第 14 章　Python 数据分析与处理

14.1 pandas 基本操作

- pandas 主要提供了 3 种数据结构：
  - Series ，带标签的一维数组；
  - DataFrame ，带标签且大小可变的二维表格结构；
  - Panel ，带标签且大小可变的三维数组。

14.1 pandas 基本操作

1. 生成一维数组

```
>>> import numpy as np
>>> import pandas as pd
>>> x = pd.Series([1, 3, 5, np.nan])
>>> x
0    1.0
1    3.0
2    5.0
3    NaN
dtype: float64
```

Numpy. array([1, 3, 5, np.nan])

14.1 pandas 基本操作

```
>>> pd.Series(range(5))                    # 把 Python 的 range 对象转换为一维数组
0    0
1    1
2    2
3    3
4    4
dtype: int32
>>> pd.Series(range(5), index=list('abcde'))
 #指定索引
```

与 pd.Series(range(5), index=['a','b','c','d','e']) 功能相同

14.1 pandas 基本操作

```
>>> pd.date_range(start='20180101', end='20181231', freq='H')
                                                        # 间隔为小时
DatetimeIndex(['2018-01-01 00:00:00', '2018-01-01 01:00:00',
               '2018-01-01 02:00:00', '2018-01-01 03:00:00',
               '2018-01-01 04:00:00', '2018-01-01 05:00:00',
               '2018-01-01 06:00:00', '2018-01-01 07:00:00',
               '2018-01-01 08:00:00', '2018-01-01 09:00:00',
               ...
               '2018-12-30 15:00:00', '2018-12-30 16:00:00',
               '2018-12-30 17:00:00', '2018-12-30 18:00:00',
               '2018-12-30 19:00:00', '2018-12-30 20:00:00',
```

工于建构 成于创造

14.1 pandas 基本操作

```
>>> pd.date_range(start='20180101', end='20181231', freq='D')
                                                        # 间隔为天
DatetimeIndex(['2018-01-01', '2018-01-02', '2018-01-03','2018-01-04',
               '2018-01-05', '2018-01-06', '2018-01-07', '2018-01-08',
               '2018-01-09', '2018-01-10',
               ...
               '2018-12-22', '2018-12-23', '2018-12-24', '2018-12-25',
               '2018-12-26', '2018-12-27', '2018-12-28', '2018-12-29',
               '2018-12-30', '2018-12-31'],
              dtype='datetime64[ns]', length=365, freq='D')
```

14.1 pandas 基本操作

```
>>> dates = pd.date_range(start='20180101', end='20181231', freq='M')
                                                    # 间隔为月
>>> dates
DatetimeIndex(['2018-01-31', '2018-02-28', '2018-03-31','2018-04-30',
               '2018-05-31', '2018-06-30', '2018-07-31', '2018-08-31',
               '2018-09-30', '2018-10-31', '2018-11-30', '2018-12-31'],
              dtype='datetime64[ns]', freq='M')
```

14.1 pandas 基本操作

2. 二维数组 DataFrame 的操作

（1）生成二维数组

```
>>> pd.DataFrame(np.random.randn(12,4),     # 数据
                 index=dates,                # 索引
                 columns=list('ABCD'))       # 列名
                   A         B         C         D
2018-01-31  1.060900  0.697288 -0.058990 -0.487499
2018-02-28 -0.353329  1.160652 -0.277649  1.076614
2018-03-31  2.323984 -0.435853 -0.591344 -0.754395
2018-04-30 -0.077860 -0.432890  1.318615  0.125510
2018-05-31 -0.993383 -1.064773 -0.430447 -3.073572
```

14.1 pandas 基本操作

```
>>> df = pd.DataFrame({'A':np.random.randint(1, 100, 4),
                       'B':pd.date_range(start='20180301', periods=4, freq='D'),
                       'C':pd.Series([1, 2, 3, 4],
                                     index=['zhang', 'li', 'zhou', 'wang'],
                                     dtype='float32'),
                       'D':np.array([3] * 4,dtype='int32'),
                       'E':pd.Categorical(["test","train","test","train"]),
                       'F':'foo'})
>>> df
        A          B    C  D      E    F
zhang  60 2018-03-01  1.0  3   test  foo
li     36 2018-03-02  2.0  3  train  foo
```

