Patrick Regan
Aug 8th, 2023
ITFDN 110 A
Assignment 05
https://github.com/egg2020/IntroToProg-Python

Lists and Dictionaries

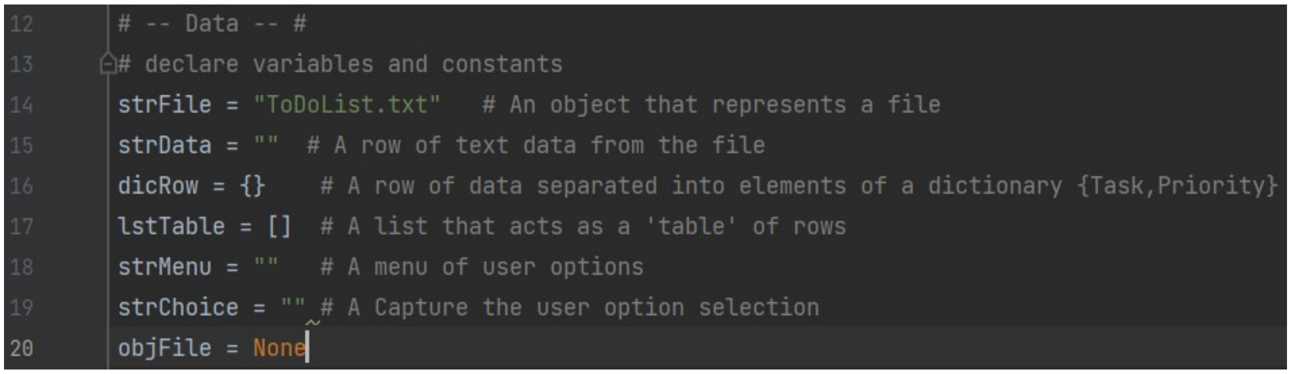
Intro

Moving Forward with our studies, lists and dictionaries help us in organizing data. For this weeks assignment, we nested dictionaries inside of a list, and performed tasks such as adding and removal of the individual dictionaries. I was challanged with the removal of of dictionaries when the chooses a corresponding value. I tried many different solutions, and each one printed out something slightly different. Once I found the solution, it became very obvious. Below is my summary for writing this weeks assignment.

Program

This week we were asked to write a program that prompts a user to enter a task, and a corresponding priority level. The user would be able to add and remove tasks from a list, as well as display and save the data to a .txt file, stored on a local computer. We worked off of a template provided by our instructor, and filled in the blanks where needed.

Below is the variables that were part of the template provided. I added 'objFile' just out of habit, even though it isn't necessary.

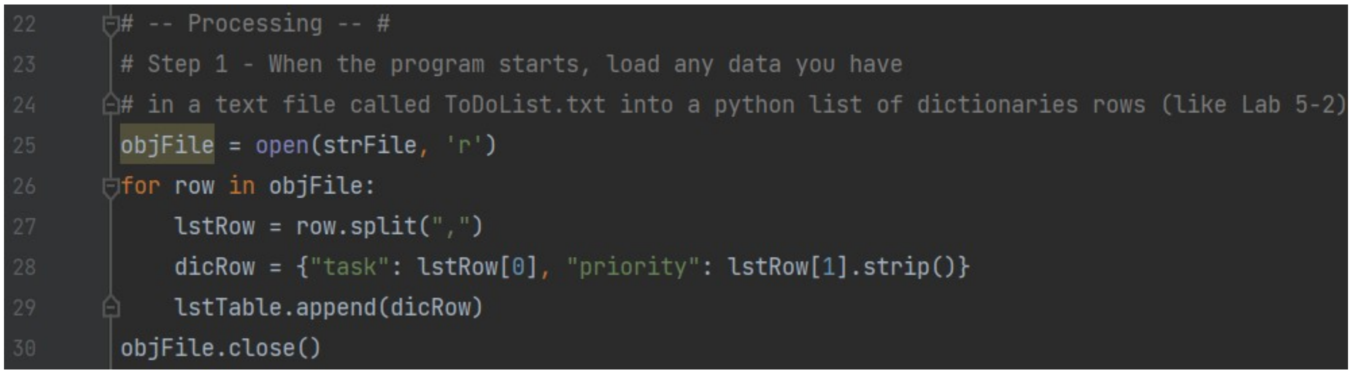


```
12  # -- Data -- #
13  # declare variables and constants
14  strFile = "ToDoList.txt"   # An object that represents a file
15  strData = ""  # A row of text data from the file
16  dicRow = {}    # A row of data separated into elements of a dictionary {Task,Priority}
17  lstTable = []  # A list that acts as a 'table' of rows
18  strMenu = ""   # A menu of user options
19  strChoice = "" # A Capture the user option selection
20  objFile = None
```

