



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

第三版：Python 48 道

找出两个列表中相同的元素和不同的元素

```
list1=[1,2,3,5,8,7,11,10]
list2=[5,15,25,10]
sim=[i for i in list1 if i in list2]
diff=[i for i in list1+list2 if i not in sim]
print(sim)
print(diff)
```

二叉树是非线性结构，栈和队列以及线性表都是线性结构，对吗？

对的

从 0-99 这 100 个数中随机取出 10 个，要求不能重复

```
import random
lis=random.sample(range(0,100),10)
print(lis)
```

有一个列表 lis=['This','is','a','Man','B','!']，对它进行大小写无关的排序

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

```
lis=['This','is','a','Man','B','!']  
lis=sorted(lis,key=str.lower)  
print(lis)
```

描述以下字典的 items()方法和 iteritems()方法有啥不同?

- 1、字典的 items 方法作用：是可以将字典中的所有项，以列表方式返回。因为字典是无序的，所以用 items 方法返回字典的所有项，也是没有顺序的。
- 2、字典的 iteritems 方法作用：与 items 方法相比作用大致相同，只是它的返回值不是列表，而是一个迭代器

请列举你所知道的 python 代码检测工具以及他们之间的区别

- 1、pylint --- 源代码分析器，可以分析 python 代码中的错误
- 2、pyflakes --- 检查源文件错误的简单程序，不会检查代码风格。
- 3、pep8 --- 检查代码规范的工具

介绍一下 try except 的用法和作用?

主要用来处理异常

完整用法如下：

```
try:
```

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

```
Normal execution block
except A:
    Exception A handle
except B:
    Exception B handle
except:
    Other exception handle
else:
    if no exception,get here
finally:
    print("finally")
```

阅读以下代码，写输出结果

```
lis = [2,4,5,6,7]
for i in lis:
    if i % 2 == 0:
        lis.remove(i)
print(lis)
```

结果：[4, 5, 7]

对列表[3,1,-4,-2]按照绝对值排序

```
lis=[3,1,-4,-2]
lis=sorted(lis,key=lambda x:abs(x))
print(lis)
```

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

获取 python 解释器版本的方法

终端执行 `python -V`

现有 mydict 和变量 onekey, 请写出从 mydict 中取出 onekey 的值得方法

方法一：`mydict[onekey]`

这种方法，如果 mydict 中键不存在的时候程序会报错

方法二：`mydict.get(onekey)`

这种方法，如果存在，返回值，不存在返回 None

方法三：`mydict.setdefault(onekey,[])`

这种方法：存在的话返回值，不存在的时候创建一个值，值得内容为第二个参数 +

列表中保留顺序和不保留顺序去重

不保留顺序

```
lis=[3, 1, 4, 2, 3]
print(list(set(lis)))
```

保留顺序

```
lis=[3, 1, 4, 2, 3]
T=[]
[T.append(i) for i in lis if i not in T]
print(T)
```

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

```
# 或者
T=sorted(set(lis), key=lis.index)
print(T)
```

在什么情况下 $y! = x - (x - y)$ 会成立?

x, y 是两个不相等的非空集合

实现 99 乘法表 (使用两种方法)

```
print('\n'.join(['\t'.join(['{}*{}={}'.format(x,y,x*y) for x in range(1,y+1)]) for
y in range(1,10)]))
```

```
for i in range(1,10):
for j in range(1,i+1):
print('%s*%s=%s'%(i,j,i*j),end='\t')
else:
print()
```

判断 dict 中有没有某个 key。

```
key in dict.keys() 判断
```

`a = dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))` 请问 a 是什么?

```
{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
```

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

在 python 中如何拷贝一个对象，并说明他们之间的区别

- 1、赋值 (=)，就是创建了对象的一个新的引用，修改其中任意一个变量都会影响到另一个。
- 2、浅拷贝：创建一个新的对象，但它包含的是对原始对象中包含项的引用（copy 模块的 copy() 函数）
- 3、深拷贝：创建一个新的对象，并且递归的复制它所包含的对象（修改其中一个，另外一个不会改变）（copy 模块的 deep.deeppcopy() 函数）

python 中进制转换

进制转换以十进制为媒介

十六进制前面加上 0x，八进制加上 0o，二进制前面加上 0b

	二进制	八进制	十进制	十六进制
二进制		bin(int(x, 8))	bin(int(x, 10))	bin(int(x, 16))
八进制	oct(int(x, 2))		oct(int(x, 10))	oct(int(x, 16))
十进制	int(x, 2)	int(x, 8)		int(x, 16)
十六进制	hex(int(x, 2))	hex(int(x, 8))	hex(int(x, 10))	

