

Python数据分析

(6)

殷传涛 教授

chuantao.yin@buaa.edu.cn

Pandas入门

► Pandas介绍

- Panel Data Analysis

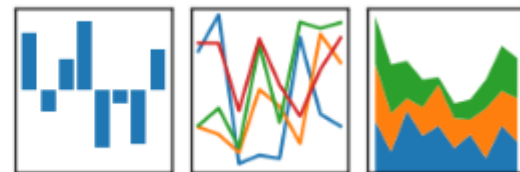
- 一个开源的Python库

- 提供Series, DataFrame, Panel等对象及其处理方法

- 官方网站是: <http://pandas.pydata.org/>

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



- 快速高效的DataFrame对象，具有默认和自定义的索引。
- 将数据从不同文件格式加载到内存中的数据对象。
- 丢失数据的数据对齐和综合处理。
- 重组和摆动日期集。
- 基于标签的切片，索引和大数据集的子集。
- 可以删除或插入来自数据结构的列。
- 按数据分组进行聚合和转换。
- 高性能合并和数据加入。
- 时间序列功能。

Pandas入门



[About us](#) ▾ [Getting started](#) [Documentation](#) [Community](#) ▾ [Contrib](#)

pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

[Install pandas now!](#)

Latest version: 2.2.2

- [What's new in 2.2.2](#)
- Release date:
Apr 10, 2024
- [Documentation \(web\)](#)
- [Download source code](#)

Follow us



Getting started

- [Install pandas](#)
- [Getting started](#)

Documentation

- [User guide](#)
- [API reference](#)
- [Contributing to pandas](#)
- [Release notes](#)

Community

- [About pandas](#)
- [Ask a question](#)
- [Ecosystem](#)

Recommended books



Pandas与Numpy



▶ 相同点

- Numpy和Pandas都是Python的第三方库，用于数据处理和分析。
- 它们都提供了高效的数据结构和函数，可以处理大规模数据。
- Numpy和Pandas都支持向量化操作，可以对整个数组或数据框进行快速计算
- 它们都具有广泛的功能和方法，可以进行数据的读取、转换、筛选、聚合等操作。

Pandas与Numpy

▶ 应用不同

- Numpy主要用于数值计算和科学计算，提供了多维数组对象(ndarray)和相关的数学函数
- Pandas则更适用于数据处理和分析，提供了数据框(DataFrame)和序列(Series)等数据结构

▶ 数据结构不同

- Numpy的数据结构是多维数组，适用于处理数值型数据
- Pandas的数据结构更灵活，可以处理不同类型的数据包括数值型、字符串型、时间序列等

▶ 操作不同

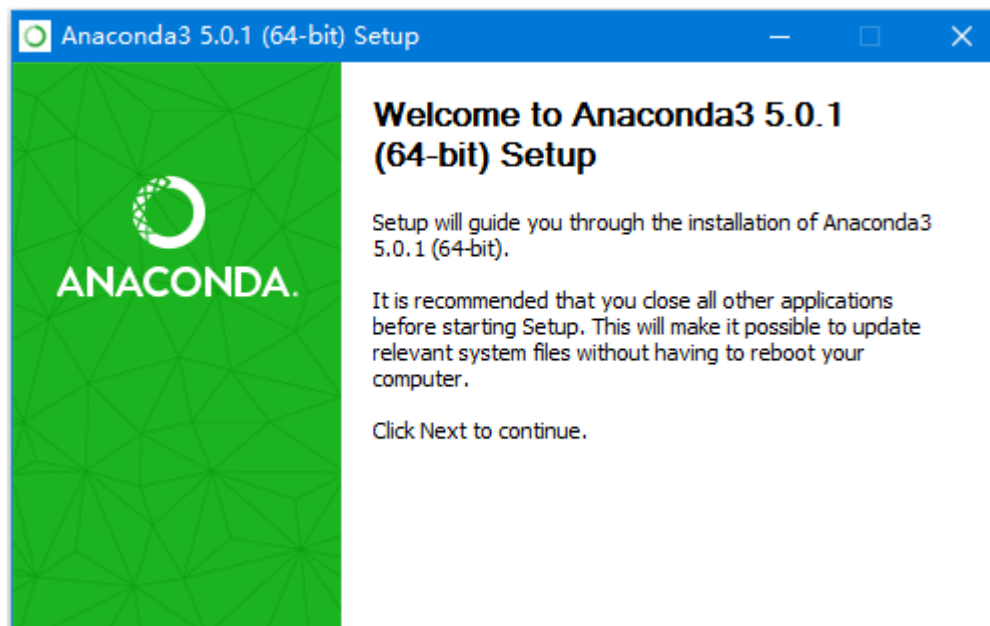
- Numpy的操作更底层，更适合进行数值计算和数组操作
- Pandas提供了更高级的数据操作和分析功能，例如数据的合并、重塑、分组、透视等

