Dmitriy Alexandrov

November 27th, 2019

IT FDN 100 A

Assignment08

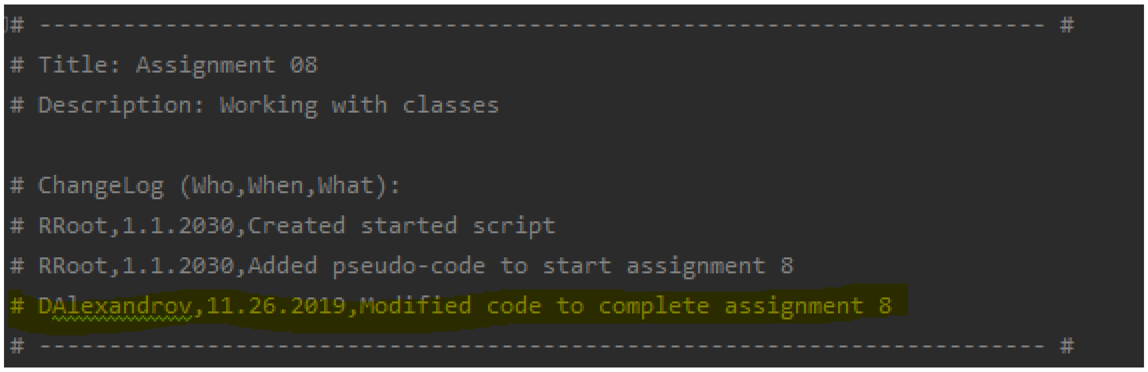
# Coding Product Management Script using Classes and Error Handling

## Introduction

In this paper I am going to go through the steps that I took in adding code to a starter script in order to create a fully functioning Product Management script with a user menu, utilizing classes and error handling.

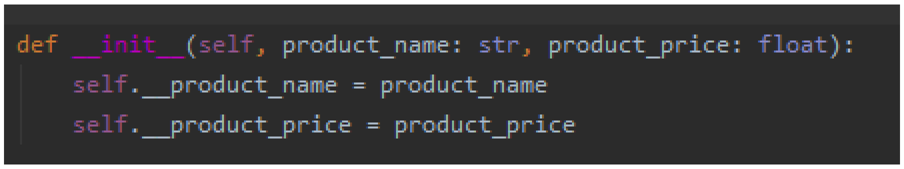
## Steps Taken in Adding Code to the Starter Script:

I began with creating a new project in PyCharm called Assignment08 and importing the Assignment08_Starter.py from the course module. After looking through the script, I first began with updating the header at the top of the script with my information in the Change log section.



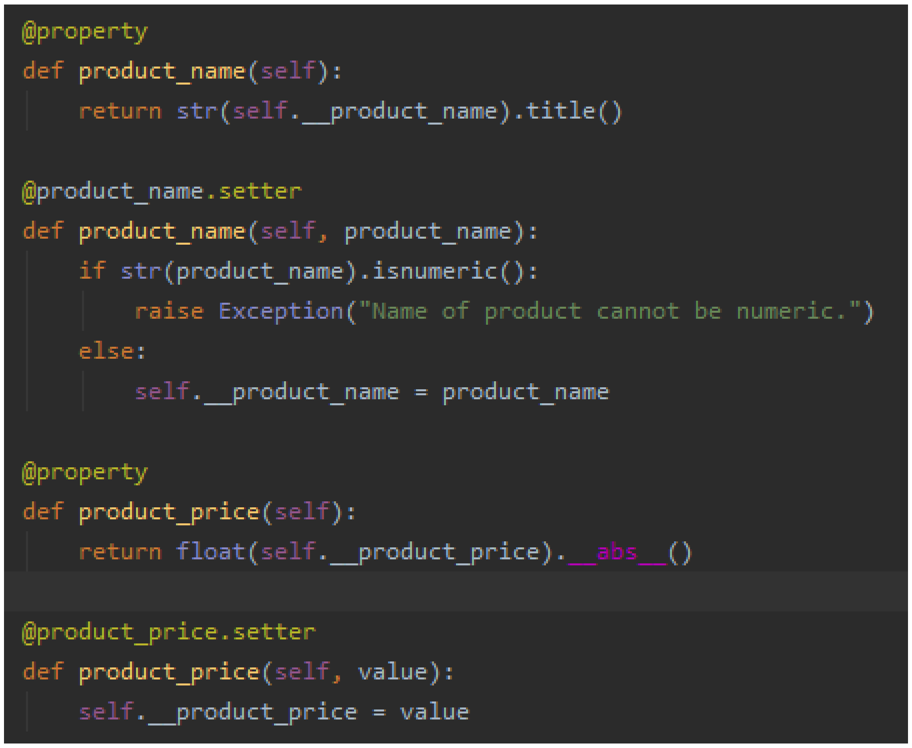
```
# ------------------------------------------------------------------------ #
# Title: Assignment 08
# Description: Working with classes

# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# RRoot,1.1.2030,Added pseudo-code to start assignment 8
# DAlexandrov,11.26.2019,Modified code to complete assignment 8
# ------------------------------------------------------------------------ #
```

After that, I began with the product class. I saw that the class doc string already had 2 properties in there (product_name that is a string, and product_price which is a float value). So I started off with creating the Product class constructor which would take in the self, product_name variable (with the preferred str), product_price (with the preferred float) as shown below.



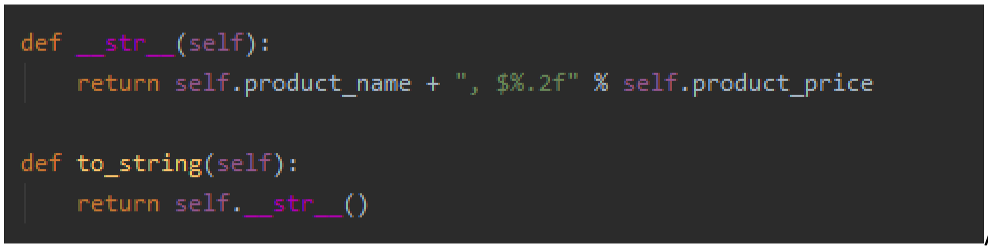
```python
def __init__(self, product_name: str, product_price: float):
    self.__product_name = product_name
    self.__product_price = product_price
```

After the creation of the constructor with its two private variables of self.__product_name and self.__product_price which take in the input of the values passed to the Product class. I created the getter and setter methods for each property as seen below. Also the setter for the product_name method included an error handle where if a numeric value was passed for the product name then an exception would be raised to let the user know that a product cannot have a numeric value.



