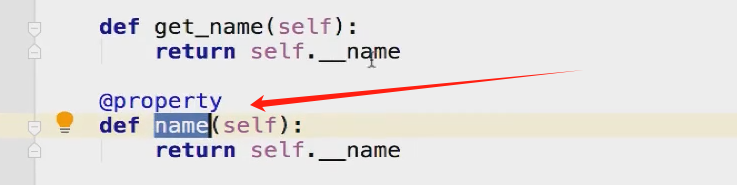
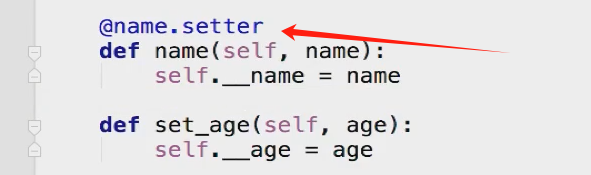
# 类

## 1. 属性

类的内部，单下划线是受保护属性，都可以访问。双下划线\_\_private\_var是私有属性，只有类内部访问，或者使用这种方式访问\_ClassName\_\_private\_var。

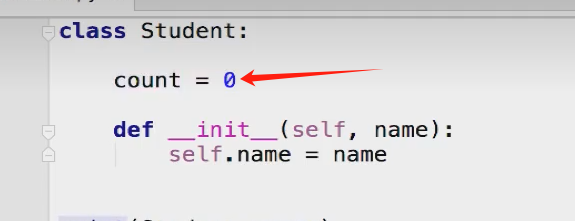


第二个可以替代第一种方法，然后使用Class.name就能获取到name参数，而不用get\_name（）



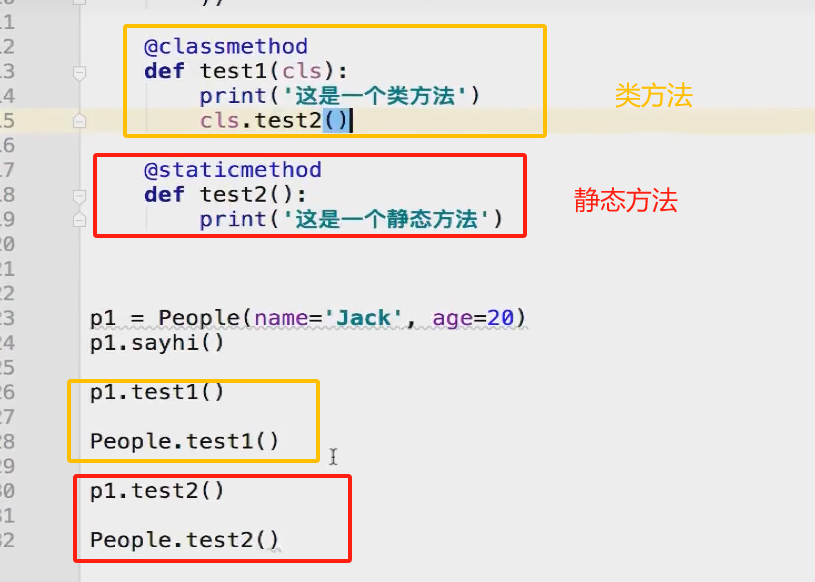
Set方法同理

类属性（相当于static属性，不需要实例化类也可以访问的属性）



这个count对于实例化以后，这个实例化对象自己也有一个count属性，和类的count是不同的，所以给实例化的count赋值，是不影响类的count的值的。

给类属性赋值之后，以后的每个实例化后的count的默认值都是这个类属性



类方法：类的公共方法，可以传cls参数，自己调用内部类方法、静态方法、类属性

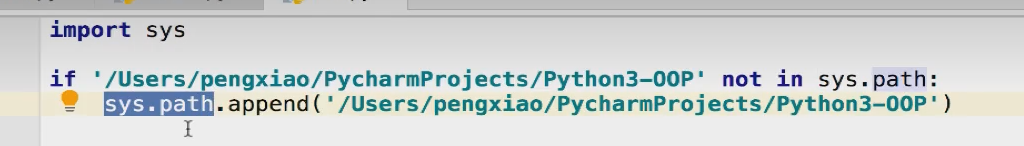
静态方法：只能有基本类型参数，不能调用自身方法

## 2. 继承



## 3. IsInstance(x,int)和Type()

## 4. package



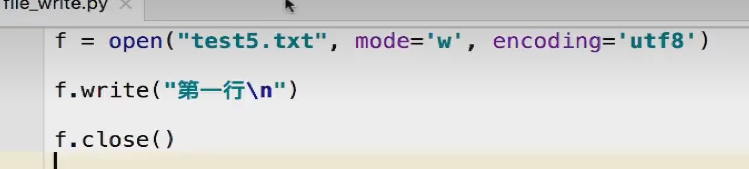
最好把总目录添加进sys里面，不然找不到目录下面的路径，虽然pycharm自己会帮我们加进去，但是如果是在其他地方命令行运行就会报错

