一、数据结构介绍

在pandas中有两类非常重要的数据结构，即序列Series和数据框DataFrame。Series类似于numpy中的一维数组，除了通吃一维数组可用的函数或方法，而且其可通过索引标签的方式获取数据，还具有索引的自动对齐功能；DataFrame类似于numpy中的二维数组，同样可以通用numpy数组的函数和方法，而且还具有其他灵活应用，后续会介绍到。

1、Series的创建

序列的创建主要有三种方式：

1）通过一维数组创建序列

import numpy as np, pandas as pd

arr1 = np.arange(10)

arr1

type(arr1)

s1 = pd.Series(arr1)

s1

type(s1)

2）通过字典的方式创建序列

dic1 = {'a':10,'b':20,'c':30,'d':40,'e':50}

dic1

type(dic1)

s2 = pd.Series(dic1)

s2

type(s2)

3）通过DataFrame中的某一行或某一列创建序列

这部分内容我们放在后面讲，因为下面就开始将DataFrame的创建。

2、DataFrame的创建

数据框的创建主要有三种方式：

1）通过二维数组创建数据框

arr2 = np.array(np.arange(12)).reshape(4,3)

arr2

type(arr2)

df1 = pd.DataFrame(arr2)

df1

type(df1)

2）通过字典的方式创建数据框

以下以两种字典来创建数据框，一个是字典列表，一个是嵌套字典。

dic2 = {'a':[1,2,3,4],'b':[5,6,7,8],

'c':[9,10,11,12],'d':[13,14,15,16]}

dic2

type(dic2)

df2 = pd.DataFrame(dic2)

df2

type(df2)

dic3 = {'one':{'a':1,'b':2,'c':3,'d':4},

'two':{'a':5,'b':6,'c':7,'d':8},

'three':{'a':9,'b':10,'c':11,'d':12}}

dic3

type(dic3)

df3 = pd.DataFrame(dic3)

df3

type(df3)

3）通过数据框的方式创建数据框

df4 = df3[['one','three']]

df4

type(df4)

s3 = df3['one']

s3

type(s3)

二、数据索引index

细致的朋友可能会发现一个现象，不论是序列也好，还是数据框也好，对象的最左边总有一个非原始数据对象，这个是什么呢？不错，就是我们接下来要介绍的索引。  
在我看来，序列或数据框的索引有两大用处，一个是通过索引值或索引标签获取目标数据，另一个是通过索引，可以使序列或数据框的计算、操作实现自动化对齐，下面我们就来看看这两个功能的应用。

1、通过索引值或索引标签获取数据

s4 = pd.Series(np.array([1,1,2,3,5,8]))

s4

如果不给序列一个指定的索引值，则序列自动生成一个从0开始的自增索引。可以通过index查看序列的索引：

s4.index

现在我们为序列设定一个自定义的索引值：

s4.index = ['a','b','c','d','e','f']

s4

序列有了索引，就可以通过索引值或索引标签进行数据的获取：

s4[3]

s4['e']

s4[[1,3,5]]

s4[['a','b','d','f']]

s4[:4]

s4['c':]

s4['b':'e']

千万注意：如果通过索引标签获取数据的话，末端标签所对应的值是可以返回的！在一维数组中，就无法通过索引标签获取数据，这也是序列不同于一维数组的一个方面。

2、自动化对齐

如果有两个序列，需要对这两个序列进行算术运算，这时索引的存在就体现的它的价值了—自动化对齐.

s5 = pd.Series(np.array([10,15,20,30,55,80]),

index = ['a','b','c','d','e','f'])

s5

s6 = pd.Series(np.array([12,11,13,15,14,16]),

index = ['a','c','g','b','d','f'])

s6

s5 + s6

s5/s6

由于s5中没有对应的g索引，s6中没有对应的e索引，所以数据的运算会产生两个缺失值NaN。注意，这里的算术结果就实现了两个序列索引的自动对齐，而非简单的将两个序列加总或相除。对于数据框的对齐，不仅仅是行索引的自动对齐，同时也会自动对齐列索引（变量名）

数据框中同样有索引，而且数据框是二维数组的推广，所以其不仅有行索引，而且还存在列索引，关于数据框中的索引相比于序列的应用要强大的多，这部分内容将放在数据查询中讲解。

