Amanda Spears

5/13/2021

Foundations of Programming: Python

Assignment 5

# **Using a Menu and Dictionaries for a Task and Priority Program**

## **Introduction**

This write-up will describe the steps necessary to modify an existing program that allows a user to create a task list and assign priorities to those tasks. The data entered by the user will then be saved in a text file that can be referenced, and updated (both adding and removing items from the list at the users will).

## **Reviewing existing code and editing the processing step**

The base code provided declared the variables we would be using, roughly outlined the data, processing, and input/output steps for the code. The first portion of the code that required editing was the processing step. Here we needed to be able to open a text file and read any existing data within that file into memory (Figure 1 below). This step was accomplished using the open() function, followed by creating a dictionary item for each row of data present, then adding to a list table for ease of display.

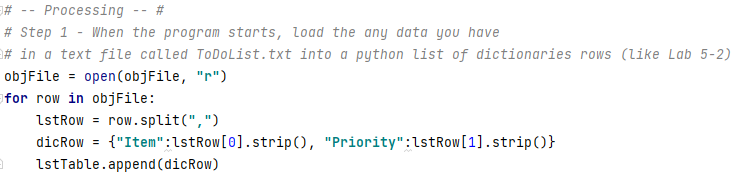


Figure 1. Reading the file into memory, and pulling data into a list table for display to the user.

## **Editing menu options**

The starting code outlined the basic while loop that runs through each of the menu options. These options included showing the current data, adding a new item, removing an existing item, saving the data to the file, then finally exiting the program.

### Option 1 – Show the current data

The first item in the menu is to show the current data that is already part of the list. In the processing step we already opened the text file and read in the data, adding each row to a dictionary item. We then added the dictionary rows into a list table for ease of display. In this step, all we need to do is a brief for loop to run through the list table, and print each row of data. Formatting was added in to extract teach element of the dictionary to make the display of data more visually appealing to the end user. See figure 2 below for the code.

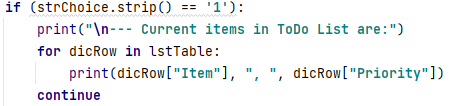


Figure 2.1. Display current data in list (the code)

Figure 2.2 below shows option 1 as run using PyCharm.

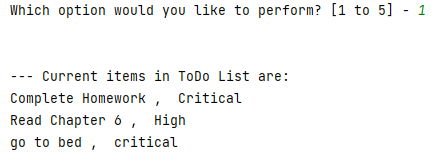


Figure 2.2. Display current data in list ( in use for PyCharm)

### Option 2 – Add a new item to the dictionary list

The second option on the menu allows a user to enter new data to create a new row of data. In this step we needed to add input() functions to enable user input of both the Item and Priority elements. Once those elements were entered by the user, they needed to be added to a dictionary row, then subsequently appended to our list table (as discussed above). Finally, the new data is printed for the user to view and ensure it was accurate.

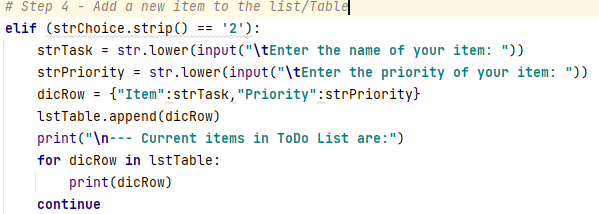


Figure 3.1. Option 2 – adding new data to the list (the code)

Figure 3.2 below shows the code in use with PyCharm. Here I have added setting up a GitHub account to my to do list, with a high priority seeing as this will be a necessary part of this assignment.

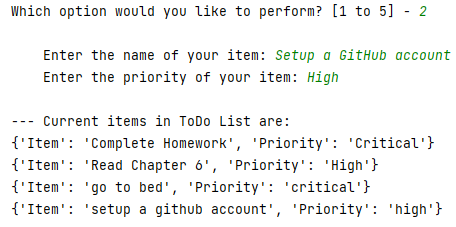


Figure 3.2. Option 2 – adding new data to the list (in use with PyCharm)

### Option 3 – Remove existing item

The third option on the menu allows the user to delete an item that is no longer necessary for their task list. This was perhaps the most complicated portion of the code edit. Here we ask the user to input the element that they wished to remove from the item list. We then initialize a Boolean value set to false, and a row number set to 0. These items then allow us to create a while loop that searches through each line of data in our table and removes every row that matches with the data input by the user at the beginning of option 3. See figure 4 below.

