Darivanh Vlachos

February 16, 2021

IT FDN 110 B Wi 21: Foundations of Programming: Python

Assignment05

https://github.com/darivanhatUW/introToProg-Python

Making a List and Checking For Loops

Introduction

The assignment this week in IT FDN 110 B Wi 21: Foundations of Programming: Python, the class was tasked to create a program that tracks a user's To Do List from a saved file and asks the user to choose from a list of actions to be performed, which includes 1) Show current data, 2) Add a new item, 3) Remove existing item, 4) Save Data to File, 5) Exit Program. Building from the previous weeks' assignments and starter code, students are tasked to read and write to a ToDoList.txt, as well as maintain a list and table for temporary storage. Below I will chronicle the challenges and thoughts as I traverse through each action to be performed within the program.

Declared Variables

The starter code, Assignment05_starter.py, was provided to the class by Randall Root, lecturer, from which are included variables the class is expected to build their programs. Below is a table of the the variables and its purpose in the program, and how I used it throughout my program.

Table 1 Variable declarations, their purposes and how they were used in my program.

Variable	Purpose	Use Instances
ObjFile	Represents an object to perform actions specific to a certain file	Where the ToDoList.txt would need to be read, written or appended.
strFile	Represents the text file ToDoList.txt	Where the ToDoList.txt would be called within the objFile.
strData	Represents a row of data from ToDoList.txt	Where data was being read from the ToDoList.txt whose data is then translated into the dictionary format.
dicRow	Represents a row of data separated by a comma and items as in a dictionary. Each item containing a key and a value	Where individual items needed to be added or deleted.
lstTable	Represents the table which contains rows of data from the dictionary	Where all items within the table were presented to or if an item were to be amended by the user.
strMenu	Represents a printed menu for the users to choose from	I did not use this in my program as its presence within the while loop is presented following the performance of action, as I expected.
strChoice	Represents the input, choice, from the user	Followed the menu so that user may choose an action from said menu.

Step 1 – Load data from ToDoList.txt into a Python List of Dictionaries Rows

The objective of Step 1 was to load data from the ToDoList.txt into a list and table within the program for temporary storage to be amended or deleted by the user. Initially, my confusion with this step stemmed from my misunderstanding that I needed to provide the data within the program rather than pulling the data from the ToDoList.txt. Figure 1, illustrates my confusion with the assignment. After a conversation with Teaching Assistant, Anubhaw Arya, where I questioned the ease of the assignment as surely, I misunderstood, Arya exclaimed, "Programming is hard." In which I whole heartedly agree.

```
30    objFile = open(strFile, "w") # open txt file
31    dicRow = {"Task": "Vacuum", "Priority": "high"}
32    lstTable.append(dicRow)
33    dicRow = {"Task": "Dusting", "Priority": "low"}
34    lstTable.append(dicRow)
35    objFile.close()
```

Figure 1 Code demonstrating my misunderstanding in adding the items within the program rather than from the ToDoList.txt.

Arya then provided context, guidance and clarity about the origin of the data and I was able to correct my code to fit the assignment. Figure 2, below, demonstrates the code which opens the ToDoList.txt file and reads its contents. The for loop, reads the data by row and returns the row signifying that the item is separated by a comma, otherwise known as comma delimited. dicRow then takes the item and assigns the first value as the key for a dictionary and the other as the value of the key. lstTable then appends each item into a table rather than replacing the row with each loop. When this portion of the code is executed, the lstTable variable is now updated with the data and ready to use throughout the program.

```
31    objFile = open(strFile, "r") # open txt file
32    for row in objFile:
33        strData = row.split(", ")
34        dicRow = {"Task": strData[0], "Priority": strData[1].strip()}
35        lstTable.append(dicRow)
```

Figure 2 My code displaying data being collected from the ToDoList.txt as the program opens

Step 2 – Display a Menu of Choices to the User

The menu of choices presented to the user is a list of choices provided to the user upon the start of the program as well as after each action is performed. This particular list was provided by in the Assignment05_starter.py within a while loop. The user is then able to input their choice with the prompt, also provided by the starter code, "Which option would you like to perform? [1 to 5]". I did not feel the need to move or alter the menu nor the prompt as they were performing as expected within the program. Following the menu and the prompt, Figure 3, were the if and elif statements that which performed an action dependent of the user's choice. User options included: 1) Show current data, 2) Add a new item, 3) Remove an existing item, 4) Save Data to File, and 5) Exit Program.

```
# Step 2 - Display a menu of choices to the user
while (True):
    print("""
    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program
    """)
    strChoice = str(input("Which option would you like to perform? [1 to 5] - "))
```

Figure 3 Menu and prompt for the user which I kept in its original state.

Step 3 – Show the Current Items in the Table

If the user chooses "1" from the menu, the program will show the user the current data within the ToDoList.txt file. In my program, Figure 4, I wanted to confirm the user's choice by letting them know that they were viewing their task list and include a header for the information being presented. In "Don't Make Me Think," author Steve Krug's states that his principle for usability in design is "as far as is humanly possible, when I look at a [program] it should be self-evident. Obvious. Self-Explanatory…without expending any effort thinking about it." I extend that principle to this program, by ensuring that when the user makes a choice their decision is confirmed and the data is presented with headings to remind the user what each column of the data represents. This same principle is not only provided for the user but for future programmers who may read my code.

```
    if (strChoice.strip() == '1'):
        # TODO: Add Code Here
        print("YOUR TASK LIST: ") # Informs the user the action that was preformed.
        print("Task | Priority") # Headings for the data
        print("----------------")
        for row in lstTable: # Loop to display all the rows in the table in a particular format
            print(row["Task"] + " | " + row["Priority"])
        continue
```

Figure 4 Code for user choice 1 which includes prints to inform the user the action that was performed and a for loop to list and display the contents of the table.

