Glenn B. Dacones

May 24, 2022

Foundation of Programming: Python

Assignment 06

Github Repository

To Do List – Function Script

Introduction

This paper is a continuation of the to do list script from Assignment 05. In this assignment, using the template provided, I will create and execute a script using functions that are already pre-defined in the template. The codes that I have added are the text following the "TODO: Add Code Here!" comment in the template.

How to Create a Menu of Options

In this step of the script, the codes have already been provided in the template. The codes are consisting of two separate functions, one function is to display the menu of options, and the other is to request for user's choice.

How to Read Data from a File

In this step of the script, the file, ToDoFile.txt must be created first and have two rows of data. The predefined function has been provided in the template as shown below (Figure 1).

```
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 1. An example of a function reading data from a file.

How to Request and Add Data to a List/Table

In this step of the script, the template provided two separate functions, one function is to request the data from the user, and two is the function to add the data to a list/table. Shown below is the first function used (Figure 2).

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

    :return: (string, string) with task and priority
    """
    pass
    # TODO: Add Code Here!
    task = input("Enter a Task: ")
    task = task.capitalize()
    priority = input("Enter a Priority, High, Medium, Low: ")
    priority = priority.capitalize()
    return task, priority
```

Figure 2. An example of a function requesting data from the user.

Below shows the second function used (Figure 3).

```
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()}
    # TODO: Add Code Here!
    table_lst.append(row)
    return list_of_rows
```

Figure 3. An example of a function adding data to a list/table.

How to Remove an Existing Task from a List/Table

Like the previous example, this step of script used two separate functions, one to request which task to remove, two to remove the data from the list/table. Shown below is the first function used (Figure 4).

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

    :return: (string) with task
    """
    pass
    # TODO: Add Code Here!
    task = input("Which task would you like to remove?: ")
    task = task.capitalize()
    return task
```

Figure 4. An example of a function requesting for a task to be removed.

And below show the second function used to remove the data the list/table (Figure 5).

```
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!
    for row in table_lst:
```

```
        if (row["Task"] == task):
            table_lst.remove(row)
            print("\nThe task has been removed from the list/Table!")
    return list_of_rows
```

Figure 5. An example of a function removing the data from the list/table.

How to Save Data to a File

In this step of the script, the data is saved using the predefined function in the template. As mentioned earlier, the codes I have added are the text below the comment, "TODO: Add Code Here!" (Figure 6).