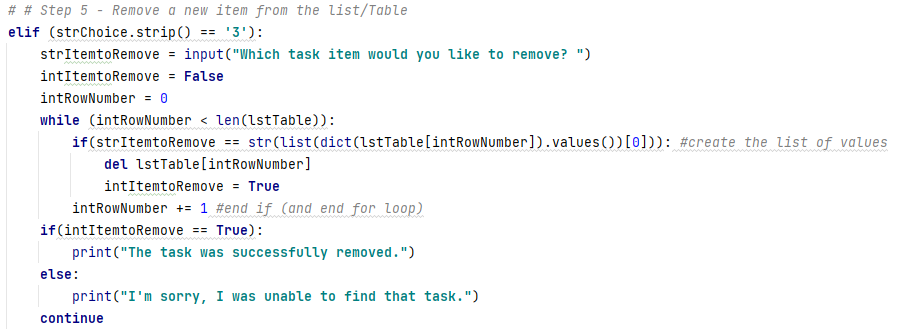


Figure 4.1: Option 3 – remove an item from the list (the code)

Figure 4.2 below shows the code in use, where I have removed the item of “Go to bed” from the to-do list. This is appropriate consider the amount of work I still need to do tonight.

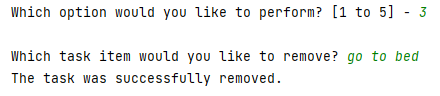


Figure 4.2: Option 3 – remove an item from the list (in use with PyCharm)

### Option 4 – Save new data to file

The next option on the menu, item 4, allows the user to then save the current data back to the text file. Since we have read that data into the program, and possibly added new data and/or deleted some data we will be overwriting the data in the text file, rather than just appending. We start this step by opening our text file object again, the we use a for loop to run through each dictionary row entry and write it to our text file. We end by closing our file and printing a statement letting the user know the data was saved successfully. See figure 5 below for the specific code.

### 

Figure 5.1: Option 4 - save new data to text file (the code)

The 4th menu option is to save the data to the file. Seeing as I am happy with the current data in the list after removing my “go to bed” item, I have elected to save the list to a file. Figure 5.2 below shows the code in use selecting option 4 with PyCharm, and the success message we coded into the output.

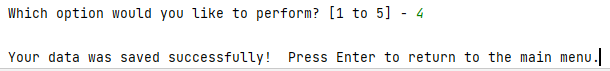


Figure 5.2: Option 4 - save new data to text file (in use with PyCharm)

### Option 5 – Exit the program

The fifth and final option on the menu is to exit the program. Here, there were no additions necessary, as the required “break” statement was already included. See figure 6 below.



Figure 6.1: Option 5– Exit the program (the code)

Again, the final menu option allows the user to exit the program. Figure 6.2 displays this functionality as seen by the end user. Seeing this displayed to the end user, I should probably also create a print statement thanking the user for using the program and acknowledging the exit…but frankly I am too tired for additional edits.

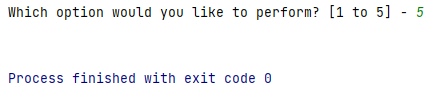


Figure 6.2: Option 5– Exit the program (in use with PyCharm)

### But what if the user enters a number other than 1-5?

The only item that was not outlined in the base code was an else statement. If an end user enters something other than 1-5, we needed a way to gently correct them and let them know that their entry was not valid. A simple else statement printing that they needed to select option 1-5 was sufficient to accomplish this task. See figure 7 below.



Figure 7.1: Else – if a user enters something other than 1-5 (the code)

The final portion of the code for this project which shows the user an error message is displayed below as if the end user had entered the number 7.

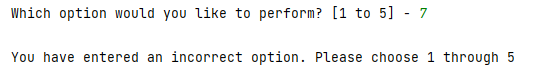


Figure 7.2: Else – if a user enters something other than 1-5 (in use with PyCharm)

## **Summary**

In summary, this document walks the reader through the process used to create a python script that displays a menu of options to end user allowing them to enter a list of tasks and the priority associated with those tasks. Similar to assignment 4 where loops and menu options are used, this week dictionaries are used to store and display the values back to the end user, along with the option to delete data that has been entered previously. Final data is then written to a text file for later use by the program.