George Kim

February 7, 2019

Foundations of Programming: Python

Assignment 05

GitHub Link: <https://github.com/gjkim44/IntroToProg-Python>

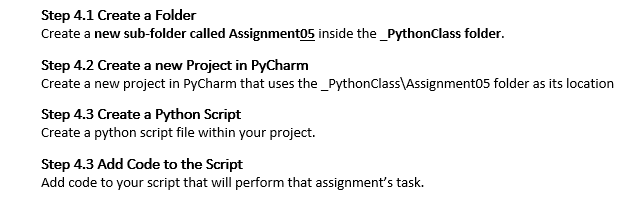
To Do List (Dictionaries and Lists)

# Introduction

Assignment 5 is an introduction to Dictionaries and Lists. This assignment is a *To Do list* with two values, that are categorized as “Task” and “Priority”. The project starts off with two lines of values already in a text file that we need to put into a python dictionary. We will then be putting data, from the user, into a dictionary then into a list, and then reversing it back into a text file. There will be menu that will have different options for the user for us to loop in and out of. We are building off the last assignment, the format will be the same, but this is way more challenging. I will be going over my process and how I accomplished this assignment.

**Starting the Script**

We start this assignment like the others in the previous assingments(**Figure 1**):



***Figure 1. Steps to create a python script*** (*Intro to Programming(Python) Chapter 5*, **Professor Randal Root**)

The python script file name ToDo.py and ToDo.txt, was saved to the project folder. I went over the directions that Professor Root gave on the assignment overview:

*This time the ToDo file will contain different columns of data (Task, Priority) which are stored in a Python Dictionary. Each Dictionary will represent one row of data and these rows of data are added to a Python List to create a table of data.*

1. ***Create*** *a text file called Todo.txt using the following data:*

*Clean House,low*

*Pay Bills,high*

1. *When the program starts,* ***load*** *each* ***row*** *of data from the ToDo.txt text file* ***into*** *a Python* ***dictionary****. (The data will be stored like a row in a table.)*

***Tip****: You can use a for loop to read a single line of text from the file and then place the data into a new dictionary object.*

1. *After you get the data in a Python dictionary,* ***Add*** *the new* ***dictionary*** *“row”* ***into*** *a Python* ***list*** *object (now the data will be managed as a table).*
2. ***Display*** *the contents of the List to the user.*
3. *Allow the user to* ***Add*** *or* ***Remove*** *tasks from the list using numbered choices. Something like this would work:*

*Menu of Options*

*1) Show current data*

*2) Add a new item.*

*3) Remove an existing item.*

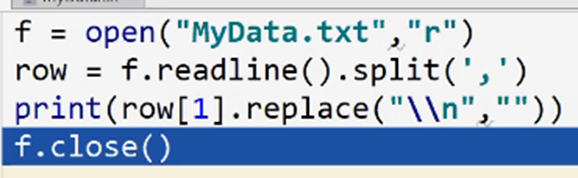
*4) Save Data to File*

*5) Exit Program*

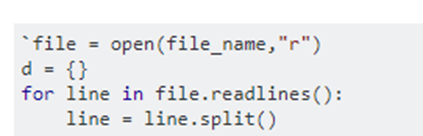
1. ***Save*** *the data from the table into the Todo.txt file when the program exits.*

*(Intro to Programming(Python) Chapter 5 Overview, Professor Randal Root)*

I began with the initial text file, ToDo.text, that had the values of Clean House,low and Pay Bills,high. I had to try and figure out how to write it to a dictionary and was at a lost till the online class session. Professor Root gave an example of reading from a text file , which was being written to a string variable called *row*. When you printed *row* , it returned a list with our values inside(**Figure 2a**). I was then able to find on StackOverflow the missing piece to creates a loop and iterate through the text file (**Figure 2b**). I used these lines of code to write my initial object file code to start the program ToDo.py.

