Irina Zubova

March 7, 2021

Foundations of Programming: Python

Assignment 08

https://github.com/i-zuzu/IntroToProg-Python-Mod08

Creating Script Using Custom Classes

Introduction

In this document I'll describe my steps to finish Assignment08. The task was to read and understand the pseudo-code in the starter code, then add code to make the application work and include error handling. I'll be using custom classes to organize functions and data and apply knowledge on how to write classes, work with Constructors, Attributes, Properties and Methods in classes, use properties to manage attribute data, create class-wide elements such as static methods.

Reviewing the Starter Code

Starter code contained 3 classes:

`class Product` - Stores data about a product
`class FileProcessor` - Processes data to and from a file and a list of product object
`class IO` - Performs Input and Output tasks

And I had to add code to allow the script to Load data from file into a list of product objects when script starts, show user a menu of options, Get user's menu option choice to:

1. Show user current data in the list of product objects
2. Let user add data to the list of product objects
3. Let user save current data to file and exit program

Parts of this script were quite similar to Assignment 06 and I applied knowledge and some functions from this Assignment to a new script.

Adding code to Classes

The docstring in the first class called `class Product` let me know I needed to create properties for both `prod_name` and `product_price` fields/attributes. It also showed me the types (string and float) for these fields/attributes:

```
"""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
    IZubova,3.7,2021,Modified code to complete assignment 8
"""
```

First I create the Constructor, special method that automatically runs when you create an object from the class and that used to set the initial values of Field data. Python Constructor's use the double underscore("duder") name of "`__init__`". Python automatically calls the "`__init__()`" method and passes any arguments you provide to the "`__init__()`" method each time you make a new object. Below is my Constructor

```
def __init__(self):
    self.__str_prod_name = ''
    self.__flt_prod_price = None
```

Then I work on the Properties for `str_prod_name` attribute. There're two Properties for each field/attribute, one for "getting" data and one for "setting data.

The one below is the Getter Property, that allows to format data (in my case it's `title ()`). `@property` directive indicates a getter function:

```
# -- Properties --
# prod_name
@property
def str_prod_name(self): # (getter or accessor)
    return str(self.__str_prod_name).title()   # Title case
```

Next one is Setter Properties that let you add code for both validation and error handling. If a value passed into the Properties parameter is valid, then it is assigned to the field or attribute. Setter must include the `@name_of_method.setter` directive, and the directive and function name must match. In my case `name_of_method` is str_prod_name and Setter property checks that the str_prod_name value is not the number.

```
@str_prod_name.setter  # The NAME MUST MATCH the property's!
def str_prod_name(self, value):  # (setter or mutator)
    if str(value).isnumeric() == False:
        self.__str_prod_name = value.strip()
    else:
        raise Exception("Names cannot be numbers")
```

Code for Properties for prod_price field is below. There's no formatting in the Getter propertes. And the Setter Properties validate that the flt_prod_price is float. I also set exception handling here.

```
@property
def flt_prod_price(self):  # (getter or accessor)
    return self.__flt_prod_price  # Title case

@flt_prod_price.setter  # The NAME MUST MATCH the property's!
def flt_prod_price(self, value):  # (setter or mutator)
    try:
        self.__flt_prod_price = float(value)
    except ValueError:
        raise Exception("Price should be a float number, instead {value}
provided")
```

```
def __str__(self):
    return self.__str_prod_name + " " + str(self.__flt_prod_price)
```

The next class id file File Processor that per its docstring should have 2 methods: `save_data_to_file` and `read_data_from_file`. Methods are just functions inside of a class.

These 2 methods have `@staticmethod` directive because they're called directly from the class, without making an object:

```
@staticmethod
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 rows
    """
    lstOfProductObjects.clear()  # clear current data
    with open(file_name, "r") as file:
        for line in file:
            prod, price = line.split(",")
            objNewProd = Product()
            objNewProd.str_prod_name =  prod
            objNewProd.flt_prod_price =  price
            lstOfProductObjects.append(objNewProd)
    return lstOfProductObjects, 'Success'

@staticmethod
def save_data_to_file(file_name, lstOfProductObjects):
    """ Saves data to a file from list of dictionary rows

        :param file_name: (string) with name of file:
        :param lstOfProductObjects: (list) you want write to file:
        :return: (list) of rows
    """
    with open(file_name, "w") as file:
        for row in lstOfProductObjects:
            file.write(row.str_prod_name + ", " + str(row.flt_prod_price)+
'\n')
    return lstOfProductObjects, 'Success'
```

I added additional `class Processor` to create method to add data from the user to the list. I addended docstring with the description for both the class and the method. Also the error handling I applied is used in case the data added in the wrong format.

