Vamei

编程，数学，设计

Python深入05 装饰器

作者：Vamei 出处：http://www.cnblogs.com/vamei 欢迎转载，也请保留这段声明。谢谢！

装饰器(decorator)是一种高级Python语法。装饰器可以对一个函数、方法或者类进行加工。在Python中，我们有多种方法对函数和类进行加工，比如在Python闭包中，我们见到函数对象作为某一个函数的返回结果。相对于其它方式，装饰器语法简单，代码可读性高。因此，装饰器在Python项目中有广泛的应用。

装饰器最早在Python 2.5中出现，它最初被用于加工函数和方法这样的可调用对象(callable object，这样的对象定义有__call__方法)。在Python 2.6以及之后的Python版本中，装饰器被进一步用于加工类。

装饰函数和方法

我们先定义两个简单的数学函数，一个用来计算平方和，一个用来计算平方差：

```
# get square sum
def square_sum(a, b):
    return a**2 + b**2

# get square diff
def square_diff(a, b):
    return a**2 - b**2

print(square_sum(3, 4))
print(square_diff(3, 4))
```

在拥有了基本的数学功能之后，我们可能想为函数增加其它的功能，比如打印输入。我们可以改写函数来实现这一点：

```
# modify: print input

# get square sum
def square_sum(a, b):
    print("intput:", a, b)
    return a**2 + b**2

# get square diff
def square_diff(a, b):
    print("input", a, b)
    return a**2 - b**2

print(square_sum(3, 4))
print(square_diff(3, 4))
```

我们修改了函数的定义，为函数增加了功能。

现在，我们使用装饰器来实现上述修改：

```
def decorator(F):
    def new_F(a, b):
        print("input", a, b)
        return F(a, b)
    return new_F

# get square sum
@decorator
def square_sum(a, b):
    return a**2 + b**2

# get square diff
@decorator
def square_diff(a, b):
```

```
    return a**2 - b**2

print(square_sum(3, 4))
print(square_diff(3, 4))
```

装饰器可以用def的形式定义，如上面代码中的decorator。装饰器接收一个可调用对象作为输入参数，并返回一个新的可调用对象。装饰器新建了一个可调用对象，也就是上面的new_F。new_F中，我们增加了打印的功能，并通过调用F(a, b)来实现原有函数的功能。

定义好装饰器后，我们就可以通过@语法使用了。在函数square_sum和square_diff定义之前调用@decorator，我们实际上将square_sum或square_diff传递给decorator，并将decorator返回的新的可调用对象赋给原来的函数名(square_sum或square_diff)。 所以，当我们调用square_sum(3, 4)的时候，就相当于：

```
square_sum = decorator(square_sum)
square_sum(3, 4)
```

我们知道，Python中的变量名和对象是分离的。变量名可以指向任意一个对象。从本质上，装饰器起到的就是这样一个重新指向变量名的作用(name binding)，让同一个变量名指向一个新返回的可调用对象，从而达到修改可调用对象的目的。

与加工函数类似，我们可以使用装饰器加工类的方法。

如果我们有其他的类似函数，我们可以继续调用decorator来修饰函数，而不用重复修改函数或者增加新的封装。这样，我们就提高了程序的可重复利用性，并增加了程序的可读性。

含参的装饰器

在上面的装饰器调用中，比如@decorator，该装饰器默认它后面的函数是唯一的参数。装饰器的语法允许我们调用decorator时，提供其它参数，比如@decorator(a)。这样，就为装饰器的编写和使用提供了更大的灵活性。

```
# a new wrapper layer
def pre_str(pre=''):
    # old decorator
    def decorator(F):
        def new_F(a, b):
            print(pre + "input", a, b)
            return F(a, b)
        return new_F
    return decorator

# get square sum
@pre_str('^_^')
def square_sum(a, b):
    return a**2 + b**2

# get square diff
@pre_str('T_T')
def square_diff(a, b):
    return a**2 - b**2

print(square_sum(3, 4))
print(square_diff(3, 4))
```

上面的pre_str是允许参数的装饰器。它实际上是对原有装饰器的一个函数封装，并返回一个装饰器。我们可以将它理解为一个含有环境参量的闭包。当我们使用@pre_str('^_^')调用的时候，Python能够发现这一层的封装，并把参数传递到装饰器的环境中。该调用相当于：

```
square_sum = pre_str('^_^') (square_sum)
```

