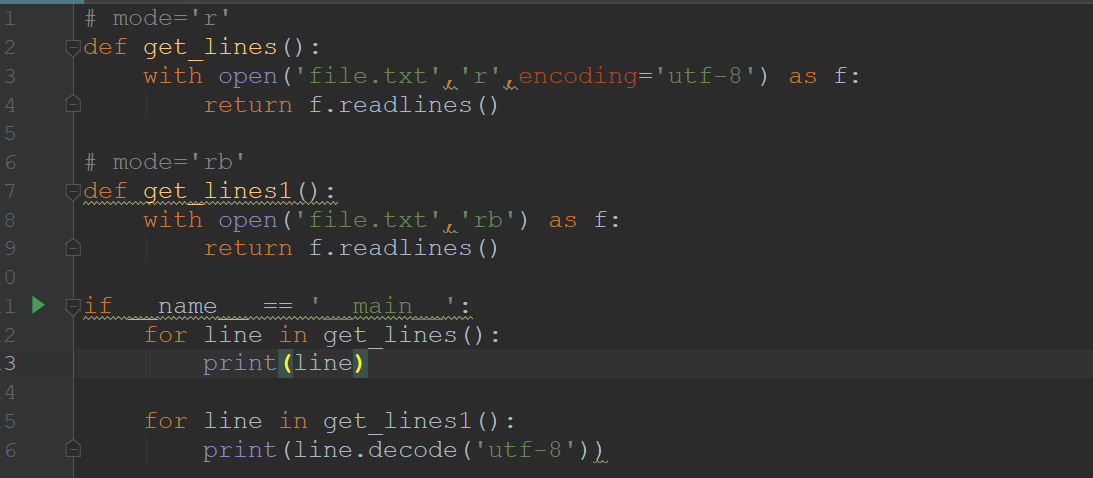
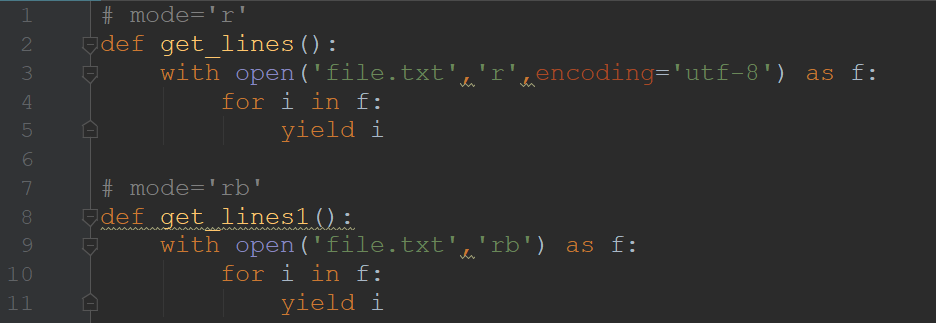
## [Python基础](https://github.com/kenwoodjw/python_interview_question" \l "python%E5%9F%BA%E7%A1%80)

## [文件操作](https://github.com/kenwoodjw/python_interview_question#%E6%96%87%E4%BB%B6%E6%93%8D%E4%BD%9C)

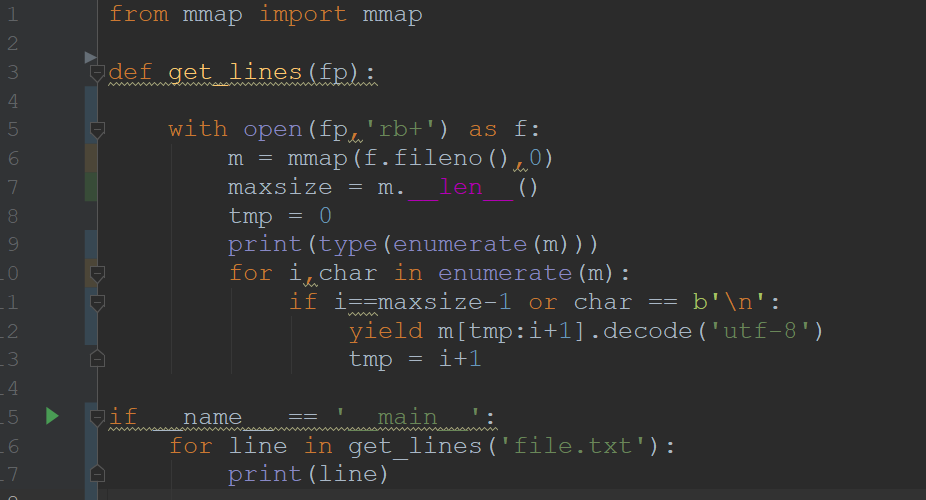
### 1.有一个jsonline格式的文件file.txt大小约为10K

****

**现在要处理一个大小为10G的文件，但是内存只有4G，如果在只修改get\_lines 函数而其他代码保持不变的情况下，应该如何实现？需要考虑的问题都有那些？**

****

**或者**

****

**要考虑的问题有：内存只有4G无法一次性读入10G文件，需要分批读入分批读入数据要记录每次读入数据的位置。分批每次读取数据的大小，太小会在读取操作花费过多时间。**

### 2.补充缺失的代码

def print\_directory\_contents(sPath):

"""

