Connor McNiff

May 25, 2020

Foundations of Programming: Python

Assignment06

<https://github.com/cmcniff/IntroToProg-Python-Mod06>

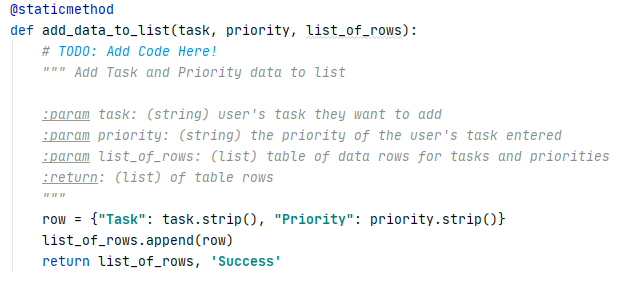
**To Do List with Functions**

Introduction

In this week’s assignment we were tasked with writing a code that captures user’s input in the form of a To Do List and Priority Ranking while navigating a menu. This week’s assignment varied from last weeks as we used functions to help build our data files. The menu gives the user the option to “1) View Data, “2) Add Data”, “3) Remove Data” “4) Load Data to a Text file” and “5) Exit the program while managing the input in a “Task” and “priority data column format. When the user inputs this data, the program will load columns of data in a separate text file via a Python dictionary object. As stated above, this is a progression from Assignment05 where we can use functions to improve our scripts.

Step 1 – Add Data to List

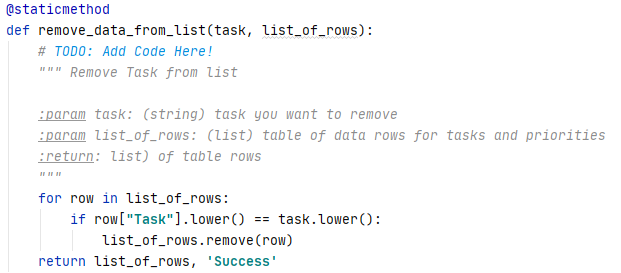
Following Assignment06 Starter script, the first step was to create a definition for the processing function “add data to list”. This function was created under the “class Processor” as the script is better organized and easier to read. Figure 1 below shows the definition for the processing function “add data to list” where we define how “rows” are made up from user’s task and priority data in a “list of rows”



**Figure 1 – Add Data to List**

Step 2 – Remove Data from List

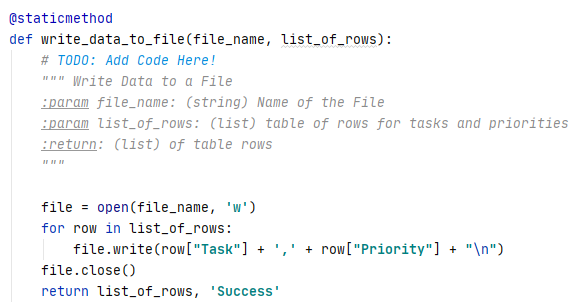
The second choice given to the user in our program’s menu is the “remove data” option where we define another processing function called “remove data from list” to remove “task data” from our stored data table. Figure 2 below shows the processing function and how the data column “Task” is identified and matched the user input of “task” for that item to be removed.



**Figure 2 – Remove Data from List**

Step 3 – Write Data to File

The Last processing function used in the “class Processor” was written to open a file and write the user’s data for “Task” and “Priority”. Figure 3 below shows the processing function and how the data is written in a new open file.

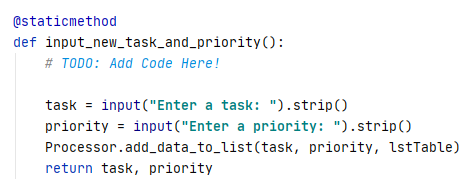


**Figure 3 – Write Data to File**

In summary for the processing functions; none of the functions are running data but instead we are defining them, similarly to how we define the data at the beginning of our script. In the later sections of our script we will be calling our functions that have since been defined.

Step 4 – Input New Data

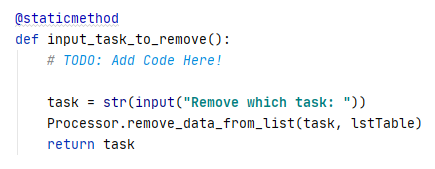
Once our processing functions are defined, we can them add in variables which will then be processed once the program is run. The IO function “Input new task and priority” defines two variables as the user input for “task” and “priority” and then are combined with the processing function “add data to list”. This IO will produce the results for the user’s input “task” and “priority” while also saving the data in our file as defined in steps 1 and 2. This IO is necessary for the user to complete selection 1 on our program’s menu. Figure 4 below displays the IO function “input new task an dpriority”.