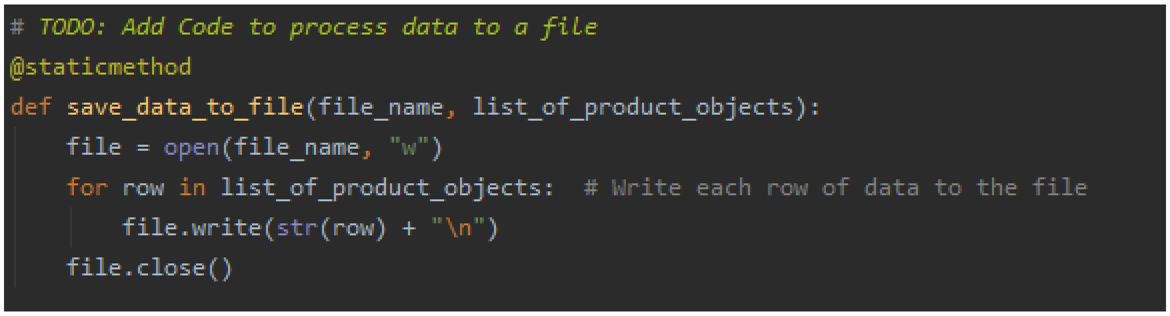
```python
@property
def product_name(self):
    return str(self.__product_name).title()

@product_name.setter
def product_name(self, product_name):
    if str(product_name).isnumeric():
        raise Exception("Name of product cannot be numeric.")
    else:
        self.__product_name = product_name

@property
def product_price(self):
    return float(self.__product_price).__abs__()

@product_price.setter
def product_price(self, value):
    self.__product_price = value
```

After the getter and setter methods were created for the Product class, I created a custom to_string and an overwrite of the original __str__ methods specifically for the Product class. Both the new __str__ and to_string methods return the name of the product follow by a ", $" and the product price for formatting purposes.



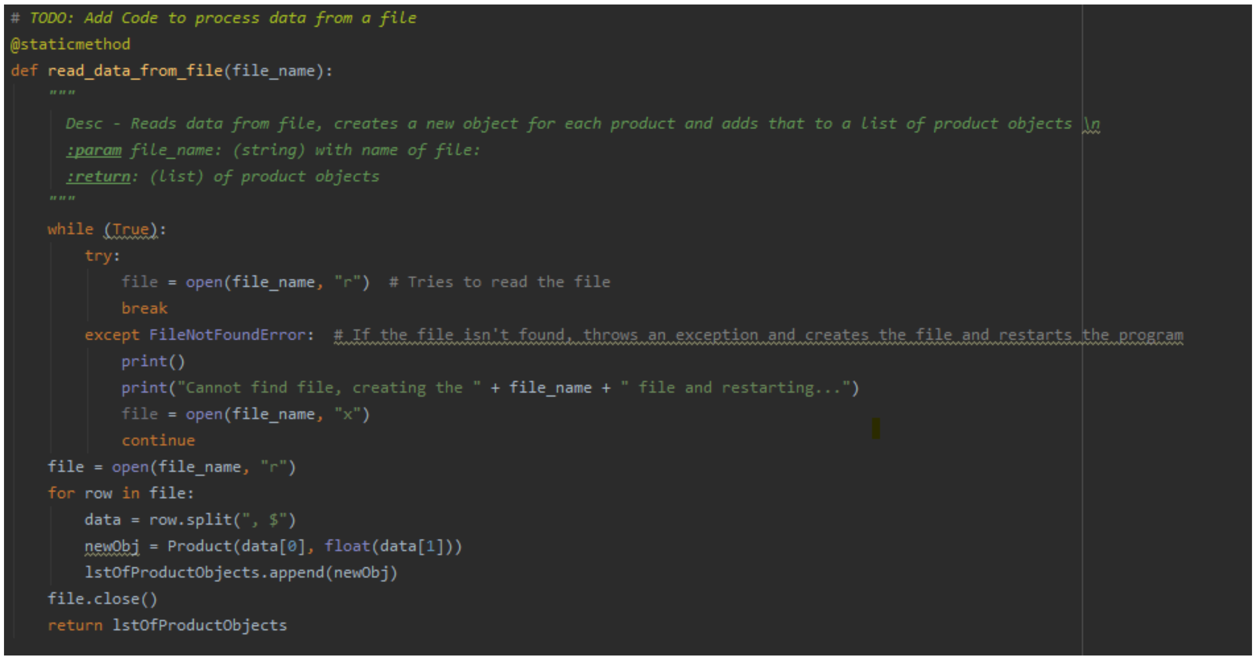
```python
def __str__(self):
    return self.product_name + ", $%.2f" % self.product_price

def to_string(self):
    return self.__str__()
```

After testing and ensuring that all the features of the Product class were functioning as needed, I moved onto creating the FileProcessor class. For this class, I started out by going back to Assignment06 and coping the save to file function as well as the read from file functions into this script as a starting point. I then updated the code for the “save_data_to_file” function to take in a file_name and a list_of_product_objects per the doc string above. The “save_data_to_file” function looked as following:



