作业要求：

1. 从ftp上下载当天作业(ftp://172.26.184.101)

2. 每一小题新建一个文件（简答题保存为txt形式，程序题保存为py形式)

3. 每个文件名保存为以下格式：PartA的第1题保存为A-1.py、PartB的第3题保存为B-3.py等

4. 提交要求：按照一个Part的题目打包为rar格式提交，打包文件名为：学号+姓名+PartA(/B/C/D/E/F).rar

5. 提交地址，上传到上述ftp的upload目录中

Part A　Python 基础

一、环境准备

1. 在 Windows 下配置 Python，使其在 Windows 命令行中能够使用。
2. 找到 Python 的安装位置及其标准库模块所在位置。
3. 在 IDLE 中打开 String.py 了解基本的 Python 语言和结构，适应阅读 Python 脚本。
4. 了解 sys 模块，输出所有 sys 模块的属性。显示 sys 模块的版本号和平台变量。

二、编程

1. 用两种方式输出 Hello World!,一为交互执行方式“print 'Hello World!'”,二为创建'Hello World!'脚本并运行。
2. 三个元素 a、b、c 排序，不用现成的排序函数，对输入的 a、b、c 这 3 个数按从小到大的顺序输出。
3. 编写程序求两个整型的最大公约数和最小公倍数。

Part B 序列和字典

1. 判断输入的一个字符串是否是回文。要求支持中文。

2. 排序的简单练习：随机输入 10 个数字，按照数字从小到大的顺序进行排列。如果把这 10 个数字看成是字符类型，按照字符的字典序进行排序，则结果与按大小排序的结果是不一样的。尝试一下 Python 实现排序和 C、JAVA 有什么不同？

3. 学期结束时，需要对学生成绩进行简单整理。从键盘输入本班若干学生的成绩，当成绩为“0”时结束输入。然后根据学分绩（A-F）的评分标准：

   A: 90–100

   B: 80–89

   C: 70–79

   D: 60–69

   F: <60

   得到每个成绩的学分绩，并和成绩配对输出。

   最后再输出所有成绩的平均成绩。

4. a) 阿拉伯数字与英文数字的翻译，给出一个整数值，返回代表该值的英文，比如输入 89 返回"eight-nine"。限定值的范围在 0 到 1,000。

   b) 如果输入 89 时，想要返回”eighty-nine”。代码应该如何写？

5. 列表与字典的转换：给定两个长度相同的列表，比如说，列表[1, 2, 3,...]和['abc', 'def', 'ghi',...]，用这两个列表里的所有数据组成一个字典，像这样：{1:'abc', 2: 'def', 3: 'ghi',...}。至少用 3 种方法实现。

6. 创建一个简单的人事管理程序。让用户输入一组雇员姓名和编号。你的程序可以提供按照姓名排序输出的功能，雇员姓名显示在前面，后面是对应的雇员编号。也可以提供按照雇员编号的顺序输出数据。（提示：可以使用 sorted 函数）

7. (a) 编写一个"rot13"翻译器。"rot13"是一个古老而又简单的加密方法，它把字母表中的每个字母用其后的第 13 个字母来代替。字母表中前半部分字母将被映射到后半部分，而后半部分字母将被映射到前半部分，大小写保持不变。举例来说，'a'将被替换为'n','X'将被替换为'K'; 数字和符号不进行翻译。

   (b) 在你的解决方案的基础上加一个应用，让它提示用户输入准备加密的字符串(这个算法同时也可以对加密后的字符串进行解密)。

   例如：不明觉厉

```
Enter string to rot13:I don't quite get it,but I think you are really terrific.
The rot13 string is:
V qba'g dhvgr trg vg,ohg V guvax lbh ner ernyyl greevsvp.
```

```
Enter string to rot13:V qba'g dhvgr trg vg,ohg V guvax lbh ner ernyyl greevsvp.
The rot13 string is:
I don't quite get it,but I think you are really terrific.
```

Part C 语句

1. 输入一元二次方程 $ax^2+bx+c=0$ 的三个系数 a、b、c（均为整数），求此方程的根，显示两位小数。若方程的根是实数，则将其保存到两个 float 变量中；若方程的根是复数，则将其保存到两个 complex 变量中。

例如，a=1,b=2,c=1 时输出 x1=-1.00, x2=-1.00

a=1,b=3,c=1 时输出 x1=-0.38, x2=-.62

a=2,b=3,c=2 时输出 x1=-0.75+0.66i,x2=-0.75+-0.66i

2. 要求将整数 m～n 范围之间每个整数的十进制表示、二进制表示、八进制表示和十六进制表示均显示出来。输入 m、n 的值。

例如，输入起始值 9，结束值 18 时，输出应为如下形式：

DEC	BIN	OCT	HEX
9	01001	11	9
10	01010	12	10
11	01011	13	11
12	01100	14	12
13	01101	15	13
14	01110	16	14
15	01111	17	15
16	10000	20	10
17	10001	21	11
18	10010	22	12

3. 输入一个起始整数 m 和一个结束整数 n，输出 m ~ n 范围内的所有满足以下条件的整数：（1）是回文数（2）是完全平方数

例如，当 m=10000,n=99999 时，输出 [10201, 12321, 14641, 40804, 44944, 69696, 94249]

4. 用牛顿迭代法求解方程 f(x)=sinx-cosx-x，f(x)=0 在-1 附近的近似实根,误差小于 0.00000001。输出 -1.258728。

提示：牛顿迭代法，给出 x_0，然后 $x_n=x_{n-1}-f(n-1)/f'(n-1)$,n=1、2、3……，当$|x_n-x_{n-1}|<\varepsilon$或$|f(x)|<\varepsilon$时，输出 x 的值。正弦函数 sin(x)，余弦函数 cos(x)，绝对值函数 fabs(x)。

5. 输出一个整数的所有素数因子列表。例如，输入 20 时输出 [2,2,5]。

6. 验证命题：任何一个数字不全相位的 4 整数,经过不超过 7 次的“重排求差”操作后,总会得到 6174，称 6174 为 4 位黑洞数。

"重排求差"操作是指用组成该数的数字重排后得到的最大数减去最小数。

例如，对 2012 重排之后得到最大数是 2210，最小数是 0122，2210-0122=2088；对 2088 重排之后得到最大数是 8820，最小数是 0288，8820-0288=8532；对 8532 重排之后得到最大数是 8532，最小数是 2358，8532-2358=6174；对 6174 重排之后得到最大数是 7641，最小数是 1467，7641-1467=6174。

