Python 知识集萃

1. print(十进六制) 如 print(0xffff) 输出十进制，其他一样，0o3474、0b101.
2. type（）函数可以识别数类型 print（type10/3（））---float
3. 对数的大小不做限制，可以无限大
4. /是除法，//整除，
5. 复数 complex 3+4i　x=3+5i　print（x.real）是实部，x.imag 为虚部
6. 乘方 x**y x 的 y 次方
7. 三引号可以用于文段类字符串 Docstring，也可以 print
8. 三引号用于文档字符串，

   def like():

   ‘’’

   函数属性

   ‘’’

   Print(like.__doc__) _——函数名.__doc__

   即可输出函数相关信息
9. Python 3.0 使用 Unicode，utf-8 编码将一个字节表示的用一个子节表示，不能则二位，转换：a=”dsfsdf”,　print(a.encode(‘utf-8’))
10. Ascii 与 int 转化：　print（ord(‘a’）)---97,　print(chr(97))---a
11. 字符串长度　len（）函数　print（len(“fddgd’）)
12. 字符串格式化：print(‘{} is {}’**.format**(‘xutao’,20))

Print(‘%s is %d ‘ **%(‘xutao’,20**))

13.序列：

列表 list：a_list=[‘xutao’, 888,true] 中括号 任何类型

Print(type(a_list))

依然可用于索引 print(a_list[2])

最大索引是 len(a_list)-1

索引是负数则表示倒数 print(a_list[-2] 倒数第二

添加：a_list.append(‘xutao’)

插入：a_list.insert(index，insertContent)把元素向后排

弹出：a_list.pop() 最后一个

反转：a.reverse()

指定弹出：a_list(index, deleteContent)

替换：a_list[index]=Change

嵌套：b_list=[‘xutao’2 ,3,[1,2,3]] 内层看做整体，且不可改变内层整体，只能改变内层中的一个元素

嵌套修改：a_list=[index][index1]=change

List 允许重复

使用 del 语句可以从一个列表中依索引而不是值来删除一个元素。这与使用 pop() 返回一个值不同。可以用 del 语句从列表中删除一个切割，或清空整个列表（我们以前介绍的方法是给该切割赋一个空列表）。

元组 tuple:　**不可改变性**

a_tuple=(1,2,3) **括号**

可以嵌套 list

Tuple.index(obj,start=0,end=len(string)) 检查 object 是不是在 tuple 中，返回 index

Tuple.count(obj) 返回 obj 在 tuple 中出现的次数

字符串列表字典序排列经典算法：

```
def strmax(x,y):  #比较字符串的函数
    if x<y:
        return x
    else:return y
list_after=[]
c=0
while list_before!=[]:
    list_xutao.append(reduce(strmax,list_before))
    list_last.remove(list_after[c])
    c+=1
```

14.数据类型：容器

dict 字典：映射 一对一　易查找，

a_dict={‘name’: ‘xutao’ , ’age’:20}

print(a_dict[‘age’]---20

修改：a_dict[‘name’]=’xu’

添加：a_dict[‘gender’]=’female’

存在性：print(‘sex’ in A_dict)——以 key 来检测

Print(a_dict , __contain__(‘ago’))

取出：a_dict.get(‘name’)

若不存在返回默认值 None，也可以修改默认值：print（a_dict.get(‘gender’,’male’）

删除：a_dict.pop(‘mane’)

A_dict.popitem()删除最后一项

A_dict.clear()清空

字典里的对象如**‘name’是不能重复**，若是重复，取最后一个定义；**不能修改**，只能修改对象后面的属性。

set 集合:

a_set={1，2, 3}

重复的会忽略

单个添加：a_set.add(5) 已有的不能成功添加

末尾成段添加：a_set.update([6,7,8,9])

删除：a_set.remove(2)

清空：a_set.clear()

Set 里的数据元素不能修改不能重复

区分：

- List: []
- Tuple()
- Dict: {key:value,,,}

- Set:{a,b,c}

15.for I in range(n):---0-n-1

for I in range(m,n):---m-n-1

for I in range(m,n,x): 间隔想，不含 n

16.在 dict 中，

For key in a_dict: 仅对象

For item in a_dict.items(): 对象与数据，合并

For item,value in a_dict.items(): 对象与数据,分开

17.Pass: 占位符，什么都不做（函数中）

18.判断类型：isinstance（A，(int,float，B）) 判断 A 是不是如 B 的类

19.四舍五入：round(x), abs:绝对值

20.Raise TypeError('content') 类型错误

21.可变参数：函数的形参可以是多个时：fun(*args),fun(*name)

两个：关键字参数（如字典）：def fun(**xt)**

调用：Fun(name='xutao',age=20)

22.文件操作：

f=open('xt.txt','w',encoding='utf-8')//不存在则创建

//w—覆盖，a—追加，r—读取，r+--可写，追加,w+（文件可无，覆盖）

f.write("shit")

```
f.close()
content=f.read()//文件必须存在
c=f.read(n)//n 个代码单元
v=f.readline()//读取一行，列表类型
   for conn in v
        conn=conn.strip()
        print(conn)
print(content)
```

```
with open('x.txt','r',encoding='utf-8') as f:
```

2	file.flush() 刷新文件内部缓冲，直接把内部缓冲区的数据立刻写入文件，而不是被动的等待输出缓冲区写入。
3	file.fileno() 返回一个整型的文件描述符(file descriptor FD 整型)，可以用在如 os 模块的 read 方法等一些底层操作上。
4	file.isatty() 如果文件连接到一个终端设备返回 True，否则返回 False。
5	file.next() 返回文件下一行。
6	file.read([size]) 从文件读取指定的字节数，如果未给定或为负则读取所有。
7	file.readline([size]) 读取整行，包括 "\n" 字符。
8	file.readlines([sizeint]) 读取所有行并返回列表，若给定 sizeint>0，返回总和大约为 sizeint 字节的行，实际读取值可能比 sizeint 为需要填充缓冲区。
9	file.seek(offset[, whence]) 设置文件当前位置

10	file.tell() 返回文件当前位置。
11	file.truncate([size]) 从文件的首行首字符开始截断，截断文件为 size 个字符，无 size 表示从当前位置截断；截断之后 V 后字符被删除，其中 Widnows 系统下的换行代表 2 个字符大小。
12	file.write(str) 将字符串写入文件，没有返回值。
13	file.writelines(sequence) 向文件写入一个序列字符串列表，如果需要换行则要自己加入每行的换行符。

23.python 中相同数据的的赋值会共享同一片空间地址

import sys

sys.getrefcount(value)

可以查看一个值被多少次引用

24.**id(value)** 查看地址

25.字符串内建函数：返回新的字符串

转化函数

str.lower()

str.upper()

str.swapcase() 交换大小写：字母大写转换为小写，小写转换为大写

str.title() 将首字母大写，句中字母变为小写，每个单词首字母大写

str.capitalize()仅首字母大写，其他小写

搜索函数：

Str.find(str,[start=0,stop=len(Str)]): 计算 Str 中出现 str 的第一个字母的索引，如无返回-1

string.index(str,[start=0,stop=len(string)]):计算 string 中出现 str 的第一个字母的索引，如无则引发异常

string.count(str,[start=0,stop=len(string)]):计算 str 在 string 中出现的次数

string.endwith(str,[start=0,stop=len(string)]):检查 string 是否已 str 结尾。若是返回 true 否则返回 false

查找范围若不写则默认整个字符串

替换函数：

String.repiace(str1,str2,[num=string.count(str1)]):将 str1 替换为 str2，num 替换次数，默认替换次数为 str1 出现次数，即全部替换

String.**strip**(**chr**):在 string 的开头和结尾删除 chr,当 chr 为空时删除空白符

String.rstrip():删除 string 末尾的空格或换行符

String.lstrip():删除 string 前面空格换行

String.strip():去掉 string 前后换行空白

分割组合函数：

String.split(chr,num) 以 chr 为分割标志将字符串分割 num 次，默认 num 为 chr 出现次数，注意返回的是一个列表

Str.join(str1,str2):**以 str 为连接符拼接,no,**

判断函数（同 C++）：返回 bool

String. Isdigit()　数字，仅限于非负数

对于负数：'-n'.lstrip('-')).isdigit())

