Where are data from? How to represent data?

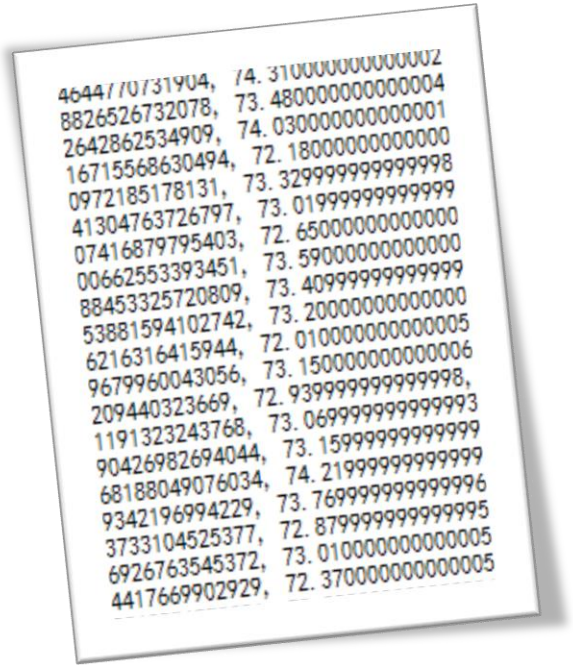
数据 获取与表示

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

用Python玩转数据

1 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

Source

```
>>> f1 = open('d:\\infile.txt')
>>> f2 = open(r'd:\outfile.txt', 'w')
>>> f3 = open('record.dat', 'wb', 0)
```

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

使用系统默认的缓冲区大小

- 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
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+）

文件相关函数

返回值

- open()函数返回一个文件（file）对象
- 文件对象可迭代
- 有关闭和读写文件相关的函数/方法
  - f.read(), f.write(), f.readline(), f.readlines(), f.writelines()
  - f.close()
  - f.seek()

写文件-f.write()

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

Source

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

firstpro.txt :
Hello, World!

Source

```
>>> with open('firstpro.txt', 'w') as f:
        f.write('Hello, World!')
```

推荐此写法，可以进行上下文管理（结束后自动关闭文件）、异常处理。

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

Source

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

Output:
Hello, World!

其他读写函数


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

- 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
with open('companies.txt') as f1:
        cNames = f1.readlines()
        for i in range(0, len(cNames)):
                cNames[i] = str(i+1) + ' ' + cNames[i]
with open('scompanies.txt', 'w') as f2:
        f2.writelines(cNames)
```

Output:

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

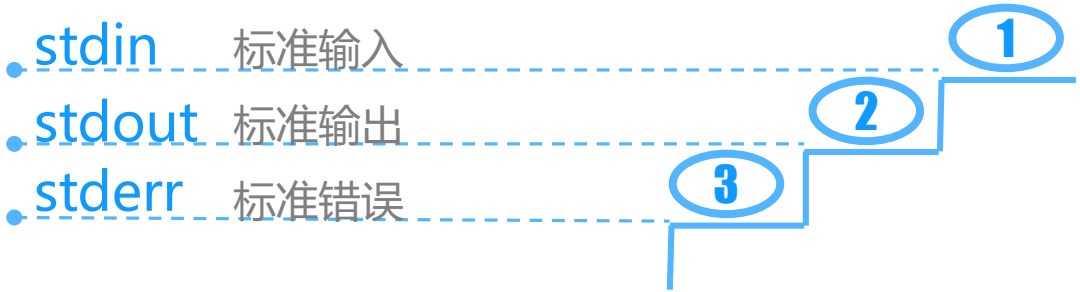
其他文件相关函数


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

- **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



Source

```
>>> newcName = input('Enter the name of new  company: ')
Enter the name of new company: Alibiabia
>>> print(newcName)
Alibiabia
```

