1. Python 是如何进行内存管理的？

答:从三个方面来说,一对象的引用计数机制,二垃圾回收机制,三内存池机制

一、对象的引用计数机制

python 内部使用引用计数，来保持追踪内存中的对象，所有对象都有引用计数。

引用计数增加的情况：

1，一个对象分配一个新名称

2，将其放入一个容器中（如列表、元组或字典）

引用计数减少的情况：

1，使用 del 语句对对象别名显示的销毁

2，引用超出作用域或被重新赋值

sys.getrefcount( )函数可以获得对象的当前引用计数

多数情况下，引用计数比你猜测得要大得多。对于不可变数据（如数字和字符串），解释器会在程序的不同部分共享内存，以便节约内存。

二、垃圾回收

1，当一个对象的引用计数归零时，它将被垃圾收集机制处理掉。

2， 当两个对象 a 和 b 相互引用时，del 语句可以减少 a 和 b 的引用计数，并销毁用于引用底层对象的名称。然而由于每个对象都包含一个对其他对象的应用，因此引 用计数不会归零，对象也不会销毁。（从而导致内存泄露）。为解决这一问题，解释器会定期执行一个循环检测器，搜索不可访问对象的循环并删除它们。

三、内存池机制

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

1，Pymalloc 机制。为了加速 Python 的执行效率，Python 引入了一个内存池机制，用于管理对小块内存的申请和释放。

2，Python 中所有小于 256 个字节的对象都使用 pymalloc 实现的分配器，而大的对象则使用系统的 malloc。

3，对于 Python 对象，如整数，浮点数和 List，都有其独立的私有内存池，对象间不共享他们的内存池。也就是说如果你分配又释放了大量的整数，用于缓存这些整数的内存就不能再分配给浮点数。

2. 什么是 lambda 函数？它有什么好处?

答：lambda 表达式，通常是在需要一个函数，但是又不想费神去命名一个函数的场合下使用，也就是指匿名函数

lambda 函数：首要用途是指点短小的回调函数

lambda [arguments]:expression

```
>>> a=lambdax,y:x+y
>>> a(3,11)
```

3. Python 里面如何实现 tuple 和 list 的转换？

答：直接使用 tuple 和 list 函数就行了，type()可以判断对象的类型

4. 请写出一段 Python 代码实现删除一个 list 里面的重复元素

答：

1,使用 set 函数，set(list)

2，使用字典函数，

```
>>>a=[1,2,4,2,4,5,6,5,7,8,9,0]
>>> b={}
>>>b=b.fromkeys(a)
>>>c=list(b.keys())
>>> c
```

5. 编程用 sort 进行排序，然后从最后一个元素开始判断

```
a=[1,2,4,2,4,5,7,10,5,5,7,8,9,0,3]
a.sort()
last=a[-1]
for i inrange(len(a)-2,-1,-1):
if last==a[i]:
del a[i]
else:last=a[i]
print(a)
```

6. Python 里面如何拷贝一个对象？（赋值，浅拷贝，深拷贝的区别）

答：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。

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（如果用引用的方式修改其中一个对象，另外一个也会修改改变）{1,完全切片方法；2，工厂函数，如 list()；3，copy 模块的 copy()函数}

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.deepcopy()函数}

7. 介绍一下 except 的用法和作用？

答：try…except…except…[else…][finally…]

执行 try 下的语句，如果引发异常，则执行过程会跳到 except 语句。对每个 except 分支顺序尝试执行，如果引发的异常与 except 中的异常组匹配，执行相应的语句。如果所有的 except 都不匹配，则异常会传递到下一个调用本代码的最高层 try 代码中。
try 下的语句正常执行，则执行 else 块代码。如果发生异常，就不会执行
如果存在 finally 语句，最后总是会执行。

