

Saglara Orellana

June 6, 2022

Foundation of Programming: Python
Assignment 08

List Of Product Objects

Introduction

The starter script had three classes created: one is data class called `Product` and two are processing classes `FileProcessor` and `IO`. In this assignment I needed to add the code to complete defining classes and write the main body of the script, using properties and methods of those classes.

Class Product

The first class, I need to add code to, is *Product* class, I will be creating objects from this class, so it will hold information about the product objects, such as product name and its price. I start with updating docstring's changelog, then I add the constructor to set the initial values for product name and product price:

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

The starter script had included two properties for this class, that I defined next. I wrote codes for getter and setter of *product_name* and *product_price* properties. Getter for *product_name* gets string value with name of the product and format it into a title, and setter checks if the string value contains numbers, if it does, it will raise an error. For *product_price* property I made sure that its value is numeric, otherwise it will raise an error.

```
# Properties ----- #  
  
# product name  
  
@property    # getter decorator
```

```

def product_name(self):
    return str(self.__product_name_str).title()

@product_name.setter    # setter decorator
def product_name(self, value):
    if str(value).isnumeric() == False:
        self.__product_name_str = value
    else:
        raise Exception('Product name cannot be number')

# product price
@property    # getter decorator
def product_price(self):
    return str(self.__product_price_str)

@product_price.setter    # setter decorator
def product_price(self, value):
    if str(value).isnumeric() == True:
        self.__product_price_str = value
    else:
        raise Exception('Price must be in numbers')

```

I also added method `__str__()` to overwrite its default statement of showing object's class and its address to return string values of its attributes instead:

```

# Methods ----- #
def __str__(self):
    return self.product_name + ',' + self.product_price

```

Class FileProcessor

Next I need to define class `FileProcessor`, I started with adding changelog to the docstring. This class will have two methods for processing the file: saving data and reading data. Generally, I would define them as static methods, since they are processing functions, but the docstring did

not specify, so I created them as instance methods instead, having *self* as one of parameters and calling them from an object in the main body of the script.

When I want to load the data from file, I start with removing current data from the list. Then I open the file and loop through each line. First, I assign values that are split by comma to variables *product* and *price*. Next, I create an object from *Product* class, that I defined earlier, and pass it values of variables *product* and *price*. Then I add the object to the list.

```
def read_data_from_file(self, file_name, list_of_product_objects):  
    """ Reads data from a file into a list of rows  
  
    :param file_name: (string) with name of file:  
    :return: list_of_product_objects: (list) of objects  
    """  
    list_of_product_objects.clear()  
    with open(file_name, 'r') as file:  
        for line in file:  
            product, price = line.split(',')  
            obj_product = Product(product, price.strip())  
            list_of_product_objects.append(obj_product)  
    return list_of_product_objects
```

To save the data to a file, I need to open the file and unpack the list by looping through each element (object) and then write it as a string to a file:

```
def save_data_to_file(self, file_name, list_of_product_objects):  
    """ Saves list of product objects to a file  
  
    :param file_name: (string) with file name  
    :param list_of_product_objects: (list) of objects  
    """  
    with open(file_name, 'w') as file:  
        for row in list_of_product_objects:  
            file.write(str(row) + '\n')  
    return 'Data was saved. Goodbye!'
```

Class IO

This class was missing docstring, so first I added it. The class will display and collect the data and will contain four static methods (each method will have `@staticmethod` decorator):

```
class IO:
    """
    Displaying and collecting user data:

    static methods:
        show_menu():
        input_user_choice(): -> (string) with menu choice
        print_current_data(list_of_product_objects):
        add_product(): -> (object)

    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
        SOrellana,06/05/2022,Modified code to complete assignment 8
    """
```

Show_menu() will display menu options to the user with a print function. Reading through pseudo-code, I identified 3 menu options: show current data, add new data and save data.

```
@staticmethod
def show_menu():
    """ Display a menu of choices to the user
    :return: nothing
    """
    print('-' * 33)
    print('Menu of Options:
    1) Show Current List of Products
    2) Add New Product
    3) Save Data to a File and Exit')
    print('-' * 33)
```

