Jay Plemons

Professor R. Root

Foundations of Programming: Python

15 May 2020

To-Do List Script

Introduction

This is a script that reads from a To-Do List text file then either adds to it or removes tasks once completed.

Drafting the Script

The script is broken into three sections - Data, Processing, and Input/Output. As with last week's assignment, the Data section at the top of script declares the variables and constants. The Processing section then loads any data from the To-Do list text file into a python list of dictionary rows (figure 5.1). "R" is passed in along with the file variable to tell the program to read the file.

```
textFile = open(objFile, "r")
for row in textFile:
    strTask, strPriority = row.split(",")
    dicRow = {"Task": strTask, "Priority": strPriority.strip()}
    lstTable.append(dicRow)
textFile.close()
```

Figure 5.1 – Processing

Next is, the Input/Output section, which is the core of the program. First a menu of options (figure 5.2) if created. White spaces are used to offset each line from each other as a stylistic option. Then a While loop starts and an input function is used to capture the user's selection.

```
strMenu = """
    Menu of Options
     1) Show To-Do list
      2) Add a New Task.
       3) Remove a Completed Task.
        4) Save Data to File
         5) Exit Program
    """
```

Figure 5.2 - Menu

A While loop is started printing the menu and an input function captures the user's choice. Option 1(figure 5.3) will display the tasks and their priority from the text file. Two print functions are used to give a title above the list of tasks, then a For Loop cycles through the list of dictionaries to display each value.

```
if strChoice.strip() == "1":
    print("| Task --- Priority")
    print("--------------------")
    for dicRow in lstTable:
        print(f"| {dicRow['Task']} --- {dicRow['Priority']}")
    continue
```

Figure 5.3 – Show To-Do list

Option 2 (figure 5.4) allows the user to add a new task and set its priority. Input functions capture the data which is the put into a dictionary. The dictionary is added to the end of the list. A print function then confirms to the user that the task was added.

```
elif strChoice.strip() == "2":
    strTask = input("What do you need to do?: ")
    strPriority = input("What is the priority level?: ")
    dicRow = {"Task": strTask, "Priority": strPriority}
    lstTable.append(dicRow)
    print(f"\n{strTask.capitalize()} has been added to the list. Let me know when it is completed.")
    continue
```

Figure 5.4 – Add a New Task

Option 3 (figure 5.5) allows the user to remove tasks already completed. First a variable, "strData," is set to False. An input function is used to ask the user which task has been completed. A For loop then checks each row to determine if the task in the list. The Lower method is used to convert both the task given by the user and the tasks in the list to lower-case letters to accurately compare them. If the task is in the list, strData is then changed to True and the task is removed from the list. A print functions congratulates the user on completing the task. Next, an If statement checks whether strData is True or False. If it is True, then a task was removed, and the script continues. If it is still False, then the task was not in the list, and a print function alerts the user. An updated list of tasks will then be printed.

```
elif strChoice.strip() == "3":
    strData = False
    strRemove = input("Which task did you complete?: ")
    for dicRow in lstTable:
        if dicRow["Task"].lower() == strRemove.lower():
            strData = True
            lstTable.remove(dicRow)
            print(f"\nCongrats on a job well done! {strRemove.capitalize()} has been removed.")
            continue
    if not strData:
        print(f"\n{strRemove.capitalize()} not found, you must have completed it earlier.")
    print("\nHere's what's left.")
    print("\n| Task --- Priority")
    print("--------------------")
    for dicRow in lstTable:
        print(f"| {dicRow['Task']} --- {dicRow['Priority']}")
    continue
```

Figure 5.5 – Remove a Completed Task

Option 4 (figure 5.6) will save the current list of tasks to the text file. If the user does not use this option, the tasks that are input will not be saved to the hard drive of the computer and will be lost when the program closes. A For loop is used to transfer each dictionary in the list from the memory of the computer into the hard drive. A print function then displays to the user confirming the data has been saved.

```
elif strChoice.strip() == "4":
    textFile = open(objFile, "w")
    for dicRow in lstTable:
        textFile.write(f"{dicRow['Task']}, {dicRow['Priority']}\n")
    print("The tasks have been saved to your To-Do list, just waiting to be completed.")
    textFile.close()
    continue
```

Figure 5.6 – Save Data to File.