8. Python 中 pass 语句的作用是什么？

答：pass 语句不会执行任何操作，一般作为占位符或者创建占位程序，whileFalse:pass

9. 介绍一下 Python 下 range()函数的用法？

答：列出一组数据，经常用在 for in range()循环中

10. 如何用 Python 来进行查询和替换一个文本字符串？

答：可以使用 re 模块中的 sub()函数或者 subn()函数来进行查询和替换，
格式：sub(replacement, string[,count=0])（replacement 是被替换成的文本，string 是需要被替换的文本，count 是一个可选参数，指最大被替换的数量）

```
>>> import re
>>>p=re.compile('blue|white|red')
>>>print(p.sub('colour','blue socks and red shoes'))
colour socks and colourshoes
>>>print(p.sub('colour','blue socks and red shoes',count=1))
colour socks and redshoes
```

subn()方法执行的效果跟 sub()一样，不过它会返回一个二维数组，包括替换后的新的字符串和总共替换的数量

11. Python 里面 match()和 search()的区别？

答：re 模块中 match(pattern,string[,flags]),检查 string 的开头是否与 pattern 匹配。
re 模块中 research(pattern,string[,flags]),在 string 搜索 pattern 的第一个匹配值。

```
>>>print(re.match('super', 'superstition').span())
(0, 5)
>>>print(re.match('super', 'insuperable'))
None
>>>print(re.search('super', 'superstition').span())
(0, 5)
```

>>>print(re.search('super', 'insuperable').span())

(2, 7)

12. 用 Python 匹配 HTML tag 的时候，<.*>和<.*?>有什么区别？

答：术语叫贪婪匹配(<.*>)和非贪婪匹配(<.*?>)

例如:

test

<.*> : test

<.*?> :

13. Python 里面如何生成随机数？

答：random 模块

随机整数：random.randint(a,b)：返回随机整数 x,a<=x<=b

random.randrange(start,stop,[,step])：返回一个范围在(start,stop,step)之间的随机整数，不包括结束值。

随机实数：random.random( ):返回 0 到 1 之间的浮点数

random.uniform(a,b):返回指定范围内的浮点数。

14. 有没有一个工具可以帮助查找 python 的 bug 和进行静态的代码分析？

答：PyChecker 是一个 python 代码的静态分析工具，它可以帮助查找 python 代码的 bug，会对代码的复杂度和格式提出警告

Pylint 是另外一个工具可以进行 codingstandard 检查

15. 如何在一个 function 里面设置一个全局的变量？

答：解决方法是在 function 的开始插入一个 global 声明：

def f()

global x

16. 单引号，双引号，三引号的区别

答：单引号和双引号是等效的，如果要换行，需要符号(\),三引号则可以直接换行，并且可以包含注释

如果要表示 Let’s go 这个字符串

单引号：s4 = ‘Let\’s go’

双引号：s5 = “Let’s go”

s6 = ‘I realy like“python”!’

这就是单引号和双引号都可以表示字符串的原因了

17. 如何用 Python 来发送邮件？

可以使用 smtplib 标准库。

以下代码可以在支持 SMTP 监听器的服务器上执行。

```
import sys, smtplib
fromaddr =raw_input(“From: “)
toaddrs = raw_input(“To: “).split(‘,’)
print “Enter message, end with ^D:”
msg = ”
while 1:
line = sys.stdin.readline()
if not line:
break
msg = msg + line
# 发送邮件部分
server = smtplib.SMTP(‘localhost’)
server.sendmail(fromaddr, toaddrs, msg)
server.quit()
```

18. Python 如何实现单例模式？其他 23 种设计模式 python 如何实现？

Python 有两种方式可以实现单例模式，下面两个例子使用了不同的方式实现单例模式：

1.

```
class Singleton(type):
def __init__(cls, name, bases, dict):
super(Singleton, cls).__init__(name, bases, dict)
cls.instance = None
def __call__(cls, *args,**kw):
if cls.instance is None:
cls.instance = super(Singleton, cls).__call__(*args, **kw)
```

```
return cls.instance
class MyClass(object):
__metaclass__ = Singleton
print MyClass()
print MyClass()
```

2. 使用 decorator 来实现单例模式

```
def singleton(cls):
instances = {}
def getinstance():
if cls not in instances:
instances[cls] = cls()
return instances[cls]
return getinstance
@singleton
class MyClass:
…
```

19. 华为一道编程

有两个序列 a,b，大小都为 n,序列元素的值任意整形数，无序；

要求：通过交换 a,b 中的元素，使[序列 a 元素的和]与[序列 b 元素的和]之间的差最小。

1. 将两序列合并为一个序列，并排序，为序列 Source
2. 拿出最大元素 Big，次大的元素 Small
3. 在余下的序列 S[:-2]进行平分，得到序列 max，min
4. 将 Small 加到 max 序列，将 Big 加大 min 序列，重新计算新序列和，和大的为 max，小的为 min。