# if \_\_name\_\_ == “\_\_main\_\_”

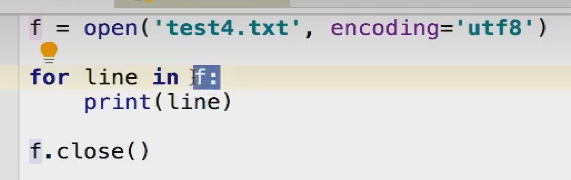
如果不加这句，会把import引入并执行的那个函数的相关代码都执行了，例如一些用于测试代码，所以在测试时最好放在if \_\_name\_\_ == “\_\_main\_\_”下面。因为\_\_name\_\_是不同的

Pypi.org

# 文件读写操作



W a x r模式



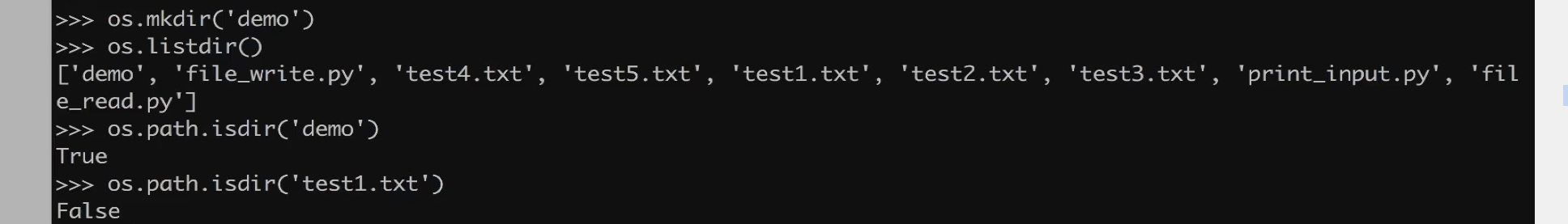
读写二进制：



或者使用这种方式就不用调用close方法了



OS包是操作系统的，包括操作目录文件等，import os



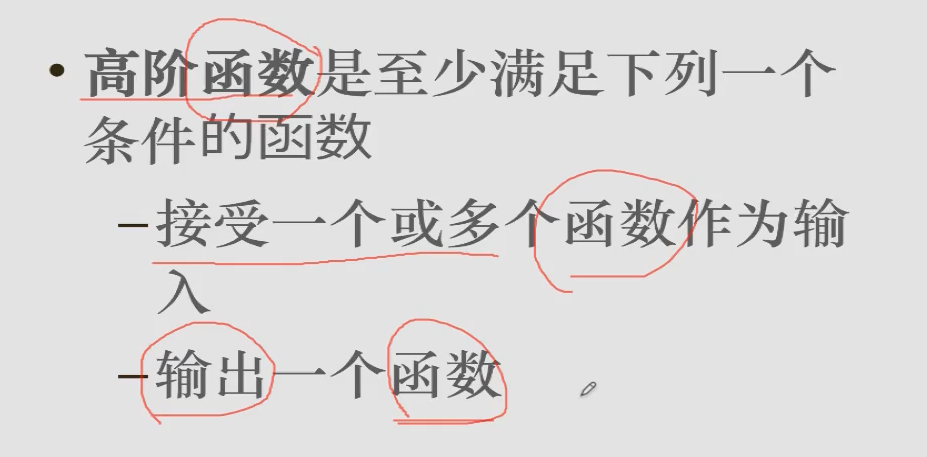