7. 输入一个英文句子，输出其中包含的单词个数、字符个数及元音字母个数。

例如，输入**"Do you like python?"**，输出 **words:4, characters: 19, vowels: 6**

8. 删除一个人名列表中重复出现的名字后，对列表中剩余的名字按字典序排序后输出。

例如，对于包含人名 **Tom, Mary , John ,Mike , Julia ,Mary ,Tom , Mary** 的列表做上述处理后输出 **[John,Julia,Mary,Mike,Tom]**。如果要求将所有重复的都删除，比如上例，输出 **[John,Julia, Mike]**，则程序应该怎么写？

Part D 文件

1. 编写文件复制程序

提示并接收用户输入的两个文件名，第一个文件名代表源文件，第二个文件名代表目标文件，将源文件的内容复制到目标文件并显示被复制的文件内容。

2. 编写文件过滤程序

提示并接收用户输入的一个 **python** 脚本程序文件名，删除程序中的所有注释内容（以#开始到行尾的字符串）后再写回到原文件中。

3. 编写计算器程序

接收包含一个算术运算算式的命令行参数（不检查算式正确性，输入时确保算式正确），完成算术算式运算，输出运算结果，最后将算式及运算结果保存到一个文件 **res.txt** 中。

提示：本题需要在命令提示符下执行：**python p8-4.py 12+23**，则输出到 **res.txt** 中的内容为**"12+23=35"**。

思考：如果想要连续+，连续-，以及四则运算呢，程序将如何修改？

Part E 函数

1.安装 PyPI，利用 pip 安装任意一个包并列举此包所包含的若干函数.

2.求前 n 个 Fibonacci 数。

3.实现 max()和 min()内建函数。

写分别带两个元素返回一个较大和较小元素，简单的 max2()核 min2()函数。他们应该可以用任意的 python 对象运作。举例来说，max2(4,8)和 min2(4,8)会各自每次返回 8 和 4。

4. 实现 printf()的函数。有一个值参数和格式字符串，根据格式化字符串显示可变参数，格式化字符串中的值允许特别的字符串格式操作指示符，如%d, %f 等。提示：无需实现字符串操作符功能性。

5. 使用 reduce()进行函数式编程以及递归。

(a) 用一分钟写一个带 x,y 并返回他们乘积的名为 mult(x,y)的简单小巧的函数。

(b) 用你在 a 中创建的 mult()函数以及 reduce 来计算阶乘。

(c) 彻底抛弃掉 mult()的使用，用 lamda 表达式替代。

(d) 我们描绘了一个递归解决方案来找到 N!

8.首先产生一个随机数序列，然后过滤（利用 filter()函数）掉所有的偶数

Part F 面向对象

1、创建一个由有序数值对(x, y) 组成的 Point 类，它代表某个点的 X 坐标和 Y 坐标。
X 坐标和 Y 坐标在实例化时被传递给构造器，如果没有给出它们的值，则默认为坐标的原点。

2、创建一个直线/直线段类。除主要的数据属性：一对坐标值外，它还具有长度和斜线的方法属性。
你需要覆盖__repr__()方法(如果需要的话，还有__str__()方法)，使得代表那条直线(或直线段)的字符串表示形式是由一对元组构成的元组，即，((x1, y1), (x2, y2)).

__repr__ 将直线的两个端点(始点和止点)显示成一对元组
length 返回直线段的长度
slope 返回此直线段的斜率(或在适当的时候返回 None)

3、对类进行定制。写一个类，名为 MoneyFmt，用来将浮点数值转换为金额。
MoneyFmt 类里只有一个数据值(即，金额)，和五个方法。

__init__()构造器对数据进行初始化，
update()方法把数据值替换成一个新值，
__nonzero__()是布尔型的，当数据值非零时返回 True，
__repr__()方法以浮点数的形式返回金额
__str__()方法以字符格式显示金额，金额数里应该有逗号(比如 1,000,000)和美元的货币符号，如果有负号，它必须出现在美元符号的左边。1234567.8901 ==> “$1,234,567.89”

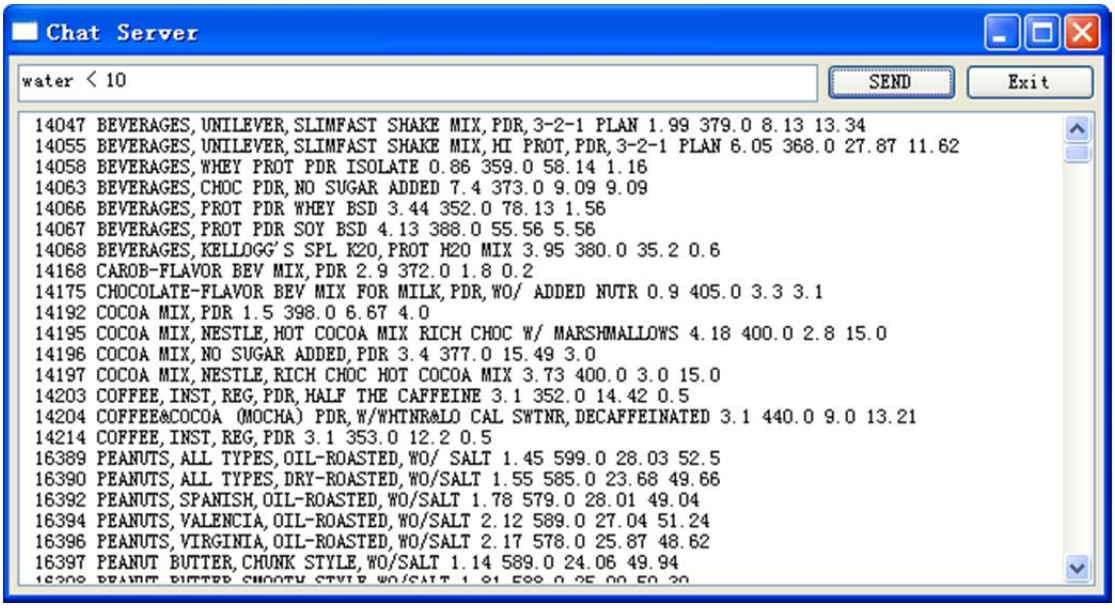
Python 应用练习

1. 简单命令行数据库查询
   （1） 在 ABBREV.txt 中存储着食物营养属性数据，使用 importdata.py 将文本文件中的数据导入到 sqlite3 数据库 food.db 中。
   （2） 编写 food_query.py，从命令行给出查询条件，然后从数据库 food.db 中提取满足条件的数据记录并显示在屏幕上，效果如下图所示：

```
E:\python\python_courses\coursefiles>c:\Python27\python.exe food_query.py "water < 10 "
```

