Vamei

编程，数学，设计

Python深入04 闭包

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

闭包(closure)是函数式编程的重要的语法结构。函数式编程是一种编程范式 （而面向过程编程和面向对象编程也都是编程范式）。在面向过程编程中，我们见到过函数(function)；在面向对象编程中，我们见过对象(object)。函数和对象的根本目的是以某种逻辑方式组织代码，并提高代码的可重复使用性(reusability)。闭包也是一种组织代码的结构，它同样提高了代码的可重复使用性。

不同的语言实现闭包的方式不同。Python以函数对象为基础，为闭包这一语法结构提供支持的 （我们在特殊方法与多范式中，已经多次看到Python使用对象来实现一些特殊的语法）。Python一切皆对象，函数这一语法结构也是一个对象。在函数对象中，我们像使用一个普通对象一样使用函数对象，比如更改函数对象的名字，或者将函数对象作为参数进行传递。

函数对象的作用域

和其他对象一样，函数对象也有其存活的范围，也就是函数对象的作用域。函数对象是使用def语句定义的，函数对象的作用域与def所在的层级相同。比如下面代码，我们在line_conf函数的隶属范围内定义的函数line，就只能在line_conf的隶属范围内调用。

```
def line_conf():
    def line(x):
        return 2*x+1
    print(line(5))    # within the scope
```

```
line_conf()
print(line(5))        # out of the scope
```

line函数定义了一条直线(y = 2x + 1)。可以看到，在line_conf()中可以调用line函数，而在作用域之外调用line将会有下面的错误：

```
NameError: name 'line' is not defined
```

说明这时已经在作用域之外。

同样，如果使用lambda定义函数，那么函数对象的作用域与lambda所在的层级相同。

闭包

函数是一个对象，所以可以作为某个函数的返回结果。

```
def line_conf():
    def line(x):
        return 2*x+1
    return line        # return a function object

my_line = line_conf()
print(my_line(5))
```

上面的代码可以成功运行。line_conf的返回结果被赋给line对象。上面的代码将打印11。

如果line()的定义中引用了外部的变量，会发生什么呢？

```
def line_conf():
    b = 15
    def line(x):
        return 2*x+b
    return line        # return a function object

b = 5
my_line = line_conf()
print(my_line(5))
```

我们可以看到，line定义的隶属程序块中引用了高层级的变量b，但b信息存在于line的定义之外 (b的定义并不在line的隶属程序块中)。我们称b为line的环境变量。事实上，line作为line_conf的返回值时，line中已经包括b的取值(尽管b并不隶属于line)。

上面的代码将打印25，也就是说，line所参照的b值是函数对象定义时可供参考的b值，而不是使用时的b值。

一个函数和它的环境变量合在一起，就构成了一个闭包(closure)。在Python中，所谓的闭包是一个包含有环境变量取值的函数对象。环境变量取值被保存在函数对象的__closure__属性中。比如下面的代码：

```
def line_conf():
    b = 15
    def line(x):
        return 2*x+b
    return line        # return a function object

b = 5
my_line = line_conf()
print(my_line.__closure__)
print(my_line.__closure__[0].cell_contents)
```

__closure__里包含了一个元组(tuple)。这个元组中的每个元素是cell类型的对象。我们看到第一个cell包含的就是整数15，也就是我们创建闭包时的环境变量b

的取值。

下面看一个闭包的实际例子：

```
def line_conf(a, b):
    def line(x):
        return ax + b
    return line

line1 = line_conf(1, 1)
line2 = line_conf(4, 5)
print(line1(5), line2(5))
```

这个例子中，函数line与环境变量a,b构成闭包。在创建闭包的时候，我们通过line_conf的参数a,b说明了这两个环境变量的取值，这样，我们就确定了函数的最终形式(y = x + 1和y = 4x + 5)。我们只需要变换参数a,b，就可以获得不同的直线表达函数。由此，我们可以看到，闭包也具有提高代码可复用性的作用。

如果没有闭包，我们需要每次创建直线函数的时候同时说明a,b,x。这样，我们就需要更多的参数传递，也减少了代码的可移植性。利用闭包，我们实际上创建了泛函。line函数定义一种广泛意义的函数。这个函数的一些方面已经确定(必须是直线)，但另一些方面(比如a和b参数待定)。随后，我们根据line_conf传递来的参数，通过闭包的形式，将最终函数确定下来。

