Stephanie Miller

November 20, 2022

IT FDN 100

Assignment 6

Git Hub Link:

To Do List Script 2

Intro

In this assignment we are again starting with a partially written code that we need to make changes to. This is also like the last assignment in the goals that we are trying to complete. The way we are going about completing those tasks, however, is very different from what we have done previously. This week we learned how to use and create functions.

Creating the Script

As with every script at the beginning we have the document information and change log. Since this script was started by someone else, we need to add to the change log what we are doing now. In this case we added our name, the date, and that we were modifying to complete the assignment. (Figure 1)

```
# ------------------------------------------------------------------------ #
# 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 the each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# Stephanie Miller, 11.20.2022, Modified code to complete assignment 06
# ------------------------------------------------------------------------ #
```

Figure 1: Document Overview and Change Log

Each section of the code that I made changes too has the #TO DO marker to make it easy to find. The first section of the code that I made modifications too was the part that added new data to the table. Here the row code was already added so we just had to add the append function so we could add the data to the table. (Figure 2)

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

Figure 2: Adding New Data to the Table

The next section of the code that require modification was to remove data from the list. For this code we have a for statement that will run through each row in the code. In the for statement is an if statement. The if statement will compare each rows task to the task that user entered to have removed. If the row matches, then the row will be removed from the list. If the task entered by the user was not in the list, then it will run through the same code but will not remove any rows. (Figure 3)

```
@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!
    for row in list_of_rows:
        if row["Task"].lower() == task.lower():
            list_of_rows.remove(row)
    return list_of_rows
```

Figure 3: Removing a Row from the Table

In this next section we need to add code that will write the data in the list to the file. This will allow us to save any changes for future use. The first part of the code is opening the file we want to write the data to. Next, we have a for loop which will run through each row in the list performing the save function. After the for loop has completed the last piece of code closing the file will be executed. (Figure 4)

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

Figure 4: Saving Data to the File

In this section we are asking for the user to input a new task and the task priority. To do this we create an input statement asking the user to enter a task. Then we have another input statement that will ask the user to input the priority of the task. This input will then be used in the earlier sections of code that we wrote. (Figure 5)

```
@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
    """
    pass  # TODO: Add Code Here!
    task = str(input("Enter Task: ")).strip()
    priority = str(input("Enter Priority: ")).strip()
    return task, priority
```

Figure 5: User Input Request for a new Task and Priority

In the last section of the code that we need to make an addition to we are again asking for the user to input data. This time we are asking the user to input a task that they would like to have removed from the list. This input will then be used in the earlier sections of code that we wrote. (Figure 6)

```
@staticmethod
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 = str(input("Task to Remove: ")).strip()
    return task
```

Figure 6: User Input Request for Task to Remove

The rest of the code that actually pulls in the different functions was already prewritten and did not require any alterations on our part.

Testing the Script

Running in PyCharm

When the code is first run in PyCharm the statement telling us what our file currently holds is printed at the top of the screen. Next, we will see the menu of options we have to choose from and the request for us to input a number from the menu. (Figure 7)

```
******* The current tasks ToDo are: *******
Clean Bathroom (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] -
```

Figure 7: Statements Printed when First Running Code

When the number 1 is selected we are telling the code that we want to add a new task to the list. After the 1 is entered we will be asked to enter a task. After this is done we will be asked to enter the priority of the task. Once this has been done all the current tasks in the list will be printed and we will be taken back to the main menu where we can select another number. (Figure 8)