帮助文档：

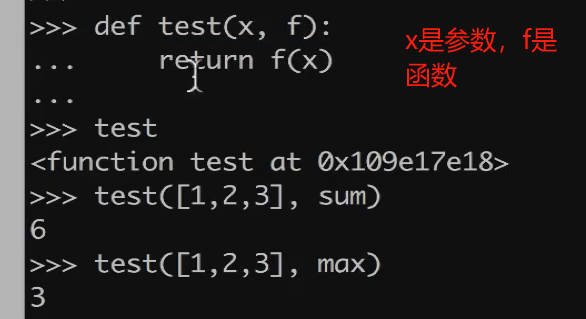
Dir（os）获取所有方法

Help(os.mkdir) 获取该方法参数和使用文档

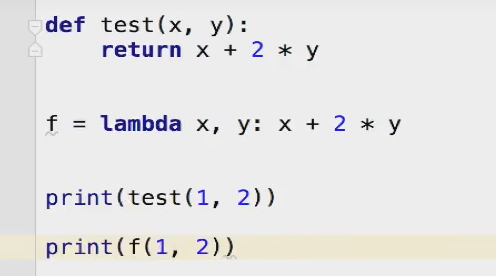
# 函数式编程

## 1. 高阶函数

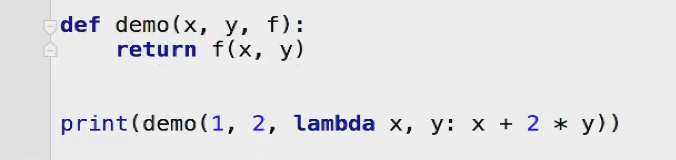




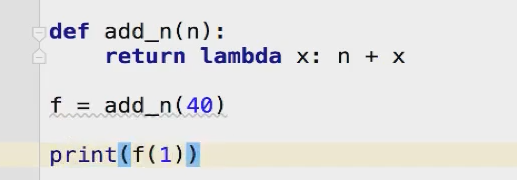
## 2. lambda匿名函数



和高阶函数结合：

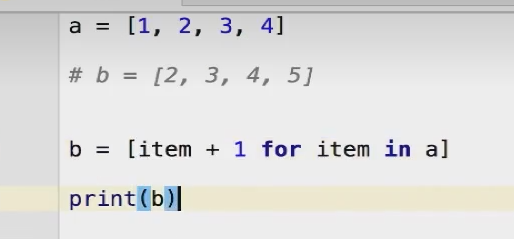


以lambda为返回函数，此时f是一个函数，是lambda返回的匿名函数