```
>>> import sys
>>> sys.stdout.write('hello')
```

用Python玩转数据

网络数据获取

用Python获取数据

rmbol	Company Name	Pric
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
IIO	iBio, Inc.	1.4
ELP	Yelp, Inc.	58.5
IFN	Infinera Corporation	13
SFT	Microsoft Corporation	44.7
O	The Coca-Cola Comp…	41.2
FE	Pfizer Inc	28.8

网络数据如何获取（爬取）？

抓取网页，解析网页内容

- 抓取
  - urllib内建模块
    - urllib.request
  - Requests第三方库
  - Scrapy框架
- 解析
  - BeautifulSoup库
  - re模块

逐渐被替代

第三方API抓取+解析

中小型网络爬虫

大型网络爬虫开发

Requests库

- Requests库是更简单、方便和人性化的Python HTTP第三方库
- Requests官网：http://www.python-requests.org/
- 基本方法

requests.get()	请求获取指定URL位置的资源，对应HTTP协议的GET方法

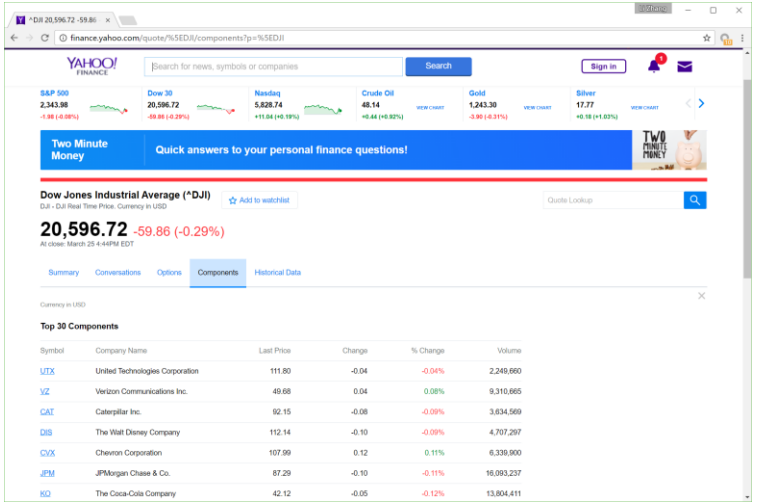
遵循网站爬虫协议 robots.txt

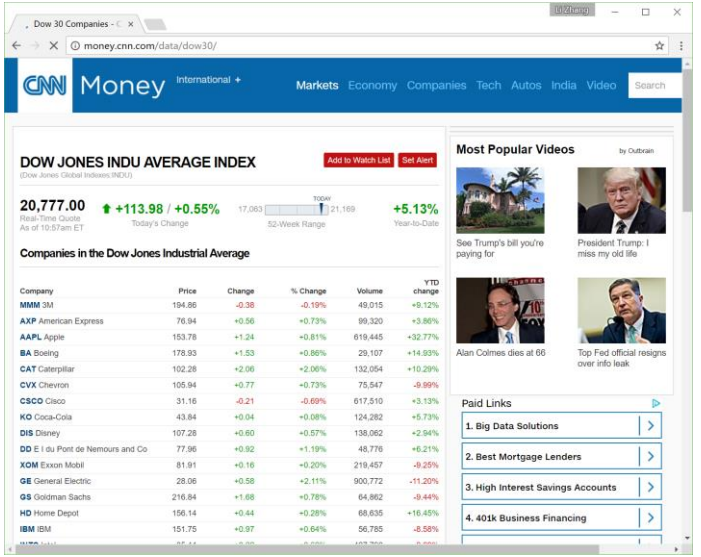
Source

```
>>> import requests
>>> r = requests.get('https://book.douban.com/subject/1084336/comments/')
>>> r.status_code
200
>>> print(r.text)
```

道指成分股数据

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





http://money.cnn.com/data/dow30/

利用Requests库获取道指成分股数据


```
<tr>
        <td class="wsod_firstCol"><a href="/quote/quote.html?symb=INTC" class="ws
        <td class="wsod_aRight"><span stream="last_167459" class="wsod_stream">35
        <td class="wsod_aRight"><span stream="change_167459" class="wsod_stream">
        <td class="wsod_aRight"><span stream="changePct_167459" class="wsod_strea
        <td class="wsod_aRight">17,171,872</td>
        <td class="wsod_aRight"><span class="negData">-3.39%</span></td>
</tr>

