Table of Contents

In [1]:
```
#全部行都能输出
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
import matplotlib as plt

# 解决坐标轴刻度负号乱码
plt.rcParams['axes.unicode_minus'] = False

# 解决中文乱码问题
plt.rcParams['font.sans-serif'] = ['Simhei']
```

1 查看基本信息

一般拿到数据，我们第一步需要做的是了解下数据的整体情况，可以使用 info 方法来查看。

1.1 df.info()

In [3]:
```
import pandas as pd,numpy as np
user_infor=pd.read_csv("new_infor.csv",index_col="索引")

user_infor.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8 entries, 0 to 7
Data columns (total 5 columns):
Hero Name     8 non-null object
Age           8 non-null int64
Sex           8 non-null object
Birthplace    8 non-null object
weapon        8 non-null object
dtypes: int64(1), object(4)
memory usage: 384.0+ bytes
```

1.2 df.head(num)

如果我们的数据量非常大，我想看看数据长啥样，我当然不希望查看所有的数据了，这时候我们可以采用只看头部的 n 条或者尾部的 n 条。

查看头部的n 条数据可以使用 head 方法，查看尾部的 n 条数据可以使用 tail 方法。

In [4]:
```
user_infor.head(3)
```

Out[4]:

索引	Hero Name	Age	Sex	Birthplace	weapon
0	蜘蛛侠	19	男	纽约	蜘蛛感应
1	灭霸	3000	男	泰坦星球	暴君屠刀
2	奇异博士	36	男	费城	魔法

In [5]:
```
user_infor.tail(3)
```

Out[5]:

索引	Hero Name	Age	Sex	Birthplace	weapon
5	索尔	1505	男	阿斯加德	暴风战斧
6	神奇女侠	2000	女	天堂岛	弒神者
7	黑寡妇	35	女	斯大林格勒	枪械

1.3 df.shape

此外，Pandas 中的数据结构都有 ndarray 中的常用方法和属性，如通过 .shape 获取数据的形状，通过 .T 获取数据的转置。

In [6]:
```
user_infor.shape
```

Out[6]: (8, 5)

1.4 df.T

In [7]:
```
user_infor.T   #返回新的DataFrame，不会修改原对象
#或者用user_info2.tansform()
```

Out[7]:

索引	0	1	2	3	4	5	6	7
Hero Name	蜘蛛侠	灭霸	奇异博士	钢铁侠	蝙蝠侠	索尔	神奇女侠	黑寡妇
Age	19	3000	36	42	40	1505	2000	35
Sex	男	男	男	男	男	男	女	女
Birthplace	纽约	泰坦星球	费城	纽约	哥谭	阿斯加德	天堂岛	斯大林格勒
weapon	蜘蛛感应	暴君屠刀	魔法	纳米战甲	有钱	暴风战斧	弑神者	枪械

1.5 df.values

如果我们想要通过 DataFrame 来获取它包含的原有数据，可以通过 .values 来获取，获取后的数据类型其实是一个 ndarray。

In [8]:
```
user_infor.values
```

Out[8]:
```
array([['蜘蛛侠', 19, '男', '纽约', '蜘蛛感应'],
       ['灭霸', 3000, '男', '泰坦星球', '暴君屠刀'],
       ['奇异博士', 36, '男', '费城', '魔法'],
       ['钢铁侠', 42, '男', '纽约', '纳米战甲'],
       ['蝙蝠侠', 40, '男', '哥谭', '有钱'],
       ['索尔', 1505, '男', '阿斯加德', '暴风战斧'],
       ['神奇女侠', 2000, '女', '天堂岛', '弑神者'],
       ['黑寡妇', 35, '女', '斯大林格勒', '枪械']], dtype=object)
```

In [9]:
```
type(user_infor.values)
```

Out[9]:
```
numpy.ndarray
```

2 描述与统计

2.1 常用描述性统计指标

有时候我们获取到数据之后，想要查看下数据的简单统计指标（最大值、最小值、平均值、中位数等），比如想要查看年龄的最大值，如何实现呢?

