Kristin Temperly

02/15/2022

Intro to Programming (Python)

Assignment05

LISTS AND DICTIONARIES

**Introduction**

**In this module, we continued to work with lists but Randal introduced a new sequence called dictionary. We also learned about a separation of concerns, and structured error handling. We ended the module learning about GitHub and we were asked to create our own GitHub account.**

**Dictionaries**

**Dictionaries are like index subsripts except instead of using an index they use a key. They use curly braces instead of square brackets. They are represented by a key, then a colon, and then the value. Both the key and the value are in quotes. A dictionary can be thought of as a row of data in a database.**

**The Item funciton allows us to unpack the items and custom print the key and the value. Another way to prepare the data for presentation is to use the values and the keys functions. These are handy funtions built into dictionaries.**

**The strip function strips out spaces and extra carriage returns.**

**Separation of Concerns**

**SoC is a programming design technique used to build multi-tiered or multi-functional programs. Specifically, SoC refers to the method we use to organize our program. Typically, we separate sections by 1) data 2) processing and 3) output/ presentation. An example of the data section occurs when the programmer declares one or more variables. An example of processing occurs when we perform a calculation or a manipulation of the data. An example of presentation is to print the data. Sometimes it’s not possible to get each piece of code in the right SoC.**

**Functions allow us to better control our code. For example, in some samle code Randal declards a “DivideValues” function. He declared this function first, and then calls it later. Programmers can have many many functions in any given program, which ultimately helps organize the SoC by separating processing from presentation. It also makes the code much more re-usable.**

**Another way to organize code is to create a script template in PyCharm. Most IDE’s support custom templates.**

**Randal recommends we declare our variables even though Python doesn’t demand it.**

**Structured Error Handling**

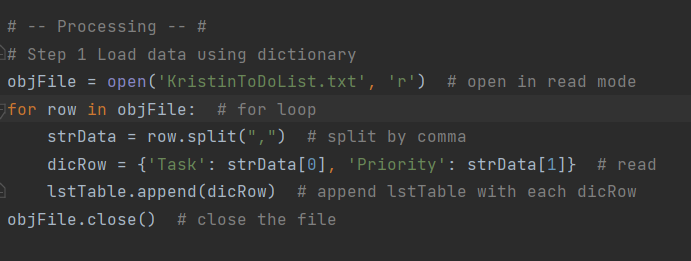
**Instead of relying on the Python programming engine to produce errors, custom error handling is a more user-friendly way to present errors to the end user. Technical details produced by Python are not necessarily valuable to the end user, so it’s best to customize simpler error messages for the user. For example, a “not divisible by 0” error is not friendly to the user, so we customize this error to simplify it by using the Try-Except block.**

**GitHub**

**GitHub is a great tool for serveral reasons. It’s an effective way of sharing code with co-workers. It’s also a way to implement source control for configuration control purposes. Furthermore, it’s a way to backup code in case the programmer’s computer or server crash. Lastly, it’s a free service. GitHub.com is a tool or website we will be using in this course.**

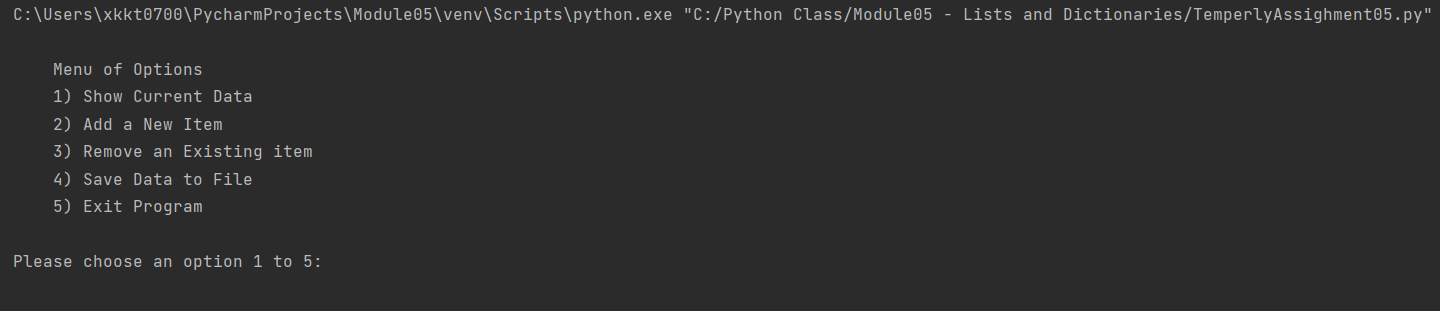
**Putting It All Togheter**

**This assignment was easier than last week because Randal supplied sample code for us to re-use. I like that he included the actual code samples in this module – very helpful! The program will begin by loading each row of data from a text file called KristinToDo.txt into a dictionary. Each row contains both a key and a value. The for loop then searches each row in the file and performs a split with a comma. Then we create the dictionary row by passing 0 and 1 position indexes. Lastly, we close the file. One way to enhance this section would be to create a custom error message for the user in case the text file does not exist yet.**

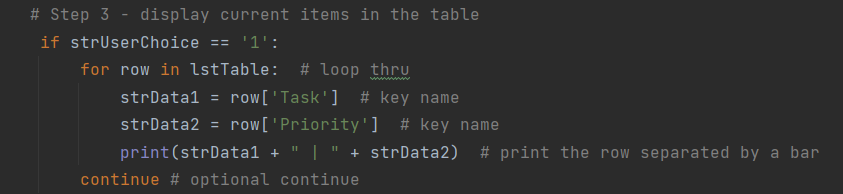


**Figure 1. Data Loading**

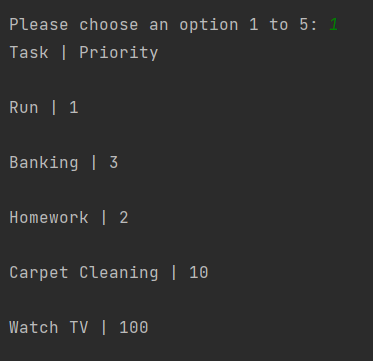
**After data has been loaded into memory, the next step is to display the main menu to the user. The menu will allow the user to 1) show current data 2) add data 3) remove data 4) save data or 5) exit.**