▶ 性能不同

- Numpy的性能更高，适用于处理大规模的数值计算
- 而Pandas的性能相对较低，但更适合处理结构化的数据和进行数据分析

Pandas的安装

- ▶ Python命令行安装
 - `pip install pandas`
- ▶ Anaconda安装
 - 自动包含了pandas



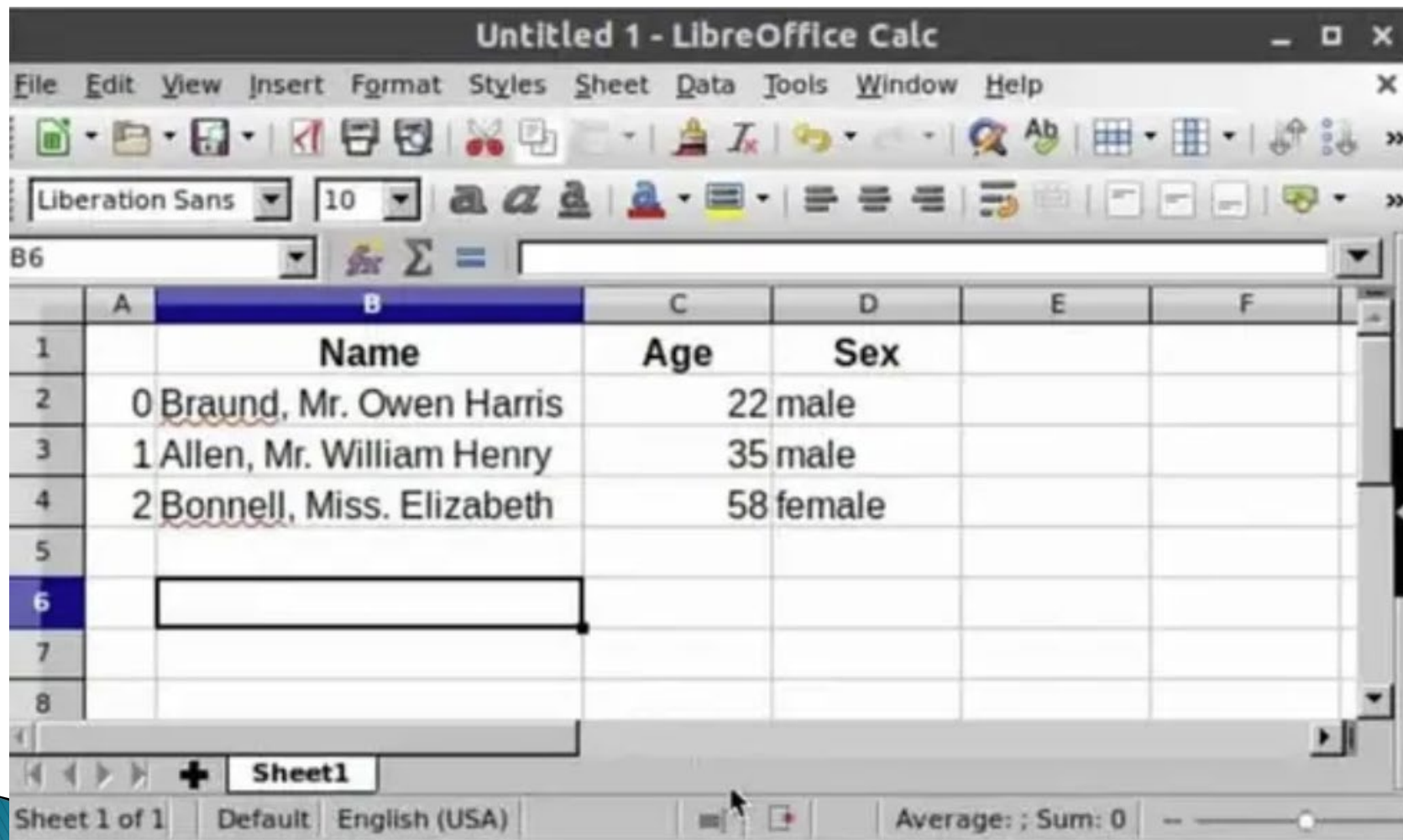
Pandas中的数据结构

- ▶ Pandas中的数据结构
 - 序列 (Series)
 - 数据框 (DataFrame)

数据结构	维数	描述
系列	1	1 D标记均匀数组，大小不变。
数据帧	2	一般 2 D标记，大小可变的表结构与潜在的异质类型的列。

- ▶ 以Numpy数组为基础进行构建

Pandas中的数据结构



Untitled 1 - LibreOffice Calc

File Edit View Insert Format Styles Sheet Data Tools Window Help

Liberation Sans 10

B6

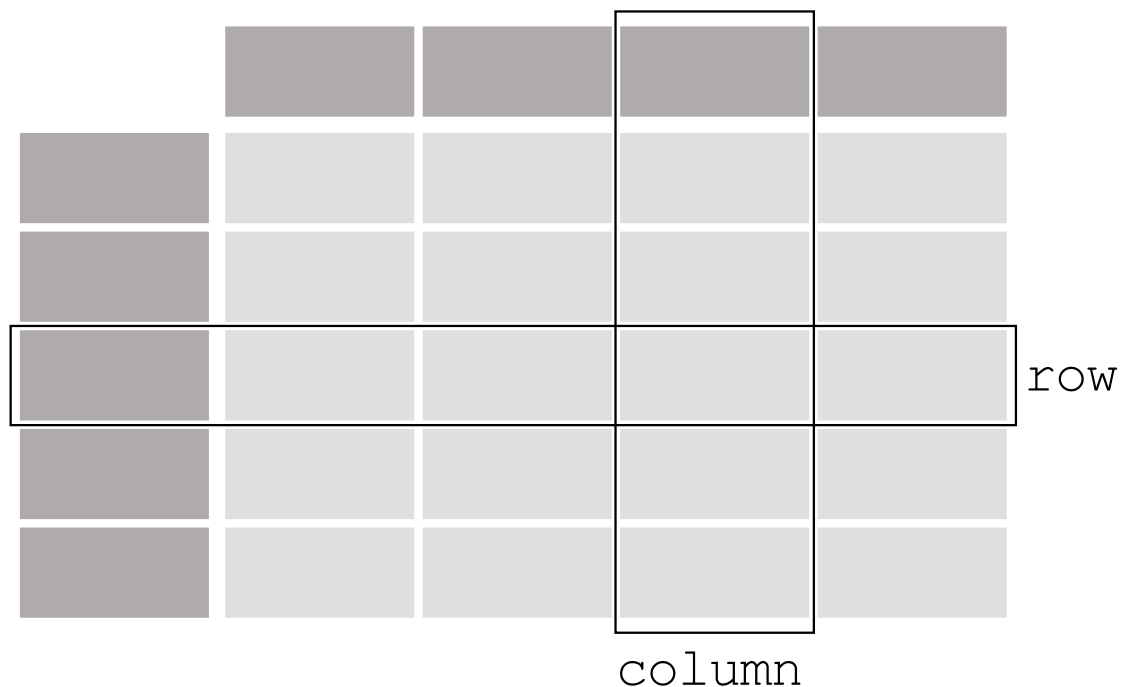
	A	B	C	D	E	F
1		Name	Age	Sex		
2	0	Braund, Mr. Owen Harris	22	male		
3	1	Allen, Mr. William Henry	35	male		
4	2	Bonnell, Miss. Elizabeth	58	female		
5						
6						
7						
8						

