

第8章 pandas库

第8章 pandas库

- 8.1 Series对象
- 8.2 Series对象的基本运算
- 8.3 DataFrame对象
- 8.4 DataFrame对象的基本运算
- 8.5 pandas数据可视化
- 8.6 pandas读写数据

第8章 pandas库

Pandas是**Python**的一个非常强大的数据分析库，提供了高性能易用的数据类型，以及大量能使我们快速便捷地处理数据的函数和方法。

Pandas的名称来自于面板数据（**panel data**）和**python**数据分析（**data analysis**）。**panel data**是经济学中关于多维数据集的一个术语。

Pandas的核心数据结构有两种，即一维数组的**Series**对象和二维表格型的**DataFrame**对象，数据分析相关的所有事务都是围绕这两种对象进行的。

第8章 pandas库

- ✓ 8.1 Series对象
- 8.2 Series对象的基本运算
- 8.3 DataFrame对象
- 8.4 DataFrame对象的基本运算
- 8.5 pandas数据可视化
- 8.6 pandas读写数据

8.1 Series对象

□ 8.1.1 Series对象创建

Series对象是一维数组结构，与NumPy中的一维数组ndarray类似，二者与Python基本的数据结构list也很相近，其区别是list中的元素可以是不同的数据类型，而一维数组ndarray和Series中则只允许存储同一数据类型的数据。

Series对象的内部结构如右表所示，由两个相互关联的数组组成，一个是数据（也称元素，本章将这两个概念等同）数组values，用来存放数据，数组values中的每个数据都有一个与之关联的索引（标签），这些索引存储在另外一个叫作index的索引数组中。

Series	
index	values
0	‘a’
1	‘b’
2	‘c’
3	‘d’

8.1 Series对象

□ 8.1.1 Series对象创建

创建一个Series对象的最基语法格式如下：

```
pandas.Series(data=None,index=None,dtype=None, name=None)
```

返回值：返回一个Series对象

data：可以是一个Python列表，**index**与列表元素个数一致；也可以是字典，将键值对中的“值”作为Series对象的数据，将“键”作为索引；也可是一个标量值，这种情况下必须设置索引，标量值会重复来匹配索引的长度。

index：为Series对象的每个数据指定索引。

dtype：为Series对象的数据指定数据类型。

name：为Series对象起个名字。

8.1 Series对象

□ 8.1.1 Series对象创建

(1) 用一维ndarray数组创建Series对象

```
>>> import numpy as np
```

```
>>> import pandas as pd
```

```
>>> s=pd.Series(data=np.arange(0,5,2))
```

```
>>> s
```

```
0    0
```

```
1    2
```

```
2    4
```

(2) 用标量值创建Series对象

```
>>> import pandas as pd
```

```
>>> s2 = pd.Series(25,index = ['a',  
'b','c'])
```

```
>>> s2
```

```
a    25
```

```
b    25
```

8.1 Series对象

□ 8.1.1 Series对象创建

(3) 用字典创建Series对象

键值对中的“键”是用来作为Series对象的索引，键值对中的“值”作为Series对象的数据。

```
>>> dict1={'Alice':'2341',  
'Beth':'9102','Cecil':'3258'}  
>>> sd = pd.Series(dict1)  
>>> sd  
  
Alice    2341  
Beth     9102  
Cecil    3258
```

(4) 用列表创建Series对象

```
>>> import pandas as pd  
>>> s3=pd.Series([1,2,3],index =  
['Java','C','Python'])  
>>> s3  
Java      1  
C          2  
Python    3  
dtype: int64
```


8.1 Series对象

□ 8.1.2 Series对象的属性

- (1) `shape`属性获取Series对象的形状。
- (2) `dtype`属性获取Series对象的数据数组中的数据的数据类型。
- (3) `values`属性获取Series对象的数据数组
- (4) `index`属性获取Series对象的数据数组的索引
- (5) Series对象本身及索引的`name`属性

```
>>> s.name='data' #为Series对象s命名'data'
```

```
>>> s.name
```

```
'data'
```

```
>>> s.index.name='idx'
```

```
>>> s.index.name
```

```
'idx'
```

8.1 Series对象

□ 8.1.3 Series对象的数据的查看和修改

(1) 通过索引和切片查看Series对象的数据

可以使用数据索引以“Series对象[id]”方式访问Series对象的数据数组中索引为id的数据。

```
>>> s=pd.Series(data=[1,2,3],index = ['Java','C','Python'])
```

```
>>> s['C']
```

```
2
```

可通过默认索引来读取。

```
>>> s[1]
```

```
2
```

通过截取（切片）的方式读取多个元素。

```
>>> s[0:2]
```

```
Java    1
```

```
C       2
```

8.1 Series对象

□ 8.1.3 Series对象的数据的查看和修改

使用多个数据对应的索引来一次读取多个元素，注意索引要放在一个列表中。

```
>>> s[['Python','C','Java']]
```

```
Python    3
```

```
C         2
```

```
Java      1
```

```
dtype: int64
```

根据筛选条件读取数据。

```
>>> s[s > 1]      #获取数据值>1的元素
```

```
C         2
```

```
Python    3
```

第8章 pandas库

- ☐ 8.1 Series对象
- ☒ 8.2 Series对象的基本运算
- ☐ 8.3 DataFrame对象
- ☐ 8.4 DataFrame对象的基本运算
- ☐ 8.5 pandas数据可视化
- ☐ 8.6 pandas读写数据

8.2 Series对象的基本运算

□ 8.2.1 算术运算与函数运算

(1) 算术运算

适用于NumPy数组的运算符（+、-、*、/）或其它数学函数，也适用于Series对象。可以将Series对象的数据数组与标量进行+、-、*、/等算术运算。

(2) 函数运算

```
>>> s = pd.Series([2,4,6],index = ["a","b","c"])
```

```
>>> np.sqrt(s)          #计算各数据的平方根
```

```
a    1.414214
```

```
b    2.000000
```

```
c    2.449490
```

```
>>> np.square(s)       #计算各数据的平方
```

```
a     4
```

```
b    16
```

```
c    36
```

8.2 Series对象的基本运算

□ 8.2.2 Series对象之间的运算

Series对象之间也可进行+、-、*、/等运算，不同Series对象运算的时候，能够通过识别索引进行匹配计算，即只有索引相同的元素才会进行相应的运算操作。

```
>>> s5=pd.Series([10,20],index=['c','d'])
>>> s6=pd.Series([2,4,6,8],index=['a','b','c','d'])
>>> s5+s6          #相同索引值的元素相加
a    NaN
b    NaN
c    16.0
d    28.0
```

