

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06 Python的内存管理

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

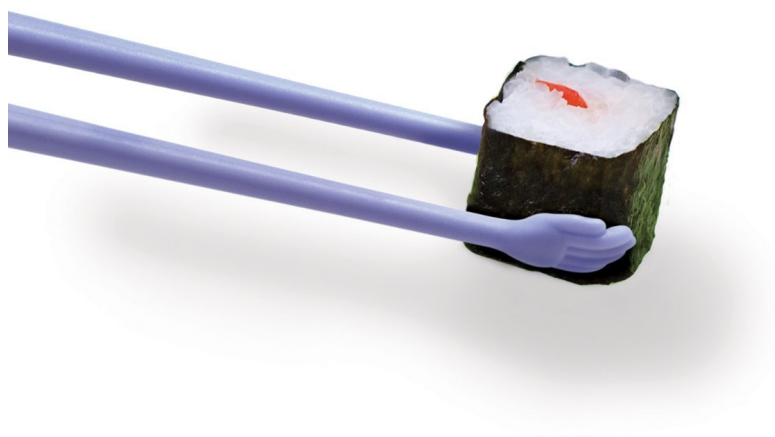
语言的内存管理是语言设计的一个重要方面。它是决定语言性能的重要因素。无论是C语言的手工管理, 还是Java的垃圾回收, 都成为语言最重要的特征。这里以Python语言为例子, 说明一门动态类型的、面向对象的语言的内存管理方式。

对象的内存使用

赋值语句是语言最常见的功能了。但即使是最简单的赋值语句, 也可以很有内涵。Python的赋值语句就很值得研究。

```
a = 1
```

整数1为一个对象。而a是一个引用。利用赋值语句, 引用a指向对象1。Python是动态类型的语言 (参考[动态类型](#)), 对象与引用分离。Python像使用“筷子”那样, 通过引用来接触和翻动真正的食物——对象。



引用和对象

为了探索对象在内存的存储, 我们可以求助于Python的内置函数`id()`。它用于返回对象的身份(identity)。其实, 这里所谓的身份, 就是该对象的内存地址。

```
a = 1

print(id(a))
print(hex(id(a)))
```

在我的计算机上，它们返回的是：

```
11246696
'0xab9c68'
```

分别为内存地址的十进制和十六进制表示。

在Python中，整数和短小的字符，Python都会缓存这些对象，以便重复使用。当我们创建多个等于1的引用时，实际上是让这些引用指向同一个对象。

```
a = 1
b = 1


print(id(a))
print(id(b))
```

上面程序返回

```
11246696
11246696
```

可见a和b实际上是指向同一个对象的两个引用。

为了检验两个引用指向同一个对象，我们可以用is关键字。is用于判断两个引用所指的对象是否相同。

```

# True
a = 1
b = 1
print(a is b)

# True
a = "good"
b = "good"
print(a is b)

# False
a = "very good morning"
b = "very good morning"
print(a is b)

# False
a = []
b = []
print(a is b)
```

上面的注释为相应的运行结果。可以看到，由于Python缓存了整数和短字符串，因此每个对象只存有一份。比如，所有整数1的引用都指向同一对象。即使使用赋值语句，也只是创造了新的引用，而不是对象本身。长的字符串和其它对象可以有多个相同的对象，可以使用赋值语句创建出新的对象。

在Python中，每个对象都有存有指向该对象的引用总数，即引用计数(reference count)。

我们可以使用`sys`包中的`getrefcount()`，来查看某个对象的引用计数。需要注意的是，当使用某个引用作为参数，传递给`getrefcount()`时，参数实际上创建了一个临时的引用。因此，`getrefcount()`所得到的结果，会比期望的多1。

```
from sys import getrefcount

a = [1, 2, 3]
print(getrefcount(a))

b = a
print(getrefcount(b))
```

由于上述原因，两个`getrefcount`将返回2和3，而不是期望的1和2。

对象引用对象

Python的一个容器对象(container)，比如表、词典等，可以包含多个对象。实际上，容器对象中包含的并不是元素对象本身，是指向各个元素对象的引用。

我们也可以自定义一个对象，并引用其它对象：

```
class from_obj(object):
    def __init__(self, to_obj):
        self.to_obj = to_obj

b = [1,2,3]
a = from_obj(b)
print(id(a.to_obj))
print(id(b))
```