**Figure 4 – Input New Data**

Step 5 – Remove Data

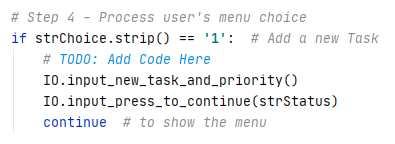
There is another IO function we used to remove “task” data from our file. Figure 5 below shows the variable “task” and how we remove that variable’s data using the processing function “remove data from list”



**Figure 5 – Remove Data**

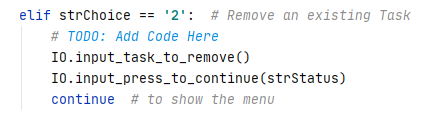
Step 6 – Main Body and User Menu Options

Completing our script, we looked to tie in our IO functions to the user’s choice within the program’s menu. Figure 6 below shows how we add new task and priority data after the user chose option 1 from our menu:



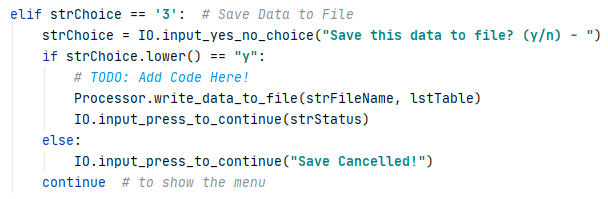
**Figure 6 – Add a New Task**

Figure 7 below shows how we remove data after the user chose option 2 from our menu. Specifically for this section , we focus on identifying a task rather than a task and a priority where we can then remove the whole row from the table memory in this program:

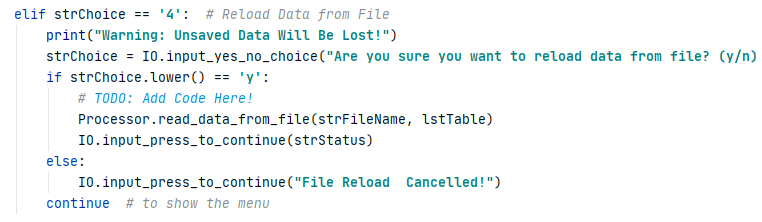


**Figure 7 – Remove an Existing Task**

Figure 8 and 9 below shows how we save an reload the data from our ToDo txt file. Using the file path strFileName = **"C:**\\**\_PythonClass**\\**Assignment06**\\**ToDoFile.txt"** we were able to read our write Table stored in our text file. In addition we can read our saved data as well shown in Figure 9.



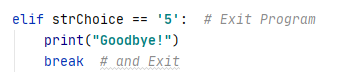
**Figure 8 – Saving Data to File**

****

**Figure 9 – Reloading Data from File**

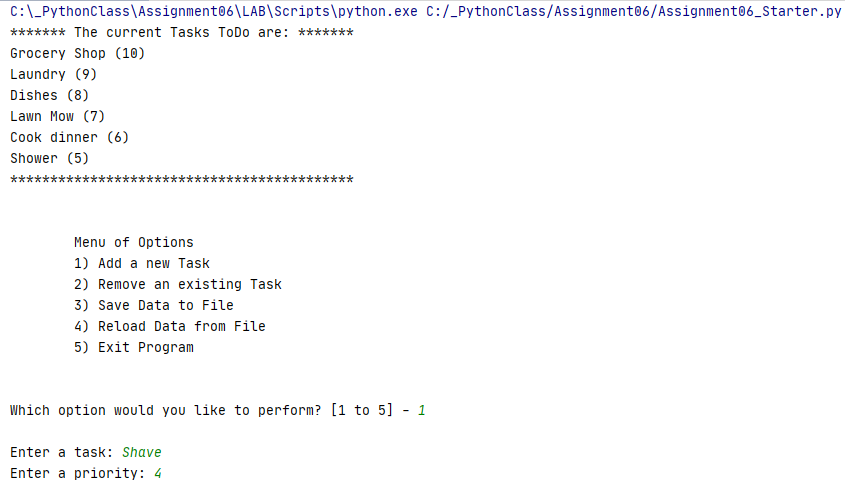
Step 7 – Exit Program

The last choice where the user can exit the program is choice 5 “Exit Program” represented in the script below in “Figure 10”:

