



Where are data from? How to represent data?

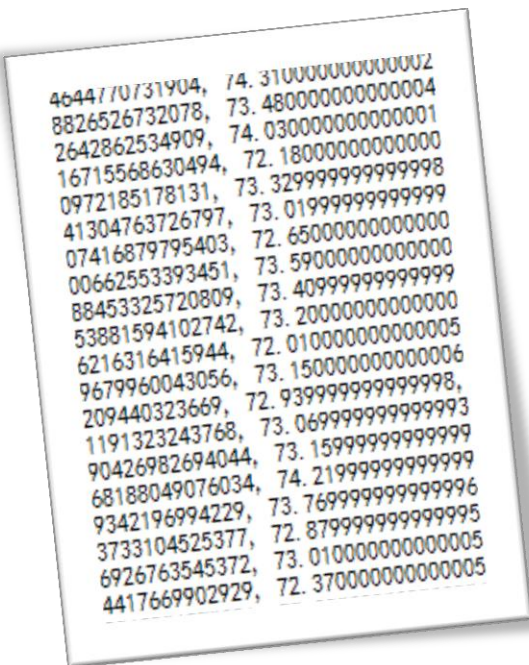
数据 获取与表示

Department of Computer Science and Technology
Department of University Basic Computer Teaching

用Python玩转数据

本地数据获取

用Python获取数据



本地数据如何获取？

文件的打开，读写和关闭

- 打开后才能进行读写
- 读文件
- 写文件
- 文件为什么需要关闭？

文件的打开



```
>>> f1 = open(r'd:\\infile.txt')
>>> f2 = open(r'd:\\outfile.txt', 'w')
>>> f3 = open('frecord.csv', 'ab', 0)
```

Python 3.x中的目录
路经常常直接用类似
“d:\\test.txt” 或
“d:/test.txt” 这样的
方式表示

file_obj = open(filename, mode='r', buffering=-1)

- mode为可选参数，默认值为r
- buffering也为可选参数，默认值为-1（0代表不缓冲，1或大于1的值表示缓冲一行或指定缓冲区大小）

open()函数-mode

Mode	Function
r	以读模式打开
rU or U	以读模式打开，并提供通用换行符支持
w	以写模式打开（清空原内容）
a	以追加模式打开（从EOF开始，必要时创建新文件）
r+	以读写模式打开
w+	以读写模式打开（清空原内容）
a+	以读和追加模式打开
rb	以二进制读模式打开
wb	以二进制写模式打开（参见w）
ab	以二进制追加模式打开（参见a）
rb+	以二进制读写模式打开（参见r+）
wb+	以二进制读写模式打开（参见w+）
ab+	以二进制读写模式打开（参见a+）

Python 3.x中不推荐
使用U模式

返回值

- `open()`函数返回一个文件（file）对象
- 文件对象可迭代
- 有关闭方法和许多读写相关的方法/函数
 - `f.read()`, `f.write()`, `f.readline()`, `f.readlines()`, `f.writelines()`
 - `f.close()`
 - `f.seek()`

- **file_obj.write(str)**
 - 将一个字符串写入文件



```
>>> f = open('firstpro.txt', 'w')
>>> f.write('Hello, World!')
>>> f.close()
```

```
firstpro.txt :
Hello, World!
```

读文件-f.read()

- **file_obj.read(size)**
 - 从文件中至多读出size字节数据，返回一个字符串
- **file_obj.read()**
 - 读文件直到文件结束，返回一个字符串



```
>>> f = open('firstpro.txt')
>>> p1 = f.read(5)
>>> p2 = f.read()
>>> print p1,p2
>>> f.close()
```

Output:
Hello, World!

其他读写函数



```
# Filename: companies_a.py  
f = open(r'companies.txt')  
cNames = f.readlines()  
print cNames  
f.close()
```

- file_obj.readlines()
- file_obj.readline()
- file_obj.writelines()

Output:

['GOOGLE Inc.\n', 'Microsoft Corporation\n', 'Apple Inc.\n',
'Facebook, Inc.']