**Figure 2. Display Main Menu**

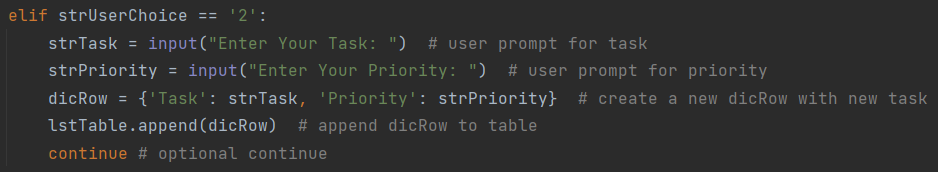


**Figure 3. Show Current Data Code**

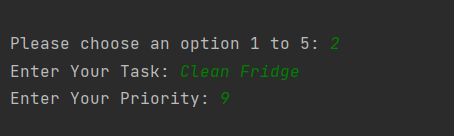


**Figure 4. Show Current Data Result**

**To add a new task, the list table is appended with the dictionary row.**

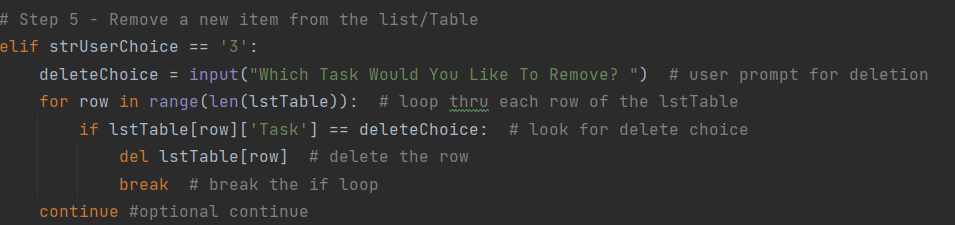


**Figure 5. Add New Data Code**

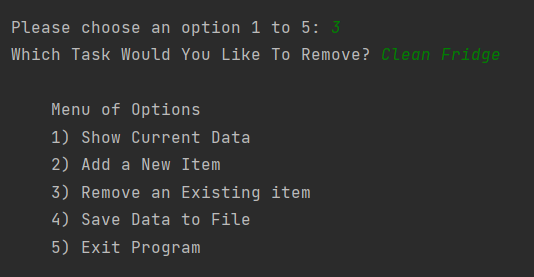


**Figure 6. Add New Data Result**

**To delete a task, the user is prompted and must know, from his or her own memory, which task to delete. A way to improve this program would be to print the data again for the user in case they don’t have it memorized. After all, the user is probably writing down the to-do list because they can not memorize it! The mechanism to delete the row is the delete function, and once that happens the if condition will break.**

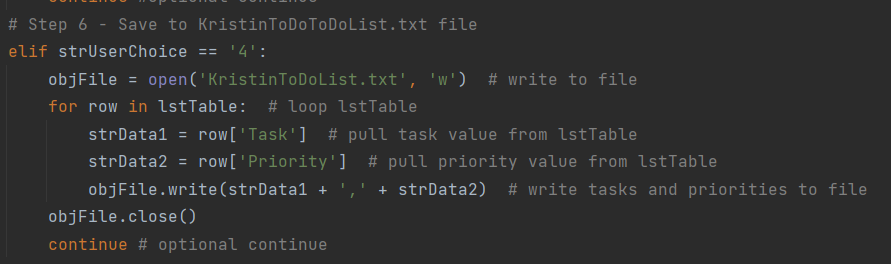


**Figure 7. Remove Data Code**



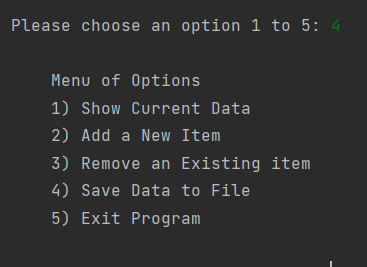
**Figure 8. Remove Data Result**

**The last step of this program writes the newly modified list of tasks and priorities back to the KristinToDo.txt file. A connection to the file is made again, but this time in write mode, and the variables are looped from the table and written to the file.**

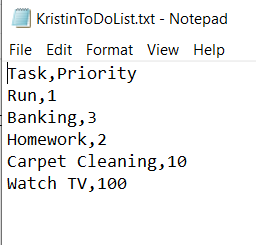


**Figure 9. Save Data to File Code**

**Figure 10. Save Data Result**

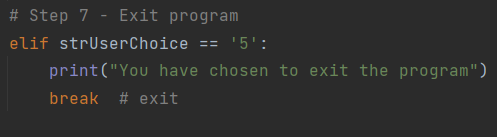


**Figure 11. Save Data Result**



**Figure 12. Kristin ToDo.txt file**

**The very last step of the program was to exit.**



**Figure 13. Exit Program**

**Summary**

**In module 05, Randal reviewed the sequence called lists and introduction a new one called the dictionary. The primary difference between them is that the dictionary sequence is created with a key and a value. A key is ultimately a column header, and the value is the data within any given row. We also learned about separation of concerns and error handling. Lastly, we created a GitHub acoount.**