Sheet1

Sheet 1 of 1 Default English (USA) Average: ; Sum: 0

Pandas中的数据结构

DataFrame



Pandas中的数据结构

► 序列 Series

- Pandas 中的一种基本数据结构，类似于二维数组或列表，
- 具有标签（索引）
- 数据格式均匀，值可变



A	B
	Name
0	Braund, Mr. Owen Harris
1	Allen, Mr. William Henry
2	Bonnell, Miss. Elizabeth



```
a    1
b    2
c    3
d    4
dtype: int64
```

Series

Pandas中的数据结构

► 序列 Series

- 用pandas.Series(data, index, dtype, copy)创建

1	<code>data</code>	数据采取各种形式, 如: <code>ndarray</code> , <code>list</code> , <code>constants</code>
2	<code>index</code>	索引值必须是唯一的和散列的, 与数据的长度相同。默认 <code>np.arange(n)</code> 如果没有索引被传递。
3	<code>dtype</code>	<code>dtype</code> 用于数据类型。如果没有, 将推断数据类型
4	<code>copy</code>	复制数据, 默认为 <code>false</code> 。

```
import numpy as np
import pandas as pd
s = pd.Series()
print(s)
```

Series([], dtype: float64)

```
import pandas as pd
import numpy as np
data = np.arange(6,11)
s = pd.Series(data)
print(s)
```

```
0    6
1    7
2    8
3    9
4   10
dtype: int32
```

```
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data)
print(s)
```

```
0    a
1    b
2    c
3    d
dtype: object
```

Pandas中的数据结构

► 序列 Series

- 自定义索引
 - 通过指定数据定义
 - 通过字典的键定义

```
import pandas as pd
import numpy as np
data = np.array(['a', 'b', 'c', 'd'])
s = pd.Series(data, index=[100, 101, 102, 103])
print(s)
```

```
100    a
101    b
102    c
103    d
dtype: object
```

```
import pandas as pd
import numpy as np
data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = pd.Series(data)
print(s)
```

```
a    0.0
b    1.0
c    2.0
dtype: float64
```

```
import pandas as pd
import numpy as np
data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = pd.Series(data, index=['b', 'c', 'd', 'a'])
print(s)
```

```
b    1.0
c    2.0
d    NaN
a    0.0
dtype: float64
```

Pandas中的数据结构

► 序列 Series

- 从单个数值创建序列
 - 必须提供索引值，以确定序列长度
 - 每个元素的值均是设定的数值

```
import pandas as pd
import numpy as np
s = pd.Series(5, index=[0, 1, 2, 3])
print (s)
```

```
0    5
1    5
2    5
3    5
dtype: int64
```

```
import pandas as pd
import numpy as np
s = pd.Series(5, index=np.arange(10))
print (s)
```

```
0    5
1    5
2    5
3    5
4    5
5    5
6    5
7    5
8    5
9    5
dtype: int64
```

Pandas中的数据结构

► 序列 Series

- 类似于numpy多维数组，可通过位置编号去访问元素

```
import pandas as pd
import numpy as np
s = pd.Series([1,2,3,4,5], index = ['a','b','c','d','e'])
print(s)
print (s[0])
print (s[:3])
print (s[3:])
print (s[2:3])
print (s[:2])
print (s[-5:-2])
```

```
a    1
b    2
c    3
d    4
e    5
dtype: int64
1
a    1
b    2
c    3
dtype: int64
d    4
e    5
dtype: int64
c    3
dtype: int64
a    1
c    3
e    5
dtype: int64
a    1
b    2
c    3
dtype: int64
```


Pandas中的数据结构

► 序列 Series

- 可通过索引值去访问元素

```
import pandas as pd
import numpy as np
s = pd.Series([1,2,3,4,5], index = ['a','b','c','d','e'])
print(s)
print(s['a'])
print(s[['a','b','e']])
print(s['f'])
```

```
a    1
b    2
c    3
d    4
e    5
dtype: int64
1
a    1
b    2
e    5
dtype: int64
```

```
TypeError                                Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_value(self, s
series, key)
    4410
-> 4411         return libindex.get_value_at(s, key)
    4412     except IndexError:
```

