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

pandas的拼接操作

pandas的拼接分为两种：

- 级联：pd.concat, pd.append
- 合并：pd.merge, pd.join

1. 使用pd.concat()级联

pandas使用pd.concat函数，与np.concatenate函数类似，只是多了一些参数：

```
objs
axis=0
keys
join='outer' / 'inner':表示的是级联的方式，outer会将所有的项进行级联（忽略匹配和不匹配），而inner只会将匹配的项级联到一起，不匹配的不级联
ignore_index=False
```

1)匹配级联

In [2]:
```
df1 = DataFrame(data=np.random.randint(0,100,size=(3,3)),index=['a','b','c'])
df2 = DataFrame(data=np.random.randint(0,100,size=(3,3)),index=['a','b','d'])
```

In [3]:
```
pd.concat([df1,df2],axis=0)
```

Out[3]:

	0	1	2
a	2	21	48
b	35	75	73
c	73	33	35
a	18	45	83
b	1	98	37
d	64	6	60

不匹配指的是级联的维度的索引不一致。例如纵向级联时列索引不一致，横向级联时行索引不一致 有2种连接方式：

- 外连接：补NaN（默认模式）
- 内连接：只连接匹配的项

In [4]:
```
pd.concat([df1,df2],axis=1,join='outer')  #默认outer
```

```
c:\users\administrator\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:1: FutureWarning: Sorting bec
ause non-concatenation axis is not aligned. A future version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  """Entry point for launching an IPython kernel.
```

Out[4]:

	0	1	2	0	1	2
a	2.0	21.0	48.0	18.0	45.0	83.0
b	35.0	75.0	73.0	1.0	98.0	37.0
c	73.0	33.0	35.0	NaN	NaN	NaN
d	NaN	NaN	NaN	64.0	6.0	60.0

In [5]:
```
pd.concat([df1,df2],axis=1,join='inner')
```

Out[5]:

	0	1	2	0	1	2
a	2	21	48	18	45	83
b	35	75	73	1	98	37

3) 使用df.append()函数添加

由于在后面级联的使用非常普遍，因此有一个函数append专门用于在后面添加

In [6]:
```
df1.append(df2)
```

Out[6]:

	0	1	2
a	2	21	48
b	35	75	73
c	73	33	35
a	18	45	83
b	1	98	37
d	64	6	60

2. 使用pd.merge()合并

merge与concat的区别在于，merge需要依据某一共同的列来进行合并

使用pd.merge()合并时，会自动根据两者相同column名称的那一列，作为key来进行合并。

注意每一列元素的顺序不要求一致

参数：

- how：out取并集 inner取交集
- on：当有多列相同的时候，可以使用on来指定使用那一列进行合并，on的值为一个列表

1) 一对一合并

In [7]:
```
df1 = DataFrame({'employee':['Bob','Jake','Lisa'],
                 'group':['Accounting','Engineering','Engineering'],
                 })
df1
```

Out[7]:

	employee	group
0	Bob	Accounting
1	Jake	Engineering
2	Lisa	Engineering

In [8]:
```
df2 = DataFrame({'employee':['Lisa','Bob','Jake'],
                 'hire_date':[2004,2008,2012],
                 })
df2
```

Out[8]:

	employee	hire_date
0	Lisa	2004
1	Bob	2008
2	Jake	2012

In [9]:
```
pd.merge(df1,df2)
```

Out[9]:

	employee	group	hire_date
0	Bob	Accounting	2008
1	Jake	Engineering	2012
2	Lisa	Engineering	2004

2) 多对一合并

In [10]:
```
df3 = DataFrame({
    'employee':['Lisa','Jake'],
    'group':['Accounting','Engineering'],
    'hire_date':[2004,2016]})
df3
```

Out[10]:

	employee	group	hire_date
0	Lisa	Accounting	2004
1	Jake	Engineering	2016

In [11]:
```
df4 = DataFrame({'group':['Accounting','Engineering','Engineering'],
                 'supervisor':['Carly','Guido','Steve']
                })
df4
```

Out[11]:

	group	supervisor
0	Accounting	Carly
1	Engineering	Guido
2	Engineering	Steve

In [12]:
```
pd.merge(df3,df4,how='outer')
```

Out[12]:

	employee	group	hire_date	supervisor
0	Lisa	Accounting	2004	Carly
1	Jake	Engineering	2016	Guido
2	Jake	Engineering	2016	Steve