第8章 pandas库

- ☐ 8.1 Series对象
- ☐ 8.2 Series对象的基本运算
- ☒ 8.3 DataFrame对象
- ☐ 8.4 DataFrame对象的基本运算
- ☐ 8.5 pandas数据可视化
- ☐ 8.6 pandas读写数据

8.3 DataFrame对象

□ 8.3.1 DataFrame对象创建

DataFrame是一个表格型的数据结构，既有行索引（保存在index）又有列索引（保存在columns），是Series对象从一维到多维的扩展。**DataFrame**对象每列相同位置处的元素共用一个行索引，每行相同位置处的元素共用一个列索引。**DataFrame**对象各列的数据类型可以不相同。

DataFrame对象的内部组成如图所示

The diagram illustrates the internal structure of a DataFrame as a table. A bracket on the left labeled 'index' spans the first column. A bracket on top labeled 'columns' spans the subsequent columns. The table contains five rows and three columns. The first column contains integer indices from 0 to 3. The second column contains course names as strings. The third column contains scores as integers.

	course	scores
0	'C'	80
1	'Java'	96
2	'Python'	90
3	'Hadoop'	88

8.3 DataFrame对象

□ 8.3.1 DataFrame对象创建

创建DataFrame对象最常用的方法是使用pandas的DataFrame()构造函数，其语法格式如下：

DataFrame(data=None, index=None, columns=None, dtype=None)

返回值：DataFrame对象

参数说明：

data：创建DataFrame对象的数据，其类型可以是字典、嵌套列表、元组列表、numpy的ndarray对象、其它DataFrame对象。

index：行索引，创建DataFrame对象的数据时，如果没有提供索引，默认赋值为arange(n)。

columns：列索引，没有提供索引时，默认赋值为arange(n)。

dtype：用来指定元素的数据类型，如果为空，自动推断类型。

8.3 DataFrame对象

□ 8.3.1 DataFrame对象创建

(1) 可将一个字典对象传递给DataFrame()函数来生成一个DataFrame对象，字典的键作为DataFrame对象的列索引，字典的值作为列索引对应的列值，pandas也会自动为其添加一列从0开始的数值作为行索引。

```
>>> data={'course':['C','Java','Python','Hadoop'],  
'scores':[82,96,92,88], 'grade':['B','A','A','B']}  
>>> df=pd.DataFrame(data)  
>>> df
```

	course	grade	scores
0	C	B	82
1	Java	A	96
2	Python	A	92
3	Hadoop	B	88

8.3 DataFrame对象

□ 8.3.1 DataFrame对象创建

(2) 可以只选择字典对象的一部分数据来创建DataFrame对象，只需在DataFrame构造函数中，用columns选项指定需要的列即可，新建的DataFrame对象各列顺序与指定的列顺序一致。

```
>>> df1=pd.DataFrame (data,columns=['course','grade'])
```

```
>>> df1
```

	course	grade
0	C	B
1	Java	A
2	Python	A
3	Hadoop	B

8.3 DataFrame对象

□ 8.3.1 DataFrame对象创建

(3) 创建DataFrame对象时，如果没有用index数组明确指定行索引，pandas也会自动为其添加一列从0开始的数值作为行索引。如果想用自己定义的行索引，则要把定义的索引放到一个数组中，赋值给index选项。

```
>>> df2=pd.DataFrame (data, index=['一','二','三','四'])
```

```
>>> df2
```

	course	grade	scores
一	C	B	82
二	Java	A	96
三	Python	A	92
四	Hadoop	B	88

8.3 DataFrame对象

□ 8.3.1 DataFrame对象创建

(4) 创建DataFrame对象时，可以同时指定行索引和列索引，这时候就需要传递三个参数给DataFrame()构造函数，三个参数的顺序是：数据、index选项和columns选项。将存放行索引的数组赋给index选项，将存放列索引的数组赋给columns选项。

```
>>> df3=pd.DataFrame ([[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]],  
index=['一','二','三','四'],columns=['A','B','C','D'])
```

```
>>> df3
```

	A	B	C	D
一	1	2	3	4
二	5	6	7	8
三	9	10	11	12
四	13	14	15	16

8.3 DataFrame对象

□ 8.3.1 DataFrame对象创建

(5) 以字典的字典或Series的字典的结构创建DataFrame对象，pandas会将外边的键解释成列名称，将里面的键解释成行索引。

```
data={  
    "name":{"one":"Jack",'two':"Mary",'three':"John",'four':"Alice"},  
    "age":{"one":10,'two':20,          'three':30,'four':40},          "weight":  
    {'one':30,'two':40,'three':50,'four':65}}
```

```
>>> df=pd.DataFrame(data)
```

```
>>> df
```

	name	age	weight
one	Jack	10	30
two	Mary	20	40
three	John	30	50
four	Alice	40	65

8.3 DataFrame对象

□ 8.3.1 DataFrame对象创建

(6) 用键值为列表的字典构建DataFrame，其中每个列表（list）代表的是一个列，字典的名字则是列索引。这里要注意的是每个列表中的元素数量应该相同，否则会报错。

```
>>> data1={ "name":["Jack","Mary","John"],  
            "age":[10,20,30],  
            "weight":[30,40,50]}
```

```
>>> df=pd.DataFrame(data1)
```

```
>>> df
```

	age	name	weight
0	10	Jack	30
1	20	Mary	40
2	30	John	50

8.3 DataFrame对象

□ 8.3.1 DataFrame对象创建

(7) 以字典的列表构建DataFrame，其中每个字典代表的是每条记录（DataFrame中的一行），字典中各个键的键值对应的是这条记录的相关属性。

```
>>> d = [{'one' : 1,'two':1},{'one' : 2,'two' : 2},{'one' : 3,'two' : 3},{'two' : 4}]
```

```
>>> df = pd.DataFrame(d,index=['a','b','c','d'],columns=['one','two'])
```

```
>>> df
```

	one	two
a	1.0	1
b	2.0	2
c	3.0	3
d	NaN	4

8.3 DataFrame对象

□ 8.3.2 DataFrame对象的属性

属性名	功能描述
T	行列转置
columns	查看列索引名，可得到各列的名称
dtypes	查看各列的数据类型
index	查看行索引名
shape	查看DataFrame对象的形状
size	返回DataFrame对象包含的元素个数，为行数、列数大小的乘积
values	获取存储在DataFrame对象中的数据，返回一个NumPy数组
ix	用ix属性和行索引可获取DataFrame对象指定行的内容
ix[[x,y,...], [x,y,...]]	对行重新索引，然后对列重新索引
index.name	行索引的名称
columns.name	列索引的名称
loc	通过行索引获取行数据
iloc	通过行号获取行数据（只接受整型索引）