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

Enter Task: Homework
Enter Priority: Medium
******* The current tasks ToDo are: *******
Clean Bathroom (High)
Homework (Medium)
*****************************************
```

Figure 8: Entering a New Task and Priority when a 1 is Entered

When a number 2 is selected we are telling the code that we want to remove a task from the list. When a two is entered the user will be asked to enter a task to remove. After this has been done the current list of data will be printed and the user will be taken back to the main menu again. (Figure 9)

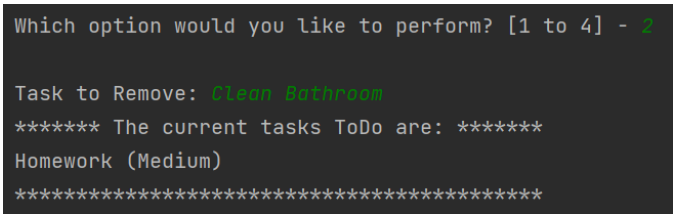

Figure 9: Removing a Task from the List after a 2 is Entered

When a number 3 is selected we are telling the code that we want to save our data to the file. When a three is entered the statement "Data Saved" will be printed, letting us know that the data was saved to the file and the code worked properly. The current tasks will also be printed under this. After this has been done we will again be returned to the main menu. (Figure 10)

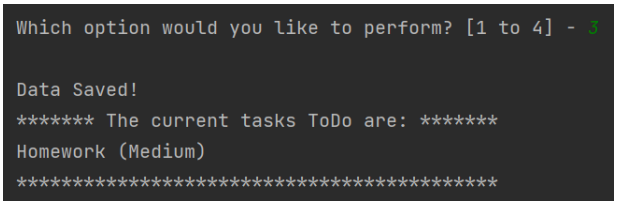

Figure 10: Saving the Data to the File after a 3 is Entered

When a number 4 is selected we are telling the code that we want to exit the script. After the four is entered the statement "Goodbye" is printed. This statement is telling us that the script has now been closed. (Figure 11)

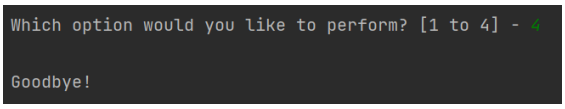

Figure 11: Closing the Script after a 4 is Entered

Running in Command Screen

When the code is first run in the Command Screen the statement telling us what our file currently holds is printed at the top of the screen. Next, we will see the menu of options we have to choose from and the request for us to input a number from the menu. (Figure 12)

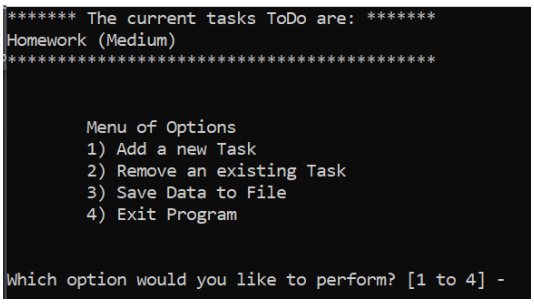

Figure 12: Statements Printed when First Running Code

When the number 1 is selected we are telling the code that we want to add a new task to the list. After the 1 is entered we will be asked to enter a task. After this is done we will be asked to enter the priority of the task. Once this has been done all the current tasks in the list will be printed and we will be taken back to the main menu where we can select another number. (Figure 13)

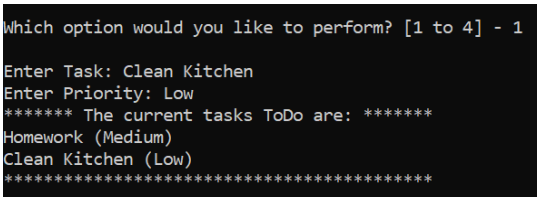

Figure 13: Entering a New Task and Priority when a 1 is Entered

When a number 2 is selected we are telling the code that we want to remove a task from the list. When a two is entered the user will be asked to enter a task to remove. After this has been done the current list of data will be printed and the user will be taken back to the main menu again. (Figure 14)

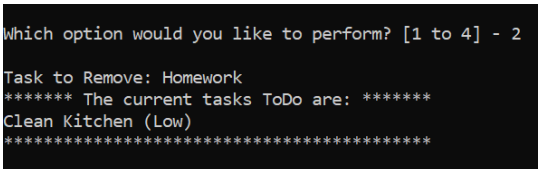

Figure 14: Removing a Task from the List after a 2 is Entered

When a number 3 is selected we are telling the code that we want to save our data to the file. When a three is entered the statement “Data Saved” will be printed, letting us know that the data was saved to the file and the code worked properly. The current tasks will also be printed under this. After this has been done we will again be returned to the main menu. (Figure 15)

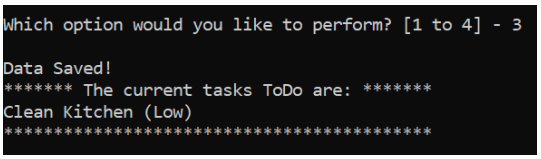

Figure 15: Saving the Data to the File after a 3 is Entered

When a number 4 is selected we are telling the code that we want to exit the script. When you are using the Command Screen to run the code the screen will simply close after the number four has been entered.

Summary

In this section we learned how to incorporate functions in writing our code. In this type of code, it feels a bit cumbersome compared to how we did it in the last assignment. However, I could see how writing functions could be very useful in other situations.