Ryan Fear

12/06/2021

IT FDN 110 A

Assignment 08

https://github.com/RFear/IntroToProg-Python-Mod08

Classes and Objects

# Introduction

This document will discuss how to recursively display an option menu for the user to navigate. The user will be able to repeatedly show the current data, add new data, and save data to a text file. The programming example in this document introduces a new concept of creating our own class objects. This concept is fundamental to python programming and will allow the programmer to vastly expand their capabilities.

# Problem Statement

The goal of this program is to load data, work with data, and save the data to a file. The data to handle will be class objects. The program must first read in data from the text file ‘products.txt’. Then the program must recursively print a menu of options to the user. From the displayed menu, the user selects an option then the program performs the selected operation. The user can select from the following three options. One, show the current data. Two, add product and price data as objects. Three, save the data to a file which also exits the program. The menu system is recursively printed to the screen after each menu option execution. Only when option three is selected will the program stop executing.

# Data

The first concern addressed in this code is data. The code has a title block, declares two global variables, and defines the product class. See Figure 1 for the first two items and see Figure 2 for the product class. The title block is created for code documentation and serves as a reference to future programmers. On line 12, the file name that will be used by the program is defined. Note the file location is a relative reference and is stored at the same location on the computer as the python script (.py) file. An empty list that will hold the product name and price objects is made on line 13.

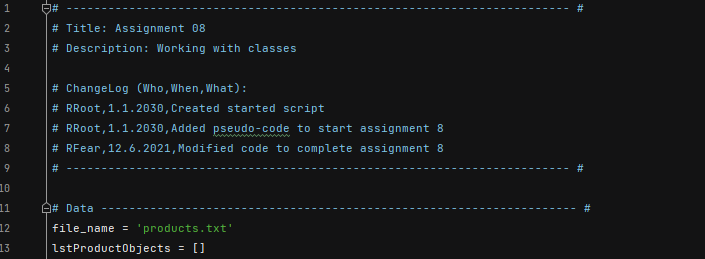


Figure : Title Block and Constants

The Product class is shown in Figure 2, note the document string on line 17 is hidden for clarity. First, the two attributes, *product\_name* and *product\_price*, are defined on line 33 and 34. Next the properties are defined for each object in lines 37-58. In this code I tried to have the exceptions on line 46 and 57 raised inside of a while loop for any incorrect input, however I was unsuccessful. So I created a workaround that is described in the class Processor. Finally, two methods that return the *product\_name* as a string and the *product\_price* as a float.

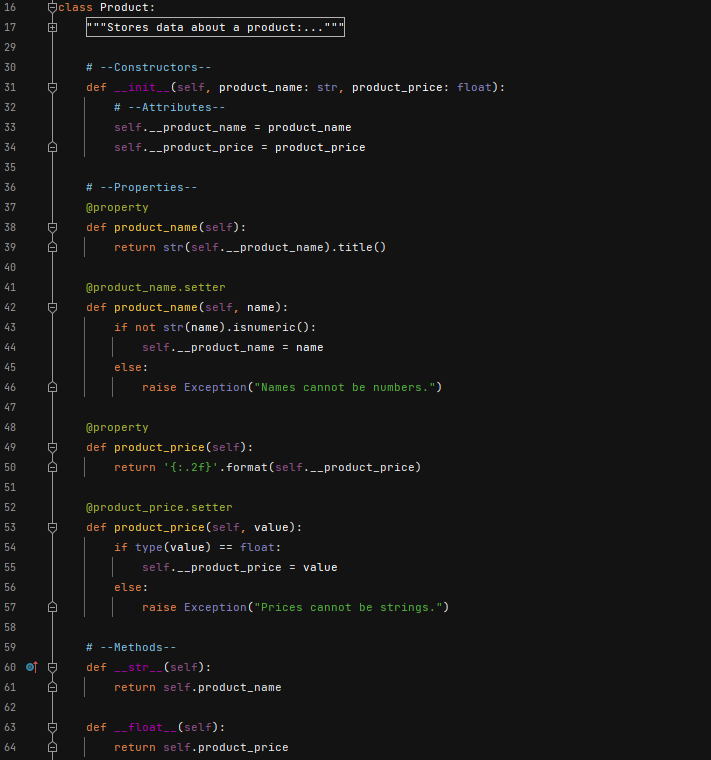


Figure : Product Class

# Processing

The next concern address is processing. The processing portion of code is contained in a class called Processor. All the methods in this class are defined with the @staticmethod decorator which means you do not have to make an object. There are three methods defined in this class and they are discussed below in their respective sections.

