**深入理解Lambda**

原创 2016年03月10日 17:18:43

* 标签：
* [lambda](http://so.csdn.net/so/search/s.do?q=lambda&t=blog) /
* [函数](http://so.csdn.net/so/search/s.do?q=%E5%87%BD%E6%95%B0&t=blog) /
* [Python](http://so.csdn.net/so/search/s.do?q=Python&t=blog)
* 9276

概述

　　Lambda是一个表达式，也可以说它是一个匿名函数。然而在使用它或是阅读Lambda代码的时候，却显得并不那么容易。因为它匿名，因为它删减了一些必要的说明信息（比如方法名）。下面就来说说Lambda是如何进行转换和工作的吧。

版权说明

著作权归作者所有。   
商业转载请联系作者获得授权，非商业转载请注明出处。   
作者：[Coding-Naga](http://blog.csdn.net/lemon_tree12138" \t "_blank)   
发表日期： 2016年3月10日   
链接：[http://blog.csdn.net/lemon\_tree12138/article/details/50774827](http://blog.csdn.net/lemon_tree12138/article/details/50774827" \t "_blank)   
来源：CSDN   
更多内容：[分类 >> Thinking In Python](http://blog.csdn.net/u013761665/article/category/3049975" \t "_blank)

目录

* [概述](http://blog.csdn.net/lemon_tree12138/article/details/50774827#概述)
* [版权说明](http://blog.csdn.net/lemon_tree12138/article/details/50774827#版权说明)
* [目录](http://blog.csdn.net/lemon_tree12138/article/details/50774827#目录)
* [Lambda](http://blog.csdn.net/lemon_tree12138/article/details/50774827#lambda)
  + [初步认识](http://blog.csdn.net/lemon_tree12138/article/details/50774827#初步认识)
    - [表达式及定义](http://blog.csdn.net/lemon_tree12138/article/details/50774827#表达式及定义)
    - [实例说明](http://blog.csdn.net/lemon_tree12138/article/details/50774827#实例说明)
  + [变量作用域说明](http://blog.csdn.net/lemon_tree12138/article/details/50774827#变量作用域说明)
  + [Lambda评价](http://blog.csdn.net/lemon_tree12138/article/details/50774827#lambda评价)
    - [优点](http://blog.csdn.net/lemon_tree12138/article/details/50774827#优点)
    - [缺点](http://blog.csdn.net/lemon_tree12138/article/details/50774827#缺点)
* [Ref](http://blog.csdn.net/lemon_tree12138/article/details/50774827#ref)

Lambda

初步认识

表达式及定义

**lambda [ arg1 [arg2, arg3, … argN] ] : expression**   
　　　　　　　　　　　　　　　　　　　　　　– 《Python核心编程》

从《Python核心编程》书本中包含了以上lambda表达式的表述。也就是说在lambda右侧的式子中，冒号左边的为参数值，右边的为计算表达式。

实例说明

**1.单层参数求和**   
现在假设需要对两个数进行求和运算。对于正常的逻辑代码，不难写出如下代码：

def sum(x, y):

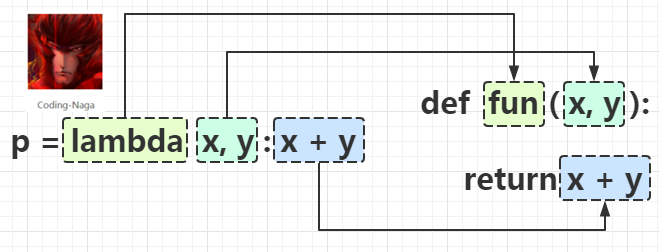
return x + y

* 1
* 2

而在lambda中，我们可以这样来写：

p = lambda x, y: x + y

* 1

代码简洁了许多，可是因为缺少方法名对此步操作进行描述，也让我们对这个求和的Lambda表达式有一些疑问，也就是它在程序中目的是什么，我们难以揣测。   


**2.嵌套参数求和**   
需求还是之前的需求，可是我们在对参数的传递发生了变化。什么叫作嵌套参数，我们可以参考如下代码：   
**注：下面的一段代码实际上是存在语法错误的，此代码的目的纯粹只是为了说明问题。请勿模仿**

def sum\_outer(x=0):

