Henry Chisolm

August 23, 2021

Document Write-up:

Assignment 05

How to Modify an Existing Python Script Using the Dictionary Categorization to Create a ToDo-List.

**Introduction:**

In this paper I am going to modify an existing python script to create a program that store information of a ToDo-List through using a dictionary categorization. This script should create a .txt file and provide a menu of options to add, delete and store the user input.

**Analyzing the Scope of the Assignment:**

This assignment asks us to modify as opposed to write our own script to create a program that stores user input in a .txt file utilizing dictionary entries. The given script is broken down into 7 steps:

(1) Loading data that we already have from a .txt file.

(2) Display a Menu of choices for the user.

(3) Show the current items in the table.

(4) Add a new item to the list/Table.

(5) Remove a new item from the list/table.

(6) Save tasks to the ToDoList.txt file.

(7) Exit the program.

Additionally, the script given is broken down into sections: Data, Processing, Input/Output. All divisions meant to make the script more organized and easier to read.

**Declaring Constants and Variables:**

The script comes with preloaded variables under the data section before the program begins. This is a good organizational process in that it’s a place where we can refer to how we are storing and utilize the data and functions that we are about to work with in the script. This process is called initializing our variables and it’s like mis en place in cooking (having all the ingredients in place to start the recipe or in this case having all our variable in place to start writing the script). The only change I made to this is I incorporated strMenu = the actual menu choice late in the program.

Text

Description automatically generated

Putting this bit of code here as opposed to step 2 was easier for me to organize.

**Processing:**

**(1) Loading Data that we already have from a .txt file.**

This is so that when the program starts it will load any data we have already from the “ToDoList.txt” text file. This file may already contain data if the program has been run or if it is the first time the program is run it will create the txt file. To do this we use our initialized data from the script given. We use the command objFile = open(strFile, “w”). Previously, the variable objFile had been initialized as None as a place holder.

(initialized given script)

Setting objFile now as opening the strFile and creating a write file. The strFile had previously been initialized and designated as “ToDoList.txt”.

Here I added dicRow and defined them within {} to give both Task and Priority key definitions. I decided to hard code a few lines of these in since so something would show up when I selected the menu option 2) Show current data and the script had not been run or executed on that computer.

(script defining dicRow)

This is followed by the command lstTable.append(dicRow), which appends or adding new dictionary entries to the file that we just opened. This is the followed by the command **for** which for all the rows in lstTable: an objFile will now write and store the into the file ToDoList.txt. Finally we objFile.close so the script doesn’t keep running. The final product has preloaded value in our program.

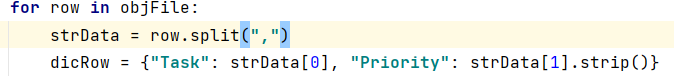
**Input/Output:**

**(2) Display a menu of choices for the user.**

The menu structure and options had already been provided in the script, so here all we are doing is starting a loop with a **while** statement setting it to true so that the program runs a menu of options with a print command. Then it sets the value strChoice = str(input(“Which option would you like to perform? [1 to 5]”. This waits for user input of 1-5 which are the choices on the menu. The structure of the remaining program in 1-5 are if statements.

**(3) Show the current Items in the table.**

This is set up with the preloaded script of **if** (strChoice.strip() == “1”):, which serves to run the task if the value 1 is selected and inputed by the user. The .strip() is added to take away anything that is entered in besides the number. This will be repeated with the remaining menu choices. Upon selecting 1, we get our preloaded values from (1) in this write up if this script has never been run on this computer and a ToDoList.txt file does not exist or we get the list if the program has already been run and altered. Here we also have an objFile command opening our str file which in turn opens or “todoList.txt” file with an “r”. This stands as read, so we are opening our file to a mode that it can’t be altered.

(Code to present data)

We then look to present the data that we have retrieved for display. The row.split(“,”) puts a comma between the values we are retrieving. We now have a dicRow dispaying our keys and the placement of strdata that we desire. It looks like this:

Text

Description automatically generated

Finally, In this option we print with a for item in lstTable and then close the file.

**(4) Add a new Item to the list/Table.**

This also begins with a preloaded script that we need to modify. In this case an elif is used and .strip is used to allow for the input of 2 from the menu which will allow a user to input new items to our list/Table in our “ToDoList.txt” file. We do this by creating two new variables for input. We create:



Both are variables that pause and wait for user input. In the case of task it is asking for a task and Priority gives options of High, Medium or Low). We next define a dicRow with a dictonary entry and key and then once again issue a commant to open our “ToDoList.txt” file through opening our strFile with “a”. This command “a” puts the ToDoList.txt in append mode which means that our added input will be stored in the file. The command objFile.write structures how the entry will be written.

Chart, scatter chart

Description automatically generated

This is followed by objFile.flush to put any data that we have entered before exiting the program into our textfile and objfile.close to start the loop over. A final print line is added to indicate that the user input has been saved.

Text

Description automatically generated(.txt file after it has been appended)

**(5) Remove a new item from the list/Table**

The preloaded elif command with the choice of 3 has been loaded into this option. We create a variable remove variable with an input function asking “Which task would you like to remove?” We then as in the previous option open our objFile but in this case we usr “r”, which opens the file in read mode. We set up a for row in objFile clause and then we then use an if command:



Where our variable removes the data in our strData from the input. The command .strip() is kept open as to remove any entry in that position

**(6) Save Tasks to the to the ToDoList.txt file.**

In this we have a precoded elif statement like the prvious option just that this one is == 4. We open our objFile, in this case to “w” or write. Additionally, we take our data that we have previously entered and store it with objFile.write in the structure we want. We finally utilize objFile.close to close the file and then add an input value that will pause the script until enter has been inputted.

**(7) Exit Program.**

Once again an elif structure has been utilized, in this case to 5. We pause the program with the an input, in this case press enter to exit. Once the input of enter has been received a line of break run. Break is a command which exits the user from the program.

**Conclusion:**

This program loads data from a .txt file which may already exist or if not creates it and then allows for a menu of options to examine, add data or edit data to the file by using dictionary entries.