2. 网络版数据库查询
   分别编写 cmdqueryclient.py 和 queryserver.py，实现网络查询功能。
   （1） queryserver.py 实现一个网络服务器，不停等待客户端提交的查询条件，当收到查询条件时，从 food.db 中提取满足条件的记录，一条一条传送给客户端；
   （2） cmdqueryclient.py 实现从命令行给出查询条件，然后由网络客户端将查询条件提交给服务器，并接收服务器传来的数据记录，一个记录一条的记录到数据文件 res.txt 中。

3. 图形界面版网络数据库查询
   分别编写 guiqueryclient.py 和 queryserver.py，实现网络查询功能。
   （1） queryserver.py 实现一个网络服务器，不停等待客户端提交的查询条件，当收到查询条件时，从 food.db 中提取满足条件的记录，一条一条传送给客户端；
   （2） guiqueryclient.py 实现从图形界面给出查询条件，然后由网络客户端将查询条件提交给服务器，并接收服务器传来的数据记录，一个记录一条记录的显示在图形界面的 LISTBOX 中，如下图所示。



Answers to Part A

一、环境准备

1. 在 Windows 下配置 Python，使其在 Windows 命令行中能够使用。

在安装有 Python 2.7.6 的 Windows 机器上，点击控制面板->系统->高级->环境变量。在“系统变量”表单中点击叫做 PATH 的变量，然后编辑这个变量，把;C:\Python27 加到它的结尾。

2. 找到 Python 的安装位置及其标准库模块所在位置。

Python 安装位置 C:\Python27；Python 标准库模块所在位置 C:\Python27\Lib

3. 在 IDLE 中打开 String.py 了解基本的 Python 语言和结构，适应阅读 Python 脚本。

String.py 文件在 C:\Python27\Lib 下，右击，选择 Edit with IDLE，打开阅读 String.py。了解 docstrings，注释，变量，函数，字符串，缩进，模块等基本内容。

4. 了解 sys 模块，输出所有 sys 模块的属性。显示 sys 模块的版本号和平台变量。

```
import sys
help(sys)
dir(sys)
sys.version
sys.platform
```

二、编程

1. 用两种方式输出 Hello World!,一为交互执行方式“print 'Hello World!'”,二为创建'Hello World!'脚本并运行。

a.打开 IDLE，输入“print 'Hello World!'”并回车运行。

b.打开 IDLE，选择 File->New File，在打开的窗口中输入“print 'Hello World!'”，并保存为 Helloworld.py 文件，选择 Run->Run Module（或按 F5 键）查看运行结果。

2. 三个元素 a、b、c 排序，不用现成的排序函数，对输入的 a、b、c 这 3 个数按从小到大的顺序输出。

```
a = raw_input('please input three integers, the first:')
b = raw_input('please input three integers, the second:')
c = raw_input('please input three integers, the third:')
if a>b:
    a,b = b,a
if b>c:
    b,c = c,b
if a>b:
    a,b = b,a
print a,b,c
```

3. 编写程序求两个整型的最大公约数和最小公倍数。

```
def gcd1(a,b):
```

```
    c = a % b
    while c!=0:
        a = b
        c = a % b
    return b

def gcd2(a,b):
    if a % b == 0:
        return b
    else:
        return gcd2(b,a % b)

a = int(raw_input('Please input a integer:'))
b = int(raw_input('Please input another integer:'))
print gcd2(a,b),a*b/gcd2(a,b)
```

Answers to Part B

1. 判断输入的一个字符串是否是回文。要求支持中文。

```
while True:
    s = raw_input('Please input a string:')
    s = s.decode('gbk')                                    #支持中文
    if s != '' :
        if s[::]==s[::-1]:
            print 'Yes! The input string is palindromic'
        else:
            print 'No! The input string is not palindrome'
    else:
        break
```

输出：

```
Please input a string:abcba
Yes! The input string is palindromic
Please input a string:we are happy
No! The input string is not palindrome
Please input a string:
```

2. 排序的简单练习：随机输入 10 个数字，按照数字从小到大的顺序进行排列。如果把这 10 个数字看成是字符类型，按照字符的字典序进行排序，则结果与按大小排序的结果是不一样的。尝试一下 Python 实现排序和 C、JAVA 有什么不同？

```
import random
usort = list()
snsort = list()
i=0
while i<10:
    x = random.randint(10,200)
    if x not in usort:
        usort.append(x)
        snsort.append(str(x))
        i += 1
print snsort
print 'The random 10 numbers are:'
print usort
usort.sort()
print '10 numbers are sorted by digit:'
print usort
snsort.sort()
for i in range(len(snsort)):
    snsort[i] = int(snsort[i])
print '10 numbers are sorted by string:'
print snsort
```

```
a=raw_input('')
```

输出：

```
['170', '50', '19', '43', '18', '64', '126', '39', '80', '122']
The random 10 numbers are:
[170, 50, 19, 43, 18, 64, 126, 39, 80, 122]
10 numbers are sorted by digit:
[18, 19, 39, 43, 50, 64, 80, 122, 126, 170]
10 numbers are sorted by string:
[122, 126, 170, 18, 19, 39, 43, 50, 64, 80]
```

参考答案是使用生成随机数得到10个数字，如果不熟悉random模块，输入10个数字也可以。

字符串的排序即是按照字典序排序，数字直接排序则是按照数字大小排。

3. 学期结束时，需要对学生成绩进行简单整理。从键盘输入本班若干学生的成绩，当成绩为“0”时结束输入。然后根据学分绩（A-F）的评分标准：

A: 90–100

B: 80–89

C: 70–79

D: 60–69

F: <60

得到每个成绩的学分绩，并和成绩配对输出。

最后再输出所有成绩的平均成绩。

```
scrList = []
while True:
    a = raw_input('Please input the score(endwith "0"):')
    if a != "0":
        ins1 = int(a)
        scrList.append(ins1)
    else:
        break
scr2List=[]
for eachscr in scrList:
    if 90 <= eachscr <= 100:
        scr2List.append('A')
    elif 80 <= eachscr <= 89:
        scr2List.append('B')
    elif 70 <= eachscr <= 79:
        scr2List.append('C')
    elif 60 <= eachscr <= 69:
        scr2List.append('D')
    else:
        scr2List.append('F')
print "All the students' score list:"
for i , j in enumerate(zip(scrList,scr2List)):
    print i,j
```

```
print "The average score is:"
print sum(scrList)/len(scrList)
a=raw_input('')
```

