**Name:** Kerry Bosworth  
**Date:** Nov 27, 2019  
**Assignment:** 08  
<https://yukkutobu777.github.io/IntroToProg-Python-Mod08/>

# Class and Objects

**Introduction**

The assignment this week is similar to assignment 6, but with an object class. Also, after assignment 7, error handling should be considered wherever possible. Before bringing in code from those assignments, I took time to test out new features as well as build the base for the classes.

**New concept - DocString**

For example, before changing any code in the main body I typed:

*# Main Body of Script ---------------------------------------------------- #***print**(FileProcessor.\_\_doc\_\_)

This accessed the DocString function that was provided for the above referenced class. It printed the following to the screen.

Processes data to and from a file and a list of product objects:

methods:

save\_data\_to\_file(file\_name, list\_of\_product\_objects):

read\_data\_from\_file(file\_name): -> (a list of product objects)

changelog: (When,Who,What)

RRoot,1.1.2030,Created Class

<Your Name>,<Today's Date>,Modified code to complete assignment 8

This is optional but is especially helpful to a user running the program who does not have access to the source code. The IO class had no DocString defined so when running the above and substituting with ‘IO’ it returned with ‘None’. Before moving move on, I added it. From there I built out the object class for product and made sure that worked with simple calls from the main body before moving forward.

**Programming Assignment**

In this assignment we added an object class. A normal class can be called directly. In the class definition you start it with @staticmethod. We did this when calling functions used in previous assignments like:

IO.OutputMenuItems() *# Shows menu*

An object class called Product can be called indirectly and have multiple objects that are stored in different places. For example:

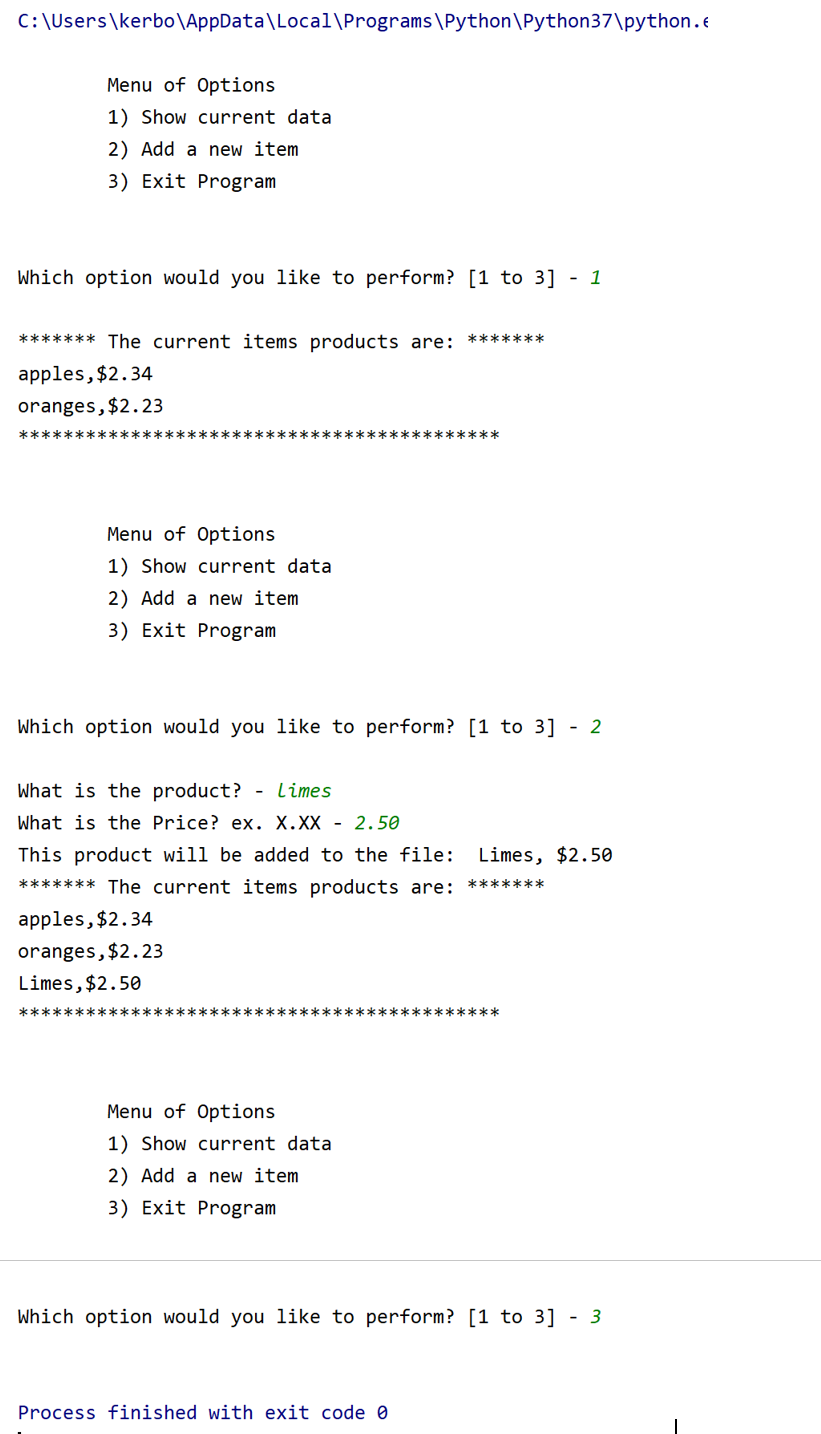
*# Let user add data to the list of product objects***elif**(strChoice.strip() == **'2'**):  
 product, price = IO.EnterNewEntry()  
 objP1 = Product(product,price)  
 **print**(**"This product will be added to the file: "**, objP1)  
 FileProcessor.AddEntry(objP1.product\_name, str(**"$"** + objP1.product\_price), lstOfProductObjects)  
 **continue** *# to show the menu*

