Kyle Banfill

2/22/2022

Foundations of Programming: Python

Assignment 05

Creating a ToDo List

Introduction

In this module, we learned how to pack and unpack Lists and Dictionaries from saved files, and some finer points of code hygiene with topics such as Separation of Concerns, Functions, and Try-Except.

Questions

1. What is the difference between a List and a Dictionary?

Dictionaries replace index subscripts with keys.

1. What is the between an Index and a Key?

An index is a numerical description of a position within a sequence while a key is a character subscript. Keys allow easy organization of types of data.

1. How do you read data from a file into a List?

To read data from a file, you open and read the data using index subscripts to sort the data into a list.

1. How do you read data from a file into a Dictionary?

You open and read the data but use keys to assign the values into a Dictionary.

1. What is the programming pattern called “Separations of Concerns?”

Separations of Concerns is organizing code into different distinct sections such as data, presentation, processing, etc.

1. How would you use a function to organize your code?

You could use functions to cleanly organize code. You could focus on presentation in one area and leave all the processing in a different area, using defining functions within the processing area, and just calling those functions in the presentation portions.

1. Why is a script template useful?

It allows for uniformity between different coders and projects.

1. Why is error handling using Try-Except recommended?

The default errors that appear may be inscrutable to people untrained in coding, so using Try-Except to present a simpler explanation of the error is recommended.

1. What is GitHub, and why is it used?

GitHub is a way for people to store and share code online. Not only can you share code so that others can work on it or view it, but you can also back up previous versions of your code so that you can rollback if you need to.

Our Basic ToDo List Script

Text

Description automatically generated

Text

Description automatically generated

One thing to note immediately is that because we’re dealing with a pre-existing text file that needs to be exist next to our script, we need to be sure to point our OS command at the right directory. If we don’t specify the directory we’re looking at, OS command will not be able to find the text file that we use in our code.

Text

Description automatically generated

The first section is to define our variables and constants. These already existed from the starter code provided. We create a temporarily empty dictionary and list as well.

Text

Description automatically generated

In step 1, we open the text file, read it with “r”, and then take the information in there and unpack the data with our row.split() and move it into a dictionary by supplying our keys “Task” and “Priority”. We then take our dictionary we’ve created and pack it in a list by appending it to our lstTable. Afterwards, we do our due diligence and close the file as we no longer need it open.

Text

Description automatically generated

In step 2, we have a menu to guide user input and output. This was provided by the starter code.

Graphical user interface, text

Description automatically generated

In step 3, to display the current data, we take the list we created earlier, which consists of rows of dictionary data, and display it row by row with a print().

Text

Description automatically generated

In step 4, we prompt the user for input. We can use .strip() when we pack the data into a dictionary to ensure that any unsightly spaces or line breaks are omitted. We also update our lstTable with the new information. We finish with a printed statement to give the user some feedback concerning the work.

Text

Description automatically generated

Step 5 was by far the trickiest. I ran into an issue while testing the script that I’d forget what exactly was written for the tasks while trying to input the task information. If I’m going to experience it, I imagine a user will too, so I just copied the block of code from step 3, but with a preceding line of printed text.

Next, we take the input from the user, lower case it and strip it to normalize the text, and then compare it to values with the “Task” key in our dictionary. If it matches, we remove the entire “row”, which catches the task and its associated priority. We also print a line of task for user feedback that a change has or has not occurred.

An issue that I was running into was that when trying to give feedback to the user, I’d have the “unable to find” line print multiple times if the task was buried, so to speak, deeper within the list. To fix this, I created another variable, RemoveConfirm, as a flag that would trigger if the task was found within the list, which would then cause the removal confirmation text to print. If this flag stayed False, then after the for loop finished and moved on, it would return the “unable to find” text.

Text

Description automatically generated

For step 6, we reopen the file since we’re going to write data to it, and then be sure to pack our list back in with their nice and tidy keys, “Task” and “Priority”, along with a line break for some visual clarity. And let’s not forget providing the user with feedback after this has finished. And as always, we close our file since we’re done with it for now.

Text

Description automatically generated

In step 7, we create one line of input to make sure the user wants to exit so they don’t immediately end the program if they mistype the number 5.



And finally, we put one catchall else statement at the end for our main while loop for user feedback if they input a value other than 1 – 5.

Conclusion

We’ve learned to pack and unpack lists and dictionaries and to write and read them files for later use. We also learned how to clean up our code so that it’s more readable to us and to others who may handle the code, as well as valuable information regarding how to share our code with others through services such as GitHub.