# python语法

# 函数

## 位置参数的可变参数

有多个数，需要累加求和

def add(\*nums):   
 sum = 0   
print(type(nums))   
for x in nums:   
 sum += x   
print(sum)   
add(3, 6, 9) # 调用

在形参前使用\*表示该形参是可变参数，可以接收多个实参

收集多个实参为一个tuple

思考：关键字参数能否也能传递任意多个吗？（不能）

## 关键字参数的可变参数

配置信息打印

def showconfig(\*\*kwargs):   
 for k,v in kwargs.items():   
 print('{} = {}'.format(k, v))   
showconfig(host='127.0.0.1',port='8080',username='wayne',password='magedu')



形参前使用\*\*符号，表示可以接收多个关键字参数

收集的实参名称和值组成一个字典

## 可变参数混合使用

配置信息打印

def showconfig(username, password, \*\*kwargs)

def showconfig(username, \*args, \*\*kwargs)

def showconfig(username, password, \*\*kwargs, \*args)

## 总结

* 有位置可变参数和关键字可变参数

* 位置可变参数在形参前使用一个星号\*

* 关键字可变参数在形参前使用两个星号\*\*

* 位置可变参数和关键字可变参数都可以收集若干个实参，位置可变参数收集形成一个tuple，关键字可变参数收集形成一个dict

* 混合使用参数的时候，可变参数要放到参数列表的最后，普通参数需要放到参数列表前面，位置可变参数需要在关键字可变参数之前

## keyword-only参数

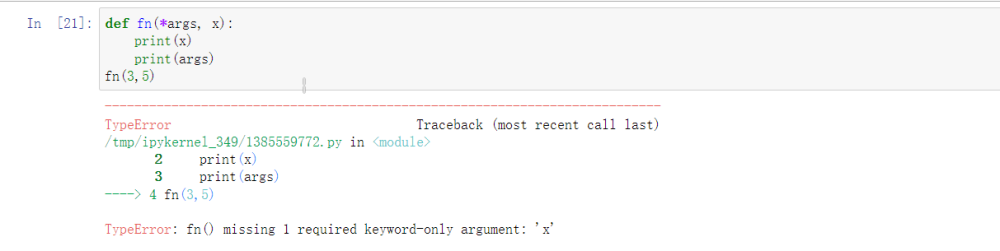
keyword-only参数（Python3加入）

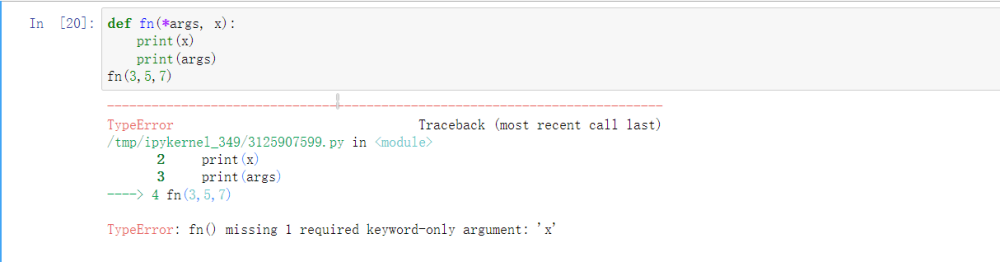
如果在一个星号参数后，或者一个位置可变参数后，出现的普通参数，实际上已经不是普通的

参数了，而是keyword-only参数

def fn(\*args, x):   
 print(x)   
 print(args)

fn(3,5)



fn(3,5,7) 

fn(3,5,x=7)



args可以看做已经截获了所有的位置参数，x不使用关键字参数就不可能拿到实参

思考：def fn(\*\*kwargs, x) 可以吗？

举例

def(\*\*kwargs, x):   
 print(x)   
 print(kwargs)