8.3 DataFrame对象

□ 8.3.3 查看和修改DataFrame对象的元素

(1) 查看DataFrame对象中的元素
要想获取存储在DataFrame对象中的一个元素，需要依次指定元素所在的列名称、行名称（索引）。

```
>>> df
```

	age	name	weight
0	10	Jack	30
1	20	Mary	40
2	30	John	55
3	40	Alice	65

```
>>> df['age'][1] #先列后行
20
```

可以通过指定条件筛选
DataFrame对象的元素。

```
>>> df[df.weight>35]
```

	age	name	weight
1	20	Mary	40
2	30	John	55
3	40	Alice	65

```
>>> df[df.age>35]
```

	age	name	weight
3	40.0	Alice	65.0

8.3 DataFrame对象

□ 8.3.3 查看和修改DataFrame对象的元素

(2) 修改DataFrame对象中的元素
可以用DataFrame对象的name属性为DataFrame对象的列索引columns和行索引index指定别的名称，以便于识别。

```
>>> df.index.name='id'
```

```
>>> df.columns.name='item'
```

```
>>> df
```

item	age	name	weight
id			
0	10	Jack	30
1	20	Mary	40
2	30	John	55
3	40	Alice	65

可以为DataFrame对象添加新的列，指定新列的名称，以及为新列赋值。

#添加新列，并都赋值为10

```
>>> df['new']=10 >>> df
```

item	age	name	weight	new
------	-----	------	--------	-----

id

0	10	Jack	30	10
1	20	Mary	40	10
2	30	John	55	10
3	40	Alice	65	10

8.3 DataFrame对象

□ 8.3.4 判断元素是否属于DataFrame对象

可通过DataFrame对象的方法
`isin()`判断一组元素是否属于
DataFrame对象。

```
>>> df
```

	age	name	weight
four	40	Alice	65
one	10	Jack	30
three	30	John	50
two	20	Mary	40

```
>>> df.isin(['Jack', 30])
```

	age	name	weight
four	False	False	False
one	False	True	True
three	True	False	False
two	False	False	False

第8章 pandas库

- ☐ 8.1 Series对象
- ☐ 8.2 Series对象的基本运算
- ☐ 8.3 DataFrame对象
- ☒ 8.4 DataFrame对象的基本运算
- ☐ 8.5 pandas数据可视化
- ☐ 8.6 pandas读写数据

8.4 DataFrame对象的基本运算

□ 8.4.1 数据筛选

DataFrame对象的常用数据筛选方法如表所示

DataFrame对象的数据筛选方法	描述
<code>df.head(N)</code>	返回前N行
<code>df.tail(M)</code>	返回后M行
<code>df[m:n]</code>	切片，选取m~n-1行
<code>df[df['列名'] > value]</code>	选取满足条件的行
<code>df.query('列名 > value')</code>	选取满足条件的行
<code>df.query('列名 == [v1,v2,...]')</code>	选取列名列表的值等于v1,v2,...的行
<code>df.loc[:, 'colname']</code>	选取colname列的所有行
<code>df.loc[row, col]</code>	选取某一元素
<code>df['col']</code>	获取col列，返回Series

8.4 DataFrame对象的基本运算

□ 8.4.1 数据筛选

```
>>> df
```

status	year	index
good	2012	1
good	2013	2
well	2014	3
well	2015	4
wonderful	2016	5

```
>>> df.loc[1:4, 'year'] #获取'year' 列的1~3行  
2013  
2014  
2015
```

8.4 DataFrame对象的基本运算

□ 8.4.2 数据预处理

DataFrame对象的数据预处理方法	描述
<code>df.duplicated(subset=None, keep='first')</code>	针对某些列，返回用布尔序列表示的重复行
<code>df.drop_duplicates(subset=None, keep='first', inplace=False)</code>	<code>df.drop_duplicates()</code> 用于删除df中重复行，并返回删除重复后的结果
<code>df.fillna(value=None, method=None, axis=None, inplace=False, limit=None)</code>	使用指定的方法填充NA/NaN缺失值
<code>df.drop(labels=None, axis=0, index=None, columns=None, inplace=False)</code>	删除指定轴上的行或列，它不改变原有的DataFrame对象中的数据，而是返回另一个DataFrame对象来存放删除后的数据

8.4 DataFrame对象的基本运算

□ 8.4.2 数据预处理

DataFrame对象的数据预处理方法	描述
<code>df.dropna(axis=0,how='any',thresh=None,subset=None,inplace=False)</code>	删除指定轴上的缺失值
<code>del df['col']</code>	直接在df对象上删除col列
<code>df.columns = col_lst</code>	重新命名列名，col_lst为自定义列名列表
<code>df.rename(index={'row1':'A'}, columns= {'col1':'A1'})</code>	重命名行索引名和列索引名。
<code>df.reindex(index=None, columns=None, fill_value='NaN')</code>	改变索引，返回一个重新索引的新对象，index用作新行索引，columns用作新列索引，将缺失值填充为fill_value

8.4 DataFrame对象的基本运算

□ 8.4.2 数据预处理

DataFrame对象的数据预处理方法	描述
<code>df.replace(to_replace=None, value=None, inplace=False, limit=None, regex=False, method='pad')</code>	用来把to_replace所列出的且在df对象中出现的元素值替换为value所表示的值
<code>df.merge(right, how='inner', on=None, left_on=None, right_on=None)</code>	通过行索引或列索引进行两个DataFrame对象的连接
<code>pandas.concat(objs, axis=0, join='outer', join_axes=None, ignore_index=False, keys=None)</code>	以指定的轴将多个对象堆叠到一起, concat()不会去重对象中重复的记录
<code>df.stack(level=-1, dropna=True)</code>	将df的列旋转成行
<code>df.unstack(level=-1, fill_value=None)</code>	将df的行旋转为列

8.4 DataFrame对象的基本运算

□ 8.4.3 数据运算与排序

DataFrame对象的数据运算与排序方法如表所示，表中的df表示一个DataFrame对象。

数据运算与排序方法	描述
df.T	df的行列转置
df*N	df的所有元素乘以N
df1+df2	将df1和df2的行名和列名都相同的元素相加，其它位置的元素用NaN填充
df1.add(other, axis='columns', level=None, fill_value=None)	将df1中的元素与other中的元素相加，other的类型可以是scalar（标量）、sequence（序列）、Series、DataFrame等形式
df1.sub(other, axis='columns', level=None, fill_value=None)	将df1中的元素与other中的元素相减
df1.div(other, axis='columns', level=None, fill_value=None)	将df1中的元素与other中的元素相除

