

**Name:** Marita Quiroz

**Date:** May 24, 2023

**Course:** IT FDN 110 A Sp 23: Foundations Of Programming: Python

**Assignment:** 06

**Github URL:** <https://github.com/mquiroz1/IntroToProg-Python-Mod06>

## Assignment 06 – INTRO TO PROGRAMMING (PYTHON)

### Introduction

The purpose of this assignment is to modify an existing program called Assignment06\_Starter.py. The file loads data from a file into a Python list of dictionary objects. The program starts as incomplete with the functions to be used within. The goal is to modify the program to complete the needed functions to allow the program to have full functionality.

### Program

The header section (Figure 1) provides information about the title, developer name, date of origination, and change log. The change log has been updated to include my name, date, and basic information regarding what was modified or updated.

```
1  # ----- #
2  # Title: Assignment 06
3  # Description: Working with functions in a class,
4  #               When the program starts, load each "row" of data
5  #               in "ToDoToDoList.txt" into a python Dictionary.
6  #               Add the each dictionary "row" to a python list "table"
7  # ChangeLog (Who,When,What):
8  # RRoot,1.1.2030,Created started script
9  # Marita Quiroz,05.24.2023,Modified code to complete assignment 06
10 # ----- #
11
```

Figure 1: Program header

Figure 2 shows the first section proceeding the header, which is where the data is initially handled. The Data section is where the variable declaration will occur. Here, the list, dictionary, and file data related variables are listed.

```

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
20

```

Figure 2: Data section with variable declaration

The Processing section provides space for functions which will be bound to the class, Processing (Figure 3). The first function takes the file name and data list as parameters. The function then reads the data from the file and inputs it into dictionary rows within a list, for further processing within the program. This allows the program to work from memory and not directly from the file, for better efficiency.

```

21 # Processing ----- #
    4 usages
22 class Processor:
23     """ Performs Processing tasks """
24
    1 usage
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 3: Processing class section with functions

The next function (Figure 4) will add task and priority data to dictionary rows within the list.

```

1 usage
42 @staticmethod
43 def add_data_to_list(task, priority, list_of_rows):
44     """ Adds data to a list of dictionary rows
45
46     :param task: (string) with name of task:
47     :param priority: (string) with name of priority:
48     :param list_of_rows: (list) you want to add more data to:
49     :return: (list) of dictionary rows
50     """
51     row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
52     # TODO: Add Code Here!
53     list_of_rows.append(row)
54     return list_of_rows
55

```

Figure 4: Add data to list function

The third function within the Processing class takes the task and list as parameters and removes it from the list of dictionary rows (Figure 5).

```

1 usage
56 @staticmethod
57 def remove_data_from_list(task, list_of_rows):
58     """ Removes data from a list of dictionary rows
59
60     :param task: (string) with name of task:
61     :param list_of_rows: (list) you want filled with file data:
62     :return: (list) of dictionary rows
63     """
64     # TODO: Add Code Here!
65     for row in list_of_rows:
66         if row["Task"].lower() == task.lower():
67             table_lst.remove(row)
68             print("task removed")
69     return list_of_rows
70

```

Figure 5: Remove data from list function

The final function within the Processing section will write the data to the file (Figure 6). The parameters for this function are the file name and the list of data.

```

1 usage
71 @staticmethod
72 def write_data_to_file(file_name, list_of_rows):
73     """ Writes data from a list of dictionary rows to a File
74
75     :param file_name: (string) with name of file:
76     :param list_of_rows: (list) you want filled with file data:
77     :return: (list) of dictionary rows
78     """
79     # TODO: Add Code Here!
80     file = open(file_name, "w")
81     for row in list_of_rows:
82         file.write(row["Task"] + "," + row["Priority"] + "\n")
83     file.close()
84     return list_of_rows
85

```

Figure 6: Write data to file function

The next section within the program is the Presentation section (Figure 7). This is where the IO class is defined. The first function within this class is the menu function. This function displays the list of menu options to the user. The IO class provides functions which focus on data input from the user and clean user interaction on the console and/ or Pycharm when the program is run.

```

87 # Presentation (Input/Output) ----- #
88
89
5 usages
90 class IO:
91     """ Performs Input and Output tasks """
92
1 usage
93     @staticmethod
94     def output_menu_tasks():
95         """ Display a menu of choices to the user
96
97         :return: nothing
98         """
99         print(''
100             Menu of Options
101             1) Add a new Task
102             2) Remove an existing Task
103             3) Save Data to File
104             4) Exit Program
105             '')
106         print() # Add an extra line for looks
107

```

Figure 7: Presentation section and menu function

The next function, in the IO class, prompts the user for a menu choice, and returns the input back to the program as a string (Figure 8).