三、利用pandas查询数据

这里的查询数据相当于R语言里的subset功能，可以通过布尔索引有针对的选取原数据的子集、指定行、指定列等。我们先导入一个student数据集：

student=pd.io.parsers.read\_csv('C:\\Users\\admin\\Desktop\\student.csv')

查询数据的前5行或末尾5行

student.head()

student.tail()

查询指定的行

student.ix[[0,2,4,5,7]] #这里的ix索引标签函数必须是中括号[]

查询指定的列

student[['Name','Height','Weight']].head() #如果多个列的话，必须使用双重中括号

也可以通过ix索引标签查询指定的列

student.ix[:,['Name','Height','Weight']].head()

查询指定的行和列

student.ix[[0,2,4,5,7],['Name','Height','Weight']].head()

以上是从行或列的角度查询数据的子集，现在我们来看看如何通过布尔索引实现数据的子集查询。  
查询所有女生的信息

student[student['Sex']=='F']

查询出所有12岁以上的女生信息

student[(student['Sex']=='F') & (student['Age']>12)]

查询出所有12岁以上的女生姓名、身高和体重

student[(student['Sex']=='F') & (student['Age']>12)][['Name','Height','Weight']]

上面的查询逻辑其实非常的简单，需要注意的是，如果是多个条件的查询，必须在&（且）或者|（或）的两端条件用括号括起来。

四、统计分析

pandas模块为我们提供了非常多的描述性统计分析的指标函数，如总和、均值、最小值、最大值等，我们来具体看看这些函数：  
首先随机生成三组数据

np.random.seed(1234)

d1 = pd.Series(2\*np.random.normal(size = 100)+3)

d2 = np.random.f(2,4,size = 100)

d3 = np.random.randint(1,100,size = 100)

d1.count() #非空元素计算

d1.min() #最小值

d1.max() #最大值

d1.idxmin() #最小值的位置，类似于R中的which.min函数

d1.idxmax() #最大值的位置，类似于R中的which.max函数

d1.quantile(0.1) #10%分位数

d1.sum() #求和

d1.mean() #均值

d1.median() #中位数

d1.mode() #众数

d1.var() #方差

d1.std() #标准差

d1.mad() #平均绝对偏差

d1.skew() #偏度

d1.kurt() #峰度

d1.describe() #一次性输出多个描述性统计指标

必须注意的是，descirbe方法只能针对序列或数据框，一维数组是没有这个方法的

这里自定义一个函数，将这些统计描述指标全部汇总到一起:

def stats(x):

return pd.Series([x.count(),x.min(),x.idxmin(),

x.quantile(.25),x.median(),

x.quantile(.75),x.mean(),

x.max(),x.idxmax(),

x.mad(),x.var(),

x.std(),x.skew(),x.kurt()],

index = ['Count','Min','Whicn\_Min',

'Q1','Median','Q3','Mean',

'Max','Which\_Max','Mad',

'Var','Std','Skew','Kurt'])

stats(d1)

在实际的工作中，我们可能需要处理的是一系列的数值型数据框，如何将这个函数应用到数据框中的每一列呢？可以使用apply函数，这个非常类似于R中的apply的应用方法。  
将之前创建的d1,d2,d3数据构建数据框:

df = pd.DataFrame(np.array([d1,d2,d3]).T,columns=['x1','x2','x3'])

df.head()

df.apply(stats)

非常完美，就这样很简单的创建了数值型数据的统计性描述。如果是离散型数据呢？就不能用这个统计口径了，我们需要统计离散变量的观测数、唯一值个数、众数水平及个数。你只需要使用describe方法就可以实现这样的统计了。

student['Sex'].describe()

除以上的简单描述性统计之外，还提供了连续变量的相关系数（corr）和协方差矩阵（cov）的求解，这个跟R语言是一致的用法。

df.corr()

关于相关系数的计算可以调用pearson方法或kendell方法或spearman方法，默认使用pearson方法。

df.corr('spearman')

如果只想关注某一个变量与其余变量的相关系数的话，可以使用corrwith,如下方只关心x1与其余变量的相关系数:

df.corrwith(df['x1'])

数值型变量间的协方差矩阵

df.cov()

五、类似于SQL的操作

在SQL中常见的操作主要是增、删、改、查几个动作，那么pandas能否实现对数据的这几项操作呢？答案是Of Course!