DataFrame中有两种方法可以实现：

- 方法一：直接通过DataFrame抽取出来的单列series使用series的相应统计方法或函数。
- 方法二：通过使用numpy对应的方法或函数，处理目标就是DataFrame抽出来的series。

DataFrame和Series均支持以下方法：

指标	描述
count()	计数项
first()、last()	第一项和最后一项
mean()、median()	均值与中位数
min()、max()	最大值与最小值
std()、var()	标准差与方差
mad()	均值绝对偏差
prod()	所有项乘积
sum()	所有项求和

In [10]:
```
user_infor
```

Out[10]:

索引	Hero Name	Age	Sex	Birthplace	weapon
0	蜘蛛侠	19	男	纽约	蜘蛛感应
1	灭霸	3000	男	泰坦星球	暴君屠刀
2	奇异博士	36	男	费城	魔法
3	钢铁侠	42	男	纽约	纳米战甲
4	蝙蝠侠	40	男	哥谭	有钱
5	索尔	1505	男	阿斯加德	暴风战斧
6	神奇女侠	2000	女	天堂岛	弒神者
7	黑寡妇	35	女	斯大林格勒	枪械

In [11]:
```
user_infor.Age
```

Out[11]:
```
索引
0      19
1    3000
2      36
3      42
4      40
5    1505
6    2000
7      35
Name: Age, dtype: int64
```

比如我们想要统计user_infor中年龄的平均值：

In [12]:
```
#方法一
user_infor.Age.mean()
```

Out[12]: 834.625

In [13]:
```
#方法二
np.mean(user_infor.Age)
```

Out[13]: 834.625

那么，series能用哪些描述性统计的指标呢？——numpy中ndarray相应的函数都能用在这里：

In [14]:
```
#numpy的函数都能用在这里
user_infor.Age.max()
user_infor.Age.idxmax()
user_infor.Age.idxmin()
```

Out[14]: 3000

Out[14]: 1

Out[14]: 0

In [15]:
```
user_infor.Age.cumsum()
```

Out[15]:
```
索引
0       19
1     3019
2     3055
3     3097
4     3137
5     4642
6     6642
7     6677
Name: Age, dtype: int64
```

类似的，通过调用 min、mean、quantile、sum 方法可以实现最小值、平均值、中位数以及求和。可以看到，对一个 Series 调用 这几个方法之后，返回的都只是一个聚合结果。

2.2 练习

In [17]:
```
grade=pd.read_csv('student_grade.txt',sep='\t')
grade.head()
```

Out[17]:

	姓名	语文	数学	英语	总分	班名次
0	杨璐	131	143	144	418	1
1	王雪	131	135	144	410	2
2	韩林霖	127	139	142	408	3
3	沙龙逸	123	148	136	407	4
4	李鉴学	126	135	140	401	5

- 计算数学成绩的平均值，以及数学成绩的标准差。
- 生成第二个总分列，计算各科成绩的总分，命名为‘总分2’。
- 计算每个同学的‘数学减去语文’的成绩，取绝对值，命名为‘差值’。

In [18]:
```

```

Out[18]: 102.82352941176471

Out[18]: 33.39583546137604

In [19]:

Out[19]:

	姓名	语文	数学	英语	总分	班名次	总分1	总分2
0	杨璐	131	143	144	418	1	418	418
1	王雪	131	135	144	410	2	410	410
2	韩林霖	127	139	142	408	3	408	408
3	沙龙逸	123	148	136	407	4	407	407
4	李鉴学	126	135	140	401	5	401	401

In [20]:

Out[20]:
```
数学    6992
语文    7344
英语    7412
dtype: int64
```

In [21]:

Out[21]:

	姓名	语文	数学	英语	总分	班名次	总分1	总分2	差值
0	杨璐	131	143	144	418	1	418	418	12
1	王雪	131	135	144	410	2	410	410	4
2	韩林霖	127	139	142	408	3	408	408	12
3	沙龙逸	123	148	136	407	4	407	407	25
4	李鉴学	126	135	140	401	5	401	401	9

如果想要获取更多的统计方法，可以参见官方链接：Descriptive statistics (http://pandas.pydata.org/pandas-docs/stable/basics.html#descriptive-statistics (http://pandas.pydata.org/pandas-docs/stable/basics.html#descriptive-statistics))

虽然说常见的各种统计值都有对应的方法，如果我想要得到多个指标的话，就需要调用多次方法，是不是显得有点麻烦呢?