Python 代码

```
def mean( sorted_list ):
if not sorted_list:
return (([],[]))
big = sorted_list[-1]
small = sorted_list[-2]
big_list, small_list =mean(sorted_list[:-2])
big_list.append(small)
small_list.append(big)
big_list_sum =sum(big_list)
small_list_sum =sum(small_list)
```

```
if big_list_sum >small_list_sum:
return ( (big_list,small_list))
else:
return (( small_list,big_list))
tests = [ [1,2,3,4,5,6,700,800],
[10001,10000,100,90,50,1],
range(1, 11),
[12312, 12311, 232, 210,30, 29, 3, 2, 1, 1]
]
for l in tests:
l.sort()
print
print “Source List:\t”,l
l1,l2 = mean(l)
print “Result List:\t”,l1, l2
print “Distance:\t”,abs(sum(l1)-sum(l2))
print ‘-*’*40
```

输出结果

Python 代码

```
Source List: [1, 2, 3, 4, 5, 6, 700, 800]
Result List: [1, 4, 5, 800] [2, 3, 6, 700]
Distance: 99
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
Source List: [1, 50, 90, 100, 10000, 10001]
Result List: [50, 90, 10000] [1, 100, 10001]
Distance: 38
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
Source List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Result List: [2, 3, 6, 7, 10] [1, 4, 5, 8, 9]
Distance: 1
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
Source List: [1, 1, 2, 3, 29, 30, 210, 232, 12311, 12312]
Result List: [1, 3, 29, 232, 12311] [1, 2, 30, 210, 12312]
Distance: 21
```

20. python 程序中文输出问题怎么解决？

方法一：

用 encode 和 decode

如：

```
import os.path
import xlrd,sys
Filename=’/home/tom/Desktop/1234.xls’
if not os.path.isfile(Filename):
raise NameError,”%s is not a valid filename”%Filename
bk=xlrd.open_workbook(Filename)
shxrange=range(bk.nsheets)
print shxrange
for x in shxrange:
p=bk.sheets()[x].name.encode(‘utf-8′)
print p.decode(‘utf-8′)
```

方法二：

在文件开头加上

```
reload(sys)
sys.setdefaultencoding(‘utf8′)
```

这 2 行，再试着运行一下

字符串在 Python 内部的表示是 unicode 编码，因此，在做编码转换时，通常需要以 unicode 作为中间编码，即先将其他编码的字符串解码（decode）成 unicode，再从 unicode 编码（encode）成另一种编码。

decode 的作用是将其他编码的字符串转换成 unicode 编码，如 str1.decode('gb2312')，表示将 gb2312 编码的字符串 str1 转换成 unicode 编码。

encode 的作用是将 unicode 编码转换成其他编码的字符串，如 str2.encode('gb2312')，表示将 unicode 编码的字符串 str2 转换成 gb2312 编码。

因此，转码的时候一定要先搞明白，字符串 str 是什么编码，然后 decode 成 unicode，然后再 encode 成其他编码

代码中字符串的默认编码与代码文件本身的编码一致。

如：s='中文'

如 果是在 utf8 的文件中，该字符串就是 utf8 编码，如果是在 gb2312 的文件中，则其编码为 gb2312。这种情况下，要进行编码转换，都需要先用 decode 方法将其转换成 unicode 编码，再使用 encode 方法将其转换成其他编码。通常，在没有指定特定的编码方式时，都是使用的系统默认编码 创建的代码文件。

如果字符串是这样定义：s=u'中文'

则该字符串的编码就被指定为 unicode 了，即 python 的内部编码，而与代码文件本身的编码无关。因此，对于这种情况做编码转换，只需要直接使用 encode 方法将其转换成指定编码即可。

如果一个字符串已经是 unicode 了，再进行解码则将出错，因此通常要对其编码方式是否为 unicode 进行判断：

isinstance(s,unicode) #用来判断是否为 unicode

用非 unicode 编码形式的 str 来 encode 会报错

如何获得系统的默认编码？

```
#!/usr/bin/env python
#coding=utf-8
import sys
print sys.getdefaultencoding()
```

该段程序在英文 WindowsXP 上输出为：ascii

21. python 代码得到列表 list 的交集与差集

交集

```
b1=[1,2,3]
b2=[2,3,4]
b3 = [val for val in b1if val in b2]
print b3
```

差集

```
b1=[1,2,3]
b2=[2,3,4]
b3 = [val for val in b1 if val not in b2]
print b3
```