增：添加新行或增加新列

In [99]: dic = {'Name':['LiuShunxiang','Zhangshan'],

...: 'Sex':['M','F'],'Age':[27,23],

...: 'Height':[165.7,167.2],'Weight':[61,63]}

In [100]: student2 = pd.DataFrame(dic)

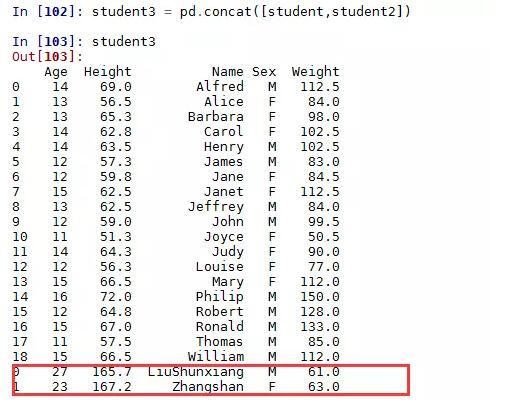
In [101]: student2

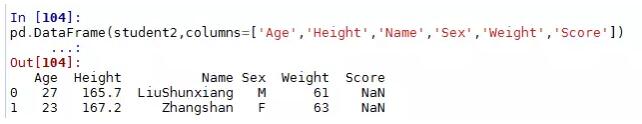
Out[101]:

Age Height Name Sex Weight

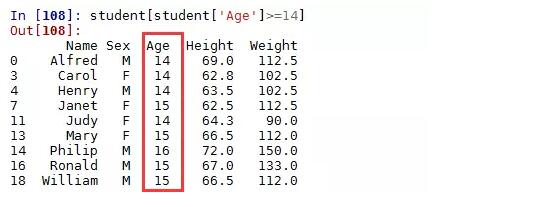
0 27 165.7 LiuShunxiang M 61

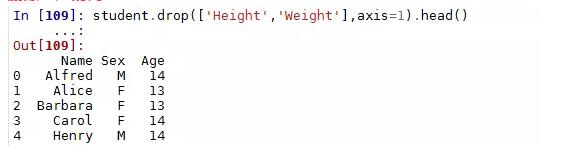
1 23 167.2 Zhangshan F 63

现在将student2中的数据新增到student中，可以通过concat函数实现：  
  
注意到了吗？在数据库中union必须要求两张表的列顺序一致，而这里concat函数可以自动对齐两个数据框的变量！

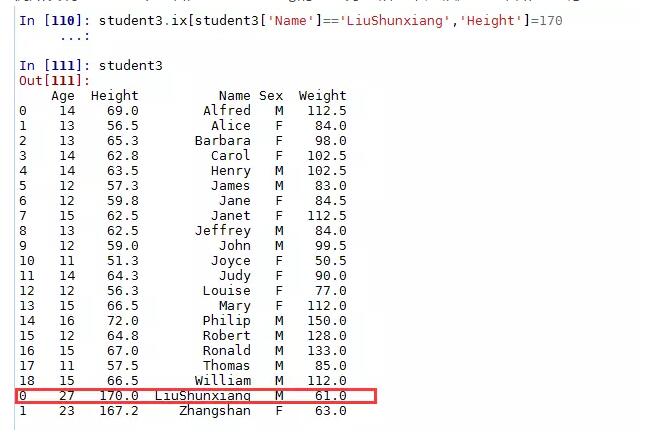
新增列的话，其实在pandas中就更简单了，例如在student2中新增一列学生成绩：  
  
对于新增的列没有赋值，就会出现空NaN的形式。

删：删除表、观测行或变量列

删除数据框student2,通过del命令实现，该命令可以删除Python的所有对象。  
  
删除指定的行  
  
原数据中的第1,2,4,7行的数据已经被删除了。  
根据布尔索引删除行数据，其实这个删除就是保留删除条件的反面数据，例如删除所有14岁以下的学生：  


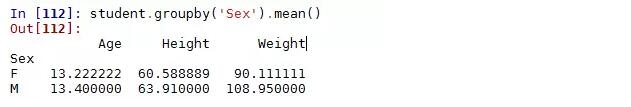
删除指定的列  
  
我们发现，不论是删除行还是删除列，都可以通过drop方法实现，只需要设定好删除的轴即可，即调整drop方法中的axis参数。默认该参数为0，表示删除行观测，如果需要删除列变量，则需设置为1。

