Hitakshi Verma

August 29, 2020

Foundations of Programming: Python

Assignment 08

<https://github.com/hitakshi01/IntroToProg-Python-Mod08>

**Create a script using object-oriented programming techniques**

**Introduction:**

In this Module 08, Randal Root, our professor, has introduced me with new object- oriented programming concepts in python. I have learnt how to create classes, objects, attributes, properties and methods or functions in a class. As a part of this assignment, I am going to share my gained knowledge and how using knowledge I have created a python script.

**Classes vs Objects:**

Python is an object- oriented programming language. Classes provide a means of bundling data and functionality together. Data in a class is defined using *variables or constants*. When data is in a class, it is called as *Field* and functions in a class is called as *Methods*. Class can be created by using *class* keyword. For example – *Student* is a class which is a collection of data elements or fields as studentId, studentName etc. We can write student class as *class student*

There are 2 ways to access class’s code – directly and indirectly.

For *directly*, we write as *classname.field* :

student.studentId

and

student.studentName

For *indirectly*, we create an *object* of a class which is an instance of a class. We can now access the data of a class using object.

Obj1= Student()

Obj1.id = 100

Obj1.name = “Hitakshi”

*In general, you use a class directly if its focus is on processing data and indirectly if its focus is on storing data.*

*Classes typically have Fields, Constructors, Properties, and Methods.*

class MyClassName(MyBaseClassName):

# -- Fields –

# -- Constructor –

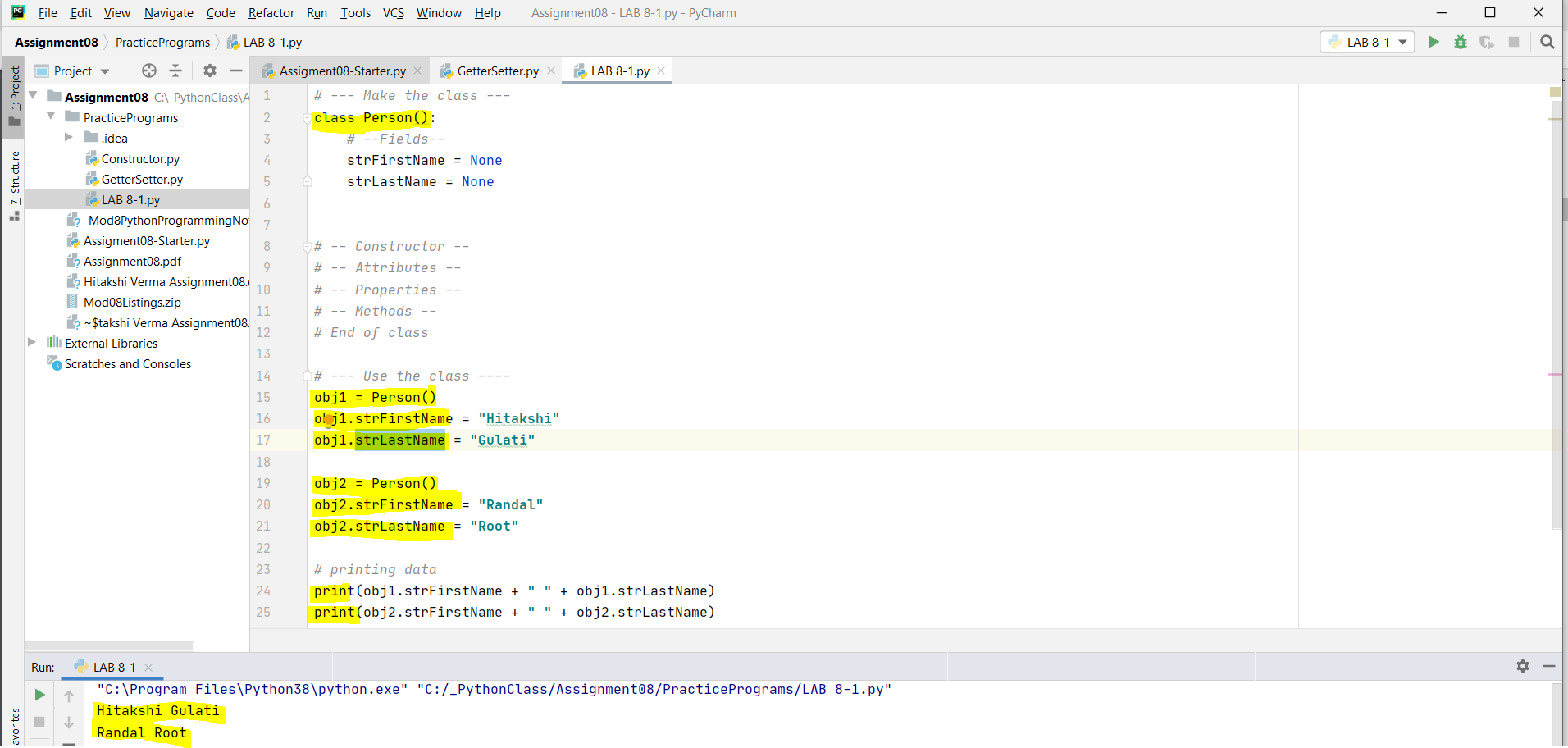
# -- Attributes –

# -- Properties –

# -- Methods –

**Working on Lab 8-1:**

Here, I have created a python script using class as Person and added class fields as strFirstName and strLastName. Then creating an object instance of the same class and setting the class values in created instance. Finally, printing the instance values.



*Figure1 demonstrates the class and object instance*

**Constructors:**

Constructors are special methods (functions) that automatically runs when you create an object from the class. It is another way of initializing or setting field values.

