12/04/21

Matthew Fuller IT FDN 110 A

https://github.com/mfuller7/IntroToProg-Python-Mod08

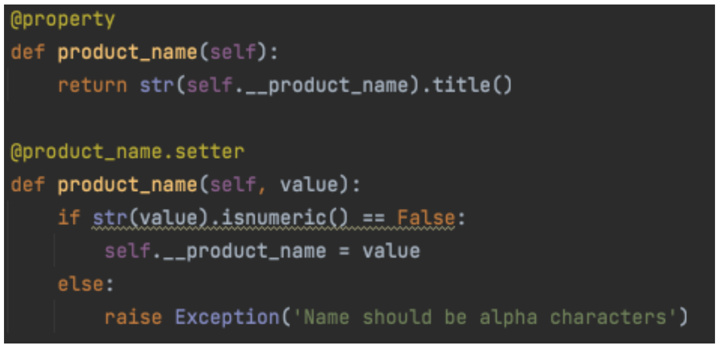
Assignment 08 Knowledge Document

Introduction

I have not had a lot of practice working with classes before. It seems like the organization and reproducibility classes offer is extremely useful. However it does also seem to make code more "complex" sometimes, as illustrated by the confusing, inconsistent behavior of setting the value of a class property in different ways.

Class Product

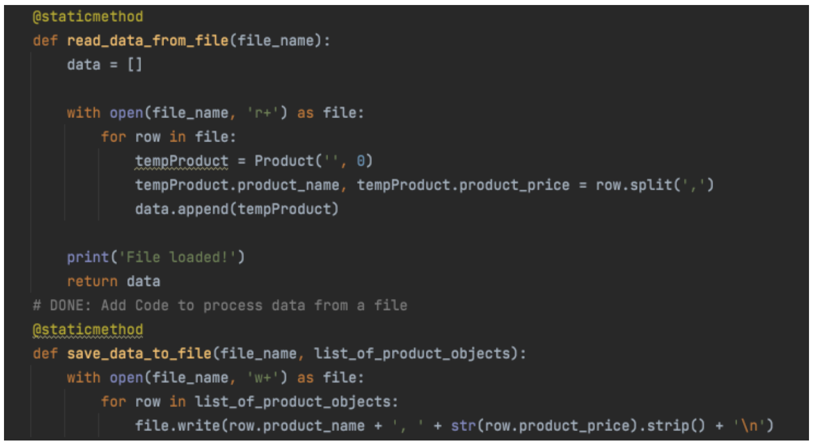
Filling in the skeleton of the *Product* class in the starter code went smoothly. Referencing the lecture notes provided clear examples of the "getter" and "setter" methods. Once I got the basic functionality in I started to add additional error handling and data validation.



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

@product_name.setter
def product_name(self, value):
    if str(value).isnumeric() == False:
        self.__product_name = value
    else:
        raise Exception('Name should be alpha characters')
```

Class FileProcesser

The *read_data_to_file* and *save_data_to_file* methods recycled code from last weeks exercise, with changes made to work with text files instead of binary. Some minor tweaks were needed after testing when I realized I missed some formatting string values in the *save* method. Additionally, for some reason when I run *open(file)* a new file is not created. I tried to figure out why this was but couldn't figure it out.



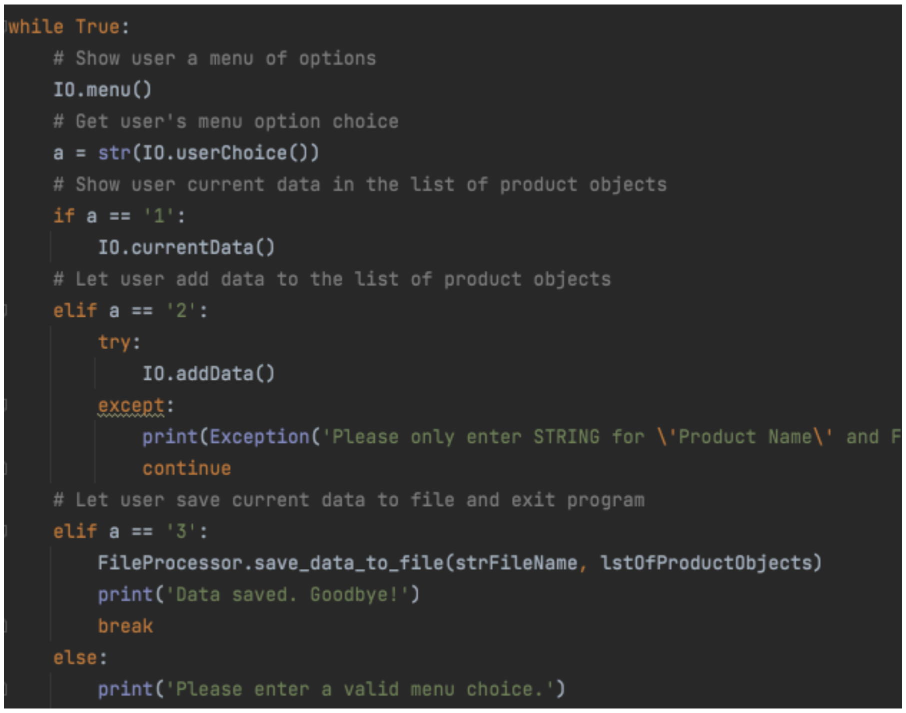
```
@staticmethod
def read_data_from_file(file_name):
    data = []

