**argparse:**

import argparse

parser = argparse.ArgumentParser(description=’demo’)

parser.add\_argument(’--net’ ,dest=’net’, type=str,

default=true, help=’how to use’)

par = parser.parse\_args()

函数传递值：

**不可更改的类型**：类似 c++ 的值传递，如 整数、字符串、元组。如fun（a），传递的只是 a 的值，没有影响 a 对象本身。比如在 fun（a）内部修改 a 的值，只是修改另一个复制的对象，不会影响 a 本身。

**可更改的类型**：类似 c++ 的引用传递，如 列表，字典。如 fun（a），则是将 a 真正的传过去，修改后 fun 外部的 a 也会受影响

例子：

#不可改变类型

def change\_number(b):

b = 1000

b = 1

change\_number(b)

print(b)

输出：1

#可改变类型

def change\_number(b):

b.append(1000)

b = [1,2]

change\_number(b)

print(b)

输出：[1,2,1000]

**只有在形参表末尾的那些参数可以有默认参数值，**这里需要特别注意的一点：默认参数的值是不可变的对象，比如None、True、False、数字或字符串。一个 lsit （列表）或者 dict （字典），可以使用 None 作为默认值。

可以通过参数名来给函数传递参数，而不用关心参数列表定义时的顺序，这被称之为**关键字参数**。使用关键参数有两个优势 ：

1、由于我们不必担心参数的顺序，使用函数变得更加简单了。

2、假设其他参数都有默认值，我们可以只给我们想要的那些参数赋值

有时我们在设计函数接口的时候，可会需要可变长的参数。也就是说，我们事先无法确定传入的参数个数。Python 提供了一种元组的方式来接受没有直接定义的参数。这种方式在参数前边加星号 \* 。如果在函数调用时没有指定参数，它就是一个空元组。我们也可以不向函数传递未命名的变量。

例如：

def print\_user\_info( name , age , sex = '男' , \* hobby):

# 打印用户信息

print('昵称：{}'.format(name) , end = ' ')

print('年龄：{}'.format(age) , end = ' ')

print('性别：{}'.format(sex) ,end = ' ' )

print('爱好：{}'.format(hobby))

return;

# 调用 print\_user\_info 函数

print\_user\_info( '两点水' ,18 , '女', '打篮球','打羽毛球','跑步')

通过输出的结果可以知道，\*hobby是可变参数，且 hobby其实就是一个 tuple （元祖）

可变长参数也支持关键参数，没有被定义的关键参数会被放到一个字典里。这种方式即是在参数前边加 \*\*,更改上面的示例如下：

# -\*- coding: UTF-8 -\*-

def print\_user\_info( name , age , sex = '男' , \*\* hobby ):

# 打印用户信息

print('昵称：{}'.format(name) , end = ' ')

print('年龄：{}'.format(age) , end = ' ')

print('性别：{}'.format(sex) ,end = ' ' )

print('爱好：{}'.format(hobby))

return;

# 调用 print\_user\_info 函数

print\_user\_info( name = '两点水' , age = 18 , sex = '女', hobby = ('打篮球','打羽毛球','跑步'))

输出的结果：

昵称：两点水 年龄：18 性别：女 爱好：{'hobby': ('打篮球', '打羽毛球', '跑步')}

通过对比上面的例子和这个例子，可以知道，\*hobby是可变参数，且 hobby其实就是一个 tuple （元祖），\*\*hobby是关键字参数，且 hobby 就是一个 dict （字典）

*匿名函数*