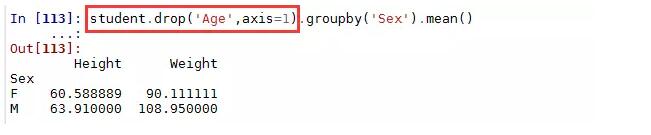
改：修改原始记录的值

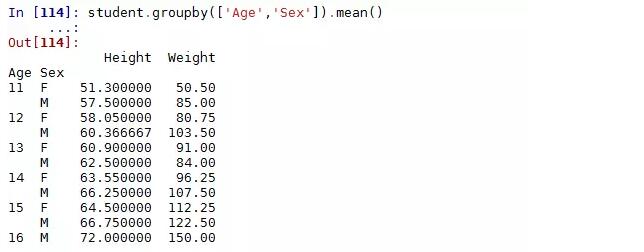
如果发现表中的某些数据错误了，如何更改原来的值呢？我们试试结合布尔索引和赋值的方法：  
例如发现student3中姓名为Liushunxiang的学生身高错了，应该是173，如何改呢？  
  
这样就可以把原来的身高修改为现在的170了。  
看，关于索引的操作非常灵活、方便吧，就这样轻松搞定数据的更改。

查：有关数据查询部分，上面已经介绍过，下面重点讲讲聚合、排序和多表连接操作。

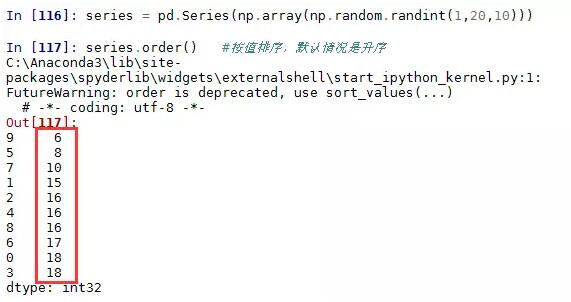
聚合：pandas模块中可以通过groupby()函数实现数据的聚合操作

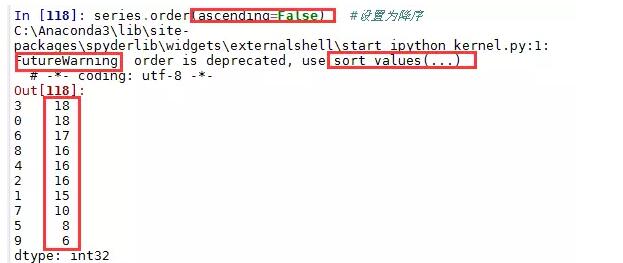
根据性别分组，计算各组别中学生身高和体重的平均值：  


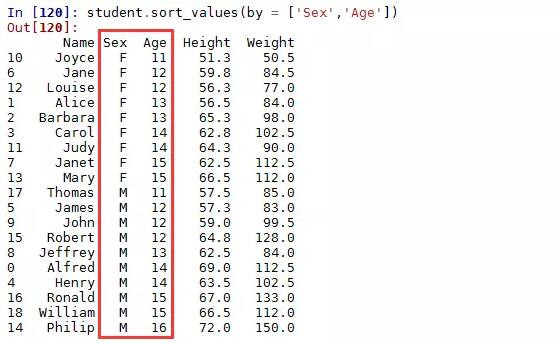
如果不对原始数据作限制的话，聚合函数会自动选择数值型数据进行聚合计算。如果不想对年龄计算平均值的话，就需要剔除改变量：  