def sum\_inner(y):

return x + y

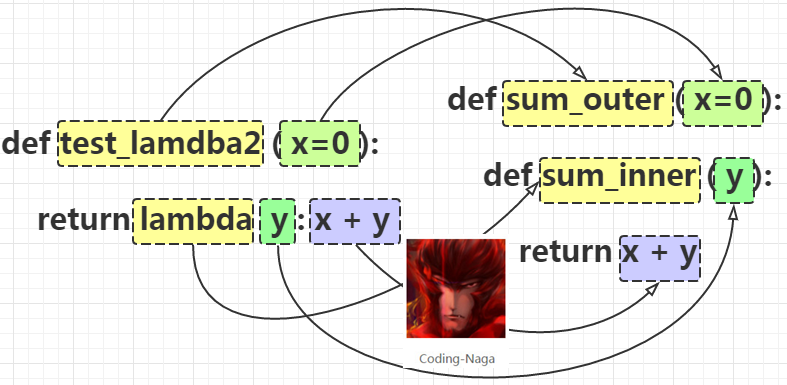
* 1
* 2
* 3

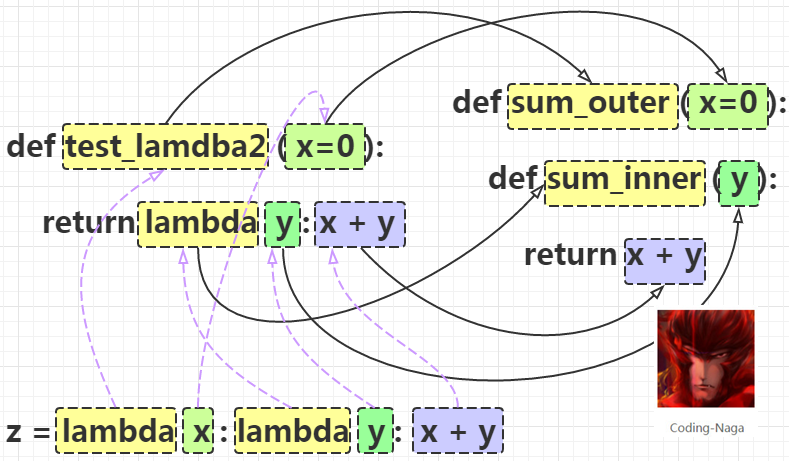
采用lambda改写后的代码如下：

def test\_lamdba2(x=0):

return lambda y: x + y

* 1
* 2

而两种代码逻辑的转换关系如下：   


**3.嵌套lambda**   
前面假想了一种方法内部嵌套方法的情况。我们说Lambda就是理解成在方法内部嵌套了一个“表达式方法”。所以，还可以能上面进行一层Lambda的方法进行改写。将代码中的方法*test\_lambda2*改写成lambda。改写后的描述如下图所示：   
  
将上图所示的转换过程与之前的转换进行一个分析，可以得到一个结论，在lambda的表达式中，前面的lambda为外层方法，后续lambda则为次外层方法，并以次由外向内递推。

变量作用域说明

关于变量在Lambda中的作用域从上面的几幅图中也可以看出一二来，主要可以做出以下几点的总结：

1. 对局部变量可见
2. 对全局变量可见
3. 对当前层传入的参数可见
4. 对上层函数传入的参数可见
5. 对上层Lambda传入的参数可见

Lambda评价

优点

1. 在普通代码里几行的代码，在Lambda中只需要一行就可以解决。所以代码比以前更简洁了
2. 可以在某一个方法内部定义，这样可以提高操作的便捷性

缺点

1. Lambda是一个匿名函数，因为是匿名，所以可读性变差了
2. 有时候有多个Lambda嵌套（就像实例中的第3点一样），让程序变得难以理解

Ref

* 《Python核心编程》
* 《Python学习手册》