链滴

由 @property 引起的对 Python 装饰器（Decorator）的学习

作者：GodFather

原文链接：https://ld246.com/article/1362722808781

来源网站：链滴

今天看别人的代码遇到了@property，让我不由想起Java中的注解，难道Python中也有这东西

经过一番查找，在Python中这个叫做装饰器（Decorator）。

一、装饰器的基本知识

1. 装饰器有两种：不带参数的装饰器，带参数的装饰器。

2. 分别用一句话来描述这两种装饰器：不带参数的装饰器是一个用来装饰函数的以该函数为参数的数；带参数的装饰器是一个返回一个用来装饰函数的以该函数为参数的函数（或者说是一个返回不带数装饰器的函数）。嘿嘿，有点拗口哈，不过貌似确实是这样的，求指导……

3. 当然，不管是带参数还是不带参数的装饰器，都需要返回一个函数，在调用被装饰的函数的时候实际上调用的就是装饰器返回的函数。

4. 不带参数的装饰器：

     定义：

<pre>def deco(func):
.....</pre>

     使用：

<pre>@deco
def foo():
pass</pre>

     Python的处理方式：

<pre>foo=deco(foo)</pre>

5. 带参数的装饰器：

     定义：

<pre>def deco(args):
.....</pre>

     使用：

<pre>   @deco(args)
    def foo():
        pass</pre>

     Python的处理方式：

<pre>   foo = deco(args)(foo)</pre>

6. 使用多个装饰器，如：

     ① 不带参数的装饰器

<pre>   @deco1
    @deco2
    def foo():
        pass</pre>

     Python的处理方式是：

<pre>   foo = deco1(deco2(foo))</pre>

     ② 带参数的装饰器

<pre>   @deco1
    @deco2(args)
    def foo():
        pass</pre>

     Python的处理方式是：

<pre>   foo = deco1(deco2(args)(foo))</pre>

7. 一个装饰器的例子，实现了打印函数调用的时间：

<pre>from time import ctime
def printCallTime(func):
    def decoratedFunc():

```
        print '[%s] %s() is called.' % (ctime(),func.__name__)
        func()
    return decoratedFunc

@printCallTime
def myFunc():
pass

myFunc()</pre>

</div>
<div>
 代码执行后会输出：
</div>
<blockquote>
 <div>
  [Fri Mar 08 13:23:27 2013] myFunc() is called.
 </div>
</blockquote>
<div>
 8. 学习了装饰器的基本知识，回到最开始的@property，通过文档我了解到，它的作用是将一个函
作为类的一个属性来访问。如：
</div>
<div>
 <pre>class Parrot(object):
    def __init__(self):
        self._voltage = 100000


@property
def voltage(self):
    &amp;quot;&amp;quot;&amp;quot;Get the current voltage.&amp;quot;&amp;quot;&amp;quot;
    return self._voltage</pre>

</div>
<div>
 这样就为类Parrot定义了一个只读属性voltage，如果p是Parrot的一个实例，那我我们就可以通过p.
oltage来访问到voltage函数的返回值，如果想让该属性可写：
</div>
<div>
 <pre>    @voltage.setter
    def voltage(self, value):
        self._voltage = value</pre> &amp;nbsp;
</div>
<div>
 9. 扩展
</div>
<div>
 &amp;nbsp; &amp;nbsp; &amp;nbsp;在Python3.0之前有old-style class 和new-style class之分，帮助文档中是
样介绍new-style class的：
</div>
<blockquote>
 <div>
```

new-style classAny class which inherits from
object. This includes all built-in types like
list and
dict. Only new-style classes can use Python’ s newer, versatile features like
__slots__, descriptors, properties, and
__getattribute__().

</blockquote>

<dl>
<dd>

也就是说继承自object的类才是new-style class 而且只有new-style class 才能设置属性，也即是使用@property的时候，所在的类必须是继承自object的。

</dd>
</dl>