Pandas 设计者自然也考虑到了这个问题，想要一次性获取多个统计指标，只需调用 describe 方法即可

2.3 df.describe()

In [22]:
```
# 只支持数值列
grade.describe()
```

Out[22]:

	语文	数学	英语	总分	班名次	总分1	总分2	差值
count	68.000000	68.000000	68.000000	68.000000	68.00000	68.000000	68.000000	68.000000
mean	108.000000	102.823529	109.000000	319.823529	34.50000	319.823529	319.823529	17.176471
std	14.519159	33.395835	30.289647	73.782241	19.77372	73.782241	73.782241	16.414356
min	66.000000	21.000000	34.000000	123.000000	1.00000	123.000000	123.000000	0.000000
25%	100.500000	85.750000	100.250000	286.750000	17.75000	286.750000	286.750000	5.500000
50%	111.500000	111.500000	119.000000	343.500000	34.50000	343.500000	343.500000	10.000000
75%	118.000000	127.250000	130.250000	378.000000	51.25000	378.000000	378.000000	26.250000
max	131.000000	148.000000	144.000000	418.000000	68.00000	418.000000	418.000000	78.000000

可以看到，直接调用 describe 方法后，会显示出数字类型的列的一些统计指标，如 总数、平均数、标准差、最小值、最大值、25%/50%/75% 分位数。

如果想要查看非数字类型的列的统计指标，可以设置 include=["object"] 来获得。

In [24]:
```
user_infor=pd.read_csv("new_infor.csv",index_col="索引")
user_infor.describe(include=["object"])  #显示数据类型是object的字段信息
```

Out[24]:

	Hero Name	Sex	Birthplace	weapon
count	8	8	8	8
unique	8	2	7	8
top	奇异博士	男	纽约	暴风战斧
freq	1	6	2	1

上面的结果展示了非数字类型的列的一些统计指标：总数，去重后的个数、最常见的值、最常见的值的频数。

2.4 Series.value_counts

In [26]:
```
user_infor=pd.read_csv("new_infor.csv",index_col="索引")
user_infor
```

Out[26]:

索引	Hero Name	Age	Sex	Birthplace	weapon
0	蜘蛛侠	19	男	纽约	蜘蛛感应
1	灭霸	3000	男	泰坦星球	暴君屠刀
2	奇异博士	36	男	费城	魔法
3	钢铁侠	42	男	纽约	纳米战甲
4	蝙蝠侠	40	男	哥谭	有钱
5	索尔	1505	男	阿斯加德	暴风战斧
6	神奇女侠	2000	女	天堂岛	弒神者
7	黑寡妇	35	女	斯大林格勒	枪械

In [27]:
```
user_infor['Birthplace'].value_counts()
```

Out[27]:
```
纽约        2
天堂岛       1
斯大林格勒     1
阿斯加德       1
费城         1
哥谭         1
泰坦星球       1
Name: Birthplace, dtype: int64
```

如果想要获取某列最大值或最小值对应的索引，可以使用 idxmax 或 idxmin 方法完成。

In [29]:
```
user_infor.Age.idxmin()
```

Out[29]: 0

注意区分，ndarray对应方法为np.argmin()

3 离散化

3.1 pd.cut()

有时候，我们会碰到这样的需求，想要将年龄进行离散化（分箱），直白来说就是将年龄分成几个区间，这里我们想要将年龄分成 3 个区间段。就可以使用 pd.cut 方法来完成,返回的也是Series。

In [31]:
```
pd.cut(user_infor['Age'], 3)
```

Out[31]:
```
索引
0       (16.019, 1012.667]
1       (2006.333, 3000.0]
2       (16.019, 1012.667]
3       (16.019, 1012.667]
4       (16.019, 1012.667]
5    (1012.667, 2006.333]
6    (1012.667, 2006.333]
7       (16.019, 1012.667]
Name: Age, dtype: category
Categories (3, interval[float64]): [(16.019, 1012.667] < (1012.667, 2006.333] < (2006.333, 300
0.0]]
```

