Jordan Keating
May 15, 2023
Foundations of Programming: Python
Assignment 05

The To-Do List Program

Introduction

This week we'll be using both lists and dictionaries to help us build an interactive To-Do List. We'll show how you can convert an existing text document into a usable list within your code, how you can edit that list (adding to it or removing from it), and how you can update the saved version on your hard drive from within the program. Let's get started!

Working with a pre-existing document

In this project, we'll be working with a partially created script. This is really exciting because it feels more like on-the-job type work rather than a class assignment. At first glance, there are some definite advantages to having the document partially created for us - all of our variables are nicely laid out at the top, the menu options are clearly labeled and prepared for display to the end user, and our while loop has been roughly constructed. Looking at that while loop, we'll see the commands within those options have been left alone with some basic pseudo-code to get us started. How kind of them to leave us with all the fun!

Reading the text file

The first step in this script will be to load our existing data from the ToDoList text file. Since no file of that name currently exists, I'm going to simply create a new one and add some dummy data for us to pull in for testing. I added several different activities and priority levels in different rows on the ToDoList text file and added code to attempt to use the *read* command to call the data from the text file and display it back to the user.

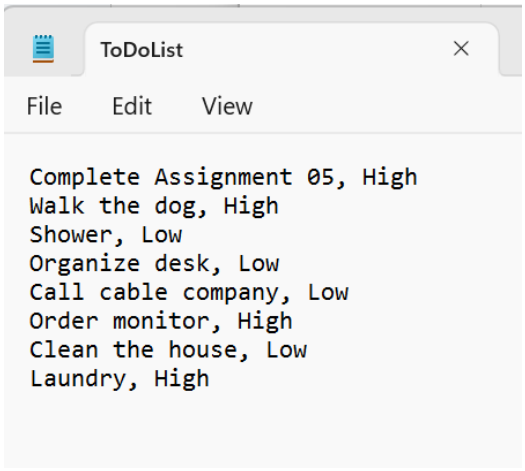

Figure 2.1 - Adding dummy data to ToDoList text file to test our read function

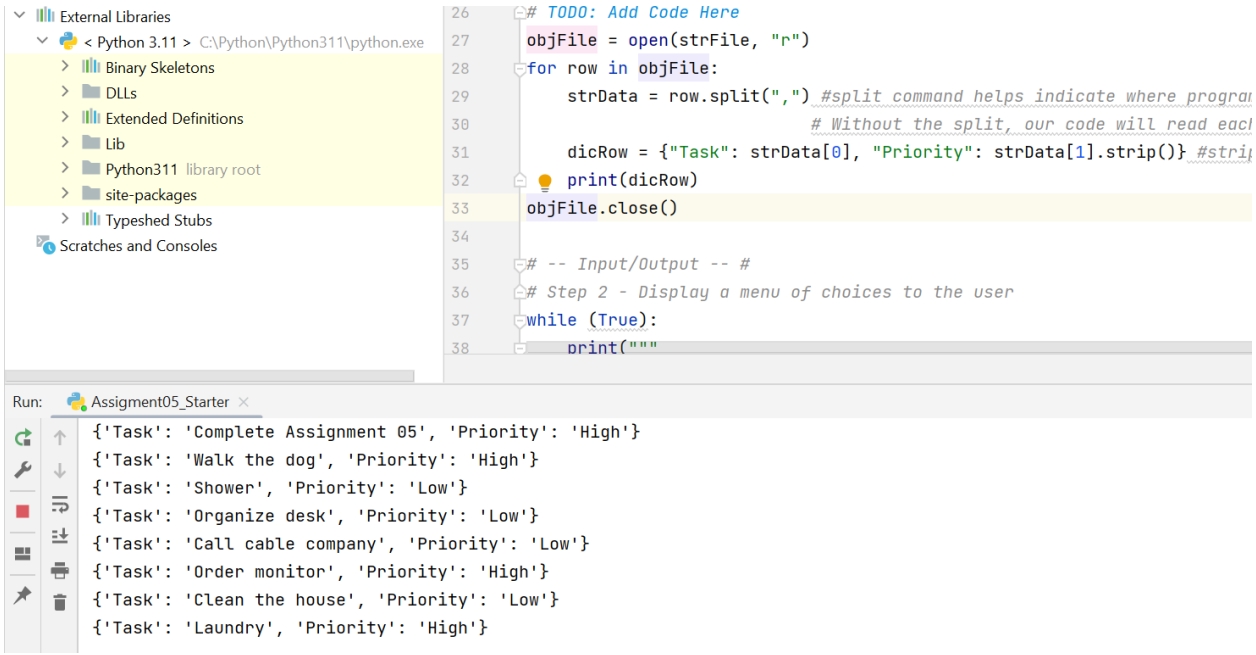

Figure 2.2 - Data from ToDoList text file is displayed back to the user!

