和大熊猫们(Pandas)一起游戏吧！

Pandas是Python的一个用于数据分析的库： http://pandas.pydata.org (http://pandas.pydata.org)

API速查：http://pandas.pydata.org/pandas-docs/stable/api.html (http://pandas.pydata.org/pandas-docs/stable/api.html)

基于NumPy,SciPy的功能，在其上补充了大量的数据操作（Data Manipulation）功能。

统计、分组、排序、透视表自由转换，如果你已经很熟悉结构化数据库（RDBMS）与Excel的功能，就会知道Pandas有过之而无不及！

0. 上手玩：Why Pandas?

普通的程序员看到一份数据会怎么做?

In [1]:

```
import codecs
import requests
import numpy as np
import scipy as sp
import pandas as pd
import datetime
import json
```

In [60]:

```
r = requests.get("http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data")
with codecs.open('S1EP3_Iris.txt','w',encoding='utf-8') as f:
    f.write(r.text)
```

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
4.8,3.4,1.9,0.2,Iris-setosa
5.0,3.0,1.6,0.2,Iris-setosa
5.0,3.4,1.6,0.4,Iris-setosa
5.2,3.5,1.5,0.2,Iris-setosa
5.2,3.4,1.4,0.2,Iris-setosa
4.7,3.2,1.6,0.2,Iris-setosa
```

```
4.8,3.1,1.6,0.2,Iris-setosa
5.4,3.4,1.5,0.4,Iris-setosa
5.2,4.1,1.5,0.1,Iris-setosa
5.5,4.2,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.0,3.2,1.2,0.2,Iris-setosa
5.5,3.5,1.3,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
4.4,3.0,1.3,0.2,Iris-setosa
5.1,3.4,1.5,0.2,Iris-setosa
5.0,3.5,1.3,0.3,Iris-setosa
4.5,2.3,1.3,0.3,Iris-setosa
4.4,3.2,1.3,0.2,Iris-setosa
5.0,3.5,1.6,0.6,Iris-setosa
5.1,3.8,1.9,0.4,Iris-setosa
4.8,3.0,1.4,0.3,Iris-setosa
5.1,3.8,1.6,0.2,Iris-setosa
4.6,3.2,1.4,0.2,Iris-setosa
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
6.5,2.8,4.6,1.5,Iris-versicolor
5.7,2.8,4.5,1.3,Iris-versicolor
6.3,3.3,4.7,1.6,Iris-versicolor
4.9,2.4,3.3,1.0,Iris-versicolor
6.6,2.9,4.6,1.3,Iris-versicolor
5.2,2.7,3.9,1.4,Iris-versicolor
5.0,2.0,3.5,1.0,Iris-versicolor
5.9,3.0,4.2,1.5,Iris-versicolor
6.0,2.2,4.0,1.0,Iris-versicolor
6.1,2.9,4.7,1.4,Iris-versicolor
5.6,2.9,3.6,1.3,Iris-versicolor
6.7,3.1,4.4,1.4,Iris-versicolor
5.6,3.0,4.5,1.5,Iris-versicolor
5.8,2.7,4.1,1.0,Iris-versicolor
6.2,2.2,4.5,1.5,Iris-versicolor
5.6,2.5,3.9,1.1,Iris-versicolor
5.9,3.2,4.8,1.8,Iris-versicolor
6.1,2.8,4.0,1.3,Iris-versicolor
6.3,2.5,4.9,1.5,Iris-versicolor
6.1,2.8,4.7,1.2,Iris-versicolor
6.4,2.9,4.3,1.3,Iris-versicolor
6.6,3.0,4.4,1.4,Iris-versicolor
6.8,2.8,4.8,1.4,Iris-versicolor
6.7,3.0,5.0,1.7,Iris-versicolor
6.0,2.9,4.5,1.5,Iris-versicolor
5.7,2.6,3.5,1.0,Iris-versicolor
5.5,2.4,3.8,1.1,Iris-versicolor
5.5,2.4,3.7,1.0,Iris-versicolor
5.8,2.7,3.9,1.2,Iris-versicolor
6.0,2.7,5.1,1.6,Iris-versicolor
5.4,3.0,4.5,1.5,Iris-versicolor
6.0,3.4,4.5,1.6,Iris-versicolor
6.7,3.1,4.7,1.5,Iris-versicolor
6.3,2.3,4.4,1.3,Iris-versicolor
5.6,3.0,4.1,1.3,Iris-versicolor
5.5,2.5,4.0,1.3,Iris-versicolor
5.5,2.6,4.4,1.2,Iris-versicolor
6.1,3.0,4.6,1.4,Iris-versicolor
5.8,2.6,4.0,1.2,Iris-versicolor
5.0,2.3,3.3,1.0,Iris-versicolor
5.6,2.7,4.2,1.3,Iris-versicolor
5.7,3.0,4.2,1.2,Iris-versicolor
5.7,2.9,4.2,1.3,Iris-versicolor
6.2,2.9,4.3,1.3,Iris-versicolor
```

```
5.1,2.5,3.0,1.1,Iris-versicolor
5.7,2.8,4.1,1.3,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
6.3,2.9,5.6,1.8,Iris-virginica
6.5,3.0,5.8,2.2,Iris-virginica
7.6,3.0,6.6,2.1,Iris-virginica
4.9,2.5,4.5,1.7,Iris-virginica
7.3,2.9,6.3,1.8,Iris-virginica
6.7,2.5,5.8,1.8,Iris-virginica
7.2,3.6,6.1,2.5,Iris-virginica
6.5,3.2,5.1,2.0,Iris-virginica
6.4,2.7,5.3,1.9,Iris-virginica
6.8,3.0,5.5,2.1,Iris-virginica
5.7,2.5,5.0,2.0,Iris-virginica
5.8,2.8,5.1,2.4,Iris-virginica
6.4,3.2,5.3,2.3,Iris-virginica
6.5,3.0,5.5,1.8,Iris-virginica
7.7,3.8,6.7,2.2,Iris-virginica
7.7,2.6,6.9,2.3,Iris-virginica
6.0,2.2,5.0,1.5,Iris-virginica
6.9,3.2,5.7,2.3,Iris-virginica
5.6,2.8,4.9,2.0,Iris-virginica
7.7,2.8,6.7,2.0,Iris-virginica
6.3,2.7,4.9,1.8,Iris-virginica
6.7,3.3,5.7,2.1,Iris-virginica
7.2,3.2,6.0,1.8,Iris-virginica
6.2,2.8,4.8,1.8,Iris-virginica
6.1,3.0,4.9,1.8,Iris-virginica
6.4,2.8,5.6,2.1,Iris-virginica
7.2,3.0,5.8,1.6,Iris-virginica
7.4,2.8,6.1,1.9,Iris-virginica
7.9,3.8,6.4,2.0,Iris-virginica
6.4,2.8,5.6,2.2,Iris-virginica
6.3,2.8,5.1,1.5,Iris-virginica
6.1,2.6,5.6,1.4,Iris-virginica
7.7,3.0,6.1,2.3,Iris-virginica
6.3,3.4,5.6,2.4,Iris-virginica
6.4,3.1,5.5,1.8,Iris-virginica
6.0,3.0,4.8,1.8,Iris-virginica
6.9,3.1,5.4,2.1,Iris-virginica
6.7,3.1,5.6,2.4,Iris-virginica
6.9,3.1,5.1,2.3,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
6.8,3.2,5.9,2.3,Iris-virginica
6.7,3.3,5.7,2.5,Iris-virginica
6.7,3.0,5.2,2.3,Iris-virginica
6.3,2.5,5.0,1.9,Iris-virginica
6.5,3.0,5.2,2.0,Iris-virginica
6.2,3.4,5.4,2.3,Iris-virginica
5.9,3.0,5.1,1.8,Iris-virginica
```

In [2]:

```
with codecs.open('S1EP3_Iris.txt','r',encoding='utf-8') as f:
    lines = f.readlines()

