Rufus Ayeni
November 28, 2020
IT FDN 110 A
Assignment 08
https://github.com/rayeni/IntroToProg-Python-Mod08

Working with Classes

Introduction:

In Assignment 08, I modified a script that uses classes. The modification entailed writing Python code based on the existing pseudocode in the script. A new concept was introduced in this assignment-- creating classes from which objects can be created (instantiated). Attributes and methods of the object are accessed using dot notation (object.attribute or object.method). The constructor method was introduced as an initializer of an object's attributes at the time the object is created. The remainder of the assignment reinforced concepts from earlier assignments-- segmentation of the script using IO and processing sections, and the creation and use of functions to perform tasks.

Requirements:

1. Write a class from which objects can be created.

2. Write functions to perform processing and presentation tasks.

3. The processing tasks are:

    a. Read data from a file
    b. Save data to a file
    c. Add object to a list

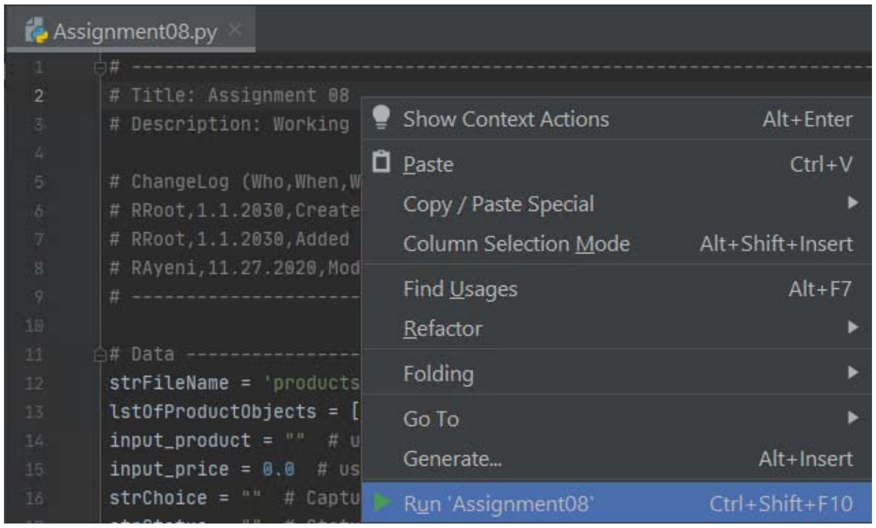
4. The presentation tasks are:

    a. Print menu options
    b. Receive menu option from user
    c. Receive input (y or no)
    d. Get current data from list
    e. Input product data
    f. Receive input from user to continue

5. Add `if` code blocks to correspond to user's selection.

Execution of Script:

1. To execute the script in PyCharm, simply right-click the code area and select “**Run ‘Assignment08.’**”

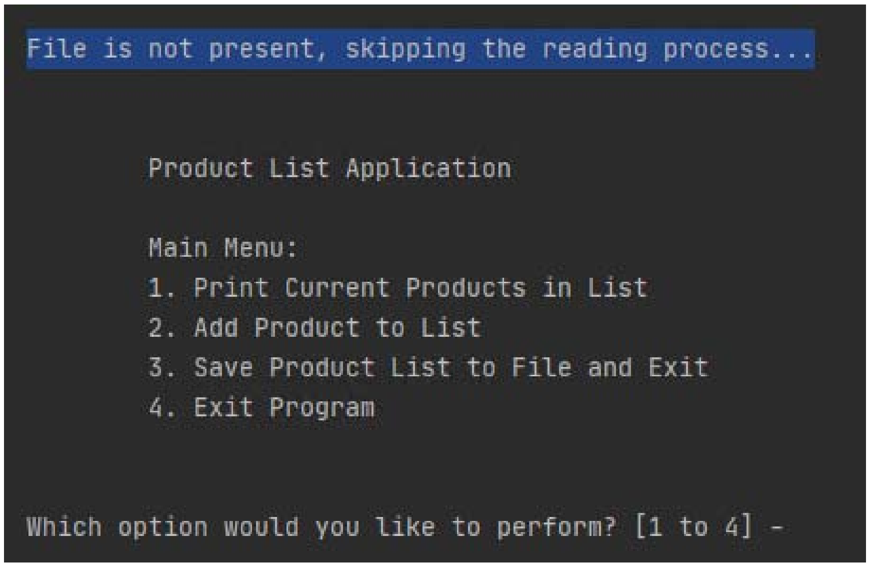


```
# -----------------------------------------------------------------------
# Title: Assignment 08
# Description: Working

# ChangeLog (Who,When,W
# RRoot,1.1.2030,Create
# RRoot,1.1.2030,Added
# RAyeni,11.27.2020,Mod
# --------------------

# Data ---------------
strFileName = 'products
lstOfProductObjects = [
input_product = ""  # u
input_price = 0.0  # us
strChoice = ""  # Captu
```

