

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).

```
1  # ----- #
2  # Title: Assignment 08
3  # Description: Working with classes
4  # Changelog (Who, When, What):
5  # FSharif,3.6.2022,Created started script
6  # FSharif,3.6.2022,Added pseudo-code to start assignment 8
7  # <Fatima Sharif>,<2.6.2022>,Modified code to complete assignment 8
8  # ----- #
9
10 # Data ----- #
11 strFileName = 'products.txt'
12 lstOfProductObjects = []
```

The Standard Class Pattern:

class MyClassName(MyBaseClassName):

-- **Fields** – “variables”
-- **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.

```
14
15 class Product:
16     # --Fields--
17     strProductName = ""
18     fltProductPrice = ""
19
20     # --Constructor--
21     def __init__(self, product_name: str, product_price: float):
22         # -- Attributes --
23         self.__product_name = product_name
24         self.__product_price = product_price
25
```

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)

```
26 # -- properties --
27 # product name
28 @property
29 def product_name(self):
30     return str(self.__product_name)
31
32 @product_name.setter
33 def product_name(self, value: str):
34     self.__product_name = value
35
36 # product_price
37 @property
38 def product_price(self):
39     return float(self.__product_price)
40
41 @product_price.setter
42 def product_price(self, value):
43     if str(value).isnumeric():
44         self.__product_price = float(value)
45     else:
46         raise Exception("Prices must be numbers")
47
```

Figure 2: Properties for the Product Class

Step Five: Methods

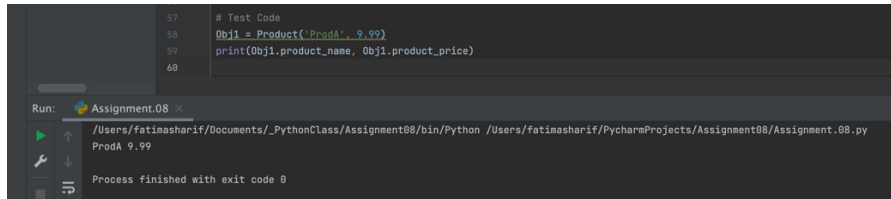
5. Add code to the method to return both the first_name and last_name with a comma separator. (Shown in Figure 3)

```
48 # -- methods --
49 def to_string(self):
50     """ alias of __str__(), converts product data to string """
51     return self.__str__()
52
53 def __str__(self):
54     """ Converts product data to string """
55     return self.product_name + ", " + str(self.product_price)
56
```