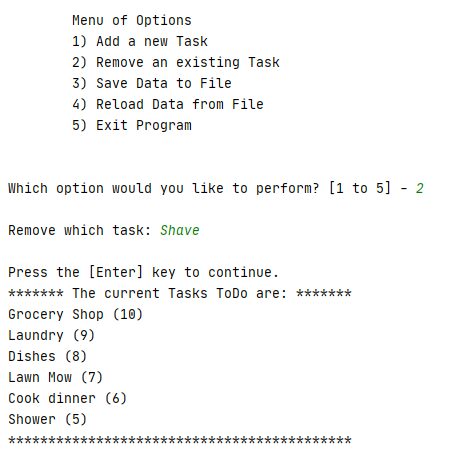


**Figure 10 – Exit Program**

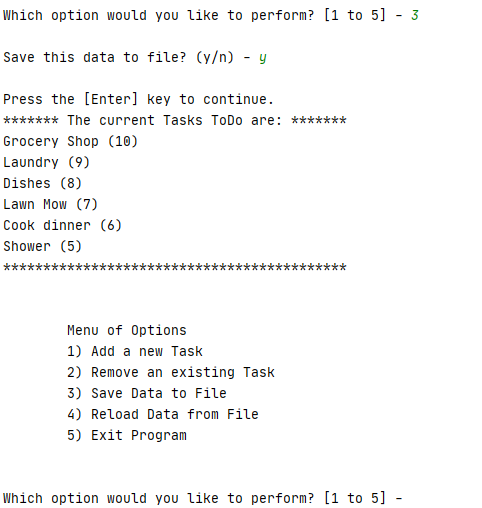
Run the Program in PyCharm, Figure 11 - 13



**Figure 11 – Stored Data in File and Addition of another task**

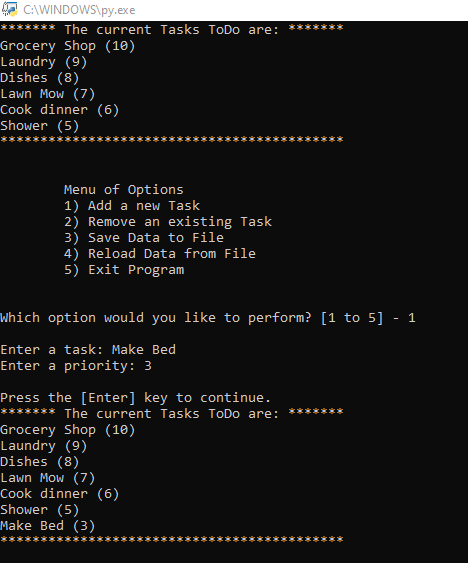
****

**Figure 12 – Removing a Task and Stored Data**

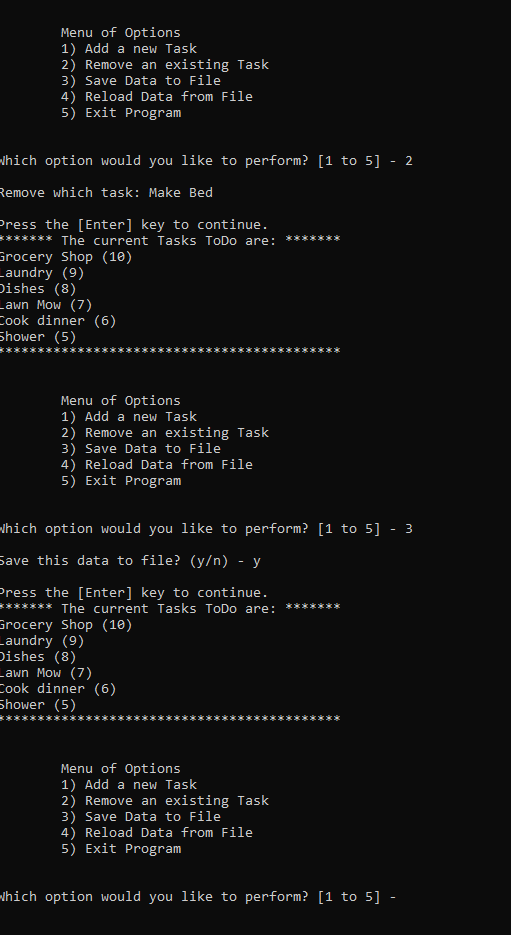


**Figure 13 – Saving Data to File**

Run the Program in the Command Shell, Figure 14 - 15

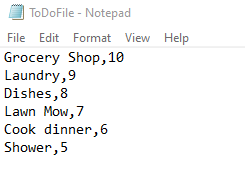


**Figure 14 – Add Task Data**



**Figure 15 – Remove Data and Save File**

Open the Text File, Figure 16

****

**Figure 16 - ToDoFile**

Summary

Progressing off of last week’s assignment we utilized the power of functions and classes to help organize our data. I found that using functions can help clean up our script and make the back end of the program more legible. I also found that I can create a more complex script while using less variables defined early in the data section which cleans things up nicely for the programmer.