差集实例

———-

```
#/bin/env python
# -*- coding:utf-8 -*-
f =open('C:\diff_dealer\excel.txt')
excel = f.readlines()
f.close()
f= open('C:\diff_dealer\db.txt')
db = f.readlines()
diff = [val for val in db if val not in excel]
f.close()
```

f =open('C:\diff_dealer\diff.txt', 'w')

f.writelines(diff)

f.close()

print diff

22. 写一个简单的 python socket 编程

python 编写 server 的步骤：

1 第一步是创建 socket 对象。调用 socket 构造函数。如：

socket = socket.socket(family, type)

family 参数代表地址家族，可为 AF_INET 或 AF_UNIX。AF_INET 家族包括 Internet 地址，AF_UNIX 家族用于同一台机器上的进程间通信。

type 参数代表套接字类型，可为 SOCK_STREAM(流套接字)和 SOCK_DGRAM(数据报套接字)。

2.第二步是将 socket 绑定到指定地址。这是通过 socket 对象的 bind 方法来实现的：

socket.bind(address)由 AF_INET 所创建的套接字，address 地址必须是一个双元素元组，格式是(host,port)。host 代表主机，port 代表端口号。如 果端口号正在使用、主机名不正确或端口已被保留，bind 方法将引发 socket.error 异常。

3.第三步是使用 socket 套接字的 listen 方法接收连接请求。

socket.listen(backlog)

backlog 指定最多允许多少个客户连接到服务器。它的值至少为 1。收到连接请求后，这些请求需要排队，如果队列满，就拒绝请求。

4.第四步是服务器套接字通过 socket 的 accept 方法等待客户请求一个连接。

connection, address =socket.accept()

调 用 accept 方法时，socket 会时入"waiting"状态。客户请求连接时，方法建立连接并返回服务器。accept 方法返回一个含有两个元素的 元组(connection,address)。第一个元素 connection 是新的 socket 对象，服务器必须通过它与客户通信；第二个元素 address 是客户的 Internet 地址。

5. 第五步是处理阶段，服务器和客户端通过 send 和 recv 方法通信(传输数据)。服务器调用 send，并采用字符串形式向客户发送信息。send 方法返回 已发送的字符个数。服务器使用 recv 方法从客户接收信息。调用 recv 时，服务器必须指定一个整数，它对应于可通过本次方法调用来接收的最大数据量。recv 方法在接收数据时会进入"blocked"状态，最后返回一个字符 串，用它表示收到的数据。如果发送的数据量超过了 recv 所允许的，数据会被截短。多余的数据将缓冲于接收端。以后调用 recv 时，多余的数据会从缓冲区 删除(以及自上次调用 recv 以来，客户可能发送的其它任何数据)。

6. 传输结束，服务器调用 socket 的 close 方法关闭连接。

python 编写 client 的步骤：

1. 创建一个 socket 以连接服务器：socket= socket.socket(family, type)

2.使用 socket 的 connect 方法连接服务器。对于 AF_INET 家族,连接格式如下：

socket.connect((host,port))

host 代表服务器主机名或 IP，port 代表服务器进程所绑定的端口号。如连接成功，客户就可通过套接字与服务器通信，如果连接失败，会引发 socket.error 异常。

3. 处理阶段，客户和服务器将通过 send 方法和 recv 方法通信。

4. 传输结束，客户通过调用 socket 的 close 方法关闭连接。

下面给个简单的例子：

server.py

```
if __name__ =='__main__':
import socket
sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
sock.bind(('localhost',8001))
sock.listen(5)
while True:
connection,address =sock.accept()
try:
connection.settimeout(5)
buf =connection.recv(1024)
if buf == '1':
connection.send('welcometo server!')
else:
connection.send('pleasego out!')
except socket.timeout:
print 'time out'
connection.close()
```

client.py

python 代码

```
if __name__ =='__main__':
import socket
sock =socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(('localhost',8001))
import time
time.sleep(2)
sock.send('1')
```

print sock.recv(1024)

sock.close()

在终端运行 server.py，然后运行 clien.py，会在终端打印“welcometo server!"。如果更改 client.py 的 sock.

