Lantian Qin

November 17, 2020

IT FDN 110 A

Assignment 05

Lists, Dictionaries and Best Practices

Introduction

In this assignment, I created a data management program that loads data from a txt file with a to-do list and displays its content. Then, it allows the user to perform a series of actions on the data, including adding a new, deleting an item and saving the changes to file. This assignment requires us to understand how lists and dictionaries work, and since it’s relatively complicated compared to the previous assignments, it’s important to organize the code with script template and pseudo codes.

Data

In the starter Python file for this assignment, professor Root had written out the variable definitions in the “Data” section of the file. It’s good practice to declare these values early on in the code since they will be re-used and referred to frequently throughout the program.

Text

Description automatically generated

*Figure 1. Variables and Constants Declarations*

Processing

In this section, the program opens the to-do list file and reads its content into a list of dictionaries in memory, which we will be used for data querying and manipulation later on. The open() function reads the txt file line by line as a string, so I split the string by comma to separate it into two items in a list, and then store both items into a dictionary. Then, I append the dictionary to the pre-defined list of dictionaries.

Graphical user interface, text, application, email

Description automatically generated

*Figure 2. Read and Process to-do list*

Input / Output

This is the part of the code that allows the program to interact with users, take their input and process them accordingly. The entire section is in a while (True) loop, which means it will keep running again and again until a “break” condition occurs, and every time the it goes back to the beginning of the loop, the program will display the menu of options and have the user enter their choice. With that in mind, I started to work each individual steps of the loop.

Graphical user interface, text

Description automatically generated

*Figure 3. Starter code for input-output section*

Show Current Items in the Table

Using what I learned from the last lecture, I wrote a for loop that goes through every item in the list of dictionaries and print out the task and the priority in a formatted way. For formatting, I used the f-string, which I find very useful in helping coder visualize what the output will look like.

A picture containing logo

Description automatically generated

*Figure 4. Show Current Items in the Table*

Add New Item to List

The next step of the program allows the user to enter a new row of data. This snippet of code prompts the user to enter a task and a priority, and stores that information into a dictionary which is appended into the list of dictionaries. I declared the dictionary inside the append() method so that I don’t have to think of a name for this temporary value.

Graphical user interface, text, application

Description automatically generated

*Figure 5. Add New Item to List*

Remove an Item from the List

I want to give user the option to choose which option they want to remove from the list, so I made the program print out all the items currently in the list and asks the user to select which one to delete by typing in the item’s number. I also want to allow users to cancel the operation if they change their mind, so I made sure if they type in “x” the removal will be aborted. In order to make this possible, I searched for a way to delete an item from a list by referring to its index and found the del function, which is perfect for this use case. To let the program handle error gracefully when user enters an invalid option, I wrapped the input part in a while True loop and used the try-except construct so that when the entry is not an “x” and can’t be used to index the list, the program will print “Invalid entry” to the screen, go back to the beginning and prompt the user to select and item to delete again.

*Graphical user interface, text, application

Description automatically generated*

*Figure 6. Remove an Item from the list*

Save and Exit

When the user chooses to save data to file, the program will open the to-do txt file for writing. I used the “w” keyword because the list of dictionaries contains everything we want in the file, and this keyword will make sure than when it opens the file it will truncate and write the list into the file instead of appending it to the end of the file. I used string concatenation to convert each dictionary into a string that’s separated by a comma and ends with a new line character. This will ensure the next entry starts on a new line.

When the user chooses the exit the program, the break statement will execute which ends the loop and the program. I want to make sure when the user enters am option that isn’t 1 to 5, the program can tell the user what’s wrong and ask them to enter again, so in the end I added a else statement that prints out “Invalid entry”. And since we are not breaking from the while loop, the program will go back to the beginning of the loop and asks the user for entry again.

Graphical user interface, text, application, email

Description automatically generated

*Figure 7. Save and exit*

Testing the Program

I tested the program on both PyCharm, and as documented in the screenshots below, the program works as expected.

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text

Description automatically generatedText

Description automatically generated

*Figure 8. Running on PyCharm*

Graphical user interface, text, application

Description automatically generated

*Figure 9. PyCharm run result*

*Text

Description automatically generated*

*Text

Description automatically generated*

*Figure 10. Running on Terminal*

Graphical user interface, text, application

Description automatically generated

*Figure 11. Terminal run result*

Summary

From this assignment I learned that lists and dictionaries are useful structures for handling data in Python, and a combination of dictionaries in a list is especially helpful in representing tables with named columns. I also learned that for complex programs, it’s important to follow the “separation of concerns” principle since leaving code unorganized can make the development and maintenance process difficult. Using code templates and pseudo codes is a good way to make sure your code is structured.