

数据汇总与统计

DATA

主讲教师：宋晖

数据汇总与统计

1. 统计的基本概念 统计的含义 | 常用统计量
2. Pandas数据结构 **Series**对象及数据访问 | **DataFrame**对象及数据访问
3. 数据文件读取 读写**CSV**和文本文件 | 读写**Excel**文件
4. 数据清洗 缺失数据处理 | 去重
5. 数据规整化 数据合并 | 数据排序
6. 统计分析 通用函数与运算 | 统计函数 | 相关性分析

1. 统计的基本概念

1. 统计的基本概念 | 统计的含义

 统计是对数据资料的获取、整理、分析、描述及推断方法的总称

在理解现有数据的基础上，统计分析进一步发现规律

- ◆ 对未来实施预测
- ◆ 如市场预测、人口预测、经济发展预测等

案例3-1：学生问卷调查统计分析

- 1、你的性别是_____；
- 2、你的年龄为_____周岁；
- 3、你的身高=_____cm，体重=_____kg；
- 4、你来自的省份是_____；
- 5、你上个月的生活费支出是_____元；
- 6、你《数据科学》课程的考试成绩是_____；
- 7、回答以下问题：
(1=完全不同意，2=比较不同意，3=无所谓，4=比较同意，5=完全同意)
(1) 我对《数据科学》课程很感兴趣_____；
(2) 案例教学法对我掌握相关知识非常重要_____；

50名同学的反馈结果

[illegible]

1. 统计的基本概念 | 总体

✚ 研究对象的全体称为**总体**

◆ 如：所有学生的身高、成绩和体重等

✚ 总体中的每一个成员都是**个体**

◆ 如：单个同学的身高、成绩

✚ 从总体中抽出部分个体组成的集合称为**样本**，样本中所含个体的数目称为**样本容量**

序 号	性 别	年 龄	身 高	体 重	省 份	成 绩	月生活费	课程兴趣	案例教学
1	male	20	170	70	LiaoNing	71	800	5	4
2	male	22	180	62	GuangXi	57	1000	2	4
3	female	20	162	47	AnHui	78	1200	4	4
4	female	22	164	53	YunNan	79	1000	4	5
5	male	19	169	76	ShanDong	88	1300	5	5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

✚ 案例3-1包括学生的性别、年龄、课程兴趣等9个总体

◆ 每个总体的样本容量为50

1. 统计的基本概念 | 常用统计量含义

统计量	含义	案例说明
均值	样本（一组数据）的算术平均值，反映数据的集中趋势	学生身高均值168.4cm，描述了全班同学身高的整体特征
方差	描述一组数据的离散程度，或样本个体距离均值的分散程度	两组数据 {1,9,30,60} 和 {24,25,25,26}，样本均值都是25，而方差分别为520.5，0.5，表示第一组数据的离散程度远大于第二组
频率	频数与样本容量的比值为频率。频数是某个值在样本中出现的次数，或样本中不同的值分别出现的次数	案例中性别的值为男和女，男生的频率是53%，女生为47%
众数	样本中出现次数最多的值，如果所有值出现的次数一样多，则认为样本没有众数。	课程兴趣反馈数据的众数是4，同学对《数据科学》比较感兴趣的同学人数最多

1. 统计的基本概念 |

统计量	含义	案例说明
分位数	将一个随机变量的概率分布范围分为几个等份的数值点	包括中位数、四分位数、百分位数等

其中：

中位数	将样本数据从小到大顺序排列，如果样本容量为奇数，处在中间的数是中位数；否则处在最中间两个数的平均值是中位数	中位数的作用类似于平均值，反应数据整体特征，但是不受最大、最小两个极端数值的影响
四分位数	将样本由小到大排列后分成四等份，处于3个分割点位置的数值就是四分位数	Q1表示下四分位数：排在25%的数值；Q2表示中位数；Q3表示上四分位数：排在75%的数值。Q3-Q1称为四分位距，反应了样本中间50%数据的取值范围

2. pandas数据结构

2. Pandas数据结构 | Why pandas

姓名	Math	English	Python	Chinese	Art	Database	Physics
王微	70	85	77	90	82	84	89
肖良英	60	64	80	75	80	92	90
方绮雯	90	93	88	87	86	90	91
刘旭阳	80	82	91	88	83	86	80
钱易铭	88	72	78	90	91	73	80

✚ 使用NumPy存储学生成绩信息

- ◆ 学生姓名，一维ndarray
- ◆ 课程名称，一维ndarray
- ◆ 课程成绩，二维ndarray

✚ 能否将这些数据组织在一个数据结构中？

- ◆ 将二维的数据的**行**与**学生姓名**数组关联
- ◆ 将二维的数据的**列**与**课程名称**数组关联

2. Pandas数据结构 | 什么是pandas

✚ pandas是基于python的数据分析工具包

- ◆ Series数据结构：一维数据
- ◆ DataFrame数据结构：二维数据和高维数据
- ◆ 汇集多种数据源数据、处理缺失数据
- ◆ 对数据进行切片、聚合和汇总统计
- ◆ 实现数据可视化

✚ 使用前需要导入

```
>>> import pandas as pd
```

```
>>> import numpy as np
```

#numpy的通用函数

✚ 为方便使用Series和DataFrame，将其导入本地命名空间

```
>>> from pandas import Series, DataFrame
```

2. Pandas数据结构 | Series数据结构

- Series 是类似于数组的一维数据结构，由索引（index）和值（values）两个相关联的数组组成：

1	158
2	170
3	178
...	...
index	values

- Series创建

Series ([data, index, ...])

data: Python的列表或Numpy的一维ndarray对象

index: 列表，若省略则自动生成0 ~n-1的序号标签

2. Pandas数据结构 | 例题3-1

✚ 创建5名篮球运动员身高的Series结构对象height，值是身高，索引为球衣号码（数字字符串作为索引）。

```
>>> height=Series([187,190,185,178,185],
index=['13','14','7','2','9'])
>>> height
13      187
14      190
7       185
2       178
9       185
dtype: int64
```

✚ 用字典创建Series对象，将字典的key作为索引：

```
>>> height=Series({'13':187,'14':190,'7':185,'2':178,
'9':185})
```

2. Pandas数据结构 | Series数据选取

选取类型	选取方法	说明
索引名选取	<code>obj [index]</code>	选取某个值
	<code>obj [indexList]</code>	选取多个值
基于位置选取	<code>obj [loc]</code>	选取某个值
	<code>obj [locList]</code>	选取多个值
	<code>obj [a:b, c]</code>	选取位置a~(b-1)以及c的值
条件筛选	<code>obj [condition]</code>	选取满足条件表达式的值

2. Pandas数据结构 | 例题3-2

✚ 使用例3-1创建的球员身高Series对象，实现球员数据的查询、增加、删除和修改。

1. 球员身高查询

```
>>> height['13']  
187
```

```
>>> height[ ['13','2','7'] ]  
13      187  
2       178  
7       185
```

```
>>> height[1:3]  
14      190  
7       185
```

```
>>> height[ height.values>=186 ]  
13      187  
14      190
```

```
>>> height  
13      187  
14      190  
7       185  
2       178  
9       185  
dtype: int64  
  
#同height[0]  
#同height[[0  
  
#检索标签序号1~3的球员身高  
#检索高于186的球员
```

