Jessica Chen

November 19, 2022

Foundations Of Programming: Python

Assignment 06

GitHub URL: https://github.com/jesscUW/IntroToProg-Python-Mod06

Modify a script that manages a To Do List

Introduction
In this assignment, I will document how I modified a Python script that displays a menu of choices to a user for managing their To-Do list. The code loads data from an existing text file into a Python list of dictionary objects when the program starts. Then the user can add new items, remove items, and also save the data to the text file.

Completing the Script

Creating a New Project in PyCharm
To begin with this assignment, I created a new project in PyCharm with the folder location "_PythonClass/Assignment06" (Figure 1).

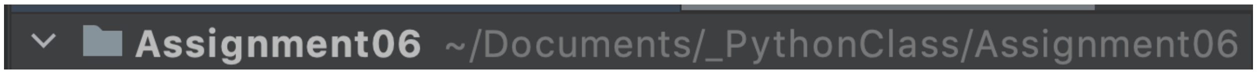

Assignment06 ~/Documents/_PythonClass/Assignment06

Figure 1: Creating a new project in PyCharm

Adding the Starter File
I opened the starter file "Assignment06_Starter.py" and saved it as "Assignment06.py" within my project (Figure 2).

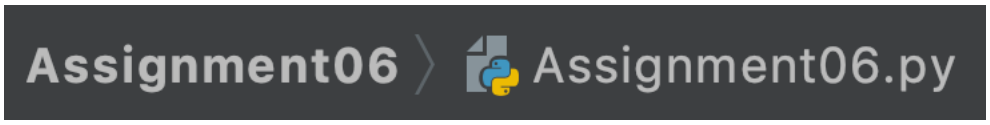
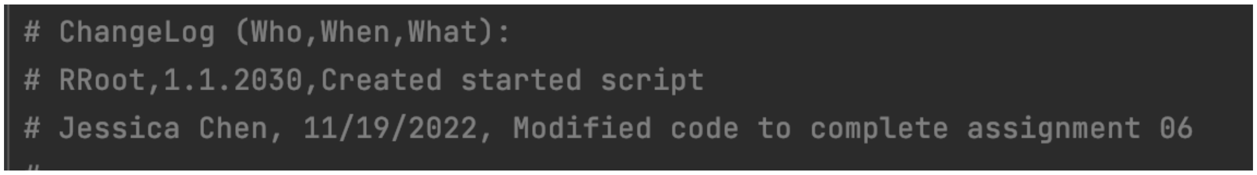

Assignment06 〉 Assignment06.py

Figure 2: Adding the starter file and saving it as "Assignment06.py" in the project

Creating "ToDoList.txt"
When the program starts, the code would load data from a text file named "ToDoList". I created "ToDoList.txt" with a row of data "Wash dishes, Low" (Task, Priority) for this step.

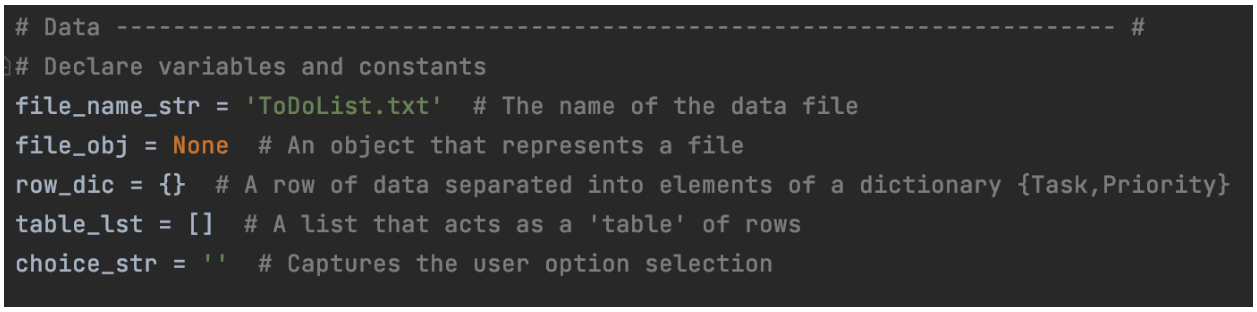
Script Header
After all the folder and files needed were created, I started to work on the code. The first thing I did was to update the change log in the header (Figure 3).



```
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# Jessica Chen, 11/19/2022, Modified code to complete assignment 06
```