直接报语法错误

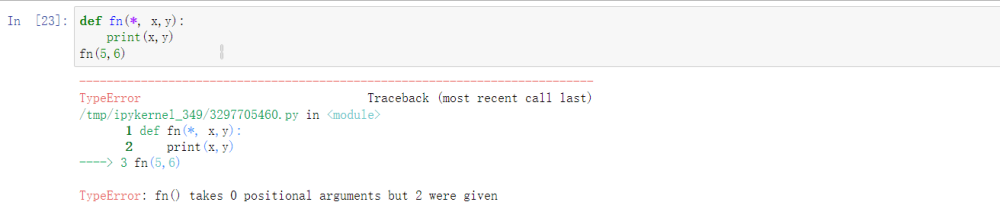
可以理解为kwargs会截获所有的关键字参数，就算你写了x=5，x也永远得不到这个值，所以语法

错误

### keyword-only 参数另一种形式

def fn(\*, x,y):   
 print(x,y)   
fn(x=5,y=6)

fn(5,6)



fn(x=5,y=6)



\*号之后，普通形参都变成了必须给出的keyword-only 参数

## 可变参数和参数默认值

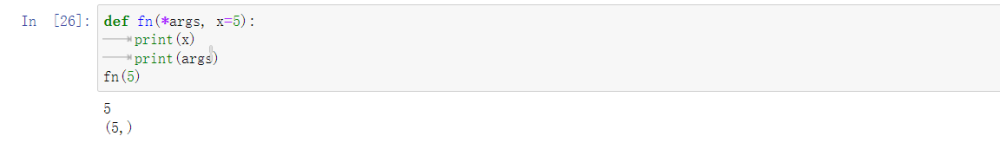
### 举例 1

def fn(\*args, x=5):   
 print(x)   
 print(args)

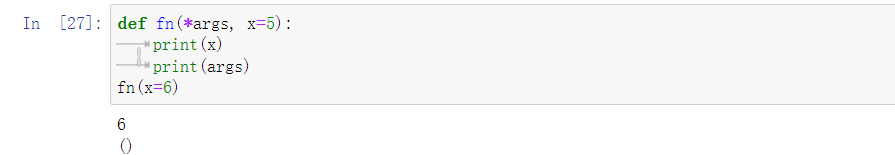
fn() # 等价于fn(x=5)



fn(5)



fn(x=6)



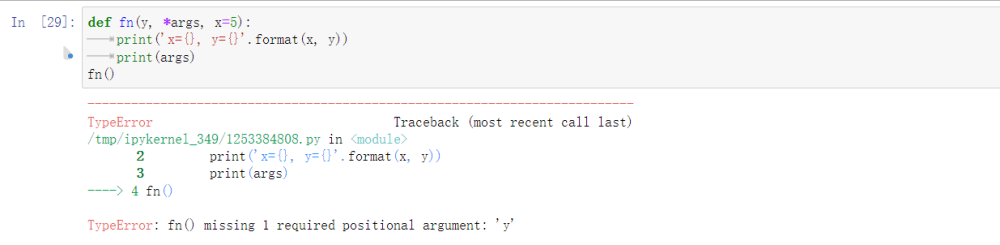
fn(1,2,3,x=10)



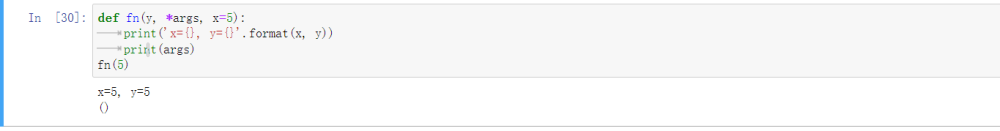
### 举例 2

def fn(y, \*args, x=5):   
 print('x={}, y={}'.format(x, y))   
 print(args)

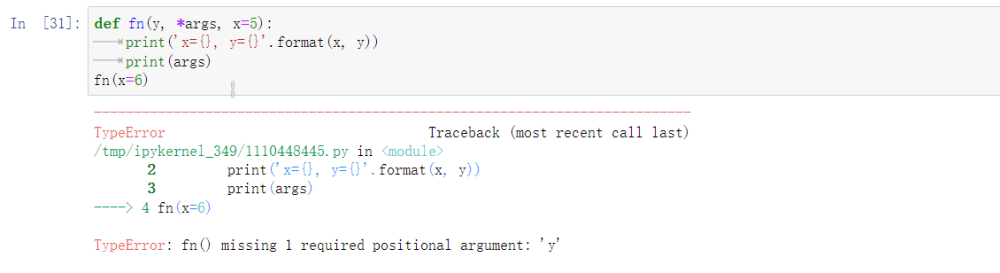
fn()