可以看到， cut 自动生成了等距的离散区间，如果自己想定义也是没问题的。

In [32]:
```
pd.cut(user_infor.Age, [0, 18, 60, 100,5000])
```

Out[32]:
```
索引
0       (18, 60]
1    (100, 5000]
2       (18, 60]
3       (18, 60]
4       (18, 60]
5    (100, 5000]
6    (100, 5000]
7       (18, 60]
Name: Age, dtype: category
Categories (4, interval[int64]): [(0, 18] < (18, 60] < (60, 100] < (100, 5000]]
```

有时候离散化之后，想要给每个区间起个名字，可以在pd.cut()中使用参数 labels 来指定。

In [34]:
```
a=pd.cut(user_infor.Age, [0, 18, 60, 100,150,5000], labels=["未成年", "成年", "老年人","超长寿","
a
```

Out[34]:
```
索引
0     成年
1    非人类
2     成年
3     成年
4     成年
5    非人类
6    非人类
7     成年
Name: Age, dtype: category
Categories (5, object): [未成年 < 成年 < 老年人 < 超长寿 < 非人类]
```

既然pd.cut()返回的是一列Series，那么可以将其添加到原DataFrame中：

In [35]:
```
user_infor["年龄段"]=a
user_infor
```

Out[35]:

索引	Hero Name	Age	Sex	Birthplace	weapon	年龄段
0	蜘蛛侠	19	男	纽约	蜘蛛感应	成年
1	灭霸	3000	男	泰坦星球	暴君屠刀	非人类
2	奇异博士	36	男	费城	魔法	成年
3	钢铁侠	42	男	纽约	纳米战甲	成年
4	蝙蝠侠	40	男	哥谭	有钱	成年
5	索尔	1505	男	阿斯加德	暴风战斧	非人类
6	神奇女侠	2000	女	天堂岛	弒神者	非人类
7	黑寡妇	35	女	斯大林格勒	枪械	成年

3.2 pd.qcut()

除了可以使用 cut 进行离散化之外，qcut 也可以实现离散化。cut 是根据**每个值的大小来进行离散化的，qcut 是根据每个值出现的次数来进行离散化，也就是基于分位数的离散化功能。**

In [37]:
```
user_infor=pd.read_csv("new_infor.csv",index_col="索引")
user_infor

pd.qcut(user_infor.Age, 3)
```

Out[37]:

索引	Hero Name	Age	Sex	Birthplace	weapon
0	蜘蛛侠	19	男	纽约	蜘蛛感应
1	灭霸	3000	男	泰坦星球	暴君屠刀
2	奇异博士	36	男	费城	魔法
3	钢铁侠	42	男	纽约	纳米战甲
4	蝙蝠侠	40	男	哥谭	有钱
5	索尔	1505	男	阿斯加德	暴风战斧
6	神奇女侠	2000	女	天堂岛	弒神者
7	黑寡妇	35	女	斯大林格勒	枪械

Out[37]:
```
索引
0       (18.999, 37.333]
1    (1017.333, 3000.0]
2       (18.999, 37.333]
3    (37.333, 1017.333]
4    (37.333, 1017.333]
5    (1017.333, 3000.0]
6    (1017.333, 3000.0]
7       (18.999, 37.333]
Name: Age, dtype: category
Categories (3, interval[float64]): [(18.999, 37.333] < (37.333, 1017.333] < (1017.333, 3000.0]]
```

3.3 练习

In [39]:
```
grade=pd.read_csv('student_grade.txt',sep='\t')
grade.head()
```

Out[39]:

	姓名	语文	数学	英语	总分	班名次
0	杨璐	131	143	144	418	1
1	王雪	131	135	144	410	2
2	韩林霖	127	139	142	408	3
3	沙龙逸	123	148	136	407	4
4	李鉴学	126	135	140	401	5

第一小题：

grade按照数学成绩进行离散化，分为“优”、“良”、“及格”、“不及格”：

[135,150]-->优

[120,135)-->良

[90,120)-->及格

[0,90)-->不及格

第二小题：

如何统计不同数学成绩等级的人数？

In [40]:
```

```

Out[40]:

	姓名	语文	数学	英语	总分	班名次	数学等级
0	杨璐	131	143	144	418	1	优
1	王雪	131	135	144	410	2	良
2	韩林霖	127	139	142	408	3	优
3	沙龙逸	123	148	136	407	4	优
4	李鉴学	126	135	140	401	5	良
5	韩雨萌	129	133	138	400	6	良
6	刘帅	116	143	140	399	7	优
7	康惠雯	114	142	139	395	8	优
8	刘钰婷	115	139	135	389	9	优
9	林世博	116	142	129	387	10	优