Figure 3: Screenshot of the updated change log in the header

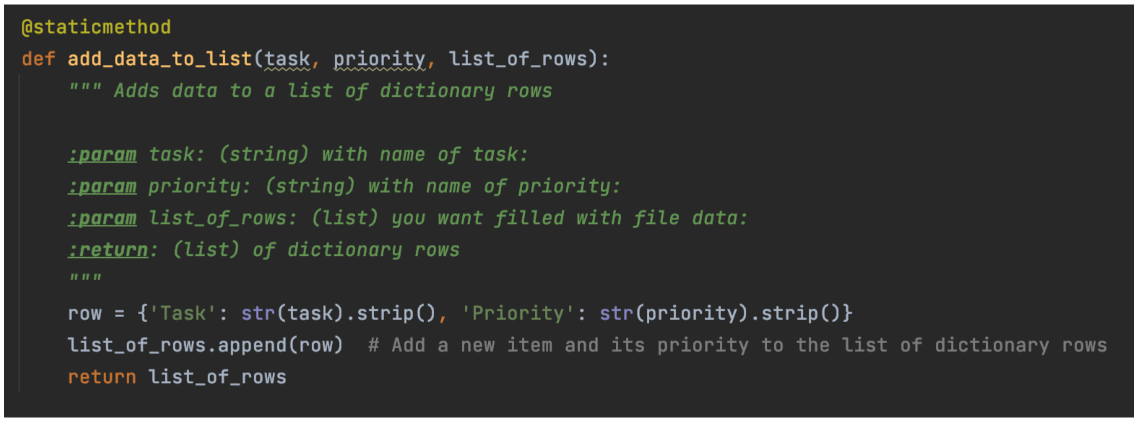
Declaring variables
There are four sections in this program: “Data”, “Processing”, “Presentation (Input/Output)” and “Main body of script”. In Data section, all the variables and constants were declared. I only changed the name of the data file from “ToDoFile.txt” to “ToDoList.txt” (Figure 4).



```
# Data -------------------------------------------------------------------- #
# Declare variables and constants
file_name_str = 'ToDoList.txt'  # The name of the data file
file_obj = None  # An object that represents a file
row_dic = {}  # A row of data separated into elements of a dictionary {Task,Priority}
table_lst = []  # A list that acts as a 'table' of rows
choice_str = ''  # Captures the user option selection
```

Figure 4: Declaring the variables

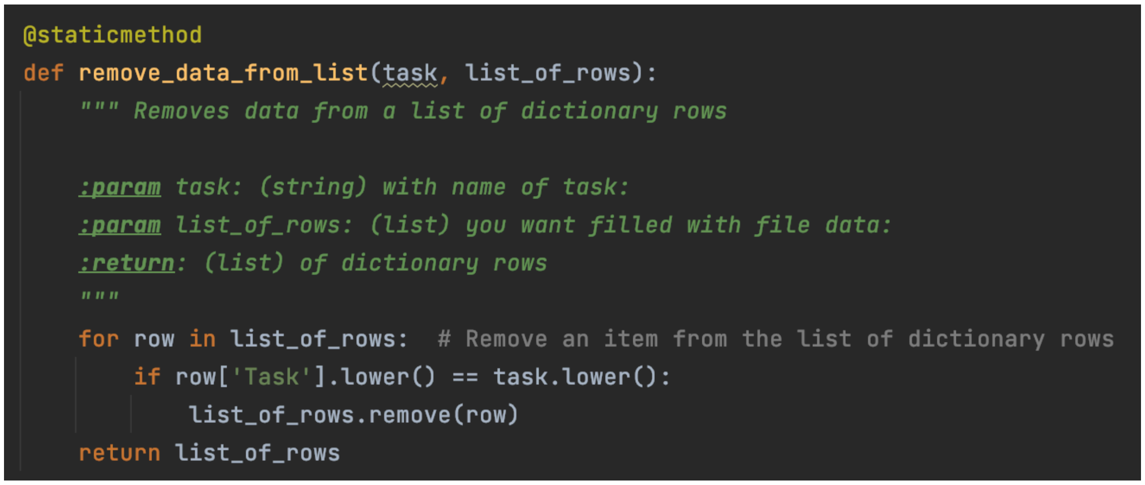
Processing
A class “Processor” was created for the Processing section. There are four functions included in it. The first one is “read_data_from_file()”. There was nothing to change so I kept it as is. The second function is “add_data_to_list()”. This function is for adding data to a list of dictionary rows. I used append() method here in order to add an item and its priority entered by the user to the list/table (Figure 5).



```
@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 filled with file data:
    :return: (list) of dictionary rows
    """
    row = {'Task': str(task).strip(), 'Priority': str(priority).strip()}
    list_of_rows.append(row)  # Add a new item and its priority to the list of dictionary rows
    return list_of_rows
```

Figure 5: Adding the append() method to complete the function “add_data_to_list()”

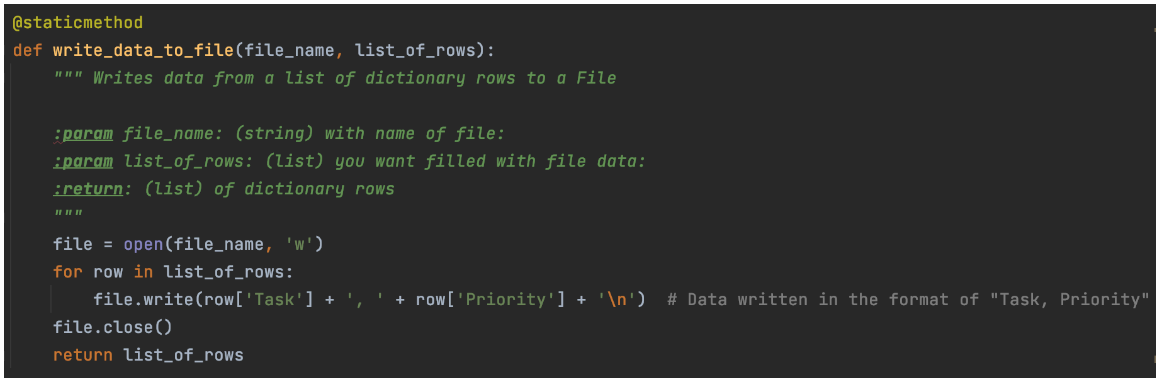
The third function “remove_data_from_list()” is to remove the data from the list/table following the user’s request. A for loop was added and a nested if statement was put in the loop body. If the value of “Task” in a row equals to the user’s input, the row would be removed. To make it not case-sensitive, I used the lower() method to create an all-lowercase-letters version of the string; the remove() method was used to do the removing job (Figure 6).



```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:  # Remove an item from the list of dictionary rows
        if row['Task'].lower() == task.lower():
            list_of_rows.remove(row)
    return list_of_rows
```