文件读写例子



将文件companies.txt 的字符串前加上序号1、2、3、...后写到另一个文件scompanies.txt中。



```
# Filename: revcopy.py
f1= open(r'companies.txt')
cNames = f1.readlines()
for i in range(0,len(cNames)):
    cNames[i] = str(i+1) + ' ' + cNames[i]
f1.close()
f2 = open(r'scompanies.txt','w')
f2.writelines(cNames)
f2.close()
```

Output:

```
1 GOOGLE Inc.
2 Microsoft Corporation
3 Apple Inc.
4 Facebook, Inc.
```

其他文件相关函数

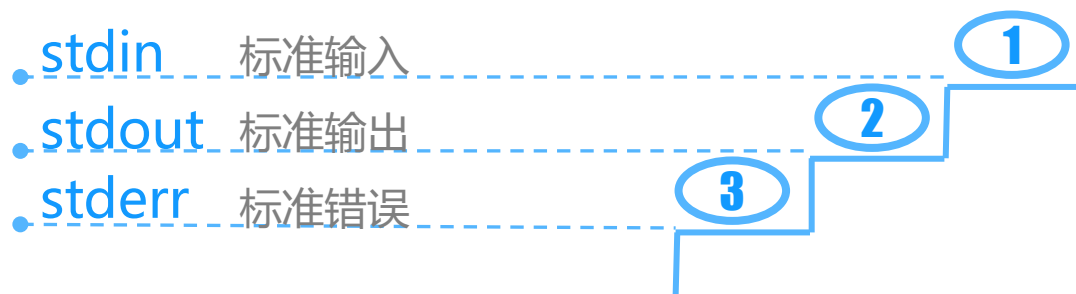


```
# Filename: companies_b.py
s = 'Tencent Technology Company Limited'
f = open(r'companies.txt', 'a+')
f.writelines('\n')
f.writelines(s)
f.seek(0,0)
cNames = f.readlines()
print cNames
f.close()
```

- **file_obj.seek(offset, whence=0)**
 - 在文件中移动文件指针，从 *whence* (0表示文件头部，1表示当前位置，2表示文件尾部) 偏移 *offset* 个字节
 - *whence* 参数可选，默认值为0

标准文件

- 当程序启动后，以下三种标准文件有效



```
>>> newcName = raw_input('Enter the name of new company: ')
Enter the name of new company: Alibaba
>>> print newcName
'Alibaba'
```

用Python玩转数据

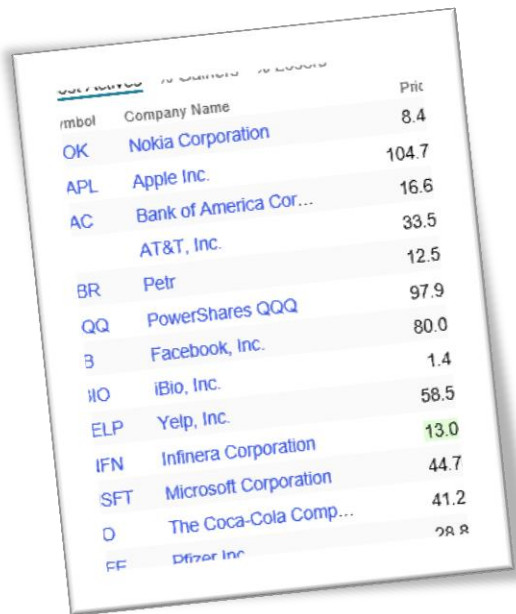
2

网络数据获取

用Python获取数据

网络数据如何获取？

抓取网页，解析网页内容



Symbol	Company Name	Price
OK	Nokia Corporation	8.4
APL	Apple Inc.	104.7
AC	Bank of America Cor...	16.6
	AT&T, Inc.	33.5
BR	Petr	12.5
QQ	PowerShares QQQ	97.9
B	Facebook, Inc.	80.0
IBIO	IBio, Inc.	1.4
ELP	Yelp, Inc.	58.5
IFN	Infinera Corporation	13.0
SFT	Microsoft Corporation	44.7
D	The Coca-Cola Comp...	41.2
FF	Difzer Inc	28.8

- urllib
 - urllib2
 - httplib
 - httplib2
- Python 3中被
urllib.request代替
- Python 3中被
http.client代替

利用urllib库获取网络数据

urllib — Open arbitrary resources by URL

Source

```
>>> import urllib
>>> r = urllib.urlopen('http://z.cn/')
>>> html = r.read()
>>> html
```