输出：

```
Please input the score(endwith "0"):90
Please input the score(endwith "0"):52
Please input the score(endwith "0"):85
Please input the score(endwith "0"):0
All the students' score list:
0 (90, 'A')
1 (52, 'F')
2 (85, 'B')
The average score is:
75
```

4. a) 阿拉伯数字与英文数字的翻译，给出一个整数值，返回代表该值的英文，比如输入 89 返回"eight-nine"。限定值的范围在 0 到 1,000。

b) 如果输入 89 时，想要返回"eighty-nine"。代码应该如何写？

a)

```
englist = ['zero','one','two','three','four','five','six','seven','eight','nine','ten']
while True:
    a = int(raw_input('Please input a number between 0-1000:'))
    if 0 <= a <=1000:
        enga=[]
        for dig in str(a):
            enga.append(englist[int(dig)])
        print '-'.join(enga)
    else:
        break
```

输出：

```
Please input a number between 0-1000:100
one-zero-zero
Please input a number between 0-1000:951
nine-five-one
Please input a number between 0-1000:35
three-five
Please input a number between 0-1000:604
six-zero-four
```

b)

```
engsingle = ['zero','one','two','three','four','five','six','seven','eight','nine']
engteen =
['ten','eleven','twelve','thirteen','fourteen','fifteen','sixteen','seventeen','eightee
n','nineteen']
engtens = ['twenty','thirty','forty','fifty','sixty','seventy','eighty','ninety']

while True:
```

```
a = int(raw_input('Please input a number between 0-1000:'))
sa = str(a)
if 0 <= a <=1000:
    n=len(sa)
    enga=[]
    if n == 1:
        enga.append(str(engsingle[a]))
        print ''.join(enga)
    elif n == 2:
        if sa[0] == '1':
            enga.append(str(engteen[int(sa[1])]))
            print ''.join(enga)
        else:
            enga.append(engtens[int(sa[0])-2])
            enga.append(engsingle[int(sa[1])])
            print '-'.join(enga)
    elif n == 3:
        enga.append(str(engsingle[int(sa[0])]))
        enga.append(engtens[int(sa[1])-2])
        enga.append(engsingle[int(sa[2])])
        print enga[0] + ' hundred' + ' and ' + '-'.join(enga[1:])
else:
    break
```

输出：

```
Please input a number between 0-1000:5
five
Please input a number between 0-1000:14
fourteen
Please input a number between 0-1000:52
fifty-two
Please input a number between 0-1000:123
one hundred and twenty-three
Please input a number between 0-1000:954
nine hundred and fifty-four
```

5. 列表与字典的转换：给定两个长度相同的列表，比如说，列表[1, 2, 3,...]和['abc', 'def', 'ghi',...]，用这两个列表里的所有数据组成一个字典，像这样：{1:'abc', 2: 'def', 3: 'ghi',...}。至少用 3 种方法实现。

第一种：

```
>>> d1=[1,2,3]
>>> d2=['a','b','c']
>>> d3={}
>>> d3=dict(zip(d1,d2))
>>> d3
{1: 'a', 2: 'b', 3: 'c'}
```

第二种：

```
>>> d1
[1, 2, 3]
>>> d2
['a', 'b', 'c']
>>> d3={}
>>> for i in range(len(d1)):
        d3.setdefault(d1[i],d2[i])
>>> d3
{1: 'a', 2: 'b', 3: 'c'}
```

第三种：

```
>>> d3={}
>>> for i,x in enumerate(d1):
        d3[x]=d2[i]
>>> d3
{1: 'a', 2: 'b', 3: 'c'}
```

第四种：

```
>>> d3={}
>>> for i in range(len(d1)):
        d3[d1[i]]=d2[i]
>>> d3
{1: 'a', 2: 'b', 3: 'c'}
```

第五种：

```
>>> d3=dict(map(None,d1,d2))
>>> d3
{1: 'a', 2: 'b', 3: 'c'}
```

#map 函数的使用：

```
>>> def lianjie(s1,s2):
        return tuple([s1,s2])

>>> map(lianjie,l1,l2)
[(1, 'a'), (2, 'b'), (3, 'c')]
```

6. 创建一个简单的人事管理程序。让用户输入一组雇员姓名和编号。你的程序可以提供按照姓名排序输出的功能，雇员姓名显示在前面，后面是对应的雇员编号。也可以提供按照雇员编号的顺序输出数据。

```
d1={}
while True:
    employee = raw_input('请输入雇员名（输入q退出）:')
    if employee == 'q':
        break
    employee.decode('gbk')
    while True:
```

```
        uid = int(raw_input('请输入雇员编号::'))
        if uid in d1:
            print '编号已经存在'
        else:
            break
    d1[uid]=employee
while True:
    order = raw_input('你想按名字(N)还是按编号(I)排序(N/I/Q退出):').lower()[0]
    if order not in 'niq':
        print '输入的指令错误!'
    elif order == 'q':
        break
    elif order == 'n':
        for k,v in sorted(d1.items(),key=lambda x:x[1]):
            print k,v
    elif order == 'i':
        for k,v in sorted(d1.items()):
            print k,v
```

输出：

```
请输入雇员名（输入q退出）:张三
请输入雇员编号:023
请输入雇员名（输入q退出）:李四
请输入雇员编号:094
请输入雇员名（输入q退出）:王五
请输入雇员编号:003
请输入雇员名（输入q退出）:q
你想按名字(N)还是按编号(I)排序(N/I/Q退出):n
94 李四
3 王五
23 张三
你想按名字(N)还是按编号(I)排序(N/I/Q退出):i
3 王五
23 张三
94 李四
你想按名字(N)还是按编号(I)排序(N/I/Q退出):
```

