

# 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03 对象的属性

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

Python一切皆对象(object), 每个对象都可能多个属性(attribute)。  
Python的属性有一套统一的管理方案。

### 属性的\_\_dict\_\_系统

对象的属性可能来自于其类定义, 叫做类属性(class attribute)。类属性可能来自类定义自身, 也可能根据类定义继承来的。一个对象的属性还可能是该对象实例定义的, 叫做对象属性(object attribute)。

对象的属性储存在对象的\_\_dict\_\_属性中。\_\_dict\_\_为一个词典, 键为属性名, 对应的值为属性本身。我们看下面的类和对象。chicken类继承自bird类, 而summer为chicken类的一个对象。



```
class bird(object):
    feather = True

class chicken(bird):
    fly = False
    def __init__(self, age):
        self.age = age

summer = chicken(2)

print(bird.__dict__)
print(chicken.__dict__)
```

```
print(summer.__dict__)
```



下面为我们的输出结果：



```
{'__dict__': <attribute '__dict__' of 'bird' objects>,
'__module__': '__main__', '__weakref__': <attribute
'__weakref__' of 'bird' objects>, 'feather': True,
'__doc__': None}

{'fly': False, '__module__': '__main__', '__doc__': None,
'__init__': <function __init__ at 0x2b91db476d70>}

{'age': 2}
```



第一行为bird类的属性，比如feather。第二行为chicken类的属性，比如fly和\_\_init\_\_方法。第三行为summer对象的属性，也就是age。有一些属性，比如\_\_doc\_\_，并不是由我们定义的，而是由Python自动生成。此外，bird类也有父类，是object类（正如我们的bird定义，class bird(object)）。这个object类是Python中所有类的父类。

可以看到，Python中的属性是分层定义的，比如这里分为object/bird/chicken/summer这四层。当我们需要调用某个属性的时候，Python会一层层向上遍历，直到找到那个属性。（某个属性可能出现再不同的层被重复定义，Python向上的过程中，会选取先遇到的那一个，也就是比较低层的属性定义）。

当我们有一个summer对象的时候，分别查询summer对象、chicken类、bird类以及object类的属性，就可以知道summer对象所有的\_\_dict\_\_，就可以找到通过对象summer可以调用和修改的所有属性了。**下面两种属性修改方法等效：**

```
summer.__dict__['age'] = 3
```


```
print(summer.__dict__['age'])

summer.age = 5
print(summer.age)
```

(上面的情况中, 我们已经知道了summer对象的类为chicken, 而chicken类的父类为bird。如果只有一个对象, 而不知道它的类以及其他信息的时候, 我们可以利用\_\_class\_\_属性找到对象的类, 然后调用类的\_\_base\_\_属性来查询父类)

## 特性

同一个对象的不同属性之间可能存在依赖关系。当某个属性被修改时, 我们希望依赖于该属性的其他属性也同时变化。这时, 我们不能通过\_\_dict\_\_的方式来静态的储存属性。Python提供了多种即时生成属性的方法。其中一种称为特性(property)。特性是特殊的属性。比如我们为chicken类增加一个特性adult。当对象的age超过1时, adult为True; 否则为False:




```
class bird(object):
    feather = True

class chicken(bird):
    fly = False
    def __init__(self, age):
        self.age = age
    def getAdult(self):
        if self.age > 1.0: return True
        else: return False
    adult = property(getAdult) # property is built-in

summer = chicken(2)

print(summer.adult)
summer.age = 0.5
print(summer.adult)
```



特性使用内置函数`property()`来创建。`property()`最多可以加载四个参数。前三个参数为函数，分别用于处理查询特性、修改特性、删除特性。最后一个参数为特性的文档，可以为一个字符串，起说明作用。

