Jeff Nelson

June 10, 2020

IT FDN 100 A

Assignment #8

https://github.com/jnelson22/IntroToProg-Python-Mod08

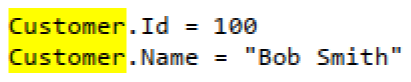
Objects and Classes

Intro

In this paper, I will go over a few interesting and important items to remember that I learned from this week's lecture. Lastly, I will show a program uses the knowledge from this week to capture a products name and price then write it to a file.

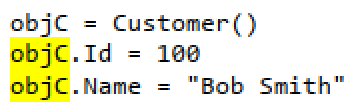
Week 8 Learnings

This week I learned more about objects and classes. Classes are used to help grouping of data and functions. Most of them are designed to with focus on data or processing. Data is class is defined using variables and constants. When they are used in a class, they are class Fields. Methods are used when you want to use a function. When using a class, you can either use it directly or indirectly.



```
Customer.Id = 100
Customer.Name = "Bob Smith"
```

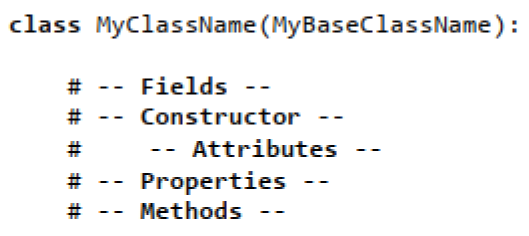
When you use the code indirectly, you create an object as shown below.



```
objC = Customer()
objC.Id = 100
objC.Name = "Bob Smith"
```

This allow you to create multiple object each with a different address in memory. Calling a class directly is usually for processing data and indirectly for storing data.

A typical class is setup having Fields, Constructor, Attributes, Properties and Methods. Below you can see the structure for Python.



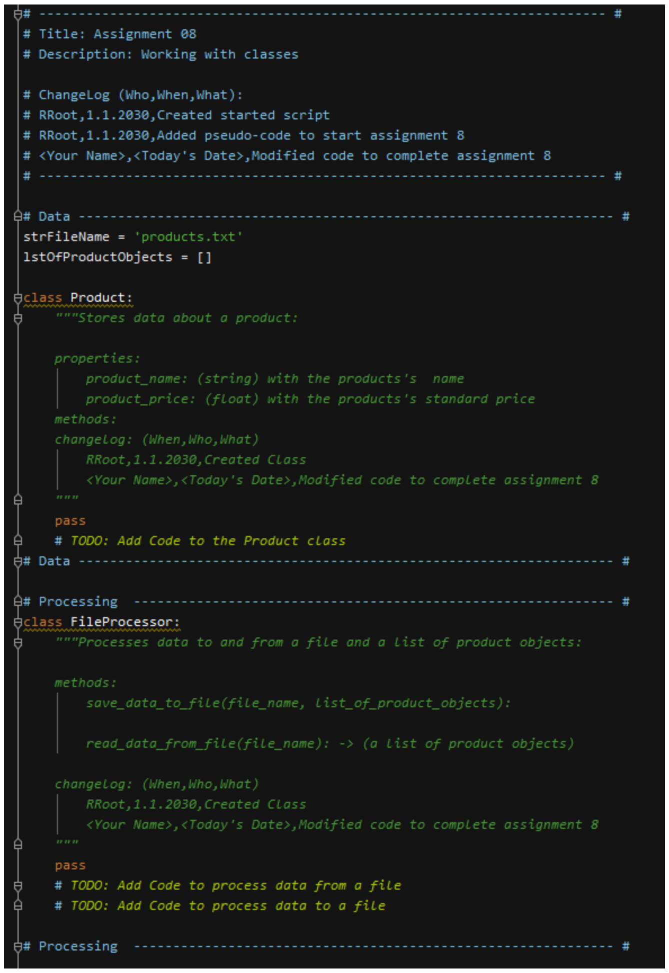
```
class MyClassName(MyBaseClassName):

    # -- Fields --
    # -- Constructor --
    #     -- Attributes --
    # -- Properties --
    # -- Methods --
```

