Name: Axel Adams
Date: Feb, 14, 2023
Course: Intro to Programming (PYTHON)

Assignment 04 - To Do List

Introduction:
Dictionaries allow the user to organize data such that a series of data are associated with a key. In this assignment. We are creating a To Do list that uses dictionaries to allow the user to add and edit items from their To Do Lists.

Topic: Pseudo-coding Assignment 04
The pseudo-coding for this assignment is provided in the prompt script from class. The variables were also defined in the script prompt.

Figure 1: Variable Definitions and Empty List Designations

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

Topic: Step 1: Generating the File and the List Table
In Step 1 of the program, we open the ToDoFile.txt. By using the “r” open mode. The rows of data in the text file are converted to a dictionary row as in Lab 5-2, and these are compiled in a list table.
Figure 2. Step

```
# -- 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(ToDoFile, "r")#Here we import the text file data and convert to the list table
for row in objFile:
    lstRow = row.split(",")
    dicRow = {"Task": lstRow[0], "Priority": lstRow[1].strip()}
    lstTable.append(dicRow)
```

Topic: Step 2: Display the Menu to the User
The menu was generated in the prompt. I added an additional prompt at the beginning of the menu to make it more explicit.

Figure 3: Step 2

```
# -- Input/Output -- #
# Step 2 - Display a menu of choices to the user
while (True):
    print("""
    How would you like to alter your To Do List?
    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
    """)
```

Topic: Show Current Data in To Do List: Step 3:
To display the list in the To Do Text file, I used method to print the dictionary rows in the list table we created in step 1. If the list table is empty, I used the length function introduced in chapter 4 of our course textbook, “Python Programming for the Absolute Beginner, 3rd ed,” to display a message stating the text file is empty if the length of the list table is zero.

Figure 4: Step 3

```
# Step 3 - Show the current items in the table
if (strChoice.strip() == '1'):
    if len(lstTable) == 0:
        print("Your To Do List is empty")#If text file is empty
    else:
        for dicRow in lstTable:
            print(dicRow["Task"]+","+dicRow["Priority"])#Print the list table
```

Topic: Add a New Item to To Do List: Step 4:
In this instance, I used a similar method as that described in Lab 5-2 to collect user input to generate a new item in the list. The user input is collected and is converted to a dictionary row. The append function described in listing 21 of the module 04 class notes is used to add the new dictionary row to the list table.

Figure 5. Step

```
# Step 4 - Add a new item to the list/Table
elif (strChoice.strip() == '2'):
    print("Add a 'Task' and 'Priority' for your To Do List Table")
    userTask = input("Enter a Task:")
    userPriority = input("Enter a Priority:")
    dicRow = {"Task": str(userTask), "Priority": str(userPriority)}
    lstTable.append(dicRow)#We add a dictionary row to the list table
```