闭包与并行运算

闭包有效的减少了函数所需定义的参数数目。这对于并行运算来说有重要的意义。在并行运算的环境下，我们可以让每台电脑负责一个函数，然后将一台电脑的输出和下一台电脑的输入串联起来。最终，我们像流水线一样工作，从串联的电脑集群一端输入数据，从另一端输出数据。这样的情境最适合只有一个参数输入的函数。闭包就可以实现这一目的。

并行运算正称为一个热点。这也是函数式编程又热起来的一个重要原因。函数式编程早在1950年代就已经存在，但应用并不广泛。然而，我们上面描述的流水线式的工作

并行集群过程，正适合函数式编程。由于函数式编程这一天然优势，越来越多的语言也开始加入对函数式编程范式的支持。

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

16 0

荣誉：推荐博客

+加关注

(请您对文章做出评价)

« 上一篇：Python深入03 对象的属性

» 下一篇：协议森林09 爱的传声筒 (TCP连接)

posted @ 2012-12-15 00:27 Vamei 阅读(23153) 评论(24) 编辑 收藏

评论列表

#1楼 2012-12-15 10:11 Chenkun

写的很好, 不过我觉得还有一个地方应该讲解一下在python2.x中 不能对外层变量赋值:

```
def line_conf(a, b):
    i = a * b
    def line(x):
        i = i + x
        return i * x + b
    return line

line1 = line_conf(4, 5)
print line1(5)
```

则会报UnboundLocalError错误

支持(3) 反对(0)

#2楼[楼主] 2012-12-15 10:17 Vamei

@ Chenkun

这个我没有注意过。Python 3没有问题么？

支持(0) 反对(0)

#3楼 2012-12-15 10:18 Chenkun

@ Vamei

python3中好像引入了新的关键字nonlocal, 可以解决这个问题!

支持(0) 反对(0)

#4楼 2012-12-15 10:24 Chenkun

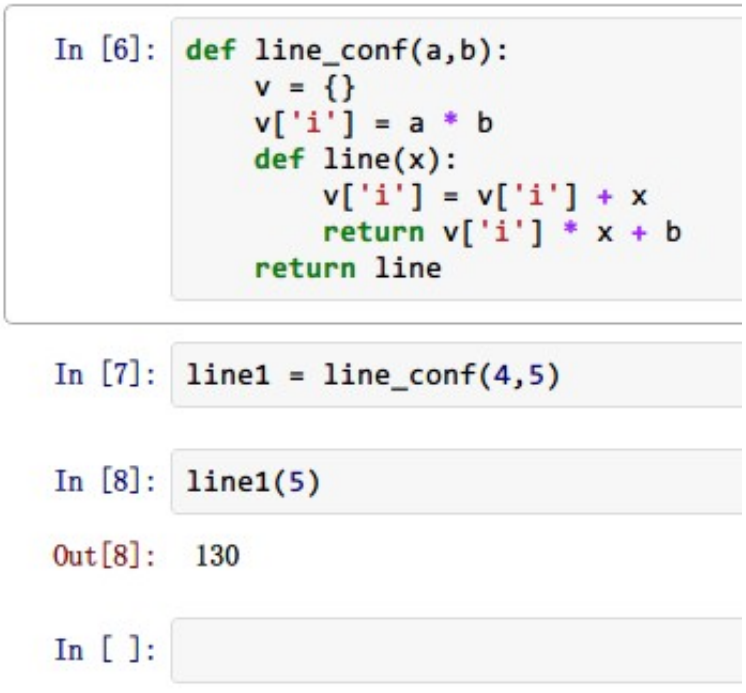
http://www.python.org/dev/peps/pep-3104/

这里有说明和这个问题的一种解决办法!

支持(0) 反对(0)

#5楼 2012-12-15 11:03 zhuangzhuang1988

python 2.*版本是有问题的.可以这样hack下.



支持(1) 反对(0)

#6楼[楼主] 2012-12-15 11:11 Vamei

@ zhuangzhuang1988

这个方法有些取巧啊

支持(0) 反对(0)

#7楼[楼主] 2012-12-15 11:12 Vamei

@ Chenkun

嗯，回来看一下，谢谢！

支持(0) 反对(0)

#8楼 2012-12-15 22:52 leopardsaga

讲解通俗易懂

支持(0) 反对(0)

#9楼 2012-12-30 22:30 真实的活

mark, 好文

支持(0) 反对(0)

#10楼 2013-01-22 16:42 鸽斑兔

马克

支持(0) 反对(0)

