Lauren Parker

November 13, 2020

IT FDN 110 A Au 20: Foundations Of Programming: Python

Assignment 05

Editing a To Do List Script

# Introduction

This paper will discuss the steps it took to adjust a Python script that was provided by our professor, to create an ongoing to-do list. The script should provide a menu of interactive options, such as, showing the current data in the list, adding data to the to-do list, removing data from the to-do list, saving the data to a file, and exiting the program.

# Step 1: Importing Starter Script & Defining Variables

Our professor provided us with a starter code for this assignment. The first thing I did was create a new file in PyCharm and copy and paste Professor Root’s starter code. I made sure to update the header with my name and date. Next, I reviewed Professor Root’s variables and made some adjustments to the names so that they were a bit clearer for my code. You can see an example of this is figure 1.

Text

Description automatically generated

***Figure 1: Example of header.***

# Step 2: Creating the File and Loading the Data from the .txt File

Judging from professor Root’s comments, he wanted us to load the data that we had in a text file called ToDoList.txt, into a Python list of dictionary rows. Since I didn’t already have a text file named ToDoList.txt, I created one by using objFile = open(ObjFileName, “w”). Once the file was created, I used the read function to load any of the data from this file. You can see an example of this in Figure 2.

Text

Description automatically generated

***Figure 2: Creating the file and reading it.***

# Step 3: Displaying the Current Data to the User

If the user types in “1”, the program needs to display the current items in the table. Since there is a chance that the table could be empty, I added an if not statement that lets the user know they don’t have any tasks. You can see this on lines 51-52 in Figure 3. The following statement tells the program that if the table is not empty, print all the rows in lstTable. You can see an example of this on lines 53-55 in Figure 3.

Text

Description automatically generated

***Figure 3: Showing how to display the info in the text file, and letting the user know if the text file is empty.***

# Step 4: Retrieving User Input to Add to the lstTable

If the user chooses option “2”, they are prompted to input a task and a priority level. Knowing that I would need to utilize this input data later, I assigned each input a variable. I used the variable strTask for the user’s task and strPriority for the user’s priority level. Once I retrieved the data from the user, I added the variables to a dictionary. You can see this on line 62 in Figure 4. The last statement makes sure that any new data is appended to the table.

Text

Description automatically generated

***Figure 4: Getting input from the user and adding it to a table.***

# Step 5: Deleting Data from the Table

The next portion of code needed to tell the computer what to do if the user input option “3”. This selection would let the user delete a new item from the list. In order to do this, I wanted to print out all the rows in the current table, so the user could see what tasks they had to choose from. I did this using a for loop to print each dictionary separately. Then I asked the user what task they would like to remove and assigned it the variable delTask. I used a second for loop to cycle through the rows in the table and search for a value that was equal to the user’s input. If delTask was equal to one of the row’s tasks, then I used the .remove() method to remove the row from the table. I also added a print statement to let the user know their task had been deleted. The last statement offers an else clause if the user’s input does not match to any of the tasks in the table. You can see an example of this in Figure 5.

Text

Description automatically generated

***Figure 5: Removing a row of data from the table with .remove().***

# Step 6: Saving Tasks to the Text File

According to the menu, if the user input option “4”, the program would need to save their info into a text file titled ToDoList.txt. To do this I had to open the file and write to it by using the code on line 80 in Figure 6. After opening the text file, I used a for loop to write each row of data to the text file separating each item with a comma. You can see an example of this on line 82 in Figure 6. I made sure to close the file and add a statement for the user that let them know their data had been saved. The very last elif statement in Figure 6 just gives a goodbye statement and exits the program.

***Text

Description automatically generated***

***Figure 6: Writing the data to a .txt file and ending the program.***

# Summary

In conclusion, we have discussed how to edit and add code in PyCharm. We have discussed how to add user input to a text file and how to read that info back to our user. We also discussed how to remove data from our table of lists.