链滴

python 学习 --- 面向对象的编程续一（五）

作者：oldcaptain

原文链接：https://ld246.com/article/1374584440747

来源网站：链滴

<h2> 题外话：关于缩进 </h2>
     python关于缩进，对于新手，或者学过其他语言，然后再学习python对于格可能有些不习惯。同时也没有{}，刚开始我也为这种格式蛋疼了一会，由于缩进有问题，连编译都编不了。但是如果你选择了一款正确的编辑器，然后再严格按照这种风格去编码，你会发现你的pytho代码是如此的漂亮。在这里我推荐pycharm ，于注册什么的，自己可以在网上搜搜注册码，这是一款我认为比较好用的编辑器。希望你也能喜欢。/p>
 注：前面四部分，都是我参看《pytho简明教程》 很容易理解，所以就将其中的一些比较重要的概念罗列了出来，接下来的小，将会加入自己的理解，并且逐步深入，请各位网友多加指点
<h2> python的面向对象深入 </h2>

<pre>#-*- coding:utf-8 -*-
__author__ = 'oldcaptain.cc'

class SchoolMember:
'''代表任何学校成员'''

def __init__(self,name,age):
    self.name = name
    self.age = age
    print '初始化的成员： %s' % self.name

def tell(self):
    '''说话细节'''
    print '名称："%s" age : "%s"'%(self.name,self.age)

class Teacher(SchoolMember):
'''代表一个老师'''
def **init**(self,name,age,salary):
SchoolMember.**init**(self,name,age)
self.salary = salary
print '初始化的老师有 : %s' %self.name
def tell(self):
SchoolMember.tell(self)
print '科目 : "%d" ' % self.sala
y

class Student(SchoolMember):
def **init**(self,name,age,marks):
SchoolMember.**init**(self,name,age)
self.marks = marks
print '初始化的学生为 : %s' %self.name
def tell(self):

```
SchoolMember.tell(self)
print 'marks : &quot;%d&quot;' %s
lf.marks

t = Teacher(&quot;oldcaptain&quot;
40,30000)
s = Student(&quot;guopeng&quot;
22,75)</pre>

<p> <br /> 上面这段代码是一段关于继承的实现，表现出了多态，并且展现出了一些关于python面
对象的一些语法知识。 </p>
<h3> 1、pass关键字 </h3>
<p> 当这个类中没有没有定义任何关键字或者方法的时候，可以使用关键字pass，相当于java中的{}
么也没有，一个空的类。 </p>
<h3> 2、缩进 </h3>
<p> 在类中的所有东西都缩进，if，函数，for，等等，当不缩进的代码则被认为不属于这个类。 </p

<h3> 3、继承 </h3>
<p> 在python中，类的基类只是简单的列在类名后面的小括号中，如果你想要多重继承，那么就在
名后面的小括号中列出你要继承的类名，并且以逗号进行分割。在java中，父类的方法在子类方法执
前被自动调用，python不是这样，你必须显示的调用父类中的合适方法。 </p>
<h3> 4、再说__init__方法 </h3>
<p> __init__在类的实例被创建后被立即调用。他可能会让你误解为构造函数。因为他看上去像（在
中，习惯被第一个定义的方法）行为也像（在一个新创建的实例中，它是第一个被调用的方法），说
不正确时因为对象再调用init方法时，已经被构造出来了，你已经有了 一个对类的的新实例的有效引
。但是__init__是在python中你可以得到最接近构造函数的东西。并且他也扮演者非常相似的角色。 <
p>
<h3> 5、参数 </h3>
<p> 每个类的方法的第一个参数，包括__init__,都是指向类当前的实例的引用。按照习惯这个参数总
被称为self。在__init__方法中，self指向新创建的对象；在其他的类方法中，它指向方法被调用的类
例。尽管当定义方法时你需要明确指定self，但在调用方法时，你不能指定他，python会给你自动加
的。 </p>
<p> __init__方法可以接受任意数目的参数，就像函数一样，参数可以用缺省值定义，即可以设置成
于调用者可选。 </p>
<h3> 6、self </h3>
<p> self相当于c++或者java中的this，但是他并不是一个保留字，它只是一个命名习惯。所以请一
坚持这个习惯。 </p>
<h3> 7、何时使用self和__init__ </h3>
<p> &amp;nbsp;当你定义自己的类方法时，你必须明确的将self作为每个方法的第一个参数列出，包括__i
it__ ,当你从类中调用一个父类的方法时，你必须包括self参数，但当你从类的外部调用你的类方法时
你不必对self参数指定任何值；你完全将其忽略，而python会自动替你增加实例的引用。刚开始这里
能不是很好理解。所以需要慢慢在代码中理解这些特性。 </p>
<p> 注： __init__方法是可选的，但是一旦定义了之后，就必须要显示调用父类的__init__方法（如果
定义了的话）这样才是正确的：无论何时子类想要扩展父类的行为，后代方法必须在适当的时机，使
适当的参数，显示调用父类方法。 </p>
<h3> 8、类的实例 </h3>
<p> 每一个类的实例有一个内置属性，_class_,它是对象的类。（注意这个表示，在内存中的地址，
过我自己测试，这里并不是物理地址。） </p>
<p> 在python中，每一个对象都有自己的元数据属性，包括__class__,__dict__,__module__等. </p>
<h3> 9、参数的重载 </h3>
<p> 在java中支持通过参数列表的重载，也就是一个类可以同名的方法，但这些方法或者是参数个数
同，或者参数的类型不同，像plsql支持参数名的重载。在python中两种都不支持，总之没有任何形
的函数重载。一个__init__方法就是一个__init__方法，不管什么样的参数，一个类中只能有一个__init_
```

方法，并且如果一个子类拥有一个__init__方法，他总是覆盖父类的__init__方法，甚至子类可以用不的参数列表来定义它。
 python作者解释： 方法的覆盖： 子类可以覆盖父类中的方法。因为方法没有特殊的优先的设，父类中的一个方法在调用同类中的另一个方法时，可能调用到的是子类中覆盖的覆盖的父类的同名法的方法。
 -----未完-----