14.1 pandas 基本操作

（ 2 ）查看二维数组数据

```
>>> df.head()          # 默认显示前 5 行，不过这里的 df 只有 4 行数据
        A          B    C  D      E    F
zhang  60 2018-03-01  1.0  3   test  foo
li     36 2018-03-02  2.0  3  train  foo
zhou   45 2018-03-03  3.0  3   test  foo
wang   98 2018-03-04  4.0  3  train  foo
>>> df.head(3)         # 查看前 3 行
        A          B    C  D      E    F
zhang  60 2018-03-01  1.0  3   test  foo
li     36 2018-03-02  2.0  3  train  foo
zhou   45 2018-03-03  3.0  3   test  foo
```

14.1 pandas 基本操作

（ 3 ）查看二维数组数据的索引、列名和值

```
>>> df.index                    # 查看索引
Index(['zhang', 'li', 'zhou', 'wang'], dtype='object')
>>> df.columns                  # 查看列名
Index(['A', 'B', 'C', 'D', 'E', 'F'], dtype='object')
>>> df.values                   # 查看值
array([[60, Timestamp('2018-03-01 00:00:00'), 1.0, 3, 'test', 'foo'],
       [36, Timestamp('2018-03-02 00:00:00'), 2.0, 3, 'train', 'foo'],
       [45, Timestamp('2018-03-03 00:00:00'), 3.0, 3, 'test', 'foo'],
       [98, Timestamp('2018-03-04 00:00:00'), 4.0, 3, 'train', 'foo']], d
type=object)
```

14.1 pandas 基本操作

（4）查看二维数组数据的统计信息

```
>>> df.describe()    # 平均值、标准差、最小值、最大值等信息
                A          C    D
count    4.000000   4.000000  4.0
mean    59.750000   2.500000  3.0
std     27.354159   1.290994  0.0
min     36.000000   1.000000  3.0
25%     42.750000   1.750000  3.0
50%     52.500000   2.500000  3.0
75%     69.500000   3.250000  3.0
max     98.000000   4.000000  3.0
```

```
         A          B    C  D      E    F
zhang   60 2018-03-01  1.0  3   test  foo
li      36 2018-03-02  2.0  3  train  foo
zhou    45 2018-03-03  3.0  3   test  foo
wang    98 2018-03-04  4.0  3  train  foo
```

14.1 pandas 基本操作

（ 5 ）对二维数组进行排序操作

```
>>> df.sort_index(axis=0, ascending=False) # 对索引进行降序排序
        A          B    C  D      E    F
zhou    45 2018-03-03  3.0  3   test  foo
zhang   60 2018-03-01  1.0  3   test  foo
wang    98 2018-03-04  4.0  3  train  foo
li      36 2018-03-02  2.0  3  train  foo
```

```
        A          B    C  D      E    F
zhang   60 2018-03-01  1.0  3   test  foo
li      36 2018-03-02  2.0  3  train  foo
zhou    45 2018-03-03  3.0  3   test  foo
wang    98 2018-03-04  4.0  3  train  foo
```

14.1 pandas 基本操作

```
>>> df.sort_index(axis=0, ascending=True)  # 对索引升序排序
        A          B    C  D      E    F
li     36 2018-03-02  2.0  3  train  foo
wang   98 2018-03-04  4.0  3  train  foo
zhang  60 2018-03-01  1.0  3   test  foo
zhou   45 2018-03-03  3.0  3   test  foo
```

14.1 pandas 基本操作

```
>>> df.sort_index(axis=1, ascending=False) # 对列进行降序排序
```

	F	E	D	C	B	A
zhang	foo	test	3	1.0	2018-03-01	60
li	foo	train	3	2.0	2018-03-02	36
zhou	foo	test	3	3.0	2018-03-03	45
wang	foo	train	3	4.0	2018-03-04	98

	A	B	C	D	E	F
zhang	60	2018-03-01	1.0	3	test	foo
li	36	2018-03-02	2.0	3	train	foo
zhou	45	2018-03-03	3.0	3	test	foo
wang	98	2018-03-04	4.0	3	train	foo

14.1 pandas 基本操作

```
>>> df.sort_values(by='A')                    # 按 A 列对数据进行升序排序
        A          B    C  D      E    F
li      36 2018-03-02  2.0  3  train  foo
zhou    45 2018-03-03  3.0  3   test  foo
zhang   60 2018-03-01  1.0  3   test  foo
wang    98 2018-03-04  4.0  3  train  foo
```

