Table of Contents

In [2]:
```
#全部行都能输出
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import numpy as np,pandas as pd
```

DataFrame 是一个带有索引的二维数据结构，每列可以有自己的名字，并且可以有不同的数据类型。你可以把它想象成一个 excel 表格或者数据库中的一张表，也可以将它想象成由多个Series拼接成的一个

DataFrame，公用一个索引，它是最常用的 Pandas 对象。

1 DataFrame的创建

我们继续使用之前的数据来继续操作, 对于每一个英雄, 除了年龄之外, 还可以储存很多其他的数据 我们来换一种方法重新生成数据

pd.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)

1.1 从数组转化过来

In [3]:
```
a=np.arange(1,10).reshape(3,3)
a

pd.DataFrame(a)
```

Out[3]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

Out[3]:

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

1.2 通过字典转化

通过字典转化为DataFrame类型时，"键"会作为列标签，"键"对应的"值"会作为该列名对应的数据列。

In [4]:
```
data = {
    "年龄":[19, 3000, 30, 37, 40, 1500],
    "出生地":["纽约皇后区", "泰坦星球","费城", "纽约", "哥谭",  "阿斯加德" ]
}

#先将dataframe的索引创建，再将索引添加到dataframe中去
index = pd.Index(data=['蜘蛛侠', '灭霸', '奇异博士', '钢铁侠','蝙蝠侠', '索尔' ], name="英雄姓名")

user_info = pd.DataFrame(data=data,index=index )

user_info
```

Out[4]:

英雄姓名	年龄	出生地
蜘蛛侠	19	纽约皇后区
灭霸	3000	泰坦星球
奇异博士	30	费城
钢铁侠	37	纽约
蝙蝠侠	40	哥谭
索尔	1500	阿斯加德

1.3 通过多维数组转化

除了上面这种传入 dict 的方式构建外，我们还可以通过另外一种方式来构建。
这种方式是先构建一个二维数组data，然后再生成一个列标签columns，最后生成一个Index对象。
这样，data、colunms、Index三个对象就构成了DataFrame。

In [8]:
```
data = [[19, "纽约皇后区"],
        [3000, "泰坦星球"],
        [30, "费城"],
        [37, "纽约"],
        [40, "哥谭"],
        [1500, "阿斯加德"]]

columns = ["年龄", "出生地"]
index = pd.Index(data=['蜘蛛侠', '灭霸', '奇异博士', '钢铁侠','蝙蝠侠', '索尔' ], name="英雄姓名")


user_info = pd.DataFrame(data=data, index=index, columns=columns)
user_info
```

Out[8]:

英雄姓名	年龄	出生地
蜘蛛侠	19	纽约皇后区
灭霸	3000	泰坦星球
奇异博士	30	费城
钢铁侠	37	纽约
蝙蝠侠	40	哥谭
索尔	1500	阿斯加德

当然，上面的嵌套列表换成ndarray也是可以的。

1.4 练习1

自行构建一个DataFrame。

In [ ]:
```

```

2 数据提取

2.1 索引和切片

可以像列表切片一样，把每一个Series当做里面的一个元素

In [15]:
```
import pandas as pd

data = [[19, "纽约皇后区",'null'],
        [3000, "泰坦星球"],
        [30, "费城"],
        [37, "纽约"],
        [40, "哥谭"],
        [1500, "阿斯加德"]]

columns = ["年龄", "出生地","血型"]
index = pd.Index(data=['蜘蛛侠', '灭霸', '奇异博士', '钢铁侠','蝙蝠侠', '索尔' ], name="英雄姓名")

user_info = pd.DataFrame(data,columns=columns,index=index)

user_info
```

Out[15]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约皇后区	null
灭霸	3000	泰坦星球	None
奇异博士	30	费城	None
钢铁侠	37	纽约	None
蝙蝠侠	40	哥谭	None
索尔	1500	阿斯加德	None

DataFrame中的列标签可以用作取每一列数据：

In [10]:
```
user_info['出生地']
```

Out[10]:
```
英雄姓名
蜘蛛侠      纽约皇后区
灭霸        泰坦星球
奇异博士        费城
钢铁侠         纽约
蝙蝠侠         哥谭
索尔        阿斯加德
Name: 出生地, dtype: object
```

In [11]:
```
user_info.出生地
```

Out[11]:
```
英雄姓名
蜘蛛侠      纽约皇后区
灭霸        泰坦星球
奇异博士        费城
钢铁侠         纽约
蝙蝠侠         哥谭
索尔        阿斯加德
Name: 出生地, dtype: object
```

如果想提取多列，在列表中写多个列标签名即可：

In [12]:
```
user_info[["出生地", "血型"]]
```

Out[12]:

英雄姓名	出生地	血型
蜘蛛侠	纽约皇后区	null
灭霸	泰坦星球	None
奇异博士	费城	None
钢铁侠	纽约	None
蝙蝠侠	哥谭	None
索尔	阿斯加德	None

2.2 df.loc[ ]

可以用loc [location]的方法来进行显示索引

语法 user_info.loc[索引行,索引列]

In [18]:
```
user_info
```

Out[18]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约皇后区	null
灭霸	3000	泰坦星球	None
奇异博士	30	费城	None
钢铁侠	37	纽约	None
蝙蝠侠	40	哥谭	None
索尔	1500	阿斯加德	None

In [19]:
```
user_info.loc["蝙蝠侠", "年龄"]
```

Out[19]: 40

如果想取多行多列数据，在对应的参数位置，以列表的形式，将要取的行和列的标签名写上即可：

In [20]:
```
user_info.loc[["蜘蛛侠","蝙蝠侠"], ["年龄","出生地"]]
```

Out[20]:

英雄姓名	年龄	出生地
蜘蛛侠	19	纽约皇后区
蝙蝠侠	40	哥谭

2.3 df.iloc[ ]