Below is the code for Processor Class:

```
class Processor:
    """Processes product and price data to add it to the list:

    methods:
        add_data_to_list(lstOfProductObjects, prod, price):

    changelog: (When,Who,What)
        IZubova,3.7,2021,Modified code to complete assignment 8
    """
@staticmethod
```

```
def add_data_to_list(lstOfProductObjects, prod, price):
    """ 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) you want filled with file
data:
        :return: (string) Success/Fail
    """
    try:
        objNewProd = Product()
        objNewProd.str_prod_name = prod
        objNewProd.flt_prod_price = price
        lstOfProductObjects.append(objNewProd)
    except Exception as e:
        # print(e)
        print('Bad format.')
        return 'Fail'
    return 'Success'
```

One more class called `class  IO` has several methods to perform Ipnut and out put tasks: `print_menu_Tasks()`, `input_menu_choice()`, `print_current_Tasks_in_list()`, `input_new_prod_and_price()`, `input_new_prod_and_price()`, `input_press_to_continue(optional_message='')`, `input_yes_no_choice(message)`.

Below is example of the method from `class  IO` that gets input from the user on product name and price:

```
@staticmethod
def input_new_prod_and_price():
    """ Gets the name of a new task and priority to add to the list from a
user
        :return: string, string
    """
    prod = input("Enter the product: ")
    price = input("Enter the price: ")
    return prod, price
```

Now when the content in all clases created I needed to put together the main body of the script and call the methods in classes defined in the previous sections to perform all actions rom the menu of options.

When the script runs it loads and displays existing data in the text file. I also created while loop to allow user pick menu choices until he/she decides to exit. I specify the class name and the method name in the code for this action:

```
# Step 1 - When the program starts, Load data from strFileName.
FileProcessor.read_data_from_file(strFileName, lstOfProductObjects)
IO.print_current_Tasks_in_list(lstOfProductObjects)  # Show current data in
the list

while(True):
    # Step 2 - Display a menu of choices to the user
    IO.print_menu_Tasks()  # Shows menu
    strChoice = IO.input_menu_choice()  # Get menu option
```

If the user chooses option 1, then the method `print_current_Tasks_in_list` from `IO class` is used:

```
# Step 3 Show current data
if strChoice.strip() == '1':  # Add a new pr
    IO.print_current_Tasks_in_list(lstOfProductObjects)  # Show current data
in the list/table
    continue
```

Then I the user picks option 2 then 2 methods are called: `input_new_prod_and_price()` in IO class and `add_data_to_list` in the `Processor class`. If the values for prod and price entered by the user match all criteria defined in Getter and Setter `Properties` (`class Product`) then the data for price and product added to the list, if not, the user gets notification about the wrong format of the data.

```
# Step 4 - Add new product and price
if strChoice.strip() == '2':  # Add a new product and price
    prod, price = IO.input_new_prod_and_price()
    result = Processor.add_data_to_list(lstOfProductObjects, prod, price)
    if result == 'Success':
        strStatus = "\n Product added.\n"
    else:
        strStatus = '\n Product was not added.\n'
    IO.input_press_to_continue(strStatus)
    continue  # to show the menu
```

The last option in the menu is option 3 and if the user picks this option then the program confirms user's choice to save the data using `input_yes_no_choice` method in IO class. If the user confirms (enters 'y') then the new data entered by the user is saved to the file because `save_data_to_file` method is called in the `FileProcessor` class and the while loop breaks. If the user doesn't want to save the data then the program just exits without saving the data.

