[Python函数及参数详解（位置、关键字、限定位置参数等）](https://www.cnblogs.com/Hardworking666/p/15866139.html)

**文章目录**

* [一、函数简介](https://www.cnblogs.com/Hardworking666/p/15866139.html#_10)
  + [1.定义函数](https://www.cnblogs.com/Hardworking666/p/15866139.html#1_22)
  + [2.调用函数](https://www.cnblogs.com/Hardworking666/p/15866139.html#2_45)
  + [3.函数返回值](https://www.cnblogs.com/Hardworking666/p/15866139.html#3_53)
* [二、函数的参数](https://www.cnblogs.com/Hardworking666/p/15866139.html#_59)
  + [1.位置参数](https://www.cnblogs.com/Hardworking666/p/15866139.html#1_69)
  + [2.关键字参数](https://www.cnblogs.com/Hardworking666/p/15866139.html#2_73)
  + [3.默认参数](https://www.cnblogs.com/Hardworking666/p/15866139.html#3_101)
  + [4.不定长参数](https://www.cnblogs.com/Hardworking666/p/15866139.html#4_118)
  + [5.限定位置形参（也叫强制位置参数）](https://www.cnblogs.com/Hardworking666/p/15866139.html#5_175)
  + [6.限定关键字形参（常叫命名关键字参数）](https://www.cnblogs.com/Hardworking666/p/15866139.html#6_189)
  + [7.参数小结](https://www.cnblogs.com/Hardworking666/p/15866139.html#7_217)
  + [8.经验建议](https://www.cnblogs.com/Hardworking666/p/15866139.html#8_229)
* [结语](https://www.cnblogs.com/Hardworking666/p/15866139.html#_245)

**一、函数简介**

**函数**是组织好的，可重复使用的，用来实现单一，或相关联功能的代码段。

函数能提**高应用的模块性，和代码的重复利用率**。

Python提供了许多内建函数，比如print()。我们也可以创建函数，即用户**自定义函数**。

函数的使用必须遵循“先定义，后调用”的原则。

函数的定义就相当于事先将内存地址赋值给定义的函数名，函数名就是对这段代码的引用，这类似于变量的定义。

**1.定义函数**

def 函数名(参数1,参数2,...):

"""文档描述"""

函数体

return 值

1. def: 定义函数的**关键字**；
2. 函数名：函数名指向函数内存地址，是对函数体代码的引用及功能的**概述**；
3. 括号：括号内定义可有可无且无需指定类型的**参数**；
4. 冒号：括号后加冒号，以便在下一行开始**缩进**编写函数体的代码；
5. 文档描述：描述函数**功能**，参数介绍等信息的文档，非必要；
6. 函数体：由**语句和表达式**组成；
7. return 值：结束函数，选择性地返回一个**值**给调用方，不带值的return相当于返回None。

**2.调用函数**

函数的使用分为**定义阶段**与**调用阶段**，定义函数时只检测语法，不执行函数体代码。

**函数名加括号**即函数调用，只有调用函数时才会执行函数体代码。

函数的**调用形式**分为：语句形式、表达式形式、作为参数形式。

**3.函数返回值**

若需要将函数体代码执行的结果返回给调用者，则需要用到return。

return是一个函数结束的标志，函数内可以有多个return，但只执行一次函数就结束了，并把return后定义的值作为本次调用的结果返回。

**二、函数的参数**

函数的参数分为形式参数和实际参数，简称形参和实参。

**形参**即在定义函数时，括号内声明的参数。形参本质就是一个**变量名**，用来接收外部传来的值。

**实参**即在调用函数时，括号内传入的值，值可以是**常量、变量、表达式**或三者的组合。

在Python中，变量名与值只是单纯的绑定关系，而对于函数来说，这种绑定关系只在函数调用时生效，在调用结束后解除。

**1.位置参数**

位置参数也称**必须参数**，位置参数须以**从左到右的顺序**传入函数。调用时的**数量**必须和声明时的一样。

**2.关键字参数**

在调用函数时，实参可以是key=value的形式，称为关键字参数。

凡是按照这种形式定义的实参，可以不按照从左到右的顺序定义，但仍能为指定的形参赋值。

因为 Python 解释器能够用参数名匹配参数值。

**注意**：在调用函数时，实参也可以是按位置或按关键字的混合使用，但必须保证

1. **关键字参数在位置参数后面**，因为python函数在解析参数时，是按照顺序来的，**位置参数**是必须先满足，才考虑其他可变参数。
2. **不可以对一个形参重复赋值**。

def register(name,age):

register(name='Tom',18) #SyntaxError：关键字参数name=‘Tom’在位置参数18之前

register()

