Deborah Yenubari

03/10/2021

Fundamentals of Programming (Python)

# Assignment 8

**­­­­­­­­­­­­­­­­Classes**

Classes are a way of grouping data and functions, and most classes are designed to focus on either data or processing.

In this assignment, a class “FileProcessor” was created to processing data to and from a file. Another class for assigning properties and methods to a product called “Product”. The focus of the "FileProcessor" class would be to perform a set of actions, while the focus of the "Product" class would be to organize data about a Product.

Data in a class is defined using variables or constants. However, when these are in a class, they are called Fields. Any functions you have in a class are called Methods.

**Objects vs. Classes**

When the class's code loads into memory, you either use that code directly or indirectly.

To use the class's code directly, you use commands like the following pseudo-code:

Product.product\_name = Apples

Product.product\_price = $0.90

To use the code indirectly, you create an object instance of the class and use the object's variable with commands like the following pseudo-code:

objectP1= IO()

objectP1.print\_menu\_options()

One advantage of using the code indirectly is that you can have multiple object instances, each with a different address in memory (Figure 1.) The data for each instance is kept separate for each object, and each object would hold data about a different customer. In general, you use a class directly if its focus is on processing data and indirectly if it’s focus is on storing data.

**A Standard Class Pattern**

Classes have Fields, Constructors, Properties, and Methods.

Like scripts, class code follows a general design pattern in most of the languages. See below a pseudo-code with the different areas that make-up a Python class:

class MyClassName(MyBaseClassName):

# -- Fields --

# -- Constructor --

# -- Attributes --

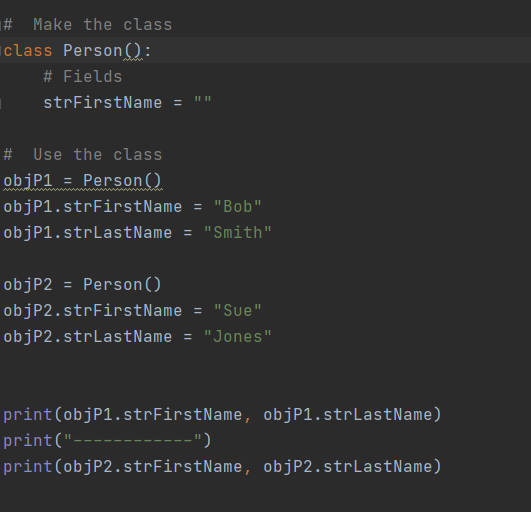
# -- Properties --

# -- Methods --

**Fields**

Fields are the data members of a class.

Fields are created using variables and constants. Listing 1 shows an example of creating two object instances using a class called Person. Each object instance can hold first name data unique to each person.



**Listing 1**: Creating two object instances using a class called Person

**Constructors**

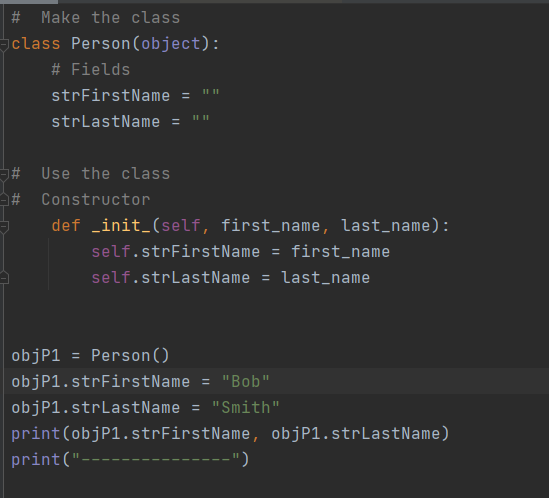
Constructors are special methods (functions) that automatically run when you create an object from the class. Constructors are often used to set the initial valuesof Field data.

Python Constructor's use the double underscore("duder") name of "\_\_init\_\_" and is the same name as the class. When you create an object instance from a class, you use the class's name as if it were a function.

objP1 = Person("Bob")

Python automatically calls the "\_\_init\_\_()" method and passes any arguments you

provide to the "\_\_init\_\_()" method each time you make a new object.



**Listing 2** shows an example of a constructor with two parameters

Since constructors are a specialized function, we use them as a function by passing arguments into the parameters. However, they only run once; when a new object instance of a class is created!

The Self Keyword

The "self" in the constructor method is used to refer to data or functions found in an object instance, but and not directly in the class. The code of a class always loads into memory when your script starts running. There the class code sits, waiting, even if it never gets used.

class Customer():

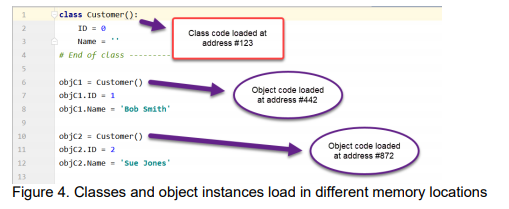
ID = 0

Name = ''

On the other hand, you must explicitly create an object instance in your script.

objC1 = Customer()

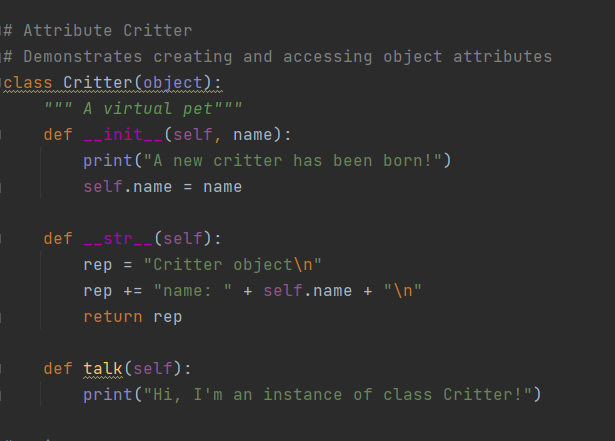
The class and the object instances are in different locations of a computer's memory. You can only load the class's code in memory once, but you can have multiple instances of a class, each representing a "copy" of the classes code.

