82.07 pd.drop()函数

删除表中的某一行或者某一列更明智的方法是使用drop，它不改变原有的df中的数据，而是返回另一个dataframe来存放删除后的数据，from《利用python进行数据分析》。

清理无效数据

```
df[df.isnull()]   #返回的是个true或false的Series对象（掩码对象），进而筛选出我们需要的特定数据。
df[df.notnull()]

df.dropna()       #将所有含有nan项的row删除
df.dropna(axis=1,thresh=3)   #将在列的方向上三个为NaN的项删除
df.dropna(how='ALL')         #将全部项都是nan的row删除
```

此处：print data.dropna() 和 print data[data.notnull()] 结果一样

填充无效值

```
df.fillna(0)
df.fillna({1:0, 2:0.5})            #对第一列nan值赋0，第二列赋值0.5
df.fillna(method='ffill') #在列方向上以前一个值作为值赋给NaN
```

drop函数的使用

（1）drop函数的使用：删除行、删除列

```
print frame.drop(['a'])
print frame.drop(['Ohio'], axis = 1)
```

drop函数默认删除行，列需要加axis = 1

（2）drop函数的使用：inplace参数

采用drop方法，有下面三种等价的表达式：

```
1. DF= DF.drop('column_name', axis=1);
2. DF.drop('column_name',axis=1, inplace=True)
3. DF.drop([DF.columns[[0,1, 3]]], axis=1, inplace=True)   # No
te: zero indexed
```

注意：凡是会对原数组作出修改并返回一个新数组的，往往都有一个 inplace可选参数。如果手动设定为True（默认为False），那么原数组直接就被替换。也就是说，采用inplace=True之后，原数组名（如2和3情况所示）对应的内存值直接改变；

而采用inplace=False之后，原数组名对应的内存值并不改变，需要将新的结果赋给一个新的数组或者覆盖原数组的内存位置（如1情况所示）。

（3）drop函数的使用：数据类型转换

```
df['Name'] = df['Name'].astype(np.datetime64)
```

DataFrame.astype() 方法可对整个DataFrame或某一列进行数据格式转换，支持Python和NumPy的数据类型。