Fields are the data member of a class. Constructors are a special method that are automatically run when you create an object. They are often used to set the initial values of the Field data. In Python they use the notation of double underscore(“duder”) name of “__init__”. Opposite the constructor is the Destructor which is used to remove objects for memory. It uses the notation “__del__”. Within the constructor method you’ll see the keyword “self”. This refers to the data or functions found in an object instance, but no directly in the class. The next setup in the class is Attributes. These are “virtual” fields that hold internal data. Properties are functions used to manage fields or attributes. The last is Methods which are used to handle functions inside of the class.

Product and Price Python Program Using Classes

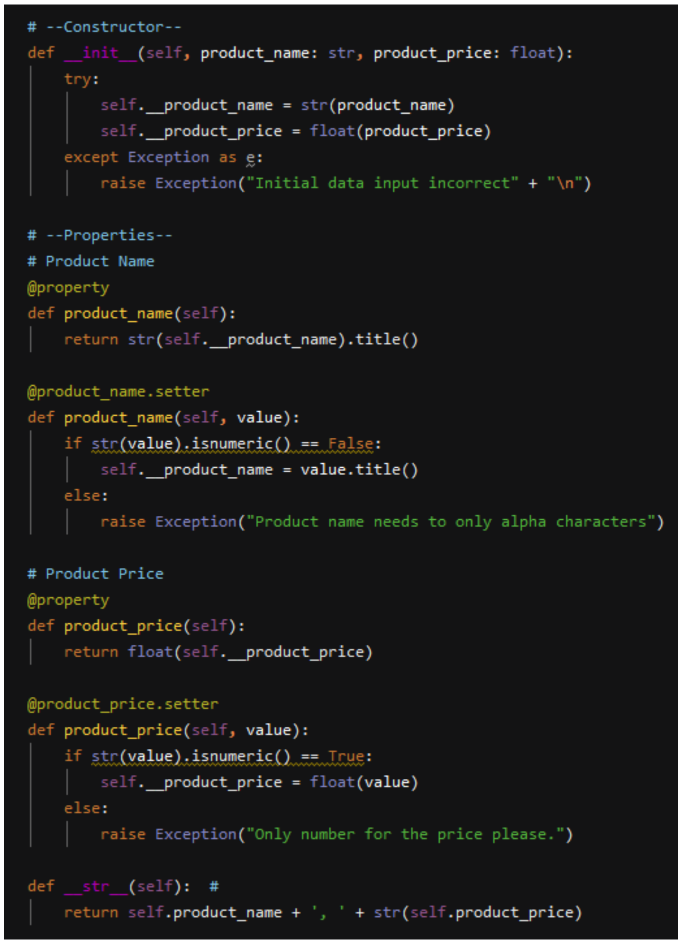
This program's problem statement is similar to the previous weeks but we now use classes and object to help organize the code.

To start out the program was given with a basic outline and structure that was a starting point.