2. Pandas数据结构 | 例题3-2

2. 球员身高修改

```
>>> height
13      187
14      190
7       185
2       178
9       185
dtype: int64
```

```
>>> height['13'] = 188
```

#将13号队员的身高修改为188

```
>>> height['13']
```

```
188
```

```
>>> height[1:3] = 160
```

#修改位置 1、2的数据，标量赋值

```
>>> height
```

```
13      188
```

```
14      160
```

```
7       160
```

```
2       178
```

```
9       185
```

2. Pandas数据结构 | 例题3-2

- ✚ Series不能直接添加新数据
- ✚ append()函数将两个Series拼接产生一个新的Series
 - ✚ 不改变原Series

3. 增加新球员

```
>>> a = Series([190,187], index=['23','5'])
```

```
>>> new = height . append( a )
```

```
>>> new
```

13	188
14	160
7	160
2	178
9	185
23	190
5	187

```
>>> height
```

13	188
14	160
7	160
2	178
9	185

2. Pandas数据结构 | 例题3-2

4. 删除离队球员

```
>>> new = height . drop( ['13', '9'] ) #删除13号球员数据
```

```
>>> new
```

```
14    160
7     160
2     178
```

Series的drop()函数缺省不删除原始对象的数据

```
>>> height
```

```
13    188
14    160
7     160
2     178
9     185
```

2. Pandas数据结构 | 例题3-2

```
>>> height
```

```
13    188
14    160
7     160
2     178
9     185
```

Series对象创建后，值可以修改，索引也修改，用新的列表替换即可。

5. 更改球员球衣号码

```
>>> height.index=[13,14,7,2,9]
```

#注意这里是数字索引

```
13    188
14    160
7     160
2     178
9     185
```

Series的索引为数字，基于位置序号访问需要使用iloc方式

```
>>> height.iloc[0]
```

```
188
```

思考与练习

1. 创建并访问Series对象。

1) 创建如下表的Series数据对象，其中a-f为索引；

a	b	c	d	e	f
30	25	27	41	25	34

2) 增加数据27，索引为g；

3) 修改索引d对应的值为40；

4) 查询值大于27的数据；

5) 删除位置为1-3的数据。

【提示】位置1-3的索引列表，可以用 `series.index[1:3]` 来得到。

2. Pandas数据结构 | DataFrame

✚ DataFrame 包括值 (values)、行索引 (index) 和列索引 (columns) 3部分:

	age	weight	height	列索引 (columns)
1	19	68	170	
2	20	65	165	
3	18	65	175	
4	19	58	168	
5	18	67	174	

行索引 (index)

值 (values)

✚ DataFrame 创建方法:

DataFrame (data, index = [...], columns=[...])

data: 列表或NumPy的二维ndarray对象

index, columns: 列表, 若省略则自动生成0 ~n-1的序号标签

2. Pandas数据结构 | 例题3-3

✚ 创建DataFrame对象students记录3名学生的信息

◆ 行索引为数字序号；列索引为age、weight和height

```
>>> data = [[19,170,68],[20,165,65],[18,175,65]]
```

```
>>> students = DataFrame(data, index=[1,2,3],  
columns=['age','height','weight'])
```

```
>>> students
```

	age	height	weight
1	19	170	68
2	20	165	65
3	18	175	65

data列表的每个元素初始化为DataFrame的一行值

2. Pandas数据结构 | DataFrame数

选取类型	选取方法	说明
索引名选取	<code>obj[col]</code>	选取某列
	<code>obj[colList]</code>	选取某几列
	<code>obj.loc[index, col]</code>	选取某行某列
	<code>obj.loc[indexList, colList]</code>	选取多行多列
位置序号选取	<code>obj.iloc[iloc, cloc]</code>	选取某行某列
	<code>obj.iloc[ilocList, clocList]</code>	选取多行多列
	<code>obj.iloc[a:b, c:d]</code>	选取a~(b-1)行, c~(d-1)列
条件筛选	<code>obj.loc[condition, colList]</code>	使用索引构造条件表达式 选取满足条件的行
	<code>obj.iloc[condition, clocList]</code>	使用位置序号构造条件表达式 选取满足条件的行

2. Pandas数据结构 | 例题3-4

✚ 使用例3-3创建的学生DataFrame对象，实现学生信息的查询、增加、删除和修改。

1. 学生信息查询

```
>>> students.loc[ 1, 'age'] #查询1号同学的年龄
```

```
19
```

```
>>> students.loc[[1,3],['height','weight']] #查询1、3号同学的身高和体重
```

```
      height  weight
1      170      68
3      175      65
```

```
>>> students.iloc[[0,2],[0,1]] #查询第0、2行的第0、1列的值
```

```
   age  height
1   19     170
3   18     175
```

```
>>> students
```

```
   age  height  weight
1   19     170      68
2   20     165      65
3   18     175      65
```

2. Pandas数据结构 | 例题3-4

```
>>> students
      age  height  weight
1     19     170     68
2     20     165     65
3     18     175     65
```

```
>>> students.loc[ : , ['height', 'weight']] # ":" 表示所有行的数据
```

```
      height  weight
1       170     68
2       165     65
3       175     65
```

```
>>> students[['height', 'weight']] #查询所有同学的身高和体重,
常用形式
```

```
      height  weight
1       170     68
2       165     65
3       175     65
```

```
>>> students.iloc[1:, 0:2] #通过切片抽取某些行和列的数据
```

```
      age  height
2     20     165
3     18     175
```

```
>>> students[1:3 ] #抽取行数据, 列的 ":" 可以省略
```

```
      age  height  weight
2     20     165     65
3     18     175     65
```


2. Pandas数据结构 | 例题3-4

```
>>> students
      age  height  weight
1     19     170     68
2     20     165     65
3     18     175     65
```

#筛选身高大于168的同学，显示其身高和体重值

```
>>> mask = students['height']>=168
```

```
>>> mask
```

```
1      True
2     False
3      True
Name: height, dtype: bool
```

#mask对象索引为2的行值为False，对应students索引为2的行未选中

```
>>> students.loc[ mask, ['height', 'weight'] ]
```

```
      height  weight
1         170     68
3         175     65
```

2. Pandas数据结构 | 例题3-4

```
>>> students
      age  height  weight
1     19     170     68
2     20     165     65
3     18     175     65
```

2. 增加学生信息

#列索引标签不存在，添加新列；存在则为值修改

```
>>> students['expense'] = [1500,1600,1200] #为学生
增加月消费数据
```

```
>>> students
```

	age	height	weight	expense
1	19	170	68	1500
2	20	165	65	1600
3	18	175	65	1200

DataFrame对象可以添加新的列，但不能直接增加新的行
增加行需要通过两个DataFrame对象的合并实现（见章节3.5）

2. Pandas数据结构 | 例题3-4

```
>>> students
```

	age	height	weight	expense
1	19	170	68	1500
2	20	165	65	1600
3	18	175	65	1200

3. 修改学生信息

```
>>> students['expense'] = 1000
```

#选中月消费列，用标量赋值

```
>>> students
```

	age	height	weight	expense
1	19	170	68	1000
2	20	165	65	1000
3	18	175	65	1000

```
>>> students.loc[1, :] = [21, 180, 70, 20]
```