String.islower()　小写

String.isupper()　注意这俩个大小写判断符 在含有数字时不影响判断

String.isspace(): 空白

dir(str)　命令行参数里

string.isalpha:字母

string.isalnum:字母与数字（可复合）

26.系统内建函数：

enumerate(iter) 返回一个枚举类型，用 for 循环可迭代出所有枚举对象

用 for 循环时可以是一个参数，也可以是两个：

For val1,val2 in enumerate(list_1)

　　Print(val1":"val2)

max("a","b"): 比较 ASCII

特定比较 ：

max("a23","b34",key=lambda x:x[1])—比较第二个 3>4

reverse(str)　for 循环迭代

list

sorted(list)—并不会真正修改

sum(list)　完全为数字时

help（n）

zip(list1,list2)—将 list1，list2 一一对应组成新的 list3，不足时报错

map(函数对象，list)—将 list 所有对象经过函数处理后返回，可用 for 迭代

```
for I in map(funcname,liat_1)
    print(i)
```

reduce(func,list):

从 functools 中导入：eg:

```
Form functools import reduce
Def func(x,y):
    Return x*y
Reduce(func,list)
```

去掉最后一个字符如\n:　方法 1：x = 'asd\n'

X=x[:-1]

方法 2：

x.strip("\n)

27. 推导式：

列表推导式

List1=

List2=[var*var for var in list1]

也可以有更多的条件控制

List3=[var*var for var in list1 if var!=n]

字典推导式：

Mydict={var:"value" for var in mylist if var >n}

集合推导式：

Myset ={var+1 for var in mylist}

28.函数构成

函数默认参数必须在最后一个

不定参数函数：

加了星号（*）的变量名会存放所有未命名的变量参数

def func(*name)

Func(1,2,3,4)

:def func(**var)—会将输入的值封装为一个字典

Func(a=1,b=2,c=3)

：联用：def func(*var,**var)—分别封装成元组与字典

返回多个元素时，会封装成一个元组，实为一个

字符串*int，重复多次

29.函数作用域：

全局 global 局部

传递一个可变对象：按引用传递—会改变原始对象的值，因为他们本质是一个地址，由不同的引用，共享空间， 可用复制方式防止，如 var=[1,2,3],var1=var[:]

Python 的作用域一共有 4 种，分别是：

- L （Local） 局部作用域
- E （Enclosing） 闭包函数外的函数中
- G （Global） 全局作用域
- B （Built-in） 内建作用域

以 L –> E –> G –>B 的规则查找，即：在局部找不到，便会去局部外的局部找（例如闭包），再找不到就会去全局找，再者去内建中找。

30.正则 re:

Import re

特殊符号与字符：

匹配一个范围：a[a-z]c:aXc,其中 X 为 a-c 任意数， 1[2-5]4

匹配前面出现的正则表达式任意次（含 0）：a*. 如 a*与’‘, a ,aaa,aaaaa 等匹配

若匹配 0 次或 1 次：a?. eg: c?匹配 ‘ ’，c

若匹配一次或无穷次：a+

点“.”匹配任意字符

组合“.*”匹配多个这种组合

\d:匹配任何数字

\s:匹配任何空白符：\n\t\r\v\f

\w: 匹配任何数字，字母，字符，相当于：[A-Za-z0-9_]

^a: 匹配任意以 a 开头的字符串

[^a]:除了 a 的其他字符串：如: a[^acv]b

$a : 匹配以 a 结尾的

.	匹配除 "\n" 之外的任何单个字符。要匹配包括 '\n' 在内的任何字符，请使用象 '[.\n]' 的模式。
\d	匹配一个数字字符。等价于 [0-9]。
\D	匹配一个非数字字符。等价于 [^0-9]。
\s	匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [\f\n\r\t\v]。
\S	匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。
\w	匹配包括下划线的任何单词字符。等价于'[A-Za-z0-9_]'。
\W	匹配任何非单词字符。等价于 '[^A-Za-z0-9_]'。
修饰符	描述
re.I	使匹配对大小写不敏感
re.L	做本地化识别（locale-aware）匹配
re.M	多行匹配，影响 ^ 和 $
re.S	使 . 匹配包括换行在内的所有字符
re.U	根据 Unicode 字符集解析字符。这个标志影响 \w, \W, \b, \B.
re.X	该标志通过给予你更灵活的格式以便你将正则表达式写得更易于理解。
模式	描述
^	匹配字符串的开头
$	匹配字符串的末尾。
.	匹配任意字符，除了换行符，当 re.DOTALL 标记被指定时，则可以匹配包括换行符的任意字符。
[...]	用来表示一组字符,单独列出：[amk] 匹配 'a'，'m'或'k'
[^...]	不在[]中的字符：[^abc] 匹配除了 a,b,c 之外的字符。

re*	匹配 0 个或多个的表达式。
re+	匹配 1 个或多个的表达式。
re?	匹配 0 个或 1 个由前面的正则表达式定义的片段，非贪婪方式
re{ n}	匹配 n 个前面表达式。例如，"o{2}"不能匹配"Bob"中的"o"，但是能匹配"food"中的两个 o。
re{ n,}	精确匹配 n 个前面表达式。例如，"o{2,}"不能匹配"Bob"中的"o"，但能匹配"foooood"中的所有 o。"o{1,}" "o+"。"o{0,}"则等价于"o*"。
re{ n, m}	匹配 n 到 m 次由前面的正则表达式定义的片段，贪婪方式
a| b	匹配 a 或 b
(re)	G 匹配括号内的表达式，也表示一个组
(?imx)	正则表达式包含三种可选标志：i, m, 或 x 。只影响括号中的区域。
(?-imx)	正则表达式关闭 i, m, 或 x 可选标志。只影响括号中的区域。
(?: re)	类似 (...), 但是不表示一个组
(?imx: re)	在括号中使用 i, m, 或 x 可选标志
(?-imx: re)	在括号中不使用 i, m, 或 x 可选标志
(?#...)	注释.
(?= re)	前向肯定界定符。如果所含正则表达式，以 ... 表示，在当前位置成功匹配时成功，否则失败。但一旦所式已经尝试，匹配引擎根本没有提高；模式的剩余部分还要尝试界定符的右边。
(?! re)	前向否定界定符。与肯定界定符相反；当所含表达式不能在字符串当前位置匹配时成功。
(?> re)	匹配的独立模式，省去回溯。
\w	匹配数字字母下划线
\W	匹配非数字字母下划线
\s	匹配任意空白字符，等价于 [\t\n\r\f]。
\S	匹配任意非空字符
\d	匹配任意数字，等价于 [0-9]。
\D	匹配任意非数字

\A	匹配字符串开始
\Z	匹配字符串结束，如果是存在换行，只匹配到换行前的结束字符串。
\z	匹配字符串结束
\G	匹配最后匹配完成的位置。
\b	匹配一个单词边界，也就是指单词和空格间的位置。例如， 'er\b' 可以匹配"never" 中的 'er'，但不能匹"verb" 中的 'er'。
\B	匹配非单词边界。'er\B' 能匹配 "verb" 中的 'er'，但不能匹配 "never" 中的 'er'。
\n, \t, 等。	匹配一个换行符。匹配一个制表符, 等
\1...\9	匹配第 n 个分组的内容。
\10	匹配第 n 个分组的内容，如果它经匹配。否则指的是八进制字符码的表达式。

31.re 模块：

re.compile(pattern) –编译正则表达式

匹配 march 从字符串开头进行匹配

搜索 search 从字符串任意位置始匹配

Re.march(pattern, string) –用正则表达式模式 pattern 匹配 string，匹配成功则返回一个匹配对象，可使用 grope 函数获取匹配到的值：name.group()，否则返回 None

Eg: import re

A=re.complie(^a)// 将需要的字符串拿来编译好，以便用

B=re.march(A,’acdvdf’)

C=re.search(A,’dgfddfsa’)