装饰类

在上面的例子中，装饰器接收一个函数，并返回一个函数，从而起到加工函数的效果。在Python 2.6以后，装饰器被拓展到类。一个装饰器可以接收一个类，并返回一个类，从而起到加工类的效果。

```
def decorator(aClass):
    class newClass:
        def __init__(self, age):
            self.total_display   = 0
            self.wrapped         = aClass(age)
        def display(self):
            self.total_display += 1
            print("total display", self.total_display)
            self.wrapped.display()
    return newClass

@decorator
class Bird:
    def __init__(self, age):
        self.age = age
    def display(self):
        print("My age is",self.age)

eagleLord = Bird(5)
for i in range(3):
    eagleLord.display()
```

在decorator中，我们返回了一个新类newClass。在新类中，我们记录了原来类生成的对象（self.wrapped），并附加了新的属性total_display，用于记录调用display的次数。我们也同时更改了display方法。

通过修改，我们的Bird类可以显示调用display的次数了。

总结

装饰器的核心作用是name binding。这种语法是Python多编程范式的又一个体现。大部分Python用户都不怎么需要定义装饰器，但有可能会使用装饰器。鉴于装饰器在Python项目中的广泛使用，了解这一语法是非常有益的。

欢迎继续阅读“Python快速教程”

标签: Python

好文要顶 关注我 收藏该文

Vamei
关注 - 26
粉丝 - 4985

14 0

荣誉：推荐博客

+加关注

(请您对文章做出评价)

« 上一篇：Python简史

» 下一篇：版本管理三国志 (CVS, Subversion, git)

posted @ 2013-02-16 17:18 Vamei 阅读(20545) 评论(25) 编辑 收藏

评论列表

#1楼 2013-02-16 20:33 Chenkun

已阅！支持～

支持(0) 反对(0)

#2楼 2013-02-16 22:37 金色驼铃

谢谢楼主分享

支持(0) 反对(0)

#3楼[楼主] 2013-02-16 22:42 Vamei

@ Chenkun

下一步不知道写什么了。

支持(0) 反对(0)

#4楼 2013-02-16 23:01 Chenkun

@ Vamei

其实挺期待Python深入这个系列的！我一直希望能在这个系列中看到讲解Python内部机制的部分，如：Python是如何处理类型，如何解释执行Python代码，Python中的模块是怎么加载的，与其他语言混合编程（如C语言）

也希望LZ能坚持下去！加油～ :-)

支持(0) 反对(0)

#5楼[楼主] 2013-02-16 23:08 Vamei

@ Chenkun

嗯，我正准备写一个混合编程的内容。

支持(0) 反对(0)

#6楼 2013-02-20 12:29 reverland

终于讲到decorator了……

支持(1) 反对(0)

#7楼 2013-07-05 11:02 zylzjj

第一个decorator时候，如果new_F(),不带参数的话会报错:TypeError: new_F() takes no arguments (2 given), 这意思是在当把F传入decorator时a, b 就已经赋值给new_F中a，b了么？不是很明白啊？希望楼主能讲解下，谢谢！

支持(0) 反对(0)

#8楼 2013-07-05 11:07 zylzjj

明白了，原来a，b是当作全局变量传过去的啊。square_sum()的话会报错：NameError: global name 'a' is not defined。没有仔细想就发言打扰了，Sorry！

支持(0) 反对(0)

#9楼[楼主] 2013-07-05 15:12 Vamei

@ zylzjj

自问自答，值得鼓励！ ^_^

支持(0) 反对(0)

#10楼 2013-11-17 15:53 杨琼

楼主“Python 深入”这一系列写的通俗易懂，冒昧问下博主是读哪一本书？

支持(0) 反对(0)

#11楼[楼主] 2013-11-17 20:54 Vamei

@ 杨琼

参考了好多本书，以及网上的许多资料。你可以看下面的豆列:

http://book.douban.com/doulist/1619790/

支持(1) 反对(0)

#12楼 2014-03-11 18:08 assasszt

```
1  #类装饰器
2  def decorator_class(classA):
3      class newClass:
4          def __init__(self,age):
5              self.a = classA
6              self.num = 0
7          def run(self):
8              self.num+=1
9              print self.num," RUN."
10             self.a.run()
11     return newClass
12
13 @decorator_class
14 class class_a:
15     def __init__(self,age):
16         self.age = age
17     def run(self):
18         print "age:",age,"run..."
19
20 a = class_a
21 a.run()
```

为什么出了异常？

TypeError: unbound method run() must be called with newClass instance as first a

rgument (got nothing instead)

跟你的代码 没差别啊？

支持(0) 反对(0)

