David Grunloh

June 1, 2021

IT FDN 110 A

Assignment 08

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

Working with Classes

# Introduction

In Assignment 08, I altered a Python application which utilized 3 classes to add and view products and their cost. This application is used to gather products and their price from a user and in the end store that data in a file. This assignment was very similar to other assignments in basic functionality, but this was the first to utilize data classes to manage data in the application. This code for this assignment was very different as it is broken into multiple classes with various methods within. This can make the code easier to manage and reuse in the future.

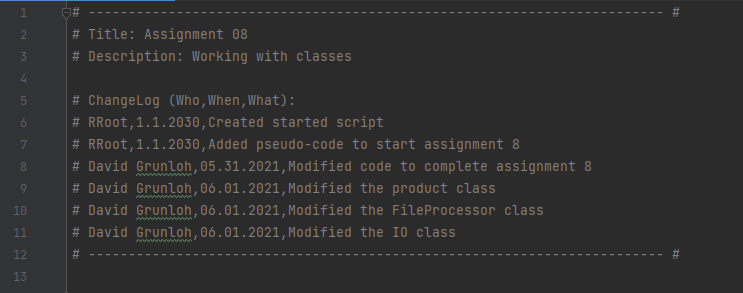
# **Writing the script**

In the script for this assignment there are 4 main sections of code, Product class, FileProcessor class, IO class, and main. These will be described in more detail below as they are used within the application.

## Header

This portion of the script is used to provide information about the purpose of this script to anyone who needs to review or edit the script in the future. The main components that should be included are title, description, author, and change log.

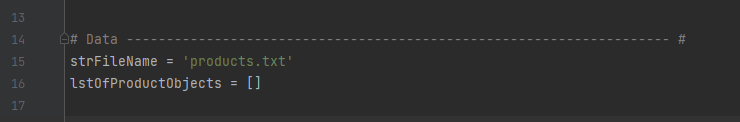
As you can see in Figure 1 below, the application was created by RRoot, but I made several updates and changes to this application. Each of those adjustments were documented in the change log portion of this header.



***Figure 1: This shows the header for the application***

## Define Variables

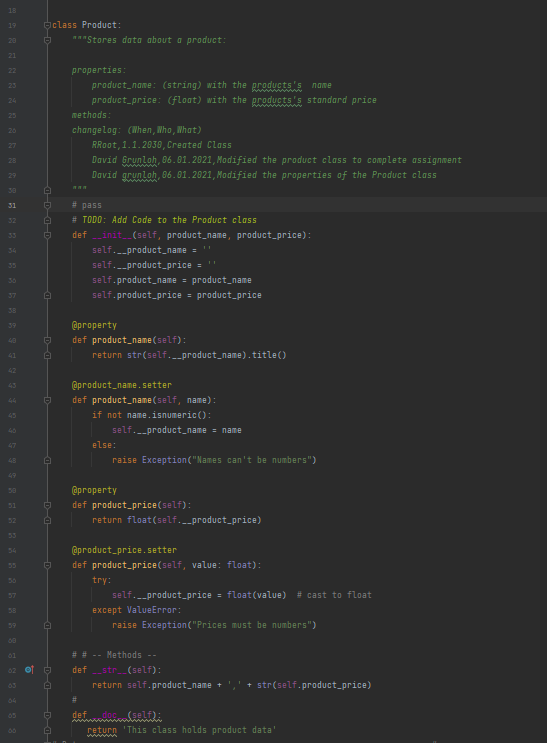
In this section of code, I define some variables that will be used throughout the application. Notice that due to the use of methods and classes, variables that apply to the entire application are not as common.



***Figure 2: This shows the variables for the application being defined.***

## Class Product

This class, which you can see defined in figure 3 below includes the product data management. Within this class you can see that restrictions for the data are built in and exceptions for errors are added. This class is utilized throughout the application when product attributes are utilized.

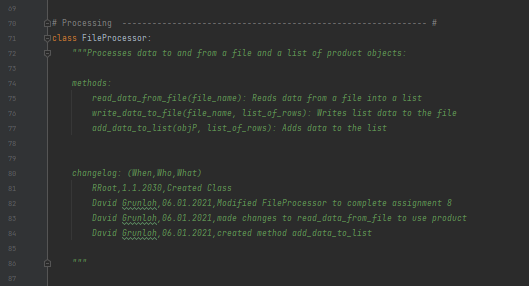


***Figure 3: Product Class***

## Class FileProcessor

This class, which you can see defined in figure 4 below includes the methods that do all the processing for the application. Within this class there are several different methods which complete task including read data from files, add data to list, and write data to file. These functions will be further explained in the description of the main class below as they are called.