## Read Data

The code to read data from a file is shown in Figure 3. The method is defined on line 82 and takes in two parameters: the file name and the list of objects. Note, the list of objects is a list in which each element is a class object. The method begins with a doc string, inside three quotations. The remainder of the method is within a try, except, and else block beginning at line 89. First the block tries to open the file name, if it exist. If it does exist, the loop on line 91 executes and the data is read in as class objects and stored to the local variable *list\_of\_obj*. If a FileNotFoundError occurs, the user is informed by a message printed to the screen. The code will then make a blank file and notify the user a file was created. The else block executes if the try block was successful, notifies the user, and closes the file.

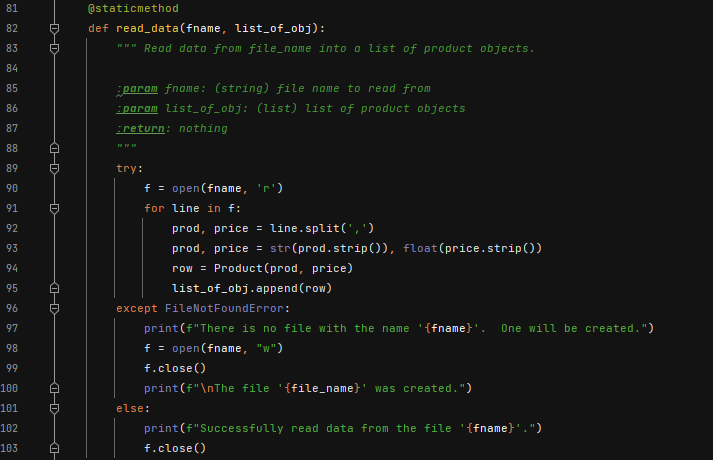


Figure : Read Data Method

## Save Data

The code to save data to a file is shown in Figure 4. The method is defined on line 106 and takes in two parameters: the file name and the list of objects. First the file is opened in line 113, then the data is written to the text file in the for loop beginning on line 114. The data is separated by a comma.

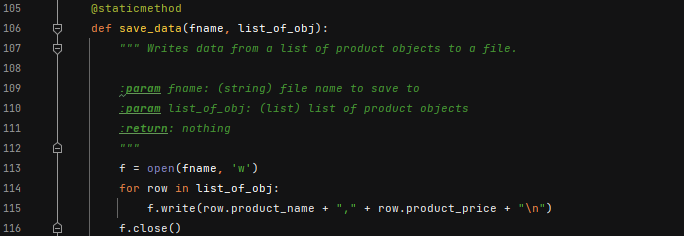


Figure : Save Data Method

## Add Data

The code to add objects to a list is shown in Figure 5. The method is defined on line 119 and takes in the one parameter: the list to add the data to. Three variables are defined to make sure the data entered make sense. This is also where I would have liked to work on more error handling with the exceptions raised in the property setter/getter portion of the Product class. First a while loop is started and the code ask the user to insert a product name. It then repeats their input and ask if it is correct. If it is not, the user is prompted to enter another product name. Once the input is acceptable to the user, the code then ask the user for the products price. Another try/except block is used here to check if a float type variable exist or not. If it does not, the code will prompt the user to enter numbers for a price. If the price is a float, then a new object is created in line 142. This data is then appended to the variable *list\_of\_obj*.