8.4 DataFrame对象的基本运算

□ 8.4.3 数据运算与排序

DataFrame对象的数据运算与排序方法如表所示，表中的df表示一个**DataFrame**对象。

数据运算与排序方法	描述
df1.(other, axis='columns', level=None, fill_value=None)	将df1中的元素与other中的元素相乘
df.apply(func,axis=0)	将func函数应用到df的行或列所构成的一维数组上
df.applymap(func)	将func函数应用到各个元素上
df.sort_index(aixs=0, ascending=True)	按行索引进行升序排序
df.sort_values(by, axis=0, ascending=True)	按指定的列或行进行值排序
df.rank(axis=0, method='average', ascending=True)	沿着行计算元素值的排名，对于相同的两个元素值，沿着行顺序排在前面的数据排名高，返回各个位置上元素值从小到大排序对应的序号

8.4 DataFrame对象的基本运算

□ 8.4.3 数据运算与排序

(1) `df.apply(func,axis=0)`

作用：将func函数应用到DataFrame对象df的行或列所构成的一维数组上。

func：应用到行或列上的函数。

axis：**axis=0**，对每一列应用func函数；**axis=1**，对每一行应用函数。

```
>>> df
```

	a	b	c	d
A	0	1	2	3
B	4	5	6	7
C	8	9	10	11
D	12	13	14	15

#计算数组元素的取值间隔

```
>>> def f(x):
```

```
    return x.max()-x.min()
```

```
#求每一列元素的取值间隔
```

```
>>> df.apply(f,axis=0)
```

```
a    12
b    12
c    12
d    12
```

8.4 DataFrame对象的基本运算

□ 8.4.3 数据运算与排序

(2) `df.sort_values(by, axis=0, ascending=True)`

作用：按指定的列或行进行值排序。

参数说明：

by: 指定某些行或列作为排序的依据。

axis: `axis=0`, 对行进行排序; `axis=1`, 对列进行排序。

```
>>> df
```

	col1	col2	col3
0	A	2	1
1	A	9	9
2	B	8	4
3	D	7	2
4	C	4	3

#按col1进行排序

```
>>> df.sort_values(by=['col1'])
```

	col1	col2	col3
0	A	2	1
1	A	9	9
2	B	8	4
4	C	4	3
3	D	7	2

8.4 DataFrame对象的基本运算

□ 8.4.4 数学统计

DataFrame对象的常用数学统计方法如表所示

数学统计方法	描述
<code>df.count(axis=0,level=None)</code>	统计每列或每行非NaN的元素个数。
<code>df.describe(percentiles=None, include=None, exclude=None)</code>	生成描述性统计，总结数据集分布的中心趋势，分散和形状，不包括NaN值。
<code>df.max(axis=0)</code>	<code>axis=0</code> 表示返回每列的最大值， <code>axis=1</code> 表示返回每行的最大值
<code>df.min(axis=0)</code>	<code>axis=0</code> 表示返回每列的最小值， <code>axis=1</code> 表示返回每行的最小值
<code>df.sum(axis=None,skipna=None, level=None)</code>	返回指定轴上元素值的和
<code>df.mean(axis=None,skipna=None, level=None)</code>	返回指定轴上元素值的平均值

8.4 DataFrame对象的基本运算

□ 8.4.4 数学统计

DataFrame对象的常用数学统计方法如表所示

数学统计方法	描述
<code>df.median(axis=None,skipna=None,level=None)</code>	返回指定轴上元素值的中位数
<code>df.var(axis=None,skipna=None,level=None)</code>	返回指定轴上元素值的均方差
<code>df.std(axis=None,skipna=None,level=None)</code>	返回指定轴上元素值的标准差
<code>df.cov()</code>	计算df的列列之间的协方差，不包括NA/空值
<code>df.corr(method='pearson')</code>	计算df的列列之间的相关性，不包括NA/空值
<code>df1.corrwith(df2)</code>	计算df1与df2的行或列之间的相关性
<code>df.cumsum(axis=0,skipna=True)</code>	对df求累加和，计算结果是与df形状相同的DataFrame对象
<code>df.cumprod(axis=None, skipna=True)</code>	返回df指定轴的元素累积积

8.4 DataFrame对象的基本运算

□ 8.4.4 数学统计

(4) `df.max(axis=0)`

作用：`axis=0`表示返回df列的最大值，`axis=1`表示返回df行的最大值。

```
>>> df1=pd.DataFrame(np.arange(20).reshape(5,4),index=list('abcde'),  
columns=['one','two','three','four'])
```

```
>>> df1
```

	one	two	three	four
a	0	1	2	3
b	4	5	6	7
c	8	9	10	11
d	12	13	14	15
e	16	17	18	19

```
>>> df1.max(axis=1) #返回每行的最大值
```

a	3
b	7
c	11
d	15
e	19

8.4 DataFrame对象的基本运算

□ 8.4.4 数学统计

(8) `df.corr(method='pearson')`

作用：计算列与列之间的相关系数，返回相关系数矩阵。

参数说明：

method：指定求何种相关系数，可选值为{'pearson', 'kendall', 'spearman'}，'pearson'相关系数用来衡量两个数据集合是否在一条线上面，即针对线性数据的相关系数计算；'kendall'用于反映分类变量相关性的指标，即针对无序序列的相关系数；'spearman'表示非线性的、非正态分布的数据的相关系数。

线性相关系数不仅表示线性相关的方向，还表示线性相关的程度，取值[-1,1]。也就是说，相关系数为正值，说明一个变量变大另一个变量也变大；取负值说明一个变量变大另一个变量变小，取0说明两个变量没有相关关系。同时，相关系数的绝对值越接近1，线性关系越显著。

8.4 DataFrame对象的基本运算

□ 8.4.4 数学统计

(8) `df.corr(method='pearson')`

作用：计算列与列之间的相关系数，返回相关系数矩阵。

```
>>> import pandas as pd
```

```
>>> data = pd.DataFrame({'c':[5,4,3,2,1,0], 's':[0,1,2,3,4,5],  
'x':[0,1,2,4,7,10], 'y':[0,3,2,4,5,7]},index = ['p1','p2','p3','p4','p5','p6'])
```

```
>>> data
```

	c	s	x	y
p1	5	0	0	0
p2	4	1	1	3
p3	3	2	2	2
p4	2	3	4	4
p5	1	4	7	5
p6	0	5	10	7

```
>>> data.corr() #计算pearson相关系数
```

	c	s	x	y
c	1.000000	-1.000000	-0.972598	-0.946256
s	-1.000000	1.000000	0.972598	0.946256
x	-0.972598	0.972598	1.000000	0.941729
y	-0.946256	0.946256	0.941729	1.000000