* read\_data\_from\_file(file\_name): Reads data from a file into a list
* write\_data\_to\_file(file\_name, list\_of\_rows): Writes list data to the file
* add\_data\_to\_list(objP, list\_of\_rows): Adds data to the list



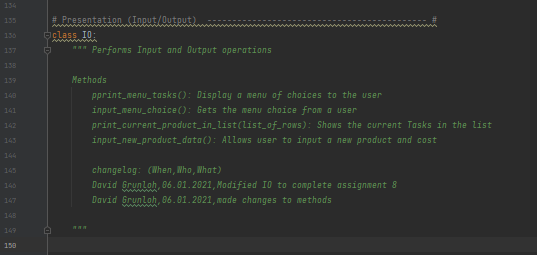
***Figure 4: Class FileProcessor***

## Class IO

The IO Class, which you can see defined in figure 5 below includes the functions that do all the input output operations that interact with the user. Within this class there are several different functions which complete task including:

* Printing the main menu
* Getting the menu choice from the user
* Printing the current product list
* Getting a new product and price from the user

These functions will be further explained in the description of the main class below as they are called.



***Figure 5: IO Class***

## Main

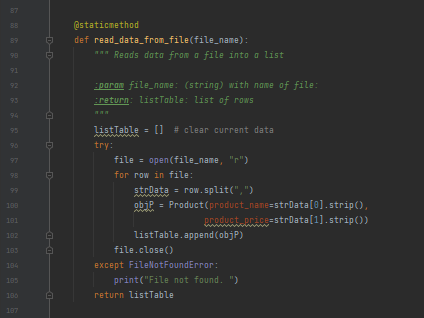
In assignment 08, the main body of code is an initial call to a method to load the list and then you enter a while loop, that calls two IO class functions to get a choice from the user, and then enters into the appropriate if condition based on the input from the user during the initial IO functions. I will walk through each of the if conditions below.

### Load Data from file

When the application starts, data is loaded from the ToDoList.txt file. You can see the method call in figure 6 below.

  
***Figure 6: Initial load of data from file to list***

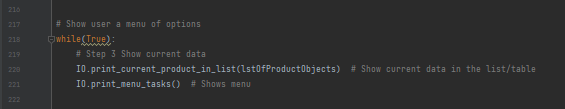
In the code below, you can see the method pulls the data into memory to allow the user to interact and manage that data available to the application. You can see when pulling the data via the method (figure 7), the data is pulled one row at a time adding it to attributes of the product class to format the data as its being pulled into a list for use throughout the application.

******

***Figure 7: read\_data\_from\_file method***

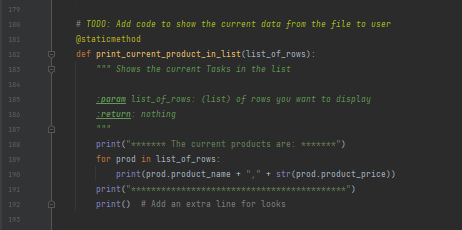
### Loop and Menu Options

In the below section of code shown in Figure 8, you can see the while loop functions calls to print the current task, print menu options and obtain the menu choice from the user. The user will only be able to break out of this condition once they have chosen the correct option, 4, which allows the user to exit the application. As users navigate through the different options, they will always be routed back to this section where the methods explained above allowing the user to use the input function that allows them to select what activity they would like to complete.



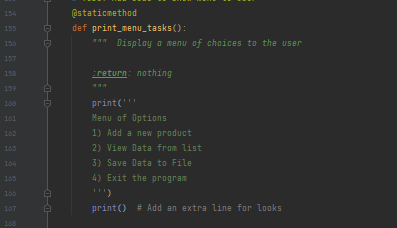
***Figure 8: Loop and Menu call***

Below is figure 9 that shows the method from the IO class that is used to print the current list. This function passes in the current list parameter, list\_of\_rows.

******

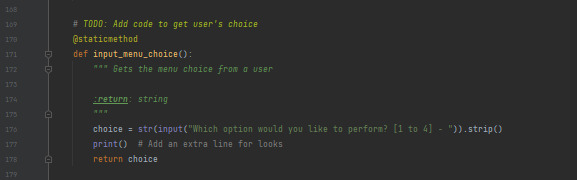
***Figure 9: print\_current\_product\_in\_list method***

Below is figure 10 that shows the method from the IO class that is used to print the menu options. This function is only used to display the list of menu options to the user and does not pass in any parameters nor returns anything to be used later.