Series 序列的属性

► Series序列具有一些基本属性或方法

系列基本功能		
编号	属性或方法	描述
1	<code>axes</code>	返回行轴标签列表。
2	<code>dtype</code>	返回对象的数据类型(<code>dtype</code>)。
3	<code>empty</code>	如果系列为空, 则返回 <code>True</code> 。
4	<code>ndim</code>	返回底层数据的维数, 默认定义: <code>1</code> 。
5	<code>size</code>	返回基础数据中的元素数。
6	<code>values</code>	将系列作为 <code>ndarray</code> 返回。
7	<code>head()</code>	返回前 <code>n</code> 行。
8	<code>tail()</code>	返回最后 <code>n</code> 行。

Series 序列的属性

► Series序列的基本属性和方法

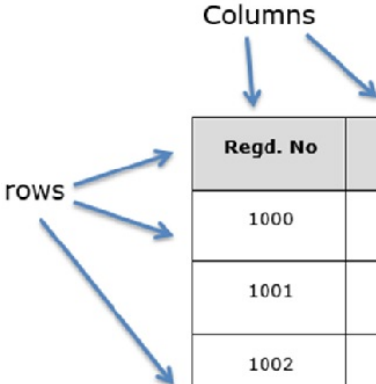
```
import pandas as pd
import numpy as np
data = np.array(['a', 'b', 'c', 'd'])
s = pd.Series(data, index=[100, 101, 102, 103])
print(s)
print (s.axes)
print (s.dtype)
print (s.empty)
print (s.ndim)
print (s.size)
print (s.values)
print (s.head(3))
print (s.tail(2))
```

```
100    a
101    b
102    c
103    d
dtype: object
[Int64Index([100, 101, 102, 103], dtype='int64')]
object
False
1
4
['a' 'b' 'c' 'd']
100    a
101    b
102    c
dtype: object
102    c
103    d
dtype: object
```

Pandas中的数据结构

▶ 数据框 DataFrame

- 列、类型、行
- 异构数据，大小可变，数据可变



Regd. No	Name	Marks%
1000	Steve	86.29
1001	Mathew	91.63
1002	Jose	72.90
1003	Patty	69.23
1004	Vin	88.30

数据帧(*DataFrame*)是一个具有异构数据的二维数组。 例如,

姓名	年龄	性别	等级
Maxsu	25	男	4.45
Katie	34	女	2.78
Vina	46	女	3.9
Lia	女	x女	4.6

列	类型
姓名	字符串
年龄	整数
性别	字符串
等级	浮点型

Pandas中的数据结构

▶ 数据框 DataFrame

- `pandas.DataFrame(data, index, columns, dtype, copy)`
- 可以从列表、字典、序列、numpy多维数组创建

1	<code>data</code>	数据采取各种形式, 如: <code>ndarray</code> , <code>series</code> , <code>map</code> , <code>lists</code> , <code>dict</code> , <code>constant</code> 和另一个 <code>DataFrame</code> 。
2	<code>index</code>	对于行标签, 要用于结果帧的索引是可选缺省值 <code>np.arange(n)</code> , 如果没有传递索引值。
3	<code>columns</code>	对于列标签, 可选的默认语法是 - <code>np.arange(n)</code> 。 这只有在没有索引传递的情况下才是这样。
4	<code>dtype</code>	每列的数据类型。
5	<code>copy</code>	如果默认值为 <code>False</code> , 则此命令(或任何它)用于复制数据。

Pandas中的数据结构

► 创建数据框 DataFrame

- 索引（行），如果缺省的话则使用np.arange()
- 列，如果缺省的话则使用np.arange()

```
import pandas as pd
df = pd.DataFrame()
print(df)
```

```
Empty DataFrame
Columns: []
Index: []
```

```
: import pandas as pd
data = [1,2,3,4,5]
df = pd.DataFrame(data)
print(df)
```

```
0
0 1
1 2
2 3
3 4
4 5
```

```
import pandas as pd
import numpy as np
df = pd.DataFrame((np.arange(24)*2).reshape(4,6))
print(df)
```

```
0 1 2 3 4 5
0 0 2 4 6 8 10
1 12 14 16 18 20 22
2 24 26 28 30 32 34
3 36 38 40 42 44 46
```


Pandas中的数据结构

- ▶ 创建数据框 DataFrame
 - 使用列表和列标签进行创建
 - 数据类型的修改

```
import pandas as pd
data = [['Alex', 10], ['Bob', 12], ['Clarke', 13]]
df = pd.DataFrame(data, columns=['Name', 'Age'])
print(df)
```

	Name	Age
0	Alex	10
1	Bob	12
2	Clarke	13

```
import pandas as pd
data = [['Alex', 10], ['Bob', 12], ['Clarke', 13]]
df = pd.DataFrame(data, columns=['Name', 'Age'], dtype=float)
print(df)
```

	Name	Age
0	Alex	10.0
1	Bob	12.0
2	Clarke	13.0

Pandas中的数据结构

► 创建数据框 DataFrame

- 使用字典来创建

```
import pandas as pd
data = {'Name': ['Tom', 'Jack', 'Steve', 'Ricky'], 'Age': [28, 34, 29, 42]}
df = pd.DataFrame(data)
print(df)
```

	Name	Age
0	Tom	28
1	Jack	34
2	Steve	29
3	Ricky	42

```
import pandas as pd
data = {'Name': ['Tom', 'Jack', 'Steve', 'Ricky'], 'Age': [28, 34, 29, 42]}
df = pd.DataFrame(data, index=['rank1', 'rank2', 'rank3', 'rank4'])
print(df)
```