Option 5 (figure 5.7) will display a message to user and break out of the While loop, ending the program.

```
elif strChoice.strip() == "5":
    print("Now go out there and complete those tasks!")
    break  # and Exit the program
```

Figure 5.7 – Exit Program

The only other option available to the user is anything other than the numbers 1 through and including 5. If the user types anything else, or nothing, and hits the carriage return, a message will be printed reminding the user of the only 5 available options (figure 5.8).

```
else:
    print("Please make a selection from 1 - 5.")
```

Figure 5.8 – Else Statement

The Result

Whether this program is run in PyCharm or Terminal, the same text file will be accessed, and updated. First, a .txt file is created in the same location as the .py file. The program is then run in PyCharm (figure 5.9). Because the file is empty, I start with option 2. I add "make bed" with a low priority (not pictured) then "mow grass" with a moderate priority. This program does not set the rules for the description of priority, so the user is free to use whichever scale they prefer.

```
Which option would you like to perform? [1 to 5]: 2

What do you need to do?: mow grass
What is the priority level?: moderate

Mow grass has been added to the list.  Let me know when it is completed.

    Menu of Options
     1) Show To-Do List
      2) Add a New Task.
       3) Remove a Completed Task.
        4) Save Data to File
         5) Exit Program

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

Figure 5.9 – Adding Tasks to To-Do List

A few more tasks are added, then option 1 is pressed to show the to-do list (figure 5.10)

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

| Task --- Priority
--------------------
| make bed --- low
| mow grass --- moderate
| eat lunch --- high
| disc golf --- ultimate
```

Figure 5.10 – Showing To-Do List

After the program is saved (not pictured) the program is then run in Terminal. Selecting Option 3 (figure 5.11) then removes a task added in PyCharm and displays a message informing the user of the removal, and the updated to-do list.

```
Which option would you like to perform? [1 to 5]: 3

Which task did you complete?: Eat Lunch

Congrats on a job well done! Eat lunch has been removed.

Here's what's left.

| Task --- Priority
--------------------
| make bed --- low
| mow grass --- moderate
| disc golf --- ultimate
```

Figure 5.11 – Removing Task in Terminal

Option 4 (figure 5.12) saves the file again from Terminal as it did in PyCharm, and a message informs the user that the information is saved.

```
Which option would you like to perform? [1 to 5]: 4

The tasks have been saved to your To-Do list, just waiting to be completed.

    Menu of Options
     1) Show To-Do List
      2) Add a New Task.
       3) Remove a Completed Task.
        4) Save Data to File
         5) Exit Program

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

Figure 5.12 – Save Data to File in Terminal

Finally, option 5 (figure 5.13) will exit the program and display a message to user.

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

Now go out there and complete those tasks!
jayplemons@Jays-iMac Assignment05 %
```

Figure 5.13 – Exit Program

The text file (figure 5.14) proves the program worked with the task and priority listed to reflect the tasks from the most recent save.

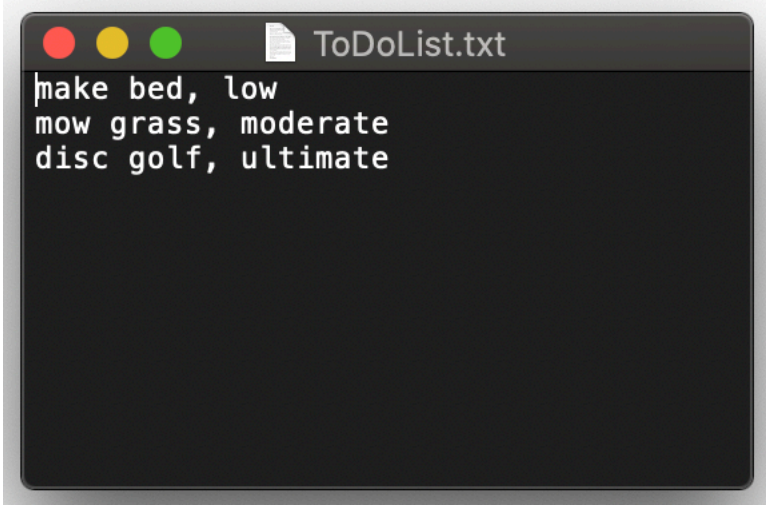

Figure 5.15. – Text File

Conclusion

This is script that will read a text file from the hard drive and display its contents to the user. The user can then add tasks and remove tasks once complete. The script can be run from various interpreters all reading and saving the information to the same text file.