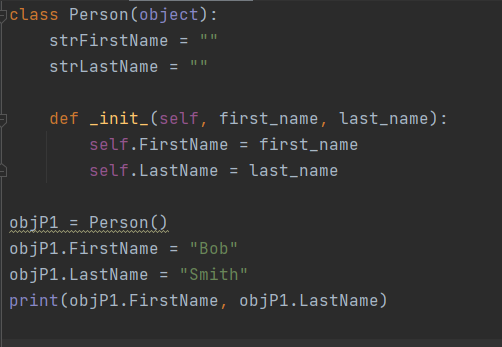


Because of this, you include a parameter called "self" in each method meant to be used from an object instance. Oddly, you do not pass arguments into this parameter. Nor is it not automatically assumed to be there if you forget to type it in.

**Attributes**

**In Python, Attributes are "virtual" fields that hold internal data.** An invisible field is created for you when you use the following syntax in the constructor.One problem with Fields or Attributes is that they are just variables. You do not have much control over what data goes into them unless you write specific code to validate values before they are assigned. To help with that, we use special methods (functions) called Properties.





Listing 3: Creating attributes in a class

**Properties**

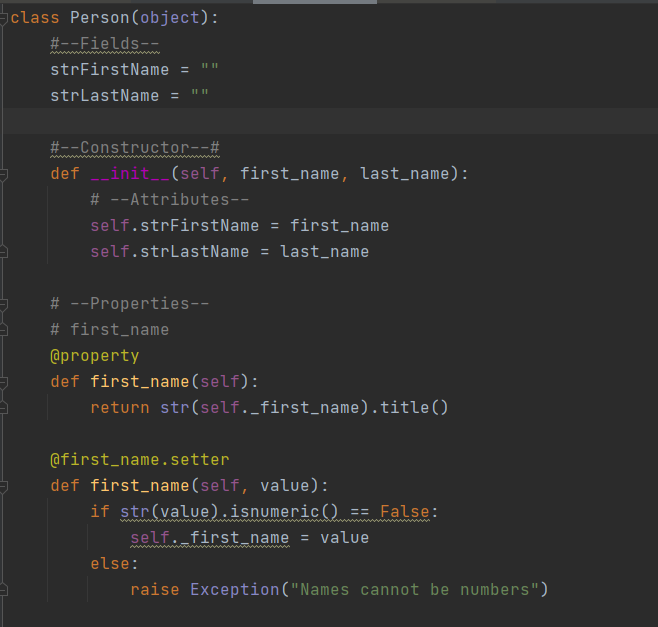
**Properties are functions used to manage field or attribute data.** We create two properties for each field/attribute: one for "getting" data and one for "setting” data. You may hear them called "Getters" and "Setters" or "Accessors" and "Mutators."

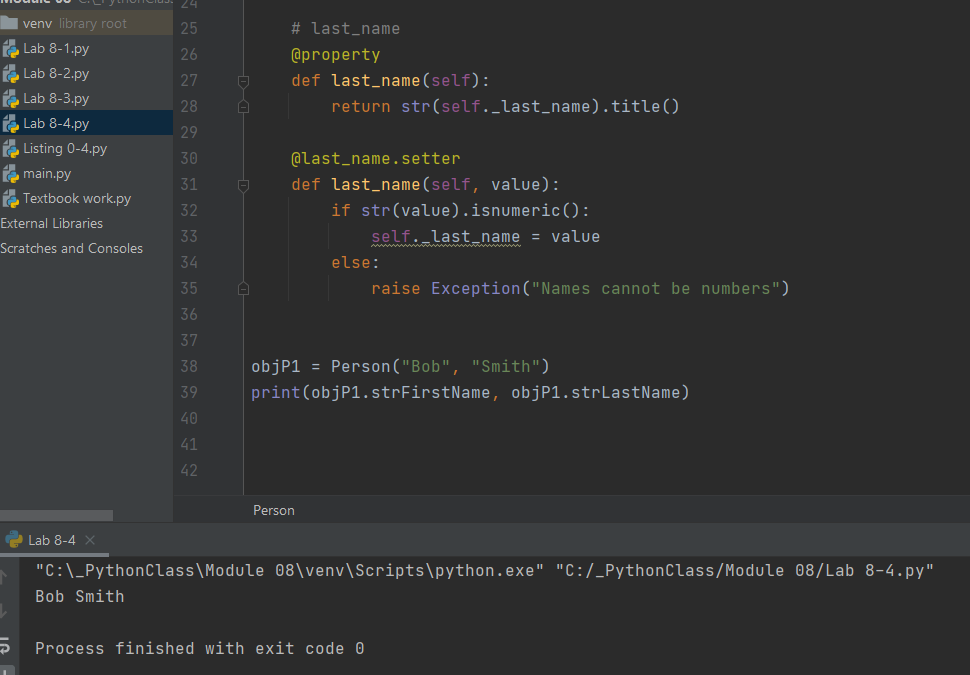
Setter Properties let you add code for both validation and error handling. If a value passed into the Properties parameter is valid, then it is assigned to the field or attribute. Creating a setter like any other function, but it must include the @name\_of\_method.setter directive, and the directive and function name must match!