	Name	Age
rank1	Tom	28
rank2	Jack	34
rank3	Steve	29
rank4	Ricky	42

Pandas中的数据结构

► 创建数据框 DataFrame

- 从字典创建
- 从series序列创建

```
import pandas as pd
data = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data)
print(df)
```

	a	b	c
0	1	2	NaN
1	5	10	20.0

```
import pandas as pd
data = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data, index=['first', 'second'])
print(df)
```

	a	b	c
first	1	2	NaN
second	5	10	20.0

```
import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
     'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)
print(df)
```

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

DataFrame 数据框的属性

► DataFrame数据框的基本属性和方法

编号	属性或方法	描述
1	<code>T</code>	转置行和列。
2	<code>axes</code>	返回一个列，行轴标签和列轴标签作为唯一的成员。
3	<code>dtypes</code>	返回此对象中的数据类型(<code>dtypes</code>)。
4	<code>empty</code>	如果 <code>NDFrame</code> 完全为空[无项目]，则返回为 <code>True</code> ；如果任何轴的长度为 <code>0</code> 。
5	<code>ndim</code>	轴/数组维度大小。
6	<code>shape</code>	返回表示 <code>DataFrame</code> 的维度的元组。
7	<code>size</code>	<code>NDFrame</code> 中的元素数。
8	<code>values</code>	NDFrame的Numpy表示。
9	<code>head()</code>	返回开头前 <code>n</code> 行。
10	<code>tail()</code>	返回最后 <code>n</code> 行。

DataFrame 数据框的属性

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve',
                      'Age':pd.Series([25,26,25,23,30,29,23]),
                      'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])

#Create a DataFrame
df = pd.DataFrame(d)
print ("Our data series is:")
print (df)
print (df.T)
print (df.axes)
print (df.dtypes)
print (df.empty)
print (df.ndim)
print (df.shape)
print (df.size)
print (df.values)
print (df.head(3))
print (df.tail(2))
```

Our data series is:

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56
4	Steve	30	3.20
5	Minsu	29	4.60
6	Jack	23	3.80

	0	1	2	3	4	5	6
Name	Tom	James	Ricky	Vin	Steve	Minsu	Jack
Age	25	26	25	23	30	29	23
Rating	4.23	3.24	3.98	2.56	3.2	4.6	3.8

[RangeIndex(start=0, stop=7, step=1), Index(['Name', 'Age', 'Rating'], dtype='object')]

Name object

Age int64

Rating float64

dtype: object

False

2

(7, 3)

21

[['Tom' 25 4.23]

['James' 26 3.24]

['Ricky' 25 3.98]

['Vin' 23 2.56]

['Steve' 30 3.2]

['Minsu' 29 4.6]

['Jack' 23 3.8]]

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98

	Name	Age	Rating
5	Minsu	29	4.6
6	Jack	23	3.8

DataFrame的数据访问

- ▶ 使用属性运算符.来选择列

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(8, 4), columns = ['A', 'B', 'C', 'D'])

print (df.C)
```

```
0    1.533123
1   -0.722758
2    0.498297
3    0.832395
4   -1.642713
5    0.041027
6   -1.012724
7   -0.741129
Name: C, dtype: float64
```


DataFrame的数据访问

- ▶ 像其它序列数据一样，使用 [] 进行索引

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(8, 4), columns = ['A', 'B', 'C', 'D'])
print (df['A'])
print (df[['A', 'C']])
print (df[1:3])
```

```
0    0.535041
1   -1.678345
2   -0.167819
3   -0.934546
4    0.348999
5    0.529115
6   -1.246337
7   -0.276454
```

Name: A, dtype: float64

```
      A      C
0  0.535041  0.190204
1 -1.678345 -1.964838
2 -0.167819  0.340459
3 -0.934546 -2.481086
4  0.348999 -1.067456
5  0.529115 -1.481631
6 -1.246337 -0.210450
7 -0.276454  0.299557
```

```
      A      B      C      D
1 -1.678345  1.385971 -1.964838  0.665639
2 -0.167819  1.642409  0.340459  0.344061
```

DataFrame的数据访问

- ▶ `loc[]`
 - 对行（索引）和列（属性）进行索引
 - `loc[行, 列]`

- ▶ 用`loc[]`选择行

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(8, 4),
index = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'], columns = ['A', 'B', 'C', 'D'])

#select all rows for a specific column
print(df)
print (df.loc['a':'d'])
print (df.loc[['a', 'b', 'd']])
```

	A	B	C	D
a	1.242329	-0.264037	0.394992	0.518599
b	-1.166217	0.544554	-2.191570	1.376396
c	0.785907	-0.890341	-1.168464	-0.225913
d	-0.329196	-0.280407	-1.053630	0.124817
e	-0.362039	-0.992861	-1.784398	-0.160440
f	0.743550	-1.191295	-0.758145	-1.364801
g	0.847752	-0.408057	-0.482668	1.753288
h	0.758198	-1.096082	-0.608332	0.228507

	A	B	C	D
a	1.242329	-0.264037	0.394992	0.518599
b	-1.166217	0.544554	-2.191570	1.376396
c	0.785907	-0.890341	-1.168464	-0.225913
d	-0.329196	-0.280407	-1.053630	0.124817

	A	B	C	D
a	1.242329	-0.264037	0.394992	0.518599
b	-1.166217	0.544554	-2.191570	1.376396
d	-0.329196	-0.280407	-1.053630	0.124817

DataFrame的数据访问

- ▶ loc[]
 - 对行（索引）和列（属性）进行索引
 - loc[行, 列]
- ▶ 用loc[]选择列

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(8, 4),
index = ['a','b','c','d','e','f','g','h'], columns = ['A', 'B', 'C', 'D'])

