pandas（2）

函数运用

In [9]:
```
import numpy as np
import pandas as pd
df1 = pd.DataFrame(np.arange(9).reshape(3,3),index=['宋江','李逵','武松'],colum
print(df1)
```

```
    语文  数学  英语
宋江   0   1   2
李逵   3   4   5
武松   6   7   8
```

In [13]:
```
#使用numpy的通用函数
#获取列的最大数
np.max(df1)
#np.max(df1,axis=0)
```

Out[13]:
```
语文    6
数学    7
英语    8
dtype: int64
```

In [11]:
```
#获取行的最大数
np.max(df1,axis=1)
```

Out[11]:
```
宋江    2
李逵    5
武松    8
dtype: int64
```

数学和统计方法

方法	描述
sum	对数组的所有或一个轴向上的元素求和。零长度的数组的和为灵。
mean	算术平均值。灵长度的数组的均值为NaN。
std, var	标准差和方差，有可选的调整自由度（默认值为n）。
min, max	最大值和最小值
argmin, argmax	索引最小和最大元素。
cumsum	所有元素的累计和
cumprod	所有元素的累计积

In [14]:
```
#使用自定义函数
```

```
def text(df):
    return df.max()-df.min()

df1.apply(text)
#pandas 的 apply() 函数可以作用于 Series 或者整个 DataFrame,
#功能是自动遍历整个 Series 或者 DataFrame, 对每一个元素运行指定的函数。
```

Out[14]:
```
语文    6
数学    6
英语    6
dtype: int64
```

In [15]:
```
def text(df):
    return df.max()-df.min()

df1.apply(text,axis=1)
```

Out[15]:
```
宋江    2
李逵    2
武松    2
dtype: int64
```

In [ ]:
```
df1.apply(lambda x:x.max() - x.min())
```

In [16]:
```
def total(df1):
    df1['总分'] = df1['语文'] + df1['数学'] + df1['英语']
    return df1
df1.apply(total,axis = 1)
```

Out[16]:

	语文	数学	英语	总分
宋江	0	1	2	3
李逵	3	4	5	12
武松	6	7	8	21

In [17]:
```
def total(df1):
    df1['总分'] = df1['语文'] + df1['数学'] + df1['英语']
    return df1
df1.apply(total,axis = 1)
```

Out[17]:

	语文	数学	英语	总分
宋江	0	1	2	3
李逵	3	4	5	12
武松	6	7	8	21

排序

In [18]:
```
df1
```

Out[18]:

	语文	数学	英语
宋江	0	1	2

	语文	数学	英语
李逵	3	4	5
武松	6	7	8

In [19]:
```
#默认是按照名字的Unicode排序的，没有实际意义
```

In [21]:
```
#转化为拼音，用拼音首字母排序
df1.rename({'宋江':'songjiang','李逵':'likui','武松':'wusong'},axis='index')
```

Out[21]:

	语文	数学	英语
songjiang	0	1	2
likui	3	4	5
wusong	6	7	8

In [22]:
```
df1.rename({'宋江':'songjiang','李逵':'likui','武松':'wusong'},axis=0)
```

Out[22]:

	语文	数学	英语
songjiang	0	1	2
likui	3	4	5
wusong	6	7	8

In [23]:
```
df1
```

Out[23]:

	语文	数学	英语
宋江	0	1	2
李逵	3	4	5
武松	6	7	8

In [24]:
```
df1.rename({'宋江':'songjiang','李逵':'likui','武松':'wusong'},axis=0,inplace=T
```

In [25]:
```
df1
```

Out[25]:

	语文	数学	英语
songjiang	0	1	2
likui	3	4	5
wusong	6	7	8

In [26]:
```
df1.sort_index()  #按行，根据拼音排序，默认升序排列
```

Out[26]:

	语文	数学	英语

	语文	数学	英语
likui	3	4	5
songjiang	0	1	2
wusong	6	7	8

In [27]:
```
df1.sort_index(ascending=False)   #按行，根据拼音排序，降序排列
```

Out[27]:

	语文	数学	英语
wusong	6	7	8
songjiang	0	1	2
likui	3	4	5

In [28]:
```
#按列排序
#重命名列
df1.rename({'语文':'yuwen','数学':'shuxue','英语':'yingyu'},axis=1,inplace=True
```