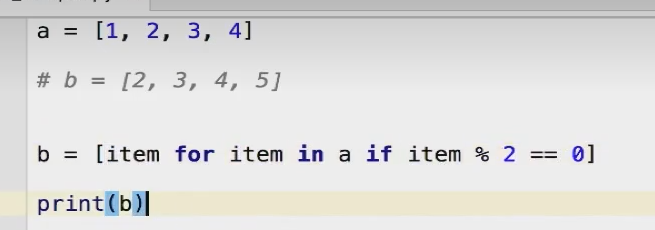


结果是41

## 3. 列表解析



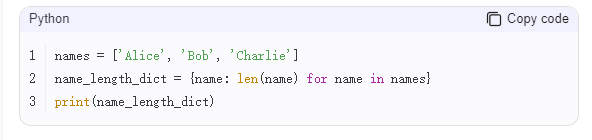
For前的item：指的是要放入b这个数组的元素



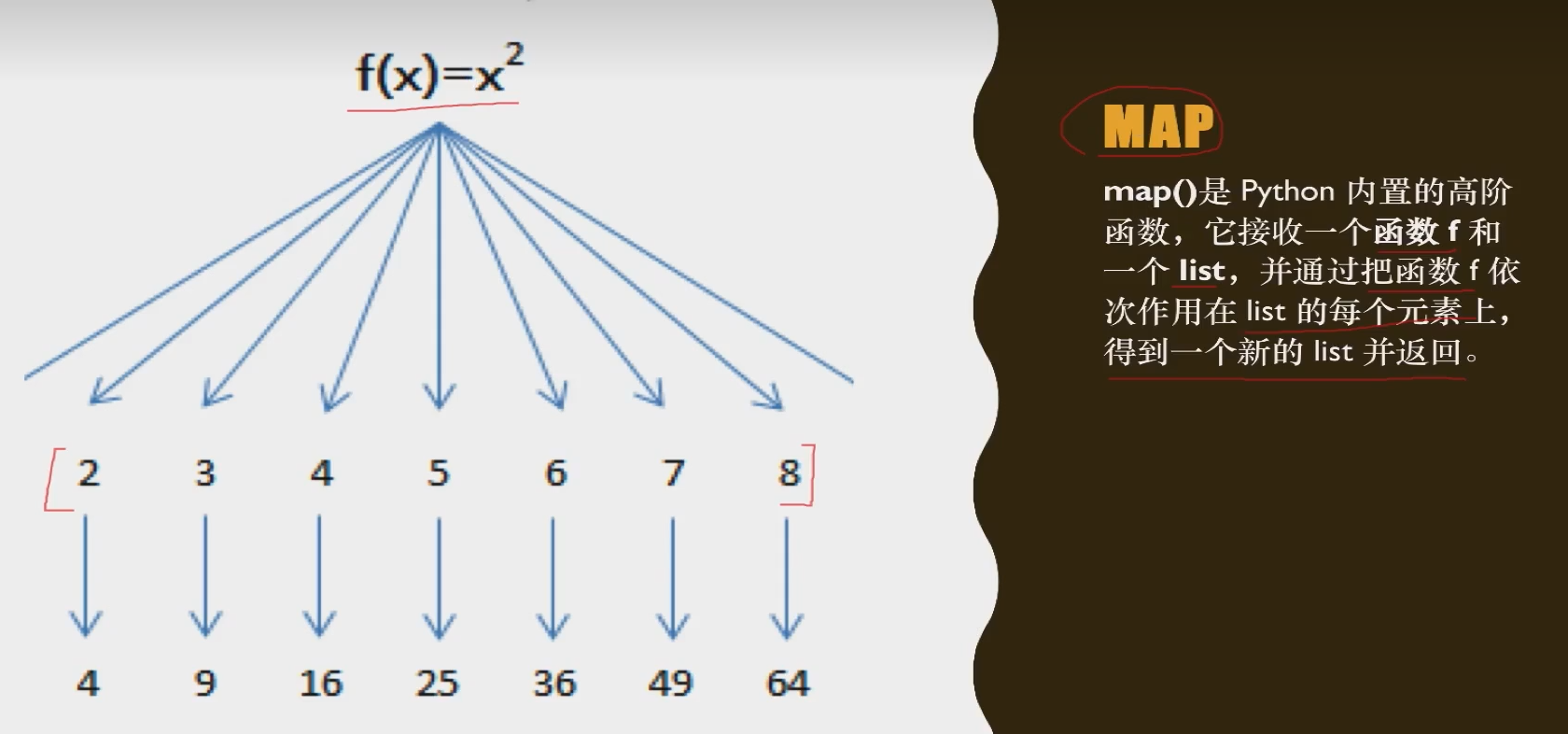
或者加上判断语句，满足if的才把item放进b

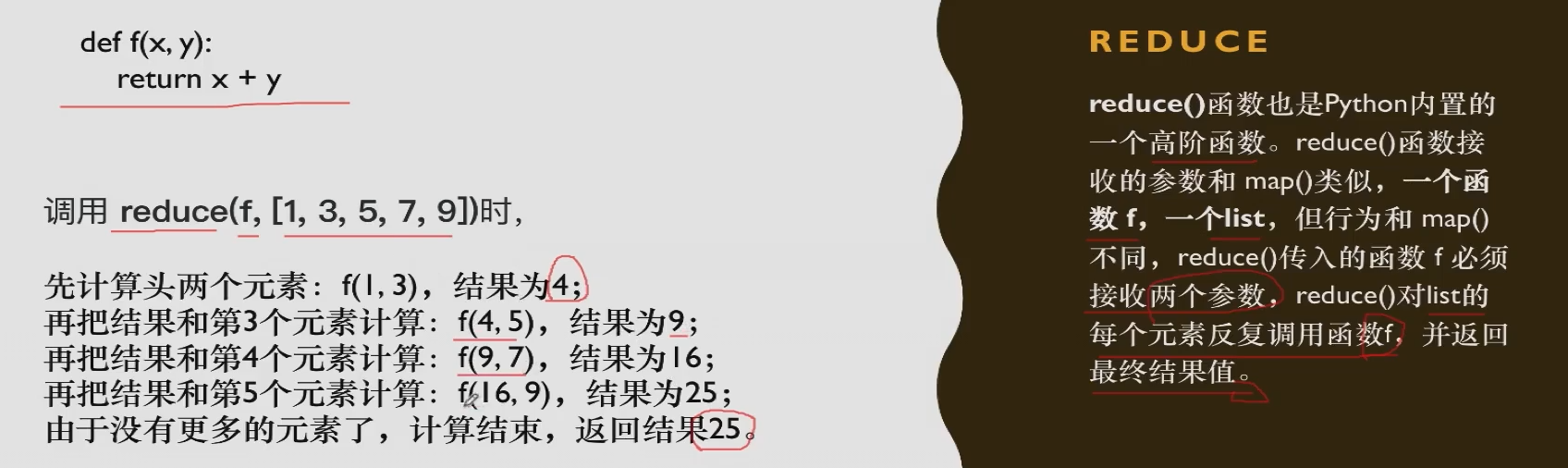
还有字典解析：

{key\_expression: value\_expression for element in iterable if condition}



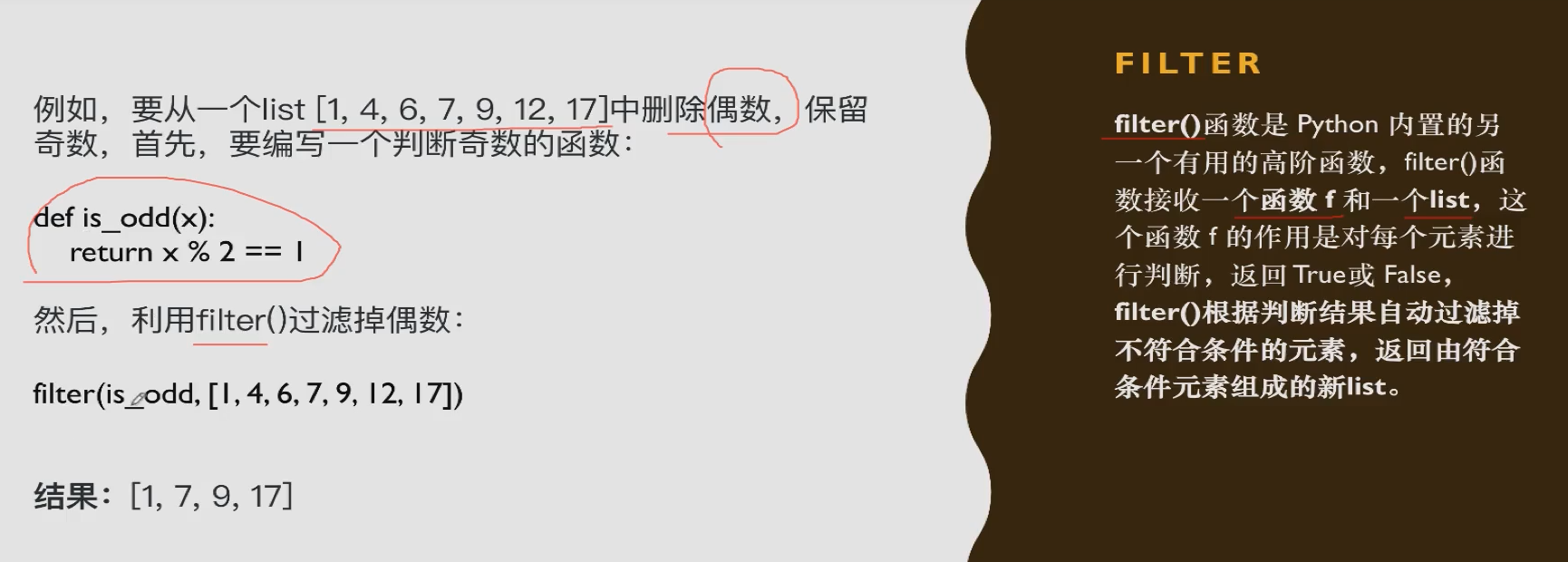
## 4. Map()函数和reduce()