In [42]:
```

```

Out[42]:
```
及格      22
不及格     19
良        18
优         9
Name: 数学等级, dtype: int64
```

4 排序功能

在进行数据分析时，少不了进行数据排序。Pandas 支持两种排序方式：按轴（索引或列）排序和按实际值排序。

4.1 df.sort_index()

sort_index 方法默认是按照索引进行正序排的。

In [44]:
```
user_info=pd.read_csv("new_infor.csv",index_col="索引")
user_infor.sort_index()
```

Out[44]:

索引	Hero Name	Age	Sex	Birthplace	weapon
0	蜘蛛侠	19	男	纽约	蜘蛛感应
1	灭霸	3000	男	泰坦星球	暴君屠刀
2	奇异博士	36	男	费城	魔法
3	钢铁侠	42	男	纽约	纳米战甲
4	蝙蝠侠	40	男	哥谭	有钱
5	索尔	1505	男	阿斯加德	暴风战斧
6	神奇女侠	2000	女	天堂岛	弒神者
7	黑寡妇	35	女	斯大林格勒	枪械

In [45]:
```
# 如果想要按照列进行倒序排，可以设置ascending=False。
user_infor.sort_index(ascending=False)
```

Out[45]:

索引	Hero Name	Age	Sex	Birthplace	weapon
7	黑寡妇	35	女	斯大林格勒	枪械
6	神奇女侠	2000	女	天堂岛	弒神者
5	索尔	1505	男	阿斯加德	暴风战斧
4	蝙蝠侠	40	男	哥谭	有钱
3	钢铁侠	42	男	纽约	纳米战甲
2	奇异博士	36	男	费城	魔法
1	灭霸	3000	男	泰坦星球	暴君屠刀
0	蜘蛛侠	19	男	纽约	蜘蛛感应

4.2 df.sort_values()

如果想要实现按照实际值来排序，例如想要按照年龄排序，如何实现呢？

使用 sort_values 方法，设置参数 by="age" 即可。

In [47]:
```
user_info=pd.read_csv("new_infor.csv",index_col="索引")
user_info.loc[8]=["星爵",35,"男","科罗拉多州","枪械"]
user_info["Power"]=[95,140,120,120,120,138,130,95,90]
user_info
```

Out[47]:

索引	Hero Name	Age	Sex	Birthplace	weapon	Power
0	蜘蛛侠	19	男	纽约	蜘蛛感应	95
1	灭霸	3000	男	泰坦星球	暴君屠刀	140
2	奇异博士	36	男	费城	魔法	120
3	钢铁侠	42	男	纽约	纳米战甲	120
4	蝙蝠侠	40	男	哥谭	有钱	120
5	索尔	1505	男	阿斯加德	暴风战斧	138
6	神奇女侠	2000	女	天堂岛	弒神者	130
7	黑寡妇	35	女	斯大林格勒	枪械	95
8	星爵	35	男	科罗拉多州	枪械	90

有时候我们可能需要按照多个值来排序，例如：按照年龄和城市来一起排序，可以设置参数 by 为一个 list 即可。

注意：list 中每个元素的顺序会影响排序优先级的。

In [48]:
```
user_info.sort_values(by=["Age","Power"])
```

Out[48]:

索引	Hero Name	Age	Sex	Birthplace	weapon	Power
0	蜘蛛侠	19	男	纽约	蜘蛛感应	95
8	星爵	35	男	科罗拉多州	枪械	90
7	黑寡妇	35	女	斯大林格勒	枪械	95
2	奇异博士	36	男	费城	魔法	120
4	蝙蝠侠	40	男	哥谭	有钱	120
3	钢铁侠	42	男	纽约	纳米战甲	120
5	索尔	1505	男	阿斯加德	暴风战斧	138
6	神奇女侠	2000	女	天堂岛	弒神者	130
1	灭霸	3000	男	泰坦星球	暴君屠刀	140

In [49]:
```
user_info.sort_values(by=["Age","Power"],ascending=[True,False])
```

