Chani Orr

8/8/21

Foundation of Programming: Python

Assignment05

Collections

Purpose of The Lab

In this module we explore collections more in depth, expanding our knowledge of lists and dictionaries, and how to write these to a file and read that file back with Python. We also made a GitHub repository, and answered the questions: What is the difference between a *List* and a *Dictionary*?; What is the difference between an *Index* and a *Key*?; How do you read *data from a file into a List*?; How do you read *data from a file into a Dictionary*?; What is the programming pattern called *“Separations of Concerns?”*; How would you use a function to organize your code? Why is a *script template* useful?; Why is *error handling using Try-Except* recommended?; What is GitHub, and why is it used?

Getting Started (Project, Script, & Pseudocode)

I departed from the decorum this time and did not write the pseudocode at the beginning of the assignment, as the starter laid out the project for me. I later regretted this, as I wished I’d had some reference to remind me where I was aside from the code itself. While I got lost in the reeds, I imagined what that pseudocode might have looked like:

| #data ⇒ declared variables   | #processing ⇒ | read ⇒ todolist.txt | print to ⇒ do list | make into ⇒ dictionary | add dictionary to Master List | | --- | --- | --- | --- | --- |   #input/output   This starts the Overarching While Loop, which contain:  \* The menu ⇒ print  \* Option 1: Show current data   | Sub For loop: For dic in list, print out each item in dic  (2 items per dic, task and priority)  ↖ Return to Master Loop | | --- |   \* Option 2: Add a new item  Take two input from user: task and priority ↴  Create dictionary with two inputs ↴  Append dictionary to Master List ↴  Continue through the Master Loop back up to the top ↴  \* Option 3: Remove an existing item  Number the list and let them choose the number to remove.   | Sub While loop: print out the numbered list↴   | Sub sub For loop: (oh we gettin fancy now)  For dic in list, print out each item in dic  (2 items per dic, task and priority)  ↖ Return to Sub While loop | | --- |   **If choice in list:** delete it ↴  ↖ Print out list again. BACK TO THE TOP!  **If choice a number but not in list:**  Tell them no ↴  ↖ Print out list again. BACK TO THE TOP!  **If choice not a number:**   Assume they want out of this loop. Break loop.↴  ↖ Return to Master Loop | | --- | --- |   \* Option 4: Save data to a file  Do you wanna save?   | SUB WHIIILLLLEEE LOOOOOPPPPP!!:  **If cool:**  Crammed items into a dictionary? yes ↴  Stuffed dictionaries into a Master List? check ↴  Zip that Master List up in a text file suitcase(“a”)   | Sub For loop: For dic in list, write out each item  in dic to the file, seperate with a comma.  (dic["priority"] + ',' + Row["task"] + '\n')  Close file  ↖ Return to Sub While Loop | | --- |   **If nah:**  ↖ Return to Master Loop | | --- | --- |   \* Option 5: Exit program  You sure?   | Not even kidding, I will do this again and that’s a threat:  **If yes:** leave  **If no:**   ↖ Return to Master Loop | | --- |     ╰ Why don’t you make a Try/Except section? “Shhh! I’m not comfortable with those yet!”╮ |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

(Image 1.0 - Chani’s imagination. Lots of blank space.)

With such a large scale project, not writing out the pseudocode ahead of time was definitely a mistake.

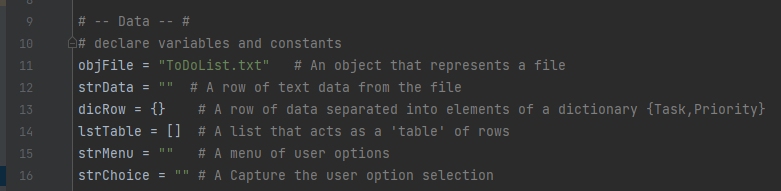
Once I had loaded the Starter Script provided by Mr. Root and familiarized myself with the contents, I set about writing the code.

Add Code To The Script

This module emphasized organizing the code into a standard format of Data, Processing, and Presentation; this is a typical practice known as Separating the Concerns. As such, I will be formatting this section of the documentation in similar style and namesake.