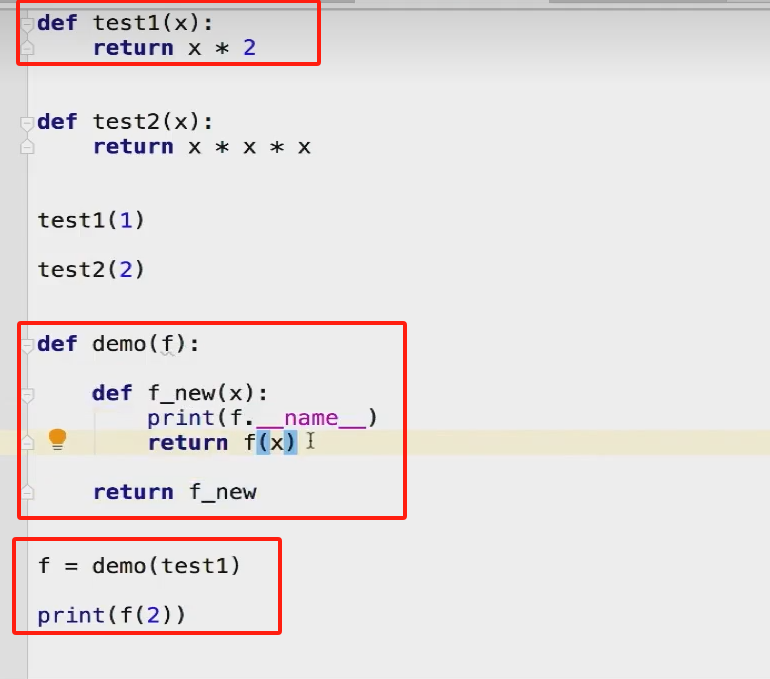


不一定是list。能进行for循环的都可以传进去

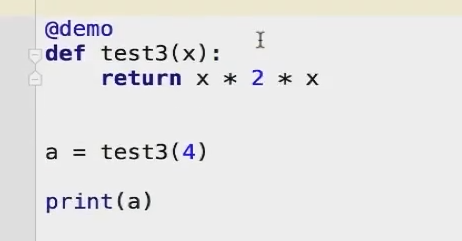
## 5. filter（）



## 6. 装饰器



或者使用装饰器：

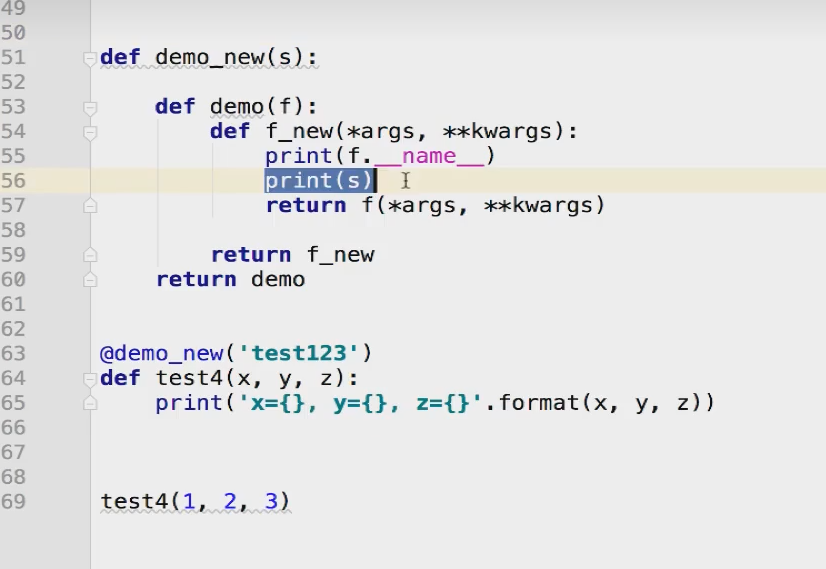


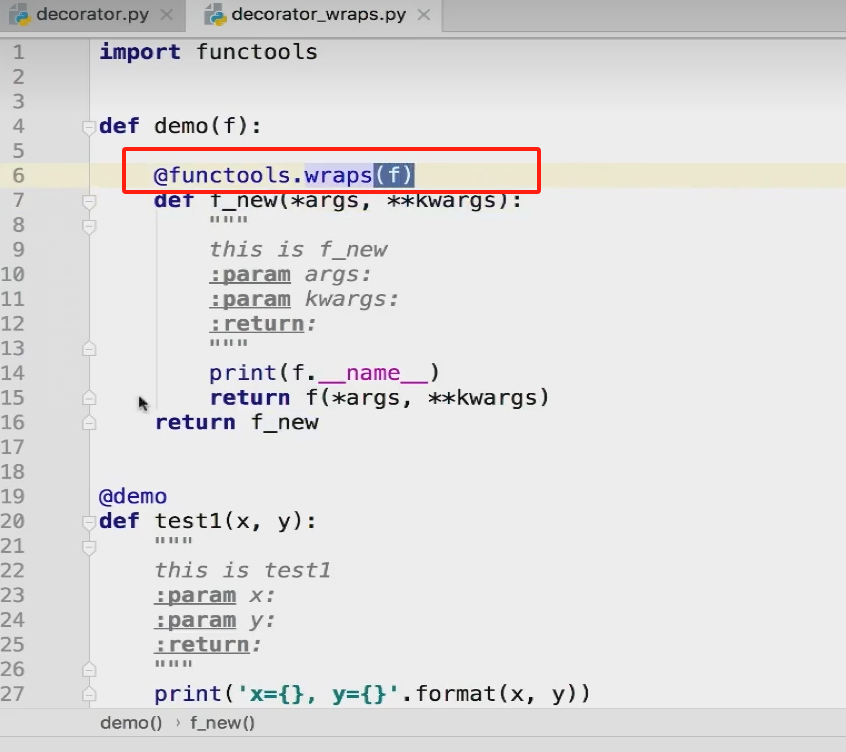


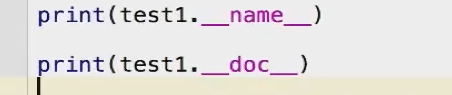
本质上还是把test函数传给demo函数。Demo作为高阶函数

如果需要在装饰器上面传参：

那就再封装一层，



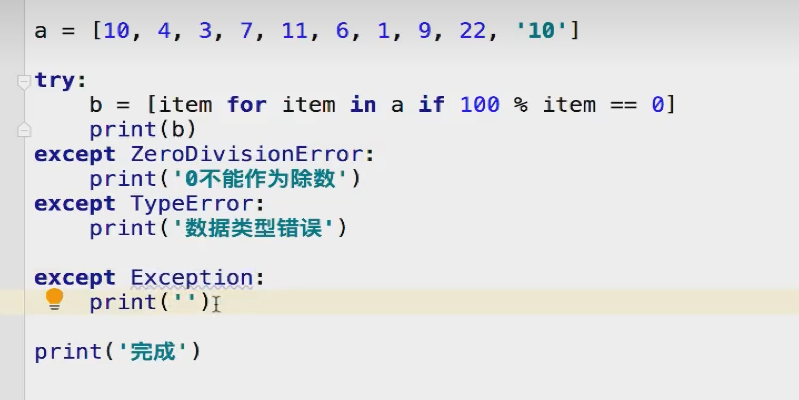




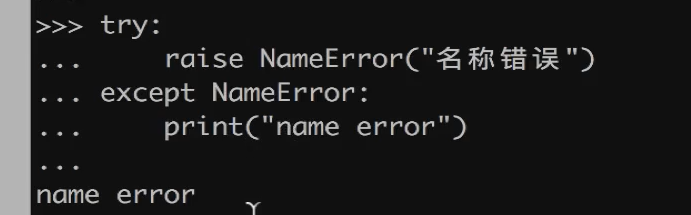
前提：调用test1的时候，\_\_name\_\_和\_\_doc\_\_的返回值都变成了f\_new的值，可以使用上面红框的方法，是的test的\_\_name\_\_和\_\_doc\_\_值是自己的

