#### 面向对象高级编程

多重继承、定制类、元类

### 使用\_\_slots\_\_

限制实例的属性

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

#### 使用@property

有没有既能检查参数，又可以用类似属性这样简单的方式来访问类的变量呢？对于追求完美的Python程序员来说，这是必须要做到的！

还记得装饰器（decorator）可以给函数动态加上功能吗？对于类的方法，装饰器一样起作用。Python内置的@property装饰器就是负责把一个方法变成属性调用的：

@property广泛应用在类的定义中，可以让调用者写出简短的代码，同时保证对参数进行必要的检查，这样，程序运行时就减少了出错的可能性。

#### 多重继承

通过多重继承，一个子类就可以同时获得多个父类的所有功能。

### MixIn

MixIn的目的就是给一个类增加多个功能，这样，在设计类的时候，我们优先考虑通过多重继承来组合多个MixIn的功能，而不是设计多层次的复杂的继承关系。

由于Python允许使用多重继承，因此，MixIn就是一种常见的设计。

只允许单一继承的语言（如Java）不能使用MixIn的设计。

#### 定制类

### \_\_str\_\_

这是因为直接显示变量调用的不是\_\_str\_\_()，而是\_\_repr\_\_()，两者的区别是\_\_str\_\_()返回用户看到的字符串，而\_\_repr\_\_()返回程序开发者看到的字符串，也就是说，\_\_repr\_\_()是为调试服务的。

解决办法是再定义一个\_\_repr\_\_()。但是通常\_\_str\_\_()和\_\_repr\_\_()代码都是一样的，所以，有个偷懒的写法：

**class Student(object):**

**def** **\_\_init\_\_**(self, name):

self.name = name

**def** **\_\_str\_\_**(self):

**return** 'Student object (name=%s)' % self.name

\_\_repr\_\_ = \_\_str\_\_

### \_\_iter\_\_

如果一个类想被用于for ... in循环，类似list或tuple那样，就必须实现一个\_\_iter\_\_()方法，该方法返回一个迭代对象，然后，Python的for循环就会不断调用该迭代对象的\_\_next\_\_()方法拿到循环的下一个值，直到遇到StopIteration错误时退出循环。

### \_\_getattr\_\_

正常情况下，当我们调用类的方法或属性时，如果不存在，就会报错。比如定义Student类：

只有在没有找到属性的情况下，才调用\_\_getattr\_\_，已有的属性，比如name，不会在\_\_getattr\_\_中查找。

#### 使用枚举类

value属性则是自动赋给成员的int常量，默认从1开始计数

#### 使用元类

### type()

动态语言和静态语言最大的不同，就是函数和类的定义，不是编译时定义的，而是运行时动态创建的。

>>> def fn(self, name='world'): # 先定义函数

... print('Hello, %s.' % name)

...

>>> Hello = type('Hello', (object,), dict(hello=fn)) # 创建Hello **class**

**>>> h = Hello()**

**>>> h.hello()Hello, world.**

**>>> print(type(Hello))**

**<class 'type'>**

**>>> print(type(h))**

**<class '\_\_main\_\_.Hello'>**

要创建一个class对象，type()函数依次传入3个参数：

1. class的名称；
2. 继承的父类集合，注意Python支持多重继承，如果只有一个父类，别忘了tuple的单元素写法；
3. class的方法名称与函数绑定，这里我们把函数fn绑定到方法名hello上。

通过type()函数创建的类和直接写class是完全一样的，因为Python解释器遇到class定义时，仅仅是扫描一下class定义的语法，然后调用type()函数创建出class。

### metaclass

metaclass，直译为元类，简单的解释就是：先定义metaclass，就可以创建类，最后创建实例。

metaclass是Python中非常具有魔术性的对象，它可以改变类创建时的行为。这种强大的功能使用起来务必小心。