B.group()

re.findall(A,'string')—找出 string 中所有的 A　(不用 group())

s=re.compile('[0-9]+')

d=re.findall(s,'dsfs3s4rfsf454dfsrtr')

re.sub(str1,str2,str3)---将 str3 中的 str1 替换为 str2

注：对于字符串中含有正则意义的字符如*，要用转义*

Eg: re.sub('[0-9],'','sdf3443fd6')—去数字

re.subn 一样，但是要用数字表示替换的个数，并封装为元组

1. 正则表达式语法

1.1 字符与字符类

1 特殊字符：\.^$?+*{}[]()|

以上特殊字符要想使用字面值，必须使用\进行转义

2 字符类

1. 包含在[]中的一个或者多个字符被称为字符类，字符类在匹配时如果没有指定量词则只会匹配其中的一个。

2. 字符类内可以指定范围，比如[a-zA-Z0-9]表示 a 到 z，A 到 Z，0 到 9 之间的任何一个字符

3. 左方括号后跟随一个^，表示否定一个字符类，比如[^0-9]表示可以匹配一个任意非数字的字符。

4. 字符类内部，除了\之外，其他特殊字符不再具备特殊意义，都表示字面值。^放在第一个位置表示否定，放在其他位置表示^本身，-放在中间表示范围，放在字符类中的第一个字符，则表示-本身。

5. 字符类内部可以使用速记法，比如\d \s \w

3 速记法

. 可以匹配除换行符之外的任何字符，如果有 re.DOTALL 标志，则匹配任意字符包括换行

\d 匹配一个 Unicode 数字，如果带 re.ASCII，则匹配 0-9

\D 匹配 Unicode 非数字

\s 匹配 Unicode 空白，如果带有 re.ASCII，则匹配\t\n\r\f\v 中的一个

\S 匹配 Unicode 非空白

\w 匹配 Unicode 单词字符，如果带有 re.ascii,则匹配[a-zA-Z0-9_]中的一个

\W 匹配 Unicode 非单子字符

1.2 量词

1. ? 匹配前面的字符 0 次或 1 次

2. * 匹配前面的字符 0 次或多次

3. + 匹配前面的字符 1 次或者多次

4. {m} 匹配前面表达式 m 次

5. {m,} 匹配前面表达式至少 m 次

6. {,n} 匹配前面的正则表达式最多 n 次

7. {m,n} 匹配前面的正则表达式至少 m 次，最多 n 次

注意点：

以上量词都是贪婪模式，会尽可能多的匹配，如果要改为非贪婪模式，通过在量词后面跟随一个?来实现

1.3 组与捕获

1 ()的作用：

1. 捕获()中正则表达式的内容以备进一步利用处理，可以通过在左括号后面跟随?:来关闭这个括号的捕获功能

2. 将正则表达式的一部分内容进行组合，以便使用量词或者|

2 反响引用前面()内捕获的内容：

1. 通过组号反向引用

每一个没有使用?:的小括号都会分配一个组好，从 1 开始，从左到右递增，可以通过\i 引用前面()内表达式捕获的内容

2. 通过组名反向引用前面小括号内捕获的内容

可以通过在左括号后面跟随?P<name>,尖括号中放入组名来为一个组起一个别名，后面通过(?P=name)来引用 前面捕获的内容。如(? P<word>\w+)\s+(?P=word)来匹配重复的单词。

3 注意点：

反向引用不能放在字符类[]中使用。

1.4 断言与标记

断言不会匹配任何文本，只是对断言所在的文本施加某些约束

1 常用断言：

1. \b 匹配单词的边界，放在字符类[]中则表示 backspace

2. \B 匹配非单词边界，受 ASCII 标记影响

3. \A 在起始处匹配

4. ^ 在起始处匹配，如果有 MULTILINE 标志，则在每个换行符后匹配

5. \Z 在结尾处匹配

6. $ 在结尾处匹配，如果有 MULTILINE 标志，则在每个换行符前匹配

7. (?=e) 正前瞻

8. (?!e) 负前瞻

9. (?<=e) 正回顾

10.(?<!e) 负回顾

2 前瞻回顾的解释

前瞻： exp1(?=exp2) exp1 后面的内容要匹配 exp2

负前瞻: exp1(?!exp2) exp1 后面的内容不能匹配 exp2

后顾: (?<=exp2)exp1 exp1 前面的内容要匹配 exp2

负后顾: (?<!exp2)exp1 exp1 前面的内容不能匹配 exp2

例如：我们要查找 hello，但是 hello 后面必须是 world，正则表达式可以这样写："(hello)\s+(?=world)",用来匹配"hello wangxing"和"hello world"只能匹配到后者的 hello

1.5 条件匹配

(?(id)yes_exp|no_exp):对应 id 的子表达式如果匹配到内容，则这里匹配 yes_exp，否则匹配 no_exp

1.6 正则表达式的标志

1. 正则表达式的标志有两种使用方法

1. 通过给 compile 方法传入标志参数，多个标志使用|分割的方法,如 re.compile(r"#[\da-f]{6}\b", re.IGNORECASE|re.MULTILINE)

2. 通过在正则表达式前面添加(?标志)的方法给正则表达式添加标志，如(?ms)#[\da-z]{6}\b

2. 常用的标志

re.A 或者 re.ASCII， 使\b \B \s \S \w \W \d \D 都假定字符串为假定字符串为 ASCII

re.I 或者 re.IGNORECASE 使正则表达式忽略大小写

re.M 或者 re.MULTILINE 多行匹配，使每个^在每个回车后，每个$在每个回车前匹配

re.S 或者 re.DOTALL 使.能匹配任意字符，包括回车

re.X 或者 re.VERBOSE 这样可以在正则表达式跨越多行，也可以添加注释，但是空白需要使用\s 或者[ ]来表示，因为默认的空白不再解释。如：

```
re.compile(r"""
    <img\s +) #标签的开始
    [^>]*? #不是 src 的属性
    src= #src 属性的开始
    (?:
    (?P<quote>['"]) #左引号
    (?P<image_name>[^\1>]+?) #图片名字
    (?P=quote) #右括号
    """,re.VERBOSE|re.IGNORECASE)
```

2. Python 正则表达式模块

2.1 正则表达式处理字符串主要有四大功能

1. 匹配 查看一个字符串是否符合正则表达式的语法，一般返回 true 或者 false
2. 获取 正则表达式来提取字符串中符合要求的文本
3. 替换 查找字符串中符合正则表达式的文本，并用相应的字符串替换
4. 分割 使用正则表达式对字符串进行分割。

2.2 Python 中 re 模块使用正则表达式的两种方法

1. 使用 re.compile(r, f)方法生成正则表达式对象，然后调用正则表达式对象的相应方法。这种做法的好处是生成正则对象之后可以多次使用。

2. re 模块中对正则表达式对象的每个对象方法都有一个对应的模块方法，唯一不同的是传入的第一个参数是正则表达式字符串。此种方法适合于只使用一次的正则表达式。

2.3 正则表达式对象的常用方法

1. rx.findall(s,start, end):

返回一个列表，如果正则表达式中没有分组，则列表中包含的是所有匹配的内容，

如果正则表达式中有分组，则列表中的每个元素是一个元组，元组中包含子分组中匹配到的内容，但是没有返回整个正则表达式匹配的内容

2. rx.finditer(s, start, end):

返回一个可迭代对象

对可迭代对象进行迭代，每一次返回一个匹配对象，可以调用匹配对象的 group()方法查看指定组匹配到的内容，0 表示整个正则表达式匹配到的内容

3. rx.search(s, start, end):

返回一个匹配对象,倘若没匹配到，就返回 None

search 方法只匹配一次就停止，不会继续往后匹配

4. rx.match(s, start, end):

如果正则表达式在字符串的起始处匹配，就返回一个匹配对象，否则返回 None

5. rx.sub(x, s, m):

返回一个字符串。每一个匹配的地方用 x 进行替换，返回替换后的字符串，如果指定 m，则最多替换 m 次。对于 x 可以使用/i 或者/g<id>id 可以是组名或者编号来引用捕获到

的内容。

模块方法 re.sub(r, x, s, m)中的 x 可以使用一个函数。此时我们就可以对捕获到的内容推过这个函数进行处理后再替换匹配到的文本。

6. rx.subn(x, s, m):

与 re.sub()方法相同，区别在于返回的是二元组，其中一项是结果字符串，一项是做替换的个数。

7. rx.split(s, m):分割字符串

返回一个列表

用正则表达式匹配到的内容对字符串进行分割

如果正则表达式中存在分组，则把分组匹配到的内容放在列表中每两个分割的中间作为列表的一部分，如：

rx = re.compile(r"(\d)[a-z]+(\d)")

s = "ab12dk3klj8jk9jks5"

result = rx.split(s)

返回['ab1', '2', '3', 'klj', '8', '9', 'jks5']

8. rx.flags():正则表达式编译时设置的标志

9. rx.pattern():正则表达式编译时使用的字符串

2.4 匹配对象的属性与方法

01. m.group(g, ...)

