Python数据分析实战

第二十一课 pandas表格匹配与拼接

本节课程目标

- 级联：concat,append
- 合并：merge, join

```
import pandas as pd
df1 = pd.DataFrame({'apts': [55000, 60000],
                    'cars': [200000, 300000],},
                   index = ['Shanghai', 'Beijing'])
print(df1)
```

```
           apts    cars
Shanghai  55000  200000
Beijing   60000  300000
```

```
df2 = pd.DataFrame({'cars': [150000, 120000],
                    'apts': [25000, 20000],
                   },
                   index = ['Hangzhou', 'Najing'])
print(df2)
```

```
            cars   apts
Hangzhou  150000  25000
Najing    120000  20000
```

```
df3 = pd.DataFrame({'apts': [30000, 10000],
                    'cars': [180000, 100000],},
                   index = ['Guangzhou', 'Chongqing'])
print(df3)
```

```
            apts    cars
Guangzhou  30000  180000
Chongqing  10000  100000
```

concatenate

```
#先定义一个列表，将上面的3张表格存起来
frames = [df1, df2, df3]
#然后使用concatenate函数实现拼接
result = pd.concat(frames, sort=False)
print(result)
```

```
            apts    cars
Shanghai   55000  200000
Beijing    60000  300000
Hangzhou   25000  150000
Najing     20000  120000
Guangzhou  30000  180000
Chongqing  10000  100000
```

```
result2 = pd.concat(frames, keys=['df1', 'df2', 'df3'], sort=False)
print(result2)
```

```
               apts    cars
df1 Shanghai   55000  200000
    Beijing    60000  300000
df2 Hangzhou   25000  150000
    Najing     20000  120000
df3 Guangzhou  30000  180000
    Chongqing  10000  100000
```

```
result2.loc["df3"]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	apts	cars
Guangzhou	30000	180000
Chongqing	10000	100000

```
#比如再来创建一个dateFrame
df4 = pd.DataFrame({'salaries': [10000, 30000, 30000, 20000, 15000]},
                   index = ['Suzhou', 'Beijing', 'Shanghai', 'Guangzhou', 'Tianjin'])
print(df4)
```

```
           salaries
Suzhou        10000
Beijing       30000
Shanghai      30000
Guangzhou     20000
Tianjin       15000
```

```
#按照列来实现两个表格的拼接
result3 = pd.concat([result, df4], axis=1, sort=False)
print(result3)
```

```
              apts      cars  salaries
Shanghai   55000.0  200000.0   30000.0
Beijing    60000.0  300000.0   30000.0
Hangzhou   25000.0  150000.0       NaN
Najing     20000.0  120000.0       NaN
Guangzhou  30000.0  180000.0   20000.0
Chongqing  10000.0  100000.0       NaN
Suzhou         NaN       NaN   10000.0
Tianjin        NaN       NaN   15000.0
```

```
result3 = pd.concat([result, df4], axis=1, join='inner')#inner,表示只取index重合的部分
print(result3)
```

```
            apts    cars  salaries
Shanghai   55000  200000     30000
Beijing    60000  300000     30000
Guangzhou  30000  180000     20000
```

append

```
df1.append(df2, sort=False)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	apts	cars
Shanghai	55000	200000
Beijing	60000	300000
Hangzhou	25000	150000
Najing	20000	120000

```
df1.append(df4, sort=False)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	apts	cars	salaries
Shanghai	55000.0	200000.0	NaN
Beijing	60000.0	300000.0	NaN
Suzhou	NaN	NaN	10000.0
Beijing	NaN	NaN	30000.0
Shanghai	NaN	NaN	30000.0
Guangzhou	NaN	NaN	20000.0
Tianjin	NaN	NaN	15000.0

```
#创建一个Series
s1 = pd.Series([60, 50], index=['Shanghai', 'Beijing'], name='meal')
print(s1)
```

```
Shanghai    60
Beijing     50
Name: meal, dtype: int64
```

```
print(df1)
```

```
           apts    cars
Shanghai  55000  200000
Beijing   60000  300000
```

```
#将Series和Dateframe进行级联
print(pd.concat([df1, s1], axis=1))#axis=1表示列
```

```
           apts    cars  meal
Shanghai  55000  200000    60
Beijing   60000  300000    50
```

```
#append一个row到DataFrame里
s2 = pd.Series([18000, 12000], index=['apts', 'cars'], name='Xiamen') #注意这里的name是必须要有的，因为要用作Index。
print(s2)
```

```
print(df1.append(s2))#默认表示行的级联
```

Merge(Join)

```
df1 = pd.DataFrame({'apts': [55000, 60000, 58000],
                    'cars': [200000, 300000,250000],
                   'city': ['Shanghai', 'Beijing','Shenzhen']})
print(df1)
```

```
    apts    cars      city
0  55000  200000  Shanghai
1  60000  300000   Beijing
2  58000  250000  Shenzhen
```

```
df4 = pd.DataFrame({'salaries': [10000, 30000, 30000, 20000, 15000],
                   'city': ['Suzhou', 'Beijing', 'Shanghai', 'Guangzhou', 'Tianjin']})
print(df4)
```

```
   salaries       city
0     10000     Suzhou
1     30000    Beijing
2     30000   Shanghai
3     20000  Guangzhou
4     15000    Tianjin
```

