Andrea Folmer

May 20th 2023

IT FDN 110 A SP 23: Foundations of Programming: Python

Assignment 05

# Lists and Dictionaries

## Introduction

Note: I have an approved date extension for this assignment to May 20th

The goal of this assignment is to learn the difference between lists and dictionaries and understand when they should be used in the program. In creating the to-do list, the value of using a dictionary is that users don't have to look up or memorize where their entry is in a list, they can just type in the key (aka value) that they want to edit. The transitions are visible in the code in line 29 where I convert a string to a dictionary row and line 100 where I turn the dictionary row back into a string. The reason for the conversion is so that I can save my data as a list of comma separated values for easy storage.

The other challenge of this assignment was starting with a script template created by someone else with the separation of concerns already mapped. This is a relatively straightforward program and the template was easy to complete, but it did take a few minutes to understand the variables defined in the program and how they were intended to be used. In a longer, more complex program or a template without comments, I might have to hand raise at the team scrum meeting and ask for help understanding the data flow in and out of the program. I might also have encountered an issue with the fact that the entire program is one big wall of code. The flow here is logical, but when bug testing or tracing inputs and out puts it's easier to separate out code blocks into functions.

## Topic 1: Load existing data to the to-do list

The first step in this program, after defining variables and constants, is to check and see if there are any existing items in the to-do list. Trying to open an empty or missing file creates an error that stops the program, so I used a for loop within a try, except decision tree to load data. To load data, I used the split function to separate strings by "," into values for my dictionary. This is one of the steps in the program where I alternate between a string and a dictionary entry. In line 29 I create a dictionary row with the keys Item and Priority based on the string keys 0 and 1.

```
# -- Processing -- #
# Step 1 - When the program starts, load any data you have
# Open To Do List file, append current data to lstTable, close To Do List
try:
    with open(objFile, "r") as currentlist:
        for task in currentlist:
            row = task.split(",")
            dicRow = {"Item": row[0], "Priority": row[1]}
            lstTable.append(dicRow)
        currentlist.close()
# Try/except used here to let user know if they don't have a to-do list file
except:
    print("You do not have an existing to-do list.")
```

***Figure 1: Code to import existing to-do items or let the user know that their list is empty***

## Topic 2: Show items in current to-do list

To show the items in the current table I used the len() function to check and see if there are any items in the list. If there are existing items, they print using a for loop on the table lstTable.

```
# Step 3 - Show the current items in the table
if strChoice.strip() == '1':
    # Notify user if to-do list is empty
    if len(lstTable) == 0:
        print("Your to-do list is empty")
    # Print formatted to-do list
    else:
        print("Item \t Priority")
        for task in lstTable:
            print(task["Item"] + "\t" + task["Priority"])
    continue
```

***Figure 2: Code to print list***

## Topic 3: Add new items to to-do list

The value of using a list vs. a dictionary is demonstrated in this step to add new items to the to-do list. I was able to add a check to see if the item was already in the dictionary using the key “Item.” If a task is already listed as an item the user gets an error message. Otherwise, a new item is added to the dictionary.

```
63      # Step 4 - Add a new item to the list/Table
64      elif strChoice.strip() == '2':
65          # Obtain user input: list item and priority
66          task = input("What is the name of the item? ")
67          # Check to see if item is already in to-do list
68          if task in lstTable:
69              print("This item is already in your list.")
70          # Obtain priority and enter item into list
71          else:
72              priority = input("What is the item priority? ")
73              dicRow = {"Item": task, "Priority": priority}
74              lstTable.append(dicRow)
75          continue
```

***Figure 3: Code to enter new items***

## Topic 4: Remove items from to-do list

Removing items from the list leverages the same benefit of using a dictionary as adding items to the list. The program first checks to see if there are any items in the list and lets the user know if it's empty. Next, it checks to see if the item the user wants to remove is in the list. I used the variable remove to flag that the if statement found one true condition, and then let the user know if the item was not found.

```
76
77      # Step 5 - Remove a new item from the list/Table
78      elif strChoice.strip() == '3':
79          # Check to see if there is data in table to remove
80          if len(lstTable) == 0:
81              print("Your to-do list is empty")
82          # Remove item from table
83          else:
84              task = input("Enter the item to remove: ")
85              remove = 0  # Internal variable to check that item is in list and notify user
86              for row in lstTable:
87                  if row["Item"] == task:
88                      lstTable.remove(row)
89                      remove = 1
90              if remove == 0:
91                  print("Item not found.")
92          continue
```