返回编号或者组名匹配到的内容，默认或者 0 表示整个表达式匹配到的内容，如果指定多个，就返回一个元组

02. m.groupdict(default)

返回一个字典。字典的键是所有命名的组的组名，值为命名组捕获到的内容

如果有 default 参数，则将其作为那些没有参与匹配的组的默认值。

03. m.groups(default)

返回一个元组。包含所有捕获到内容的子分组，从 1 开始，如果指定了 default 值，则这个值作为那些没有捕获到内容的组的值

04. m.lastgroup()

匹配到内容的编号最高的捕获组的名称，如果没有或者没有使用名称则返回 None(不常用)

05. m.lastindex()

匹配到内容的编号最高的捕获组的编号，如果没有就返回 None。

06. m.start(g):

当前匹配对象的子分组是从字符串的那个位置开始匹配的,如果当前组没有参与匹配就返回-1

07. m.end(g)

当前匹配对象的子分组是从字符串的那个位置匹配结束的，如果当前组没有参与匹配就返回-1

08. m.span()

返回一个二元组，内容分别是 m.start(g)和 m.end(g)的返回值

09. m.re()

产生这一匹配对象的正则表达式

10. m.string()

传递给 match 或者 search 用于匹配的字符串

11. m.pos()

搜索的起始位置。即字符串的开头，或者 start 指定的位置(不常用)

12. m.endpos()

搜索的结束位置。即字符串的末尾位置，或者 end 指定的位置(不常用)

2.5 总结

1. 对于正则表达式的匹配功能，Python 没有返回 true 和 false 的方法，但可以通过对 match 或者 search 方法的返回值是否是 None 来判断

2. 对于正则表达式的搜索功能，如果只搜索一次可以使用 search 或者 match 方法返回的匹配对象得到，对于搜索多次可以使用 finditer 方法返回的可迭代对象来迭代访问

3. 对于正则表达式的替换功能，可以使用正则表达式对象的 sub 或者 subn 方法来实现，也可以通过 re 模块方法 sub 或者 subn 来实现，区别在于模块的 sub 方法的替换文本可以使用一个函数来生成

4. 对于正则表达式的分割功能，可以使用正则表达式对象的 split 方法，需要注意如果正则表达式对象有分组的话，分组捕获的内容也会放到返回的列表中

32. OS 模块：

OS 负责程序与操作系统的交互，sys 负责程序与 python 解释器的交互

Sys.path—PATH 环境变量， os.path—模块，提供方法

Os.name---输出字符串指示正在使用的平台:win-‘nt’;linux/unix: ‘posix;

Os.getcwd()—脚本运行目录

运行目录即程序开始执行的地方

工作目录即程序调用其他文件等时操作的目录

os.listdir()–返回指定目录下的所有文件和目录名的一个列表，未列出什么目录什么文件

注意：命令提示符下输入 tree 命令即可返回当前文件下的结构树如：

C:\Users\xutao>tree

或： os.system(‘tree’)

用 quit()可退出 python 解释器

Os.listdir(‘参数’)—参数中\要用转移字符（加 r）:“\\ ”--参数即是地址，用引号括起来

os.remove(‘file_name’)—删除指定文件

os.rmdir(‘dir_name’)—删除指定目录

os.mkdir(‘dirname’)—创建目录

os.makedirs(‘a\b\c’)—递归创建目录

os.system(‘Is’)—执行 shell 命令

os.chdir(‘filename’)—改变工作目录

os.chmod(‘filename’)—改变文件或目录的权限

dir(os)

33.os.path 模板：

os.path.abspath(‘filename’)—返回文件或目录的绝对路径，不会检验存在否

os.path.split(‘filen_path’)—将路径分为目录和文件名，用一个元组返回

os.path.basename(‘path’)—返回路径最后的文件名

os.path.exists(‘fileopath’)—如果路径存在返回 true 反之返回 false

os.path.join(‘file_path’,’file_name’)—路径拼接，不会判断存在合法否

os.path.isdir(‘name’—判断是否为目录 bool

os.path.isfile(‘file’)—

os.path.islink(‘name’)—判断是否为链接

os.path.getsize(‘path’)—返回文件大小，不存在则返回错误

33.datetime 模板：

Import datetime

datetime.datetime.now()—查看当前时间

34.. 数字类型 Number：

不允许改变的,这就意味着如果改变数字数据类型的值，将重新分配内存空间。
del 语句删除一些数字对象的引用

math: 需要导入 math 模板

abs(x)	返回数字的绝对值，如 abs(-10) 返回 10
ceil(x)	返回数字的上入整数，如 math.ceil(4.1) 返回 5
exp(x)	返回 e 的 x 次幂(e^x),如 math.exp(1) 返回 2.718281828459045
fabs(x)	返回数字的绝对值，如 math.fabs(-10) 返回 10.0
floor(x)	返回数字的下舍整数，如 math.floor(4.9)返回 4
log(x)	如 math.log(math.e)返回 1.0,math.log(100,10)返回 2.0
log10(x)	返回以 10 为基数的 x 的对数，如 math.log10(100)返回 2.0
max(x1, x2,...)	返回给定参数的最大值，参数可以为序列。
min(x1, x2,...)	返回给定参数的最小值，参数可以为序列。
modf(x)	返回 x 的整数部分与小数部分，两部分的数值符号与 x 相同，整数部分以浮点型表示。
pow(x, y)	x**y 运算后的值。
round(x [,n])	返回浮点数 x 的四舍五入值，如给出 n 值，则代表舍入到小数点后的位数。

sqrt(x) 下面要导入：random 模块	返回数字 x 的平方根。
choice(seq)	从序列的元素中随机挑选一个元素，比如 random.choice(range(10))，从 0 到 9 中随 个整数。
randrange ([start,] stop [,step])	从指定范围内，按指定基数递增的集合中获取一个随机数，基数缺省值为 1
random()	随机生成下一个实数，它在[0,1)范围内。
seed([x])	改变随机数生成器的种子 seed。如果你不了解其原理，你不必特别去设定 seed，Pyt 你选择 seed。
shuffle(lst)	将序列的所有元素随机排序
uniform(x, y)	随机生成下一个实数，它在[x,y]范围内。

三角函数：math

acos(x)	返回 x 的反余弦弧度值。
asin(x)	返回 x 的反正弦弧度值。
atan(x)	返回 x 的反正切弧度值。
atan2(y, x)	返回给定的 X 及 Y 坐标值的反正切值。
cos(x)	返回 x 的弧度的余弦值。
hypot(x, y)	返回欧几里德范数 sqrt(x*x + y*y)。
sin(x)	返回的 x 弧度的正弦值。
tan(x)	返回 x 弧度的正切值。
degrees(x)	将弧度转换为角度,如 degrees(math.pi/2) ， 返回 90.0
radians(x)	将角度转换为弧度

Python 可以如此初始变量

```
a, b = 0, 1
while b < 1000:
    print(b, end=',')
    a, b = b, a+b
```

迭代是 Python 最强大的功能之一，是访问集合元素的一种方式。

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。迭代器只能往前不会后退。

迭代器有两个基本的方法：**iter()** 和 **next()**。

字符串，列表或元组对象都可用于创建迭代器

```
it = iter(list)      # 创建迭代器对象
print (next(it))     # 输出迭代器的下一个元素
```

在 Python 中，使用了 yield 的函数被称为生成器（generator）。**生成器导入 sys**

跟普通函数不同的是，生成器是一个返回迭代器的函数，只能用于迭代操作，更简单点理解生成器就是一个迭代器。

在调用生成器运行的过程中，每次遇到 yield 时函数会暂停并保存当前所有的运行信息，返回 yield 的值，并在下一次执行 next() 方法时从当前位置继续运行。