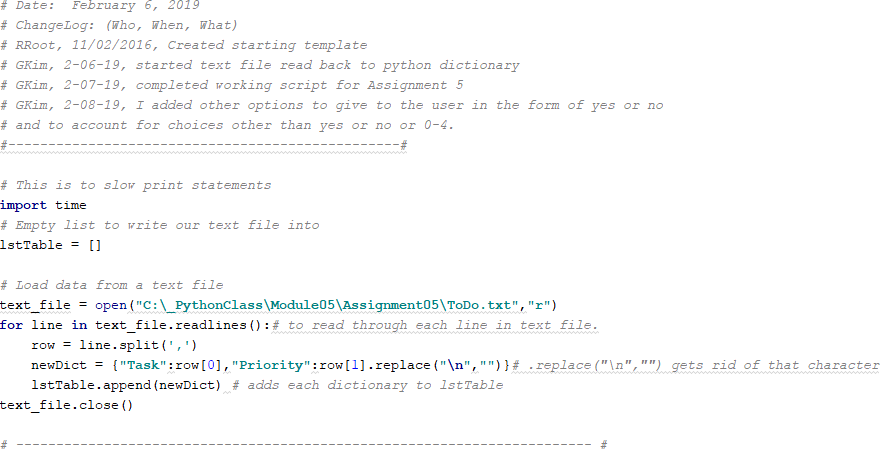


***Figure 2a. Reading from a text file and writing it to a variable the returns a list***



***Figure 2b.*** ***Reads through each line in text file*** (stackoverflow,<https://stackoverflow.com/questions/53491406/how-can-i-take-multiple-lines-of-a-text-file-as-the-values-in-a-tuple-for-a-ke>,2019 answered by user10642683)(External site)

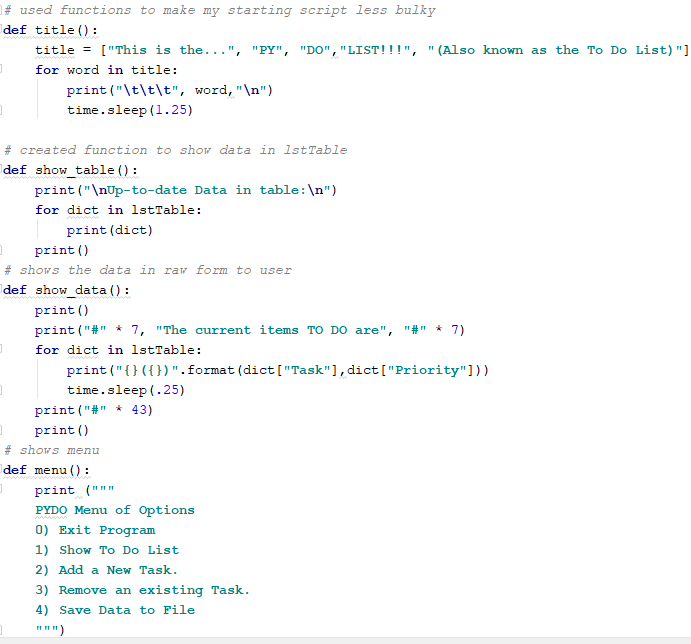
In the section comments, *Load data from a text file* (**Figure 2c***)*, you can see I open up the file with the directory in which it’s at: ***open("C:\\_PythonClass\Module05\Assignment05\ToDo.txt","r")***. I then start a *for loop* to iterate through the lines in the text file. The key here was the ***.readlines()*** in the *for loop*. This ensured to go through each row in the text file. I then had the data written to a variable, *row,* where it is split at the comma from the text file(***line.split(‘,’)***). Remember from earlier in what I wrote, that in python it writes that data in a *list* format. I was able to call each value by *index* and place them in a varible called *newDict* , which is a *dictionary* variable, and have each value from the row placed into the *newDict* variable and then appended to the *lstTable* (empty list). This gives us a list of dictionaries with the keys of *Task* and *Priority* and their values being the data from the text file. Trying to find the way to do this was my most challenging piece to figure out for this whole python script. Again, with more practice makes for better coding. The rest of the script came a lot easier to me this time around.