***Figure 4: Code to remove items from list***

## Topic 5: Saving to-do list to a text file

The final step in the program is to save the user's to-do list to a text file that can be opened and edited later or just printed out and stuck to the refrigerator. This is done by converting our dictionary values into a string with each value separated by a comma, which will be used to convert back into a dictionary when the program is run again (lines 26 – 34).

```
93
94      # Step 6 - Save tasks to the ToDoToDoList.txt file
95      elif strChoice.strip() == '4':
96          # Create new save file or overwrite existing
97          currentlist = open(objFile, "w")
98          # Save dictionary rows as comma separated values
99          for row in lstTable:
100             strData = (row["Item"] + "," + row["Priority"] + "\n")
101             currentlist.write(strData)
102         currentlist.close()
103         print("List saved")
104         continue
```

***Figure 4: Code to save items to file as comma separated values***

## Summary

In this project I practiced filling in a template created by someone else, converting my data between lists and dictionaries, and posting my work to GitHub. This will prepare me to work on complex projects where multiple people work on the same code. I've also learned how to save and import data as comma separated values, one of the most common ways to move data between applications.

```
What is the name of the item? mow lawn
What is the item priority? high

    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] - 2

What is the name of the item? paint fence
What is the item priority? low

    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] - 1

Item     Priority
mow lawn    high
paint fence low
```

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

Enter the item to remove: mow lawn

    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] - 2

What is the name of the item? buy gum
What is the item priority? medium

    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] - 1

Item     Priority
paint fence low
buy gum medium
```

***Figure 5: Code executing in PyCharm***

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

What is the name of the item? mow lawn
What is the item priority? high

    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] - 1

Item     Priority
mow lawn        high

    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] -
```

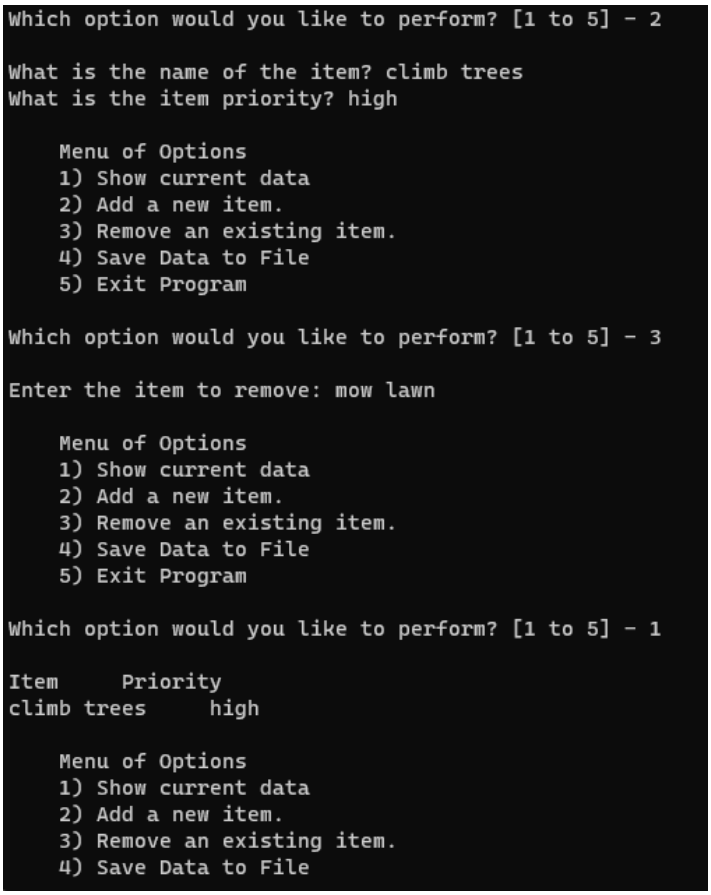


***Figure 6: Code executing in command prompt***

## References


1. Dawson, Michael, Python Programming for the Absolute Beginner, 3rd ed. Boston, MA: Cengage Learning, Inc, 2010.
2. W3 Schools, "Python Try Except." [Online]. Available https://www.w3schools.com/python/python_try_except.asp [Accessed May 20th 2023].
3. W3 Schools, "Python String Split." [Online]. Available Python String split() Method (w3schools.com) [Accessed May 20th 2023].