pandas模块中有2种数据结构经常被使用，一种是Series, 另一种就是DataFrame. 可以说，DataFrame由Series组成，所以在掌握DataFrame之前先学习Series.

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

从以下几个方面掌握Series:

1. 创建
2. 属性
3. 基本操作
   - 索引
   - 切片
   - 和python常用数据类型list、dict的转换
   - 检测缺失值
   - 迭代, 计数，布尔索引，排序, 去重

1. 创建

Series(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)

Series常用的创建方式有2种

由列表或numpy数组创建

In [3]:

```
from pandas import Series
import pandas as pd
import numpy as np
import string
```

In [5]:

```
# 指定显式索引，默认有隐式索引0，1，2，3
Series(np.arange(4), index=list(string.ascii_letters[:4]), name="s")
```

Out[5]:

```
a    0
b    1
c    2
d    3
Name: s, dtype: int64
```

由字典创建

不能在使用index.但是依然存在默认索引

In [6]:

```
data = {"name": "jack", "age": 23}
Series(data)
```

Out[6]:

```
name    jack
age       23
dtype: object
```

2. 属性

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

常用属性：index, values, name, size

In [39]:

```
s = Series(data, name="boo")
```

In [40]:

```
s.index
```

Out[40]:

```
Index(['name', 'age'], dtype='object')
```

In [41]:

```
s.values
```

Out[41]:

```
array(['jack', 23], dtype=object)
```

In [42]:

```
s.name
```

Out[42]:

```
'boo'
```

In [43]:

```
s.size
```

Out[43]:

```
2
```

3. 基本操作

索引

不同于DataFrame，Series只有一个行索引，所以Series取索引也是比较简单的

In [30]:

```
s
```

Out[30]:

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

In [18]:

```
s[0]
```

Out[18]:

```
'jack'
```

In [31]:

```
# 取多个索引值
s[['b', 'a']]
```

Out[31]:

```
b    1
a    2
dtype: int64
```

切片

值得说明的是，Series的切片，对于显式索引遵循左闭右闭原则(所以如果右索引不存在会报错)，对于隐式索引遵循左闭右开原则.(显式索引指的是通过index参数指定的索引, 隐式索引指的是Series默认给的0,1,2等索引).

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

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 [32]:

```
s = Series({"b": 1, "a": 2, "d": 3})
```

In [33]:

```
s[0: 10]
```

Out[33]:

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

In [34]:

```
# 先取b元素，然后再找到d元素， 然后结合两者之间的元素
s["b": "d"]
```

Out[34]:

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

In [35]:

```
s.iloc[0:3]
```

Out[35]:

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

In [36]:

```
s.loc['a': 'd']
```

Out[36]:

```
a    2
d    3
dtype: int64
```

In [37]:

```
s.loc["a"]
```

Out[37]:

```
2
```

In [38]:

```
s.iloc[1]
```

Out[38]:

```
2
```

和python常用数据类型list、dict的转换

In [44]:

```
s
```

Out[44]:

```
name    jack
age       23
Name: boo, dtype: object
```

In [46]:

```
s.to_dict()
```

Out[46]:

```
{'name': 'jack', 'age': 23}
```

In [47]:

```
# 这里的为啥不是和to_dict一样使用to_list? 搞不懂, 看不起dict?
s.tolist()
```

Out[47]:

```
['jack', 23]
```

检测缺失值

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

In [48]:

```
s = Series(data=range(12))
```

In [49]:

```
s
```

Out[49]:

```
0       0
1       1
2       2
3       3
4       4
5       5
6       6
7       7
8       8
9       9
10     10
11     11
dtype: int64
```

In [50]:

```
s[[0, 3, 5, 9]] = None
```

In [51]:

```
s
```

Out[51]:

```
0      NaN
1      1.0
2      2.0
3      NaN
4      4.0
5      NaN
6      6.0
7      7.0
8      8.0
9      NaN
10    10.0
11    11.0
dtype: float64
```

In [52]:

```
s.isnull()
```

Out[52]:

```
0      True
1     False
2     False
3      True
4     False
5      True
6     False
7     False
8     False
9      True
10    False
11    False
dtype: bool
```

In [54]:

```
s.notnull()
```

Out[54]:

```
0     False
1      True
2      True
3     False
4      True
5     False
6      True
7      True
8      True
9     False
10     True
11     True
dtype: bool
```

In [55]:

```
# 还记得在numpy中的索引机制吗？在这里同样使用，使用isnull() 或notnull() 检测得到一列bool值，可
s[s.notnull()]
```

Out[55]:

```
1      1.0
2      2.0
4      4.0
6      6.0
7      7.0
8      8.0
10    10.0
11    11.0
dtype: float64
```

迭代, 计数， 排序, 去重

In [62]:

```
s = s[s.notnull()]
import random
index = list(string.ascii_letters[:s.size])
random.shuffle(index)
s.index = index
```

In [58]:

```
for i in s:
    print(i)
```

```
1.0
2.0
4.0
6.0
7.0
8.0
10.0
11.0
```

In [59]:

```
# 等同于s.size
s.count()
```

Out[59]:

```
8
```

In [60]:

```
s.size
```

Out[60]:

```
8
```

In [63]:

```
s.index
```

Out[63]:

```
Index(['e', 'f', 'g', 'c', 'b', 'h', 'a', 'd'], dtype='object')
```

In [65]:

```
s
```

Out[65]:

```
e     1.0
f     2.0
g     4.0
c     6.0
b     7.0
h     8.0
a    10.0
d    11.0
dtype: float64
```

In [64]:

```
# Series的排序有2种，一种是根据索引排序，另一种是根据值排序
s.sort_index()
```

Out[64]:

```
a    10.0
b     7.0
c     6.0
d    11.0
e     1.0
f     2.0
g     4.0
h     8.0
dtype: float64
```

In [66]:

```
s.sort_values()
```

Out[66]:

```
e     1.0
f     2.0
g     4.0
c     6.0
b     7.0
h     8.0
a    10.0
d    11.0
dtype: float64
```

In [67]:

```
# 因为去重是根据value来去重的，value 是 array类型，所以去重得到的结果也是array类型
s.unique()
```

Out[67]:

```
array([ 1.,  2.,  4.,  6.,  7.,  8., 10., 11.])
```