It uses double underscore name of “\_\_init\_\_”

Obj1= Person(“Hitakshi”) *# constructor using one argument*

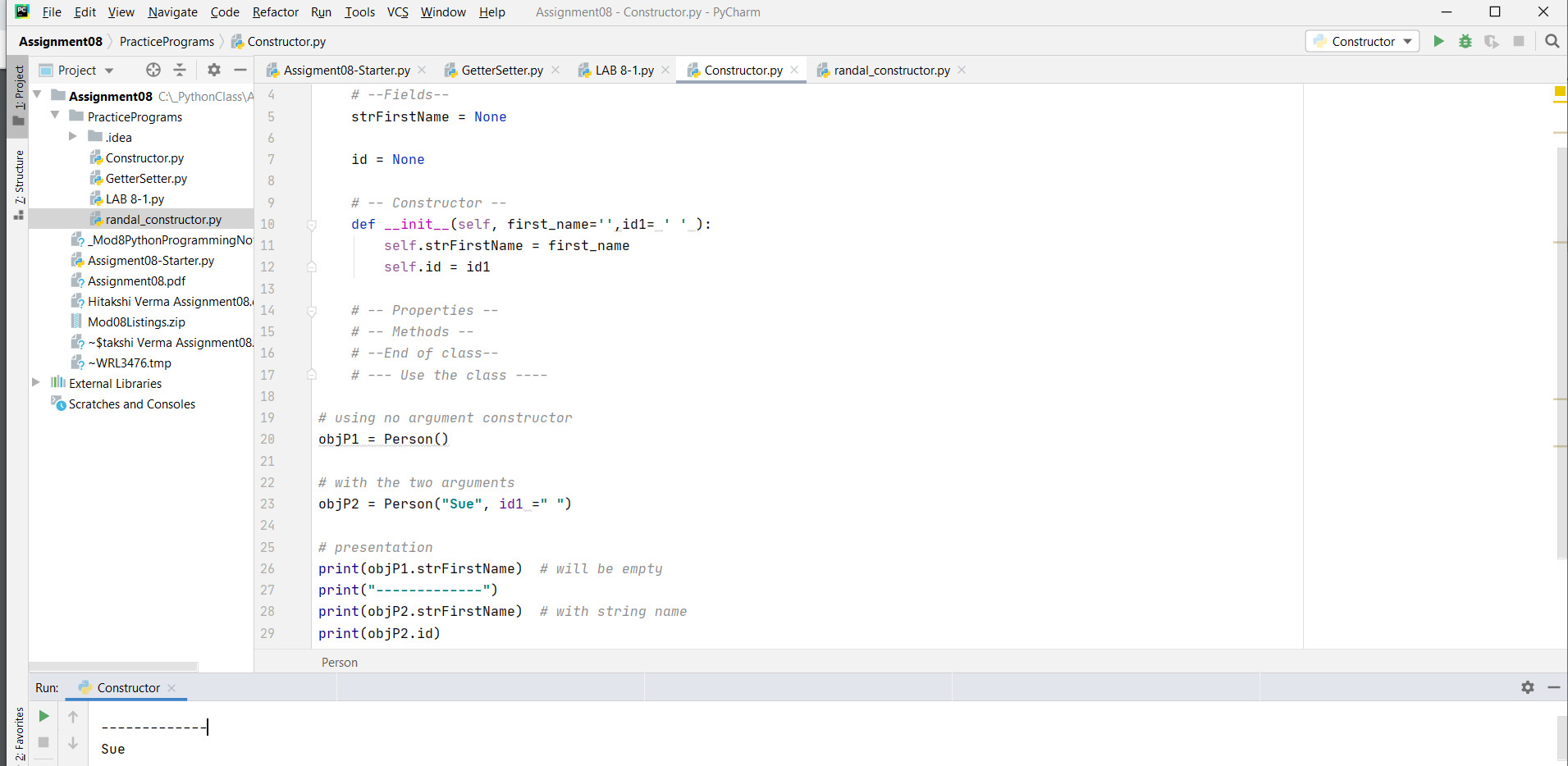
*Python automatically calls the "\_\_init\_\_()" method and passes this arguments which I provided to the "\_\_init\_\_()" method*

