

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进阶04 函数的参数对应

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

我们已经接触过函数(function)的**参数(arguments)**传递。当时我们根据**位置**，传递对应的参数。我们将接触更多的参数传递方式。

回忆一下位置传递：

```
def f(a,b,c):  
    return a+b+c  
  
print(f(1,2,3))
```

在调用f时，1，2，3根据位置分别传递给了a,b,c。

关键字传递

有些情况下，用位置传递会感觉比较死板。**关键字(keyword)**传递是根据每个参数的**名字传递参数**。关键字并不用遵守位置的对应关系。依然沿用上面f的定义，更改调用方式：

```
print(f(c=3,b=2,a=1))
```

关键字传递可以和位置传递混用。但位置参数要出现在关键字参数之前：

```
print(f(1,c=3,b=2))
```

参数默认值

在定义函数的时候，使用形如a=19的方式，可以给参数赋予默认值(default)。如果该参数最终没有被传递值，将使用该默认值。

```
def f(a,b,c=10):  
    return a+b+c  
  
print(f(3,2))  
print(f(3,2,1))
```

在第一次调用函数f时，我们并没有足够的值，c没有被赋值，c将使用默认值10。第二次调用函数的时候，c被赋值为1，不再使用默认值。

包裹传递

在定义函数时，我们有时候并不知道调用的时候会传递多少个参数。这时候，包裹(packing)位置参数，或者包裹关键字参数，来进行参数传递，会非常有用。

下面是包裹位置传递的例子：

```
def func(*name):  
    print type(name)  
    print name  
  
func(1,4,6)  
func(5,6,7,1,2,3)
```

两次调用，尽管参数个数不同，都基于同一个func定义。在func的参数表中，所有的参数被name收集，根据位置合并成一个元组(tuple)，这就是包裹位置传递。

为了提醒Python参数，name是包裹位置传递所用的元组名，在定义func时，在name前加*号。

下面是包裹关键字传递的例子：

```
def func(**dict):
```

```
print type(dict)
print dict

func(a=1,b=9)
func(m=2,n=1,c=11)
```

与上面一个例子类似，`dict`是一个字典，收集所有的关键字，传递给函数`func`。为了提醒Python，参数`dict`是包裹关键字传递所用的字典，在`dict`前加`**`。

包裹传递的关键在于定义函数时，在相应元组或字典前加`*`或`**`。

解包裹

`*`和`**`，也可以在调用的时候使用，即解包裹(`unpacking`)，下面为例：

```
def func(a,b,c):
    print a,b,c

args = (1,3,4)
func(*args)
```

在这个例子中，所谓的解包裹，就是在传递`tuple`时，让`tuple`的每一个元素对应一个位置参数。在调用`func`时使用`*`，是为了提醒Python：我想要把`args`拆成分散的三个元素，分别传递给`a,b,c`。（设想一下在调用`func`时，`args`前面没有`*`会是什么后果？）

相应的，也存在对词典的解包裹，使用相同的`func`定义，然后：

```
dict = {'a':1,'b':2,'c':3}
func(**dict)
```

在传递词典`dict`时，让词典的每个键值对作为一个关键字传递给`func`。

混合

在定义或者调用参数时，参数的几种传递方式可以混合。但在过程中要小心前后顺序。基本原则是，先位置，再关键字，再包裹位置，再包裹关键字，并且根据上面所

说的原理细细分辨。

注意：请注意定义时和调用时的区分。包裹和解包裹并不是相反操作，是两个相对独立的过程。

总结

关键字，默认值，

包裹位置，包裹关键字

解包裹

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

200

(请您对文章做出评价)

« 上一篇: Python进阶03 模块

» 下一篇: Python进阶05 循环设计

posted @ 2012-07-08 11:03 Vamei 阅读(34934) 评论(32) 编辑 收藏

评论列表

- #1楼 2013-03-23 23:03 chowlerste

清晰

支持(0) 反对(0)
- #2楼 2013-05-16 01:16 红烧狮子头

清晰易懂，楼主霸气

支持(0) 反对(0)

#3楼 2013-08-22 21:47 非线性进化完全体

3. 包裹(packing)位置传递和包裹关键字传递:

在定义函数时, 我们有时候并不知道调用的时候会传递多少个函数。这时候, 使用包裹位置传递和包裹关键字传递会非常有用。

应该是传递多少个参数吧。。

支持(0) 反对(0)

#4楼[楼主] 2013-08-24 10:47 Vamei

@ 非线性进化完全体

是的, 应该是参数, 谢谢你的提醒。

支持(0) 反对(0)

#5楼 2013-12-28 11:25 itfanr

解包裹 包裹 明白了 第一次见这个概念

支持(0) 反对(0)