Figure 6: A for loop used and a nested if statement put in the loop body to complete the function “remove_data_from_list()”

The last function created in this section is “write_data_to_file()”. This part was written for the option to save the user’s entered data. In order to add the data to a file, the first thing I did was open the file in write mode. Then I used a for loop to format the data shown in the text file–it would look like this: “Task, Priority”. The close() method was added to close the file (Figure 7).

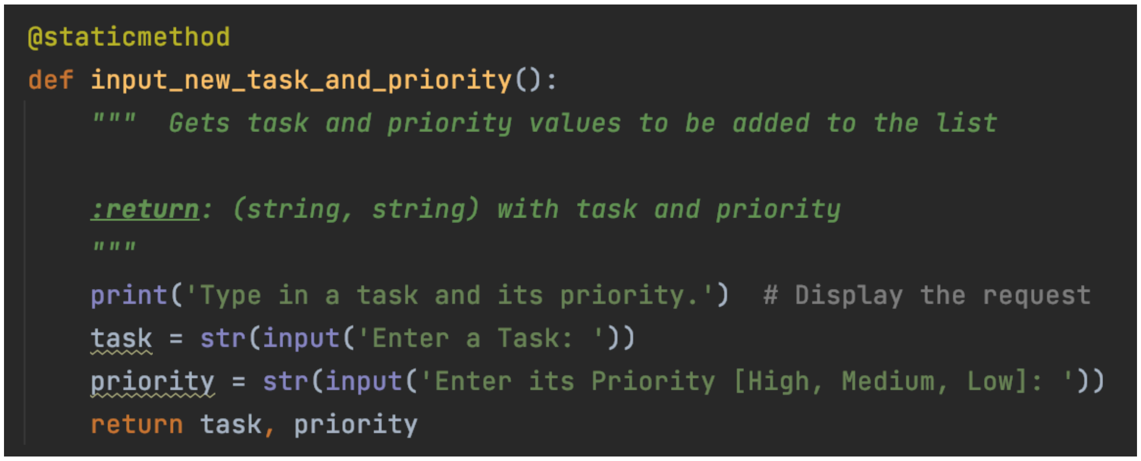


```python
@staticmethod
def write_data_to_file(file_name, list_of_rows):
    """ Writes data from a list of dictionary rows to a File

    :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:
        file.write(row['Task'] + ', ' + row['Priority'] + '\n')  # Data written in the format of "Task, Priority"
    file.close()
    return list_of_rows
```

Figure 7: Screenshot of the complete code for function “write_data_to_file()”

Presentation (Input/Output)

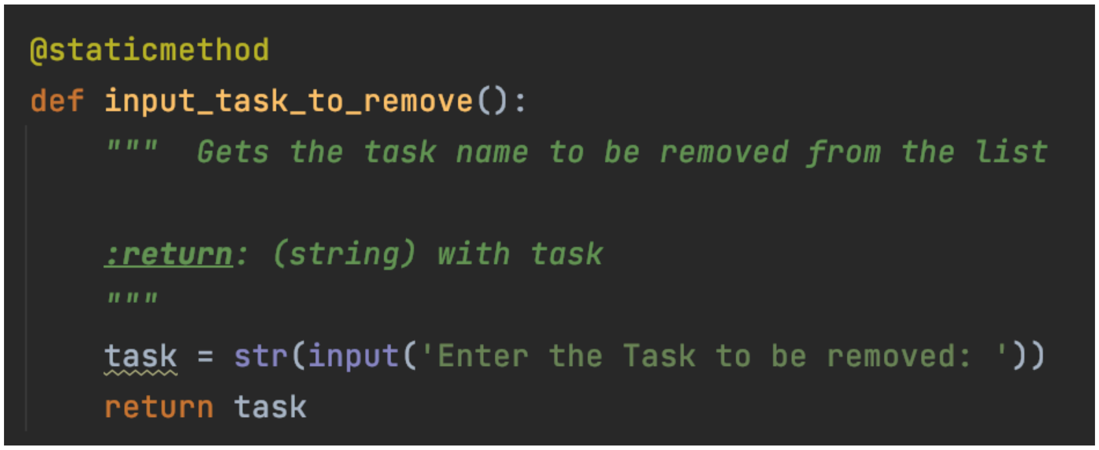
A class “IO” was created in this section. There are five functions included. The first three “output_menu_tasks()”, “input_menu_choice()” and “output_current_tasks_in_list()” worked so I did not change the code. The fourth function is “input_new_task_and_priority()”. I used the print() function to print out the message telling the user what they are going to do. Then two variables “task” and “priority” were named and assigned the user’s input (Figure 8).



```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
    """
    print('Type in a task and its priority.')  # Display the request
    task = str(input('Enter a Task: '))
    priority = str(input('Enter its Priority [High, Medium, Low]: '))
    return task, priority
```

Figure 8: Screenshot of the complete code for function “input_new_task_and_priority()”

The last function in Class IO is “input_task_to_remove()”. I assigned the user’s input to “task”. It would get the task to be removed from the user (Figure 9).

