Kip Keeney

Randal Root

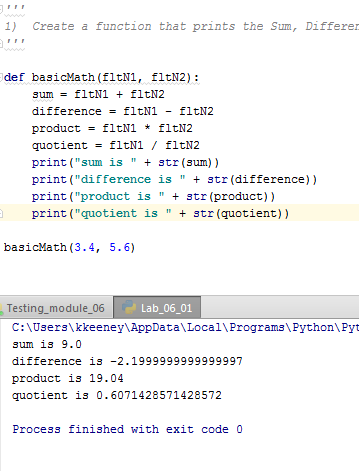
IT FDN 100 A (Python)

7 November 2016

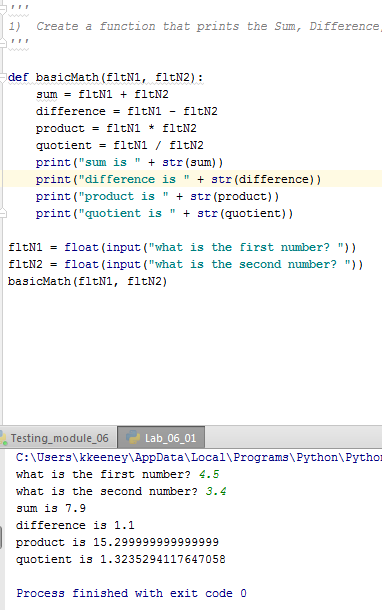
Module 06

In module 06 we covered Abstraction (encapsulation), Functions, and Classes and Methods. Overall I thought this lesson was far more straightforward and easy to follow than the previous session. The labs seemed to apply directly to the end assignment and that made completing the assignment much easier.

For the first lab, I found it relatively straightforward and quickly came up with this code:

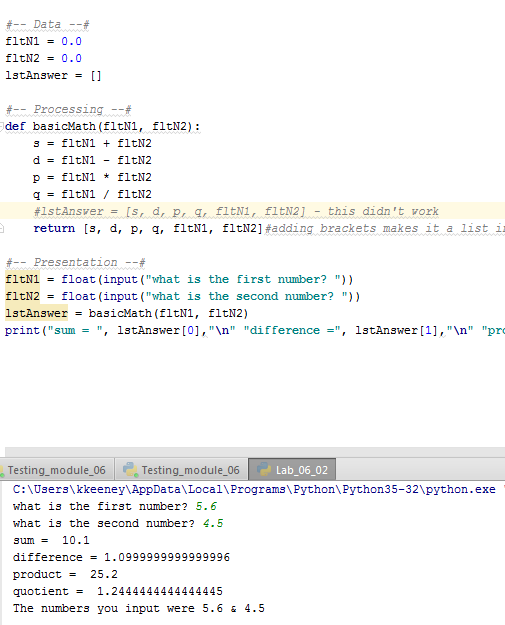


This was fine and all, but I didn’t like that I had to edit the code to change the numbers that were being calculated. I attempted to add a little more complexity to the code and I ended up with this:

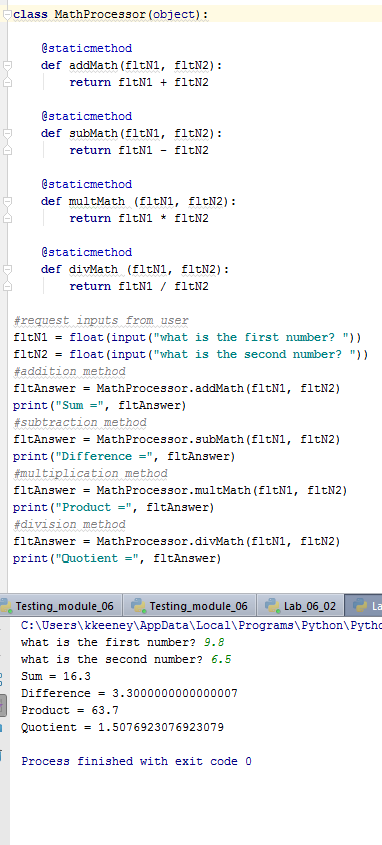


Now my code requests two numbers from the user and will allow me to test other numbers without hard coding them. Granted I could have gone much further and implemented some error handling, but I was happy with the results and moved on.