这个函数接收文件夹的名称作为输入参数

返回该文件夹中文件的路径

以及其包含文件夹中文件的路径

"""

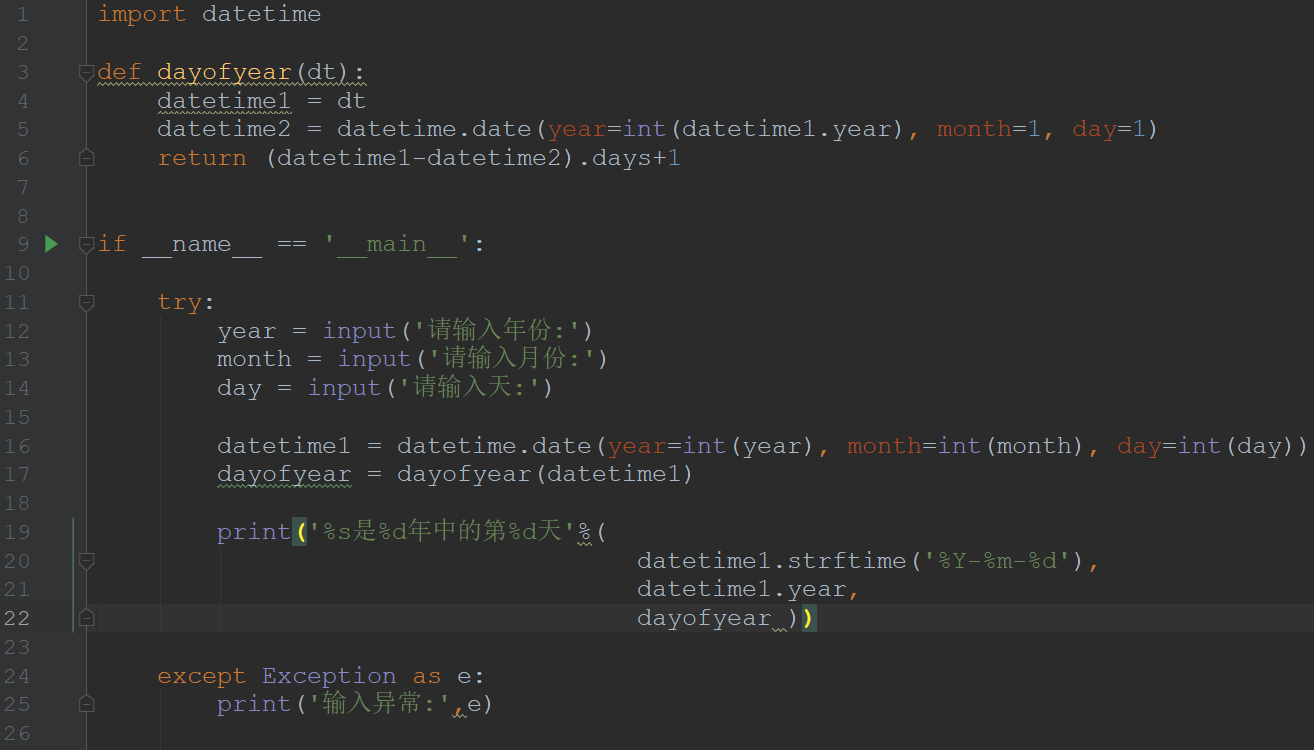
**代码1**

****

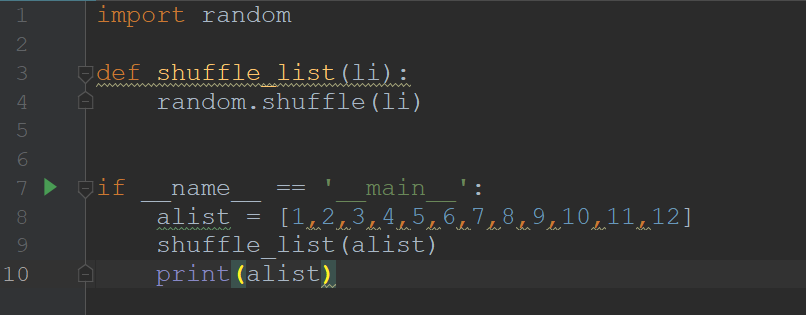
**代码2**

****

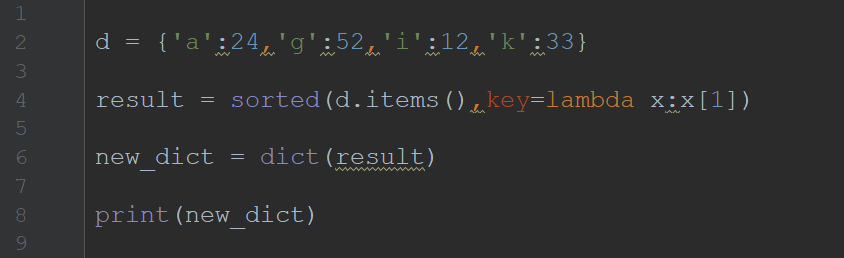
### 3.输入日期， 判断这一天是这一年的第几天？

****

### 4.打乱一个排好序的list对象alist？

****

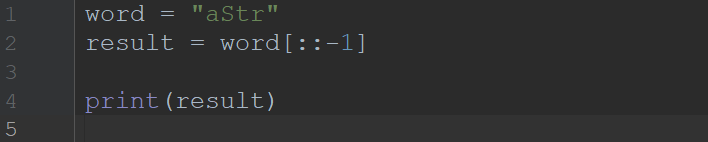
### 5.现有字典 d= {'a':24,'g':52,'i':12,'k':33}请按value值进行排序?