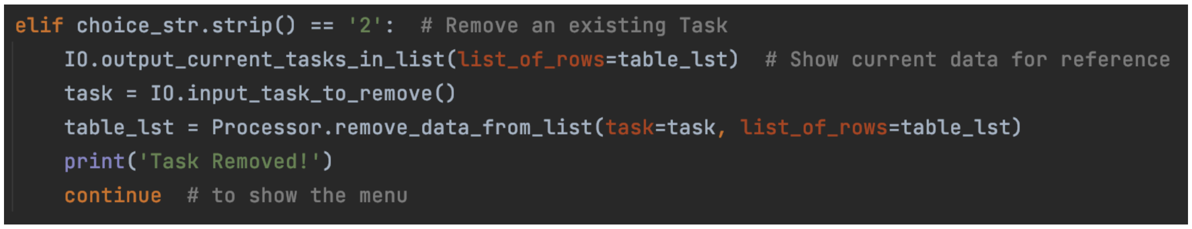


```python
@staticmethod
def input_task_to_remove():
    """  Gets the task name to be removed from the list

    :return: (string) with task
    """
    task = str(input('Enter the Task to be removed: '))
    return task
```

Figure 9: Screenshot of the complete code for function “input_task_to_remove()”

Main Body of Script

I reviewed the code in this section, and it worked correctly. The only two things I modified were adding a line “IO.output_current_tasks_in_list(list_of_rows=table_lst)” and another line “print(‘Task removed!’)” in the elif clause for removing an existing task. They would show current data again to the user for visibility and let the user know that their requested data has been removed (Figure 10).

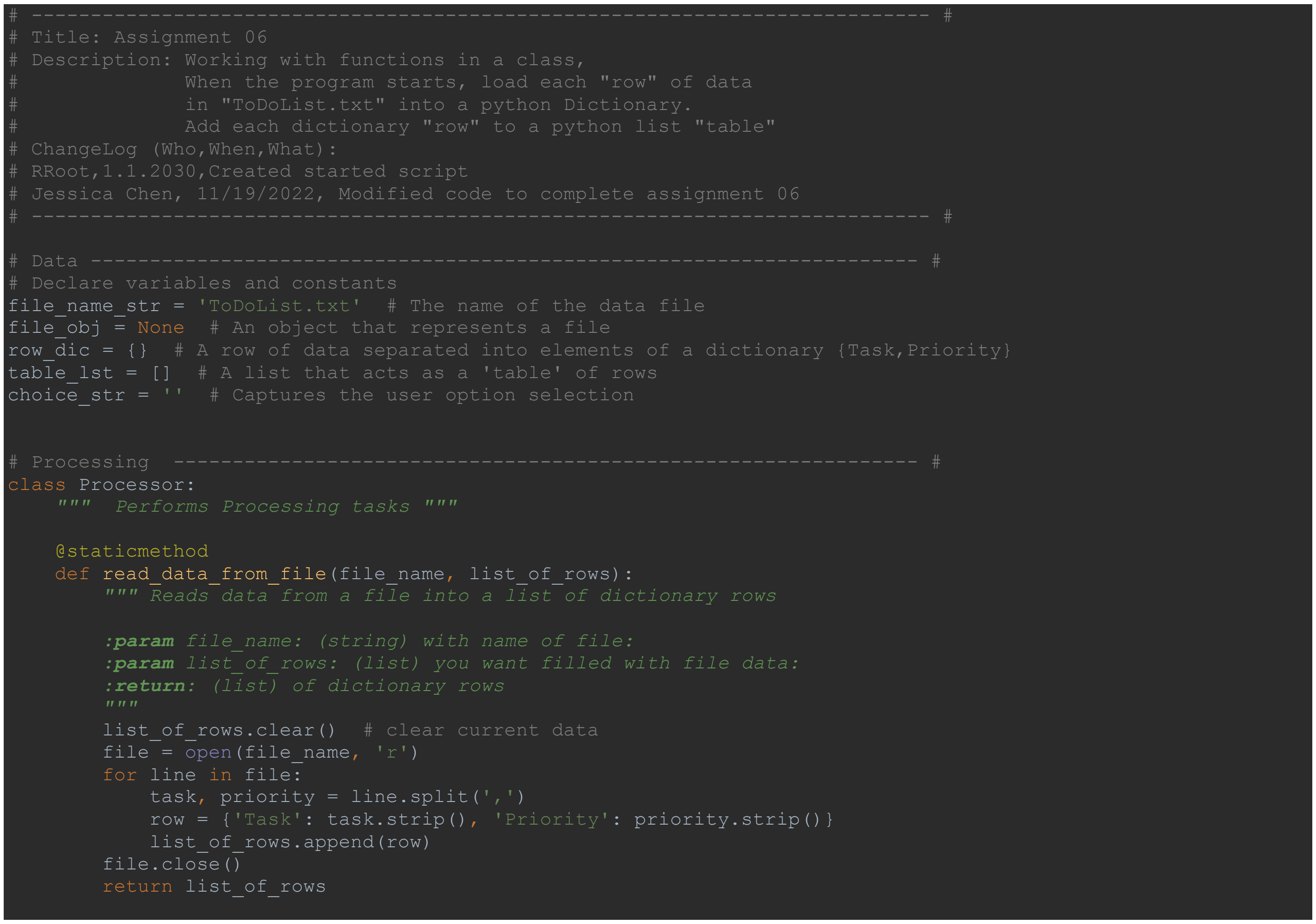


```
elif choice_str.strip() == '2':  # Remove an existing Task
    IO.output_current_tasks_in_list(list_of_rows=table_lst)  # Show current data for reference
    task = IO.input_task_to_remove()
    table_lst = Processor.remove_data_from_list(task=task, list_of_rows=table_lst)
    print('Task Removed!')
    continue  # to show the menu
```