Below is the code for option 3:

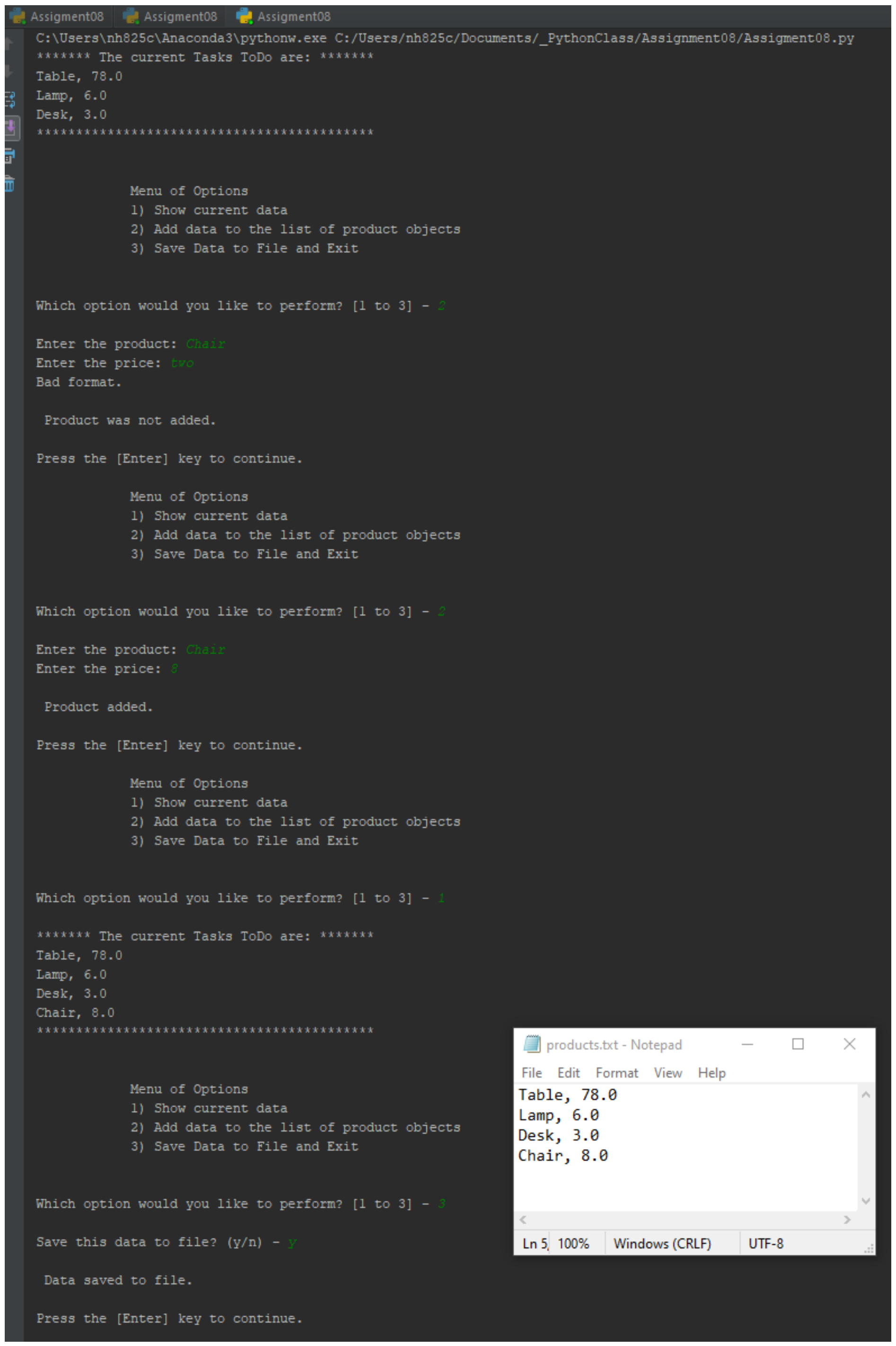
```
# Step 5 - Save product and price and exit
if strChoice == '3':              # Save Data to File
    strChoice = IO.input_yes_no_choice("Save this data to file? (y/n) - ")
    if strChoice.lower() == "y":
        FileProcessor.save_data_to_file(strFileName, lstOfProductObjects)
        strStatus = "\n Data saved to file.\n"
        IO.input_press_to_continue(strStatus)
        break
    else:
        IO.input_press_to_continue("Save Cancelled!")
        break
```