for line in lines:
    print line,
```

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
```

```
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
4.8,3.4,1.9,0.2,Iris-setosa
5.0,3.0,1.6,0.2,Iris-setosa
5.0,3.4,1.6,0.4,Iris-setosa
5.2,3.5,1.5,0.2,Iris-setosa
5.2,3.4,1.4,0.2,Iris-setosa
4.7,3.2,1.6,0.2,Iris-setosa
4.8,3.1,1.6,0.2,Iris-setosa
5.4,3.4,1.5,0.4,Iris-setosa
5.2,4.1,1.5,0.1,Iris-setosa
5.5,4.2,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.0,3.2,1.2,0.2,Iris-setosa
5.5,3.5,1.3,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
4.4,3.0,1.3,0.2,Iris-setosa
5.1,3.4,1.5,0.2,Iris-setosa
5.0,3.5,1.3,0.3,Iris-setosa
4.5,2.3,1.3,0.3,Iris-setosa
4.4,3.2,1.3,0.2,Iris-setosa
5.0,3.5,1.6,0.6,Iris-setosa
5.1,3.8,1.9,0.4,Iris-setosa
4.8,3.0,1.4,0.3,Iris-setosa
5.1,3.8,1.6,0.2,Iris-setosa
4.6,3.2,1.4,0.2,Iris-setosa
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
6.5,2.8,4.6,1.5,Iris-versicolor
5.7,2.8,4.5,1.3,Iris-versicolor
6.3,3.3,4.7,1.6,Iris-versicolor
4.9,2.4,3.3,1.0,Iris-versicolor
6.6,2.9,4.6,1.3,Iris-versicolor
5.2,2.7,3.9,1.4,Iris-versicolor
5.0,2.0,3.5,1.0,Iris-versicolor
5.9,3.0,4.2,1.5,Iris-versicolor
6.0,2.2,4.0,1.0,Iris-versicolor
6.1,2.9,4.7,1.4,Iris-versicolor
5.6,2.9,3.6,1.3,Iris-versicolor
6.7,3.1,4.4,1.4,Iris-versicolor
5.6,3.0,4.5,1.5,Iris-versicolor
5.8,2.7,4.1,1.0,Iris-versicolor
6.2,2.2,4.5,1.5,Iris-versicolor
5.6,2.5,3.9,1.1,Iris-versicolor
5.9,3.2,4.8,1.8,Iris-versicolor
6.1,2.8,4.0,1.3,Iris-versicolor
6.3,2.5,4.9,1.5,Iris-versicolor
6.1,2.8,4.7,1.2,Iris-versicolor
```

```
6.4,2.9,4.3,1.3,Iris-versicolor
6.6,3.0,4.4,1.4,Iris-versicolor
6.8,2.8,4.8,1.4,Iris-versicolor
6.7,3.0,5.0,1.7,Iris-versicolor
6.0,2.9,4.5,1.5,Iris-versicolor
5.7,2.6,3.5,1.0,Iris-versicolor
5.5,2.4,3.8,1.1,Iris-versicolor
5.5,2.4,3.7,1.0,Iris-versicolor
5.8,2.7,3.9,1.2,Iris-versicolor
6.0,2.7,5.1,1.6,Iris-versicolor
5.4,3.0,4.5,1.5,Iris-versicolor
6.0,3.4,4.5,1.6,Iris-versicolor
6.7,3.1,4.7,1.5,Iris-versicolor
6.3,2.3,4.4,1.3,Iris-versicolor
5.6,3.0,4.1,1.3,Iris-versicolor
5.5,2.5,4.0,1.3,Iris-versicolor
5.5,2.6,4.4,1.2,Iris-versicolor
6.1,3.0,4.6,1.4,Iris-versicolor
5.8,2.6,4.0,1.2,Iris-versicolor
5.0,2.3,3.3,1.0,Iris-versicolor
5.6,2.7,4.2,1.3,Iris-versicolor
5.7,3.0,4.2,1.2,Iris-versicolor
5.7,2.9,4.2,1.3,Iris-versicolor
6.2,2.9,4.3,1.3,Iris-versicolor
5.1,2.5,3.0,1.1,Iris-versicolor
5.7,2.8,4.1,1.3,Iris-versicolor
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
6.3,2.9,5.6,1.8,Iris-virginica
6.5,3.0,5.8,2.2,Iris-virginica
7.6,3.0,6.6,2.1,Iris-virginica
4.9,2.5,4.5,1.7,Iris-virginica
7.3,2.9,6.3,1.8,Iris-virginica
6.7,2.5,5.8,1.8,Iris-virginica
7.2,3.6,6.1,2.5,Iris-virginica
6.5,3.2,5.1,2.0,Iris-virginica
6.4,2.7,5.3,1.9,Iris-virginica
6.8,3.0,5.5,2.1,Iris-virginica
5.7,2.5,5.0,2.0,Iris-virginica
5.8,2.8,5.1,2.4,Iris-virginica
6.4,3.2,5.3,2.3,Iris-virginica
6.5,3.0,5.5,1.8,Iris-virginica
7.7,3.8,6.7,2.2,Iris-virginica
7.7,2.6,6.9,2.3,Iris-virginica
6.0,2.2,5.0,1.5,Iris-virginica
6.9,3.2,5.7,2.3,Iris-virginica
5.6,2.8,4.9,2.0,Iris-virginica
7.7,2.8,6.7,2.0,Iris-virginica
6.3,2.7,4.9,1.8,Iris-virginica
6.7,3.3,5.7,2.1,Iris-virginica
7.2,3.2,6.0,1.8,Iris-virginica
6.2,2.8,4.8,1.8,Iris-virginica
6.1,3.0,4.9,1.8,Iris-virginica
6.4,2.8,5.6,2.1,Iris-virginica
7.2,3.0,5.8,1.6,Iris-virginica
7.4,2.8,6.1,1.9,Iris-virginica
7.9,3.8,6.4,2.0,Iris-virginica
6.4,2.8,5.6,2.2,Iris-virginica
6.3,2.8,5.1,1.5,Iris-virginica
6.1,2.6,5.6,1.4,Iris-virginica
7.7,3.0,6.1,2.3,Iris-virginica
6.3,3.4,5.6,2.4,Iris-virginica
6.4,3.1,5.5,1.8,Iris-virginica
6.0,3.0,4.8,1.8,Iris-virginica
6.9,3.1,5.4,2.1,Iris-virginica
6.7,3.1,5.6,2.4,Iris-virginica
6.9,3.1,5.1,2.3,Iris-virginica
```

```
5.8,2.7,5.1,1.9,Iris-virginica
6.8,3.2,5.9,2.3,Iris-virginica
6.7,3.3,5.7,2.5,Iris-virginica
6.7,3.0,5.2,2.3,Iris-virginica
6.3,2.5,5.0,1.9,Iris-virginica
6.5,3.0,5.2,2.0,Iris-virginica
6.2,3.4,5.4,2.3,Iris-virginica
5.9,3.0,5.1,1.8,Iris-virginica
```

Pandas的意义就在于

快速的识别结构化数据

In [3]:

```
import pandas as pd
irisdata = pd.read_csv('S1EP3_Iris.txt',header = None, encoding='utf-8')
irisdata
```

Out[3]:

	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa
20	5.4	3.4	1.7	0.2	Iris-setosa
21	5.1	3.7	1.5	0.4	Iris-setosa
22	4.6	3.6	1.0	0.2	Iris-setosa
23	5.1	3.3	1.7	0.5	Iris-setosa
24	4.8	3.4	1.9	0.2	Iris-setosa
25	5.0	3.0	1.6	0.2	Iris-setosa

26	5.0	3.4	1.6	0.4	Iris-setosa
27	5.2	3.5	1.5	0.2	Iris-setosa
28	5.2	3.4	1.4	0.2	Iris-setosa
29	4.7	3.2	1.6	0.2	Iris-setosa
...	...	...	...	...	...
120	6.9	3.2	5.7	2.3	Iris-virginica
121	5.6	2.8	4.9	2.0	Iris-virginica
122	7.7	2.8	6.7	2.0	Iris-virginica
123	6.3	2.7	4.9	1.8	Iris-virginica
124	6.7	3.3	5.7	2.1	Iris-virginica
125	7.2	3.2	6.0	1.8	Iris-virginica
126	6.2	2.8	4.8	1.8	Iris-virginica
127	6.1	3.0	4.9	1.8	Iris-virginica
128	6.4	2.8	5.6	2.1	Iris-virginica
129	7.2	3.0	5.8	1.6	Iris-virginica
130	7.4	2.8	6.1	1.9	Iris-virginica
131	7.9	3.8	6.4	2.0	Iris-virginica
132	6.4	2.8	5.6	2.2	Iris-virginica
133	6.3	2.8	5.1	1.5	Iris-virginica
134	6.1	2.6	5.6	1.4	Iris-virginica
135	7.7	3.0	6.1	2.3	Iris-virginica
136	6.3	3.4	5.6	2.4	Iris-virginica
137	6.4	3.1	5.5	1.8	Iris-virginica
138	6.0	3.0	4.8	1.8	Iris-virginica
139	6.9	3.1	5.4	2.1	Iris-virginica
140	6.7	3.1	5.6	2.4	Iris-virginica
141	6.9	3.1	5.1	2.3	Iris-virginica
142	5.8	2.7	5.1	1.9	Iris-virginica
143	6.8	3.2	5.9	2.3	Iris-virginica
144	6.7	3.3	5.7	2.5	Iris-virginica
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

快速的操作元数据

In [4]:

```
cnames = ['sepal_length','sepal_width','petal_length','petal_width','class']
irisdata.columns = cnames
irisdata
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa
20	5.4	3.4	1.7	0.2	Iris-setosa
21	5.1	3.7	1.5	0.4	Iris-setosa
22	4.6	3.6	1.0	0.2	Iris-setosa
23	5.1	3.3	1.7	0.5	Iris-setosa
24	4.8	3.4	1.9	0.2	Iris-setosa
25	5.0	3.0	1.6	0.2	Iris-setosa
26	5.0	3.4	1.6	0.4	Iris-setosa
27	5.2	3.5	1.5	0.2	Iris-setosa
28	5.2	3.4	1.4	0.2	Iris-setosa
29	4.7	3.2	1.6	0.2	Iris-setosa
...	...	...	...	...	...
120	6.9	3.2	5.7	2.3	Iris-virginica
121	5.6	2.8	4.9	2.0	Iris-virginica
122	7.7	2.8	6.7	2.0	Iris-virginica

123	6.3	2.7	4.9	1.8	Iris-virginica
124	6.7	3.3	5.7	2.1	Iris-virginica
125	7.2	3.2	6.0	1.8	Iris-virginica
126	6.2	2.8	4.8	1.8	Iris-virginica
127	6.1	3.0	4.9	1.8	Iris-virginica
128	6.4	2.8	5.6	2.1	Iris-virginica
129	7.2	3.0	5.8	1.6	Iris-virginica
130	7.4	2.8	6.1	1.9	Iris-virginica
131	7.9	3.8	6.4	2.0	Iris-virginica
132	6.4	2.8	5.6	2.2	Iris-virginica
133	6.3	2.8	5.1	1.5	Iris-virginica
134	6.1	2.6	5.6	1.4	Iris-virginica
135	7.7	3.0	6.1	2.3	Iris-virginica
136	6.3	3.4	5.6	2.4	Iris-virginica
137	6.4	3.1	5.5	1.8	Iris-virginica
138	6.0	3.0	4.8	1.8	Iris-virginica
139	6.9	3.1	5.4	2.1	Iris-virginica
140	6.7	3.1	5.6	2.4	Iris-virginica
141	6.9	3.1	5.1	2.3	Iris-virginica
142	5.8	2.7	5.1	1.9	Iris-virginica
143	6.8	3.2	5.9	2.3	Iris-virginica
144	6.7	3.3	5.7	2.5	Iris-virginica
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

快速过滤

In [5]:

```
irisdata[irisdata['petal_width']==irisdata.petal_width.max()]
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width	class
100	6.3	3.3	6.0	2.5	Iris-virginica
109	7.2	3.6	6.1	2.5	Iris-virginica
144	6.7	3.3	5.7	2.5	Iris-virginica

快速切片

In [6]:

```
irisdata.iloc[::30,:2]
```

Out[6]:

	sepal_length	sepal_width
0	5.1	3.5
30	4.8	3.1
60	5.0	2.0
90	5.5	2.6
120	6.9	3.2

快速统计

In [7]:

```
print irisdata['class'].value_counts()

for x in xrange(4):
    s = irisdata.iloc[:,x]
    print '{0:<12}'.format(s.name.upper()), " Statistics: ", \
    '{0:>5}  {1:>5}  {2:>5}  {3:>5}'.format(s.max(), s.min(), round(s.mean(),2),round(s.std(),
2))
```

```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
SEPAL_LENGTH  Statistics:    7.9    4.3   5.84   0.83
SEPAL_WIDTH   Statistics:    4.4    2.0   3.05   0.43
PETAL_LENGTH  Statistics:    6.9    1.0   3.76   1.76
PETAL_WIDTH   Statistics:    2.5    0.1    1.2   0.76
```

快速“MapReduce”

In [8]:

```
slogs = lambda x:sp.log(x)*x
entpy = lambda x:sp.exp((slogs(x.sum())-x.map(slogs).sum())/x.sum())
irisdata.groupby('class').agg(entpy)
```

