

# 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02 上下文管理器

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

**上下文管理器**(context manager)是Python2.5开始支持的一种语法, 用于规定**某个对象的使用范围**。一旦进入或者离开该使用范围, 会有特殊操作被调用 (比如为对象分配或者释放内存)。它的语法形式是`with...as...`

## 关闭文件

我们会进行这样的操作: 打开文件, 读写, 关闭文件。程序员经常会忘记关闭文件。上下文管理器可以在不需要文件的时候, **自动关闭文件**。

下面我们看一下两段程序:

```
# without context manager
f = open("new.txt", "w")
print(f.closed)           # whether the file is open
f.write("Hello World!")
f.close()
print(f.closed)
```

以及:

```
# with context manager
with open("new.txt", "w") as f:
    print(f.closed)
    f.write("Hello World!")
print(f.closed)
```

两段程序实际上执行的是相同的操作。我们的第二段程序就使用了上下文管理器

(`with...as...`)。上下文管理器有**隶属于它的程序块**。当隶属的程序块执行结束的时候(也就是不再缩进),上下文管理器自动关闭了文件(我们通过`f.closed`来查询文件是否关闭)。我们相当于使用**缩进**规定了文件对象`f`的**使用范围**。

上面的上下文管理器基于`f`对象的`__exit__()`特殊方法(还记得我们如何利用特殊方法来实现各种语法? 参看特殊方法与多范式)。当我们使用上下文管理器的语法时,我们实际上要求Python在进入程序块之前调用对象的`__enter__()`方法,在结束程序块的时候调用`__exit__()`方法。对于文件对象`f`来说,它定义了`__enter__()`和`__exit__()`方法(可以通过`dir(f)`看到)。在`f`的`__exit__()`方法中,有`self.close()`语句。所以在使用上下文管理器时,我们就不用明文关闭`f`文件了。

## 自定义

任何定义了`__enter__()`和`__exit__()`方法的对象都可以用于上下文管理器。文件对象`f`是内置对象,所以`f`自动带有这两个特殊方法,不需要自定义。

下面,我们自定义用于上下文管理器的对象,就是下面的`myvow`:



```
# customized object

class VOW(object):
    def __init__(self, text):
        self.text = text
    def __enter__(self):
        self.text = "I say: " + self.text      # add prefix
        return self                            # note: return
an object
    def __exit__(self, exc_type, exc_value, traceback):
        self.text = self.text + "!"           # add suffix

with VOW("I'm fine") as myvow:
    print(myvow.text)
```

```
print(myvow.text)
```



我们的运行结果如下：

```
I say: I'm fine
I say: I'm fine!
```

我们可以看到，在进入上下文和离开上下文时，对象的`text`属性发生了改变（最初的`text`属性是"`I'm fine`"）。

`__enter__()` 返回一个对象。上下文管理器会使用这一对象作为`as`所指的变量，也就是`myvow`。在`__enter__()`中，我们为`myvow.text`增加了前缀（"`I say:`"）。在`__exit__()`中，我们为`myvow.text`增加了后缀（"`!`"）。

注意：`__exit__()`中有四个参数。当程序块中出现异常（exception），`__exit__()`的参数中`exc_type`, `exc_value`, `traceback`用于描述异常。我们可以根据这三个参数进行相应的处理。如果正常运行结束，这三个参数都是`None`。在我们的程序中，我们并没有用到这一特性。

## 总结：

通过上下文管理器，我们控制对象在程序不同区间的特性。上下文管理器(`with EXPR as VAR`)大致相当于如下流程：



```
# with EXPR as VAR:

VAR = EXPR
VAR = VAR.__enter__()
try:
    BLOCK
finally:
    VAR.__exit__()
```



由于上下文管理器带来的便利，它是一个值得使用的工具。

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](#)

5

0

(请您对文章做出评价)

« 上一篇: [Python深入01 特殊方法与多范式](#)

» 下一篇: [协议森林01 邮差与邮局 \(网络协议概观\)](#)

posted @ 2012-11-23 15:41 Vamei 阅读(18970) 评论(9) 编辑 收藏

## 评论列表

#1楼 2012-11-23 16:56 zhuangzhuang1988

突然想到了 刘未鹏大侠的 Blog <http://mindhacks.cn/2012/08/27/modern-cpp-practices/> 后面介绍的c++中的资源管理.

支持(0) 反对(0)

#2楼[楼主] 2012-11-23 17:02 Vamei

@ zhuangzhuang1988

嗯，每种语言的设计都有从其他语言借鉴的地方。

支持(0) 反对(0)

#3楼 2012-11-23 22:16 Chenkun

写的很好， 其实我感觉跟c#中的using还有析构函数性质一样

python的上下文管理器是实现对象的\_\_enter\_\_()和\_\_exit\_\_()

而c#是实现IDisposable这个接口

文中的“使用范围”用词，可以看出楼主的良苦用心呀，完全为读者考虑！

“使用范围”的官话就是“作用域”：)

支持(1) 反对(0)

#### #4楼[楼主] 2012-11-23 22:56 Vamei

@ Chenkun

不好意思，我不懂C#。

我没有用“作用域”的原因是怕和函数的概念混淆。毕竟，上下文管理器本质上只是调用了对象两个方法。这两个方法完全可以什么也不做。这样的效果就和“作用域”不太一样了。

支持(0) 反对(0)

#### #5楼 2012-11-23 23:05 Chenkun

@ Vamei

恩，确实是这样的，看来是我理解的片面了！

支持(0) 反对(0)

#### #6楼 2014-05-10 16:51 laocan

总结部分的finally应该要加上none参数吧，不然会报缺参数的错

但是不知道怎么样可以把错误信息给打印出来（试着将它改错，没成功）

支持(0) 反对(0)

#### #7楼 2014-05-27 22:40 冥河划桨手

```
1 with open("new.txt", "w") as f:
2     print(f.closed)
3     f.write("Hello World!")
4     print(f.closed)
```

这里，当f.write("Hello World!")这句结束后，f已经不再有效了吧  
所以最后一句的f.closed也就不对了？

支持(0) 反对(0)

#### #8楼[楼主] 2014-05-27 23:34 Vamei

@ 冥河划桨手

文件虽然关闭了，但f这个引用还是存在的。

```
>>> print f
```

```
<closed file 'new.txt', mode 'w' at 0x105d856f0>
```

支持(3) 反对(0)

## #9楼 2016-03-22 17:12 GoingMyWay

@ 冥河划桨手

f这个对象还是有的,只不过已经调出缩进区域的时候调用了close,所以f.closed属性为True

[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



SpreadJS 表格控件

Excel界面数据处理  
可视化、可定制

[了解详情](#)



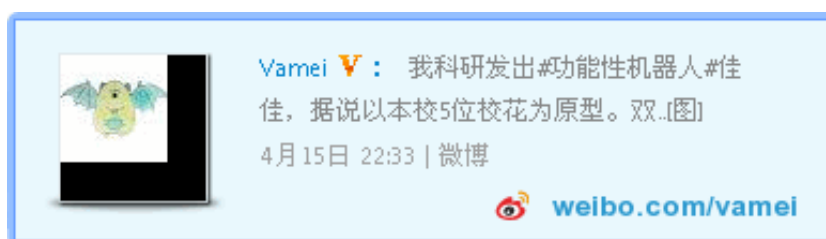
JPush 极光推送 消息推送领导品牌全面升级


[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。

## 我的微博



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...[图]

4月15日 22:33 | 微博

[weibo.com/vamei](https://weibo.com/vamei)

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-edeaa

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

**05370163**