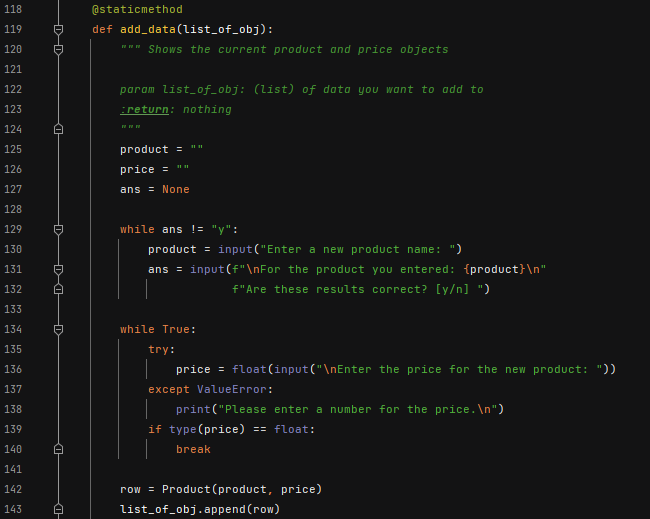


Figure : Add Data Method

# Input/Output

The last concern addressed is the presentation or input/output concern. The input/output portion of code is contained in a class called IO. There are four methods defined in this class and they are discussed below in their respective sections.

## Print Menu

The code to print the menu shown in Figure 6. The method takes in no parameters. This method simply prints out the menu options to the screen.

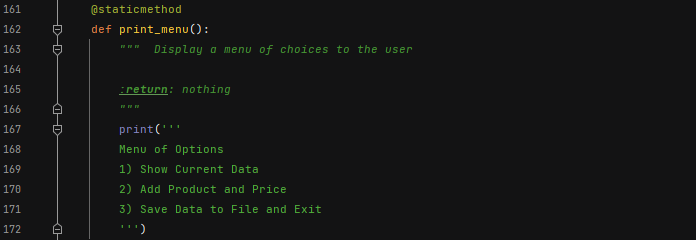


Figure : Print Menu Method

## Input Menu Choice

The code to print the menu shown in Figure 7. The method takes in no parameters. This method ask for the user for the menu option they would like to perform. The answer (*ans*) is returned to be used by the main script.

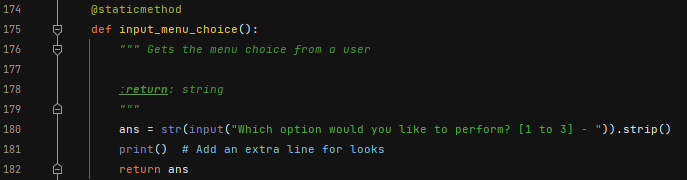


Figure : Input Menu Choice Method

## Input Yes No Choice

The code to return a user yes or no selection is shown in Figure 8. The method takes one parameter: a message. The message is a string that is a yes or no question which will take the input of “y” or “n” representing (y)es or (n)o. The input is returned on the same line 190.

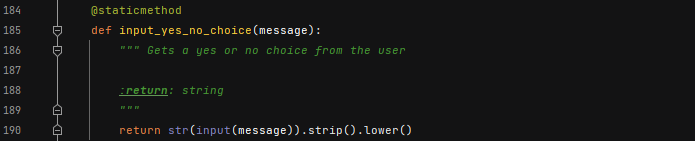


Figure : Input Yes/No Choice

## Print Current Data

The code to print the menu shown in Figure 9. The method takes one parameter: the list of objects. The method also adds some formatting to the screen. This method also displays data contained in the list.

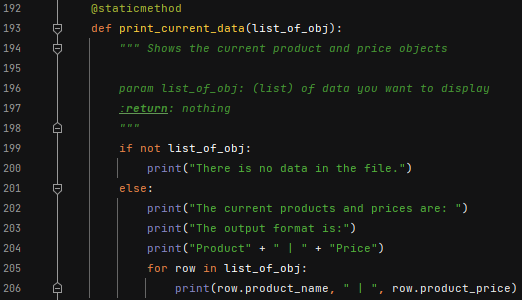


Figure : Print Current Data Method

# Main Body of Script

With all classes and methods defined, the main part of the script executes. The main script will refer to the classes and methods depending on the user menu selection. The main script consist of loading and displaying the data as well as all the execution associated with a menu selection. Each of these sections are discussed in the following sections.

