x : 希望进行分段的变量列名称
bins : 具体的分段设定
int : 被等距等分的段数
sequence of scalars : 具体的每一个分段起点，必须包括最值，可不等距
right = True : 每段是否包括右侧界值
labels = None : 为每个分段提供自定义标签
include_lowest = False : 第一段是否包括最左侧界值，需要和right参数配合

) # 分段结果是数值类型为Categories的序列

pd.qcut() # 按照频数，而不是按照取值范围进行等分

In [ ]:

```
print(df2.head())
df2['cls'] = pd.cut(df2.名次, bins=[1,3,7], right= True, include_lowest = True)
df2.head(10)
```

6.6 实战：进一步整理PM2.5数据

要求:

在数据中剔除PM2.5为-900的案例（均为缺失数据）
建立一个新变量high，当PM2.5 >= 200时为1，否则为0
建立一个新变量high2，按照PM2.5 在100，200，500分为四段，分别取值0，1，2，3
将high2转换为哑变量组
按照50一个组段，将PM2.5数值转换为分段变量high3

7 文件级别的数据管理

7.1 数据拆分

7.1.1 标记数据拆分组

df.groupby(

by : 用于分组的变量名/函数
axis = 0 :
level = None : 相应的轴存在多重索引时，指定用于分组的级别
as_index = True : 在结果中将组标签作为索引
sort = True : 结果是否按照分组关键字进行排序

) # 生成的是分组索引标记，而不是新的DF

在数据分组之后，许多数据处理/分析/绘图命令都可以在各组间单独执行

In [ ]:

```
df2g = df2.groupby('类型')
df2g
```

In [ ]:

```
df2g.groups
```

In [ ]:

```
df2g.describe()
```

In [ ]:

```
df3g = df2.groupby(['类型', '所在省份'])
```

In [ ]:

```
df3g.mean() # 可以使用tab调用语法参考
```

7.1.2 基于拆分进行筛选

筛选出出其中的一组

dfgroup.get_group()

In [ ]:

```
df2g.get_group('农林').mean()
```

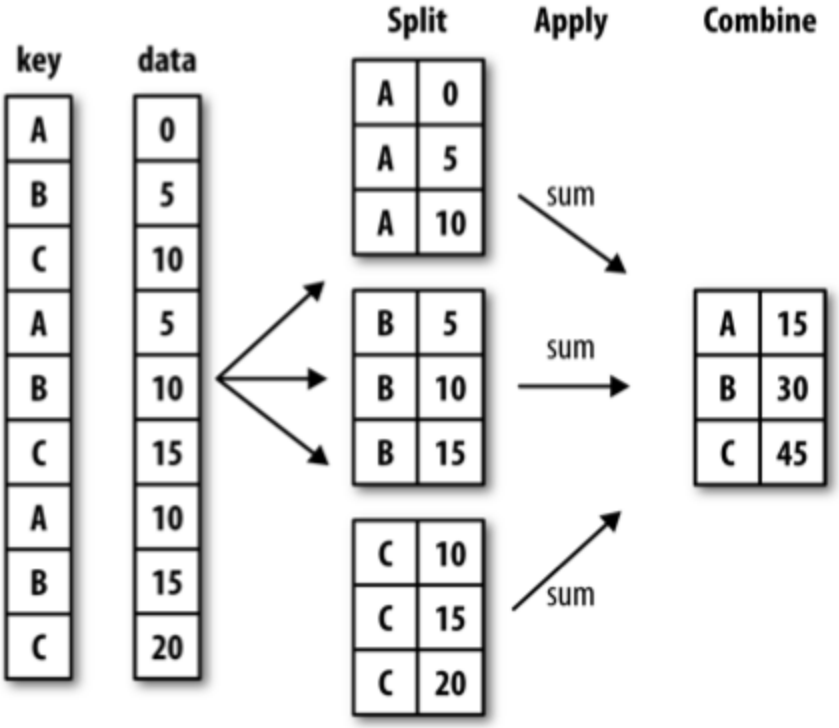
筛选出所需的列

该操作也适用于希望对不同的变量列进行不同操作时

In [ ]:

```
df2g['名次'].max()
```

7.2 分组汇总



在使用groupby完成数据拆分后，就可以按照需求分组进行信息汇总，此时可以使用其它专门的汇总命令，如agg来完成汇总操作

7.2.1 使用agg函数进行汇总

df.aggregate()