    with open(file_name, 'r+') as file:
        for row in file:
            tempProduct = Product('', 0)
            tempProduct.product_name, tempProduct.product_price = row.split(',')
            data.append(tempProduct)

    print('File loaded!')
    return data
# DONE: Add Code to process data from a file
@staticmethod
def save_data_to_file(file_name, list_of_product_objects):
    with open(file_name, 'w+') as file:
        for row in list_of_product_objects:
            file.write(row.product_name + ', ' + str(row.product_price).strip() + '\n')
```

Class IO

This class was also pretty straightforward. It was interesting to repackage elements of code that we have used throughout the course into individual methods. Even though it seemed like moving the simple “menu” code into a separate method was trivial it made the “body” of the code look so much cleaner.

Main Body

Putting everything together also felt pretty straightforward. Calling all the methods was simple because of the intentional naming conventions. I added some menu error handling with some *if* statements. Once I got to this point there was significant adjustments needed. Many times I ran into errors involving expected data types or missing expected arguments which were not hard to resolve.

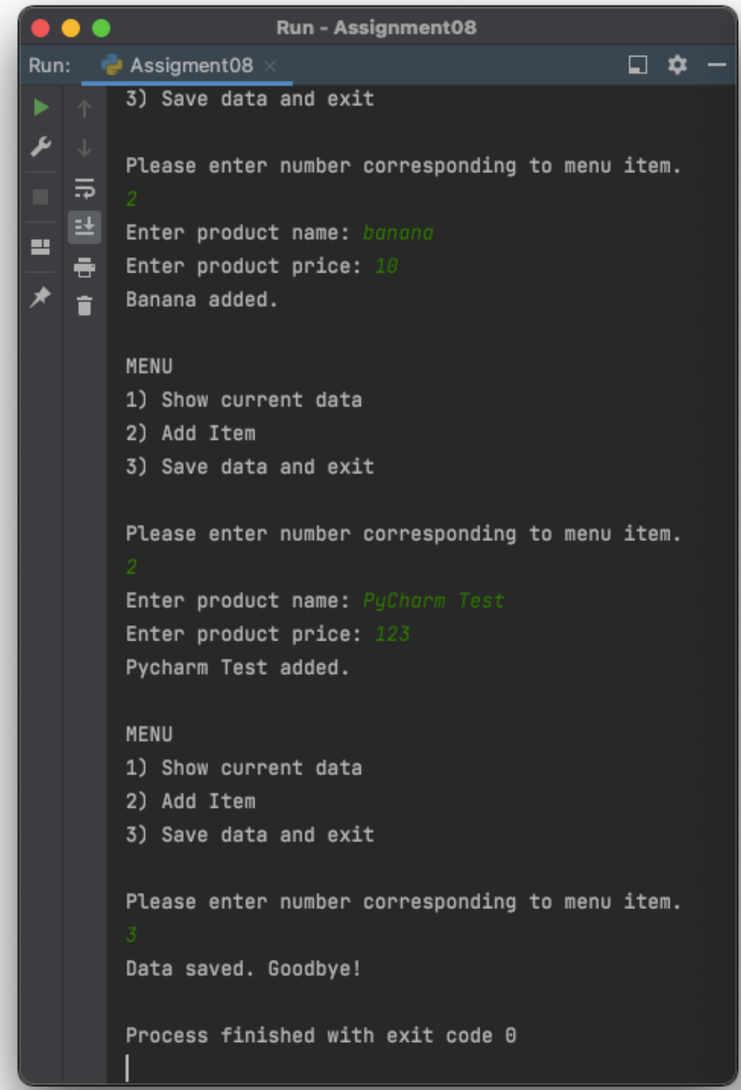