可以看到，`a`引用了对象`b`。


对象引用对象，是Python最基本的构成方式。即使是`a = 1`这一赋值方式，实际上是让词典的一个键值"`a`"的元素引用整数对象1。该词典对象用于记录所有的全局引用。该词典引用了整数对象1。我们可以通过内置函数`globals()`来查看该词典。

当一个对象A被另一个对象B引用时，A的引用计数将增加1。

```
from sys import getrefcount
```

```
a = [1, 2, 3]
print(getrefcount(a))

b = [a, a]
print(getrefcount(a))
```

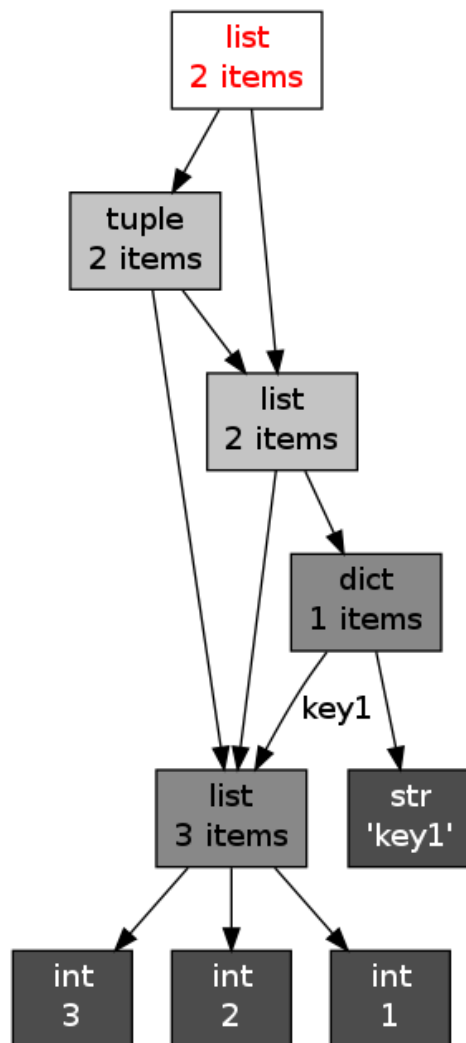


由于对象b引用了两次a，a的引用计数增加了2。

容器对象的引用可能构成很复杂的拓扑结构。我们可以用objgraph包来绘制其引用关系，比如

```
x = [1, 2, 3]
y = [x, dict(key1=x)]
z = [y, (x, y)]

import objgraph
objgraph.show_refs([z], filename='ref_topo.png')
```



objgraph是Python的一个第三方包。安装之前需要安装xdot。

```
sudo apt-get install xdot
```

```
sudo pip install objgraph
```

[objgraph官网](#)

两个对象可能相互引用，从而构成所谓的引用环(reference cycle)。

```
a = []
b = [a]
a.append(b)
```


即使是一个对象，只需要自己引用自己，也能构成引用环。

```
a = []
a.append(a)
print(getrefcount(a))
```


引用环会给垃圾回收机制带来很大的麻烦，我将在后面详细叙述这一点。

引用减少

某个对象的引用计数可能减少。比如，可以使用del关键字删除某个引用：

```

from sys import getrefcount


a = [1, 2, 3]
b = a
print(getrefcount(b))

del a
print(getrefcount(b))

```

del也可以用于删除容器元素中的元素，比如：

```
a = [1, 2, 3]
del a[0]
print(a)
```

如果某个引用指向对象A，当这个引用被重新定向到某个其他对象B时，对象A的引用计数减少：

```

from sys import getrefcount

a = [1, 2, 3]
b = a
```

```
print(getrefcount(b))

a = 1
print(getrefcount(b))
```

垃圾回收

吃太多，总会变胖，Python也是这样。当Python中的对象越来越多，它们将占据越来越大的内存。不过你不用太担心Python的体形，它会乖巧的在适当的时候“减肥”，启动**垃圾回收**(garbage collection)，将没用的对象清除。在许多语言中都有垃圾回收机制，比如Java和Ruby。尽管最终目的都是塑造苗条的提醒，但不同语言的减肥方案有很大的差异（这一点可以对比本文和[Java内存管理与垃圾回收](#)）。