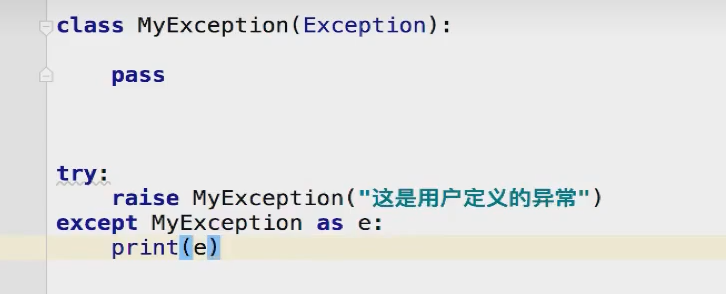
# 异常



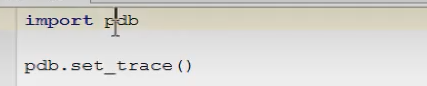


自己抛出异常：





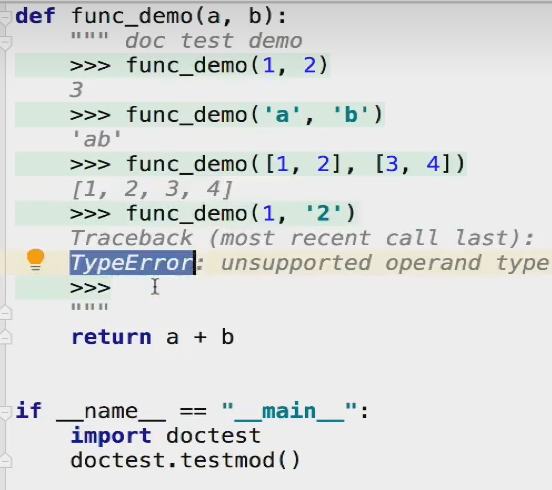
# 通过Pdb进行debug



通过？查看使用方式，N = Next、s = step、p = print

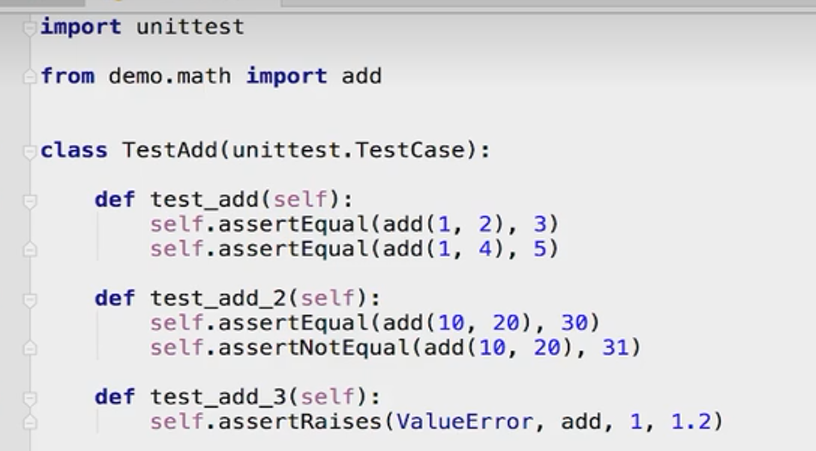
P可以打印参数

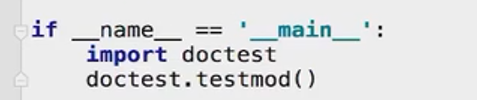