*# -- Constructor –*

*def \_\_init\_\_(self, first\_name =''):*

*#-- Attributes –*

*self.strFirstName = first\_name*



*Figure 2 demonstrates constructor*

**Destructors:**

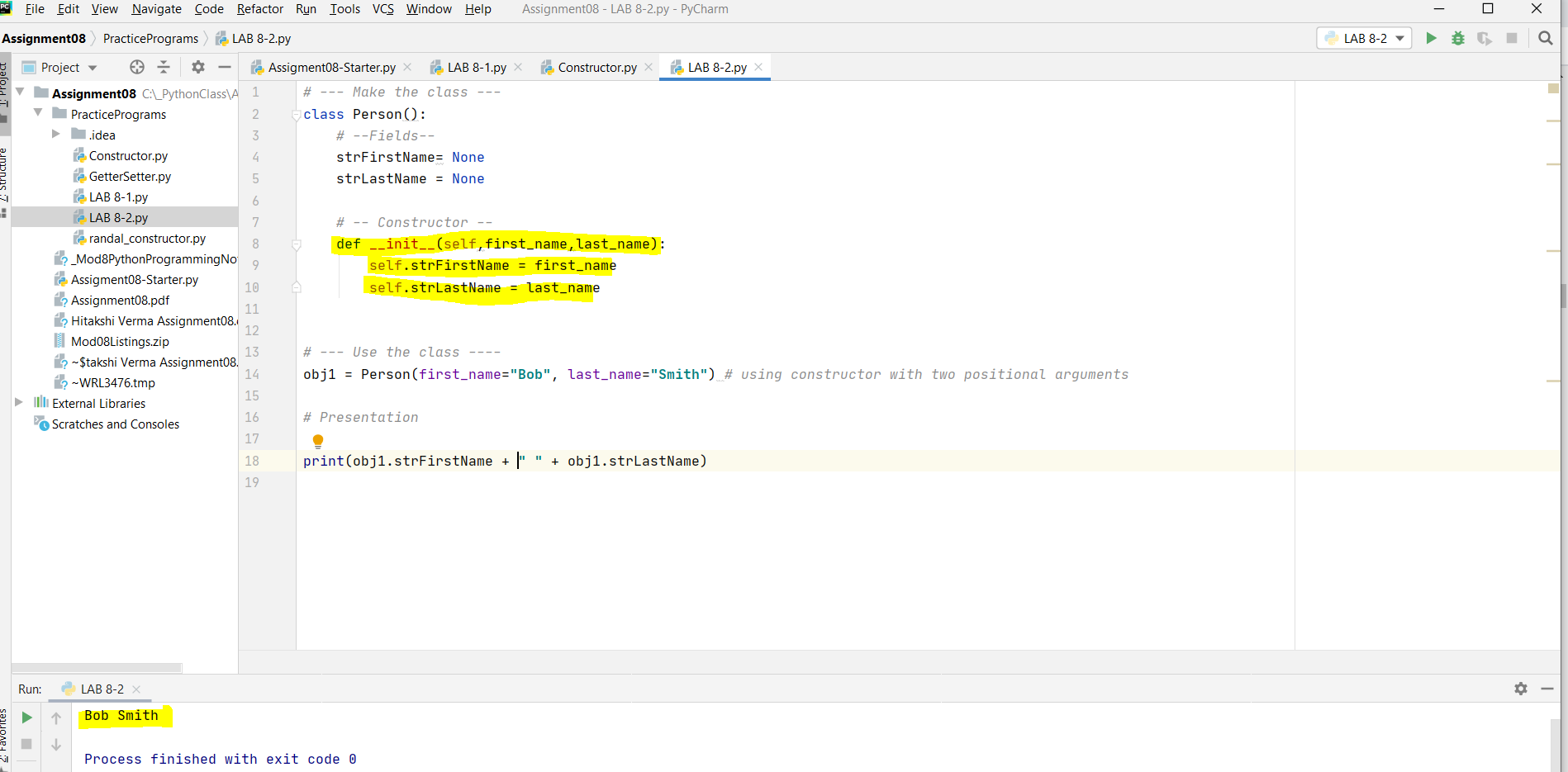
Destructors are advanced functions that python automatically runs when object instance is not needed in running program.

*def \_\_del\_\_(self):*

*""" automatically called when object is destroyed"""*

**Working on Lab 8-2:**

I have created a python script using constructor passing two arguments in its parameter and finally printing those values of class fields.



*Figure 3 demonstrates the passing two argument values in constructor’s parameter*

**Attributes:**

Attributes are "virtual" fields that hold internal data. *Class attributes* belong to the *class* itself they will be shared by all the instances. Such attributes are defined in the *class* body parts usually at the top, for legibility.

**Working on Lab 8-3:**

I am creating a python script where I am using constructor and passing argument’s FirstName and LastName values into parameters. Used implicit attributes instead of explicit class fields

*Figure 4 demonstrates the passing two argument values in constructor’s parameter and using implicit attributes*

**Properties:**

Properties are functions used to manage field or attribute data. Typically, we create two properties for each field/attribute, one for "getting" data and one for "setting data. In fact, you may hear them called "Getters" and "Setters".

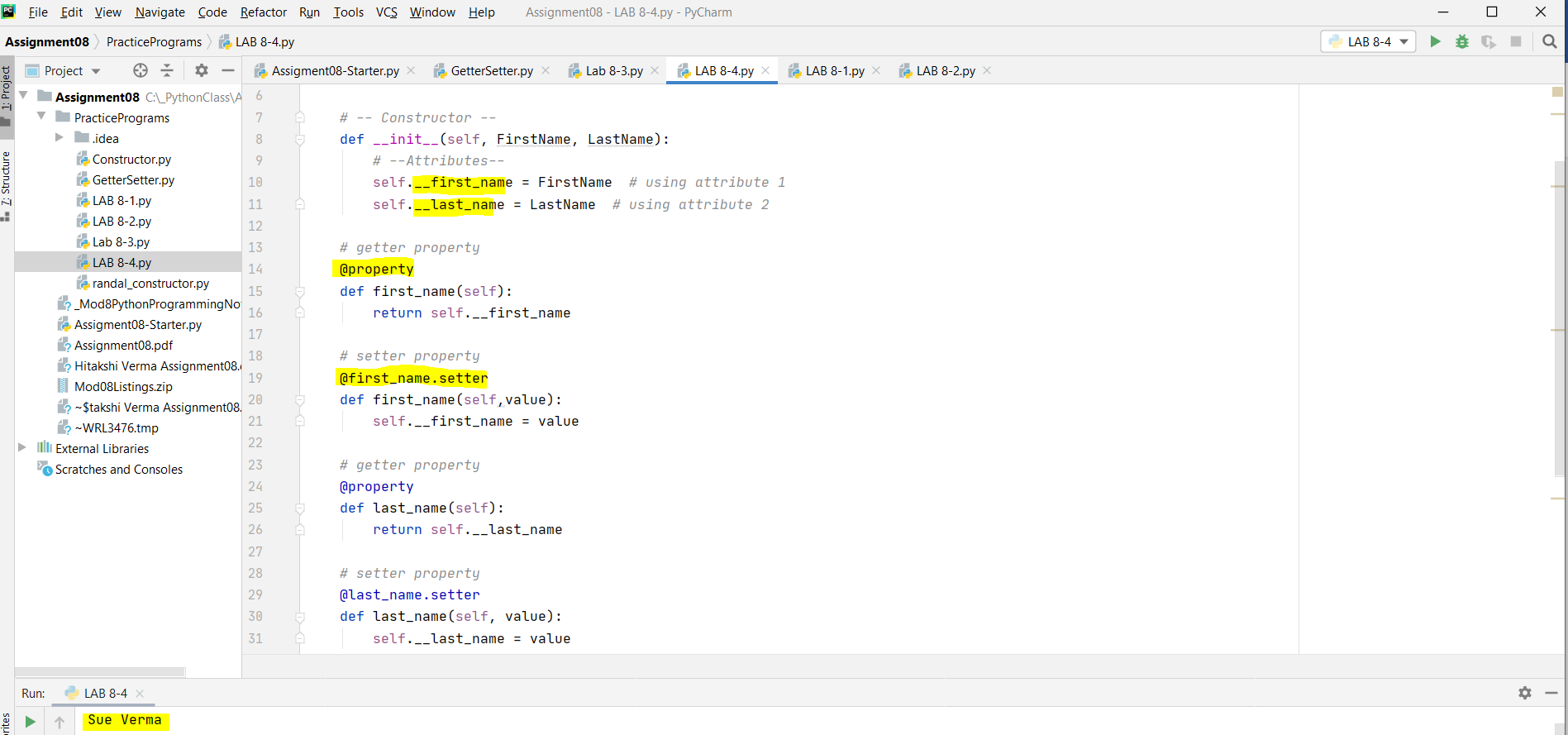
*Setter Properties* let us add code for both validation and error handling. If a value passed into the Properties parameter is valid, then it is assigned to the field or attribute.