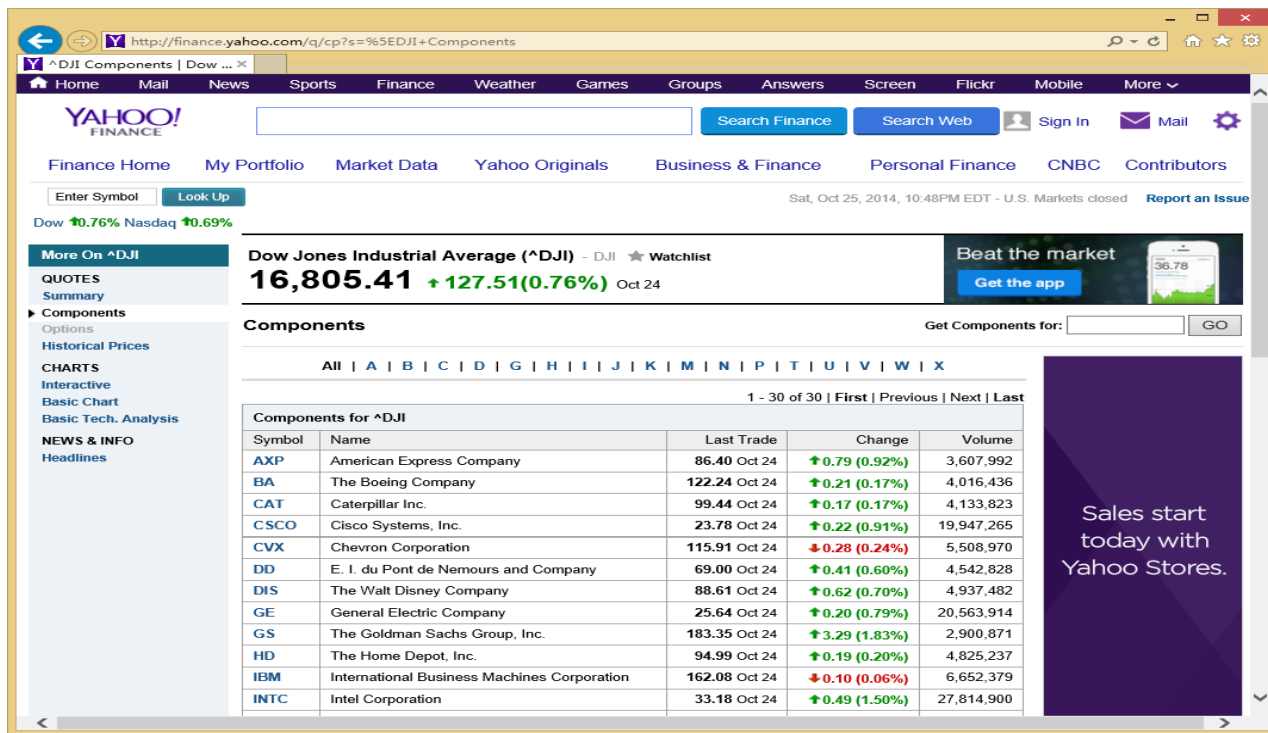
urllib

抓取网页，解析网页内容

- urllib.urlopen()
- f.read(), f.readline(), f.readlines(), f.close()等方法

urllib.urlopen在Python 3用urllib.request（即Python 2中的urllib2）中的urlopen方法代替

http://finance.yahoo.com/q/cp?s=%5EDJI+Component



利用urllib库获取yahoo财经数据(Python 2)

17



Filename: dji.py

import urllib2

import re

dStr = urllib2.urlopen('http://finance.yahoo.com/q/cp?s=%5EDJI+Components').read()

m = re.findall('<tr><td class=\'yfnc_tabledata1\'>
(.*?)</td><td class=\'yfnc_tabledata1\'>
(.*?)</td>.*?(.*?).*?</tr>', dStr)

if m:

print m

print '\n'

print len(m)

else:

print 'not match'

m = re.findall('<tr><td class="yfnc_tabledata1">
(.*?)</td><td class="yfnc_tabledata1">(.*?)</td>.*?
(.*?).*?</tr>', dStr) (2016年4月网站正则表达式更新)