Figure 10: Adding “IO.output_current_tasks_in_list(list_of_rows=table_lst)” and “print(‘Task removed!’)”

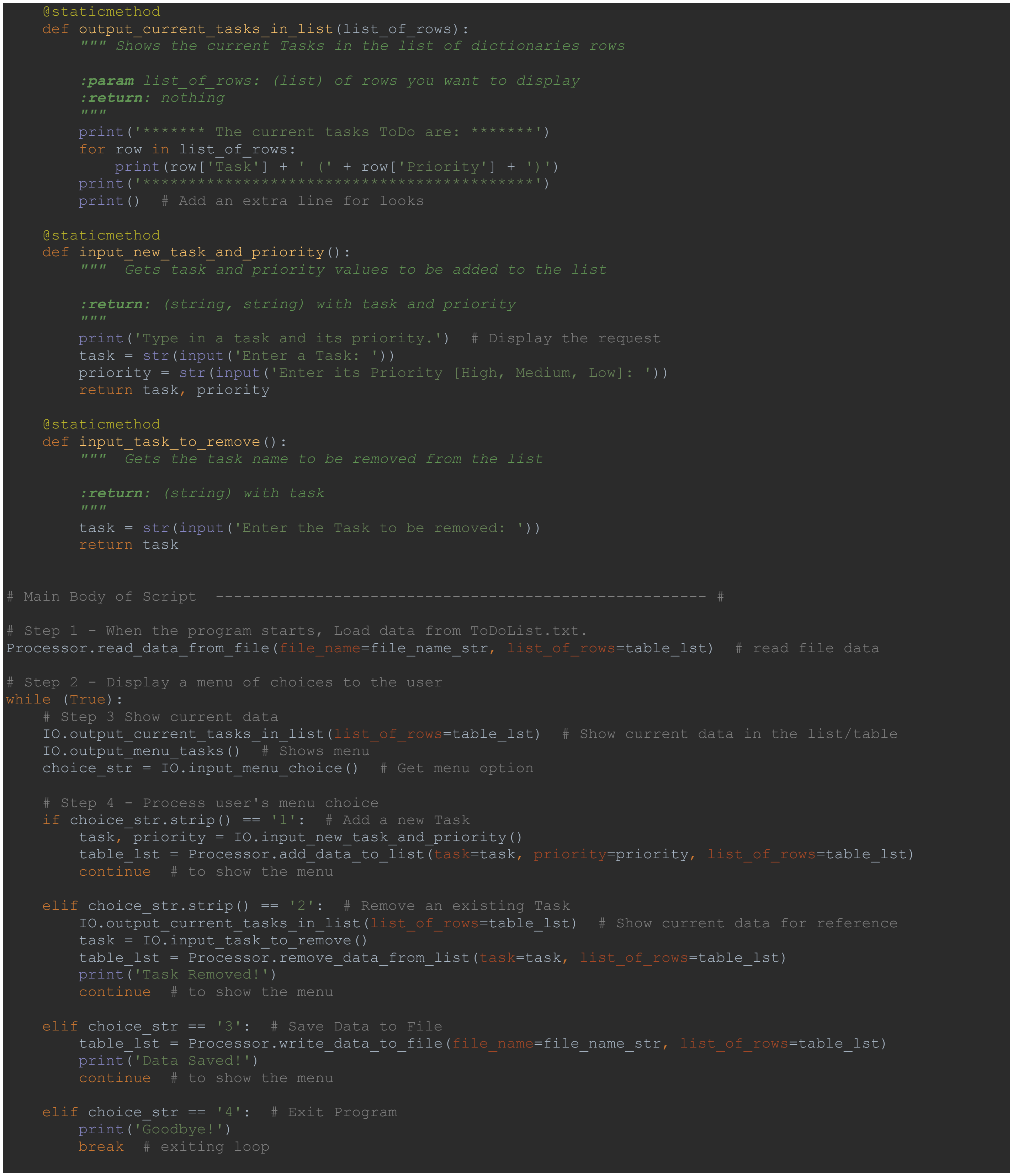
Comments

To help other people understand my code and also be able to remind myself my original intentions in the future, I added some notes. My script is shown as below (Figure 11):



```
# ------------------------------------------------------------------------ #
# Title: Assignment 06
# Description: Working with functions in a class,
#              When the program starts, load each "row" of data
#              in "ToDoList.txt" into a python Dictionary.
#              Add each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# Jessica Chen, 11/19/2022, Modified code to complete assignment 06
# ------------------------------------------------------------------------ #

# Data ----------------------------------------------------------------- #
# Declare variables and constants
file_name_str = 'ToDoList.txt'  # The name of the data file
file_obj = None  # An object that represents a file
row_dic = {}  # A row of data separated into elements of a dictionary {Task,Priority}
table_lst = []  # A list that acts as a 'table' of rows
choice_str = ''  # Captures the user option selection


# Processing  --------------------------------------------------------- #
class Processor:
    """  Performs Processing tasks """

    @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
```

```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 filled with file data:
        :return: (list) of dictionary rows
        """
        row = {'Task': str(task).strip(), 'Priority': str(priority).strip()}
        list_of_rows.append(row)  # Add a new item and its priority to the list of dictionary rows
        return list_of_rows

    @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:  # Remove an item from the list of dictionary rows
            if row['Task'].lower() == task.lower():
                list_of_rows.remove(row)
        return list_of_rows

    @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:
            file.write(row['Task'] + ', ' + row['Priority'] + '\n')  # Data written in the format of "Task,
