Sara Tupponce

05/09/2021

IT FDN 110 A Sp21: Foundations of Programming: Python

Assignment 05

GitHub URL: <https://github.com/Ssara-Tt/IntroToProg-Python>

To-Do List Script

## Introduction

The goal for this assignment was to modify a starter script in order to be able to manage a to-do list. It would allow the user to see current tasks and task priorities, add a new task, remove a task, save the data to a text file called “ToDoList.txt”, and exit the program. This script required the use of sequence types like Dictionaries and Lists, as well as file reading and writing methods and various types of loops.

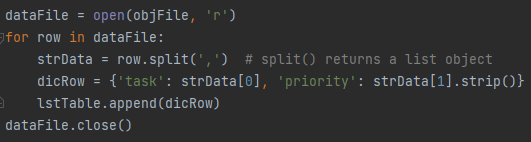
## Creating Python Script

Set up

The initial set up for this assignment was different than for past assignments in that a starter script was used as a starting point. It already had declared most of the necessary variables in a Data section, had a few comments, a Processing section, and the necessary loops for when the user made a menu choice selection in an Input/Output section. With all this information already provided, the only set up left was to fill in the header and to begin filling in the various sections.

Processing Layer

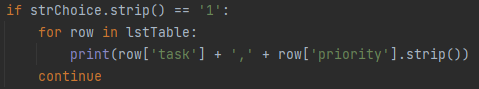
Since the Data layer information already contained declared variables, the Processing layer was the first section that needed code added to it. In this layer, the file named “ToDoList.txt” was opened and any data already saved in the file was loaded into the script. For each row of existing data, the data was saved to a dictionary variable named “dicRow” and then appended to a list variable named “lstTable”, which was acting as the table. The code can be seen below in Figure 1: Processing Layer.



**Figure 1: Processing Layer**

Input/Output Layer – Menu Choice 1

In the Input/Output layer, a menu had already been created that presented the user with a menu of choices for what to do with the to-do list data in the text file, as well as an input function to capture the user’s choice. If the user selected choice 1, the script would display to the user the current to-do items in the table. To do this, a “for” loop stepped through each row in “lstTable” and printed that row. The data was formatted by using a key to grab each part of the dictionary, each part was separated by a comma, and the “strip()” command was used to get rid of the extra “\n” character. The code can be seen below in Figure 2: I/O Layer – Show Current Data.



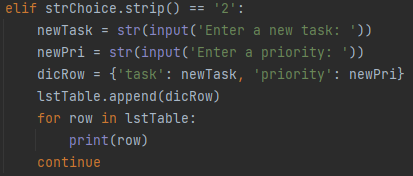
**Figure 2: I/O Layer – Show Current Data**

Test 1

A quick test of the script so far was run to make sure everything was working as expected. Since there were no error messages and the correct data was displayed to the user, work on the next menu choice script could begin.

Input/Output Layer – Menu Choice 2

If choice 2 was selected, the script would execute that would allow the user to input a new to-do item and priority for that item into the table. Two “input” functions would capture the user’s new data and save them to variables named “newTask” and “newPri”. These inputs were then added to a dictionary, which was appended to “lstTable”. The code can be seen below in Figure 3: I/O Layer – Add Item to List.



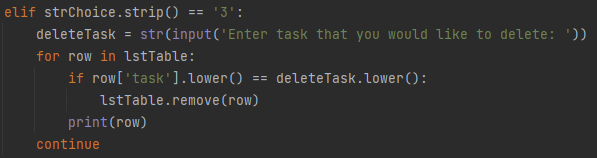
**Figure 3: I/O Layer – Add Item to List**

Test 2

In order to see if this next section of the script worked, a statement was included at the end of the section to print the table after the data had been added to the list to be able to see the data. The data printed appeared correct and the print statement was removed from the script.

Input/Output Layer – Menu Choice 3

The next menu choice allowed the user to remove an item from the list. An input statement asked the user which task they want to delete and saved it to a variable named “deleteTask”. A “for” loop stepped through every row in the table and the “if” statement checked every row to see if the “task” field was equal to the task that the user entered that they wanted to delete. If a match was found, that row containing the match was removed using the “remove()” method. The code can be seen below in Figure 4: I/O Layer – Remove Item from List.



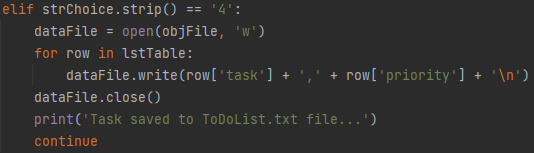
**Figure 4: I/O Layer – Remove Item from List**

Test 3

Much like in the test for the previous section, a print statement was temporarily included in the section of the script to see if the item had been removed from the list. The data printed showed that the selected data was deleted from the data set in the list.

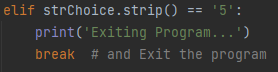
Input/Output Layer – Menu Choice 4 & 5

Menu choice 4 allowed the user to save the tasks to the file “ToDoList.txt”. The file was opened with the append setting “a”. For each row in the table, the “write()” method was used to save the data in the table to the file. After saving the data, the file was closed, and a print statement told the user that the task had been saved and the name of the file it had been saved to. The code can be seen below in Figure 5: I/O Layer – Save Tasks to File.



**Figure 5: I/O Layer – Save Tasks to File**

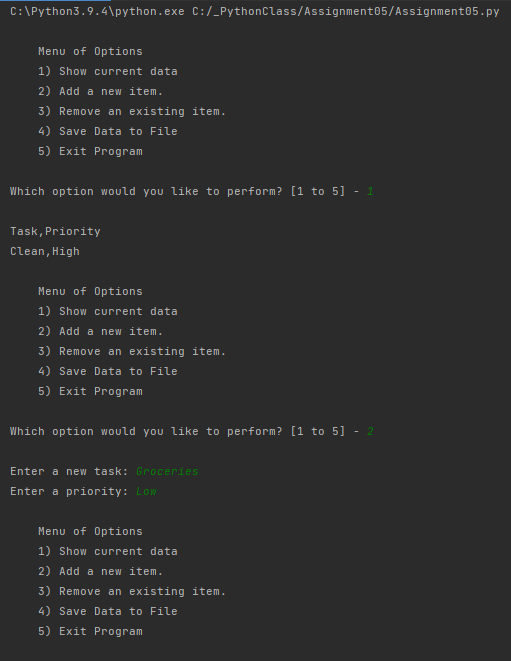
The code for menu choice 5 was for exiting the program and had already been included in the starter script. The only addition made was to include a print statement telling the user that the program was exiting. The code can be seen below in Figure 6: I/O Layer – Exit Program.



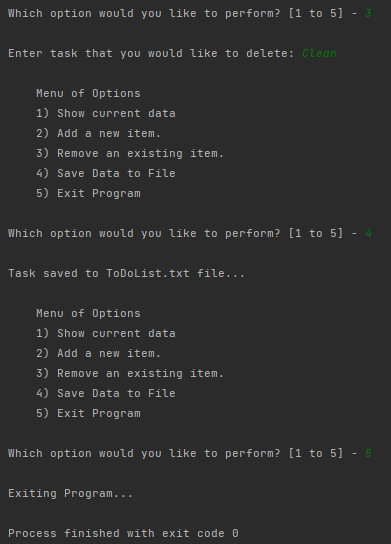
**Figure 6: I/O Layer – Exit Program**

Test 4

In the last round of testing, the script was run in both PyCharm and the Command Window. Because incremental tests had been done along the way, everything appeared to work correctly, except for a few cosmetic changes such as spaces between lines. Figure 7: PyCharm Running Script show the script being tested with PyCharm.

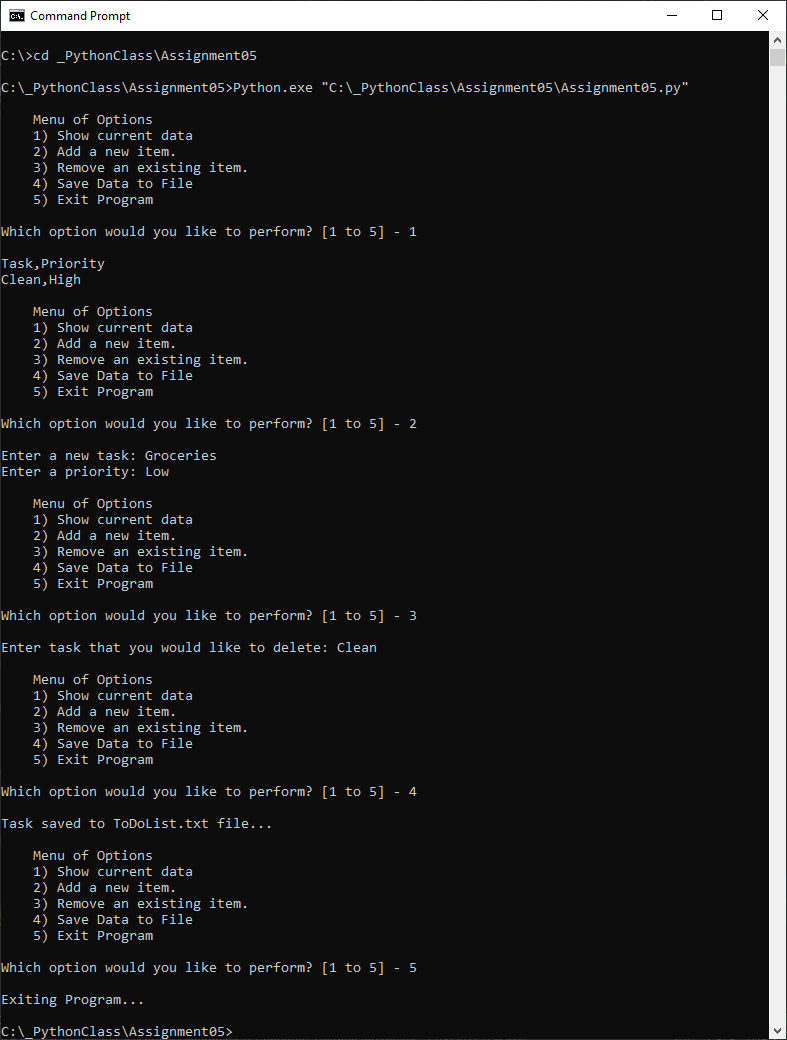


**Figure 7: PyCharm Running Script**



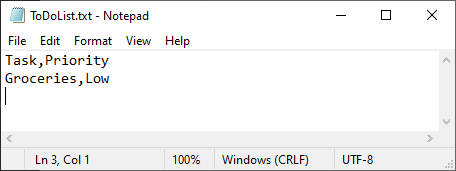
**Figure 7: PyCharm Running Script Cont.**

It should be noted that when the Command Window was used to try to run the Python script, an error was presented that said that the text file “ToDoList.txt” could not be found. The working directory had to be changed to the folder that contained both the assignment and the text file for the script to run. Figure 8: Command Window Running Script show the script being run in the Command Window.



**Figure 8: Command Window Running Script**

The resulting saved data can be seen in Figure 9: Data Saved in Text Document.



**Figure 9: Data Saved in Text Document**

## Summary

This assignment was more difficult than previous assignments because it required keeping track of the different lists and dictionaries and understanding how they were put together. Trying to run the script in the Command Window presented an additional challenge since the previous assignments had not required changing the working directory. However, having a starter script was a great help and did facilitated the process.