8.4 DataFrame对象的基本运算

□ 8.4.5 数据分组与聚合

对数据集进行分组并对各分组应用函数是数据分析中的重要环节。在pandas中，分组运算主要通过groupby函数来完成，聚合操作主要通过agg函数来完成。

groupby对数据进行数据分组运算的过程分为三个阶段：分组、用函数处理分组和分组结果合并。

(1) 分组。按照键（key）或者分组变量将数据分组。分组键可以有多种形式，且类型不必相同：①**列表或数组**，其长度与待分组的轴一样。②**DataFrame对象的某个列名**。③**字典或Series**，给出待分组轴上的值与分组名之间的对应关系。④**函数**，用于处理轴索引或索引中的各个标签。

(2) 用函数处理。对于每个分组应用我们指定的函数，这些函数可以是Python自带函数，可以是自定义的函数。

(3) 合并分组处理结果。把每个分组的计算结果合并起来。

8.4 DataFrame对象的基本运算

□ 8.4.5 数据分组与聚合

`df.groupby(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=False)`

作用：通过指定列索引或行索引，对df的数据元素进行分组。

by：用于指定分组的依据，其数据形式可以是映射、函数、索引以及索引列表。

axis：默认axis=0按行分组，可指定axis=1对列分组。

level：int值，默认为None，如果axis是一个MultiIndex（分层索引），则按特定的级别分组。

sort：排序。boolean值，默认True。

group_keys：当调用apply时，添加group_keys来索引来识别片断。

squeeze：尽可能减少返回类型的维度，否则返回一致的类型。

8.4 DataFrame对象的基本运算

□ 8.4.5 数据分组与聚合

(1) 按列分组

```
>>> df
```

	data1	data2	key1	key2
0	5	2	a	one
1	5	4	a	two
2	3	5	b	one
3	4	3	b	two
4	1	2	a	one

```
>>> group = df.groupby('key1')    #按'key1'进行分组
```

```
>>> group.groups                  #通过groups属性来查看分组情况
```

```
{'a': Int64Index([0, 1, 4], dtype='int64'), 'b': Int64Index([2, 3],  
dtype='int64')}
```

8.4 DataFrame对象的基本运算

□ 8.4.5 数据分组与聚合

(2) 按分组统计

在`df.groupby()`所生成的分组上应用`size()`、`sum()`、`count()`、`mean()`等统计函数，能分别统计分组数量、不同列的分组和、不同列的分组数量、分组不同列的平均值。

`>>> group.size() #统计分组数量` `>>> group.sum()#求不同列的分组和`

key1		data1	data2
a	3		
b	2		
dtype: int64			

key1		
a	11	8
b	7	8

8.4 DataFrame对象的基本运算

□ 8.4.5 数据分组与聚合

2. 数据聚合

对于聚合，一般指的是从数组产生标量值的数据转换过程，常见的聚合运算都有相关的统计函数快速实现，当然也可以自定义聚合运算。聚合操作主要通过agg函数来完成，agg函数的语法格式如下：

DataFrame.agg(func, axis=0)

作用：通过func在指定的轴上进行聚合操作。

func：用来指定聚合操作的方式，其数据形式有函数、字符串、字典以及字符串或函数所构成的列表。

axis：axis=0表示在列上操作，axis=1表示在行上操作。

8.4 DataFrame对象的基本运算

□ 8.4.5 数据分组与聚合

2. 数据聚合

(1) 在DataFrame对象的行或列上执行聚合操作

```
>>> df3
```

	A	B	C
0	1.0	2.0	3.0
1	4.0	5.0	6.0
2	7.0	8.0	9.0
3	NaN	NaN	NaN

```
>>> df3.agg(['sum', 'min']) #在df的各列上执行'sum'和'min'聚合操作
```

	A	B	C
sum	12.0	15.0	18.0
min	1.0	2.0	3.0

8.4 DataFrame对象的基本运算

□ 8.4.5 数据分组与聚合

2. 数据聚合

(2) 在df.groupby()所生成的分组上应用agg()

对于分组的某一列或者多个列，应用agg(func)可以对分组后的数据应用func函数。例如：用group['data1'].agg('key1')对分组后的'data1'列求均值。也可以推广到同时作用于多个列和使用多个函数。

```
>>> df4
```

	data1	data2	key1	key2
0	1	7	a	one
1	6	2	b	two
2	8	3	c	three
3	7	3	d	one
4	7	4	a	two
5	2	4	b	three
6	8	5	c	one
7	7	4	d	two

```
>>> group5=df4.groupby('key1')
>>> group5.agg('mean')
      data1  data2
key1
a         4.0    5.5
b         4.0    3.0
c         8.0    4.0
d         7.0    3.5
```

8.4 DataFrame对象的基本运算

□ 8.4.5 数据分组与聚合

2. 数据聚合

(2) 在df.groupby()所生成的分组上应用agg()

```
>>> df4.groupby('key2').agg(['mean','sum']) #在每列上使用两个函数
```

	data1		data2	
	mean	sum	mean	sum
key2				
one	5.333333	16	5.000000	15
three	5.000000	10	3.500000	7
two	6.666667	20	3.333333	10

8.4 DataFrame对象的基本运算

□ 8.4.5 数据分组与聚合

2. 数据聚合

(3) 应用apply()函数执行聚合操作

```
>>> df4.groupby('key2').apply(sum)
```

	data1	data2	key1	key2
key2				
one	16	15	adc	oneoneone
three	10	7	cb	threethree
two	20	10	bad	twotwotwo

第8章 pandas库

- ☐ 8.1 Series对象
- ☐ 8.2 Series对象的基本运算
- ☐ 8.3 DataFrame对象
- ☐ 8.4 DataFrame对象的基本运算
- ☒ 8.5 pandas数据可视化
- ☐ 8.6 pandas读写数据

8.5 pandas数据可视化

DataFrame对象df通过调用它的plot()方法，可以快速地将df的数据绘制成各种类型的图，plot()方法的语法格式如下：

```
df.plot(x=None, y=None, kind='line', figsize=None, use_index=True,  
title=None, grid=None, legend=True, style=None, xticks=None,  
yticks=None, xlim=None, ylim=None, rot=None, fontsize=None,  
alpha)
```

参数说明：

x： 设置x轴标签或位置，默认情况下，plot会将行索引作为x轴标签。

y： 设置y轴标签或位置，默认情况下，plot会将列索引作为y轴标签。

8.5 pandas数据可视化

DataFrame对象df通过调用它的plot()方法，可以快速地将df的数据绘制成各种类型的图，plot()方法的语法格式如下：

```
df.plot(x=None, y=None, kind='line', figsize=None, use_index=True,  
title=None, grid=None, legend=True, style=None, xticks=None,  
yticks=None, xlim=None, ylim=None, rot=None, fontsize=None,  
alpha)
```

kind: 所要绘制的图类型，kind='line'，绘制折线图；kind='bar'，绘制条形图；kind='barh'，绘制横向条形图；kind='hist'，绘制直方图（柱状图）；kind='box'，绘制箱线图；kind='kde'，绘制Kernel的密度估计图，主要对柱状图添加Kernel概率密度线；kind='density'，绘制的图与kind='kde'的图相同；kind='area'，绘制区域图；kind='pie'，绘制饼图；kind='scatter'，绘制散点图。