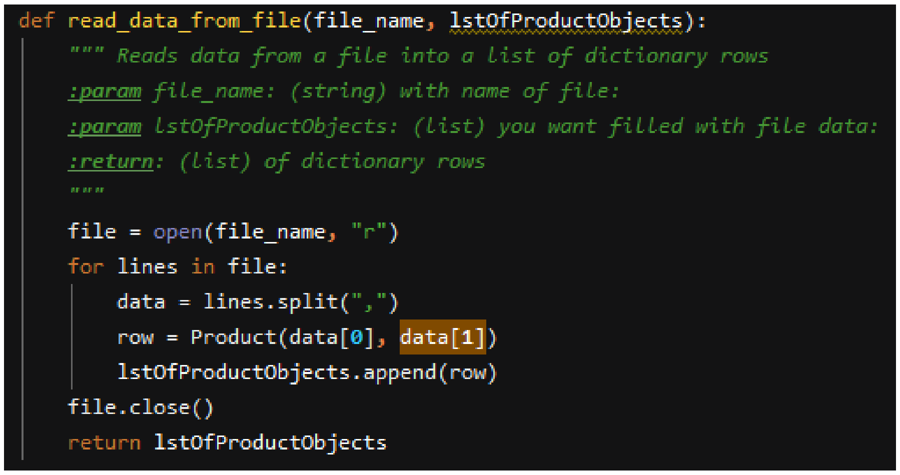
```
# ------------------------------------------------------------------------ #
# Title: Assignment 08
# Description: Working with classes

# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# RRoot,1.1.2030,Added pseudo-code to start assignment 8
# <Your Name>,<Today's Date>,Modified code to complete assignment 8
# ------------------------------------------------------------------------ #

# Data -------------------------------------------------------------------- #
strFileName = 'products.txt'
lstOfProductObjects = []

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
        <Your Name>,<Today's Date>,Modified code to complete assignment 8
    """
    pass
    # TODO: Add Code to the Product class
# Data -------------------------------------------------------------------- #

# Processing  ------------------------------------------------------------- #
class FileProcessor:
    """Processes data to and from a file and a list of product objects:

    methods:
        save_data_to_file(file_name, list_of_product_objects):

        read_data_from_file(file_name): -> (a list of product objects)

    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
        <Your Name>,<Today's Date>,Modified code to complete assignment 8
    """
    pass
    # TODO: Add Code to process data from a file
    # TODO: Add Code to process data to a file

# Processing  ------------------------------------------------------------- #
```

First section that I worked on what the “Product” class. This was the main addition to this week. I when through and created the constructor, attributes, properties and methods for the class.



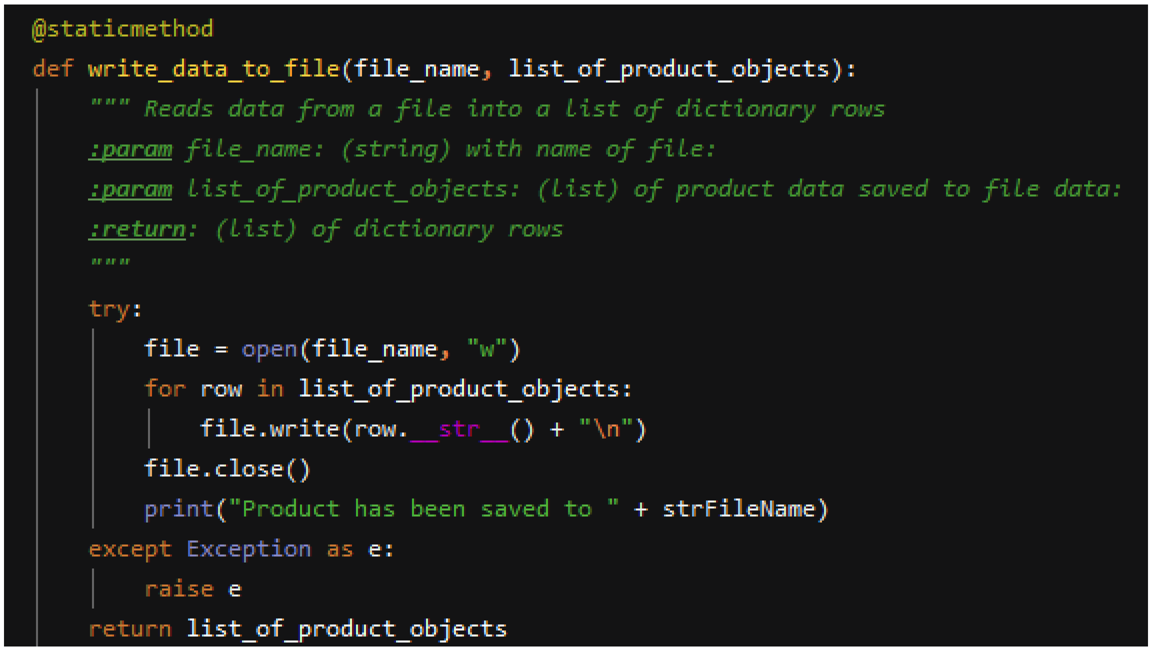
```python
# --Constructor--
def __init__(self, product_name: str, product_price: float):
    try:
        self.__product_name = str(product_name)
        self.__product_price = float(product_price)
    except Exception as e:
        raise Exception("Initial data input incorrect" + "\n")

# --Properties--
# Product Name
@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.title()
    else:
        raise Exception("Product name needs to only alpha characters")

# Product Price
@property
def product_price(self):
    return float(self.__product_price)

@product_price.setter
def product_price(self, value):
    if str(value).isnumeric() == True:
        self.__product_price = float(value)
    else:
        raise Exception("Only number for the price please.")

def __str__(self):  #
    return self.product_name + ', ' + str(self.product_price)
```

