Name: Britany Vesel
Date: February 21, 2023
Course: Foundations of Programming: Python
GitHub URL:

Assignment 05 – Lists and Dictionaries

Introduction

This week in Foundations of Programming: Python we focused on creating a list of dictionaries and using a menu with user input to manipulate the data.

Dictionaries

Dictionaries are similar to lists and tuples in that they can contain strings of data and could be compared to rows in a database. Dictionaries, however, are more versatile and can be manipulated and modified more than the other two storage types. One important aspect of dictionaries is that the column names, or keys, can now be addressed and called as part of the data manipulation returning all the values in that column. Adding data, removing data, or changing part of a row of data can all be within a dictionary.

Homework

This week's homework involved lists and dictionaries and having a user input into a menu to manipulate the data. We used the professor's starter code to begin the programming file and added the ToDo code sections he indicated to make the menu work. The first step to this was to import a previously created file and create dictionary rows from the data contained within, then adding them together in memory to create a table. Instead of using strings for column names that I could retype incorrectly, I chose to set the column names, keys, as variables and then used those variables through the length of my program to reduce potential mistakes.

Figure 1: Creating the Dictionary Rows and Assigning Variables as Keys

```
constStrKey_Task = "Task"
constStrKey_Priority = "Priority"



# -- Processing -- #
# Step 1 - When the program starts, load the any data you have
# in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)
objFile = open(strFile, "r")
for row in objFile:
    lstRow = row.split(",")  # Returns a list!
    dicRow = {constStrKey_Task: lstRow[0], constStrKey_Priority: lstRow[1].strip()}
    lstTable.append(dicRow)
#    print(dicRow[constStrKey_Task] + " ~ " + dicRow[constStrKey_Priority]) #Using for debug to see original data
objFile.close()
```

The first menu option the user could indicate was to display data currently in the table. I reused code for this from last week's assignment and changed the text inside to reflect the newly assigned variables for the keys. After each section of added code, I ran the program to ensure that each user input correctly performed the intended way.

The second menu option was also easier. I copied the code from last week and then reused lines of the import code above to assign the user input to the data and add the data to the dictionary row and table list. I initially also added lines to display the new version of the table to the user to use visually as a debug method to ensure the add was correctly working, but commented them out of the final program.

Figure 2: Adding Data and Commenting Debug Lines Out

```
    # Step 4 - Add a new item to the list/Table
    elif (strChoice.strip() == '2'):
        strTask = input("Enter a Task: ")
        strPriority = input("Enter a Priority: ")
        dicRow = {constStrKey_Task:strTask, constStrKey_Priority:strPriority}
        lstTable.append(dicRow)
#        for row in lstTable:
#            print(row[constStrKey_Task], row[constStrKey_Priority], sep="\t") #Used to debug and see results
        continue
```

The third menu option was difficult for me and took much of my time on this assignment. Since the book's example for removing data from a dictionary didn't also involve lists, the delete did not work on my data. I returned to Randall Roots homework hints and tried to use that code, but only the simple delete was working for me. It either got stuck in the while loop, or if I had a for loop, it would skip some lines in the newly mutated list due to it not looking at the same line twice for the duplicates. I solved this by continuing to use the while loop, but adding a break for the while loop first, then adding variables for it to keep searching if it did find a match to user input. This method is inefficient in the long run for massive databases of data because it is checking every line every time for a match, but I did get working code here, which I am proud of since it took so long to understand. **Note:** I did look at Randall Root's solution, and copied and ran his code. If you add multiple taxes with different priorities to his table, it doesn't remove all

the examples like my for loop initially did. I ended up not using the iterator because I couldn't understand how to use the counter's correctly to account for this problem.

Figure 3: Removing Data for Multiple Matches

```
    # Step 5 - Remove a new item from the list/Table
    elif (strChoice.strip() == '3'):
        strTask = input("What task do you want to remove?: ")
        rowFound = False
        keepSearching = True
        while(keepSearching):
            keepSearching = False
            for dicRow in lstTable:
                if dicRow["Task"].lower() == strTask.lower():
                    lstTable.remove(dicRow)
                    print("\nOkay, I deleted", strTask, "and it had priority: ", dicRow[constStrKey_Priority])
                    rowFound = True
                    keepSearching = True
                    break
            continue
        if (rowFound == False):
            print(strTask, "was not found!")
        continue
```

The fourth menu option for the user was to save the data to the file. I again reused code from last week and changed the variables and file name to reflect the current dictionary and file name Randall Root indicated in his starter code. The last menu option for the user was to exit the program and I added a print line indicating it was closing for flavor. These were again easy to execute after the Remove Data undertaking.

Conclusion

The Remove part of the assignment was the most difficult to execute for me, and because it helps me to talk things out with someone, I bounced my code off of my programmer husband. He helped me understand the remove code by talking it out with someone so that I could implement a fix for the problems encountered. It was interesting for me to see that Randall Root's code also had the same issue, and I would be interested to see how he would fix it in a future class video.