Next I added code for collecting user input to choose menu option:

```
@staticmethod
```

```

def input_user_choice():
    """ Gets menu choice from user
    :return: string
    """
    choice = str(input('Which option would you like to perform? [1 to 3]: '))
    print('-' * 33) # for looks
    return choice

```

Third static method is to display current data. It will require a parameter with name of the list to be displayed:

```

@staticmethod
def print_current_data(list_of_product_objects):
    """ Displays current list of products
    :param list_of_product_objects: (list) with products and prices
    :return: nothing
    """
    print('-----Current List Of Products-----')
    for row in list_of_product_objects: # looping through objects in the
list
        print(str(row))
    print() # extra line for looks

```

The last method in this class is *add_product*. It also takes parameter with the name of the list, so that it know where to add new product. This function first collects user input for product name and its price, then creates a new object and adds it to the existing list. To catch erroneous user input, I added try-except statement, where the exception will be raised when program tries to create a Product class object with invalid attributes. It is going to raise Exception error, if product name contains numbers or product price was entered as word.

```

@staticmethod
def add_product(list_of_product_objects):
    """ Adds a new product object to the list
    :param list_of_product_objects: (list) of product objects
    :return: (list) of product objects
    """
    try:
        product = input('Please enter a product: ')

```

```

        price = input('Please enter product price: ')
        obj_product = Product(product, price)    # new object from Product
class
        list_of_product_objects.append(obj_product)
    except Exception as e:    # catching errors for invalid input values
        print(e)
    return list_of_product_objects

```

Writing and Testing The Script

Now that all my classes were defined, I can start writing the main body of the script by following pseudocode. The first thing I need to do is to load data from the file into list of product objects. I need to call for one of the FileProcessor's methods. Since it was not static method, I have to create an object first and call for function, using that object. The first run gave me an error, so I added try-except to handle the error.

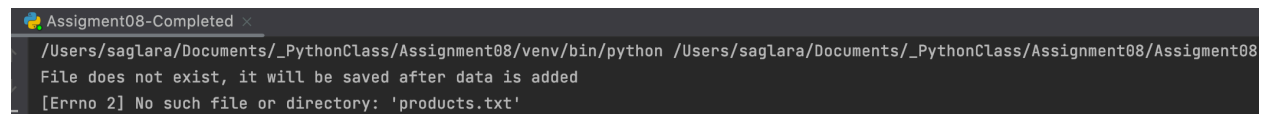
```

# Load data from file into a list of product objects when script starts
try:
    objF = FileProcessor()
    lstOfProductObjects = objF.read_data_from_file(strFileName,
lstOfProductObjects)
    print('-----Current data in file-----')
    for row in lstOfProductObjects:    # looping through objects in the list
        print(str(row))

except FileNotFoundError as e:
    print('File does not exist, it will be saved after data is added')
    print(e)

```

After adding error handling the program notifies me, that the file will be saved after data is added. Figure 01 shows the program error message.



```

Assignment08-Completed x
/Users/saglara/Documents/_PythonClass/Assignment08/venv/bin/python /Users/saglara/Documents/_PythonClass/Assignment08/Assignment08
File does not exist, it will be saved after data is added
[Errno 2] No such file or directory: 'products.txt'

```

Figure 01. Error: File does not exist.

Right after the data is loaded, I entered the *while* loop, so that the user is allowed to add multiple products to the list before saving the data and exiting. Inside the loop the program first shows the menu options to the user. It is a static method of IO class. Next, I collect the user input *strChoice* by calling function *IO.input_user_choice()*.

```
while True:
    # Show user a menu of options
    IO.show_menu()
    # Get user's menu option choice
    strChoice = IO.input_user_choice()
```

When the user chooses option 1, the program shows current data - prints *lstOfProductObjects*.

```
# Show user current data in the list of product objects
if strChoice == '1':
    IO.print_current_data(lstOfProductObjects)
```

Menu option 2 allows to add new product to the list. I assign *lstOfProductObjects* to the value of static method *add_product* from *IO* class with parameter *lstOfProductObjects*.