14.1 pandas 基本操作

```
>>> df.sort_values(by=['E', 'C'])          # 先按 E 列升序排序
                                           # 如果 E 列相同，再按 C 列升序排序
        A           B    C  D      E    F
zhang  60 2018-03-01  1.0  3   test  foo
zhou   45 2018-03-03  3.0  3   test  foo
li     36 2018-03-02  2.0  3  train  foo
wang   98 2018-03-04  4.0  3  train  foo
```

14.1 pandas 基本操作

（7）二维数组数据的选择与访问

```
>>> df['A']                                  # 选择某一列数据
zhang    60
li       36
zhou     45
wang     98
Name: A, dtype: int32
>>> 60 in df['A']                            #df['A'] 是一个类似于字典的结构
                                             # 索引类似于字典的键
                                             # 默认是访问字典的键，而不是值
False
```

14.1 pandas 基本操作

```
>>> df[0:2]                                    # 使用切片选择多行
        A          B    C  D      E    F
zhang  60 2018-03-01  1.0  3   test  foo
li     36 2018-03-02  2.0  3  train  foo
>>> df.loc[:, ['A', 'C']]                      # 选择多列（行，列）
        A    C
zhang  60  1.0
li     36  2.0
zhou   45  3.0
wang   98  4.0
```

14.1 pandas 基本操作

```
>>> df.loc[['zhang', 'zhou'], ['A', 'D', 'E']]# 同时指定多行和多列
        A  D     E
zhang  60  3  test
zhou   45  3  test
>>> df.loc['zhang', ['A', 'D', 'E']]          # 查看 'zhang' 的三列数据
A      60
D       3
E    test
Name: zhang, dtype: object
>>> df.loc[['zhang'], ['A', 'D', 'E']]          # 与上例数据显示的方式不同
        A  D     E
zhang  78  3  test
```

14.1 pandas 基本操作

```
>>> df.iloc[3]                    # 查询二维数组第 3 行数据
A                          98
B     2018-03-04 00:00:00
C                           4
D                           3
E                       train
F                         foo
Name: wang, dtype: object
>>> df.iloc[0:3, 0:4]             # 查询二维数组前
          A           B    C  D
zhang    60  2018-03-01  1.0  3
li       36  2018-03-02  2.0  3
zhou     45  2018-03-03  3.0  3
wang     98  2018-03-04  4.0  3
```

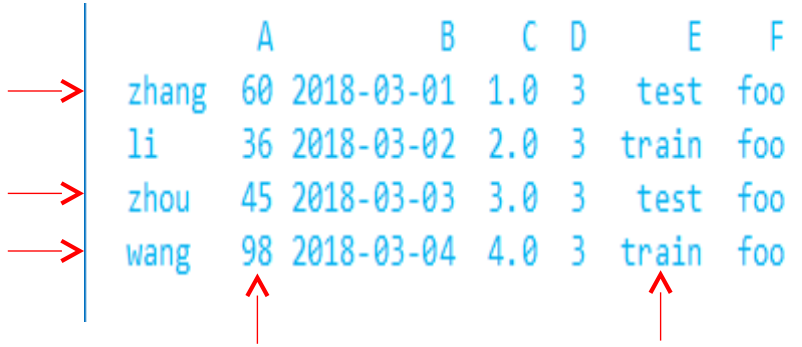
	A	B	C	D	E	F
zhang	60	2018-03-01	1.0	3	test	foo
li	36	2018-03-02	2.0	3	train	foo
zhou	45	2018-03-03	3.0	3	test	foo
wang	98	2018-03-04	4.0	3	train	foo

	A	B	C	D	E	F
zhang	60	2018-03-01	1.0	3	test	foo
li	36	2018-03-02	2.0	3	train	foo
zhou	45	2018-03-03	3.0	3	test	foo
wang	98	2018-03-04	4.0	3	train	foo

14.1 pandas 基本操作

```
>>> df.iloc[[0, 2, 3], [0, 4]]
         A      E
zhang   60   test
zhou    45   test
wang    98  train
>>> df.iloc[0,1]
Timestamp('2018-03-01 00:00:00')
>>> df.iloc[2,2]
3.0
```