In [29]:
```
df1
```

Out[29]:

	yuwen	shuxue	yingyu
songjiang	0	1	2
likui	3	4	5
wusong	6	7	8

In [30]:
```
df1.sort_index(axis=1) #按行升序排序
```

Out[30]:

	shuxue	yingyu	yuwen
songjiang	1	2	0
likui	4	5	3
wusong	7	8	6

In [ ]:
```
df1.sort_index(axis=1) #按行降序排序
```

In [31]:
```
#根据值进行排序
df1.sort_values(by=['shuxue'])
```

Out[31]:

	yuwen	shuxue	yingyu
songjiang	0	1	2
likui	3	4	5
wusong	6	7	8

In [32]:
```
df1.sort_values(by=['shuxue','yingyu'])#指定第二个排序方式
```

Out[32]:

	yuwen	shuxue	yingyu
songjiang	0	1	2
likui	3	4	5
wusong	6	7	8

描述和汇总统计

函数	描述
count	统计非空值数量
sum	汇总值
mean	平均值
mad	平均绝对偏差
median	算数中位数
min	最小值
max	最大值
mode	众数
abs	绝对值
prod	乘积
std	贝塞尔校正的样本标准偏差
var	无偏方差
sem	平均值的标准误差
skew	样本偏度 (第三阶)
kurt	样本峰度 (第四阶)

In [33]:
```
df1
```

Out[33]:

	yuwen	shuxue	yingyu
songjiang	0	1	2
likui	3	4	5
wusong	6	7	8

In [34]:
```
df1.count()
```

Out[34]:
```
yuwen      3
shuxue     3
yingyu     3
dtype: int64
```

In [35]:
```
df1.max()
```

Out[35]:
```
yuwen      6
shuxue     7
yingyu     8
dtype: int64
```

In [36]:
```
df1.sum()  #默认是列的求和
```

Out[36]:
```
yuwen      9
shuxue    12
yingyu    15
dtype: int64
```

In [37]:
```
df1.sum(axis = 1)  #求行的和
```

Out[37]:
```
songjiang     3
likui        12
wusong       21
dtype: int64
```

In [38]:
```
df1.describe()  #总体描述
```

Out[38]:

	yuwen	shuxue	yingyu
count	3.0	3.0	3.0
mean	3.0	4.0	5.0
std	3.0	3.0	3.0
min	0.0	1.0	2.0
25%	1.5	2.5	3.5
50%	3.0	4.0	5.0
75%	4.5	5.5	6.5
max	6.0	7.0	8.0

In [39]:
```
#汇总文字
data = {'name':['andy','jacky'],'age':['18','20']}
df = pd.DataFrame(data)
print(df)
print(df.describe())
```

```
    name age
0   andy  18
1  jacky  20
        name age
count      2   2
unique     2   2
top     jacky  18
freq       1   1
```

In [40]:
```
df1['yingyu'] = [np.nan,5,8]
df1
```

Out[40]:

	yuwen	shuxue	yingyu
songjiang	0	1	NaN
likui	3	4	5.0
wusong	6	7	8.0

In [41]:
```
df1.sum()  #默认会忽略掉缺失值
```

Out[41]:
```
yuwen      9.0
shuxue    12.0
yingyu    13.0
dtype: float64
```

处理缺失值

In [42]:
```
df1
```

Out[42]:

	yuwen	shuxue	yingyu
songjiang	0	1	NaN
likui	3	4	5.0
wusong	6	7	8.0

In [43]:
```
#需要先找到缺失值，然后对缺失值进行处理，最常用的处理方式有两种：删除和填充
```

In [44]:
```
#查找缺失值
df1.isnull()    #返回为true则为缺失值
```

Out[44]:

	yuwen	shuxue	yingyu
songjiang	False	False	True
likui	False	False	False
wusong	False	False	False

In [45]:
```
df1.notnull()
```

Out[45]:

	yuwen	shuxue	yingyu
songjiang	True	True	False
likui	True	True	True
wusong	True	True	True

In [47]:
```
df1.info()  #对数据的概括
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3 entries, songjiang to wusong
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   yuwen   3 non-null      int64  
 1   shuxue  3 non-null      int64  
 2   yingyu  2 non-null      float64
dtypes: float64(1), int64(2)
memory usage: 204.0+ bytes
```

In [48]:
```
#处理方法1：删除
df1.dropna()  #把包含缺失值的整行删除
```

Out[48]:

	yuwen	shuxue	yingyu

	yuwen	shuxue	yingyu
likui	3	4	5.0
wusong	6	7	8.0

In [49]:
```
df1.dropna(axis = 1)
```

Out[49]:

	yuwen	shuxue
songjiang	0	1
likui	3	4
wusong	6	7

In [53]:
```
df1
```

Out[53]:

	yuwen	shuxue	yingyu
songjiang	0	1	NaN
likui	3	4	5.0
wusong	6	7	8.0

In [55]:
```
df1.loc['wuyong']=[np.nan,np.nan,np.nan]
df1
```

Out[55]:

	yuwen	shuxue	yingyu
songjiang	0.0	1.0	NaN
likui	3.0	4.0	5.0
wusong	6.0	7.0	8.0
wuyong	NaN	NaN	NaN