Priority"
        file.close()
        return list_of_rows


# Presentation (Input/Output)  -------------------------------------------- #
class IO:
    """ Performs Input and Output tasks """

    @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

    @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
```

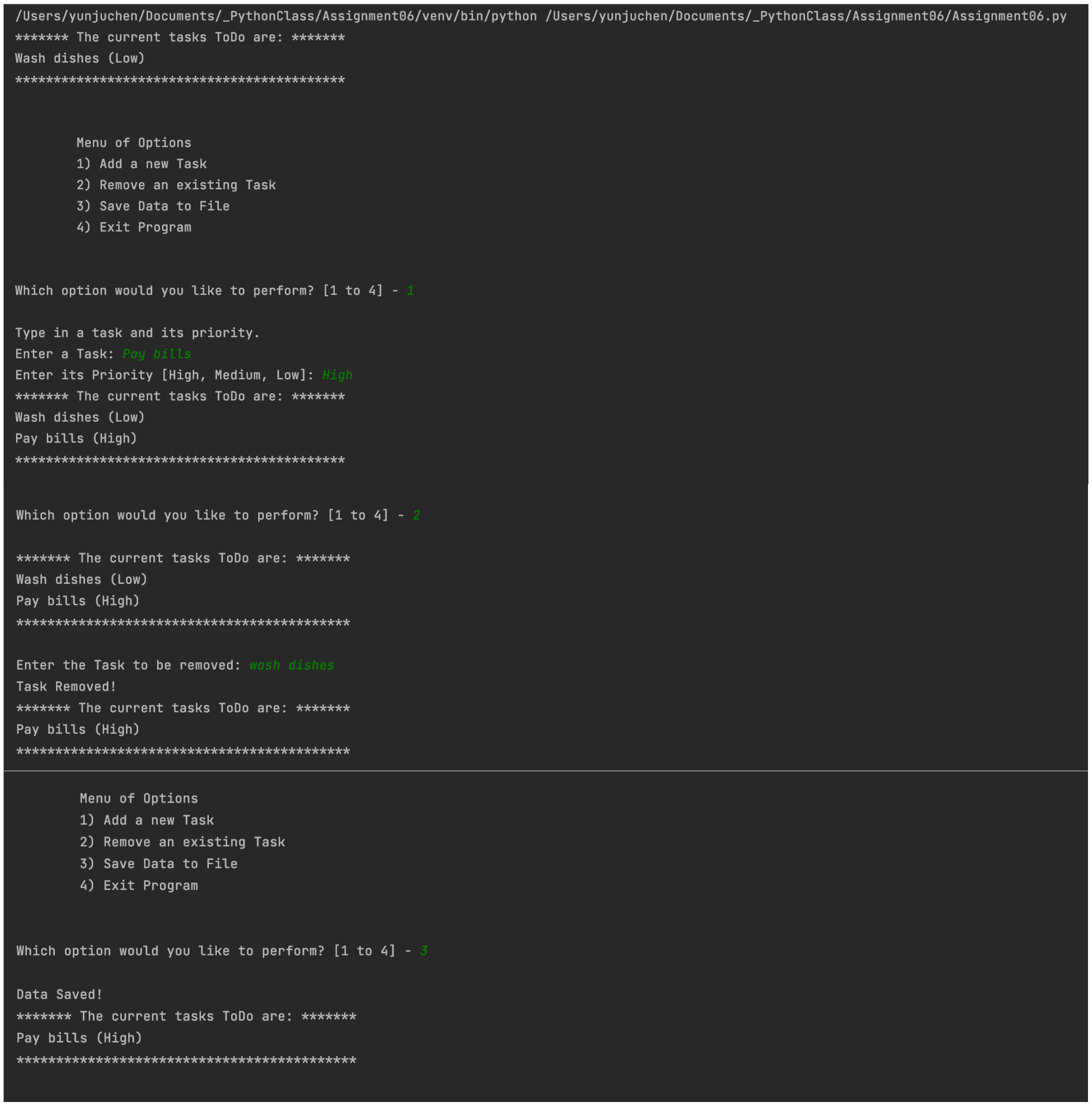


```
    @staticmethod
    def output_current_tasks_in_list(list_of_rows):
        """ Shows the current Tasks in the list of dictionaries rows

        :param list_of_rows: (list) of rows you want to display
        :return: nothing
        """
        print('******* The current tasks ToDo are: *******')
        for row in list_of_rows:
            print(row['Task'] + ' (' + row['Priority'] + ')')
        print('*******************************************')
        print()  # Add an extra line for looks

    @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
        """
        print('Type in a task and its priority.')  # Display the request
        task = str(input('Enter a Task: '))
        priority = str(input('Enter its Priority [High, Medium, Low]: '))
        return task, priority

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

        :return: (string) with task
        """
        task = str(input('Enter the Task to be removed: '))
        return task


# Main Body of Script  ------------------------------------------------------ #

# Step 1 - When the program starts, Load data from ToDoList.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)
        continue  # to show the menu

    elif choice_str.strip() == '2':  # Remove an existing Task
        IO.output_current_tasks_in_list(list_of_rows=table_lst)  # Show current data for reference
        task = IO.input_task_to_remove()
        table_lst = Processor.remove_data_from_list(task=task, list_of_rows=table_lst)
        print('Task Removed!')
        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)
        print('Data Saved!')
        continue  # to show the menu

    elif choice_str == '4':  # Exit Program
        print('Goodbye!')
        break  # exiting loop
```