8.5 pandas数据可视化

```
df.plot(x=None, y=None, kind='line', figsize=None,  
use_index=True, title=None, grid=None, legend=True, style=None,  
xticks=None, yticks=None, xlim=None, ylim=None, rot=None,  
fontsize=None, alpha)
```

参数说明：

figsize: 图片尺寸大小。

use_index: 默认用索引做x轴。

title: 图片的标题。

grid: 图片是否有网格。

legend: 子图的图例。

style: 对每列折线图设置线的类型。

8.5 pandas数据可视化

□ 8.4.5 数据分组与聚合

```
df.plot(x=None, y=None, kind='line', figsize=None,  
use_index=True, title=None, grid=None, legend=True, style=None,  
xticks=None, yticks=None, xlim=None, ylim=None, rot=None,  
fontsize=None, alpha)
```

参数说明：

xticks: 设置x轴刻度值，序列形式（比如列表）。

yticks: 设置y轴刻度，序列形式（比如列表）。

xlim: 设置x坐标轴的范围，列表或元组形式。

ylim: 设置y坐标轴的范围，列表或元组形式。

rot: 设置轴标签（轴刻度）的显示旋转度数。

fontsize: 设置轴刻度的字体大小。

alpha: 设置图表填充的不透明(0-1)度。

8.5 pandas数据可视化

□ 8.5.1 绘制折线图

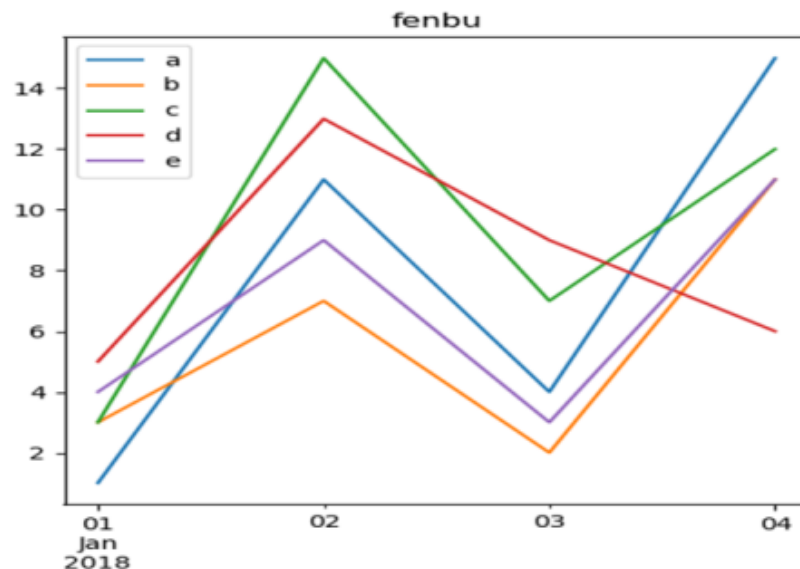
```
>>> df
```

	a	b	c	d
2018-01-01	1	3	3	5
2018-01-02	11	7	15	13
2018-01-03	4	2	7	9
2018-01-04	15	11	12	6

title='fenbu'设置图片的标题, figsize=[5,5] 设置图片尺寸大小

```
>>> df.plot(kind='line', figsize=[5,5], legend=True, title='fenbu')
```

```
>>> plt.show() #显示df.plot()绘制的'line'图
```



第8章 pandas库

- ☐ 8.1 Series对象
- ☐ 8.2 Series对象的基本运算
- ☐ 8.3 DataFrame对象
- ☐ 8.4 DataFrame对象的基本运算
- ☐ 8.5 pandas数据可视化
- ☒ 8.6 pandas读写数据

8.6 pandas读写数据

pandas常用的读写不同格式文件的函数如表所示。

读取函数	写入函数	描述
<code>read_csv()</code>	<code>to_csv()</code>	读写csv格式的数据
<code>read_table()</code>		读取普通分隔符分割的数据
<code>read_excel()</code>	<code>to_excel()</code>	读写excel格式的数据
<code>read_json()</code>	<code>to_json()</code>	读写json格式的数据
<code>read_html()</code>	<code>to_html()</code>	读写html格式的数据
<code>read_sql()</code>	<code>to_sql()</code>	读写数据库中的数据

8.6 pandas读写数据

□ 8.6.1 读写csv文件

(1) 读取csv文件中的数据

student.csv文件内容如下：

Name, Math, Physics, Chemistry

WangLi, 93, 88, 90

ZhangHua, 97, 86, 92

LiMing, 84, 72, 77

ZhouBin, 97, 94, 80

```
>>> csvframe
```

	Name	Math	Physics	Chemistry
0	WangLi	93	88	90
1	ZhangHua	97	86	92
2	LiMing	84	72	77
3	ZhouBin	97	94	80

这个文件以逗号作为分隔符，可使用
pandas的read_csv()函数读取它的内容，返回DataFrame格式的文件。

#从csv中读取数据

```
>>> csvframe = pd.read_csv('student.csv')
```

8.6 pandas读写数据

□ 8.6.1 读写csv文件

(1) 读取csv文件中的数据

csv文件中的数据为列表数据，位于不同列的元素用逗号隔开，csv文件被视作文本文件，也可以使用pandas的read_table()函数读取，但需要指定分隔符。

```
>>> pd.read_table('student.csv',sep=',')
```

	Name	Math	Physics	Chemistry
0	WangLi	93	88	90
1	ZhangHua	97	86	92
2	LiMing	84	72	77
3	ZhouBin	97	94	80

8.6 pandas读写数据

□ 8.6.1 读写csv文件

`pd.read_csv(filepath_or_buffer, sep=',', header='infer', names=None, index_col=None, usecols=None)`

作用：读取csv（逗号分割）文件到DataFrame对象。

filepath_or_buffer：拟要读取的文件的路径。

sep：其类型是str，默认','，用来指定分隔符，csv文件中的分隔符一般为逗号分隔符。

header：指定第几行作为列名，默认为0（第一行）。如果第一行不是列名，是内容，可以设置header=None，以便不把第一行当作列名。

names：用于结果的列名列表，对各列重命名，即添加表头。

index_col：用作行索引的列编号或者列名，可使用index_col=[0, 1]来指定文件中的第1和2列为行索引。

usecols：选取某几列，不读取整个文件的内容，如usecols=[1, 2]。

8.6 pandas读写数据

□ 8.6.1 读写csv文件

header参数可以是一个列表，例如[0, 2]，这个列表表示将文件中的这些行作为列标题，介于中间的行将被忽略掉（例如本例中的行号为0、2的行将被作为多级标题出现，行号为1的行将被丢弃，**dataframe**的数据从行号为3的行开始）。

