类方法、静态方法

1. 类方法

- **第一个形参是类对象的方法**
- 需要用装饰器 `@classmethod` 来标识其为类方法，对于类方法，**第一个参数必须是类对象**，一般以 `cls` 作为第一个参数。

```
class Dog(object):
    __type = "狗"

    # 类方法，用classmethod来进行修饰
    @classmethod
    def get_type(cls):
        return cls.__type
print(Dog.get_type())
```

> 使用场景： - 当方法中 **需要使用类对象** (如访问私有类属性等)时，定义类方法 - 类方法一般和类属性配合使用

2. 静态方法

- 需要通过装饰器 `@staticmethod` 来进行修饰，**静态方法既不需要传递类对象也不需要传递实例对象（形参没有self/cls）**。
- 静态方法 也能够通过 **实例对象** 和 **类对象** 去访问。

```
class Dog(object):
    type = "狗"

    def __init__(self):
        name = None

    # 静态方法
    @staticmethod
    def introduce():  # 静态方法不会自动传递实例对象和类对象
        print("犬科哺乳动物,属于食肉目..")

dog1 = Dog()
Dog.introduce()     # 可以用 实例对象 来调用 静态方法
dog1.introduce()     # 可以用 类对象 来调用 静态方法
```

> 使用场景： - 当方法中 **既不需要使用实例对象**(如实例对象，实例属性)，**也不需要使用类对象** (如类属

> 性、类方法、创建实例等)时，定义静态方法 - **取消不需要的参数传递**，有利于 **减少不必要的内存占用和性能消耗**

注意点：

- 类中定义了同名的方法时，调用方法会执行最后定义的方法

```
class Dog:

    def demo_method(self):
        print("对象方法")

    @classmethod
    def demo_method(cls):
        print("类方法")

    @staticmethod
    def demo_method():  # 被最后定义
        print("静态方法")

dog1 = Dog()
Dog.demo_method()  # 结果：静态方法
dog1.demo_method()  # 结果：静态方法
```