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？how do 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])
```