3) 多对多合并

In [13]:
```
df1 = DataFrame({'employee':['Bob','Jake','Lisa'],
                 'group':['Accounting','Engineering','Engineering']})
df1
```

Out[13]:

	employee	group
0	Bob	Accounting
1	Jake	Engineering
2	Lisa	Engineering

In [14]:
```
df5 = DataFrame({'group':['Engineering','Engineering','HR'],
                 'supervisor':['Carly','Guido','Steve']
                 })
df5
```

Out[14]:

	group	supervisor
0	Engineering	Carly
1	Engineering	Guido
2	HR	Steve

In [15]:
```
pd.merge(df1,df5,how='left')
```

Out[15]:

	employee	group	supervisor
0	Bob	Accounting	NaN
1	Jake	Engineering	Carly
2	Jake	Engineering	Guido
3	Lisa	Engineering	Carly
4	Lisa	Engineering	Guido

- 加载excl数据:pd.read_excel('excl_path',sheetname=1)

```
df = pd.read_excel('./data.xlsx',sheet_name=1)
```

4) key的规范化

- 当列冲突时，即有多个列名称相同时，需要使用on=来指定哪一个列作为key，配合suffixes指定冲突列名

In [16]:
```
df1 = DataFrame({'employee':['Jack',"Summer","Steve"],
                 'group':['Accounting','Finance','Marketing']})
df1
```

Out[16]:

	employee	group
0	Jack	Accounting
1	Summer	Finance
2	Steve	Marketing

In [17]:
```
df2 = DataFrame({'employee':['Jack','Bob',"Jake"],
                 'hire_date':[2003,2009,2012],
                'group':['Accounting','sell','ceo']})
df2
```

Out[17]:

	employee	hire_date	group
0	Jack	2003	Accounting
1	Bob	2009	sell
2	Jake	2012	ceo

In [18]:
```
pd.merge(df1,df2,on='group',how='outer')
```

Out[18]:

	employee_x	group	employee_y	hire_date
0	Jack	Accounting	Jack	2003.0
1	Summer	Finance	NaN	NaN
2	Steve	Marketing	NaN	NaN
3	NaN	sell	Bob	2009.0
4	NaN	ceo	Jake	2012.0

- 当两张表没有可进行连接的列时，可使用left_on和right_on手动指定merge中左右两边的哪一列列作为连接的列

In [19]:
```
df1 = DataFrame({'employee':['Bobs','Linda','Bill'],
                 'group':['Accounting','Product','Marketing'],
                'hire_date':[1998,2017,2018]})
df1
```

Out[19]:

	employee	group	hire_date
0	Bobs	Accounting	1998
1	Linda	Product	2017
2	Bill	Marketing	2018

In [20]:
```
df5 = DataFrame({'name':['Lisa','Bobs','Bill'],
                 'hire_dates':[1998,2016,2007]})
df5
```

Out[20]:

	name	hire_dates
0	Lisa	1998
1	Bobs	2016
2	Bill	2007

In [21]:
```
pd.merge(df1,df5,left_on='employee',right_on='name',how='outer')
```

Out[21]:

	employee	group	hire_date	name	hire_dates
0	Bobs	Accounting	1998.0	Bobs	2016.0
1	Linda	Product	2017.0	NaN	NaN
2	Bill	Marketing	2018.0	Bill	2007.0
3	NaN	NaN	NaN	Lisa	1998.0

==========

练习:

1. 自行练习多对一，多对多的情况
2. 自学left_index,right_index

==========

5) 内合并与外合并:out取并集 inner取交集

- 内合并：只保留两者都有的key（默认模式）

In [22]:
```
df6 = DataFrame({'name':['Peter','Paul','Mary'],
                 'food':['fish','beans','bread']}
                )
df7 = DataFrame({'name':['Mary','Joseph'],
                 'drink':['wine','beer']})
pd.merge(df6,df7,how='inner')
```

Out[22]:

	name	food	drink
0	Mary	bread	wine

In [23]:
```
df6 = DataFrame({'name':['Peter','Paul','Mary'],
                 'food':['fish','beans','bread']}
                )
df7 = DataFrame({'name':['Mary','Joseph'],
                 'drink':['wine','beer']})
pd.merge(df6,df7,how='outer')
```

Out[23]:

	name	food	drink
0	Peter	fish	NaN
1	Paul	beans	NaN
2	Mary	bread	wine
3	Joseph	NaN	beer

In [ ]:
```

```