从基本原理上，当Python的某个对象的引用计数降为0时，说明没有任何引用指向该对象，该对象就成为要被回收的垃圾了。比如某个新建对象，它被分配给某个引用，对象的引用计数变为1。如果引用被删除，对象的引用计数为0，那么该对象就可以被垃圾回收。比如下面的表：

```
a = [1, 2, 3]
del a
```

del a后，已经没有任何引用指向之前建立的[1, 2, 3]这个表。用户不可能通过任何方式接触或者动用这个对象。这个对象如果继续待在内存里，就成了不健康的脂肪。当垃圾回收启动时，Python扫描到这个引用计数为0的对象，就将它所占据的内存清空。

然而，减肥是个昂贵而费力的事情。垃圾回收时，Python不能进行其它的任务。频繁的垃圾回收将大大降低Python的工作效率。如果内存中的对象不多，就没有必要总启动垃圾回收。所以，Python只会在特定条件下，**自动启动**垃圾回收。当Python运行时，**会记录其中分配对象(object allocation)和取消分配对象(object deallocation)的次数。当两者的差值高于某个阈值时，垃圾回收才会启动。**

我们可以通过gc模块的**get_threshold()**方法，查看该阈值：

```
import gc
print(gc.get_threshold())
```

返回(700, 10, 10)，后面的两个10是与分代回收相关的阈值，后面可以看到。700即是垃圾回收启动的阈值。可以通过gc中的`set_threshold()`方法重新设置。

我们也可以手动启动垃圾回收，即使用`gc.collect()`。

分代回收

Python同时采用了分代(generation)回收的策略。这一策略的基本假设是，存活时间越久的对象，越不可能在后面的程序中变成垃圾。我们的程序往往会产生大量的对象，许多对象很快产生和消失，但也有一些对象长期被使用。出于信任和效率，对于这样一些“长寿”对象，我们相信它们的用处，所以减少在垃圾回收中扫描它们的频率。



小家伙要多检查

Python将所有的对象分为0, 1, 2三代。所有的新建对象都是0代对象。当某一代对象经历过垃圾回收，依然存活，那么它就被归入下一代对象。垃圾回收启动时，一定会扫描所有的0代对象。如果0代经过一定次数垃圾回收，那么就启动对0代和1代的扫描清理。当1代也经历了一定次数的垃圾回收后，那么会启动对0, 1, 2，即对所有对象进行扫描。

这两个次数即上面`get_threshold()`返回的(700, 10, 10)返回的两个10。也就是说，每10次0代垃圾回收，会配合1次1代的垃圾回收；而每10次1代的垃圾回收，才会有1次的2代垃圾回收。

同样可以用`set_threshold()`来调整，比如对2代对象进行更频繁的扫描。

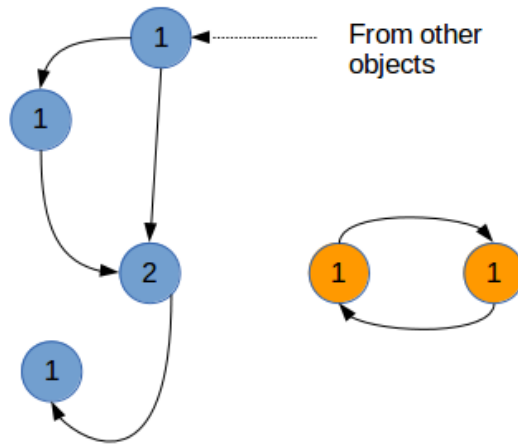
```
import gc
gc.set_threshold(700, 10, 5)
```

孤立的引用环

引用环的存在会给上面的垃圾回收机制带来很大的困难。这些引用环可能构成无法使用，但引用计数不为0的一些对象。

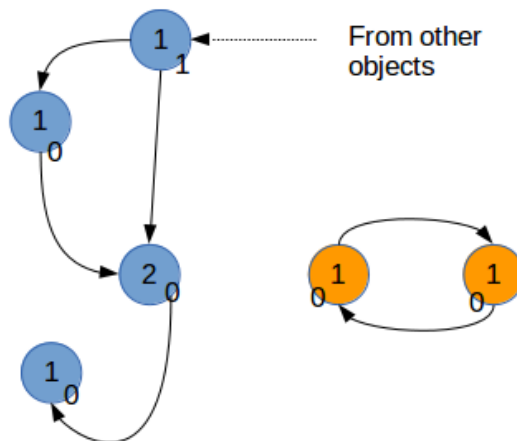
```
a = []  
b = [a]  
a.append(b)  
  
del a  
del b
```