#select all rows for a specific column
print(df)
print (df.loc[:, ['A', 'C']])
print (df.loc[:, 'B':'D'])
```

	A	B	C	D
a	-0.602487	1.635987	0.302517	1.440742
b	0.348930	-0.604218	-0.254037	0.843374
c	0.537780	-0.014482	-2.151472	-1.765111
d	0.734376	-0.373812	-0.580278	-2.002795
e	-1.084220	1.107000	0.390228	0.113948
f	-0.124250	0.400670	-0.652855	1.087088
g	-0.491426	0.440912	-0.364870	0.283550
h	0.404972	0.497218	0.956159	0.185127

	A	C
a	-0.602487	0.302517
b	0.348930	-0.254037
c	0.537780	-2.151472
d	0.734376	-0.580278
e	-1.084220	0.390228
f	-0.124250	-0.652855
g	-0.491426	-0.364870
h	0.404972	0.956159

	B	C	D
a	1.635987	0.302517	1.440742
b	-0.604218	-0.254037	0.843374
c	-0.014482	-2.151472	-1.765111
d	-0.373812	-0.580278	-2.002795
e	1.107000	0.390228	0.113948
f	0.400670	-0.652855	1.087088
g	0.440912	-0.364870	0.283550
h	0.497218	0.956159	0.185127

DataFrame的数据访问

- ▶ 用loc[]选择特定的行与列，然后进行计算

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(8, 4),
index = ['a','b','c','d','e','f','g','h'], columns = ['A', 'B', 'C', 'D'])

print (df.loc['e':'g','B':'D']>0)

a= (df.loc[['a','d','h'],['A','C']]*3).min()
print(a)
```

```
      B      C      D
e  True  False  True
f  True   True  False
g  True  False  False
A -0.867449
C -3.261867
dtype: float64
```

DataFrame的数据访问

- ▶ `iloc[]`
 - 用纯整数获得位置索引
 - `iloc[: , :]`

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(8, 4),
index = ['a','b','c','d','e','f','g','h'], columns = ['A', 'B', 'C', 'D'])

#select all rows for a specific column
print(df)
print(df.iloc[2,3])
print(df.iloc[0:4,2:4])
print(df.iloc[:,1:4])
```

	A	B	C	D
a	-0.273230	1.123049	1.240904	-0.747489
b	0.785923	0.104906	0.356446	3.300909
c	1.227198	0.878074	0.801194	2.193329
d	-0.264067	-0.175093	-0.932446	0.462531
e	-0.972333	1.026779	-0.971462	0.848920
f	-0.473528	-0.008426	-0.664078	-0.028129
g	0.357411	-1.172513	-1.380107	0.318769
h	-0.311122	0.869946	0.532915	0.200803

	C	D
a	1.240904	-0.747489
b	0.356446	3.300909
c	0.801194	2.193329
d	-0.932446	0.462531

	B	C	D
a	1.123049	1.240904	-0.747489
b	0.104906	0.356446	3.300909
c	0.878074	0.801194	2.193329
d	-0.175093	-0.932446	0.462531
e	1.026779	-0.971462	0.848920
f	-0.008426	-0.664078	-0.028129
g	-1.172513	-1.380107	0.318769
h	0.869946	0.532915	0.200803

DataFrame的数据分组

► 利用groupby()产生分组

```
import pandas as pd

ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',
                    'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],
            'Rank': [1, 2, 2, 3, 3, 4, 1, 1, 2, 4, 1, 2],
            'Year': [2014, 2015, 2014, 2015, 2014, 2015, 2016, 2017, 2016, 2014, 2015, 2017],
            'Points': [876, 789, 863, 673, 741, 812, 756, 788, 694, 701, 804, 690]}

df = pd.DataFrame(ipl_data)

print (df)
print ()
print (df.groupby('Team').groups)
print ()
print (df.groupby(['Team', 'Year']).groups)
```

	Team	Rank	Year	Points
0	Riders	1	2014	876
1	Riders	2	2015	789
2	Devils	2	2014	863
3	Devils	3	2015	673
4	Kings	3	2014	741
5	kings	4	2015	812
6	Kings	1	2016	756
7	Kings	1	2017	788
8	Riders	2	2016	694
9	Royals	4	2014	701
10	Royals	1	2015	804
11	Riders	2	2017	690

```
{'Devils': Int64Index([2, 3], dtype='int64'), 'Kings': Int64Index([4, 6, 7], dtype='int64'), 'Riders': Int64Index([0, 1, 8, 11], dtype='int64'), 'Royals': Int64Index([9, 10], dtype='int64'), 'kings': Int64Index([5], dtype='int64')}
```

```
{('Devils', 2014): Int64Index([2], dtype='int64'), ('Devils', 2015): Int64Index([3], dtype='int64'), ('Kings', 2014): Int64Index([4], dtype='int64'), ('Kings', 2016): Int64Index([6], dtype='int64'), ('Kings', 2017): Int64Index([7], dtype='int64'), ('Riders', 2014): Int64Index([0], dtype='int64'), ('Riders', 2015): Int64Index([1], dtype='int64'), ('Riders', 2016): Int64Index([8], dtype='int64'), ('Riders', 2017): Int64Index([11], dtype='int64'), ('Royals', 2014): Int64Index([9], dtype='int64'), ('Royals', 2015): Int64Index([10], dtype='int64'), ('kings', 2015): Int64Index([5], dtype='int64')}
```


DataFrame的数据分组

► 利用get_group获得分组