figure 1: Data section

Notice line 12 # -- Data -- #. This is a concept we are learning about called Seperation of Concerns. Basically it is a way to organize code in a way that is consistant with other programmers. This makes it easy for others to hop right into code, and have some familiararity.

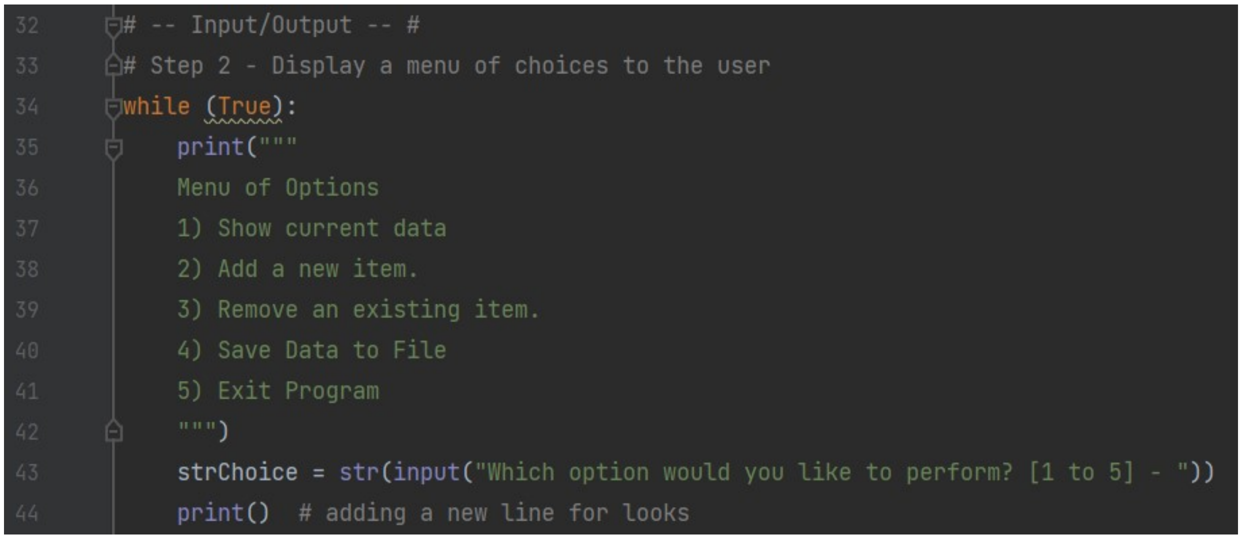
"Data" is the first section, "Processing" is second, and "Presentation" is third. "Data" is used for items that will be used throughout the program. They are generally established at the beginning of the code. From my understanding, "Processing" is where values are being calculated or generated. Again, from my understanding, "Presentation" is where I/O is handled, interaction between the computer and user. I will need to continue to keep this concept in mind for future projects, but didn't get much use out of it for this assignment. Anyways, moving on.



```python
# -- 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)
objFile = open(strFile, 'r')
for row in objFile:
    lstRow = row.split(",")
    dicRow = {"task": lstRow[0], "priority": lstRow[1].strip()}
    lstTable.append(dicRow)
objFile.close()
```

