第10章 数据分析利器：pandas

本章提要

1.pandas库简介

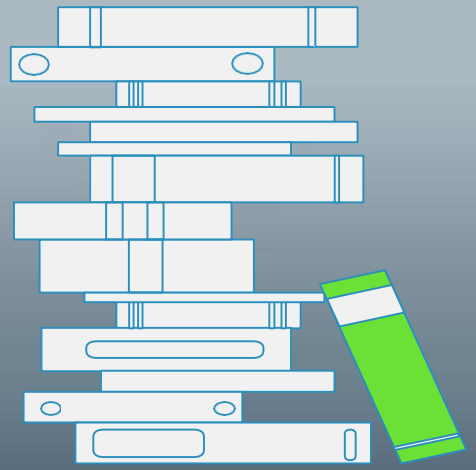
2.Series对象

3.DataFrame对象

4.DataFrame实例：
手机数据分析

10.1

pandas库简介



pandas库简介

- pandas是一个开源的Python第三方库，诞生于2008年，最初是作为一种金融数据分析工具被开发出来。
- pandas 名称的由来：

  Panel Data（多维面板数据） + Data Analysis（数据分析）
- 主要应用方向：

  pandas库是以numpy库为基础构建的，通常用来分析处理多维结构化（表格型）数据集，戒跟时间序列相关的数据集，广泛应用于各商业及科学计算领域。

pandas的优点

- pandas提供了快速高效的DataFrame对象，可用于集成索引的数据集操作；
- 提供了对各种格式数据的读写工具，例如：CSV文件、Excel、SQL数据库等；
- 智能的数据对齐和缺失数据处理方式，方便将凌乱数据处理成有序的形式；
- 可以很灵活的进行数据集的维度变换、切片、花式索引、大数据集的拆分、数据列的添加和删除；
- pandas提供了强大的分组引擎，可以对分组数据集进行拆分、应用、组合等操作，也可以方便的对数据集进行汇总、统计、合并、连接；
- 分层轴索引提供了在低维数据结构中处理高维数据的直观方式，也提供了时间序列操作功能；
- pandas经过了高度的性能优化，执行效率高；

导入库

Pandas使用前需要先安装，并在程序中导入该库，在做数据分析时也会经常用到numpy库的函数，以及matplotlib库的绘图功能，所以我们通常会同时导入这几个库：

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

两种数据对象

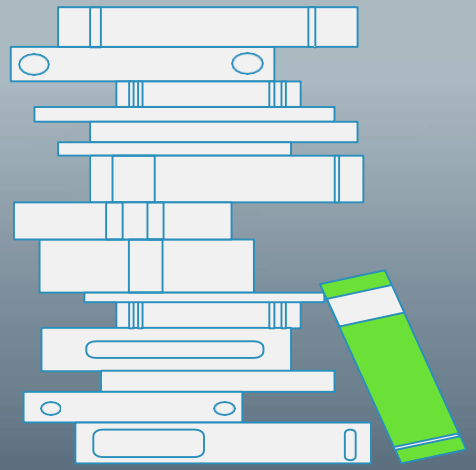
pandas有两个最主要的数据结构对象：

- Series对象
- DataFrame对象

pandas对于数据的绝大多数处理功能是在这两种对象上构建出来的，因此对于这两种对象的熟练掌握尤其重要，我们首先来学习第一个对象：Series。

10.2

Series 对象



Series对象

Series对象有着与一维数组相似的结构，由一组数据及其对应的标签（即索引）所组成，结构示意图如下：

Series对象	
索引	**数据**
index 1	value 1
index 2	value 2
…	…
…	…
index n	value n

Series的创建

创建Series对象时，可以使用index参数为对象指定字符串类型的索引。

```
>>> ls = ['a','b','c','d']
>>> s2 = pd.Series([10, 20, 30, 10], index=ls)
>>> s2
a   10
b   20
c   30
d   10
dtype: int64
```

```
>>> s1.values
array([2, 4, 6, 8], dtype=int64)

>>> s1.index
RangeIndex(start=0, stop=4, step=1)
```