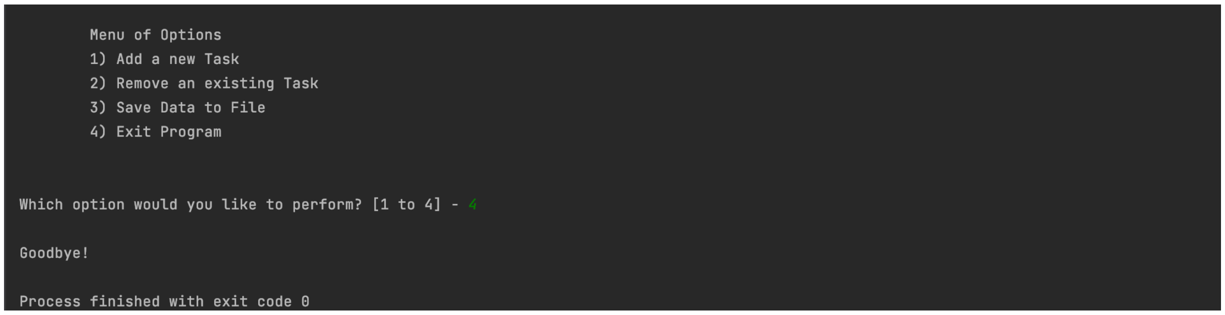
Figure 11: Screenshot of my script

Running the Script

I ran Assignment06.py in both PyCharm and Shell. When the program started, the data "Wash dishes, Low" was correctly loaded as "Wash dishes (Low)" from the "ToDoList.txt". Following the prompt, I chose option 1 to add a task and its priority. The program immediately showed current data of the existing task and the one I added. Then I selected option 2 and entered "wash dishes" to remove this task. It displayed "Task removed!" and also there was no "Wash dishes (Low)" in the list anymore. And then I typed 3 to save the data. Finally, I chose option 4 to exit the program. The script ran as expected (Figure 12 and 13); the data was correctly updated ("Wash dishes" deleted) and written to the text file "ToDoList.txt" (Figure 14).

