Chris Perry

November 6, 2019

Foundations of Programming: Python Assignment 05

Task:Priority List Python Script

Introduction

In this assignment, I will go over how to create and run a python script that will present the user with 5 choices and ask them to input some tasks that need to be done and assign a priority to them. Also from the main menu the user can view the current data they have input, remove an item from the data the user input, choose to save it to a local file (in the working directory) and/or quit the program. I will be creating the script using Python3, PyCharm CE and macOS.

Coming up with the Code

To write my code, I used PyCharm CE - a popular python programming editor/application. I decided when coming up with this script to try to apply all the lessons from this week's assignment, if applicable, as well as some past lessons.

As apart of the assignment, I was given a template python script that had most of the variables defined and was a good starting point but needed the various code and logic added in order to make the script perform successfully.

The first decision was do I create a local txt file ahead of time in the working directory and pre-populate it with some initial data or is there a better way? It turns out, there is a better way. I was able to use the open function to create the file (in memory) if it wasn't there which seemed like a better idea. Why not let it start from scratch each time the program/script is launched and only have a file there if the user ends up deciding to save it locally. For showing the currently input data, I just used a for loop for each row in our list table and printed it to the screen.

For adding values, I decided to present some input functions to gather the task name and priority ranking. From here, the script would need to take these inputs and add them to our already established list table.

For removing, the script again needed to take some input (task name) and remove that task and its priority value from the script's list table in memory. For this, I knew that using dictionaries for validation before removing is probably the way to go.

Lastly, I needed to ensure to ask the user if they would like to save their data and if so, to save it to a local txt file in the working directory. I also recycled more code here from the last assignment but with some additional modification. If the user chose not to save their data, the script goes back to the main menu where the user can then chose the last option to simply exit the script.

How the Code Works

To start the script, I make sure the script has a proper heading and comments and the script defines all appropriate variables so they are available throughout the script. The script uses an if not loop with an open function to check if our local file is present in memory. If it's not, with the 'w+' syntax added, it will create the file if it isn't there. The script then closes the file before moving on.

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment 05
# Description: Working with Dictionaries and Files
#              When the program starts, load each "row" of data
#              in "ToDoToDoList.txt" into a python Dictionary.
#              Add the each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# ChrisPerry,11/4/19,Added code to complete assignment 5
# ChrisPerry,11/5/19,Added append functionality
# ChrisPerry,11/5/19,Added custom dictionary and dictionary.keys logic to verify
# input keys are listed before we attempt to remove them.
# ChrisPerry,11/6/19,Final code clean up and commenting.
# ------------------------------------------------------------------------------------------ #

# -- Data -- #
# declare variables and constants
objFile = "ToDoList.txt"   # 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 dictionary that acts as a 'table' of rows
strMenu = ""   # A menu of user options
strChoice = ""# A Capture the user option selection


# -- Processing -- #
# Step 1 - When the program starts, load the any data you have
# in a text file called ToDoList.txt into a python Dictionary.
# We're checking for the existence of out file in memory.
# If the file isn't there, using 'w+' creates it and also writes
# to the file in memory
if not objFile:
    open(objFile, 'w+').close()
```

The second part of the script is a menu that gets presented to the user each time a user starts the program. This section of the script was part of the template that was provided and therefore already written out.

```
# -- Input/Output -- #
# Step 2 - Display a menu of choices to the user
while (True):
    print("""
    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
    """)
    strChoice = str(input("Which option would you like to perform? [1 to 5] - "))
    print()  # adding a new line for looks
    # Step 3 - Show the current items in the table
    if (strChoice.strip() == '1'):
        for objRow in lstTable:
            print(objRow)
        continue
```

For section 3 of the script, it corresponded to choice number 1 which was to show all data. In order to show all presently stored data, the script uses a for loop for each row in the list table and then a print function to print each row back to the user as live output. The script then uses continue to circle back to the main menu.

On to section 4 of our code, this section represents action for when the user chooses choice number 2 from our main menu. This action lets the user add a task and priority to the overall list and store it in memory.