G*etter Properties* let us 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.

When using Properties, we "*hide*" the attribute using (2) underscores before the attribute's name, which makes the attribute "*private*"

**Working on Lab 8-4:**

I have created a python script using getter and setter properties to change attribute values.

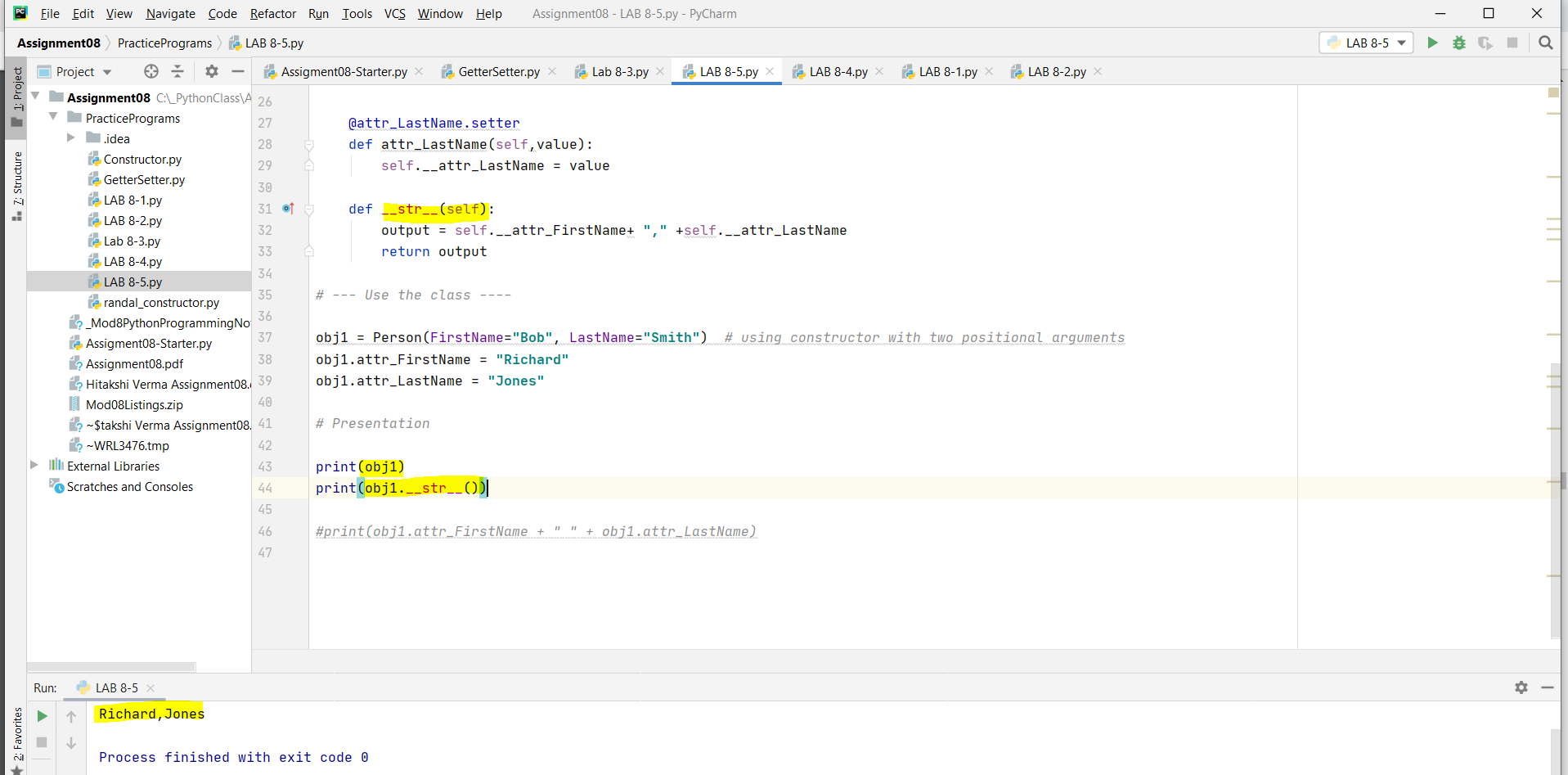
*Figure 5 demonstrates getter and setter properties*

**Methods:**

Functions inside a class other than properties is known as “*Methods*”. Methods allow you to organize your processing statements into named groups.

**Working on Lab 8-5:**

I have created a custom script to override the \_\_str\_\_ method which returns comma separated first name and last name.