23. Python 文件操作的面试题

1. 如何用 Python 删除一个文件？

使用 os.remove(filename)或者 os.unlink(filename);

2. Python 如何 copy 一个文件？

shutil 模块有一个 copyfile 函数可以实现文件拷贝

1：Python 如何实现单例模式？

Python 有两种方式可以实现单例模式，下面两个例子使用了不同的方式实现单例模式：

1.

```
class Singleton(type):
def __init__(cls, name, bases, dict):
super(Singleton, cls).__init__(name, bases, dict)
cls.instance = None
def __call__(cls, *args, **kw):
if cls.instance is None:
cls.instance = super(Singleton, cls).__call__(*args, **kw)
return cls.instance
class MyClass(object):
__metaclass__ = Singleton
print MyClass()
print MyClass()
```

2. 使用 decorator 来实现单例模式

```
def singleton(cls):
instances = {}
def getinstance():
if cls not in instances:
instances[cls] = cls()
return instances[cls]
return getinstance
```

@singleton

class MyClass:

…

2：什么是 lambda 函数？

Python 允许你定义一种单行的小函数。定义 lambda 函数的形式如下：labmda 参数：表达式 lambda 函数默认返回表达式的值。你也可以将其赋值给一个变量。lambda 函数可以接受任意个参数，包括可选参数，但是表达式只有一个：

```
>>> g = lambda x, y: x*y
>>> g(3,4)
12
>>> g = lambda x, y=0, z=0: x+y+z
>>> g(1)
1
>>> g(3, 4, 7)
14
```

也能够直接使用 lambda 函数，不把它赋值给变量：

```
>>> (lambda x,y=0,z=0:x+y+z)(3,5,6)
14
```

如果你的函数非常简单，只有一个表达式，不包含命令，可以考虑 lambda 函数。否则，你还是定义函数才对，毕竟函数没有这么多限制。

3：Python 是如何进行类型转换的？

Python 提供了将变量或值从一种类型转换成另一种类型的内置函数。int 函数能够将符合数学格式数字型字符串转换成整数。否则，返回错误信息。

```
>>> int(”34″)
34
>>> int(”1234ab”) #不能转换成整数
ValueError: invalid literal for int(): 1234ab
```

函数 int 也能够把浮点数转换成整数，但浮点数的小数部分被截去。

```
>>> int(34.1234)
34
>>> int(-2.46)
-2
```

函数°oat 将整数和字符串转换成浮点数：

```
>>> float(”12″)
12.0
```

>>> float(”1.111111″)

1.111111

函数 str 将数字转换成字符：

>>> str(98)

‘98′

>>> str(”76.765″)

‘76.765′

整数 1 和浮点数 1.0 在 python 中是不同的。虽然它们的值相等的，但却属于不同的类型。这两个数在计算机的存储形式也是不一样。

4：Python 如何定义一个函数

函数的定义形式如下：

def <name>(arg1, arg2,… argN):

<statements>

函数的名字也必须以字母开头，可以包括下划线“ ”,但不能把 Python 的关键字定义成函数的名字。函数内的语句数量是任意的，每个语句至少有一个空格的缩进，以表示此语句属于这个函数的。缩进结束的地方，函数自然结束。

下面定义了一个两个数相加的函数：

>>> def add(p1, p2):

print p1, “+”, p2, “=”, p1+p2

>>> add(1, 2)

1 + 2 = 3

函数的目的是把一些复杂的操作隐藏，来简化程序的结构，使其容易阅读。函数在调用前，必须先定义。也可以在一个函数内部定义函数，内部函数只有在外部函数调用时才能够被执行。程序调用函数时，转到函数内部执行函数内部的语句，函数执行完毕后，返回到它离开程序的地方，执行程序的下一条语句。

5：Python 是如何进行内存管理的？

Python 的内存管理是由 Python 得解释器负责的，开发人员可以从内存管理事务中解放出来，致力于应用程序的开发，这样就使得开发的程序错误更少，程序更健壮，开发周期更短

6：如何反序的迭代一个序列？howdo I iterate over a sequence in reverse order

如果是一个 list，最快的解决方案是：

list.reverse()

try:

for x in list:

“do something with x”

finally:

list.reverse()

如果不是 list, 最通用但是稍慢的解决方案是：

for i in range(len(sequence)-1, -1, -1):

x = sequence[i]

<do something with x>

7：Python 里面如何实现 tuple 和 list 的转换？

函数 tuple(seq)可以把所有可迭代的(iterable)序列转换成一个 tuple, 元素不变，排序也不变。例 如，tuple([1,2,3])返回(1,2,3),tuple(’abc’)返回(’a’.’b','c').如果参数已经是一个 tuple 的话，函 数不做任何拷贝而直接返回原来的对象，所以在不确定对象是不是 tuple 的时候来调用 tuple()函数也不是很耗费的。

