处理丢失数据

有两种丢失数据:

- None
- np.nan(NaN)

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

1. None

None是Python自带的，其类型为python object。因此，None不能参与到任何计算中。

In [2]:
```
# 查看None的数据类型
type(None)
```

Out[2]:
```
NoneType
```

In [3]:
```
type(np.NaN)
```

Out[3]:
```
float
```

2. np.nan（NaN）

np.nan 是浮点类型,能参与到计算中. 但计算的结果总是NaN

In [4]:
```
# 查看 np.nan 的数据类型
type(np.nan)
```

Out[4]:
```
float
```

3. pandas中的None与NaN

1) pandas中None与np.nan都视作np.nan

创建DataFrame

In [5]:
```
np.random.seed(2) #时间种子 random不再随机
df = DataFrame(data=np.random.randint(0,100,size=(7,6)))
df
```

Out[5]:

	0	1	2	3	4	5
0	40	15	72	22	43	82
1	75	7	34	49	95	75
2	85	47	63	31	90	20
3	37	39	67	4	42	51
4	38	33	58	67	69	88
5	68	46	70	95	83	31
6	66	80	52	76	50	4

In [6]:
```
# 将某些数组元素赋值为 nan
df.iloc[1,2] = None
df.iloc[2,1] = np.nan
df.iloc[3,4] = None
df
```

Out[6]:

	0	1	2	3	4	5
0	40	15.0	72.0	22	43.0	82
1	75	7.0	NaN	49	95.0	75
2	85	NaN	63.0	31	90.0	20
3	37	39.0	67.0	4	NaN	51
4	38	33.0	58.0	67	69.0	88
5	68	46.0	70.0	95	83.0	31
6	66	80.0	52.0	76	50.0	4

2) pandas处理空值操作

- isnull()
- notnull()
- dropna()：过滤丢失数据
- fillna()：填充丢失数据

In [7]:
```
df.isnull()
```

Out[7]:

	0	1	2	3	4	5
0	False	False	False	False	False	False
1	False	False	True	False	False	False
2	False	True	False	False	False	False
3	False	False	False	False	True	False
4	False	False	False	False	False	False
5	False	False	False	False	False	False
6	False	False	False	False	False	False

In [8]:
```
b = [True, False, False, False, True, True, True]
```

In [9]:
```
df.loc[b] #行索引
```

Out[9]:

	0	1	2	3	4	5
0	40	15.0	72.0	22	43.0	82
4	38	33.0	58.0	67	69.0	88
5	68	46.0	70.0	95	83.0	31
6	66	80.0	52.0	76	50.0	4

- df.notnull/isnull().any()/all()

In [10]:
```
df.isnull()
```

Out[10]:

	0	1	2	3	4	5
0	False	False	False	False	False	False
1	False	False	True	False	False	False
2	False	True	False	False	False	False
3	False	False	False	False	True	False
4	False	False	False	False	False	False
5	False	False	False	False	False	False
6	False	False	False	False	False	False

In [11]:
```
df.notnull().all(axis=0) #整列没有空值
```

Out[11]:
```
0     True
1    False
2    False
3     True
4    False
5     True
dtype: bool
```

In [12]:
```
df.isnull().any(axis=0) #整列有空值    0>>>每列
```

Out[12]:
```
0    False
1     True
2     True
3    False
4     True
5    False
dtype: bool
```

In [13]:
```
df.isnull().all(axis=0) # 每列所有数值都为空
```

Out[13]:
```
0    False
1    False
2    False
3    False
4    False
5    False
dtype: bool
```

In [14]:
```
# notnull() ===> all() 没有空值的行 要保留   (作为过滤条件)
# isnull() ==> any()  有空值的行  要删除     (作为删除的条件)
```

In [15]:
```
df
```

Out[15]:

	0	1	2	3	4	5
0	40	15.0	72.0	22	43.0	82
1	75	7.0	NaN	49	95.0	75
2	85	NaN	63.0	31	90.0	20
3	37	39.0	67.0	4	NaN	51
4	38	33.0	58.0	67	69.0	88
5	68	46.0	70.0	95	83.0	31
6	66	80.0	52.0	76	50.0	4

In [16]:
```
df[df.notnull().all(axis=1)]  # 切取 没有空的行
```

Out[16]:

	0	1	2	3	4	5
0	40	15.0	72.0	22	43.0	82
4	38	33.0	58.0	67	69.0	88
5	68	46.0	70.0	95	83.0	31
6	66	80.0	52.0	76	50.0	4

In [17]:
```
df[df.isnull().any(axis=1)]  # 切取 有空值的行
```

Out[17]:

	0	1	2	3	4	5
1	75	7.0	NaN	49	95.0	75
2	85	NaN	63.0	31	90.0	20
3	37	39.0	67.0	4	NaN	51

In [18]:
```
 # inplace 写回原数据  默认为false
df.dropna(axis=0,inplace=False)  # 删除 有空值的行 (在drop系列的函数中 axis=0(行) 1(列) ====  正常  0是列  1是列)
```

Out[18]:

	0	1	2	3	4	5
0	40	15.0	72.0	22	43.0	82
4	38	33.0	58.0	67	69.0	88
5	68	46.0	70.0	95	83.0	31
6	66	80.0	52.0	76	50.0	4

(3) 填充函数 Series/DataFrame

- `fillna()` :value和method参数

In [19]:
```
df
```

Out[19]:

	0	1	2	3	4	5
0	40	15.0	72.0	22	43.0	82
1	75	7.0	NaN	49	95.0	75
2	85	NaN	63.0	31	90.0	20
3	37	39.0	67.0	4	NaN	51
4	38	33.0	58.0	67	69.0	88
5	68	46.0	70.0	95	83.0	31
6	66	80.0	52.0	76	50.0	4

In [20]:
```
df.fillna(method='ffill',axis=1)# 看空值 行前边的数   正常 1 是行
```

Out[20]:

	0	1	2	3	4	5
0	40.0	15.0	72.0	22.0	43.0	82.0
1	75.0	7.0	7.0	49.0	95.0	75.0
2	85.0	85.0	63.0	31.0	90.0	20.0
3	37.0	39.0	67.0	4.0	4.0	51.0
4	38.0	33.0	58.0	67.0	69.0	88.0
5	68.0	46.0	70.0	95.0	83.0	31.0
6	66.0	80.0	52.0	76.0	50.0	4.0

In [21]:
```
df.fillna(method='bfill',axis=1) # 看空值 行后边的数  正常 1 是行
```

Out[21]:

	0	1	2	3	4	5
0	40.0	15.0	72.0	22.0	43.0	82.0
1	75.0	7.0	49.0	49.0	95.0	75.0
2	85.0	63.0	63.0	31.0	90.0	20.0
3	37.0	39.0	67.0	4.0	51.0	51.0
4	38.0	33.0	58.0	67.0	69.0	88.0
5	68.0	46.0	70.0	95.0	83.0	31.0
6	66.0	80.0	52.0	76.0	50.0	4.0

method 控制填充的方式 bfill ffill

In [ ]:
```
```