You'll notice we used a couple of in line command functions within our script to start manipulating the data from the text file. *Split* searches the text file for the given item (in this case a comma) and breaks the row into separate strings after the given value. Since our text file is essentially a Comma Separated Value file (CSV), this will give us the desired result. Additionally, we use the *Strip* command to remove the hidden new line value from the last string of each row. You can see this exists when looking at the *print(lstRow)* output and see that the final item has a *\n* after the email. This happens when the input file has separate rows - each is a new line, so when the program reads the doc, it inputs that new line command at the end of your script items. Since we don't want it with our output here, we use the *strip* command to remove it!

Lastly here, we want to save each dictionary row into a table for use later. We can simply add a bit more code to our loop and add each new dicRow to our lstTable using the *append* command. I'll remove the dictionary print statement from that initial block of code as well and we'll give our lstTable a test run in the next section when we build out option 1!

Option 1 - Display current data

We'll start by creating a simple for loop to iterate through the data we saved into lstTable above and print it out for the user - this will help us to make sure our data saved as we intended it to when we loaded the program from the text file.

```
if (strChoice.strip() == '1'):
    # TODO: Add Code Here
    print("Here is your current To Do List:")
    for row in lstTable:
        print(row)
    continue
```

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

Here is your current To Do List:
{'Task': 'Complete Assignment 05', 'Priority': 'High'}
{'Task': 'Walk the dog', 'Priority': 'High'}
{'Task': 'Shower', 'Priority': 'Low'}
{'Task': 'Organize desk', 'Priority': 'Low'}
{'Task': 'Call cable company', 'Priority': 'Low'}
{'Task': 'Order monitor', 'Priority': 'High'}
{'Task': 'Clean the house', 'Priority': 'Low'}
{'Task': 'Laundry', 'Priority': 'High'}
```

Figure 3.1 & 3.2 - Scripting and printing the lstTable we created

With all of our data looking good, we can format the data from the list table to look a little more user friendly. I'm going to use the dictionary keys (Task & Priority in this case) to call the specific tasks and priority levels for each row of our lstTable.

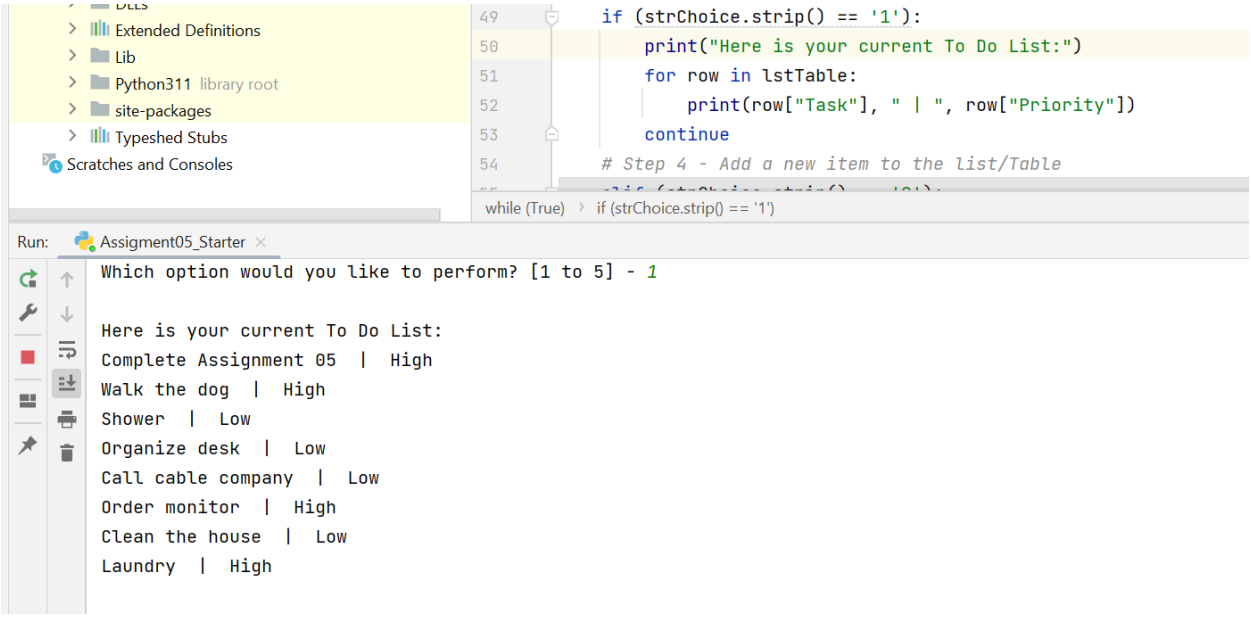

Figure 3.3 - Formatting our current list

Option 2 - Add new user input tasks to dictionary

Alright, we move on to our second option! In this section, we'll essentially be doing almost the same thing that we did when we opened up our pre-existing file and loaded that data into the dictionary table. We'll use two *input* functions to ask for user input (one for the task itself and one for the priority level), then we'll add those two pieces of data into a new dictionary row, and finally *append* that dictionary row into our existing lstTable.