Figure 1

Exception Generation and Menu Presentation:

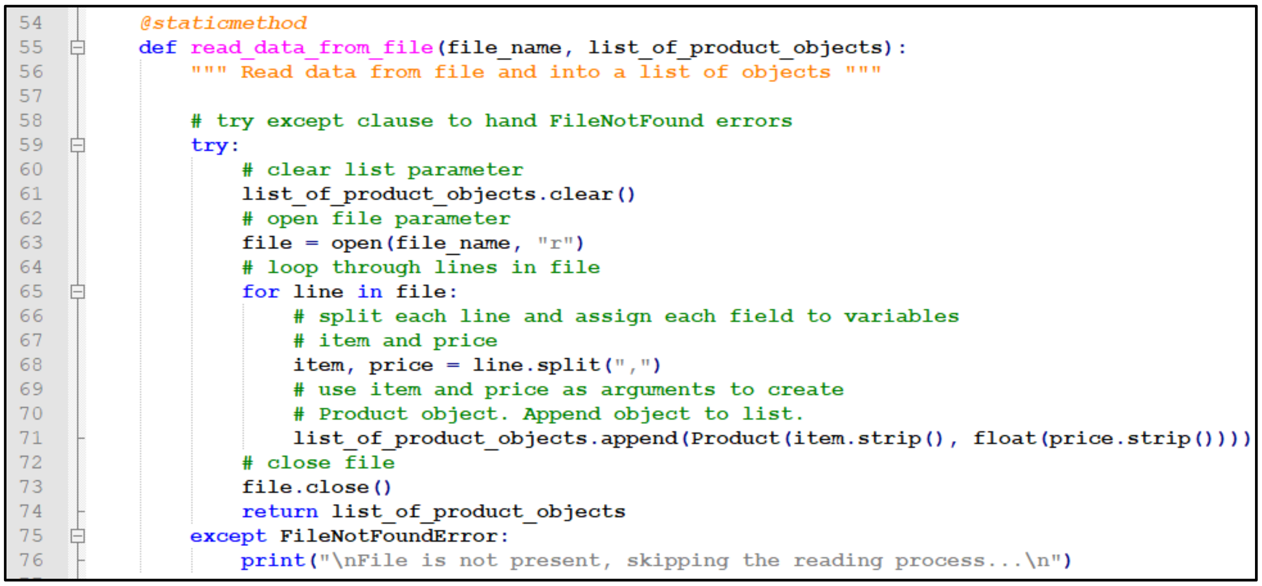
2. Running the script opens a terminal window through which a user can interface with the application, through a menu of options. The script begins by informing the user that the file is missing-- “`File is not present, skipping the reading process...`”



```
File is not present, skipping the reading process...


        Product List Application

        Main Menu:
        1. Print Current Products in List
        2. Add Product to List
        3. Save Product List to File and Exit
        4. Exit Program


Which option would you like to perform? [1 to 4] -
```

Figure 2

The message is presented through the use of exception handling. The code for exception handling can be viewed in **Figure 3**:

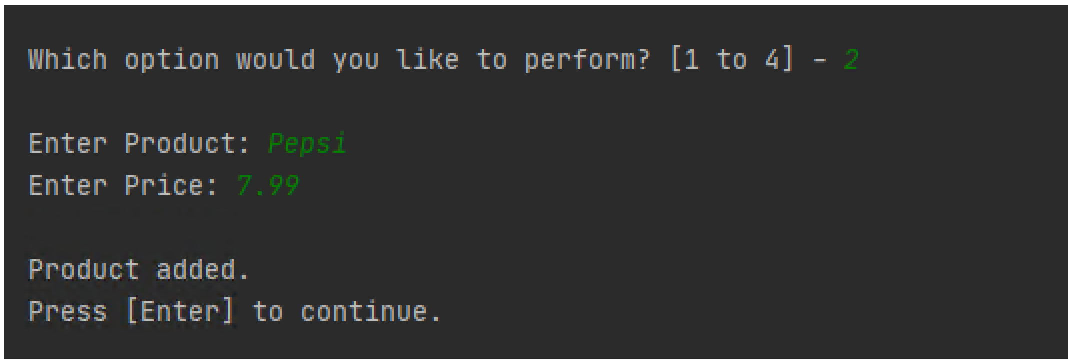


```python
    @staticmethod
    def read_data_from_file(file_name, list_of_product_objects):
        """ Read data from file and into a list of objects """

        # try except clause to hand FileNotFound errors
        try:
            # clear list parameter
            list_of_product_objects.clear()
            # open file parameter
            file = open(file_name, "r")
            # loop through lines in file
            for line in file:
                # split each line and assign each field to variables
                # item and price
                item, price = line.split(",")
                # use item and price as arguments to create
                # Product object. Append object to list.
                list_of_product_objects.append(Product(item.strip(), float(price.strip())))
            # close file
            file.close()
            return list_of_product_objects
        except FileNotFoundError:
            print("\nFile is not present, skipping the reading process...\n")
```

Figure 3

Option #2, Add Product to List:

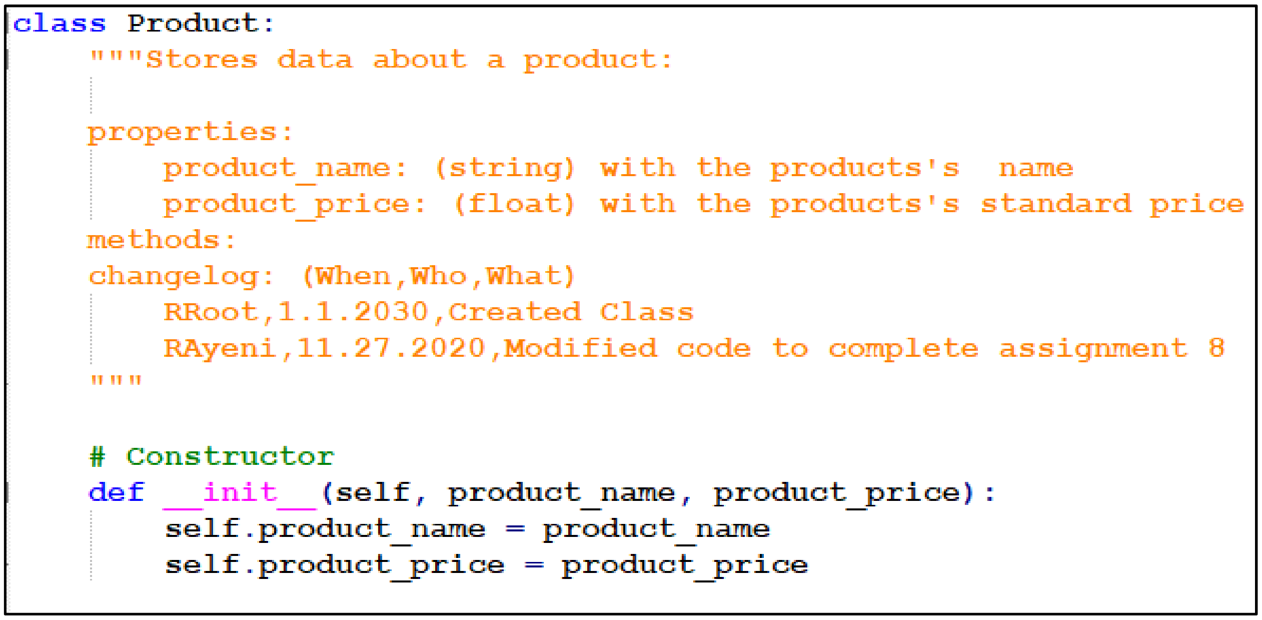
3. Since there are no product objects in the list, let's choose **Option #2**, *Add Product to List*. When Option #2 is selected, the user is prompted for the product's name and price.



```
Which option would you like to perform? [1 to 4] - 2

Enter Product: Pepsi
Enter Price: 7.99

Product added.
Press [Enter] to continue.
```

