Jacob Bentzen

May 23st, 2022

Foundations Of Programming: Python

Assignment6

Adding to a Code Template : Functions

**Introduction**

In assignment 6, I was tasked with using a code template to re-frame assignment 5 using functions. Since there was already quite a bit of code already in place, we only needed to work within the functions to make sure they were coded properly.

**Understanding the Template**

Finding what is already completed:

First, I will need to run through the code template and find out what I need to complete. As fig. 1 shows, the entire header and data sections have been completed.

Text

Description automatically generated

*Fig. 1: Header and data section of the template.*

It looks like there are plenty of variables already declared that are ready to be used in the rest of the code. Speaking of the rest of the code, fig. 2 shows most of what is expected to be completed.

Text

Description automatically generated

*Fig. 2: Shows the IO class, which includes several Functions left to be completed.*

Basically, the template is requesting me to complete the incomplete functions so that the program will run completely when the functions are called. There is also a processor class with similar functions to be complete.

Functions in the Processor Class:

As you can see in Fig. 3, the functions being used are basically the same code that I wrote in assignment 5.

The add data function uses .append, the remove data function uses .remove, and the write data to file function utilizes a for loop to cycle through the dictionaries in the list and adds them to the file.

Text

Description automatically generated

*Fig. 3: Completed functions under the processor class.*

The main difference for this assignment is making sure the functions are receiving and sending the parameters and arguments correctly. For example, in the remove\_data\_from\_list function I had to make sure that I was returning the list\_of\_rows in multiple places otherwise I would receive an error of an “none” list.

Functions in the IO Class:

The IO class includes functions that are input and output related (Fig. 4).

Again, much of the thinking work was completed last week with assignment 5. However, I made sure to adopt the code so that it made sense in a function context and return the correct values.

Text

Description automatically generated

*Fig. 4: Completed functions under the IO class.*

For the input\_new\_task\_and\_priority function, I made sure that the proper values were returned as a tuple as the code was written to receive the data as such.

Checking the Main Body:

Of course, all these functions are used to run the actual script, so they will need to be called in the main body (Fig. 5).

Text

Description automatically generated

*Fig. 5: Main body of the script with all functions being utilized.*

The layout of the body is similar to assignment five, with everything being contained in a while loop and the users choice deciding the programs action.

Again, the difference is the use of functions. Since this portion was already completed, I was able to use the calls of the functions to make sure that my code was correct. For example, step 4 requests both task and priority from the function, so it is necessary to return both pieces of information in the code.

**Testing the Script**

Using PyCharm:

Since the program was written in PyCharm, it is straightforward to run and test the completed program (Fig. 6).

Text

Description automatically generated

*Fig. 6: Testing the program in the PyCharm run window.*

Using the Command Window:

After running the script in PyCharm, I made sure to test it using the Command Window on my computer (Fig. 7).

Text

Description automatically generated

*Fig. 7: Command Window with the Home Inventory Program completed.*

**Summary**

I was able to complete this program by utilizing my understanding of functions, as well as both arguments and parameters and how they move to and from functions within Python.

Additionally, here is my GitHub link:

<https://github.com/bentzj2/IntroToProg-Python-Mod06>