# 测试：



文档测试，会执行doc里面四条测试用例

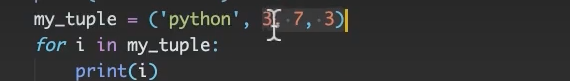
单元测试：

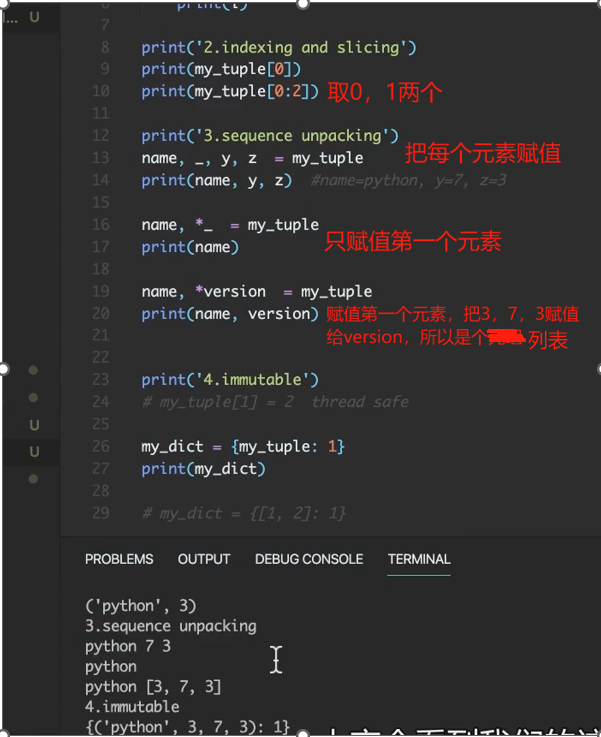




# Collections

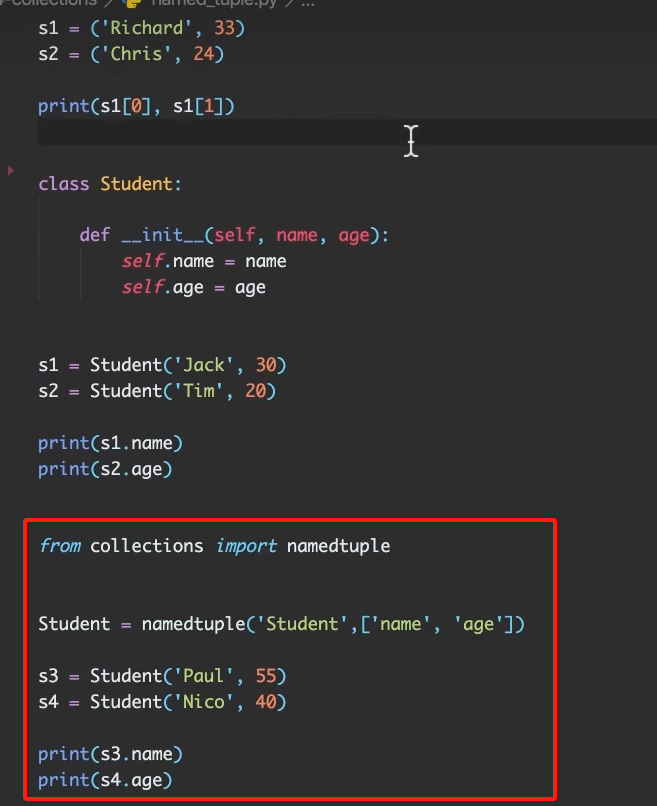
## 元组：





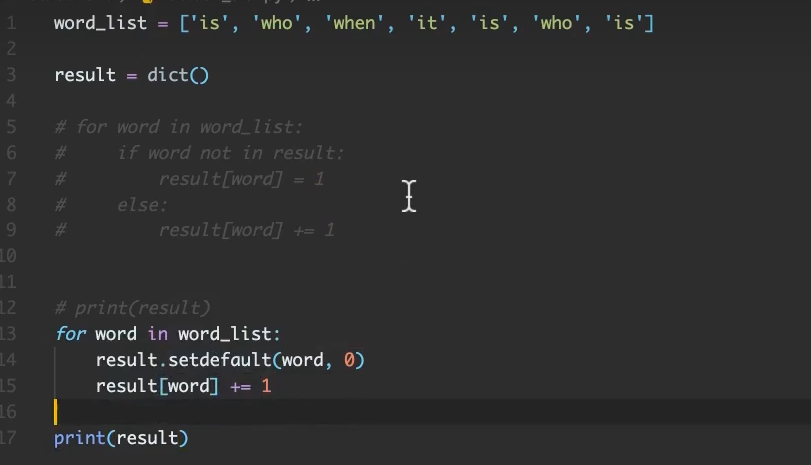
最下面，如果想作为字典的key，必须是不可变的，元组就是不可变的。而list则不能作为字典的key