查询二维数组指定的多行、多列数据



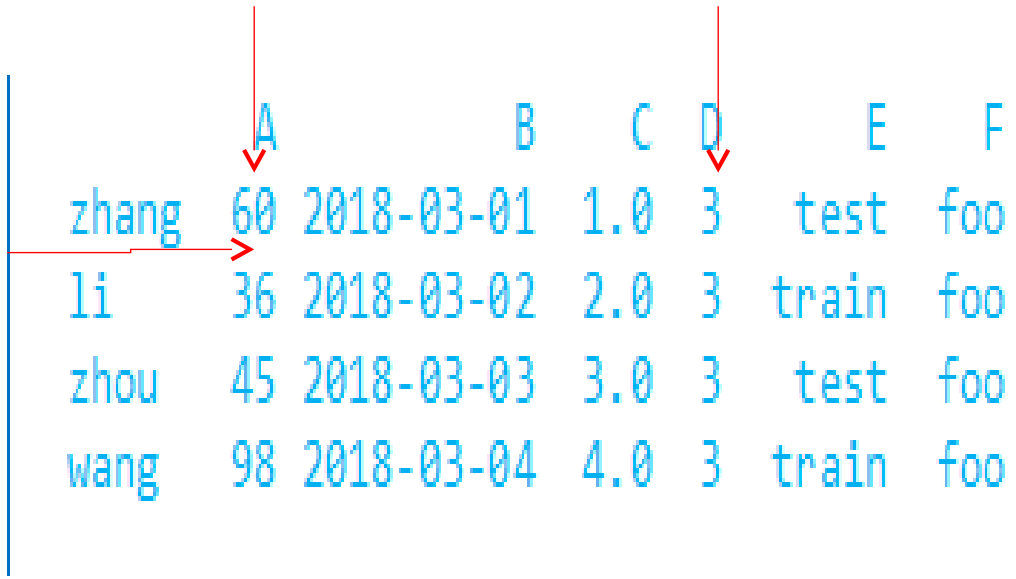
	A	B	C	D	E	F
zhang	60	2018-03-01	1.0	3	test	foo
li	36	2018-03-02	2.0	3	train	foo
zhou	45	2018-03-03	3.0	3	test	foo
wang	98	2018-03-04	4.0	3	train	foo

查询二维数组第 0 行第 1 列位置的数据值

查询二维数组第 2 行第 2 列位置的数据值

14.1 pandas 基本操作

```
>>> df.at['zhang', 'A']          # 查询指定行、列位置的数据值
60
>>> df.at['zhang', 'D']
3
```



	A	B	C	D	E	F
zhang	60	2018-03-01	1.0	3	test	foo
li	36	2018-03-02	2.0	3	train	foo
zhou	45	2018-03-03	3.0	3	test	foo
wang	98	2018-03-04	4.0	3	train	foo

14.1 pandas 基本操作

```
>>> df[df.A>50]                          # 查询 A 列大于 50 的所有行
        A          B    C  D      E    F
zhang  60 2018-03-01  1.0  3   test  foo
wang   98 2018-03-04  4.0  3  train  foo
>>> df[df['E']=='test']                  # 查询 E 列为 'test' 的所有行
        A          B    C  D     E    F
zhang  60 2018-03-01  1.0  3  test  foo
zhou   45 2018-03-03  3.0  3  test  foo
```

14.1 pandas 基本操作

```
>>> df.nlargest(3, ['C'])          # 返回 C 列值最大的前 3 行
       A          B    C  D      E    F
wang  98 2018-03-04  4.0  3  train  foo
zhou  45 2018-03-03  3.0  3   test  foo
li    36 2018-03-02  2.0  3  train  foo
>>> df.nlargest(3, ['A'])          # 返回 A 列值最大的前 3 行
        A          B    C  D      E    F
wang   98 2018-03-04  4.0  3  train  foo
zhang  60 2018-03-01  1.0  3   test  foo
zhou   45 2018-03-03  3.0  3   test  foo
```

14.1 pandas 基本操作

（8）二维数组的数据修改

```
>>> df.iat[0, 2] = 3                          # 修改指定行、列位置的数据值
>>> df.loc[:, 'D'] = np.random.randint(50, 60, 4)
                                              # 修改某列的值
>>> df['C'] = -df['C']                        # 对指定列数据取反
>>> df                                        # 查看上面三个修改操作的最终结果
```

	A	B	C	D	E	F
zhang	60	2018-03-01	-3.0	52	test	foo
li	36	2018-03-02	-2.0	52	train	foo
zhou	45	2018-03-03	-3.0	59	test	foo
wang	98	2018-03-04	-4.0	54	train	foo

	A	B	C	D	E	F
zhang	60	2018-03-01	1.0	3	test	foo
li	36	2018-03-02	2.0	3	train	foo
zhou	45	2018-03-03	3.0	3	test	foo
wang	98	2018-03-04	4.0	3	train	foo