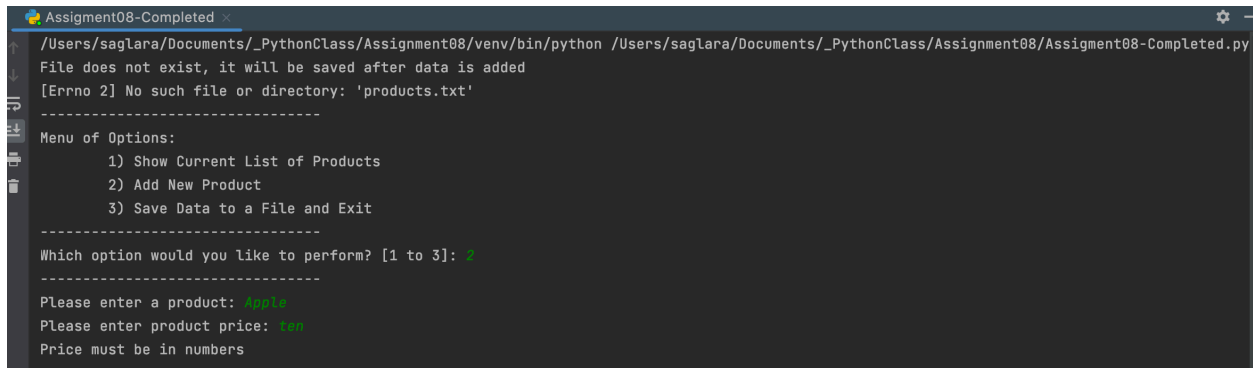
```
# Let user add data to the list of product objects
elif strChoice == '2':
    lstOfProductObjects = IO.add_product(lstOfProductObjects)
```

Option 3 of the menu saves the data into a file and ends the program. I need to call for function from *FileProcessor* class, using object *objF*. This function returns a string with a status message, so I decided to use function *print* to display it. Then program ends.

```
# let user save current data to file and exit program
elif strChoice == '3':
    objF = FileProcessor()
    print(objF.save_data_to_file(strFileName, lstOfProductObjects))
    break
```

I run the program again and since I do not have a text file existed, I choose option 2 to add products. Figure 02 shows error message, when I enter product price as a word instead of number. If I enter product name as numbers, the program still asks to enter its price before

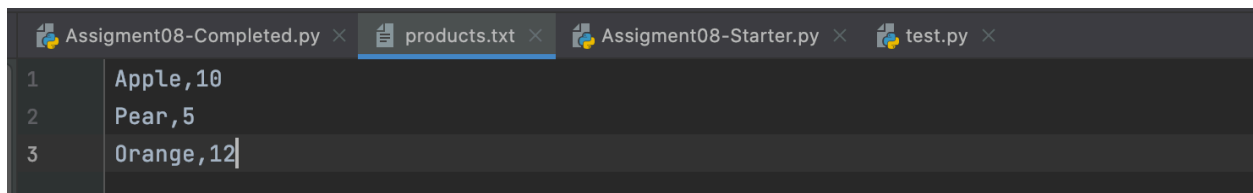
showing error message. The error is raised, when the new object from Product class is created, which happens after both inputs are collected.



```
Assignment08-Completed x
/Users/saglara/Documents/_PythonClass/Assignment08/venv/bin/python /Users/saglara/Documents/_PythonClass/Assignment08/Assignment08-Completed.py
File does not exist, it will be saved after data is added
[Errno 2] No such file or directory: 'products.txt'
-----
Menu of Options:
  1) Show Current List of Products
  2) Add New Product
  3) Save Data to a File and Exit
-----
Which option would you like to perform? [1 to 3]: 2
-----
Please enter a product: Apple
Please enter product price: ten
Price must be in numbers
-----
```

Figure 02. Adding a new product with invalid price.

