DataFrame是pandas里比较重要的一个类, 也是平常工作中经常使用的一个类。本人之前曾做过一个族谱分析的项目，其中主要运用的知识就是DataFrame。

DataFrame是一个【表格型】的数据结构。DataFrame由按一定顺序排列的多列数据组成。设计初衷是将Series的使用场景从一维拓展到多维。DataFrame既有行索引，也有列索引。

行索引：index 列索引：columns 值：values

学习DataFrame大致分两块：一块是DataFrame基础，另一块就是DataFrame相关的数据清洗。

DataFrame基础：

1. 创建
2. 属性
3. 索引
    - 行索引
    - 列索引
4. 切片
5. 运算

1. 创建

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

使用ndarray创建DataFrame

In [1]:

```
from pandas import DataFrame
import pandas as pd
import numpy as np
```

In [2]:

```
DataFrame(data=np.random.randint(1,23, size=(4, 4)))
```

Out[2]:

	0	1	2	3
0	10	4	4	11
1	10	15	3	17
2	11	5	18	11
3	10	12	1	13

In [3]:

```
DataFrame(data=np.random.randint(1,23, size=(4, 4)), index=["a", "b", "c", "d"], col
```

Out[3]:

	A	B	C	D
a	21	21	5	1
b	9	4	6	7
c	2	17	21	21
d	7	16	1	7

使用字典进行创建

传递一个字典来创建。DataFrame以字典的键作为每一【列】的名称，以字典的值（一个数组）作为每一列。

此外，DataFrame会自动加上每一行的索引。

使用字典创建的DataFrame后，则columns参数将不可被使用。

In [4]:

```
dic = {
    '张三':[77,88,99,90],
    '李四':[67,88,99,78]
}
df = DataFrame(data=dic,index=['语文','数学','英语','理综'])
df
```

Out[4]:

	张三	李四
语文	77	67
数学	88	88
英语	99	99
理综	90	78

使用列表套字典创建

这种方式还是比较常用的，因为mongo查询出来的数据一般都是列表套字典的形式

In [13]:

```
data = [{'name':'jack', 'age':12}, {'name':'tuple', 'nation':'china'}]
DataFrame(data)
```

Out[13]:

	age	name	nation
0	12.0	jack	NaN
1	NaN	tuple	china

2. 属性

常用属性：shape, size, columns, index, values

In [8]:

```
df.shape
```

Out[8]:

```
(4, 2)
```

In [9]:

```
df.size
```

Out[9]:

```
8
```

In [10]:

```
df.columns
```

Out[10]:

```
Index(['张三', '李四'], dtype='object')
```

In [11]:

```
df.index
```

Out[11]:

```
Index(['语文', '数学', '英语', '理综'], dtype='object')
```

In [12]:

```
df.values
```

Out[12]:

```
array([[77, 67],
       [88, 88],
       [99, 99],
       [90, 78]])
```

3. 索引

DataFrame的索引分为行索引和列索引, DataFrame默认索引就是列索引。

对列进行索引

- 通过类似字典的方式 df['q']
- 通过属性的方式 df.q

可以将DataFrame的列获取为一个Series。返回的Series拥有原DataFrame相同的行索引，且name属性也已经设置好了，就是相应的列名。

In [14]:

```
df = DataFrame(data=np.random.randint(1,23, size=(4, 4)), index=["a", "b", "c", "d"]
```

In [15]:

```
df.A
```

Out[15]:

```
a     3
b    11
c    13
d    18
Name: A, dtype: int64
```

In [16]:

```
df["B"]
```

Out[16]:

```
a    17
b    17
c     8
d     7
Name: B, dtype: int64
```

In [18]:

```
# 取多列, 返回一个DataFrame
df[["A", "C"]]
```

Out[18]:

	A	C
a	3	9
b	11	19
c	13	19
d	18	10

对行进行索引

记得之前numpy中默认是行索引, 对列进行索引使用的是最low的逗号。DataFrame默认是列索引, 对行索引必须使用特定的方法, 即loc和iloc.

对行进行索引

- 使用.loc[]加index来进行行索引
- 使用.iloc[]加整数来进行行索引

同样返回一个Series，index为原来的columns。

In [19]:

```
df
```

Out[19]:

	A	B	C	D
a	3	17	9	7
b	11	17	19	12
c	13	8	19	11
d	18	7	10	8

In [21]:

```
df.loc["a"]
```

Out[21]:

```
A     3
B    17
C     9
D     7
Name: a, dtype: int64
```

In [22]:

```
df.iloc[0]
```

Out[22]:

```
A     3
B    17
C     9
D     7
Name: a, dtype: int64
```

In [32]:

```
df.loc[["a", "c"]]
```

Out[32]:

	A	B	C	D
a	3	17	9	7
c	13	8	19	11

取出特定元素

对元素索引的方法

- 使用列索引
- 使用行索引(iloc[3,1] or loc['C','q']) 行索引在前，列索引在后

In [23]:

```
df
```

Out[23]:

	A	B	C	D
a	3	17	9	7
b	11	17	19	12
c	13	8	19	11
d	18	7	10	8

In [24]:

```
df["A"]["a"]
```

Out[24]:

```
3
```

In [25]:

```
df.loc["a", "A"]
```

Out[25]:

```
3
```

4. 切片

直接用中括号时：

1. 索引表示的是列索引
2. 切片表示的是行切片

换句话说, DataFrame默认的索引是列索引，但是默认的切片却是行切片

行切片

In [26]:

```
df
```

Out[26]:

	A	B	C	D
a	3	17	9	7
b	11	17	19	12
c	13	8	19	11
d	18	7	10	8

In [28]:

```
# 隐式索引的切片是左闭右开
df[0:2]
```

Out[28]:

	A	B	C	D
a	3	17	9	7
b	11	17	19	12

In [29]:

```
# 显式索引的切片是左闭右闭
df["a": "c"]
```

Out[29]:

	A	B	C	D
a	3	17	9	7
b	11	17	19	12
c	13	8	19	11

列切片

在之前的索引中, 默认是列索引，要想使用行索引，引出了loc方法。同样，对于切片来说，默认是行切片，要想使用列切片，也可以使用loc方法

In [30]:

```
df
```

Out[30]:

	A	B	C	D
a	3	17	9	7
b	11	17	19	12
c	13	8	19	11
d	18	7	10	8

In [31]:

```
df.loc[:, "A": "C"]
```

Out[31]:

	A	B	C
a	3	17	9
b	11	17	19
c	13	8	19
d	18	7	10

In [ ]:

```

```