exercise

June 1, 2020

1 OOP Syntax Exercise - Part 2

Now that you've had some practice instantiating objects, it's time to write your own class from scratch. This lesson has two parts. In the first part, you'll write a Pants class. This class is similar to the shirt class with a couple of changes. Then you'll practice instantiating Pants objects

In the second part, you'll write another class called SalesPerson. You'll also instantiate objects for the SalesPerson.

For this exercise, you can do all of your work in this Jupyter notebook. You will not need to import the class because all of your code will be in this Jupyter notebook.

Answers are also provided. If you click on the Jupyter icon, you can open a folder called 2.OOP_syntax_pants_practice, which contains this Jupyter notebook ('exercise.ipynb') and a file called answer.py.

2 Pants class

Write a Pants class with the following characteristics: * the class name should be Pants * the class attributes should include * color * waist_size * length * price * the class should have an init function that initializes all of the attributes * the class should have two methods * change_price() a method to change the price attribute * discount() to calculate a discount

```
In [1]: ### TODO:
        #   - code a Pants class with the following attributes
        #   - color (string) eg 'red', 'yellow', 'orange'
        #   - waist_size (integer) eg 8, 9, 10, 32, 33, 34
        #   - length (integer) eg 27, 28, 29, 30, 31
        #   - price (float) eg 9.28

        ### TODO: Declare the Pants Class

        class Pants:

        ### TODO: write an __init__ function to initialize the attributes

            def __init__(self, color, waist_size, length, price):
                self.color = color
                self.waist_size = waist_size
                self.length = length
```

```
            self.price = price

    ### TODO: write a change_price method:
    #    Args:
    #        new_price (float): the new price of the shirt
    #    Returns:
    #        None

        def change_price(self, new_price):
            self.price = new_price

    ### TODO: write a discount method:
    #    Args:
    #        discount (float): a decimal value for the discount.
    #            For example 0.05 for a 5% discount.
    #
    #    Returns:
    #        float: the discounted price

        def discount(self, discount):
            return self.price * (1 - discount)
```

3 Run the code cell below to check results

If you run the next code cell and get an error, then revise your code until the code cell doesn't output anything.

```
In [2]: def check_results():
            pants = Pants('red', 35, 36, 15.12)
            assert pants.color == 'red'
            assert pants.waist_size == 35
            assert pants.length == 36
            assert pants.price == 15.12

            pants.change_price(10) == 10
            assert pants.price == 10

            assert pants.discount(.1) == 9

            print('You made it to the end of the check. Nice job!')

        check_results()
```

```
You made it to the end of the check. Nice job!
```

4 SalesPerson class

The Pants class and Shirt class are quite similar. Here is an exercise to give you more practice writing a class. **This exercise is trickier than the previous exercises.**