#修改1号同学数据，使用列表赋值

```
>>> students
```

	age	height	weight	expense
1	21	180	70	20
2	20	165	65	1000
3	18	175	65	1000

2. Pandas数据结构 | 例题3-4

```
>>> students
```

	age	height	weight	expense
1	21	180	70	20
2	20	165	65	1000
3	18	175	65	1000

3. 修改学生信息

#筛选不合理数据，重新赋值

```
>>> students.loc[students['expense']<500, 'expense'] = 1200
```

```
>>> students
```

	age	height	weight	expense
1	21	180	70	1200
2	20	165	65	1000
3	18	175	65	1000

2. Pandas数据结构 | 例题3-4

4. 删除学生信息

缺省不修改原始数据对象

```
>>> students
```

	age	height	weight	expense
1	21	180	70	1200
2	20	165	65	1000
3	18	175	65	1000

```
>>> students.drop(1, axis=0) #axis=0表示行
```

	age	height	weight	expense
2	20	165	65	1600
3	18	175	65	1200

```
>>> students.drop('expense', axis=1) #删除expense列, axis=1  
表示列
```

	age	height	weight
1	21	180	70
2	20	165	65
3	18	175	65

```
>>> students.drop([1, 2], axis=0) #删除多行, 给出行索引名列表
```

	age	height	weight	expense
3	18	175	65	1000

2. Pandas数据结构 | 例题3-4

4. 删除学生信息

```
>>> students
```

	age	height	weight	expense
1	21	180	70	1200
2	20	165	65	1000
3	18	175	65	1000

如果需要直接删除原始对象的行或列，设置参数 `inplace=True`

#删除多列，并修改students对象

```
>>> students.drop(['age', 'weight'], axis=1,
inplace=True)
```

	height	expense
1	180	1200
2	165	1000
3	175	1000

```
>>> students
```

	height	expense
1	180	1200
2	165	1000
3	175	1000

课后练习

1. 创建并访问DataFrame对象。

- a) 创建 3×3 DataFrame数据对象：数据内容为1-9；行索引为字符a, b, c；列索引为字符串one, two, three；
- b) 查询列索引为two和three两列数据；
- c) 查询第0行、第2行、第0列、第2列数据；
- d) 筛选第1列中值大于2的所有行数据，另存为data1对象；
- e) 为data1添加一列数据，列索引为four，值都为10；
- f) 将data1所有值大于9的数据修改为8；
- g) 删除data1中第0行和第1行数据。

【提示】

- 1) 生成数据，使用numpy的arange()函数和reshape()函数；
- 2) 使用 `data>9`生成布尔型的DataFrame，用于整个DataFrame的数据过滤。

3.数据文件读写

3. 数据文件读写 | 支持的文件格式?

✚ Pandas支持多种格式的数据导入和导出

- CSV、TXT、Excel、HTML等文件格式
- MySQL、SQLServer等数据库格式
- JSON等Web API数据交换格式

✚ 常用CSV、TXT、Excel 3种文件的数据读写

3. 数据文件读写 | 读取CSV文件

✚ CSV是一种特殊的文本文件，通常使用：

✚ 逗号：字段之间的分隔符

✚ 换行符：记录之间分隔符

✚ 读取CSV文件方法

```
read_csv( file, sep=',', header='infer', index_col=None,  
          names=None, skiprows,...)
```

参数说明：

file	字符串，文件路径和文件名
sep	字符串，每行各数据之间的分隔符，默认为 ‘,’
header	header=None，文件中第一行不是列索引
index_col	数字，用作行索引的列
name	列表，定义列索引，默认文件中第一行为列索引
skiprows	整数或列表，需要忽略的行数或需要跳过的行号列表

3. 数据文件读写 | 例题3-5

✚ 从students1.csv文件读出数据，保存为DataFrame对象

```
>>> student = pd.read_csv( 'data\student1.csv ' )
```

```
>>> student[-3:] #显示最后3条数据
```

	序号	性别	年龄	身高	体重	省份	成绩
2	3	male	22	180	62	FuJian	57
3	4	male	20	177	72	LiaoNing	79
4	5	male	20	172	74	ShanDong	91



文件中每个同学已有序号，读取时作为行索引

```
>>> student = pd.read_csv( 'data\student1.csv ',  
index_col = 0 )
```

```
>>> student[:3] #从开始到序号为3的行
```

	性别	年龄	身高	体重	省份	成绩
序号						
1	male	20	170	70	LiaoNing	71
2	male	22	180	71	GuangXi	77
3	male	22	180	62	FuJian	5

3. 数据文件读写 | 文本文件编码格式

✚ 文本文件包含中文，使用“UTF-8”编码格式保存

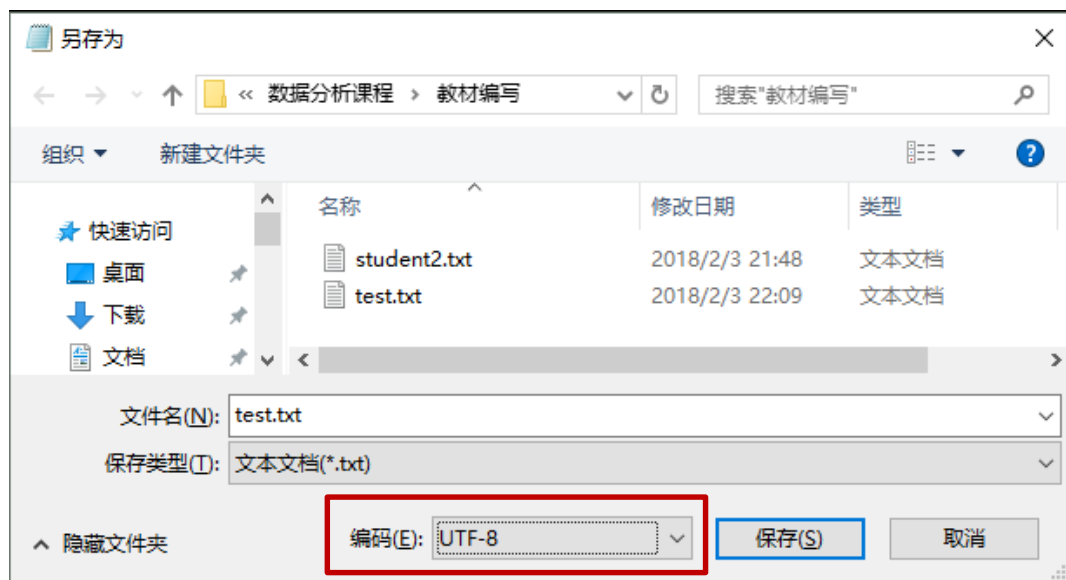
◆ 其他格式，Python 3读取时报“utf-8”错误

✚ 保存方法

◆ 用“记事本”程序打开文件，选择“文件”的“另存为”菜单

◆ 点击最下方的“编码”下拉列表

◆ 选择“UTF-8” → “保存”