调用一个生成器函数，返回的是一个迭代器对象。

如果要修改嵌套作用域（enclosing 作用域，外层非全局作用域）中的变量则需要 nonlocal 关键字了

global 适用于函数内部修改全局变量的值

nonlocal 适用于嵌套函数中内部函数修改外部变量的值

35. 输出格式美化

- **str()：** 函数返回一个用户易读的表达形式。
- **repr()：** 产生一个解释器易读的表达形式。

函数可以转义字符串中的特殊字符，即不会执行'\n\...

```
repr() 的参数可以是 Python 的任何对象
字符串对象的 rjust(n) 方法，它可以将字符串靠右，并在左边填充空格,宽度 n
方法 zfill()，它会在数字的左边填充 0
print(repr(x).rjust(2), repr(x*x).rjust(3), end=' ')
print('{0:2d} {1:3d} {2:4d}'.format(x, x*x, x*x*x))

'!a' (使用 ascii()), '!s' (使用 str()) 和 '!r' (使用 repr()) 可以用于在格式化某个值之前对其进行转化
print('常量 PI 的值近似为： {!r}。'.format(math.pi))
可选项 ':' 和格式标识符可以跟着字段名。 这就允许对值进行更好的格式化
print('常量 PI 的值近似为 {0:.3f}。'.format(math.pi))
print('{0:10} ==> {1:10d}'.format(name, number))


文件通配符: glob 模块提供了一个函数用于从目录通配符搜索中生成文件列表
glob.glob('*.py')
```

访问 互联网

有几个模块用于访问互联网以及处理网络通信协议。其中最简单的两个是用于处理从 urls 接收的数据的 **urllib.request** 以及用于发送电子邮件的 smtplib:

数据压缩

以下模块直接支持通用的数据打包和压缩格式：zlib，gzip，bz2，zipfile，以及 tarfile。

测试模块

开发高质量软件的方法之一是为每一个函数开发测试代码，并且在开发过程中经常进行测试

doctest 模块提供了一个工具，扫描模块并根据程序中内嵌的文档

字符串执行测试。

测试构造如同简单的将它的输出结果剪切并粘贴到文档字符串中。

通过用户提供的例子，它强化了文档，允许 doctest 模块确认代码的结果是否与文档一致: def average(values):

```
    """Computes the arithmetic mean of a list of numbers.
    >>> print(average([20, 30, 70]))
    40.0
    """
    return sum(values) / len(values)


import doctest
print(doctest.testmod()) # 自动验证嵌入测试
```

unittest 模块不像 doctest 模块那么容易使用，不过它可以在一个独立的文件里提供一个更全面的测试集

36.decimal.

十进制数学计算

decimal.getcontext().prec=x 默认是 28 位

精度值的修改只在运算中才会体现出来

setcontext()来一次性设置 context

```
decimal.getcontext().prec=6
print(Decimal(1)/Decimal(3))
```

可以传递给 Decimal 整型或者字符串参数，但不能是浮点数据，因为浮点数据本身就不准确

用 Decimal(小数)的方式对浮点数进行操作无误差

Decimal()的构造中如果是小数或字符的话，需要加上单引号；如果为整数，则不需要

quantize()，当我们希望在运算过程中保持较高的精度，而在结果中以某种方式保留几位小数时可以用这个函数

```
def decimal_retain(number, ndigits=0):
    context = decimal.getcontext()
    context.rounding = decimal.ROUND_UP
    return round(Decimal(str(number)), ndigits)
```

可将浮点数精准保留 n 位，向上舍入

Dict.items() –字典里的数据：

For k , v in dict.items() if k ==100

关键字参数：函数调用时将形参写出来： 形参=value ，好处 ：可以不按顺序写

多参数： *args 位置参数 **kargs 关键字参数

装饰器 可以将函数赋值给另一个函数

类

class person: 定义一个类

def __init__(self, name，age):

self._name=name 实例化对象的属性，

注：该对象属性名称前**加下划线表示该属性为私有，不可用户访问**

```
        self._age =age
    def getname(self)                    类方法
        return self._name
xutao= person(‘xurao’,21)                创建一个对象
```

类的继承：

```
Class student(person):
    Pass
```

爬虫

主要库： requests　获取页面内容

bs4 –BeautifulSoup　分析页面信息

数据处理

库：numpy pandas,

绘图：matplotlib

数组：只能存储一种数据类型（相对列表）

Import array

更多用在 numpy

Numpy:

Import numpy as np (别名)

List_1=list(range10)

B=np.array(liast_1)

type(b): numpy.ndaarray

b.np.zeros(10,dtype=int)

a.dtype

b.npzeros((4,5),dtype=int) two demon

np.ones((4,5),dtype=int)

np.full((3,4), 3.14)　这个可以赋值 ， ones ,zeros 不可以，已经赋值

np.zeros_like(b)

np.ones_likes(b)

np.full_like(b, newvalue,dtype=int)

```
numpy.array(object, dtype = None, copy = True, order = None, subok = False,
ndmin = 0)
```

1.	object 任何暴露数组接口方法的对象都会返回一个数组或任何(嵌套)序列。
2.	dtype 数组的所需数据类型，可选。
3.	copy 可选，默认为 true，对象是否被复制。
4.	order C(按行)、F(按列)或 A(任意，默认)。
5.	subok 默认情况下，返回的数组被强制为基类数组。 如果为 true，则返回子类
6.	ndimin 指定返回数组的最小维数。

数据类型：

1.	bool_存储为一个字节的布尔值(真或假)
2.	int_默认整数，相当于 C 的 long，通常为 int32 或 int64
3.	intc 相当于 C 的 int，通常为 int32 或 int64
4.	intp 用于索引的整数，相当于 C 的 size_t，通常为 int32 或 int64

5.	int8 字节(-128 ~ 127)
6.	int1616 位整数(-32768 ~ 32767)
7.	int3232 位整数(-2147483648 ~ 2147483647)
8.	int6464 位整数(-9223372036854775808 ~ 9223372036854775807)
9.	uint88 位无符号整数(0 ~ 255)
10.	uint1616 位无符号整数(0 ~ 65535)
11.	uint3232 位无符号整数(0 ~ 4294967295)
12.	uint6464 位无符号整数(0 ~ 18446744073709551615)
13.	float_float64 的简写
14.	float16 半精度浮点：符号位，5 位指数，10 位尾数
15.	float32 单精度浮点：符号位，8 位指数，23 位尾数
16.	float64 双精度浮点：符号位，11 位指数，52 位尾数
17.	complex_complex128 的简写
18.	complex64 复数，由两个 32 位浮点表示(实部和虚部)
19.	complex128 复数，由两个 64 位浮点表示(实部和虚部)

NumPy 数字类型是 dtype(数据类型)对象的实例，每个对象具有唯一的特征

数组属性：

a = np.array([[1,2,3],[4,5,6]])

print a.shape

shape 函数返回数组的维度（元组）

a = np.array([[1,2,3],[4,5,6]])

a.shape = (3,2)

调整数组维度

b = a.**reshape**(3,2)　同理

ndarray.ndim 这一数组属性返回数组的维数

a = np.arange(24) 等间隔数字的数组

numpy.flags

ndarray 对象拥有以下属性。这个函数返回了它们的当前值。

序号	属性及描述
1.	C_CONTIGUOUS (C) 数组位于单一的、C 风格的连续区段内
2.	F_CONTIGUOUS (F) 数组位于单一的、Fortran 风格的连续区段内
3.	OWNDATA (O) 数组的内存从其它对象处借用

序号	属性及描述
4.	WRITEABLE (W) 数据区域可写入。 将它设置为 flase 会锁定数据，使其
5.	ALIGNED (A) 数据和任何元素会为硬件适当对齐
6.	UPDATEIFCOPY (U) 这个数组是另一数组的副本。当这个数组释放时，

numpy.empty

它创建指定形状和 dtype 的未初始化数组。 它使用以下构造函数：

numpy.empty(shape, dtype = float, order = 'C')

Order 'C'为按行的 C 风格数组，'F'为按列的 Fortran 风格数组

numpy.asarray

此函数类似于 numpy.array，除了它有较少的参数。 这个例程对于将

Python 序列转换为 ndarray 非常有用。

numpy.asarray(a, dtype = None, order = None)

1.	a 任意形式的输入参数，比如列表、列表的元组、元组、元组的元组、元组
2.	dtype 通常，输入数据的类型会应用到返回的 ndarray
3.	order 'C'为按行的 C 风格数组，'F'为按列的 Fortran 风格数组

numpy.frombuffer

此函数将缓冲区解释为一维数组。 暴露缓冲区接口的任何对象都用作参数来返回 ndarray。

numpy.frombuffer(buffer, dtype = float, count = -1, offset = 0)