函数 list(seq)可以把所有的序列和可迭代的对象转换成一个 list,元素不变，排序也不变。例如 list([1,2,3])返回(1,2,3),list(’abc’)返回['a', 'b', 'c']。如果参数是一个 list,她会像 set[:]一样做一个拷贝

8：Python 面试题：请写出一段 Python 代码实现删除一个 list 里面的重复元素

可以先把 list 重新排序，然后从 list 的最后开始扫描，代码如下：

if List:

List.sort()

last = List[-1]

for i in range(len(List)-2, -1, -1):

if last==List[i]: del List[i]

else: last=List[i]

9：Python 文件操作的面试题

1. 如何用 Python 删除一个文件？

使用 os.remove(filename)或者 os.unlink(filename);

2. Python 如何 copy 一个文件？

shutil 模块有一个 copyfile 函数可以实现文件拷贝

10：Python 里面如何生成随机数？

标准库 random 实现了一个随机数生成器，实例代码如下：

import random

random.random()

它会返回一个随机的 0 和 1 之间的浮点数

11：如何用 Python 来发送邮件？

可以使用 smtplib 标准库。

以下代码可以在支持 SMTP 监听器的服务器上执行。

```
import sys, smtplib
fromaddr = raw_input(”From: “)
toaddrs = raw_input(”To: “).split(’,')
print “Enter message, end with ^D:”
msg = ”
while 1:
line = sys.stdin.readline()
if not line:
break
msg = msg + line
# 发送邮件部分
server = smtplib.SMTP(’localhost’)
server.sendmail(fromaddr, toaddrs, msg)
server.quit()
```

12：Python 里面如何拷贝一个对象？

一般来说可以使用 copy.copy()方法或者 copy.deepcopy()方法，几乎所有的对象都可以被拷贝

一些对象可以更容易的拷贝，Dictionaries 有一个 copy 方法：

```
newdict = olddict.copy()
```

13：有没有一个工具可以帮助查找 python 的 bug 和进行静态的代码分析？

有，PyChecker 是一个 python 代码的静态分析工具，它可以帮助查找 python 代码的 bug，会对代码的复杂度和格式提出警告

Pylint 是另外一个工具可以进行 coding standard 检查。

14：如何在一个 function 里面设置一个全局的变量？

解决方法是在 function 的开始插入一个 global 声明：

```
def f()
global x
```

14：有两个序列 a,b，大小都为 n,序列元素的值任意整形数，无序；要求：通过交换 a,b 中的元素，使[序列 a 元素的和]与[序列 b 元素的和]之间的差最小。

1. 将两序列合并为一个序列，并排序，为序列 Source
2. 拿出最大元素 Big，次大的元素 Small
3. 在余下的序列 S[:-2]进行平分，得到序列 max，min

4. 将 Small 加到 max 序列，将 Big 加大 min 序列，重新计算新序列和，和大的为 max，小的为 min。

Python 代码

```
def mean( sorted_list ):
if not sorted_list:
return (([],[]))
big = sorted_list[-1]
small = sorted_list[-2]
big_list, small_list = mean(sorted_list[:-2])
big_list.append(small)
small_list.append(big)
big_list_sum = sum(big_list)
small_list_sum = sum(small_list)
if big_list_sum > small_list_sum:
return ( (big_list, small_list))
else:
return (( small_list, big_list))
tests = [ [1,2,3,4,5,6,700,800],
[10001,10000,100,90,50,1],
range(1, 11),
[12312, 12311, 232, 210, 30, 29, 3, 2, 1, 1]
]
for l in tests:
l.sort()
print
print “Source List:\t”, l
l1,l2 = mean(l)
print “Result List:\t”, l1, l2
print “Distance:\t”, abs(sum(l1)-sum(l2))
print ‘-*’*40
```

输出结果

Python 代码

```
Source List: [1, 2, 3, 4, 5, 6, 700, 800]
Result List: [1, 4, 5, 800] [2, 3, 6, 700]
Distance: 99
```

-*

Source List: [1, 50, 90, 100, 10000, 10001]

Result List: [50, 90, 10000] [1, 100, 10001]

Distance: 38

-*

Source List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Result List: [2, 3, 6, 7, 10] [1, 4, 5, 8, 9]