我们使用下面一个例子进一步说明：



```
class num(object):
    def __init__(self, value):
        self.value = value
    def getNeg(self):
        return -self.value
    def setNeg(self, value):
        self.value = -value
    def delNeg(self):
        print("value also deleted")
        del self.value
    neg = property(getNeg, setNeg, delNeg, "I'm negative")

x = num(1.1)
print(x.neg)
x.neg = -22
print(x.value)
print(num.neg.__doc__)
del x.neg
```



上面的`num`为一个数字，而`neg`为一个特性，用来表示数字的负数。当一个数字确定的时候，它的负数总是确定的；而当我们修改一个数的负数时，它本身的值也应该变化。这两点由`getNeg`和`setNeg`来实现。而`delNeg`表示的是，如果删除特性`neg`，那么应该执行的操作是删除属性`value`。`property()`的最后一个参数(`"I'm negative"`)为特性`negative`的说明文档。

## 使用特殊方法 `__getattr__`

我们可以用 `__getattr__(self, name)` 来查询即时生成的属性。当我们查询一个属性时，如果通过 `__dict__` 方法无法找到该属性，那么Python会调用对象的 `__getattr__` 方法，来即时生成该属性。比如：



```
class bird(object):
    feather = True

class chicken(bird):
    fly = False
    def __init__(self, age):
        self.age = age
    def __getattr__(self, name):
        if name == 'adult':
            if self.age > 1.0: return True
            else: return False
        else: raise AttributeError(name)

summer = chicken(2)

print(summer.adult)
summer.age = 0.5
print(summer.adult)

print(summer.male)
```



每个特性需要有自己的处理函数，而 `__getattr__` 可以将所有的即时生成属性放在同一个函数中处理。`__getattr__` 可以根据函数名区别处理不同的属性。比如上面我们查询属性名 `male` 的时候，`raise AttributeError`。

(Python中还有一个 `__getattribute__` 特殊方法，用于查询任意属性。

`__getattr__` 只能用来查询不在 `__dict__` 系统中的属性)

`__setattr__(self, name, value)` 和 `__delattr__(self, name)` 可用于修改和删除属性。它们的应用面更广，可用于任意属性。

## 即时生成属性的其他方式

即时生成属性还可以使用其他方式，比如descriptor(descriptor类实际上是property()函数的底层，property()实际上创建了一个该类的对象)。有兴趣可以进一步查阅。

## 总结

\_\_dict\_\_ 分层存储属性。每一层的\_\_dict\_\_只存储该层新增的属性。子类不需要重复存储父类中的属性。

即时生成属性是值得了解的概念。在Python开发中，你有可能使用这种方法来更合理的管理对象的属性。

分类: [Python](#)

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

9

0

(请您对文章做出评价)

« 上一篇: [协议森林08 不放弃 \(TCP协议与流通信\)](#)

» 下一篇: [Python深入04 闭包](#)

posted @ 2012-12-11 22:16 Vamei 阅读(23182) 评论(24) 编辑 收藏

### 评论列表

#1楼 2012-12-11 22:33 Chenkun

nice! 很详细!

支持(0) 反对(0)

---

## #2楼 2012-12-12 08:55 zhuangzhuang1988

不错不错不错,很好很好

查了下还有这种方法

```
1 class C(object):
2     @property
3     def x(self): return self._x
4     @x.setter
5     def x(self, value): self._x = value
6     @x.deleter
7     def x(self): del self._x
```

支持(1) 反对(0)

---

## #3楼[楼主] 2012-12-12 09:13 Vamei

@ zhuangzhuang1988

这个就是一种property的用法，只是用了decorator.

