Python OOPs concept demonstration lab

Exercise 1:(LAB 1 – OOP)

Create one class called FruitShop. This class should receive a name which should be a string, and fruits as a list.

Create also a method called get_fruits_count() which should return amount of fruits in the shop.

Input:
my_shop = FruitShop("My Fruit Shop", ["Banana" , "Orange", "Kiwi", "Apple"])

Output: 4

Exercise 2:(LAB 2 – OOP)

Create the following classes: Animal, Horse and Dog.

The **Animal** class should have a method eat ( ) and it should return "eating.. nom.. nom.."
The **Horse** class should have a method called neigh ( ) and it should return "neigh! "
The **Dog** class should have a method called bark ( ) and it should return "voof voof!"

And remember that the Horse and Dog class should inherit form the Animal class.

Exercise 3:(LAB 3 – OOP)

Create the following classes:

Person
Staff
Busdriver

Person should have one method named walk( ) that should return "walking"
Staff should have one method called work( ) that should return "working"
Busdriver should have one method called driving ( ) that returns "diving the bus"

The Busdriver class should inherit from both Person and Staff

Exercise 4:(LAB 4 – OOP)

In this lab you have one person in a band that wants to play some instruments...

Create one function and name it play_instrument( ) which will receive an instance of an instrument and will print its play () method.

Testing code:
Note: You need to create seprate files for each class as well.

Text

```
class Drums:
    def play(self):
        print("playing the drums")  |   OUTPUT--------> playing the drums...

drums = Drums()
play_instrument(drums)
--------------------------------------------------------------------------------

class Synth:
    def play(self):
        print("playing the synth")  |   OUTPUT --------> playing the synth...
synth = Synth()
play_instrument(synth)
--------------------------------------------------------------------------------
```

Collapse

Exercise 5:(LAB 5 – OOP)

Create an abstract class Shape with abstract methods `calculate_area` and `calculate_perimeter`

Create classes `Circle` (receives radius upon initialization) and `Rectangle` (receives height and width upon initialization) that implement those methods (returning the result)

And do the same thing as you did with the previous lab (Lab4). Seperate files for different classes.

Text

```
------------------------------------------------------------------------
circle = Circle(5)
print(circle.calculate_area())
print(circle.calculate_perimeter())

//OUTPUT
78.539816...
31.415926...
------------------------------------------------------------------------









------------------------------------------------------------------------
rectangle = Rectangle(10, 20)
print(rectangle.calculate_area())
print(rectangle.calculate_perimeter())

//OUTPUT
200
60
------------------------------------------------------------------------
```

Exercise 6:(LAB 6 – OOP)

You are given this code:

Text

```
def even_numbers(function):
    def wrap (numbers):
        #your code here...
    return wrap
```

Complete this code so it works using decorators.

Exercise 7:(LAB 7 – OOP)

You are provided with the following phrase:
phrase = 'This is a string and it has some numbers like 5533 and a symbol #hashtag'

Use python lib re and try to find:

1. **Sequence of digits**
2. **Sequence of non-digits**
3. **Sequence of whitespace**
4. **Sequence of non-whitespace**
5. **alphanumeric characters**
6. **non alphanumeric**