Matthew Lord
2023.02.20
IT FDN 110 A Wi 23: Foundations of Programming: Python
Assignment05
<GITHUB LINK>

Reading/Writing to Text Files with Lists/Dictionaries While Gathering User Input

Introduction

In this paper we will discuss creating a Python script that will accept user input to collect and edit a table of tasks and their associated priority. To do so we will employ List and Dictionary objects within the Python language. We will read and write this data to a text file for use later when needed. To gain the user input we will utilize a menu system within a loop which will provide the user with several options for feedback and collecting or editing data until they choose to exit the program.

Program Setup

A large portion of the code was already written as part of the assignment. It provided the menu block complete with user selectable options to access associated blocks of code within the script.

```
# -- BEGIN INPUT/OUTPUT -- #
# 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 = input("Which option would you like to perform? [1 to 5] - ")
    print()  # adding a new line for looks

    # Step 3 - Show the current items in the table
    if (strChoice.strip() == '1'):
    . . .
```

Prior to the menu system there is a block of code for “Step 1” which loads data from a text file. We first check to see if the file exists before reading from it and if it doesn’t exist, we create the file.

```
# -- BEGIN PROCESSING -- #
# Step 1 - When the program starts, load any data you have
# in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)
if (strFile):  # see if the file exists to be read from
    objFile = open(strFile, "r")  # if the file exists, open it to be read
    for row in objFile:  # loop through each row of the file
        strData = row  # assign the text file row to a string
        lstRow = strData.split(",")  # create a temporary list from the string
        # use the temporary list to create a dictionary object
        dicRow = {"task": lstRow[0], "priority": lstRow[1].strip()}
        lstTable.append(dicRow)  # add the dictionary row to a list/table
    # close the file for now
    objFile.close()
else:
    # if the file does not exist, create the file
    objFile = open(strFile, "w")
    # close the file
    objFile.close()

# -- END PROCESSING -- #
```

The above portion of code required some alterations to our global variables in my case. While running the script in PyCharm I did not have any issues with the code checking for the "ToDoList.txt" file. The path was kept relative since the file is in the same folder as the script.

However, while running the code in Terminal, I received an error and needed to make adjustments to the code.

```
Traceback (most recent call last):
  File "/Users/spacetimeshift/Documents/_PythonClass/Assignment05/Assignment05.py", line 52, in
<module>
    objFile = open(strFile, "r")  # if the file exists, open it to be read
              ^^^^^^^^^^^^^^^^^^
FileNotFoundError: [Errno 2] No such file or directory: 'ToDoList.txt'
```

After a short search on the internet it became clear that I needed to make an explicit path although I am still a bit confused as to why (D'cruz, Arbie. "Python FileNotFoundError: [Errno 2] No Such File or Directory Solution." *Careerkarma.com*, https://careerkarma.com/wiki/python-file-not-found-error/).

Developing the Script Code

The part of this code which took the most time for me to develop was the Step #5 code block for user selection "3)". This, I believe was due in large part to not fully understand how to access data in lists and how this differentiates from accessing data in dictionaries. I went back and forth between the course book, the supplied notes, and the online web references, before I began to catch on to how I was handling the code wrongly for the intended results. There seemed to be some confusion on my part in how best to handle the data.

After gaining a better understanding on the methods to access lists/dictionaries the formatting of the user display as well as the data saved and retrieved from a file the challenge became checking for a "task" to be removed in a dictionary/list pairing. To do this I used several nested statements within the for loop for menu selection "3)"

The code as written in my version of the script:

```
# Step 5 - Remove a new item from the list/Table
elif (strChoice.strip() == '3'):
    # set counter to check when looping through the full list/table
    counter = 1
    # gather input from user as to which item they would like removed
    taskToDelete = input("What task would you like to remove?: ")
    # loop through each row in the list
    for item in lstTable:
        # get the row of text and assign it to a variable
        dicRow = item
        # check if user entry is NOT found in the table
        if dicRow["task"] != taskToDelete:
            # if the entry is NOT found and we have reached the end of our list/table
            if counter >= len(lstTable):
                # inform the user that the task they entered is not in the list/table
                print("\nThe item '", taskToDelete, "' has not been found.")
        # check if user entry is found in the table
        elif dicRow["task"] == taskToDelete:
            # remove the item from the list/table
            lstTable.remove(item)
            # inform the user that the item has been removed
            print("\nThe item '", taskToDelete, "' has been removed from the list.")
            break  # continue to the while loop when task is found and deleted from list/table
        counter += 1  # increase counter for each item in the list/table
    continue  # continue to the while/menu loop
```

In addition to the nested statements I used a counter variable to check if and when we had iterated through the entire list without finding the task the user had entered. In keeping track of this I could then let the user know the the word they were requesting to remove could not be found in the list at which point the script would return the user to the main menu.

Summary

In this paper we discussed editing and adding to pre-existing code in order to gather and store user data. The data was stored in both random access memory in the lists and dictionaries while the code was running and hard drive memory when we saved to a file at the user's request.

I experienced several challenges while working on this script which involved differentiating between the ways data in lists and data in dictionaries can be iterated through, modified, and formatted for user feedback and saving to file.