Out[49]:

索引	Hero Name	Age	Sex	Birthplace	weapon	Power
0	蜘蛛侠	19	男	纽约	蜘蛛感应	95
7	黑寡妇	35	女	斯大林格勒	枪械	95
8	星爵	35	男	科罗拉多州	枪械	90
2	奇异博士	36	男	费城	魔法	120
4	蝙蝠侠	40	男	哥谭	有钱	120
3	钢铁侠	42	男	纽约	纳米战甲	120
5	索尔	1505	男	阿斯加德	暴风战斧	138
6	神奇女侠	2000	女	天堂岛	弒神者	130
1	灭霸	3000	男	泰坦星球	暴君屠刀	140

In [50]:
```
heros=user_info.copy()
```

4.3 series.nlargest()

一般在排序后，我们可能需要获取最大的n个值或最小值的n个值，我们可以使用 nlargest 和 nsmallest 方法来完成，这比先排序再使用 head(n)方法快得多。

In [51]:
```
heros
```

Out[51]:

索引	Hero Name	Age	Sex	Birthplace	weapon	Power
0	蜘蛛侠	19	男	纽约	蜘蛛感应	95
1	灭霸	3000	男	泰坦星球	暴君屠刀	140
2	奇异博士	36	男	费城	魔法	120
3	钢铁侠	42	男	纽约	纳米战甲	120
4	蝙蝠侠	40	男	哥谭	有钱	120
5	索尔	1505	男	阿斯加德	暴风战斧	138
6	神奇女侠	2000	女	天堂岛	弒神者	130
7	黑寡妇	35	女	斯大林格勒	枪械	95
8	星爵	35	男	科罗拉多州	枪械	90

In [52]:
```
heros.Age.nlargest(3)
heros.Age.nsmallest(3)
```

Out[52]:
```
索引
1    3000
6    2000
5    1505
Name: Age, dtype: int64
```

Out[52]:
```
索引
0    19
7    35
8    35
Name: Age, dtype: int64
```

5 函数应用及映射方法

虽说 Pandas 为我们提供了非常丰富的函数，有时候我们可能需要自己使用高级函数来实现自定义功能，并将它应用到 DataFrame 或 Series。常用到的函数有：

1. map
2. apply
3. applymap

5.1 Series.map()

map 是 Series 中特有的方法，通过它可以对 Series 中的每个元素实现转换。

如果我想通过年龄判断用户是否属于中年人（50岁以上为中年），通过 map 可以轻松搞定它。

In [53]:
```
di = {
"纽约皇后区": "地球人",
"泰坦星球": "外星人",
"费城": "地球人",
"纽约": "地球人",
"哥谭":"地球人",
"阿斯加德":"外星人",
"天堂岛":"地球人",
"斯大林格勒":"地球人"
""
}

heros['Alien'] = heros.Birthplace.map(di)
```

In [54]:
```
heros.head()
```

Out[54]:

索引	Hero Name	Age	Sex	Birthplace	weapon	Power	Alien
0	蜘蛛侠	19	男	纽约	蜘蛛感应	95	地球人
1	灭霸	3000	男	泰坦星球	暴君屠刀	140	外星人
2	奇异博士	36	男	费城	魔法	120	地球人
3	钢铁侠	42	男	纽约	纳米战甲	120	地球人
4	蝙蝠侠	40	男	哥谭	有钱	120	地球人

另外，可以使用自定义函数来形成这种映射关系：

In [55]:
```
earth_city=['纽约','费城','纽约','哥谭','天堂岛',"斯大林格勒","科罗拉多州"]

def func(x):
    if x in earth_city:
        return "地球人"
    else:
        return "外星人"
```

In [56]:
```
heros['Alien_02'] = heros.Birthplace.map(func)
heros
```

Out[56]:

索引	Hero Name	Age	Sex	Birthplace	weapon	Power	Alien	Alien_02
0	蜘蛛侠	19	男	纽约	蜘蛛感应	95	地球人	地球人
1	灭霸	3000	男	泰坦星球	暴君屠刀	140	外星人	外星人
2	奇异博士	36	男	费城	魔法	120	地球人	地球人
3	钢铁侠	42	男	纽约	纳米战甲	120	地球人	地球人
4	蝙蝠侠	40	男	哥谭	有钱	120	地球人	地球人
5	索尔	1505	男	阿斯加德	暴风战斧	138	外星人	外星人
6	神奇女侠	2000	女	天堂岛	弒神者	130	地球人	地球人
7	黑寡妇	35	女	斯大林格勒	枪械	95	地球人	地球人
8	星爵	35	男	科罗拉多州	枪械	90	NaN	地球人