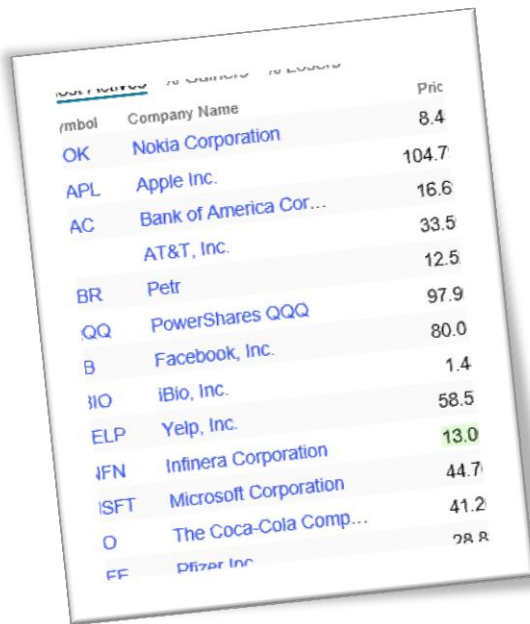
利用urllib库获取yahoo财经数据 (Python 3)

18



```
# Filename: dji.py
import urllib.request
import re
dBytes = urllib.request.urlopen('http://finance.yahoo.com/q/cp?s=%5EDJI+Components').read()
dStr = dBytes.decode('GBK') #在python3中urllib.read()返回bytes对象而非str，语句功能是将dStr转换成str
m = re.findall('<tr><td class="yfnc_tabledata1"><b><a href=".*?">(.*?)</a></b></td><td  
class="yfnc_tabledata1">(.*?)</td>.*?<b>(.*?)</b>.*?</tr>', dStr)
if m:
    print(m)
    print ('\n')
    print (len(m))
else:
    print ('not match')
```

- 包含多个字符串
 - 'AXP', 'American Express Company', '86.40'
 - 'BA', 'The Boeing Company', '122.24'
 - 'CAT', 'Caterpillar Inc.', '99.44 '
 - 'CSCO', 'Cisco Systems, Inc.', '23.78'
 - 'CVX', 'Chevron Corporation', '115.91'
 - ...



Symbol	Company Name	Price
OK	Nokia Corporation	8.4
APL	Apple Inc.	104.7
AC	Bank of America Cor...	16.6
	AT&T, Inc.	33.5
BR	Petr	12.5
QQ	PowerShares QQQ	97.9
B	Facebook, Inc.	80.0
IBIO	iBio, Inc.	1.4
ELP	Yelp, Inc.	58.5
IFN	Infinera Corporation	13.0
ISFT	Microsoft Corporation	44.7
O	The Coca-Cola Comp...	41.2
PF	Pfizer Inc.	28.8

用Python玩转数据

序列

- `str = 'Hello, World!'`
- `aList = [2, 3, 5, 7, 11]`
- `aTuple = ('Sunday', 'happy')`
- `pList = [('AXP', 'American Express Company', '86.40'),
('BA', 'The Boeing Company', '122.64'),
('CAT', 'Caterpillar Inc.', '99.44'),
('CSCO', 'Cisco Systems, Inc.', '23.78')
('CVX', 'Chevron Corporation', '115.91')]`



字符串

Strings



列表

Lists



元组

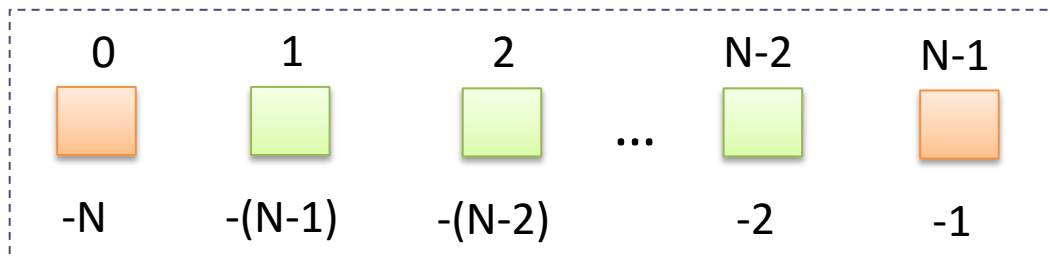
Tuples

Python中的序列

23

week	0	1	2	3	4	5	6
	'Monday'	'Tuesday'	'Wednesday'	'Thursday'	'Friday'	'Saturday'	'Sunday'
	-7	-6	-5	-4	-3	-2	-1