******

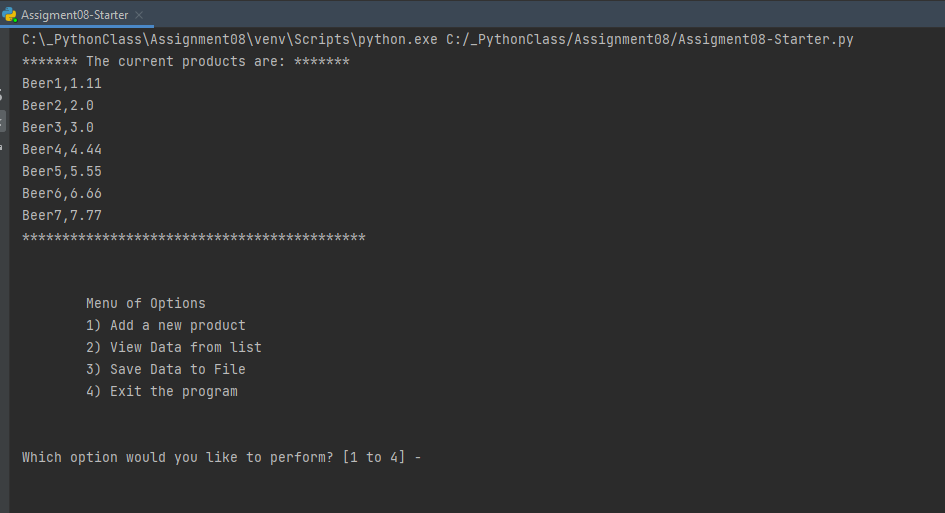
***Figure 10: print\_menu\_tasks method***

Below in figure 11, you can see the input\_menu\_choice function. This function is used to allow the user to determine which section of the application they would like to execute. You can see this function returns the choice to be used in each option if, elif statement.

******

***Figure 11: input\_menu\_choice method***

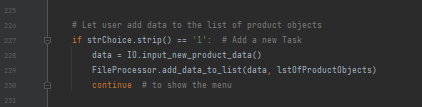
Below in figure 12, you can see this section of code being executed in PyCharm.



***Figure 12: Menu of Options***

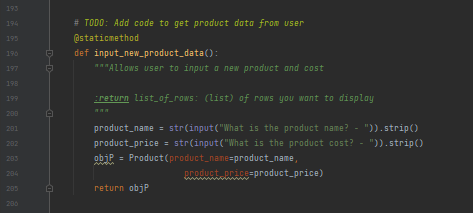
### Option 1 – Gather Input and Store in List

In the below section of code shown in figure 13, which is initiated when the user enters option 1, 2 methods, IO.input\_new\_product\_data and ileProcessor.add\_data\_to\_list are executed. These methods include all the code to gather a product and price from a user and then add that to the current list in memory.

******

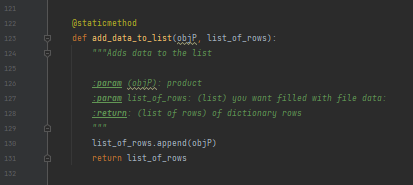
***Figure 13: Option 1***

You can see below in figure 14 the method input\_new\_product\_data uses variables and the input() function to gather data from the user. You can see in this method, we utilize the product class as part of preparing the data to be added to the list. This data is then passed out of this method as return values to be processed in the next method, add\_data\_to\_list.

******

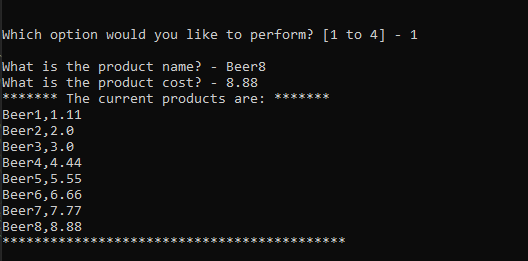
***Figure 14: input\_new\_product\_data***

In figure 15 below, you can see the code for the function add\_data\_to\_list, which uses the return values from the previous function to add the data to the list stored in memory. In addition to the parameters that are passed in, it also returns an updated list of rows to be used throughout the application.

******

***Figure 15: add\_data\_to\_list***

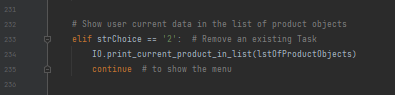
Below in figure 16, you can see this section of code being executed in a command prompt.



***Figure 16: executing option 1***

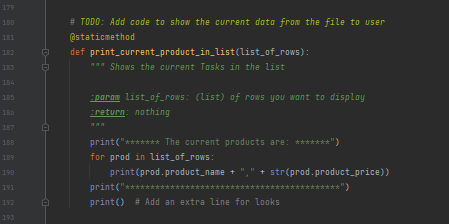
### Option 2 – View Data in list

In this section shown in figure 17, the user has chosen to view data in the list. Within this section, I allow the user to choose which item they would like to remove by entering the Task name. As long as this task name matches a dictionary row Task Key. If there is a match, the code removes that line and returns the user to the main menu. This is completed through the use of the 2 functions below IO.input\_task\_to\_remove and Processsor.remove\_data\_from\_list.