***Figure 2c. Start of loading data from a text file into a dictionary then into a list***.

**Creating Functions**

I decided to have a little fun and do some functions to make my body of code less chaotic with all the potential code that could be in it. If I wanted the function to be called it would just be a few characters instead a few lines of code over and over. I made four total functions to be incorporated into my body of code. Two of the functions would only be called on once, **def title()** and **def menu()**. In Figure 3 , you can see **def title()** on top and **def menu()** on the bottom. The title function starts at the opening of the script, which is displyed to the user line by line. Then menu function is called on with in the while loop, and is reffered back to with in each selction except for the *exit* option in the menu.

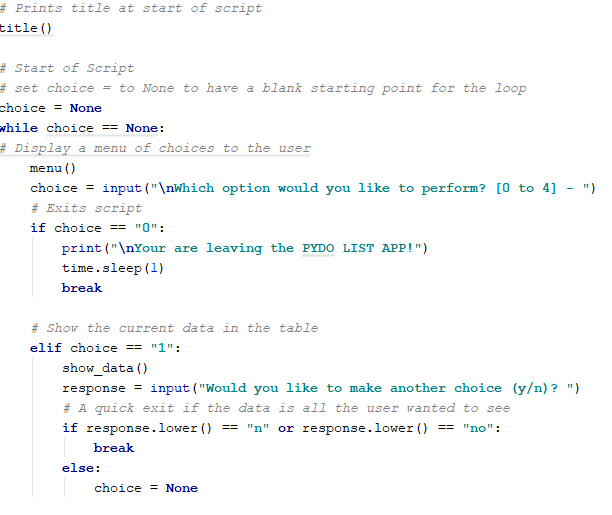


***Figure 3. Four Functions: title, show\_table, show\_data, menu***

Functions of **show\_table** and **show\_data,** were called on most in the python script. Show\_table, iterates through the *lstTable* variable, and prints out each dictionary for the user to view. Show\_data, does the same thing but shows the *raw* data to the user( ex. Clean House(low)). These functions mimic Professor Roots example that he had made a video on : <https://www.youtube.com/watch?v=hsFHXz3kvUI&t=0s&list=PLfycUyp06LG9I4194n8OvgCbmEXQt3a8z&index=15>. The output of these functions will be shown later in this paper, but I wanted to show you the lines of code that you would have to write over again if it weren’t for functions.