Series的创建

以使用index参数为对象指定字符串类型的索引，以在对象上做常见的数组运算，例如标量乘法、数据过滤、应用数学函数等。

```
>>> ls = ['a','b','c','d']
>>> s2 = pd.Series([10, 20, 30, 10], index=ls)
>>> s2
a.   10
b.   20
c.   30
d.10  dtype:
int64
```

```
>>> s2[s2>10]
b.   20
c.   30
dtype: int64
```

```
>>> s2 ** 2
a    100
b    400
c    900
d    100
dtype: int64
```

```
>>> np.sqrt(s2)
a    3.162278
b    4.472136
c    5.477226
d    3.162278
dtype: float64
```

Series的数组运算

通过字典来创建Series对象，因为字典中的数据都是以键值对形式存储的，所以创建Series对象时就用字典中的主键（key）作为索引，字典中的值（value）作为数据部分

```
>>> dic = {'郭靖':20, '萧峰':19, '杨过':18, '令狐冲':13, '张无忌':20}
>>> s3 = pd.Series(dic)
>>> s3
郭靖     20
萧峰     19
杨过     18
令狐冲    13
张无忌    20
dtype: int64
```

Series的两种索引

Series对象中的字符串索引，通常称为标签索引，在对象内部，还存在着一个整数型索引。利用这两种索引都可以获取对象数据，但是整数型索引不包含右侧边界值。

```
>>> s3[1:4]
萧峰    19
杨过    18
令狐冲   13
dtype: int64
>>> s3['萧峰':'令狐冲']
萧峰    19
杨过    18
令狐冲   13
dtype: int64
```

```
>>> s3['令狐冲'] = 10
>>> s3['张三丰'] = 25
>>> s3
郭靖    20
萧峰    19
杨过    18
令狐冲   10
张无忌   20
张三丰   25
dtype: int64
```

Series的数据对齐

Series对象最重要的一个功能就是：算术运算中的数据对齐，就是多个Series对象中的数据依据索引进行匹配后再做算术运算。

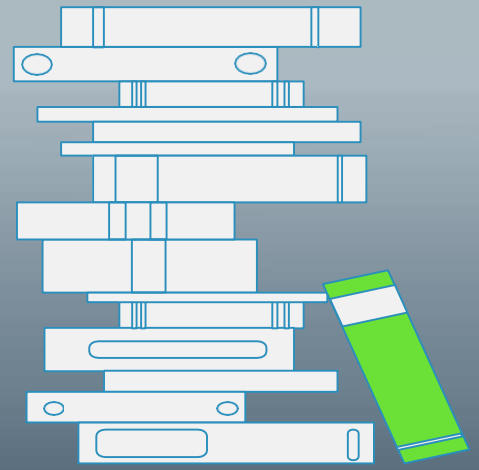
```
>>> dic = {'郭靖':20, '萧峰':19, '杨过':18,
    '令狐冲':13, '张无忌':20}
>>> s3 = pd.Series(dic)
>>> s3
郭靖     20
萧峰     19
杨过     18
令狐冲    13
张无忌    20
dtype: int64
```

```
>>> dic2 = {'郭靖':18, '萧峰':17, '杨过':18, '
    令狐冲':19, '韦小宝':5}
>>> s4 = pd.Series(dic2)
>>> s4
郭靖     18
萧峰     17
杨过     18
令狐冲    19
韦小宝    5
dtype: int64
```

```
>>> s3 + s4
令狐冲     32.0
张无忌     NaN
杨过    36.0
萧峰      36.0
郭靖      38.0
韦小宝     NaN
dtype: float64
```