上面我们先创建了两个表对象，并引用对方，构成一个引用环。删除了a，b引用之后，这两个对象不可能再从程序中调用，就没有什么用处了。但是由于引用环的存在，这两个对象的引用计数都没有降到0，不会被垃圾回收。



孤立的引用环

为了回收这样的引用环，Python复制每个对象的引用计数，可以记为`gc_ref`。假设，每个对象`i`，该计数为`gc_ref_i`。Python会遍历所有的对象`i`。对于每个对象`i`引用的对象`j`，将相应的`gc_ref_j`减1。



遍历后的结果

在结束遍历后，gc_ref不为0的对象，和这些对象引用的对象，以及继续更下游引用的对象，需要被保留。而其它的对象则被垃圾回收。

总结

Python作为一种动态类型的语言，其对象和引用分离。这与曾经的面向过程语言有很大的区别。为了有效的释放内存，Python内置了垃圾回收的支持。Python采取了一种相对简单的垃圾回收机制，即引用计数，并因此需要解决孤立引用环的问题。Python与其它语言既有共通性，又有特别的地方。对该内存管理机制的理解，是提高Python性能的重要一步。

欢迎继续阅读“[Python快速教程](#)”

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](#)

23

0

(请您对文章做出评价)

« 上一篇: [概率论11 协方差与相关系数](#)
» 下一篇: [概率论12 矩与矩生成函数](#)

posted @ 2013-11-17 00:13 [Vamei](#) 阅读(31298) 评论(44) 编辑 收藏

评论列表

#1楼 2013-11-17 13:28 [24K纯开源](#)

谢谢分享，objgraph这个包不错！

[支持\(0\)](#) [反对\(0\)](#)

#2楼[楼主] 2013-11-17 20:57 [Vamei](#)

@ [24K纯开源](#)

嗯，我觉得这个包的功能很有趣。

[支持\(0\)](#) [反对\(0\)](#)

#3楼 2013-11-17 23:01 [caochao88](#)

比如某个新建对象，它被分配给某个引用，对象的引用计数变为1。如果引用被删除，对象的引用计数为2，那么该对象就可以被垃圾回收。比如下面的表：

```
a = [1, 2, 3]
del a
```

这里描述有误，被删了后应该为0

支持(2) 反对(0)

#4楼 2013-11-18 21:13 kylinfish

讲得非常好，很容易理解！

支持(0) 反对(0)

#5楼 2013-11-18 21:19 寒霭

很好~

支持(0) 反对(0)

#6楼 2013-11-19 11:45 俊仁

赞！！

支持(0) 反对(0)

#7楼[楼主] 2013-11-20 12:19 Vamei

@ [tudas](#)

好的，谢谢你的指出！

支持(0) 反对(0)

#8楼 2013-11-20 20:40 Alexia(minmin)

```
from sys import getrefcount
```

```
a = [1, 2, 3]
```

```
b = a
```

```
print(getrefcount(b))
```

```
a = 1
```

```
print(getrefcount(b))
```

b = a后是不是b和a的引用计数是一样的，所以print(getrefcount(b))结果与print(getrefcount(a))相同？

支持(0) 反对(0)

#9楼[楼主] 2013-11-20 22:22 Vamei

@ [Alexia\(minmin\)](#)

我没有太明白你的意思。a和b如果指向同一个对象的，那么它们的引用计数相同。

支持(0) 反对(0)

#10楼 2013-11-21 08:42 Alexia(minmin)

@ [Vamei](#)

嗯，就是说如果a = 1, b =a,那么getrefcount(a)=getrefcount(b)

支持(0) 反对(0)

#11楼 2014-02-01 12:35 vxciu

挺不错，我也是对垃圾回收机制有点疑虑，博主看法和我一样啊.....