3. 数据文件读写 | 读取文本文件

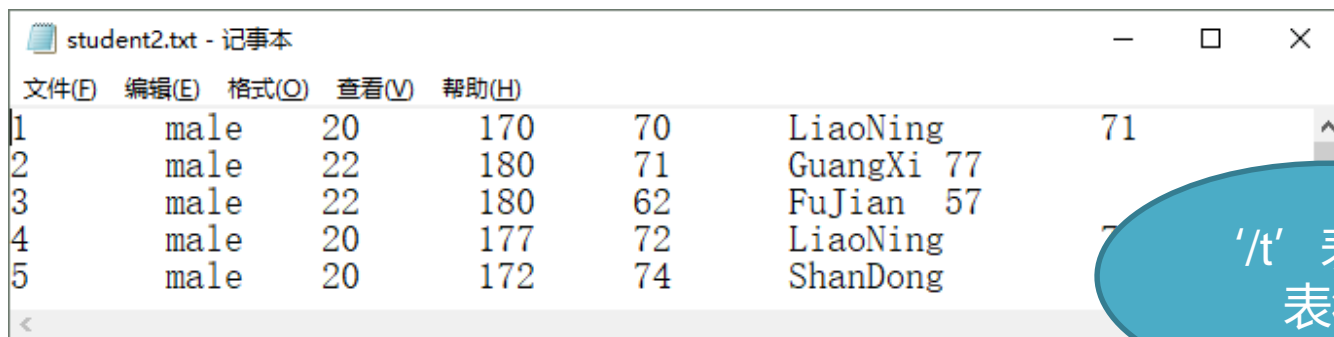
- ✚ 不是以逗号隔开的文本文件，读取时需要设置分隔符参数sep
 - ✚ 分隔符既可以是指定字符串，也可以是正则表达式
- ✚ 常用的通配符

通配符	描述
\s	空格等空白字符
\S	非空白字符
\t	制表符
\n	换行符
\d	数字
\D	非数字字符

3. 数据文件读写 | 例题3-6

✚ 从student2.txt文件中读取数据，保存至DataFrame对象

- ◆ student2.txt以制表符为分割符
- ◆ 文件中不包含列索引



1	male	20	170	70	LiaoNing	71
2	male	22	180	71	GuangXi	77
3	male	22	180	62	FuJian	57
4	male	20	177	72	LiaoNing	7
5	male	20	172	74	ShanDong	

'\t' 表示制表符

```
>>> colNames = ['性别','年龄','身高','体重','省份','成绩']
```

```
>>> student = pd.read_csv('data\student2.txt', sep='\t', index_col=0,  
                           header=None, names= colNames )
```

```
>>> student[:2]
```

```
  性别  年龄  身高  体重  省份  成绩  
序号  
1  male  20  170  70  LiaoNing  71  
2  male  22  180  71  GuangXi  77
```

指明:

- 1) 文件中不包括列索引
- 2) 列索引名由指定列表给出

3. 数据文件读写 | 保存CSV文件

数据保存到文件

to_csv (file, sep, mode, index, header,...)

参数说明:

file	文件路径和文件名
sep	分隔符, 默认为逗号
mode	导出模式, w为导出到新文件, a为追加到现有文件
index	是否导出行索引, 默认为True
header	是否导出列索引, 默认为True

3. 数据文件读写 | 例题3-7

🚩 新建DataFrame对象student, 并将数据保存到out.csv文件

```
>>> data = [[19,68,170],[20,65,165],[18,65,175]]
>>> student =DataFrame( data,index=[1,2,3],
columns=['age','weight','height'] )
>>> student . to_csv('out.csv', mode='w', header=True,
index=False)
```


3. 数据文件读写 | 读取Excel文件

- ✚ 从Excel文件中读取数据的函数类似CSV文件

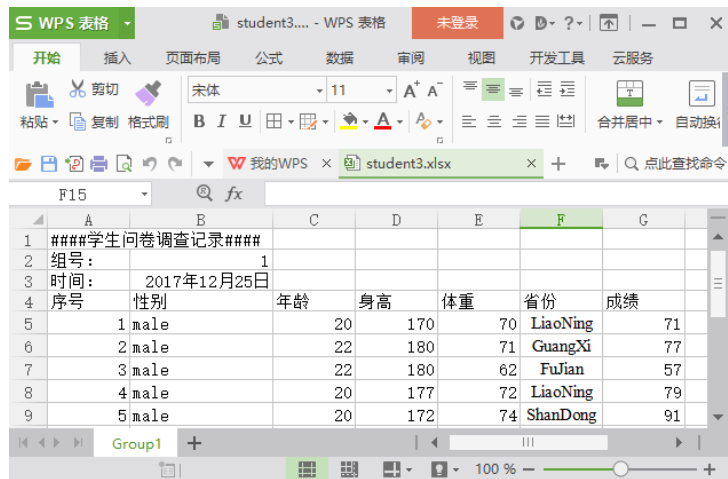
 - ✚ 需给出数据所在的Sheet表单名

- ✚ 读取方法：

 - `read_excel(file, sheetname, ...)`

3. 数据文件读写 | 例题3-8

✚ 从如图所示的student3.xlsx文件 “Group1” 页中读取数据，保存至 DataFrame对象



序号	性别	年龄	身高	体重	省份	成绩
1	male	20	170	70	LiaoNing	71
2	male	22	180	71	GuangXi	77
3	male	22	180	62	Fujian	57
4	male	20	177	72	LiaoNing	79
5	male	20	172	74	ShanDong	91

#将序号列作为index，跳过前3行

```
>>> student = pd.read_excel('data\student3.xlsx',  
                             'Group1', index_col=0, skiprows=3 )
```

```
>>> student[:2]
```

	性别	年龄	身高	体重	省份	成绩
序号						
1	male	20	170	70	LiaoNing	71
2	male	22	180	71	GuangXi	77

kiprows = 3, 忽略前3行，即0、1、2行
忽略指定行：给出行号列表
如:忽略第2、3行，skiprows=[1,2]

思考与练习

1. 创建 50×7 的DataFrame对象，数据为 [10,99]之间的随机整数；columns为字符a-g；将DataFrame对象保存到csv文件中。

【提示】使用NumPy的随机生成函数randint()生成数据。

2. 海伦一直使用在线交友网站寻找适合的约会对象, 她将交友数据存放在datingTestSet.xls文件中。

- 1) 从文件中读取有效数据保存到Dataframe对象中，跳过所有文字解释行；
- 2) 列索引名设为 ['flymiles','videogame','icecream','type']；
- 3) 显示读取到的前面5条数据；
- 4) 显示所有'type'为'largeDoses'的数据；
- 5) 将平均每周玩视频游戏时间超过10的数据都改成10；
- 6)

4.数据清洗

4. 数据清洗 | 数据清洗?

✚ 数据清洗对采集的数据进行重新审查和校验

◆ 删除重复信息、纠正存在的错误，保证数据一致性

✚ 例：数据缺失问题、数据错误问题、数据重复问题

The diagram illustrates data cleaning issues using two Excel spreadsheets. Callouts point to specific cells:

- 数据缺失 (Data Missing):** Points to empty cells in the first spreadsheet (e.g., G2, H5, H6).
- 数据错误 (Data Error):** Points to incorrect values in the second spreadsheet (e.g., G4=12, I4=5, E7=20).
- 数据重复 (Data Repeat):** Points to the duplicate row (row 9) in the second spreadsheet.

