[一 Pandas 1](#_Toc759010900)

[1 Series 1](#_Toc860174493)

[1.1 Series的构造 1](#_Toc1385132989)

[1.1.1 创建一个空的Series对象： 1](#_Toc295196335)

[1.1.2 通过一个ndarray对象传递到data参数 1](#_Toc1512396497)

[1.1.3 从字典创建一个Series对象 2](#_Toc1426502558)

[1.1.4 通过标量值创建对象 3](#_Toc1176632045)

[1.2 Series基础操作 3](#_Toc223210623)

[1.2.1 Series对象元素的查看 3](#_Toc399831430)

[1.2.2 Series对象的修改 5](#_Toc1598584556)

[1.2.3 Series的数学运算 7](#_Toc541237354)

[2 DataFrame 12](#_Toc1085581637)

[2.1 DataFrame对象的构造 12](#_Toc1478485615)

[2.1.1 通过一个字典构建DataFrame对象 13](#_Toc2030875964)

[2.1.2 通过字典列表 13](#_Toc79819364)

[2.1.3 通过Series列表 14](#_Toc70642592)

[2.2 DataFrame的基本操作 14](#_Toc490385946)

[2.2.1 DataFrame的增删改查 14](#_Toc1983736462)

[① DataFrame的查找 14](#_Toc734669559)

[② 删除某一列属性 15](#_Toc126601959)

[③ 插入某一列属性 17](#_Toc642454214)

[④ 插入或修改一个新元组 18](#_Toc272415212)

[2.2.2 DataFrame的算术运算 20](#_Toc1548720608)

[3 Pandas进阶操作 21](#_Toc167194604)

[3.1 文件读写操作 21](#_Toc351805921)

[3.1.1 csv模文件处理 21](#_Toc1654239174)

[3.1.2 Excel文件处理 23](#_Toc2044795852)

[3.2 处理缺失数据 25](#_Toc748454319)

[3.2.1 isna()方法 25](#_Toc689623597)

[3.2.2 notna() 25](#_Toc733646191)

[3.2.3 notnull()方法 26](#_Toc383125979)

[3.2.4 dropna()方法 28](#_Toc1448634497)

[3.3 Pandas高阶操作 29](#_Toc1593820684)

[3.3.1 连接、合并 29](#_Toc1768258968)

[① concat()方法 29](#_Toc1743830832)

[② merga()合并函数 30](#_Toc958733533)

[③ join() 32](#_Toc1047277878)

[3.3.2 apply()函数 32](#_Toc772979229)

[3.3.3 Groupby分组 33](#_Toc1181944156)

[二 Sklearn学习 34](#_Toc1447109308)

1. Pandas

Pandas是为Python编程语言编写的一个用于数据操作和分析的开源软件库。在Pandas库出现之前，Python主要用于数据迁移和准备，因为Pandas高效，灵活，便捷的特点而，使得Python在数据分析与处理上变得得心应手，从而广泛应用于金融，统计，数据分析等学术和商业领域，可以说Pandas是当下Python大热的一个重要推手。

Pandas是基于Python Numpy库开发，并构建了一些更为高级的数据结构(Series,DataFrame,Panel)。因为在Pandas0.20.0以后的版本中，Panel遭到官方的摒弃，淘汰于历史尘埃中，故在本章Pandas学习中主要介绍Series和DataFrame。环境如下:操作系统，Ubuntu16.04，Python版本为3.6.5，IDE为Pycharm，基于IPython有着语法高亮，自动补全等特点，Python Shell使用了IPython。

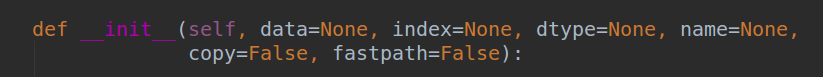
* 1. Series

Series，官方的定义如下：

**[Series](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html" \l "pandas.Series" \o "pandas.Series)** is a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.). The axis labels are collectively referred to as the **index**.