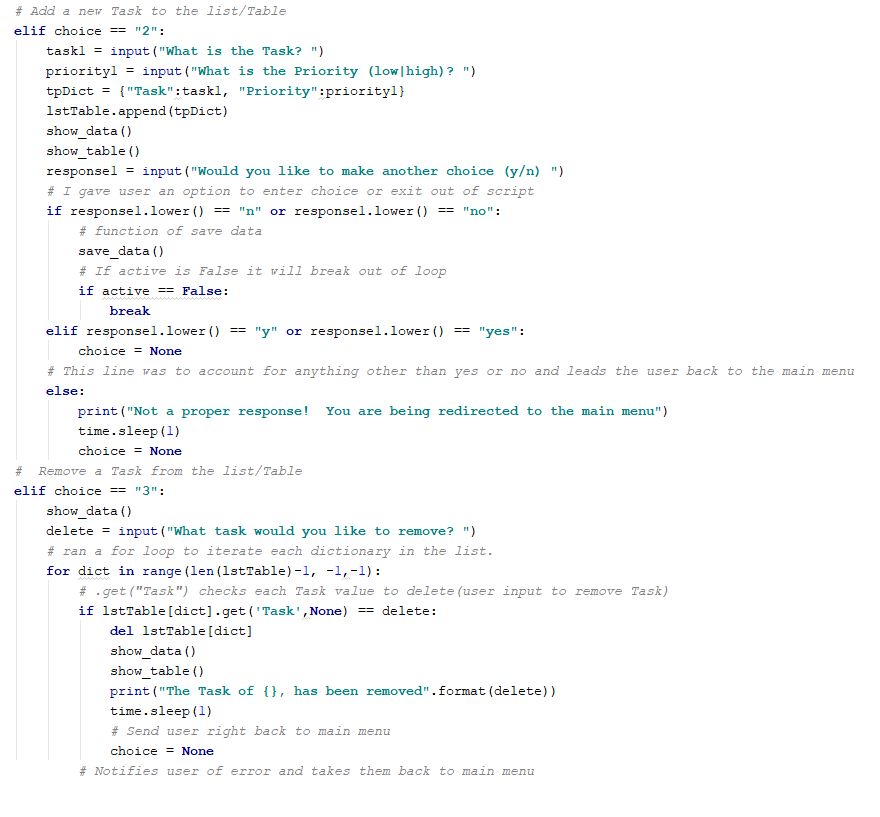
**Body of the script**

As you can can see in **Figure 4a**, the function of **title()** is called on to open the outside of the while loop. I have learned from doing the challenges in our text book, to set the choice *variable* equal to **None**. Now **None** doesn’t mean what you think. It has a value that is non-numeric and gives me another value to work with when I am dealing with multiple choices the user can input. I start my while loop at **None** and it goes through the script calling on the **menu()** function and then askes the user what choice. The first two choices , I kept fairly simple , giving the user the exit option at “0” and to show the data(**show\_data()** function) that is in the *lstTable,* in option “1”. I nested a condition in there for the user to be able to leave the program right away if all he/she wanted to do was just see the data or go back to the main menu.

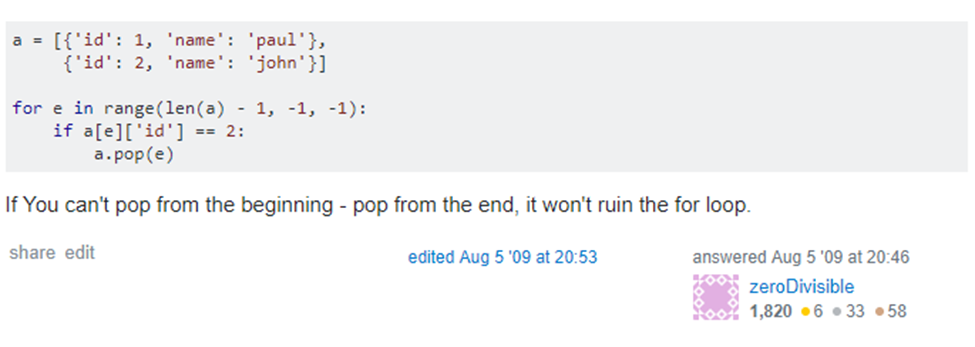


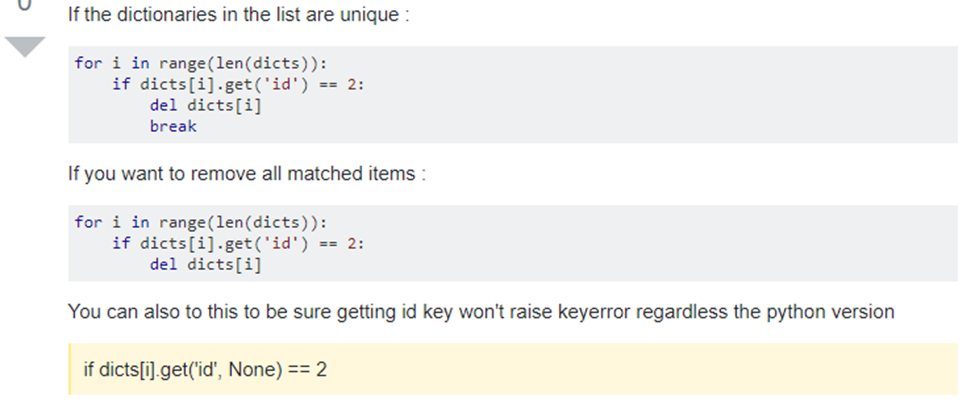
***Figure 4a. Start of Script***

**Figure 4b** shows the next two options : “2” , add a task ,“3” remove a task. In “2”, we get input from the user on what *task* they would like to add and the *priority* level (*low|high*) they would like to assign the task. We then put those values linked to their respective **keys**, in a dictionary, and then append that to the *lstTable* variable. The functions for **show\_data()** and **show\_table()** are displayed to the user to show the user the data entered. Do you see how much code there would be if the functions were not created? Instead of taking the user right back to the main menu, I ask the user if they would like to make another choice, thus giving them the option to just leave and not save the data that they just entered in case it was a mistake. You will also see if there any values other than yes or no it will prompt a message and take the user back to the menu. I added a conditional to account for user error in spelling. It redirects the user back to the main menu.