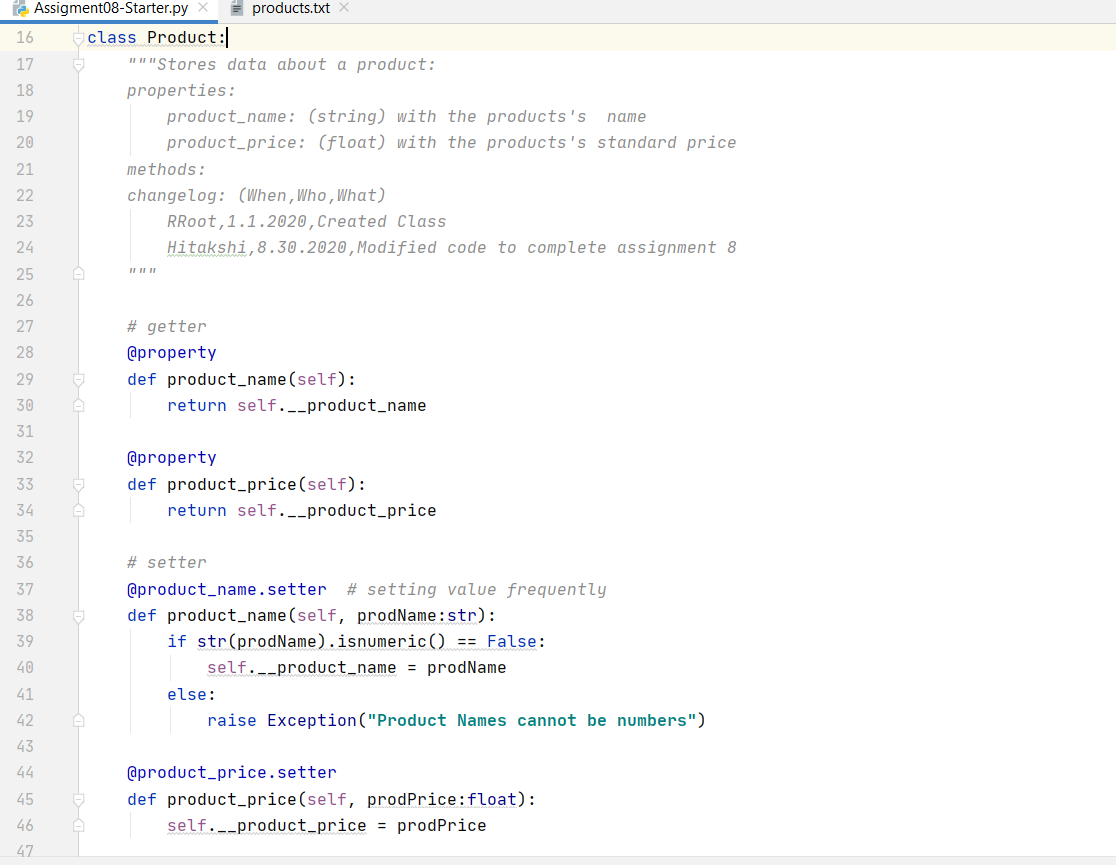
*Figure 6 demonstrates the overridden of str function*

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).

**Working on Assignment 08:**

For this assignment, I was provided with a pseudo code. As part of the code, I added individual functions and error handling. The following classes were available as a placeholder –

**Product class:** This class is used to store product name and product price. I added getter and setter for both the attributes. I used annotations like *property* and *<attribute\_name>.setter*