构造器接受下列参数：

序号	参数及描述
1.	buffer 任何暴露缓冲区借口的对象
2.	dtype 返回数组的数据类型，默认为 float
3.	count 需要读取的数据数量，默认为-1，读取所有数据
4.	offset 需要读取的起始位置，默认为 0

numpy.fromiter

此函数从任何可迭代对象构建一个 ndarray 对象，返回一个新的一维数组。

numpy.fromiter(iterable, dtype, count = -1)

构造器接受下列参数：

序号	参数及描述
1.	iterable 任何可迭代对象
2.	dtype 返回数组的数据类型
3.	count 需要读取的数据数量，默认为-1，读取所有数据

切片：

a = np.arange(10)

s = slice(2,7,2)

通过将由冒号分隔的切片参数(start:stop:step)直接提供给 ndarray 对象，也可以获得相同的结果

b = a[2:7:2]

如果只输入一个参数，则将返回与索引对应的单个项目。 如果使用 a:，则从该索引向后的所有项目将被提取。 如果使用两个参数(以:分隔)，则对两个索引(不包括停止索引)之间的元素以默认步骤进行切片。

切片还可以包括省略号(...)，来使选择元组的长度与数组的维度相同。 如果在行位置使用省略号，它将返回包含行中元素的 ndarray。

示例 7

高级索引：

x = np.array([[1, 2], [3, 4], [5, 6]])

y = x[[0,1,2], [0,1,0]]

y: [1 4 5] 结果包括数组中(0,0)，(1,1)和(2,0)位置处的元素

广播：

广播是指 NumPy 在算术运算期间处理不同形状的数组的能力。 对数组的算术运算通常在相应的元素上进行。 如果两个阵列具有完全相同的形状，则这些操作被无缝执行。

```
a = np.array([1,2,3,4])
b = np.array([10,20,30,40])
c = a * b
```

两个数组的维数不相同，则元素到元素的操作是不可能的

如果满足以下规则，可以进行广播：

- ndim 较小的数组会在前面追加一个长度为 1 的维度。
- 输出数组的每个维度的大小是输入数组该维度大小的最大值。
- 如果输入在每个维度中的大小与输出大小匹配，或其值正好为 1，则在计算中可它。
- 如果输入的某个维度大小为 1，则该维度中的第一个数据元素将用于该维度的所有计算。

如果上述规则产生有效结果，并且满足以下条件之一，那么数组被称为**可广播的**。

- 数组拥有相同形状。
- 数组拥有相同的维数，每个维度拥有相同长度，或者长度为 1。
- 数组拥有极少的维度，可以在其前面追加长度为 1 的维度，使上述条件成立。

Np.random.randn(100)　-0-1 之间的值

Pandas

Import pandas as pd
Df = pd.read_csv{‘address’}　路径中不能有中文；
众多参数：
pandas.read_csv(filepath_or_buffer, sep=', ', delimiter=None, header='infer', names=N

one, index_col=None, usecols=None, squeeze=False, prefix=None, mangle_dupe_cols=True, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skipinitialspace=False, skiprows=None, nrows=None, na_values=None, keep_default_na=True, na_filter=True, verbose=False, skip_blank_lines=True, parse_dates=False, infer_datetime_format=False, keep_date_col=False, date_parser=None, dayfirst=False, iterator=False, chunksize=None, compression='infer', thousands=None, decimal=b'.', lineterminator=None, quotechar='"', quoting=0, escapechar=None, comment=None, encoding=None, dialect=None, tupleize_cols=False, error_bad_lines=True, warn_bad_lines=True, skipfooter=0, skip_footer=0, doublequote=True, delim_whitespace=False, as_recarray=False, compact_ints=False, use_unsigned=False, low_memory=True, buffer_lines=None, memory_map=False, float_precision=None)
Df.head(first n hang)

DataFrame
df.columns 列名
df.index
df.loc(index)
筛选：
Df.数学 >120 ----输出的是布尔类型
Df [Df.数学>130]---输出满足要求的整行
多个条件： 用与运算& ---注意：要用括号
排序：
df.sort_values(['数学','语文']).head(n)
第一个相同 按第二个排序，排出前 n 名

将字典格式化：
dict={"a":[1,2],'b':[3,4]}
d=pd.DataFrame(dict,index=['one','two'],columns=)
print(d)
选取多行数据的某几项：
df[['数学','语文']].head(10)

按数据添加项目：
df['数学分类'] = df.数学.map(func) #执行函数的同时会添加到原数据中
print(df.数学分类)
对所有数据进行操作：
print(df.applymap(lambda x:str(x)+"-"))

产生一个新的列：
Df['total'] = df.apply(lambda x:x.数学+x.语文,axis=1) axis –列

Matplotlib 绘图

Matplotlib 是 Python 绘图库，可以与 numpy 一起使用，也可以和 pyqt 等图形工具一起使用

Import matplotlib.pyplot as plt

(%matpoltlib inline) -ipython

（from matplotlib import pyplot as plt）

plt.title()

plt.xlabel()

plt.ylabel()

plt.scatter()

plt.ylim(-0.5,1) -限定范围

plt.xlim()

plt.plot(x, y, 'ob',**color = 'red', label='name'， markersize=9,linewidth=8**)

(plot 含大量参数)

plt.legend()用于控制 label 的显示效果，如 loc 控制位置

注意：用如 help（plt . legend ）的形式查看帮助文档

plt.show()

作为线性图的替代，可以通过向 plot()函数添加格式字符串来显示离散值。 可以使用以下格式化字符。

'-'	实线样式
'--'	短横线样式
'-.'	点划线样式
':'	虚线样式
'.'	点标记

','	像素标记
'o'	圆标记
'v'	倒三角标记
'^'	正三角标记
'<'	左三角标记
'>'	右三角标记
'1'	下箭头标记
'2'	上箭头标记
'3'	左箭头标记
'4'	右箭头标记
's'	正方形标记
'p'	五边形标记
'*'	星形标记
'h'	六边形标记 1
'H'	六边形标记 2
'+'	加号标记
'x'	X 标记
'D'	菱形标记
'd'	窄菱形标记
'|'	竖直线标记
'_'	水平线标记

还定义了以下颜色缩写。

字符	**颜色**
'b'	蓝色
'g'	绿色
'r'	红色
'c'	青色
'm'	品红色
'y'	黄色
'k'	黑色

字符	颜色
'w'	白色

要显示圆来代表点，而不是上面示例中的线，请使用 ob 作为 plot()函数中的格式字符串

subplot()函数允许你在同一图中绘制不同的东西

```
# 建立 subplot 网格，高为 2，宽为 1
# 激活第一个 subplot
plt.subplot(2,  1,  1)
# 绘制第一个图像
plt.plot(x, y_sin)
```

bar()

pyplot 子模块提供 bar()函数来生成条形图

```
x =  [5,8,10]
y =  [12,16,6]
x2 =  [6,9,11]
y2 =  [6,15,7]
plt.bar(x, y, align =  'center')      y=f(x)
plt.bar(x2, y2, color =  'g', align =  'center')
```

直方图： **plt.hist()**

NumPy 有一个 numpy.histogram()函数，它是数据的频率分布的图形表示。 水平尺寸相等的矩形对应于类间隔，称为 bin，变量 height 对应于频率。

umpy.histogram()函数将输入数组和 bin 作为两个参数。 bin 数组中的连续元素用作每个 bin 的边界。

```
from matplotlib import pyplot as plt
import numpy as np
a = np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])  一个数组
plt.hist(a, bins =  [0,20,40,60,80,100])  bins 数组表示分布轴
plt.show()
```

```
plt.style.use('classic')
x = np.random.randn(100)
y = np.random.randn(100)
colors = np.random.randn(100)
size = 1000*np.random.rand(100)
plt.scatter(x,y,c=colors,s=size,alpha=0.5)
plt.colorbar()
```

Sklearn

Train_test_spilt() 函数

作用：交叉验证，随机划分训练集 ，测试集，从样本中随机的按比例选取 train_data 和 test_data

形式为：

X_train,X_test,y_train,y_test

=cross_validation.train_test_split(train_data,train_target,test_size=0.4, random_state=0)

test_size：样本占比，如果是整数的话就是样本的数量

random_state：是随机数的种子。

随机数种子：其实就是该组随机数的编号，在需要重复试验的时候，保证得到一组一样的随机数。比如你每次都填 1，其他参数一样的情况下你得到的随机数组是一样的。但填 0 或不填，每次都会不一样。