	A	B	C	D	E	F	G	H	I	J
1	序号	性别	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学
2	1	male	20	170	70	LiaoNing		800	5	4
3	2	male	22	180	71	GuangXi	77	1300	3	4
4	3	male		180	62	FuJian	57	1000	2	4
5	4	male	20	177	72	LiaoNing	79	900	4	4
6	5	male	20	172		ShanDong	91		5	5
7	6	male	20	179	75	YunNan	92	950	5	5
8	7	female	21	166	53	LiaoNing	80	1200	4	5
9	8	female	20	162	47	AnHui	78	1000	4	4
10	9	female	20	162	47	AnHui	78	1000	4	4
11	10	male	19	169	76	HeiLongJiang	8			

	A	B	C	D	E	F	G	H	I	J
1	序号	性别	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学
2	1	female	21	162	49	YunNan	89	800	5	5
3	2	female	20	164	53	GuiZhou	79	1000	4	5
4	3	female	20	166	43	HaiNan	12	1000	5	5
5	4	female	21	162	52	TianJin	86	500	5	5
6	5	male	20	175	73	AnHui		700	4	4
7	6	male	19	172	20	ShanDong	75	900	5	5
8	7	male	21	178	79	GuiZhou	74	650	4	5
9	8	female	20	163	54	XinJiang	79	1400	4	5
10	9	female	21	160	44	ShangHai	61	600	5	5
11	10	female	19	163	48	GuiZhou	56	700	2	5

4. 数据清洗 | 缺失数据处理

✚ 主要有数据滤除和数据填充两类方法

1. 数据滤除

```
obj.dropna(axis, how, thresh, ...)
```

参数说明：

axis	0表示按行滤除，1按列滤除，默认axis=0
how	'all'表示滤除全部值都为NaN的行或列
thresh	只留下有效数据大于等于thresh值的行或列

4. 数据清洗 | 例题3-9

- ✚ 从文件studentsInfo.xlsx的“Group1”表单中读取数据，滤除部分缺失数据，填充部分缺失数据。

```
>>> stu = pd.read_excel('data\studentsInfo.xlsx',  
                        'Group1', index_col=0)
```

```
>>> stu
```

性别	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学
----	----	----	----	----	----	------	------	------

序号

1	male	20.0	170	70.0	LiaoNing	NaN	800.0	5	4
2	male	22.0	180	71.0	GuangXi	77.0	1300.0	3	4
3	male	NaN	180	62.0	FuJian	57.0	1000.0	2	4
4	male	20.0	177	72.0	LiaoNing	79.0	900.0	4	4
5	male	20.0	172	NaN	ShanDong	91.0	NaN	5	5

.....

- ✚ 缺失数据被表示为NaN，赋值时使用np.nan

- ◆ 样本容量大，忽略缺失行
- ◆ 样本容量较小，采用合适的值来填充

4. 数据清洗 | 例题3-9续

```
>>> stu.dropna() #缺省删除包含有缺失值的行 (序号1、3、5的行被滤除)
```

性别	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学	
序号									
2	male	22.0	180	71.0	GuangXi	77.0	1300.0	3	4
4	male	20.0	177	72.0	LiaoNing	79.0	900.0	4	4
6	male	20.0	179	75.0	YunNan	92.0	950.0	5	5
.....									

```
>>> stu.dropna(thresh=8) #保留有效数据个数≥8的行(序号5的行被滤除)
```

性别	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学
序号								
1	male	20.0	170	70.0	LiaoNing	NaN	800.0	5
2	male	22.0	180	71.0	GuangXi	77.0	1300.0	3
3	male	NaN	180	62.0	FuJian	57.0	1000.0	2
4	male	20.0	177	72.0	LiaoNing	79.0	900.0	4
6	male	20.0	179	75.0	YunNan	92.0	950.0	5
.....								

4. 数据清洗 | 缺失数据处理

2. 数据填充

填充有两种基本思路：

- ◆ 用默认值填充
- ◆ 用已有数据的均值/中位数来填充

格式：

```
obj.fillna (value, method, inplace...)
```

参数说明：

value	填充值，可以是标量、字典、Series或DataFrame
method	'ffill'：同列前一行数据填充缺失值，'bfill'：用后一行数据填充
inplace	是否修改原始数据的值，默认为False，产生一个新的数据对象

4. 数据清洗 | 例题3-9续

✚ 案例3-1 “年龄” 和 “体重” 列有缺失数据

- ◆ 年龄用默认值填充
- ◆ 体重用平均值来填充

✚ 列填充，构造{列索引名:值}形式的字典对象作为实际参数

```
>>> stu.fillna( { '年龄':20, '体重':stu['体重'].mean() } )
```

	性别	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学
--	----	----	----	----	----	----	------	------	------

1	male	20.0	170	70.000000	LiaoNing	NaN	800.0	5	4
2	male	22.0	180	71.000000	GuangXi	77.0	1300.0	3	4
3	male	20.0	180	62.000000	FuJian	57.0	1000.0	2	4
4	male	20.0	177	72.000000	LiaoNing	79.0	900.0	4	4
5	male	20.0	172	63.666667	ShanDong	91.0	NaN	5	5
.....									

4. 数据清洗 | 例题3-9续

✚ 用前一行数据替换当前行的空值

```
>>> stu.fillna(method='ffill')
```

#每个空值用上一行同列的值填充

	性别	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学
序号									
1	male	20.0	170	70.0	LiaoNing	NaN	800.0	5	4
2	male	22.0	180	71.0	GuangXi	77.0	150		
3	male	22.0	180	62.0	FuJian	57.0	1000		
4	male	20.0	177	72.0	LiaoNing	79.0	900.0		
5	male	20.0	172	72.0	ShanDong	91.0	900.0	5	5
.....									

没有前一行，
不填充

✚ 填充操作产生新的数据对象，原始数据不会被修改

✚ 直接填充原始数据中的缺失值

✚ fillna() 增加参数设置: inplace=True

4. 数据清洗 | 去重 例题3-10

✚ 例题：从文件studentsInfo.xlsx的 “Group1”页中读取数据，去除重复数据。

✚ 去重函数

```
obj.drop_duplicates()
```

```
>>> stu = pd.read_excel('data\studentsInfo.xlsx',  
'Group1', index_col=0)
```

```
>>> stu.drop_duplicates() #去重（序号9的行被滤除）
```

序号	性别	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学
1	male	20.0	170	70.0	LiaoNing	NaN	800.0	5	4
2	male	22.0	180	71.0	GuangXi	77.0	1300.0	3	4
.....									
8	female	20.0	162	47.0	AnHui	78.0	1000.0	4	4
10	male	19.0	169	76.0	HeiLongJiang	88.0	1100.0	5	5

思考与练习

1. 数据清洗。

- 1) 从studentsInfo.xlsx 文件的 “Group1” 表单中读取数据；
- 2) 将 “案例教学” 列数据值全改为NaN；
- 3) 滤除每行数据中缺失3项以上（包括3项）的行；
- 4) 滤除值全部为NaN的列；

2. 数据填充。

- 1) 使用习题1的数据；
- 2) 使用列的平均值填充 “体重” 和 “成绩” 列的NaN数据；
- 3) 使用上一行数据填充 “年龄” 列的NaN数据；
- 4) 使用 “中位数” 填充 “生活费用” NaN数据。