#11楼 2013-01-22 16:44 鸽斑兔

楼主顺便帮忙解释一下，这几个是啥意思啊？

闭包有什么用：

惰性求值 || 延迟求值 || 在一系列函数调用中保持某个状态

来自：http://www.cnblogs.com/mess4u/archive/2012/10/23/2735468.html

支持(0) 反对(0)

#12楼[楼主] 2013-01-22 22:48 Vamei

@ tuzkee

面向过程的话，是非延迟求值，比如：

```
a = 5
b = a + 9
c = b - 1
```

print b

就是要先求a, 再求b，再求c，再求d。想得到最后的结果的话，必须一步步运算并保存中间结果。

而延迟求值是说你可以先设计好整个函数，到最后的时候调用一个函数，并传递给参量，一步求出结果。

在一系列参数中保持某个状态我不太明白，应该是说可以保存函数环境吧。

支持(0) 反对(0)

#13楼 2013-01-25 12:45 Sinkup

元旦过来后就开始看您的Python系列了，请问后面还会写吗？

另外讲linux和网络协议的文章也看了，都很有收获，非常感谢！

支持(0) 反对(0)

#14楼[楼主] 2013-01-25 13:03 Vamei

@ sinkup

我不是很确定。有计划写一些Python第三方包的文章，主要是数据分析方面的。

支持(0) 反对(0)

#15楼 2013-02-17 16:07 霍克依毒间

@ Chenkun

试试这样

```
def line_conf(a, b):
i = a * b
def line(x):
i1 = i + x
return i1 * x + b
return line

line1 = line_conf(4, 5)
print line1(5)
```

支持(0) 反对(0)

#16楼 2014-02-21 16:28 展望天空

有点深入了，得细细研究了~

支持(0) 反对(0)

#17楼 2014-04-22 14:29 非洲の白人

@ Chenkun

在内部函数中，环境变量只能用，不能改。（2.7中）

支持(0) 反对(0)

#18楼 2014-10-28 09:23 圣让

写的挺好的~~

支持(0) 反对(0)

#19楼 2014-11-14 10:37 清明-心若淡定

@ Chenkun

在python2.x中报UnboundLocalError错误，会不会是因为变量i是函数line_conf()的局部变量，函数调用结束之后也就清除了这个变量，而line1（）中需要依赖变量i，才会报这个错误

支持(0) 反对(0)

#20楼 2015-03-15 21:28 放作夥

博主好啊，你说的闭包我能理解了。但是从串联的电脑集群一端输入数据，从另一端输出数据。这样的情境最适合只有一个参数输入的函数。闭包就可以实现这一目的。这句话我理解不了，闭包怎么就实现了呢。

支持(0) 反对(0)

#21楼 2015-03-25 15:11 lsj8924

写的非常不错。

支持(0) 反对(0)

#22楼 2015-12-02 12:20 疯狂乱抓

```
1  def line_conf(a, b):
2      def line(x):
3          return ax + b
4      return line
5
6  line1 = line_conf(1, 1)
7  line2 = line_conf(4, 5)
8  print(line1(5), line2(5))
9
10 应该修改成
```

```
11 def line_conf(a, b):
12     def line(x):
13         return a * x + b
14     return line
15 
16 line1 = line_conf(1, 1)
17 line2 = line_conf(4, 5)
18 print(line1(5), line2(5))
```

支持(0) 反对(0)

#23楼 2015-12-22 21:44 遠離塵世の方舟

return a*x + b

支持(0) 反对(0)

#24楼 2016-06-01 12:18 杨桓

@ Chenkun

python2.x中,默认情况下不允许修改外层嵌套函数 line_conf的名称.在python3.x中,可以使用nonlocal语句来声明,这样就可以修改嵌套作用域的变量了.

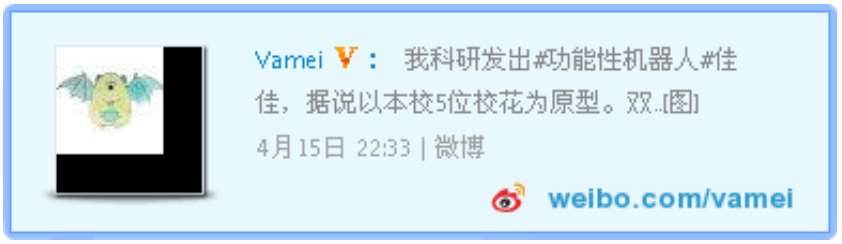
支持(0) 反对(0)

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)
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)