Final code for the script with Custom Classes is below:

```
# ------------------------------------------------------------------------ #
# 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
# IZubova,3.7.2021,Modified code to complete assignment 8
# ------------------------------------------------------------------------ #

# Data -------------------------------------------------------------------- #
strFileName = 'products.txt'
lstOfProductObjects = []
strChoice = '' # Captures the user option selection
strStatus = '' # Captures the status of the processing functions

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
        IZubova,3.7,2021,Modified code to complete assignment 8
    """

    #-- Constructor --
    def __init__(self):
        self.__str_prod_name = ''
        self.__flt_prod_price = None

    # -- Properties --
    # prod_name
    @property
    def str_prod_name(self): # (getter or accessor)
        return str(self.__str_prod_name).title()  # Title case

    @str_prod_name.setter  # The NAME MUST MATCH the property's!
    def str_prod_name(self, value):  # (setter or mutator)
        if str(value).isnumeric() == False:
            self.__str_prod_name = value.strip()
        else:
            raise Exception("Names cannot be numbers")

                # prod_price
    @property
    def flt_prod_price(self):  # (getter or accessor)
        return self.__flt_prod_price  # Title case

    @flt_prod_price.setter  # The NAME MUST MATCH the property's!
    def flt_prod_price(self, value):  # (setter or mutator)
        try:
            self.__flt_prod_price = float(value)
        except ValueError:
            raise Exception("Price should be a float number, instead {value}
provided")

    # def __str__(self):
    #     return self.__str_prod_name + " " + str(self.__flt_prod_price)

# 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
        IZubova,3.7,2021,Modified code to complete assignment 8
    """

    @staticmethod
    def read_data_from_file(file_name, lstOfProductObjects):
        """ Reads data from a file into a list of dictionary rows
```

```python
            :param file_name: (string) with name of file:
            :param lstOfProductObjects: (list) you want filled with file data:
            :return: (list) of rows
        """
        lstOfProductObjects.clear()  # clear current data
        try:
            with open(file_name, "r") as file:
                for line in file:
                    prod, price = line.split(",")
                    objNewProd = Product()
                    objNewProd.str_prod_name =  prod
                    objNewProd.flt_prod_price =  price
                    lstOfProductObjects.append(objNewProd)
        except (FileNotFoundError):
            print ("\n Text file is not created\n")
        return lstOfProductObjects, 'Success'

    @staticmethod
    def save_data_to_file(file_name, lstOfProductObjects):
        """ Saves data to a file from list of dictionary rows

            :param file_name: (string) with name of file:
            :param lstOfProductObjects: (list) you want write to file:
            :return: (list) of rows
        """
        with open(file_name, "w") as file:
            for row in lstOfProductObjects:
                file.write(row.str_prod_name + ", " + str(row.flt_prod_price)+ '\n')
        return lstOfProductObjects, 'Success'

# Processing  ------------------------------------------------------------- #
class Processor:
    """Processes product and price data to add it to the list:

    methods:
        add_data_to_list(lstOfProductObjects, prod, price):

    changelog: (When,Who,What)
        IZubova,3.7,2021,Modified code to complete assignment 8
    """

    @staticmethod
    def add_data_to_list(lstOfProductObjects, prod, price):
        """ 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) you want filled with file data:
            :return: (string) Success/Fail
        """
        try:
            objNewProd = Product()
            objNewProd.str_prod_name = prod
            objNewProd.flt_prod_price = price
            lstOfProductObjects.append(objNewProd)
        except Exception as e:
            # print(e)
            print('Bad format.')
            return 'Fail'
        return 'Success'

# Presentation (Input/Output)  -------------------------------------------- #
class IO:
    """Performs Input and Output tasks:

       methods:
           print_menu_Tasks():
           input_menu_choice():
           print_current_Tasks_in_list():
           input_new_prod_and_price():
           input_press_to_continue(optional_message=''):
           input_yes_no_choice(message)

       changelog: (When,Who,What)
           IZubova,3.7,2021,Modified code to complete assignment 8
       """

    @staticmethod
    def print_menu_Tasks():
        """  Display a menu of choices to the user
            :return: nothing
        """
        print('''
            Menu of Options
            1) Show current data
            2) Add data to the list of product objects
            3) Save Data to File and Exit
            ''')
        print()  # Add an extra line for looks

    @staticmethod
    def input_menu_choice():
        """ Gets the menu choice from a user
            :return: string
        """
        choice = str(input("Which option would you like to perform? [1 to 3] -
")).strip()
        print()  # Add an extra line for looks
        return choice

    @staticmethod
    def print_current_Tasks_in_list(lstOfProductObjects):
        """ Shows the current Tasks in the list of dictionaries rows

            :param list_of_product_objects: (list) of rows you want to display
            :return: nothing
        """
        print("******* The current Tasks ToDo are: *******")
        for row in lstOfProductObjects:
            print(row.str_prod_name + ", " + str(row.flt_prod_price))
        print("*******************************************")
        print()  # Add an extra line for looks

    @staticmethod
    def input_new_prod_and_price():
        """ Gets the name of a new task and priority to add to the list from a user
            :return: string, string
        """
        prod = input("Enter the product: ")
        price = input("Enter the price: ")
        return prod, price

    @staticmethod
    def input_press_to_continue(optional_message=''):
        """ Pause program and show a message before continuing
        :param optional_message:  An optional message you want to display
        :return: nothing
        """
        print(optional_message)
        input('Press the [Enter] key to continue.')

    @staticmethod
    def input_yes_no_choice(message):
        """ Gets a yes or no choice from the user
        :return: string
        """
        return str(input(message)).strip().lower()

# Presentation (Input/Output)  -------------------------------------------- #

# Main Body of Script  ---------------------------------------------------- #
# Load data from file into a list of product objects when script starts
# Show user a menu of options
# Get user's menu option choice
    # Show user current data in the list of product objects
    # Let user add data to the list of product objects
    # let user save current data to file and exit program

# Main Body of Script  ---------------------------------------------------- #

# Step 1 - When the program starts, Load data from strFileName.
FileProcessor.read_data_from_file(strFileName, lstOfProductObjects)
IO.print_current_Tasks_in_list(lstOfProductObjects)  # Show current data in the list

while(True):
    # Step 2 - Display a menu of choices to the user
    IO.print_menu_Tasks()  # Shows menu
    strChoice = IO.input_menu_choice()  # Get menu option

    # Step 3 Show current data
    if strChoice.strip() == '1':  # Print current data
        IO.print_current_Tasks_in_list(lstOfProductObjects)  # Show current data in
the list/table
        continue

    # Step 4 - Add new product and price
    if strChoice.strip() == '2':  # Add a new product and price
        prod, price = IO.input_new_prod_and_price()
        result = Processor.add_data_to_list(lstOfProductObjects, prod, price)
        if result == 'Success':
            strStatus = "\n Product added.\n"
        else:
            strStatus = '\n Product was not added.\n'
        IO.input_press_to_continue(strStatus)
        continue  # to show the menu

    # Step 5 - Save product and price and exit
    if strChoice == '3':              # Save Data to File
        strChoice = IO.input_yes_no_choice("Save this data to file? (y/n) - ")
        if strChoice.lower() == "y":
            FileProcessor.save_data_to_file(strFileName, lstOfProductObjects)
            strStatus = "\n Data saved to file.\n"
            IO.input_press_to_continue(strStatus)
            break
        else:
            IO.input_press_to_continue("Save Cancelled!")
            break
print('\n Good bye!')
```