#指定csv文件中的行号为0、2的行为列标题

```
>>> csvframe = pd.read_csv('student.csv',header=[0,2])
```

```
>>> csvframe
```

	Name	Math	Physics	Chemistry
	ZhangHua	97	86	92
0	LiMing	84	72	77
1	ZhouBin	97	94	80

8.6 pandas读写数据

□ 8.6.1 读写csv文件

(2) 往csv文件写入数据

把DataFrame对象中的数据写入csv文件，要用到to_csv()函数：

```
DataFrame.to_csv(path_or_buf=None, sep=',', na_rep='',  
columns=None, header=True, index=True)
```

作用：以逗号为分隔将DataFrame对象中的数据写入csv文件中。

filepath_or_buffer：拟要写入的文件的路径或对象。

sep：默认字符','，用来指定输出文件的字段分隔符。

na_rep：字符串，默认''，缺失数据表示。

columns：指定要写入文件的列。

header：是否保存列名，默认为True，保存。如果给定字符串列表，则将其作为列名的别名。

index：是否保存行索引，默认为True，保存。

8.6 pandas读写数据

□ 8.6.1 读写csv文件

(2) 往csv文件写入数据

```
>>> df
```

	book	box	pen
2018-08-01	12	3	5
2018-08-02	13	8	7
2018-08-03	15	13	12
2018-08-04	22	18	15

```
>>> df.to_csv('bbp1.csv', index=False, header=False)
```

生成的bbp1.csv文件的内容如下：

12, 3, 5

13, 8, 7

15, 13, 12

22, 18, 15

8.6 pandas读写数据

□ 8.6.2 读取txt文件

pandas的函数`read_table()`可读取txt文本文件。

```
pandas.read_table(filepath_or_buffer, sep='\t', header='infer',  
names=None, index_col=None, skiprows=None, nrows=None,  
delim_whitespace=False)
```

作用：读取以'\t'分割的文件，返回DataFrame对象。

参数说明：

sep：其类型是str，用来指定分隔符，默认为制表符，可以是正则表达式。

index_col：指定行索引。

skiprows：用来指定读取时要排除的行。

nrows：从文件中要读取的行数。

delim_whitespace：`delim_whitespace=True`表示用空格来分隔每行。

8.6 pandas读写数据

□ 8.6.2 读取txt文件

首先在工作目录下创建名为1.txt的文本文件，其内容如下：

C Python Java

1 4 5

3 3 4

4 2 3

2 1 1

>>> pd.read_table('1.txt') #读取1.txt文本文件

C Python Java

0 1 4 5

1 3 3 4

2 4 2 3

3 2 1 1

8.6 pandas读写数据

□ 8.6.3 读写Excel文件

pandas提供了`read_excel()`函数来读取Excel文件，用`to_excel()`函数往Excel文件写入数据。

`pandas.read_excel(io, sheet_name=0, header=0, names=None, index_col=None, usecols=None, skiprows=None, skip_footer=0)`

作用：读取Excel文件中的数据，返回一个DataFrame对象。

参数说明：

io：Excel文件路径，是一个字符串。

sheet_name：返回指定的sheet（表），如果将`sheet_name`指定为`None`，则返回全表；如果需要返回多个表，可以将`sheet_name`指定为一个列表，例如`['sheet1', 'sheet2']`；可以根据sheet的名字字符串或索引来值指定所要选取的sheet，例如`[0, 1, 'SEET5']`将返回第一、第二和第五个表；默认返回第一个表。

8.6 pandas读写数据

□ 8.6.3 读写Excel文件

`pandas.read_excel(io, sheet_name=0, header=0, names=None, index_col=None, usecols=None, skiprows=None, skip_footer=0)`