将列表

`alist=[{'name':'a','age':25},{'name':'b','age':30},{'name':'c','age':20}]`，按照 age 的值从大到小排列。

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

```
alist=[{'name':'a','age':25},{'name':'b','age':30},{'name':'c','age':20}]
blist=sorted(alist,key=lambda x:x['age'],reverse=True)
print(blist)
```

关于 Python 程序的运行方面，有什么手段能提升性能？

- 1、 使用多进程，充分利用机器的多核性能
- 2、 对于性能影响较大的部分代码，可以使用 C 或 C++ 编写
- 3、 对于 IO 阻塞造成的性能影响，可以使用 IO 多路复用来解决
- 4、 尽量使用 Python 的内建函数 5、 尽量使用局部变量

python 是如何进行内存管理的？ python 的程序会内存泄漏吗？说说有没有什么方面阻止或者检测内存泄漏？

python 是如何进行内存管理的[参考文章](#)

1、 引用计数

Python 内部使用引用计数，来保持追踪内存中的对象，Python 内部记录了对象有多少个引用，就是引用计数，当对象被创建时就创建了一个引用计数，当对象不再需要的时候，这个对象的引用计数为 0 时，他被垃圾回收。

2、 垃圾回收

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

当内存中有不再使用的部分时，垃圾收集器就会把他们清理掉。他会去检查那些引用计数为 0 的对象，然后清除其在内存中的空间。当然除了引用计数为 0 的会被清除，还有一种情况也会被垃圾收集器清掉，当两个对象相互引用时，他们本身其他引用已经为 0 了。

3、内存池机制

1、Python 提供了对内存的垃圾收集机制，但是他将不用的内存放到内存池而不是返回给操作系统。

2、python 的程序会内存泄漏吗？

3、会发生内存泄漏，在 Python 程序里，内存泄漏是由于一个长期持有的对象不断的往一个 dict 或者 list 对象里添加新的对象，而又没有即时释放，就会导致这些对象占用的内存越来越多，从而造成内存泄漏。另外，对象的交叉引用也会造成内存无法释放的问题。

说说有没有什么方面阻止或者检测内存泄漏？

1、程序员管理好每个 python 对象的引用，尽量在不需要使用对象的时候，断开所有引用

2、尽量少通过循环引用组织数据，可以改用 weakref 做弱引用或者用 id 之类的句柄访问对象

3、通过 gc 模块的接口可以检查出每次垃圾回收有哪些对象不能自动处理，再逐个处理

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

一个大小为 100G 的文件 etl_log.txt，要读取文件的内容，写出具体过程代码

```
with open("etl_log.txt",'r',encoding='utf8') as f:  
for line in f:  
print(line,end="")
```

python 代码如何获取命令行参数

[获取命令行参数的方法参考](#)

1、使用 sys 模块

通过 sys.argv 来获取

2、使用 getopt 模块

写代码：如何由 tuple1=('a','b','c','d','e')，和

tuple2=(1,2,3,4,5)得到 res={'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}

```
tuple1=('a','b','c','d','e')  
tuple2=(1,2,3,4,5)  
res=dict(zip(tuple1,tuple2))  
print(res)
```

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

1 < (22) 和 1 < 22 的结果分别是什么？

第一个为 False，第二个为 True，暂时按照第一个类型进行相应的转换

如何打乱一个排好序的列表

使用 random.shuffle()

```
import random
alist=[1,2,3,4,5,6]
random.shuffle(alist)
print(alist)
```

如何查找一个字符串中特定的字符？ find 和 index 的差异？

使用 find 和 index 方法查找

- 1、 find()方法：查找子字符串，若找到返回从 0 开始的下标值，若找不到返回-1
- 2、 index()方法：python 的 index 方法是在字符串里查找子串第一次出现的位置，类似字符串的 find 方法，不过比 find 方法更好的是，如果查找不到子串，会抛出异常，而不是返回-1

把 a='aaabbccddde'这种形式的字符串，压缩成

a3b2c3d4e1 这种形式。

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

```
a='aaabbcccdde'
aa=""
for i in sorted(list(set(a)),key=a.index):
aa=aa+i+str(a.count(i))
print(aa)
```

一个数如果恰好等于它的因子之和，这个数就称为‘完数’，比如 $6=1+2+3$ ，编程找出 1000 以内的所有的完数。

```
wanshu=[]
for i in range(1,1001):
s=0
for j in range(1,i//2+1):
if i % j ==0:
s+=j
else:
if i==s:
wanshu.append(i)
print(wanshu)
```

输入一个字符串，输出该字符串的字符的所有组合。如输入'abc'，输出 a,b,c,ab,ac,abc.

```
def getC(s):
if not s:
return
```