支持(0) 反对(0)

---

## #4楼[楼主] 2012-12-12 09:14 Vamei

@ Chenkun

谢谢

支持(0) 反对(0)

---

## #5楼 2013-02-22 13:55 mmufhy

hi, 我对property()不是很理解:

property(fget=None, fset=None, fdel=None, doc=None) -> property  
attribute

如果我要修改property，需要先执行一个fset函数，然后得到property值，那为什么不直接每次都使用fget函数呢？这样就可以不用定义property了。

支持(0) 反对(0)

---

## #6楼[楼主] 2013-02-22 14:03 Vamei

@ mmufhy

没有太明白你的意思

当你在调用属于property的属性时候，fget是自动执行的。

fset是你想手动修改属于property的属性时执行的。

支持(0) 反对(0)

#7楼 2013-02-22 14:32 mmufhy

@ Vamei

拿你上面的“negative”代码举个例子。在这个例子中，我使用的时候，x.neg和x.getNeg()应该是一个意思。那我既然已经有了x.getNeg()方法，为什么还要定义一个property 'neg'呢？反正两者是等效的。

支持(0) 反对(0)

#8楼[楼主] 2013-02-22 14:43 Vamei

@ mmufhy

neg虽然是属性，但不是方法，而getNeg()是一个方法。

你当然可以每次都调用getNeg，并使用函数返回结果。

但有时你想要的是一个非方法的属性，比如一个整数或者字符串，直接作为属性。

我想，审美是这样做的主要原因。

比如，我定义一个“人”的对象，你会更加自然的将：

身高(float)、年龄(int)、体重(float)作为属性，而不是总是去调用“获得身高”，“获得年龄”，“获得体重”这样的方法。

支持(1) 反对(0)

#9楼 2013-02-22 16:39 mmufhy

@ Vamei

了解了。这跟JAVA语法推荐的不大一样。JAVA是非常不推荐直接访问对象的属性的，一般都是用getxxx()的方法返回要使用的属性值。

谢谢。

支持(0) 反对(1)

#10楼 2013-09-23 00:05 lwkjob

```
1 class bird(object):
2     feather = True
3
4 class chicken(bird):
5     fly = False
6     def __init__(self, age):
7         self.age = age
8     def __getattr__(self, name):
```



```
9         if name=='adult':
10             if self.age > 1.0:return True
11             else:return False
12         else:raise AttributeError(name)
13
14     summer = chicken(2)
15
16     print(summer.adult)
17     summer.age = 0.5
18     print(summer.adult)
19
20     print(summer.male)
```

奇怪不知道哪里写错了 对照了无数遍还是 没找到什么地方不一样导致执行不成功

支持(0) 反对(0)

#11楼 2014-03-05 14:13 特务小强

@ Vamei

引用

@zhuangzhuang1988

这个就是一种property的用法，只是用了decorator.

这个也太不利于代码维护了，代码可读性不好

支持(0) 反对(0)

#12楼 2014-03-07 16:24 assasszt

感谢博主分享，清晰、易懂、不深入，学完之后就会有框架在那里，太好了。

我一点一点的看过来的，问个问题

我练习 写了特性的 的第二个例子，就是三个方法的那个：

```
neg = property(getNeg, setNeg, delNeg, "I'm negative")
```

结果打印neg 是三个 方法的内存地址，（类似<function setNum at 0x00000000025E3AC8） + 最后说明的那句话，这就是neg的值？

那“自动改变值”在哪里体现了（第一个例子很明确）

支持(0) 反对(0)

### #13楼[楼主] 2014-03-08 00:23 Vamei

@ assasszt

-1.1

22

I'm negative

value also deleted

我直接运行那段代码后的结果是这样的。

比如说我输入的是1.1，这里出现的是-1.1。

这就是getNeg()发挥了作用。

你直接打印neg，是打印类里的neg，还是对象的neg？可否把原码贴出来？

支持(0) 反对(0)

---

### #14楼 2014-03-11 17:42 assasszt

@ Vamei

没事了，我又仔细检查了一下

是我把 set和get 的位置放反了

支持(0) 反对(0)

---

### #15楼 2014-05-10 23:18 逍遥22

\_\_getattr\_\_可以根据函数名区别处理不同的属性。比如上面我们查询属性名male的时候，raise AttributeError。楼主，那个函数名感觉更应该是属性名吧？

支持(1) 反对(0)

---

### #16楼 2014-06-12 22:35 Dust.ww

你真的好厉害啊，会这么多程序语言还这么精。小白一个，现在正在学习力两年前的博文，求大神指导。

支持(0) 反对(0)

---

### #17楼 2014-06-13 13:06 Dust.ww

这一节感觉好难啊。。。没怎么懂。

支持(0) 反对(0)

---

### #18楼 2014-08-02 16:24 Mr\_Q

python的很多思想确实和java不一样...