<tr>
        <td class="wsod_firstCol"><a href="/quote/quote.html?symb=JNJ" class="wso
        <td class="wsod_aRight"><span stream="last_174239" class="wsod_stream">12
        <td class="wsod_aRight"><span stream="change_174239" class="wsod_stream">
        <td class="wsod_aRight"><span stream="changePct_174239" class="wsod_strea
        <td class="wsod_aRight">6,571,254</td>
        <td class="wsod_aRight"><span class="posData">+10.21%</span></td>
</tr>
```

- 包含多个字符串
  - 'AXP', 'American Express Company', '77.77'
  - 'BA', 'The Boeing Company', '177.83'
  - 'CAT', 'Caterpillar Inc.', '96.39'
  - …

File

```
# Filename: dji.py
import requests
re = requests.get('http://money.cnn.com/data/dow30/')    # the url may change
print(re.text)
```

网页数据解析

- **BeautifulSoup**是一个可以从HTML或XML文件中提取数据的Python库
- 官方网站：https://www.crummy.com/software/BeautifulSoup/bs4/doc/

- **re**正则表达式模块进行各类正则表达式处理
- 参考网站：https://docs.python.org/3.5/library/re.html

```
soup.find_all('p', 'comment-content')
```

```
<p class="comment-content">不知道第几次重读。每过一段时间再读，都有新的收获。心变得很柔软，脑里的迷雾被驱散。更多的关注他人，关心这个世界，自私是多么无趣的事情啊。我想，写一本能温暖人心，帮助困难的人们的书，比世界上很多事情都有意义。</p>
```

```
'<span class="user-stars allstar(.*?) rating"'
```

```
<span class="user-stars allstar50 rating" title="力荐"></span>
```

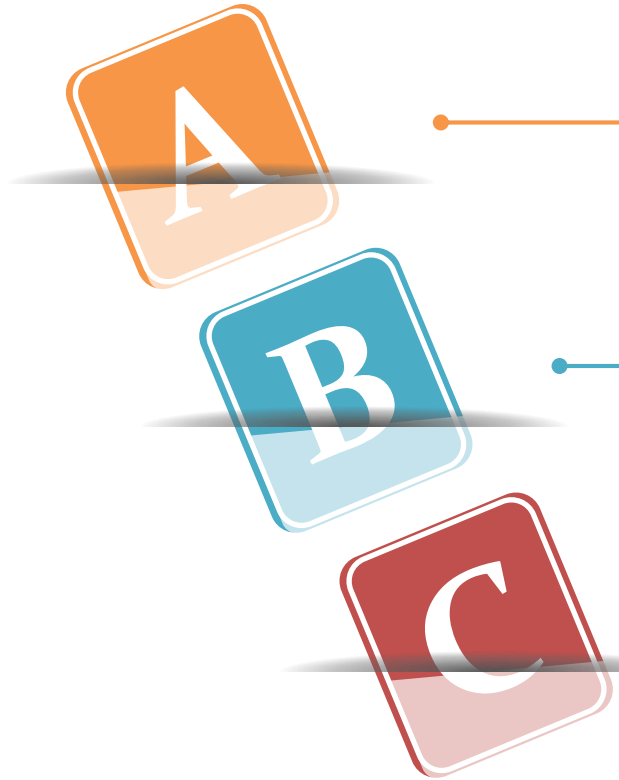
用Python玩转数据

3

序列

序列

- aStr = 'Hello, World!'
- aList = [2, 3, 5, 7, 11]
- aTuple = ('Sunday', 'happy')
- pList = [('AXP', 'American Express Company', '78.51'),
  ('BA', 'The Boeing Company', '184.76'),
  ('CAT', 'Caterpillar Inc.', '96.39'),
  ('CSCO', 'Cisco Systems, Inc.', '33.71'),
  ('CVX', 'Chevron Corporation', '106.09')]



字符串
Strings

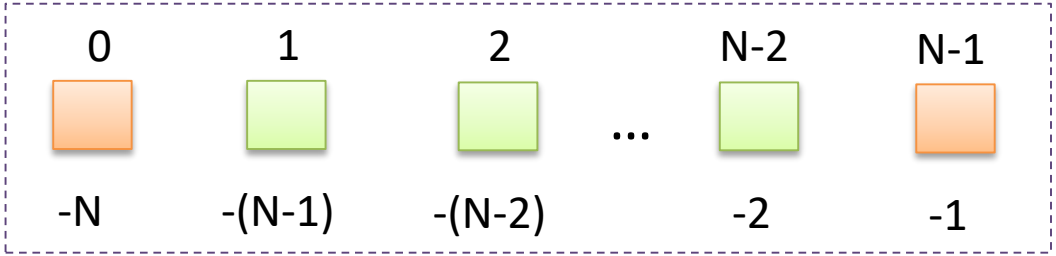
列表
Lists

元组
Tuples

Python中的序列

	0	1	2	3	4	5	6
week	'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

序列类型转换内建函数
序列类型可用内建函数

标准类型运算符

Source