Next, I copied over some of my code for assignment 6 to read in the data from the file. Now that we are using objects, I had to change over how the data was being used in the for loop and calling the new product class in the process.



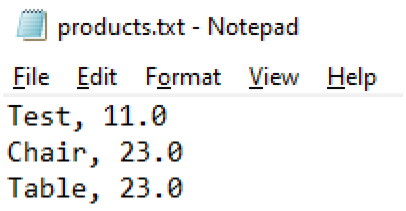
```python
def read_data_from_file(file_name, lstOfProductObjects):
    """ Reads data from a file into a list of dictionary rows
    :param file_name: (string) with name of file:
    :param lstOfProductObjects: (list) you want filled with file data:
    :return: (list) of dictionary rows
    """
    file = open(file_name, "r")
    for lines in file:
        data = lines.split(",")
        row = Product(data[0], data[1])
        lstOfProductObjects.append(row)
    file.close()
    return lstOfProductObjects
```

Then I went on to the write data to a file. I used the same process as reading gather an outline form my assignment 6 and modifying where needed.



```python
@staticmethod
def write_data_to_file(file_name, list_of_product_objects):
    """ Reads data from a file into a list of dictionary rows
    :param file_name: (string) with name of file:
    :param list_of_product_objects: (list) of product data saved to file data:
    :return: (list) of dictionary rows
    """
    try:
        file = open(file_name, "w")
        for row in list_of_product_objects:
            file.write(row.__str__() + "\n")
        file.close()
        print("Product has been saved to " + strFileName)
    except Exception as e:
        raise e
    return list_of_product_objects
```

Here is the output in the text file.

