Pandas的数据结构

导入pandas：
三剑客

In [1]:
```
import pandas as pd
import numpy as np
from pandas import Series,DataFrame
```

1.Series

Series是一种类似与一维数组的对象，由下面两个部分组成：

Series是一种类似与一维数组的对象，由下面两个部分组成：

Series是一种类似与一维数组的对象，由下面两个部分组成：

- values：一组数据（ndarray类型）
- index：相关的数据索引标签

1）Series的创建

两种创建方式：

(1) 由列表或numpy数组创建

默认索引为0到N-1的整数型索引

In [2]:
```
# 使用列表创建Series
Series(data=[1,2,3,4,5])
```

Out[2]:
```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

In [3]:
```
s = Series(data=[1,2,3,4,5],index=['a','b','c','d','e'])
s
```

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

In [9]:
```
# 使用 numpy 创建 Series  #使用name参数
Series(data=np.random.random(size=(10,)),name='np')
```

Out[9]:
```
0    0.110273
1    0.823596
2    0.878166
3    0.695236
4    0.222624
5    0.179289
6    0.151988
7    0.337229
8    0.671042
9    0.009094
Name: np, dtype: float64
```

- 还可以通过设置index参数指定索引

(2) 由字典创建:不能在使用index.但是依然存在默认索引

注意：数据源必须为一维数据

In [12]:
```
dic = {
    'math':100,
    'English':99,
}
s = Series(data=dic)
s
```

Out[12]:
```
math       100
English     99
dtype: int64
```

2）Series的索引和切片

可以使用中括号取单个索引（此时返回的是元素类型），或者中括号里一个列表取多个索引（此时返回的是一个Series类型）。

(1) 显式索引：

- 使用index中的元素作为索引值
- 使用s.loc[]（推荐）:注意，loc中括号中放置的一定是显示索引

注意，此时是闭区间

In [18]:
```
s
```

Out[18]:
```
math       100
English     99
dtype: int64
```

In [13]:
```
s.math
```

Out[13]:
```
100
```

In [16]:
```
s[0]
```

Out[16]:
```
100
```

In [17]:
```
s['math']
```

Out[17]: 100

In [19]:
```
s.loc['math']
```

Out[19]: 100

(2) 隐式索引:

- 使用整数作为索引值
- 使用.iloc[]（推荐）:iloc中的中括号中必须放置隐式索引

注意，此时是半开区间

In [22]:
```
s.iloc[0:2]
```

Out[22]:
```
math       100
English     99
dtype: int64
```

3）Series的基本概念

可以把Series看成一个定长的有序字典

向Series增加一行: 相当于给字典增加一组键值对

可以通过shape, size, index, values等得到series的属性

In [23]:
```
s.shape
```

Out[23]: (2,)

In [24]:
```
s.size
```

Out[24]:
```
2
```

In [25]:
```
s.values
```

Out[25]:
```
array([100,  99], dtype=int64)
```

In [27]:
```
s.index        #行索引
```

Out[27]:
```
Index(['math', 'English'], dtype='object')
```

可以使用s.head(),tail()分别查看前n个和后n个值

In [28]:
```
s.head(1)
```

Out[28]:
```
math    100
dtype: int64
```

In [29]:
```
s.tail(1)
```

Out[29]:
```
English    99
dtype: int64
```

对Series元素进行去重

In [30]:
```
s = Series([1,1,1,1,2,2,2,3,4,4,5,6,5,6,])
s
```

Out[30]:
```
0     1
1     1
2     1
3     1
4     2
5     2
6     2
7     3
8     4
9     4
10    5
11    6
12    5
13    6
dtype: int64
```

In [31]:
```
s.unique() #去重函数
```

Out[31]:
```
array([1, 2, 3, 4, 5, 6], dtype=int64)
```

当索引没有对应的值时，可能出现缺失数据显示NaN（not a number）的情况

- 使得两个Series进行相加

In [33]:
```
s1 = Series([1,2,3,4,5],index=['a','b','c','d','e'])
s2 = Series([1,2,3,4,5],index=['a','b','c','f','e'])
s1
```

Out[33]:
```
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

In [34]:
```
s2
```

Out[34]:
```
a    1
b    2
c    3
f    4
e    5
dtype: int64
```

数据清洗

(3) Series之间的运算

- 在运算中自动对齐不同索引的数据
- 如果索引不对应，则补NaN

In [37]:
```
s3 = s1+s2  #数据清洗
s3  # 2组数据 必须都有 才能相加 否则 对应 NaN
```

Out[37]:
```
a     2.0
b     4.0
c     6.0
d     NaN
e    10.0
f     NaN
dtype: float64
```

可以使用pd.isnull()，pd.notnull()，或s.isnull(),notnull()函数检测缺失数据

In [39]:
```
s3.isnull() #pd.isnull(s3)
```

Out[39]:
```
a    False
b    False
c    False
d     True
e    False
f     True
dtype: bool
```

In [43]:
```
s3.notnull() #pd.notnull()
```

Out[43]:
```
a     True
b     True
c     True
d    False
e     True
f    False
dtype: bool
```

In [51]:
```
s3
```

Out[51]:
```
a     2.0
b     4.0
c     6.0
d     NaN
e    10.0
f     NaN
dtype: float64
```

In [49]:
```
s3[[0,2,]]
```

Out[49]:
```
a    2.0
c    6.0
dtype: float64
```

In [52]:
```
s3[[True,False,True,False,False,False]]
```

Out[52]:
```
a    2.0
c    6.0
dtype: float64
```

In [53]:
```
s3.notnull()
```

Out[53]:
```
a     True
b     True
c     True
d    False
e     True
f    False
dtype: bool
```

In [55]:
```
s3[s3.notnull()]  # 清洗 NaN值
```

Out[55]:
```
a     2.0
b     4.0
c     6.0
e    10.0
dtype: float64
```

4）Series的运算

(1) + - * /

(2) add() sub() mul() div() : s1.add(s2,fill_value=0)

In [57]:
```
s1.add(s2)  # s1+s2 其他同理
```

Out[57]:
```
a     2.0
b     4.0
c     6.0
d     NaN
e    10.0
f     NaN
dtype: float64
```

(3) Series之间的运算

- 在运算中自动对齐不同索引的数据
- 如果索引不对应，则补NaN

1. 想一想Series运算和ndarray运算的规则有什么不同?
   - series有显示索引 numpy没有显示,遵循广播机制

In [ ]:
```

```