```
# Step 4 - Add a new item to the list/Table
elif (strChoice.strip() == '2'):
    # Collecting input for our user's tasks and priorities, and
    # and ensuring the values are strings, making them a dictionary row and
    # then using that to append to our file in memory.
    strData = open(objFile, "a")
    task = input("What task would you like to add? ")
    prior = int(input("What priority should this task have? "))
    lstRow = [task, str(prior)]
    dicRow = {"Task": lstRow[0], "Priority": lstRow[1].strip()}
    lstTable.append(dicRow)
    # strData.write(lstRow[0] + ',' + lstRow[1] + '\n')
    strData.close()
    continue
```

The code opens up our table stored in memory (lstTable) and tells our script that we're going to want to append some values to the table. The script then has 2 input functions to prompt the user to first input a name of the task they would like to store and then to give that task a priority. Once the script gets input from the user, the script assigns the input to 2 variables; one for task and one for priority of the task - 'task' and 'priority' respectively.

Using those variables, the script then creates and uses a list row variable (lstRow) and assigned the variables as values in the scripts list table. Next, the script converts those list values into a dictionary row (dicRow) and assigns the variables from our list row to a key "Task": and a value "Priority":. We are able to use lstRow[0] and lstRow[1] to signify we want to use the first key and value in our list row for our dictionary row. At the end of our dictionary row, we use the strip function to strip out any extra characters at the end of our row.

Once the dictionary row is set, we then use the append function to add our new values to the overall list table in memory for our script. The script then closes the table by using the close function. Without properly closing the table, the script will error out and/or not write any values. The script then uses continue to go back to the main menu.

For section 5 of our code, the script runs logic for when the user selects choice 3 from our main menu. Choice 3 represents removing an item from the list table in memory.

At first, I was trying to just remove it from the list table all in one fail swoop but I quickly realized that was not going to work. Instead, I decided to take full advantage of dictionaries and the listing through a dictionary easily.

I started by asking for input from the user to get the task they would like to remove from their current list. After that, I made a brand new dictionary separate from the list table of the script. The script now has a searchable dictionary that won't impact the current list in memory until we're ready to modify it. With that dictionary, I can now leverage the dictionary for verification on our user's input. I could easily just gather the input and pass it along to our variable and then use that variable to remove the line from the list table but what happens when the user inputs something that isn't in the list? I needed to account for that.

So I decided to make a simple if loop using the dictionary keys function. Taking the input from the user, I then will search the dictionary to see if that key is present. If it is, I will simply print a message saying "Task found! Removing.." and then pull the matching value for that key using built-in functionality of dictionaries in Python. If that input the user puts in is not a key in our dictionary which then means it's not in our list table, I print a message to the user saying "Task not found. Try again." and it bounces them back to the main menu.

This is important because if we just depended on the user always putting in the right values, we would get a KeyError exception and/or a list value error when we went to remove the incorrect value to the list. By checking it against the dictionary, we can ensure we are using the correct value when the script goes to remove the value from the list table. Since we are using a dictionary value for validation of the right key value, we can automatically pull the value to make sure that the key (task) and value (priority) are correct and match, and assign it to our value variable (priord).

We end this section of the script by using the remove function to remove the task from our list table in memory.

```
# Step 5 - Remove a new item to the list/Table
elif (strChoice.strip() == '3'):
    strData = open(objFile, "a")
    taskd = input("What is the name of the task? ")