```
# Step 4 - Add a new item to the list/Table
elif (strChoice.strip() == '2'):
    NewTask = input("Please enter your new task: ")
    NewPriority = input("What priority level is this task? (Low, Med, or High): ")
    NewDicRow = {"Task": NewTask, "Priority": NewPriority}
    lstTable.append(NewDicRow)
```

Figure 4.1 - Adding scripting for our new user input task and priority level

In an attempt to simplify the program and clean up the user experience, I decided to prompt the user for another new task immediately following a message letting them know the first new task was saved. To do this, I created a *while* loop inside our elif statement that essentially will

continue to loop through asking for new tasks and priority levels until the user says they have no further tasks to add. This way, they aren't prompted with the entire menu each time. This does not add any real functionality, but it does (at least in my mind) present a little cleaner experience for the end user.

```
# Step 4 - Add a new item to the list/Table
elif (strChoice.strip() == '2'):
    NewTask = input("Please enter your new task: ")
    NewPriority = input("What priority level is this task? (Low, Med, or High): ")
    NewDicRow = {"Task": NewTask, "Priority": NewPriority}
    lstTable.append(NewDicRow)
    while(True):
        another = input("\nYour new task has been saved! Add another? (Y or N): ")
        if another.lower().strip() == "y":
            NewTask = input("Please enter your new task: ")
            NewPriority = input("What priority level is this task? (Low, Med, or High): ")
            NewDicRow = {"Task": NewTask, "Priority": NewPriority}
            lstTable.append(NewDicRow)
        else:
            break
    continue
```

Figure 4.2 - Including a *while* loop for repeated task and priority additions without returning to main menu

One other fun thing I discovered while adding that extra loop - you can nest string commands on top of each other. For the data input "another", I used both .lower and .strip to ensure the code will work properly with upper or lower case letter choices and whether or not the user adds any spaces before or following their letter selection.

As you'll see in Figure 4.3, our script is working great so far. I added a couple new items and then recalled them using option 1 from the main menu!

```
Please enter your new task: Wash the car
What priority level is this task? (Low, Med, or High): Low

Your new task has been saved! Add another? (Y or N):  Y
Please enter your new task: Reading
What priority level is this task? (Low, Med, or High): High

Your new task has been saved! Add another? (Y or N): n

    Menu of Options
    1) Show current data
    2) Add a new task.
    3) Remove an existing task.
    4) Save Data to File
    5) Exit Program

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

Here is your current To Do List:

Complete Assignment 05  |  High
Walk the dog  |  High
Shower  |  Low
Organize desk  |  Low
Call cable company  |  Low
Order monitor  |  High
Clean the house  |  Low
Laundry  |  High
Wash the car  |  Low
Reading  |  High
```

Figure 4.3 - Successfully added two tasks to our list!

Option 3 - User selected removal from dictionary

To remove an item from our dictionary, I'm going to be using the *remove()* command and a *for* loop to iterate through our existing table and remove the item that's been requested from the user. First, we'll need to get the item to be removed from the user, so a simple *input* function is up first which we'll save to a new variable itemRemove. To make things easier on our user, I'm also going to have option 3 automatically call up all of the existing Tasks on the lstTable so far - this will hopefully help with the user inputs to make sure they're usable when we're looking for matches later.