```
import pandas as pd

ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',
                    'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],
            'Rank': [1, 2, 2, 3, 3, 4, 1, 1, 2, 4, 1, 2],
            'Year': [2014, 2015, 2014, 2015, 2014, 2015, 2016, 2017, 2016, 2014, 2015, 2017],
            'Points': [876, 789, 863, 673, 741, 812, 756, 788, 694, 701, 804, 690]}

df = pd.DataFrame(ipl_data)

grouped = df.groupby('Year')
print (grouped.get_group(2014))

teamgroup = df.groupby('Team')
print (teamgroup.get_group("Kings"))
```

	Team	Rank	Year	Points
0	Riders	1	2014	876
1	Riders	2	2015	789
2	Devils	2	2014	863
3	Devils	3	2015	673
4	Kings	3	2014	741
5	kings	4	2015	812
6	Kings	1	2016	756
7	Kings	1	2017	788
8	Riders	2	2016	694
9	Royals	4	2014	701
10	Royals	1	2015	804
11	Riders	2	2017	690

	Team	Rank	Year	Points
0	Riders	1	2014	876
2	Devils	2	2014	863
4	Kings	3	2014	741
9	Royals	4	2014	701

	Team	Rank	Year	Points
4	Kings	3	2014	741
6	Kings	1	2016	756
7	Kings	1	2017	788

DataFrame的数据统计

► Pandas中的统计函数

编号	函数	描述
1	<code>count()</code>	非空观测数量
2	<code>sum()</code>	所有值之和
3	<code>mean()</code>	所有值的平均值
4	<code>median()</code>	所有值的中位数
5	<code>mode()</code>	值的模值
6	<code>std()</code>	值的标准偏差
7	<code>min()</code>	所有值中的最小值
8	<code>max()</code>	所有值中的最大值
9	<code>abs()</code>	绝对值
10	<code>prod()</code>	数组元素的乘积
11	<code>cumsum()</code>	累计总和
12	<code>cumprod()</code>	累计乘积

DataFrame的数据统计

► Pandas中的统计函数 sum ()

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Minsu','Jack',
    'Lee','David','Gasper','Betina','Andres']),
    'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
    'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print (df)
print (df.sum())
print (df.sum(axis=1))
```

```
Name      TomJamesRickyVinSteveMinsuJackLeeDavidGasperBe...
Age                                     382
Rating                                     44.92
dtype: object
```

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56
4	Steve	30	3.20
5	Minsu	29	4.60
6	Jack	23	3.80
7	Lee	34	3.78
8	David	40	2.98
9	Gasper	30	4.80
10	Betina	51	4.10
11	Andres	46	3.65

```
0      29.23
1      29.24
2      28.98
3      25.56
4      33.20
5      33.60
6      26.80
7      37.78
8      42.98
9      34.80
10     55.10
11     49.65
dtype: float64
```

DataFrame的数据统计

- ▶ 平均值 `mean()`
- ▶ 中位数 `median()`

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56
4	Steve	30	3.20
5	Minsu	29	4.60
6	Jack	23	3.80
7	Lee	34	3.78
8	David	40	2.98
9	Gaspar	30	4.80
10	Betina	51	4.10
11	Andres	46	3.65

```
print(df.mean())  
print(df.median())
```

```
Age      31.833333  
Rating    3.743333  
dtype: float64  
Age      29.50  
Rating    3.79  
dtype: float64
```

DataFrame的数据统计

- ▶ 标准差 std()
- ▶ 最大值 max() 最小值 min()

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56
4	Steve	30	3.20
5	Minsu	29	4.60
6	Jack	23	3.80
7	Lee	34	3.78
8	David	40	2.98
9	Gaspar	30	4.80
10	Betina	51	4.10
11	Andres	46	3.65

```
print(df.std())  
print(df.max())  
print(df.min())
```

```
Age          9.232682  
Rating       0.661628  
dtype: float64  
Name         Vin  
Age          51  
Rating       4.8  
dtype: object  
Name         Andres  
Age          23  
Rating       2.56  
dtype: object
```

DataFrame的数据统计

- ▶ 汇总描述数据 describe()
 - 默认只统计数值列，排除字符列

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Minsu','Jack',
    'Lee','David','Gasper','Betina','Andres']),
    'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
    'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print (df)

print(df.describe())
```

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56
4	Steve	30	3.20
5	Minsu	29	4.60
6	Jack	23	3.80
7	Lee	34	3.78
8	David	40	2.98
9	Gasper	30	4.80
10	Betina	51	4.10
11	Andres	46	3.65

	Age	Rating
count	12.000000	12.000000
mean	31.833333	3.743333
std	9.232682	0.661628
min	23.000000	2.560000
25%	25.000000	3.230000
50%	29.500000	3.790000
75%	35.500000	4.132500
max	51.000000	4.800000

DataFrame的数据统计

► 汇总描述数据 describe()

- 由include参数进行控制
 - all, number, object

```
import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Minsu','Jack',
    'Lee','David','Gasper','Betina','Andres']),
    'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
    'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame
df = pd.DataFrame(d)
print (df)

print(df.describe(include="all"))
print(df.describe(include="number"))
print(df.describe(include="object"))
```

	Name	Age	Rating
count	12	12.000000	12.000000
unique	12	NaN	NaN
top	Lee	NaN	NaN
freq	1	NaN	NaN
mean	NaN	31.833333	3.743333
std	NaN	9.232682	0.661628
min	NaN	23.000000	2.560000
25%	NaN	25.000000	3.230000
50%	NaN	29.500000	3.790000
75%	NaN	35.500000	4.132500
max	NaN	51.000000	4.800000

	Age	Rating
count	12.000000	12.000000
mean	31.833333	3.743333
std	9.232682	0.661628
min	23.000000	2.560000
25%	25.000000	3.230000
50%	29.500000	3.790000
75%	35.500000	4.132500
max	51.000000	4.800000

	Name
count	12
unique	12
top	Lee
freq	1

DataFrame的日期数据

► 生成日期序列 date_range()