    # creating a searchable dictionary from our list table.
    # using the keys function for dictionaries so we can
    # just ask the user for the task name (key) and auto pull
    # the priority (value)
    d = dict((i['Task'], i['Priority']) for i in lstTable)
    if taskd in d.keys():
        print("Task found! Removing.. ")
        priord = d[taskd]
    else:
        print("Task not found. Try again. ")
        continue
    # Taking our input after verifying it's in the dictionary
    # and removing the selected task out of our list table in memory
    lstRow = [taskd, str(priord)]
    dicRow = {"Task": lstRow[0], "Priority": lstRow[1].strip()}
    lstTable.remove(dicRow)
    strData.close()
    continue
```

For our final 2 sections of our script, section 6 and 7, we will explore the logic behind choices 4 and 5 which are asking the user if they want to save their data in memory to a local file and then a choice to exit the script.

In section 6, I used some code from the last assignment since it was already written. The script first displays a print function to ask the user if they would like to save their data and then presents an input function asking them to enter y or n. If the user selects y for yes, the script will use the open function with "w" and output.write function to write all the data in memory in our list table to a local file. This will then save a txt file called "ToDoList.txt" to our working directory.

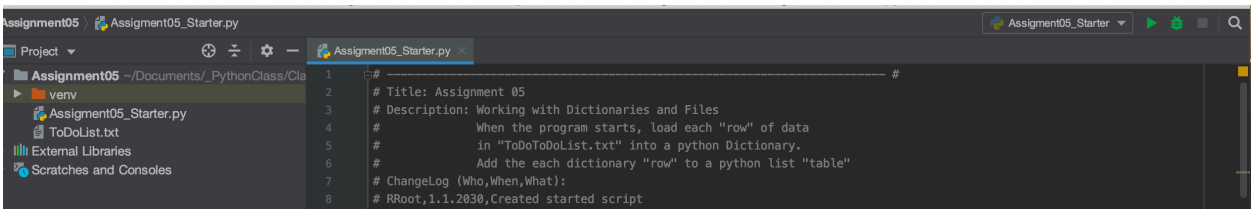
If the user selects n for no, the while loop will just break and go back to the main menu. If the user inputs anything besides y or n, the script will continually ask to input y or n until they do or they force exit the script.

For the final section, section 7, which corresponds to choice 5 (Exit the program), the program will just immediately break and the script will end.

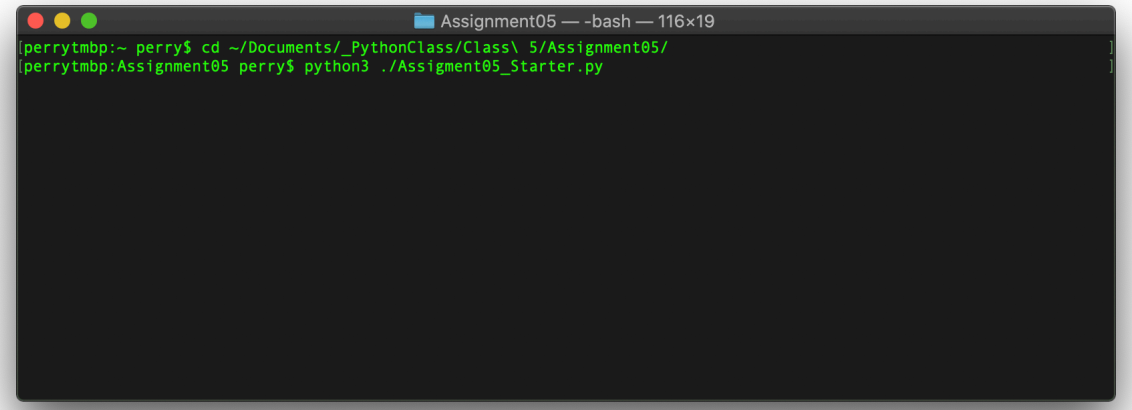
```
# Step 6 - Save tasks to the ToDoList.txt file
elif (strChoice.strip() == '4'):
    print("Would you like to save your data?: ")
    # while loop to present a yes or no prompt to the user for saving.
    while True:
        try:
            saveExit=input("Enter 'y' or 'n': ")
            if saveExit == 'y':
                with open("ToDoList.txt", "w") as output:
                    output.write(str(lstTable) + "\n")
                break
            elif saveExit == 'n':
                break
            else:
                print("Please enter 'y' or 'n'")
                continue
        except ValueError:
            print("Invalid choice. Please Try Again")
    exit

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

Launching and Saving the Script

We are using PyCharm to run a script for an assignment. You can press the play button in the upper right corner or hit run from the menu bar to run the script. Below is a screenshot of how I run it from PyCharm:



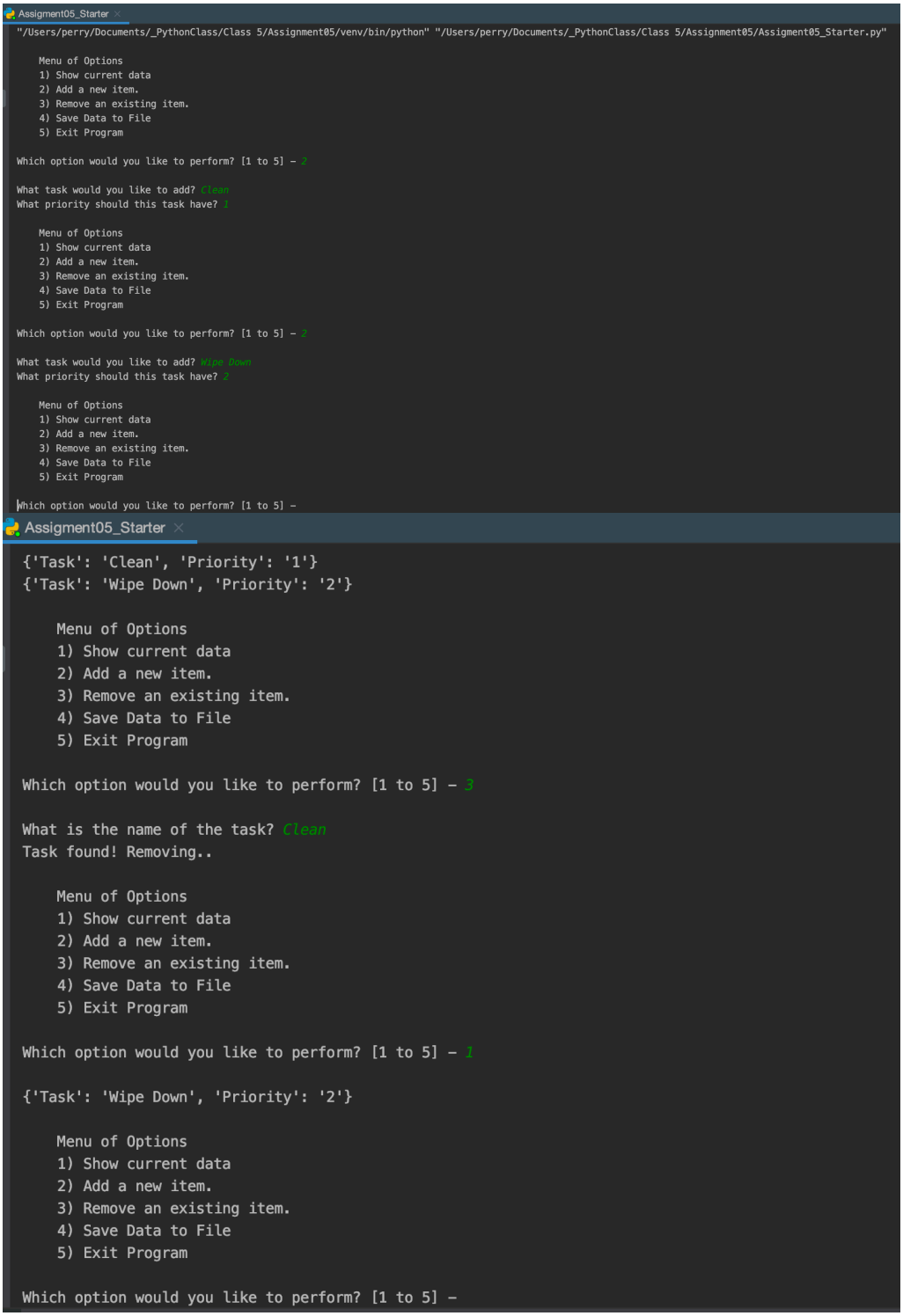
When launching from the shell, since I am using Python3 on my Mac and macOS can have more than one version of Python installed, I have to make sure to designate Python3 when I call the script to run. Also because of the way I saved the path to the txt file that is going to be written, I have to run the script from the directory I want the txt file saved to. Otherwise, I would just need to define the path in the script. Below is a screenshot of how I call the script to run from terminal:



The script is saved as a .py file (for python script) in my class directory: /Users/perry/Documents/_PythonClass/Class1/Assignment05/Assigment05_Starter.py