注：sorted 函数的用法：

```
>>> d
{8: 'k', 2: 'm', 5: 'a'}
>>> k=lambda x:x[0]
>>> sorted(d.items(),key=k)
[(2, 'm'), (5, 'a'), (8, 'k')]
>>> k=lambda x:x[1]
>>> sorted(d.items(),key=k)
[(5, 'a'), (8, 'k'), (2, 'm')]
```

7. (a) 编写一个"rot13"翻译器。"rot13"是一个古老而又简单的加密方法，它把字母表中的每个字母用其后的第 **13** 个字母来代替。字母表中前半部分字母将被映射到后半部分，而后半部分字母将被映射到前半部分，大小写保持不变。举例来说，'a'将被替换为'n','X'将被替换为'K'；数字和符号不进行翻译。

(b) 在你的解决方案的基础上加一个应用，让它提示用户输入准备加密的字符串(这个算法同时也可以对加密后的字符串进行解密)，如下所示:

```
def rot13(clear):
    news=''
    for i in clear:
        if 'A'<=i<='M' or 'a'<=i<='m':
            i=chr(ord(i)+13)
        elif 'N'<=i<='Z' or 'n'<=i<='z':
            i=chr(ord(i)-13)
        news = news + chr(ord(i))
    return news

s = raw_input('Enter string to rot13:')
print 'The rot13 string is:'
ens = rot13(s)
print ens
```

```
Enter string to rot13:I don't quite get it,but I think you are really terrific.
The rot13 string is:
V qba'g dhvgr trg vg,ohg V guvax lbh ner ernyyl greevsvp.
```

```
Enter string to rot13:V qba'g dhvgr trg vg,ohg V guvax lbh ner ernyyl greevsvp.
The rot13 string is:
I don't quite get it,but I think you are really terrific.
```

Answers to Part C

1. 输入一元二次方程 $ax^2+bx+c=0$ 的三个系数 a、b、c（均为整数），求此方程的根，显示两位小数。若方程的根是实数，则将其保存到两个 float 变量中；若方程的根是复数，则将其保存到两个 complex 变量中。

例如，a=1,b=2,c=1 时输出 x1=-1.00, x2=-1.00

a=1,b=3,c=1 时输出 x1=-0.38, x2=-.62

a=2,b=3,c=2 时输出 x1=-0.75+0.66i,x2=-0.75+-0.66i

```
from math import sqrt
a=int(raw_input('enter a:'))
b=int(raw_input('enter b:'))
c=int(raw_input('enter c:'))
d=b*b-4*a*c
if d>0:
   x1=(-b+sqrt(d))/(2*a)
   x2=(-b-sqrt(d))/(2*a)
   print 'x1=%.2f,x2=%.2f' % (x1,x2)
elif d<0:
   x1=complex(float(-b)/(2*a),sqrt(-d)/(2*a))
   x2=complex(float(-b)/(2*a),-sqrt(-d)/(2*a))
   print 'x1=%.2f+%.2fi,x2=%.2f+%.2fi' % (x1.real,x1.imag,x2.real,x2.imag)
else:
   x1=x2=float(-b)/(2*a)
   print 'x1=%.2f,x2=%.2f' % (x1,x2)
a = raw_input('press any key ...')
```

输出：

```
enter a:1
enter b:2
enter c:1
x1=-1.00,x2=-1.00
press any key ...
```

```
enter a:1
enter b:3
enter c:1
x1=-0.38,x2=-2.62
press any key ...
```

```
enter a:2
enter b:3
enter c:2
x1=-0.75+0.66i,x2=-0.75+-0.66i
press any key ...
```

2. 要求将整数 m～n 范围之间每个整数的十进制表示、二进制表示、八进制表示和十六进制表示均显示出来。输入 m、n 的值。

例如，输入起始值 9，结束值 18 时，输出应为如下形式：

DEC	BIN	OCT	HEX
9	01001	11	9
10	01010	12	10
11	01011	13	11
12	01100	14	12
13	01101	15	13
14	01110	16	14
15	01111	17	15
16	10000	20	10

17　　10001　　21　　11

18　　10010　　22　　12

方法一：二进制使用 bin()函数获得

```
m=input('m=')
n=input('n=')
print "%6s%10s%5s%5s" % ('DEC','BIN','OCT','HEX')
print "-"*35
for i in range(m,n+1):
    print "%6d%10s%5o%5x" % (i,bin(i)[2:],i,i)
a=raw_input('Press Enter to quit……')
```

方法二：二进制用基本概念的短除法获得

```
import math
m=int(raw_input('m='))
n=int(raw_input('n='))
x=n
c=0
while x>0:
   c+=1
   x/=2
print "DEC\tBIN\t\tOCT\tHEX"
print "-"*35
for i in range(m,n+1):
    print "%d\t" % i,
    x=i
    mode=2**(c-1)
    for j in range(1,c+1):
         print x/mode,
         x=x%mode
         mode/=2
    print "\t%o\t%x" % (i,i)
```

3. 输入一个起始整数 m 和一个结束整数 n，输出 m ~ n 范围内的所有满足以下条件的整数：

（1）是回文数（2）是完全平方数

例如，当 m=10000,n=99999 时，输出 [10201, 12321, 14641, 40804, 44944, 69696, 94249]

方法一：

```
from math import sqrt
m=int(raw_input('m='))
n=int(raw_input('n='))
z=[]
x=m
while x<=n:
    y=0
    k=x
    while k>0:
```

```
        y=y*10+k%10
        k=k/10
    root=sqrt(x)
    if x==y and root==int(root):
      z.append(x)
    x+=1
print z
a = raw_input('press Enter to quit ...')
```

方法二： #换成字符串判断回文

```
for i in range(m,n+1):
    if str(i)[::] == str(i)[::-1] and sqrt(i)==int(sqrt(i)): #math.sqrt()
        print str(i) + " is ok"
'''
```

输出：

```
m=10000
n=19999
[10201, 12321, 14641]
press Enter to quit ...
```

4. 用牛顿迭代法求解方程 f(x)=sinx-cosx-x，f(x)=0 在-1 附近的近似实根,误差小于 0.00000001。输出 -1.258728。

提示：牛顿迭代法，给出 x_0，然后 $x_n=x_{n-1}-f(n-1)/f'(n-1)$,n=1、2、3……，当$|x_n-x_{n-1}|<\varepsilon$或$|f(x)|<\varepsilon$时，输出 x 的值。正弦函数 sin(x)，余弦函数 cos(x)，绝对值函数 fabs(x)。

```
from math import sin,cos,fabs
def f(x):
    return sin(x)-cos(x)-x
def f1(x):
    return cos(x)+sin(x)-1
x0 = -1
while True:
  x=x0-f(x0)/f1(x0)
  if fabs(x-x0) < 1e-8 or fabs(f(x))< 1e-8:
      break
  x0=x
print "x=%f" % x
a = raw_input('Press Enter to quit……-……-')
```