5.2 Series.apply()

apply 方法既支持 Series，也支持 DataFrame，在对 Series 操作时会作用到每个值上，在对 DataFrame 操作时会作用到所有行或所有列（通过 axis参数控制）。

In [57]:
```
# 对 Series 来说，apply 方法 与 map 方法区别不大。
earth_city=['纽约','费城','纽约','哥谭','天堂岛',"斯大林格勒","科罗拉多州"]

def func(x):
    if x in earth_city:
        return "地球人"
    else:
        return "外星人"


heros['Alien'] = heros.Birthplace.apply(func)  #这里只是将上一小节中的Series.map()换成了.apply(),
heros
```

Out[57]:

索引	Hero Name	Age	Sex	Birthplace	weapon	Power	Alien	Alien_02
0	蜘蛛侠	19	男	纽约	蜘蛛感应	95	地球人	地球人
1	灭霸	3000	男	泰坦星球	暴君屠刀	140	外星人	外星人
2	奇异博士	36	男	费城	魔法	120	地球人	地球人
3	钢铁侠	42	男	纽约	纳米战甲	120	地球人	地球人
4	蝙蝠侠	40	男	哥谭	有钱	120	地球人	地球人
5	索尔	1505	男	阿斯加德	暴风战斧	138	外星人	外星人
6	神奇女侠	2000	女	天堂岛	弒神者	130	地球人	地球人
7	黑寡妇	35	女	斯大林格勒	枪械	95	地球人	地球人
8	星爵	35	男	科罗拉多州	枪械	90	地球人	地球人

对 DataFrame 来说，apply 方法的作用对象是一行或一列数据（一个Series）

- axis为 0或'index'：将函数应用于每列。
- axis为1或'columns'：将函数应用于每一行。

In [58]:
```
def re_max(x):
    return x.max()

heros.apply(re_max, axis=0)
```

Out[58]:
```
Hero Name       黑寡妇
Age            3000
Sex               男
Birthplace     阿斯加德
weapon           魔法
Power             140
Alien          外星人
Alien_02       外星人
dtype: object
```

上面就将每一列中的"最大值"求出。

为什么汉字也能求“最大值”？因为Python3支持的是Unicode，每汉字都对应一个编码数字：

In [133]:
```
ord("魔")
ord("蜘")
```

Out[133]: 39764

Out[133]: 34584

In [136]:
```
chr(39764)
chr(34584)
```

Out[136]: '魔'

Out[136]: '蜘'

5.3 df.applymap()

applymap 方法针对于 DataFrame，它作用于 DataFrame 中的每个元素，它对 DataFrame 的效果类似于 apply 对 Series 的效果。

比如我们要将上表中的所有的"侠"字换成"人"字。

In [59]:
```
def replace(x):
    x=str(x)
    if x.find("侠")!=-1:
        x=x.replace("侠","人")
    elif x.find("人")!=-1:
        x=x.replace("人","者")
    return x

heros.applymap(replace)
```

Out[59]:

索引	Hero Name	Age	Sex	Birthplace	weapon	Power	Alien	Alien_02
0	蜘蛛人	19	男	纽约	蜘蛛感应	95	地球者	地球者
1	灭霸	3000	男	泰坦星球	暴君屠刀	140	外星者	外星者
2	奇异博士	36	男	费城	魔法	120	地球者	地球者
3	钢铁人	42	男	纽约	纳米战甲	120	地球者	地球者
4	蝙蝠人	40	男	哥谭	有钱	120	地球者	地球者
5	索尔	1505	男	阿斯加德	暴风战斧	138	外星者	外星者
6	神奇女人	2000	女	天堂岛	弒神者	130	地球者	地球者
7	黑寡妇	35	女	斯大林格勒	枪械	95	地球者	地球者
8	星爵	35	男	科罗拉多州	枪械	90	地球者	地球者