Getter Properties let you add code to format a field's or attribute's data. Often, a Getter is included in a class, even if there is no formatting code. Inconsistently, Python use the @property directive to indicate a getter function.

**It is considered a best practice to only work with the data in a class through a Method or Property. This practice creates a layer of "Abstraction" and protects software using your class from internal changes to the Fields or Attributes**.

**Lab 8.4** shows properties for first\_name and last\_name in a class ‘Person’ with its output





**Lab 8.4** shows properties for first\_name and last\_name in a class ‘Person’ with its output

**Static Methods**

**If you want to include methods called directly from the class, without making an object first, you add the @staticmethod directive**:

class Math(object):

@staticmethod

def Add(Value1, Value2): # You do not need the self keyword

return Value1 + Value2

Now, you can call the method by using the name of the class and the name of the method.

Sum = Math.Add(5, 6)

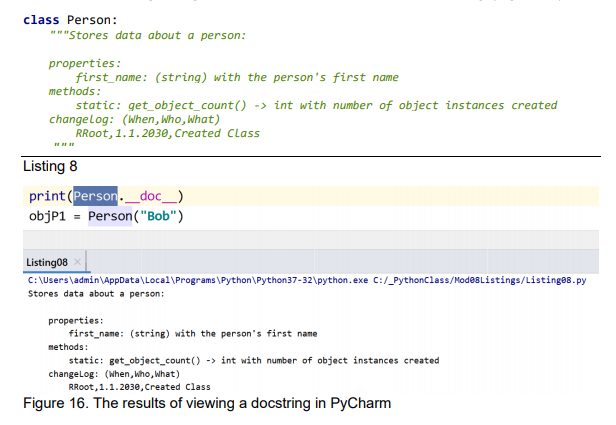
print(Sum)

Important: You do not need the "self" keyword with a static method since you are not calling the function from an object instance, and because the class is only loaded one, there is no need for a confusing pronoun!

Classes can have both instance methods and static methods, but most will not. **In general, when a class focuses on processing data, use "static" methods. However, when a class focuses on storing data, use "instance" methods (the ones with self**). For example, a class that holds a customer's name, email address, phone number, and other data about the customer, so use mostly instance methods because you would want to make multiple copies of the class as you program runs, each with its own unique customer data**.** On the other hand, a class that processes data from a list object into a file and from a file to a list object would be most about performing actions that would not require multiple copies of the class with unique data. This generalization may not always be true, but often enough that it provides a good starting point.

**DocStrings**

Just as we did with functions, we should include a docstring for our classes. Integrated development environments like PyCharm can display tooltips to show a developer's notes (use ctrl + q to activate this option in PyCharm). You can also show the "DocString" using the built-in and inherited \_\_doc\_\_ property (Figure 16).

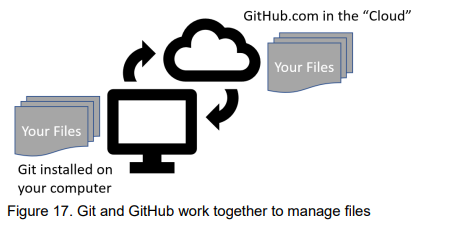


**GIT**

We've used GitHub's website through a web browser, but often developers work with GitHub differently. Instead of browser, they work with GitHub on their local computers using either with a command prompt application or a desktop application. In both cases, the communication between the GitHub website and the local computer is handled by a program called "Git."

**GIT**

**The Git software manages versions of one or more files. It allows to make a clone (copy) of a file, then make changes to clone, and save it as a new version of the same file. All while maintaining a copy of the original version. In addition to managing the cloned file on your computer, by default, Git uses the GitHub website to store backup files in the "Cloud."**

****

**GitHub Desktop**

In most organizations, developers use command shell to interact with the website from their local computers, but we start with something more visual; GitHub

Desktop is a free application you can install on both Windows and Mac OS.

**Git** is an open-source version control system that was started by Linus Torvalds—the same person who created Linux. Git is similar to other version control systems—[Subversion](https://www.howtogeek.com/66731/version-tracking-with-subversion-svn-for-beginners/), CVS, and Mercurial to name a few.

When developers create something (an app, for example), they make constant changes to the code, releasing new versions up to and after the first official (non-beta) release.

Version control systems keep these revisions straight, storing the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute.

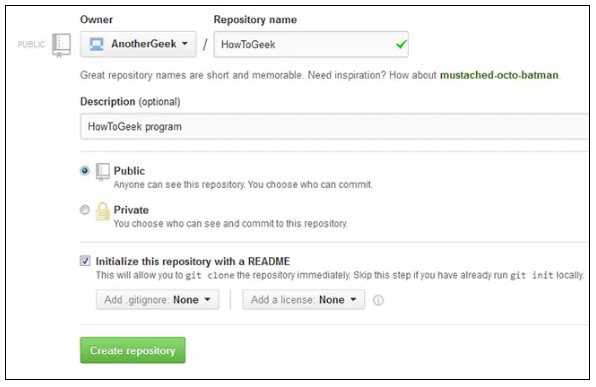
## **The “Hub” in GitHub**

We’ve established that Git is a version control system, similar but better than the many alternatives available.