随机数的产生取决于种子，随机数和种子之间的关系遵从以下两个规则：

种子不同，产生不同的随机数；种子相同，即使实例不同也产生相同的随机数。

Python 基础知识

字符串转整数：float（str）

取小数位数：round(num, n)

split:

str.split(str="", num=string.count(str)).

str -- 分隔符，默认为所有的空字符，包括空格、换行(\n)、制表符(\t)等。

num -- 分割次数。默认为 -1, 即分隔所有。

```
print(i+" ",end="")
```

三元运算符
res = 值 1 if 条件 else 值 2

+:
作用 1：数学运算
作用 2：字符串连接
所以不能使用 字符串+数字模式（不能正确解释）

arr.reverse() 无返回值

字符串反转：
1.result = s[::-1]
2.l = list(s)
l.reverse()
result = "".join(l)
3.result = reduce(lambda x,y:y+x,s)

List ——》String：
"分隔符".join(list)

is 用于判断两个变量引用对象是否为同一个， == 用于判断引用变量的值是否相等。

"s".join(seq)注意：seq 可以使数组、元组、列表、但是必须是字符串

指定小数位数：
print('{:.2f} {:.2f} {:.2f}'.format(A,B,C))
'%.2f'%3.1415926

去除字符串前端空格：
"str".lstrip()
去除字符串后端空格：
"str".rstrip()
去除字符串前端空格：
"str".strip()

```
a = [1,2,3]
for i in a:
    i=1
```
对 i 的操作不会改变原数组的值

```
try:
    while True:
```

```
            inp = input()
            print(inp)
except EOFError:
    pass
```

eval(s)将字符串自动转化成 Python 类型：
s = {"A":1}
就会自动转化为字典

```
# print(b&c)
# print(b|c)
# print(b^c)
```
~b 取反

sum(集合)——计算和

int("任何进制数字"，进制)

普通闰年:能被 4 整除但不能被 100 整除的年份为普通闰年。(如 2004 年就是闰年，1999 年不是闰年)；
世纪闰年:能被 400 整除的为世纪闰年。(如 2000 年是世纪闰年，1900 年不是世纪闰年)；

except (SyntaxError , ValueError):（多个异常）

re.split(pattern, string[, maxsplit=0, flags=0]) ——正则

Python 支持无限位数运算，不管有多大

re-findall:

```
import re
pat = re.compile("[A-Za-z]+")
X = pat.findall("str")
```

从以上可以看出，使用 a[:], list(a), a*1, copy.copy(a)四种方式复制列表结果都可以得到一个新的列表，但是如果列表中含有列表，所有 b, c, d, e 四个新列表的子列表都是指引到同一个对象上。只有使用 copy.deepcopy(a)(import copy)方法得到的新列表 f 才是包括子列表在内的完全复制。

map 函数
内置高阶函数，接受一个函数 f 和一个 List，并通过把函数 f 依次作用在 List 的每个元素上，得到一个新的 List 并返回（并未改变原来的 List）
f 处还可以使用类型代替：

a = list(map(float,eval(input())))

容器：
列表：[ ]
元组：（）
字典：{key:value}
集合：{a,b,c}

集合操作：
a = t | s　　#t 和 s 的并集
b = t & s　　#t 和 s 的交集
c = t - s　　# 求差集（项在 t 中，但不在 s 中）
d = t ^ s　　# 对称差集（项在 t 或 s 中，但不会同时出现在二者中）
集合函数：
add()——添加一项
update()——添加多项
remove()
s.issubset(t)——检查是否子集
s.isupperset(t)——检查是否父集
s.union(t)——相当于交集
s.intersection(t)——相当于交集
s.difference(t)——相当于差集
s.symmetric_differeence()——对称差集
s.copy()——浅复制（只复制了地址）
clear()
s.discard(x)——如果 x 在 s 中，则删除

sorted()函数：
sorted(iterable,key,reverse)
iterable：可迭代的对象，例如可以是 dict.items()、dict.keys()等
key：一个函数，用来选取参与比较的元素
reverse：指定排序是倒序还是顺 序，rtrue 则是倒序，false 时则是顺序，默认时 reverse=false。
返回值：返回列表中套用的元组
eg: 按降序反转字典：
a = dict(eval(input()))
b = {value:key for key,value in a.items()}
c = sorted(b.items(),key=lambda item:item[1],reverse=True)
d = dict(c)

math.factorial（）阶乘
math.fabs()绝对值

statistics 库：

Calculating averages

Function	Description
mean	Arithmetic mean (average) of data.
harmonic_mean	Harmonic mean of data.
median	Median (middle value) of data.
median_low	Low median of data.
median_high	High median of data.
median_grouped	Median, or 50th percentile, of grouped data.
mode	Mode (most common value) of data.

Calculating variability or spread

Function	Description
pvariance	Population variance of data.
variance	Sample variance of data. 方差
pstdev	Population standard deviation of data.
stdev	Sample standard deviation of data. 标准差

datetime

注意：

from datetime import datetime

（模块） （模块中的类）

dt = datetime(2015, 4, 19, 12, 20) # 用指定日期时间创建 datetime

在计算机中，时间实际上是用数字表示的。我们把 1970 年 1 月 1 日 00:00:00 UTC+00:00 时区的时刻称为 epoch time，记为 0（1970 年以前的时间 timestamp 为负数），当前时间就是相对于 epoch time 的秒数，称为 timestamp。可见 timestamp 的值与时区毫无关系，因为 timestamp 一旦确定，其 UTC 时间就确定了，转换到任意时区的时间也是完全确定的，这就是为什么计算机存储的当前时间是以 timestamp 表示的，因为全球各地的计算机在任意时刻的 timestamp 都是完全相同的（假定时间已校准）。

```
>>> dt = datetime(2015, 4, 19, 12, 20) # 用指定日期时间创建 datetime
>>> dt.timestamp() # 把 datetime 转换为 timestamp
1429417200.0
```

timestamp 转换为 datetime

要把 timestamp 转换为 datetime，使用 datetime 提供的 fromtimestamp()方法

datetime 加减

对日期和时间进行加减实际上就是把 datetime 往后或往前计算，得到新的 datetime。加减可以直接用+和-运算符，不过需要导入 timedelta 这个类：

from datetime import datetime, timedelta

str 转换为 datetime
很多时候，用户输入的日期和时间是字符串，要处理日期和时间，首先必须把 str 转换为 datetime。转换方法是通过 datetime.strptime()实现，需要一个日期和时间的格式化字符串：

```
>>> from datetime import datetime
>>> cday = datetime.strptime('2015-6-1 18:19:59', '%Y-%m-%d %H:%M:%S')
>>> print(cday)
2015-06-01 18:19:59
```

注意是： strptime

namedtuple:
元组的升级版本——具名元组
内存消耗和普通的元组一样多
格式：
collectionois.namedtuple(typename, field_names, verbose=false, rename=false)
field_names: 元组中元素的名称。
rename：如果元素名称中有 Python 的关键字，则必须设置为 true

```
#User = collections.namedtuple('User', ['name', 'age', 'id'])
User = collections.namedtuple('User', 'name age id')
user = User('tester', '22', '464643123')
```

Python 中 abs()和 math.fabs()区别
abs()是一个内置函数，而 fabs()在 math 模块中定义的。
fabs()函数只适用于 float 和 integer 类型，而 abs()也适用于复数。
abs()返回是 float 和 int 类型，math.fabs()返回是 float 类型
eg: 取绝对值，输入可能为 int float 复数

```
print(abs(eval(input())))
```

