课程：面向对象-其他

目标

- 面向对象三大特性
- 类属性和实例属性
- 类方法和静态方法

一. 面向对象三大特性

- 封装
  - 将属性和方法书写到类的里面的操作即为封装
  - 封装可以为属性和方法添加私有权限
- 继承
  - 子类默认继承父类的所有属性和方法
  - 子类可以重写父类属性和方法
- 多态
  - 传入不同的对象，产生不同的结果

二. 多态

2.1 了解多态

多态指的是一类事物有多种形态，（一个抽象类有多个子类，因而多态的概念依赖于继承）。

- 定义：多态是一种使用对象的方式，子类重写父类方法，调用不同子类对象的相同父类方法，可以产生不同的执行结果
- 好处：调用灵活，有了多态，更容易编写出通用的代码，做出通用的编程，以适应需求的不断变化！
- 实现步骤：
  - 定义父类，并提供公共方法
  - 定义子类，并重写父类方法
  - 传递子类对象给调用者，可以看到不同子类执行效果不同

2.2 体验多态

```
class Dog(object):
    def work(self):  # 父类提供统一的方法，哪怕是空方法
```

```
        print('指哪打哪...')


class ArmyDog(Dog):  # 继承Dog类
    def work(self):  # 子类重写父类同名方法
        print('追击敌人...')


class DrugDog(Dog):
    def work(self):
        print('追查毒品...')


class Person(object):
    def work_with_dog(self, dog):  # 传入不同的对象，执行不同的代码，即不同的work函数
        dog.work()


ad = ArmyDog()
dd = DrugDog()

daqiu = Person()
daqiu.work_with_dog(ad)
daqiu.work_with_dog(dd)
```

三. 类属性和实例属性

3.1 类属性

3.1.1 设置和访问类属性

- 类属性就是 **类对象** 所拥有的属性，它被 **该类的所有实例对象 所共有**。
- 类属性可以使用 **类对象** 或 **实例对象** 访问。

```
class Dog(object):
    tooth = 10


wangcai = Dog()
xiaohei = Dog()

print(Dog.tooth)  # 10
print(wangcai.tooth)  # 10
print(xiaohei.tooth)  # 10
```

> 类属性的优点
>
> - **记录的某项数据 始终保持一致时**，则定义类属性。
> - **实例属性** 要求 **每个对象** 为其 **单独开辟一份内存空间** 来记录数据，而 **类属性** 为全类所共有，**仅占用一份内存，更加节省内存空间。**

3.1.2 修改类属性

类属性只能通过类对象修改，不能通过实例对象修改，如果通过实例对象修改类属性，表示的是创建了一个实例属性。

```python
class Dog(object):
    tooth = 10


wangcai = Dog()
xiaohei = Dog()

# 修改类属性
Dog.tooth = 12
print(Dog.tooth)  # 12
print(wangcai.tooth)  # 12
print(xiaohei.tooth)  # 12

# 不能通过对象修改属性，如果这样操作，实则是创建了一个实例属性
wangcai.tooth = 20
print(Dog.tooth)  # 12
print(wangcai.tooth)  # 20
print(xiaohei.tooth)  # 12
```

3.2 实例属性

```python
class Dog(object):
    def __init__(self):
        self.age = 5

    def info_print(self):
        print(self.age)


wangcai = Dog()
print(wangcai.age)  # 5
# print(Dog.age)  # 报错：实例属性不能通过类访问
wangcai.info_print()  # 5
```

四. 类方法和静态方法

4.1 类方法

4.1.1 类方法特点

- 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。

4.1.2 类方法使用场景

- 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

```python
class Dog(object):
    __tooth = 10

    @classmethod
    def get_tooth(cls):
        return cls.__tooth


wangcai = Dog()
result = wangcai.get_tooth()
print(result)  # 10
```

4.2 静态方法

4.2.1 静态方法特点

- 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。

4.2.2 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)，**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**

```python
class Dog(object):
    @staticmethod
    def info_print():
        print('这是一个狗类，用于创建狗实例....')


wangcai = Dog()
# 静态方法既可以使用对象访问又可以使用类访问
wangcai.info_print()
Dog.info_print()
```

五. 总结

- 面向对象三大特性
  - 封装
  - 继承
  - 多态
- 类属性
  - 归属于类对象的属性，所有对象共有的属性
- 实例属性
- 类方法

```python
@classmethod
def xx():
  代码
```

- 静态方法

```python
@staticmethod
def xx():
  代码
```