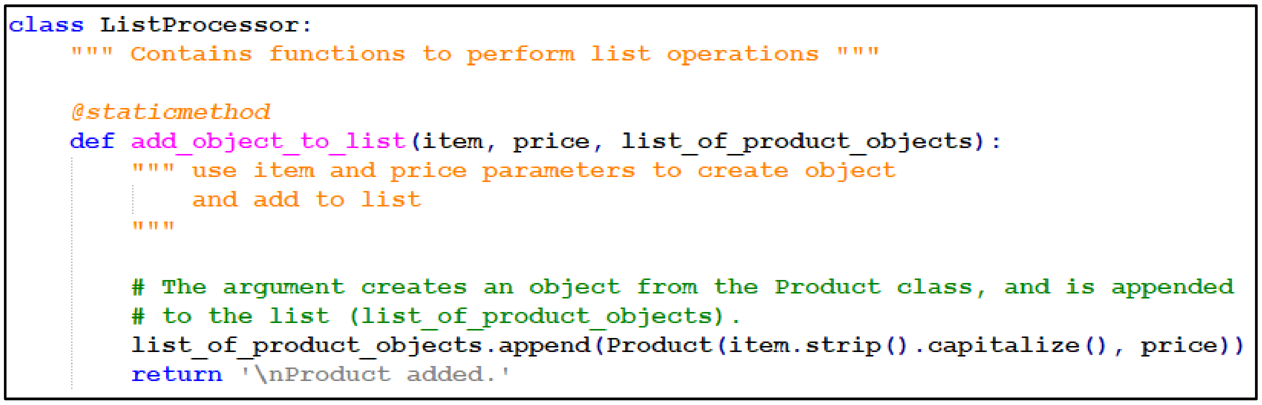
Figure 4

After the user inputs the product and prices, the script uses the data to create an object from the `Product` class. The code for the `Product` class, and the function, `def add_object_to_list()`, that creates the object and stores it in a list is shown in **Figures 5 and 6**.



```
class Product:
    """Stores data about a product:

    properties:
        product_name: (string) with the products's  name
        product_price: (float) with the products's standard price
    methods:
    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
        RAyeni,11.27.2020,Modified code to complete assignment 8
    """

    # Constructor
    def __init__(self, product_name, product_price):
        self.product_name = product_name
        self.product_price = product_price
```

Figure 5: Product class

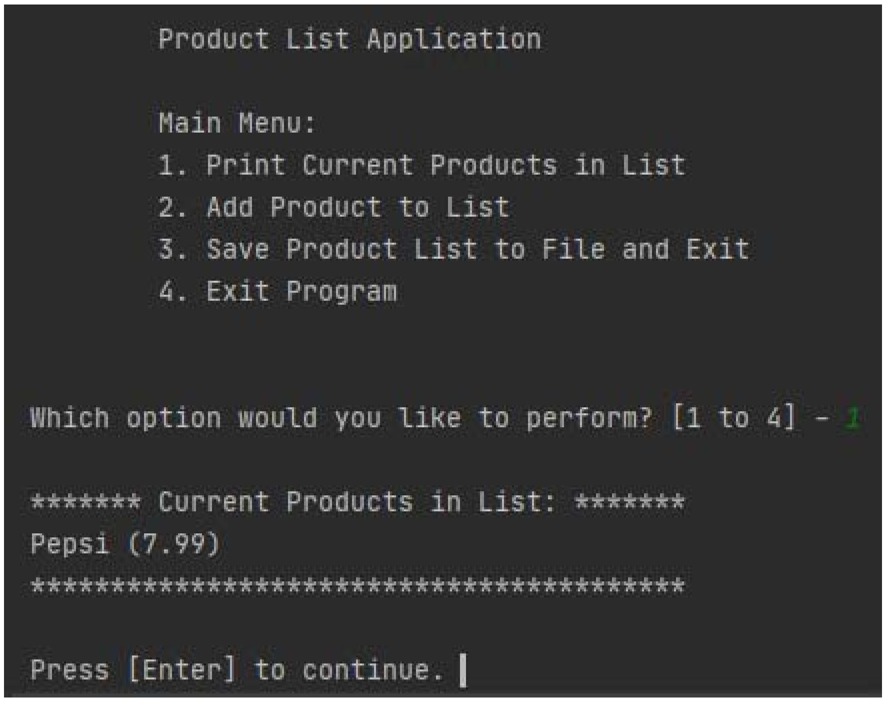


```
class ListProcessor:
    """ Contains functions to perform list operations """

    @staticmethod
    def add_object_to_list(item, price, list_of_product_objects):
        """ use item and price parameters to create object
            and add to list
        """

        # The argument creates an object from the Product class, and is appended
        # to the list (list_of_product_objects).
        list_of_product_objects.append(Product(item.strip().capitalize(), price))
        return '\nProduct added.'
```