groupby还可以使用多个分组变量，例如根本年龄和性别分组，计算身高与体重的平均值：  
  
当然，还可以对每个分组计算多个统计量：  
  
是不是很简单，只需一句就能完成SQL中的SELECT…FROM…GROUP BY…功能，何乐而不为呢？

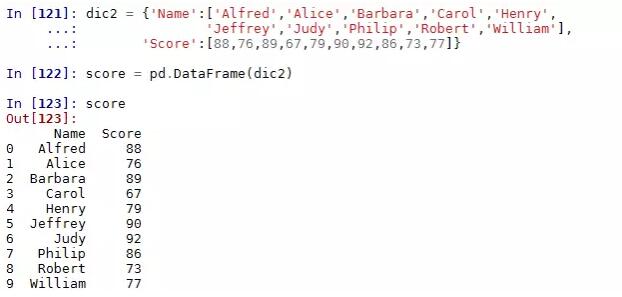
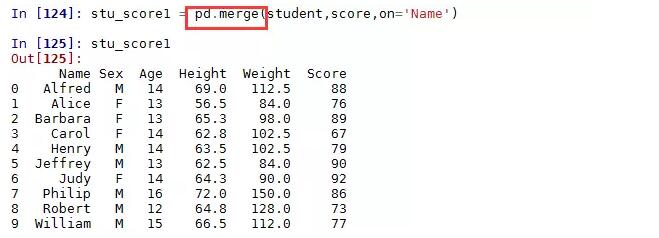
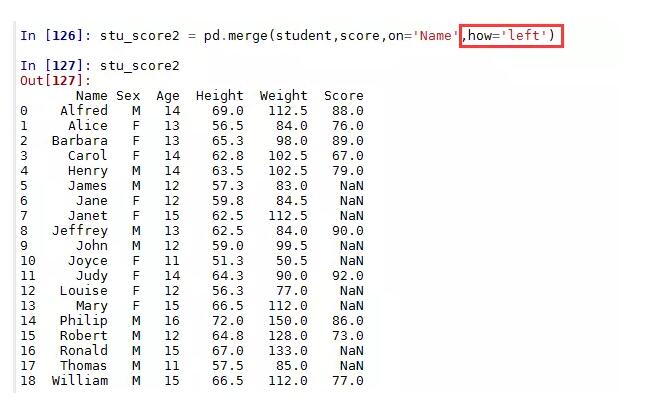
排序：

排序在日常的统计分析中还是比较常见的操作，我们可以使用order、sort\_index和sort\_values实现序列和数据框的排序工作：  


我们再试试降序排序的设置：  
  
上面两个结果其实都是按值排序，并且结果中都给出了警告信息，即建议使用sort\_values()函数进行按值排序。

在数据框中一般都是按值排序，例如：  


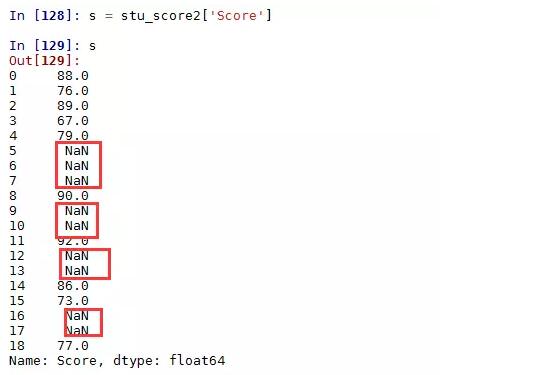
多表连接:

多表之间的连接也是非常常见的数据库操作，连接分内连接和外连接，在数据库语言中通过join关键字实现，pandas我比较建议使用merger函数实现数据的各种连接操作。  
如下是构造一张学生的成绩表：  
  
现在想把学生表student与学生成绩表score做一个关联，该如何操作呢？  
  
注意，默认情况下，merge函数实现的是两个表之间的内连接，即返回两张表中共同部分的数据。可以通过how参数设置连接的方式，left为左连接；right为右连接；outer为外连接。  