Data: Introducing the Data To Be Referenced

At first I used the data provided by Mr. Root, but later expanded this list to include some constants I referenced multiple times in the code.

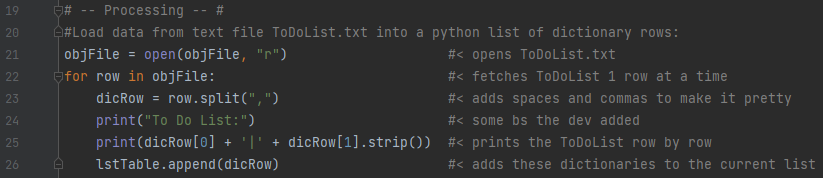


(Image 2.0 - Original script of #Data#)

(Image 3.0 - Later changes to #Data#)

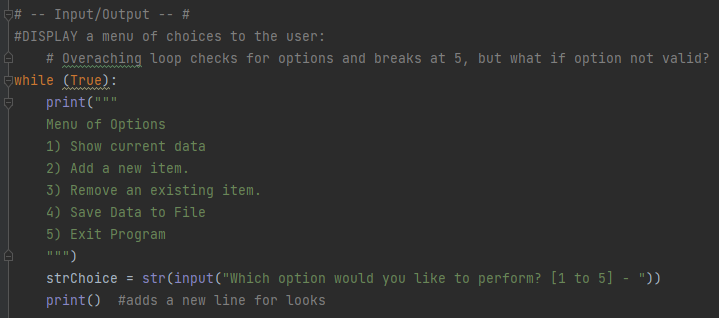
Processing: Perform Tasks On Relevant Data

This section assumes a previous text file called ToDoList and pulls the data from it. The desired effect is to pull the string data from that file and convert it to a Dictionary, then add that Dictionary to a List for each line of text in the file. I asked the computer to append these Dictionaries to the Master List that I would be referencing for the rest of the program.

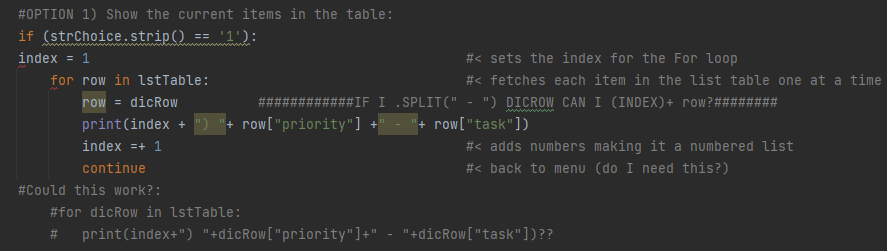
  
(Image 4.0 - Initial #Processing#)

Presentation: Input and Output

The first section of this part was provided by Mr. Root, and it describes the layout of the rest of the program. This section initiates an overarching while loop that encapsulates the remainder of the program.

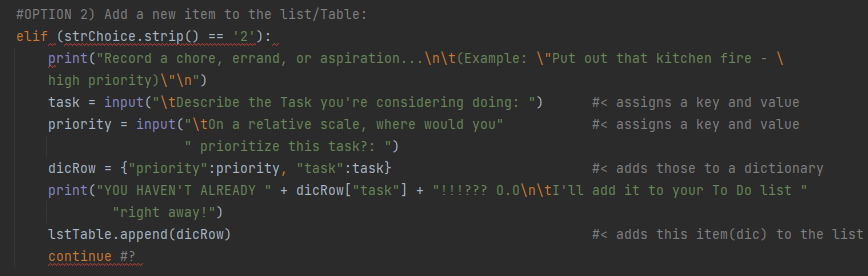
  
(Image 6.0 - While Loop and Menu)

In the first option, I noted some things I would like to try for a cleaner, more succinct code. The idea behind this section was to display the List as it currently exists. To do so I initiated a For loop within the Master Loop and asked it to print each item in the Dictionaries within the List Table. I called these items by their Key, separating them with a dash(-). I also set an index to print in front of them and asked it to integer upward by one with each pass, for aesthetics sake. It was my intention to use this same For loop later in the code, at which time the numbered index would be of more utility. I began this numbered list at one(1). By contrast, the next time it appears in the code it begins at zero(0); I did this as a hint to the user that while the list was similar, the option they were laboring under was different - in case they forgot where in the program they were.