支持(0) 反对(0)

#12楼 2014-02-28 09:04 特务小强

from sys import getrefcount() 多了一个()

支持(0) 反对(0)

#13楼 2014-02-28 13:05 特务小强

同样可以用get_threshold()来调整 应该是set_threshold()

支持(0) 反对(0)

#14楼[楼主] 2014-02-28 14:02 Vamei

@ 特务小强

谢谢订正！

支持(0) 反对(0)

#15楼 2014-03-10 22:28 yvivid

python好像只有常用的 整数，是固定id。

如果 a=1000, b=1000, 一般两个id是不同的。

支持(0) 反对(0)

#16楼 2014-03-15 15:51 霍克依毒间

循环引用的地方有点错误，应该是gc_ref为0的对象保留，而不是不为0的保留

Python遍历时无法达到循环引用的对象，所以这些对象的gc_ref不是0

支持(0) 反对(0)

#17楼[楼主] 2014-03-17 20:34 Vamei

@ 霍克依毒间

gc_ref不是引用计数，是一个新的变量。Python内部机制可以遍历所有对象，包括循环引用的对象。

支持(0) 反对(0)

#18楼 2014-03-17 23:12 霍克依毒间

@ Vamei

明白了，Python遍历的是内部对象而不是变量符号，gc_ref变为0表明只有对象内的引用，而没有外部变量符号的引用，多谢楼主：)

支持(0) 反对(0)

#19楼 2014-04-25 09:46 cnshen

感谢 @Vamei 的分享，这个比看书效果好的多。

顺便说一下，我在 ubuntu 系统 python 2.7.3 下安装 objgraph1.8 版本导入模块时总是报错找不到模块，安装低版本 sudo pip install 'objgraph<1.8' 导入模块成功。

支持(0) 反对(0)

#20楼 2014-05-11 10:07 逍遥22

楼主，我用的是opensuse13.1 用的是搜狐的源，可是找不到objgraph呀？

支持(0) 反对(0)

#21楼[楼主] 2014-05-11 10:44 Vamei

@ 逍遥22

去objgraph官网上看看？

支持(0) 反对(0)

你用pip试试先

支持(0) 反对(0)

原来是我没有装python-pip管理器，所以找不到pip命令。现在好了，太谢谢楼主啦

支持(0) 反对(0)

```
1 #False
2 a = 'a very very very very very very very very very very very very very very very very very very very very very'
3 b = 'a very very very very very very very very very very very very very very very very very very very very very'
4 print(a is b)
5 print(id(a))
6 print(id(b))
```

3077673952

之后我把a,b变成几千字符的字符串，同样返回True,不知道这个短字符串的界限是多少，请楼主赐教？

支持(2) 反对(0)

@Vamei

```
1 Python 2.7.2 (default, Sep 6 2012, 12:21:50)
2 [GCC 4.5.3] on linux2
3 Type "help", "copyright", "credits" or "license" for more information.
4 >>> a = "aaa bbb"
5 >>> b = "aaa bbb"
6 >>> a is b
7 False
8 >>> c = "cccddd"
9 >>> d = "cccddd"
10 >>> c is d
11 True
```

为啥我这个中间只要有“空格”，就不是同一个对象了呢？

支持(0) 反对(0)

#26楼 2014-08-21 16:39 JohnTsai

@ flowjacky

我用python2.7测试结果是false

支持(0) 反对(0)

#27楼 2014-08-28 14:51 燧人氏

请教博主，引用环的回收，是python自主完成，还是需要我们手动写代码实现？

支持(0) 反对(0)

#28楼 2014-11-14 15:30 清明-心若淡定

```
x = [1, 2, 3]
y = [x, dict(key1=x)]
z = [y, (x, y)]
```

```
import objgraph
objgraph.show_refs([y],filename='sample.png')
```

以上代码运行时报如下错误信息：

Graph viewer (xdot) and image renderer (dot) not found, not doing anything else

支持(1) 反对(0)

#29楼 2015-01-29 14:55 尼航

请问下windows底下如何安装objgraph呢

支持(0) 反对(0)

#30楼 2015-02-02 10:38 yqy2011

@ 燧人氏

我感觉引用环的回收，应该是python自主完成的吧。这个还需要博主出来解答一下。^_^

支持(0) 反对(0)

#31楼 2015-02-02 11:24 yqy2011

博主你好，你的文章中提到“分配对象(object allocation)和取消分配对象(object deallocation)”，所谓分配对象是指新建的对象，取消分配对象是引用次数为0的对象吗？