10.3

DataFrame 对象



DataFrame对象

DataFrame是一个表格型的数据结构，由行和列所组成。

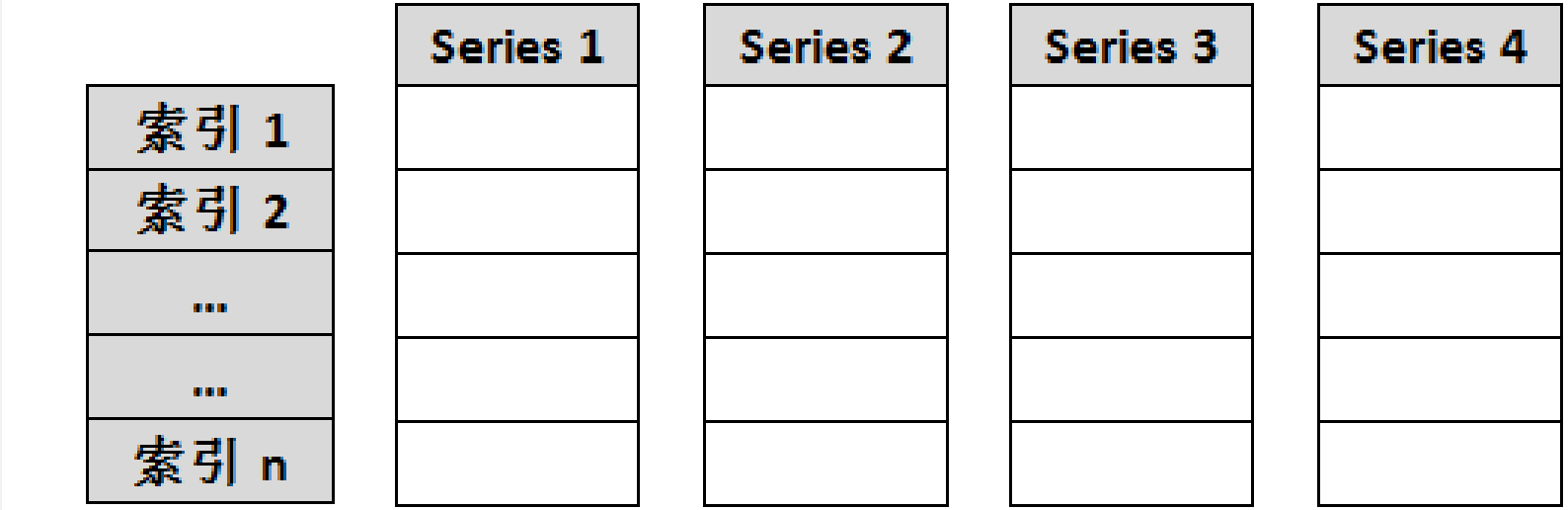
DataFrame的列是有序的，列与列之间的数据类型可以互不相同。

DataFrame的每一行存在一个行索引（index），每一列同样有一个 列索引（columns），如下图所示：

	列 1	列 2	列 3	列 4
索引 1				
索引 2				
...				
...				
索引 n				

DataFrame对象

实际上DataFrame对象中的每一列都是一个Series对象，所以也可以将DataFrame对象看作“共享行索引的多个Series的集合”。



创建DataFrame对象

创建DataFrame对象的方式很多，最常用的是通过一个字典直接转换，字典中每个主键的对应数据是等长的列表或numpy数组。
未定义行索引时自动创建行索引。

```
# 2017年中国大陆城市GDP数据，人口单位：（万人），GDP单位：（亿元）
dic = { '城市':  ['北京', '上海', '广州', '深圳', '重庆'],
        '人口':  [2171, 2418, 1090, 1404, 3372],
        'GDP': [28000, 30133, 21500, 22286, 19530]   }

df = pd.DataFrame(dic, columns=['城市', 'GDP', '人口'])
```

	城市	GDP	人口
0	北京	28000	2171
1	上海	30133	2418
2	广州	21500	1090
3	深圳	22286	1404
4	重庆	19530	3372

DataFrame的常见属性

DataFrame对象可以通过一些属性查看它的基本信息。

属性名称	功能
df.shape	DataFrame的形状
df.index	DataFrame的行索引
df.columns	DataFrame的列索引
df.values	以numpy.ndarray对象类型返回DataFrame所有数据
df.info()	DataFrame的摘要信息