Out[8]:

	sepal_length	sepal_width	petal_length	petal_width
class				
Iris-setosa	49.878745	49.695242	49.654909	45.810069
Iris-versicolor	49.815081	49.680665	49.694505	49.452305
Iris-virginica	49.772059	49.714500	49.761700	49.545918

1. 欢迎来到大熊猫世界

Pandas的重要数据类型

- DataFrame(二维表)
- Series(一维序列)
- Index(行索引，行级元数据)

1.1 Series：pandas的长枪(数据表中的一列或一行,观测向量,一维数组...)

数据世界中对于任意一个个体的全面观测，或者对于任意一组个体某一属性的观测，全部可以抽象为Series的概念。

用值构建一个Series：

由默认index和values组成。

In [9]:

```
Series1 = pd.Series(np.random.randn(4))
print Series1,type(Series1)
print Series1.index
print Series1.values
```

```
0    0.030480
1    0.072746
2   -0.186607
3   -1.412244
dtype: float64 <class 'pandas.core.series.Series'>
Int64Index([0, 1, 2, 3], dtype='int64')
[ 0.03048042  0.07274621 -0.18660749 -1.41224432]
```

Series支持过滤的原理就如同NumPy：

In [10]:

```
print Series1>0
print Series1[Series1>0]
```

```
0     True
1     True
2    False
3    False
dtype: bool
0    0.030480
1    0.072746
dtype: float64
```

当然也支持**Broadcasting**：

In [11]:

```
print Series1*2
print Series1+5
```

```
0    0.060961
1    0.145492
2   -0.373215
3   -2.824489
dtype: float64
0    5.030480
1    5.072746
2    4.813393
3    3.587756
dtype: float64
```

以及**Universal Function**：

In [12]:

```
print np.exp(Series1)
#NumPy Universal Function
f_np = np.frompyfunc(lambda x:np.exp(x*2+5),1,1)
print f_np(Series1)
```

```
0    1.030950
1    1.075458
2    0.829769
3    0.243596
dtype: float64
0     157.742
1     171.656
2     102.185
3    8.806687
dtype: object
```

在序列上就使用行标，而不是创建一个2列的数据表，能够轻松辨别哪里是数据，哪里是元数据：

In [13]:

```
Series2 = pd.Series(Series1.values,index=['norm_'+unicode(i) for i in xrange(4)])
print Series2,type(Series2)
print Series2.index
print type(Series2.index)
print Series2.values
```

```
norm_0    0.030480
norm_1    0.072746
norm_2   -0.186607
norm_3   -1.412244
dtype: float64 <class 'pandas.core.series.Series'>
Index([u'norm_0', u'norm_1', u'norm_2', u'norm_3'], dtype='object')
<class 'pandas.core.index.Index'>
[ 0.03048042  0.07274621 -0.18660749 -1.41224432]
```

虽然行是有顺序的，但是仍然能够通过行级的index来访问到数据：

（当然也不尽然像Ordered Dict，因为行索引甚至可以重复，不推荐重复的行索引不代表不能用）

In [14]:

```
print Series2[['norm_0','norm_3']]
```

```
norm_0    0.030480
norm_3   -1.412244
dtype: float64
```

In [15]:

```
print 'norm_0' in Series2
print 'norm_6' in Series2
```

```
True
False
```

默认行索引就像行号一样：

In [16]:

```
print Series1.index
```

```
Int64Index([0, 1, 2, 3], dtype='int64')
```

从Key不重复的Ordered Dict或者从Dict来定义Series就不需要担心行索引重复：

In [17]:

```
Series3_Dict = {"Japan":"Tokyo","S.Korea":"Seoul","China":"Beijing"}
Series3_pdSeries = pd.Series(Series3_Dict)
print Series3_pdSeries
print Series3_pdSeries.values
print Series3_pdSeries.index
```

```
China      Beijing
Japan        Tokyo
S.Korea      Seoul
dtype: object
['Beijing' 'Tokyo' 'Seoul']
Index([u'China', u'Japan', u'S.Korea'], dtype='object')
```

想让序列按你的排序方式保存？就算有缺失值都毫无问题

In [18]:

```
Series4_IndexList = ["Japan","China","Singapore","S.Korea"]
Series4_pdSeries = pd.Series( Series3_Dict ,index = Series4_IndexList)
print Series4_pdSeries
print Series4_pdSeries.values
print Series4_pdSeries.index
print Series4_pdSeries.isnull()
print Series4_pdSeries.notnull()
```

```
Japan          Tokyo
China        Beijing
Singapore        NaN
S.Korea        Seoul
dtype: object
['Tokyo' 'Beijing' nan 'Seoul']
Index([u'Japan', u'China', u'Singapore', u'S.Korea'], dtype='object')
Japan        False
China        False
Singapore     True
S.Korea      False
dtype: bool
Japan         True
China         True
Singapore    False
S.Korea       True
dtype: bool
```

整个序列级别的元数据信息：name

当数据序列以及index本身有了名字，就可以更方便的进行后续的数据关联啦！

In [19]:

```
print Series4_pdSeries.name
print Series4_pdSeries.index.name
```

```
None
None
```

In [20]:

```
Series4_pdSeries.name = "Capital Series"
Series4_pdSeries.index.name = "Nation"
print Series4_pdSeries
```

```
Nation
Japan          Tokyo
China        Beijing
Singapore        NaN
S.Korea        Seoul
Name: Capital Series, dtype: object
```

"字典"？不是的，行index可以重复，尽管不推荐。

In [21]:

```
Series5_IndexList = ['A','B','B','C']
Series5 = pd.Series(Series1.values,index = Series5_IndexList)
print Series5
print Series5[['B','A']]
```

```
A    0.030480
B    0.072746
B   -0.186607
C   -1.412244
dtype: float64
B    0.072746
B   -0.186607
A    0.030480
dtype: float64
```

1.2 DataFrame：pandas的战锤(数据表，二维数组)

Series的有序集合，就像R的DataFrame一样方便。

仔细想想，绝大部分的数据形式都可以表现为DataFrame。

从NumPy二维数组、从文件或者从数据库定义：数据虽好，勿忘列名

In [22]:

```
dataNumPy = np.asarray([('Japan','Tokyo',4000),('S.Korea','Seoul',1300),('China','Beijing',910
0)])
DF1 = pd.DataFrame(dataNumPy,columns=['nation','capital','GDP'])
DF1
```

Out[22]:

	nation	capital	GDP
0	Japan	Tokyo	4000
1	S.Korea	Seoul	1300
2	China	Beijing	9100

等长的列数据保存在一个字典里（JSON）：很不幸，字典key是无序的

In [23]:

```
dataDict = {'nation':['Japan','S.Korea','China'],'capital':['Tokyo','Seoul','Beijing'],'GDP':[
4900,1300,9100]}
DF2 = pd.DataFrame(dataDict)
DF2
```

Out[23]:

	GDP	capital	nation
0	4900	Tokyo	Japan
1	1300	Seoul	S.Korea
2	9100	Beijing	China

从另一个DataFrame定义DataFrame：啊，强迫症犯了！

In [24]:

```
DF21 = pd.DataFrame(DF2,columns=['nation','capital','GDP'])
DF21
```

Out[24]:

	nation	capital	GDP
0	Japan	Tokyo	4900
1	S.Korea	Seoul	1300
2	China	Beijing	9100

In [25]:

```
DF22 = pd.DataFrame(DF2,columns=['nation','capital','GDP'],index = [2,0,1])
DF22
```

Out[25]:

	nation	capital	GDP
2	China	Beijing	9100
0	Japan	Tokyo	4900
1	S.Korea	Seoul	1300

从DataFrame中取出列？两种方法（与JavaScript完全一致！）

- '.'的写法容易与其他预留关键字产生冲突
- '[ ]'的写法最安全。

In [26]:

```
print DF22.nation,DF22.capital
print DF22['GDP']
```

```
2      China
0      Japan
1    S.Korea
Name: nation, dtype: object 2    Beijing
0      Tokyo
1      Seoul
Name: capital, dtype: object
2    9100
0    4900
1    1300
Name: GDP, dtype: int64
```

从**DataFrame**中取出行？（至少）两种方法：

In [27]:

```
print DF22[0:1] #给出的实际是DataFrame
print DF22.ix[0] #通过对应Index给出行
```

```
  nation  capital   GDP
2  China  Beijing  9100
nation     Japan
capital    Tokyo
GDP         4900
Name: 0, dtype: object
```

像**NumPy**切片一样的终极招式：**iloc**

In [28]:

```
print DF22.iloc[0,:]
print DF22.iloc[:,0]
```

```
nation       China
capital    Beijing
GDP           9100
Name: 2, dtype: object
2      China
0      Japan
1    S.Korea
Name: nation, dtype: object
```

听说你从**Alter Table**地狱来，大熊猫笑了

然而动态增加列无法用"."的方式完成，只能用"[]"

In [29]:

```
DF22['population'] = [1600,130,55]
DF22
```

Out[29]:

	nation	capital	GDP	population
2	China	Beijing	9100	1600
0	Japan	Tokyo	4900	130
1	S.Korea	Seoul	1300	55

1.3 Index：pandas进行数据操纵的鬼牌（行级索引）

行级索引是

- 元数据
- 可能由真实数据产生，因此可以视作数据
- 可以由多重索引也就是多个列组合而成
- 可以和列名进行交换，也可以进行堆叠和展开，达到Excel透视表效果

Index有四种...哦不，很多种写法，一些重要的索引类型包括

- pd.Index（普通）
- Int64Index（数值型索引）
- MultiIndex（多重索引，在数据操纵中更详细描述）
- DatetimeIndex（以时间格式作为索引）
- PeriodIndex （含周期的时间格式作为索引）

直接定义普通索引，长得就和普通的Series一样

In [30]:

```
index_names = ['a','b','c']
Series_for_Index = pd.Series(index_names)
print pd.Index(index_names)
print pd.Index(Series_for_Index)
```

```
Index([u'a', u'b', u'c'], dtype='object')
Index([u'a', u'b', u'c'], dtype='object')
```

可惜Immutable，牢记！

In [31]:

```
index_names = ['a','b','c']
index0 = pd.Index(index_names)
print index0.get_values()
index0[2] = 'd'
```

```
['a' 'b' 'c']

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-31-f34da0a8623c> in <module>()
      2 index0 = pd.Index(index_names)
      3 print index0.get_values()
----> 4 index0[2] = 'd'

/Users/wangweiyang/anaconda/anaconda/lib/python2.7/site-packages/pandas/core/index.pyc in __se
titem__(self, key, value)
   1055
   1056     def __setitem__(self, key, value):
-> 1057         raise TypeError("Indexes does not support mutable operations")
   1058
   1059     def __getitem__(self, key):

TypeError: Indexes does not support mutable operations
```