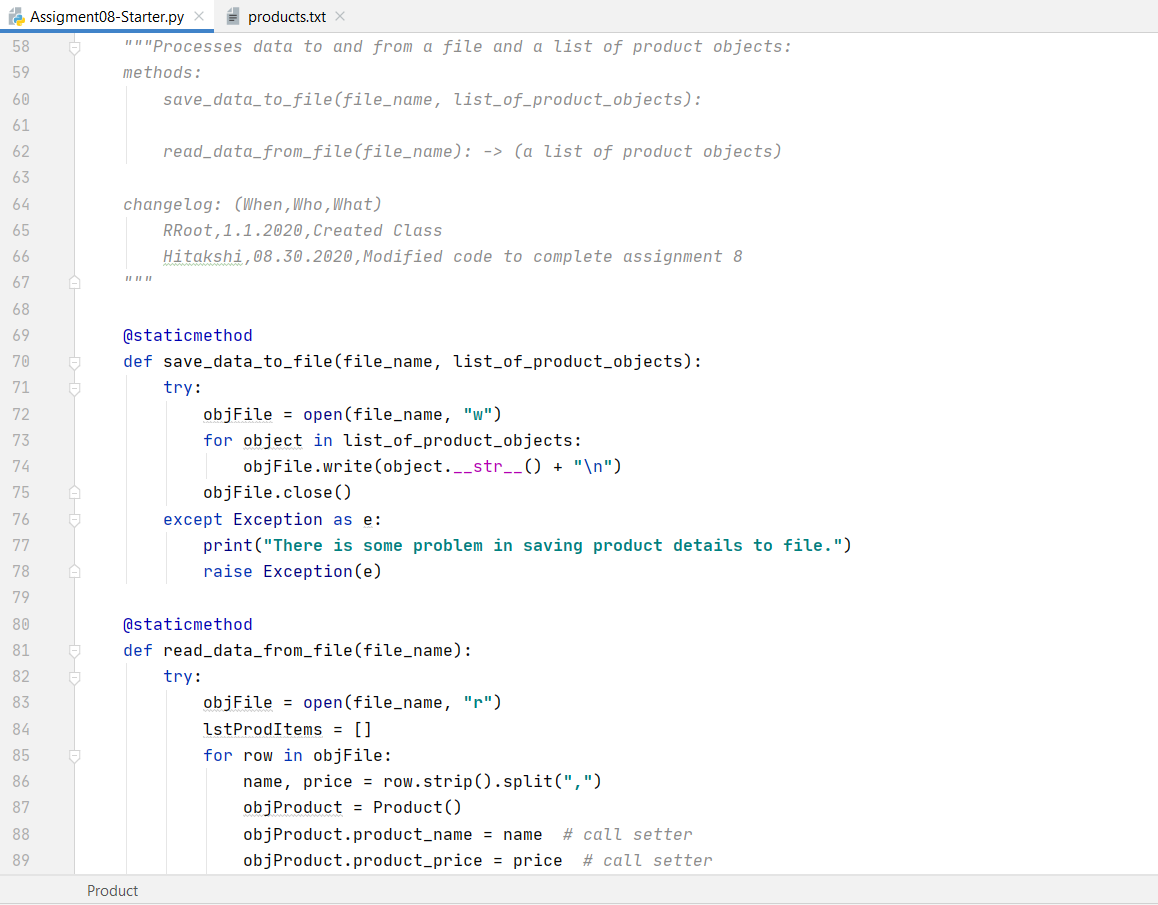
*Figure 7 demonstrates Product class*

I also overrode the existing \_\_str\_\_ method and returned a custom- comma separated product\_name and product\_price.

**FileProcessor class:**

The FileProcessor class consisted of 2 methods. Since this class was used for processing data, I used the @staticmethod annotation.

* save\_data\_to\_file() – takes filename and a list of product objects as input. Similar to older assignments, I opened a file in “w” mode and looped through the list to store the data in file. Further, I also added the code in try-except to ensure error handling.
* read\_data\_from\_file() – takes file name in the input. I opened the file in read mode. I iterated over the file, sanitized the input using strip() and split() using comma. Further, I created an instance of the Product class and used setter to store the name and price. Finally I appended the object to the list. I put the code in try-except to catch any processing errors.



*Figure 8 demonstrates in continuation of Product class*

**IO class:**

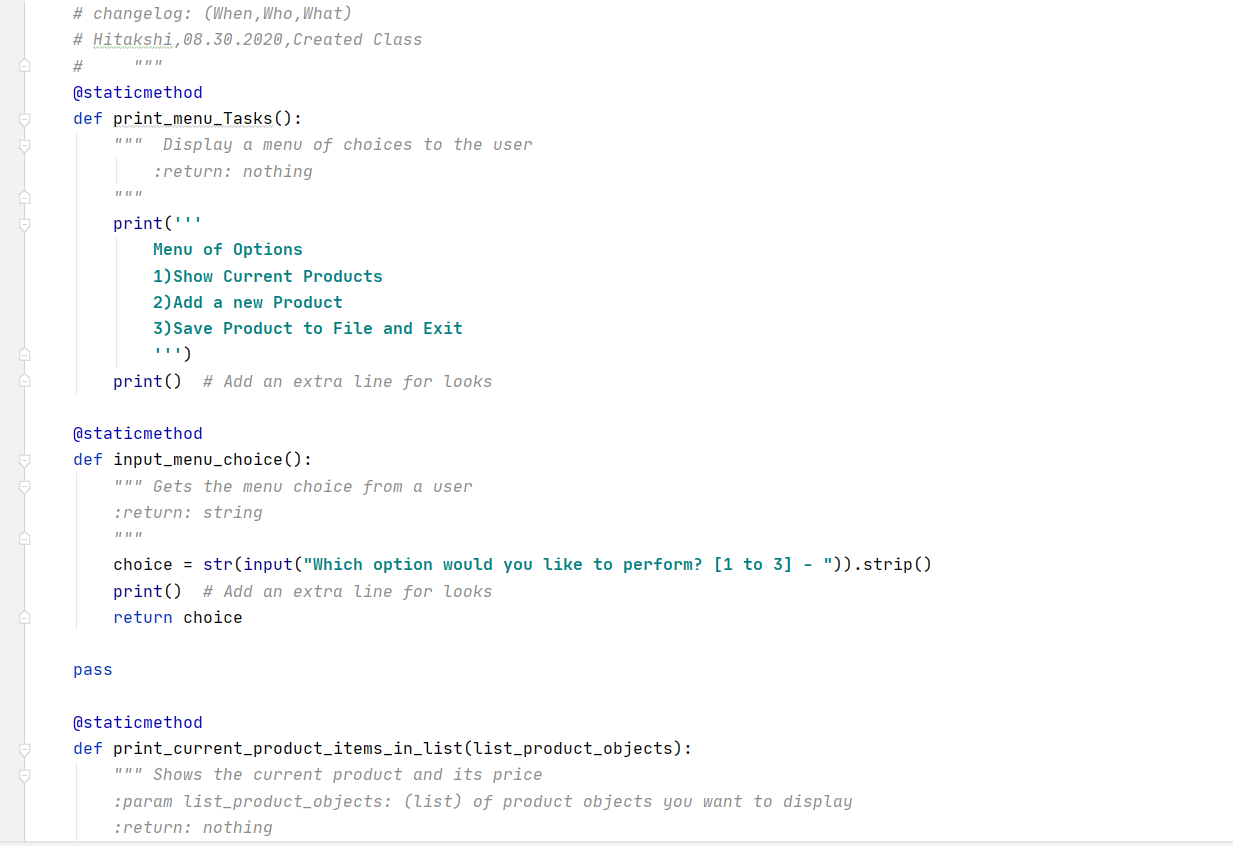
This class was used to process data and perform input/ output operations. this class was used for processing data, I used the @staticmethod annotation.