```
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!
    file_obj = open(file_name, "w")
    for row in table_lst:
        file_obj.write(row["Task"] + "," + row["Priority"] + "\n")
    file_obj.close()
    return list_of_rows
```

Figure 6. An example of a function saving data to a file.

How to Exit the Program

In this step of the script, the code has already been provided in template.

Below shows the script running in the command shell (Figure 7).

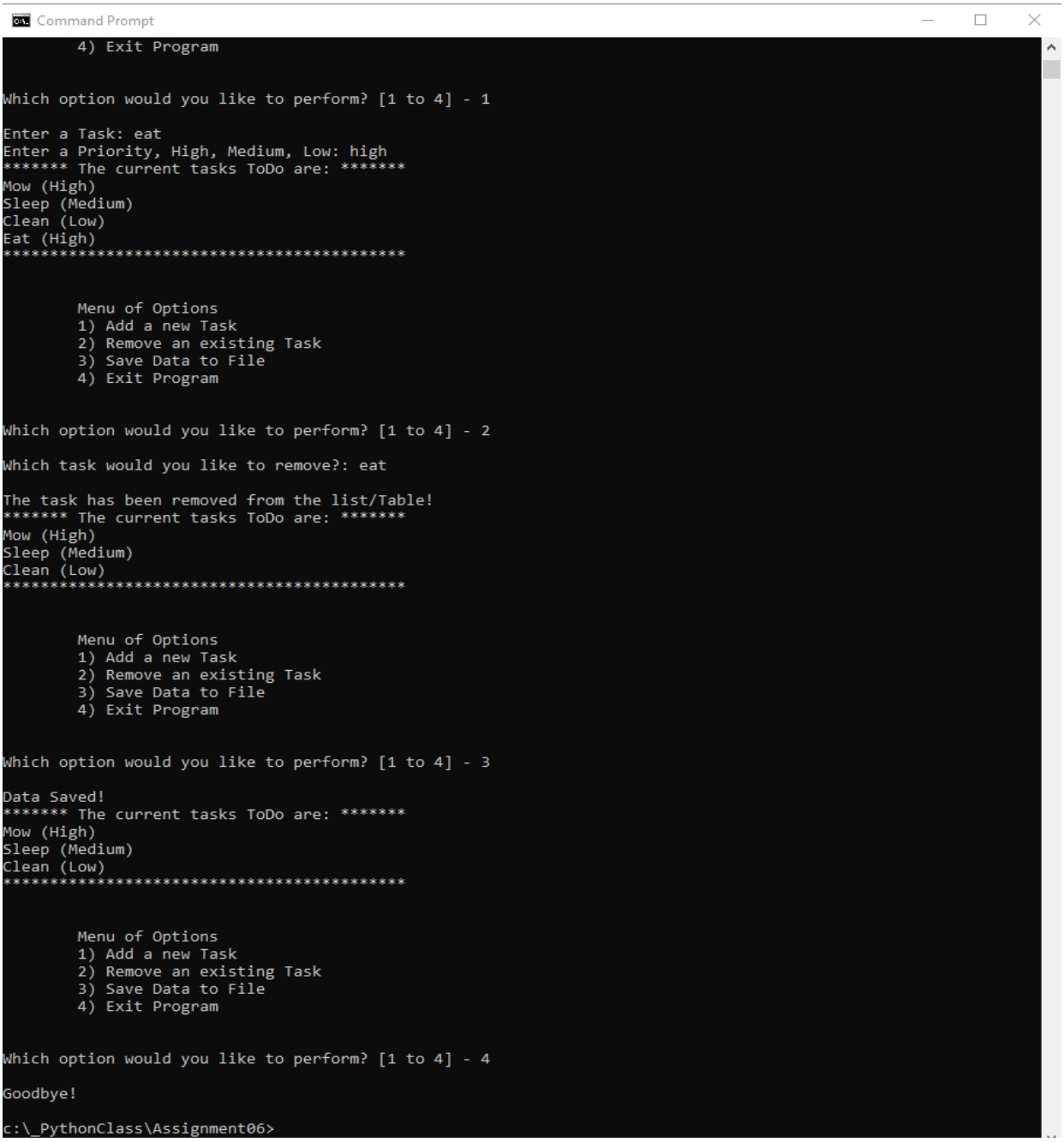

Figure 7. A screenshot of the script running in the command shell.

Below shows the script running in PyCharm (Figure 8).