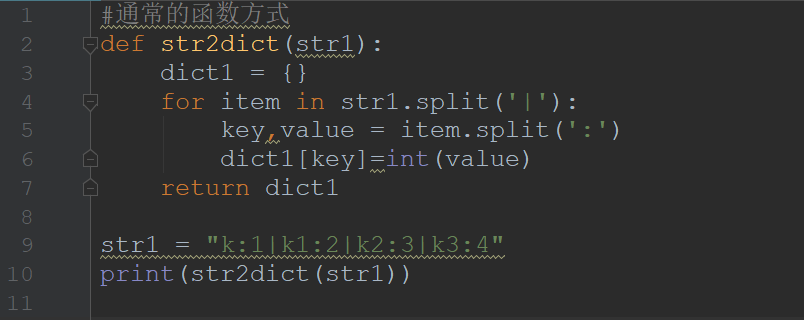
### 6.字典推导式

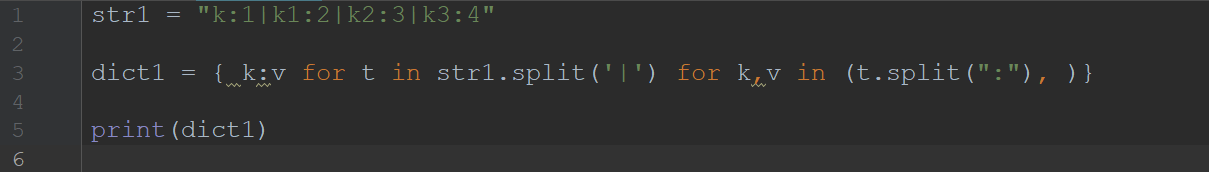
d = {key:value for (key,value) in iterable}