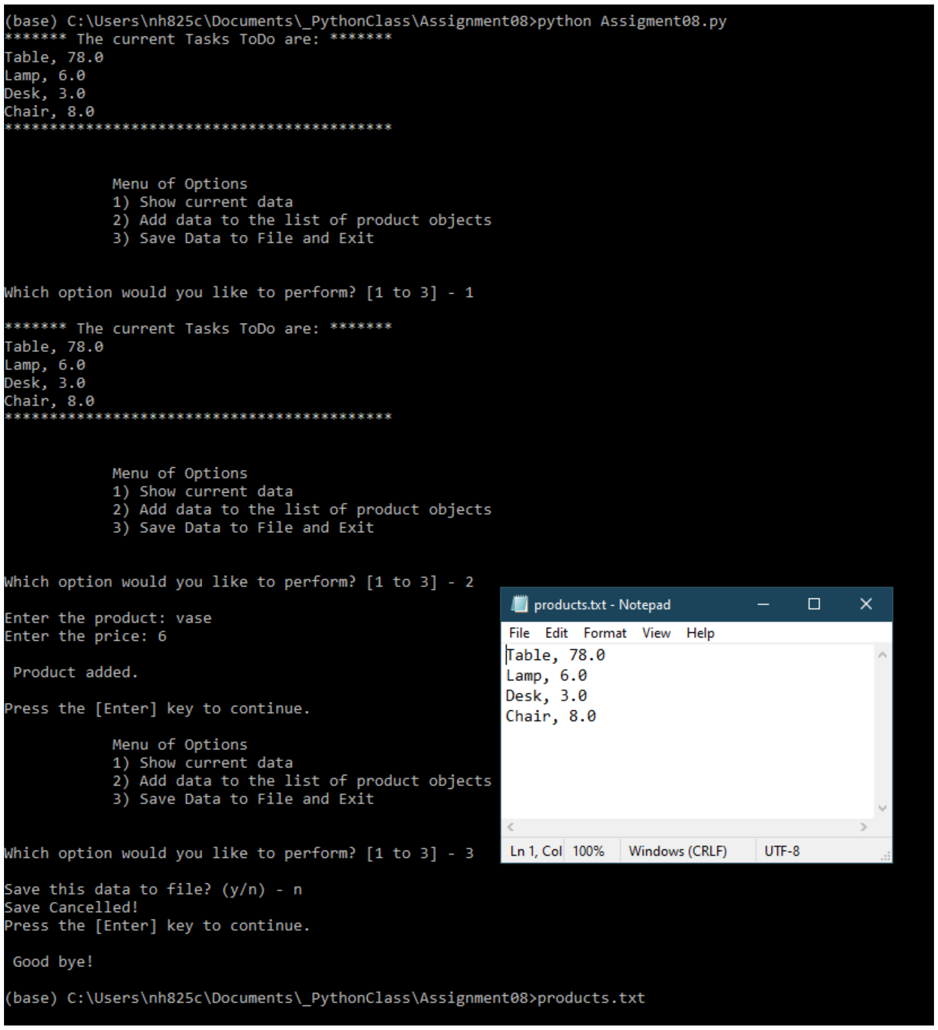
Figure 1 shows the script running in PyCharm:



```
C:\Users\nh825c\Anaconda3\pythonw.exe C:/Users/nh825c/Documents/_PythonClass/Assignment08/Assigment08.py
******* The current Tasks ToDo are: *******
Table, 78.0
Lamp, 6.0
Desk, 3.0
*******************************************



            Menu of Options
            1) Show current data
            2) Add data to the list of product objects
            3) Save Data to File and Exit


Which option would you like to perform? [1 to 3] - 2

Enter the product: Chair
Enter the price: two
Bad format.

 Product was not added.

Press the [Enter] key to continue.

            Menu of Options
            1) Show current data
            2) Add data to the list of product objects
            3) Save Data to File and Exit


Which option would you like to perform? [1 to 3] - 2

Enter the product: Chair
Enter the price: 8

 Product added.

Press the [Enter] key to continue.

            Menu of Options
            1) Show current data
            2) Add data to the list of product objects
            3) Save Data to File and Exit


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

******* The current Tasks ToDo are: *******
Table, 78.0
Lamp, 6.0
Desk, 3.0
Chair, 8.0
*******************************************



            Menu of Options
            1) Show current data
            2) Add data to the list of product objects
            3) Save Data to File and Exit


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

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

 Data saved to file.

Press the [Enter] key to continue.
```

```
Table, 78.0
Lamp, 6.0
Desk, 3.0
Chair, 8.0
```