#13楼 2014-04-03 13:31 少帅胆小混混

@ assasszt

看了下代码错了 3个地方，

I：self.a=classA()即self.a存放的是被装饰类的对象实例不是类

II：第18行print "age:",age,"run..."应为print "age:",self.age,"run..."，作用域不对

III:a=class_a 第20行应这样实例化对象a=class_a(2)

貌似你比较急躁，心情复杂O(∩_∩)O

支持(0) 反对(0)

#14楼[楼主] 2014-04-04 15:49 Vamei

@ 少帅胆小混混

谢谢帮忙改正哈～

支持(0) 反对(0)

#15楼 2014-04-17 11:50 assasszt

@ 少帅胆小混混

引用

@assasszt

看了下代码错了 3个地方，

I：self.a=classA()即self.a存放的是被装饰类的对象实例不是类

II：第18行print "age:",age,"run..."应为print "age:",self.age,"run..."，作用域不对

III:a=class_a 第20行应这样实例化对象a=class_a(2)

貌似你比较急躁，心情复杂O(∩_∩)O

@Vamei

引用

@少帅胆小混混

谢谢帮忙改正哈～

我给自己每天安排一些任务，其中就有每天完成一个博主的python系列。

当然还有其他的，还有公司的工作

所以确实有些急。。

谢了啊。。。

支持(0) 反对(0)

#16楼 2014-05-21 02:57 本本乱

问个问题，装饰器装饰函数的例子，是可以在原来函数运行之前，进行一些额外的处理。最后把新函数return出去。

那如果我想在原来的函数运行结束之后，进行一些额外处理，那应该怎么写？

支持(0) 反对(0)

#17楼[楼主] 2014-05-21 10:02 Vamei

@ 本本乱

你可以在上面的F(a, b)先不返回，再多做些处理，然后再返回F(a, b)

支持(0) 反对(0)

#18楼 2014-05-21 20:24 本本乱

@ Vamei

谢谢回复！自己捣鼓了一下，貌似可以。

暂且贴一下代码，给其他人做参考吧。

```
#!/usr/bin/env python

def printLogs(F):
    def new_F(a, b):
        print 'function start'
        funcToReturn = F(a, b)
        print 'function end'
        return funcToReturn
    return new_F

@printLogs
def square_sum(a, b):
    print 'square_sum() is processing...'
    return a ** 2 + b ** 2

print square_sum(3, 4)
```

运行结果：

```
PastgiftMacbookPro:python pastgift$ ./decoratorTest.py
function start
square_sum() is processing...
function end
```

```
25
```

支持(0) 反对(0)

#19楼 2014-05-21 20:36 本本乱

另外，多个装饰器也是可以的：

```
#!/usr/bin/env python

def printLog(F):
    def new_F(a, b):
        print 'function start'
        funcToReturn = F(a, b)
        print 'function end'
        return funcToReturn
    return new_F

def printArgs(F):
    def new_F(a, b):
        print 'args:', a, b
        funcToReturn = F(a, b)
        return funcToReturn
    return new_F

@printLog
@printArgs
def square_sum(a, b):
    print 'square_sum() is processing...'
    return a ** 2 + b ** 2

print square_sum(3, 4)
```

运行结果：

```
PastgiftMacbookPro:python pastgift$ ./decoratorTest.py
function start
args: 3 4
square_sum() is processing...
function end
25
```

支持(2) 反对(0)

#20楼[楼主] 2014-05-22 11:05 Vamei

@ 本本乱

赞这个探索

支持(0) 反对(0)

#21楼 2014-07-25 13:20 laocan

@decorator(f2)=lambda x,y:x+y+1

@decorator f2=lambda x,y:x+y+1

decorator不能使用lambda表达式吗？

像上面这样子都会报错误

谢谢～

支持(0) 反对(0)

#22楼 2015-03-24 21:41 赵明威

```
>>> print square_diff(3,4)
('input', 3, 4)
-7
>>> square_sum = decorator(square_sum)
>>> square_sum(3,4)
('input', 3, 4)
('input', 3, 4)
25
```

我这怎么出来两个输出啊?求指导.我是你的粉丝

支持(0) 反对(0)

#23楼 2015-04-03 10:47 Leo Yun

@ 赵明威

仔细看这段话 "定义好装饰器后，我们就可以通过@语法使用了。在函数square_sum和..." 就明白了。

你的写法，在square_sum定义上用@decorator之后，就不需要再手动调用一遍decorator(square_sum)，直接用square_sum即可。 输出两次('input', 3, 4）是因为你的写法相当于装饰了两次。