序列



访问模式

- 元素从0开始通过下标偏移量访问
- 一次可访问一个或多个元素

序列相关操作

标准
类型
运算符

值比较
对象身份比较
布尔运算

序列
类型
运算符

获取
重复
连接
判断

内建
函数

序列类型转换工厂函数
序列类型可用内建函数

标准类型运算符



```
>>> 'apple' < 'banana'
```

```
True
```

```
>>> [1,3,5] != [2,4,6]
```

```
True
```

```
>>> aTuple = ('BA', 'The Boeing Company', '122.64')
```

```
>>> bTuple = aTuple
```

```
>>> bTuple is not aTuple
```

```
False
```

```
>>> ('86.40' < '122.64') and ('apple' > 'banana')
```

```
False
```

标准类型运算符

值比较

<	>
<=	>=
==	
!=	<>

Python 3.x中
不再支持

对象身份比较

is
is not

布尔运算

not
and
or

序列类型运算符

Source

```
>>> week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
>>> print week[1], week[-2], '\n', week[1:4], '\n', week[:6], '\n', week[::-1]
Tuesday Saturday
['Tuesday', 'Wednesday', 'Thursday']
['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
['Sunday', 'Saturday', 'Friday', 'Thursday', 'Wednesday', 'Tuesday', 'Monday']
>>> 'apple' * 3
'appleappleapple'
>>> 'pine' + 'apple'
'pineapple'
>>> 'BA' in ('BA', 'The Boeing Company', '122.64')
True
```

序列类型运算符

`x in s`

`x not in s`

`s + t`

`s * n, n * s`

`s[i]`

`s[i:j]`

`s[i:j:k]`

序列类型转换工厂函数

list()

str()

unicode()

basestring()

tuple()



```
>>> list('Hello, World!')
['H', 'e', 'l', 'l', 'o', ',', ' ', 'W', 'o', 'r', 'l', 'd', '!']
>>> tuple("Hello, World!")
('H', 'e', 'l', 'l', 'o', ',', ' ', 'W', 'o', 'r', 'l', 'd', '!')
```

Python 3.x中移除了该函数，因为
其字符串已为Unicode字符串

序列类型可用内建函数

30

<code>enumerate()</code>	<code>reversed()</code>
<code>len()</code>	<code>sorted()</code>
<code>max()</code>	<code>sum()</code>
<code>min()</code>	<code>zip()</code>



```
>>> aStr = 'Hello, World!'
```

```
>>> len(aStr)
```

```
13
```

```
>>> sorted(aStr)
```

```
[' ', '!', ',', 'H', 'W', 'd', 'e', 'l', 'l', 'l', 'o', 'o', 'r']
```

用Python玩转数据

字符串



字符串的不同表示形式

- If = [('AXP', 'American Express Company', '86.40'), ('BA', 'The Boeing Company', '122.64'), ('CAT', 'Caterpillar Inc.', '99.44'), ('CSCO', 'Cisco Systems, Inc.', '23.78') ('CVX', 'Chevron Corporation', '115.91')]



```
>>> aStr = 'The Boeing Company'  
>>> bStr = "The Boeing Company"  
>>> cStr = """The Boeing  
company"""
```