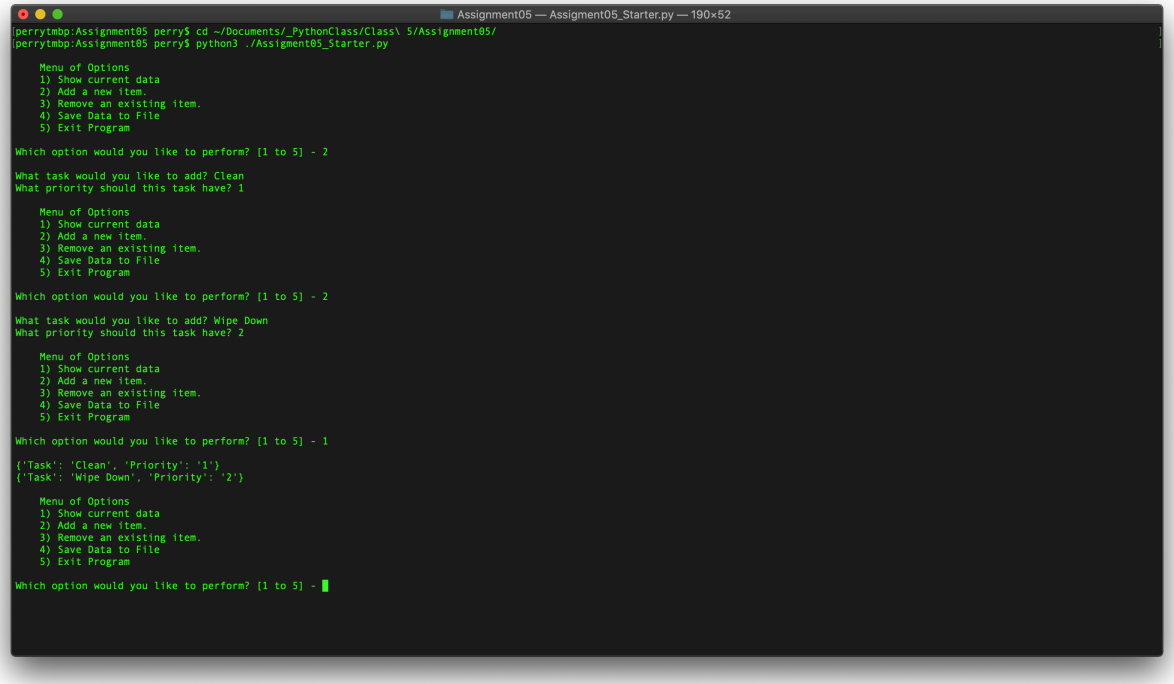
Output of the Final Script

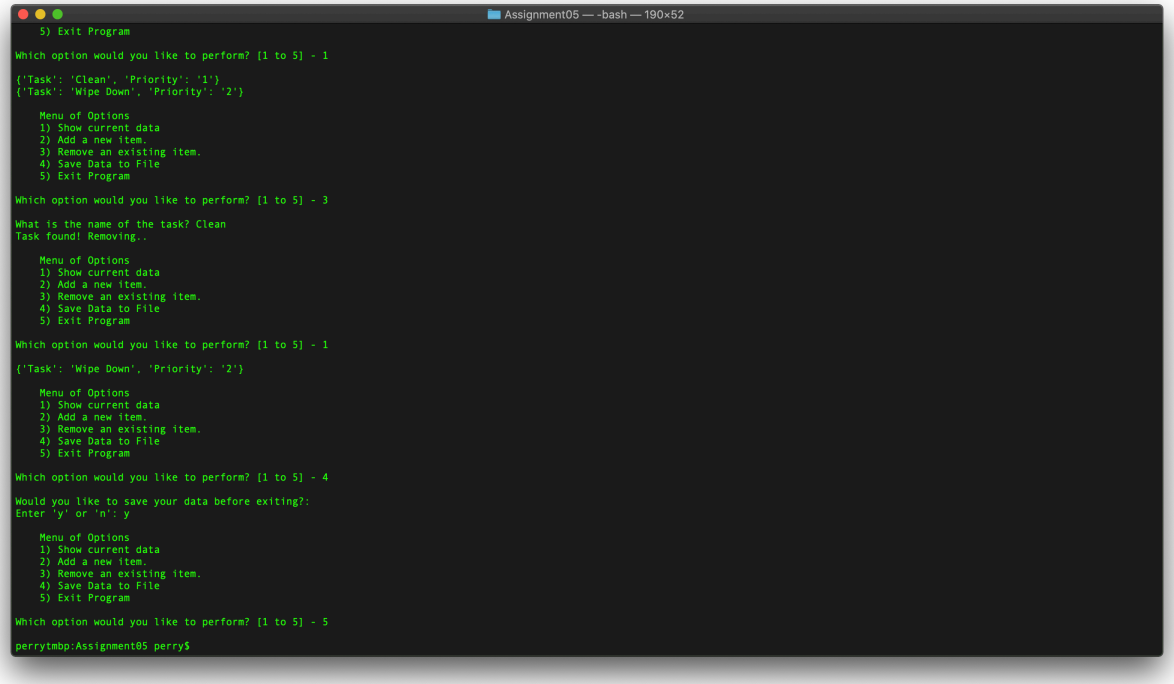
Please see below for a final output screenshot of my home inventory list Python script. First, the output while running in PyCharm:



```
    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

{'Task': 'Wipe Down', 'Priority': '2'}

    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] - 4

Would you like to save your data before exiting?:
Enter 'y' or 'n': y

    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] - 5


Process finished with exit code 0
```

Output from running the same script from the shell (terminal) in macOS:





Final Summary

During this fifth assignment, I was able to successfully show how to present a user with a menu of options, with each option doing something different. This was a combination of while loops, try and except, if, elif and else for conditional statements and to ensure that the user was entering acceptable values.

Once the user input data, we were able to successfully save the data to a list, store it, display it back to the user and if the user wanted, save the data to a local txt file. I also incorporated dictionaries to make a separate, searchable database, to verify that the input the user chose matched a value in the list table in memory. This assignment required applying logic from past lessons as well as new knowledge from this week's assignment about dictionaries and appending/removing values.