```

1 usage
108     @staticmethod
109     def input_menu_choice():
110         """ Gets the menu choice from a user
111
112         :return: string
113         """
114         choice = str(input("Which option would you like to perform? [1 to 4] - ")).strip()
115         print() # Add an extra line for looks
116         return choice
117

```

Figure 8: User choice function

Figure 9 shows the next function, which will display the current list of tasks back to the user. It will go through each row within the list, and display each row of data back to the user in a readable format.

```
1 usage
118 @staticmethod
119 def output_current_tasks_in_list(list_of_rows):
120     """ Shows the current Tasks in the list of dictionaries rows
121
122     :param list_of_rows: (list) of rows you want to display
123     :return: nothing
124     """
125     print("***** The current tasks ToDo are: *****")
126     for row in list_of_rows:
127         print(row["Task"] + " (" + row["Priority"] + ")")
128     print("*****")
129     print() # Add an extra line for looks
```

Figure 9: Function to display current list of tasks

The function shown in Figure 10 prompts the user for a task and priority, to be returned to the program for use in the main body of code.

```
1 usage
131 @staticmethod
132 def input_new_task_and_priority():
133     """ Gets task and priority values to be added to the list
134
135     :return: (string, string) with task and priority
136     """
137     pass
138     # TODO: Add Code Here!
139     task = str(input("What is the task? - ")).strip()
140     priority = str(input("What is the priority? - ")).strip()
141     print() # Add an extra line for looks
142     return task, priority
```

Figure 10: Task and Priority prompt for user

The last function within the IO class prompts the user for a task name to be removed.

```
1 usage
143 @staticmethod
144 def input_task_to_remove():
145     """ Gets the task name to be removed from the list
146
147     :return: (string) with task
148     """
149     pass
150     # TODO: Add Code Here!
151     task = str(input("Remove which task? - ")).strip()
152     print() # Add an extra line for looks
153     return task
154
```

Figure 11: Task to remove prompt

Next, begins the main body of the script, where the classes and functions, previously defined in the program, will be used (Figure 12). Step 1 reads the data from the file using the appropriate function from the Processor class. Step 2 will run the IO related menu functions and take input from the user when selecting each choice. Step 2 runs as long as the program is active, as the while loop continues to be true. Step 3 displays the user the menu and prompts the user for their choice. Step 4 will run if the user selects 1, run the add new task function from the IO class, then add the data to the list using the function from the Processor class.

```
156 # Main Body of Script ----- #
157
158
159 # Step 1 - When the program starts, Load data from ToDoFile.txt.
160 Processor.read_data_from_file(_file_name=file_name_str, list_of_rows=table_lst) # read file data
161
162 # Step 2 - Display a menu of choices to the user
163 while (True):
164     # Step 3 Show current data
165     IO.output_current_tasks_in_list(list_of_rows=table_lst) # Show current data in the list/table
166     IO.output_menu_tasks() # Shows menu
167     choice_str = IO.input_menu_choice() # Get menu option
168
169     # Step 4 - Process user's menu choice
170     if choice_str.strip() == '1': # Add a new Task
171         task, priority = IO.input_new_task_and_priority()
172         table_lst = Processor.add_data_to_list(task=task, priority=priority, list_of_rows=table_lst)
173         continue # to show the menu
174
```

Figure 12: Main body of script (Steps 1-4)

Figure 13 shows the continuation of Step 4 with how the other user menu choices are run. If the user selects menu choice 2, the IO function to remove the task is run. This will prompt the user for the task information, and take that data to run the Processor function to then remove the task from the list.

If the user selects 3, the Processor class function is called to write the current list data to the file. The user is displayed a confirmation message that the data was saved.

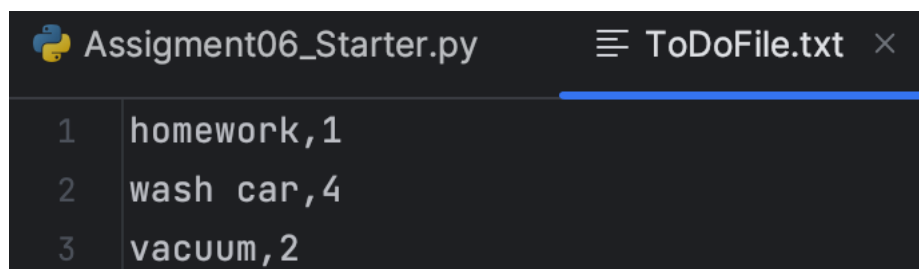
If the user selects menu option 4, the program displays a goodbye message and the while loop is stopped with the break command.