支持(0) 反对(0)

### #19楼 2014-10-12 20:31 fupeng

内容非常有货， 赞。

支持(0) 反对(0)

### #20楼 2014-10-12 20:39 fupeng

@ lwkjob

从你代码来看 16 18 20 这三行个有一个打印函数，前两句是可以正常执行的，第20行执行会有错误。

这个错误是正常的，因为你没有定义male这个属性，这章不就是讲这个自定义属性吗？

因为18行的属性定义了，所以执行不出错，

20行 调用了一个没有定义的属性，就出错了。不出错才 坏事了呢。

支持(0) 反对(0)

### #21楼 2014-11-30 21:32 jeffsoft.h

@ Vamei

引用

@mmufhy

neg虽然是属性，但不是方法，而getNeg()是一个方法。

你当然可以每次都调用getNeg，并使用函数返回结果。

但有时你想要的是一个非方法的属性，比如一个整数或者字符串，直接作为属性。

我想，审美是这样做的主要原因。

比如，我定义一个“人”的对象，你会更加自然的将：

身高(float)、年龄(int)、体重(float)作为属性，而不是总是去调用“获得身高”，“获得年龄”，“获得体重”这样的方法。

=====

因为python的类成员没有private,public之分啊。属性本质上是有一个对庆的get()和set()方法，即使像这里的“特殊属性”，在其他语言中，get()方法肯定是private或proceted的，但python没法private,所以，只好连get也暴露出来了

支持(0) 反对(0)

### #22楼 2015-01-08 19:45 黑夜不是我

@ jeffsoft.h

引用

@Vamei

引用

引用@mmufhy

neg虽然是属性，但不是方法，而getNeg()是一个方法。

你当然可以每次都调用getNeg，并使用函数返回结果。

但有时你想要的是一个非方法的属性，比如一个整数或者字符串，直接作为属性。

我想，审美是这样做的主要原因。

比如，我定义一个“人”的对象，你会更加自然的将：

身高(float)、年龄(int)、体重(float)作为属性，而不是总是去调用“获得身高”，“获得年龄”，“获得体重”这样的方法。

=====

因为python的类成员没有private,public之分啊。属性本质上是有一个对庆的...

我记得python的类成员中只要在成员名前加加两个下划线前缀就是私有的，不能直接访问，但可以用\_classname+私有成员名 访问

支持(0) 反对(0)

#23楼 2016-04-01 15:54 Missingsour

提个问题，为什么class bird(object)没有\_\_init\_\_()属性？

支持(0) 反对(0)

#24楼 2016-04-01 16:05 Missingsour

不好意思，看岔了，bird就是没有用到\_\_init\_\_()

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。

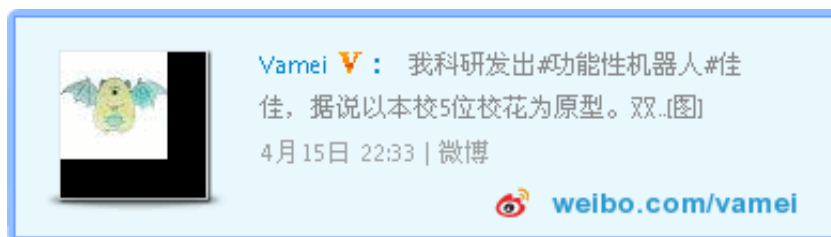
【推荐】50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库

【推荐】融云即时通讯云一豆果美食、Faceu等亿级APP都在用



## 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！](#)

--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/assets/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

05370162