In [57]:
```
df1.dropna(how= 'all')   #删除方式为所有都为nan的
```

Out[57]:

	yuwen	shuxue	yingyu
songjiang	0.0	1.0	NaN
likui	3.0	4.0	5.0
wusong	6.0	7.0	8.0

In [58]:
```
df1
```

Out[58]:

	yuwen	shuxue	yingyu
songjiang	0.0	1.0	NaN
likui	3.0	4.0	5.0

	yuwen	shuxue	yingyu
wusong	6.0	7.0	8.0
wuyong	NaN	NaN	NaN

In [61]:
```
df1.dropna(how= 'all',inplace = True)   #删除方式为所有都为nan的,并且替换源数据
df1
```

Out[61]:

	yuwen	shuxue	yingyu
songjiang	0.0	1.0	NaN
likui	3.0	4.0	5.0
wusong	6.0	7.0	8.0

In [62]:
```
#处理方法2：填充
df1.fillna(0)  #填充0的误差太大
```

Out[62]:

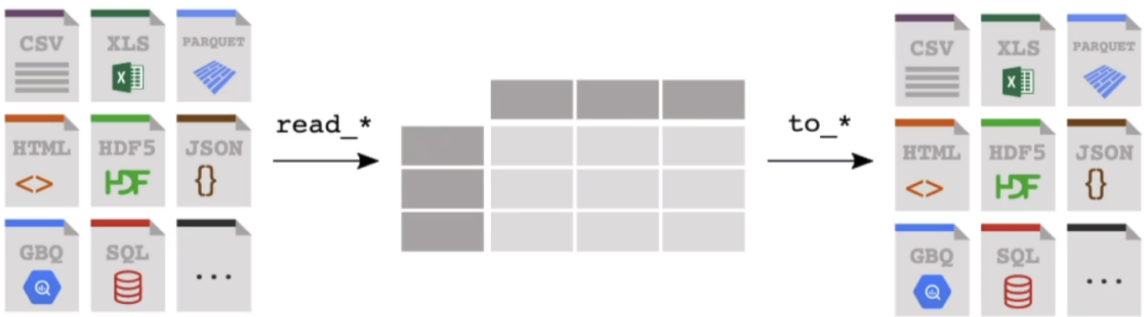
	yuwen	shuxue	yingyu
songjiang	0.0	1.0	0.0
likui	3.0	4.0	5.0
wusong	6.0	7.0	8.0

In [63]:
```
df1.fillna(df1.mean())
```

Out[63]:

	yuwen	shuxue	yingyu
songjiang	0.0	1.0	6.5
likui	3.0	4.0	5.0
wusong	6.0	7.0	8.0

读取和存储



In [65]:
```
movies = pd.read_csv('data/豆瓣TOP250.txt')
movies
```

Out[65]:

	肖申克的救赎 The Shawshank Redemption|9.6|(1005725人评价)
0	这个杀手不太冷 Léon|9.4|(945608人评价)
1	阿甘正传 Forrest Gump|9.4|(804207人评价)
2	霸王别姬|9.5|(731356人评价)
3	美丽人生 La vita è bella|9.5|(469717人评价)
4	海上钢琴师 La leggenda del pianista sull'oceano|9.2...
...	...
244	你看起来好像很好吃 おまえうまそうだな|8.8|(166414人评价)
245	那些年，我们一起追的女孩 那些年，我們一起追的女孩|8.1|(522910人评价)
246	再见我们的幼儿园 さよならぼくたちのようちえん|8.7|(94329人评价)
247	就是这样 This Is It|8.7|(77153人评价)
248	被解救的姜戈 Django Unchained|8.5|(268718人评价)

249 rows × 1 columns

In [66]:
```
print(movies.columns)
```

```
Index(['肖申克的救赎 The Shawshank Redemption|9.6|(1005725人评价)'], dtype='object')
```

In [67]:
```
#读取时添加分隔符
movies = pd.read_csv('data/豆瓣TOP250.txt',sep='|')
movies
```

Out[67]:

	肖申克的救赎 The Shawshank Redemption	9.6	(1005725人评价)
0	这个杀手不太冷 Léon	9.4	(945608人评价)
1	阿甘正传 Forrest Gump	9.4	(804207人评价)
2	霸王别姬	9.5	(731356人评价)
3	美丽人生 La vita è bella	9.5	(469717人评价)
4	海上钢琴师 La leggenda del pianista sull'oceano	9.2	(651756人评价)
...	...	...	...
244	你看起来好像很好吃 おまえうまそうだな	8.8	(166414人评价)
245	那些年，我们一起追的女孩 那些年，我們一起追的女孩	8.1	(522910人评价)
246	再见我们的幼儿园 さよならぼくたちのようちえん	8.7	(94329人评价)
247	就是这样 This Is It	8.7	(77153人评价)
248	被解救的姜戈 Django Unchained	8.5	(268718人评价)