## Name Tuple:

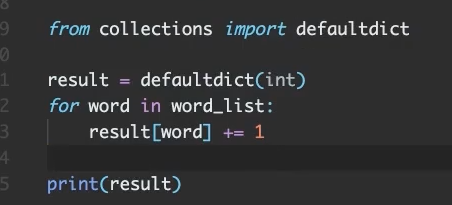


## Default Dict

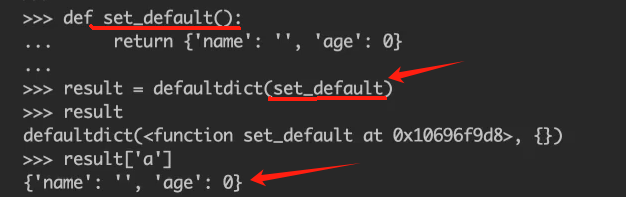
统计单词出现次数：



上面是不用default dict方式



default dict如果没有某个key，会帮我们自动创建这个key和默认值，这里传的是int，所以默认值是0，也可以传函数等（下面代码传函数案例）



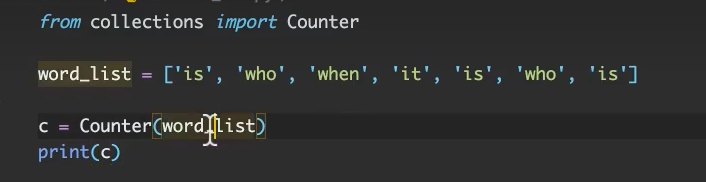
## Deque

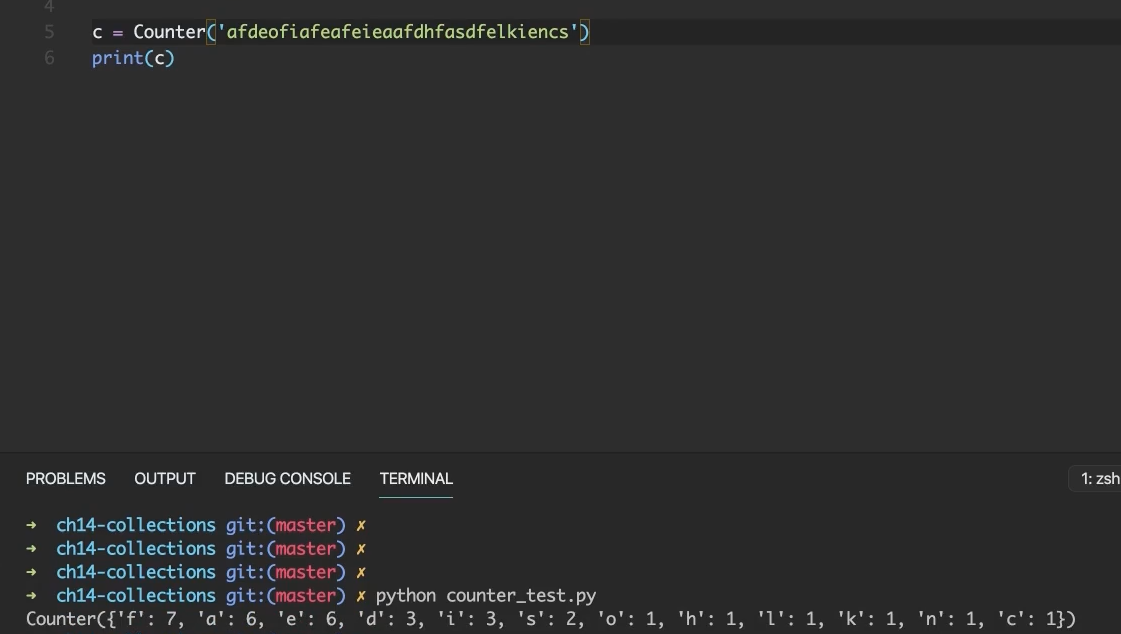
双向队列，数据量大的情况下，deque插入某个下标时效率比list高很多，因为list需要把下标后的值都往后移动一位，这就和java里面的list和双向列表的区别一样。

## Counter

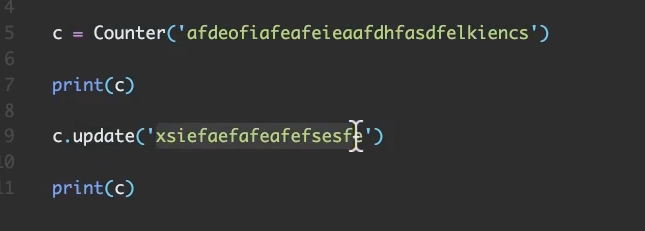
统计工作

对比上面的统计字母出现次数的效率比dict高





使用update（）可以追加参数



还有很多关于求和、top等方法

## Order dict

普通的dict是根据我们赋值的顺序来排序的