### 7.请反转字符串 "aStr"?

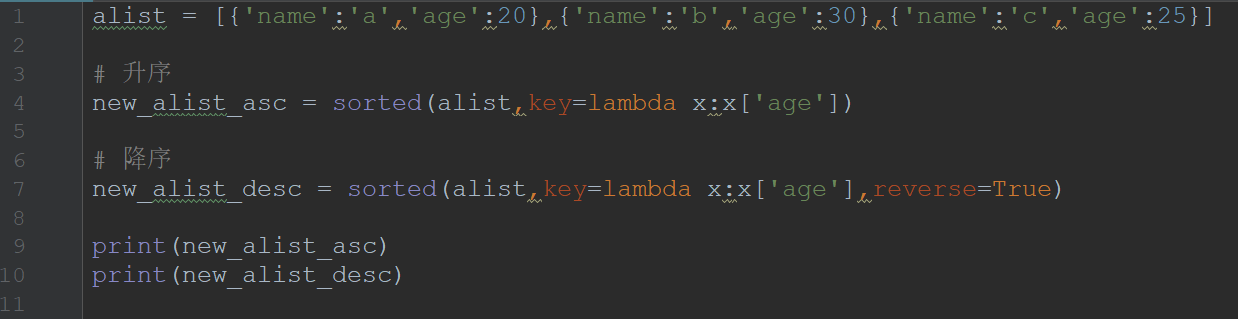


### 8.将字符串 "k:1 |k1:2|k2:3|k3:4"，处理成字典 {k:1,k1:2,...}

****



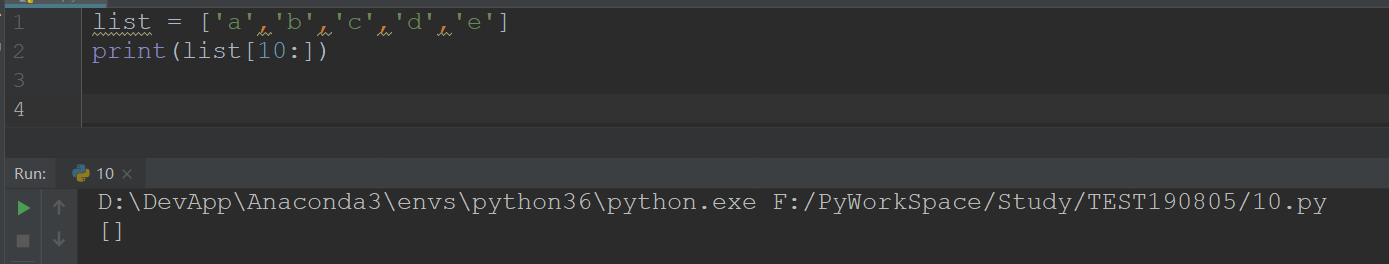
### 9.请按alist中元素的age由大到小排序

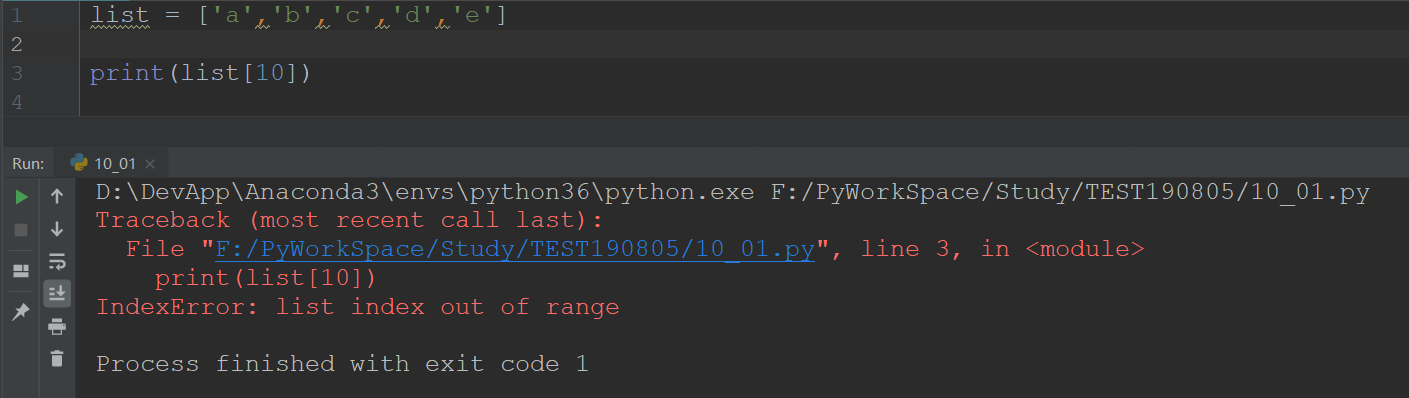


### 10.下面代码的输出结果将是什么？

list = ['a','b','c','d','e']

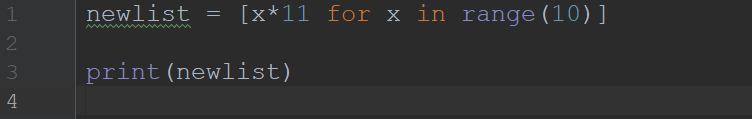
print(list[10:])



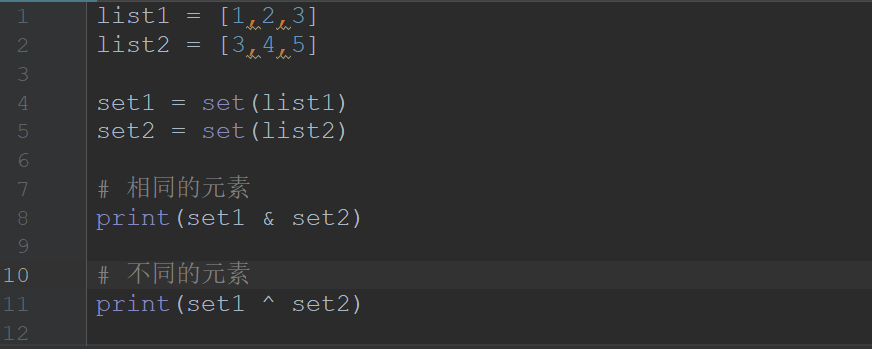


代码将输出[],不会产生IndexError错误，就像所期望的那样，尝试用超出成员的个数的index来获取某个列表的成员。例如，尝试获取list[10]和之后的成员，会导致IndexError。然而，尝试获取列表的切片，开始的index超过了成员个数不会产生IndexError，而是仅仅返回一个空列表。这成为特别让人恶心的疑难杂症，因为运行的时候没有错误产生，导致Bug很难被追踪到。

