**Kate Pollock**

**November 11, 2021**

**Foundations of Programming: Python**

**Assignment05**

**GitHubURL <**[katepollock/IntrotoProg-Python: For reviewing homework files (github.com)](https://github.com/katepollock/IntrotoProg-Python)>

# **To Do List**

# Introduction

In Module 5, we continued working with lists and were introduced to dictionaries. We additionally practiced storing data in files and then accessing this data in a file and loading it into memory. Regarding dictionaries, we learned that they use keys as opposed to indices which can be useful depending on the type of data that one wants to store. We learned how to better organize our code by using separation of concerns (data, processing and presentation sections) and were introduced to functions. We additionally opened GitHub accounts, a popular source control software, to display this assignments code.

# Planning my “To Do ” Script

I followed the Assignment 5 starter file. Subsequent to filling out the header, I declared all of the variables that I intended to use during my program (**Figure 1**).

Graphical user interface, text, application, email

Description automatically generated

***Figure 1 – To Do List - Variables***

Main Body of To Do Script

After identifying and declaring the variables, the next step was to load the data from a text file into PyCharm using a nested list with dictionary rows. It occurred to me that if a user did not yet have a text file, the code would generate a “File not Found” exception. Therefore, I used a try except block to proceed with the code should the user not yet have a file. In order to properly load my data into a nested list with dictionary rows, I used a “for loop” to unpack the text rows and add them to a dictionary row with “Task” and “Priority” keys. I used the strip method to make sure that carriage returns per the text file were removed when adding to the rows. After creating a dictionary row per each line in the text file, I then appended the row to a list.

I used a while loop on the menu options that would continue to loop using “if-elif” statements until the user elected to exit the program. The user is initially prompted to select from 5 options using the input function (**Figure 2**).

Graphical user interface, text, application, email

Description automatically generated

***Figure 2 – To Do List – Loading Data and Menu***

Main Body of To Do Script continued

The following are the menu option choices with a description of the code.

Option 1 is to display current data. I used the “len” function to determine whether the table held data. If it did, I unpacked the keys from the first dictionary row. I then used a loop to display the values in each row.

Option 2 is to add a task and priority. I created two new variables “strTask” and “strPriority” and used the input function for both. Next I added both variables as values with their corresponding keys and appended them as a dictionary to the table. I additionally created a dataChange variable which I set to a Boolean value of True. This will be used to determine if data should be saved in Option 4.

Option 3 prompts the user to remove a task. I used a for loop to scan for a match between the task added by the user and the tasks per the dictionary row. If there is a match, I used the remove method to remove the dictionary row from the list. I additionally set the variable Found to a Boolean value of false. When the task is removed the Boolean changes to True. Should the variable not change to True, an if statement prints a statement telling the user the task was not found. As in Option 2, I also used the dataChange variable.

Option 4 writes the data to the file. Because the initial code saved what was in the file to the list, it was appropriate to use the write function to the file (not append which would result in repetitive data). When the user writes to the text file, dataChange becomes False indicating that the data has been saved to the file.

Option 5 breaks out the while loop, thereby exiting the program, through the use of a ‘break statement’. If data was added or removed in Option 2 or Option 3 and not added to the file in Option 4, dataChange would be set to True and prompt the user to continue the program.

The Else statement captures cases where the user has inadvertently entered an option that isn’t valid (**Figure 3**).

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

***Figure 3 – To Do List – Menu Options, Writing to File, Exiting***

# Results of Script

I ran the code in the command prompt (**Figure 4**).

Text

Description automatically generated

Text

Description automatically generated

***Figure 4: Output in Command Prompt***

I additionally ran the code in PyCharm. Below is the text file with the data. (**Figure 5**).

Graphical user interface, text, application, Word

Description automatically generated

***Figure 5: Output in File***

Both results were as expected.

# Summary

I have written the Python program above by utilizing the new concepts learned in Module 5 of this course. These concepts include collections of data in dictionaries and lists, while and for loops, nested lists, rows and tables, and opening and writing data from and to a file. I’m looking forward to learning about functions next week.