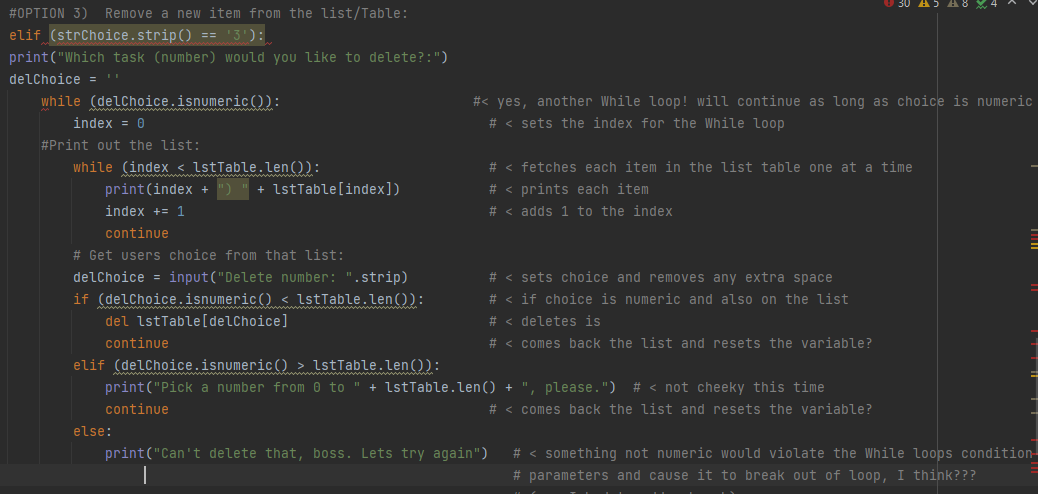
(Image 7.0 - Initial code of Option 1)

To create the second option, I used input from the user to assign two key/value pairs to a Dictionary. I requested the task first as this felt like a more natural flow of thought, even though when printing it back to the user I asked that the priority be set to the left of the task. I then appended this Dictionary to the Master List as well.



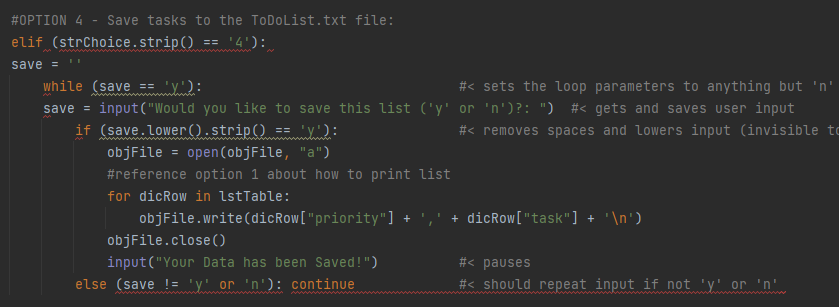
(Image 8.0 - Initial code of Option 2)

To code the third option, I started with the For loop I created earlier to assign a number to each task on the List, and asked the user to choose which item to delete based on its index value. I wanted this list to repeat - including any changes made to it - once at the top of the option and then again after each time the user inputs a number value, so I nested the For loop itself within a While sub-loop. The first If condition of the While loop takes effect if the user chooses a valid number; The row is deleted and the program loops, printing out the newly appended To Do list. If the number is not valid but is still an integer, the second If statement assumes the user mis-typed and brings them back through the list again. Anything not a number, the code interprets as an attempt to break out of the sub-loop and gently returns the user to the larger, ever-present Over Loop.



(Image 9.0 - Initial code of Option 3)

When coding the fourth option, I invoked another While sub-loop to ask the user if they were certain they meant to save. The input is again stripped and lowered. If ‘y’, the code opens the text file and write to that file each item in each Dictionary for each Dictionary in the Master List, including a comma separation between entries and an invisible newline after each row. Then the file closes. I wanted the loop to repeat if the user chose neither ‘y’ nor ‘n’, and to close if the user chose ‘n’, however a glance at this code suggests that it will not do that. I have no explanation for what the developer was thinking; She is a clown.



