一、介绍pandas包前期工作

1、运行本文提供的代码，需引入包有：

```
pandas as pd
numpy as np
matplotlib.pyplot as plt
```

2、部分快捷查询方法

对象属性查询（pandas包定义的数组对象，或矩阵对象）：在对象名后加点，然后按tab键，则会显示

二、pandas包中的重要方法介绍

1、对象定义

1）、类似数组类型

```
pd.Series([1,np.nan,3])
//标明每个数组元素，通过pd.Series封装成数组
//其中数组元素的生成可以调用numpy中的方法，如:
np.random.randint(0,7,size=10))。
```

```
pd.Series(1,index=list(range(4)),dtype='float32')
pd.Series(1,index=pd.date_range('20130102', periods=4),dtype='float32')
//除了给出具体的数值，还可以指定对应的index（类型不局限于数字），以及数字的类型dtype
```

2）、类似矩阵类型

```
pd.DataFrame(data={'col1': [1, 2], 'col2': [3, 4]})
//可以标明矩阵具体的数字，如上列一共两列，一列为1,2；一列为3,4。生成一个举证
pd.DataFrame(np.random.randn(6,4), index=pd.date_range('20130102', periods=4), columns=list('ABCD'))
//可以标明具体的数值，index范围即矩阵行数，columns范围即矩阵列数
pd.DataFrame({ 'A' : 1.,
            'B' : pd.Timestamp('20130102'),
            'C' : pd.Series(1,index=list(range(4)),dtype='float32'),
            'D' : np.array([3] * 4,dtype='int32'),
            'E' : pd.Categorical(["test","train","test","train"]),
```

'F' : 'foo' })
//也可以通过定义键值对，定义矩阵；每个键值对表示一个列

3）、对象赋值

df.at[dates[0],'A'] = 0 //对某一个值赋值
df.iat[0,1] = 0 //对某一个值赋值
df.loc[:,'D'] = np.array([5] * len(df)) //对某一列赋值
df.fillna(value=5) //对矩阵中为NaN的元素填充为5

2、对pandas中定义对象的操作

1）、获取头尾数据

pd.head(1) //数组则为第一个数字，对于矩阵则为第一行数据；2则为前两个。
pd.tail(1) //数组则为最后一个数字，对于矩阵则为最后一行数据。

2）、查看数据的行、列、数值

pd.index //获取数组的下标，对于矩阵获取矩阵的行
pd.columns //获取矩阵的列，但对于数组则无该属性
pd.values //获取数据及举证的具体数值
pd.describle()//获取pd对象全部基础属性

3）、类似矩阵计算 相关操作（注：df为一个实例对象）

转置

df.T //获取矩阵的转置

矩阵部分数据提取

通过行名和列名，对整行或者整列进行
df['A'] //获取矩阵中，列名为A的该列数据(对某列的提取，对行不行)
df[0:1] //获取矩阵中，第0行（行数的下标从0开始），df[0:2]则取第0行和第1行，不包括2
//也可以通过具体的行名进行提取，如df['20130102':'20130104']。提取'20130102'到'20130104'之间数值

通过df自带函数，和具体的行名和列名进行提取

df.loc[dates[0]] //dates数组为行名数组，其中dates[0]表示提取第一行数据。同样可以用具体标签进行筛选df.loc['20130102':'20130104']
df.loc[:,['A','B']] //提取矩阵中列名为A和B，两列的数据
df.loc[1,['A']] //可以结合行、列，对矩阵进行提取；
//如果列名也有数组，可以通过以下方法提取某个值df.at[dates[0],'A']

通过df自带函数，和所在位置进行提取
df.iloc[2:5,0:2] //提取第3至5行，以及第1至2列的数据
df.iat[1,1] //选取第二行、第二列的数据

通过运算符选取符合条件的数据
df[df.A > 0] //选取第A列中，大于0的对应行
df[df > 0] //选取矩阵中大于0的值，如果小于0，则显示为NaN
df[df['E'].isin(['two','four'])] //选取矩阵中，列名为E且数值为two和four的数据
df.dropna(how='any') //去掉元素中包含NaN的行
pd.isna(df1) //判断df1矩阵中是否为NaN，如果为空则为True

矩阵中增、删、改
df['E'] = ['one','two','three','four'] //在df矩阵中，增加一列，列名为E，数值为'one','two','three','four'，其中数组的长度要和矩阵的行数相同
df1 = df.reindex(index=dates[0:4], columns=list(df.columns) + ['E']) //提取df矩阵中第1到4行，并在原有的列中新加入一列E
df.append(df.iloc[3], ignore_index=True) //取出df矩阵中的第4行，比添加都矩阵df下面

对矩阵中的统计方法
df.mean() //对每一列进行统计，计算平均值
df.mean(1) //对每一行进行统计，计算平均值

矩阵中的加减运算
df.sub(s, axis='index') //按照index进行匹配，完成两个矩阵相减（df-s），如果s为数组，其为补全为一个矩阵后进行计算
df.apply(np.cumsum) //每行数值向上求和，如数组1，2，3 计算结果为 1，3，6
df.max() - df.min() //去数据每列最大值 - 每列最小值 其与df.apply(lambda x : x.max() -x.min()) 的区别
s.value_counts() //对数组中，数值次数的统计，仅对矩阵有帮助
s.str.lower() //把数组中的大写变为小写

4）、类似SQL语言 相关操作
类似order by
df.sort_index(axis=1,ascending=False) //根据矩阵的行名进行排序
df.sort(columns = 'B') //根据矩阵的某一列的值进行排序，如根据列名为B的值进行排序

类似group by

df.groupby(['A','B']).sum() //先根据B列的数据进行聚合，在根据A列的数据聚合。A为大类，B为小类

类似union

pieces = [df[:3], df[3:7], df[7:]] //
pd.concat(pieces) //每行的数据个数要求一样，才能进行union

类似join

left = pd.DataFrame({'key': ['foo', 'foo'], 'lval': [1, 2]})
right = pd.DataFrame({'key': ['foo', 'foo'], 'rval': [4, 5]})
pd.merge(left, right, on='key') //根据key列进行关联

5）、类似Excel处理方法

df.stack() //类似excel中根据不同维度统计，变换表头
df.unstack(1) //类似excel中根据不同维度统计，变换表头
pd.pivot_table(df, values='D', index=['A', 'B'], columns=['C']) //类似透视表，详情需查询官网

3、数据的写入

1）、csv的读取和写入

df.to_csv('foo.csv') //把df矩阵写入foo.csv中
pd.read_csv('foo.csv') //把foo.csv数据读入

2）、xls的读取和写入

df.to_excel('foo.xlsx', sheet_name='Sheet1') //把df矩阵写入foo.xlsx中
pd.read_excel('foo.xlsx', 'Sheet1', index_col=None, na_values=['NA']) //读取foo.xlxs中的数据