The following methods were used –

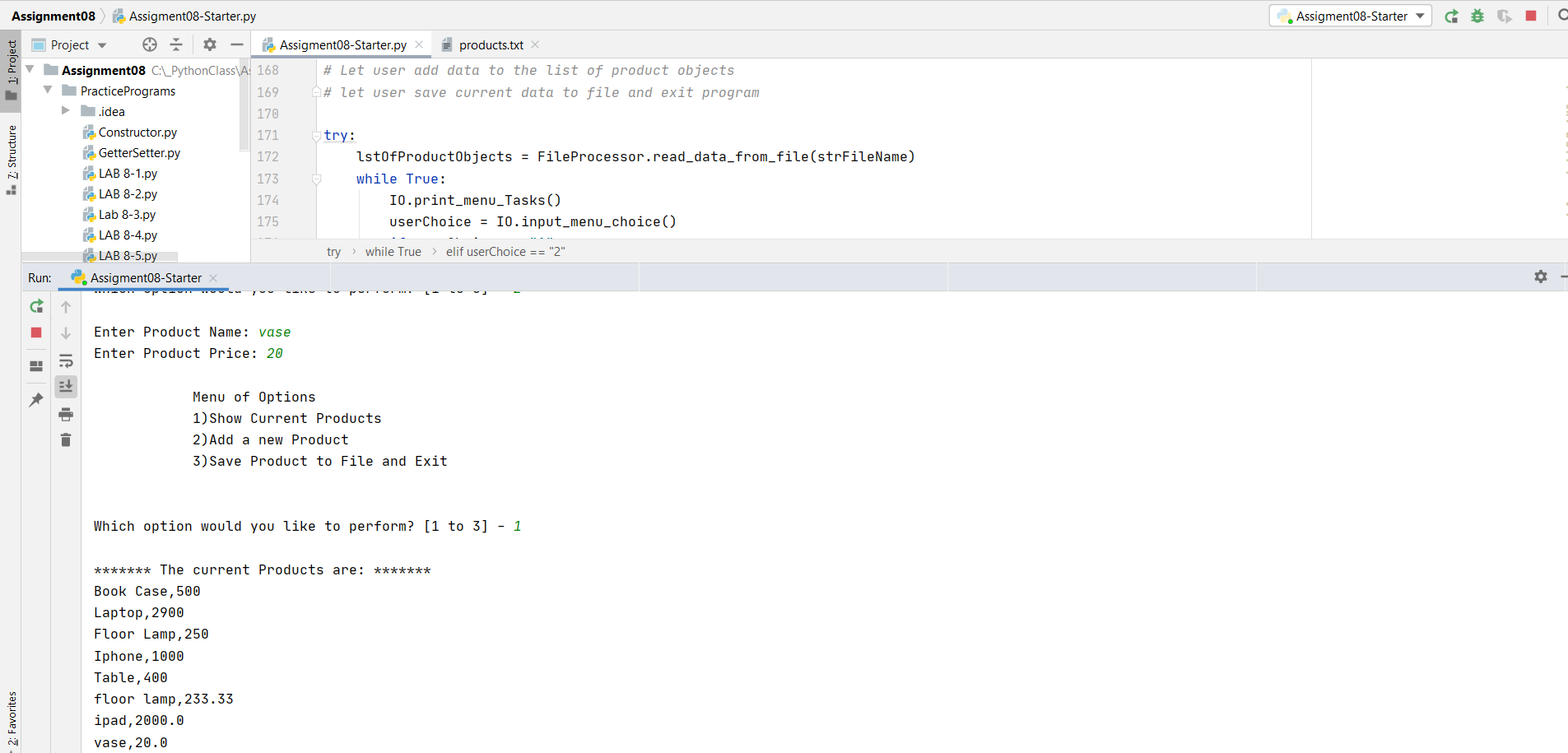
* print\_menu\_Tasks() – displays the 3 options to the user to choose.
* input\_menu\_choice() – takes an input from the user to choose the operation
* print\_current\_product\_items\_in\_list() – takes list of product items in the input. I used a for loop to iterate over the list and simply use the print statement. This used the \_\_str\_\_ method of the Product class and prints the data to user in the format *productName,productPrice*
* add\_product() – This function takes 2 input from the user. The name is taken in string format and price is taken in float format. This was returned as a tuple.

In the main body of the script, I started with loading the data from the file into a list of product object. I used a while loop which runs unless the user selects option 3 – *Save and exit*. Within the loop, I used the IO. print\_menu\_Tasks() to display the choice to the user and IO.input\_menu\_choice() to take the input. I used if-elif-else condition to run the appropriate function.

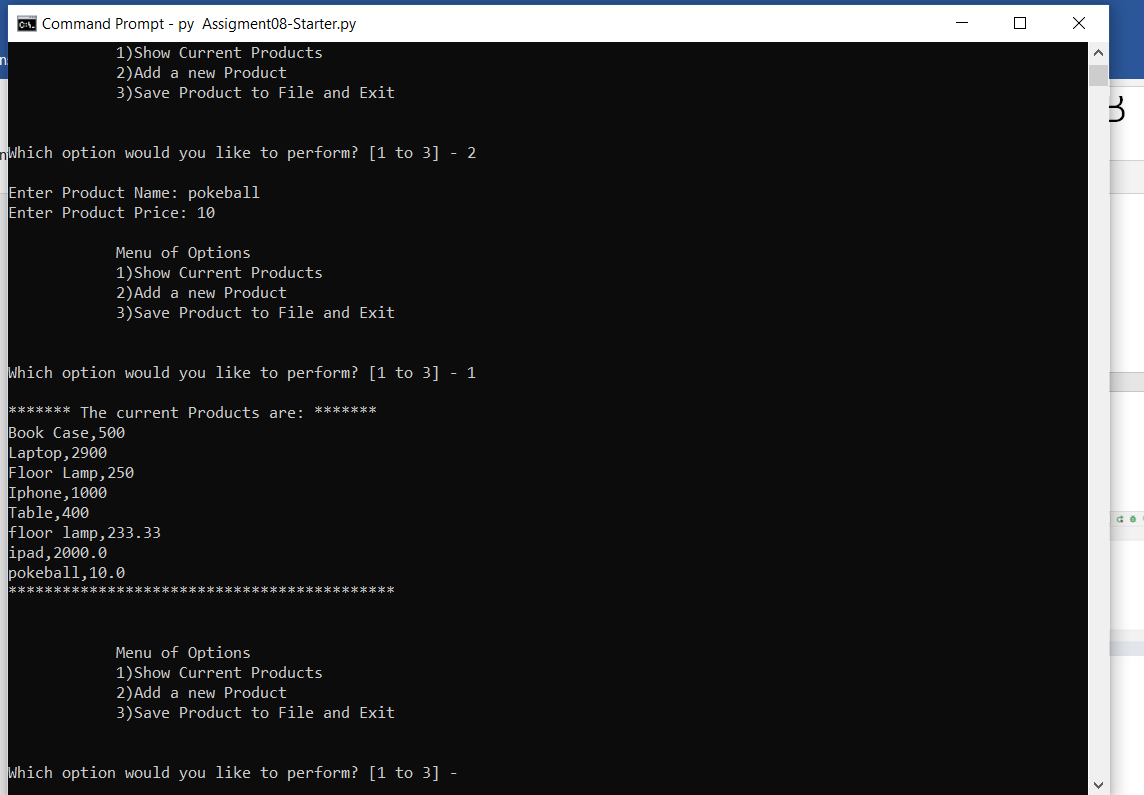
* If “1” was selected – I used IO.print\_current\_product\_items\_in\_list()
* If “2” was selected – I used IO.add\_product() to return data as a tuple and create a new instance of the Product class which was appended to the list.
* If “3” was selected, I saved the data into the file and exited the program after displaying a print statement.
* For everything else, the script will ask the user to select correct option.