【提示：】使用df[“生活费用”].median() 计算中位数。

5.数据规整化

5.数据规整化 | 数据合并

同一实体的数据来自不同的业务系统

- ◆ 学生的基本信息来自教务系统
- ◆ 学生刷卡数据来自一卡通系统

相同实体的多个数据集

- ◆ 案例3-1中反馈数据存放在5张Excel表中

数据合并可分为两种处理方式

- ◆ 行数据追加
- ◆ 列数据连接

5.数据规整化 | 数据合并 | 行数据追加 例题3-11

🚦 将新同学的信息（右表）添加学生基本信息（左表）中

学号	姓名	专业
202003101	赵成	软件工程
202005114	李斌丽	机械制造
202009111	孙武一	工业设计

学号	姓名	专业
202003103	王芳	软件工程
202005116	袁一凡	工业设计

#分别建立原有数据和新数据的DataFrame对象

```
>>> colName = ['学号', '姓名', '专业']      #列索引
>>> data1 = [ ['202003101', '赵成', '软件工程'], ['202005114', '李斌丽', '机械制造'],
               ['202009111', '孙武一', '工业设计'] ]      #值列表
>>> stu1 = DataFrame( data1, columns=colName )      #行索引自动生成
>>> data2 = [['202003103', '王芳', '软件工程'], ['202005116', '袁一凡', '工业设计']]
>>> stu2 = DataFrame( data2, columns=colName )
```


5.数据规整化 | 数据合并 | 行数据追加 例题3-11续

✚ 原数据的列与新增数据的列完全相同

◆ 轴向连接: **concat()函数**

```
>>> newstu = pd.concat([stu1, stu2], axis=0)      #axis=0, 表示按行进行数据追加
```

```
>>> newstu
```

	学号	姓名	专业
0	202003101	赵成	软件工程
1	202005114	李斌丽	机械制造
2	202009111	孙武一	工业设计
0	202003103	王芳	软件工程
1	202005116	袁一凡	工业设计

5.数据规整化 | 数据合并 | 列数据连接 例题3-12



分析各专业学生上图书馆的习惯

- ◆ 去图书馆的信息保存在学生刷卡表中
- ◆ 将教务数据表与刷卡信息表拼接起来

ID	刷卡地点	刷卡时间	消费金额
202003101	一食堂	20180305 11:45	14.2
104574	教育超市	20180307 17:30	25.2
202003103	图书馆	20180311 18:23	
202005116	图书馆	20180312 08:32	
202005114	二食堂	20180312 17:08	12.5
202003101	图书馆	20180314 13:45	

■ 教务表“学号”与一卡通表“ID”表示相同概念

■ 比较两张表每行的“学号”与“ID”（键）进行拼接

```
merge ( x,y,how,left_on,right_on ... )
```

参数说明:

x 左数据对象

y 右数据对象

how 数据对象连接的方式, 'inner'、'outer'、'left'、'right'

left_on 左数据对象用于连接的键

right_on 右数据对象用于连接的键

5.数据规整化 | 数据合并 | 列数据连接 例题3-12续

✚ 参数how定义了四种合并方式

- 1) inner: 内连接, 拼接两个数据对象中键值交集的行, 其余忽略
- 2) outer: 外连接, 拼接两个数据对象中键值并集的行
- 3) left: 左连接, 取出x的全部行, 拼接y中匹配的键值行。
- 4) right: 右连接, 取出y的全部行, 拼接x中匹配的键值行。

✚ 2、3或4种合并方法

◆ 当某列数据不存在则自动填充NaN

✚ 本例中分析学生去图书馆的信息

- ◆ 采用 “left”方式合并 “学生表” 和 “一卡通表”
- ◆ 忽略一卡通记录中非学生记录

5.数据规整化 | 数据合并 | 列数据合并 例题3-12续

```
>>> cardcol = ['ID','刷卡地点','刷卡时间','消费金额']

>>> data3 = [ ['202003101','一食堂','20180305 1145',14.2], ['104574','教育超市',
', '20180307 1730',25.2], ['202003103','图书馆','20180311 1823'], ['202005116','图书馆',
', '20180312 0832'], ['202005114','二食堂','20180312 1708',12.5], ['202003101','图书馆',
', '20180314 1345']]
```

```
>>> card = DataFrame( data3, columns=cardcol )           #创建一卡通数据对象
#左连接
```

```
>>> pd.merge(newstu,card, how='left', left_on='学号', right_on='ID')
```

	学号	姓名	专业	ID	刷卡地点	刷卡时间	消费金额	
0	202003101	赵成	软件工程	202003101	一食堂	20180305 1145	14.2	
1	202003101	赵成	软件工程	202003101	图书馆	20180314 1345	NaN	
2	202005114	李斌丽	机械制造	202005114	二食堂	20180312 1708	12.5	
3	202009111	孙武一	工业设计	NaN	NaN	NaN	NaN	
4	202003103	王芳	软件工程	202003103	图书馆	20180311 1823	NaN	
5	202005116	袁一凡	工业设计	202005116	图书馆	20180312 0832	NaN	

5.数据规整化 | 数据排序

 Series和DataFrame 都支持排序

- ◆ 按照列数据值排序
- ◆ 按列数据生成排名

1. DataFrame 值排序

obj.sort_values(by, ascending,inplace...)

参数说明:

by	列索引, 定义用于排序的列
ascending	排序方式, True为升序, False为降序
inplace	是否修改原始数据数据, True为修改, 默认为False

5.数据规整化 | 例题3-13续

✚ 指定多个列排序，如：by=['身高','体重']

◆ 先按“身高”排序，

◆ 若某些行的“身高”相同，这几行再按“体重”排序

```
>>> stu.sort_values(by=['身高','体重'], ascending=True)
```

	性别	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学
24	female	21	160	49	HeBei	59	1100	3	5
28	female	22	160	52	ShanXi	73	800	3	4
29	female	20	161	51	GuangXi	80	1250	5	5
27	female	21	162	49	ShanDong	65	950	4	4

5.数据规整化 | 数据排序

2. 排名

✚ 排名给出每行的名次

◆ 定义等值数据的处理方式，如并列名次取最小值或最大值，也可以取均值。

✚ 排名函数形式

```
obj.rank(axis, method, ascending, ...)
```

参数说明：

axis	0：按行数据排名，1：按列数据排名
method	并列时取值方式：min、max、mean
ascending	排序方式，True为升序，False为降序

5.数据规整化 | 例题3-14

✚ 对例3-13的成绩数据降序排名，增加“成绩排名”列。

```
>>> stu['成绩排名'] = stu['成绩'].rank(method='min', ascending=False)
```

```
>>> stu
```

序号	性别	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学	成绩排名
21	female	21	165	45	ShangHai	93	1200	5	5	2.0
22	female	19	167	42	HuBei	89	800	5	5	4.0
23	male	21	169	80	GanSu	93	900	5	5	2.0
24	female	21	160	49	HeBei	59	1100	3	5	10.0
...										