```
>>> 'apple' < 'banana'
True
>>> [1,3,5]  != [2,4,6]
True
>>> aTuple = ('BA', 'The Boeing Company', '184.76')
>>> 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

<	>
<=	>=
==	!=

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', '184.76')
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]

序列类型转换内建函数

list()
str()
tuple()

Source

```
>>> 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', '!')
```

序列类型其他常用内建函数

enumerate()	reversed()
len()	sorted()
max()	sum()
min()	zip()

Source

```
>>> 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


```
lf = [('AXP', 'American Express Company', '78.51'),
      ('BA', 'The Boeing Company', '184.76'),
      ('CAT', 'Caterpillar Inc.', '96.39'),
      ('CSCO', 'Cisco Systems, Inc.', '33.71'),
      ('CVX', 'Chevron Corporation', '106.09')]
```

Source

```
>>> aStr = 'The Boeing Company'
>>> bStr = "The Boeing Company "
>>> cStr = "I'm a student."
>>> dStr = '''The Boeing
company'''
```

字符串小例子

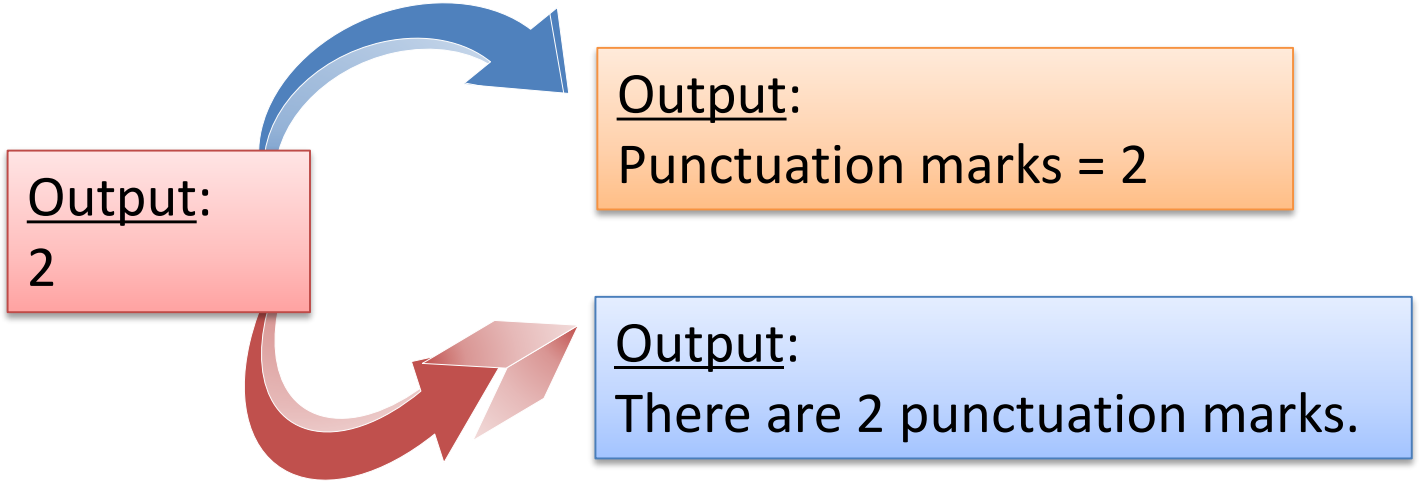
将字符串“Hello, World!”中的“World”替换成“Python”，并计算其包含的标点符号（由逗号、句号、感叹号和问号组成）的个数。

File