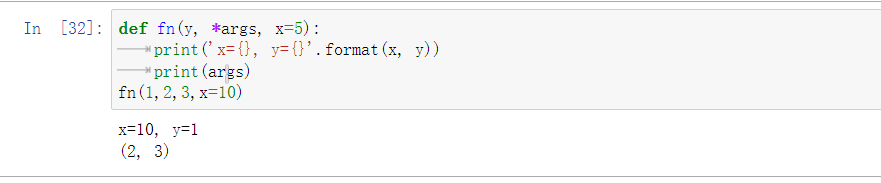
fn(5)



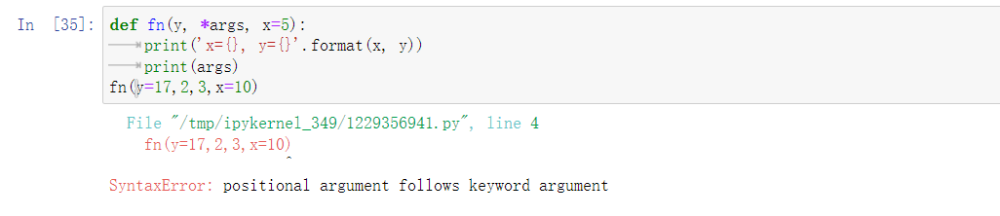
fn(x=6)



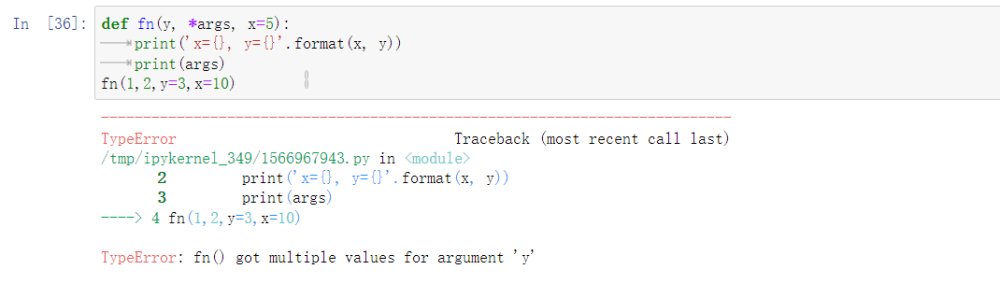
fn(1,2,3,x=10)



fn(y=17,2,3,x=10)



fn(1,2,y=3,x=10)



x 是 keyword-only参数

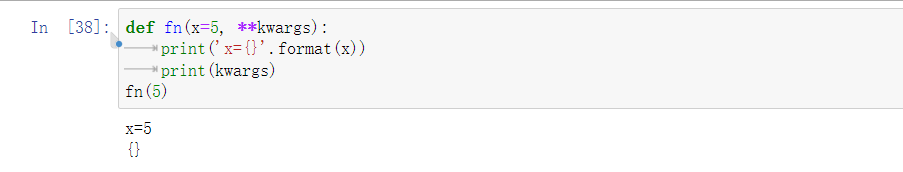
### 举例 3

def fn(x=5, \*\*kwargs):   
 print('x={}'.format(x))   
 print(kwargs)

fn()



fn(5)



fn(x=6)



fn(y=3,x=10)



fn(3,y=10)



## 参数规则

参数列表参数一般顺序是，普通参数、缺省参数、可变位置参数、keyword-only参数（可带缺

省值）、可变关键字参数

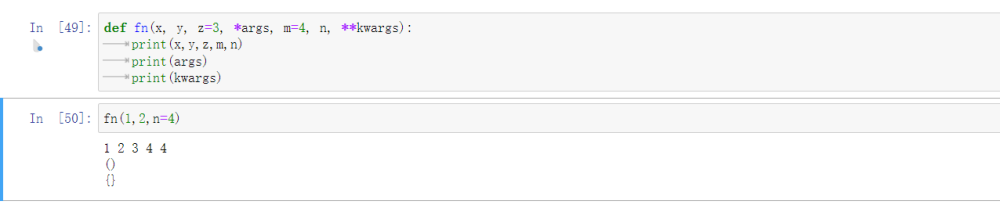
def fn(x, y, z=3, \*args, m=4, n, \*\*kwargs):   
 print(x,y,z,m,n)   
 print(args)   
 print(kwargs)