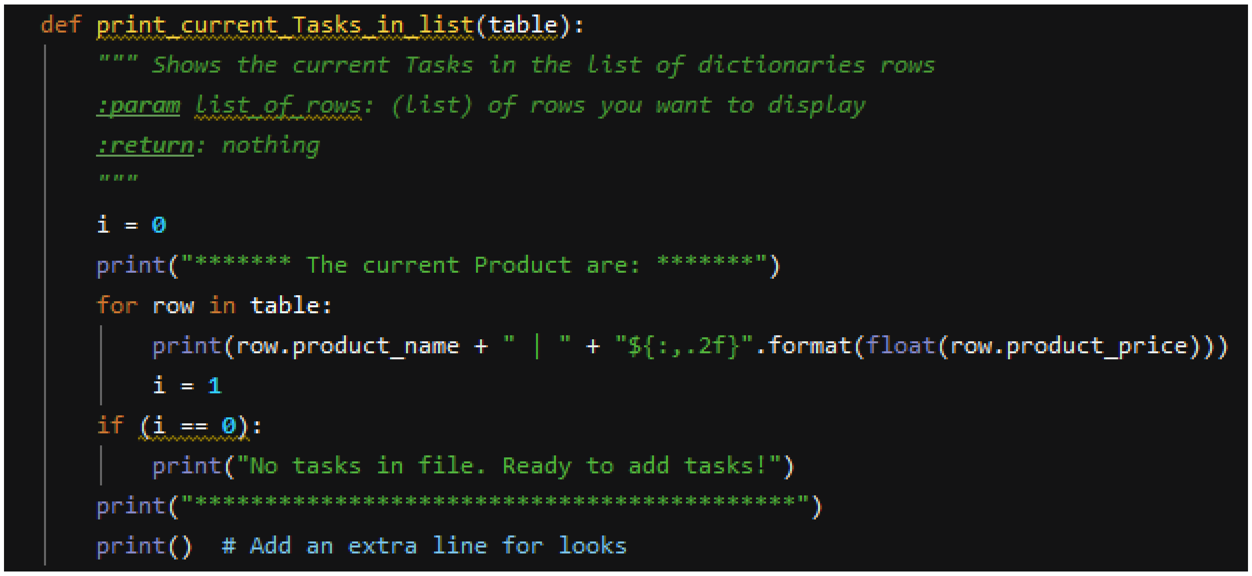


products.txt - Notepad

File Edit Format View Help

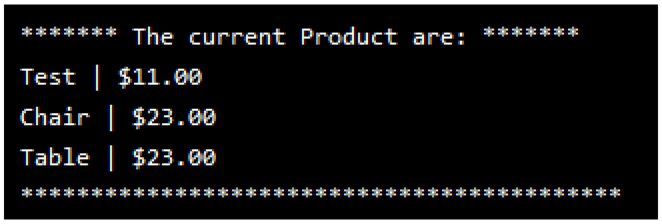
```
Test, 11.0
Chair, 23.0
Table, 23.0
```

The next function was printing the current list. This was a copy from assignment 6 by with update to all the product objects.



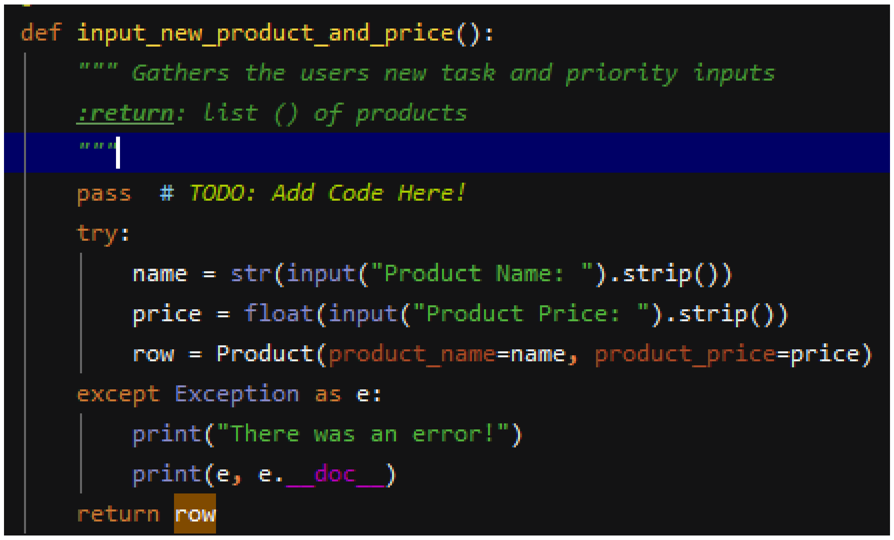
```python
def print_current_Tasks_in_list(table):
    """ Shows the current Tasks in the list of dictionaries rows
    :param list_of_rows: (list) of rows you want to display
    :return: nothing
    """
    i = 0
    print("******* The current Product are: *******")
    for row in table:
        print(row.product_name + " | " + "${:,.2f}".format(float(row.product_price)))
        i = 1
    if (i == 0):
        print("No tasks in file. Ready to add tasks!")
    print("********************************************")
    print()  # Add an extra line for looks
```

Here is an example of the print out from the text file.