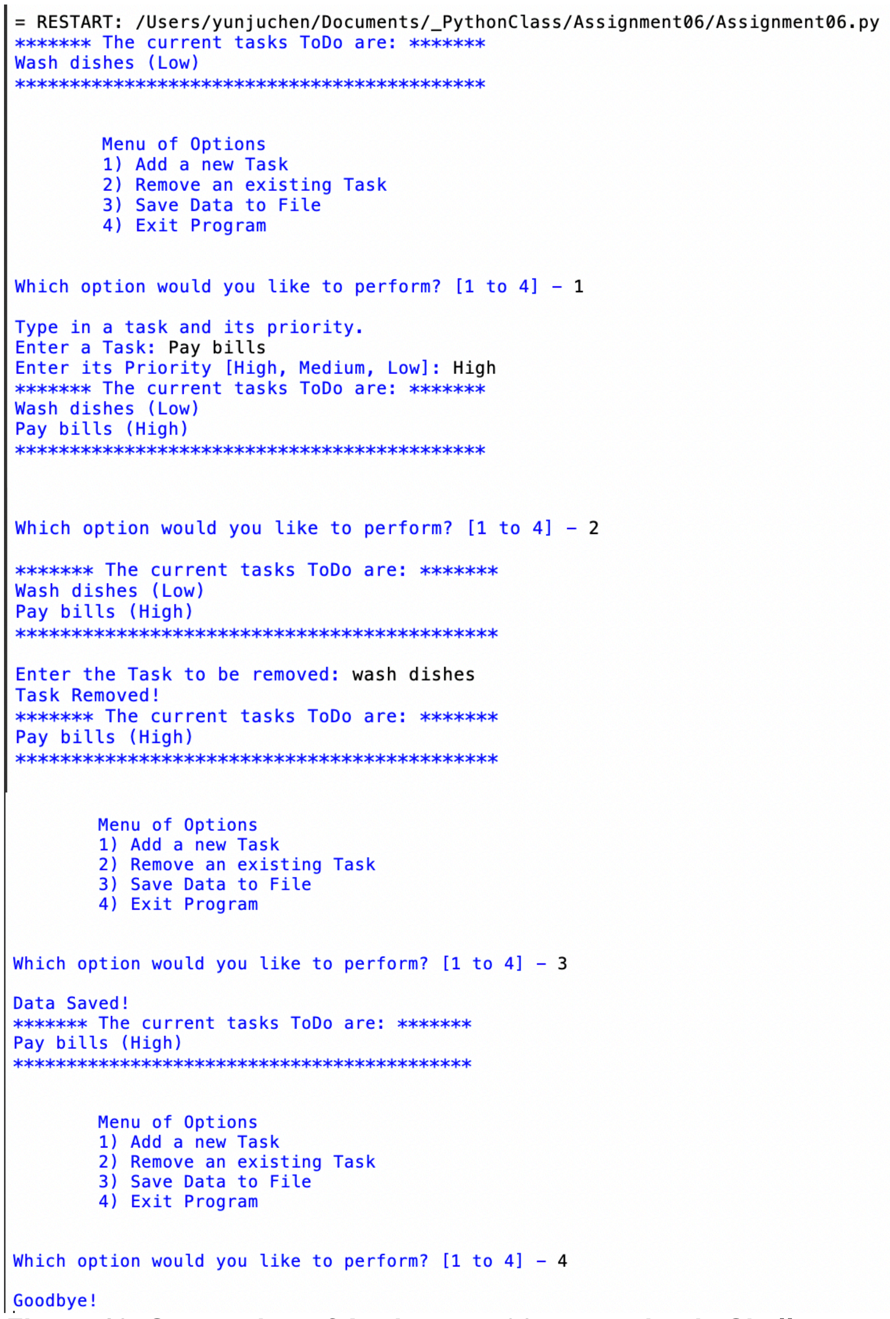


```
/Users/yunjuchen/Documents/_PythonClass/Assignment06/venv/bin/python /Users/yunjuchen/Documents/_PythonClass/Assignment06/Assignment06.py
******* The current tasks ToDo are: *******
Wash dishes (Low)
*******************************************


        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

Type in a task and its priority.
Enter a Task: Pay bills
Enter its Priority [High, Medium, Low]: High
******* The current tasks ToDo are: *******
Wash dishes (Low)
Pay bills (High)
*******************************************


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

******* The current tasks ToDo are: *******
Wash dishes (Low)
Pay bills (High)
*******************************************

Enter the Task to be removed: wash dishes
Task Removed!
******* The current tasks ToDo are: *******
Pay bills (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!
******* The current tasks ToDo are: *******
Pay bills (High)
*******************************************
```



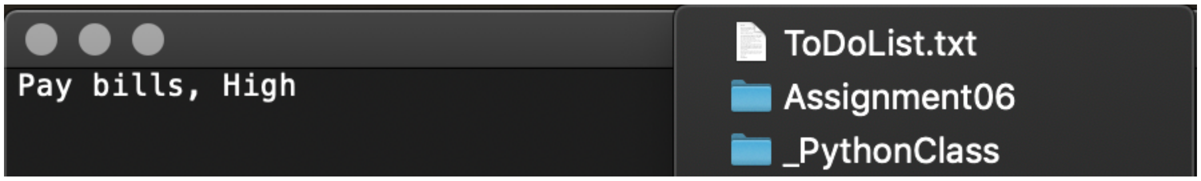
```
        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


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

Goodbye!

Process finished with exit code 0
```

Figure 12: Screenshot of Assignment06.py running in PyCharm



```
= RESTART: /Users/yunjuchen/Documents/_PythonClass/Assignment06/Assignment06.py
******* The current tasks ToDo are: *******
Wash dishes (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

Type in a task and its priority.
Enter a Task: Pay bills
Enter its Priority [High, Medium, Low]: High
******* The current tasks ToDo are: *******
Wash dishes (Low)
Pay bills (High)
*******************************************


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

******* The current tasks ToDo are: *******
Wash dishes (Low)
Pay bills (High)
*******************************************

Enter the Task to be removed: wash dishes
Task Removed!
******* The current tasks ToDo are: *******
Pay bills (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!
******* The current tasks ToDo are: *******
Pay bills (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] - 4

Goodbye!
```

Figure 13: Screenshot of Assignment06.py running in Shell



```
Pay bills, High
```

Figure 14: Verifying that the file has correct data

Summary

Through Module 06, I learned to work with functions and gained an understanding of simple classes. I got to work with a script created by another person and modified it. The program allows a user to choose which action to be performed until they exit the program; the data can be loaded from an existing text file and updated as requested, and then written into the same text file.