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

```
len_s=len(s)
ss=[]
for i in range(len_s):
    combination(s,0,i,ss)
aaa=[]
def combination(s,index,num,ss):
    global aaa
    if num== -1:
        return
    if index==len(s):
        return
    ss.append(s[index])
    aaa.append("".join(ss))
    combination(s,index+1,num-1,ss)
    ss.pop()
    combination(s,index+1,num,ss)

getC('123')
print(aaa)
print(sorted(set(aaa),key=lambda x:len(str(x))))
```

给定一个非空的字符串，判断它是否可以由它的一个子串重复多次构成。给定的字符串只含有小写英文字母，并且长度不超过10000。例如：'ababab'返回 True，'ababa'，返回 False

```
def solution(s):
    ll=len(s)
    for i in range(1,ll//2+1):
```

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

```
if ll % i == 0:
j=0
while s[:i]==s[j:j+i] and j<ll:
    j=j+i
if j==ll:
    return True
return False

print(solution('abababa'))
```

python 递归的最大层数?

1000

filter、map、reduce 的作用。

1、 filter() 相当于过滤器的作用

```
s=[1,2,3,5,6,8,9,10,25,12,30]
# 筛选出 3 的倍数
# 第一个参数为一个返回 True 或者 False 的函数，第二个参数为可迭代对象
# 该函数把可迭代对象依次传入第一个函数，如果为 True，则筛选
d=filter(lambda x:True if x % 3 == 0 else False,s)
print(list(d))
```

2、 map()函数，

```
# 第一个参数为函数，依次将后面的参数传给第一个函数，并执行函数
# 如果有多个参数则，依次将后面的对应传给参数
```

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

```
s=map(lambda x,y:x+y,range(10),range(10))
print(list(s))
ss=map(lambda x:x*x,range(10))
print(list(ss))
```

3、reduce()函数

```
from functools import reduce
# 开始的时候将可迭代对象的第一个数和第二个数当成 x 和 y
# 然后将第一次函数的执行结果当成 x，然后再传入一个数当成 y
# 再执行函数
s=reduce(lambda x,y:x+y,range(101))
print(s) # 相当于 0+1+2+.....+99+100
```

什么是闭包

在函数中可以（嵌套）定义另一个函数时，如果内部的函数引用了外部的函数的变量，则可能产生闭包。闭包可以用来在一个函数与一组“私有”变量之间创建关联关系。在给定函数被多次调用的过程中，这些私有变量能够保持其持久性。

```
# 内部函数使用了外部函数的变量，就相当于闭包
def func1():
    a=1
    def inner():
        return a
    return inner
print(func1())
```

简述生成器，迭代器，装饰器以及应用场景

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

- 1、 迭代器对象实现了 iter() 方法
- 2、 迭代器实现了 iter() 和 next() 方法, 迭代器对象从集合的第一个元素开始访问, 直到所有的元素被访问完结束
- 3、 生成器是迭代器的一种, 一个函数调用时返回一个迭代器, 这个函数就叫生成器。通常带有 yield
- 4、 装饰器是一个以函数作为参数, 并返回一个替换函数的可执行函数, 是闭包的一种应用。通常用来给一个函数添加功能

使用生成器编写一个函数实现生成指定个数的斐波那契数列

```
def fib2(imax):  
    t,a,b=0,0,1  
    while t<imax:  
        yield b  
        a,b=b,a+b  
        t+=1  
  
for i in fib2(10):  
    print(i)
```

一行代码通过 filter 和 lambda 函数输出

alist=[1,22,2,33,23,32]中索引为奇数的值

```
alist=[1,22,2,33,23,32]
```

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

```
ss=[x[1] for x in filter(lambda x:x[0]%2==1,enumerate(alist))]  
print(ss)
```

编写一个函数实现十进制转 62 进制，分别用 0-9A-Za-z,表示 62 位字母

```
import string  
print(string.ascii_lowercase) # 小写字母  
print(string.ascii_uppercase) # 大写字母  
print(string.digits) # 0-9  
  
s=string.digits+string.ascii_uppercase+string.ascii_lowercase  
def _10_to_62(num):  
    ss=""  
    while True:  
        ss=s[num%62]+ss  
        if num//62==0:  
            break  
        num=num//62  
    return ss  
print(_10_to_62(65))
```

实现一个装饰器，限制该函数被调用的频率，如 10 秒一次

```
import time  
from functools import wraps  
def dec(func):  
    key=func.__name__
```

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

```
cache={key:None}
@wraps(func)
def inner(*args,**kwargs):
    result=None
    if cache.get(key) is None:
        cache[key]=time.time()
        result=func(*args,**kwargs)
        print('执行函数中')
    else:
        now=time.time()
        if now-cache[key]> 10:
            cache[key]=now
            result=func(*args,**kwargs)
            print('执行函数中')
        else:
            print('函数执行受限')
    return result
return inner

@dec
def add(x,y):
    print(x+y)

add(1,2)
add(1,3)
time.sleep(10)
add(3,4)
```