Git is a command-line tool, but the center around which all things involving Git revolve is the hub—GitHub.com—where developers store their projects and network with like-minded people.

**Terminology**

### **Repository**

A repository (usually abbreviated to “repo”) is a location where all the files for a particular project are stored. Each project has its own repo, and you can access it with a unique URL.

### **Forking a Repo**

“Forking” is when you create a new project based off of another project that already exists. This encourages the further development of programs and other projects.

### **Pull Requests**

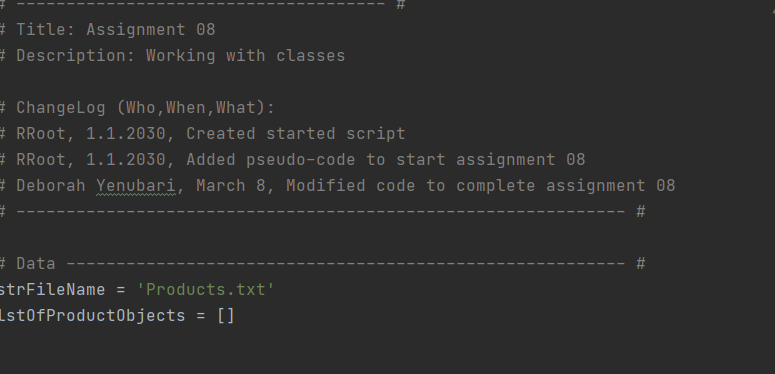
You’ve forked a repository, made a revision to the project, and want it to be recognized by the original developers—maybe even included in the official project/repository. You can do so by creating a pull request. The authors of the original repository can see your work, and then choose whether or not to accept it into the official project.

### **Social networking**

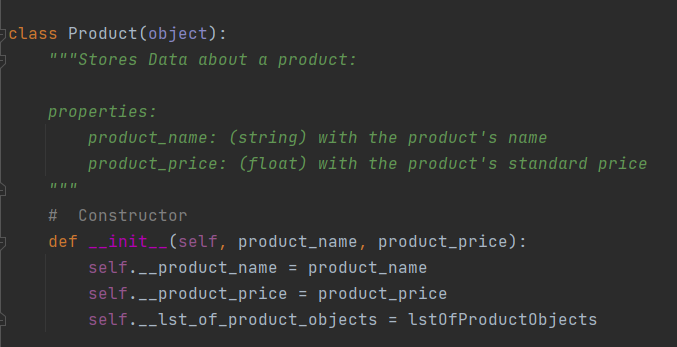
The social networking aspect of GitHub is probably its most powerful feature, allowing projects to grow more than just about any of the other features offered.

**Assignment 08**

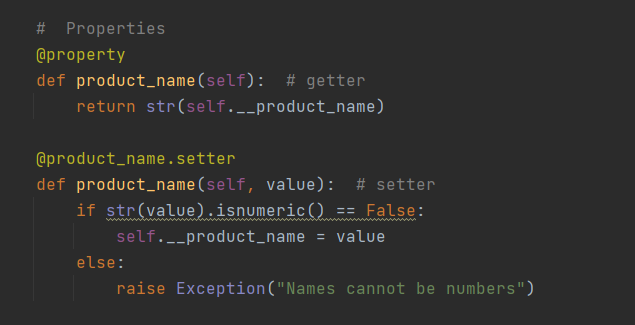
**Data includes the variables used in the program**

strFileName with the file’s name, list table with list of rows, product name and price – string variable

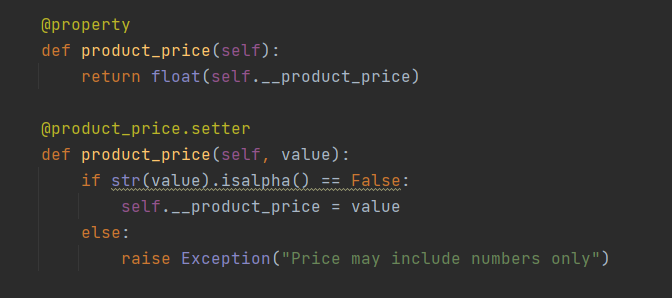
The first class used is **class Product** (object). The class stores data about each product, its name and price. Return as string arguments. List of product objects, a list table with rows. Data variables placed inside a class are ‘fields’.



Creating Constructor's using the double underscore("duder") name of "\_\_init\_\_" and is the same name as the class. Python automatically calls the "\_\_init\_\_()" method and passes any arguments we provide to the "\_\_init\_\_()" method each time we make a new object.



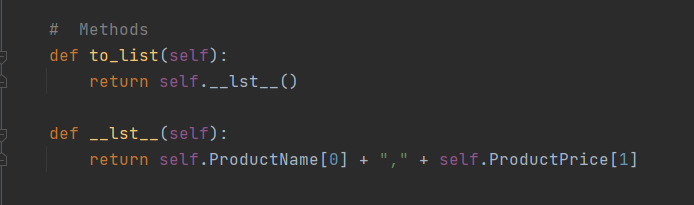
Setting initial values for data, product\_name, product\_price and lstOfProductObjects. Each time product\_name is used in the method, product\_name’s value is assigned to the variable**.**

****

To access the data, a property for product\_name and product\_price to return variables product\_name and product\_price.

One setter each for product\_name and product\_price , to set a value to each variable. (Setter Properties let you add code for both validation and error handling. If a value passed into the Properties parameter is valid, then it is assigned to the field or attribute. Creating a setter like any other function, but it must include the @name\_of\_method.setter directive, and the directive and function name must match!)