扔进去一个含有多元组的List，就有了MultiIndex

可惜，如果这个List Comprehension改成小括号，就不对了。

In [32]:

```
multi1 = pd.Index([('Row_'+str(x+1),'Col_'+str(y+1)) for x in xrange(4) for y in xrange(4)])
multi1.name = ['index1','index2']
print multi1
```

```
MultiIndex(levels=[[u'Row_1', u'Row_2', u'Row_3', u'Row_4'], [u'Col_1', u'Col_2', u'Col_3', u'
Col_4']],
           labels=[[0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3], [0, 1, 2, 3, 0, 1, 2, 3,
0, 1, 2, 3, 0, 1, 2, 3]])
```

对于Series来说，如果拥有了多重Index，数据，变形！

下列代码说明：

- 二重MultiIndex的Series可以unstack()成DataFrame
- DataFrame可以stack成拥有二重MultiIndex的Series

In [33]:

```
data_for_multi1 = pd.Series(xrange(0,16),index=multi1)
data_for_multi1
```

Out[33]:

```
Row_1  Col_1     0
       Col_2     1
       Col_3     2
       Col_4     3
Row_2  Col_1     4
       Col_2     5
       Col_3     6
       Col_4     7
Row_3  Col_1     8
       Col_2     9
       Col_3    10
       Col_4    11
Row_4  Col_1    12
       Col_2    13
       Col_3    14
       Col_4    15
dtype: int64
```

In [34]:

```
data_for_multi1.unstack()
```

Out[34]:

	Col_1	Col_2	Col_3	Col_4
Row_1	0	1	2	3
Row_2	4	5	6	7
Row_3	8	9	10	11
Row_4	12	13	14	15

In [35]:

```
data_for_multi1.unstack().stack()
```

Out[35]:

```
Row_1  Col_1     0
       Col_2     1
       Col_3     2
       Col_4     3
Row_2  Col_1     4
       Col_2     5
       Col_3     6
       Col_4     7
Row_3  Col_1     8
       Col_2     9
       Col_3    10
       Col_4    11
Row_4  Col_1    12
       Col_2    13
       Col_3    14
       Col_4    15
dtype: int64
```

我们来看一下非平衡数据的例子：

Row_1,2,3,4和Col_1,2,3,4并不是全组合的。

In [36]:

```
multi2 = pd.Index([('Row_'+str(x+1),'Col_'+str(y+1)) for x in xrange(5) for y in xrange(x)])
multi2
```

Out[36]:

```
MultiIndex(levels=[[u'Row_2', u'Row_3', u'Row_4', u'Row_5'], [u'Col_1', u'Col_2', u'Col_3', u'
Col_4']],
           labels=[[0, 1, 1, 2, 2, 2, 3, 3, 3, 3], [0, 0, 1, 0, 1, 2, 0, 1, 2, 3]])
```

In [37]:

```
data_for_multi2 = pd.Series(np.arange(10),index = multi2)
data_for_multi2
```

Out[37]:

```
Row_2  Col_1    0
Row_3  Col_1    1
       Col_2    2
Row_4  Col_1    3
       Col_2    4
       Col_3    5
Row_5  Col_1    6
       Col_2    7
       Col_3    8
       Col_4    9
dtype: int64
```

In [38]:

```
data_for_multi2.unstack()
```

Out[38]:

	Col_1	Col_2	Col_3	Col_4
Row_2	0	NaN	NaN	NaN
Row_3	1	2	NaN	NaN
Row_4	3	4	5	NaN
Row_5	6	7	8	9

In [39]:

```
data_for_multi2.unstack().stack()
```

Out[39]:

```
Row_2  Col_1    0
Row_3  Col_1    1
       Col_2    2
Row_4  Col_1    3
       Col_2    4
       Col_3    5
Row_5  Col_1    6
       Col_2    7
       Col_3    8
       Col_4    9
dtype: float64
```

DateTime标准库如此好用，你值得拥有

In [40]:

```
dates = [datetime.datetime(2015,1,1),datetime.datetime(2015,1,8),datetime.datetime(2015,1,30)]
pd.DatetimeIndex(dates)
```

Out[40]:

```
DatetimeIndex(['2015-01-01', '2015-01-08', '2015-01-30'], dtype='datetime64[ns]', freq=None, t
z=None)
```

如果你不仅需要时间格式统一，时间频率也要统一的话

In [41]:

```
periodindex1 = pd.period_range('2015-01','2015-04',freq='M')
print periodindex1
```

```
PeriodIndex(['2015-01', '2015-02', '2015-03', '2015-04'], dtype='int64', freq='M')
```

月级精度和日级精度如何转换?

有的公司统一以1号代表当月，有的公司统一以最后一天代表当月，转化起来很麻烦，可以asfreq

In [42]:

```
print periodindex1.asfreq('D',how='start')
print periodindex1.asfreq('D',how='end')
```

```
PeriodIndex(['2015-01-01', '2015-02-01', '2015-03-01', '2015-04-01'], dtype='int64', freq='D')
PeriodIndex(['2015-01-31', '2015-02-28', '2015-03-31', '2015-04-30'], dtype='int64', freq='D')
```

最后的最后，我要真正把两种频率的时间精度匹配上?

In [43]:

```
periodindex_mon = pd.period_range('2015-01','2015-03',freq='M').asfreq('D',how='start')
periodindex_day = pd.period_range('2015-01-01','2015-03-31',freq='D')

print periodindex_mon
print periodindex_day
```

```
PeriodIndex(['2015-01-01', '2015-02-01', '2015-03-01'], dtype='int64', freq='D')
PeriodIndex(['2015-01-01', '2015-01-02', '2015-01-03', '2015-01-04',
             '2015-01-05', '2015-01-06', '2015-01-07', '2015-01-08',
             '2015-01-09', '2015-01-10', '2015-01-11', '2015-01-12',
             '2015-01-13', '2015-01-14', '2015-01-15', '2015-01-16',
             '2015-01-17', '2015-01-18', '2015-01-19', '2015-01-20',
             '2015-01-21', '2015-01-22', '2015-01-23', '2015-01-24',
             '2015-01-25', '2015-01-26', '2015-01-27', '2015-01-28',
             '2015-01-29', '2015-01-30', '2015-01-31', '2015-02-01',
             '2015-02-02', '2015-02-03', '2015-02-04', '2015-02-05',
             '2015-02-06', '2015-02-07', '2015-02-08', '2015-02-09',
             '2015-02-10', '2015-02-11', '2015-02-12', '2015-02-13',
             '2015-02-14', '2015-02-15', '2015-02-16', '2015-02-17',
             '2015-02-18', '2015-02-19', '2015-02-20', '2015-02-21',
             '2015-02-22', '2015-02-23', '2015-02-24', '2015-02-25',
             '2015-02-26', '2015-02-27', '2015-02-28', '2015-03-01',
             '2015-03-02', '2015-03-03', '2015-03-04', '2015-03-05',
             '2015-03-06', '2015-03-07', '2015-03-08', '2015-03-09',
             '2015-03-10', '2015-03-11', '2015-03-12', '2015-03-13',
             '2015-03-14', '2015-03-15', '2015-03-16', '2015-03-17',
             '2015-03-18', '2015-03-19', '2015-03-20', '2015-03-21',
             '2015-03-22', '2015-03-23', '2015-03-24', '2015-03-25',
             '2015-03-26', '2015-03-27', '2015-03-28', '2015-03-29',
             '2015-03-30', '2015-03-31'],
            dtype='int64', freq='D')
```

粗粒度数据 + **reindex** + **ffill/bfill**

In [44]:

```
full_ts = pd.Series(periodindex_mon,index=periodindex_mon).reindex(periodindex_day,method='ffi
ll')
full_ts
```

Out[44]:

```
2015-01-01    2015-01-01
2015-01-02    2015-01-01
2015-01-03    2015-01-01
2015-01-04    2015-01-01
2015-01-05    2015-01-01
2015-01-06    2015-01-01
2015-01-07    2015-01-01
2015-01-08    2015-01-01
2015-01-09    2015-01-01
2015-01-10    2015-01-01
2015-01-11    2015-01-01
2015-01-12    2015-01-01
2015-01-13    2015-01-01
2015-01-14    2015-01-01
2015-01-15    2015-01-01
2015-01-16    2015-01-01
2015-01-17    2015-01-01
2015-01-18    2015-01-01
2015-01-19    2015-01-01
2015-01-20    2015-01-01
2015-01-21    2015-01-01
2015-01-22    2015-01-01
2015-01-23    2015-01-01
2015-01-24    2015-01-01
2015-01-25    2015-01-01
2015-01-26    2015-01-01
2015-01-27    2015-01-01
2015-01-28    2015-01-01
2015-01-29    2015-01-01
2015-01-30    2015-01-01
                 ...
2015-03-02    2015-03-01
2015-03-03    2015-03-01
2015-03-04    2015-03-01
2015-03-05    2015-03-01
2015-03-06    2015-03-01
2015-03-07    2015-03-01
2015-03-08    2015-03-01
2015-03-09    2015-03-01
2015-03-10    2015-03-01
2015-03-11    2015-03-01
2015-03-12    2015-03-01
2015-03-13    2015-03-01
2015-03-14    2015-03-01
2015-03-15    2015-03-01
2015-03-16    2015-03-01
2015-03-17    2015-03-01
2015-03-18    2015-03-01
2015-03-19    2015-03-01
2015-03-20    2015-03-01
2015-03-21    2015-03-01
2015-03-22    2015-03-01
2015-03-23    2015-03-01
2015-03-24    2015-03-01
2015-03-25    2015-03-01
2015-03-26    2015-03-01
2015-03-27    2015-03-01
2015-03-28    2015-03-01
2015-03-29    2015-03-01
2015-03-30    2015-03-01
2015-03-31    2015-03-01
Freq: D, dtype: object
```

关于索引，方便的操作有?