左连接实现的是保留student表中的所有信息，同时将score表的信息与之配对，能配多少配多少，对于没有配对上的Name，将会显示成绩为NaN。

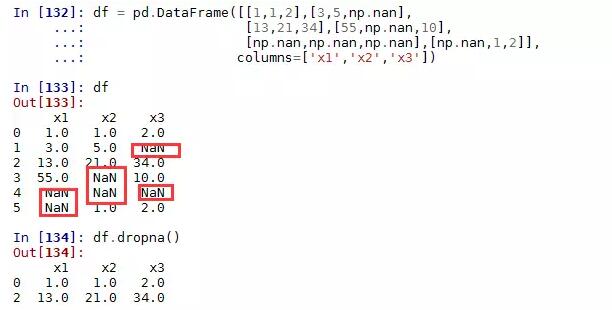
六、缺失值处理

现实生活中的数据是非常杂乱的，其中缺失值也是非常常见的，对于缺失值的存在可能会影响到后期的数据分析或挖掘工作，那么我们该如何处理这些缺失值呢？常用的有三大类方法，即删除法、填补法和插值法。  
删除法：当数据中的某个变量大部分值都是缺失值，可以考虑删除改变量；当缺失值是随机分布的，且缺失的数量并不是很多是，也可以删除这些缺失的观测。  
替补法：对于连续型变量，如果变量的分布近似或就是正态分布的话，可以用均值替代那些缺失值；如果变量是有偏的，可以使用中位数来代替那些缺失值；对于离散型变量，我们一般用众数去替换那些存在缺失的观测。  
插补法：插补法是基于蒙特卡洛模拟法，结合线性模型、广义线性模型、决策树等方法计算出来的预测值替换缺失值。

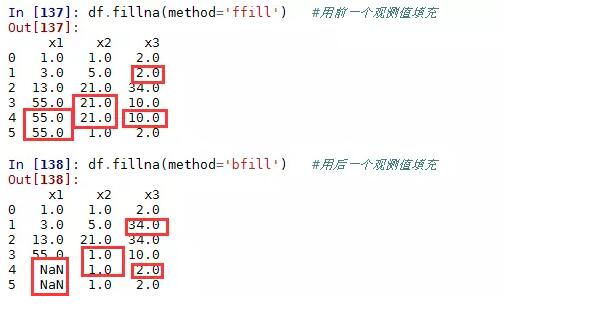
我们这里就介绍简单的删除法和替补法：  
  
这是一组含有缺失值的序列，我们可以结合sum函数和isnull函数来检测数据中含有多少缺失值：

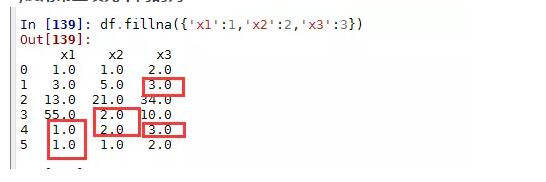
In [130]: sum(pd.isnull(s))

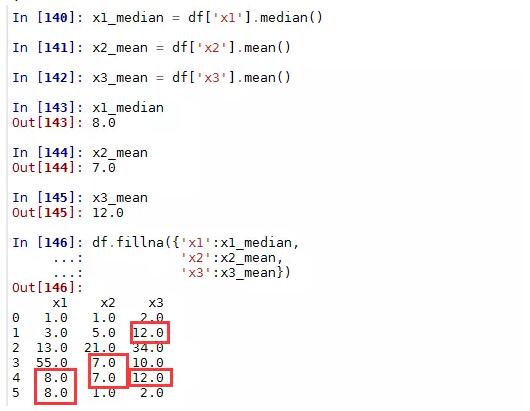
Out[130]: 9

直接删除缺失值  
  
默认情况下，dropna会删除任何含有缺失值的行，我们再构造一个数据框试试：  
  
返回结果表明，数据中只要含有缺失值NaN,该数据行就会被删除，如果使用参数how=’all’，则表明只删除所有行为缺失值的观测。  


使用一个常量来填补缺失值，可以使用fillna函数实现简单的填补工作：  
1）用0填补所有缺失值  


2)采用前项填充或后向填充  


3)使用常量填充不同的列  


4)用均值或中位数填充各自的列  
  
很显然，在使用填充法时，相对于常数填充或前项、后项填充，使用各列的众数、均值或中位数填充要更加合理一点，这也是工作中常用的一个快捷手段。

七、数据透视表

在Excel中有一个非常强大的功能就是数据透视表，通过托拉拽的方式可以迅速的查看数据的聚合情况，这里的聚合可以是计数、求和、均值、标准差等。  
pandas为我们提供了非常强大的函数pivot\_table()，该函数就是实现数据透视表功能的。对于上面所说的一些聚合函数，可以通过参数aggfunc设定。我们先看看这个函数的语法和参数吧：