输出：

```
x=-1.258728
Press Enter to quit……
```

5. 输出一个整数的所有素数因子列表。例如，输入 20 时输出 [2,2,5]。

```
def getfactors(x):
```

```
    y=[]
    for i in range(2,x+1):
        while x>0 and x%i==0:
            y.append(i)
            x=x/i
    return y
a=input('enter an integer:')
print getfactors(a)
a = raw_input('press Enter to quit ...')
```

输出：

```
enter an integer:20
[2, 2, 5]
```

6. 验证命题：任何一个数字不全相同的 4 整数,经过不超过 7 次的“重排求差”操作后,总会得到 6174，称 6174 为 4 位黑洞数。

"重排求差"操作是指用组成该数的数字重排后得到的最大数减去最小数。

例如，对 2012 重排之后得到最大数是 2210，最小数是 0122，2210-0122=2088；对 2088 重排之后得到最大数是 8820，最小数是 0288，8820-0288=8532；对 8532 重排之后得到最大数是 8532，最小数是 2358，8532-2358=6174；对 6174 重排之后得到最大数是 7641，最小数是 1467，7641-1467=6174。

```
a=[0,0,0,0]
while True:
    x=int(raw_input('enter a integer between 1000 and 9999 and the all numbers are not same:'))
    if x%1111 != 0:
        break
c=0
while x!=6174 and c<7:
    for i in range(0,4):
        a[i] = x%10
        x = x/10
    a.sort()
    max=0;  min=0
    for i in range(4):
        min = min*10+a[i]
        max = max*10+a[3-i]
    x=max-min
    print x,
if c>=7:
    print "\nerror"
a = raw_input('Press Enter to quit……-…-')
```

```
enter a integer between 1000 and 9999 and the all numbers are not same:1234
3087 8352 6174 Press Enter to quit……
```

7. 输入一个英文句子，输出其中包含的单词个数、字符个数及元音字母个数。

例如，输入"Do you like python?"，输出 words:4, characters: 19, vowels: 6

```
s=raw_input('Enter a string :')
word=len(s.split())
ch=len(s)
v=0
for i in range(ch):
    if s[i] in 'aeiouAEIOU':
      v+=1
print 'words: %d, characters: %d, vowels: %d' %(word,ch,v)
```

输出：

```
Enter a string :Enter a string
words: 3, characters: 14, vowels: 4
```

8. 删除一个人名列表中重复出现的名字后，对列表中剩余的名字按字典序排序后输出。

例如，对于包含人名 Tom, Mary , John ,Mike , Julia ,Mary ,Tom , Mary 的列表做上述处理后输出 [John,Julia,Mary,Mike,Tom]

```
name=[]
while True:
    a = raw_input('input a name:  ')
    if a != '':
        name.append(a)
    else:
        break
name = list(set(name))     #利用集合的值的唯一性，简单直接
name.sort()
print name
```

输出：

```
input a name: Tom
input a name: Mary
input a name: Jill
input a name: Tom
input a name: Mary
input a name:
['Jill', 'Mary', 'Tom']
```

讲解：在列表删除列表项时，必须注意方法的比较！

Answers to Part D

1. 编写文件复制程序

提示并接收用户输入的两个文件名，第一个文件名代表源文件，第二个文件名代表目标文件，将源文件的内容复制到目标文件并显示被复制的文件内容。

```
fname1=raw_input('Enter file name of source: ')
fname2=raw_input('Enter file name of destination: ')
f1=open(fname1,'r')
f2=open(fname2,'w')
s=f1.read()
print s
f2.write(s)
f1.close()
f2.close()
```

```
Enter file name of source: p8-1.py
Enter file name of destination: copy.txt
fname1=raw_input('Enter file name of source: ')
fname2=raw_input('Enter file name of destination: ')

f1=open(fname1,'r')
f2=open(fname2,'w')

s=f1.read()
print s
f2.write(s)

f1.close()
f2.close()
```

2. 编写文件过滤程序

提示并接收用户输入的一个 python 脚本程序文件名，删除程序中的所有注释内容（以#开始到行尾的字符串）后再写回到原文件中。

```
fname=raw_input('Enter file name : ')
f=open(fname,'r+')
s=f.readlines()
t=[]
print s
for line in s:
    if '#' not in line:
        t.append(line)
    elif not line.startswith('#'):
        k=line.partition('#')
        t.append(k[0]+'\n')
print t
f.seek(0)
f.truncate()
print f.tell()
```

```
f.writelines(t)
f.close()
raw_input('Press Enter to quit…-…-')
```

输出：

```
>>>
Enter file name : a.txt
['ai  \n', 'fdzila #88\n', 'inmz \n', 'afwiao  #00\n', 'xppqk\n']
['ai  \n', 'fdzila \n', 'inmz \n', 'afwiao  \n', 'xppqk\n']
0
Press Enter to quit……
```

a.txt：

之前：

```
ai
fdzila #88
inmz
afwiao #00
xppqk
```

，之后：

```
ai
fdzila
inmz
afwiao
xppqk
```

3. 编写计算器程序

接收包含一个算术运算算式的命令行参数（不检查算式正确性，输入时确保算式正确），完成算术算式运算，输出运算结果，最后将算式及运算结果保存到一个文件 res.txt 中。

提示：本题需要在命令提示符下执行：python p8-4.py 12+23，则输出到 res.txt 中的内容为"12+23=35"。

如果想要连续+，连续-，以及四则运算呢，程序将如何修改？

方法一：

```
import sys
fname=raw_input('Enter file name : ')
f=open(fname,"w")
s=sys.argv[1]
if '+' in s:
    a=s.partition('+')
    res=int(a[0])+int(a[2])
elif '-' in s:
    a=s.partition('-')
    res=int(a[0])-int(a[2])
elif '*' in s:
    a=s.partition('*')
    res=int(a[0])*int(a[2])
elif '/' in s:
    a=s.partition('/')
    if int(a[2]):
        res=int(a[0])/int(a[2])
    else:
        print "devided by 0"
```

```
      res=0
else:
   print 'error'
print res
f.write(sys.argv[1]+'=')
f.write(str(res))
f.close()
```

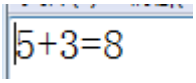
方法二：

```
import sys
fname=raw_input('Enter file name : ')
f=open(fname,'w')
res=eval(sys.argv[1])
f.write(sys.argv[1])
f.write('=')
f.write(str(res))
f.close()
print '%d'% res
```

输出：

```
D-4>python p8-4.py 5+3
Enter file name : a.txt
8
```

文件 a.txt 中：



Answers to Part E

1.安装 PyPI，利用 pip 安装任意一个包并列举此包所包含的若干函数.

