链滴

python 学习 --- 面向对象的编程（四）

作者：oldcaptain
原文链接：https://ld246.com/article/1374498637982
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

<h1> 简介 </h1>
 到目前为止，在我们的程序中，我们都是根据操作数据的函数或语句块来设计程序的。这被称为nbsp;面向过程的 编程。还有一种把数据和功能结合起来，用称为对象的东西包裹起来组织程的方法。这种方法称为 面向对象的 编程理念。在大多数时候你可以使用过程性编程，是有些时候当你想要编写大型程序或是寻求一个更加合适的解决方案的时候，你就得使用面向对象的程技术。
 类和对象是面向对象编程的两个主要方面。类创建一个新类型，而<stron>对象这个类的 实例 。这类似于你有一个int类型的变量，这存储整数的变是int类的实例（对象）。
 注意，即便是整数也被作为对象（属于int类）。这和C++、Java（1.5版之前）把整数纯粹作为型是不同的。通过help(int)了解更多这个类的详情。 C#和Java 1.5程序员会熟悉这个概念，因为它类与 封装与解封装 的概念。
 对象可以使用普通的 属于 对象的变量存储数据。属于一个对象或类的变量被称为strong>域。对象也可以使用 属于 类的函数来具有功能。这样的函数被称类的方法。这些术语帮助我们把它们与孤立的函数和变量区分开来。域和方法以合称为类的属性。
 域有两种类型——属于每个实例/类的对象或属于类本身。它们分别被称为实例变量</trong>和类变量。
 类使用class关键字创建。类的域和方法被列在一个缩进块中。
 1、self
 类的方法与普通的函数只有一个特别的区别——它们必须有一个额外的第一个参数名称，但是调用这个方法的时候你不为这个参数赋值，Python会提供这个值。这个特别的量指对象本身，按照惯例它的名称是<a>self。
 虽然你可以给这个参数任何名称，但是 强烈建议 你使用<a>self这个名称—其他名称都是不赞成你使用的。使用一个标准的名称有很多优点——你的程序读者可以迅速识别它如果使用<a>self的话，还有些IDE（集成开发环境）也可以帮助你。
 给C++/Java/C#程序员的注释
 Python中的<a>self等价于C++中的<a>self针和Java、C#中的this参考。
 你一定很奇怪Python如何给<a>self赋值以及为何你不需要给它赋值。举一个例子会使此得清晰。假如你有一个类称为MyClass和这个类的一个实例MyObject。当你调用这个对象的方法Myject.method(arg1, arg2)的时候，这会由Python自动转为MyClass.method(MyObject, arg1, arg2——这就是<a>self的原理了。
 这也意味着如果你有一个不需要参数的方法，你还是得给这个方法定义一个<a>self参数。
 2、类
<pre>>>> class person:
    pass

>>> p = person

>>> p =person()

>>> print p

<**main**.person instance at 0x01E621C0>

>>> p = person

>>> print p

main.person

>>>

</pre>
 我们使用class语句后跟<a>类名，创建了一个新的<a>类。这后面跟着一个缩进的句块形成<a>类体。在这个例子中，我们使用了一个空白块，它由pass语句表示。

 接下来，我们使用<a>类名后跟一对圆括号来创建一个对象/实例。
 为了验证，我们简单地打印了这个变量的<a>类型。它告诉我们我们已经在__main__模块有了一个Person<a>类的实例。
 可以注意到存储对象的计算机内存地址也打印了出来。这个地址在你的计算机上会是另外一个，因为Python可以在任何空位存储对象。
 3、对象的方法
 我们已经讨论了类/对象可以拥有像函数一样的方法，这些方法与函数的区别只是一个额外的sel变量。现在我们来学习一个例子。
<pre>>>> class person:
    def sayHi(self):
        print 'hello how are you?'

>>> p = person()

>>> p.sayHi()

hello how are you?

>>> </pre>

 里我们看到了self的用法。注意sayHi方法没有任何参数，但仍然在函数定义时有self。
 4、__init__方法
 __init__方法在类的一个对象被建立时，马上运行。这个方法可以用来对你的对象做一些你希望 初始化 。注意，这个名称的开始和结尾都是双下划线。
 这里的__init__和java或者c++的构造函数有略微的区别，虽然都是最先执行，但是java是在构对象的时候先执行，而在python中，是将对象构造完之后，最先执行

<pre>>>> class person:
    def __init__(self,name):
        self.name = name
    def sayhi(self):
        print 'hello,my name is',self.name

>>> p = person('oldcaptain.cc')

>>> p.sayhi()

hello,my name is oldcaptain.cc

>>> </pre>

 这里，我们把__init__方法定义为取一个参数name（以及普通的参数self）。在这个__init__里，们只是创建一个新的域，也称为name。注意它们是两个不同的变量，尽管它们有相同的名字。点号我们能够区分它们。
 最重要的是，我们没有专门调用__init__方法，只是在创建一个类的新实例的时候，把参数包括圆括号内跟在类名后面，从而传递给__init__方法。这是这种方法的重要之处。
 现在，我们能够在我们的方法中使用self.name域。这在sayHi方法中得到了验证。
 给C++/Java/C#程序员的注释
 __init__方法类似于C++、C#和Java中的 construc or 。
 5、类与对象的方法
 我们已经讨论了类与对象的功能部分，现在我们来看一下他的数据部分。事实上，它们只是与和对象的名称空间绑定的普通变量，即这些名称只在这些类与对象的前提下奏效。
 有两种类型的域--类的变量和对象的变量他们根据是类还是对象，拥有这个变量的而区分。类的量由一个类的所有对象（实例）共享使用。只是一个类变量的拷贝，所以当某个对象的变量做了改动时候。这个改动会反应到所有其他的实例上。
 对象的变量由类的每个对象（实例）拥有。因此每个对象有自己对这个域的拷贝即他们是不共

的，在同一个类的不同实例中，虽然对象的变量有相同的名称，但是是互不相关的。（这一点和java类的静态变量和对象的变量类似）