pivot\_table(data,values=None,

index=None,

columns=None,

aggfunc='mean',

fill\_value=None,

margins=False,

dropna=True,

margins\_name='All')

data：需要进行数据透视表操作的数据框

values：指定需要聚合的字段

index：指定某些原始变量作为行索引

columns：指定哪些离散的分组变量

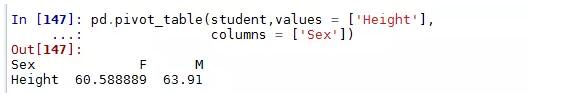
aggfunc：指定相应的聚合函数

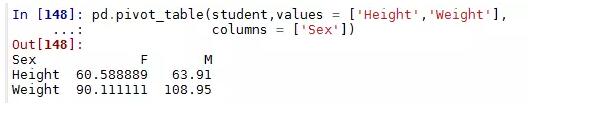
fill\_value：使用一个常数替代缺失值，默认不替换

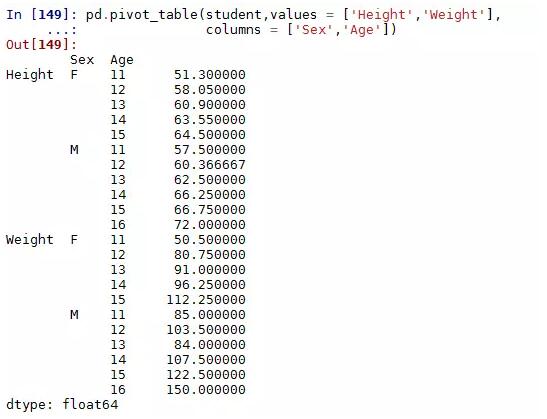
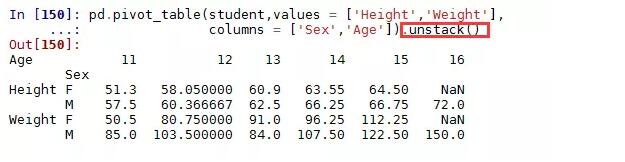
margins：是否进行行或列的汇总，默认不汇总

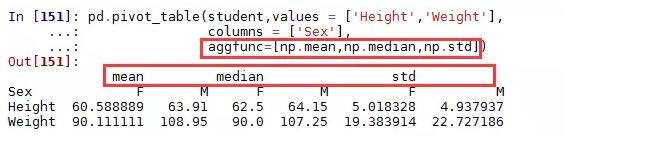
dropna：默认所有观测为缺失的列

margins\_name：默认行汇总或列汇总的名称为'All'

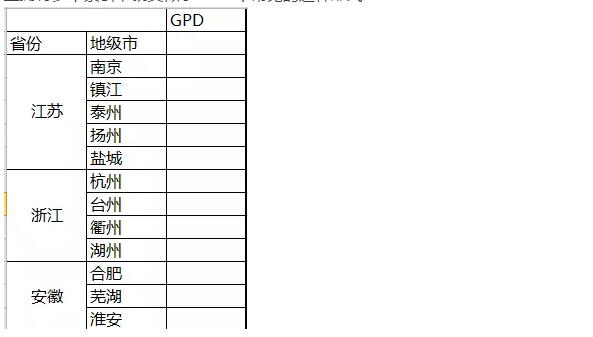
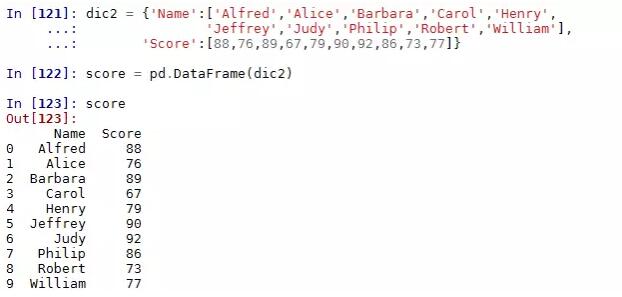
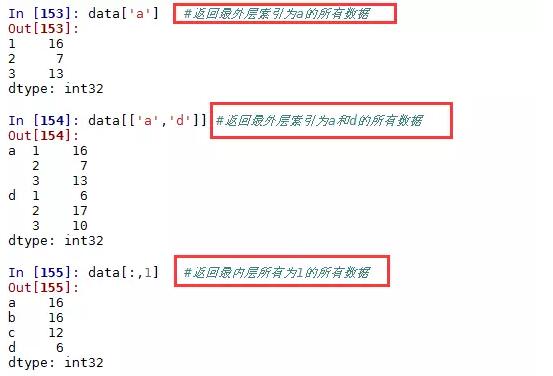
我们仍然以student表为例，来认识一下数据透视表pivot\_table函数的用法：  
对一个分组变量（Sex），一个数值变量（Height）作统计汇总  