249 rows × 3 columns

In [69]:
```
#把第一行数据行不作为索引
movies = pd.read_csv('data/豆瓣TOP250.txt',sep='|',header=None)
movies
```

Out[69]:

	0	1	2
0	肖申克的救赎 The Shawshank Redemption	9.6	(1005725人评价)
1	这个杀手不太冷 Léon	9.4	(945608人评价)
2	阿甘正传 Forrest Gump	9.4	(804207人评价)
3	霸王别姬	9.5	(731356人评价)
4	美丽人生 La vita è bella	9.5	(469717人评价)
...	...	...	...
245	你看起来好像很好吃 おまえうまそうだな	8.8	(166414人评价)
246	那些年，我们一起追的女孩 那些年，我們一起追的女孩	8.1	(522910人评价)
247	再见我们的幼儿园 さよならぼくたちのようちえん	8.7	(94329人评价)
248	就是这样 This Is It	8.7	(77153人评价)
249	被解救的姜戈 Django Unchained	8.5	(268718人评价)

250 rows × 3 columns

In [70]:

```
#添加自定义索引方法1
movies = pd.read_csv('data/豆瓣TOP250.txt',sep='|',header=None)
movies.columns=['电影名','评分','评论数']
movies
```

Out[70]:

	电影名	评分	评论数
0	肖申克的救赎 The Shawshank Redemption	9.6	(1005725人评价)
1	这个杀手不太冷 Léon	9.4	(945608人评价)
2	阿甘正传 Forrest Gump	9.4	(804207人评价)
3	霸王别姬	9.5	(731356人评价)
4	美丽人生 La vita è bella	9.5	(469717人评价)
...	...	...	...
245	你看起来好像很好吃 おまえうまそうだな	8.8	(166414人评价)
246	那些年，我们一起追的女孩 那些年，我們一起追的女孩	8.1	(522910人评价)
247	再见我们的幼儿园 さよならぼくたちのようちえん	8.7	(94329人评价)
248	就是这样 This Is It	8.7	(77153人评价)
249	被解救的姜戈 Django Unchained	8.5	(268718人评价)

250 rows × 3 columns

In [73]:

```
#添加自定义索引方法2
movies = pd.read_csv('data/豆瓣TOP250.txt',sep='|',header=None,names =['电影名'
movies
```

Out[73]:

	电影名	评分	评论数
0	肖申克的救赎 The Shawshank Redemption	9.6	(1005725人评价)

	电影名	评分	评论数
1	这个杀手不太冷 Léon	9.4	(945608人评价)
2	阿甘正传 Forrest Gump	9.4	(804207人评价)
3	霸王别姬	9.5	(731356人评价)
4	美丽人生 La vita è bella	9.5	(469717人评价)
...	...	...	...
245	你看起来好像很好吃 おまえうまそうだな	8.8	(166414人评价)
246	那些年，我们一起追的女孩 那些年，我們一起追的女孩	8.1	(522910人评价)
247	再见我们的幼儿园 さよならぼくたちのようちえん	8.7	(94329人评价)
248	就是这样 This Is It	8.7	(77153人评价)
249	被解救的姜戈 Django Unchained	8.5	(268718人评价)

250 rows × 3 columns

In [74]:
```
#文件存储为txt
movies.to_csv('豆瓣电影排名前250位.txt')
```

In [75]:
```
#去除列索引及行索引
movies.to_csv('豆瓣电影排名前250位.txt',header=None,index=False)
```

In [76]:
```
#保存为csv格式文件
movies.to_csv('豆瓣电影排名前250位.csv',index=False)
```

In [78]:
```
#读取csv文件
data = pd.read_csv('豆瓣电影排名前250位.csv',nrows=10,na_values=0,skiprows=10)
#nrows表示读取前几行，na_values表示如果数据中有NAN空值用什么来替换，skiprows为跳过前几行
data
```

Out[78]:

	三傻大闹宝莱坞 3 Idiots	9.2	(758367人评价)
0	泰坦尼克号 Titanic	9.2	(743837人评价)
1	盗梦空间 Inception	9.3	(844576人评价)
2	放牛班的春天 Les choristes	9.2	(506961人评价)
3	龙猫 となりのトトロ	9.1	(467815人评价)
4	忠犬八公的故事 Hachi: A Dog's Tale	9.2	(522090人评价)
5	教父 The Godfather	9.2	(381041人评价)
6	大话西游之大圣娶亲 西遊記大結局之仙履奇緣	9.2	(554001人评价)
7	乱世佳人 Gone with the Wind	9.2	(296621人评价)
8	天堂电影院 Nuovo Cinema Paradiso	9.1	(335287人评价)
9	当幸福来敲门 The Pursuit of Happyness	8.9	(599730人评价)

In [ ]: