Steven Huang

May 18, 2020

IT FDN 100 A Sp 20: Foundations Of Programming: Python

Assignment05

<https://github.com/shpy086/IntroToProg-Python>

Assignment Recap

Introduction

In this assignment, we are asked to add code to a python script that is used to manage a To-Do list. This To-Do list contains two columns of data, “Task” and “Priority” that are loaded into a python dictionary object. Using the existing script, we need to define the inputs and outputs for a menu of operations use to manipulate the data and then write the data to a text file.

Review the Starter Script

To start, I read through the existing script several times to familiarize myself with the layout. I then compared it to some previous examples that required multiple operations (primarily Lab 5-2 and Assignment04). Using these other examples as reference points, I started building out the script.

Create and Test

The existing script starts with a prescribed menu of options that perform different operations on the data in the To-Do list. We were instructed to use each of these 5 operations to manipulate the data. See Figure 1 Below:

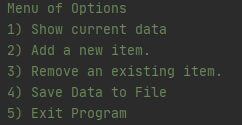


Figure 1

As you can see, the options on this menu are pretty straightforward. The first option, “show current data” will not display any data (since nothing has been added yet) so we will start with option 2, “add a new item”. In order to add a new item, the user needs to input the name of the task and its priority level. Since the program specifically asks us to input these two columns of data as part of a dictionary object, we will need to define them as two separate keys: “Task” and “Priority”. The corresponding values for these keys are inputted by the user when a new item is added. Figure 2 below shows the result of option 2:

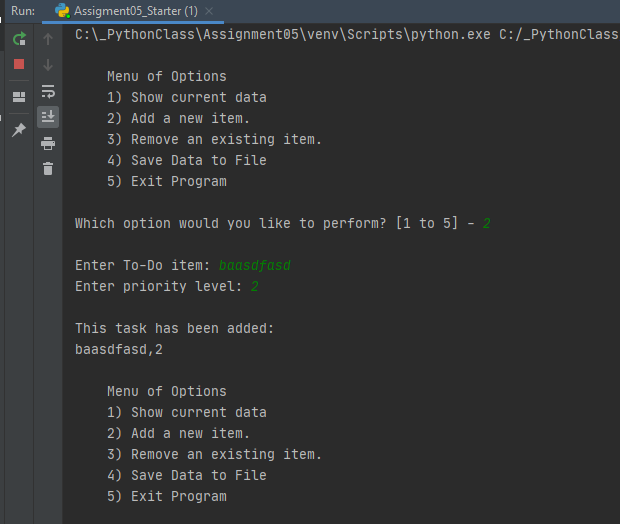


Figure 2

When the user selects option 2, they are prompted to input the name of the task and the priority level. Once both pieces of data have been entered, the program informs the user that their data has been saved to memory. Now that we have actually added some data to this To-Do list, we can choose option 1. This option will show the user all the data that has been stored in the lstTable to this point. Figure 3 below shows the output of option 1:

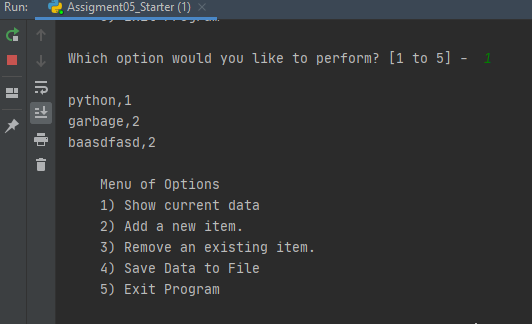


Figure 3

As you can see, when the user selects option 1, all the data that has been entered and stored into memory is displayed back to the user. Now that we have added rows of data to the lstTable and we’re able to view that data, we can move onto option 3, “removing an existing item”. Figure 4 below shows the outcome of option 3:

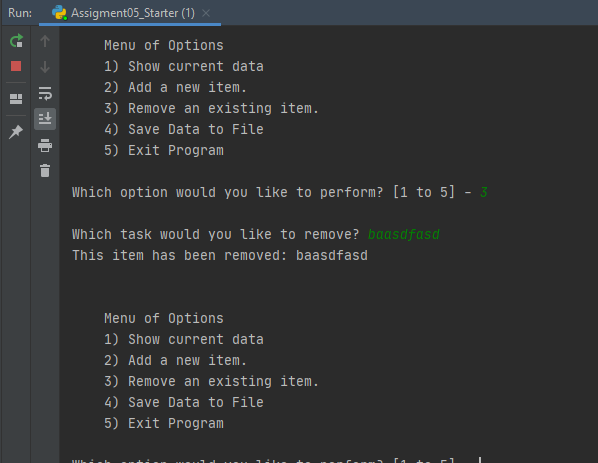


Figure 4

When the user selects option 3 from the menu, they’re asked “which task would you like to remove” and prompted to enter a task. In this case (per Figure 4), I selected one of the items that was previously added to the lstTable and stored in memory. As a result, the program confirms this with a print statement saying “this item has been removed.” In the event that a user selects an item that has not been previously stored in memory as a key, the condition would be false. If this happens, the program would return a statement saying “this item does not exist.

Option 4 is the last option for manipulating the data, this allows us to save data to an external text file. Figure 5 below shows the output of this option:

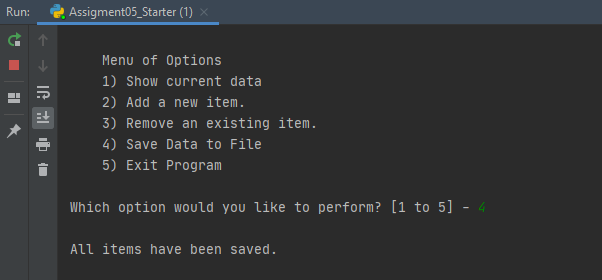


Figure 5

As you can see in Figure 5, the user selects option 4 and the program confirms that all items added to the lstTable have been saved to the text file. Figure 6 below shows the To-Do list text document containing the data entered and saved to the lstTable:

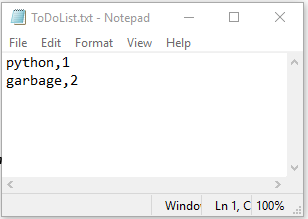


Figure 6

Writing the Code

Now that we have reviewed each option on the program menu and shown the desired outcomes, let’s take a deeper dive into the code written for each of these different options. In the template provided, the code was broken down into 7 steps. The first step asks us to open a text file and load the data from that text file into a python list of dictionaries. We start with a load file command that opens the objFIle (ToDoList.txt) and reads each row in the file. It then stores each row of data as a dictionary row and these dictionary rows are appended and stored to our lstTable. The lstTable will be the table of data that is stored in memory while the program is running and eventually saved to the external test file. Figure 7 shows the code from this first step:

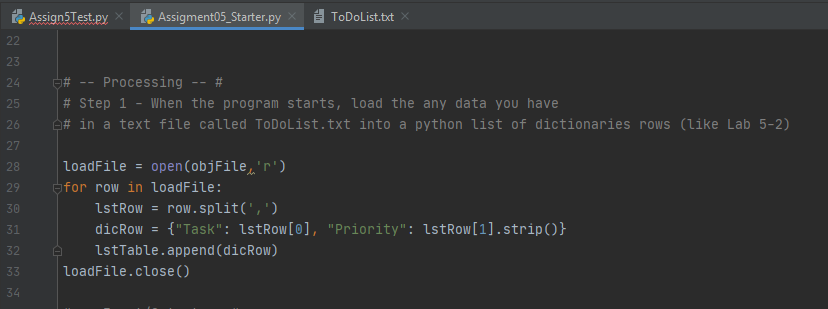


Figure 7

The second step prints out the menu of options. This this step was provided for us in the template, no code was added here. The third step asks us to create code to perform option 1, this shows all the data that has been added to the data table. Here, I used a condition that cycles through each row in the data table and displays it back to the user. See Figure 8 below:

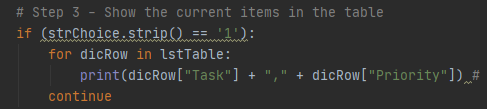


Figure 8

The fourth step asks us to write the code required for option 2, add a new item to the data table. This step is more interactive, it prompts the user to enter a specific task followed by the task’s priority level. After these two pieces of data are entered, they’re stored as values in a dictionary row and appended as a new row to the data table. See Figure 9 below:

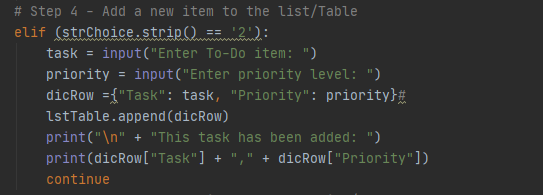


Figure 9

The fifth step asks us to perform option 3, remove a new item from the data table. Writing the code for this menu option was especially difficult. First, we created a new variable called itemRemoved. Then, we prompt the user to enter a task that they would like to remove from their To-Do list. We store the user’s input into the removeTask variable and add “.lower”. Since python is case sensitive, we have to ensure this is lower case. From there, we created a loop that cycles through each row of the data table to find the row that matches the value of removeTask.

This loop will produce one of two outcomes: if the loop finds a row that matches the value, it removes that item and returns a print statement saying “this item has been removed.” If the loop does not find a row that matches the value of removeTask, it returns a print statement saying “this item does not exist.” See Figure 10 below:

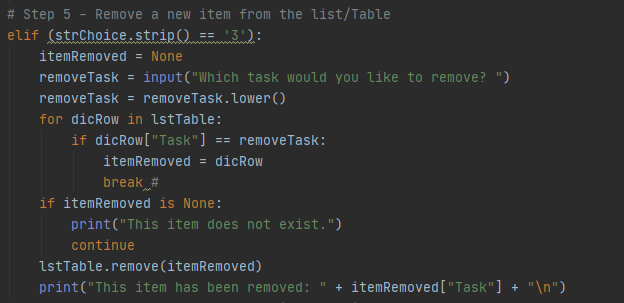


Figure 10

The sixth step asks us to save the data table to an external file (ToDoList.txt). Here, we use the loadFile command to open the text file, write each row the data table to that text file and close the file. The seventh (and final) step just asks us to exit the program and break the loop. See Figure 11 below:

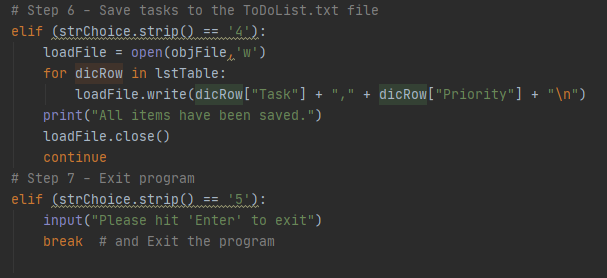


Figure 11

Run the Script

We have reviewed each option of the program menu and evaluated the code that was required to accomplish each step. While reviewing each step, we verified that the complete script ran successfully in PyCharm. Figure 12 below shows the program running successfully as a console application in the command shell:

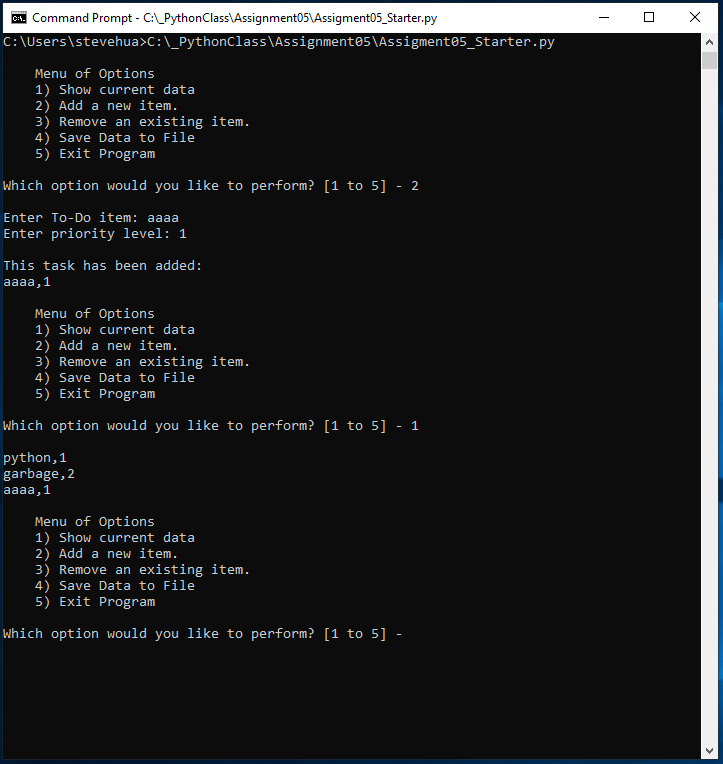


Figure 12

Conclusion

This assignment was undoubtedly the most challenging assignment we have had to date. In addition to reviewing an existing template to familiarize ourselves with the overall format of the program, we had to figure out a way to incorporate a complex loop to accomplish multiple operations. Although it was helpful to see the script broken up into 7 steps, the challenge was writing the code needed to perform each of those steps. In addition to the challenges associate with the logic and syntax of the code, I encountered other common errors. For example, the program ran seamlessly in PyCharm but when I ran it in the command console, it couldn’t locate our text file (ToDoList.txt). After attempting to troubleshoot it, I changed the location of the ObjFile from a relative file path to an absolute file path and the program ran successfully as a terminal application.