Fatima Sharif
March 8, 2022
Foundations of Programming: Python
Assignment08
https://github.com/Fmsharif3/IntroToProg-Python-Mod08

Working With Classes

Introduction: In this paper I will explain to you the steps I took to create a script with three classes using the Standard Class Pattern illustrated below.

Starter Script:

To start, I opened PyCharm and created a new project titled, "Assignment08." I then created a python file in my project titled, "Assignment08." While using the starter script provided by my instructor, I edited my program header and viewed the list of declared variables (Shown in the figure below).

```
# ------------------------------------------------------------------------ #
# Title: Assignment 08
# Description: Working with classes
# ChangeLog (Who,When,What):
# FSharif,3.6.2022,Created started script
# FSharif,3.6.2022,Added pseudo-code to start assignment 8
# <Fatima Sharif>,<2.6.2022>,Modified code to complete assignment 8
# ------------------------------------------------------------------------ #

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

The Standard Class Pattern:

class MyClassName(MyBaseClassName):

-- **Fields** – "varibles"
-- **Constructor** --
-- **Attributes** --
-- **Properties** – "special function"
-- **Methods** --

Class 1: Product
Refencing the standard class pattern shown above, I then began to create the Product Class.

Step One: Fields

1. Create a class called Product and add both strProductName and strProductPrice fields to Product Class (Shown in Figure 1)

Step Two/Three: Constructors & Attributes

2. Define the two attributes: product name and product price.
3. Use the constructor parameter values to set product_name and product_price attributes. (Shown in Figure 1) define the two main attributes of the code, product price and product name.

```python
class Product:
    # --Fields--
    strProductName = ""
    fltProductPrice = ""

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

Figure 1:Fields, Constructors and Attributes for the Product Class

Step Four: Properties

4. Modify the constructor's attribute name to __product_name and __product_price and Create a getter and setter Property for both the __product_name and __product_price attributes. (Shown in figure 2)

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

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

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

    @product_price.setter
    def product_price(self, value):
        if str(value).isnumeric():
            self.__product_price = float(value)
        else:
            raise Exception("Prices must be numbers")
```

Figure 2: Properties for the Product Class

Step Five: Methods

5. Add code to the method to return both the first_name and last_name with a comma separator. (Shown in Figure 3)

```python
    # -- methods --
    def to_string(self):
        """ alias of __str__(), converts product data to string """
        return self.__str__()

    def __str__(self):
        """ Converts product data to string """
        return self.product_name + "," + str(self.product_price)
```

Figure 3: Methods for the Product Class

Step 6: Testing the Product Class

6. Test the code by creating an object instance, setting the properties, then using print function. (Shown in Figure 4)

```python
# Test Code
Obj1 = Product('ProdA', 9.99)
print(Obj1.product_name, Obj1.product_price)
```

```
Run: Assignment.08
/Users/fatimasharif/Documents/_PythonClass/Assignment08/bin/Python /Users/fatimasharif/PycharmProjects/Assignment08/Assignment.08.py
ProdA 9.99

Process finished with exit code 0
```

Figure 4: Output of testing the Product Class

```python
# Processing  ------------------------------------------------------------ #
class FileProcessor:

    # TODO: Add Code to process data from a file
    @staticmethod
    def save_data_to_file(file_name: str, list_of_product_objects: list):
        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
        except Exception as e:
            print("There was a general error!")
            print(e, e.__doc__, type(e), sep='\n')
        return success_status
    # TODO: Add Code to process data to a file
    @staticmethod
    def read_data_from_file(file_name: str):
        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 a general error!")
            print(e, e.__doc__, type(e), sep='\n')
        return list_of_product_rows
```

Figure 5: File Processor Class

Class 2: File Processor

Then, I created methods for the File Processor Class. The methods created were two. One to read data from a file and add it to a list of lists and the other to save current data to a text file. (Shown in Figure 5)

Class 3: IO

Next, I created four methods for the IO Class. The first method created was to print a menu to the user. The second method is to get input from the user. The third method created was to display current data in text file to the user. And the last method was for requesting new data from the user. (Shown in Figure 6)

```python

class IO:  # TODO: Add docstring
    # TODO: Add code to show menu to user
    @staticmethod
    def print_menu_items():
        """ Print a menu of choices to the user """
        print('''
    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Save Data to File
    4) Exit Program
    ''')

    print()  # Add an extra line for looks

    # TODO: Add code to get user's choice
    @staticmethod
    def input_menu_choice():

        choice = str(input("Which option would you like to perform? [1 to 4] - ")).strip()
        print()  # Add an extra line for looks
        return choice
    # TODO: Add code to show the current data from the file to user
    @staticmethod
    def print_current_list_items(list_of_rows: list):

        print("******* The current items products are: *******")
        for row in list_of_rows:
            print(row.product_name + " (" + str(row.product_price) + ")")
        print("*******************************************")
```

Figure 6: IO Class

Main Body Script

Lastly, I used all the classes created. (Shown in Figure 7)

```python
# Main Body of Script  ------------------------------------------------------ #
try:
    lstOfProductObjects = FileProcessor.read_data_from_file(strFileName)
    while True:
        IO.print_menu_items()_# Show user a menu of options
        strChoice = IO.input_menu_choice()_# Get user's menu option choice
        if strChoice.strip() == '1':
            IO.print_current_list_items(lstOfProductObjects)_# Show user current data in the list of product objects
            continue
        elif strChoice.strip() == '2':
            # Let user add data to the list of product objects
            lstOfProductObjects.append(IO.input_product_data())
            continue
        elif strChoice.strip() == '3':
            # let user save current data to file and exit program
            FileProcessor.save_data_to_file(strFileName, lstOfProductObjects)
            continue
        elif strChoice.strip() == '4':
            break
except Exception as e:
    print("There was an error! Check file permissions.")
    print(e, e.__doc__, type(e), sep='\n')
```

Figure 7:Main Body Script

Summary: In this paper I explained to you the steps I took to create a script with three classes using the Standard Class Pattern.