Vamei

编程，数学，设计

Python基础09 面向对象的进一步拓展

我们熟悉了对象和类的基本概念。我们将进一步拓展，以便能实际运用对象和类。

调用类的其它信息

上一讲中提到，在定义方法时，必须有self这一参数。这个参数表示某个对象。对象拥有类的所有性质，那么我们可以通过self，调用类属性。

```
class Human(object):
    laugh = 'hahahaha'
    def show_laugh(self):
        print self.laugh
    def laugh_100th(self):
        for i in range(100):
            self.show_laugh()

li_lei = Human()
li_lei.laugh_100th()
```

这里有一个类属性laugh。在方法show_laugh()中，通过self.laugh，调用了该属性的值。

还可以用相同的方式调用其它方法。方法show_laugh()，在方法laugh_100th中()被调用。

通过对象可以修改类属性值。但这是危险的。类属性被所有同一类及其子类的对象共享。类属性值的改变会影响所有的对象。

__init__()方法

__init__()是一个特殊方法(special method)。Python有一些特殊方法。Python会特殊的对待它们。特殊方法的特点是名字前后有两个下划线。

如果你在类中定义了__init__()这个方法，创建对象时，Python会自动调用这个方法。这个过程也叫初始化。

```
class happyBird(Bird):
    def __init__(self,more_words):
        print 'We are happy birds.',more_words

summer = happyBird('Happy,Happy!')
```

这里继承了Bird类，它的定义见上一讲。

屏幕上打印：

```
We are happy birds.Happy,Happy!
```

我们看到，尽管我们只是创建了summer对象，但__init__()方法被自动调用了。最后一行的语句(summer = happyBird...)先创建了对象，然后执行：

summer.__init__(more_words)

'Happy,Happy!' 被传递给了__init__()的参数more_words

对象的性质

我们讲到了许多属性，但这些属性是类的属性。所有属于该类的对象会共享这些属性。比如说，鸟都有羽毛，鸡都不会飞。

在一些情况下，我们定义对象的性质，用于记录该对象的特别信息。比如说，人这个

类。性别是某个人的一个性质，不是所有的人类都是男，或者都是女。这个性质的值随着对象的不同而不同。李雷是人类的一个对象，性别是男；韩美美也是人类的一个对象，性别是女。

当定义类的方法时，必须要传递一个self的参数。这个参数指代的就是类的一个对象。我们可以通过操纵self，来修改某个对象的性质。比如用类来新建一个对象，即下面例子中的li_lei，那么li_lei就被self表示。我们通过赋值给self.attribute，给li_lei这一对象增加一些性质，比如说性别的男女。self会传递给各个方法。在方法内部，可以通过引用self.attribute，查询或修改对象的性质。

这样，在类属性的之外，又给每个对象增添了各自特色的性质，从而能描述多样的世界。

```
class Human(object):
    def __init__(self, input_gender):
        self.gender = input_gender
    def printGender(self):
        print self.gender

li_lei = Human('male') # 这里，'male'作为参数传递给__init__()方法的input_gender变量。
print li_lei.gender
li_lei.printGender()
```

在初始化中，将参数input_gender，赋值给对象的性质，即self.gender。

li_lei拥有了对象性质gender。gender不是一个类属性。Python在建立了li_lei这一对象之后，使用li_lei.gender这一对象性质，专门储存属于对象li_lei的特有信息。

对象的性质也可以被其它方法调用，调用方法与类属性的调用相似，正如在printGender()方法中的调用。

总结

通过self调用类属性

__init__()：在建立对象时自动执行

类属性和对象的性质的区别

标签: Python

好文要顶 关注我 收藏该文

Vamei
关注 - 26
粉丝 - 4985

27 0

荣誉：推荐博客

+加关注

(请您对文章做出评价)

« 上一篇：Python基础08 面向对象的基本概念
» 下一篇：Python基础10 反过头来看看

posted @ 2012-06-02 17:03 Vamei 阅读(48573) 评论(35) 编辑 收藏

评论列表

#1楼 2012-07-14 21:23 micfan

简明易懂，注释得当，希望以后可以看到python小项目

支持(1) 反对(0)

#2楼 2012-09-02 21:52 luoboiqingcai

nice,对于初学者甚好

支持(0) 反对(0)

#3楼 2012-12-27 17:27 next163

self相当于 *this

__init__() 差不多算是构造。

支持(2) 反对(0)

#4楼 2013-02-08 15:37 峻祁连

非常好的教程！，感谢！

不过有一句不太明白，您提到 ： （在方法中更改类变量属性的值是危险的，这样会影响根据这个类定义的所有对象的这一属性！！）

我做了个实验：

```
class Human(object):
    Can_Talk = True
    Can_Walk = True
    Age = 0
    Name = ""

    def Say(self, msg):
        print "I am saying: " + msg


class Child(Human):
    def Cry(self):
        print "wa wa ...."

    def ShowAge(self):
        print self.Name, " is " , self.Age ," years old."

    def Grow(self, yr) :
        self.Age = yr


Jerry = Child()
Jerry.Name = "Jerry"
Jerry.Age = 3
Jerry.Grow(4)
Jerry.ShowAge()

Daniel = Child()
Daniel.Name = "Daniel"
Daniel.Grow(1)
Daniel.ShowAge()
```

输出结果：

C:\Python27>python c:\if.py

Jerry is 4 years old.