DataFrame自定义索引

如果希望自定义索引，那么在创建对象时，就需要增加索引参数：

```
# 2017年中国大陆城市GDP数据
dic = {'城市': ['北京', '上海', '广州', '深圳', '重庆'],
       '人口': [2171, 2418, 1090, 1404, 3372],
       'GDP': [28000, 30133, 21500, 22286, 19530] }

df = pd.DataFrame(dic,
                  index = [2, 1, 4, 3, 5],
                  columns=['城市', 'GDP', '人口'])
print(df)
```

	城市	GDP	人口
2	北京	28000	2171
1	上海	30133	2418
4	广州	21500	1090
3	深圳	22286	1404
5	重庆	19530	3372

DataFrame自定义索引

pandas允许用户使用 set_index() 方法将某列设置为新索引，

也可以使用 reindex() 方法改变数据行的顺序，生成一个匹配新索引的新对象

```
# 以城市列作为新索引创建一个新对象
df = df.set_index(['城市'])

# 根据给定索引改变数据行顺序，创建一个新对象
df = df.reindex(['上海', '北京', '深圳', '广州', '重庆'])
print(df)
```

城市	GDP	人口
上海	30133	2418
北京	28000	2171
深圳	22286	1404
广州	21500	1090
重庆	19530	3372

DataFrame数据选择

如果想对DataFrame对象进行切片，获取部分数据，根据想获取的区域不同，有几种常用的数据选择方式：选择行、选择列、选择数据区域、选择单个数据、条件筛选等。

城市	GDP	人口
上海	30133	2418
北京	28000	2171
深圳	22286	1404
广州	21500	1090
重庆	19530	3372

```
# 选择行数据
print(df[0:1])                  # 获取第一行
print(df[1:3])                  # 获取第1、2行：北京、深圳
print(df['北京':'广州'])         # 获取第1-3行：北京、深圳、广州

print(df.head())                # 默认获取前5行数据
print(df.head(3))               # 获取前3行数据
print(df.tail(1))               # 获取最后1行数据

# 选择列
print(df['GDP'])                # 获取GDP列
```

DataFrame的区域选择

选择区域数据时可能用到的方式有：loc、iloc、at、iat等，我们先讲一下它们的大致区别。一种是基于行列索引标签迪行选择，如：loc、at；另一种是基于行列位置关系迪行切片，如：iloc、iat。

使用格式	功能
`loc[i]`	选取行索引为i的行
`loc[i1:i2]`	选取行索引从i1至i2的行，包含i2行
`loc[i1:i2, c1:c2]`	选取行索引从i1至i2, 列索引c1至c2的矩形区域
`at[i, c]`	选取行索引为i和列索引c的单个数据
`iloc[r]`	选取位置为第r行的数据，r从0开始
`iloc[r1:r2]`	选取位置为第r1行至第r2行的数据，丌包括r2行
`iloc[r1:r2, c1:c2]`	选取位置为第r1行至第r2行，c1至c2列的矩形区域
`iat[r, c]`	选取位置为第r行、第c列的单个数据，r和c从0开始

DataFrame的区域选择

对于右图示例对象，下列代码演示了如何使用loc、iloc、at、iat。

```
# 基于行列索引标签选择
>>> df.loc['北京']                                  # 选取北京行
>>> df.loc['北京':'广州']                           # 选取北京、深圳、广州三行
>>> df.loc['北京':'广州', 'GDP':'人口'] # 选取指定的三行两列

# 基于数据所在的行列位置进行选择，结果同上
>>> df.iloc[1]
>>> df.iloc[1:4]
>>> df.iloc[1:4, 0:]

# 基于标签选择深圳的人口：1404
>>> df.at['深圳', '人口']
# 基于位置选择深圳的人口：1404
>>> df.iat[2, 1]
```