支持(0) 反对(0)

#24楼 2015-09-02 11:17 Nefeltari

@ 本本乱

多个装饰器，执行顺序是怎样的？

输出结果为什么是这样的？每个装饰器里面都有funcToReturn=F(a,b)，不会执行两次吗？

```
function start
args: 3 4
square_sum() is processing...
function end
25
```

支持(1) 反对(0)

#25楼 2015-12-02 17:40 疯狂乱抓

@ Nefeltari

顺序按照@的顺序。

同问为什么只执行一次

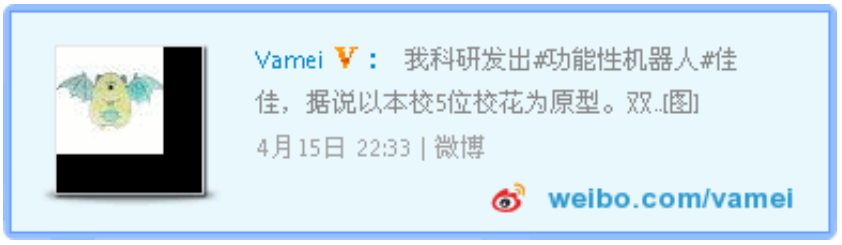
支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

公告

你好，这里是Vamei，一名编程爱好者。我在博客里写了**Python/Linux/**网络协议**/**算法**/Java/**数据科学系列文章，从这里开始阅读。非常期待和你的交流。

我的微博 

下列教程已经做成电子出版物，内容经过修订，也方便离线阅读：

协议森林

欢迎阅读我写的其他书籍：

现代小城的考古学家

天气与历史的相爱相杀

随手拍光影

昵称：Vamei
园龄：4年1个月
荣誉：推荐博客
粉丝：4985
关注：26
+加关注

常用链接

我的随笔

我的评论

我的参与

最新评论

我的标签

我的标签

Python(61)

Java(42)

大数据(22)

Linux(17)

网络(16)

算法(15)

文青(14)

技普(9)

系列索引(6)

开发工具(4)

更多

系列文章

Java快速教程

Linux的概念与体系

Python快速教程

数据科学

协议森林

纸上谈兵: 算法与数据结构

积分与排名

积分 - 659668

排名 - 122

最新评论

1. Re:Java基础11 对象引用

受教!

--MisslLost

2. Re:Python快速教程

看评论区一片喝彩！看来我得在此扎营了！

--测试小蚂蚁

3. Re:Python进阶06 循环对象

好好地列表解析变成了表推导

--ashic

4. Re:“不给力啊，老湿！”：RSA加密与破解

感谢楼主精彩分享

--worldball

5. Re:概率论04 随机变量

你写的这一系列太棒了，刚加入博客园就在你这里学到了，我要转载过去学习一下

--yixius

6. Re:Python基础03 序列

挺好的教程、、、、

--王小拽的号

7. Re:Python进阶07 函数对象

def func(x,y): print x**ydef test(f,a,b): print 'test' print f(a,b)test (func,3,2)

输出的内容:tes.....

--M-edea

8. Re:Python进阶02 文本文件的输入输出

@coderXT换行符：\n...

--行者之印

9. Re:数据科学

博主啊，这里是一枚即将大二的计算机新人，大一学了python，java，还有一些算法，数据结构，图论了，感觉我对数学又一些反感，但是听说离散数学对计算机专业的很重要，不知道怎么去学比较好呢，我想像您写.....

--Acokil

10. Re:为什么要写技术博

楼主是用自己自定义的模板吗？在博客园里找不到这种风格的blog模板？

--行者之印

11. Re:来玩Play框架01 简介

挖煤哥,我补充了一下Windows下的搭建play框架,希望有点帮助,谢谢!

--Sungeek

12. Re:来玩Play框架07 静态文件

@helper.form(action = routes.Application.upload, 'enctype -> "multipart/form-data") {--action = rout.....

--quxiaozha

13. Re:来玩Play框架07 静态文件

该记录将/assets/下的URL，对应到项目的/public文件夹内的文件。比如在项目的/public/images/test.jpg，就可以通过/assests/images/test.jpg这一.....

--quxiaozha

14. Re:来玩Play框架06 用户验证

支持挖煤哥~~~

--quxiaozha

15. Re:“不给力啊，老湿！”：RSA加密与破解

@maanshancss请你仔细阅读了这个文章再来评价。...

--Vamei

推荐排行榜