Kyle Odland
May 17, 2020
Foundations of Programming: Python
Assignment 05

To-Do List

Introduction

The fifth assignment for Foundations of Programming: Python was to modify an existing script for a To-Do list, to add functionality to the program. The program works by loading an existing to-do list, then giving the user a menu of options - to view the current to-do list, to add a new task, to remove a task, to save the to-do list to a file, and to exit the program.

Coding

The starter code given in the assignment provided a structure for how the program should operate. Step 1 was to load data from a ToDo List text file into memory. As discussed in Module 5, this can be accomplished with python dictionary sequences. The ToDo list text file is structured like Task,Priority, with a new line for each item. I added the data in each line to a dictionary, with keys of “Task” and “Priority”, and values pulled from the file. Each dictionary (corresponding to one line of the text file) became a row in the table of data. Listing 1 shows the code to accomplish Step 1. When I started the program, there was no seed file with the ToDo tasks. I had to comment out this portion of the program that loaded the file, until later after I had saved a task list to the text file.

```
# Step 1 - When the program starts, load the any data you have
# in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)
loadFile = open(objFile, "r")
for row in loadFile:
  lstRow = row.split(",")  # returns a list with data from each line of file
  dicRow = {"Task":lstRow[0], "Priority":lstRow[1]}
  lstTable.append(dicRow)
loadFile.close()
```

Listing 1: Code for Step 1, to load in data from file

The rest of the starter code was structured as a menu of options for the user, with each choice leading to a different task for the program to accomplish. The menu of options presented to the user are shown in Figure 1.

```
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
```

Figure 1: Menu of options for user

The program was structured like a while loop with a series of if/elif statements, with each of the user choices corresponding to a different step of the program that I needed to fill in. The easiest to fill in was choice 5, to exit the program. This just required a “break” statement, to get out of the while loop.

When I started this program, there was no existing ToDo text file. My first step for creating one was to tackle choice 2, adding a new item to the list (Step 4 in the starter file for the assignment). This was very similar to the example presented in Lab 5-2. I have the program ask for user input about a task and a priority, then I create a dictionary with keys of “Task” and “Priority”, and values defined by the user input. I appended this dictionary to the existing table of data. Listing 2 shows the code for Step 4, adding a task to the list.

```
# Step 4 - Add a new item to the list/Table
elif (strChoice.strip() == '2'):
  strTask = input("Enter a Task: ")
  strPriority = input("Enter a Priority: ")
  dicRow = {"Task":strTask, "Priority":strPriority + "\n"}  # make a dictionary with user inputs
  lstTable.append(dicRow)  # add the dictionary to the table of data
  continue
```

Listing 2: Code for Step 4, to add a new task to the list

Now that I could add tasks, I wanted to be able to see what was on the list. This was also similar to Lab 5-2, printing out the rows in the table of data. This is just a for loop through the rows of the table. Every row is a dictionary, so the program prints out the values in the dictionary for the keys “Task” and “Priority”. Listing 3 shows the code that accomplishes this.

```
# Step 3 - Show the current items in the table
if (strChoice.strip() == '1'):
  print("Your current to-do items are:")
  print("Task | Priority")
  for row in lstTable:
    print(row["Task"] + " | " + row["Priority"].strip())
  input("\nPress enter to continue")  # pauses for user input, so they can see the list
  continue
```

Listing 3: Code for Step 3, to print the ToDo list to the display

The next step was to write the table of data back to a file, Step 6 in the assignment starter code. This was a repeat of what was done in the HomeInventory.py program for Assignment 04, except writing dictionary elements instead of list elements to the file.

The final step for my program was to give the user the ability to remove a task from the to-do list, Step 5 in the starter code. I had the program ask the user for a task, then go through a for loop of the table of data to see if the user input matched a task in the list. If it matched, the program uses the .remove() function to remove that row from the table.

```
# Step 5 - Remove a new item from the list/Table
elif (strChoice.strip() == '3'):
  removeCounter = False
  strRemove = input("Please type in the task you'd like to remove: ")
  for row in lstTable:
    if row["Task"].lower() == strRemove.lower():  # if the task in the dictionary matches the task to
remove
      lstTable.remove(row)  # remove the row from the table
      removeCounter = True  # switch the counter to show that an item was removed
      break
  if not removeCounter:
    print("Your task was not on the list!")
  else:
    print("You removed this task")
  input("\nPress enter to continue")
  continue
```

Listing 4: Code for Step 5, to remove an item from task list

Running the Code

With every user choice covered, I was ready to test out the final product. Figure 2 shows a sample of the program running in PyCharm.

```
    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

Enter a Task: Cook dinner
Enter a Priority: 4

    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

Your current to-do items are:
Task | Priority
Run | 4
Golf | 5
dishes | 1
Cook dinner | 4

Press enter to continue
```

Figure 2: Running the code for Assignment05 in PyCharm

Running the code in the terminal gives the output shown in Figure 3.

