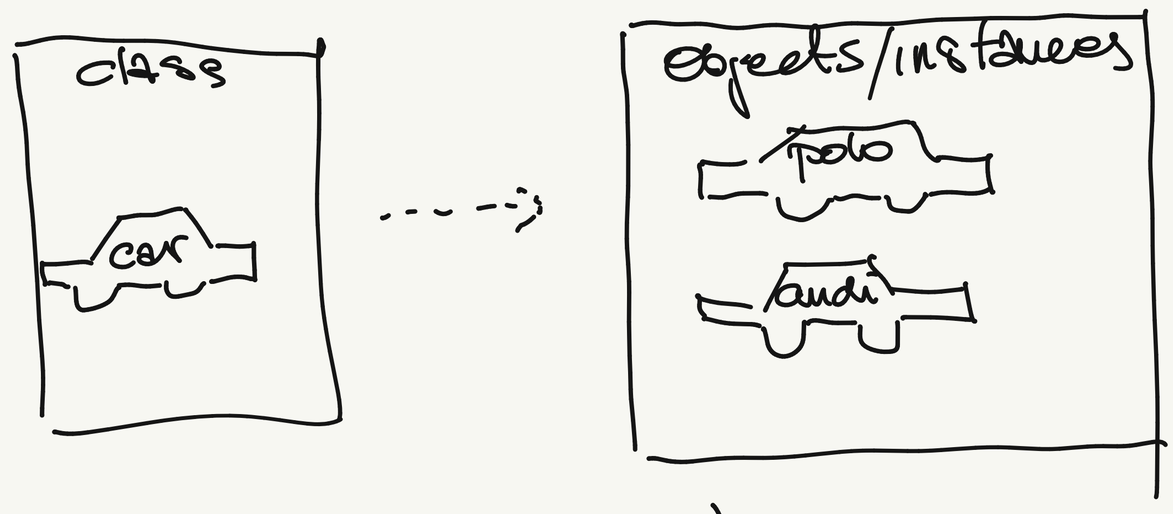
OBJECT ORIENTED PROGRAMMING - NOTES 1

1) Instances - individual object of a certain class.

EXAMPLE:



2) Attributes (instance variables)

EXAMPLE:

```
class Car(object):
    number_of_wheels = 4   <- attribute example
    color
```

Then we can assign the value to an attr.

```
Polo = Car()
Polo.color = "red"
```

3) Methods - functions inside a class, having an acces to attributes of that class

EXAMPLE:

```
Polo.how_old_is_car()
class Car(object)
    production_year

    def how_old_is_car(self):
        return current_year - self.production_year
```

→ the interpreter first looks up Polo and finds that is an instance of a class Car.

→ then looks up the attribute (if any) and finds that is a method.

→ the only difference between a method invocation and other function calls is that the object instance itself is also passed as a parameter.

OBJECT ORIENTED PROGRAMMING - NOTES 12

PYTHON'S INSTANCE, CLASS, AND STATIC METHODS

- for Python 3
- for Python 2 class MYCLASS(object):

```
class MYCLASS:
    def method(self):
        return 'instance method called', self

    @classmethod
    def classmethod(cls):
        return 'class method called', cls

    @staticmethod
    def staticmethod():
        return 'static method called'
```

DECORATORS

regular instance method
- takes one parameter "self"
- can freely access attributes and other methods on the same object
- can modify object state but also by using self.__class__ can modify class state

class method
- instead of accepting "self" parameter class method takes a "cls" param that points to the class — and not the object instance — when the method is called
  - it cannot modify object state
  - can modify class state that applies across all.

static method
- restricted in what data they can access