Figure 6: Product object created and appended to list.

Option #1, Print Current Products in List:

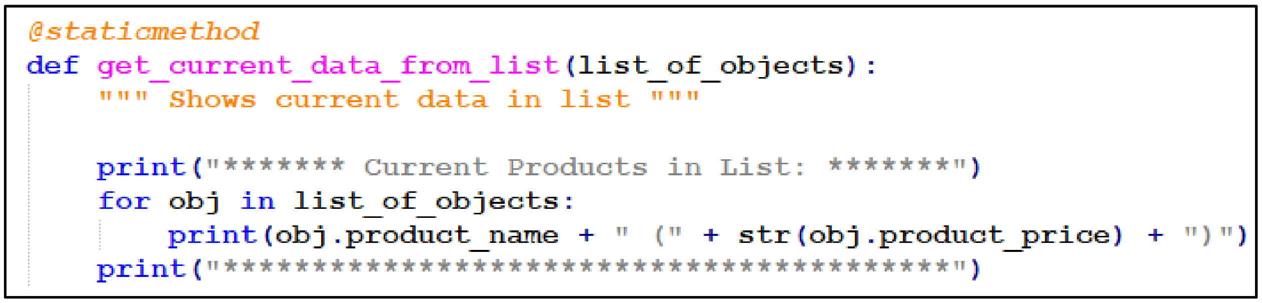
4. Since we now have a `Product` object in our list, we can select **Option #1**, *Print Current Products in List*, to show the current object in the list:



```
        Product List Application

        Main Menu:
        1. Print Current Products in List
        2. Add Product to List
        3. Save Product List to File and Exit
        4. Exit Program


Which option would you like to perform? [1 to 4] - 1

******* Current Products in List: *******
Pepsi (7.99)
*****************************************

Press [Enter] to continue.
```

Figure 7: One product is in the list

The code to print the current product objects can be found the `def get_current_data_from_list` function, in the IO class, as shown in **Figure 8**.

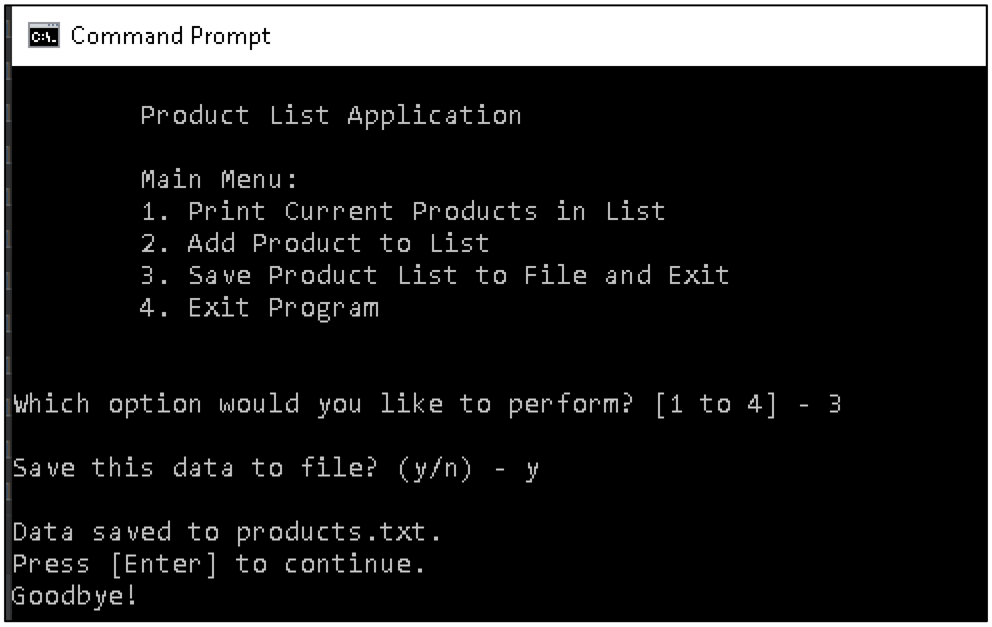


```
@staticmethod
def get_current_data_from_list(list_of_objects):
    """ Shows current data in list """

    print("******* Current Products in List: *******")
    for obj in list_of_objects:
        print(obj.product_name + " (" + str(obj.product_price) + ")")
    print("*****************************************")
```