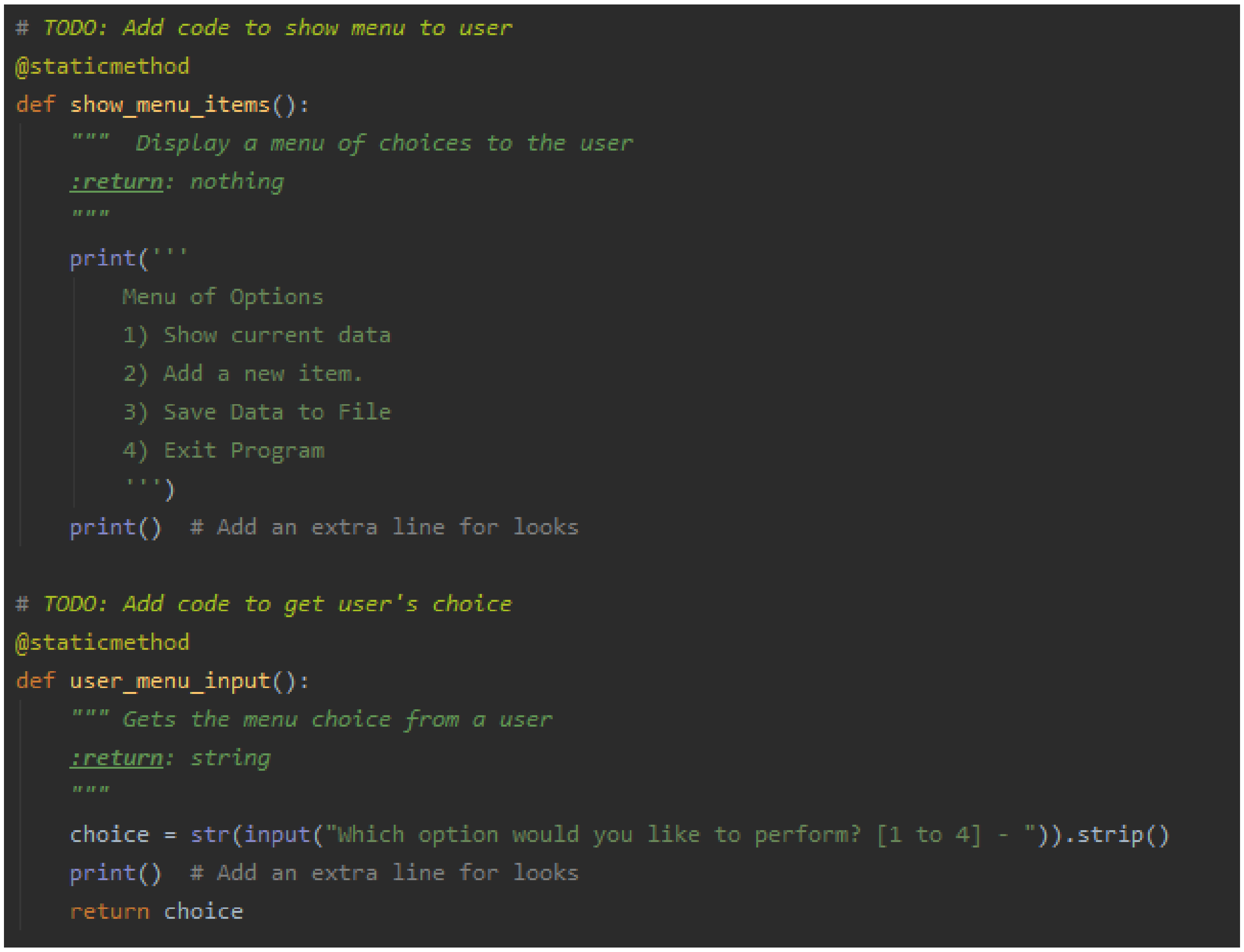
```
# TODO: Add Code to process data to a file
@staticmethod
def save_data_to_file(file_name, list_of_product_objects):
    file = open(file_name, "w")
    for row in list_of_product_objects:  # Write each row of data to the file
        file.write(str(row) + "\n")
    file.close()
```

For the “read from file” function, I also modified the code that I had written for Assignment06 and changed it so that instead of creating a dictionary row from the file data, instead the data was split on the “, $” (which is what my formatting for products was above) and a new object of the Product class was created for each item in the file. Then these new product objects are appended to the lstOfProductObjects list.