字符串小例子



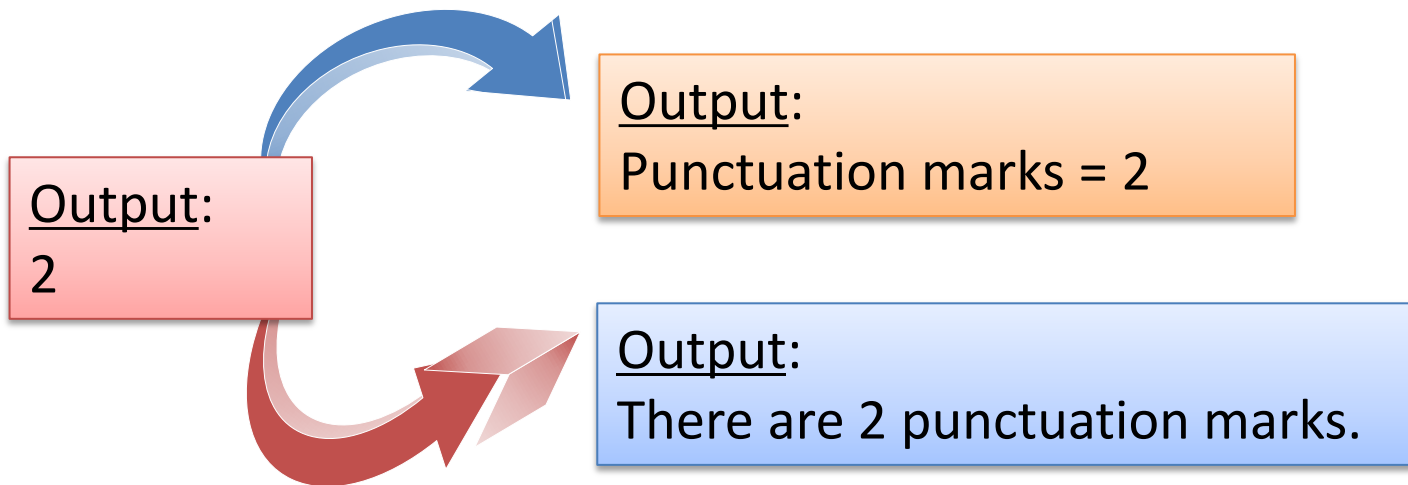
将字符串 “Hello, World!” 中的 “World” 替换成 “Python” , 并计算其包含的标点符号（由逗号、句号、感叹号和问号组成）的个数。



```
# Filename: puncount.py
aStr = "Hello, World!"
bStr = aStr[:7] + "Python!"
count = 0
for ch in bStr[:]:
    if ch in ',.!?':
        count += 1
print count
```

Output:

2



```
print 'There are %d punctuation marks. ' % (count)
format_string % (arguments_to_convert)
```

格式运算符%

格式化字符	转换方式
%c	转换成字符 (ASCII值或长度为1的字符串)
%r	优先使用repr()函数进行字符串转换
%s	优先使用str()函数进行字符串转换
%d / %i	转换成带符号十进制整数
%u	转换成无符号十进制整数
%o	转换成无符号八进制数
%x/ %X	转换成无符号十六进制数 (x/X表示转换后的十六进制字符的小/大写)
%e / %E	转换成科学计数法 (e/E控制输出e/E)
%f / %F	转换成浮点数 (部分小数自然截断)
%g / %G	选择%e/%f或%E/%F中输出更短的形式输出
%%	输出%

格式运算符辅助符

符号	功能
*	定义宽度或精度
-	用作左对齐
+	在正数前显示加号 ('+')
<sp>	在正数前显示空格
#	在八进制数前显示零 ('0') , 在十六进制前显示'0x'或'0X' (取决于使用了'x'还是'X')
0	显示数字时前面用零 ('0') 填充而不是默认的空格
%	'%%'输出一个'%'
(var)	映射变量 (参数为字典)
m.n	m是显示的最小总宽度 , n是小数点后的位数 (如果可用的话)

格式运算符

Source

```
>>> cCode = ['AXP' , 'BA' , 'CAT' , 'CSCO' , 'CVX' ]
>>> cPrice = ['86.40' , '122.64' , '99.44' , '23.78' , '115.91' ]
>>> for i in range(5):
    print '%d%8s:%8s.' % (i, cCode[i], cPrice[i])
0    AXP:   86.40.
1     BA:  122.64.
2    CAT:   99.44.
3   CSCO:   23.78.
4    CVX:  115.91.
>>> print 'I get %d%%!' % 32
I get 32 %!
```

字符串的应用



有一个字符串“acdhdca”，判断其是否是回文串。接着判断一个数字354435是否是回文数字。

File

```
# Filename: compare.py
sStr = "acdhdca"
if (sStr == ''.join(reversed(sStr))):
    print 'Yes'
else:
    print 'No'
```

File

```
# Filename: compare.py
sStr = "acdhdca"
if cmp(sStr, ''.join(reversed(sStr)))==0:
    print 'Yes'
else:
    print 'No'
```