This type of class is made up of the following:

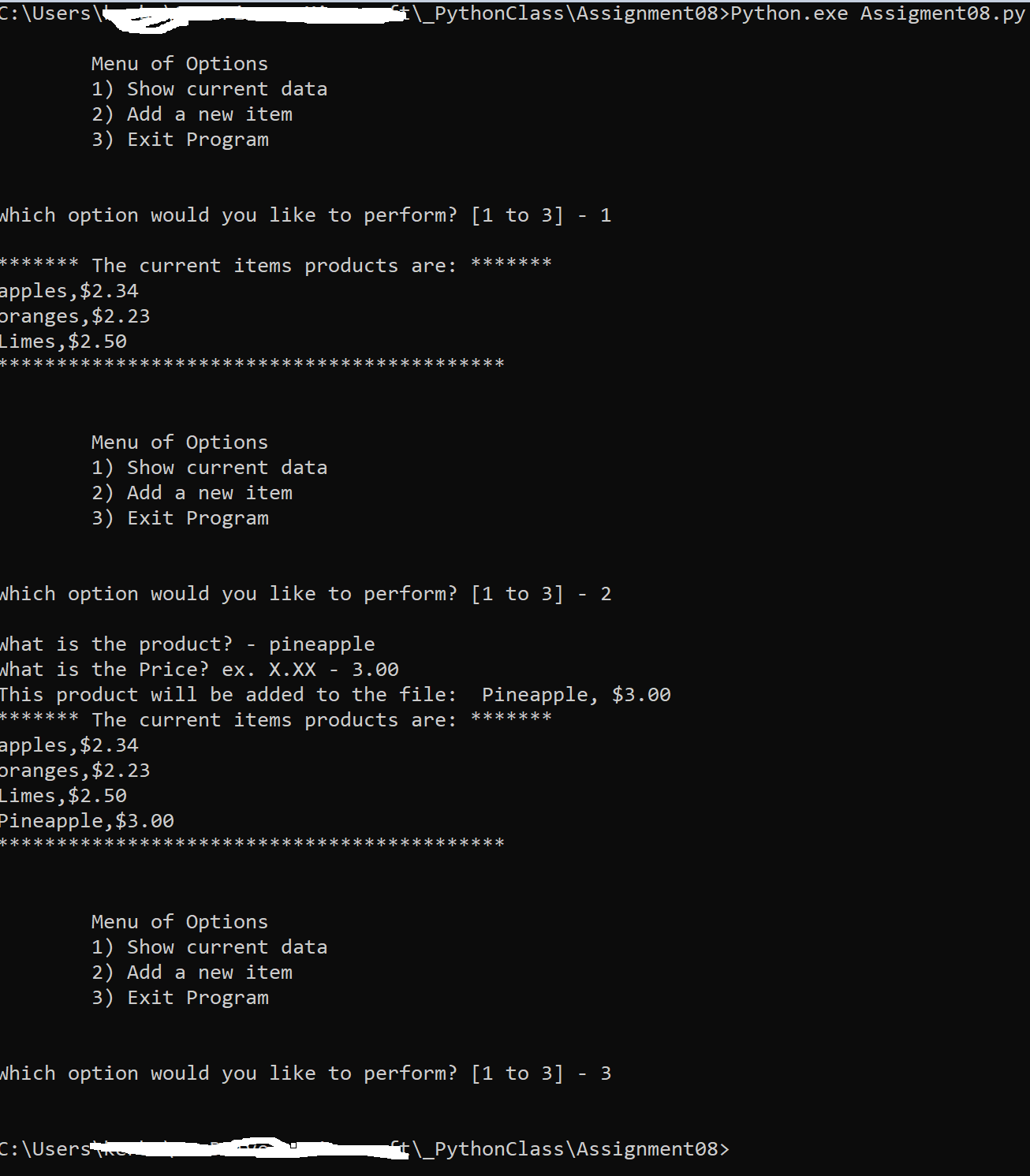
class MyClassName(MyBaseClassName):   
 # -- Fields --   
 # -- Constructor --  
 # -- Attributes --  
 # -- Properties --   
 # -- Methods --

A class constructor defines the ‘self’. This allows you to load the class once but have multiple instances of a class (a copy of self). Properties use/manage attributes. Methods do the processing.

After filling out the rest of the classes and building out the main body of the assignment here is the output of the code (see Figure 1 and 2).



**Figure 1:** Run of program in PyCharm



**Figure 2:** Run of program in command line

**Summary**

The objective of this assignment was to create and use the object class. I ran out of time to add more error handling. For example, checking to ensure the price entered was a number. We also learned how to use GitHub desktop and Git. These allow you to work on code locally and either graphically, (GitHub Desktop) or via the command line (Git) and then add and commit the code to a shared repository on the Internet.