也可以进行隐式索引

使用 iloc 也就是 index_loc

这种方式不看你的行列的索引标签是什么,只看数据是处于表中的一个什么位置,即，依据真正的行列索引值来取。

In [21]:
```
a=user_info
a
type(a)
```

Out[21]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约皇后区	null
灭霸	3000	泰坦星球	None
奇异博士	30	费城	None
钢铁侠	37	纽约	None
蝙蝠侠	40	哥谭	None
索尔	1500	阿斯加德	None

Out[21]:
```
pandas.core.frame.DataFrame
```

如果只是取第一行:

In [23]:
```
a=user_info.iloc[0,:]  #直接写user_info.iloc[0]也可以
a
type(a)
```

Out[23]:
```
年龄          19
出生地      纽约皇后区
血型        null
Name: 蜘蛛侠, dtype: object
```

Out[23]:
```
pandas.core.series.Series
```

如果取第一列:

In [25]:
```
a=user_info.iloc[:,0]
a
type(a)
```

Out[25]:
```
英雄姓名
蜘蛛侠        19
灭霸        3000
奇异博士        30
钢铁侠         37
蝙蝠侠         40
索尔        1500
Name: 年龄, dtype: int64
```

Out[25]:
```
pandas.core.series.Series
```

如果取第一行第一列的值：

In [26]:
```
a=user_info.iloc[0,0]
a
type(a)
```

Out[26]:
```
19
```

Out[26]:
```
numpy.int64
```

In [27]:
```
a=user_info.iloc[0:, 0:]
a
type(a)
```

Out[27]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约皇后区	null
灭霸	3000	泰坦星球	None
奇异博士	30	费城	None
钢铁侠	37	纽约	None
蝙蝠侠	40	哥谭	None
索尔	1500	阿斯加德	None

Out[27]:
```
pandas.core.frame.DataFrame
```

In [28]:
```
user_info.iloc[1::2,1:2 ]
type(a)
```

Out[28]:

英雄姓名	出生地
灭霸	泰坦星球
钢铁侠	纽约
索尔	阿斯加德

Out[28]:
```
pandas.core.frame.DataFrame
```

与df.loc[ ]类似，df.iloc[ ]也可以用列表写上行列索引值，来取不同的行列数据：

In [29]:
```
user_info.iloc[[0,3],[0,2]]
```

Out[29]:

英雄姓名	年龄	血型
蜘蛛侠	19	null
钢铁侠	37	None

3 行列增删改

我们先建立一个DataFrame对象，以此作为下面操作说明的例子。

In [30]:
```
user_info=pd.read_csv("infor00",index_col=0)
user_info
```

Out[30]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约	NaN
灭霸	3000	泰坦星球	NaN
奇异博士	30	费城	NaN
钢铁侠	37	纽约	NaN
蝙蝠侠	40	哥谭	NaN
索尔	1500	阿斯加德	NaN

3.1 行的增加与修改

3.1.1 行的增加

In [29]:
```
user_info.loc['神奇女侠'] = [2000,'天堂岛', None]
user_info
```

Out[29]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约皇后区	NaN
灭霸	3000	泰坦星球	NaN
奇异博士	30	费城	NaN
钢铁侠	37	纽约	NaN
蝙蝠侠	40	哥谭	NaN
索尔	1500	阿斯加德	NaN
神奇女侠	2000	天堂岛	None

3.1.2 行的修改

In [31]:
```
#修改奇异博士的年龄：30岁变35岁
user_info.loc['奇异博士'] = [35,'费城', None]
user_info
```

Out[31]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约	NaN
灭霸	3000	泰坦星球	NaN
奇异博士	35	费城	NaN
钢铁侠	37	纽约	NaN
蝙蝠侠	40	哥谭	NaN
索尔	1500	阿斯加德	NaN

In [32]:
```
#单独修改奇异博士的年龄：35岁变36岁
user_info.loc['奇异博士','年龄'] = 36
user_info
```

Out[32]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约	NaN
灭霸	3000	泰坦星球	NaN
奇异博士	36	费城	NaN
钢铁侠	37	纽约	NaN
蝙蝠侠	40	哥谭	NaN
索尔	1500	阿斯加德	NaN

3.2 列的添加与修改

在生成了 DataFrame 之后，突然你发现好像缺失了用户的性别这个信息，如何添加呢?

如果所有的性别都一样，我们可以通过传入一个标量，Pandas 会自动帮我们广播来填充所有的位置。

3.2.1 列的增加

In [42]:
```
user_info=pd.read_csv("infor00",index_col=0)

user_info['性别'] = '男'
user_info
```

Out[42]:

英雄姓名	年龄	出生地	血型	性别
蜘蛛侠	19	纽约	NaN	男
灭霸	3000	泰坦星球	NaN	男
奇异博士	30	费城	NaN	男
钢铁侠	37	纽约	NaN	男
蝙蝠侠	40	哥谭	NaN	男
索尔	1500	阿斯加德	NaN	男

In [44]:
```
user_info.loc["神奇女侠"]=np.nan
user_info
```

Out[44]:

英雄姓名	年龄	出生地	血型	性别
蜘蛛侠	19.0	纽约	NaN	男
灭霸	3000.0	泰坦星球	NaN	男
奇异博士	30.0	费城	NaN	男
钢铁侠	37.0	纽约	NaN	男
蝙蝠侠	40.0	哥谭	NaN	男
索尔	1500.0	阿斯加德	NaN	男
神奇女侠	NaN	NaN	NaN	NaN

当然，将一个序列赋值到新的列标签来增加新列，这才是最常用的方法:

In [45]:
```
user_info['性别']=["男","男","男","男","男","男","女"]
user_info
```

Out[45]:

