面向对象基本语法

在Python中，对象几乎是无处不在的，我们可以使用 `dir` 内置函数来查看这个对象里的方法。

定义简单的类（只包含方法）

> 面向对象是更大的封装，在一个类中封装多个方法，这样通过这个类创建出来的对象，就可以直接调用这些方法了！

定义类

在Python中要定义一个只包含方法的类，语法格式如下：

```
class 类名:
    def 方法1(self,参数列表):
        pass
    def 方法2(self,参数列表):
        pass
```

> 1. 方法的定义格式和之前学习过的函数一样
> 2. 方法里的**第一个参数必须是self**，大家暂时先记住，稍后介绍 self.
> 3. 类名要遵守大驼峰命名法。

创建实例对象

当一个类定义完成之后，要使用这个类来创建对象，语法格式如下：

```
对象变量名 = 类名()
```

第一个面向对象代码

需求

- 小猫 爱 吃 鱼，小猫 要 喝 水

分析

- 定义一个猫类 Cat
- 定义两个方法 eat 和 drink
- 按照需求 —— 不需要定义属性

```
class Cat:
    """这是个猫类"""

    def eat(self):
        print("小猫在吃东西")

    def drink(self):
        print("小猫在喝水")

tom = Cat()  # 创建了一个Cat对象
tom.eat()
tom.drink()

hello_kitty = Cat()  # 又创建了一个新的Cat对象
hello_kitty.eat()
hello_kitty.drink()
```

思考：`tom` 和 `hello_kitty` 是同一个对象吗?

self的使用

给对象添加属性

python支持动态属性，当一个对象创建好了以后，直接使用 `对象.属性名 = 属性值` 就可以很方便的给对象添加一个属性。

```
tom = Cat()
tom.name = 'Tom'  # 可以直接给 tom 对象添加一个 name 属性
```

这种方法很方便，但是，不建议使用这种方式给对象添加属性。

self的概念

哪个对象调用了方法，方法里的 `self` 指的就是谁。 通过 `self.属性名` 可以访问到这个对象的属性；通过 `self.方法名()` 可以调用这个对象的方法。

```
class Cat:
    def eat(self):
        print("%s爱吃鱼" %self.name)

tom = Cat()
tom.name = 'Tom'  # 给 tom 对象添加了一个name属性
tom.eat()  # Tom爱吃鱼

lazy_cat = Cat()
lazy_cat.name = "大懒猫"
lazy_cat.eat() # 大懒猫爱吃鱼
```

直接给对象添加属性的缺点

上述代码中，我们是先创建对象，然后再给对象添加 `name` 属性，但是这样做会有问题。

```
tom = Cat()
tom.eat()
tom.anme = "Tom"
```

程序运行时会报错：

```
AttributeError: 'Cat' object has no attribute 'name'
错误提示：'Cat'对象没有 'name' 属性
```

在日常开发中，不推荐在类的外部直接给对象添加属性这种方式。对象应该具有哪些属性，我们应该封装在类的内部。