作用：读取Excel文件中的数据，返回一个DataFrame对象。

参数说明：

header：指定作为列名的行，默认0，即取第一行，数据为列名行以下的的数据；若数据不含列名，则设定**header = None**。

names：指定所生成的DataFrame对象的列的名字，传入一个list数据。

index_col：指定某列为行索引。

usecols：通过名字或索引值读取指定的列。

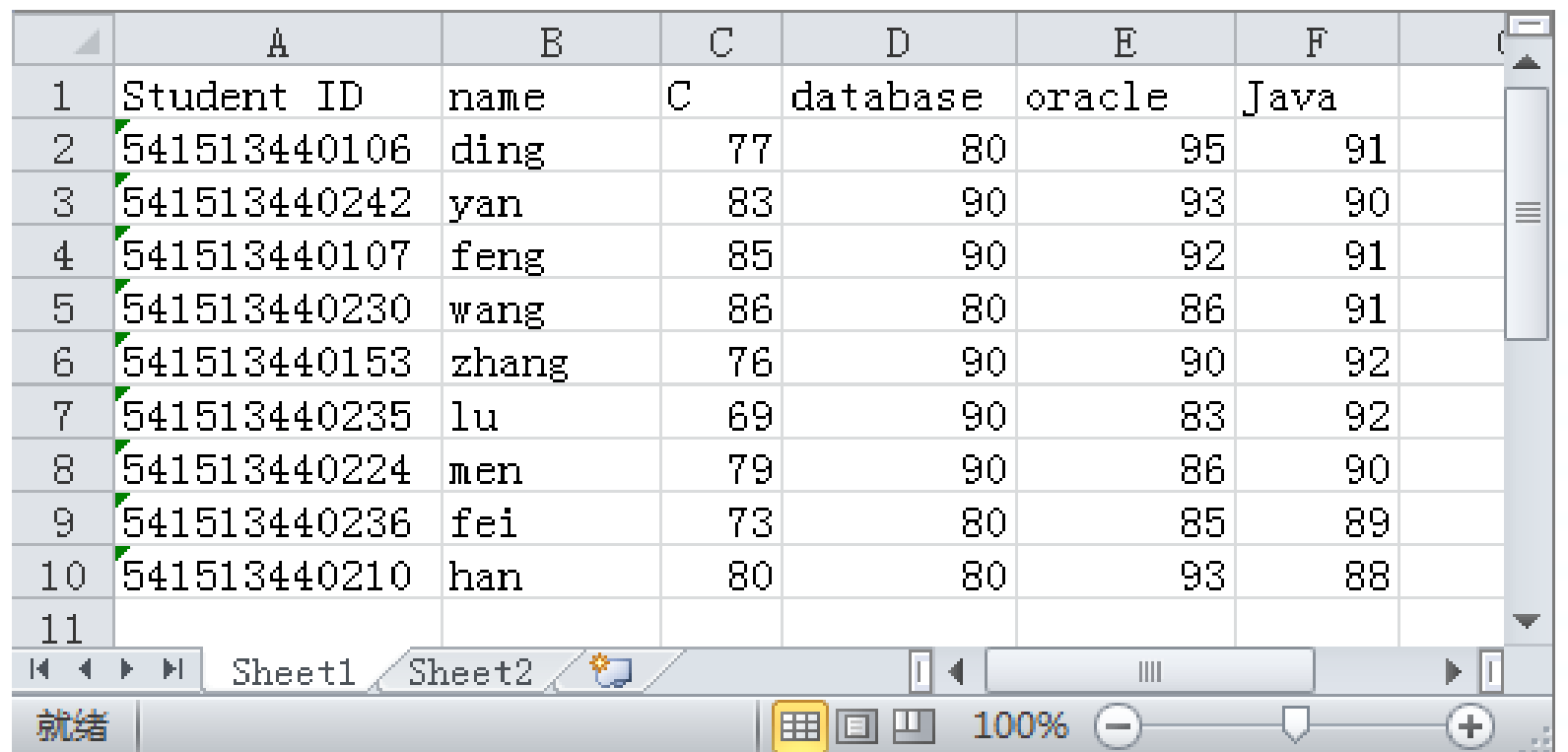
skiprows：省略指定行数的数据。

skip_footer：int，默认值为0，读取数据时省略最后int行。

8.6 pandas读写数据

□ 8.6.3 读写Excel文件

首先在工作目录下创建名为chengji.xlsx的Excel文件，Sheet1的内容表所示。



The image shows a screenshot of an Excel spreadsheet with the following data:

	A	B	C	D	E	F
1	Student ID	name	C	database	oracle	Java
2	541513440106	ding	77	80	95	91
3	541513440242	yan	83	90	93	90
4	541513440107	feng	85	90	92	91
5	541513440230	wang	86	80	86	91
6	541513440153	zhang	76	90	90	92
7	541513440235	lu	69	90	83	92
8	541513440224	men	79	90	86	90
9	541513440236	fei	73	80	85	89
10	541513440210	han	80	80	93	88
11						

The spreadsheet interface includes a status bar at the bottom with the text "就绪" (Ready) and a zoom level of 100%.

8.6 pandas读写数据

□ 8.6.3 读写Excel文件

#将chengji.xlsx的列名作为所生成的DataFrame对象的第一行数据，并重新生成索引

```
>>> pd.read_excel('chengji.xlsx',header=None)
```

	0	1	2	3	4	5
0	Student ID	name	C	database	oracle	Java
1	541513440106	ding	77	80	95	91
2	541513440242	yan	83	90	93	90
3	541513440107	feng	85	90	92	91
.....						
7	541513440224	men	79	90	86	90
8	541513440236	fei	73	80	85	89
9	541513440210	han	80	80	93	88

8.6 pandas读写数据

□ 8.6.3 读写Excel文件

(2) 往Excel文件写入数据

把DataFrame对象df中的数据写入Excel文件的函数为df.to_excel()。

df.to_excel(excel_writer, sheet_name='Sheet1', na_rep='', columns=

None, header=True, index=True, startcol=0, engine=None)

startcol=0, engine=None)

excel_writer: ^{df}输出路径。

sheet_name: 将数据存储在Excel的哪个Sheet页面，如Sheet1页面。

na_rep: 缺失值填充。

columns: 选择输出的列。

header: 指定列名，布尔或字符串列表，默认为True，如果给定字符串列表，则假定它是列名称的别名。header = False则不输出题头。

index: 布尔型，默认True，显示行索引（名字），当index=False，则不显示行索引（名字）。

注意：使用to_excel()函数之前，需要先通过“pip install openpyxl”安装openpyxl模块。

8.6 pandas读写数据

□ 8.6.3 读写Excel文件

(2) 往Excel文件写入数据

```
>>> df
```

```
   course grade
0        C    B
1    Java    A
2  Python    A
3  Hadoop    B
```

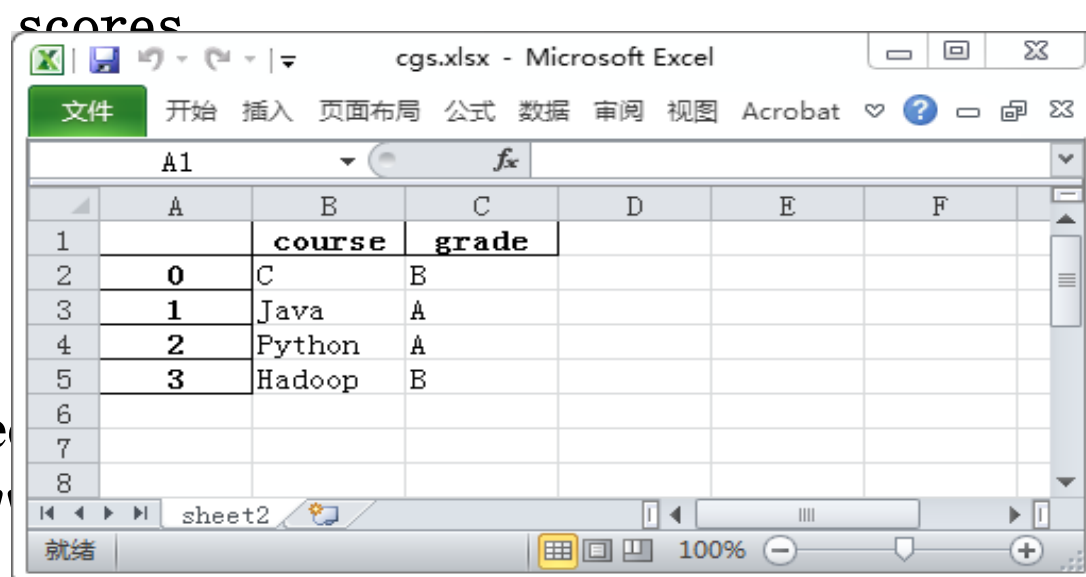
```
'''sheet_name="sheet2"'''
```

```
df.to_excel(excel_writer="cgs.xlsx",sheet_name="sheet2",columns
```

```
=["course","grade"])
```

```
df.to_excel(excel_writer="cgs.xlsx",sheet_name="sheet2",columns
```

生成的cgs.xlsx文件表其内容如表所示。



	A	B	C	D	E	F
1		course	grade			
2	0	C	B			
3	1	Java	A			
4	2	Python	A			
5	3	Hadoop	B			
6						
7						
8						

umns

第8章 pandas库

练习：demo1

由于体育课全被数学老师抢了，所以学生成绩表中的体育期末成绩全为空，将其删除。并添加一列总成绩来求得每位学生的总成绩。

THE END