24、利用pandas处理数据（2）

1、Series和DataFrame

在pandas中提供了Series和DataFrame两种数据结构，Series适合处理一维数据，而表格数据往往是二维的，更适合用DataFrame来进行处理。DataFrame可以看作是多个相同索引的Series的集合，其中一个Series为一列。因此很多Series的操作也可以用到DataFrame中

2、DataFrame的构建

构建DataFrame有两种常用的方法。

1）通过字典构建DataFrame

```
import pandas as pd
#定义字典data，每个键值是相等长度的列表
data={'姓名':['王静怡','张佳妮','李臣武'],'性别':['女','女','男'],'借阅次数':[28,56,37]}
#根据字典data来创建DataFrame,并设定数据列顺序
df1=pd.DataFrame(data,columns=['性别','姓名','借阅次数'])
print(df1)
```

```
  性别   姓名  借阅次数
0  女  王静怡    28
1  女  张佳妮    56
2  男  李臣武    37
```

2）通过读取csv或读取excel文件来构建DataFrame

```
import pandas as pd
df2=pd.read_csv("name.csv")
df2
```

	性别	姓名	借阅次数
0	女	王静怡	28
1	女	张佳妮	56
2	男	李臣武	37

```
df2=pd.read_excel("name.xlsx",sheet_name='Sheet1')
df2
```

	性别	姓名	借阅次数
0	女	王静怡	28
1	女	张佳妮	56
2	男	李臣武	37

3、DataFrame常用属性

1）columns属性：

```
for i in df1.columns:
    print(i)
```

```
性别
姓名
借阅次数
```

2）values属性：

```
for i in df1.values:
    print(i)
```

```
['女' '王静怡' 28]
['女' '张佳妮' 56]
['男' '李臣武' 37]
```

3）T属性，得到转置（行列变换）后的结果：

```
df1.T
```

	0	1	2
性别	女	女	男
姓名	王静怡	张佳妮	李臣武
借阅次数	28	56	37

4、用DataFrame处理橡皮筋实验数据

1）读取excel数据到data对象，并通过head方法和tail方法查看前5行数据和末尾5行数据：

```
import pandas as pd
data=pd.read_excel("橡皮筋实验.xlsx",'Sheet1')
data.columns=['length','force','k'] # 重命名列
print(data.head())
print(data.tail())
```

```
   length  force   k
0       1   0.50 NaN
1       2   0.95 NaN
2       3   1.31 NaN
3       4   1.60 NaN
4       5   1.84 NaN
    length  force   k
15      15   4.28 NaN
16      16   4.70 NaN
17      17   5.33 NaN
18      18   6.10 NaN
19      19   0.00 NaN
```

2）找出force为0的数据并删除：

```
print("force小于等于0的数据行为：")
print(data[data['force']<=0])
data=data.drop(data[data['force']==0].index)
print("新的数据为：")
print(data)
```

```
force小于等于0的列为：
Empty DataFrame
Columns: [length, force, k]
Index: []
新的数据为：
    length  force   k
0        1   0.50 NaN
1        2   0.95 NaN
2        3   1.31 NaN
3        4   1.60 NaN
4        5   1.84 NaN
5        6   2.05 NaN
6        7   2.26 NaN
7        8   2.47 NaN
8        9   2.68 NaN
9       10   2.89 NaN
10      10   2.89 NaN
11      11   3.11 NaN
12      12   3.36 NaN
13      13   3.65 NaN
14      14   3.95 NaN
15      15   4.28 NaN
16      16   4.70 NaN
17      17   5.33 NaN
18      18   6.10 NaN
```

3）找出length重复的数据并删除：

```
print("force小于等于0的列为：")
print(data[data['length'].duplicated()])
data=data.drop(data[data['length'].duplicated()].index)
data=data.reset_index(drop=True) #重新设置索引号
print("新的数据为：")
print(data)
```

```
force小于等于0的列为：
Empty DataFrame
Columns: [length, force, k]
Index: []
新的数据为：
    length  force   k
0        1   0.50 NaN
1        2   0.95 NaN
2        3   1.31 NaN
3        4   1.60 NaN
4        5   1.84 NaN
5        6   2.05 NaN
6        7   2.26 NaN
7        8   2.47 NaN
8        9   2.68 NaN
9       10   2.89 NaN
10      11   3.11 NaN
11      12   3.36 NaN
12      13   3.65 NaN
13      14   3.95 NaN
14      15   4.28 NaN
15      16   4.70 NaN
16      17   5.33 NaN
17      18   6.10 NaN
```

4）计算劲度系数k：

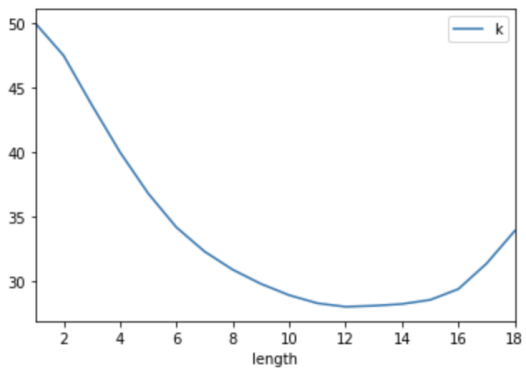
```
data['k']=data['force']/data['length']*100
print(data)
```

```
    length  force          k
0        1   0.50  50.000000
1        2   0.95  47.500000
2        3   1.31  43.666667
3        4   1.60  40.000000
4        5   1.84  36.800000
5        6   2.05  34.166667
6        7   2.26  32.285714
7        8   2.47  30.875000
8        9   2.68  29.777778
9       10   2.89  28.900000
10      11   3.11  28.272727
11      12   3.36  28.000000
12      13   3.65  28.076923
13      14   3.95  28.214286
14      15   4.28  28.533333
15      16   4.70  29.375000
16      17   5.33  31.352941
17      18   6.10  33.888889
```

5）绘制劲度系数折线图：

```
data.plot(x='length',y='k')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcdc14f88d0>
```



tips）如果要按照劲度系数降序排序怎么办：

```
data=data.sort_values(['k'],ascending=False)
print(data)
```

```
    length  force          k
0        1   0.50  50.000000
1        2   0.95  47.500000
2        3   1.31  43.666667
3        4   1.60  40.000000
4        5   1.84  36.800000
5        6   2.05  34.166667
17      18   6.10  33.888889
6        7   2.26  32.285714
16      17   5.33  31.352941
7        8   2.47  30.875000
8        9   2.68  29.777778
15      16   4.70  29.375000
9       10   2.89  28.900000
14      15   4.28  28.533333
10      11   3.11  28.272727
13      14   3.95  28.214286
12      13   3.65  28.076923
11      12   3.36  28.000000
```