figure 2: Processing Section

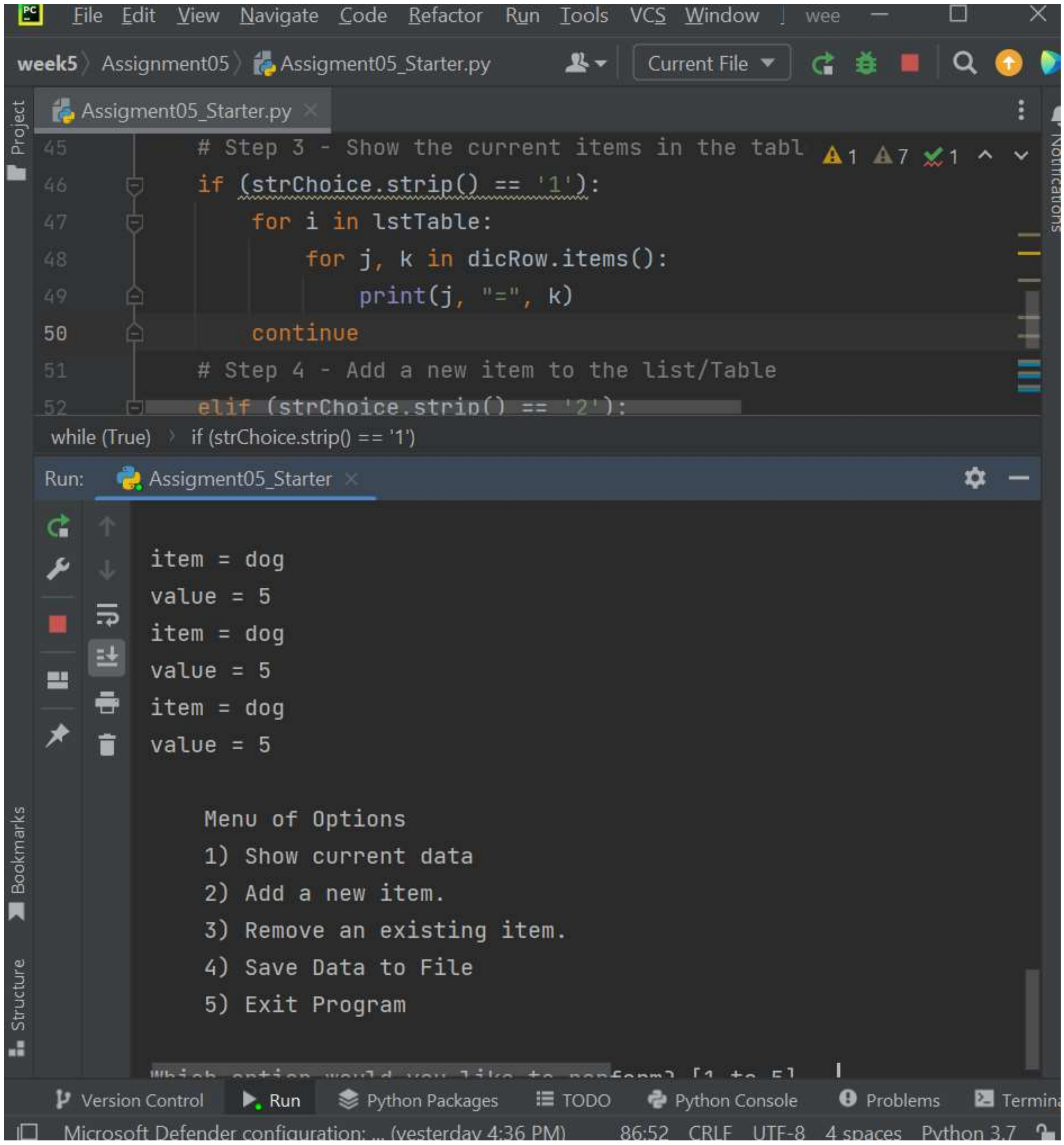
Next I open a file and read it's contents, and store the data to a list full of individual dictionaries. Here I open the file and assign it to the variable "objFile" with read permissions. Then I loop through all rows in the file, and assign those values to individual dictionaries. Last I append each dictionary into the list "lstTable", and close the file. I was able to follow the example in the book, so I didn't have any trouble with this part. It is getting a bit harder for me to remember all the exact syntax, from all the new things we are learning, however it comes to me quickly when I see examples.