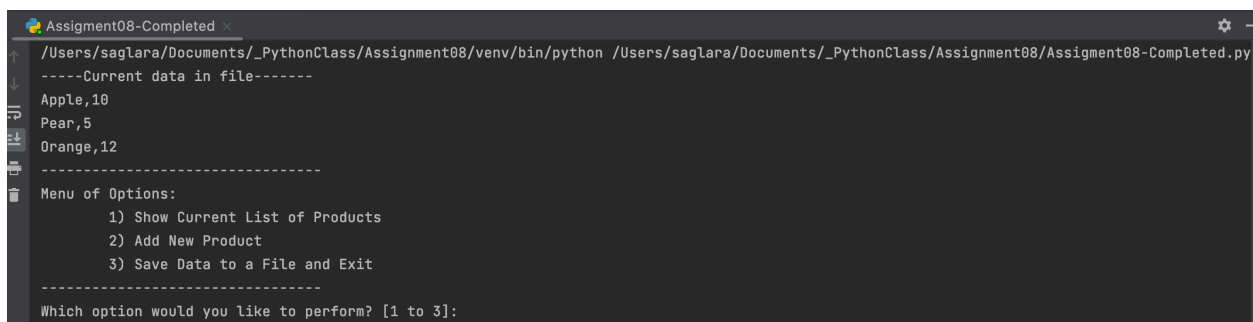
Once I've added few products, I enter option 3 to save data to a file. The program ends with a message 'Data was saved. Goodbye!'. New file appeared in the project folder called products.txt and it contains the data, I have previously entered.



```
Assignment08-Completed.py x products.txt x Assignment08-Starter.py x test.py x
1 Apple,10
2 Pear,5
3 Orange,12
```

Figure 03. Products.txt.

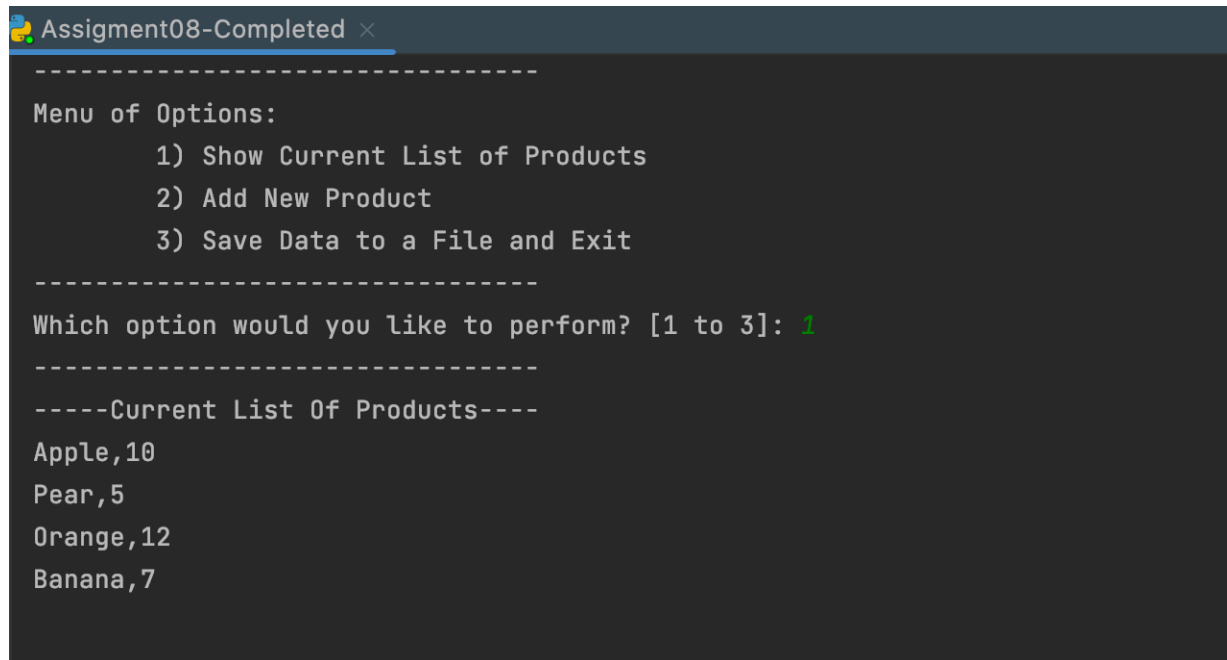
Now that I have a file created, I tested the program again to make sure it loads the data correctly. Figure 04 shows the results.



```
Assignment08-Completed x
/Users/saglara/Documents/_PythonClass/Assignment08/venv/bin/python /Users/saglara/Documents/_PythonClass/Assignment08/Assignment08-Completed.py
----Current data in file-----
Apple,10
Pear,5
Orange,12
-----
Menu of Options:
  1) Show Current List of Products
  2) Add New Product
  3) Save Data to a File and Exit
-----
Which option would you like to perform? [1 to 3]:
```