//一条语句就搞定，如果你不知道的话，可能要写几十行，卧槽

split()
不加参数时，默认以空格分，空格数量不影响，末尾空格舍去，强大

一个 python 对象可能拥有两个属性，__class__ 和 __base__，__class__ 表示这个对象是谁创建的，__base__ 表示一个类的父类是谁。

isinstance() 与 type() 区别：
type() 不会认为子类是一种父类类型，不考虑继承关系
isinstance() 会认为子类是一种父类类型，考虑继承关系

class type(name, bases, dict)
>>> type(1)==int
True

issubclass(class, classinfo)
如果 class 是 classinfo 的子类（直接、间接或 虚拟 的），则返回 true。classinfo 可以是类对象的元组，此时 classinfo 中的每个元素都会被检查。其他情况，会触发 TypeError 异常。
ascii(a)
返回一个可打印的对象字符串方式表示，如果是非 ascii 字符就会输出\x, \u 或\U 等字符来表示
repr() 返回的字符串中非 ASCII 编码的字符

eval 可以直接解析表达式
a = input() #1+2
print(eval(a)) #3

enumerate(iterable, start=0)
返回一个枚举对象。iterable 必须是一个序列，或 iterator，或其他支持迭代的对象。 enumerate() 返回的迭代器的 __next__() 方法返回一个元组，里面包含一个计数值（从 start 开始，默认为 0）和通过迭代 iterable 获得的值。
>>> seasons = ['Spring', 'Summer', 'Fall', 'Winter']
>>> list(enumerate(seasons))
[(0, 'Spring'), (1, 'Summer'), (2, 'Fall'), (3, 'Winter')]
>>> list(enumerate(seasons, start=1))
[(1, 'Spring'), (2, 'Summer'), (3, 'Fall'), (4, 'Winter')]

any(iterable)
如果*iterable*的任一元素为真则返回``True``。如果迭代器为空，返回``False``。
等价于:
```
def any(iterable):
    for element in iterable:
        if element:
            return True
    return False
```

两个列表--->字典：

dict(zip(list1,list2))
嵌套列表转字典：
2、将嵌套列表转为字典，有两种方法，
>>>new_list= [['key1','value1'],['key2','value2'],['key3','value3']]
>>>dict(list)

zip:
1.合并：
>>>a = [1,2,3]
>>> b = [4,5,6]
>>> zipped = zip(a,b)　　# 打包为元组的列表
[(1, 4), (2, 5), (3, 6)]
2.拆解
a = [(1, 4), (2, 5), (3, 6)]
x,y=zip(*a)
　　x= [1,2,3]
　　y = [4,5,6]

callable(object)
如果实参 object 是可调用的，返回 True，否则返回 False。如果返回真，调用仍可能会失败；但如果返回假，则调用 object 肯定会失败。注意类是可调用的（调用类会返回一个新的实例）。如果实例的类有 __call__() 方法，则它是可调用。

setattr(object, name, value)
此函数与 getattr() 两相对应。 其参数为一个对象、一个字符串和一个任意值。字符串指定一个现有属性或者新增属性。 函数会将值赋给该属性，只要对象允许这种操作。 例如，setattr(x, 'foobar', 123) 等价于 x.foobar = 123。

__repr__()方法：显示属性
定义后，使用 print(该对象）即可输出信息

None 是一种数据类型

Counter:
Counter 类的目的是用来跟踪值出现的次数。它是一个无序的容器类型，以字典的键值对形式存储，其中元素作为 key，其计数作为 value。计数值可以是任意的 Interger（包括 0 和负数）。

创建:
四种方法：

```
>>> c = Counter()  # 创建一个空的 Counter 类
>>> c = Counter('gallahad')  # 从一个可 iterable 对象（list、tuple、dict、字符串等）创建
>>> c = Counter({'a': 4, 'b': 2})  # 从一个字典对象创建
>>> c = Counter(a=4, b=2)  # 从一组键值对创建
```

使用：
当所访问的键不存在时，返回 0，而不是 KeyError；否则返回它的计数。

```
c=Counter("abcdefgab")
c["a"]
```

更多见：
https://blog.csdn.net/screaming/article/details/51713615

sorted 按多个条件排序：前一个相同，则使用后一个
在 key= 使用多个排序函数：
sorted(dictRes.items(),key=lambda a:(a[1],-ord(a[0])),reverse=True)

lambda a:(a[1],-ord(a[0]))——字典{str: int}
先按 value 大小排序
相同的按 key 的字典序排列

re
pat 如果使用变量设置匹配次数，使用+n+pat
re.match() 仅在开头处判断是否匹配
re.search() 搜索
注：
    match 和 search 一旦匹配成功，就返回一个 match 对象，match 对象有如下方法：
group() 返回被 RE 匹配的字符串
start() 返回匹配开始的位置
end() 返回匹配结束的位置
span() 返回一个元组包含匹配 (开始,结束) 的位置
group() 返回 re 整体匹配的字符串，可以一次输入多个组号，对应组号匹配的字符串。

Re.findAll() 返回列表
finditer() 搜索 string，返回一个顺序访问每一个匹配结果（Match 对象）的迭代器。找到 RE 匹配的所有子串，并把它们作为一个迭代器返回。

sub()：使用 re 替换 string 中每一个匹配的子串后返回替换后的字符串。
re.sub(pattern, repl, string, count)
re.sub 还允许使用函数对匹配项的替换进行复杂的处理
如：re.sub(r'\s', lambda m: '[' + m.group(0) + ']', text, 0)；将字符串中的空格' '替换为'[ ]'。

subn()：返回替换次数
subn(pattern, repl, string, count=0, flags=0)

abs() 是一个内置函数，而 fabs() 在 math 模块中定义的。
fabs() 函数只适用于 float 和 integer 类型，而 abs() 也适用于复数。

random 模块：
import random as r
r.randint(a,b):产生 a-b 的随机整数

排列组合：
>> from scipy.special import comb, perm
>> perm(3, 2)
6.0
>> comb(3, 2)
3.0

Numpy:

Ndarray 对象：
N 维数组对象，同类型数据集合
ndarray 内部由以下内容组成：
1.一个指向数据（内存或内存映射文件中的一块数据）的指针。
2.数据类型或 dtype，描述在数组中的固定大小值的格子。
3.一个表示数组形状（shape）的元组，表示各维度大小的元组。
4.一个跨度元组（stride），其中的整数指的是为了前进到当前维度下一个元素需要"跨过"的字节数。

创建：

numpy.array(object, dtype = None, copy = True, order = None, subok = False, ndmin = 0)
object　数组或嵌套的数列
dtype　数组元素的数据类型，可选
copy　对象是否需要复制，可选
order　创建数组的样式，C 为行方向，F 为列方向，A 为任意方向（默认）
subok　默认返回一个与基类类型一致的数组
ndmin　指定生成数组的最小维度

指定函数参数类型：

```
def B(k: int, x):
    return

print('{:.4f} {:.4f}'.format(A(k, x), B(k, x)))
```

python 使用 pandas 来计算偏度和峰度

```
import pandas as pd
x = [53, 61, 49, 66, 78, 47]
s = pd.Series(x)
print(s.skew())
print(s.kurt())
```

生成 n 维随机数组：

```
import numpy as np
a=np.random.randint(0,10,size=[维度 1，维度 2...])
```

```
rint("ha1:")
13
print(" ",end="")
14
print(ha1)
15
# print(ha2)
16
```

```
va1,va2,va3 = np.vsplit(a,3)
17
print("va1:")
18
print(" ",end="")
19
print(va1)
20
?
21
b = np.hstack((ha1,ha2))
22
print("hstack:")
23
print(" ",end="")
24
print(b)
25
?
26
c = np.hstack((va2,va2))
27
print("vstack:")
28
print(" ",end="")
29
print(c)
30
#6:
31
print("a:")
32
print(" ",end="")
33
t = np.reshape(a,(1,12))
34
print(t)
```

rong Answer
期望结果为:
a:
[[11 12 7 6]
[10 10 16 8]
[13 11 0 4]]

```
ha1:
 [[11 12]
 [10 10]
 [13 11]]
va1:
 [[11 12  7  6]]
hstack:
 [[11 12  7  6]
 [10 10 16  8]
 [13 11  0  4]]
vstack:
 [[11 12  7  6]
 [10 10 16  8]]
a:
 [11 12  7  6 10 10 16  8 13 11  0  4]
```

你的结果为:

```
a:
 [[11 12  7  6]
 [10 10 16  8]
 [13 11  0  4]]
ha1:
 [[11 12]
 [10 10]
 [13 11]]
va1:
 [[11 12  7  6]]
hstack:
 [[11 12  7  6]
 [10 10 16  8]
 [13 11  0  4]]
vstack:
 [[10 10 16  8]
 [10 10 16  8]]
a:
 [[11 12  7  6 10 10 16  8 13 11  0  4]]
```

```
print("a:\n",t)
```

这种输出居然也可以