Write a SalesPerson class with the following characteristics: * the class name should be SalesPerson * the class attributes should include * first_name * last_name * employee_id * salary * pants_sold * total_sales * the class should have an init function that initializes all of the attributes * the class should have four methods * sell_pants() a method to change the price attribute * calculate_sales() a method to calculate the sales * display_sales() a method to print out all the pants sold with nice formatting * calculate_commission() a method to calculate the salesperson commission based on total sales and a percentage

```
In [29]: ### TODO:
         #   Code a SalesPerson class with the following attributes
         #   - first_name (string), the first name of the salesperson
         #   - last_name (string), the last name of the salesperson
         #   - employee_id (int), the employee ID number like 5681923
         #   - salary (float), the monthly salary of the employee
         #   - pants_sold (list of Pants objects),
         #             pants that the salesperson has sold
         #   - total_sales (float), sum of sales of pants sold

         ### TODO: Declare the SalesPerson Class

         class SalesPerson:

         ### TODO: write an __init__ function to initialize the attributes
         ###    Input Args for the __init__ function:
         #        first_name (str)
         #        last_name (str)
         #        employee_id (int)
         # .      salary (float)
         #
         # You can initialize pants_sold as an empty list
         # You can initialize total_sales to zero.
         #
         ###

             def __init__(self, first_name, last_name, employee_id, salary):
                 self.first_name = first_name
                 self.last_name = last_name
                 self.employee_id = employee_id
                 self.salary = salary
                 self.pants_sold = []
                 self.total_sales = 0

         ### TODO: write a sell_pants method:
         #
```

```
#    This method receives a Pants object and appends
#    the object to the pants_sold attribute list
#
#    Args:
#        pants (Pants object): a pants object
#    Returns:
#        None

    def sell_pants(self, pants):
        self.pants_sold.append(pants)

### TODO: write a display_sales method:
#
#    This method has no input or outputs. When this method
#    is called, the code iterates through the pants_sold list
#    and prints out the characteristics of each pair of pants
#    line by line. The print out should look something like this
#
#   color: blue, waist_size: 34, length: 34, price: 10
#   color: red, waist_size: 36, length: 30, price: 14.15
#
#
#
###

    def display_sales(self):
        for pants in self.pants_sold:
            print("self.color: {}, self.waist_size: {}, self.length: {}, self.price: {}
                  .format(pants.color, pants.waist_size, pants.length, pants.price))

### TODO: write a calculate_sales method:
#      This method calculates the total sales for the sales person.
#      The method should iterate through the pants_sold attribute list
#      and sum the prices of the pants sold. The sum should be stored
#      in the total_sales attribute and then return the total.
#
#      Args:
#        None
#      Returns:
#        float: total sales
#
###

    def calculate_sales(self):
        for pants in self.pants_sold:
            self.total_sales += pants.price

        return self.total_sales
```

```
    ### TODO: write a calculate_commission method:
    #
    #   The salesperson receives a commission based on the total
    #   sales of pants. The method receives a percentage, and then
    #   calculate the total sales of pants based on the price,
    #   and then returns the commission as (percentage * total sales)
    #
    #    Args:
    #        percentage (float): comission percentage as a decimal
    #
    #    Returns:
    #        float: total commission
    #
    #
    ###

        def calculate_commission(self, percentage):
            return self.total_sales * percentage
```

5 Run the code cell below to check results

If you run the next code cell and get an error, then revise your code until the code cell doesn't output anything.

```
In [30]: def check_results():
             pants_one = Pants('red', 35, 36, 15.12)
             pants_two = Pants('blue', 40, 38, 24.12)
             pants_three = Pants('tan', 28, 30, 8.12)

             salesperson = SalesPerson('Amy', 'Gonzalez', 2581923, 40000)

             assert salesperson.first_name == 'Amy'
             assert salesperson.last_name == 'Gonzalez'
             assert salesperson.employee_id == 2581923
             assert salesperson.salary == 40000
             assert salesperson.pants_sold == []
             assert salesperson.total_sales == 0

             salesperson.sell_pants(pants_one)
             salesperson.pants_sold[0] == pants_one.color

             salesperson.sell_pants(pants_two)
             salesperson.sell_pants(pants_three)

             assert len(salesperson.pants_sold) == 3
             assert round(salesperson.calculate_sales(),2) == 47.36
```

```
            assert round(salesperson.calculate_commission(.1),2) == 4.74

            print('Great job, you made it to the end of the code checks!')

        check_results()
```

```
Great job, you made it to the end of the code checks!
```

5.0.1 Check display_sales() method

If you run the code cell below, you should get output similar to this:

```
color: red, waist_size: 35, length: 36, price: 15.12
color: blue, waist_size: 40, length: 38, price: 24.12
color: tan, waist_size: 28, length: 30, price: 8.12
```

```
In [24]: pants_one = Pants('red', 35, 36, 15.12)
         pants_two = Pants('blue', 40, 38, 24.12)
         pants_three = Pants('tan', 28, 30, 8.12)

         salesperson = SalesPerson('Amy', 'Gonzalez', 2581923, 40000)

         salesperson.sell_pants(pants_one)
         salesperson.sell_pants(pants_two)
         salesperson.sell_pants(pants_three)

         salesperson.display_sales()
```

```
self.color: red, self.waist_size: 35, self.length: 36, self.price: 15.12
self.color: blue, self.waist_size: 40, self.length: 38, self.price: 24.12
self.color: tan, self.waist_size: 28, self.length: 30, self.price: 8.12
```

6 Solution

As a reminder, answers are also provided. If you click on the Jupyter icon, you can open a folder called 2.OOP_syntax_pants_practice, which contains this Jupyter notebook and a file called answer.py.