Daniel is 1 years old.

我在Jerry实例的Grow（）方法中更改了类变量属性Age的值，没发现影响到其他对象Daniel啊？ 能详细解释一下吗？

支持(1) 反对(0)

#5楼[楼主] 2013-02-08 18:52 Vamei

@ 峻祁连

事实上，我也不知道当时为什么这么写！！！

等我过年后回来看吧，要回家休假了。

支持(1) 反对(0)

#6楼[楼主] 2013-02-08 18:53 Vamei

@ 峻祁连

谢谢你提醒我。

支持(0) 反对(0)

#7楼[楼主] 2013-02-16 16:41 Vamei

@ 峻祁连

你的这种写法可行，是因为你的属性都是immutable的(比如整数、字符串)。但如果属性是mutable的话(比如list)，就会出现问题。比如下面的代码：

```
1  class Human(object):
2      Can_Talk = True
3      Can_Walk = True
4      Age = 0
5      Name = ["Li", "Lei"]
6
7
8  a = Human()
9  b = Human()
10
11 a.Age += 1
12 print a.Age
13 print b.Age
14
15 a.Name[0] = "Wang"
16 print a.Name
17 print b.Name
```

为什么immutable是可行的呢？原因是，在更改对象属性时，如果属性是immutable的，该属性会被复制出一个副本，存放在对象的__dict__中。你可以通过下面的方式查看：

print a.__class__.__dict__

print a.__dict__

注意到类中和对象中各有一个Age。一个为0， 一个为1。所以我们在查找a.Age的时候，会先查到对象的__dict__的值，也就是1。

但mutable的类属性，在更改属性值时，并不会有新的副本。所以更改会被所有的对象看到。

所以，为了避免混淆，最好总是区分类属性和对象的属性，而不能依赖上述的immutable属性的复制机制。

支持(7) 反对(0)

#8楼[楼主] 2013-02-16 16:44 Vamei

@ 峻祁连

嗯，我修改了一个原文的叙述。

支持(0) 反对(0)

#9楼 2013-02-21 15:55 Hejin.Wong

@ 峻祁连

不清楚Python的对象内存怎么分配的

C#的类（对象）的不同实例会开辟出新的内存，不同实例的相同属性内存不一样的

以此类推：你修改了Jerry的age这个属性的值，不会影响到Daniel 的age属性 两个age只是同名 但是地址不一样 没有必然联系

支持(0) 反对(0)

#10楼[楼主] 2013-02-21 22:12 Vamei

@ egger

具体的内存实现我没有查。

在Python中, 类的属性和对象的属性是两个概念，尽管在@峻祁连举出的例子中，Python(有时)会自动创建与类属性同名的对象属性，这有点让人混淆。

我觉得好的习惯是遵循简单的原则：使用__init__初始化对象属性值是好习惯。

支持(2) 反对(0)

#11楼 2013-02-23 21:57 pang18a

@ egger

http://blog.csdn.net/hsuxu/article/details/7785835

我也对这里有点迷惑 后来百度了下

大概的意思好像是说 如果是个可变类型的变量 那么赋值时传递的是引用修改的是同一块内存 如果是不可变类型的变量 赋值时 会重新开辟一块内存

支持(2) 反对(0)

#12楼[楼主] 2013-02-24 09:46 Vamei

@ pang18a

对，是这样的。

支持(0) 反对(0)

#13楼 2013-06-18 16:38 MeT

（通过对象来修改类属性是危险的，这样可能会影响根据这个类定义的所有对象的这一属性！！）

关于这一句，情况是这样的：

1、对象不能修改类的属性，只能修改自己的，也就是说，修改了之后对同类的其他对象没有影响；

2、动态修改类属性可以用类名.属性 = xxx来进行修改；

3、修改的类属性一般会影响所辖对象的属性，除非对象在此之前对该属性进行过修改。

支持(4) 反对(0)

#14楼 2014-02-12 12:45 helloray

@MeT

这才是正解

支持(0) 反对(0)

#15楼 2014-02-17 14:03 特务小强

immutable这是基础知识了。

支持(0) 反对(0)

#16楼 2014-02-22 10:33 smirkymonkey

楼主真是可爱

深入浅出 看着happyhappy

支持(0) 反对(0)

#17楼 2014-03-20 17:07 yexuan910812

所谓的类成员和对象成员，是不是就是就是Java中static成员与对象成员的区别？

支持(0) 反对(0)

#18楼 2014-03-27 14:41 Triangle23

最后一个例子中定义的Human类，类中的gender不需要声明吗？

支持(0) 反对(0)

#19楼 2014-04-06 09:45 choovin

支持

支持(0) 反对(0)