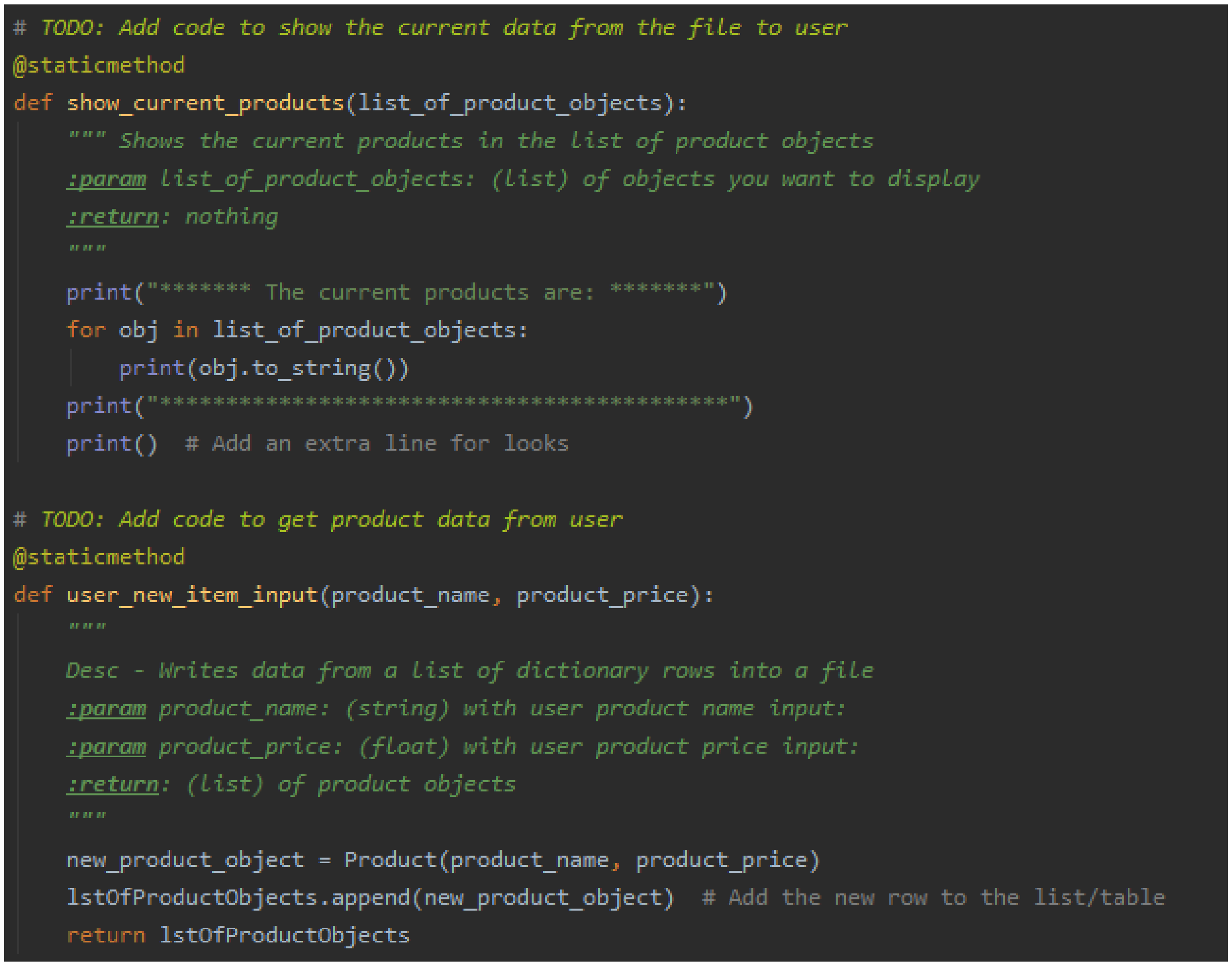


```
# TODO: Add Code to process data from a file
@staticmethod
def read_data_from_file(file_name):
    """
      Desc - Reads data from file, creates a new object for each product and adds that to a list of product objects \n
      :param file_name: (string) with name of file:
      :return: (list) of product objects
    """
    while (True):
        try:
            file = open(file_name, "r")  # Tries to read the file
            break
        except FileNotFoundError:  # If the file isn't found, throws an exception and creates the file and restarts the program
            print()
            print("Cannot find file, creating the " + file_name + " file and restarting...")
            file = open(file_name, "x")
            continue
    file = open(file_name, "r")
    for row in file:
        data = row.split(", $")
        newObj = Product(data[0], float(data[1]))
        lstOfProductObjects.append(newObj)
    file.close()
    return lstOfProductObjects
```

This completed the tasks in the FileProcessor class, and after testing I moved onto to the IO class. In this class I started out with once again copying the static functions from Assignment06 that were similar to what was needed to be achieved in this assignment. I copied over the menu function that just showed the menu options and changed it to reflect the options of this assignment. I also copied over the user_menu_input function that takes the user input and returns what the user chose.

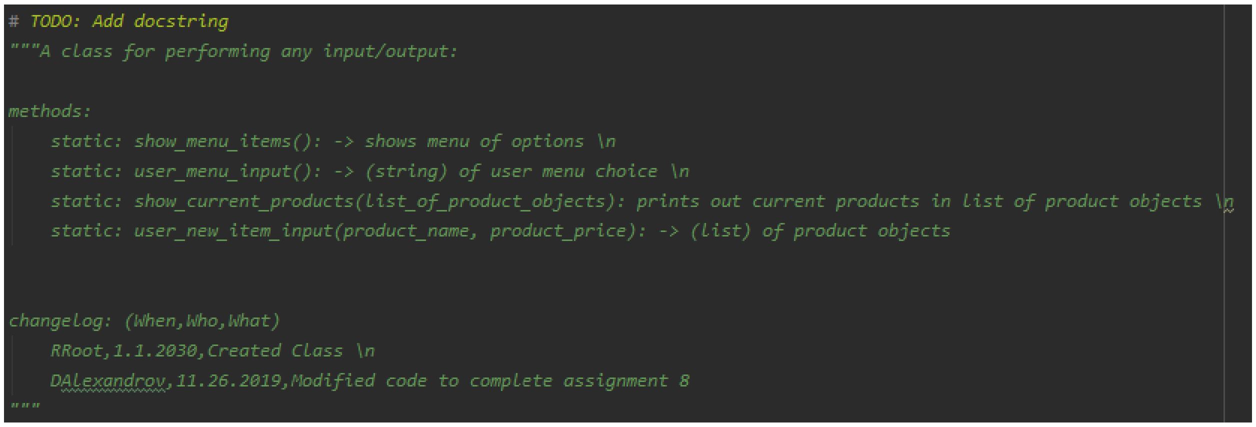

```python
# TODO: Add code to show menu to user
@staticmethod
def show_menu_items():
    """  Display a menu of choices to the user
    :return: nothing
    """
    print('''
        Menu of Options
        1) Show current data
        2) Add a new item.
        3) Save Data to File
        4) Exit Program
        ''')
    print()  # Add an extra line for looks

# TODO: Add code to get user's choice
@staticmethod
def user_menu_input():
    """ Gets the menu choice from a user
    :return: string
    """
    choice = str(input("Which option would you like to perform? [1 to 4] - ")).strip()
    print()  # Add an extra line for looks
    return choice
```

Then I copied over and modified the show current data function, which now takes in the list of product objects (lstOfProductObjects) and for each item in that list prints out using the custom "to_string" method that I created in the Product class above. Next, I added the method to add a new product named "user_new_item_input" since this method would be processing the user input and passing it over to the Product class to create a new product object which in turn would be appended to the list of objects.