Figure 3: Methods for the Product Class

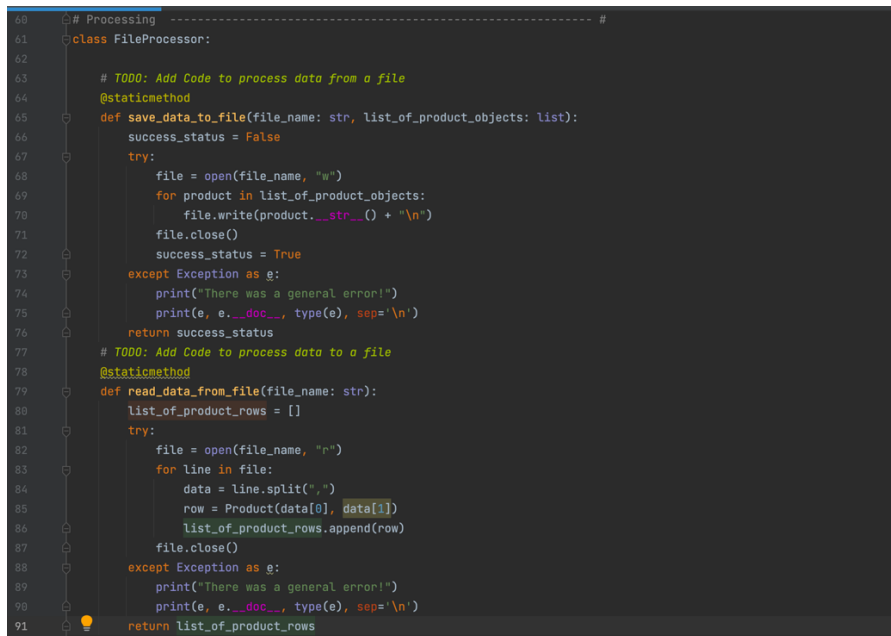
Step 6: Testing the Product Class

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



```
57: # Test Code
58: Obj1 = Product('Prada', 9.99)
59: print(Obj1.product_name, Obj1.product_price)
60:
Run: Assignment08
/Users/fatimasharif/Documents/_PythonClass/Assignment08/bin/Python /Users/fatimasharif/PycharmProjects/Assignment08/Assignment.08.py
Prada 9.99
Process finished with exit code 0
```

Figure 4: Output of testing the Product Class



```
60: # Processing ----- #
61: class FileProcessor:
62:
63:     # TODO: Add Code to process data from a file
64:     @staticmethod
65:     def save_data_to_file(file_name: str, list_of_product_objects: list):
66:         success_status = False
67:         try:
68:             file = open(file_name, "w")
69:             for product in list_of_product_objects:
70:                 file.write(product.__str__() + "\n")
71:             file.close()
72:             success_status = True
73:         except Exception as e:
74:             print("There was a general error!")
75:             print(e, e.__doc__, type(e), sep='\n')
76:         return success_status
77:
78:     # TODO: Add Code to process data to a file
79:     @staticmethod
80:     def read_data_from_file(file_name: str):
81:         list_of_product_rows = []
82:         try:
83:             file = open(file_name, "r")
84:             for line in file:
85:                 data = line.split(",")
86:                 row = Product(data[0], data[1])
87:                 list_of_product_rows.append(row)
88:             file.close()
89:         except Exception as e:
90:             print("There was a general error!")
91:             print(e, e.__doc__, type(e), sep='\n')
92:         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)

```

106
107 class IO: # TODO: Add docstring
108     # TODO: Add code to show menu to user
109     @staticmethod
110     def print_menu_items():
111         """ Print a menu of choices to the user """
112         print('')
113         Menu of Options
114         1) Show current data
115         2) Add a new item.
116         3) Save Data to File
117         4) Exit Program
118         ''')
119
120     print() # Add an extra line for looks
121
122     # TODO: Add code to get user's choice
123     @staticmethod
124     def input_menu_choice():
125
126         choice = str(input("Which option would you like to perform? [1 to 4] - ")).strip()
127         print() # Add an extra line for looks
128         return choice
129
130     # TODO: Add code to show the current data from the file to user
131     @staticmethod
132     def print_current_list_items(list_of_rows: list):
133
134         print("***** The current items products are: *****")
135         for row in list_of_rows:
136             print(row.product_name + " (" + str(row.product_price) + ")")
137         print("*****")

```

Figure 6: IO Class

Main Body Script

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

```

153 # Main Body of Script ----- #
154 try:
155     lstOfProductObjects = FileProcessor.read_data_from_file(strFileName)
156     while True:
157         IO.print_menu_items() # Show user a menu of options
158         strChoice = IO.input_menu_choice() # Get user's menu option choice
159         if strChoice.strip() == '1':
160             IO.print_current_list_items(lstOfProductObjects) # Show user current data in the list of product objects
161             continue
162         elif strChoice.strip() == '2':
163             # Let user add data to the list of product objects
164             lstOfProductObjects.append(IO.input_product_data())
165             continue
166         elif strChoice.strip() == '3':
167             # Let user save current data to file and exit program
168             FileProcessor.save_data_to_file(strFileName, lstOfProductObjects)
169             continue
170         elif strChoice.strip() == '4':
171             break
172 except Exception as e:
173     print("There was an error! Check file permissions.")
174     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.