支持(0) 反对(0)

#32楼 2015-02-02 14:24 laughingtree

我用PIP安装了XDOT和OBJGRAPH，运行以下代码：

```
x = [1, 2, 3]
y = [x, dict(key1=x)]
z = [y, (x, y)]
```

```
import objgraph
objgraph.show_refs([y],filename='sample.png')
```

报以下错误：

Graph viewer (xdot) and image renderer (dot) not found, not doing anything else

在网上搜了半天也不知道怎么解决，楼主能告诉我怎么解决吗？好捉急啊！

支持(0) 反对(0)

#33楼 2015-04-03 16:28 Leo Yun

同问，这个问题，哪位大拿知道

Graph viewer (xdot) and image renderer (dot) not found, not doing anything else

查了这个报错原因，是dot这个文件在PATH里没有找到，很奇怪，为什么要找dot而不是xdot

支持(0) 反对(0)

#34楼 2015-06-30 15:28 Tsingcoo

问题跟楼上一样，系统是win8.1,求教博主解答方法

支持(0) 反对(0)

#35楼 2015-08-03 21:14 woodyle

win7 64bit下和上面几位一样报错：

Image renderer (dot) not found, not doing anything else

后来在虚拟机里面的ubuntu上试，是好的，会在同一目录下生成相应的 .png文件。

猜测也许和xdot有关，windows下都是用pip安装的xdot和objgraph，作为python库文件，而ubuntu下为程序文件（sudo apt-get install的软件包）。

支持(0) 反对(0)

#36楼 2015-08-03 21:19 woodyle

@ Vamei

请教博主帮助回答一下我们几位的windows下objgraph报错原因：

Image renderer (dot) not found, not doing anything else

支持(0) 反对(0)

#37楼 2015-09-02 11:36 Nefeltari

@ JohnTsai

我也是2.7，但是测试结果是True

支持(0) 反对(0)

#38楼 2015-09-02 11:39 Nefeltari

@ vamei

我的版本是2.7 运行结果是True

```
1 | a = "very good morning"
2 | b = "very good morning"
3 | print(a is b)
```

支持(0) 反对(0)

#39楼 2015-09-12 20:52 Eledim

为了回收这样的引用环，Python复制每个对象的引用计数，可以记为gc_ref。假设，每个对象i，该计数为gc_ref_i。Python会遍历所有的对象i。对于每个对象i引用的对象j，将相应的gc_ref_j减1。

每个对象i, 该计数为gc_ref_i, 指的是对象1被引用的次数为gc_ref_1。对象2 被引用的次数为gc_ref_2吗

支持(0) 反对(0)

#40楼 2015-09-18 10:53 信息时代来了

楼主讲的很详细, 必须赞一个! 但是对于我们初学者很费劲, 要是能图文并茂深入讲解就好了, 我给你推荐一个不错的博文。

[Python基本语法, python入门到精通\[二\]](#)

最近园子新开的python系列, 一步步从入门开始学习。

支持(0) 反对(2)

#41楼 2016-01-22 21:08 yeayee

内存这块一直似懂非懂, pass

支持(0) 反对(0)

#42楼 2016-05-07 01:03 dreamlofter

@ Nefeltari

如果是控制台中操作, 一句句输入, 结果就是False,

如果是运行整个文件, 结果就是True。

这个过程中解释器应该有做优化吧。

支持(0) 反对(0)

#43楼 2016-05-07 01:07 dreamlofter

查了下, 确切的说在交互环境下, 如果包含了除字母、数字、下划线之外的字符, Python会自动建立新的字符串对象。

支持(0) 反对(0)

#44楼 2016-06-01 18:46 会长

好文章, 我在学习python, 但很多书都没有将python的内存管理模型, 只是讲一些语法。

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

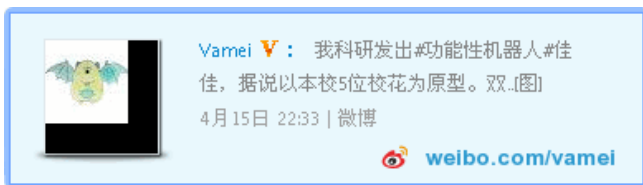


历史上的今天:

2012-11-17 为什么要写技术博

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-edeal

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

05370059