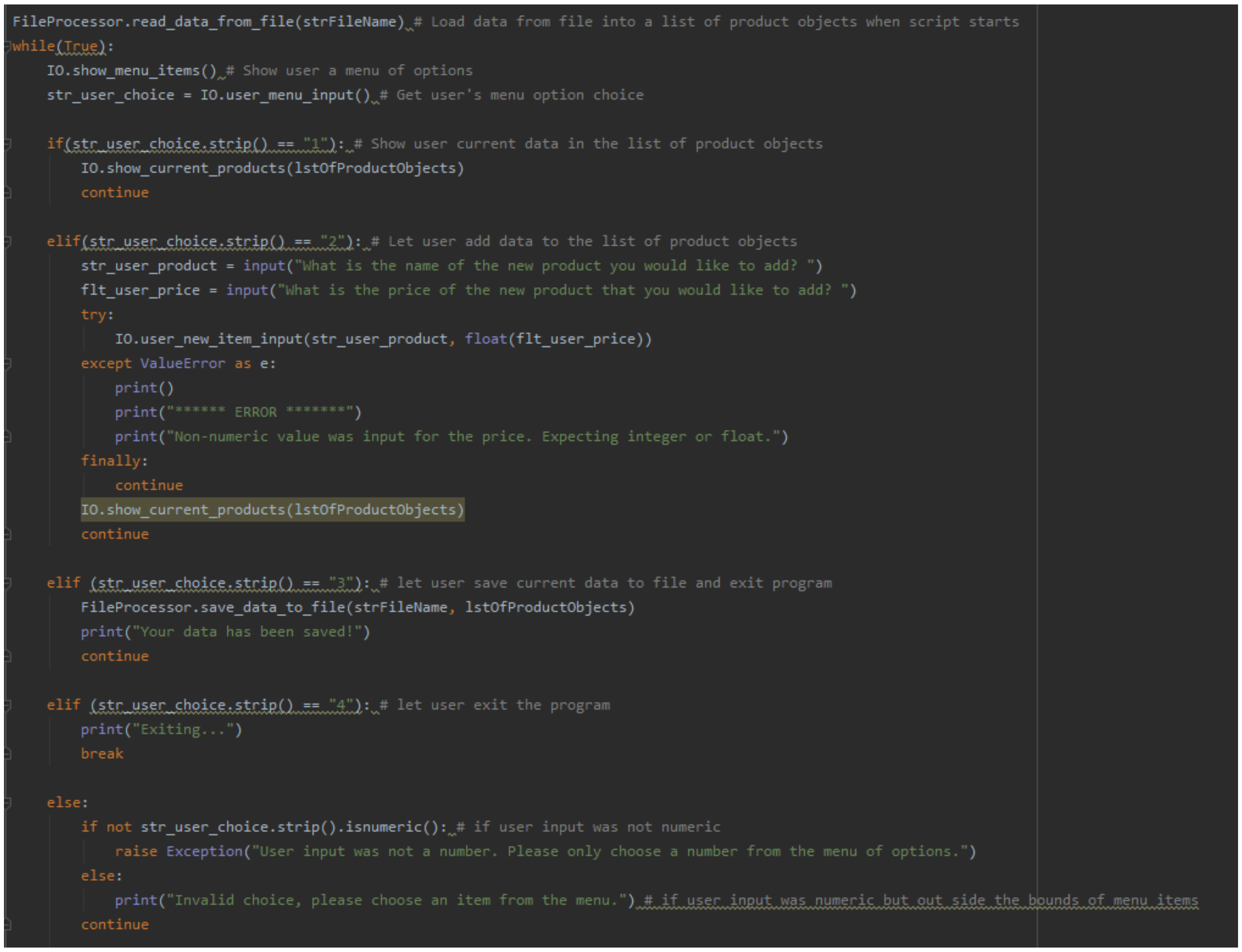
```
# TODO: Add code to show the current data from the file to user
@staticmethod
def show_current_products(list_of_product_objects):
    """ Shows the current products in the list of product objects
    :param list_of_product_objects: (list) of objects you want to display
    :return: nothing
    """
    print("******* The current products are: *******")
    for obj in list_of_product_objects:
        print(obj.to_string())
    print("*******************************************")
    print()  # Add an extra line for looks

# TODO: Add code to get product data from user
@staticmethod
def user_new_item_input(product_name, product_price):
    """
    Desc - Writes data from a list of dictionary rows into a file
    :param product_name: (string) with user product name input:
    :param product_price: (float) with user product price input:
    :return: (list) of product objects
    """
    new_product_object = Product(product_name, product_price)
    lstOfProductObjects.append(new_product_object)  # Add the new row to the list/table
    return lstOfProductObjects
```

Lastly after testing and ensuring that all the methods that I had just added to the IO class were functioning as needed. I went back to the top of the IO class and added the Doc string at the top so that another person using this code could understand what the class is doing and what methods were in it.



```
# TODO: Add docstring
"""A class for performing any input/output:

methods:
    static: show_menu_items(): -> shows menu of options \n
    static: user_menu_input(): -> (string) of user menu choice \n
    static: show_current_products(list_of_product_objects): prints out current products in list of product objects \n
    static: user_new_item_input(product_name, product_price): -> (list) of product objects


changelog: (When,Who,What)
    RRoot,1.1.2030,Created Class \n
    DAlexandrov,11.26.2019,Modified code to complete assignment 8
"""
```