英雄姓名	年龄	出生地	血型	性别
蜘蛛侠	19.0	纽约	NaN	男
灭霸	3000.0	泰坦星球	NaN	男
奇异博士	30.0	费城	NaN	男
钢铁侠	37.0	纽约	NaN	男
蝙蝠侠	40.0	哥谭	NaN	男
索尔	1500.0	阿斯加德	NaN	男
神奇女侠	NaN	NaN	NaN	女

列值修改也是如此，用一个新的序列数据赋值到对应的列标签中即可，在此不再赘述。

3.2.2 np.where()的妙用

如果新列是依据原有的列生成的“衍生变量”，那么需要用到np.where()，比如要生成一列变量，将性别的“男”转化为“1”,“女”转化为“0”：

In [46]:
```
np.where(user_info.性别=="男",1,0)        #注意，这里返回的是numpy的ndarray

user_info["sex"]=np.where(user_info.性别=="男",1,0)
user_info
```

Out[46]:
```
array([1, 1, 1, 1, 1, 1, 0])
```

Out[46]:

英雄姓名	年龄	出生地	血型	性别	sex
蜘蛛侠	19.0	纽约	NaN	男	1
灭霸	3000.0	泰坦星球	NaN	男	1
奇异博士	30.0	费城	NaN	男	1
钢铁侠	37.0	纽约	NaN	男	1
蝙蝠侠	40.0	哥谭	NaN	男	1
索尔	1500.0	阿斯加德	NaN	男	1
神奇女侠	NaN	NaN	NaN	女	0

3.2.3 df.assign()

通过上面的例子可以看出，我们创建新的列的时候都是在原有的 DataFrame 上修改的，也就是说如果添加了新的一列之后，原有的 DataFrame 会发生改变。

我们可以通过 assign 方法来创建新的一列,并返回一个新DataFrame，不修改原DataFrame。

In [50]:
```
user_info.assign(新列 = 88)
```

Out[50]:

英雄姓名	年龄	出生地	血型	性别	sex	新列
蜘蛛侠	19.0	纽约	NaN	男	1	88
灭霸	3000.0	泰坦星球	NaN	男	1	88
奇异博士	30.0	费城	NaN	男	1	88
钢铁侠	37.0	纽约	NaN	男	1	88
蝙蝠侠	40.0	哥谭	NaN	男	1	88
索尔	1500.0	阿斯加德	NaN	男	1	88
神奇女侠	NaN	NaN	NaN	女	0	88

In [49]:
```
user_info
```

Out[49]:

英雄姓名	年龄	出生地	血型	性别	sex
蜘蛛侠	19.0	纽约	NaN	男	1
灭霸	3000.0	泰坦星球	NaN	男	1
奇异博士	30.0	费城	NaN	男	1
钢铁侠	37.0	纽约	NaN	男	1
蝙蝠侠	40.0	哥谭	NaN	男	1
索尔	1500.0	阿斯加德	NaN	男	1
神奇女侠	NaN	NaN	NaN	女	0

DataFrame.assign()往往会搭配np.where()使用：

比如，依据原有的user_infor表，生成一个新的DataFrame，新的DataFrame新增一列变量，如果英雄性别为男，年龄增加10岁，性别为女增加5岁，新列列名为“新年龄”。

3.3 练习2

如何在不修改原表的基础上，用df.assign()的方法对下表的"Sex"列值进行转换：“男"转换为"1"，"女"转化为"0"：

In [55]:
```
new_infor=pd.read_csv("new_infor.csv",index_col="Unnamed: 0")
new_infor
```

Out[55]:

	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	弑神者
7	黑寡妇	35	女	斯大林格勒	枪械

In [64]:
```

```

Out[64]:

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

Out[64]:

	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	弑神者
7	黑寡妇	35	女	斯大林格勒	枪械

3.4 行列值修改

In [65]:
```
#也可以同时修改多个行列信息
user_info.loc[['钢铁侠','索尔'],["年龄","血型"]] =[ [43,"A"],[1505,"?"]]
user_info
```

Out[65]:

英雄姓名	年龄	出生地	血型	性别	sex
蜘蛛侠	19.0	纽约	NaN	男	1
灭霸	3000.0	泰坦星球	NaN	男	1
奇异博士	30.0	费城	NaN	男	1
钢铁侠	43.0	纽约	A	男	1
蝙蝠侠	40.0	哥谭	NaN	男	1
索尔	1505.0	阿斯加德	?	男	1
神奇女侠	NaN	NaN	NaN	女	0

同样地，也可以用DataFrame.iloc[ ]方法，只不过参数换成了行索引值和列索引值：

In [66]:
```
#修改钢铁侠和索尔的年龄
user_info.iloc[3,0] =42
user_info.iloc[5,0] =1505
user_info

# #也可以这样写：
# user_infor.iloc[[3,5],0] =[42,1505]
# user_infor
```

Out[66]:

英雄姓名	年龄	出生地	血型	性别	sex
蜘蛛侠	19.0	纽约	NaN	男	1
灭霸	3000.0	泰坦星球	NaN	男	1
奇异博士	30.0	费城	NaN	男	1
钢铁侠	42.0	纽约	A	男	1
蝙蝠侠	40.0	哥谭	NaN	男	1
索尔	1505.0	阿斯加德	?	男	1
神奇女侠	NaN	NaN	NaN	女	0

3.5 练习3

为user_info新生成一列，列名为年龄段，要求：

- 如果年龄小于40岁，返回"青年";
- 如果年龄40以上，则返回"非青年";
- 新增加一列的表作为新的对象返回，不改变原表。

In [74]:
```
user_info=pd.read_csv("infor00")
user_info
```

Out[74]:

	英雄姓名	年龄	出生地	血型
0	蜘蛛侠	19	纽约	NaN
1	灭霸	3000	泰坦星球	NaN
2	奇异博士	30	费城	NaN
3	钢铁侠	37	纽约	NaN
4	蝙蝠侠	40	哥谭	NaN
5	索尔	1500	阿斯加德	NaN