**Methods**



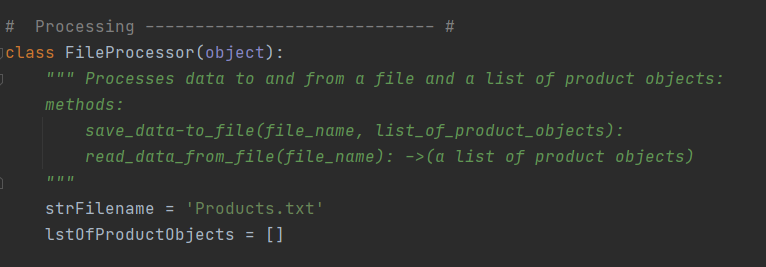
While functions that manage attribute data are called properties, other functions inside of a class are called Methods. Methods organize processing statements into named groups, just like functions in scripts do!

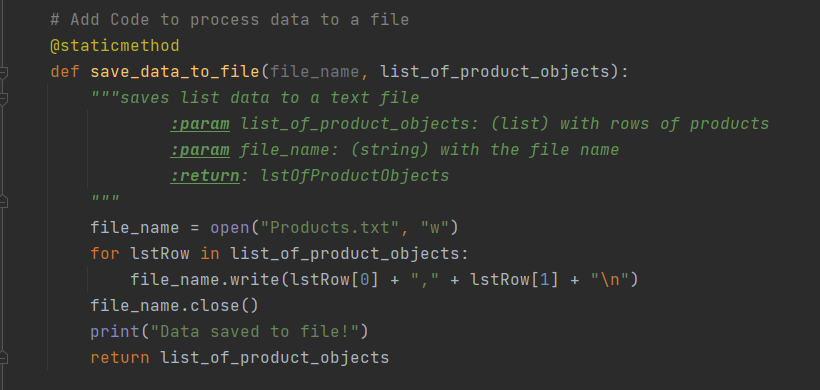
The "\_\_lst\_\_()" Method

Most classes, in most languages, include a method that returns some or all the class's data as a string. Python includes an invisible "\_\_str\_\_()" method if you do not add one to a class. This default invisible method only returns the name of class and an address identifier.

In this assignment, used the list method “\_\_lst\_\_()” to return the data as a list of rows, lstOfProductObjects each lstRow containing a ProductName and ProductPrice.

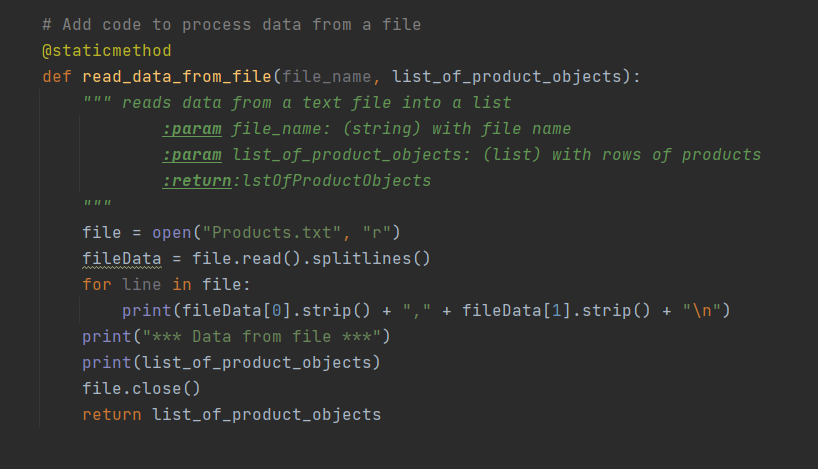
**class FileProcessor**

**** This class has two methods to save data into and read data from a file.



Under staticmethod, **def save\_data\_to\_file** with two parameters, file\_name , list\_of\_product\_objects

Open file, ‘Products.txt’ ,for each lstRow in lstTable, write item at index 0, followed by comma, followed by item at index 1 in lstRow. Close file. Print statement’Data saved to file’. Return variable list\_of\_product\_objects



Under staticmethod, **def read\_data\_from\_file** with parameters, file\_name, list\_of\_product\_objects. Open file ‘Products.txt’, read data in file and split lines.

For each line in file, print data into list per index position. Print statement, print list\_of\_product\_objects, close file and return variable list\_of\_product\_objects

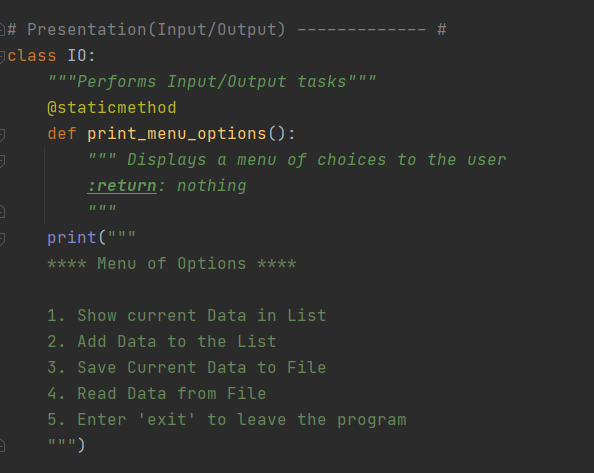
**Class IO**

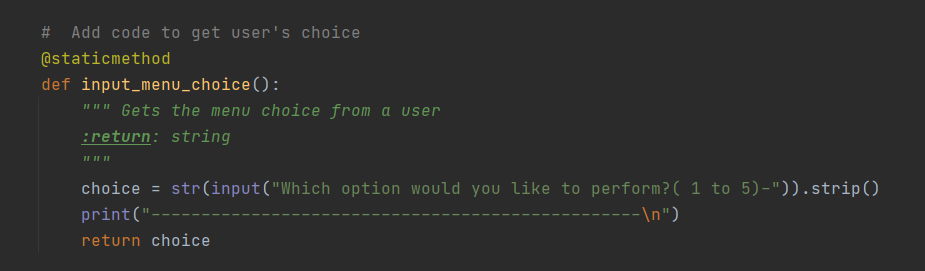
A third class for input/output functions