Figure 8

Option #3, Save Product List to File and Exit:

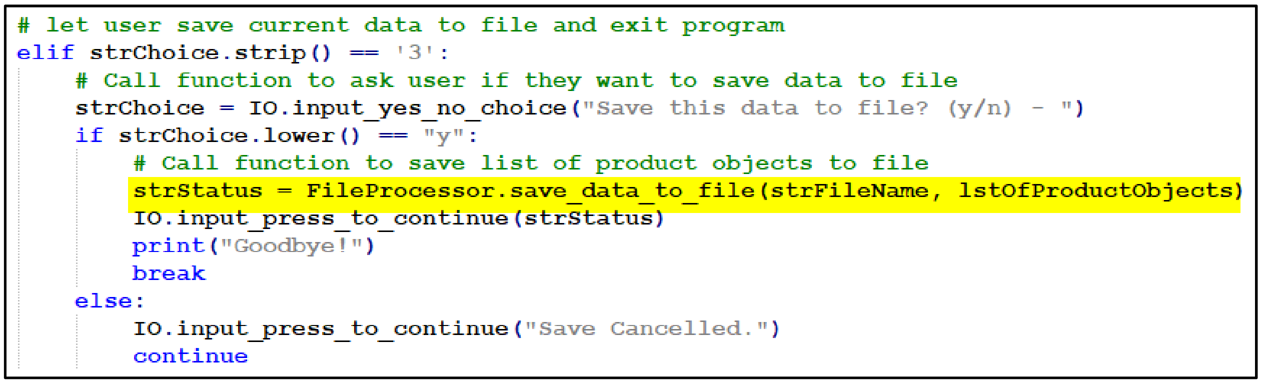
5. To save the list of product objects to a file, the user selects **Option #3**, *Save Product List to File and Exit*. The script asks the user if he/she wants to save the file. When the user selects `y`, the script notifies the user that the data was saved to `products.txt` and ends the program, as seen in **Figure 9**:



```
Command Prompt

        Product List Application

        Main Menu:
        1. Print Current Products in List
        2. Add Product to List
        3. Save Product List to File and Exit
        4. Exit Program


Which option would you like to perform? [1 to 4] - 3

Save this data to file? (y/n) - y

Data saved to products.txt.
Press [Enter] to continue.
Goodbye!
```