Distance: 1

-*

Source List: [1, 1, 2, 3, 29, 30, 210, 232, 12311, 12312]

Result List: [1, 3, 29, 232, 12311] [1, 2, 30, 210, 12312]

Distance: 21

-*

15：用 Python 匹配 HTML tag 的时候，<.*>和<.*?>有什么区别？

当重复匹配一个正则表达式时候， 例如<.*>，当程序执行匹配的时候，会返回最大的匹配值

例如：

import re

s = ‘<html><head><title>Title</title>’

print(re.match(’<.*>’, s).group())

会返回一个匹配<html><head><title>Title</title>而不是<html>

而

import re

s = ‘<html><head><title>Title</title>’

print(re.match(’<.*?>’, s).group())

则会返回<html>

<.*>这种匹配称作贪心匹配 <.*?>称作非贪心匹配

16：Python 里面 search()和 match()的区别？

match（）函数只检测 RE 是不是在 string 的开始位置匹配， search()会扫描整个 string 查找匹配，也就是说 match（）只有在 0 位置匹配成功的话才有返回，如果不是开始位置匹配成功的话，match()就返回 none

例如：

print(re.match(’super’, ’superstition’).span())会返回(0,5)

而 print(re.match(’super’, ‘insuperable’))则返回 None

search()会扫描整个字符串并返回第一个成功的匹配

例如：print(re.search(’super’, ’superstition’).span())返回(0, 5)

print(re.search(’super’, ‘insuperable’).span())返回(2,7)

17：如何用 Python 来进行查询和替换一个文本字符串？

可以使用 sub()方法来进行查询和替换，sub 方法的格式为：sub(replacement, string[, count=0])

replacement 是被替换成的文本

string 是需要被替换的文本

count 是一个可选参数，指最大被替换的数量

例子：

```
import re
p = re.compile(’(blue|white|red)’)
print(p.sub(’colour’,'blue socks and red shoes’))
print(p.sub(’colour’,'blue socks and red shoes’, count=1))
```

输出：

```
colour socks and colour shoes
colour socks and red shoes
```

subn()方法执行的效果跟 sub()一样，不过它会返回一个二维数组，包括替换后的新的字符串和总共替换的数量

例如：

```
import re
p = re.compile(’(blue|white|red)’)
print(p.subn(’colour’,'blue socks and red shoes’))
print(p.subn(’colour’,'blue socks and red shoes’, count=1))
```

输出

```
(’colour socks and colour shoes’, 2)
(’colour socks and red shoes’, 1)
```

18：介绍一下 except 的用法和作用？

Python 的 except 用来捕获所有异常， 因为 Python 里面的每次错误都会抛出 一个异常，所以每个程序的错误都被当作一个运行时错误。

一下是使用 except 的一个例子：

```
try:
foo = opne(”file”) #open 被错写为 opne
except:
sys.exit(”could not open file!”)
```

因为这个错误是由于 open 被拼写成 opne 而造成的，然后被 except 捕获，所以 debug 程序的时候很容易不知道出了什么问题

下面这个例子更好点：

```
try:
foo = opne(”file”) # 这时候 except 只捕获 IOError
except IOError:
sys.exit(”could not open file”)
```

19：Python 中 pass 语句的作用是什么？

pass 语句什么也不做，一般作为占位符或者创建占位程序，pass 语句不会执行任何操作，比如：

```
while False:
pass
```

pass 通常用来创建一个最简单的类：

```
class MyEmptyClass:
pass
```

pass 在软件设计阶段也经常用来作为 TODO，提醒实现相应的实现，比如：

```
def initlog(*args):
pass #please implement this
```

20：介绍一下 Python 下 range()函数的用法？

如果需要迭代一个数字序列的话，可以使用 range()函数，range()函数可以生成等差级数。如例：

```
for i in range(5)
print(i)
```

这段代码将输出 0, 1, 2, 3, 4 五个数字

range(10)会产生 10 个值， 也可以让 range()从另外一个数字开始，或者定义一个不同的增量，甚至是负数增量

range(5, 10)从 5 到 9 的五个数字

range(0, 10, 3) 增量为三， 包括 0,3,6,9 四个数字

range(-10, -100, -30) 增量为-30， 包括-10, -40, -70

可以一起使用 range()和 len()来迭代一个索引序列

例如：

```
a = ['Nina', 'Jim', 'Rainman', 'Hello']
for i in range(len(a)):
    print(i, a[i])
```