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

figure 3: Presentation Section

The screenshot above shows the menu that was provided in the template. I made no changes here.



```
    # Step 3 - Show the current items in the tabl
    if (strChoice.strip() == '1'):
        for i in lstTable:
            for j, k in dicRow.items():
                print(j, "=", k)
        continue
    # Step 4 - Add a new item to the list/Table
    elif (strChoice.strip() == '2'):
```

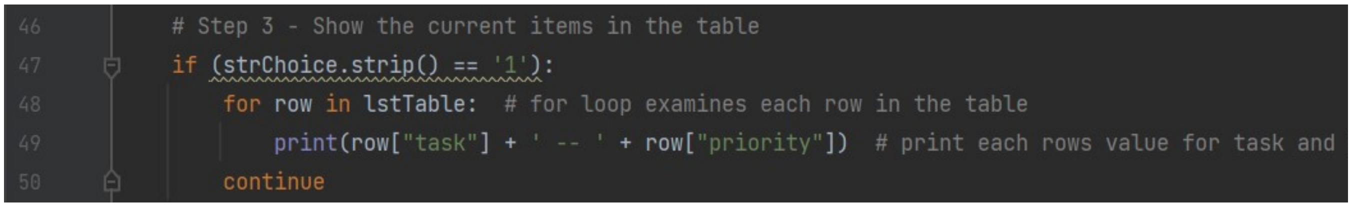
```
item = dog
value = 5
item = dog
value = 5
item = dog
value = 5

    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

Which option would you like to perform? [1 to 5]
```

figure 4: First attempt at displaying list

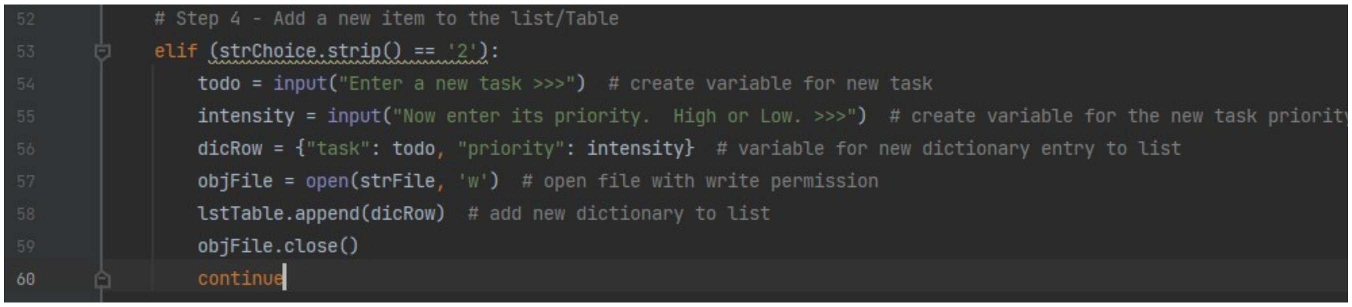
The above screenshot shows a most excellant failure at getting the list to display on the screen. I tried a few different iterations of this, before giving up and looking at the examples in the homework.



```
46      # Step 3 - Show the current items in the table
47      if (strChoice.strip() == '1'):
48          for row in lstTable:  # for loop examines each row in the table
49              print(row["task"] + ' -- ' + row["priority"])  # print each rows value for task and
50          continue
```