14.1 pandas 基本操作

```
>>> dff = df[:]                          # 切片（：所有行所有列）
>>> dff['C'] = dff['C'] ** 2             # 替换列数据
>>> dff
          A          B     C   D      E    F
zhang    60 2018-03-01   9.0  52   test  foo
li       36 2018-03-02   4.0  52  train  foo
zhou     45 2018-03-03   9.0  59   test  foo
wang     98 2018-03-04  16.0  54  train  foo
```

14.1 pandas 基本操作

```
>>> data = pd.DataFrame({'k1':['one'] * 3 + ['two'] * 4,
                         'k2':[1, 1, 2, 3, 3, 4, 4]})
>>> data.replace(1, 5)                          # 把所有 1 替换为 5
    k1  k2
0  one   5
1  one   5
2  one   2
3  two   3
4  two   3
5  two   4
6  two   4
```

14.1 pandas 基本操作

（ 9 ）二维数组数据预处理

1 ）二维数组缺失值的处理。

```
>>> df1 = df.reindex(columns=list(df.columns) + ['G'])
                                                    # 增加一列，列名为 G
>>> df1                                             # 其中 NaN 表示缺失值
        A           B     C   D      E    F   G
zhang  60 2018-03-01   9.0  52   test  foo NaN
li     36 2018-03-02   4.0  52  train  foo NaN
zhou   45 2018-03-03   9.0  59   test  foo NaN
wang   98 2018-03-04  16.0  54  train  foo NaN
```

14.1 pandas 基本操作

```
>>> df1.iat[0, 6] = 3      # 修改指定位置元素值，该列其他元素仍为缺失值
        A          B     C   D      E    F    G
zhang  60 2018-03-01   9.0  52   test  foo  3.0
li     36 2018-03-02   4.0  52  train  foo  NaN
zhou   45 2018-03-03   9.0  59   test  foo  NaN
wang   98 2018-03-04  16.0  54  train  foo  NaN
>>> df1.dropna()           # 返回不包含缺失值的行
        A          B    C   D      E    F    G
zhang  60 2018-03-01  9.0  52  test  foo  3.0
```

14.1 pandas 基本操作

```
>>> df1['G'].fillna(5, inplace=True)  # 使用指定值原地填充缺失值
>>> df1
         A           B     C   D      E    F    G
zhang   60 2018-03-01   9.0  52   test  foo  3.0
li      36 2018-03-02   4.0  52  train  foo  5.0
zhou    45 2018-03-03   9.0  59   test  foo  5.0
wang    98 2018-03-04  16.0  54  train  foo  5.0
```

14.1 pandas 基本操作

2 ）二维数组重复值的处理。

```
>>> data = pd.DataFrame({'k1':['one'] * 3 + ['two'] * 4,
                         'k2':[1, 1, 2, 3, 3, 4, 4]})

>>> data
    k1  k2
0  one   1
1  one   1
2  one   2
3  two   3
4  two   3
5  two   4
```

14.1 pandas 基本操作

	k1	k2
0	one	1
1	one	1
2	one	2
3	two	3
4	two	3
5	two	4
6	two	4

```
>>> data.drop_duplicates()          # 返回新数组，删除重复行
    k1  k2
0  one   1
2  one   2
3  two   3
5  two   4
>>> data.drop_duplicates(['k1'])    # 删除 k1 列的重复数据，保留第一条数据
    k1  k2
0  one   1
3  two   3
>>> data.drop_duplicates(['k1'], keep='last')
                                    # 对于重复的数据，只保留最后一个
```

14.1 pandas 基本操作

3 ）二维数组异常值的处理。

```
>>> import numpy as np
>>> import pandas as pd
>>> data = pd.DataFrame(np.random.randn(500, 4))
>>> data.describe()                                    # 查看数据的统计信息
                0           1           2
3
count  500.000000  500.000000  500.000000  500.00000
0
mean    -0.077138    0.052644   -0.045360    0.02427
```

14.1 pandas 基本操作

```
>>> col2 = data[2]                    # 获取第 2 列的数据
>>> col2[col2>3.5]                    # 查询该列中大于 3.5 的数值
12    3.986027
Name: 2, dtype: float64
>>> col2[col2>3.0]                    # 查看该列中大于 3.0 的数值
12    3.986027
Name: 2, dtype: float64
>>> col2[col2>2.5]                    # 查看该列中大于 2.5 的数值
11     2.528325
12     3.986027
41     2.775205
157    2.707940
```