In [75]:
```

```

Out[75]:
```
array(['青年', '非青年', '青年', '青年', '非青年', '非青年'], dtype='<U3')
```

In [76]:
```

```

Out[76]:

	英雄姓名	年龄	出生地	血型	年龄段
0	蜘蛛侠	19	纽约	NaN	青年
1	灭霸	3000	泰坦星球	NaN	非青年
2	奇异博士	30	费城	NaN	青年
3	钢铁侠	37	纽约	NaN	青年
4	蝙蝠侠	40	哥谭	NaN	非青年
5	索尔	1500	阿斯加德	NaN	非青年

3.6 练习4

为user_info新生成一列，列名为“年龄段”，要求：

- 如果年龄小于40岁，返回"青年";
- 如果年龄40到60岁（包括40），返回"中年";
- 如果年龄60到130岁（包括60），返回"老年";
- 如果130岁以上，返回"超长寿";
- 新增加一列的表作为新的对象返回，不改变原表。
  提示：灵活运用自定义函数和np.vectorize()或者np.frompyfunc()

In [78]:
```
user_info=pd.read_csv("infor00")
user_info
```

Out[78]:

	英雄姓名	年龄	出生地	血型
0	蜘蛛侠	19	纽约	NaN
1	灭霸	3000	泰坦星球	NaN
2	奇异博士	30	费城	NaN
3	钢铁侠	37	纽约	NaN
4	蝙蝠侠	40	哥谭	NaN
5	索尔	1500	阿斯加德	NaN

In [79]:
```

```

Out[79]: array(['青年', '超长寿', '青年', '青年', '中年', '超长寿'], dtype='<U3')

Out[79]:

	英雄姓名	年龄	出生地	血型	年龄段
0	蜘蛛侠	19	纽约	NaN	青年
1	灭霸	3000	泰坦星球	NaN	超长寿
2	奇异博士	30	费城	NaN	青年
3	钢铁侠	37	纽约	NaN	青年
4	蝙蝠侠	40	哥谭	NaN	中年
5	索尔	1500	阿斯加德	NaN	超长寿

In [80]:
```

```

Out[80]: array(['青年', '超长寿', '青年', '青年', '中年', '超长寿'], dtype=object)

Out[80]:

	英雄姓名	年龄	出生地	血型	年龄段
0	蜘蛛侠	19	纽约	NaN	青年
1	灭霸	3000	泰坦星球	NaN	超长寿
2	奇异博士	30	费城	NaN	青年
3	钢铁侠	37	纽约	NaN	青年
4	蝙蝠侠	40	哥谭	NaN	中年
5	索尔	1500	阿斯加德	NaN	超长寿

3.7 行列删除

3.7.1 df.drop()

df.drop(['labels=None', 'axis=0', 'index=None', 'columns=None', 'level=None', 'inplace=False'],)

- labels：单个标签或类似列表要删除的索引或列标签。

- axis：{0或'index'，1或'columns'}，默认为0,决定从索引中删除标签（0或'索引'）或列（1或'列'）。
- index，columns：单个标签或类似列表替代指定轴（ 标签，轴= 1 相当于 columns = labels ）。
- level：int或level name，可选。对于MultiIndex，将从中删除标签的级别。
- inplace：bool，默认为False。如果为True，则执行就地操作并返回None。

比如删除灭霸这行记录：

In [81]:
```
user_info=pd.read_csv("infor00",index_col=0)
user_info
```

Out[81]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约	NaN
灭霸	3000	泰坦星球	NaN
奇异博士	30	费城	NaN
钢铁侠	37	纽约	NaN
蝙蝠侠	40	哥谭	NaN
索尔	1500	阿斯加德	NaN

使用索引标签名来删除行：

In [82]:
```
user_info.drop(index='灭霸')
```

Out[82]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约	NaN
奇异博士	30	费城	NaN
钢铁侠	37	纽约	NaN
蝙蝠侠	40	哥谭	NaN
索尔	1500	阿斯加德	NaN

使用列标签名来删除列：

In [83]:
```
user_info.drop(columns='血型')
```

Out[83]:

英雄姓名	年龄	出生地
蜘蛛侠	19	纽约
灭霸	3000	泰坦星球
奇异博士	30	费城
钢铁侠	37	纽约
蝙蝠侠	40	哥谭
索尔	1500	阿斯加德

也可以通过指定轴的形式来进行行列删除：

In [84]:
```
user_info.drop(axis=0,labels='灭霸')
```

Out[84]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约	NaN
奇异博士	30	费城	NaN
钢铁侠	37	纽约	NaN
蝙蝠侠	40	哥谭	NaN
索尔	1500	阿斯加德	NaN

In [85]:
```
user_info.drop(axis=0,labels=['灭霸',"钢铁侠"])
```

Out[85]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约	NaN
奇异博士	30	费城	NaN
蝙蝠侠	40	哥谭	NaN
索尔	1500	阿斯加德	NaN

In [86]:
```
user_info.drop(axis=1,labels=['年龄',"血型"])
```

Out[86]:

英雄姓名	出生地
蜘蛛侠	纽约
灭霸	泰坦星球
奇异博士	费城
钢铁侠	纽约
蝙蝠侠	哥谭
索尔	阿斯加德

df.drop()操作并没有删除原表记录，而是返回了一个新的DataFrame对象。

In [87]:
```
user_info
```

Out[87]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约	NaN
灭霸	3000	泰坦星球	NaN
奇异博士	30	费城	NaN
钢铁侠	37	纽约	NaN
蝙蝠侠	40	哥谭	NaN
索尔	1500	阿斯加德	NaN

```
```

df.pop()

df.pop()方法只能删除列。

In [96]:
```
user_info=pd.read_csv("infor00",index_col=0)
user_info
```

Out[96]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约	NaN
灭霸	3000	泰坦星球	NaN
奇异博士	30	费城	NaN
钢铁侠	37	纽约	NaN
蝙蝠侠	40	哥谭	NaN
索尔	1500	阿斯加德	NaN