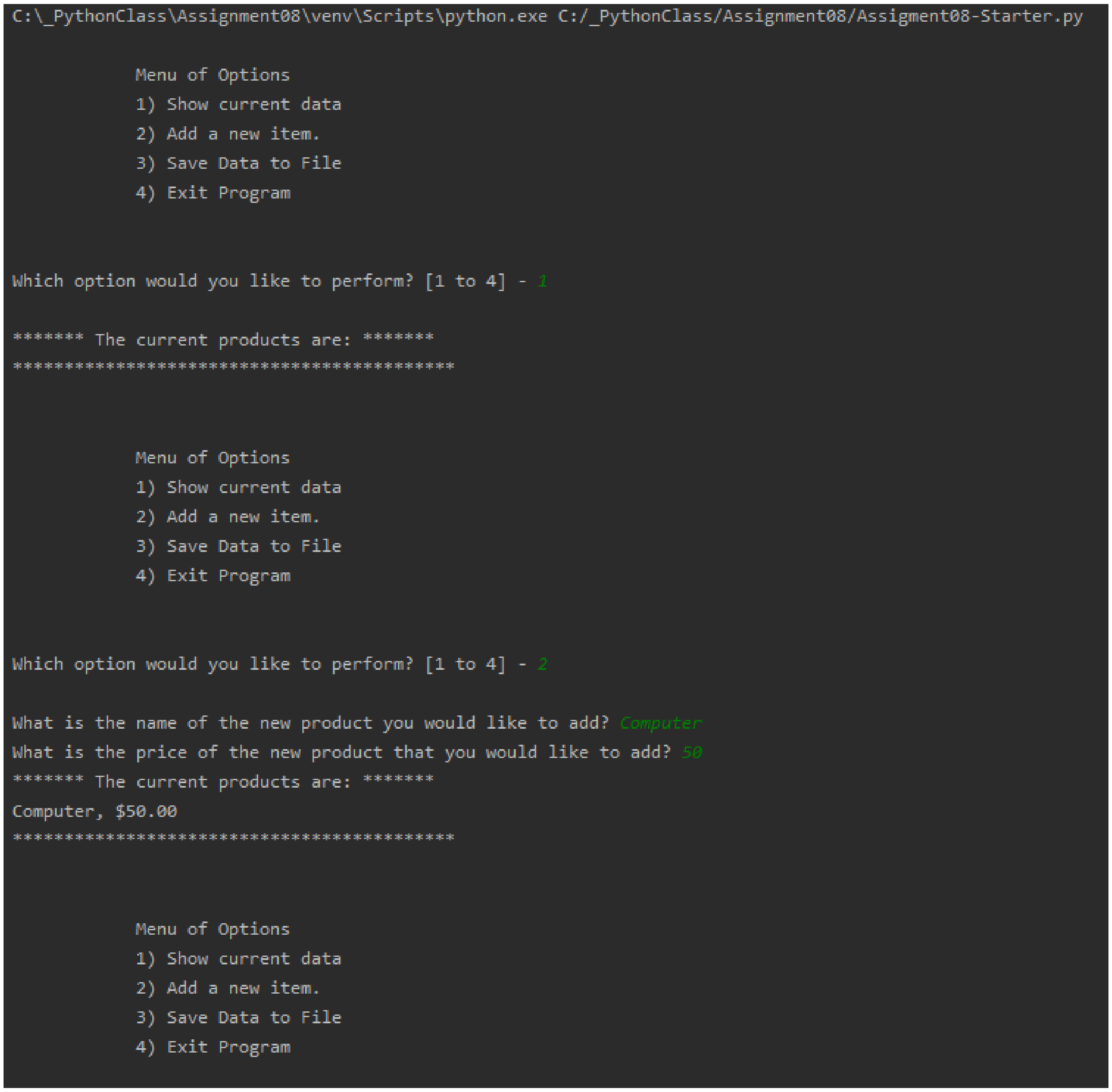
After all the classes, properties and methods were functioning as needed I moved onto the main body of the script. Where I began by calling the FileProcessor class and the “read_data_from_file” method passing it the name of the file (strFileName) to load the file contents into memory. Then I created a while loop as we have done before to keep cycling the user through the menu as they chose different options. The menu and user input were taken in each time cycle of the while loop using IO class and the “show_menu_items” and “user_menu_input” methods. Depending on which option the user chose, a different class and method would be called to perform those actions. Error handling for the creation of a new item was introduced in case the user did not input a numeric value for the price. And an option to exit the script was added. Also error handling was added if a user did not choose a numeric value from the menu.



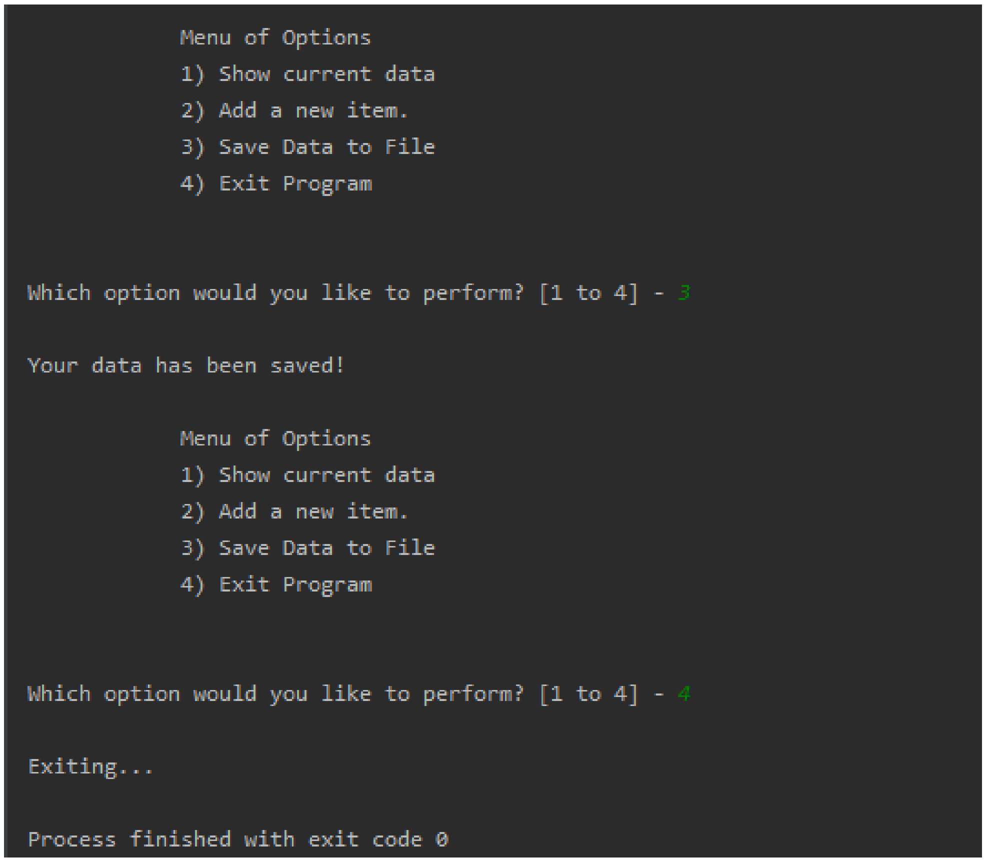
```
FileProcessor.read_data_from_file(strFileName) # Load data from file into a list of product objects when script starts
while(True):
    IO.show_menu_items() # Show user a menu of options
    str_user_choice = IO.user_menu_input() # Get user's menu option choice

    if(str_user_choice.strip() == "1"): # Show user current data in the list of product objects
        IO.show_current_products(lstOfProductObjects)
        continue

    elif(str_user_choice.strip() == "2"): # Let user add data to the list of product objects
        str_user_product = input("What is the name of the new product you would like to add? ")
        flt_user_price = input("What is the price of the new product that you would like to add? ")
        try:
            IO.user_new_item_input(str_user_product, float(flt_user_price))
        except ValueError as e:
            print()
            print("****** ERROR *******")
            print("Non-numeric value was input for the price. Expecting integer or float.")
        finally:
            continue
        IO.show_current_products(lstOfProductObjects)
        continue

    elif (str_user_choice.strip() == "3"): # let user save current data to file and exit program
        FileProcessor.save_data_to_file(strFileName, lstOfProductObjects)
        print("Your data has been saved!")
        continue

    elif (str_user_choice.strip() == "4"): # let user exit the program
        print("Exiting...")
        break

    else:
        if not str_user_choice.strip().isnumeric(): # if user input was not numeric
            raise Exception("User input was not a number. Please only choose a number from the menu of options.")
        else:
            print("Invalid choice, please choose an item from the menu.") # if user input was numeric but out side the bounds of menu items
        continue
```