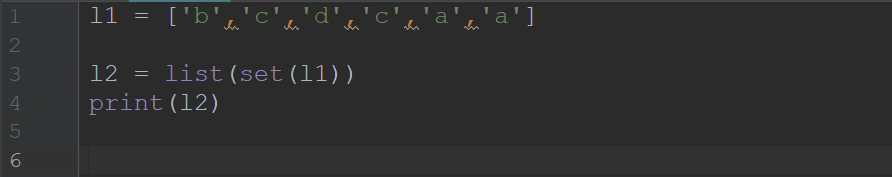
### 11.写一个列表生成式，产生一个公差为11的等差数列



### 12.给定两个列表，怎么找出他们相同的元素和不同的元素？

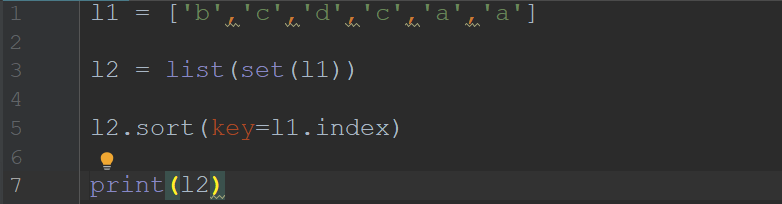


### 13.请写出一段python代码实现删除list里面的重复元素？

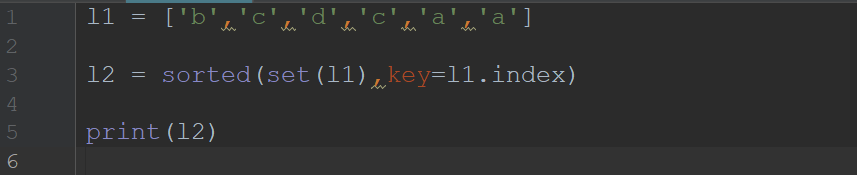


虽然很简单就可是实现去重的的效果，但是结果和我们想象的不一样，因为集合是无序的，会打乱原先列表的顺序，当然这是有解决办法的，我们只需要加一行代码就可以了，

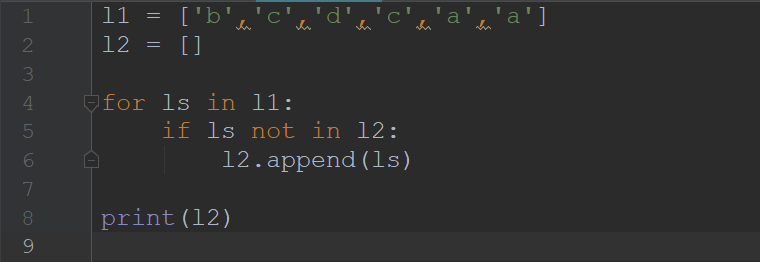
**下面是完整版：（用list类的sort方法）**



**也可以这样写**



**也可以用遍历**

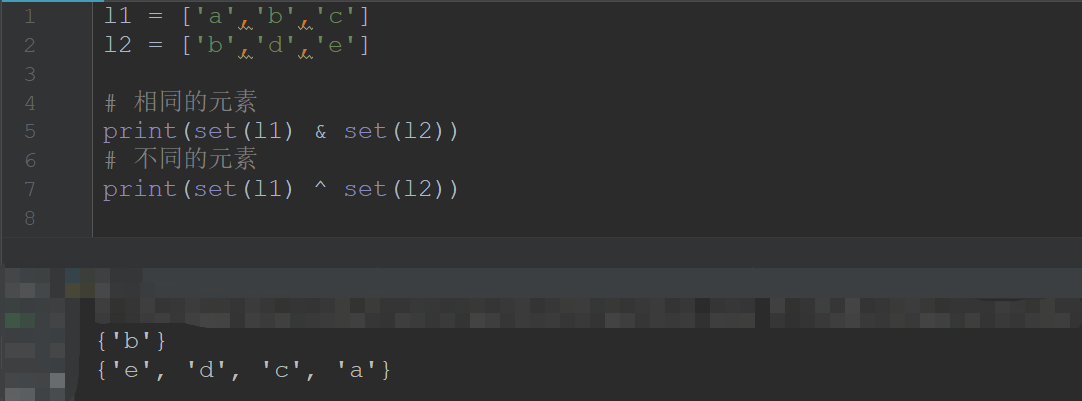


### 14.给定两个list A，B ,请用找出A，B中相同与不同的元素

A,B 中相同元素： print(set(A)&set(B))

A,B 中不同元素: print(set(A)^set(B))

**实现代码如下**



## 企业面试题

### 15.python新式类和经典类的区别？

a. 在python里凡是继承了object的类，都是新式类

b. Python3里只有新式类

c. Python2里面继承object的是新式类，没有写父类的是经典类

d. 经典类目前在Python里基本没有应用

### 16.python中内置的数据结构有几种？

a. 整型 int、 长整型 long、浮点型 float、 复数 complex

b. 字符串 str、 列表 list、 元组 tuple

c. 字典 dict 、 集合 set

d. Python3 中没有 long，只有无限精度的 int

### 17.python如何实现单例模式?请写出两种实现方式?

第一种方法:使用装饰器

def singleton(cls):

instances = {}

def wrapper(\*args, \*\*kwargs):

if cls not in instances:

instances[cls] = cls(\*args, \*\*kwargs)

return instances[cls]

return wrapper

@singleton

class Foo(object):

pass

foo1 = Foo()

foo2 = Foo()

print(foo1 is foo2) # True

第二种方法：使用基类 New 是真正创建实例对象的方法，所以重写基类的new 方法，以此保证创建对象的时候只生成一个实例

class Singleton(object):

def \_\_new\_\_(cls, \*args, \*\*kwargs):

if not hasattr(cls, '\_instance'):

cls.\_instance = super(Singleton, cls).\_\_new\_\_(cls, \*args, \*\*kwargs)

return cls.\_instance

class Foo(Singleton):

pass

foo1 = Foo()

foo2 = Foo()

print(foo1 is foo2) # True

第三种方法：元类，元类是用于创建类对象的类，类对象创建实例对象时一定要调用call方法，因此在调用call时候保证始终只创建一个实例即可，type是python的元类

class Singleton(type):

def \_\_call\_\_(cls, \*args, \*\*kwargs):

if not hasattr(cls, '\_instance'):

cls.\_instance = super(Singleton, cls).\_\_call\_\_(\*args, \*\*kwargs)

return cls.\_instance

# Python2

class Foo(object):

\_\_metaclass\_\_ = Singleton

# Python3

class Foo(metaclass=Singleton):