14.1 pandas 基本操作

（12）二维数组的映射

```
>>> data = pd.DataFrame({'k1':['one'] * 3 + ['two'] * 4,
                         'k2':[1, 1, 2, 3, 3, 4, 4]})
>>> data
    k1  k2
0  one   1
1  one   1
2  one   2
3  two   3
4  two   3
5  two   4
```

14.1 pandas 基本操作

```
>>> data['k1'] = data['k1'].map(str.upper)  # 使用函数进行映射
>>> data
    k1  k2
0  ONE   1
1  ONE   1
2  ONE   2
3  TWO   3
4  TWO   3
5  TWO   4
6  TWO   4
```

14.1 pandas 基本操作

```
>>> data['k1'] = data['k1'].map({'ONE':'one', 'TWO':'two'})
                                        # 使用字典表示映射关系

>>> data
    k1  k2
0  one   1
1  one   1
2  one   2
3  two   3
4  two   3
5  two   4
6  two   4
```

14.1 pandas 基本操作

（13）二维数组数据离散化

```
>>> from random import randrange
>>> data = [randrange(100) for _ in range(10)]  # 生成随机数
>>> data
[89, 55, 79, 73, 90, 69, 92, 46, 37, 37]
>>> category = [0, 30, 70, 100]                 # 指定数据切分的区间边界
>>> pd.cut(data, category)
[(70, 100], (30, 70], (70, 100], (70, 100], (70, 100], (30, 70], (70, 10
0], (30, 70], (30, 70], (30, 70]]
Categories (3, interval[int64]): [(0, 30] < (30, 70] < (70, 100]]
>>> pd.cut(data, category, right=False)         # 左闭右开区间
[[70, 100), [30, 70), [70, 100), [70, 100), [70, 100), [30, 70), [70, 10
```

14.1 pandas 基本操作

```
>>> labels = ['low', 'middle', 'high']
>>> pd.cut(data, category, right=False, labels=labels) # 指定标签
[high, middle, high, high, high, middle, high, middle, middle, middle]
Categories (3, object): [low < middle < high ]
```

14.1 pandas 基本操作

（ 14 ）移位与频次统计

```
>>> df1
          A          B     C   D      E    F    G
zhang    60 2018-03-01   9.0  52   test  foo  3.0
li       36 2018-03-02   4.0  52  train  foo  5.0
zhou     45 2018-03-03   9.0  59   test  foo  5.0
wang     98 2018-03-04  16.0  54  train  foo  5.0
>>> df1.shift(1)                          # 数据下移一行
            A          B    C     D      E    F    G
zhang     NaN        NaT  NaN   NaN    NaN  NaN  NaN
li       60.0 2018-03-01  9.0  52.0   test  foo  3.0
zhou     36.0 2018-03-02  4.0  52.0  train  foo  5.0
wang     45.0 2018-03-03  9.0  59.0   test  foo  5.0
```

14.1 pandas 基本操作

```
>>> df1['D'].value_counts()                    #  统计 D 列数据分布情况
52    2
59    1
54    1
Name: D, dtype: int64
>>> df1['G'].value_counts()                    # 统计 G 列数据分布情况
5.0    3
3.0    1
Name: G, dtype: int64
```

14.1 pandas 基本操作

（15）拆分与合并/连接

```
>>> df2 = pd.DataFrame(np.random.randn(10, 4))
>>> df2
          0         1         2         3
0  2.064867 -0.888018  0.586441 -0.660901
1 -0.465664 -0.496101  0.249952  0.627771
2  1.974986  1.304449 -0.168889 -0.334622
3  0.715677  2.017427  1.750627 -0.787901
4 -0.370020 -0.878282  0.499584  0.269102
```

14.1 pandas 基本操作

```
>>> p1 = df2[:3]                          # 拆分，得到前 3 行数据
>>> p1
          0         1         2         3
0  2.064867 -0.888018  0.586441 -0.660901
1 -0.465664 -0.496101  0.249952  0.627771
2  1.974986  1.304449 -0.168889 -0.334622
>>> p2 = df2[3:7]                         # 获取第 3 到 6 行数据
>>> p3 = df2[7:]                          # 获取下标为 7 之后所有行的数据
>>> df3 = pd.concat([p1, p2, p3])         # 数据行合并
```