名称可以直接简写为agg
0.20版新增
可以用axis指定汇总维度

可以直接使用的汇总函数

```
count()    Number of non-null observations
size()    group sizes
sum()    Sum of values
mean()    Mean of values
median()    Arithmetic median of values
min()    Minimum
max()    Maximum
std()    Unbiased standard deviation
var()    Unbiased variance
skew()    Unbiased skewness (3rd moment)
kurt()    Unbiased kurtosis (4th moment)
quantile()    Sample quantile (value at %)
apply()    Generic apply
cov()    Unbiased covariance (binary)
corr()    Correlation (binary)
```

In [ ]:

```
df2g.agg('count')
```

In [ ]:

```
df2.agg('median')
```

In [ ]:

```
df2.agg(['mean', 'median'])
```

In [ ]:

```
df2g.agg(['mean', 'median'])
```

In [ ]:

```
# 引用非内置函数
import numpy as np
print(df2.总分.agg(np.sum))
df2g.总分.agg(np.sum)
```

In [ ]:

```
# 引用自定义函数
def mymean(x):
    return x.mean()
df2.总分.agg(mymean)
```

In [ ]:

```
df2g.agg(mymean)
```

7.2.2 其他分组汇总方法

在生成交叉表的同时对单元格指定具体的汇总指标和汇总函数

```
df.pivot_table()
pd.crosstab()
```

In [ ]:

```
pd.crosstab(df2.办学方向, df2.类型)
```

7.3 重复测量数据的格式转换

基于多重索引，Pandas可以很容易地完成长型、宽型数据格式的相互转换。

7.3.1 转换为最简格式

df.stack(

```
level = -1 : 需要处理的索引级别，默认为全部，int/string/list
dropna = True : 是否删除为缺失值的行
```

) # 转换后的结果可能为Series

In [ ]:

```
df3 = pd.read_excel("儿童生长研究.xlsx", index_col= [0, 2])
df3
```

In [ ]:

```
df3s = df3.stack()
df3s
```

7.3.2 长宽型格式的自由互转

df.unstack(

level = -1 : 需要处理的索引级别，默认为全部，int/string/list
fill_value : 用于填充缺失值的数值

)

In [ ]:

```
df3s.unstack(1)
```

In [ ]:

```
df3s.unstack([1,2])
```

数据转置

df.T

In [ ]:

```
df3.T
```

7.3.3 其他可用于格式转换的命令

df.melt()

df.pivot()

df.pivot_table()

7.4 多个数据源的合并

7.4.1 数据的纵向合并

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

df3

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

Result

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

df.append(

`other` : 希望添加的DF/Series/字典/上述对象的列表
使用列表方式，就可以实现一次合并多个新对象
`ignore_index = False` : 添加时是否忽略索引
`verify_integrity = False` : 是否检查索引值的唯一性，有重复时报错

)

In [ ]:

```
df21 = pd.read_excel("高校信息.xlsx", sheet_name = 'part1')
df22 = pd.read_excel("高校信息.xlsx", sheet_name = 'part2')
df21 = df21.sort_values('总分')
df2 = df21.append(df22) # 注意索引值的大小, ignore_index = True
df2
```

In [ ]:

```
df2 = df21.append([df22, df22[:10], df22[40:]]) # 用list实现多个数据源的合并
df2
```

