

Vamei

编程, 数学, 设计

博客园 首页 订阅 管理

随笔-209 文章-1 评论-3802

Python深入01 特殊方法与多范式

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

Python一切皆对象, 但同时, Python还是一个多范式语言 (multi-paradigm), 你不仅可以使⤵面向对象的方式来编写程序, 还可以用面向过程的方式来编写相同功能的程序 (还有函数式、声明式等, 我们暂不深入)。Python的多范式依赖于Python对象中的特殊方法 (special method)。

特殊方法名的前后各有两个下划线。特殊方法又被成为魔法方法 (magic method), 定义了许多Python语法和表达方式, 正如我们在下面的例子中将要看到的。当对象中定义了特殊方法的时候, Python也会对它们有“特殊优待”。比如定义了__init__()方法的类, 会在创建对象的时候自动执行__init__()方法中的操作。

(可以通过dir()来查看对象所拥有的特殊方法, 比如dir(1))

运算符

Python的运算符是通过调用对象的特殊方法实现的。比如:

```
'abc' + 'xyz' # 连接字符串
```

实际执行了如下操作:

```
'abc'.__add__('xyz')
```

所以, 在Python中, 两个对象是否能进行加法运算, 首先就要看相应的对象是否有__add__()方法。一旦相应的对象有__add__()方法, 即使这个对象从数学上不可加, 我们都可以用加法的形式, 来表达obj.__add__()所定义的操作。在Python中, 运算符起到简化书写的功能, 但它依靠特殊方法实现。

Python不强制用户使用面向对象的编程方法。用户可以选择自己喜欢的使用方式(比如选择使用+符号, 还是使用更加面向对象的`__add__()`方法)。特殊方法写起来总是要更费事一点。

尝试下面的操作, 看看效果, 再想想它的对应运算符

```
(1.8).__mul__(2.0)
```

```
True.__or__(False)
```

内置函数

与运算符类似, 许多内置函数也都是调用对象的特殊方法。比如

```
len([1, 2, 3])          # 返回表中元素的总数
```

实际上做的是

```
[1, 2, 3].__len__()
```

相对与`__len__()`, 内置函数`len()`也起到了简化书写的作用。

尝试下面的操作, 想一下它的对应内置函数

```
(-1).__abs__()
```

```
(2.3).__int__()
```

表 (list) 元素引用

下面是我们常见的表元素引用方式

```
li = [1, 2, 3, 4, 5, 6]
print(li[3])
```

上面的程序运行到`li[3]`的时候, Python发现并理解`[]`符号, 然后调用`__getitem__()`方法。

```
li = [1, 2, 3, 4, 5, 6]
print(li.__getitem__(3))
```

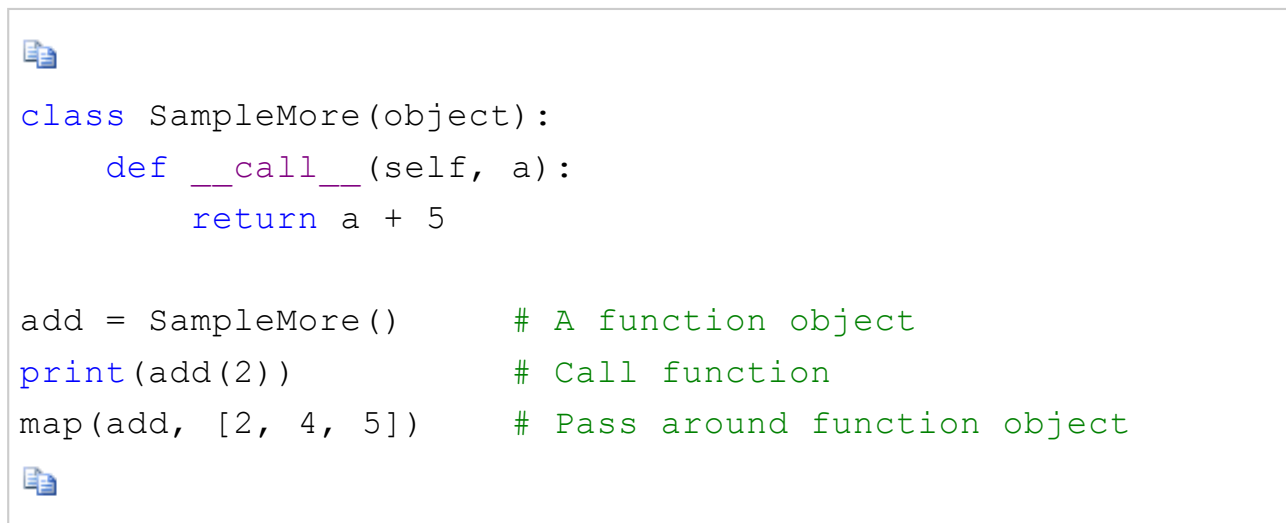
尝试看下面的操作，想想它的对应

```
li.__setitem__(3, 0)

{'a':1, 'b':2}.__delitem__('a')
```