14.1 pandas 基本操作

（ 16 ）分组计算

```
>>> df4 = pd.DataFrame({'A':np.random.randint(1,5,8),
                        'B':np.random.randint(10,15,8),
                        'C':np.random.randint(20,30,8),
                        'D':np.random.randint(80,100,8)})
```

```
>>> df4
   A   B   C   D
0  4  14  29  84
1  3  10  28  86
2  3  10  24  83
3  2  13  21  80
```

14.1 pandas 基本操作

```
>>> df4.groupby('A').sum()                    # 数据分组计算
    B    C     D
A
1   32   81   268
2   13   21    80
3   20   52   169
4   27   49   182
>>> df4.groupby(['A','B']).mean()
          C     D
A B
1 10   27.0  91.0
  11   27.0  88.5
```

14.1 pandas 基本操作

```
>>> df4.groupby(['A','B'], as_index=False).mean()
                                    # 加 as_index=False 参数可防止分组名变为索引
   A   B     C     D
0  1  10  27.0  91.0
1  1  11  27.0  88.5
2  2  13  21.0  80.0
3  3  10  26.0  84.5
4  4  13  20.0  98.0
5  4  14  29.0  84.0
```

14.1 pandas 基本操作

（ 18 ）读写文件

```
>>> df.to_excel('d:\\test.xlsx', sheet_name='dfg')
                                          # 将数据保存为 Excel 文件
>>> df = pd.read_excel('d:\\test.xlsx', 'dfg',
                       index_col=None, na_values=['NA'])
>>> df.to_csv('d:\\test.csv')           # 将数据保存为 csv 文件
>>> df = pd.read_csv('d:\\test.csv')    # 读取 csv 文件中的数据
```

14.3 pandas 应用案例

· **例 14-1** 模拟转盘抽奖游戏，统计不同奖项的获奖概率。

```
import numpy as np
import pandas as pd

# 模拟转盘 100000 次
data = np.random.ranf(100000)
# 奖项等级划分
category = (0.0, 0.08, 0.3, 1.0)
labels = ('一等奖', '二等奖', '三等奖')
# 对模拟数据进行划分
```

14.3 pandas 应用案例

- **例 14-2** 假设有个 Excel 2007 文件“电影导演演员 .xlsx”，其中有三列分别为电影名称、导演和演员列表（同一个电影可能会有多个演员，每个演员姓名之间使用逗号分隔，如图 14-4 所示），要求统计每个演员的参演电影数量，并统计最受欢迎的前 3 个

	A	B	C
1	电影名称	导演	演员
2	电影1	导演1	演员1，演员2，演员3，演员4
3	电影2	导演2	演员3，演员2，演员4，演员5
4	电影3	导演3	演员1，演员5，演员3，演员6
5	电影4	导演1	演员1，演员4，演员3，演员7
6	电影5	导演2	演员1，演员2，演员3，演员8
7	电影6	导演3	演员5，演员7，演员3，演员9
8	电影7	导演4	演员1，演员4，演员6，演员7
9	电影8	导演1	演员1，演员4，演员3，演员8
10	电影9	导演2	演员5，演员4，演员3，演员9
11	电影10	导演3	演员1，演员4，演员5，演员10
12	电影11	导演1	演员1，演员4，演员3，演员11
13	电影12	导演2	演员7，演员4，演员9，演员12
14	电影13	导演3	演员1，演员7，演员3，演员13
15	电影14	导演4	演员10，演员4，演员9，演员14
16	电影15	导演5	演员1，演员8，演员11，演员15
17	电影16	导演6	演员14，演员4，演员13，演员16
18	电影17	导演7	演员3，演员4，演员9
19	电影18	导演8	演员3，演员4，演员10

14.3 pandas 应用案例

```
>>> import pandas as pd
>>> df = pd.read_excel('电影导演演员.xlsx')  #从 Excel 文件中读取数据
>>> df
   电影名称  导演          演员
0    电影1   导演1      演员1，演员2，演员3，演员4
1    电影2   导演2      演员3，演员2，演员4，演员5
2    电影3   导演3      演员1，演员5，演员3，演员6
3    电影4   导演1      演员1，演员4，演员3，演员7
4    电影5   导演2      演员1，演员2，演员3，演员8
5    电影6   导演3      演员5，演员7，演员3，演员9
6    电影7   导演4      演员1，演员4，演员6，演员7
7    电影8   导演1      演员1，演员4，演员3，演员8
8    电影9   导演2      演员5，演员4，演员3，演员9
9    电影10  导演3      演员1，演员4，演员5，演员10
```