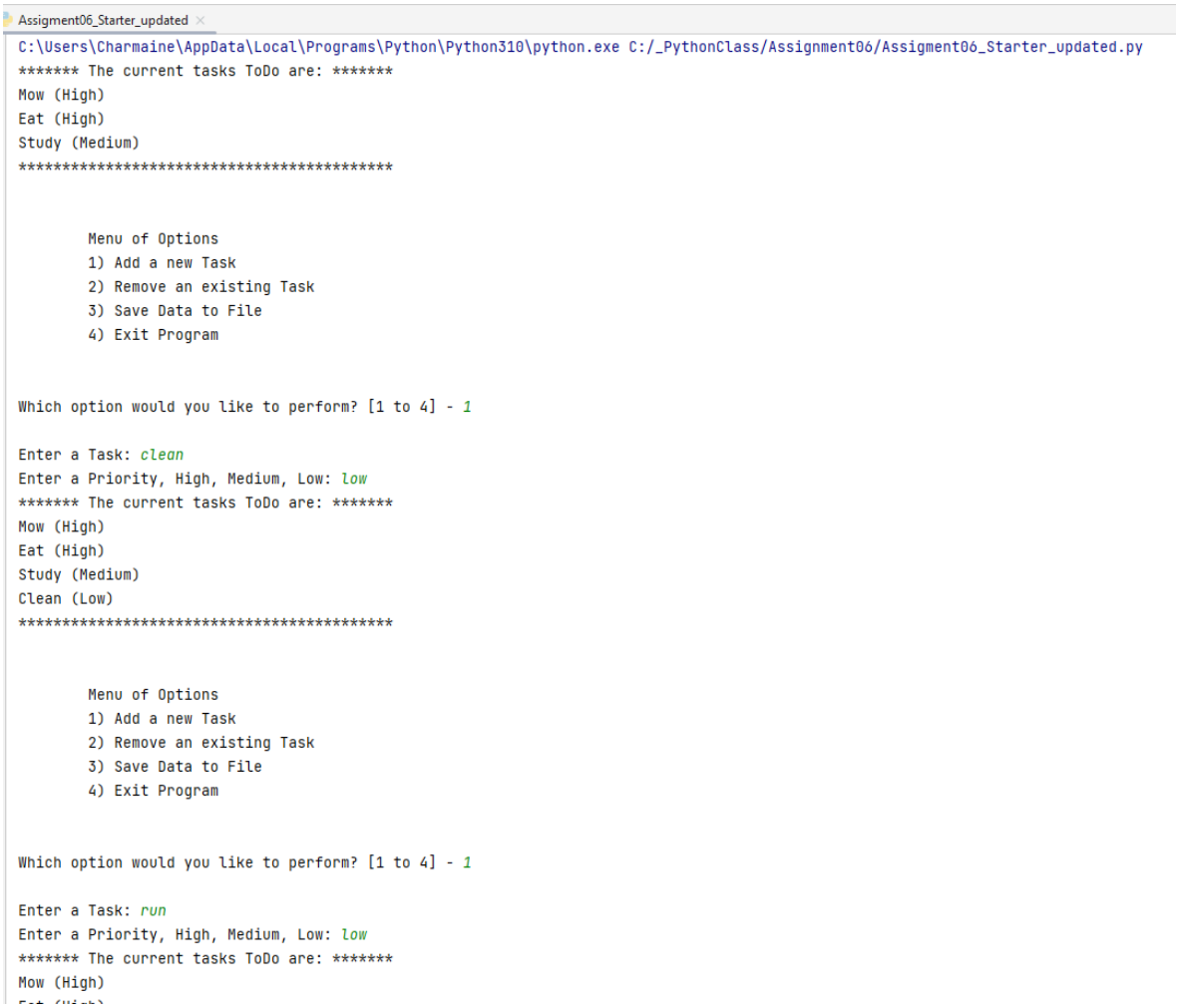

Figure 8. A screenshot of the script running in PyCharm.

Below shows the tasks and priorities written to the ToDoList.text file (Figure 9).

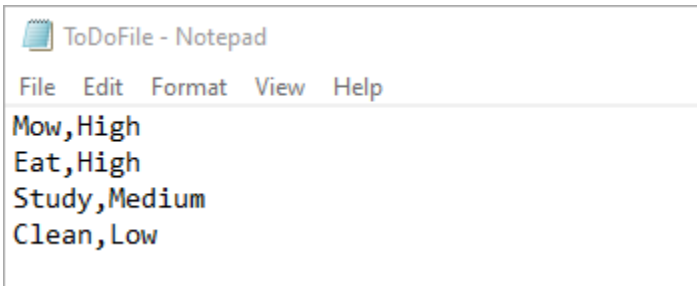

Figure 9. Verifying that the file has data.

Summary

In this assignment, I was able to complete the assigned tasks. Using the provided template with predefined functions, it was confusing at first because the functions were divided into two classes, 1) the processor, 2) the input and output. Also modifying an existing script is much harder than creating a new one. Nonetheless, with perseverance, I successfully completed this assignment.