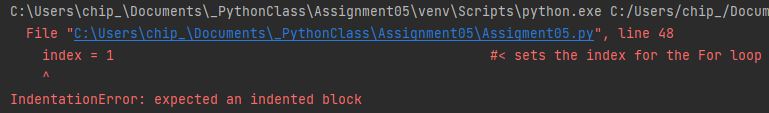
(Image 10.0 - Initial code of Option 4)

With all options accounted for, it was now time to Run The Script.

Run The Script (Errors and Changes)

Cleaning House

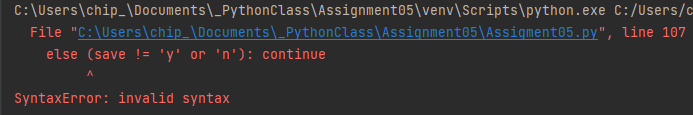
The first bombardment of errors were mostly about errant indentations. This prompted me to change the value of index to zero(0) and scour the code for other constants that I could move to the Declaring The Variables section.

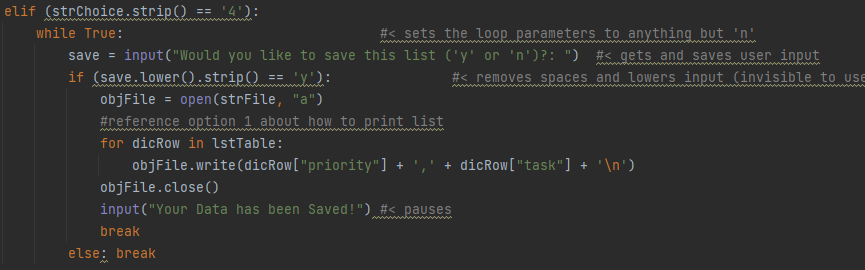
  
(Image 11.0 - Example of indentation errors)

Option Four(4)

Surprisingly, the very first syntax error appeared on line 107. Unsurprisingly, it was to do with option four(4) - saving the text.

I set the While condition to True instead of assessing if the variable ‘save’ is ‘y’. I left the first If condition as ‘y’, stripped and lowered of course. The only other condition has been changed to properly break the sub-loop if the user inputs anything other than a hearty ‘y’, and bring them back to the Menu.

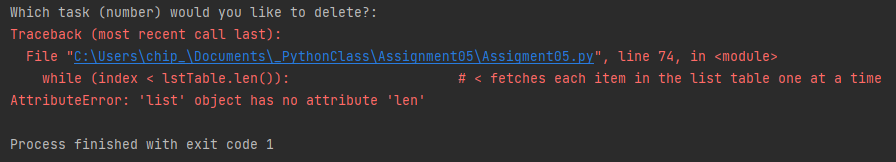


(Image 12.0 - Initial code: syntax error in Option 4)  


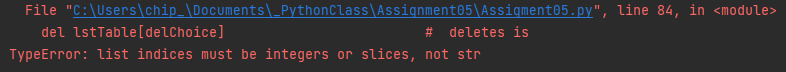
(Image 12.1 - Changes made to Option 4)

Option Three(3)

The next issue I encountered was a logic error. As written, the program would not allow me to delete Dictionaries, but would instead move back to the Menu. Upon inspection and experimentation, I found that the loop was reaching its fail condition immediately because I had set delChoice to a blank string, meaning it met the condition of not being a number immediately. While I was addressing this, my roommate (My Hero from Assignment02) happened by and pointed out that I did not have any plan for if the user input was a negative number, hence the addition of an And operator. When it eventually got around to running the For loop, it did not run the For loop. The syntax I was looking for was len.list, not the reverse. I was also reminded that I would need to put “int” before the variable when that variable refers to the index value of and item in a List.