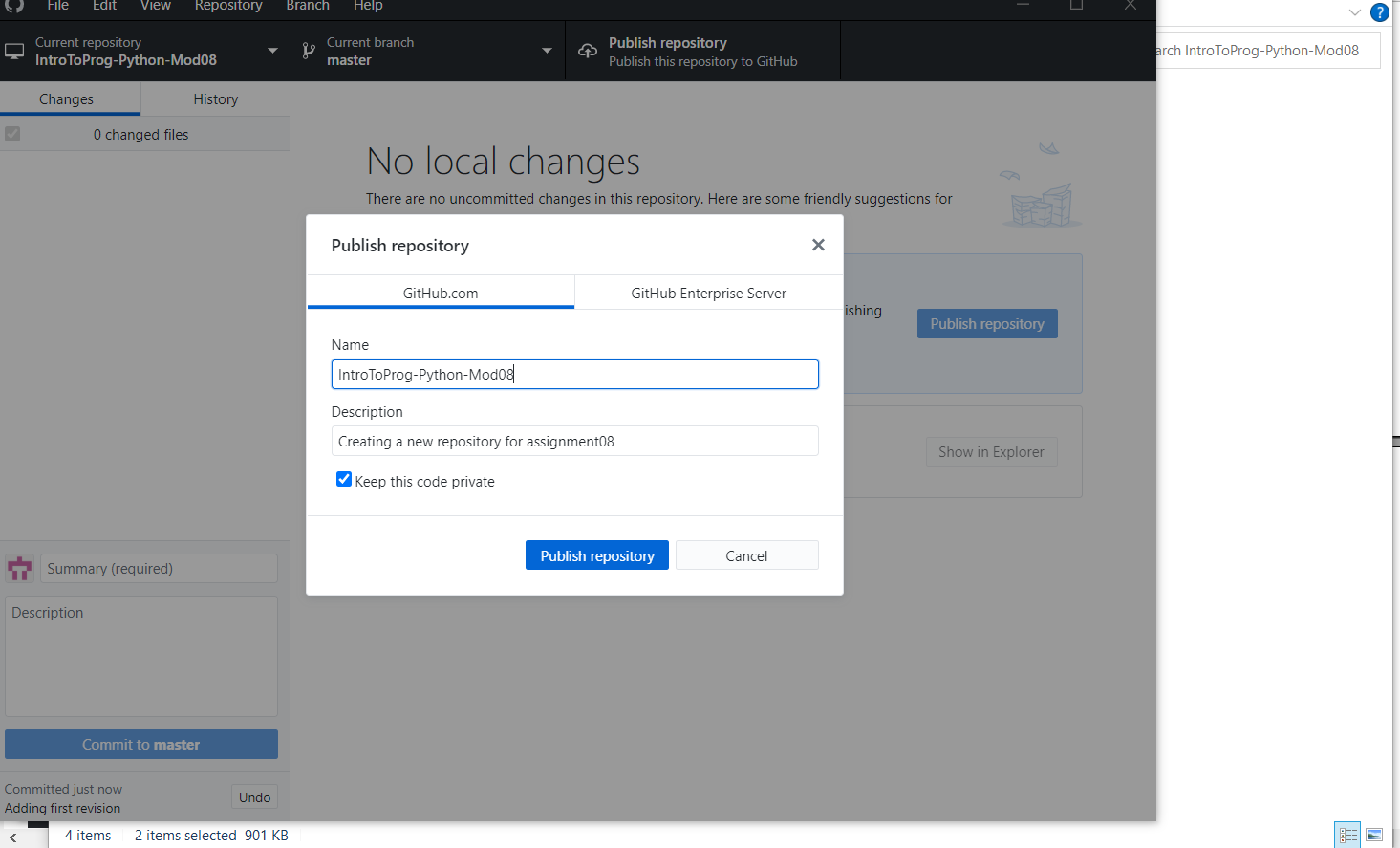




*Figure 10 demonstrates running script in PyCharm*



*Figure 11 demonstrates code script running in OS Shell*



*Figure 12 demonstrates creating repository in GitHub desktop*

**Summary:**

I have learned about classes and its properties, attributes and methods. Created a python script where i used the pseudo code and completed the script by navigating back and forth from one section to another. This made me realized; how important is for each class to interact with another and as a whole. At the end, familiarized with a new platform “GitHub Desktop” where I now started using to push my changes.

*Getting a feeling of being a professional python programmer!*