Inderjot Brar

March 9, 2021

IT FDN 110 B

Assignment 08

<[github\_url](https://github.com/IBRAR21/IntroToProg-Python-Mod08)>

ASSIGNMENT 8

**Introduction**

This assignment documents the steps I took to create and test a Python Script that uses classes and objects to organize the code that allows user to add product data to the file. The script makes use of class constructors, properties, methods, attributes and exception handling.

## **Product Class**

In the Assignment06\_Starter.py script, I first added the code for the “*Produc***t**” class object. I created a class constructor that would be initialized whenever a new product object is created. Within the constructor, I created 2 private attributes: “*name*” of string type and “*price*” of float type. Private Attributes are useful to prevent access outside of object’s own methods. Next, I created the properties and name setters for each of these attributes. I also created 2 methods within the Product class: a) *“\_\_str\_\_”* to return product data as a string, and b) “*add\_data\_to\_list*” to append the product attributes to a list of rows.(Figure 1)

A picture containing table

Description automatically generated

***Figure 1: Code added for ‘Product’ class***

## **Processor Class**

Next, I created 2 functions in the Processor class to process data to and from a file. The ‘*save\_data\_to\_list’* function receives *2 parameters* – *file\_name* and *list of rows*, opens the file in *write* mode and uses ‘*for*’ to *write* each row of data from the list table to the file. The ‘*read\_data\_from\_file’* function receives 1 parameter – *file name*, used *for* loop to creates a *row* of product data, *appends* it to the list of rows, and *returns* the updated list. “*Try-Except*” is used to catch a “FileNotFoundError” and create a new file if none exists. (Figure 2)

**Text

Description automatically generated**

***Figure 2: Code added for Processor class***

## **IO (Input/Output) Class**

Next, I added the required functions in the ‘IO’ class for user choice options. Functions were added to print a menu of options for users, get input from user of their menu choice, display current data in the list and get user input for new product. (Figure 3) For example, if the user chooses to add a new product, ‘*input\_new\_product\_data’*  function of the ‘*IO*’ class is called which will run the code to get user input and then return the data in the form of a tuple which is assigned to the variables ‘*name*’ and ‘*price*’. I also added a code to catch any error, if user inputs a number for ‘name’ or does not input a number for ‘price’.

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

***Figure 3: Code added to display current data and input new data under IO Class*Main Body**

In the main body of the script, I used a while loop to run the classes corresponding to each of the user choice. (Figure 4)

Graphical user interface, text, application, email

Description automatically generated

***Figure 4: Code added to main body of script***

After saving the script, I ran the script in PyCharm to test my code. I successfully entered a product name and its price and saved the data in the text file before exiting. (Figure 5)

Text, letter

Description automatically generated

***Figure 5: Testing in PyCharm***

## **Running the Script in OS Command**

Next, I ran the script from the Terminal on my Mac to ensure that it works properly. (Figure 5)

Table

Description automatically generated with low confidence

***Figure 5: Script executed from Terminal on Mac OS.***

## **Summary**

Using the Module 08 instructions, I was able to successfully create and run a script that uses class objects and methods to write blocks of code. Using objects is a great tool to organize and structure code and to achieve simplicity in the main body of the script.

**Link to GitHub repository:** <https://github.com/IBRAR21/IntroToProg-Python-Mod08>