***Figure 4b. Adding/Deleting a Task***





***Figure 4c.*** ***Looping through the list table and removing the value given to us by the user and removing that dictionary from the lstTable*** (stackoverflow,<https://stackoverflow.com/questions/1235618/python-remove-dictionary-from-list>,2019 answered by nixmind and zeroDivisible.)(External Site)

Choice “3” gives the user the option to delete a task that is already in the *lstTable*. The function **show\_data()** is displayed again to have the user choose which task to remove from the *lstTable*. I got help again from my favorite website of stackoverflow (**Figure 4c**), to help iterate through a list of dictionaries. I noticed one thing wrong in **Figure 4c**, *range(len(dicts))* should subtract 1 from the length to properly index through the list of dictionaries. I realized it wasn’t working properly and remembered that the range function needed to start at the len(lstTable) -1 , up to but not including -1 and start from the last index in the table. This ensured it would go through the list and not miss an index. The ***dict[i] .get (‘id’)*** will go through each dictionary *key* and looks at the *value* to see if it matches what the user inputted. If it does it will delete the dictionary from the list. The table and data are shown again to the user with the dictionary removed and the user returns to the main menu.

The last choice, “4”, is to save the data to a text file. This part of the scripts looks very similar to the last project we did (**Figure 4e**) in Assignment 4. The main difference in that section of code is that we have to loop through the *lstTable* and write the specific data in the dictionary to the text file. I knew how to loop through the table , but to write from a dictionary to a text , well that was another thing. The web is a wonderful resource to find information on python. I found a way to write to a text file by using the **.get()** method (**Figure 4d**) again. Remember when we are writing to a file , it will be in a string format. So we can join the data with a comma using the .join() and not forgetting that when we initially wrote our text file to python, it was in a list format hence the **[ ]** in the parentheses .



***Figure 4d*** (stackoverflow, <https://stackoverflow.com/questions/29907715/write-list-of-dictionary-values-to-file>, 2019 answered by TigerhawkT3) (External Site)