In this class, used the @staticmethod, as the Input output functions will be directly called, without making an object or instance first. We do not need the "self" keyword with a static method since we are not calling the function from an object instance.

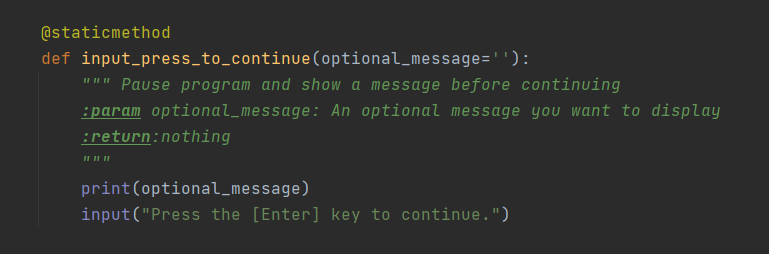
Classes can have both instance methods and static methods, but most will not. In this class IO, used three @staticmethod for 3 methods and two under instance methods.

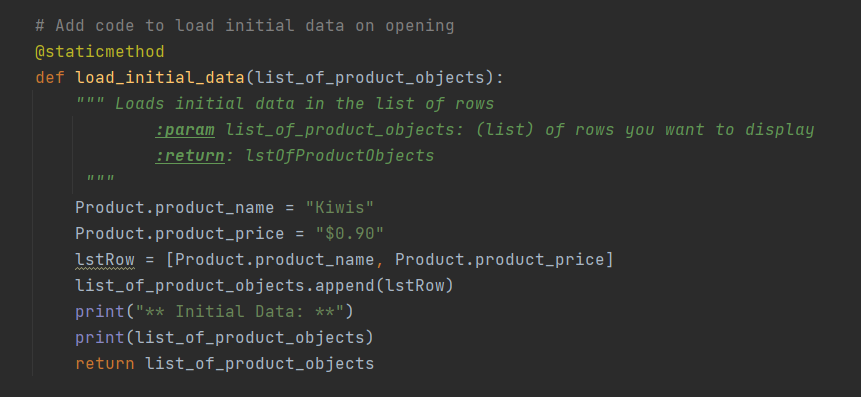
The first **def print\_menu\_options():** prints the User a Menu of Options from 1 to 5.



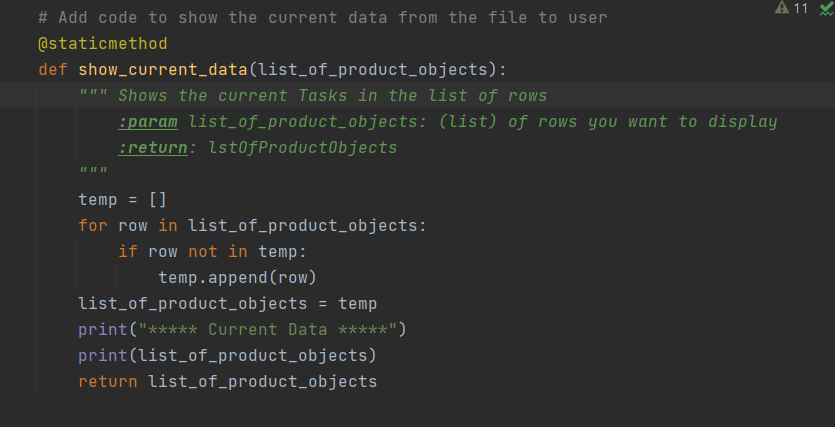
**def input\_menu\_choice():** allows the user to choose from 1 to 5 in the Menu of Options. It returns the str argument ‘choice’ which will be used in the Main Script.

**def input\_press\_to\_continue():** prompts the user to press Enter to continue with the Menu of Options.



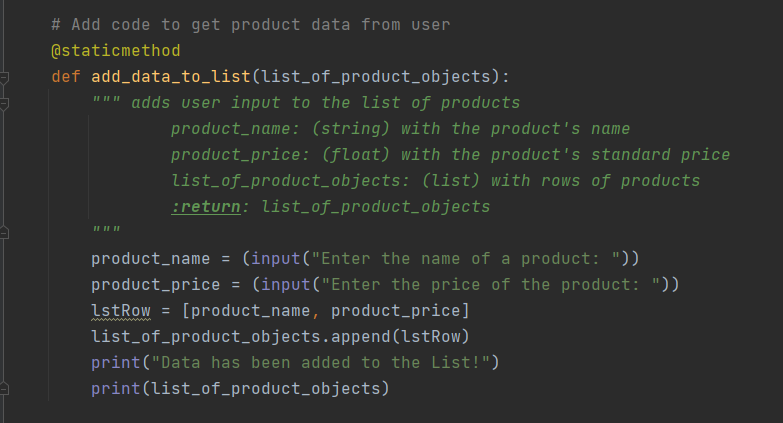
As data in lstTable kept repeating itself after initial load at start, have created two different functions, one to load initial data and another to show current data as updated by user.