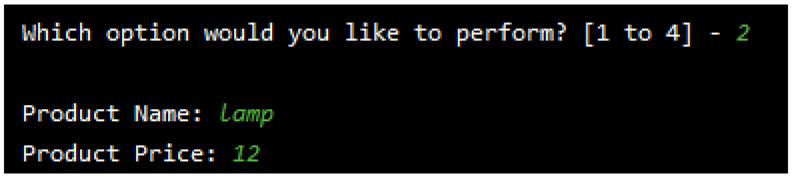
```
******* The current Product are: *******
Test | $11.00
Chair | $23.00
Table | $23.00
*******************************************
```

The next function was gathering the user's input. This one I had some trouble with but was able to figure out how to use the product class and store the objects in a list.



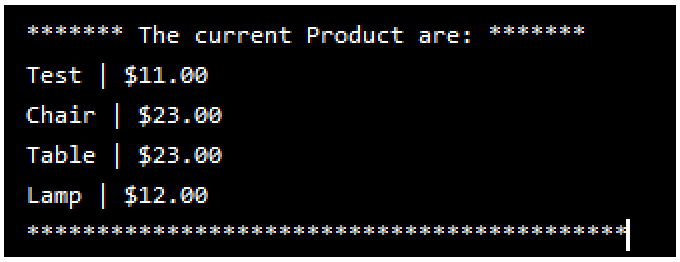
```python
def input_new_product_and_price():
    """ Gathers the users new task and priority inputs
    :return: list () of products
    """
    pass  # TODO: Add Code Here!
    try:
        name = str(input("Product Name: ").strip())
        price = float(input("Product Price: ").strip())
        row = Product(product_name=name, product_price=price)
    except Exception as e:
        print("There was an error!")
        print(e, e.__doc__)
    return row
```

Here is an example of data the use input.



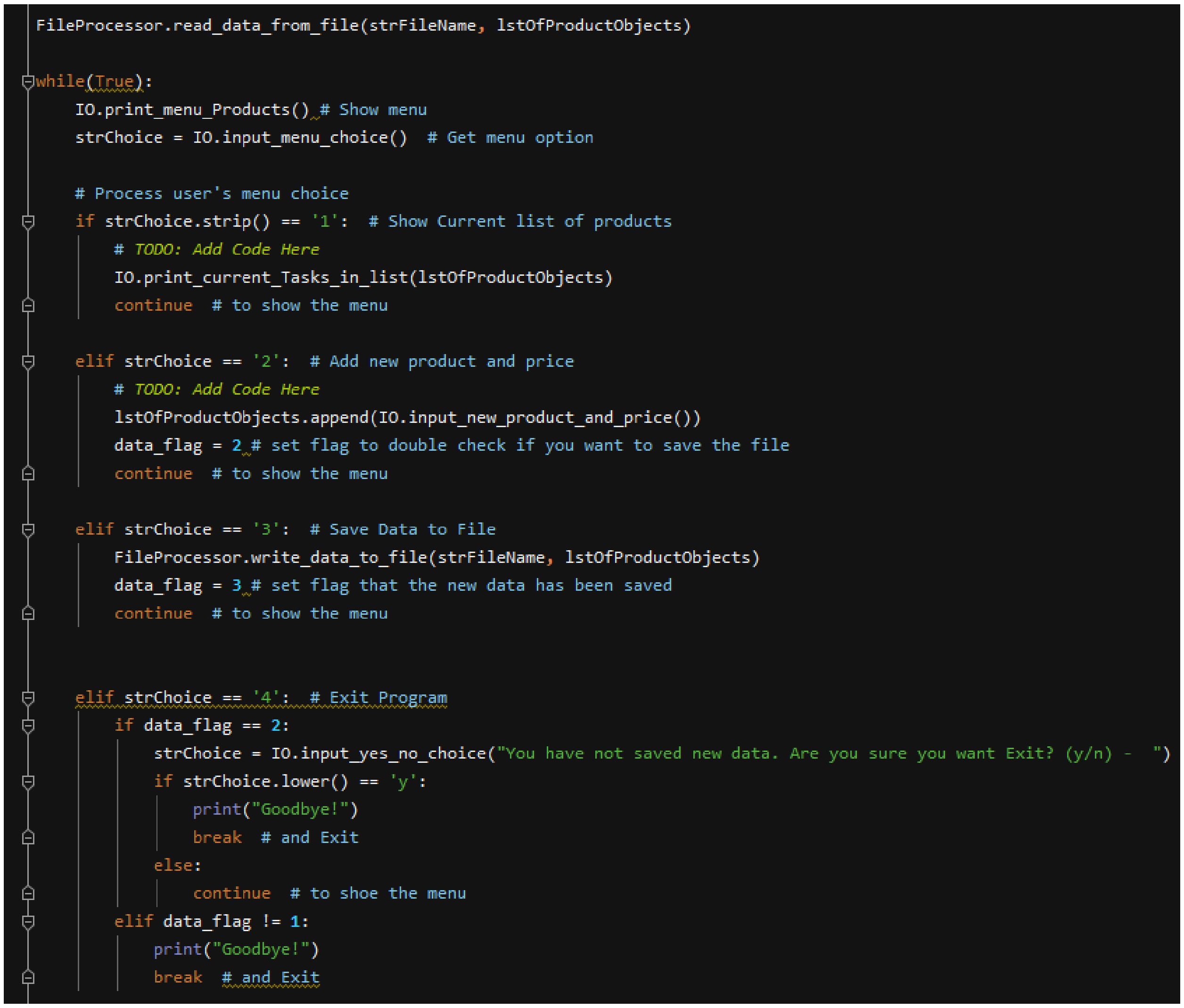
```
Which option would you like to perform? [1 to 4] - 2

Product Name: Lamp
Product Price: 12
```