## Summary

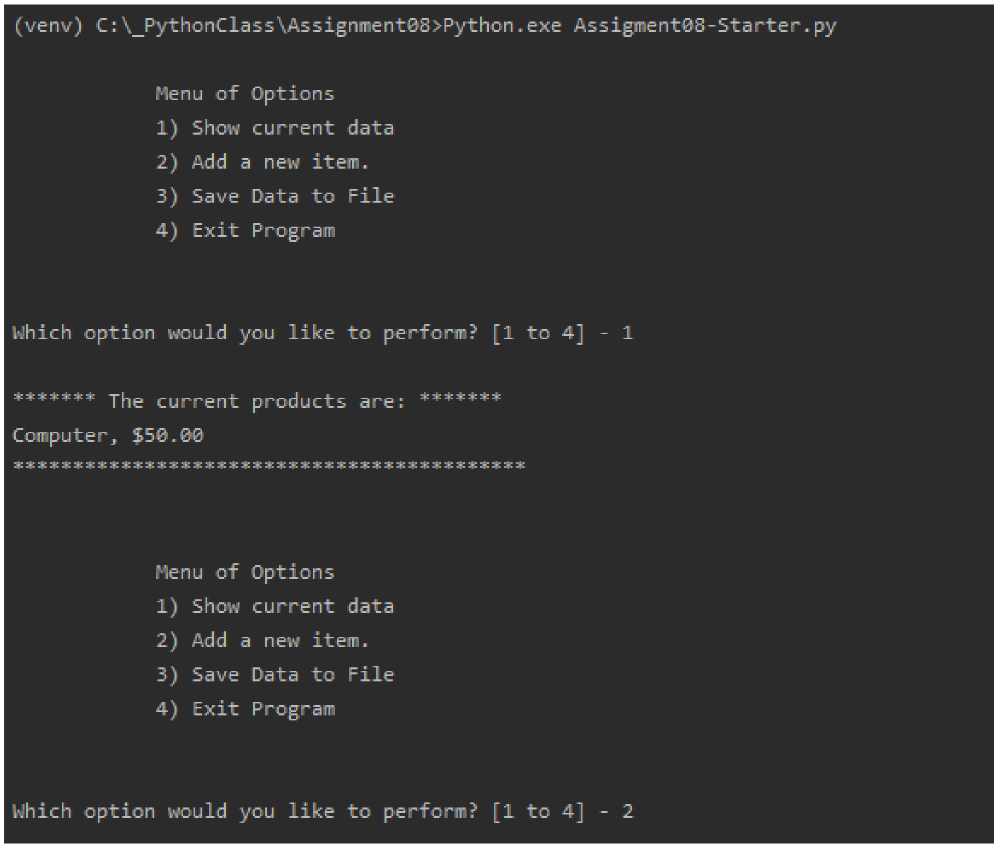
Overall, the script performed the functions that were asked of it in the assignment. It was really interesting to practice the use of classes and being able to call upon your own created classes to perform actions in your body of code. Below are the outputs of the code running in PyCharm and in the Terminal:



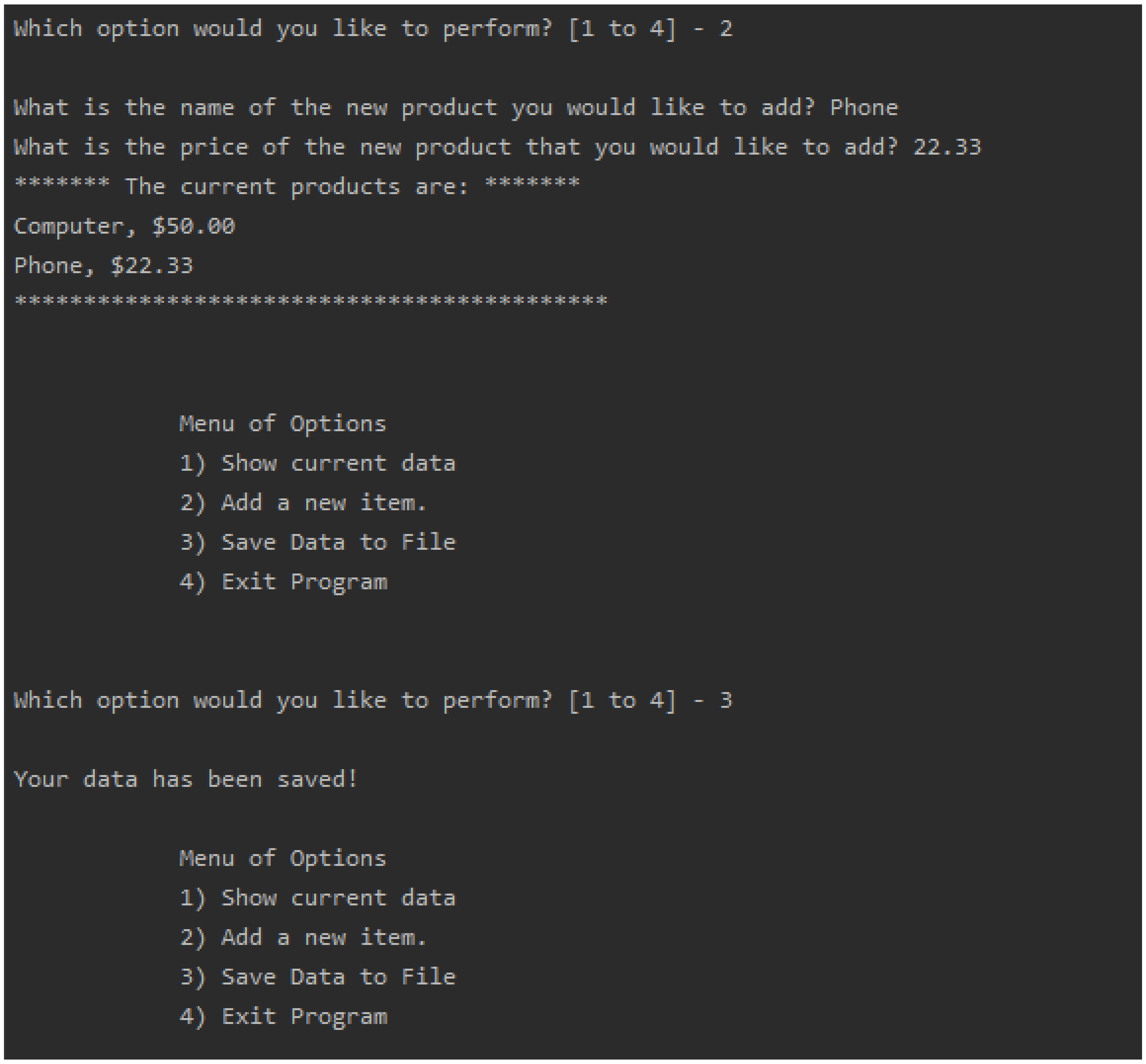
```
C:\_PythonClass\Assignment08\venv\Scripts\python.exe C:/_PythonClass/Assignment08/Assigment08-Starter.py

            Menu of Options
            1) Show current data
            2) Add a new item.
            3) Save Data to File
            4) Exit Program


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

******* The current products are: *******
*******************************************



            Menu of Options
            1) Show current data
            2) Add a new item.
            3) Save Data to File
            4) Exit Program


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

What is the name of the new product you would like to add? Computer
What is the price of the new product that you would like to add? 50
******* The current products are: *******
Computer, $50.00
*******************************************



            Menu of Options
            1) Show current data
            2) Add a new item.
            3) Save Data to File
            4) Exit Program
```



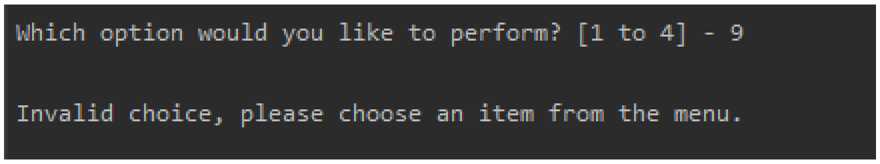
```
            Menu of Options
            1) Show current data
            2) Add a new item.
            3) Save Data to File
            4) Exit Program


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

Your data has been saved!

            Menu of Options
            1) Show current data
            2) Add a new item.
            3) Save Data to File
            4) Exit Program


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

Exiting...

Process finished with exit code 0
```



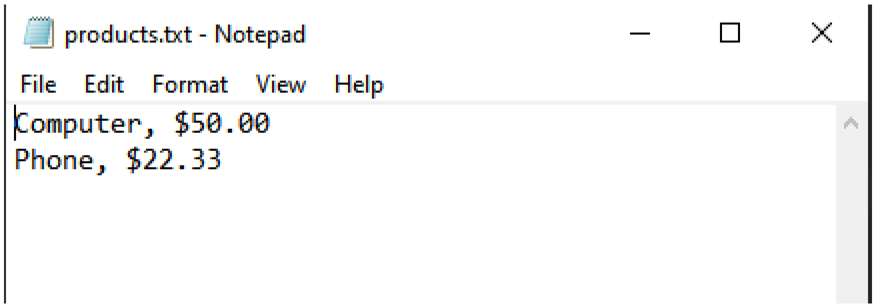
```
(venv) C:\_PythonClass\Assignment08>Python.exe Assigment08-Starter.py

            Menu of Options
            1) Show current data
            2) Add a new item.
            3) Save Data to File
            4) Exit Program


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

******* The current products are: *******
Computer, $50.00
*******************************************


            Menu of Options
            1) Show current data
            2) Add a new item.
            3) Save Data to File
            4) Exit Program


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



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

What is the name of the new product you would like to add? Phone
What is the price of the new product that you would like to add? 22.33
******* The current products are: *******
Computer, $50.00
Phone, $22.33
*******************************************



            Menu of Options
            1) Show current data
            2) Add a new item.
            3) Save Data to File
            4) Exit Program



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

Your data has been saved!

            Menu of Options
            1) Show current data
            2) Add a new item.
            3) Save Data to File
            4) Exit Program
```



```
Which option would you like to perform? [1 to 4] - 9

Invalid choice, please choose an item from the menu.
```



```
Computer, $50.00
Phone, $22.33
```