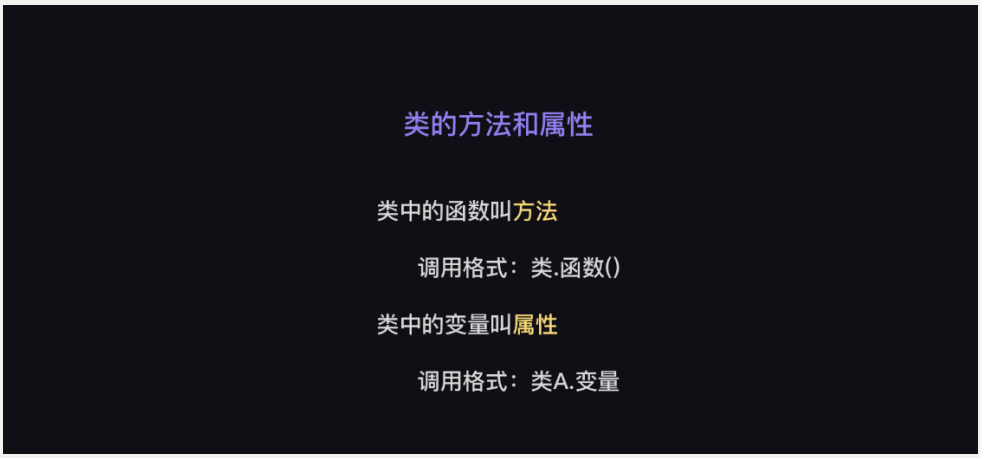
类

类是一个函数包，其中放置函数和变量

定义实例

```
class ClassNmae():
  var1 = 1     //类中存放变量为属性
  var2 = 3
  def function():  //类中函数叫方法
    print()

#调用类中的方法
ClassName.function()
#新增类的变量
ClassNmae.var3 = 4
```



类方法也是函数，那和之前学的单独定义函数有什么区别吗?

它们两者最大的区别，一个是它的调用格式：类.函数名()比函数名()多了一个【类.】，但更重要的是，“类”中的函数可以利用“类”中的变量（也就是类方法可以调用类属性）。

关于 @classmethod

```python
class 类A():
    变量1 = 100
    变量2 = 200

    @classmethod ①
    def 函数1(cls):②
        print(cls.变量1)
        print(cls.变量2) ③
类A.函数1()
```

by 风变编程

① 第一个格式 `@classmethod` 的中文意思就是“类方法”， `@classmethod` 声明了 `函数1` 是类方法，这样才能允许 `函数1` 使用类属性中的数据。

② 第二个格式 `cls` 的意思是 `class` 的缩写。如果类方法 `函数1` 想使用类属性（也就是类中的变量），就要写上 `cls` 为 `函数1` 的第一个参数，也就是把这个类作为参数传给自己，这样就能被允许使用类中的数据。

③ 第三个格式是 `cls.变量` 。类方法想使用类属性的时候，需要在这些变量名称前加上 `cls.` 。

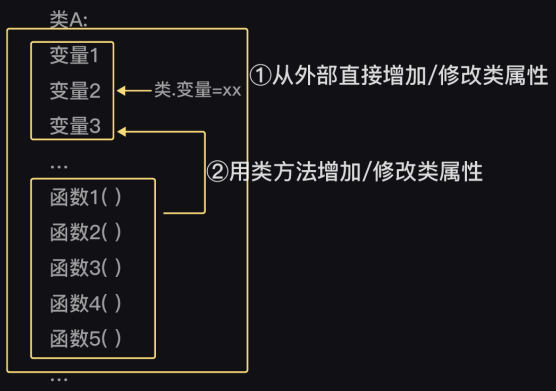
这就好比 `类方法` 和 `类` 之间的约法三章，所以但凡有任何格式错误都会报错。