```
174
175     elif choice_str == '2': # Remove an existing Task
176         task = IO.input_task_to_remove()
177         table_lst = Processor.remove_data_from_list(task=task, list_of_rows=table_lst)
178         continue # to show the menu
179
180     elif choice_str == '3': # Save Data to File
181         table_lst = Processor.write_data_to_file(file_name=file_name_str, list_of_rows=table_lst)
182         print("Data Saved!")
183         continue # to show the menu
184
185     elif choice_str == '4': # Exit Program
186         print("Goodbye!")
187         break # by exiting loop
188
```

Figure 13: Second part of main body of script (Continuing step 4)

## Running the Program with PyCharm

Figure 14 shows the initial ToDoFile.txt contents before it was run with PyCharm.



```
Assignment06_Starter.py  ToDoFile.txt x
1 homework,1
2 wash car,4
3 vacuum,2
```

Figure 14: ToDoFile.txt contents before program was run within PyCharm



Figure 15 shows the program as initially run, displaying the current contents of the ToDoFile.txt file back to the user and when the user selects option 1, adding a new task, “wash clothes” with priority 5. The updated list is displayed back to the user.

```
/Users/maritaquiroz/Documents/_PythonClass/Assignment06/venv/bin
***** The current tasks ToDo are: *****
homework (1)
wash car (4)
vacuum (2)
*****

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] - 1w

What is the task? - wash clothes
What is the priority? - 5

***** The current tasks ToDo are: *****
homework (1)
wash car (4)
vacuum (2)
wash clothes (5)
*****

Menu of Options
```

Figure 15: First part of program run within PyCharm and menu choice 1

Figure 16 is the output when the user selects menu choice 3, to save the data to file.

```
*****

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: *****
homework (1)
wash car (4)
vacuum (2)
wash clothes (5)
*****

Menu of Options
```

Figure 16: Menu choice 3 with saving data to file

Figure 17 shows the output of when the user selects menu option 4, to exit the program.

```
*****
```

### 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 17: Menu choice 4

When menu choice 2 is selected, Figure 18 shows the output if the user selects “wash clothes” as the task to be removed. The updated list is displayed back to the user.

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

```
Remove which task? - wash clothes
```

```
task removed
```

```
***** The current tasks ToDo are: *****
```

```
homework (1)
```

```
wash car (4)
```

```
vacuum (2)
```

```
*****
```

Figure 18: Menu choice 2

## Running the Program with the Console

Figure 19 shows the program initially run in the console with the current data displayed to the user. In this example, the user selects menu option 1 and adds another list item that exists from when previously run in PyCharm. The updated list is displayed back to the user.

```

(venv) Maritas-MacBook-Pro:Assignment06 maritaquiroz$ python Assigment06_Starter.py
***** The current tasks ToDo are: *****
homework (1)
wash car (4)
vacuum (2)
wash clothes (5)
*****

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 task? - wash clothes
What is the priority? - 5

***** The current tasks ToDo are: *****
homework (1)
wash car (4)
vacuum (2)
wash clothes (5)
wash clothes (5)
*****

```

Figure 19: Menu choice 1 run in console

Next, the user selects menu choice 3, to save the data to file (Figure 20). Confirmation of data being saved and current list are displayed back to the user.

```
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: *****
```

```
homework (1)
```

```
wash car (4)
```

```
vacuum (2)
```

```
wash clothes (5)
```

```
wash clothes (5)
```

```
*****
```

Figure 20: Menu choice 3

Figure 21 shows the user removing the duplicate task, using menu option 2.

```
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

Remove which item? - wash clothes

task removed
***** The current tasks ToDo are: *****
homework (1)
wash car (4)
vacuum (2)
wash clothes (5)
*****
```

Figure 21: Menu option 2 to remove duplicate task

Figure 22 shows the output of the user exiting the program with menu option 4.

```
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!
(venv) Maritas-MacBook-Pro:Assignment06 maritaquiroz$
```

Figure 22: Menu option 4

## Challenges

The first challenge faced was initially looking at the code to see how it was organized by the previous developer. Familiarizing myself with the code and the different section was important to figure out how to be most efficient with adding the additional functions and knowing how the code and program flowed.

The next challenge was ensuring the variables matched up within the classes for each of the functions being added or modified. The variable declaration area within the Data section was very helpful with seeing the variables at a glance.

## Summary

This program was easy to see the flow with the comments and documentation provided by the previous developer. The setup of the Data, Processing, IO, and main body of the script were very intuitive and it made it easy for me to familiarize myself with how the code runs for modification. For this assignment, I consulted the lecture, labs, the book, and the python documentation on the internet for file manipulation. I also referenced the additional Help-if-needed files provided with the module materials.