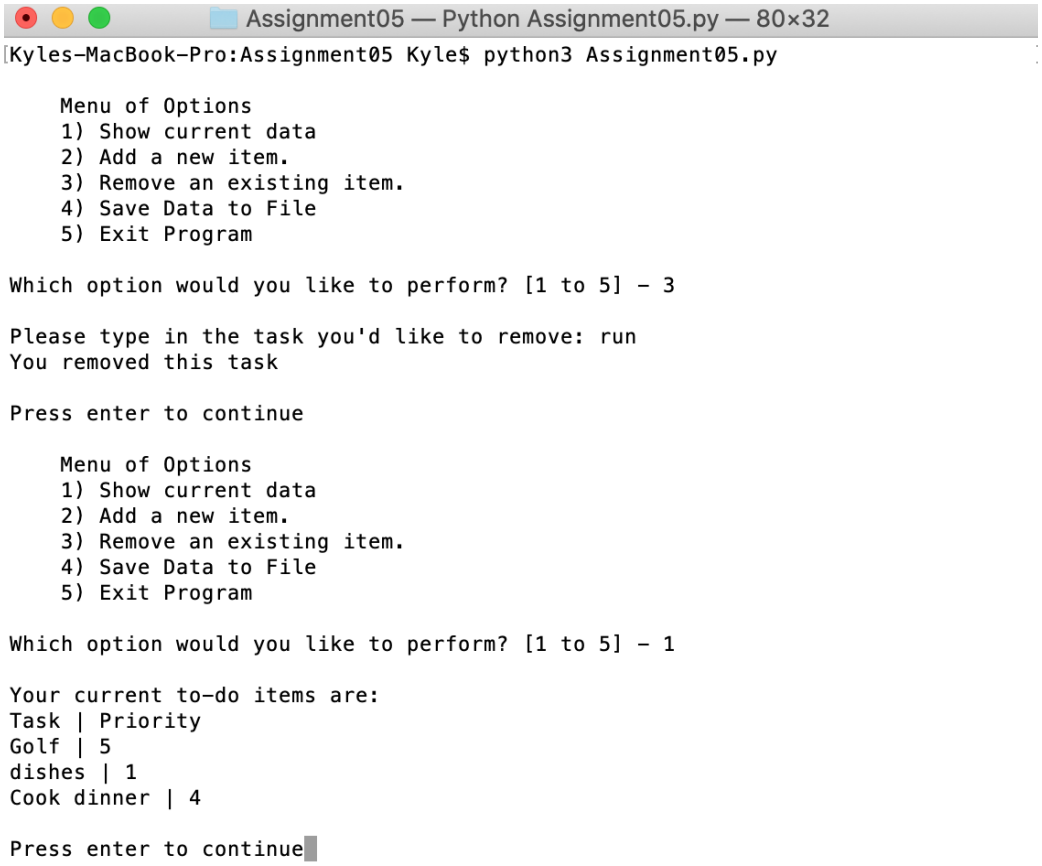

Figure 3: Running the code for Assignment05 on OSX Terminal

The code appears to work well in both PyCharm and the Terminal. The output file from after the saving the two shown iterations of the program is shown in Figure 4.

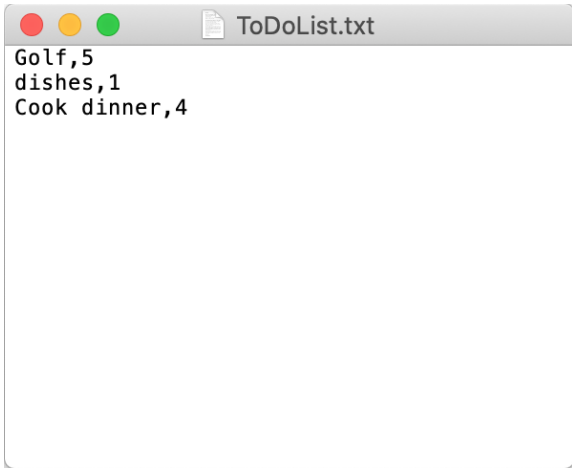

Figure 4: Output text file with a task list

Summary

In previous modules, we wrote programs to get input from the user, and print the data either to the display or to a file. This assignment added functionality by using python dictionary sequences. Now the user can perform more tasks with the data, such as removing items from the list.