Gao Yujia

10 May 2021

Foundations of Programming: Python

Assignment05

**Creating a To Do List**

**Introduction:**

**This assignment builds upon the previous learning modules on how to create as well as append data within a table. The additional component is the use of dictionaries.**

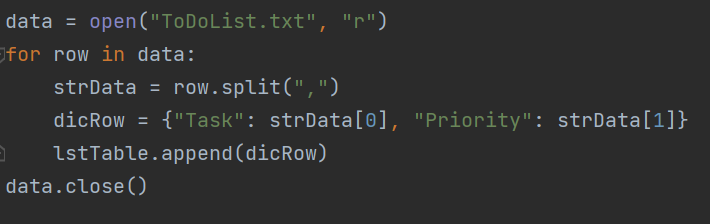
**The main difference between a list/tuple and a dictionary is that we are able to define the keys in a dictionary, instead of just using indexes. Indexes refer to the referencing of contents of a list / tuple based on its position in that row. However with keys in dictionaries, we are able to allocate a specific term to a variable in the row of content, and we will be able to call up these variables by calling up the key assigned to that content variable. It is much easier to use compared to lists / tuples and much more intuitive, negating to need for us to count the position of content variables.**

**Creating the Python File**

**There are multiple steps required in the creation of this ‘ToDoList’ file.**

**Step 1:**

**In step 1, we are required to load any existing data available in the txt file “ToDoList.txt”. This is to enable us to call up any information that may already be stored in the file (Fig. 1)**



**Fig. 1**

**One problem that I had was that this file did not exist, hence I had to create this file first using a ‘write’ sentence, before changing it to a ‘read’ sentence.**

**Step 2:**

**Step 2 involves printing a reference menu to create a dropdown list of options which the user can choose from. This is a relatively straight forward task and is already given in the template.**

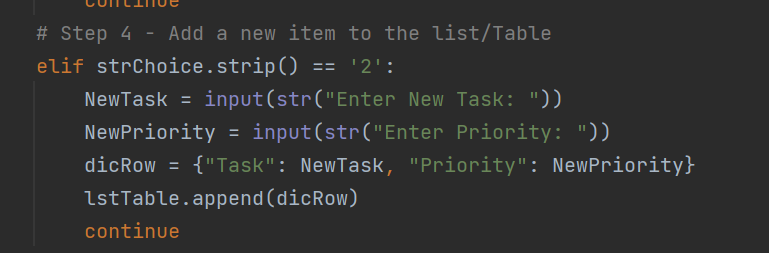
**Step 3:**

**In this step, we need to create the lines of codes that will enable the user to perform the tasks in the dropdown menu.**

**For the first task of showing all current data, as the list is empty to begin with, keying in ‘1’ will not yield any output if we are running this scrip for the first time.**

**Step 4:**

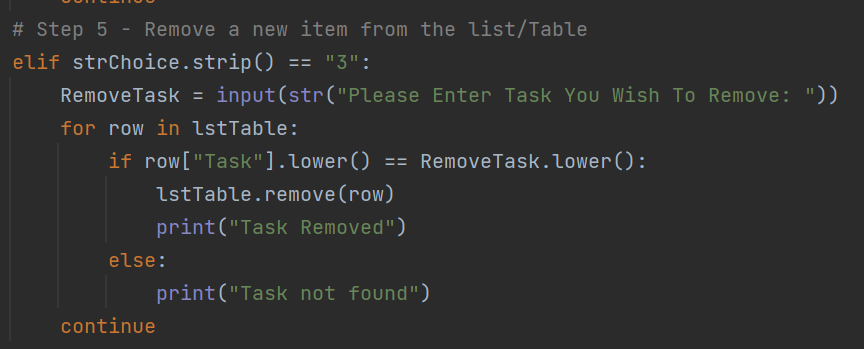
**For task 2 where we are required to add new items to the table, user input is required for both the task type, as well as the priority for the task. These will be converted from a string input into a dictionary rather than a list / tuple, and the keys ‘Task’ and ‘Priority’ will be assigned to the variables. Each entry of new data will be appended to the table for continuous data input (Fig. 2).**



**Fig. 2**

**Step 5:**

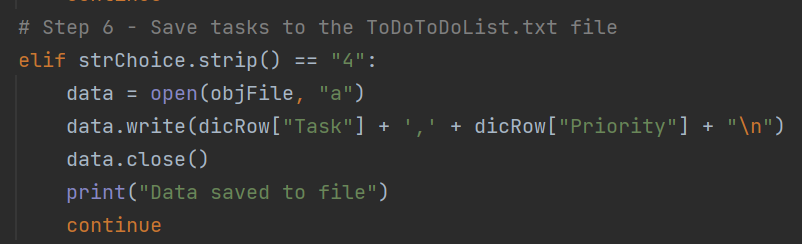
**In order to remove data from the table, we will need to enter a line of code that will prompt the user on which row of data they wish to remove. To achieve this, the user input much match the ‘Task’ description. To mitigate input errors due to incorrect use of upper or lower case, the .lower code is used so that the input and task description will both be converted to lower case for an easy match. Once the user input and the task description matches, that row of data will be removed (Fig. 3)**



**Fig. 3**

**Step 6:**

**In order to not lose our data, the entries will need to be saved onto the harddrive. This line of codes will first allow us to open the desired file destination, and the rows of the dictionary will be written into this file and saved (Fig. 4)**



**Fig. 4**

**Step 7:**

**And finally, when the user has completed all the tasks, they can exit the program.**

**Conclusion:**

**The ability to create table with lists / tuples / dictionaries gives us the flexibility to arrange the data in a way that makes sense for us. However, we must be sure to document our intentions in the Python file so that other users who use our file in the future will be able to follow our thought process and make sense of our codes.**