函数

我们已经说过，在Python中，函数也是一种对象。实际上，任何一个有`__call__()`特殊方法的对象都被当作是函数。比如下面的例子：



```
class SampleMore(object):
    def __call__(self, a):
        return a + 5

add = SampleMore()      # A function object
print(add(2))           # Call function
map(add, [2, 4, 5])     # Pass around function object
```

`add`为`SampleMore`类的一个对象，当被调用时，`add`执行加5的操作。`add`还可以作为函数对象，被传递给`map()`函数。

当然，我们还可以使用更“优美”的方式，想想是什么。

总结

对于内置的对象来说（比如整数、表、字符串等），它们所需要的特殊方法都已经在Python中准备好了。而用户自己定义的对象也可以通过增加特殊方法，来实现自定义的语法。特殊方法比较靠近Python的底层，许多Python功能的实现都要依赖于特殊方法。我们将在以后看到更多的例子。



大黄蜂，还是Camaro跑车

Python的许多语法都是基于其面向对象模型的封装。对象模型是Python的骨架，是功能完备、火力强大的大黄蜂。但是Python也提供更加简洁的语法，让你使用不同的编程形态，从而在必要时隐藏一些面向对象的接口。正如我们看到的Camaro跑车，将自己威风的火药库收起来，提供方便人类使用的车门和座椅。

标签: [Python](#)

好文要顶

关注我

收藏该文



Vamei

关注 - 26

粉丝 - 4985

荣誉: [推荐博客](#)

[+加关注](#)

10

0

(请您对文章做出评价)

« 上一篇: [为什么要写技术博](#)

» 下一篇: [Python深入02 上下文管理器](#)

posted @ 2012-11-19 23:20 Vamei 阅读(22431) 评论(28) 编辑 收藏

评论列表

#1楼 2012-11-20 09:41 Chenkun

写的很好，支持一下！

另外：楼主用的是哪个版本的python， 文中的属性小节中__getattr__应该是__getattribute__吧， 通常我们使用的时候会封装一下， 如下：

```
1 class student(object):
2     def __getattr__(self, name):
3         return object.__getattribute__(name)
```

支持(0) 反对(0)

#2楼 2012-11-20 10:21 zhuangzhuang1988

@ Chenkun

我也是的 没看到..

支持(0) 反对(0)

#3楼 2012-11-20 10:33 Chenkun

@ zhuangzhuang1988

我又去翻了一下官方的文档， 在3.4.2. Customizing attribute access 节中有对__getattr__的解释：)

支持(0) 反对(0)

#4楼[楼主] 2012-11-20 10:42 Vamei

@ Chenkun

我把它想简单了。等我再读一下官方文档看怎么修改，谢谢。

支持(0) 反对(0)

#5楼[楼主] 2012-11-20 10:59 Vamei

@ Chenkun

@zhuangzhuang1988

我的原意是用__dict__，但是由于还没有提到过__dict__，所以用了__getattr__，但看来是用错了。

整个过程是先用__dict__搜查属性，如果没有，向上找__base__的属性。如果整个

树的没有，那么调用__getattr__来生成属性。所以说__dict__和__getattr__是相互配合工作的关系。

而__getattribute__则是无条件的返回属性，无论这一属性是否存在，所以比较“暴力”。

我参考了

<http://stackoverflow.com/questions/4295678/understanding-the-difference-between-getattr-and-getattribute>

谢谢你们的提醒。

支持(0) 反对(0)

#6楼 2012-11-20 13:51 snake117

受教了

支持(0) 反对(0)

#7楼 2012-11-23 10:33 沐訖

想问下

```
def xxx():
```

支持(0) 反对(1)

#8楼 2012-11-23 10:34 沐訖

```
def xx():
```

```
def yy():
```

```
pass
```

```
def zz():
```

```
pass
```

这样写有什么优势 总是觉得这个很奇怪

支持(0) 反对(0)

#9楼[楼主] 2012-11-23 11:27 Vamei

@ 沐訖

你在哪里看到的呢？

pass可以在开发时候预留空间。

支持(0) 反对(0)

#10楼 2012-11-23 13:44 沐訖

```
1  def xx():
2  def yy():
3      pass
4  def zz():
5      pass
```

贴代码乱了 囧

就是在方法内在定义方法

支持(0) 反对(0)

#11楼 2012-11-23 13:45 沐訖

```
def xx():
---def yy():
----pass
---def zz():
----pass
```