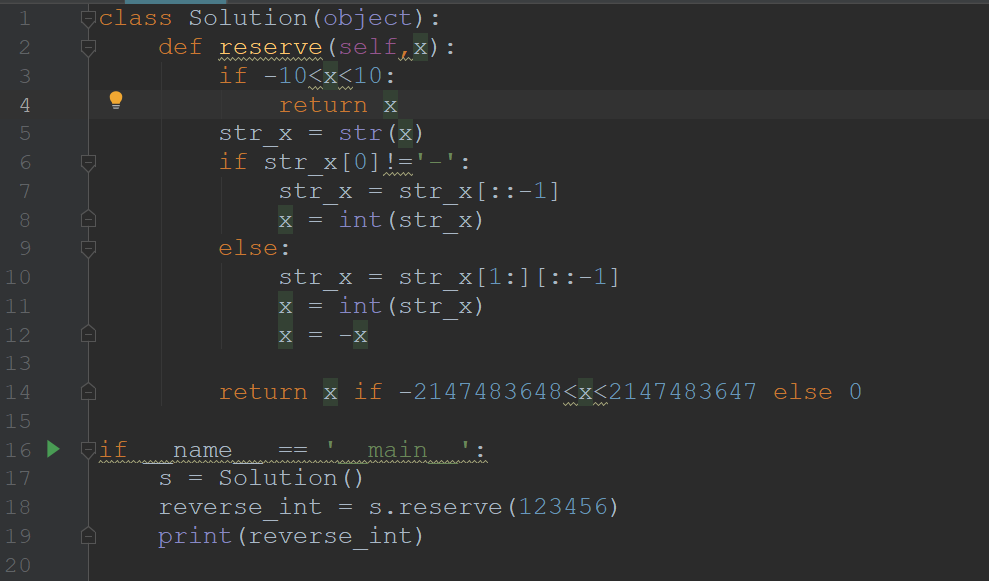
pass

foo1 = Foo()

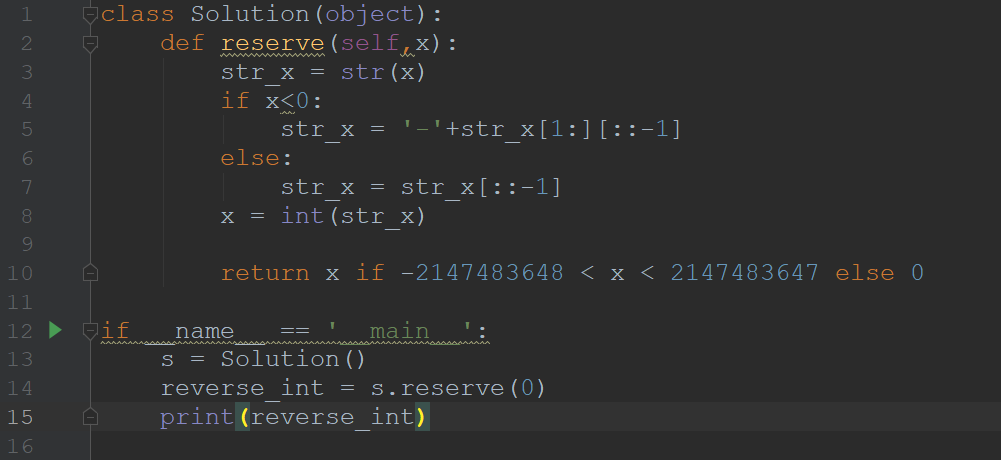
foo2 = Foo()