#20楼 2014-04-22 11:58 red.hu

python 类在实例化的时候，如果成员变量的为引用类型，那么实例化的属性值均为同样的引用值；

可以理解为，class在加载的时候，引用类型的数据就已经在内存块里了；

而在java类实例化的时候，会在内存中开辟一块新的内存，再赋值引用类型的给成员变量。

支持(0) 反对(0)

#21楼 2014-04-22 15:15 Seandor

恩，list immutable，不能乱改。

支持(0) 反对(0)

#22楼 2014-06-13 00:53 ZJA+

@ Triangle23

试了一下，不需要声明，Python应该自动识别为self特有性质，然后保存在该self下吧。

支持(0) 反对(0)

#23楼 2014-08-01 16:23 Riordon

```
>>> class Human(object):
Can_Talk = True
Can_Walk = True
Age = 0
Name = ["Li", "Lei"]
```

```
>>> a = Human()
>>> b = Human()
>>> a.Age += 1
>>> print a.Age
1
>>> print b.Age
0
>>> a.Name[0] = "Wang"
>>> print a.Name
['Wang', 'Lei']
>>> print b.Name
['Wang', 'Lei']
```

>>>@ helloray 楼主才是正解....

支持(0) 反对(0)

#24楼 2014-09-10 22:29 fosmj

@ MeT

你说的第一点无法解释楼主在评论中给的例子，你的用在java中适用

支持(0) 反对(0)

#25楼 2014-11-26 17:10 jeffsoft.h

是不是可以这样说：

对于对象中的引用型属性，同一对象的不同实例，共享一个引用地址（直到某个实例重新为属性赋值——这时将开辟新的存储空间）。

这一点和C#是不一样的。

支持(0) 反对(0)

#26楼 2014-12-02 20:01 Agureo

@ ZJA+

a=1，也不用事先声明int a，是不是一个道理呀

支持(0) 反对(0)

#27楼 2014-12-24 15:00 arzard

好棒的，大赞，对于初学者特别的有帮助，深入浅出，谢谢啦！

支持(0) 反对(0)

#28楼 2015-03-12 15:00 sowings

通过讨论可以理解的更透彻。赞一个

支持(0) 反对(0)

#29楼 2015-03-30 16:46 pythonerJ

@ pang18a

没错，我测试了下，修改字符串类型的类属性，的确修改的是同一内存地址的值。

```
class Human(object):
laugh='hahahhaha'
def show_laugh(self):
print self.laugh
def mod_laugh(self):
self.laugh="55555555"
def mod_property(self,x):
self.laugh=x



richie=Human()
richie.mod_property("wuwuwuwuw")
richie.show_laugh()
```

支持(0) 反对(0)

#30楼 2015-05-08 17:13 elivans

我有点疑惑，Human类的子类对象会继承gender这个性质吗？

支持(0) 反对(0)

#31楼 2015-08-10 10:42 p木

求解

```
>>> class Human(object):
def __init__(self, input_gender):
self.gender = input_gender
def printGender(self):
print(self.gender)
```

```
>>> lilei=Human('男')
>>> print(lilei.gender)
男
>>> lilei.printGender()
男
>>> class boy(Human):
name='john'
def showage(self,age):
print('my age is',age)


>>> john=boy()
Traceback (most recent call last):
File "<pyshell#54>", line 1, in <module>
john=boy()
TypeError: __init__() missing 1 required positional argument:
'input_gender'
```

支持(0) 反对(0)

#32楼 2016-04-01 17:37 wqh2016

这一节看的 还是 比较吃力的

支持(0) 反对(0)

#33楼 2016-04-05 16:28 Jinkingsley

@ MeT

引用

（通过对象来修改类属性是危险的，这样可能会影响根据这个类定义的所有对象的这一属性！！）

关于这一句，情况是这样的：

1、对象不能修改类的属性，只能修改自己的，也就是说，修改了之后对同类的其他对象没有影响；

2、动态修改类属性可以用类名.属性 = xxx来进行修改；

3、修改的类属性一般会影响所辖对象的属性，除非对象在此之前对该属性进行过修改。

感觉第三句话前后矛盾啊，不懂，可以解释一下吗？

支持(0) 反对(0)

#34楼 2016-04-15 17:09 一小步

@ p木

boy继承的是Human，创建对象的时候没有传递给__init__()方法参数，这样肯定会报错的啊

支持(1) 反对(0)

#35楼 2016-05-20 14:46 hao123shu

感谢楼主分享的这个教程，前面的几章都还能够读懂，这章看得有点吃力。

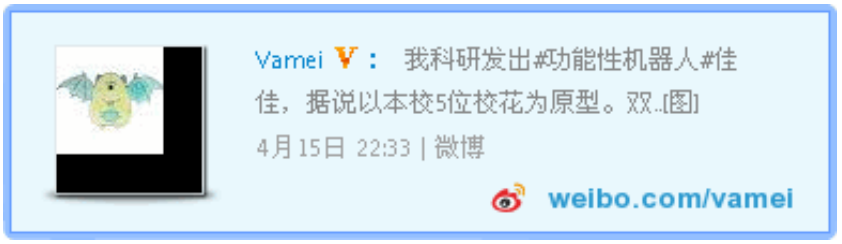
支持(0) 反对(0)

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

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)