Figure1. Output of the script after running in PyCharm



```
(base) C:\Users\nh825c\Documents\_PythonClass\Assignment08>python Assigment08.py
******* The current Tasks ToDo are: *******
Table, 78.0
Lamp, 6.0
Desk, 3.0
Chair, 8.0
*******************************************


            Menu of Options
            1) Show current data
            2) Add data to the list of product objects
            3) Save Data to File and Exit


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

******* The current Tasks ToDo are: *******
Table, 78.0
Lamp, 6.0
Desk, 3.0
Chair, 8.0
*******************************************


            Menu of Options
            1) Show current data
            2) Add data to the list of product objects
            3) Save Data to File and Exit


Which option would you like to perform? [1 to 3] - 2

Enter the product: vase
Enter the price: 6

 Product added.

Press the [Enter] key to continue.

            Menu of Options
            1) Show current data
            2) Add data to the list of product objects
            3) Save Data to File and Exit


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

Save this data to file? (y/n) - n
Save Cancelled!
Press the [Enter] key to continue.

 Good bye!

(base) C:\Users\nh825c\Documents\_PythonClass\Assignment08>products.txt
```

products.txt - Notepad
File Edit Format View Help
Ln 1, Col 100% Windows (CRLF) UTF-8

```
Table, 78.0
Lamp, 6.0
Desk, 3.0
Chair, 8.0
```

Figure2. Script running in Terminal

Summary

In this assignment I practiced in creating the code with custom classes. I learned how to crate and use different components of the class like Constructor, Properties, Fields, Methods, Docstring. I also learned about Object-oriented programming (OOP), a methodology used in the software industry that has the object as a basic building block.