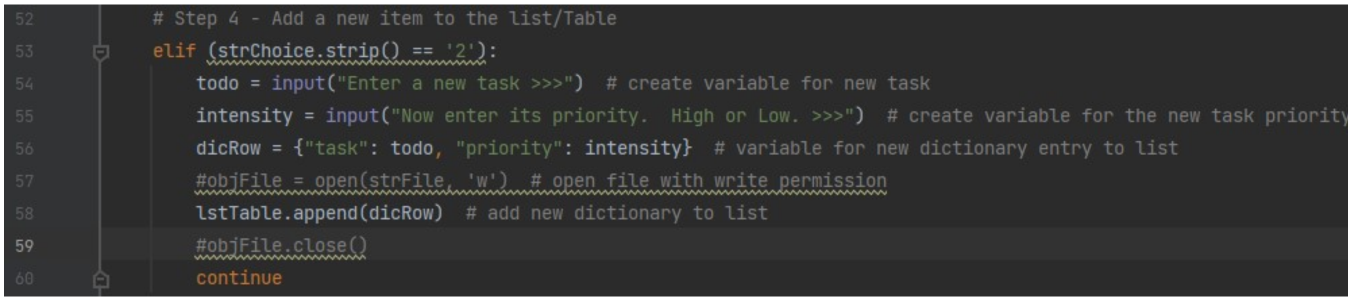
figure 5: Choice #1

In figure 5, we get to the first code for the users selection. strChoice.strip() is intending to strip out any spacing before or after the selection. When 1 is selected, the code loops through the list "lstTable", and not from the file. The same data exists in the file and "lstTable" at this point because the information was brought in at lines 25 thru 30. So each row (or dictionary) is examined, and reformatted to (task -- priority). 'task' and 'priority' are the keys for these dictionaries, and their values are printed in line 49.



```
52      # Step 4 - Add a new item to the list/Table
53      elif (strChoice.strip() == '2'):
54          todo = input("Enter a new task >>>")  # create variable for new task
55          intensity = input("Now enter its priority.  High or Low. >>>")  # create variable for the new task priorit
56          dicRow = {"task": todo, "priority": intensity}  # variable for new dictionary entry to list
57          objFile = open(strFile, 'w')  # open file with write permission
58          lstTable.append(dicRow)  # add new dictionary to list
59          objFile.close()
60          continue
```

figure 6: Choice #2 with unnecessary coding

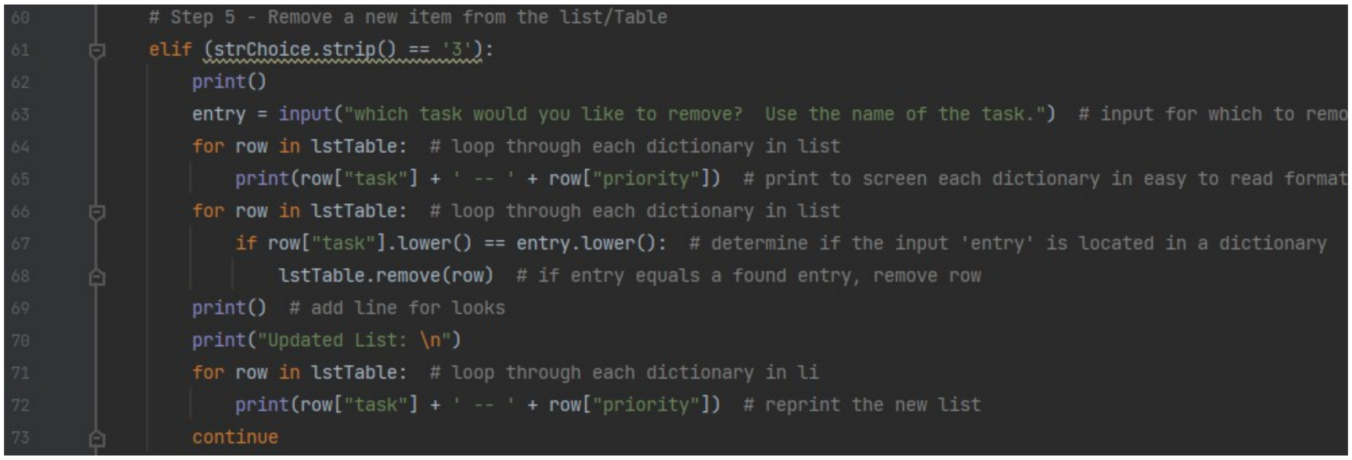


```
52      # Step 4 - Add a new item to the list/Table
53      elif (strChoice.strip() == '2'):
54          todo = input("Enter a new task >>>")  # create variable for new task
55          intensity = input("Now enter its priority.  High or Low. >>>")  # create variable for the new task priority
56          dicRow = {"task": todo, "priority": intensity}  # variable for new dictionary entry to list
57          #objFile = open(strFile, 'w')  # open file with write permission
58          lstTable.append(dicRow)  # add new dictionary to list
59          #objFile.close()
60          continue
```

figure 7: Choice #2 with updated coding

If the user chooses choice 2, they can add a new dictionary to the list. First two variables are created by user input, and a dictionary is created with these values. Each value is associated with either the key 'todo' or 'intensity'. In my previous code, i intended to open the file, write to it, and close. But now I realize that it wouldn't have even written to the file, the way it was written. Oh well, I don't need it to write to the file yet anyways, until the user decides to save.

in line 58, "dicRow" is the new dictionary, and it is added to "lstTable". Afterwards, the user is prompted with the main choices once again.