**def load\_initial\_data** with parameter, list\_of\_product\_objects. Adding initial input to product\_name ,product\_price to a lstRow and appending to a table, list\_of\_product\_objects. Return variable list\_of\_product\_objects



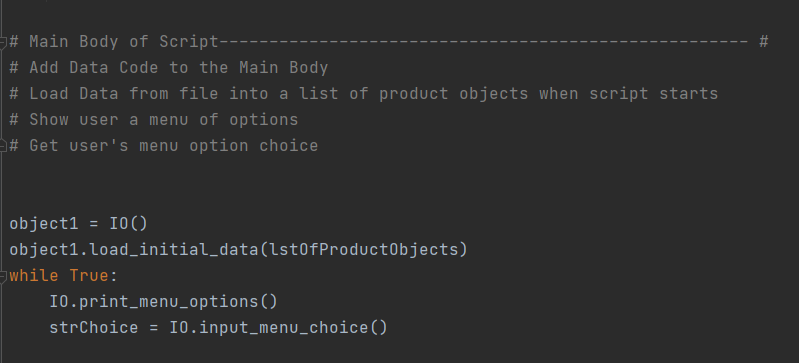
**def show\_current\_data** function with parameter, list\_of\_product\_objects. Creating a temp lstTable [], for each row in list\_of\_product\_objects, if row is not already found in temp lstTable, append to temp. Then equating it to list\_of\_product\_objects. So duplicates in the lstTable are removed and print.

**def add\_data\_to\_list** with parameter list\_of\_product\_objects. Allows user input for a product\_name and product\_price , prints in a lstRow and appends to list\_of\_product\_objects. Function calls for method under class ‘Product’

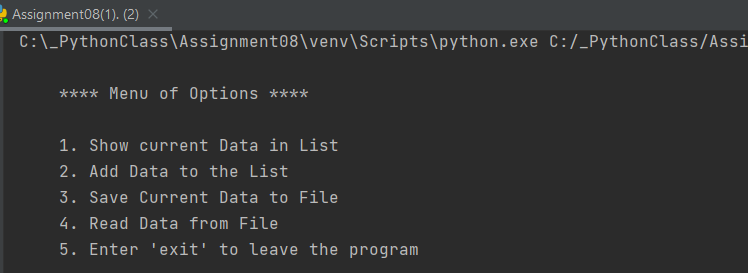


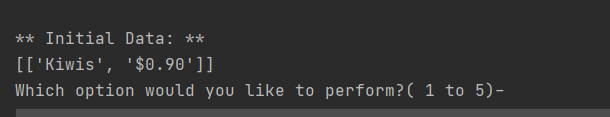
**Main Script**

Using a ‘while: True’ loop to print Menu of Options from class IO. The IO method is directly called as it’s a staticmethod function. The argument returned from the method input\_menu\_choice in class IO, needs to be assigned to a string variable, so it can be used in the main script. It needs to be accessed to be used.



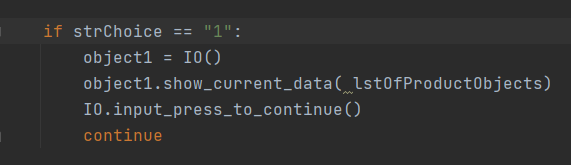
P.S Was unable to print the Menu of Options each time ‘Enter’ was used, though it did print the input\_menu\_choice as long as required within the ‘while’ loop.

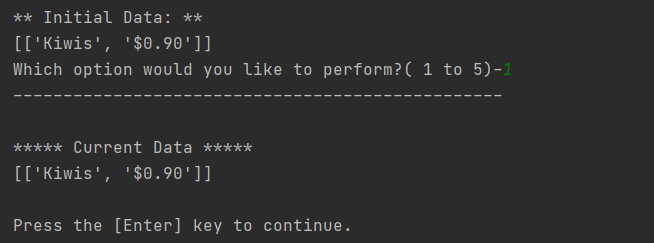




Output 1: Printing Menu of Options and asking for option.

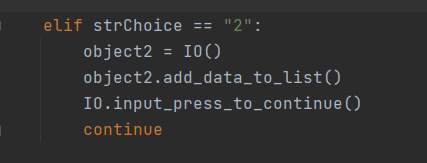
If user\_input\_choice or the variable strChoice equates 1, indirectly calling for method show\_current\_data(lstOfProductObjects) to display current data in lstOfProductObjects. Calling for method input\_press\_to\_continue from the same class IO.

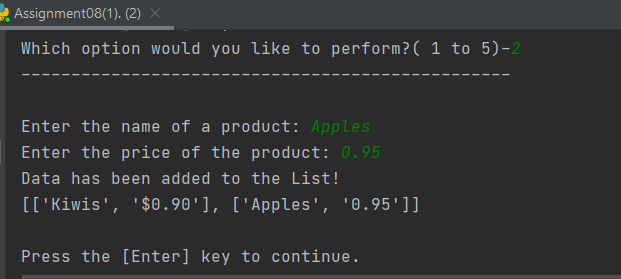




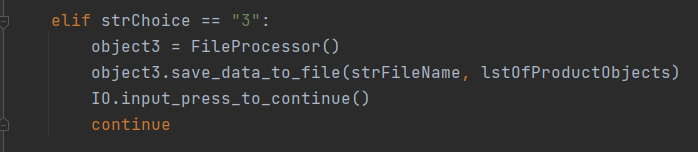
Output 2: On choice ‘1’, prints current data