In [97]:
```
user_info.pop("血型")
```

Out[97]:
```
英雄姓名
蜘蛛侠    NaN
灭霸      NaN
奇异博士    NaN
钢铁侠     NaN
蝙蝠侠     NaN
索尔      NaN
Name: 血型, dtype: float64
```

df.pop()会直接删除原表的列！谨慎操作！

In [98]:
```
user_info
```

Out[98]:

英雄姓名	年龄	出生地
蜘蛛侠	19	纽约
灭霸	3000	泰坦星球
奇异博士	30	费城
钢铁侠	37	纽约
蝙蝠侠	40	哥谭
索尔	1500	阿斯加德

3.8 列顺序的更改

如果我们想新修改user_infor2的列顺序，有两个方法：

In [99]:
```
new_infor=pd.read_csv("new_infor.csv",index_col="Unnamed: 0")
new_infor
```

Out[99]:

	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	弑神者
7	黑寡妇	35	女	斯大林格勒	枪械

3.8.1 df [列名组]

先用列表定义行的列名顺序，再按照这个新的顺序重新赋值回原DataFrame：

In [102]:
```
change=['Hero Name', 'Sex','Age', 'weapon',  'Birthplace']

new_infor[change]     #这里只是返回新DataFrame，并没有修改原DataFrame
```

Out[102]:

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

3.8.2 df.pop() + df.insert()

- 将要移动位置的列用DataFrame.pop()删除，弹出的该列series赋值到新变量。
- 然后用DataFrame.insert()将弹出来的这列指定位置插回原DataFrame。
- 这种做法当然会修改原DataFrame。

比如我们想要将new_infor表中的Sex列换在原表的第二列的位置中，即“Hero Name”后面：

In [106]:
```
new_infor=pd.read_csv("new_infor.csv",index_col="Unnamed: 0")
new_infor
```

Out[106]:

	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

先将弹出的一列重新赋值到新变量中：

In [107]:
```
Sex=new_infor.pop("Sex")
```

然后重新插入到原列中：

In [108]:
```
new_infor.insert(1,"性别",Sex)
new_infor
```

Out[108]:

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

4 索引和列名的修改

在使用 DataFrame 的过程中，经常会遇到修改列名，修改索引名、修改索引等情况。使用 rename 轻松可以实现。

- 修改索引名用需要在df.rename()中设置index参数。
- 修改列名只需要设置参数 columns。

In [125]:
```
new_infor=pd.read_csv("new_infor.csv",index_col="Unnamed: 0")
new_infor
```

Out[125]:

	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	弑神者
7	黑寡妇	35	女	斯大林格勒	枪械

4.1 修改索引名

我们首先讲最为简单的修改索引名（记住，是索引名而不是索引标签），将上面的索引名从无改为字符串“索引”：

In [126]:
```
new_infor.index.name="索引"

new_infor
```

Out[126]:

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	弑神者
7	黑寡妇	35	女	斯大林格勒	枪械

4.2 修改列名

修改列名只需要设置参数 columns。

In [127]:
```
new_infor.columns.name="角色属性"
new_infor
```

Out[127]:

角色属性 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	弑神者
7	黑寡妇	35	女	斯大林格勒	枪械

4.3 df.rename()

使用df.rename()将整个索引标签替换掉，只需要建立一个字典：

- 该字典的“键”对应着“旧索引”
- 该字典的“值”对应着“新索引”

In [131]:
```
infor00=pd.read_csv("infor00.csv",index_col="英雄姓名")
infor00
```

Out[131]:

英雄姓名	年龄	出生地	血型
蜘蛛侠	19	纽约	NaN
灭霸	3000	泰坦星球	NaN
奇异博士	30	费城	NaN
钢铁侠	37	纽约	NaN
蝙蝠侠	40	哥谭	NaN
索尔	1500	阿斯加德	NaN

In [132]:
```
a=['蜘蛛侠', '灭霸', '奇异博士', '钢铁侠', '蝙蝠侠', '索尔']
b=["荷兰弟","乔什·布洛林","本尼迪克特·康伯巴奇","小罗伯特·唐尼","本·阿弗莱克","克里斯·海姆斯
c=dict(zip(a,b))

infor00=infor00.rename(index=c)    #将行索引改为上面字典的映射关系
infor00.index.name="演员"  #添加行索引名
infor00
```

Out[132]:

演员	年龄	出生地	血型
荷兰弟	19	纽约	NaN
乔什·布洛林	3000	泰坦星球	NaN
本尼迪克特·康伯巴奇	30	费城	NaN
小罗伯特·唐尼	37	纽约	NaN
本·阿弗莱克	40	哥谭	NaN
克里斯·海姆斯沃斯	1500	阿斯加德	NaN

同样地，df.rename()也可以通过字典映射的方法，修改DataFrame的列表标签：

In [133]:
```
c={"年龄":"Age","出生地":"Birthplace","血型":" blood group"}

infor00=infor00.rename(columns=c)
infor00
```

Out[133]:

演员	Age	Birthplace	blood group
荷兰弟	19	纽约	NaN
乔什·布洛林	3000	泰坦星球	NaN
本尼迪克特·康伯巴奇	30	费城	NaN
小罗伯特·唐尼	37	纽约	NaN
本·阿弗莱克	40	哥谭	NaN
克里斯·海姆斯沃斯	1500	阿斯加德	NaN

