2. 类和对象的使用

Python 从创建之初就已经是一门面向对象的语言，因此在 Python 中可以很容易的实现面向对象编程。

（1）创建类及对象

Python 中定义一个类，使用 class 关键字。

【语法】

class 类名():

类属性和方法

其中

- class 是关键字
- 类名通常是大写开头的单词
- 类名后面的()，表示继承关系，即该类继承自谁，可选，默认继承自 object 类。

如下，创建了一个学校类 School。

```
#创建类 School
class School:
    pass
```

注意，此类现在是一个没有属性和方法的类，故使用 pass 关键字占位。

小贴士

> **骆驼命名法**
> 如果变量名或函数名是由一个或多个单词连结在一起构成，则第一个单词以小写字母开始，从第二个单词开始以后的每个单词的首字母都采用大写，例如：myFirstName、myLastName

创建好类之后，就可以用这个类来创建实例对象了，创建语法如下。

【语法】

实例变量名= 类名()

如下，创建了 School 类的实例对象：光明学校和雄鹰学校。

```
#创建对象：光明学校和雄鹰学校
guangMing = School()
xiongYing = School()
```

（2）添加属性变量

刚才已经提到过，每个类都有属性和方法。如何添加属性呢。在编程中，对象的属性一般使用变量存储，对象的方法就是功能函数。

为 School 类添加两个属性变量：name 和 address。

```
#创建类 School
class School:
    name = ''
    address = ''
```

这样为类添加属性是完全可以的，但是对于面向对象程序，我们更希望类在创建时有个初始状态。这就用到了构造方法和析构方法。

1）构造方法和析构方法

在类中有两个非常特殊的方法：__init__()和__del__()。__init__()方法被称为构造方法，它在对象被创建时自动调用。__del__()方法被称为析构方法，它在对象被销毁时自动调用。如此一来，就可以把类的一些初始化信息放在__init__()里，把对象销毁时要处理的内容放在__del__()里。

小贴士

这两个方法前后的横线“__”是两个下划线，不要写错。

示例：思考如下代码片段结果，体会一下“自动”调用的含义。

```
# 理解__init()__()方法自动调用
class School:
    def __init__(self):
        print('我被自动调用')
guangMing = School()
xiongYing = School()
```

输出结果为：

我被自动调用

我被自动调用

通过观察输出结果，我们可以知道，在实例化两个学校对象 guangMing 和 xiongYing 时，自动调用了构造方法__init__()，实施了打印输出。

了解了构造方法和析构方法的这个特点，初始化的一些数据就可以直接放在构造方法里进行，对象销毁时的数据处理就可以放在析构方法里了。构造方法__init__()一定有一个参数 self，self 其实是指向创建的对象本身，self 代表类的实例，而非类。

2）实例变量

在创建类的时候，有一些属性是必须的，这样，我们就可以利用构造方法，把一些必须绑定的属性强制填写进去，这样，只要对象被创建，这些变量就会被创建，这被称为实例变量，实例变量必须在创建对象后才能使用。

如下是定义在构造方法中的实例变量（构造方法中的实例变量也称为成员变量）。

学校名称、学校地址是类 School 的属性，每个实体对象学校都有这两个属性，我们可以这么做：

```
# 实例变量
class School:
    def __init__(self,name='',address=''):
        self.name = name
        self.address = address
```

在构造方法内部，就把 name 和 address 属性绑定到 self，在创建对象的时候，就不能传入空的参数了，必须传入与构造方法匹配的参数，但 self 不需要传，Python 解释器自己会把实例变量传进去：

```
#创建对象：光明学校和雄鹰学校
guangMing = School('光明中学','北京')
xiongYing = School('雄鹰中学','上海')
print(guangMing.name ,guangMing.address)
print(xiongYing.name,xiongYing.address)
```

输出结果如下：

光明中学 北京

雄鹰中学 上海

如果某些属性不用一开始就赋值，而是通过实例化对象来赋值，该怎么做呢？只要构造方法__init__()不设置参数就可以了。

```
# 实例变量
class School:
    def __init__(self):
        self.name = ''
        self.address = ''
#创建对象：光明学校和雄鹰学校
guangMing = School()
guangMing .name = '光明中学'
guangMing .address = '北京'
xiongYing = School()
xiongYing .name = '雄鹰中学'
xiongYing .address = '上海'
print(guangMing.name,guangMing.address)
print(xiongYing .name,xiongYing .address)
```

输出结果同前面定义的一样，如下：

光明中学 北京

雄鹰中学 上海

除了定义在构造方法中的实例变量，实例变量还可以在其他地方创建，例如类方法，但是只要记住实例变量只有对象被创建后才能使用的变量。实例对象的调用方式一定是：

【语法】

self.变量名 = 变量值

3）类变量

实例变量是必须在创建对象以后才能使用的变量。在某些场景下，希望能够通过类名直接调用类的变量或者希望所有类能够公用某个变量，在这种情况下，就可以使用类的类变量。

类变量相当于类的一个全局变量，只要能够使用类的地方都能够访问或修改类变量的值，记住它有个特点，就是可以不创建对象直接调用。

示例：思考如下代码片段运行结果，体会一下类变量 entrance_age 的使用。

```
class School:
    entrance_age = 15
    def __init__(self):
        self.name = ''
        self.address = ''
guangMing = School()
guangMing .name = '光明中学'
guangMing .address = '北京'
guangMing .entrance_age = 16#重赋值类变量
xiongYing = School()
xiongYing .name = '雄鹰中学'
xiongYing .address = '上海'
print(guangMing .name,guangMing .address,guangMing .entrance_age)
print(xiongYing .name,xiongYing .address,xiongYing .entrance_age)
print(School.entrance_age)#直接通过类名调用
```

输出结果为：

光明中学 北京 16

雄鹰中学 上海 15

15