In the second lab I found that I could reuse all of my code from the previous lab. It was pretty simple to divide the code up into data, processing and presentation. Here is the end result:



I declared all of the variables and gave them values in the data section. I initially I tried to add a list that was populated with the sum, difference, etc, but I wasn’t able to make it work the way I wanted to and forgot to remove it from the final solution (thus the commented out line in the processing section). However I did use that same list in another way. I populated it with all of the data from the function and then called the different list items by their index to display the answers to the end user.

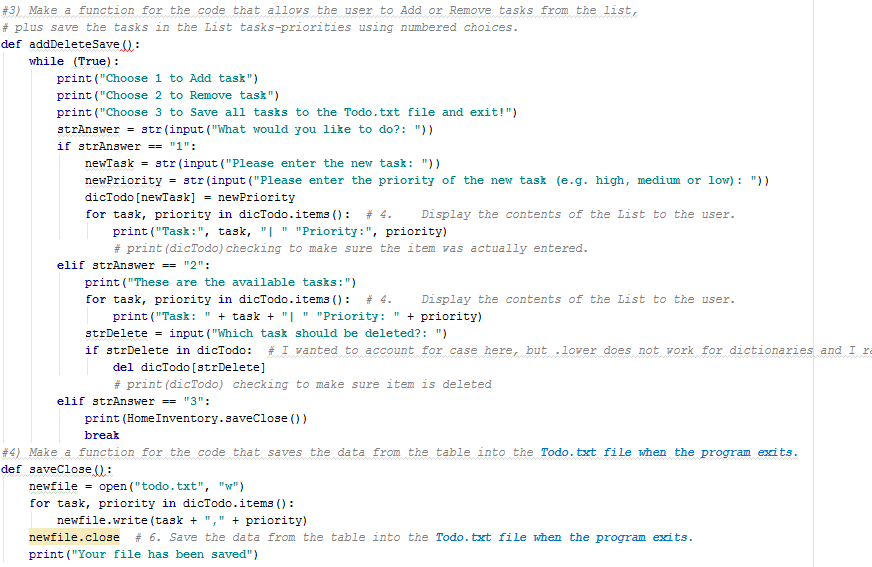
The third lab was yet again another iteration of the same base code. I didn’t follow the instructions to the letter and you will note that my method names deviate from what was requested in step 3. I still don’t understand what @staticmethod is for, but I included it per the instructions. I did find that the code appeared to work without it. Not sure if that’s a problem or not. Here is my version of Lab 06-03:

I find the concept of Classes, Methods and Functions to be very clear and I think I implemented them well in this code. Moving on to the assignment itself, I found this to be one of the easier assignments we’ve had, especially after all the struggles I had with the previous assignment. The first couple of items were easy-peasy and here is the resulting code:



I put a document header on this code to indicate, who, what when. I then divided the code up into sections. My data section is pretty small and only includes the dictionary where I store the inventory. Then you will see the processing section where I have the class called out. Below the class are two functions in the image above. The first one is loadData where I loop through the data in the txt file and then load it into the dictionary. I originally had the display of the task and priority in this part of the code, but I broke it out to make my second function displayData. This loops through the dictionary to display all of the tasks and priorities in the dictionary.

Next I moved on to the multiple options available to the end users. Here is what I came up with:

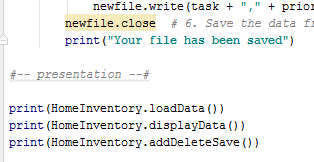


I kept my while loop and essentially just nested it under a new method called addDeleteSave. This name isn’t quite correct since save is technically in another method. I only ran into one thing that I was unsure of. I had my save functionality coded as part of option 3, so when I tried to follow the instructions to create a separate function for saving the data, I had to break up the original code from module 5. I ended up using a reference in option 3 to the last method, the save method. Specifically I’m talking about this:

**elif** strAnswer == **"3"**:  
 print(HomeInventory.saveClose())  
 **break**

I’m not sure if that is okay or not, but it works. My last method is outlined above and is called saveClose. It essentially just writes the elements that have been added (or deleted) to the .txt and closes the txt.

I wanted to be able to see all of these methods in action, so I added something to the presentation section. It was simple, but it allowed me to run through the code. Here’s what that looks like:



Those simple print statements allow me to call each of the methods and use the program. Again, I’m unsure if this is the proper procedure for calling these methods, but it works!