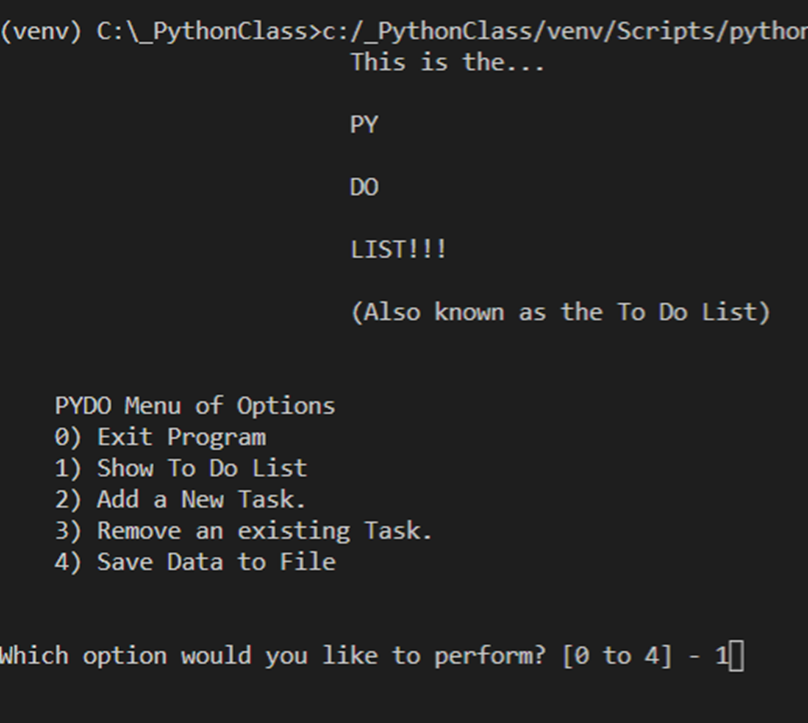


***Figure 4e. Text document with saved data which is unpacked from the tuple for loop.***

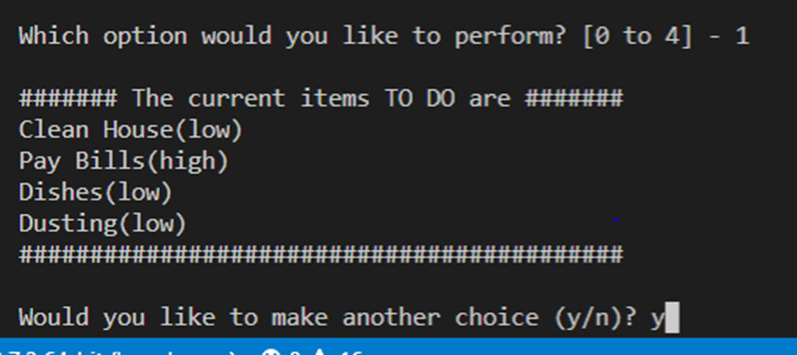
I incorporated another nested condition for user error again. If the user didn’t mean to hit the save choice option, I gave the user another out to return to the main menu or leave the script.

**Output of the Script in the Command Line**

Now we bring the code all together and run it! **Figure 5a, b** shows the script running with the title and the menu option waiting for the user to input a choice of “1”. That option will show the data in raw form to the user with a little twist to what Professor Root had to his version.

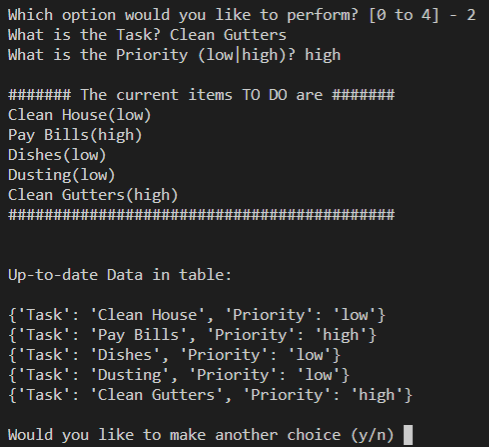


***Figure 5a. Title and Menu option***.



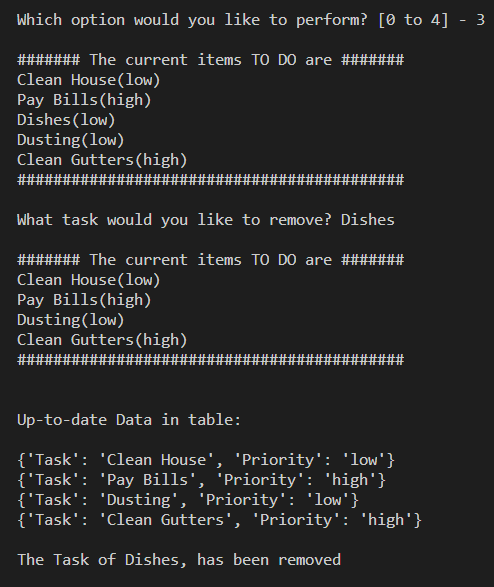
***Figure 5b. Shows Data to user in raw form.***

Choice “2” (**Figure 5c**) is adding a new task and setting the priority. It verifies this to the user by showing both data form, in a table and raw form.



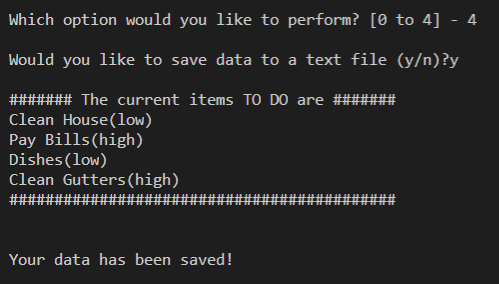
***Figure 5c. Adding a task and showing in the data table***

**Figure 5d** displays the *To-do list* and ask the user which task is to be removed. The program runs an reaffirms the updated information back to the user. The one thing I was experimenting in this section was to account for user error. I wanted to account for misspelling of the user information