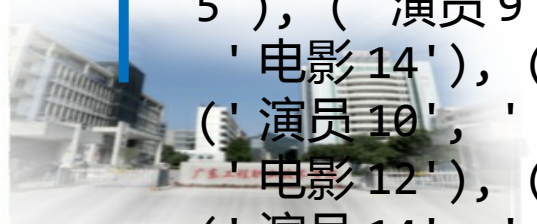
14.3 pandas 应用案例

```
>>> pairs = []
>>> for i in range(len(df)):                          # 遍历每一行数据
    actors = df.at[i, '演员'].split(',')               # 获取当前行的演员清单
    for actor in actors:                               # 遍历每个演员
        pair = (actor, df.at[i, '电影名称'])           # 每个演员参演的电影名称
        pairs.append(pair)
>>> pairs = sorted(pairs, key=lambda item:int(item[0][2:]))
                                                       # 按演员编号进行排序
```

14.3 pandas 应用案例

```
>>> pairs

[('演员1', '电影1'), ('演员1', '电影3'), ('演员1', '电影4'), ('演员1', '电影
5'), ('演员1', '电影7'), ('演员1', '电影8'), ('演员1', '电影10'), ('演员1',
 '电影11'), ('演员1', '电影13'), ('演员1', '电影15'), ('演员2', '电影1'), ('演
员2', '电影2'), ('演员2', '电影5'), ('演员3', '电影1'), ('演员3', '电影2'),
('演员3', '电影3'), ('演员3', '电影4'), ('演员3', '电影5'), ('演员3', '电影6')
, ('演员3', '电影8'), ('演员3', '电影9'), ('演员3', '电影11'), ('演员3', '电影
13'), ('演员3', '电影17'), ('演员3', '电影18'), ('演员4', '电影1'), ('演员4',
 '电影2'), ('演员4', '电影4'), ('演员4', '电影7'), ('演员4', '电影8'), ('演员
4', '电影9'), ('演员4', '电影10'), ('演员4', '电影11'), ('演员4', '电影12'),
('演员4', '电影14'), ('演员4', '电影16'), ('演员4', '电影17'), ('演员4', '电影
18'), ('演员5', '电影2'), ('演员5', '电影3'), ('演员5', '电影6'), ('演员5',
 '电影9'), ('演员5', '电影10'), ('演员6', '电影3'), ('演员6', '电影7'), ('演员
7', '电影4'), ('演员7', '电影6'), ('演员7', '电影7'), ('演员7', '电影12'),
('演员7', '电影13'), ('演员8', '电影5'), ('演员8', '电影8'), ('演员8', '电影1
5'), ('演员9', '电影6'), ('演员9', '电影9'), ('演员9', '电影12'), ('演员9',
 '电影14'), ('演员9', '电影17'), ('演员10', '电影10'), ('演员10', '电影14'),
('演员10', '电影18'), ('演员11', '电影11'), ('演员11', '电影15'), ('演员12',
 '电影12'), ('演员13', '电影13'), ('演员13', '电影16'), ('演员14', '电影14'),
```



14.3 pandas 应用案例

```
>>> index = [item[0] for item in pairs] # 生成演员列表
>>> data = [item[1] for item in pairs]  # 生成电影名称列表
>>> df1 = pd.DataFrame({'演员':index, '电影名称':data}) # 生成演员参演电影表格
>>> result = df1.groupby('演员', as_index=False).count()
                                        # 分组，统计每个演员的参演电影数量
                                        # 加 as_index=False 参数可防止分组名变为索引
```

14.3 pandas 应用案例

```
>>> result
    演员  电影名称
0     演员 1     10
1    演员 10      3
2    演员 11      2
3    演员 12      1
4    演员 13      2
5    演员 14      2
6    演员 15      1
7    演员 16      1
8     演员 2      3
9     演员 3     12
```

14.3 pandas 应用案例

```
>>> result.columns = ['演员', '参演电影数量']  #修改列名
>>> result
    演员  参演电影数量
0     演员1        10
1    演员10         3
2    演员11         2
3    演员12         1
4    演员13         2
5    演员14         2
6    演员15         1
7    演员16         1
8     演员2         3
```

14.3 pandas 应用案例

```
>>> result.nlargest(3, '参演电影数量') #参演电影数量最多的3个演员
    演员  参演电影数量
10   演员4        13
9    演员3        12
0    演员1        10
```