7.4.2 数据的横向合并

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

Result

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	NaN	NaN	NaN	NaN	B6	D6	F6
7	NaN	NaN	NaN	NaN	B7	D7	F7

merge命令使用类SQL的连接方式

pd.merge(

需要合并的DF
left : 需要合并的左侧DF
right : 需要合并的右侧DF
how = 'inner' : 具体的连接类型
{'left', 'right', 'outer', 'inner'}
两个DF的连接方式
on : 用于连接两个DF的关键变量（多个时为列表），必须在两侧都出现
left_on : 左侧DF用于连接的关键变量（多个时为列表）
right_on : 右侧DF用于连接的关键变量（多个时为列表）

left_index = False : 是否将左侧DF的索引用于连接
right_index = False : 是否将右侧DF的索引用于连接
其他附加设定
sort = False : 是否在合并前按照关键变量排序（会影响合并后的案例顺序）
suffixes : 重名变量的处理方式，提供长度为2的列表元素，分别作为后缀
suffixes=('_x', '_y')
copy = True
indicator = False : 在结果DF中增加'_merge'列，用于记录数据来源
也可以直接提供相应的变量列名
Categorical类型，取值: 'left_only', 'right_only', 'both'
validate = None : 核查合并类型是否为所指定的情况
'one_to_one' or '1:1'
'one_to_many' or '1:m'
'many_to_one' or 'm:1'
'many_to_many' or 'm:m'（实际上不做检查）
0.21版新增

)

In [ ]:

```
df2a = pd.read_excel("高校信息.xlsx", sheet_name = 'var6')
df2b = pd.read_excel("高校信息.xlsx", sheet_name = 'var3')
pd.merge(df2a, df2b)
```

In [ ]:

```
pd.merge(df2a, df2b[:20])
```

In [ ]:

```
df2ai = df2a.set_index('学校名称')
df2bi = df2b.set_index('学校名称')
pd.merge(df2ai, df2bi, left_index = True, right_index = True)
```

In [ ]:

```
df2ai = df2a.set_index('学校名称')
df2bi = df2b
pd.merge(df2ai, df2bi, left_index = True, right_on = '学校名称' )
```

7.4.3 Concat命令简介

同时支持横向合并与纵向合并

pd.concat(

```
    objs : 需要合并的对象，列表形式提供
    axis = 0 : 对行还是对列方向进行合并
        {0/'index', 1/'columns'}
    join = 'outer' : 对另一个轴向的索引值如何进行处理
        {'inner', 'outer'}
    ignore_index = False
    keys = None : 为不同数据源的提供合并后的索引值
    verify_integrity = False
    copy = True
```

)

In [ ]:

```
# 纵向合并
df21 = pd.read_excel("高校信息.xlsx", sheet_name = 'part1')
df22 = pd.read_excel("高校信息.xlsx", sheet_name = 'part2')
pd.concat([df21, df22])
```

In [ ]:

```
pd.concat([df21, df22], keys = ['A', 'B'])
```

In [ ]:

```
# 横向合并
df2a = pd.read_excel("高校信息.xlsx", sheet_name = 'var6')
df2b = pd.read_excel("高校信息.xlsx", sheet_name = 'var3')
pd.concat([df2a, df2b], axis = 1)
```

In [ ]:

```
df2ai = df2a.set_index('学校名称')
df2bi = df2b.set_index('学校名称')
pd.concat([df2ai, df2bi], axis = 1)
```

7.5 实战：对PM2.5数据做基本整理

要求:

对PM2.5数据按照年进行分组拆分，然后计算出:
    每年PM2.5的平均值、中位数、最大值、最小值
    每年PM2.5值大于200、300、500的天数
将PM2.5数据整理为以年为行，月为列，单元格为最大值的宽表形式
    提示：先使用分组汇总，然后进行长宽格式的转换
将2009年和2012年的数据分别提取出来，然后合并为一个数据框
分别使用长宽格式转换、筛选然后横向合并两种方式，将数据转换为每年一列的宽表格式