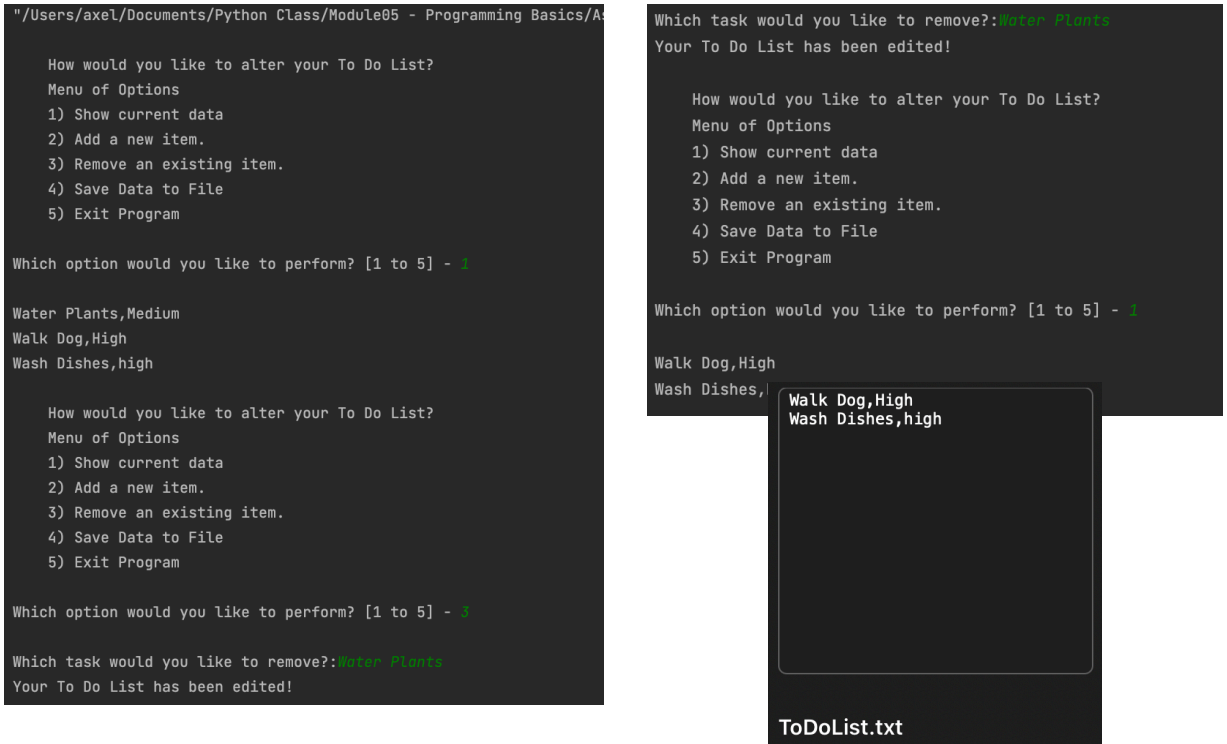
Topic: Remove a New Item from the To Do List: Step 5
This was the most challenging aspect of the code. I owe Charles Lin and reviewing his code in order to figure out a schema that would work. At first, I was looking at operations that could remove items from a list such as “del,” “pop(),” and “popitem().” However, after playing with these, I could not determine a way to make them work within current outlay of the code (based heavily on Lab 5-2). A for loop is used to identify if the user input is located in any of the rows in the list table, and the remove function listed in Listing 21 of the Module 04 class notes. Similarly to step 3, the length operation is used to determine if the list table is empty and display a message as such.

Figure 6. Step 5

```
# Step 5 - Remove a new item from the list/Table
elif (strChoice.strip() == '3'):
    if len(lstTable) == 0:
        print("The To Do List is empty.")
    else:
        strTask = str(input("Which task would you like to remove?:"))
        for row in lstTable:
            if row["Task"].lower() == strTask.lower():
                lstTable.remove(row)#we remove the item from the list table
        print("Your To Do List has been edited!")
    continue
```

One advantage of this strategy is that it allows the user to delete whatever items are in the To Do List, not just the most recent item added.

Figure 7: This screenshot demonstrates a to do list that was saved to the text file from a previous run of the code. I chose option 3 to remove “Water Plants.” This change is reflected in the associated text file.



Topic: Save Data to File: Step 6:
In this step, we open the ToDoFile.txt in a format that we can write the file, and then we write each dictionary row from the list table back to the ToDoFile.txt.

Figure 8: Step 6:

```
# Step 6 - Save tasks to the ToDoToDoList.txt file
elif (strChoice.strip() == '4'):
    objFile = open(ToDoFile, "w") #We open the list in such a way as to be able to write it
    for dicRow in lstTable: #We convert the lstTable back to dictionary format to write in text file
        objFile.write(dicRow["Task"]+","+dicRow["Priority"]+"\n")
    print("Your To Do List has been updated!")
    continue
```

Topic: Exit the Program: Step 7
In the last step, we close the program and break.

Figure 9: Step 7

```
# Step 7 - Exit program
elif (strChoice.strip() == '5'):
    objFile.close()
    break  # and Exit the program

else:
    print("Please choose from the choices above.")
```

Summary:
In this assignment, we used a prompt code to generate a menu that allows the user to add and edit items from their to do list. Crucial to this assignment is organizing the data in a series of dictionaries. This allows the user to search the keys in their to do list and remove items as they complete them.