实现一个装饰器，通过一次调用，使函数重复执行 5 次

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

```
from functools import wraps
def dec(func):
    @wraps(func)
    def inner(*args, **kwargs):
        result = [func(*args, **kwargs) for i in range(5)]
        return result
    return inner

@dec
def add(x,y):
    return x+y
print(add(1,2))
```

生成器与函数的区别？

生成器和函数的主要区别在于函数 return a value，生成器 yield a value 同时标记或记忆 point of the yield 以便于在下次调用时从标记点恢复执行。 yield 使函数转换成生成器，而生成器反过来又返回迭代器。

```
# 简单实现生成器
def dec():
    n=0
    for i in range(10):
        yield n
        n+=i

for i in dec():
    print(i)
```

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

列表推导式[i for i in range(10)]和生成式表达式(i for i in range(10))的区别

[参考文章](#)

- 1、列表推导式的结果是一个列表。
- 2、生成器表达式的结果是一个生成器，它和列表推导式类似，它一次处理一个对象，而不是一口气处理和构造整个数据结构，可以节约内存。

python 如何定义函数时如何书写可变参数和关键字参数?

```
def func(a,*args,b=1,**kwargs):  
    pass
```

python 中 enumerate 的意思是什么?

枚举的意思，同时得到可迭代对象，如列表和元组的索引和值，以元组形式返回

描述以下 dict 的 items 和 iteritems 的区别

python3 中没有 iteritems

items 和 iteritems 大致相同，只是 items 返回的是一个列表，iteritems 返回的是一个迭代器。

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

是否使用过 `functools` 中的函数？他的作用是什么？

1、 `functools.wraps()`

在装饰器中用过，如果不使用 `wraps`，则原始函数的 `name` 和 `doc` 的值就会丢失

2、 `functools.reduce()`

第一个参数是一个函数，第二个参数是一个可迭代对象，代码如下：

```
# 下面代码相当于从 1 加到 9
from functools import reduce
a=reduce(lambda x,y:x+y,range(10))
print(a)
```

如何判断一个值是方法还是函数？

- 1、使用 `type()` 来判断，如果是 `method` 为方法，如果是 `function` 则是函数。
- 2、与类和实例无绑定关系的 `function` 都属于函数 (`function`)
- 3、与类和实例有绑定关系的 `function` 都属于方法

请编写一个函数将 ip 地址转换成一个整数。如 10.3.9.12 转换成 00001010 00000011 00001001 00001100，然后转换成整数

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

```
def ip2int(ip):
    nums=ip.split('.')
    # zfill()函数是补0
    to_bin=[bin(int(i))[2:].zfill(8) for i in nums]
    return int("".join(to_bin),2)
i=ip2int('127.0.0.1')
print(i)
```

lambda 表达式格式以及应用场景?

格式：lambda 参数列表：返回表达式

应用场景：常见的在 filter, reduce 以及 map 中使用。

pass 的使用

通常用来标记一个还未写的代码的位置，pass 不做任何事情，一般用来做占位语句，保持程序结构的完整性

*arg 和**kwargs 的作用

用来接收不确定个数的参数，*args 通常用来接收不确定个数的非关键字参数，而**kwargs 通常用来接收不确定个数的关键字参数

如何在函数中设置一个全局变量?

在函数中使用 global 关键字定义变量

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

求以下代码结果：

```
def num():  
    return [lambda x:i*x for i in range(4)]  
print([m(2) for m in num()])
```

答案：[6,6,6,6]

yield from 和 yield 的区别

[简述 yield 和 yield from](#)

下面 a()和 b()是等价的

```
def a():  
    yield from [1,2,3,4,5]  
def b():  
    for i in [1,2,3,4,5]:  
        yield i  
for i in a():  
    print(i)  
for i in b():  
    print(i)
```

yield 将一个函数变成一个生成器

yield 返回一个值

yield from 后面接可迭代对象，一个一个返回值。

关注公众号：磊哥聊编程，回复：面试题，获取最新版面试题



微信搜一搜

磊哥聊编程

扫码关注



回复：面试题 获取最新版面试题

求以下代码的输出结果

```
collapse=True  
processFunc=collapse and (lambda s: ' '.join(s.split())) or (lambda s:s)  
print(processFunc('i\tam\ntest\tproject!'))
```

```
collapse=False  
processFunc=collapse and (lambda s: ' '.join(s.split())) or (lambda s:s)  
print(processFunc('i\tam\ntest\tproject!'))
```

答案：

i am test project!

i am

test project!