print(foo1 is foo2) # True

### 18.反转一个整数，例如-123 --> -321

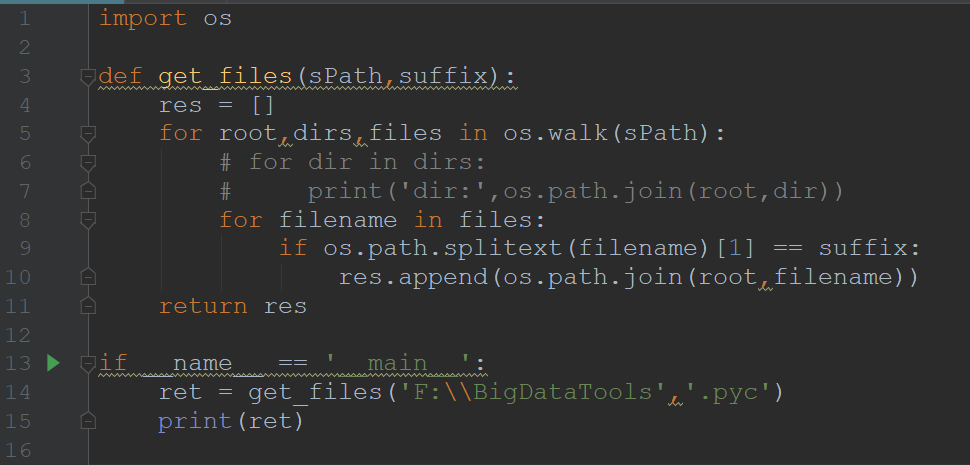
****

**也可以这样写**

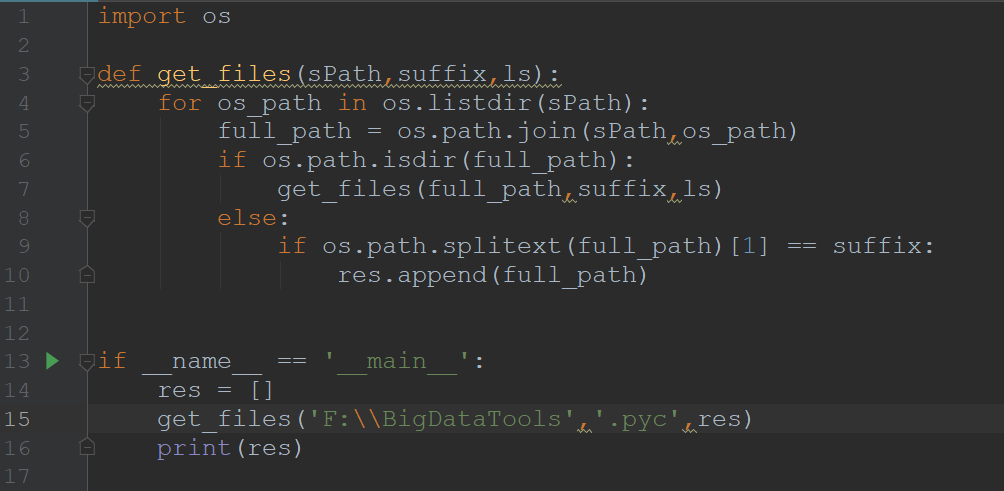


### 19.设计实现遍历目录与子目录，抓取.pyc文件

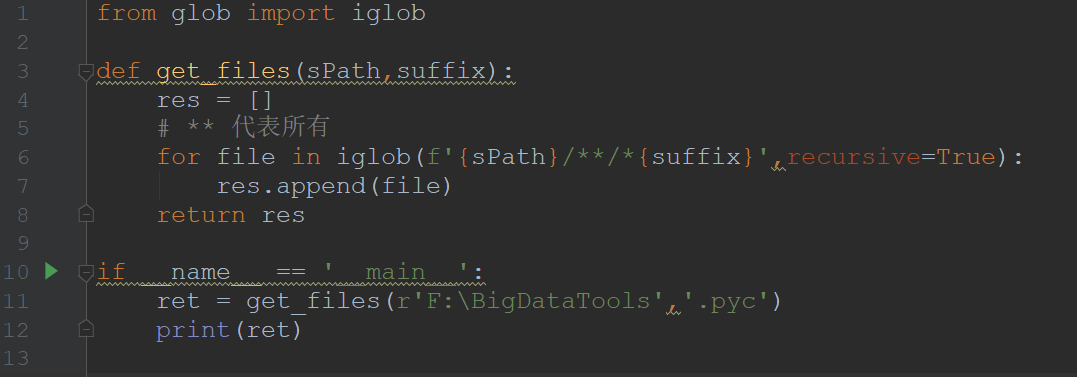
**第一种方法**



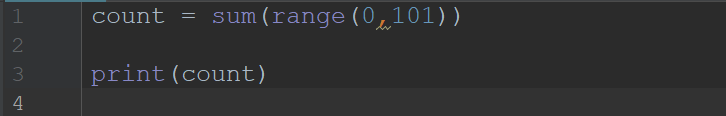
**第二种方法**



**第三种方法**

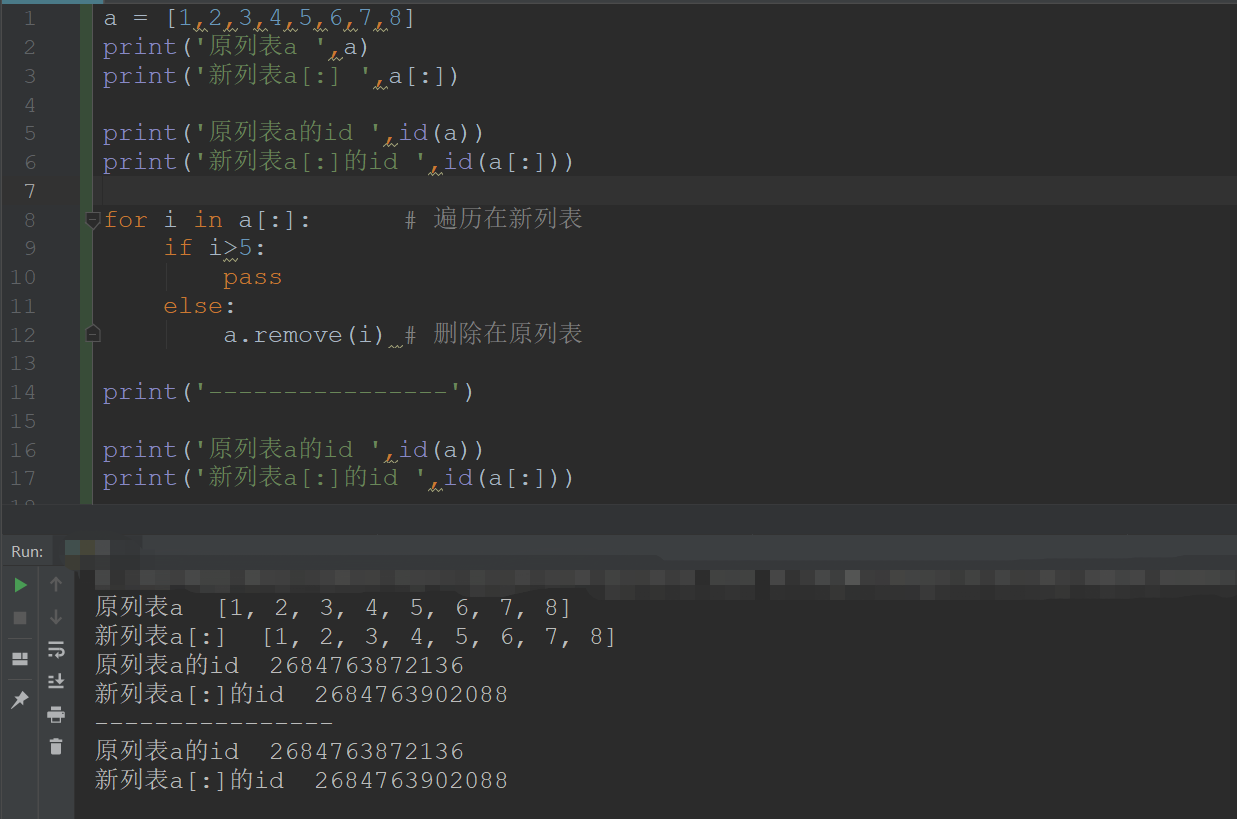


### 20.一行代码实现1-100之和

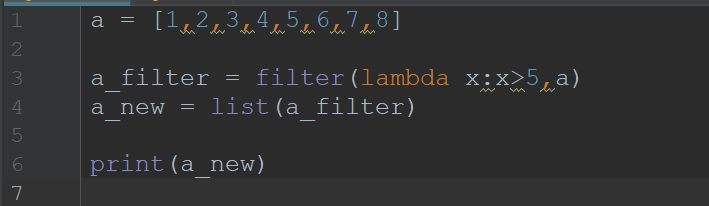


### 21.Python-遍历列表时删除元素的正确做法

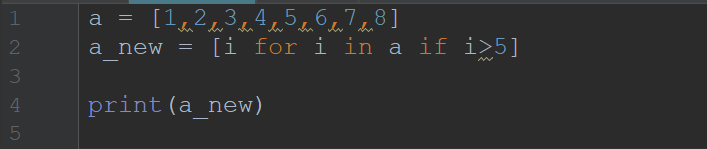
**遍历在新的列表操作，删除时在原来的列表操作**



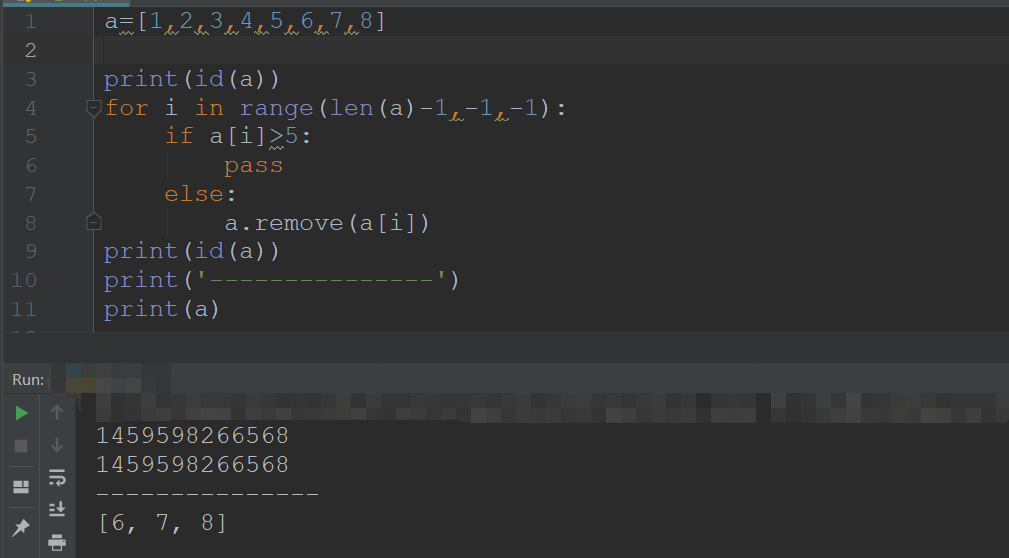
**Filter方式的代码如下**

****

**列表解析（列表推导式）**



**倒序删除，因为列表总是‘向前移’，所以可以倒序遍历，即使后面的元素被修改了，还没有被遍历的元素和其坐标还是保持不变的**



### 22.字符串的操作题目

全字母短句 PANGRAM 是包含所有英文字母的句子，比如：A QUICK BROWN FOX JUMPS OVER THE LAZY DOG. 定义并实现一个方法 get\_missing\_letter, 传入一个字符串采纳数，返回参数字符串变成一个 PANGRAM 中所缺失的字符。应该忽略传入字符串参数中的大小写，返回应该都是小写字符并按字母顺序排序（请忽略所有非 ACSII 字符）

**下面示例是用来解释，双引号不需要考虑:**

(0)输入: "A quick brown for jumps over the lazy dog"

返回： ""

(1)输入: "A slow yellow fox crawls under the proactive dog"

返回: "bjkmqz"

(2)输入: "Lions, and tigers, and bears, oh my!"

返回: "cfjkpquvwxz"

(3)输入: ""

返回："abcdefghijklmnopqrstuvwxyz"