Step 4 – Add a New Item to the List/Table

If the user chooses "2" from the menu, the program will execute the second elif statement which allows the user to input a task and priority for said task. Although not explicitly stated within the starter code, Figure 5, I chose to provide prompts for the user to input the data rather than my original statement that had the addition of the task item within the code. This provides the user the ability to add more than one task item if they so choose. Although Krug provides a case against making the user think, in this, even he would agree, that a user would be confused if a task list meant for their personal use would not allow them to provide their own tasks and their priorities. To that point, confusing the user would make them think and possibly quit the program out of frustration, which is never our intent, unless of course it is the user's intent and they are fully satisfied with their experience.

```
60      elif (strChoice.strip() == '2'):
61          # TODO: Add Code Here
62          taskInput = input("Which task would you like to add? ") # Ask the user to input a task
63          priorityInput = input("What is its priority? (Choose: high, medium, low) ") # Ask the user to input the priority level
64          print("\nHERE'S YOUR NEW TASK LIST:") # Informs the user the action that was preformed.
65          print("Task | Priority")  # Headings for the data
66          print("---------------")
67          dicRow = {"Task": taskInput, "Priority": priorityInput.strip()} # Adds the user input in the dictionary
68          lstTable.append(dicRow) # Appends the new user input to the table
69          for row in lstTable: # Loops through the table, per row, to display the data in a particular format
70              print(row["Task"] + " | " + row["Priority"])
71          continue
```

Figure 5 Included is the prompt for user input. Allows for users to add more tasks and priorities. Once the action is performed the print informs the user what action has been performed and what tasks and their priorities were added.

Step 5 – Remove a New Item from the List/Table

If the user chooses “3” from the menu, the program will execute the third elif statement which allows the user to delete the last task item in the table. I used the .pop(), in Figure 6, method in order to remove the last item in the table. As the previous statements, I provide the user with the list to confirm that the action performed is what the user anticipated, which is the existing list without the last task item.

```
72      # Step 5 - Remove a new item from the list/Table
73      elif (strChoice.strip() == '3'):
74          # TODO: Add Code Here
75          lstTable.pop() # Removes the last entry in the table
76          print("HERE'S YOUR NEW TASK LIST: ") # Informs the user the action that was preformed.
77          print("Task | Priority") # Headings for the data
78          print("---------------")
79          for row in lstTable: # Displays the items in table in a particular format
80              print(row["Task"] + " | " + row["Priority"])
81          continue
```

Figure 6 The .pop() method is used in order for the user to delete the last entry.

Step 6 – Save tasks to the ToDoList.txt File

If the user chooses “4” from the menu, the program will execute the fourth elif statement which allows the user will have chosen to save the task items into the ToDoList.txt file. To do this, Figure 7, I open the ToDoList.txt file, enable the file to be written to, and because I want a particular formatting when the data transfer is executed, I use the row[] and include the key, “Task” and “Priority,” within the square brackets which prints the corresponding values in order to overwrite the text file. Formatting of the data is performed within object.write() method. I then include the print which displays saved data, which ensures the user that the data has been saved and the file name so that the user may find it at will.

```
83      elif (strChoice.strip() == '4'):
84          # TODO: Add Code Here
85          objFile = open(strFile, "w")# Opens the "ToDoList.txt" to write into
86          print("YOUR TASK LIST HAS BEEN SAVED TO TODOLIST.TXT:")# Informs the user the action that was preformed.
87          print("Task | Priority")# Headings for the data
88          print("----------------")
89          for row in lstTable:# Adds items from table to text file
90              objFile.write(row["Task"] + ", " + row["Priority"] + "\n")
91              print(row["Task"] + " | " + row["Priority"])
92          objFile.close()
93          continue
```

Figure 7 Informs the user that their data has been saved and the file name.

Step 7 – Exit Program

If the user chooses “5” from the menu, the program will execute the fifth elif statement which allows the user to exit the program. At the end of the statement, Figure 8, there is a break which breaks out of the while loop. I eventually added a print statement to display so that the user is aware they have exited the program.

```
95      elif (strChoice.strip() == '5'):
96          # TODO: Add Code Here
97          print("Thank you for using Task List. See you back soon!")# Tells the user they have ended the program.
98          break  # and Exit the program
```

Figure 8 I added a print statement to inform the user that they have chosen to end the program.

Summary

This week’s assignment, Assingment05, students were tasked to create an interactive program that provided a menu of actions that allow the users to maintain a task list and their priorities. As recounted above, although there was some confusion initially, I was able to use my knowledge from previous classes and resources in order to complete my program.

Reference:

Root, Randall. IT FDN 110 B Wi 21: Foundations of Programming: Python. *Zoom*. 12 Jan. 2021 to 09 Feb. 2021. https://canvas.uw.edu/courses/1424625. February 10-15, 2021

Arya, Anubhaw. IT FDN 110 B Wi 21: Foundations of Programming: Python, Office Hours. *Zoom*. 11 Feb 2021.

Krug, Steve. *Don't Make Me Think, Revisited A Common Sense Approach to Web Usability*. Berkeley, CA, New Riders, 2014.

"Data Structures." *Python.org*.https://docs.python.org/3/tutorial/datastructures.html. 11- 16 Feb 2021