字符串的方法

<code>capitalize()</code>	<code>center()</code>	<code>count()</code>	<code>decode()</code>	<code>encode()</code>	<code>endswith()</code>
<code>expandtabs()</code>	<code>find()</code>	<code>index()</code>	<code>isalnum()</code>	<code>isalpha()</code>	<code>isdecimal()</code>
<code>isdigit()</code>	<code>islower()</code>	<code>isnumeric()</code>	<code>isspace()</code>	<code>istitle()</code>	<code>isupper()</code>
<code>join()</code>	<code>ljust()</code>	<code>lower()</code>	<code>lstrip()</code>	<code>partition()</code>	<code>replace()</code>
<code>rfind()</code>	<code>rindex()</code>	<code>rjust()</code>	<code>rpartition()</code>	<code>rstrip()</code>	<code>split()</code>
<code>splitlines()</code>	<code>startswith()</code>	<code>strip()</code>	<code>swapcase()</code>	<code>title()</code>	<code>translate()</code>
<code>upper()</code>	<code>zfill()</code>				

字符串的应用



有一些从网络上下载的类似如下形式的一些句子：

What do you think of this saying "No pain, No gain"?

对于句子中双引号中的内容，首先判断其是否满足标题格式，不管满足与否最终都将其转换为标题格式输出。



```
# Filename: totitle.py
```

```
aStr = 'What do you think of this saying "No pain, No gain"?'
index = aStr.index("\"", 0, len(aStr))
rindex = aStr.rindex("\"", 0, len(aStr))
tempStr = aStr[index+1:rindex]
```

```
if tempStr.istitle():
```

```
    print 'It is title format.'
```

```
else:
```

```
    print 'It is not title format.'
```

```
print tempStr.title()
```

```
tempstr= aStr.split("\"")[1]
```


转义字符

字符	说明
\0	空字符
\a	响铃
\b	退格
\t	横向制表符
\n	换行
\v	纵向制表符
\f	换页
\r	回车
\e	转义
\"	双引号
\'	单引号
\\	反斜杠
\"(在行尾时)	续行符

\000 八进制数000代表的字符

\xXX 十六进制数xx代表的字符



```
>>> aStr = '\101\t\x41\n'
```

```
>>> bStr = '\141\t\x61\n'
```

```
>>> print aStr, bStr
```

```
A
```

```
A
```

```
a
```

```
a
```



用Python玩转数据

列表

可扩展的 容器对象



```
>>> aList = list('Hello.')
>>> aList
['H', 'e', 'l', 'l', 'o', '.']
>>> aList = list('hello.')
>>> aList
['h', 'e', 'l', 'l', 'o', '.']
>>> aList[0] = 'H'
>>> aList
['H', 'e', 'l', 'l', 'o', '.']
```

包含不同 类型对象



```
>>> bList = [1,2,'a',3.5]
```

列表的形式

- `aList = [1, 2, 3, 4, 5]`
- `names = ['Zhao', 'Qian', 'Sun', 'Li']`
- `bList = [3, 2, 1, 'Action']`
- `pList = [('AXP', 'American Express Company', '86.40'),
('BA', 'The Boeing Company', '122.64'),
('CAT', 'Caterpillar Inc.', '99.44'),
('CSCO', 'Cisco Systems, Inc.', '23.78'),
('CVX', 'Chevron Corporation', '115.91')]`

列表



某学校组织了一场校园歌手比赛，每个歌手的得分由10名评委和观众决定，最终得分的规则是去掉10名评委所打分数中的一个最高分和一个最低分，再加上所有观众评委分数后的平均值。评委打出的10个分数为：9、9、8.5、10、7、8、8、9、8和10，观众评委打出的综合评分为9，请计算该歌手的最终得分。



```
# Filename: scoring.py
jScores = [9, 9, 8.5, 10, 7, 8, 8, 9, 8, 10]
aScore = 9
jScores.sort()
jScores.pop()
jScores.pop(0)
jScores.append(aScore)
aveScore = sum(jScores)/len(jScores)
print aveScore
```

```
[7, 8, 8, 8, 8.5, 9, 9, 9, 10, 10]
[8, 8, 8, 8.5, 9, 9, 9, 10]
[8, 8, 8, 8.5, 9, 9, 9, 10, 9]
8.72222222222
```



将工作日 (['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']) 和周末 (['Saturday', 'Sunday']) 的表示形式合并，并将它们用序号标出并分行显示。



