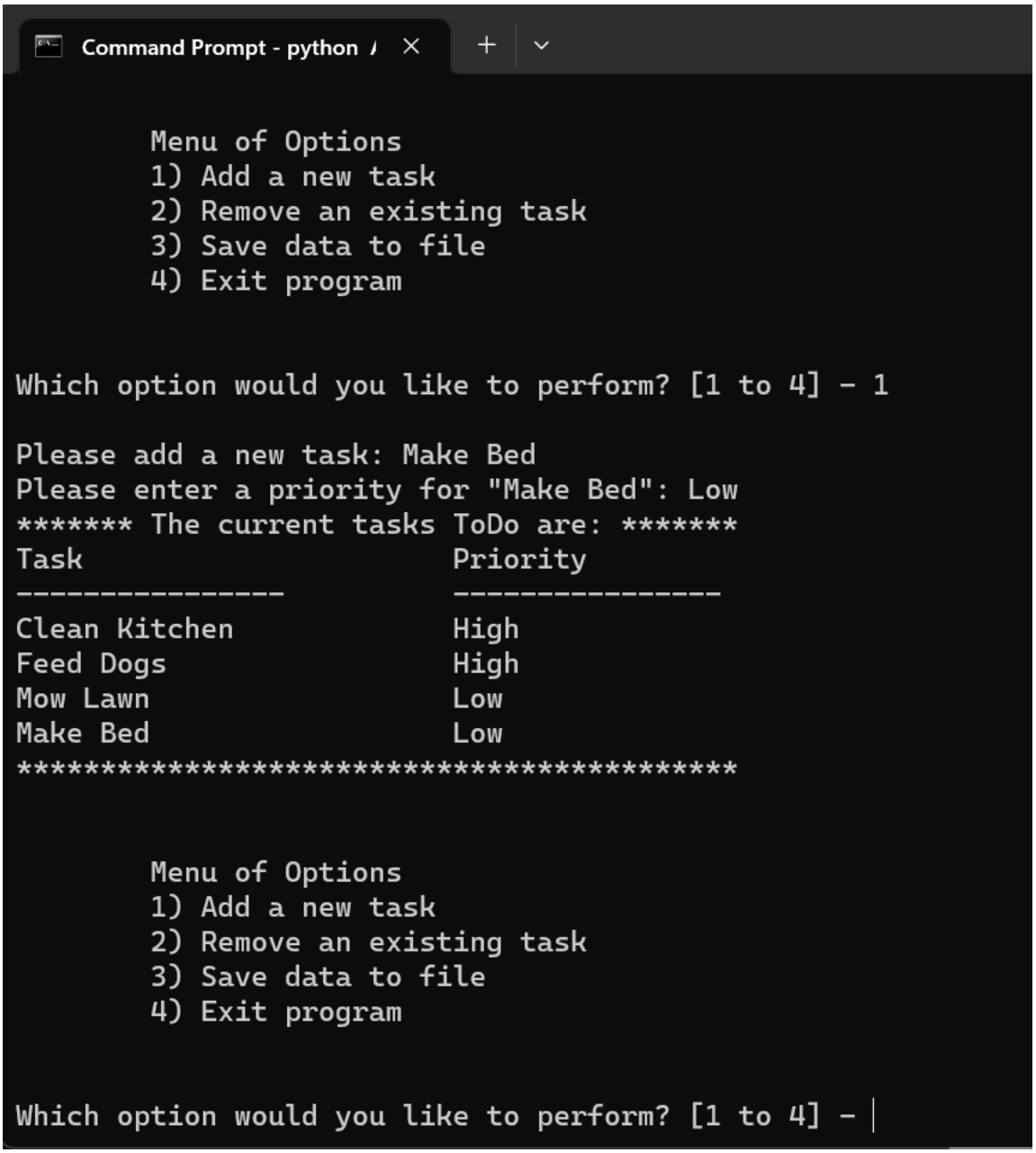

Figure 16. Testing the application from a command prompt.

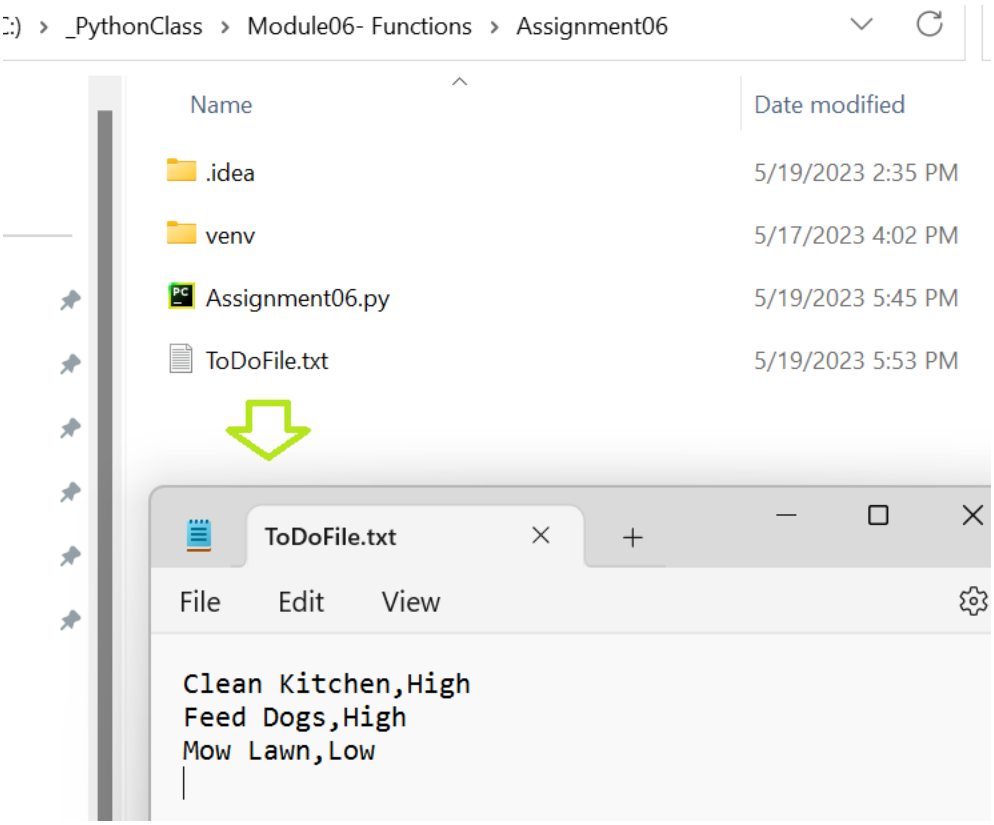

Figure 17. Verifying the contents of the data file.

Figure 14: Verifying results in the output file.

Summary

The goal of this assignment was to modify an existing program and add functionality to existing functions that edit data from a file called “ToDoFile.txt” in the form of Tasks and Priorities. My job was to implement those functions.

I then tested the application in both PyCharm and the command line and verified the contents of the data file.