**注意**

**代码应该易读易懂，而不是为难别人**

**请按照书写习惯定义函数参数**

fn(1,2,n=4)



fn(1,2,10,11,t=7,n=5)







## 参数解构

* 参数解构

* + 给函数提供实参的时候，可以在集合类型前使用\*或者\*\*，把集合类型的结构解开，提取出所有元素作为函数的实参

* + 非字典类型使用\*解构成位置参数

* + 字典类型使用\*\*解构成关键字参数

* + 提取出来的元素数目要和参数的要求匹配，也要和参数的类型匹配

### 实例1

def add(x, y):  
return x+y

add(\*(4,5))



add(\*[4,5])



add(\*{4,6})

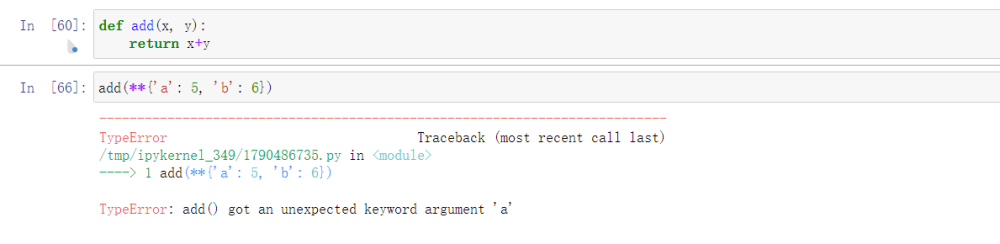


d = {'x': 5, 'y': 6}

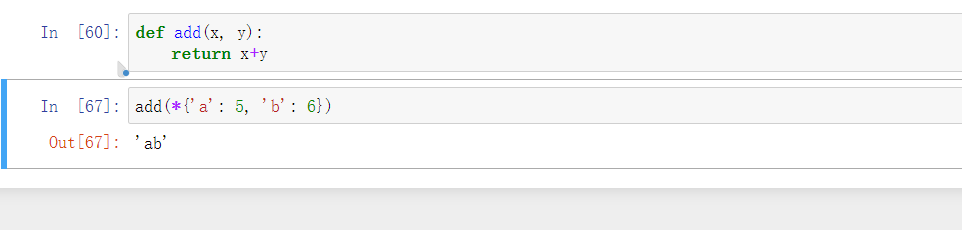
add(\*\*d)



add(\*\*{'a': 5, 'b': 6})



add(\*{'a': 5, 'b': 6})



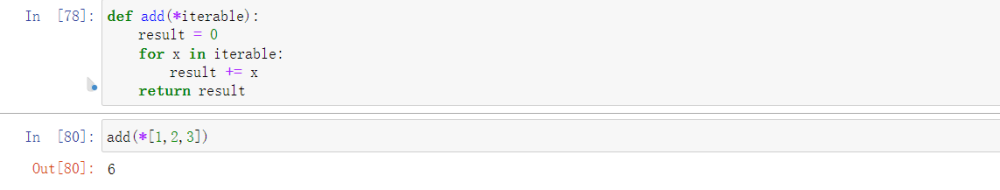
### 实例2

def add(\*iterable):  
 result = 0  
 for x in iterable:  
 result += x  
return result

add(1,2,3)



add(\*[1,2,3])



add(\*range(10))



## global使用原则

外部作用域变量会内部作用域可见，但也不要在这个内部的局部作用域中直接使用，因为

函数的目的就是为了封装，尽量与外界隔离

如果函数需要使用外部全局变量，请使用函数的形参传参解决

一句话：不用global。学习它就是为了深入理解变量作用域