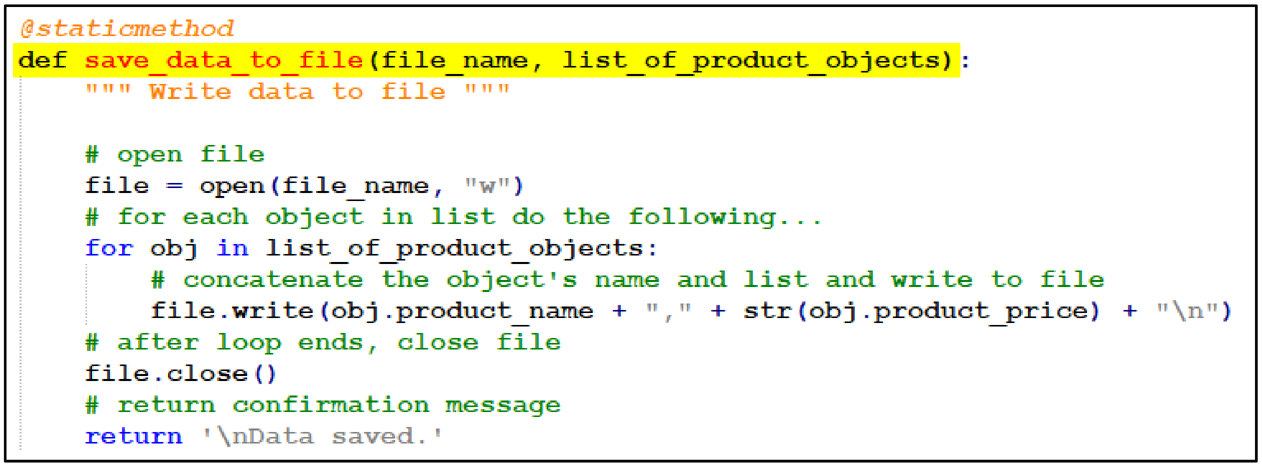
Figure 9

6. The code to save the product list to a file and exit can be viewed in **Figures 10 and 11**:



```
# let user save current data to file and exit program
elif strChoice.strip() == '3':
    # Call function to ask user if they want to save data to file
    strChoice = IO.input_yes_no_choice("Save this data to file? (y/n) - ")
    if strChoice.lower() == "y":
        # Call function to save list of product objects to file
        strStatus = FileProcessor.save_data_to_file(strFileName, lstOfProductObjects)
        IO.input_press_to_continue(strStatus)
        print("Goodbye!")
        break
    else:
        IO.input_press_to_continue("Save Cancelled.")
        continue
```

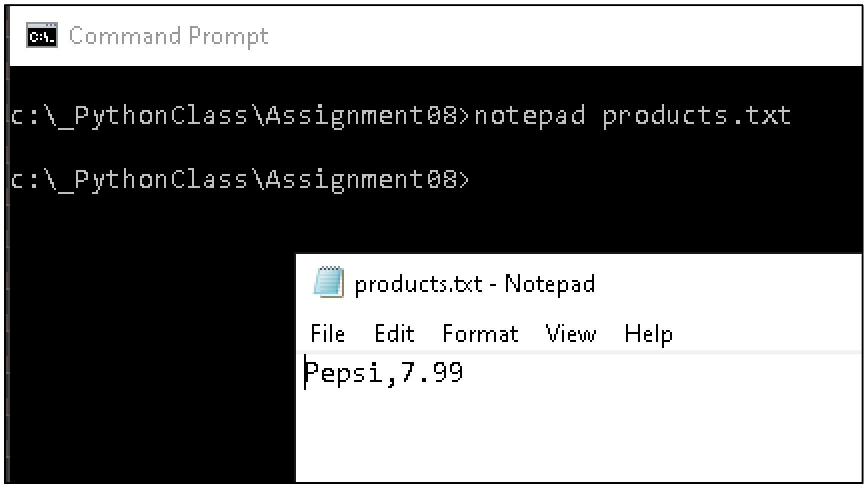
Figure 10: elif to handle menu selection



```
@staticmethod
def save_data_to_file(file_name, list_of_product_objects):
    """ Write data to file """

    # open file
    file = open(file_name, "w")
    # for each object in list do the following...
    for obj in list_of_product_objects:
        # concatenate the object's name and list and write to file
        file.write(obj.product_name + "," + str(obj.product_price) + "\n")
    # after loop ends, close file
    file.close()
    # return confirmation message
    return '\nData saved.'
```

Figure 12: Function called to save product list to file

7. After the script exits, the user can check the contents of `products.txt` to confirm that the script saved the data to the file:



```
c:\_PythonClass\Assignment08>notepad products.txt

c:\_PythonClass\Assignment08>
```

```
Pepsi,7.99
```

Figure 13

Summary:

This assignment focused on the use of classes and creating objects from them. The object that was created in this assignment was simple-- two attributes, one constructor, and no additional methods. Object attributes were accessed using dot notation (object.attribute) to write them to a file.