```
import pandas as pd
datelist = pd.date_range('2020/11/21', periods=20)
print(datelist)
```

```
DatetimeIndex(['2020-11-21', '2020-11-22', '2020-11-23', '2020-11-24',
               '2020-11-25', '2020-11-26', '2020-11-27', '2020-11-28',
               '2020-11-29', '2020-11-30', '2020-12-01', '2020-12-02',
               '2020-12-03', '2020-12-04', '2020-12-05', '2020-12-06',
               '2020-12-07', '2020-12-08', '2020-12-09', '2020-12-10'],
              dtype='datetime64[ns]', freq='D')
```

► 更改日期频率 freq参数

```
In [90]: import pandas as pd
          datelist = pd.date_range('2020/11/21', periods=20, freq="M")
          print(datelist)
```

```
DatetimeIndex(['2020-11-30', '2020-12-31', '2021-01-31', '2021-02-28',
               '2021-03-31', '2021-04-30', '2021-05-31', '2021-06-30',
               '2021-07-31', '2021-08-31', '2021-09-30', '2021-10-31',
               '2021-11-30', '2021-12-31', '2022-01-31', '2022-02-28',
               '2022-03-31', '2022-04-30', '2022-05-31', '2022-06-30'],
              dtype='datetime64[ns]', freq='M')
```


DataFrame的日期数据

- ▶ 生成商业日期序列 `bdate_range()`
 - 只包含工作日

```
import pandas as pd
datelist = pd.bdate_range('2011/11/01', periods=7)
print(datelist)
```

```
DatetimeIndex(['2011-11-01', '2011-11-02', '2011-11-03', '2011-11-04',
               '2011-11-07', '2011-11-08', '2011-11-09'],
              dtype='datetime64[ns]', freq='B')
```

```
: import pandas as pd
import datetime as dt
start = dt.date(2020, 6, 1)
end = dt.date(2020, 6, 8)
dates = pd.date_range(start, end)
bdates = pd.bdate_range(start, end)
print(dates)
print(bdates)
```

```
DatetimeIndex(['2020-06-01', '2020-06-02', '2020-06-03', '2020-06-04',
               '2020-06-05', '2020-06-06', '2020-06-07', '2020-06-08'],
              dtype='datetime64[ns]', freq='D')
DatetimeIndex(['2020-06-01', '2020-06-02', '2020-06-03', '2020-06-04',
               '2020-06-05', '2020-06-08'],
              dtype='datetime64[ns]', freq='B')
```

DataFrame的日期数据

► 用日期索引生成DataFrame数据

```
In [5]: dates = pd.date_range('20130101', periods=6)
```

```
In [6]: dates
```

```
Out[6]:
```

```
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',  
               '2013-01-05', '2013-01-06'],  
              dtype='datetime64[ns]', freq='D')
```

```
In [7]: df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))
```

```
In [8]: df
```

```
Out[8]:
```

	A	B	C	D
2013-01-01	0.469112	-0.282863	-1.509059	-1.135632
2013-01-02	1.212112	-0.173215	0.119209	-1.044236
2013-01-03	-0.861849	-2.104569	-0.494929	1.071804
2013-01-04	0.721555	-0.706771	-1.039575	0.271860
2013-01-05	-0.424972	0.567020	0.276232	-1.087401
2013-01-06	-0.673690	0.113648	-1.478427	0.524988

Pandas 文件操作

- ▶ 写入.csv文件
 - to_csv()

```
: import pandas as pd
import numpy as np

#Create a Dictionary of series
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Minsu','Jack']),
     'Age':pd.Series([25,26,25,23,30,29,23]),
     'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

#Create a DataFrame
df = pd.DataFrame(d)
print ("Our data series is:")
print (df)

df.to_csv("student.csv")
```

	A	B	C	D	E
1		Name	Age	Rating	
2	0	Tom	25	4.23	
3	1	James	26	3.24	
4	2	Ricky	25	3.98	
5	3	Vin	23	2.56	
6	4	Steve	30	3.2	
7	5	Minsu	29	4.6	
8	6	Jack	23	3.8	
9					
10					

Pandas 文件操作

- ▶ 读入csv文件
- ▶ read_csv()

```
df2=pd.read_csv("student.csv")
print(df2)

df3=pd.read_csv("student.csv",index_col=[0])
print(df3)
```

	Unnamed: 0	Name	Age	Rating
0	0	Tom	25	4.23
1	1	James	26	3.24
2	2	Ricky	25	3.98
3	3	Vin	23	2.56
4	4	Steve	30	3.20
5	5	Minsu	29	4.60
6	6	Jack	23	3.80

	Name	Age	Rating
0	Tom	25	4.23
1	James	26	3.24
2	Ricky	25	3.98
3	Vin	23	2.56
4	Steve	30	3.20
5	Minsu	29	4.60
6	Jack	23	3.80

Pandas更多应用

- ▶ 参见Pandas官方网站的说明文档：
 - 快速入门：
https://pandas.pydata.org/docs/getting_started/index.html
 - 用户手册：
https://pandas.pydata.org/docs/user_guide/indexing.html

谢谢！