

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

面向对象高级编程

使用\_\_slots\_\_

使用@property

多重继承

定制类

使用枚举类

使用元类

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

使用\_\_slots\_\_

Reads: 13919722

正常情况下，当我们定义了一个class，创建了一个class的实例后，我们可以给该实例绑定任何属性和方法，这就是动态语言的灵活性。先定义class：

```
class Student(object):  
    pass
```

然后，尝试给实例绑定一个属性：

```
>>> s = Student()  
>>> s.name = 'Michael' # 动态给实例绑定一个属性  
>>> print(s.name)  
Michael
```

还可以尝试给实例绑定一个方法：

```
>>> def set_age(self, age): # 定义一个函数作为实例方法  
...     self.age = age  
...  
>>> from types import MethodType  
>>> s.set_age = MethodType(set_age, s) # 给实例绑定一个方法  
>>> s.set_age(25) # 调用实例方法  
>>> s.age # 测试结果  
25
```

但是，给一个实例绑定的方法，对另一个实例是不起作用的：

```
>>> s2 = Student() # 创建新的实例  
>>> s2.set_age(25) # 尝试调用方法  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
AttributeError: 'Student' object has no attribute 'set_age'
```

为了给所有实例都绑定方法，可以给class绑定方法：

```
>>> def set_score(self, score):  
...     self.score = score  
...  
>>> Student.set_score = set_score
```

给class绑定方法后，所有实例均可调用：

```
>>> s.set_score(100)  
>>> s.score  
100  
>>> s2.set_score(99)  
>>> s2.score  
99
```

通常情况下，上面的 set\_score 方法可以直接定义在class中，但动态绑定允许我们在程序运行的过程中动态给class加上功能，这在静态语言中很难实现。

使用\_\_slots\_\_

但是，如果我们想要限制实例的属性怎么办？比如，只允许对Student实例添加 name 和 age 属性。

为了达到限制的目的，Python允许在定义class的时候，定义一个特殊的 \_\_slots\_\_ 变量，来限制该class实例能添加的属性：

```
class Student(object):  
    __slots__ = ('name', 'age') # 用tuple定义允许绑定的属性名称
```

然后，我们试试：

```
>>> s = Student() # 创建新的实例  
>>> s.name = 'Michael' # 绑定属性'name'  
>>> s.age = 25 # 绑定属性'age'  
>>> s.score = 99 # 绑定属性'score'  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
AttributeError: 'Student' object has no attribute 'score'
```

由于 'score' 没有被放到 \_\_slots\_\_ 中，所以不能绑定 score 属性，试图绑定 score 将得到 AttributeError 的错误。

使用 \_\_slots\_\_ 要注意， \_\_slots\_\_ 定义的属性仅对当前类实例起作用，对继承的子类是不起作用的：

```
>>> class GraduateStudent(Student):  
...     pass  
...  
>>> g = GraduateStudent()  
>>> g.score = 9999
```

除非在子类中也定义 \_\_slots\_\_，这样，子类实例允许定义的属性就是自身的 \_\_slots\_\_ 加上父类的 \_\_slots\_\_。

参考源码

use\_slots.py

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