如果缺①，即缺了“@classmethod”，类方法就不能直接利用类中的属性，于是报错。（请运行代码，报错后，修改格式到正确的样式就能运行通过）

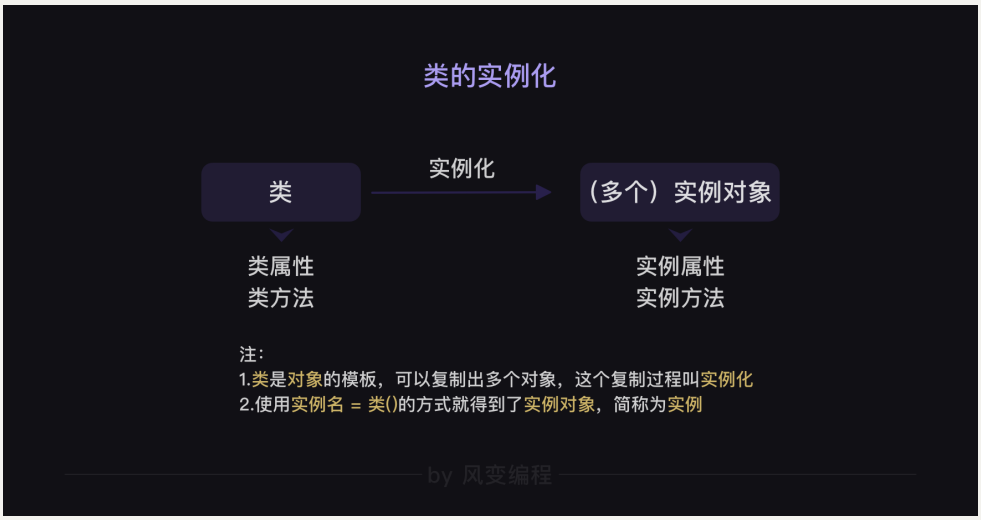
给类方法传参

- 类方法仅使用外部参数
- 类方法仅使用类属性
- 类方法同时使用类属性和外部参数

by 风变编程



by 风变编程



类方法重写

```python
class 类():
    def 原始函数(self):
        print('我是原始函数！')

def 新函数(self):
    print('我是重写后的新函数！')

a = 类()  # 实例化
a.原始函数()

# 用新函数代替原始函数，也就是【重写类方法】
类.原始函数 = 新函数

# 现在原始函数已经被替换了
a.原始函数()
```

要注意的是，这里的赋值是在替换方法，并不是调用函数，所以【不要加上括号】—— 写成类.原始函数() = 新函数()是错误的。

初始化函数

初始化函数的意思是，当你创建一个实例的时候，这个函数就会被调用。上面的代码在执行实例 = 类()的语句时，就自动调用了**init**(self)函数。

初始化函数的写法是固定的格式：中间是“init”，这个单词的中文意思是“初始化”，然后前后都要有【两个下划线】，然后**init**()的括号中，第一个参数一定要写上self，不然会报错。

初始化示例

```
class 成绩单():
    def __init__(self,学生姓名,语文_成绩,数学_成绩):
        self.学生姓名 = 学生姓名
        self.语文_成绩 = 语文_成绩
        self.数学_成绩 = 数学_成绩

    def 打印成绩单(self):
        print(self.学生姓名 + '的成绩单如下：')
        print('语文成绩：'+ str(self.语文_成绩))
        print('数学成绩：'+ str(self.数学_成绩))


成绩单1 = 成绩单('张三',99,88)
成绩单2 = 成绩单('李四',64,73)
成绩单3 = 成绩单('王五',33,22)

成绩单1.打印成绩单()
成绩单2.打印成绩单()
成绩单3.打印成绩单()
```

类的继承

类的继承很大程度也是为了避免重复性劳动。比如说当我们要写一个新的类，如果新的类有许多代码都和旧类相同，又有一部分不同的时候，就可以用“继承”的方式避免重复写代码。

在Python里，我们统一把旧的类称为父类，新写的类称为子类。子类可以在父类的基础上改造类方法，所以我们可以说子类继承了父类。