前面描述过了，索引有序，重复，但一定程度上又能通过key来访问，也就是说，某些集合操作都是可以支持的。

In [45]:

```
index1 = pd.Index(['A','B','B','C','C'])
index2 = pd.Index(['C','D','E','E','F'])
index3 = pd.Index(['B','C','A'])
print index1.append(index2)
print index1.difference(index2)
print index1.intersection(index2)
print index1.union(index2) # Support unique-value Index well
print index1.isin(index2)
print index1.delete(2)
print index1.insert(0,'K') # Not suggested
print index3.drop('A') # Support unique-value Index well
print index1.is_monotonic,index2.is_monotonic,index3.is_monotonic
print index1.is_unique,index2.is_unique,index3.is_unique
```

```
Index([u'A', u'B', u'B', u'C', u'C', u'C', u'D', u'E', u'E', u'F'], dtype='object')
Index([u'A', u'B'], dtype='object')
Index([u'C', u'C'], dtype='object')
Index([u'A', u'B', u'B', u'C', u'C', u'D', u'E', u'E', u'F'], dtype='object')
[False False False  True  True]
Index([u'A', u'B', u'C', u'C'], dtype='object')
Index([u'K', u'A', u'B', u'B', u'C', u'C'], dtype='object')
Index([u'B', u'C'], dtype='object')
True True False
False False True
```

2. 大熊猫世界来去自如：Pandas的I/O

老生常谈，从基础来看，我们仍然关心pandas对于与外部数据是如何交互的。

2.1 结构化数据输入输出

- read_csv与to_csv 是一对输入输出的工具，read_csv直接返回pandas.DataFrame，而to_csv只要执行命令即可写文件
  - read_table：功能类似
  - read_fwf：操作fixed width file
- read_excel与to_excel方便的与excel交互

还记得刚开始的例子吗?

- header 表示数据中是否存在列名，如果在第0行就写就写0，并且开始读数据时跳过相应的行数，不存在可以写none
- names 表示要用给定的列名来作为最终的列名
- encoding 表示数据集的字符编码，通常而言一份数据为了方便的进行文件传输都以utf-8作为标准

提问：下列例子中，header=4，names=cnames时，究竟会读到怎样的数据?

In [46]:

```
print cnames
irisdata = pd.read_csv('S1EP3_Iris.txt',header = None, names = cnames, encoding='utf-8')
irisdata
```

```
['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
```

Out[46]:

	sepal_length	sepal_width	petal_length	petal_width	class

0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa
20	5.4	3.4	1.7	0.2	Iris-setosa
21	5.1	3.7	1.5	0.4	Iris-setosa
22	4.6	3.6	1.0	0.2	Iris-setosa
23	5.1	3.3	1.7	0.5	Iris-setosa
24	4.8	3.4	1.9	0.2	Iris-setosa
25	5.0	3.0	1.6	0.2	Iris-setosa
26	5.0	3.4	1.6	0.4	Iris-setosa
27	5.2	3.5	1.5	0.2	Iris-setosa
28	5.2	3.4	1.4	0.2	Iris-setosa
29	4.7	3.2	1.6	0.2	Iris-setosa
...	...	...	...	...	...
120	6.9	3.2	5.7	2.3	Iris-virginica
121	5.6	2.8	4.9	2.0	Iris-virginica
122	7.7	2.8	6.7	2.0	Iris-virginica
123	6.3	2.7	4.9	1.8	Iris-virginica
124	6.7	3.3	5.7	2.1	Iris-virginica
125	7.2	3.2	6.0	1.8	Iris-virginica
126	6.2	2.8	4.8	1.8	Iris-virginica
127	6.1	3.0	4.9	1.8	Iris-virginica
128	6.4	2.8	5.6	2.1	Iris-virginica

129	7.2	3.0	5.8	1.6	Iris-virginica
130	7.4	2.8	6.1	1.9	Iris-virginica
131	7.9	3.8	6.4	2.0	Iris-virginica
132	6.4	2.8	5.6	2.2	Iris-virginica
133	6.3	2.8	5.1	1.5	Iris-virginica
134	6.1	2.6	5.6	1.4	Iris-virginica
135	7.7	3.0	6.1	2.3	Iris-virginica
136	6.3	3.4	5.6	2.4	Iris-virginica
137	6.4	3.1	5.5	1.8	Iris-virginica
138	6.0	3.0	4.8	1.8	Iris-virginica
139	6.9	3.1	5.4	2.1	Iris-virginica
140	6.7	3.1	5.6	2.4	Iris-virginica
141	6.9	3.1	5.1	2.3	Iris-virginica
142	5.8	2.7	5.1	1.9	Iris-virginica
143	6.8	3.2	5.9	2.3	Iris-virginica
144	6.7	3.3	5.7	2.5	Iris-virginica
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

希望了解全部参数的请移步API：

http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html#pandas.read_csv
(http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html#pandas.read_csv)

这里介绍一些常用的参数：

读取处理：

- skiprows：跳过一定的行数
- nrows：仅读取一定的行数
- skipfooter：尾部有固定的行数永不读取
- skip_blank_lines：空行跳过

内容处理：

- sep/delimiter：分隔符很重要，常见的有逗号，空格和Tab('\t')
- na_values：指定应该被当作na_values的数值
- thousands：处理数值类型时，每千位分隔符并不统一 (1.234.567,89或者1,234,567.89都可能)，此时要把字符串转化为数字需要指明千位分隔符

收尾处理：

- index_col：将真实的某列（列的数目，甚至列名）当作index
- squeeze：仅读到一列时，不再保存为pandas.DataFrame而是pandas.Series

2.1.x Excel ... ?

对于存储着极为规整数据的Excel而言，其实是没必要一定用Excel来存，尽管Pandas也十分友好的提供了I/O接口。

In [47]:

```
irisdata.to_excel('S1EP3_irisdata.xls',index = None,encoding='utf-8')
irisdata_from_excel = pd.read_excel('S1EP3_irisdata.xls',header=0, encoding='utf-8')
irisdata_from_excel
```

Out[47]:

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa
20	5.4	3.4	1.7	0.2	Iris-setosa
21	5.1	3.7	1.5	0.4	Iris-setosa
22	4.6	3.6	1.0	0.2	Iris-setosa
23	5.1	3.3	1.7	0.5	Iris-setosa
24	4.8	3.4	1.9	0.2	Iris-setosa
25	5.0	3.0	1.6	0.2	Iris-setosa
26	5.0	3.4	1.6	0.4	Iris-setosa
27	5.2	3.5	1.5	0.2	Iris-setosa
28	5.2	3.4	1.4	0.2	Iris-setosa
29	4.7	3.2	1.6	0.2	Iris-setosa

...	...	...	...	...	...
120	6.9	3.2	5.7	2.3	Iris-virginica
121	5.6	2.8	4.9	2.0	Iris-virginica
122	7.7	2.8	6.7	2.0	Iris-virginica
123	6.3	2.7	4.9	1.8	Iris-virginica
124	6.7	3.3	5.7	2.1	Iris-virginica
125	7.2	3.2	6.0	1.8	Iris-virginica
126	6.2	2.8	4.8	1.8	Iris-virginica
127	6.1	3.0	4.9	1.8	Iris-virginica
128	6.4	2.8	5.6	2.1	Iris-virginica
129	7.2	3.0	5.8	1.6	Iris-virginica
130	7.4	2.8	6.1	1.9	Iris-virginica
131	7.9	3.8	6.4	2.0	Iris-virginica
132	6.4	2.8	5.6	2.2	Iris-virginica
133	6.3	2.8	5.1	1.5	Iris-virginica
134	6.1	2.6	5.6	1.4	Iris-virginica
135	7.7	3.0	6.1	2.3	Iris-virginica
136	6.3	3.4	5.6	2.4	Iris-virginica
137	6.4	3.1	5.5	1.8	Iris-virginica
138	6.0	3.0	4.8	1.8	Iris-virginica
139	6.9	3.1	5.4	2.1	Iris-virginica
140	6.7	3.1	5.6	2.4	Iris-virginica
141	6.9	3.1	5.1	2.3	Iris-virginica
142	5.8	2.7	5.1	1.9	Iris-virginica
143	6.8	3.2	5.9	2.3	Iris-virginica
144	6.7	3.3	5.7	2.5	Iris-virginica
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

唯一重要的参数：sheetname=k，标志着一个excel的第k个sheet页将会被取出。（从0开始）

2.2 半结构化数据

JSON：网络传输中常用的一种数据格式。

仔细看一下，实际上这就是我们平时收集到异源数据的风格是一致的：

- 列名不能完全匹配
- key可能并不唯一
- 元数据被保存在数据里

In [95]:

```
json_data = [{'name':'Wang','sal':50000,'job':'VP'},\
             {'name':'Zhang','job':'Manager','report':'VP'},\
             {'name':'Li','sal':5000,'report':'IT'}]
data_employee = pd.read_json(json.dumps(json_data))
data_employee_ri = data_employee.reindex(columns=['name','job','sal','report'])
data_employee_ri
```

Out[95]:

	name	job	sal	report
0	Wang	VP	50000	NaN
1	Zhang	Manager	NaN	VP
2	Li	NaN	5000	IT

3. 深入Pandas数据操纵

在第一部分的基础上，数据会有更多种操纵方式：

- 通过列名、行index来取数据，结合ix、iloc灵活的获取数据的一个子集（第一部分已经介绍）
- 按记录拼接（就像Union All）或者关联（join）
- 方便的统计函数与自定义函数映射
- 排序
- 缺失值处理
- 与Excel一样灵活的数据透视表（在第四部分更详细介绍）

3.1 数据集整合

3.1.1 横向拼接：直接DataFrame

In [106]:

```
pd.DataFrame([np.random.rand(2),np.random.rand(2),np.random.rand(2)],columns=['C1','C2'])
```

Out[106]:

	C1	C2
0	0.565397	0.901534
1	0.626292	0.889969
2	0.175616	0.316424

3.1.2 横向拼接：Concatenate

In [107]:

```
pd.concat([data_employee_ri,data_employee_ri,data_employee_ri])
```

Out[107]:

	name	job	sal	report
0	Wang	VP	50000	NaN
1	Zhang	Manager	NaN	VP
2	Li	NaN	5000	IT
0	Wang	VP	50000	NaN
1	Zhang	Manager	NaN	VP
2	Li	NaN	5000	IT
0	Wang	VP	50000	NaN
1	Zhang	Manager	NaN	VP
2	Li	NaN	5000	IT

3.1.3 纵向拼接：Merge

根据数据列关联，使用on关键字

- 可以指定一列或多列
- 可以使用left_on和right_on

In [109]:

```
pd.merge(data_employee_ri,data_employee_ri,on='name')
```

Out[109]:

	name	job_x	sal_x	report_x	job_y	sal_y	report_y
0	Wang	VP	50000	NaN	VP	50000	NaN
1	Zhang	Manager	NaN	VP	Manager	NaN	VP
2	Li	NaN	5000	IT	NaN	5000	IT

In [110]:

```
pd.merge(data_employee_ri,data_employee_ri,on=['name','job'])
```

Out[110]:

	name	job	sal_x	report_x	sal_y	report_y
0	Wang	VP	50000	NaN	50000	NaN
1	Zhang	Manager	NaN	VP	NaN	VP
2	Li	NaN	5000	IT	5000	IT

根据index关联，可以直接使用left_index和right_index

In [111]:

```
data_employee_ri.index.name = 'index1'
pd.merge(data_employee_ri,data_employee_ri,left_index='index1',right_index='index1')
```