## Load and Display Menu

The code to load data from a file and display the menu system are shown in Figure 10. When the program starts data is first read from the text file in line 218. This is completed by using the method read\_data in the Processor class. The recursive while loop is then started on line 220. This will allow repetitive printing out the menu of options. The user option selection is captured on line 222.

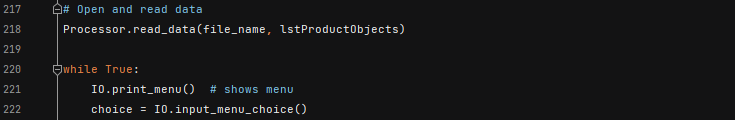


Figure : Load and Display Menu

## Menu Option 1 – Show Current Data

When menu option 1 is selected the code in Figure 11 executes. The data currently in the file is printed to the screen using the method print\_current\_data from the IO class on line 225.



Figure : Menu Option 1

## Menu Option 2 – Add Product and Price

When menu option 2 is selected the code in Figure 12 executes. This calls the add\_data method from the Processor class.



Figure : Menu Option 2

## Menu Option 3 – Save Data to File and Exit

When menu option 3 is selected the code in Figure 13 executes. The user is asked if they would like to save their data. The message will continually repeat until either “y” or “n” is entered. If “y” is selected then the file is saved and the program exits. If “n” is selected the file is not saved and the program exits.

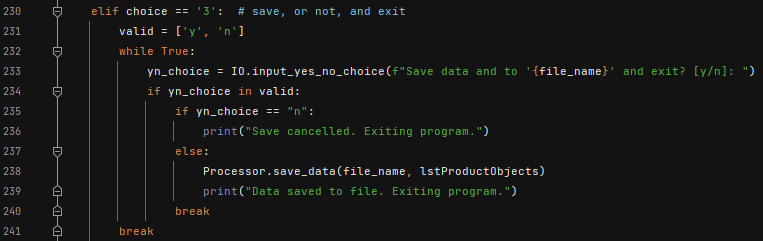


Figure : Menu Option 3

## Invalid Option

If at any time the user doesn’t select menu option 1, 2, or 3, the code in Figure 14 executes. The user is then prompted to enter a valid option and the menu of options is reprinted to the screen.

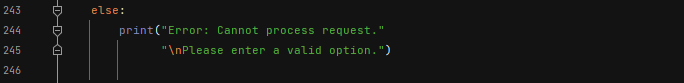


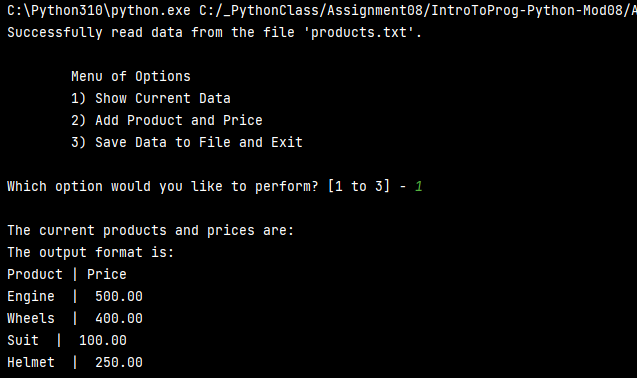
Figure : Invalid Menu Option

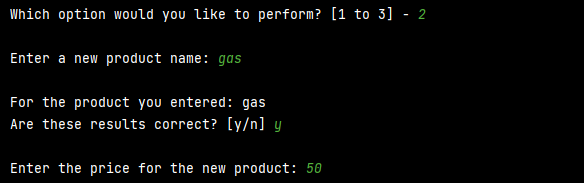
# Code Testing

The code was tested both in the PyCharm environment and the Command Window (CMD). See the respective sections for example of code execution.

## PyCharm

The PyCharm execution of ‘Assignment08.py’ is shown in Figure 15. The text file ‘products.txt’ before and after code execution are shown in Figure 16. Note the data read into and out of the program are saved in products.txt, I renamed the files products\_before.txt and products\_afterPyCharm.txt for comparison purposes only.