#6楼 2014-04-14 08:15 laocan

在包裹关键字传递时, 如果想要得到{1: "b"}的话
不知道怎么弄~

用: 会出现无效字符

用""会报关键字是表达式的错

求教~

支持(0) 反对(0)

#7楼[楼主] 2014-04-14 10:09 Vamei

@ laocan

1恐怕不能作为参数名啊。

支持(1) 反对(0)

#8楼 2014-04-15 23:47 laocan

@ Vamei

谢谢回答~

这里, 是不是相当于把'1'看成新的函数参数, 但是调用时估计会被看成具体值, 并和后面的赋值语句(貌似这时候就报错了)结合?

如果不报错的话，是不是会传递一个bool变量给函数？

支持(0) 反对(0)

#9楼[楼主] 2014-04-16 14:49 Vamei

@ laocan

是把1看作了参数的名字。应该是{"b":1}吧？

支持(0) 反对(0)

#10楼 2014-04-25 19:48 shenhuayu

```
dict = {'a':1,'b':2,'c':3}
```

```
func(**dict) # there is a error!
```

```
change to func(*dict) , and the program will be OK!
```

支持(0) 反对(0)

#11楼 2014-06-19 14:21 转译阳光

```
def func(a,b,c):
```

```
    print a,b,c
```

```
dict = {'a':1,'b':2,'c':3}
```

```
func(**dict) # 得到value
```

```
func(*dict) # 得到key
```

支持(4) 反对(0)

#12楼 2014-07-01 16:59 阿鲁巴

感谢分享

支持(0) 反对(0)

#13楼 2014-08-01 17:20 Riordon

tuple是list

```
>>> def func(a,b,c):
```

```
    print a,b,c
```

```
>>> listTest = [1,2,3]
```

```
>>> func(*listTest)
```

```
1 2 3
```

谢谢楼主的分享...

支持(0) 反对(0)

#14楼 2014-08-01 17:24 Riordon

@ 转译阳光

```
>>> def func(a,b,c):
print a,b,c
>>> dic = {"a":1, "b":2, "c":3}
>>> func(**dic)
1 2 3
>>> func(*dic)
a c b
>>> you are right !
```

支持(0) 反对(0)

#15楼 2014-08-18 11:35 11_xiao_7

楼主请教一个问题:

```
def func(*name):
print type(name)
print name
args = {'a':1,'b':2,'c':3}
func(*args)
('a','b','c') //为什么能够成功并且返回的是('a','b','c')
```

```
def func1(**name):
print type(name)
print name
dict = {'a':1,'b':2,'c':3}
func1(**dict)
<type 'dict'>
{'a': 1, 'c': 3, 'b': 2}
为什么不能使用func(*dict)和func(dict)
新手还请不要见笑~~~
```

支持(0) 反对(0)

#16楼 2014-10-10 09:58 NUST小文

@ Riordon

为什么打印出来的是a, c, b而不是a, b, c? 求教

支持(0) 反对(0)

#17楼 2014-10-10 10:14 Riordon

@ NUST小文

在python2.7是那样的，不知道是不是bug,但在python3中正常了：

```
>>> def func(a,b,c):
print(a,b,c)
>>> dic = {"a":1, "b":2, "c":3}
>>> func(*dic)
a b c
>>> func(**dic)
1 2 3
```

支持(0) 反对(0)

#18楼 2015-03-01 13:53 年轻的水兵湾

楼主讲的很好，以前学c++时候没怎么懂的东西现在看看都差不多的

支持(0) 反对(0)

#19楼 2015-04-01 17:00 pythonerJ

求解答，楼主，请问你说的：

关键字传递可以和位置传递混用。但位置参数要出现在关键字参数之前

```
print(f(1,c=3,b=2))
```

原例中，c=3, c应该是关键词=3，c=3是位置参数？

还是你指的是 必须要用一个 纯位置参数，才可以混用？

1 在c=3之前？ 是这么理解吗？

有点乱~~

支持(0) 反对(0)

#20楼 2015-05-12 15:46 _hsin

@ pythonerJ

问题描述是很乱~~

纯位置参数是什么？如果可以被称为混用，那么肯定是既有位置参数（non-keyword arg），也有关键字参数（keyword arg），然后位置参数必须在关键字参数的前面，这里就是对 a 使用位置参数传递，然后后面两个使用关键字参数传递，所以 b 、 c 之间的位置随意，只要它们俩都在 1 的后面就行。


```
1 func(a=1,b=2,3)
2     File "<stdin>", line 1
3     SyntaxError: non-keyword arg after keyword arg
```

支持(0) 反对(0)

