Chloe Wei

May 13, 2020

IT FDN 100 A

Module 05 Assignment 05

GitHub: https://github.com/cvarw/[IntroToProg-Python](https://github.com/cVarW/IntroToProg-Python)

To Do List

Introduction

As with last week’s Home Inventory assignment, this week I will read from and write to a text file called, ‘ToDoList.txt’. The data will be stored as a list that will be modified via the user interface, UI, then stored back into the text file. Unlike previous assignments, I will not have an example of the UI nor the text file but must use “Task and “Priority” as the dictionary keys. All code must be inserted into the “*TODO*” section of the script and no functions are used at this time.

TextFile.txt

TextFile.txt

Process Data

Figure : To Do List

Formulating a Plan and Solution

After reviewing this weeks lecture, reviewing the web sources for Lists and Dictionaries, and working through the examples in the Python Programming textbook, I knew I had to first write data into a text file for testing out the code. After understanding what the given data variables represented, I worked my way down the code by assigning value to the list variable, lstTable, in figure 2.

A screenshot of a cell phone

Description automatically generated

Figure Read data from text file

To do this, I referred to last weeks ‘HomeInventory.py’ file and changed the variables to the given declared variables in the Data section. Then I looked online for how to separate the data based on the keys and found on the site: <https://docs.python.org/3/tutorial/datastructures.html>, more information on how to manipulate lists and dictionaries. Aesthetically, I prefer short and slim lines of code and the approach to loop through the dictionary worked well for me. So instead of using assigning the dictionary to a variable, I used the letters, ‘t’ and ‘v’ to represent the two keys associated with the index of the list read from ‘ToDoList.txt’ (figure 3 below).

A picture containing black, sitting, table, white

Description automatically generated

Figure revised read data from file

Now that I have the list of dictionaries stored to the variable lstTable, I can now proceed to the Input/Output, I/O, portion of the script.

Using the given menu, pseudo-code, last and this week’s notes on lists, files, and dictionaries, I used the for-loop to work through the lstTable and print out the rows to the UI. I only wrote one task and value at the time of testing, so only one row printed out (figure 4). Since I was able to display the list of dictionaries, I put tweaking the output aside and moved onto how to add a new entry to the list stack.

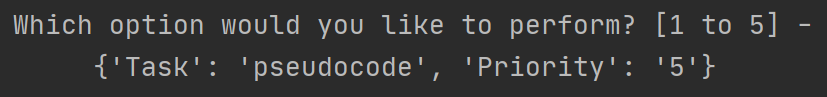


Figure Data from file

I learned from a fellow classmate last week (and in the recent live sessions), that input variables were not necessary for creating lists and decreases extra typing. However, I prefer assigning inputs to variables, so the new task and its priority level were associated with the local variables, ‘strT’ and ‘strP’, respectively. Using lists built-in append-function for lists, the new task and associated priority ranking were then added to the end of the lstTable stack as a new dictionary. To be sure it is stored how I wanted it, I used the for-loop to display the updated list and commented out later. For now, I only allowed the user to input one task per instance of the add new task from the menu option.

A close up of a street

Description automatically generated

Figure Add new data

The next item to code for the ‘ToDoList’ script is to enable to the user to remove the new entry. At first, I considered a quite lone, convoluted way of finding the end of the stack and removing the new dictionary. But after watching this week’s live session, I learned there was a more effective way of removing dictionaries from the lstTable found on figure 7.

A close up of a sign

Description automatically generated

Figure 6 Initial code to remove item

The lstTable acts as a collection of dictionaries, so using the Randal’s solution to create a search through each dictionary and remove the match to the input. This allows not just the new task & priority level to be removed, but previous task stored in the list as well! As always, I included feedback for the UI when the task is found or not. If it is not found, then the UI will display the menu. I held back from showing the list and prompting for another chance at removing a task for now. If time permits, I will go back to this section and adjust the loop.

A screenshot of a cell phone

Description automatically generated

Figure 7 Improved removal of task

Now that the ‘ToDoList’ is able to open and store the data from the file to memory, display the stored data and new data entered, add and remove data, I can now address how to store the data back to the associated file. Writing back to the file is somewhat universal for data collections, so I used the same methods and loops for the Home Inventory script and our notes. However, instead of referencing the indices of the list, I used the keys of the dictionary to determine how the data will be stored. I also attached the newline character to the “priority” key instead of a separate string element. This make it easier to strip the character when importing the data back into memory through the “ToDoList” script and eliminated an unnecessary element for each dictionary stored.

A picture containing table, black, screen, holding

Description automatically generated

Figure 8 Save data to file

Summary

Through this week’s assignment, I was able to utilize PyCharm’s debugging tool and perform tests through the debugging console. Using the debugger along with following the steps from the starter code, helped make each selection of the menu manageable. There were times, that I would need to create new python files in PyCharm to work through how to add and remove dictionaries, as well as how to add the list of dictionaries back to the file. I found writing the data back to a file the most difficult aspect of lists and dictionaries. I do feel I had some growth understanding the relationship of dictionaries to lists as well as the flexibility of lists for data manipulations as well. After working my way towards the write to file, I went back to address the appearance of the display of list data and other potential modifications to the code. I was able to adjust the aesthetic of the list printed to the UI but at this time, unable to streamline a way for the user to retry either adding or deleting multiples tasks without huge blocks of code unnecessary for this assignment, so I will wait to modify this later in the week.