Out[111]:

	name_x	job_x	sal_x	report_x	name_y	job_y	sal_y	report_y
index1								
0	Wang	VP	50000	NaN	Wang	VP	50000	NaN
1	Zhang	Manager	NaN	VP	Zhang	Manager	NaN	VP
2	Li	NaN	5000	IT	Li	NaN	5000	IT

TIPS: 增加how关键字，并指定

- how = 'inner'
- how = 'left'
- how = 'right'
- how = 'outer'

结合how，可以看到merge基本再现了SQL应有的功能，并保持代码整洁

3.2 自定义函数映射

In [151]:

```
dataNumPy32 = np.asarray([('Japan','Tokyo',4000),('S.Korea','Seoul',1300),('China','Beijing',9
100)])
DF32 = pd.DataFrame(dataNumPy,columns=['nation','capital','GDP'])
DF32
```

Out[151]:

	nation	capital	GDP
0	Japan	Tokyo	4000
1	S.Korea	Seoul	1300
2	China	Beijing	9100

map: 以相同规则将一列数据作一个映射，也就是进行相同函数的处理

In [185]:

```
def GDP_Factorize(v):
    fv = np.float64(v)
    if fv > 6000.0:
        return 'High'
    elif fv < 2000.0:
        return 'Low'
    else:
        return 'Medium'

DF32['GDP_Level'] = DF32['GDP'].map(GDP_Factorize)
DF32['NATION'] = DF32.nation.map(str.upper)
DF32
```

Out[185]:

	nation	capital	GDP	GDP_Level	NATION
0	Japan	Tokyo	4000	Medium	JAPAN
1	S.Korea	Seoul	1300	Low	S.KOREA
2	China	Beijing	9100	High	CHINA

3.3 排序

- sort: 按一列或者多列的值进行行级排序
- sort_index: 根据index里的取值进行排序，而且可以根据axis决定是重排行还是列

In [124]:

```
dataNumPy33 = np.asarray([('Japan','Tokyo',4000),('S.Korea','Seoul',1300),('China','Beijing',9
100)])
DF33 = pd.DataFrame(dataNumPy,columns=['nation','capital','GDP'])
DF33
```

Out[124]:

	nation	capital	GDP
0	Japan	Tokyo	4000
1	S.Korea	Seoul	1300
2	China	Beijing	9100

In [121]:

```
DF33.sort('GDP')
```

Out[121]:

	nation	capital	GDP
1	S.Korea	Seoul	1300
0	Japan	Tokyo	4000
2	China	Beijing	9100

In [125]:

```
DF33.sort(['capital','nation'],ascending=False)
```

Out[125]:

	nation	capital	GDP
0	Japan	Tokyo	4000
1	S.Korea	Seoul	1300
2	China	Beijing	9100

In [126]:

```
DF33.sort('GDP').sort(ascending=False)
```

Out[126]:

	nation	capital	GDP
2	China	Beijing	9100
1	S.Korea	Seoul	1300
0	Japan	Tokyo	4000

In [130]:

```
DF33.sort_index(axis=1,ascending=True)
```

Out[130]:

	GDP	capital	nation
0	4000	Tokyo	Japan
1	1300	Seoul	S.Korea
2	9100	Beijing	China

一个好用的功能：Rank

In [149]:

```
DF33.rank()
```

Out[149]:

	nation	capital	GDP
0	2	3	2
1	3	2	1
2	1	1	3

In [150]:

```
DF33.rank(ascending=False)
```

Out[150]:

	nation	capital	GDP
0	2	1	2
1	1	2	3
2	3	3	1

注意tied data（相同值）的处理：

- method = 'average'
- method = 'min'
- method = 'max'
- method = 'first'

3.4 缺失数据处理

In [132]:

```
DF34 = data_for_multi2.unstack()
DF34
```

Out[132]:

	Col_1	Col_2	Col_3	Col_4
Row_2	0	NaN	NaN	NaN
Row_3	1	2	NaN	NaN
Row_4	3	4	5	NaN
Row_5	6	7	8	9

忽略缺失值：

In [133]:

```
DF34.mean(skipna=True)
```

Out[133]:

```
Col_1    2.500000
Col_2    4.333333
Col_3    6.500000
Col_4    9.000000
dtype: float64
```

如果不想忽略缺失值的话，就需要祭出fillna了：

In [141]:

```
DF34.mean(skipna=False)
```

Out[141]:

```
Col_1    2.5
Col_2    NaN
Col_3    NaN
Col_4    NaN
dtype: float64
```

In [145]:

```
DF34.fillna(0).mean(axis=1,skipna=False)
```

Out[145]:

```
Row_2    0.00
Row_3    0.75
Row_4    3.00
Row_5    7.50
dtype: float64
```

4. “一组”大熊猫：Pandas的groupby

groupby的功能类似SQL的group by关键字：

Split-Apply-Combine

- Split，就是按照规则分组
- Apply，通过一定的agg函数来获得输入pd.Series返回一个值的效果
- Combine，把结果收集起来

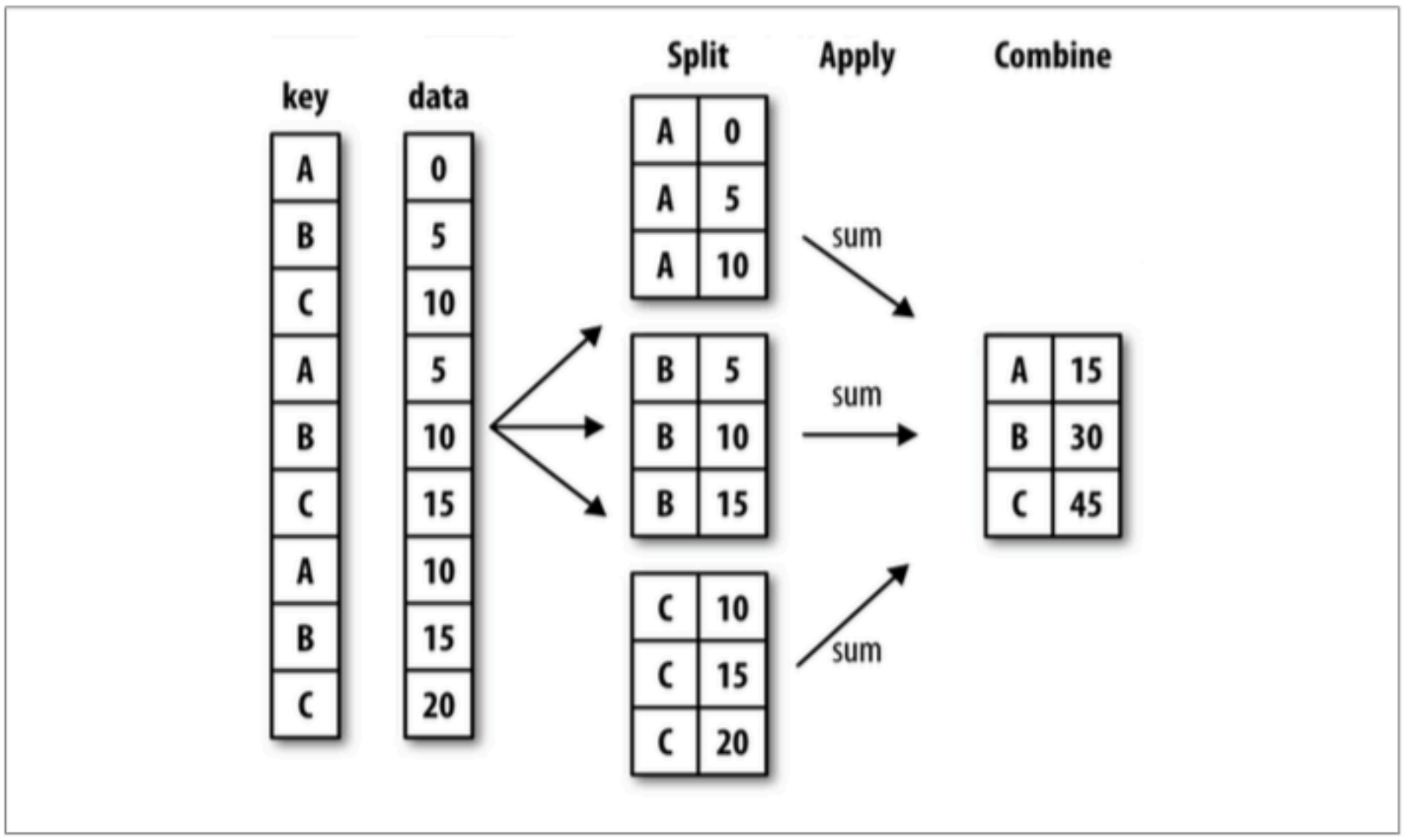
Pandas的groupby的灵活性：

- 分组的关键字可以来自于index，也可以来自于真实的列数据
- 分组规则可以通过一列或者多列

In [49]:

```
from IPython.display import Image
Image(filename="S1EP3_group.png")
```

Out[49]:



分组的具体逻辑

In [61]:

```
irisdata_group = irisdata.groupby('class')
irisdata_group
```

Out[61]:

```
<pandas.core.groupby.DataFrameGroupBy object at 0x106a66d10>
```

In [62]:

```
for level,subsetDF in irisdata_group:
    print level
    print subsetDF
```

```
Iris-setosa
    sepal_length  sepal_width  petal_length  petal_width        class
0            5.1          3.5           1.4          0.2  Iris-setosa
1            4.9          3.0           1.4          0.2  Iris-setosa
2            4.7          3.2           1.3          0.2  Iris-setosa
3            4.6          3.1           1.5          0.2  Iris-setosa
4            5.0          3.6           1.4          0.2  Iris-setosa
5            5.4          3.9           1.7          0.4  Iris-setosa
6            4.6          3.4           1.4          0.3  Iris-setosa
7            5.0          3.4           1.5          0.2  Iris-setosa
8            4.4          2.9           1.4          0.2  Iris-setosa
9            4.9          3.1           1.5          0.1  Iris-setosa
10           5.4          3.7           1.5          0.2  Iris-setosa
11           4.8          3.4           1.6          0.2  Iris-setosa
12           4.8          3.0           1.4          0.1  Iris-setosa
13           4.3          3.0           1.1          0.1  Iris-setosa
14           5.8          4.0           1.2          0.2  Iris-setosa
15           5.7          4.4           1.5          0.4  Iris-setosa
```