#21楼 2015-05-12 15:58 pythonerJ

@ _hsin

感谢你的回复！

支持(0) 反对(0)

#22楼 2015-05-31 13:05 jiahit

@ 转译阳光

引用

```
def func(a,b,c):
    print a,b,c

dict = {'a':1,'b':2,'c':3}
func(**dict) # 得到value
func(*dict) # 得到key
```

但，得到的key的顺序和value的顺序却不一致，是怎么回事？随机的么？

支持(0) 反对(0)

#23楼 2015-08-10 15:05 Nefeltari

@ 转译阳光

```
1 def func2(a,b,c):
2     print a,b,c
3     args = (1,2,3)
4     func2(*args) # 包裹关键字传递
5     dict = {'a':1,'b':2,'c':3}
6     func2(**dict) # 包裹关键字传递
7     func2(*dict)
```

这段代码运行结果是：

1 2 3

1 2 3

a c b

func2(*dict)输出的结果有顺序吗？为什么不是acb不是abc呢？

支持(0) 反对(0)

#24楼 2015-08-10 15:13 Nefeltari

@ Vamei

```
1 def func(*name):
2     print type(name)
3     print name
4 func(1,4,6) # 包裹传递
5 func(5,6,7,1,2,3)
6 func([1,2,3])
```

这段代码输出结果：

<type 'tuple'>

(1, 4, 6)

<type 'tuple'>

(5, 6, 7, 1, 2, 3)

<type 'tuple'>

([1, 2, 3],)

包裹传递只能输出tuple吗？不然为什么func([1,2,3])输出结果是<type 'tuple'>

([1, 2, 3],) ？

支持(0) 反对(0)

#25楼 2015-08-10 15:41 Nefeltari

```
1 def func2(a,b,c):
2     print a,b,c
3 dict = {'a':1,'b':2,'c':3}
4 func2(**dict) # 包裹关键字传递
5 func2(*dict)
```

这段代码输出结果是：

```
1 2 3
a c b
```

怎样输出词典里面的一个元素，比如 'a':1 ？

支持(0) 反对(0)

#26楼 2015-10-22 14:52 蚂蚁快跑

补充一点,函数参数的默认值只能放在参数的最后,
如 `def(x,y=1)`有效, `def(y=1,x)`无效

支持(0) 反对(0)

#27楼 2015-10-30 14:16 Hear_Yang

学习了 之前陆陆续续的学但都没怎么进步 现在跟着楼主的教程学 感觉收获很大 可能也和我做开发时间有关系 总觉得之前难懂的 现在懂了不少

支持(0) 反对(0)

#28楼 2015-11-30 13:00 irunner

```
def func(*name):
    #print type(name)
    print name
    func(1,2,3,4)
    func(5,6,7)

def func1(**dict):
    print type(dict)
    print dict
    func1(a=1,b=2,c=3)
    func1(a=2,b=3,c=4)

def func2(a,b,c):
    print a,b,c
    dict = {'a':2,'b':4,'c':5}
    func2(**dict)
    func2(**{'a':2,'b':4,'c':5})
```

测试通过

支持(0) 反对(0)

#29楼 2016-01-08 10:23 xiaosanyu

先位置，再关键字，再包裹位置，再包裹关键字

博主看看是否有误

先位置(tuple)，再包裹位置(*tuple)，再关键字(dict)，再包裹关键字(**dict)

```
1 def func(a=0,b=0,c=0,d=0,e=0,f=0,g=0):
2     print (a,b,c,d,e,f,g)
3     args = (4,5)#b,c赋值
4     args1={'e':6,'d':7}#d,e赋值
5     func(1,*args,g=2,f=3,**args1)
6
7 #输出
8 1 4 5 7 6 3 2
```

支持(0) 反对(0)

#30楼 2016-03-31 13:15 Missingsour

@ Nefeltari

```
def func(*name):
print type(name)
print list(name)
```

可以这样，不知道有没有更好的方法。

支持(0) 反对(0)

#31楼 2016-04-07 13:15 wqh2016

学习 完毕

支持(0) 反对(0)

#32楼 2016-04-09 10:42 Suckseedeva

很棒的教程，谢谢楼主！！

请教两个

Q1: 模块包下的文件需要一个一个 导入？ folder.module1?folder.module2?

Q2: 模块包的模块导入后，子模块自己直接运行输出？

[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



ActiveReports

企业级报表服务平台

单独部署、集成应用、报表制作、数据整合
权限管理、移动办公、二次集成开发


[立即了解](#)


 **极光推送** **消息推送领导品牌全面升级**  **JIGUANG | 极光**
[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

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)

Copyright ©2016 Vamei

05370296