排名结果显示，序号为21和22的两位学生**并列第2名，第3名空缺。**

课后作业

1. 数据合并。

1) 从studentsInfo.xlsx的“Group3”页读取数据，将序号、性别、年龄项保存到data1对象；

2) 从studentsInfo.xlsx的“Group3”页读取数据，将序号、身高、体重、成绩项保存到data2对象；

3) 将data2合并到data1中，连接方式为内连接。

2. 数据排序和排名。

1) 使用练习1最后合并的数据；

2) 按月生活费对数据升序排序；

3) 按身高对数据降序排名，并列取值方式设置为min。

6.统计分析

6.统计分析 | 通用函数与运算

📌 DataFrame、Series、标量之间的算术运算

运算符	描述
df.T	DataFrame转置
df1 + df2	按照索引和列相加，得到并集，NaN填充
df1.add(df2, fill_value=0)	按照索引和列相加，NaN用指定值填充
df1.add/sub//mul/div	四则运算
df - sr	DataFrame的所有行同时减去Series
df * n	所有元素乘以n

📌 DataFrame元素级的函数运算

- ◆ 通过numpy的一元通用函数ufunc实现
- ◆ 格式为：np.ufunc(df)

函数	描述
abs、fabs	计算整数、浮点数或复数的绝对值
sqrt	计算各元素的平方根
square	计算各元素的平方
exp	计算各元素的指数

6.统计分析 | 例题3-15

✚ 分析例3-13中同学的“身体质量”，即BMI（Body Mass Index）指数

◆ 世界卫生组织对于BMI的定义：

$$\text{BMI (kg/m}^2\text{)} = \text{体重} / \text{身高}^2$$

我国体质评判标准为：

BMI≤18.5, 过轻; 18.5~24, 正常; 24~28, 偏胖; ≥28肥胖。

```
>>> stu['BMI'] = stu['体重'] / ( np.square(stu['身高']/100) )  
#增加BMI列
```

```
>>> stu
```

性别 序号	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学	
21	female	21	165	45	ShangHai	93	1200	5	5 16.52
22	female	19	167	42	HuBei	89	800	5	5 15.059701
23	male	21	169	80	GanSu	93	900	5	5 28.010224

对列的每个元素计算sqrt值

结论：两位女同学体重偏轻，男同学达到了肥胖

6.统计分析 | 统计函数

pandas的常用统计函数

函数	描述
<code>sr.value_counts()</code>	Series统计频率
<code>sr.describe()</code>	返回基本统计量和分位数
<code>sr1.corr(sr2)</code>	sr1与sr2的相关系数
<code>df.count()</code>	统计每列数据个数
<code>df.max()</code> 、 <code>df.min()</code>	最大值和最小值
<code>df.idxmax()</code> 、 <code>df.idxmin()</code>	最大值、最小值对应的索引
<code>df.sum()</code>	按行或列求和
<code>df.mean()</code> 、 <code>df.median()</code>	计算均值、中位数
<code>df.quantile()</code>	计算给定的四分位数
<code>df.var()</code> 、 <code>df.std()</code>	计算方差、标准差
<code>df.mode()</code>	计算众数
<code>df.cumsum()</code>	从0开始向前累加各元素
<code>df.cov()</code>	计算协方差矩阵
<code>pd.crosstab(df[col1],df[col2])</code>	pandas函数，交叉表，计算分组的频率

6.统计分析 | 例题3-16

对例3-13同学数据中的“成绩/月生活费”进行统计分析

```
>>> stu['成绩'].mean() #计算成绩的平均值
```

```
78.0
```

```
>>> stu['月生活费'].quantile([.25, .75]) #计算月生活费的上、下四分位数
```

```
0.25      800.0
```

```
0.75     1175.0
```

```
Name: 月生活费, dtype: float64
```

描述统计函数describe() 一次计算多项统计值

```
>>> stu[['身高', '体重', '成绩']].describe() #对身高体重和成绩3列数据描述统计
```

	身高	体重	成绩
count	10.000000	10.0000	10.000000
mean	165.500000	55.1000	78.000000
std	6.381397	12.8448	14.476034
min	160.000000	42.0000	59.000000
25%	161.250000	49.0000	65.750000
50%	163.500000	51.5000	76.500000
75%	167.750000	53.5000	92.000000
max	181.000000	80.0000	98.000000

6.统计分析 | 分组

✚ 根据某些索引将数据对象划分为多个组

◆ 对每个分组进行排序或统计计算

```
grouped = obj.groupby(col)
```

```
grouped.aggregate({'col1':f1, 'col2':f2,...})
```

参数说明:

col	统计列索引名
f	Numpy的聚合函数名, 如: sum、mean、std等

6.统计分析 | 例题3-17

对例3-13同学数据中的“身高、月生活费”按“性别”和“年龄”进行分组分析

```
>>> grouped = stu.groupby(['性别', '年龄'])  
>>> grouped.agg({'身高': np.mean, '月生活费': np.max})
```

		身高	月生活费
性别	年龄		
female	19	167	800
	20	161	1250
	21	160	1300
	22	160	800
male	21	169	900

字典格式: { }

6.统计分析 | 例题3-17续

🌈 统计函数crosstab() 类似Excel交叉表

◆ 按照给定的第一列分组，对第二列计数

```
>>> pd.crosstab( stu['性别'], stu['月生活费'] )
```

#pandas函数

统计列

月生活费

性别

female

male

700

800

900

950

1100

1200

1250

1300

1

2

0

1

1

1

1

1

0

1

1

0

0

0

0

0

分组列

6.统计分析 | 相关性分析

✚ 研究不同总体之间是否存在依存关系

◆ 绘制散点图矩阵，直观地观察列之间的相关性

- `pd.plotting.scatter_matrix(data,diagonal='kde',color='k')` #绘图

◆ 计算样本之间的相关系数 r 推断总体的相关程度

◆ 相关系数具有以下特征

- 相关系数的值介于-1与+1之间；
- $r=1$ ：两个总体正相关； $r=0$ ：不相关； $r=-1$ ：负相关；
- $|r|<0.3$ ：低度相关； $0.3\leq|r|<0.8$ ：中等相关； $0.8\leq|r|<1$ ：高度相关。

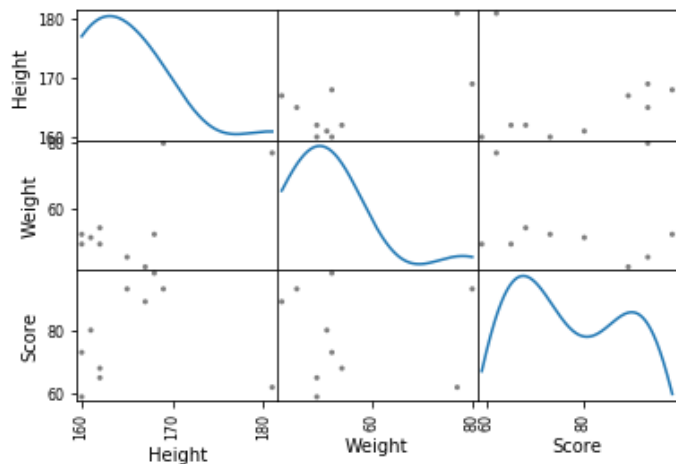
✚ 当样本容量较大 (≥ 30) 时，相关性分析判断准确性较高

✚ DataFrame相关性分析函数

`DataFrame.corr()`

6.统计分析 | 例题3-18

✚ 分析例3-13中同学身高、体重与成绩之间的相关性



#两列数据之间的相关性

```
>>> stu['身高'].corr( stu['体重'] )  
0.67573990985276822
```

#多列数据之间的相关性

```
>>> stu[ ['身高', '体重', '成绩'  
'] ].corr()
```

	身高	体重	成绩
身高	1.000000	0.675740	0.080587
体重	0.675740	1.000000	-0.072305
成绩	0.080587	-0.072305	1.000000

分析表明:

- 1) 身高与体重有一定关系，但不是很高
- 2) 两者都与成绩没有相关性

6.统计分析 | 案例-调查反馈

✚ 案例3-1对50名学生进行抽样调查，反馈数据保存在studentInfo.xlsx文件的5张表中。综合5组数据实现以下分析目标：

- 男、女生对《数据科学》课程的兴趣程度和成绩的变化趋势；
- 学生来自的省份以及性别与成绩是否存在关系；
- 学生身高、体重达标状况。

✚ 步骤1：导入所需的方法库

```
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
```

6.统计分析 | 案例-调查反馈

🚦 步骤2: 从Excel文件的5张表中读取数据, 拼接为一个DataFrame对象

#从Excel文件的5张表中读取数据

```
df1=pd.read_excel('data\studentsInfo.xlsx','Group1',index_col=0)
df2=pd.read_excel('data\studentsInfo.xlsx','Group2',index_col=0)
df3=pd.read_excel('data\studentsInfo.xlsx','Group3',index_col=0)
df4=pd.read_excel('data\studentsInfo.xlsx','Group4',index_col=0)
df5=pd.read_excel('data\studentsInfo.xlsx','Group5',index_col=0)
```

#按行追加形式, 拼接数据集

```
stu = pd.concat([df1,df2,df3,df4,df5], axis = 0)
print( 'Data Size:', stu.shape )
```

6.统计分析 | 案例-调查反馈表

- 步骤3: 去除完全重复以及缺失项较多 (≥ 2) 的数据行, 检测是否还有缺失数据

```
stu.drop_duplicates(inplace = True) #去除重复行, 更新方式
stu.dropna(thresh=8,inplace = True ) #去除有缺失数据行, 更新方式
print( 'Data Size after drop:', stu.shape )
print( "Nan Columns:\n",stu.isnull().any() ) #缺失数据列检测
```

- `stu.isnull()`检测对象每个值是否是NaN, 得到布尔型 DataFrame。
- `any()`函数缺省**按列检测**是否存在为False的值, 得到布尔型的Series对象。

```
Nan Columns:
性别      False
年龄      True
身高      False
体重      False
省份      False
成绩      True
月生活费  False
课程兴趣  False
案例教学  False
dtype: bool
```

结果表明 “年龄” 和 “成绩” 列存在缺失值需要填充

6.统计分析 | 案例-调查反馈表分析

- 步骤4：填充缺失值：成绩按照平均分填充；接受调查同学为二年级，用默认值“20”来填充

```
stu.fillna({'年龄':20, '成绩':stu['成绩'].mean()},  
           inplace=True )  
print( "Nan Columns:\n",stu.isnull().any() )
```

- 步骤5：将同学数据按照“成绩”排序，统计优秀（ ≥ 90 ）和不合格（ < 60 ）学生个数。并分别计算优秀与不合格同学的平均课程兴趣度，以及全体同学课程的平均分与课程兴趣度

#按照成绩排序

```
stu_grade = stu.sort_values(by='成绩', ascending=False)  
ex = (stu_grade['成绩']>=90 ).sum() #计算优秀人数  
fail = (stu_grade['成绩']<60 ).sum() #计算不及格人数  
print("Excellent: {}, Fail: {}".format(ex,fail))
```


6.统计分析 | 案例-调查反馈表分析

✚ 条件表达式`stu_grade['成绩']>=90` 得到布尔型Series对象, `sum()` 函数统计其中True的个数

◆ 优秀9个, 不及格4个

```
ex_mean = stu_grade[0:9][['成绩','课程兴趣']].mean()    #前9行优秀
total_mean = stu_grade[['成绩','课程兴趣']].mean()
fail_mean = stu_grade[-4:][['成绩','课程兴趣']].mean()  #后4行不及格
print("ex_mean:\n", ex_mean, "\ntotal_mean\n",total_mean,
      "\nfail_mean\n", fail_mean)

#计算两列相关度
print( stu_grade['成绩'].corr(stu_grade['课程兴趣']) )
```

结果表明:

- 1) 3类统计 “成绩” 均值为93.8、76.3和46.0, 而 “课程兴趣” 的均值为5.0、4.2和3.0
- 2) 从趋势上看, 大学课程学习的成绩与兴趣的变化具有一致性
- 3) 两列数据的相似度为0.44, 说明从个体上两者相关度并不是特别高, 也有可能存在错误数据

6.统计分析 | 案例-调查反馈表分析续

✚ 步骤6：分析性别、省份与成绩是否存在相关性，由于性别和省份数据均为字符型，无法用corr()函数来计算。简单的方法是分组计算均值：

```
sex_grouped = stu.groupby(['性别'])
sex_counts = sex_grouped.count()          #统计每个分组的行数
#分组统计成绩平均值
sex_mean = stu.groupby(['性别']).aggregate( {'成绩':np.mean } )
print(sex_counts, '\n', sex_mean)
pro_counts = stu.groupby(['省份']).count()
pro_mean = stu.groupby(['省份']).aggregate( {'成绩':np.mean } )
print(pro_counts, '\n', pro_mean)
```

结果表明：

- 1) 男、女同学的各24人，平均成绩分别是79.0和73.7
- 2) 说明男同学在该门课程中成绩更好一些。
- 3) 按省份分组，各省平均分相去甚远，但观察每个省份只有1~3名同学，分组样本太少，导致分析结果不具参考价值。

6.统计分析 | 案例-调查反馈表分析续

✚ 步骤7：计算同学的BMI值，找出各个四分位数，并与国家标准进行比较：

```
stu['BMI'] = stu['体重'] / ( np.square(stu['身高']/100) )  
#计算四分位数  
print( stu['BMI'].quantile( [.25,0.5,.75] ) )  
#计算BMI值>28的个数  
print('BMI>28 肥胖人数:', (stu['BMI']>=28 ).sum() )
```

结果说明：

- 1) 25%的同学BMI值为18.6，体重偏轻；
- 2) 75%的BMI值为23.4，都在正常范围内。
- 3) 1位同学BMI超过了28，属于肥胖。

综合练习题

✚ 根据某系的实验教学计划，完成以下分析：

1. 读取DataScience.xls文件数据，创建为data数据对象；
2. 查询df的数据量和基本结构（df.index，df.columns）；
3. 查询df中是否含有NaN数据？将含有NaN数据的行导出为数据文件pre.csv，判断采用何种数据清洗模式：填充、删除或手工填充；
4. 查询课程名称、实验项目名称、实验类型和二级实验室四列数据内容；
5. 统计每一门课程的实验课时数；
6. 统计每周开设所有实验课时数；
7. 统计每门课程的实验类型分布（crosstab）；
8. 统计每个班级的实验课课表；
9. 分析各二级实验室承担的实验课时量；
10. 分析各二级实验室能够支持的实验类型。

展示要求：

选用熟悉的应用开发语言（Python, Java, JavaScript），实现分析操作交互界面，将分析项目组织为“菜单”或“按钮”形式，调用后台完成的功能，

综合练习题（续）

应用开发要求：

1. 学习Python对象和方法的封装方式，将综合练习的数据和分析目标，封装为对象；
2. 选用熟悉的应用开发语言（Python, Java, JavaScript），实现分析操作交互界面，将分析项目组织为“菜单”或“按钮”形式；
3. 按钮功能，调用对象的方法完成，并将结果展示在当前界面中。