支持(0) 反对(0)

#12楼[楼主] 2012-11-23 14:02 Vamei

@ 沐訖

方法内定义方法有很多原因。一个是可以减少memory的占用。在Python里很常见的就是用于闭包，以便保存函数所在的情境。

支持(0) 反对(0)

#13楼 2012-11-23 14:14 沐訖

我见过一种情况是吧xx 当作装饰器 根据情况调用xx 或者yy 很有意思
感谢 解答

支持(0) 反对(0)

#14楼[楼主] 2012-11-23 15:53 Vamei

@ 沐訖

:-)

支持(0) 反对(0)

#15楼 2012-12-17 14:03 Honghe

之前觉得Python的对象很凌乱，现在觉得是对其对象模型理解不够深。

期待博主继续深入

支持(0) 反对(0)

#16楼 2013-01-16 23:35 bells

"当然，我们还可以使用更“优美”的方式，想想是什么。"

是什么？？ 没想到呀

支持(0) 反对(0)

#17楼[楼主] 2013-01-17 09:57 Vamei

@ bells

直接用def定义函数...

支持(0) 反对(0)

#18楼 2013-01-23 10:58 鸠斑兔

即使这个对象是否从数学上（不）可加

少了个字吧

支持(0) 反对(0)

#19楼[楼主] 2013-01-23 11:18 Vamei

@ tuzkee

你说的对，语法错误。谢谢提醒。

支持(0) 反对(0)

#20楼 2013-09-23 17:30 miqingren

更优雅的办法应该是 `lambda x: x+5`, list的写法吧

支持(1) 反对(0)

#21楼 2013-11-17 14:59 杨琼

学习了。

支持(0) 反对(0)

#22楼 2015-04-28 11:18 劈马砍柴

同问 更优雅的是 lambda写法么?

支持(0) 反对(0)

#23楼 2015-06-06 13:27 pyers

```
>>> map(lambda a:a+5,[2,4,5])  
[7, 9, 10]  
>>>
```

支持(0) 反对(0)

#24楼 2015-09-01 21:12 ArtinHuang

博主你好，我想问一下下面两行代码对应的具体含义是什么？个人不是很清楚，谢谢

```
1 | li.__setitem__(3, 0)  
2 | {'a':1, 'b':2).__delitem__('a')
```

支持(1) 反对(0)

#25楼 2016-01-09 16:15 madm4n

我是新手，有个问题想问一下

```
1 | class SampleMore(object):  
2 |     hases_feather = True  
3 |     def __call__(self,a):  
4 |         return a+5  
5 | lol = SampleMore()  
6 | print lol(2)
```

这里__call__换成任意名称qwer,然后print lol.qwer(2)

这两个之间有什么区别

支持(0) 反对(0)

#26楼 2016-02-20 13:14 ErikY

@ madm4n

qwer这个是类的实例属性的调用。

__call__是使这个实例属性变成一个方法并调用

支持(0) 反对(0)

#27楼 2016-04-21 17:08 casalye

```
>>> class sample(object):  
def _call_(self,a):  
return a+5
```

```
>>> add=sample()  
>>> print add(2)
```

```
Traceback (most recent call last):  
File "<pyshell#14>", line 1, in <module>  
print add(2)  
TypeError: 'sample' object is not callable
```

为什么会这样啊0.0

支持(0) 反对(0)

#28楼 2016-06-08 15:36 python自学土鳖

受教了，看了你的第一篇，我就受益匪浅啊，谢谢

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。

【推荐】50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库

【推荐】融云即时通讯云—豆果美食、Faceu等亿级APP都在用



JPush 极光推送

消息推送领导品牌全面升级




JIGUANG | 极光

[详情点击](#)

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。



Vamei ：我科研发出#功能性机器人#佳佳，据说以本校5位校花为原型。双双[图]

4月15日 22:33 | 微博



weibo.com/vamei

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！](#)

--MissLost

[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

1. “不给力啊，老湿！”：RSA加密与破解(218)
2. Python快速教程(140)
3. 野蛮生长又五年(91)
4. Java快速教程(88)
5. 协议森林01 邮差与邮局 (网络协议概观)(79)
6. 为什么要写技术博(71)
7. 编程异闻录(54)
8. 博客一年：心理之旅(49)
9. 协议森林08 不放弃 (TCP协议与流通信)(45)
10. Python快速教程 尾声(43)
11. 协议森林(42)
12. Java基础01 从HelloWorld到面向对象(42)
13. Python基础08 面向对象的基本概念(40)
14. 一天能学会的计算机技术(34)
15. 博客第二年，杂谈(33)

Copyright ©2016 Vamei

05370166