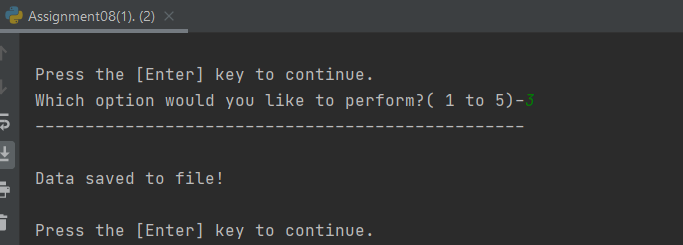
If user\_input\_choice or the variable strChoice equates 2, indirectly calling for class IO method add\_data\_to\_list() to add data in a lstRow to a lstTable called lstOfProductObjects. Directly calling for method input\_press\_to\_continue from the same class IO.



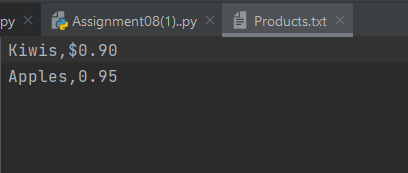
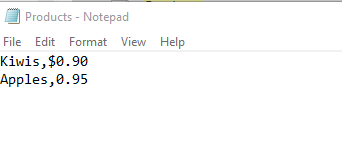


If user\_input\_choice or the variable strChoice equates 3, indirectly calling for class FileProcessor method save\_data\_to\_filet() to save the lstOfProductObjects to a text file, ‘Products’ .Directly calling for method input\_press\_to\_continue from the same class IO.

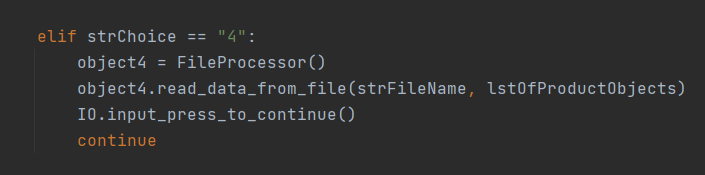




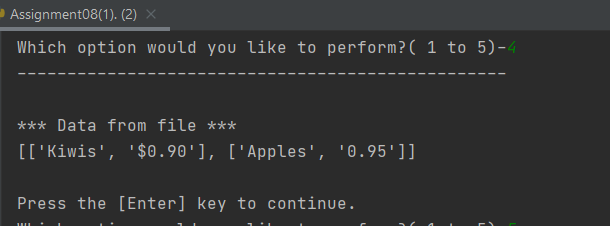
Output 4: On strChoice equating 3, Data saved to file.



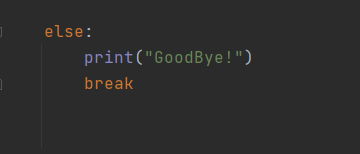
Data saved in ‘Products.txt’ on Pycharm and Notepad

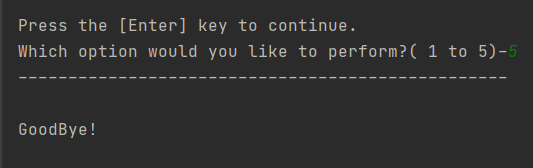


If user\_input\_choice or the variable strChoice equates 4, indirectly calling for class FileProcessor method read\_data\_from\_file() data from a text file named ‘Products’ into a lstTable named lstOfProductObjects. Directly calling for method input\_press\_to\_continue from the same class IO.



Output on strChoice == 4, data read from file





Exit the program on strChoice == 5. Entering exit or another input aside from the listed choices breaks the while loop and exits the program.

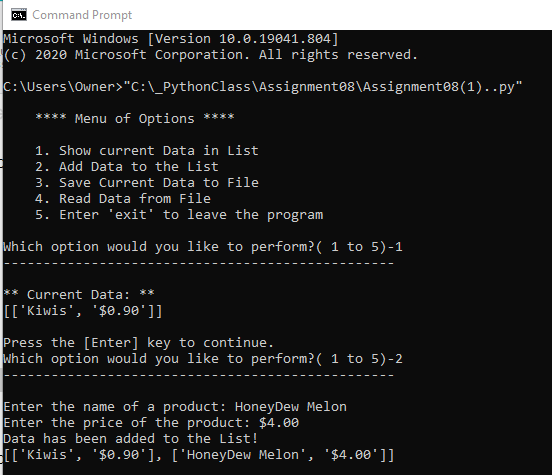


Image 1: The program run using Command prompt

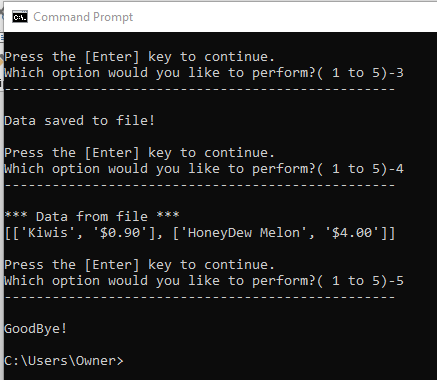


Image 2: The program run using Command prompt

**Conclusion**

The program used constructor class, getter and setter properties for a class ‘Product’, instance methods for class ‘FileProcessor’, static methods for a class ‘IO’. An option to try would be running the main script using a def main(). Was unable to execute the ‘while’ loop to print’ menu of options, though otherwise was able to execute the program successfully.