对一个分组变量（Sex），两个数值变量（Height,Weight）作统计汇总  


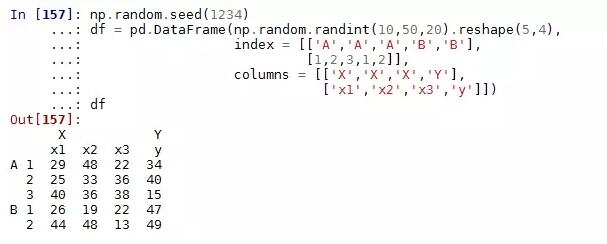
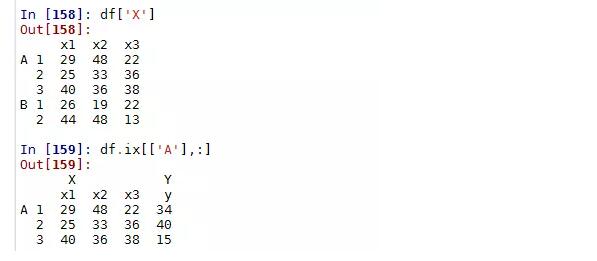
对两个分组变量（Sex，Age)，两个数值变量（Height,Weight）作统计汇总  
  
很显然这样的结果并不像Excel中预期的那样，该如何变成列联表的形式的？很简单，只需将结果进行非堆叠操作（unstack）即可：  
  
看，这样的结果是不是比上面那种看起来更舒服一点？

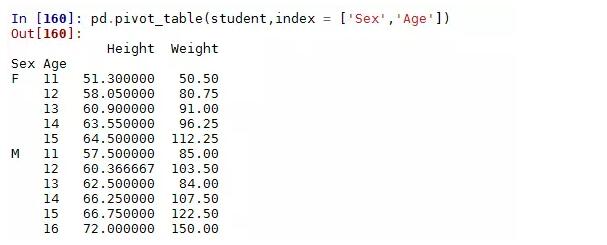
使用多个聚合函数  
  
有关更多数据透视表的操作，可参考《Pandas透视表（pivot\_table）详解》一文，链接地址：<http://python.jobbole.com/81212/>

八、多层索引的使用

最后我们再来讲讲pandas中的一个重要功能，那就是多层索引。在序列中它可以实现在一个轴上拥有多个索引，就类似于Excel中常见的这种形式：  
  
对于这样的数据格式有什么好处呢？pandas可以帮我们实现用低维度形式处理高维数数据，这里举个例子也许你就能明白了：  
  
对于这种多层次索引的序列，取数据就显得非常简单了：  


对于这种多层次索引的序列，我们还可以非常方便的将其转换为数据框的形式：  
  
以上针对的是序列的多层次索引，数据框也同样有多层次的索引，而且每条轴上都可以有这样的索引，就类似于Excel中常见的这种形式：  


我们不妨构造一个类似的高维数据框：  
  
同样，数据框中的多层索引也可以非常便捷的取出大块数据：  


在数据框中使用多层索引，可以将整个数据集控制在二维表结构中，这对于数据重塑和基于分组的操作（如数据透视表的生成）比较有帮助。  
就拿student二维数据框为例，我们构造一个多层索引数据集：  


讲到这里，我们关于pandas模块的学习基本完成，其实在掌握了pandas这8个主要的应用方法就可以灵活的解决很多工作中的数据处理、统计分析等任务