16	5.4	3.9	1.3	0.4	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa
20	5.4	3.4	1.7	0.2	Iris-setosa
21	5.1	3.7	1.5	0.4	Iris-setosa
22	4.6	3.6	1.0	0.2	Iris-setosa
23	5.1	3.3	1.7	0.5	Iris-setosa
24	4.8	3.4	1.9	0.2	Iris-setosa
25	5.0	3.0	1.6	0.2	Iris-setosa
26	5.0	3.4	1.6	0.4	Iris-setosa
27	5.2	3.5	1.5	0.2	Iris-setosa
28	5.2	3.4	1.4	0.2	Iris-setosa
29	4.7	3.2	1.6	0.2	Iris-setosa
30	4.8	3.1	1.6	0.2	Iris-setosa
31	5.4	3.4	1.5	0.4	Iris-setosa
32	5.2	4.1	1.5	0.1	Iris-setosa
33	5.5	4.2	1.4	0.2	Iris-setosa
34	4.9	3.1	1.5	0.1	Iris-setosa
35	5.0	3.2	1.2	0.2	Iris-setosa
36	5.5	3.5	1.3	0.2	Iris-setosa
37	4.9	3.1	1.5	0.1	Iris-setosa
38	4.4	3.0	1.3	0.2	Iris-setosa
39	5.1	3.4	1.5	0.2	Iris-setosa
40	5.0	3.5	1.3	0.3	Iris-setosa
41	4.5	2.3	1.3	0.3	Iris-setosa
42	4.4	3.2	1.3	0.2	Iris-setosa
43	5.0	3.5	1.6	0.6	Iris-setosa
44	5.1	3.8	1.9	0.4	Iris-setosa
45	4.8	3.0	1.4	0.3	Iris-setosa
46	5.1	3.8	1.6	0.2	Iris-setosa
47	4.6	3.2	1.4	0.2	Iris-setosa
48	5.3	3.7	1.5	0.2	Iris-setosa
49	5.0	3.3	1.4	0.2	Iris-setosa

Iris-versicolor

	sepal_length	sepal_width	petal_length	petal_width	class
50	7.0	3.2	4.7	1.4	Iris-versicolor
51	6.4	3.2	4.5	1.5	Iris-versicolor
52	6.9	3.1	4.9	1.5	Iris-versicolor
53	5.5	2.3	4.0	1.3	Iris-versicolor
54	6.5	2.8	4.6	1.5	Iris-versicolor
55	5.7	2.8	4.5	1.3	Iris-versicolor
56	6.3	3.3	4.7	1.6	Iris-versicolor
57	4.9	2.4	3.3	1.0	Iris-versicolor
58	6.6	2.9	4.6	1.3	Iris-versicolor
59	5.2	2.7	3.9	1.4	Iris-versicolor
60	5.0	2.0	3.5	1.0	Iris-versicolor
61	5.9	3.0	4.2	1.5	Iris-versicolor
62	6.0	2.2	4.0	1.0	Iris-versicolor
63	6.1	2.9	4.7	1.4	Iris-versicolor
64	5.6	2.9	3.6	1.3	Iris-versicolor
65	6.7	3.1	4.4	1.4	Iris-versicolor
66	5.6	3.0	4.5	1.5	Iris-versicolor
67	5.8	2.7	4.1	1.0	Iris-versicolor
68	6.2	2.2	4.5	1.5	Iris-versicolor
69	5.6	2.5	3.9	1.1	Iris-versicolor
70	5.9	3.2	4.8	1.8	Iris-versicolor
71	6.1	2.8	4.0	1.3	Iris-versicolor
72	6.3	2.5	4.9	1.5	Iris-versicolor
73	6.1	2.8	4.7	1.2	Iris-versicolor
74	6.4	2.9	4.3	1.3	Iris-versicolor
75	6.6	3.0	4.4	1.4	Iris-versicolor
76	6.8	2.8	4.8	1.4	Iris-versicolor
77	6.7	3.0	5.0	1.7	Iris-versicolor
78	6.0	2.9	4.5	1.5	Iris-versicolor
79	5.7	2.6	3.5	1.0	Iris-versicolor
80	5.5	2.4	3.8	1.1	Iris-versicolor
81	5.5	2.4	3.7	1.0	Iris-versicolor

82	5.8	2.7	3.9	1.2	Iris-versicolor
83	6.0	2.7	5.1	1.6	Iris-versicolor
84	5.4	3.0	4.5	1.5	Iris-versicolor
85	6.0	3.4	4.5	1.6	Iris-versicolor
86	6.7	3.1	4.7	1.5	Iris-versicolor
87	6.3	2.3	4.4	1.3	Iris-versicolor
88	5.6	3.0	4.1	1.3	Iris-versicolor
89	5.5	2.5	4.0	1.3	Iris-versicolor
90	5.5	2.6	4.4	1.2	Iris-versicolor
91	6.1	3.0	4.6	1.4	Iris-versicolor
92	5.8	2.6	4.0	1.2	Iris-versicolor
93	5.0	2.3	3.3	1.0	Iris-versicolor
94	5.6	2.7	4.2	1.3	Iris-versicolor
95	5.7	3.0	4.2	1.2	Iris-versicolor
96	5.7	2.9	4.2	1.3	Iris-versicolor
97	6.2	2.9	4.3	1.3	Iris-versicolor
98	5.1	2.5	3.0	1.1	Iris-versicolor
99	5.7	2.8	4.1	1.3	Iris-versicolor

Iris-virginica

	sepal_length	sepal_width	petal_length	petal_width	class
100	6.3	3.3	6.0	2.5	Iris-virginica
101	5.8	2.7	5.1	1.9	Iris-virginica
102	7.1	3.0	5.9	2.1	Iris-virginica
103	6.3	2.9	5.6	1.8	Iris-virginica
104	6.5	3.0	5.8	2.2	Iris-virginica
105	7.6	3.0	6.6	2.1	Iris-virginica
106	4.9	2.5	4.5	1.7	Iris-virginica
107	7.3	2.9	6.3	1.8	Iris-virginica
108	6.7	2.5	5.8	1.8	Iris-virginica
109	7.2	3.6	6.1	2.5	Iris-virginica
110	6.5	3.2	5.1	2.0	Iris-virginica
111	6.4	2.7	5.3	1.9	Iris-virginica
112	6.8	3.0	5.5	2.1	Iris-virginica
113	5.7	2.5	5.0	2.0	Iris-virginica
114	5.8	2.8	5.1	2.4	Iris-virginica
115	6.4	3.2	5.3	2.3	Iris-virginica
116	6.5	3.0	5.5	1.8	Iris-virginica
117	7.7	3.8	6.7	2.2	Iris-virginica
118	7.7	2.6	6.9	2.3	Iris-virginica
119	6.0	2.2	5.0	1.5	Iris-virginica
120	6.9	3.2	5.7	2.3	Iris-virginica
121	5.6	2.8	4.9	2.0	Iris-virginica
122	7.7	2.8	6.7	2.0	Iris-virginica
123	6.3	2.7	4.9	1.8	Iris-virginica
124	6.7	3.3	5.7	2.1	Iris-virginica
125	7.2	3.2	6.0	1.8	Iris-virginica
126	6.2	2.8	4.8	1.8	Iris-virginica
127	6.1	3.0	4.9	1.8	Iris-virginica
128	6.4	2.8	5.6	2.1	Iris-virginica
129	7.2	3.0	5.8	1.6	Iris-virginica
130	7.4	2.8	6.1	1.9	Iris-virginica
131	7.9	3.8	6.4	2.0	Iris-virginica
132	6.4	2.8	5.6	2.2	Iris-virginica
133	6.3	2.8	5.1	1.5	Iris-virginica
134	6.1	2.6	5.6	1.4	Iris-virginica
135	7.7	3.0	6.1	2.3	Iris-virginica
136	6.3	3.4	5.6	2.4	Iris-virginica
137	6.4	3.1	5.5	1.8	Iris-virginica
138	6.0	3.0	4.8	1.8	Iris-virginica
139	6.9	3.1	5.4	2.1	Iris-virginica
140	6.7	3.1	5.6	2.4	Iris-virginica
141	6.9	3.1	5.1	2.3	Iris-virginica
142	5.8	2.7	5.1	1.9	Iris-virginica
143	6.8	3.2	5.9	2.3	Iris-virginica
144	6.7	3.3	5.7	2.5	Iris-virginica
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica

```
148            6.2           3.4            5.4           2.3  Iris-virginica
149            5.9           3.0            5.1           1.8  Iris-virginica
```

分组可以快速实现MapReduce的逻辑

- Map: 指定分组的列标签，不同的值就会被扔到不同的分组处理
- Reduce: 输入多个值，返回一个值，一般可以通过agg实现，agg能接受一个函数

In [67]:

```
irisdata.groupby('class').agg(\
        lambda x:  ((x-x.mean())**3).sum()*len(x)/(len(x)-1)/(len(x)-2)/x.std()**3 if len(x)>2
else None)
```

Out[67]:

	sepal_length	sepal_width	petal_length	petal_width
class				
Iris-setosa	0.120087	0.107053	0.071846	1.197243
Iris-versicolor	0.105378	-0.362845	-0.606508	-0.031180
Iris-virginica	0.118015	0.365949	0.549445	-0.129477

汇总之后的广播操作

在OLAP数据库上，为了避免groupby+join的二次操作，提出了sum()over(partition by)的开窗操作。

在Pandas中，这种操作能够进一步被transform所取代。

In [188]:

```
pd.concat([irisdata,irisdata.groupby('class').transform('mean')],axis=1)
```

Out[188]:

	sepal_length	sepal_width	petal_length	petal_width	class	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2	Iris-setosa	5.006	3.418	1.464	0.244
1	4.9	3.0	1.4	0.2	Iris-setosa	5.006	3.418	1.464	0.244
2	4.7	3.2	1.3	0.2	Iris-setosa	5.006	3.418	1.464	0.244
3	4.6	3.1	1.5	0.2	Iris-setosa	5.006	3.418	1.464	0.244
4	5.0	3.6	1.4	0.2	Iris-setosa	5.006	3.418	1.464	0.244
5	5.4	3.9	1.7	0.4	Iris-setosa	5.006	3.418	1.464	0.244
6	4.6	3.4	1.4	0.3	Iris-setosa	5.006	3.418	1.464	0.244
7	5.0	3.4	1.5	0.2	Iris-setosa	5.006	3.418	1.464	0.244
8	4.4	2.9	1.4	0.2	Iris-setosa	5.006	3.418	1.464	0.244
					Iris-				