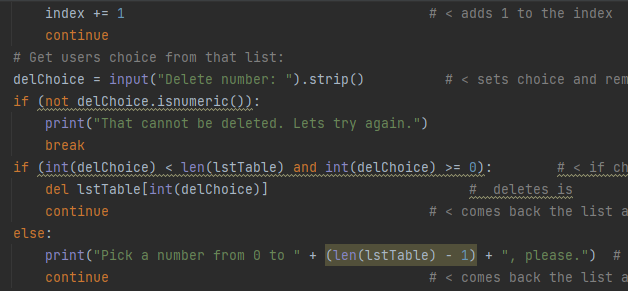


(Image 13.0 - Initial Code: attribute error in Option 3)



(Image 13.1 - Initial Code: type error in Option 3)

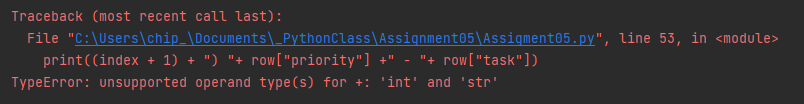
He also explained at length about how sequences in Python begin at the zero position and how this relates to their index and general length - something we had covered thoroughly in a previous module. I was deeply confused until I later read the line print("Pick a number from 0 to " + (len(lstTable) + ", please.", again. My delete command was based on the user picking the proper index from the list, but this line would always return a single interval higher than the highest valid option. With these changes, this section of the program operated satisfactorily.



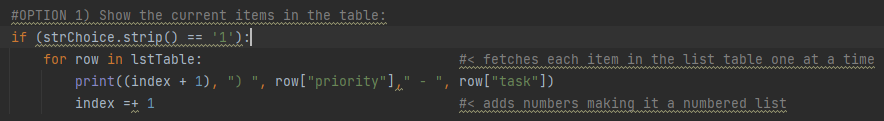
(Image 13.2 - Changes made to Option 3)

Option One(1)

The next error appeared under option one(1), where I left a mess of conjecture and questions rather than functional code. I changed the plus signs(+) to commas(,) as they should have been, and the code worked, but I still wanted to try some things. I was told this was called Refactoring. I removed some things that felt redundant, and when I found that it still ran and I was happy with this brevity, I refrained from experimentation. Sometimes the most obvious answer is truest.



(Image 14.0 - Initial code: TypeError in Option 1)



(Image 14.1 - Changes made to Option 1)

Option Five(5)

This predilection for simplicity behoved me in option five(5), as no changes were required.

In Conclusion

L O O P S . . . @.@

| What is the difference between a *List* and a *Dictionary*?: | Lists contain a sequence of single items that are accessed via their index position, where a Dictionary consists of pairs of items that are accessed via a key. |
| --- | --- |
| What is the difference between an *Index* and a *Key*?: | An index position is unchanging, and will always exist in the same position in a list, while the value it refers to may change. By contrast, a key refers to a specific value in a list and will always call that value even if its position changes. |
| How do you read *data from a file into a List*?: | Use open(file) with file as the path to open said file to return the values within as a string. You would then need to append that string into a list using the list function, append method, and reference said string. Ex: list.append(string). |
| How do you read *data from a file into a Dictionary*?: | Use open(file) with file as the path to open said file to return the values within as a string. Use a loop to access each line of the file. Then, use str.split() with each line as str to split the line into a key and value separated by a space or tab. Use the key and value to create a new key-value pair in a dictionary. |
| What is the programming pattern called *“Separations of Concerns?”*: | Separation of Concerns (SoC) is a design principle for separating and organizing a program into distinct sections such that each section addresses a separate concern. A concern is a set of information that affects the code of a computer program. |
| How would you use a function to organize your code?: | The sort() method sorts the elements in the list in ascending—smallest values first. |
| Why is a *script template* useful?: | A script template can save time and cut back on the possibility of typo by giving the user a premade code ready for alteration. |
| Why is *error handling using Try-Except* recommended?: | Try-Except allows you to preempt the kind of errors that may occur in the course of your program and funnel those into acceptable code set up to handle them. |
| What is GitHub, and why is it used?: | Github is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code. It allows many people to read, make suggestions, and even work on different parts of a code together, and it organizes these changes between developers. |