```
while True:
    # Show user a menu of options
    IO.menu()
    # Get user's menu option choice
    a = str(IO.userChoice())
    # Show user current data in the list of product objects
    if a == '1':
        IO.currentData()
    # Let user add data to the list of product objects
    elif a == '2':
        try:
            IO.addData()
        except:
            print(Exception('Please only enter STRING for \'Product Name\' and F
            continue
    # Let user save current data to file and exit program
    elif a == '3':
        FileProcessor.save_data_to_file(strFileName, lstOfProductObjects)
        print('Data saved. Goodbye!')
        break
    else:
        print('Please enter a valid menu choice.')
```

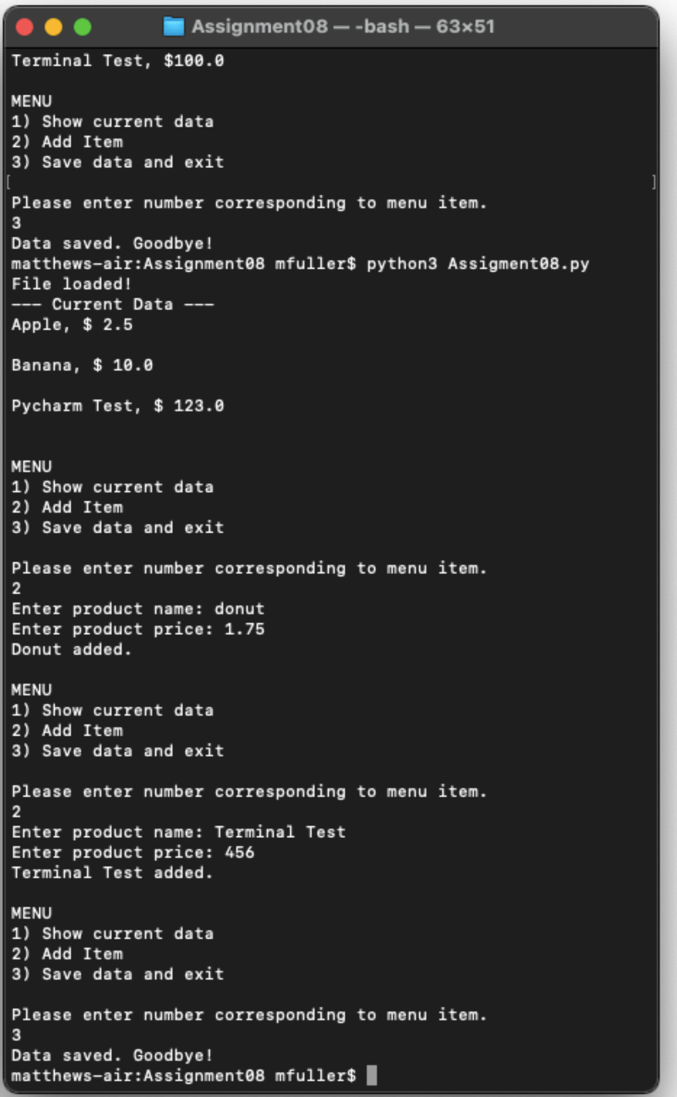
Summary

At first it took a lot of mental effort to track the flow of the code when organizing everything into classes. It was also initially challenging for me to grasp what was going on with “getter” and “setter” properties though eventually it made more sense and I chocked some of it to Python being Python. It was nice to begin to feel like everything we have practiced up till now is coalescing into a truly functional form.

Execution Verification



```
3) Save data and exit

Please enter number corresponding to menu item.
2
Enter product name: banana
Enter product price: 10
Banana added.

MENU
1) Show current data
2) Add Item
3) Save data and exit

Please enter number corresponding to menu item.
2
Enter product name: PyCharm Test
Enter product price: 123
Pycharm Test added.

MENU
1) Show current data
2) Add Item
3) Save data and exit

Please enter number corresponding to menu item.
3
Data saved. Goodbye!

Process finished with exit code 0
```



```
Terminal Test, $100.0

MENU
1) Show current data
2) Add Item
3) Save data and exit

Please enter number corresponding to menu item.
3
Data saved. Goodbye!
matthews-air:Assignment08 mfuller$ python3 Assigment08.py
File loaded!
--- Current Data ---
Apple, $ 2.5

Banana, $ 10.0

Pycharm Test, $ 123.0


MENU
1) Show current data
2) Add Item
3) Save data and exit

Please enter number corresponding to menu item.
2
Enter product name: donut
Enter product price: 1.75
Donut added.

MENU
1) Show current data
2) Add Item
3) Save data and exit

Please enter number corresponding to menu item.
2
Enter product name: Terminal Test
Enter product price: 456
Terminal Test added.

MENU
1) Show current data
2) Add Item
3) Save data and exit

Please enter number corresponding to menu item.
3
Data saved. Goodbye!
matthews-air:Assignment08 mfuller$
```