Figure 04. Loading data from file.

I have added another product (Banana, 7) to the list and chose option 1 to display current list of products.



```
Assigment08-Completed x
-----
Menu of Options:
    1) Show Current List of Products
    2) Add New Product
    3) Save Data to a File and Exit
-----
Which option would you like to perform? [1 to 3]: 1
-----
-----Current List Of Products-----
Apple,10
Pear,5
Orange,12
Banana,7
-----
```

Figure 05. Displaying current list of products.

Running script in Terminal initially will give me an error message, that file doesn't exist. I added few items to the list and saved it first. Figure 06 shows the script running in command shell.

```
saglara — -bash — 89x35
Saglaras-MBP:~ saglara$ python3 /Users/saglara/Documents/_PythonClass/Assignment08/Assignment08-Completed.py
File does not exist, it will be saved after data is added
[Errno 2] No such file or directory: 'products.txt'
-----
Menu of Options:
  1) Show Current List of Products
  2) Add New Product
  3) Save Data to a File and Exit
-----
Which option would you like to perform? [1 to 3]: 2
-----
Please enter a product: Peach
Please enter product price: 5
-----
Menu of Options:
  1) Show Current List of Products
  2) Add New Product
  3) Save Data to a File and Exit
-----
Which option would you like to perform? [1 to 3]: 2
-----
Please enter a product: Berry
Please enter product price: 4
-----
Menu of Options:
  1) Show Current List of Products
  2) Add New Product
  3) Save Data to a File and Exit
-----
Which option would you like to perform? [1 to 3]: 3
-----
Data was saved. Goodbye!
Saglaras-MBP:~ saglara$
```

Figure 06. Running script in command shell.

New file is created, I tried to re-run it again to make sure it loads the data correctly and capture it as Figure 07.

```
saglara — -bash — 91x38
Saglaras-MBP:~ saglara$ python3 /Users/saglara/Documents/_PythonClass/Assignment08/Assignment08-Completed.py
-----Current data in file-----
Peach,5
Berry,4
-----
Menu of Options:
  1) Show Current List of Products
  2) Add New Product
  3) Save Data to a File and Exit
-----
Which option would you like to perform? [1 to 3]: 2
-----
Please enter a product: cherry
Please enter product price: 6
-----
Menu of Options:
  1) Show Current List of Products
  2) Add New Product
  3) Save Data to a File and Exit
-----
Which option would you like to perform? [1 to 3]: 1
-----
-----Current List Of Products-----
Peach,5
Berry,4
Cherry,6
-----
Menu of Options:
  1) Show Current List of Products
  2) Add New Product
  3) Save Data to a File and Exit
-----
Which option would you like to perform? [1 to 3]: 3
-----
Data was saved. Goodbye!
Saglaras-MBP:~ saglara$
```

Figure 07. Loading and showing current data.

Summary

I completed the script by adding codes to define three classes and then later calling them in the main body of the script. This script loads data from the file as objects and adds it to the list. When the user wants to add new data, new object is created from class with user input data as attributes. If the user enters incorrect values, the program errors out and proceeds to run again. I have defined method `__str__()` to return custom values, so that when the list of objects is saved or displayed, it uses objects' attributes instead of its class and address.