2.求前 n 个 Fibonacci 数。

```
def fib(n):
    a, b = 0, 1
    i = 0
    while i < n:
        print a
        a, b = b, a+b
        i += 1
fib(8)
```

3.实现 max()和 min()内建函数。

(a) 写分别带两个元素返回一个较大和较小元素，简单的 max2()和 min2()函数，如果相等输出该元素。他们应该可以用任意的 python 对象运作。举例来说，max2(4,8)和 min2(4,8)会各自每次返回 8 和 4。

(b) 创建使用了在 a 部分中的解来重构 max()和 min()的新函数 my_max()和 my_min()。这些函数分别返回非空队列中一个最大和最小值。它们也能带一个参数集合作为输入。用数字和字符串来测试你的解。

(a)

```
def max2(x,y):
    if x > y:
        return x
    else:
        return y

def min2(x,y):
    if x > y:
        return y
    else:
        return x
```

4. 实现 printf()的函数。有一个值参数和格式字符串，根据格式化字符串显示可变参数，格式化字符串中的值允许特别的字符串格式操作指示符，如%d, %f 等。提示：无需实现字符串操作符功能性。

```
def printf(format,*arg):
     print format%arg

printf ("%d is not equal to %d",1,2)
```

5. 使用 **reduce()**进行函数式编程以及递归。

(a) 用一分钟写一个带 **x,y** 并返回他们乘积的名为 **mult(x,y)**的简单小巧的函数。

(b) 用你在 **a** 中创建的 **mult()**函数以及 **reduce** 来计算阶乘。

(c) 彻底抛弃掉 **mult()**的使用，用 **lamda** 表达式替代。

(d) 我们描绘了一个递归解决方案来找到 **N!**

(a)

```
def mult(x,y):
    return x*y
```

(b)

```
def fac(n):
    def fac(n):
    a = reduce(mult,range(1,6))
    return a
```

(c)

```
reduce ((lambda x,y:x*y),range(1,6))
```

(d)

```
def fac(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n*fac(n - 1)
```

6.首先产生一个随机数序列，然后过滤（利用 **filter()**函数）掉所有的偶数

```
from random import randint
def odd(n):
    return n % 2
allNums = []
for eachNum in range(9):
    allNums.append(randint(1, 99))
print filter(odd, allNums)
```

Answers to Part F

1、创建一个由有序数值对(x, y) 组成的 Point 类，它代表某个点的 X 坐标和 Y 坐标。
X 坐标和 Y 坐标在实例化时被传递给构造器，如果没有给出它们的值，则默认为坐标的原点。

```
class point(object):
    "creat point"
    def __init__(self, x = 0, y = 0):
        "creat point initialize by x,y"
        self.x = x
        self.y = y
    def output(self):
        "output coordinate(x,y)"
        print "coordinate:(%d,%d)" %(self.x,self.y)
```

测试输出：

```
>>> p=point()
>>> p.output()
coordinate:(0,0)
>>> p=point(4,5)
>>> p.output()
coordinate:(4,5)
>>> p.x
4
>>> p.y
5
```

2、创建一个直线/直线段类。除主要的数据属性：一对坐标值外，它还具有长度和斜线属性。
你需要覆盖__repr__()方法(如果需要的话，还有__str__()方法)，使得代表那条直线(或直线段)的字符串表示形式是由一对元组构成的元组，即，((x1, y1), (x2, y2)).

__repr__ 将直线的两个端点(始点和止点)显示成一对元组
length 返回直线段的长度
slope 返回此直线段的斜率(或在适当的时候返回 **None**)

```
import math
class straight_line(object):
    "creat straight line"
    def __init__(self, x1 = 0,y1 = 0,x2 =0,y2 = 0 ):
        "通过两个端点创建直线（或直线段）"
        self.x1 = x1
        self.y1 = y1
        self.x2 = x2
        self.y2 = y2
    def length(self):
        return pow(((self.x2 - self.x1) ** 2 + (self.y2 - self.y1) ** 2) ,0.5)
```

```
    def slope(self):
        if self.x1 == self.x2:
            return None
        else:
            return      float(self.y2 - self.y1)/(self.x2 - self.x1)
    def __repr__(self):
        return "((%f,%f),(%f,%f))" % (self.x1,self.y1,self.x2,self.y2)
```

测试输出：

```
>>> line=straight_line(4,5,9,12)
>>> line
((4.000000,5.000000),(9.000000,12.000000))
>>> line.length()
8.602325267042627
>>> line.slope()
1.4
>>> line=straight_line(4,4,5,5)
>>> line.length()
1.4142135623730951
>>> line.slope()
1.0
```

3、对类进行定制。写一个类，名为 MoneyFmt，用来将浮点数值转换为金额。

MoneyFmt 类里只有一个数据值(即，金额)，和五个方法。

__init__()构造器对数据进行初始化，

update()方法把数据值替换成一个新值，

__nonzero__()是布尔型的，当数据值非零时返回 True，

__repr__()方法以浮点数的形式返回金额，

__str__()方法以字符格式显示金额，金额数里应该有逗号(比如 1,000,000)和美元的货币符号，如果有负号，它必须出现在美元符号的左边。1234567.8901 ==> “$1,234,567.89”

```
class MoneyFmt(object):
    "creat money class"
    def __init__(self, x):
        self.x = float(x)
    def update(self,x):
        self.x = float(x)
    def    __nonzero__(self):
        if self.x==0:
            return False
        else:
            return True
    def __repr__(self):
        return str(self.x)
    def __str__(self):
        x = abs(self.x)
```

```
        x = round(x,2)
        if(x*10) % 1 ==0:
            s = format(x,',')+'0'
        else:
            s = format(x,',')
        if self.x<0:
            s = "-$" + s
        else:
            s = "$" + s
        return s
```