5 单列数据类型转换

如果想要转换数据类型的话，可以通过 astype 来完成。

5.1 Series.astype()

比如，想将年龄的数据类型从"int64"转为"float":

In [134]:
```
infor00.Age
```

Out[134]:
```
演员
荷兰弟                19
乔什·布洛林          3000
本尼迪克特·康伯巴奇     30
小罗伯特·唐尼         37
本·阿弗莱克           40
克里斯·海姆斯沃斯     1500
Name: Age, dtype: int64
```

In [135]:
```
infor00.Age = infor00.Age.astype(float)  #用dtype就要写成"float64",记得加双引号
infor00.info()
infor00
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6 entries, 荷兰弟 to 克里斯·海姆斯沃斯
Data columns (total 3 columns):
Age             6 non-null float64
Birthplace      6 non-null object
 blood group    0 non-null float64
dtypes: float64(2), object(1)
memory usage: 352.0+ bytes
```

Out[135]:

演员	Age	Birthplace	blood group
荷兰弟	19.0	纽约	NaN
乔什·布洛林	3000.0	泰坦星球	NaN
本尼迪克特·康伯巴奇	30.0	费城	NaN
小罗伯特·唐尼	37.0	纽约	NaN
本·阿弗莱克	40.0	哥谭	NaN
克里斯·海姆斯沃斯	1500.0	阿斯加德	NaN

5.2 Series.to_numeric()

有时候会涉及到将 object 类型转为其他类型，常见的有转为数字、日期、时间差。
Pandas 中分别对应 to_numeric、to_datetime、to_timedelta 方法。
比如，如果我们给infor00增加了一列新的字段“身高”：

In [137]:
```
infor00["Height"] = ["175", "270", "178", "177", "185", "188"]
infor00
```

Out[137]:

演员	Age	Birthplace	blood group	Height
荷兰弟	19.0	纽约	NaN	175
乔什·布洛林	3000.0	泰坦星球	NaN	270
本尼迪克特·康伯巴奇	30.0	费城	NaN	178
小罗伯特·唐尼	37.0	纽约	NaN	177
本·阿弗莱克	40.0	哥谭	NaN	185
克里斯·海姆斯沃斯	1500.0	阿斯加德	NaN	188

这时候就可以用Series.to_numeric()方法：

In [138]:
```
pd.to_numeric(infor00.Height)
```

Out[138]:
```
演员
荷兰弟              175
乔什·布洛林         270
本尼迪克特·康伯巴奇     178
小罗伯特·唐尼         177
本·阿弗莱克          185
克里斯·海姆斯沃斯       188
Name: Height, dtype: int64
```

注意，此时，pd.to_numeric()还没有将infor00.Height转为为数值型：

In [139]:
```
infor00.Height
```

Out[139]:
```
演员
荷兰弟              175
乔什·布洛林         270
本尼迪克特·康伯巴奇     178
小罗伯特·唐尼         177
本·阿弗莱克          185
克里斯·海姆斯沃斯       188
Name: Height, dtype: object
```

要通过重新赋值的方式，将转换之后的Series重新赋值到infor00.Height中：

In [140]:
```
infor00["Height"] = pd.to_numeric(infor00.Height)
infor00
```

Out[140]:

演员	Age	Birthplace	blood group	Height
荷兰弟	19.0	纽约	NaN	175
乔什·布洛林	3000.0	泰坦星球	NaN	270
本尼迪克特·康伯巴奇	30.0	费城	NaN	178
小罗伯特·唐尼	37.0	纽约	NaN	177
本·阿弗莱克	40.0	哥谭	NaN	185
克里斯·海姆斯沃斯	1500.0	阿斯加德	NaN	188

这样子就可以对身高字段进行统计描述：

In [142]:
```
infor00["Height"].describe()
```

Out[142]:
```
count      6.000000
mean     195.500000
std       36.838838
min      175.000000
25%      177.250000
50%      181.500000
75%      187.250000
max      270.000000
Name: Height, dtype: float64
```

如果身高里面多了一些"cm"的字符，使用Series.to_numeric()会转换失败：

In [144]:
```
infor00["Height"] = ["175", "270cm", "178cm", "177", "185", "188cm"]
infor00
```

Out[144]:

演员	Age	Birthplace	blood group	Height
荷兰弟	19.0	纽约	NaN	175
乔什·布洛林	3000.0	泰坦星球	NaN	270cm
本尼迪克特·康伯巴奇	30.0	费城	NaN	178cm
小罗伯特·唐尼	37.0	纽约	NaN	177
本·阿弗莱克	40.0	哥谭	NaN	185
克里斯·海姆斯沃斯	1500.0	阿斯加德	NaN	188cm

In [145]:
```
infor00["Height"] = pd.to_numeric(infor00.Height)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
pandas/_libs/src\inference.pyx in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string "270cm"

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
<ipython-input-145-2bd6245d230e> in <module>
----> 1 infor00["Height"] = pd.to_numeric(infor00.Height)

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\tools\numeric.py in to_numeric(arg, e
rrors, downcast)
    131             coerce_numeric = False if errors in ('ignore', 'raise') else True
    132             values = lib.maybe_convert_numeric(values, set(),
--> 133                                                coerce_numeric=coerce_numeric)
    134
    135     except Exception:

pandas/_libs/src\inference.pyx in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string "270cm" at position 1
```

这时候可以通过Series.to_numeric()方法里面errors参数的设定来进行区分处理,errors可以设置参数'ignore', 'raise', 'coerce':

- 如果'raise'，则无效的解析将引发异常。
- 如果'coerce'，则无效解析将被设置为NaN。
- 如果'ignore'，则无效的解析将返回输入。

In [147]:
```
infor00["Height"] = ["175", "270cm", "178cm", "177", "185", "188cm"]
infor00["Height"] = pd.to_numeric(infor00.Height,errors="coerce")
infor00
```

Out[147]:

演员	Age	Birthplace	blood group	Height
荷兰弟	19.0	纽约	NaN	175.0
乔什·布洛林	3000.0	泰坦星球	NaN	NaN
本尼迪克特·康伯巴奇	30.0	费城	NaN	NaN
小罗伯特·唐尼	37.0	纽约	NaN	177.0
本·阿弗莱克	40.0	哥谭	NaN	185.0
克里斯·海姆斯沃斯	1500.0	阿斯加德	NaN	NaN

In [148]:
```
infor00["Height"] = ["175", "270cm", "178cm", "177", "185", "188cm"]
infor00["Height"] = pd.to_numeric(infor00.Height,errors="ignore")
infor00
```

Out[148]:

演员	Age	Birthplace	blood group	Height
荷兰弟	19.0	纽约	NaN	175
乔什·布洛林	3000.0	泰坦星球	NaN	270cm
本尼迪克特·康伯巴奇	30.0	费城	NaN	178cm
小罗伯特·唐尼	37.0	纽约	NaN	177
本·阿弗莱克	40.0	哥谭	NaN	185
克里斯·海姆斯沃斯	1500.0	阿斯加德	NaN	188cm

6 表合并方式

In [149]:
```
user_infor=pd.read_csv("user_infor01",index_col="演员")
user_infor
```

Out[149]:

演员	Hero Name	Age	Sex	Birthplace
荷兰弟	蜘蛛侠	19	男	纽约
乔什·布洛林	灭霸	3000	男	泰坦星球
本尼迪克特·康伯巴奇	奇异博士	36	男	费城
小罗伯特·唐尼	钢铁侠	42	男	纽约
本·阿弗莱克	蝙蝠侠	40	男	哥谭
克里斯·海姆斯沃斯	索尔	1505	男	阿斯加德
盖尔·加朵	神奇女侠	2000	女	天堂岛

6.1 df.append(df)

In [150]:
```
user_infor.append(user_infor)
```

Out[150]:

演员	Hero Name	Age	Sex	Birthplace
荷兰弟	蜘蛛侠	19	男	纽约
乔什·布洛林	灭霸	3000	男	泰坦星球
本尼迪克特·康伯巴奇	奇异博士	36	男	费城
小罗伯特·唐尼	钢铁侠	42	男	纽约
本·阿弗莱克	蝙蝠侠	40	男	哥谭
克里斯·海姆斯沃斯	索尔	1505	男	阿斯加德
盖尔·加朵	神奇女侠	2000	女	天堂岛
荷兰弟	蜘蛛侠	19	男	纽约
乔什·布洛林	灭霸	3000	男	泰坦星球
本尼迪克特·康伯巴奇	奇异博士	36	男	费城
小罗伯特·唐尼	钢铁侠	42	男	纽约
本·阿弗莱克	蝙蝠侠	40	男	哥谭
克里斯·海姆斯沃斯	索尔	1505	男	阿斯加德
盖尔·加朵	神奇女侠	2000	女	天堂岛

6.2 pd.concat([df_01,df_02])

In [151]:
```
pd.concat([user_infor,user_infor],axis=0)
```

Out[151]:

演员	Hero Name	Age	Sex	Birthplace
荷兰弟	蜘蛛侠	19	男	纽约
乔什·布洛林	灭霸	3000	男	泰坦星球
本尼迪克特·康伯巴奇	奇异博士	36	男	费城
小罗伯特·唐尼	钢铁侠	42	男	纽约
本·阿弗莱克	蝙蝠侠	40	男	哥谭
克里斯·海姆斯沃斯	索尔	1505	男	阿斯加德
盖尔·加朵	神奇女侠	2000	女	天堂岛
荷兰弟	蜘蛛侠	19	男	纽约
乔什·布洛林	灭霸	3000	男	泰坦星球
本尼迪克特·康伯巴奇	奇异博士	36	男	费城
小罗伯特·唐尼	钢铁侠	42	男	纽约
本·阿弗莱克	蝙蝠侠	40	男	哥谭
克里斯·海姆斯沃斯	索尔	1505	男	阿斯加德
盖尔·加朵	神奇女侠	2000	女	天堂岛

In [152]:
```
pd.concat([user_infor,user_infor],axis=1)
```

Out[152]:

演员	Hero Name	Age	Sex	Birthplace	Hero Name	Age	Sex	Birthplace
荷兰弟	蜘蛛侠	19	男	纽约	蜘蛛侠	19	男	纽约
乔什·布洛林	灭霸	3000	男	泰坦星球	灭霸	3000	男	泰坦星球
本尼迪克特·康伯巴奇	奇异博士	36	男	费城	奇异博士	36	男	费城
小罗伯特·唐尼	钢铁侠	42	男	纽约	钢铁侠	42	男	纽约
本·阿弗莱克	蝙蝠侠	40	男	哥谭	蝙蝠侠	40	男	哥谭
克里斯·海姆斯沃斯	索尔	1505	男	阿斯加德	索尔	1505	男	阿斯加德
盖尔·加朵	神奇女侠	2000	女	天堂岛	神奇女侠	2000	女	天堂岛

6.3 pd.merge()

pd.merge('left', 'right', "how='inner'", 'on=None', 'left_on=None', 'right_on=None)-

- how:
  - left：仅使用左框架中的键，类似于SQL左外连接;保留关键顺序
  - right：仅使用右框架中的键，类似于SQL右外连接;保留关键顺序
  - outer：使用来自两个帧的键的并集，类似于SQL full outer加入;按字典顺序排序键
  - inner：使用两个帧的交集，类似于SQL内部加入;保留左键的顺序

我们先建立两个表：user_info_01、user_info_02

In [155]:
```
user_info_01=pd.read_csv("user_infor01",index_col="演员")
user_info_01
```

Out[155]:

演员	Hero Name	Age	Sex	Birthplace
荷兰弟	蜘蛛侠	19	男	纽约
乔什·布洛林	灭霸	3000	男	泰坦星球
本尼迪克特·康伯巴奇	奇异博士	36	男	费城
小罗伯特·唐尼	钢铁侠	42	男	纽约
本·阿弗莱克	蝙蝠侠	40	男	哥谭
克里斯·海姆斯沃斯	索尔	1505	男	阿斯加德
盖尔·加朵	神奇女侠	2000	女	天堂岛

In [156]:
```
data={"英雄名":["灭霸","奇异博士","钢铁侠","蝙蝠侠","索尔","神奇女侠","蜘蛛侠","黑寡妇"],
  "weapon":["暴君屠刀","魔法","纳米战甲","有钱","暴风战斧","弒神者","蜘蛛感应","枪械"]}

user_info_02=pd.DataFrame(data=data)
user_info_02
```

Out[156]:

	英雄名	weapon
0	灭霸	暴君屠刀
1	奇异博士	魔法
2	钢铁侠	纳米战甲
3	蝙蝠侠	有钱
4	索尔	暴风战斧
5	神奇女侠	弒神者
6	蜘蛛侠	蜘蛛感应
7	黑寡妇	枪械

6.3.1 内连接

user_info_01

演员	Hero Name	Age	Sex	Birthplace
荷兰弟	蜘蛛侠	19	男	纽约
乔什·布洛林	灭霸	3000	男	泰坦星球
本尼迪克特·康伯巴奇	奇异博士	36	男	费城
小罗伯特·唐尼	钢铁侠	42	男	纽约
本·阿弗莱克	蝙蝠侠	40	男	哥谭
克里斯·海姆斯沃斯	索尔	1505	男	阿斯加德
盖尔·加朵	神奇女侠	2000	女	天堂岛

user_info_02

	英雄名	weapon
0	灭霸	暴君屠刀
1	奇异博士	魔法
2	钢铁侠	纳米战甲
3	蝙蝠侠	有钱
4	索尔	暴风战斧
5	神奇女侠	弑神者
6	蜘蛛侠	蜘蛛感应
7	黑寡妇	枪械

In [157]:
```
#内连接：how="inner"
pd.merge(user_info_01,user_info_02,how="inner",left_on="Hero Name",right_on="英雄名")
```

Out[157]:

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

6.3.2 左连接

user_info_01

演员	Hero Name	Age	Sex	Birthplace
荷兰弟	蜘蛛侠	19	男	纽约
乔什·布洛林	灭霸	3000	男	泰坦星球
本尼迪克特·康伯巴奇	奇异博士	36	男	费城
小罗伯特·唐尼	钢铁侠	42	男	纽约
本·阿弗莱克	蝙蝠侠	40	男	哥谭
克里斯·海姆斯沃斯	索尔	1505	男	阿斯加德
盖尔·加朵	神奇女侠	2000	女	天堂岛

user_info_02

	英雄名	weapon
0	灭霸	暴君屠刀
1	奇异博士	魔法
2	钢铁侠	纳米战甲
3	蝙蝠侠	有钱
4	索尔	暴风战斧
5	神奇女侠	弑神者
6	蜘蛛侠	蜘蛛感应
7	黑寡妇	枪械

In [158]:
```
#左连接how="left"
pd.merge(user_info_01,user_info_02,how="left",left_on="Hero Name",right_on="英雄名")
```

Out[158]:

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

6.3.3 右连接

user_info_01

演员	Hero Name	Age	Sex	Birthplace
荷兰弟	蜘蛛侠	19	男	纽约
乔什·布洛林	灭霸	3000	男	泰坦星球
本尼迪克特·康伯巴奇	奇异博士	36	男	费城
小罗伯特·唐尼	钢铁侠	42	男	纽约
本·阿弗莱克	蝙蝠侠	40	男	哥谭
克里斯·海姆斯沃斯	索尔	1505	男	阿斯加德
盖尔·加朵	神奇女侠	2000	女	天堂岛

user_info_02

	英雄名	weapon
0	灭霸	暴君屠刀
1	奇异博士	魔法
2	钢铁侠	纳米战甲
3	蝙蝠侠	有钱
4	索尔	暴风战斧
5	神奇女侠	弑神者
6	蜘蛛侠	蜘蛛感应
7	黑寡妇	枪械

In [159]:
```
#右连接how="right"
pd.merge(user_info_01,user_info_02,how="right",left_on="Hero Name",right_on="英雄名")
```

Out[159]:

	Hero Name	Age	Sex	Birthplace	英雄名	weapon
0	蜘蛛侠	19.0	男	纽约	蜘蛛侠	蜘蛛感应
1	灭霸	3000.0	男	泰坦星球	灭霸	暴君屠刀
2	奇异博士	36.0	男	费城	奇异博士	魔法
3	钢铁侠	42.0	男	纽约	钢铁侠	纳米战甲
4	蝙蝠侠	40.0	男	哥谭	蝙蝠侠	有钱
5	索尔	1505.0	男	阿斯加德	索尔	暴风战斧
6	神奇女侠	2000.0	女	天堂岛	神奇女侠	弑神者
7	NaN	NaN	NaN	NaN	黑寡妇	枪械

7 数据保存与读取

In [160]:
```
new_infor.to_csv("new_infor.csv")
```

In [161]:
```
pd.read_csv("new_infor.csv")
```

Out[161]:

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

原本的new_infor中的索引在读取的时候变成了新的一列字段，该如何读取才能避免这种状况呢?

In [163]:
```
pd.read_csv("new_infor.csv",index_col="索引")
```

Out[163]:

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

如果数据有索引，但是没有索引名，导入进来后该索引会作为新的一列，导入的时候设置好参数index_col="Unnamed: 0"即可。