```
# 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

字符串与输出形式



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

format_string % (arguments_to_convert)

更推荐！！！

print('There are {0:d} punctuation marks. '.format(count))

format_string.format(arguments_to_convert)

类型说明符

字符	描述
b	二进制，以2为基数输出数字
o	八进制，以8为基数输出数字
x	十六进制，以16为基数输出数字，9以上的数字用小写字母（类型符为X时用大写字母）表示
c	字符，将整数转换成对应的Unicode字符输出
d	十进制整数，以10为基数输出数字
f	浮点数，以浮点数输出数字
e	指数记法，以科学计数法输出数字，用e（类型符是E时用大写E）表示幂

其他常用格式说明符

符号	描述
+m.nf	输出带符号（若正整数输出“+”号）的数，保留n位小数，整个输出占m列（若实际宽度超过m则突破m的限制）
<	左对齐，默认用空格填充右边
0>5d	右对齐，用0填充左边，宽度为5
^	居中对齐
{{}}	输出一个{}

[对齐说明符][符号说明符][最小宽度说明符][.精度说明符][类型说明符]

```
>>> age, height = 21, 1.758
>>> print("Age:{0:<5d}, Height:{1:5.2f}".format(age, height))
Age:21   , Height: 1.76
```

用format函数格式化字符串例

Source

```
>>> cCode = ['AXP' , 'BA' , 'CAT' , 'CSCO' , 'CVX' ]
>>> cPrice = ['78.51' , '184.76' , '96.39' , '33.71' , '106.09' ]
>>> for i in range(5):
        print('{:<8d}{:8s}{:8s}'.format(i, cCode[i], cPrice[i]))
0       AXP      78.51
1       BA       184.76
2       CAT      96.39
3       CSCO     33.71
4       CVX      106.09
>>> print('I get {:d}{{}}!'.format(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
import operator
sStr = "acdhdca"
if operator.eq(sStr, ''.join(reversed(sStr)))==0:
    print('Yes')
else:
    print('No')
```

```
sStr == sStr[::-1]
```

字符串常用方法

capitalize()	center()	count()	encode()	endswith()	find()
format()	index()	isalnum()	isalpha()	isdigit()	islower()
isspace()	istitle()	isupper()	join()	ljust()	lower()
lstrip()	maketrans()	partition()	replace()	rfind()	rindex()
rjust()	rpartition()	rstrip()	split()	splitlines()	startswith()
strip()	swapcase()	title()	translate()	upper()	zfill()

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。

File

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

\ooo　八进制数ooo代表的字符

\xXX　十六进制数xx代表的字符

Source

```
>>> aStr = '\101\t\x41\n'
>>> bStr = '\141\t\x61\n'
>>> print(aStr, bStr)
A	A
a	a
```

用Python玩转数据

5

列表

列表

可扩展的容器对象

Source

```
>>> 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

Source

```
>>> 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', '78.51'),
  ('BA', 'The Boeing Company', '184.76'),
  ('CAT', 'Caterpillar Inc.', '96.39'),
  ('CSCO', 'Cisco Systems, Inc.', '33.71'),
  ('CVX', 'Chevron Corporation', '106.09')]

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。

File

```
# 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
```

列表

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。

File

```
# 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
```

列表方法

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

参数的作用：

list.sort(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

Source

```
>>> [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)]
```

生成器表达式
Generator expression

```
>>> sum(x for x in range(10))
45
```

lazy evaluation

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

用Python玩转数据

6

元组

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)
```

元组

- 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] = 'Alibiabia'
>>> print(aList)
['AXP', 'Alibiabia', 'CAT']
>>> aTuple[1]= 'Alibiabia'
>>> print(aTuple)
aTuple[1]='Alibiabia'
TypeError: 'tuple' object does not support item assignment
```

元组

- 函数的适用类型

Source

```
>>> 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]
```

Source

```
>>> aTuple = (3, 5, 2, 4)
>>> sorted(aTuple)
[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？

可变长位置参数（元组）

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）

Source

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

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

可变长位置参数（元组）

Source

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

元组作为函数特殊返回类型

返回对象的个数	返回类型
0	None
1	object
>1	tuple

Source

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