```
result = pd.merge(df1, df4, on='city')#只有上海和北京
result
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	apts	cars	city	salaries
0	55000	200000	Shanghai	30000
1	60000	300000	Beijing	30000

```
result = pd.merge(df1, df4)
result
```


```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```


	apts	cars	city	salaries
0	55000	200000	Shanghai	30000
1	60000	300000	Beijing	30000

```
#outer-->不匹配也可以留下来
result = pd.merge(df1, df4, on='city', how='outer')
print(result)
```

```
      apts      cars       city  salaries
0  55000.0  200000.0   Shanghai   30000.0
1  60000.0  300000.0    Beijing   30000.0
2  58000.0  250000.0   Shenzhen       NaN
3      NaN       NaN     Suzhou   10000.0
4      NaN       NaN  Guangzhou   20000.0
5      NaN       NaN    Tianjin   15000.0
```

```
#right以右边的表格为准，留下右边的
result = pd.merge(df1, df4, on='city', how='right')
print(result)
```

```
      apts      cars       city  salaries
0  55000.0  200000.0   Shanghai     30000
1  60000.0  300000.0    Beijing     30000
2      NaN       NaN     Suzhou     10000
3      NaN       NaN  Guangzhou     20000
4      NaN       NaN    Tianjin     15000
```

```
#left已左边的表格为准，留下左边的
result = pd.merge(df1, df4, on='city', how='left')
print(result)
```

```
    apts    cars      city  salaries
0  55000  200000  Shanghai   30000.0
1  60000  300000   Beijing   30000.0
2  58000  250000  Shenzhen       NaN
```

```
#也可以使用concat
pd.concat([df1.set_index("city"), df4.set_index('city')], sort=False, axis=1,
join="inner")
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	apts	cars	salaries
city			
Shanghai	55000	200000	30000
Beijing	60000	300000	30000

join

```
df1 = pd.DataFrame({'apts': [55000, 60000, 58000],
                    'cars': [200000, 300000,250000]},
                   index=['Shanghai', 'Beijing','Shenzhen'])
print(df1)
```

```
           apts    cars
Shanghai  55000  200000
Beijing   60000  300000
Shenzhen  58000  250000
```

```
df4 = pd.DataFrame({'salaries': [10000, 30000, 30000, 20000, 15000]},
                   index=['Suzhou', 'Beijing', 'Shanghai', 'Guangzhou', 'Tianjin'])
print(df4)
```

```
           salaries
Suzhou        10000
Beijing       30000
Shanghai      30000
Guangzhou     20000
Tianjin       15000
```

```
# 包含，df1的部分，对两者进行数据操作
print(df1.join(df4))
```

```
           apts    cars  salaries
Shanghai  55000  200000   30000.0
Beijing   60000  300000   30000.0
Shenzhen  58000  250000       NaN
```

```
#也可以用merge实现
pd.merge(df1, df4)
```

```
---------------------------------------------------------------------------

MergeError                                Traceback (most recent call last)

<ipython-input-48-cc2bfb0cc312> in <module>
      1 #也可以用merge实现
----> 2 pd.merge(df1, df4)
```

```
/anaconda3/lib/python3.7/site-packages/pandas/core/reshape/merge.py in merge(left,
right, how, on, left_on, right_on, left_index, right_index, sort, suffixes, copy,
indicator, validate)
     45                          right_index=right_index, sort=sort, suffixes=suffixes,
     46                          copy=copy, indicator=indicator,
---> 47                          validate=validate)
     48     return op.get_result()
     49
```

```
/anaconda3/lib/python3.7/site-packages/pandas/core/reshape/merge.py in __init__(self,
left, right, how, on, left_on, right_on, axis, left_index, right_index, sort, suffixes,
copy, indicator, validate)
    522             warnings.warn(msg, UserWarning)
    523
--> 524         self._validate_specification()
    525
    526         # note this function has side effects
```

```
/anaconda3/lib/python3.7/site-packages/pandas/core/reshape/merge.py in
_validate_specification(self)
   1031                          'left_index={lidx}, right_index={ridx}'
   1032                          .format(lon=self.left_on, ron=self.right_on,
-> 1033                                  lidx=self.left_index, ridx=self.right_index))
   1034                 if not common_cols.is_unique:
   1035                     raise MergeError("Data columns not unique: {common!r}"
```

```
MergeError: No common columns to perform merge on. Merge options: left_on=None,
right_on=None, left_index=False, right_index=False
```

```
pd.merge(df1, df4, left_index=True, right_index=True, how='outer')
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	apts	cars	salaries
Beijing	60000.0	300000.0	30000.0
Guangzhou	NaN	NaN	20000.0
Shanghai	55000.0	200000.0	30000.0
Shenzhen	58000.0	250000.0	NaN
Suzhou	NaN	NaN	10000.0
Tianjin	NaN	NaN	15000.0

```
pd.concat([df1, df4], sort=False, axis=1)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	apts	cars	salaries
Shanghai	55000.0	200000.0	30000.0
Beijing	60000.0	300000.0	30000.0
Shenzhen	58000.0	250000.0	NaN
Suzhou	NaN	NaN	10000.0
Guangzhou	NaN	NaN	20000.0
Tianjin	NaN	NaN	15000.0