Thomas Farmer

3-7-2022

Foundations Of Programming

Assignment08

https://github.com/MtnWolf82/IntroToProg-Python-Mod08

# Working With Classes

## INTRODUCTION

This week's assignment saw fit to put everything we've learned so far together in one place. This time around we're introduced to classes. Classes allow the programmer to define attributes and methods. We can then create objects, based off a class, to be used where needed within the program. While the interface of this assignment is similar to that of our prior assignment 6, the behind-the-scenes is quite different.

## THE PROCESS –

1. Open a new PyCharm file and enter your program's header comments. An example of this assignment's comments can be seen 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
# TFarmer,3.6.2022,Modified code to complete assignment 8
# ------------------------------------------------------------------------ #
```

2. We'll start out by creating our first class – “Product”. This class will contain the functions and methods to work with our data. We first create our docstring. Enter the code as shown below.

```
# 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
        TFarmer,3.6.2022,Modified code to complete assignment 8
    """
```

3. As part of the Product class, we create a constructor to define our hidden product_name and product_price attributes. Enter the code as shown below.

```
    # -- Constructor --
    # product info
    def __init__(self, product_name, product_price):
        # -- Attributes --
        self.__product_name = product_name
        self.__product_price = product_price
```

4. Next we will work create several functions that will allow for us to work with the product name and product price attributes. Enter the code as shown below.

```
    # -- Properties --
    # product name
    @property
    def product_name(self):
        return str(self.__product_name).title()

    @product_name.setter
    def product_name(self, value):
        self.__product_name = value

    # product price
    @property
    def product_price(self):
        return float(self.__product_price)

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

5. Last but not least, we'll wrap up our class by adding a couple method functions. Enter the code shown below.

```
    # -- Methods --
    def to_string(self):
        return self.__str__()

    def __str__(self):
        return self.product_name + " , " + str(self.product_price)

# Data ---------------------------------------------------------------- #
```

6. The next class - “FileProcessor” - will be used to actually process data from our text file and our list of product objects. Be sure to enter an associated docstring. Enter the code shown below.

```
# 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): -> (boolean
        status)

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

    changelog: (When,Who,What)
        RRoot,1.1.2030,Created Class
        TFarmer,3.6.2022,Modified code to complete assignment 8
    """
```

7. The first method of our FileProcessor class will be used to save data to our text file. Enter the code shown below.

```
    @staticmethod
    def save_data_to_file(file_name, list_of_product_objects):
        success_status = False
        try:
            file = open(file_name, "w")
            for product in list_of_product_objects:
                file.write(product.__str__() + "\n")
            file.close()
            success_status = True
            print("Data has been saved to " + strFileName)
        except Exception as e:
            print("There was an error!")
            print(e, e.__doc__, type(e), sep="\n")
        return success_status
```

8. The second method of our FileProcessor class is used to read data from our text file. Enter the code shown below.

```
    @staticmethod
    def read_data_from_file(file_name):
        list_of_product_rows = []
        try:
            file = open(file_name, "r")
            for line in file:
                data = line.split(" , ")
                row = Product(data[0], data[1])
                list_of_product_rows.append(row)
            file.close()
        except Exception as e:
            print("There was an error!")
            print(e, e.__doc__, type(e), sep="\n")
        return list_of_product_rows

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

9. The last class we'll create is “IO”. It will work with our user provided inputs and provide outputs. Like before, the docstring should be entered first. Enter the code shown below.

```
# Presentation (Input/Output)  -------------------------------------------- #
class IO:
    """Interface to allow current file data to be viewed, in addition
     to obtaining product name and price data from the user:

    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
        TFarmer,3.6.2022,Modified code to complete assignment 8
    """
```

10. As in prior assignments, we'll create a menu for the user to interact with. However, it is now created as a function. Enter the code shown below.

```
    @staticmethod
    def print_menu_items():
        print("""
        Menu Options:
        1. Show Current Data
        2. Add A New Product
        3. Save Data To File
        4. Exit The Program
        """)
        print()
```

11. There are three additional functions left to create. Each one is based around the user's selection from the menu defined in Step 10. This will wrap up our three distinct classes. Enter the code shown below.

```
    @staticmethod
    def input_menu_options():
        choice = str(input("Please select an option: [1-4] - "))
        print()
        return choice

    @staticmethod
    def print_current_list_items(list_of_rows):
        print("***** Current Products: *****")
        for row in list_of_rows:
            print(row.product_name + " | " + str(row.product_price))
        print("******************************")

    @staticmethod
    def add_product_data():
        name = input("Enter the product name: ")
        price = float(input("Enter the product price: "))
        print()
        prodInfo = Product(product_name=name, product_price=price)
        print(name + " - added to your product list.")
        return prodInfo

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

12. All that's left is the main body of our script which utilizes functions in our previously defined classes to perform the user selected action. Enter the code shown below.

```
# Main Body of Script  ------------------------------------------------ #
lstOfProductObjects = FileProcessor.read_data_from_file(strFileName)

while True:
    IO.print_menu_items()
    choice = IO.input_menu_options()
    if choice.strip() == "1":
        IO.print_current_list_items(lstOfProductObjects)
    elif choice.strip() == "2":
        lstOfProductObjects.append(IO.add_product_data())
    elif choice.strip() == "3":
        FileProcessor.save_data_to_file(strFileName, lstOfProductObjects)
    elif choice.strip() == "4":
        break

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

**SUMMARY**

This was quite the culmination of lessons. I found it very useful seeing a lot of the prior assignments lessons being called back. It helped to reinforce much of those lessons, as well as expose those parts I needed further practice with. Looking at the difference between the script's “main body” on this assignment, versus those of prior assignments is pretty enlightening and definitely shows how much more efficient the use of classes and functions are in larger, more complicated programs.