

INDEX



- Python教程
 - Python简介
 - 安装Python
 - 第一个Python程序
 - Python基础
 - 函数
 - 高级特性
 - 函数式编程
 - 模块
- 面向对象编程
 - 类和实例
 - 访问限制
 - 继承和多态
 - 获取对象信息
 - 实例属性和类属性**
- 面向对象高级编程
- 错误、调试和测试
- IO编程
- 进程和线程
 - 正则表达式
- 常用内建模块
- 常用第三方模块
 - virtualenv
- 图形界面
- 网络编程
- 电子邮件
- 访问数据库
- Web开发
- 异步IO
- 实战
- FAQ
- 期末总结

关于作者



廖雪峰

北京 朝阳区

+ 加关注

实例属性和类属性

Reads: 11360817

由于Python是动态语言，根据类创建的实例可以任意绑定属性。

给实例绑定属性的方法是通过实例变量，或者通过 `self` 变量：

```
class Student(object):
    def __init__(self, name):
        self.name = name

s = Student('Bob')
s.score = 90
```

但是，如果 `Student` 类本身需要绑定一个属性呢？可以直接在class中定义属性，这种属性是类属性，归 `Student` 类所有：

```
class Student(object):
    name = 'Student'
```

当我们定义了一个类属性后，这个属性虽然归类所有，但类的所有实例都可以访问到。来测试一下：

```
>>> class Student(object):
...     name = 'Student'
...
>>> s = Student() # 创建实例s
>>> print(s.name) # 打印name属性，因为实例并没有name属性，所以会继续查找class的name属性
Student
>>> print(Student.name) # 打印类的name属性
Student
>>> s.name = 'Michael' # 给实例绑定name属性
>>> print(s.name) # 由于实例属性优先级比类属性高，因此，它会屏蔽掉类的name属性
Michael
>>> print(Student.name) # 但是类属性并未消失，用Student.name仍然可以访问
Student
>>> del s.name # 如果删除实例的name属性
>>> print(s.name) # 再次调用s.name，由于实例的name属性没有找到，类的name属性就显示出来了
Student
```

从上面的例子可以看出，在编写程序的时候，千万不要对实例属性和类属性使用相同的名字，因为相同名称的实例属性将屏蔽掉类属性，但是当你删除实例属性后，再使用相同的名称，访问到的将是类属性。

练习

为了统计学生人数，可以给Student类增加一个类属性，每创建一个实例，该属性自动增加：

```
# -*- coding: utf-8 -*-

class Student(object):
    count = 0

    def __init__(self, name):
        self.name = name

# 测试:
if Student.count != 0:
    print('测试失败!')
else:
    bart = Student('Bart')
    if Student.count != 1:
        print('测试失败!')
    else:
        lisa = Student('Bart')
        if Student.count != 2:
            print('测试失败!')
        else:
            print('Students:', Student.count)
            print('测试通过!')
```

▶ Run

小结

实例属性属于各个实例所有，互不干扰；

类属性属于类所有，所有实例共享一个属性；

不要对实例属性和类属性使用相同的名字，否则将产生难以发现的错误。

读后有收获可以支付宝请作者喝咖啡，读后有疑问请加微信群讨论：



还可以分享给朋友：

分享到微博

◀ Previous Page

Next Page ▶

Comments

Make a comment

Sign in to make a comment