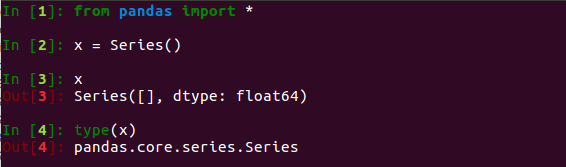
简单的说，Series类似于字典数据类型，他是一个一维的标记数组，每个元素可以储存任意的数据类型，包括整型，浮点型，字符型甚至一个Python对象。Series的标签称为索引，其与Series中的数据严格对照。索引的类型任意，可以为类似与列表的数字下标，也可以是一个字符串等等，当索引对应的数据为空时，数据为NAN。

* + 1. Series的构造

使用Pandas.Series()构造一个Series对象。查看Series构造函数，有如下参数：

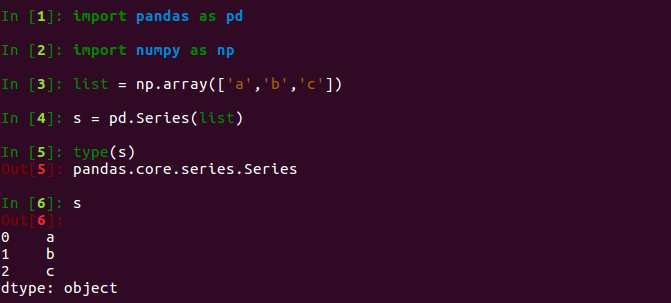
data即为传递过来的数据列表，其数据可为任意数据类型，index对应的是Series对象的索引列表，长度与数据列表，相同当index为空时(none)，默认为data列表的索引(0,1,2……)，dtype用于指定数据的数据类型，若dtype为none则根据实际情况推断真实类型，若类型不统一有冲突则为object类型。name参数允许我们给一个Series对象起名字，默认为none，copy实参为布尔类型，其用于选择是否拷贝data数据。fastpath也为布尔类型，其通常与copy结合使用。以下为具体代码：

* + - 1. 创建一个空的Series对象：



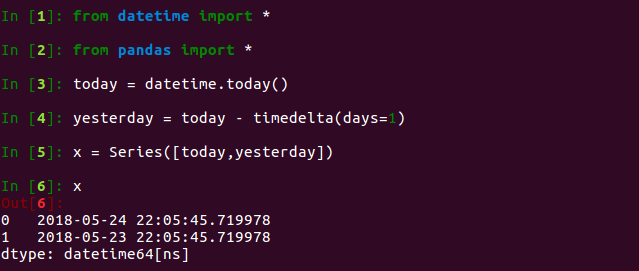
当Series对象为空数据为空时，默认数据类型为float64.

* + - 1. 通过一个ndarray对象传递到data参数



发现，当数据列表元素的数据类型为字符串型时，Series对应的数据类型为object。除此之外，Series支持的数据类型还有整数浮点数，布尔类型，datatime类型等等……

代码：

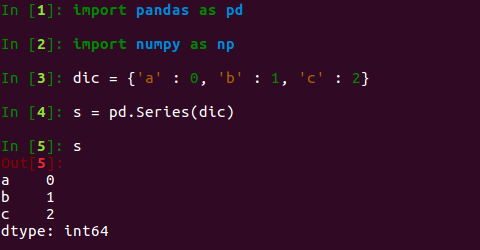


* + - 1. 从字典创建一个Series对象

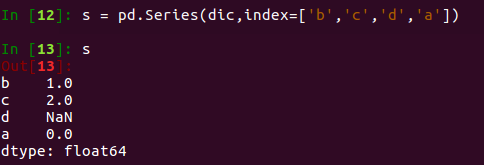
Series与dict有着很大的相似，都有着索引(键)，属性值（键值）。所以Series支持直接将一个字典对象作为data参数的一个实参传递。若构造中没有指定索引，者按照字典顺序取得对应的字典键及其键值分别传递到索引与数据中。若构造时指定了索引，索引中与标签对应的数据的值得到保留，空缺的键值以NaN（NaN并非一个数字，其在pandas中是数据缺失的一个特殊标志）填充，最后实际的顺序由索引列表决定。