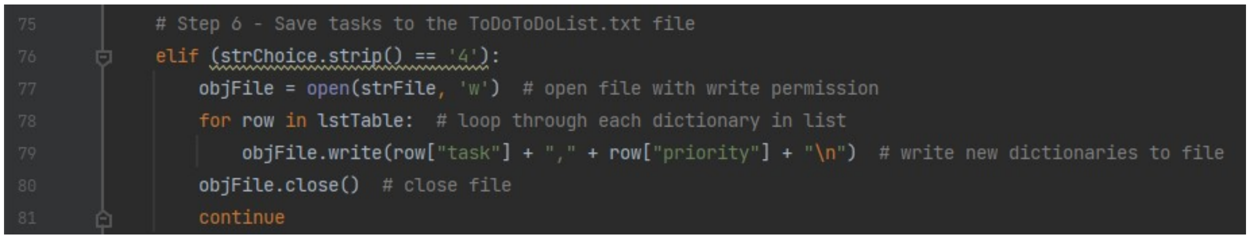


```
60    # Step 5 - Remove a new item from the list/Table
61    elif (strChoice.strip() == '3'):
62        print()
63        entry = input("which task would you like to remove?  Use the name of the task.")  # input for which to remo
64        for row in lstTable:  # loop through each dictionary in list
65            print(row["task"] + ' -- ' + row["priority"])  # print to screen each dictionary in easy to read format
66        for row in lstTable:  # loop through each dictionary in list
67            if row["task"].lower() == entry.lower():  # determine if the input 'entry' is located in a dictionary
68                lstTable.remove(row)  # if entry equals a found entry, remove row
69        print()  # add line for looks
70        print("Updated List: \n")
71        for row in lstTable:  # loop through each dictionary in li
72            print(row["task"] + ' -- ' + row["priority"])  # reprint the new list
73        continue
```

figure 8: Choice #3

Figure 8 shows Choice number #3. Here, the user can remove a dictionary by typing in the value of the first key. I begin with creating the variable "entry" and the user assignes it data. Lines 64 and 65 simply print the list, so it's easy for the user to remember which items are on it. Lines 66 thru 68 look thru the table for the string that was entered by the user. If it is found, that row is removed from the table. From here, the code prints out the new list, for the user to see.

I had the most difficulty with this part of the code. At first I tried to associate a number to each dictionary, where the user could enter the number in association, and eventually delete that row. This became frustrating, as I am not yet confident in my coding enough to write intuitively. This is where I spent the most of my time, but in doing so, I began to understand the relationship between lists and dictionaries much better. The code above is what I ended up with, which works well.

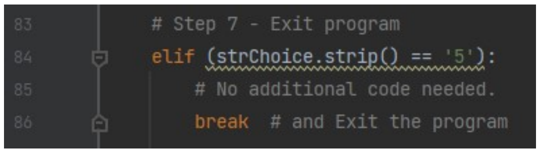


```
# Step 6 - Save tasks to the ToDoToDoList.txt file
elif (strChoice.strip() == '4'):
    objFile = open(strFile, 'w')  # open file with write permission
    for row in lstTable:  # loop through each dictionary in list
        objFile.write(row["task"] + "," + row["priority"] + "\n")  # write new dictionaries to file
    objFile.close()  # close file
    continue
```

figure 9: Choice #4

Once the user is done editing the list, selection 4 enables the saving of the list to a .txt file. The code starts by opening the file with write permissions. Each row in the list is examined, and copied to the .txt file with a certain format. The file is then closed.

I also had a bit of trouble at first, with saving the data to the file. I was trying to use .append to the .txt file, when that supposed to be used for a list. I originally use .write, but i wasn't getting return lines on the .txt file. I was hoping append would do the trick, but all I got were errors. It was pretty easy to figure that one out with the feedback I was getting when the code stopped.



```
    # Step 7 - Exit program
    elif (strChoice.strip() == '5'):
        # No additional code needed.
        break  # and Exit the program
```

figure 10: Choice #5

Last in the code is choice #5. I didn't add any code here. I noticed that other students used this area to close their file, and that probably makes the most sense, instead of opening and closing the file the entire time through the code, now that I think about it.

Conclusion

Because of my own schedule problems, I didn't have time to play around with error handling, or seperation of concerns. I will be sure to look into those during the next section. All in all, I like how things are progressing in the class. I think that this section has been the hardest for me so far. Dictionaries inside of lists has made my brain hurt on occasion, but I am slowly coming around. It was helpful to see how others are creating the same code, and I look forward to going through the rest of the work that was posted.