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:
```

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