```
# Step 5 - Remove a new item from the list/Table
elif (strChoice.strip() == '3'):
    print("As a reminder, here is your current To-Do List:")
    for row in lstTable:
        print(row["Task"])
    itemRemove = input("\nPlease type the name of the item you'd like to remove: ")
```

Figure 5.1 - Displaying the current to-do list and asking for which item user would like removed

From there, we'll create another *for* loop to again iterate through the lstTable and look for a match. We'll use an *if* to trigger our script for removing the code when the chosen Task has been found.

```
itemRemove = input("\nPlease type the name of the task you'd like to remove: ")
for row in lstTable:
    if row["Task"].lower() == itemRemove.lower():
        lstTable.remove(row)
        print("\n" + itemRemove + " has been removed!")
```

Figure 5.2 - Removing the task requested by the user and confirming

Option 4 - Save the data to the text file

We're officially in the home stretch now! We already learned about saving the data back to a text file in the last module with the HomeInventory, so we'll just go ahead and copy some of that code and use it here (adjusting for the correct txt file name of course). The other change I made was adjusting what we're actually "writing" to the txt file. Since the data is stored in dictionary rows, we have to call each key ("Task" and "Priority") rather than the index like we did with the lists (0, 1, etc).

```
ToDoList = open("ToDoList.txt", "w")
for row in lstTable:
    ToDoList.write(row["Task"] + ', ' + row["Priority"] + '\n')
ToDoList.close()
print("Data was saved! See you again soon!")
continue
```

Figure 6.1 - Saving to ToDoList.txt

Option 5 - Exit Program

Finally, we've reached the end. While we could have the program simply *break* the while loop and exit, I decided to give our user one more chance to ensure they were all finished. I did a simple *input* asking for their yes or no on exiting the program - if they say yes, we break and the program closes. If no, we continue back to the top of the *while* loop and the main menu reappears.

```
elif (strChoice.strip() == '5'):
    confirm = input("Press Y to confirm exit or N to return to menu: ")
    if confirm.lower().strip() == "y":
        break  # and Exit the program
    else:
        continue
```

Figure 7.1 - User option to return to menu or end the program

Testing

Alright, we'll now test all features in both PyCharm and CMD to ensure our script is working as intended.

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

Here is your current To Do List:

Complete Assignment 05  |  High
Walk the dog  |  Med
Shower  |  Low
Organize desk  |  Low
Call cable company  |  Low
Order monitor  |  Med
Clean the house  |  Low
Read Chapter 6  |  Med

    Menu of Options
    1) Show current data
    2) Add a new task.
    3) Remove an existing task.
    4) Save Data to File
    5) Exit Program

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

Please enter your new task: Feed Baby
What priority level is this task? (Low, Med, or High): High

Your new task has been saved! Add another? (Y or N): Y
Please enter your new task: Change Baby
What priority level is this task? (Low, Med, or High): High
```

Figure 8.1 - Running options 1 & 2 - adding two items to test our y/n functionality addition

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

As a reminder, here is your current To-Do List:
Complete Assignment 05
Walk the dog
Shower
Organize desk
Call cable company
Order monitor
Clean the house
Read Chapter 6
Feed Baby
Change Baby

Please type the name of the task you'd like to remove: Complete Assignment 05

Complete Assignment 05 has been removed!

    Menu of Options
    1) Show current data
    2) Add a new task.
    3) Remove an existing task.
    4) Save Data to File
    5) Exit Program

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

Data was saved! See you again soon!
```

```
    Menu of Options
    1) Show current data
    2) Add a new task.
    3) Remove an existing task.
    4) Save Data to File
    5) Exit Program