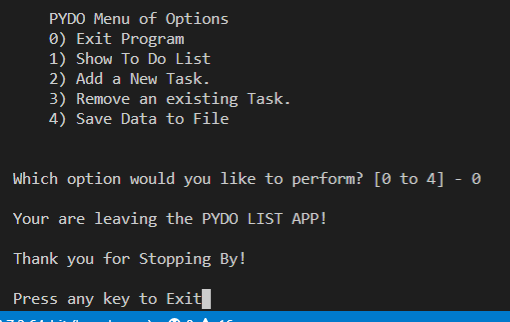


***Figure 5d. Deleting a task and updating in both forms to the user***

**Figure 5e, f** shows the file being saved to the text file and the exiting the program after going back to the main menu and choosing “0”.

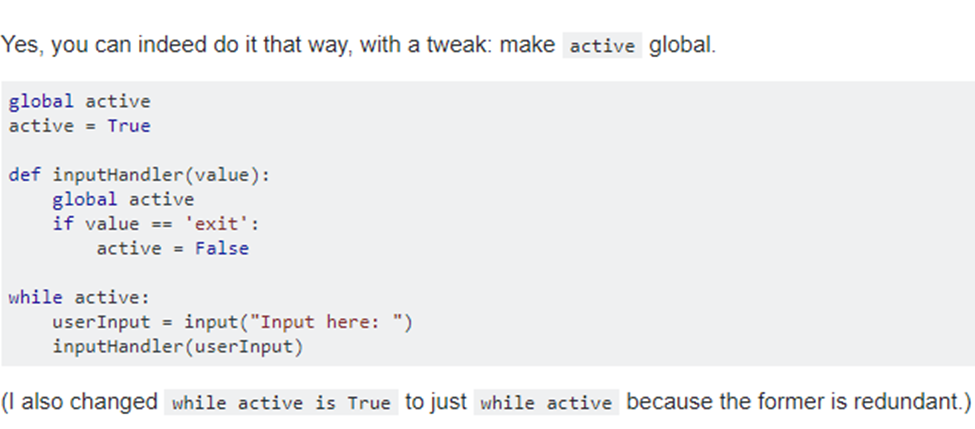


***Figure 5e. Saving the data to text file.***



***Figure 5f. Exiting the program***

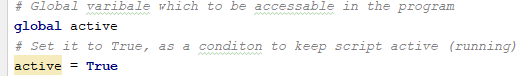
**Update**

As I was reading ahead and going over my code again, I noticed that in choice “2” that there was a nested condition that didn’t give an option for the user. I asked in choice”2” if the user would like to make another choice, which would lead them back to the main menu. What if they said “no”? And from “no” what other options could I give? The only option I could think of was if they wanted to save the date from right there in choice “2”. I started coding and ran into a bunch of issues because instead of rewriting the save code for “2”, I decided to make it a function and copy the top half of my code to the function section of my script. I found out that you can’t **break** inside a function. This led me to *global variables,* which is in chapter 6, and I started looking on the web to find some examples. Of course *stackoverlow* came to the rescue and I was able to use this code to build off of (**Figure 6a**). 

***Figure 6a. Global Variable example***

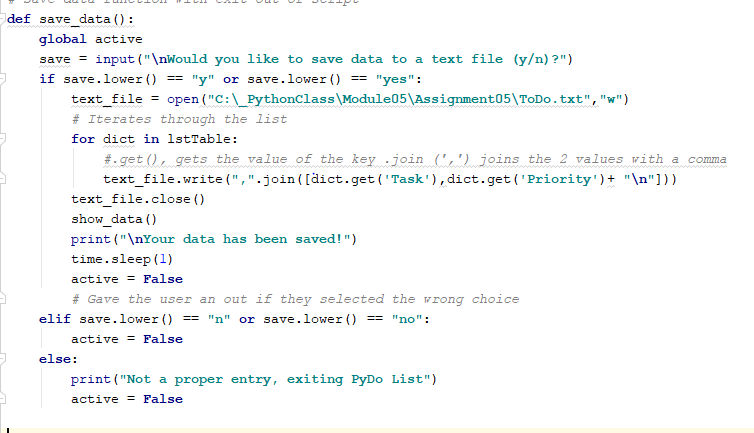
(stackoverflow,<https://stackoverflow.com/questions/33906813/can-you-break-a-while-loop-from-outside-the-loop>, 2019 answered by El'endia Starman)(External Website)

I made a global variable and used the same name, because it made sense to me, outside the while loop at the very top of the script (**Figure 6b**). Active is True for the script to run.