结果报错：SyntaxError: positional argument follows keyword argument

positional argument：位置参数，即通过在参数列表中的相对位置确定传递给哪个形参，

keyword argument：关键字参数，通过name=value这样的形式，根据name确定传递给哪个形参。

argument(actual parameter)指的是函数调用时的实际参数。

**3.默认参数**

在定义函数时，就已经为形参赋值，这类**形参**称之为默认参数。

当函数有多个参数时，需要将值经常改变的参数定义成位置参数，

而将值改变较少的参数定义成默认参数，从而降低函数调用的复杂度。

注意：

1. **默认参数必须在位置参数之后**。
2. 默认参数的值仅在函数定义阶段被赋值一次。
3. 默认参数的值通常应设为不可变类型。
4. 每次调用是在上一次的基础上向同一**列表**增加值。

**4.不定长参数**

当需要一个函数能**处理比当初声明时更多的参数**，则这些参数叫做不定长参数。

**不定长的位置参数：**

如果在最后一个形参名前加\*号,那么在调用函数时，溢出的位置实参，都会被以**元组**的形式保存下来赋值给该形参。

def text(x=8, y=9, z=10, \*args): # 在最后一个形参名args前加\*号

print(x)

print(y)

print(z)

print(args)

text(1, 2, 3, 4, 5, 6, 7)

# 实参1、2、3按位置为形参x、y、z赋值

# 多余的位置实参4、5、6、7都被\*接收，以元组的形式保存下来，赋值给args，即args=(4,5,6,7)

运行结果：  
1  
2  
3  
(4, 5, 6, 7)  
注意：这里x、y、z以用户传的参数为准，不用默认的8、9、10。

例如：求多个值的和

def add(\*args):

res = 0

for i in args:

res += i

return res

print(add(1, 2, 3, 4, 5))

结果：15

**不定长的关键字参数**

在最后一个形参名前加\*\*号，调用时溢出的关键字参数被以*字典*的形式赋值给该形参。

def func(x,\*\*kwargs): #在最后一个参数kwargs前加\*\*

print(x)

print(kwargs)

fun(y=2,x=1,z=3) #溢出的关键字实参y=2，z=3都被\*\*接收，以字典的形式保存下来，赋值给kwargs

结果：  
1  
{‘y’: 2, ‘z’: 3}

使用\*\*kwargs可以为日后的扩展供良好的环境，同时保持了函数的简洁性。

以下是Python 3.8引入的：

**5.限定位置形参（也叫强制位置参数）**

从Python 3.8开始，positional-only形参将可正式用于自定义函数中。

它们必须放在形参表的最前面，并在后面使用斜杠/（独占一个参数位）与普通形参分隔，比如下面这样

# a, b, c成为限定位置形参

def func(a, b, c, /, d):

pass

这时fun的形参a, b, c将只能按位置接收实际参数，d仍是普通形参，可以兼容两种形式。

**6.限定关键字形参（常叫命名关键字参数）**

限定关键字形参，当然就是为了限制后面几个参数只能按关键字传递，

这往往是因为后面几个形参名具有十分明显的含义，显式写出**有利于可读性**；

或者后面几个形参随着版本更迭很可能发生变化，强制关键字形式**有利于保证跨版本兼容性**。

**限定关键字形参**（keyword-only）

限制调用者不能按位置传递，需要放在形参表的后面，并在前面使用星号\*（独占一个参数位）与普通形参分隔，即类似这样

def func(其他形参, \*, kw1, kw2):

pass

这时参数kw1, kw2在函数调用时必须**显式写出**，即类似fun(其他实参, kw1=var1, kw2=var2)的形式。

下面举一个三种混合形参的例子

def func(a, b, c, /, m, n, \*, kw1, kw2):

pass

其中的m, n是兼容两种形式的普通形参。

**7.参数小结**

如果形参中已经有一个args了，**命名关键字参数**就不再需要一个单独的\*作为分隔符号了。

可变参数*args与关键字参数****kwargs通常是****组合在一起使用\*\*的，如果一个函数的形参为*args与\*\*kwargs，那么代表该函数可以接收任何形式、任意长度的参数。

提示： \*args、\*\*kwargs中的args和kwargs被替换成其他名字并无语法错误，但使用args、kwargs是约定俗成的。

综上所述，所有参数可任意组合使用，但定义顺序必须是：

位置参数、默认参数、\*args、命名关键字参数、\*\*kwargs

**8.经验建议**

建议使用**不可变对象**，如整数、浮点数、字符串、True、False、None或以上类型组成的元组等**设定默认值**。

因为默认值只会在函数定义时被设定一次，如果是可变对象，一旦在函数内部被原地修改，效果会保留至以后每次的函数调用，不会被重新初始化。

如果非要使用某个可变对象作为默认值，比如**列表**，或者要设定依赖于其他参数的默认值，建议设成None，然后写成类似这样的代码

def func(x, default=None):

if default is None:

# 这里可以书写更复杂的初始化行为

default = []

# 剩余的函数体