测试输出：

```
>>> m=MoneyFmt(30)
>>> m
30.0
>>> print m
$30.00
```

Python 试卷

2014 年 7 月 18 日 14：00 - 16：00

【要求】按照每题后面的文件名要求保存代码，最后压缩成一个 rar 或者 zip 文件提交，提交压缩包文件名为 “学号+姓名+Python” 。
提交至 Ftp://172.26.184.101/Upload/Python 提交处/Python 考试提交处

一. 改错题——在 Python 的 IDLE 环境中新建 py 文件，将代码复制进去，修改程序中的错误，使代码能够符合题目的要求运行正确，请提交正确的程序代码。（提示：每一题的错误数不一样）

1. 改正以下代码中的错误，正确输出字符串"Happy New Year"的逆序“raeY weN yppaH”。（参考错误数：1）保存为 1-1.py

```
def foo(s):
    return s[::0]
print foo("Happy New Year")
```

2. 求 1 到 10 的和，要求输出“10+9+8+7+6+5+4+3+2+1=55”（参考错误数：2）保存为 1-2.py

```
s=0
st=[]
for i in range(10, 0):
    s = s + i
    st.append(str(i))
print '+'.join(st) + '=' + s
```

3. 在屏幕上输出 1-20，要求分 2 行输出，第一行 1-10，第二行 11-20（参考错误数：1）保存为 1-3.py

```
for i in range(1,21):
    print i,
    if i%10 = 0:
        print
```

4. 将元组 t 中的 'b' 和 'c' 互换位置，输出 t 为('a', 'c', 'b', 'd')（参考错误数：2）保存为 1-4.py

```
t=('a','b','c','d')
t[1] , t[2] = t[2] , t[1]
print t
```

5. 输入一个数字 x，在列表 L 中查找是否存在 x，如果存在则输出“x 在 L 中”，否则输出“找不到 x”。（参考错误数：1）保存为 1-5.py

```
x = input('Enter the need num: ')
L = [2,13,54,50,22,73,72]
if x in L:
    print '%d is in L' % x
```

```
else:
print 'Can not find %d' % x
```

6. lambda 函数完成乘方运算，输出正确的结果 2 的 3 次方为 8。（参考错误数：1）保存为 1-6.py

```
z=lambda x,y: x**y
print z(2,3,4)
```

二. 简单编程题

1. 已知列表 a=['apple', 'pear', 'orange']，采用列表解析由列表 a 动态生成列表 b，列表 b 包含的元素为['elppa', 'raep', 'egnaro']。代码保存为 2-1.py

2. 将无序字符串 s=”maieinzl”排序为有序的字符串输出，输出为 “aeiilmnz“。代码保存为 2-2.py

3. 已知列表 a=['apple', 'pear', 'orange']，采用列表解析由列表 a 动态生成列表 b，列表 b 包含的元素为['elppa', 'raep', 'egnaro']。代码保存为 2-3.py

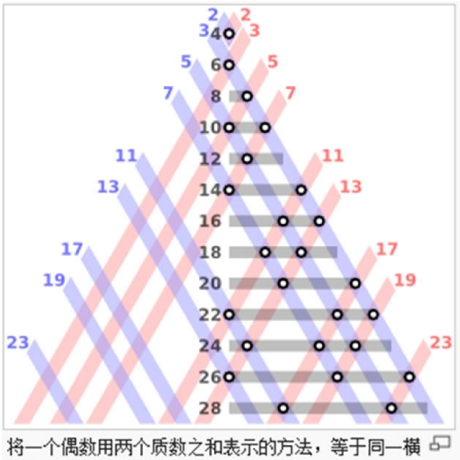
4. 已知列表 a=['12','3','45','36','890','105']，统计列表 a 中数字字符’0’~’9’各自出现的次数，输出统计结果为 {0: 2, 1: 2, 2: 1, 3: 2, 4: 1, 5: 2, 6: 1, 7: 0, 8: 1, 9: 1}。代码保存为 2-4.py

5. 使用内建函数 reduce()以及 lambda 进行函数式编程，实现列表 a 中所有数值的总和。
假设 a = [3, 4, 5, 9]，则 reduce(……)输出“21”。 代码保存为 2-5.py

6. 定义一个表示班级的类，数据属性包括班主任姓名和学生姓名列表，具有以下方法：代码保存为 2-6.py
（1）__init__()构造器对数据进行初始化，以班主任的姓名初始化；
（2）add()：加入一个学生姓名；
（3）remove()：按照姓名移除学生；
（4）print_all()：输出班主任姓名和所有学生名单。

三. 应用题

1. 哥德巴赫猜想是数论中存在最久的未解问题之一,也是二十世纪初希尔伯特第八问题中的一个子问题。这个猜想最早出现在 1742 年普鲁士人克里斯蒂安·哥德巴赫与瑞士数学家莱昂哈德·欧拉的通信中。用现代的数学语言，哥德巴赫猜想可以陈述为：“任一大于 2 的偶数，都可表示成两个素数之和。”



将一个偶数用两个质数之和表示的方法，等于同一横线上，蓝线和红线的交点数。

输入起始值 m，终值 n，用程序验证哥德巴赫猜想在 m ~ n 范围内成立。代码保存为 3-1.py

2. 假设需要对看过的电影进行整理，由于每次看完电影时记录下来的电影资料比较乱，然后把每次的记录作为一个列表项，结果得到这样一个列表：

movies = ["The Holy Grail",1975,"Terry Jones & Terry Gilliam",91,["Graham Chapman",["Michael Palin","john Cleese","Terry Gilliam","Eric Idle", "Terry Jones"]]]

现在要将该列表中的所有列表项，包括列表项中的列表项（我们称之为列表的嵌套）全部进行输出。输出为：

```
The Holy Grail
1975
Terry Jones & Terry Gilliam
91
Graham Chapman
Michael Palin
john Cleese
Terry Gilliam
Eric Idle
Terry Jones
```

要求写一个这样的 Python 程序：输出任意一个嵌套列表的所有列表项，即该列表的嵌套的层数不确定。（提示：用递归实现多层嵌套的列表项访问。假设不确定列表的嵌套有多少层，现在要求将列表所有元素都一一输出。则考虑用递归实现。否则不知道要多少层的循环嵌套。）代码保存为 3-2.py

四. 完成调查问卷。打开 IE 浏览器，输入网址 http://172.26.184.101:9587