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

Press Y to confirm exit or N to return to menu: y

Process finished with exit code 0
```

Figure 8.2 - Running option 3 (confirms addition of items from option 2 have been added to task list), then saving data with option 4 and exiting with option 5

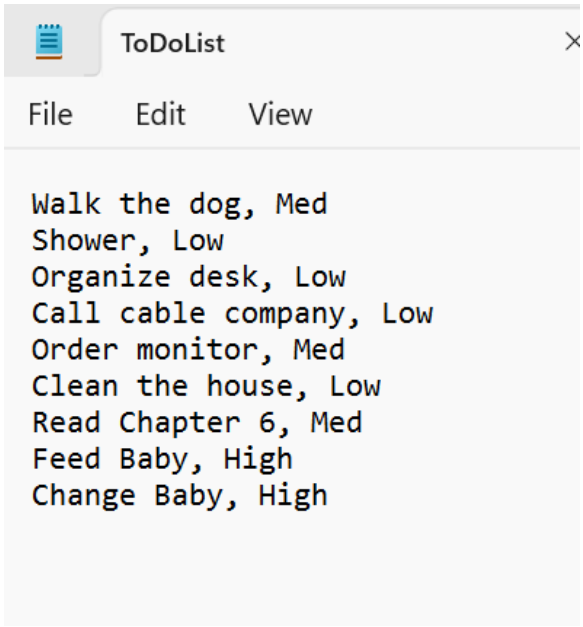

Figure 8.3 - Confirming all changes appear in ToDoList.txt - bingo! Feed and Change Baby are added and Complete Assignment 05 has been removed!

```
PS C:\_PythonClass\Assignments\Assignment05> python.exe todolist.py

    Menu of Options
    1) Show current data
    2) Add a new task.
    3) Remove an existing task.
    4) Save Data to File
    5) Exit Program

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

Here is your current To Do List:

Walk the dog  |  Med
Shower  |  Low
Organize desk  |  Low
Call cable company  |  Low
Order monitor  |  Med
Clean the house  |  Low
Read Chapter 6  |  Med
Feed Baby  |  High
Change Baby  |  High

    Menu of Options
    1) Show current data
    2) Add a new task.
    3) Remove an existing task.
    4) Save Data to File
    5) Exit Program

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

Please enter your new task: Text Mom for Mother's Day
What priority level is this task? (Low, Med, or High): High

Your new task has been saved! Add another? (Y or N): n
```

Figure 8.4 - Running Option 1 & 2 in CMD

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

As a reminder, here is your current To-Do List:
Walk the dog
Shower
Organize desk
Call cable company
Order monitor
Clean the house
Read Chapter 6
Feed Baby
Change Baby
Text Mom for Mother's Day

Please type the name of the task you'd like to remove: Organize desk

Organize desk has been removed!

    Menu of Options
    1) Show current data
    2) Add a new task.
    3) Remove an existing task.
    4) Save Data to File
    5) Exit Program

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

Data was saved!

    Menu of Options
    1) Show current data
    2) Add a new task.
    3) Remove an existing task.
    4) Save Data to File
    5) Exit Program

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

Press Y to confirm exit or N to return to menu: Y
PS C:\_PythonClass\Assignments\Assignment05>
```

Figure 8.5 - Running Option 3, 4 & 5 in CMD

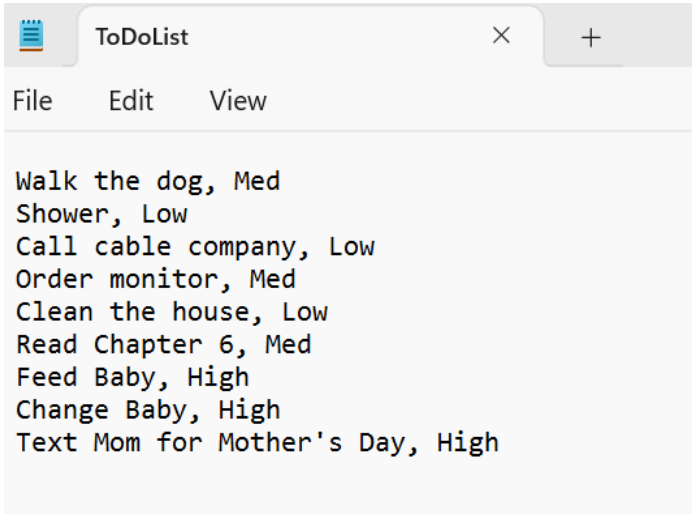

Figure 8.6 - Confirming ToDoList.txt changes

Summary

Assignment 5 was a great exercise in using dictionaries and learning how best to manipulate them. The dictionary keys provide a great way to organize pairs of data. This assignment also gave us our first look at working off another person's pre-existing code and helped us to see some of the benefits and struggles that can come from that. Overall, another super interesting assignment that will just add more flexibility to what we can do next!