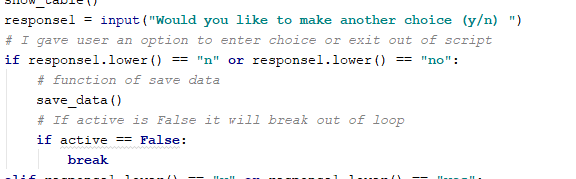
***Figure 6b***

I then copied the code from choice *“4”*, and only used the top half. The reason for this was to ask the user, after entering a new task and not wanting to make another choice in the menu, if they would like to save the data from there in choice *“2”* . Then I had to account for a no answer and ran into issues, finding out later that you cant break out of a loop in a function. I made the *Global active* **False** to exit out of the script, but this would be used for the code within the *while loop* (**Figure 6c**).



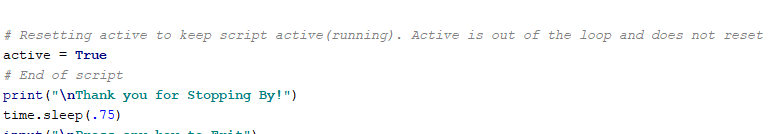
***Figure 6c***

Now that my code has gotten the response from the function *save\_data()*, if it is **False** it will break out of the loop (**Figure 6d**).



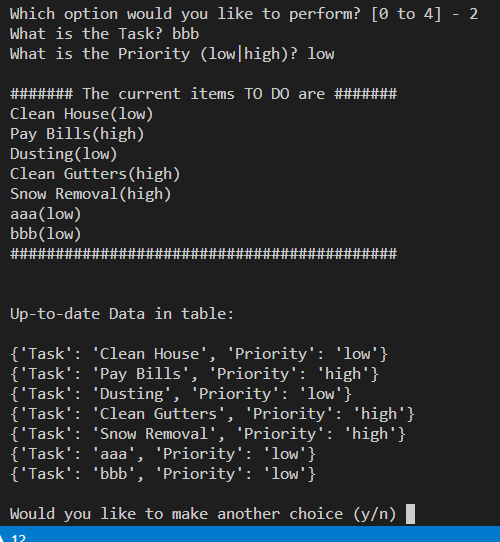
***Figure 6d.***

While I was testing my code, and I got it to work, I found out that the script would not open and run again. Then it hit me, that I had to reset the global variable back to **True** so the script would run, since it was changed to **False** it was set to exit the script and wouldn’t run. Upon exiting the script, I reset *active* back to **True** and the script runs smoothly (Figure 6e).

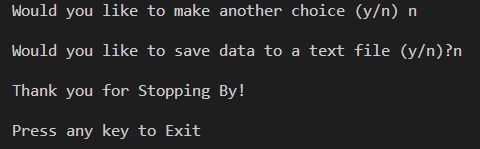


***Figure 6e***

The outputs go over the choice *“2”*, where we enter a task and don’t want to go back to the main menu. Let’s say the user made an error and doesn’t want to save, *“n”* is entered, and we exit the program. If the user wanted to save the data, it would go through the normal save process and then exit the program (**Figure 7a, b**).



***Figure 7a.***



***Figure 7b.***

**Summary**

Getting familiar with list and dictionaries will be beneficial, especially when we get into OOP. This was a great step for that, and I hope to have more practice on projects like this. This project took longer than the previous ones due to the challenging nature of writing from a text file. Having menu options and setting conditions for each one, made for great practicing in coding. Writing some functions to reduce the amount of code in the main body, helps reduce the potential clutter that can confuse other developers looking at your code. During this paper I was able to account for user errors, *task* to be removed, and made the necessary corrections and also a choice issue for the user in option “2”, which I had to use a *global variable* to break out of a loop. I hope to improve my Python coding with the *exception statements* and *global variables* to help with user error issues.