******

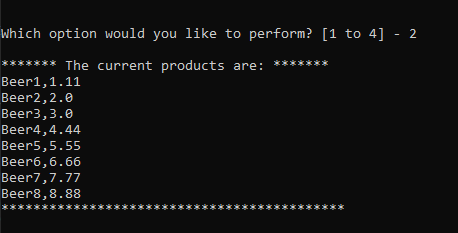
***Figure 17: Option 2***

Below in figure 18, you can see the method print\_current\_product\_in\_list. This code displays all the current data in memory to the end user. This would include all data originally loaded from the file as well as items that have been added since starting the application.

******

***Figure 18: print\_current\_product\_in\_list method***

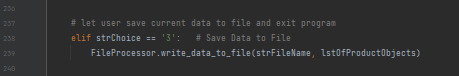
Below in figure 19, you can see this section of code being executed in a command prompt.

******

***Figure 19: Command line output of option 2***

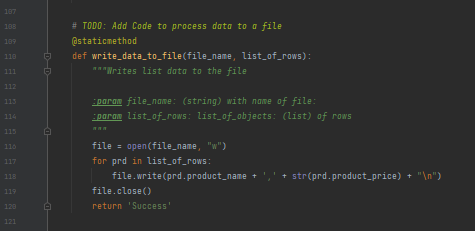
### Option 3 – Save Data to File

In this section shown in figure 20, the user has chosen to save the data to the file. Within this section, the application will write all lines from the list that is stored in memory to the text file that is specified in the variables section. This code will overwrite the text file each time it is saved.

******

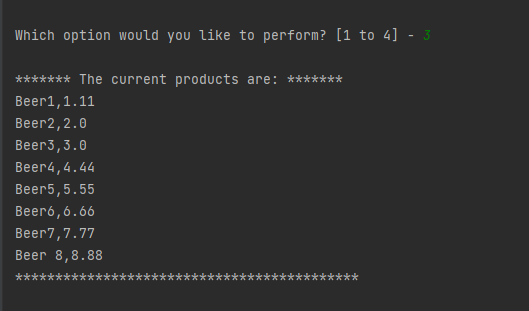
***Figure 20: Option 3 – Save Data to file***

In the below method, you can see that the data from the list, is written to the file to be used later.

******

***Figure 21: write\_data\_to\_file method***

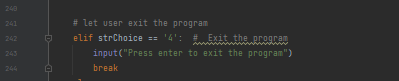
Below in figure 22, you can see this section of code being executed in PyCharm.

******

***Figure 22: save data to file in PyCharm***

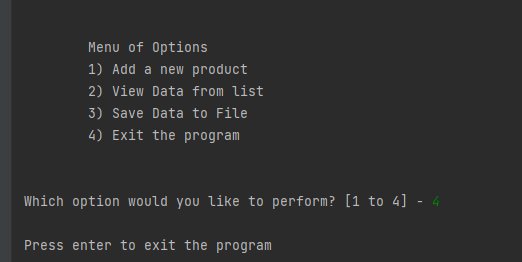
### Option 4 – Exit the program

In this section of code shown in Figure 23, the user receives a prompt asking them to hit enter to exit the program. This is completed via the input function. Upon hitting enter, the user will exit the program due to the break statement.



***Figure 23 – Exit the program***

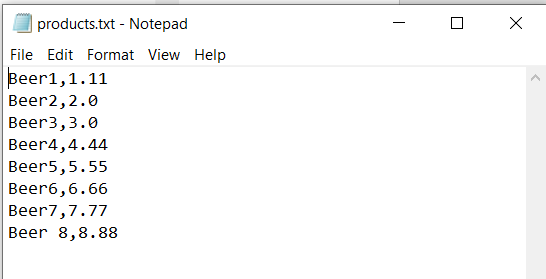
You can see option 4 in action below in Figure 24. This shows the user selecting option 4 and exiting the program using PyCharm.

******

***Figure 24***

## Output Results

Below in figure 25, you can see the output that was written to our data file, product.txt.



***Figure 25***

# Summary

In this assignment, I utilized python classes, methods, in addition to variables, and list, to gather input from a user and process it into a list. Users have the ability to loop through options multiple times to add (option 1) view the data (option 2) items to the list. Then option 3, allows the user to save the data to a text file, while option 4 allows the user exit the program. This code was also formatted and commented to ensure that it could be reused and is editable in the future.