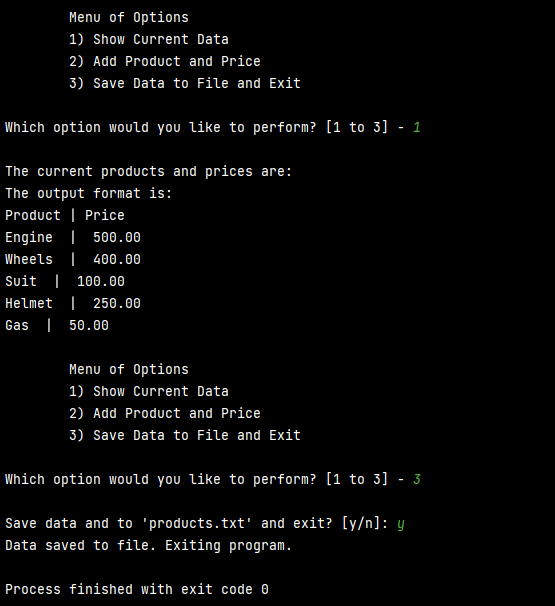


Figure : PyCharm Code Execution

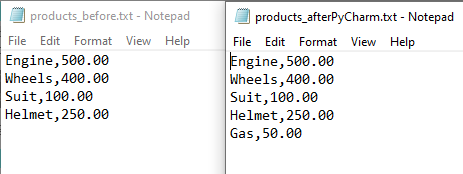
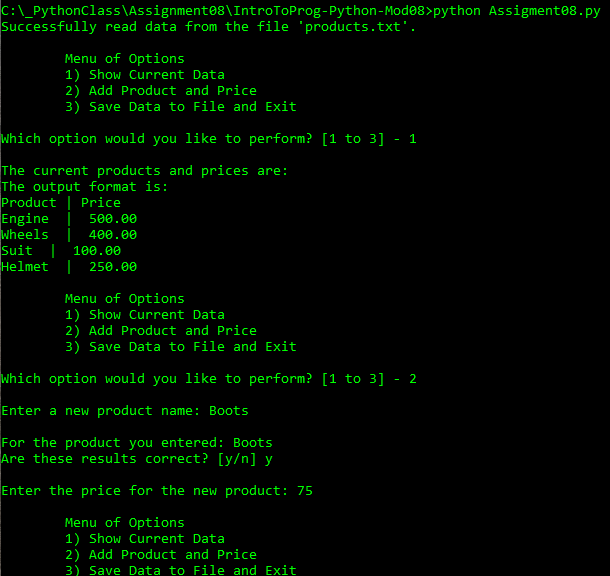


Figure : products.txt After PyCharm

## Command Window

The Command Window execution of ‘Assignment08.py’ is shown in Figure 17. The text file ‘products.txt’ before and after code execution are shown in Figure 18. Note the data read into and out of the program are saved in products.txt, I renamed the files products\_before.txt and products\_afterCMD.txt for comparison purposes only.



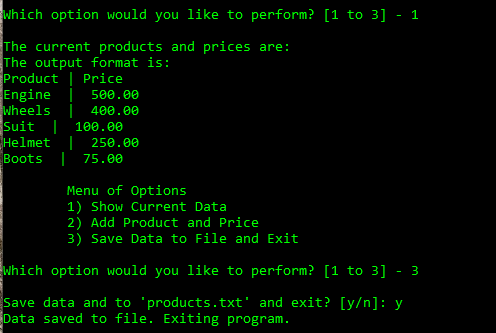


Figure : Command Window Code Execution

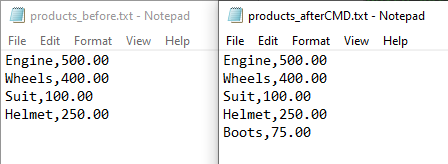


Figure : products.txt After Command Window

# Summary

In this document an example of how to recursively display an option menu for the user to navigate was discussed. First the problem statement of the example was discussed. The remainder of the document explained each of the concerns addressed. The creation of a class and two objects were introduced in this example as well. Finally code execution examples within PyCharm and the Windows OS command window were given. With the addition of the skills gained from this exercise a newer python programmer can gain greater control over their code flow and make more complex programs.