Filename: week.py

```
week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
```

```
weekend = ['Saturday', 'Sunday']
```

```
week.extend(weekend)
```

```
for i,j in enumerate(week):
```

```
    print i+1, j
```

Output:

1 Monday

2 Tuesday

3 Wednesday

4 Thursday

5 Friday

6 Saturday

7 Sunday

列表的方法

append()
count()
extend()
index()
insert()
pop()
remove()
reverse()
sort()

参数的作用：

list.sort(func=None, key=None, reverse=False)



```
>>> numList = [3, 11, 5, 8, 16, 1]
```

```
>>> fruitList = ['apple', 'banana', 'pear', 'lemon', 'avocado']
```

```
>>> numList.sort(reverse = True)
```

```
>>> numList
```

```
[16, 11, 8, 5, 3, 1]
```

```
>>> fruitList.sort(key = len)
```

```
>>> fruitList
```

```
['pear', 'apple', 'lemon', 'banana', 'avocado']
```

列表解析

List comprehensions ,
list comps

动态创建列表
简单灵活有用



```
>>> [x for x in range(10)]  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
>>> [x ** 2 for x in range(10)]  
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]  
>>> [x ** 2 for x in range(10) if x ** 2 < 50]  
[0, 1, 4, 9, 16, 25, 36, 49]  
>>> [(x+1,y+1) for x in range(2) for y in range(2)]  
[(1, 1), (1, 2), (2, 1), (2, 2)]
```

```
[ expression for expr in sequence1  
                for expr2 in sequence2 ...  
                for exprN in sequenceN  
                if condition ]
```




用Python玩转数据

元组

- 元组的一般使用与列表类似

Source

```
>>> 2014
2014
>>> 2014,
(2014,)
```

Source

```
>>> bTuple = (['Monday', 1], 2, 3)
>>> bTuple
(['Monday', 1], 2, 3)
>>> bTuple[0][1]
1
>>> len(bTuple)
3
>>> bTuple[1:]
(2, 3)
```

- 列表元素可以改变
- 元组元素不可以改变



```
>>> aList = ['AXP', 'BA', 'CAT']
>>> aTuple = ('AXP', 'BA', 'CAT')
>>> aList[1] = 'Alibaba'
>>> print aList
>>> aTuple[1]= 'Alibaba'
>>> print aTuple
```

```
['AXP', 'Alibaba', 'CAT']
```

```
aTuple[1]='Alibaba'
```

```
TypeError: 'tuple' object does not support item assignment
```

- 函数的适用类型



```
>>> aList = [3, 5, 2, 4]
>>> aList
[3, 5, 2, 4]
>>> sorted(aList)
[2, 3, 4, 5]
>>> aList
[3, 5, 2, 4]
>>> aList.sort()
>>> aList
[2, 3, 4, 5]
```



```
>>> aTuple = (3, 5, 2, 4)
>>> sorted(aTuple)
array([2, 3, 4, 5])
>>> aTuple
(3, 5, 2, 4)
>>> aTuple.sort()
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'tuple' object has no attribute 'sort'

元组的作用

元组用在什么地方？

在映射类型
中当作键使
用

函数的特殊
类型参数

作为很多内
建函数的返
回值

元组作为函数的形式参数

Python中函数的参数形式

- 位置或关键字参数
 - 仅位置的参数
 - 可变长位置参数
 - 可变长关键字参数
- (参数可以设定默认值)



```
>>> def func(args1, args2 = 'World!'):
    print args1, args2
>>> func('Hello,')
Hello, World!
>>> func('Hello,',args2 = 'Python!')
Hello, Python!
>>> func(args2 = 'Apple!',args1 = 'Hello,')
Hello, Apple!
```

```
>>> def func(args1, *argst):
    print args1
    print argst
```

元组作为函数的形式参数



```
>>> def func(args1, *argst):  
    print args1  
    print argst  
>>> func('Hello','Wangdachui','Niuyun','Linling')  
Hello,  
( 'Wangdachui', 'Niuyun', 'Linling')
```

元组作为函数的常见返回类型

返回对象的个数	返回类型
0	None
1	object
>1	tuple

返回值类型为元组的函数

- enumerate()
- coerce()
- ...



```
>>> def func():  
        return 1,2,3  
>>> func()  
(1, 2, 3)
```