Here is the output of the current table.



```
******* The current Product are: *******
Test | $11.00
Chair | $23.00
Table | $23.00
Lamp | $12.00
*******************************************
```

Finally, was the process of call all the functions. This code is very similar to assignment 6 with some modification for calling and using classes.



```
FileProcessor.read_data_from_file(strFileName, lstOfProductObjects)

while(True):
    IO.print_menu_Products() # Show menu
    strChoice = IO.input_menu_choice()  # Get menu option

    # Process user's menu choice
    if strChoice.strip() == '1':  # Show Current list of products
        # TODO: Add Code Here
        IO.print_current_Tasks_in_list(lstOfProductObjects)
        continue  # to show the menu

    elif strChoice == '2':  # Add new product and price
        # TODO: Add Code Here
        lstOfProductObjects.append(IO.input_new_product_and_price())
        data_flag = 2 # set flag to double check if you want to save the file
        continue  # to show the menu

    elif strChoice == '3':  # Save Data to File
        FileProcessor.write_data_to_file(strFileName, lstOfProductObjects)
        data_flag = 3 # set flag that the new data has been saved
        continue  # to show the menu


    elif strChoice == '4':  # Exit Program
        if data_flag == 2:
            strChoice = IO.input_yes_no_choice("You have not saved new data. Are you sure you want Exit? (y/n) -  ")
            if strChoice.lower() == 'y':
                print("Goodbye!")
                break  # and Exit
            else:
                continue  # to shoe the menu
        elif data_flag != 1:
            print("Goodbye!")
            break  # and Exit
```

Summary

In summary, I learned about objects and classes. I went through the process of how I created the classes and functions in product and price program. I found this assignment to be the most difficult so far. I really had trouble understand the concept of the classes. I really got stuck on the Product class part of the code. After doing some further reading and when professor Root said that “the objects were just a list” that really help it click for me. Overall, I’m happy with this assignment and the use of classes really shows the power of programming.