城市	GDP	人口
上海	30133	2418
北京	28000	2171
深圳	22286	1404
广州	21500	1090
重庆	19530	3372

数据文件的导入和导出

DataFrame对象可以很方便的从各种常见格式文件中导入数据，也可以方便的将数据导出，生成各种文件。

```
# 导入数据，假定已存在两个数据文件：data1.csv、data2.xlsx
# header参数表示第一行作为标题行
df = pd.read_csv('data1.csv', header=1)
df = pd.read_excel('data2.xlsx')

# DataFrame对象数据导出为文件
df.to_excel('result1.xlsx', sheet_name='sheet1')
df.to_csv('result2.csv')
df.to_html('result3.html')
```

数据筛选

DataFrame中的数据，经常需要根据条件进行筛选，这时候可以对指定列直接设定条件。

```
# 筛选出人口大于2000万的城市
df_filter = df[df['人口']>2000]
print(df_filter)
```

城市	GDP	人口
上海	30133	2418
北京	28000	2171
重庆	19530	3372

```
# 筛选出GDP超过20000万且人口大于2000万的城市
df_loc = df.loc[(df['GDP']>20000) & (df['人口']>2000)]
print(df_loc)
```

城市	GDP	人口
上海	30133	2418
北京	28000	2171

数据排序

DataFrame中的数据，经常需要根据条件进行筛选，这时候可以对指定列直接设定条件。

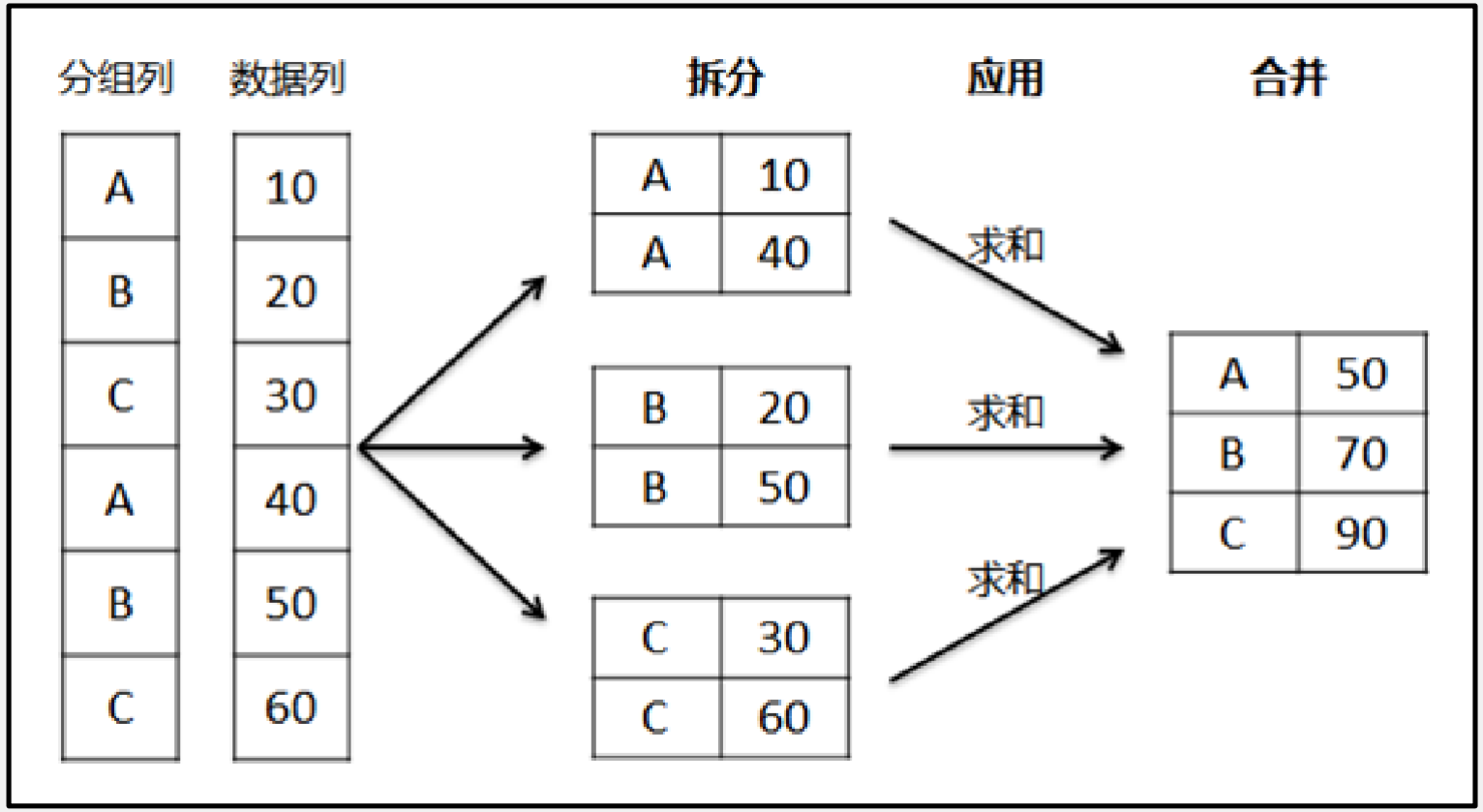
```
# 筛选出GDP超过20000万且人口大于2000万的城市
df = df.sort_values(by=['GDP'], ascending=False)

# 新增一列：排序，取值 1-5
df['排名'] = list(range(1,6))
print(df)
```