9	4.9	3.1	1.5	0.1	setosa	5.006	3.418	1.464	0.244
10	5.4	3.7	1.5	0.2	Iris-setosa	5.006	3.418	1.464	0.244
11	4.8	3.4	1.6	0.2	Iris-setosa	5.006	3.418	1.464	0.244
12	4.8	3.0	1.4	0.1	Iris-setosa	5.006	3.418	1.464	0.244
13	4.3	3.0	1.1	0.1	Iris-setosa	5.006	3.418	1.464	0.244
14	5.8	4.0	1.2	0.2	Iris-setosa	5.006	3.418	1.464	0.244
15	5.7	4.4	1.5	0.4	Iris-setosa	5.006	3.418	1.464	0.244
16	5.4	3.9	1.3	0.4	Iris-setosa	5.006	3.418	1.464	0.244
17	5.1	3.5	1.4	0.3	Iris-setosa	5.006	3.418	1.464	0.244
18	5.7	3.8	1.7	0.3	Iris-setosa	5.006	3.418	1.464	0.244
19	5.1	3.8	1.5	0.3	Iris-setosa	5.006	3.418	1.464	0.244
20	5.4	3.4	1.7	0.2	Iris-setosa	5.006	3.418	1.464	0.244
21	5.1	3.7	1.5	0.4	Iris-setosa	5.006	3.418	1.464	0.244
22	4.6	3.6	1.0	0.2	Iris-setosa	5.006	3.418	1.464	0.244
23	5.1	3.3	1.7	0.5	Iris-setosa	5.006	3.418	1.464	0.244
24	4.8	3.4	1.9	0.2	Iris-setosa	5.006	3.418	1.464	0.244
25	5.0	3.0	1.6	0.2	Iris-setosa	5.006	3.418	1.464	0.244
26	5.0	3.4	1.6	0.4	Iris-setosa	5.006	3.418	1.464	0.244
27	5.2	3.5	1.5	0.2	Iris-setosa	5.006	3.418	1.464	0.244
28	5.2	3.4	1.4	0.2	Iris-setosa	5.006	3.418	1.464	0.244
29	4.7	3.2	1.6	0.2	Iris-setosa	5.006	3.418	1.464	0.244
...	...	...	...	...	...	...	...	...	...
120	6.9	3.2	5.7	2.3	Iris-virginica	6.588	2.974	5.552	2.026
121	5.6	2.8	4.9	2.0	Iris-virginica	6.588	2.974	5.552	2.026

122	7.7	2.8	6.7	2.0	Iris-virginica	6.588	2.974	5.552	2.026
123	6.3	2.7	4.9	1.8	Iris-virginica	6.588	2.974	5.552	2.026
124	6.7	3.3	5.7	2.1	Iris-virginica	6.588	2.974	5.552	2.026
125	7.2	3.2	6.0	1.8	Iris-virginica	6.588	2.974	5.552	2.026
126	6.2	2.8	4.8	1.8	Iris-virginica	6.588	2.974	5.552	2.026
127	6.1	3.0	4.9	1.8	Iris-virginica	6.588	2.974	5.552	2.026
128	6.4	2.8	5.6	2.1	Iris-virginica	6.588	2.974	5.552	2.026
129	7.2	3.0	5.8	1.6	Iris-virginica	6.588	2.974	5.552	2.026
130	7.4	2.8	6.1	1.9	Iris-virginica	6.588	2.974	5.552	2.026
131	7.9	3.8	6.4	2.0	Iris-virginica	6.588	2.974	5.552	2.026
132	6.4	2.8	5.6	2.2	Iris-virginica	6.588	2.974	5.552	2.026
133	6.3	2.8	5.1	1.5	Iris-virginica	6.588	2.974	5.552	2.026
134	6.1	2.6	5.6	1.4	Iris-virginica	6.588	2.974	5.552	2.026
135	7.7	3.0	6.1	2.3	Iris-virginica	6.588	2.974	5.552	2.026
136	6.3	3.4	5.6	2.4	Iris-virginica	6.588	2.974	5.552	2.026
137	6.4	3.1	5.5	1.8	Iris-virginica	6.588	2.974	5.552	2.026
138	6.0	3.0	4.8	1.8	Iris-virginica	6.588	2.974	5.552	2.026
139	6.9	3.1	5.4	2.1	Iris-virginica	6.588	2.974	5.552	2.026
140	6.7	3.1	5.6	2.4	Iris-virginica	6.588	2.974	5.552	2.026
141	6.9	3.1	5.1	2.3	Iris-virginica	6.588	2.974	5.552	2.026
142	5.8	2.7	5.1	1.9	Iris-virginica	6.588	2.974	5.552	2.026
143	6.8	3.2	5.9	2.3	Iris-virginica	6.588	2.974	5.552	2.026
144	6.7	3.3	5.7	2.5	Iris-virginica	6.588	2.974	5.552	2.026
145	6.7	3.0	5.2	2.3	Iris-virginica	6.588	2.974	5.552	2.026

146	6.3	2.5	5.0	1.9	Iris-virginica	6.588	2.974	5.552	2.026
147	6.5	3.0	5.2	2.0	Iris-virginica	6.588	2.974	5.552	2.026
148	6.2	3.4	5.4	2.3	Iris-virginica	6.588	2.974	5.552	2.026
149	5.9	3.0	5.1	1.8	Iris-virginica	6.588	2.974	5.552	2.026

150 rows × 9 columns

HierarchicalIndex（多列分组）后的数据透视表操作

一般来说，多列groupby的一个副作用就是.groupby().agg()之后你的行index已经变成了一个多列分组的分级索引。

如果我们希望达到Excel的数据透视表的效果，行和列的索引自由交换，达到统计目的，究竟应该怎么办呢?

In [53]:

```
factor1 = np.random.randint(0,3,50)
factor2 = np.random.randint(0,2,50)
factor3 = np.random.randint(0,3,50)
values = np.random.randn(50)
```

In [59]:

```
hierindexDF = pd.DataFrame({'F1':factor1,'F2':factor2,'F3':factor3,'F4':values})
hierindexDF
```

Out[59]:

	F1	F2	F3	F4
0	1	0	1	0.222404
1	0	0	1	-0.548535
2	0	1	2	-1.440883
3	2	1	2	-0.014220
4	0	0	1	0.376058
5	1	1	0	-1.407690
6	0	1	1	1.250202
7	1	1	2	-0.965264
8	1	0	2	-0.168896
9	2	1	1	-1.263269
10	0	0	1	-0.275786
11	1	0	2	-0.315656
12	2	1	0	-0.627063
13	0	0	0	-2.176746
14	2	1	1	1.845389
15	1	1	0	-1.467753
16	2	1	1	0.460780
17	2	0	0	-0.596958

18	1	1	2	1.219037
19	0	1	1	0.185074
20	1	1	0	-0.032669
21	2	1	0	0.247771
22	1	0	1	-1.582100
23	1	1	0	-0.725630
24	0	1	0	-0.441145
25	2	1	1	1.807921
26	1	1	0	-1.508745
27	0	1	0	-0.007337
28	1	0	2	0.390230
29	0	0	0	0.164061
30	0	0	0	-1.594304
31	2	0	2	1.631381
32	2	1	1	-0.336894
33	2	1	0	-0.666307
34	1	1	1	-1.909391
35	2	0	2	-0.348489
36	1	0	1	0.363768
37	0	0	1	-0.992691
38	2	1	2	-0.074077
39	2	0	0	-1.060918
40	2	0	0	-0.227080
41	0	0	1	0.618373
42	1	1	0	1.451127
43	0	0	0	2.570664
44	1	1	1	2.239915
45	1	1	2	0.028498
46	2	1	1	-0.498728
47	1	1	2	0.490157
48	0	0	2	0.875079
49	0	0	0	-0.412605

In [71]:

```
hierindexDF_gbsum = hierindexDF.groupby(['F1','F2','F3']).sum()
hierindexDF_gbsum
```

Out[71]:

			F4
F1	**F2**	**F3**	
0	**0**	**0**	-1.448930
		1	-0.822580
		2	0.875079
	1	**0**	-0.448482
		1	1.435276
		2	-1.440883
1	**0**	**1**	-0.995928
		2	-0.094322
	1	**0**	-3.691359
		1	0.330524
		2	0.772429
2	**0**	**0**	-1.884956
		2	1.282892
	1	**0**	-1.045598
		1	2.015199
		2	-0.088298

观察Index：

In [73]:

```
hierindexDF_gbsum.index
```

Out[73]:

```
MultiIndex(levels=[[0, 1, 2], [0, 1], [0, 1, 2]],
           labels=[[0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2], [0, 0, 0, 1, 1, 1, 0, 0,
1, 1, 1, 0, 0, 1, 1, 1], [0, 1, 2, 0, 1, 2, 1, 2, 0, 1, 2, 0, 2, 0, 1, 2]],
           names=[u'F1', u'F2', u'F3'])
```

unstack：

- 无参数时，把最末index置换到column上
- 有数字参数时，把指定位置的index置换到column上
- 有列表参数时，依次把特定位置的index置换到column上

In [119]:

```
hierindexDF_gbsum.unstack(0)
```

Out[119]:

		F4		
	F1	0	1	2
F2	F3			
0	0	-1.448930	NaN	-1.884956
	1	-0.822580	-0.995928	NaN
	2	0.875079	-0.094322	1.282892
1	0	-0.448482	-3.691359	-1.045598
	1	1.435276	0.330524	2.015199
	2	-1.440883	0.772429	-0.088298

In [114]:

```
hierindexDF_gbsum.unstack(1)
```

Out[114]:

		F4	
	F2	0	1
F1	F3		
0	0	-1.448930	-0.448482
	1	-0.822580	1.435276
	2	0.875079	-1.440883
1	0	NaN	-3.691359
	1	-0.995928	0.330524
	2	-0.094322	0.772429
2	0	-1.884956	-1.045598
	1	NaN	2.015199
	2	1.282892	-0.088298

In [89]:

```
hierindexDF_gbsum.unstack([2,0])
```

Out[89]:

	F4								
F3	0	1	2	1	2	0		2	1
F1	0	0	0	1	1	1	2	2	2
F2									
0	-1.448930	-0.822580	0.875079	-0.995928	-0.094322	NaN	-1.884956	1.282892	NaN
1	-0.448482	1.435276	-1.440883	0.330524	0.772429	-3.691359	-1.045598	-0.088298	2.015199

更进一步的，stack的功能是和unstack对应，把column上的多级索引换到index上去

In [92]:

```
hierindexDF_gbsum.unstack([2,0]).stack([1,2])
```

Out[92]:

			F4
F2	F3	F1	
0	0	0	-1.448930
		2	-1.884956
	1	0	-0.822580
		1	-0.995928
	2	0	0.875079
		1	-0.094322
		2	1.282892
1	0	0	-0.448482
		1	-3.691359
		2	-1.045598
	1	0	1.435276
		1	0.330524
		2	2.015199
	2	0	-1.440883
		1	0.772429
		2	-0.088298

In [ ]:

```

```