In [ ]:

```
df2.query("名次 >10 and 名次 < 90 and 类型 != '综合'")
```

In [ ]:

```
limit = 5
df2.query("名次<=@limit & 类型 == '综合'")
```

In [ ]:

```
df2.query("名次<=@limit & 类型 != '综合'")
```

5.3 实战：筛选数据中所需的案例

高校信息数据：

分别使用索引、非索引和类SQL方式，筛选出 教育部主管的总分低于70分的大学
只将主管部门设定为索引，重新实现上述需求

北京PM2.5数据：

筛选出 PM2.5 > 200 的案例
筛选出 2016年10月，PM2.5 > 100 的案例
筛选出 2016年10月上班时间（9:00AM-5:00PM），PM2.5 > 100 的案例

6 变量变换

6.1 计算新变量

6.1.1 新变量为常数

df['varname'] = value

In [ ]:

```
df2.newvar = 1 # 注意该命令不会报错！
df2.head()
```

In [ ]:

```
df2['cons'] = 1
df2
```

6.1.2 基于原变量做简单四则运算

df['varname'] = df['oldvar'] * 100

df['varname'] = df.oldvar * 100

In [ ]:

```
df2['new2'] = df2.总分 + df2.名次 + 1
df2
```

In [ ]:

```
import numpy
df2.new2 = numpy.sqrt(df2.总分) # numpy内置函数自动支持serial格式数据
df2
```

In [ ]:

```
import math
df2['new3'] = math.sqrt(df2.总分) # 此处会报错
df2
```

6.1.3 基于一个原变量做函数运算

df.apply(

```
    func : 希望对行/列执行的函数表达式
    axis = 0 : 针对行还是列进行计算
        0/'index': 针对每列进行计算
        1/'columns': 针对每行进行计算
```

)

简化的用法

```
    df['varname'] = df.oldvar.apply(函数表达式)
```

In [ ]:

```
import math
df2['new3'] = df2.总分.apply(math.sqrt)
df2
```

In [ ]:

```
df2['new3'] = df2.总分.apply(numpy.sqrt)
df2
```

特殊运算的实现方式

例：取变量的第一个字符

def m_head(tmpstr):

```
    return tmpstr[:1]
```

对所有单元格进行相同的函数运算

```
    dfnew = df.applymap(函数表达式) # 是以cell为单位在进行操作
```

In [ ]:

```
df2[['名次', '总分']].applymap(math.sqrt)
```

不修改原df，而是生成新的df

```
df.assign(varname = expression)
```

In [ ]:

```
df3 = df2.assign(new = df2.总分.apply(math.sqrt))
df2
```

6.2 在指定位置插入新变量列

df.insert(

```
loc : 插入位置的索引值，0 <= loc <= len(columns)
column : 插入的新列名称
value : Series或者类数组结构的变量值
allow_duplicates = False : 是否允许新列重名
```

) # 该方法会直接修改原df

In [ ]:

```
df2.insert(1,'newvar','cons')
df2
```

6.3 修改/替换变量值

本质上是如何直接指定到单元格的问题，只要能准确定位单元格地址，就能够做到准确替换。

In [ ]:

```
df2.所在城市.isin(['上海'])
```

In [ ]:

```
df2.所在城市[70] = '上海市' # 这种引用方式会给出警告
df2[69:75]
```

In [ ]:

```
df2['所在城市'][70] = '上海市' # 这种引用方式会给出警告
```

In [ ]:

```
df2.loc[70, '所在城市'] = '上海市' # 精确引用，消停了
df2
```

6.3.1 对应数值的替换

df.replace(

    to_replace = None : 将被替换的原数值，所有严格匹配的数值将被用value替换
        可以是str/regex/list/dict/Series/numeric/None
    value = None : 希望填充的新数值
    inplace = False

)

In [ ]:

```
df2.所在城市.replace('北京市', '帝都') # 单个值替换
```

In [ ]:

```
df2.所在城市.replace(['北京市', '上海市'], ['帝都', '魔都']) # 列表值批量替换
```

In [ ]:

```
# 字典批量映射替换
df2.所在城市.replace({'北京市' : '帝都',  '上海市' : '魔都'})
```

6.3.2 指定数值范围的替换

方法一：使用正则表达式完成替换

df.replace(regex, newvalue)

方法二：使用行筛选方式完成替换

    用行筛选方式得到行索引，然后用loc命令定位替换
    目前也支持直接筛选出单元格进行数值替换
    注意：query命令的类SQL语句可以进行检索，但不直接支持数值替换

In [ ]:

```
df2.总分.iloc[0:2] = 10
df2
```

In [ ]:

```
df2.loc[3:5,'总分'] = 20
df2.head(10)
```

In [ ]:

```
# 用loc命令完成替换
df2.loc[df2.名次 < 10, '总分'] = 20 # 用index引用出相应的索引
df2.head(10)
```

In [ ]:

```
# 直接进行定位和替换
df2.总分[df2.名次 < 10] = 25
df2
```

In [ ]:

```
# 注意这里的出错原因
df2[df2.名次 < 10].总分 = 30
df2
```

In [ ]:

```
# query语句可以用于index定位，然后实现数值替换
df2.loc[df2.query("名次 < 10 and 类型 == '综合'").index, '总分'] = 10
df2
```

In [ ]:

```
# query语句无法直接实现数值替换，原因：query生成的是数据copy，不是地址引用
df2.query("名次 < 10 and 类型 != '综合'")['总分'] = 25
df2
```

6.4 哑变量变换

pd.get_dummies(

data : 希望转换的数据框/变量列
prefix = None : 哑变量名称前缀
prefix_sep = '_': 前缀和序号之间的连接字符，设定有prefix或列名时生效
dummy_na = False : 是否为NaNs专门设定一个哑变量列
columns = None : 希望转换的原始列名，如果不设定，则转换所有符合条件的列
drop_first = False : 是否返回k-1个哑变量，而不是k个哑变量

) # 返回值为数据框

In [ ]:

```
df2.head()
```

In [ ]:

```
pd.get_dummies(df2.类型, prefix = "pre" )
```

In [ ]:

```
pd.get_dummies(df2, columns=['类型'] )
```

6.5 数值变量分段

pd.cut(

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
```