城市	GDP	人口	排名
上海	30133	2418	1
北京	28000	2171	2
深圳	22286	1404	3
广州	21500	1090	4
重庆	19530	3372	5

数据分组

在数据处理时经常会涉及到数据的分组操作，即根据某一列或某几列的取值，将大数据集拆分成几个小数据集，然后在这几个小数据集上应用统计函数进行相关统计，这就是分组操作。



数据分组

我们使用2017年三个经济大省前两名城市的GDP数据来做一次分组操作，统计其平均值等。

```
dic = {
    '省份': ['广东', '广东', '江苏', '浙江', '江苏', '浙江'],
    '城市': ['深圳', '广州', '苏州', '杭州', '南京', '宁波'],
    'GDP': [22286, 21500, 17319, 12556, 11715, 9846],
    '人口': [1090, 1404, 1065, 919, 827, 788],
}
df = pd.DataFrame(dic)
print(df)

# 创建一个分组对象
group = df.groupby('省份')
```

	省份	城市	GDP	人口
0	广东	深圳	22286	1090
1	广东	广州	21500	1404
2	江苏	苏州	17319	1065
3	浙江	杭州	12556	919
4	江苏	南京	11715	827
5	浙江	宁波	9846	788

数据分组

变量group是一个分组对象，它实际上并没有进行任何计算，只是包含关于“省份”分组的中间数据。

```
# 对各省GDP、人口数据求平均值
avg = group.mean()
print(avg)
```

省份	GDP	人口
广东	21893.0	1247.0
江苏	14517.0	946.0
浙江	11201.0	853.5

```
# 对各省GDP、人口数据求和
total = group.sum()
print(total)
```

省份	GDP	人口
广东	43786	2494
江苏	29034	1892
浙江	22402	1707

```
#求各省GDP值最高的城市数据
max = group.max()
print(max)
```

省份	城市	GDP	人口
广东	深圳	22286	1404
江苏	苏州	17319	1065
浙江	杭州	12556	919

```
#计算各省人均GDP值，精确到2位小数
avgPro = (total['GDP'] / total['人口']).round(2)
print(avgPro)
```

```
省份
广东    17.56
江苏    15.35
浙江    13.12
dtype: float64
```

DataFrame统计函数

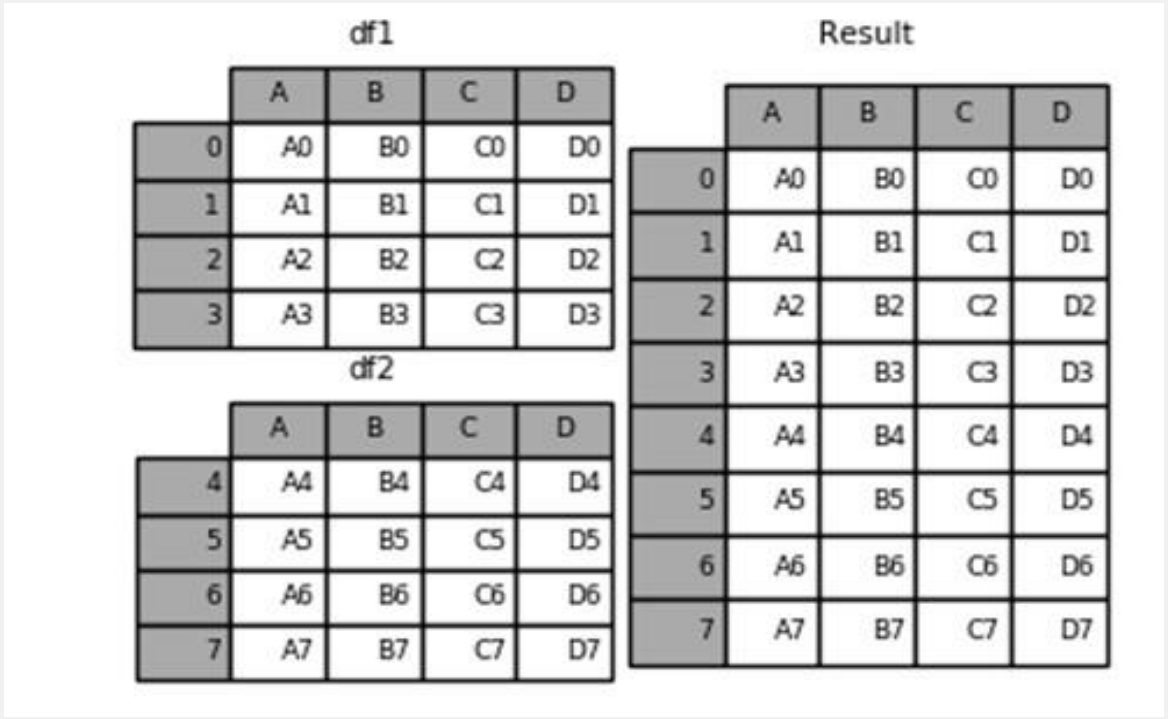
上面例子中，我们已经用到了DataFrame的统计函数，例如sum、mean等，在此列出另外一些常用的数据统计函数。

函数名	函数功能
df.describe()	查看数据列的汇总统计信息
df.mean()	返回所有列平均值
df.count()	返回每一列中非空值的个数
df.max()	返回每一列的最大值
df.min()	返回每一列的最小值
df.median()	返回每一列的中位数
df.std()	返回每一列的标准差

DataFrame数据合并

有时需要把几个DataFrame对象合并为一个DataFrame的情况，我们会用到 以下几种方法：append、merge等。

```
# 合并两个对象
result = df1.append(df2)
# 合并三个对象
result = df1.append([df2, df3])
```



df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

df3

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

Result

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

DataFrame绘图

DataFrame对象绘图时调用的还是matplotlib库，所以绘制图表的方式与之前讲解matplotlib绘图方式一样。

```
#用2017年几个城市的GDP和人口数据为例绘制一个柱形图
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
dic = { '省份': ['广东', '广东', '江苏', '浙江', '江苏', '浙江'],
        '城市': ['深圳', '广州', '苏州', '杭州', '南京', '宁波'],
    'GDP(亿元)': [22286, 21500, 17319, 12556, 11715, 9846],
    '人口(万)': [1090, 1404, 1065, 919, 827, 788] }

df = pd.DataFrame(dic)
df = df.set_index('城市') # 重新设定城市名称为行索引

# 为了在绘制图表中支持中文显示，设置字体为黑体
matplotlib.rcParams['font.family']='SimHei'
# 绘制GDP及人口柱形图
df.plot(kind='bar', title='2017年城市GDP及人口数据')
plt.show()
```