代码：

未指定索引时：



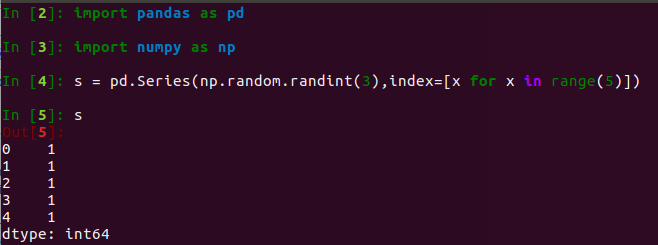
指定索引时：



* + - 1. 通过标量值创建对象

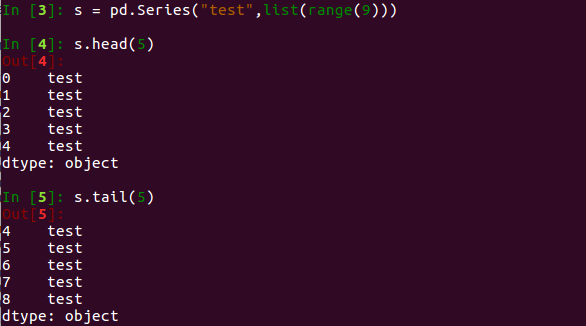
当data参数为一个数（也可为其他类型），可通过索引长度（此情况下索引的指定为必须）动态调整Series长度，此时索引对应的数据为同一的。

代码：



* + 1. Series基础操作
       1. Series对象元素的查看
          1. 通过head()和tail()查看Series对象的前n个或末尾n个元素。

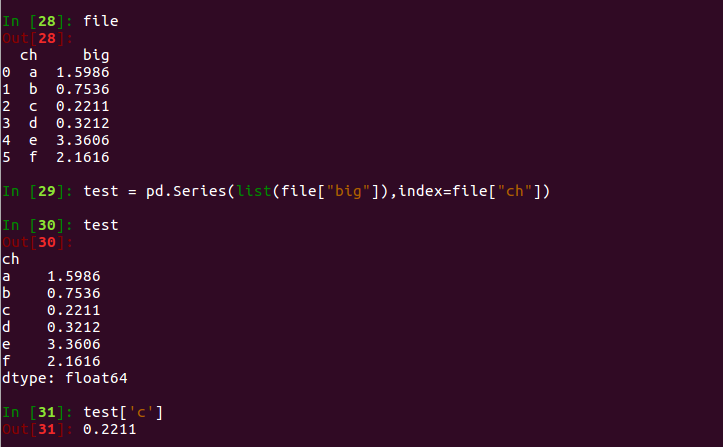
代码：



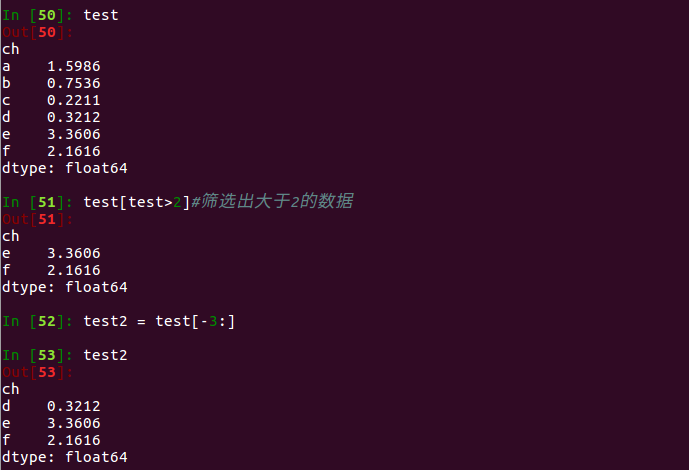
* + - * 1. 通过指定索引查看

显式得类似字典一般指定索引值，得到数据元素。

代码:

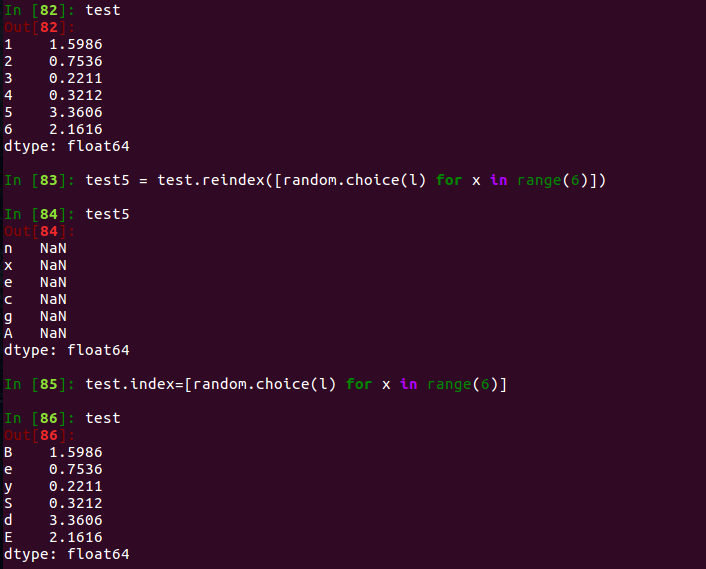


Series与npdarray类似，支持切片操作及筛选数据等操作。



* + - 1. Series对象的修改
         1. 重新修改索引

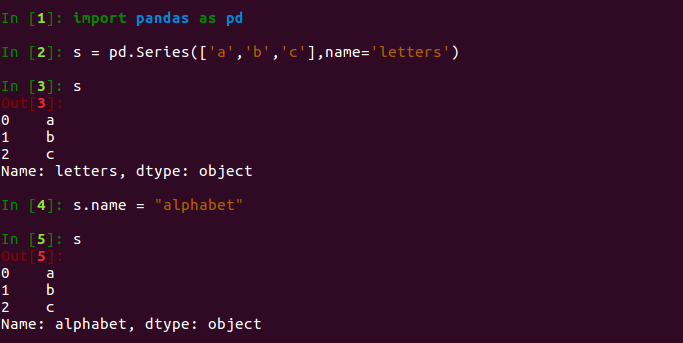
Series对象的index属性可以通过传递Series.index参数进行修改，索引的修改不会影响到原始数据。此外还可以使用reindex()方法指定新索引创建一个新的Series对象。这个方法会对新的Series对象的内容进行重写，旧索引的保留，新的以NaN补充。



* + - * 1. 设置name属性

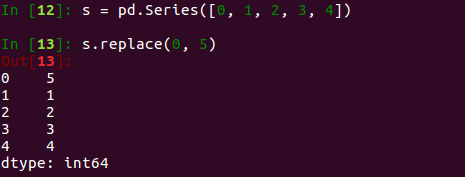
可通过Series直接构造时传递name参数，也可以通过Series.name()来修改。

代码：

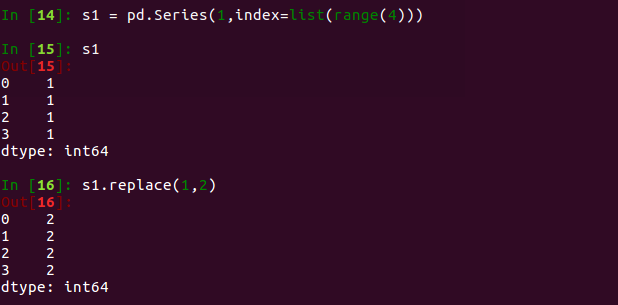


* + - * 1. 替换Series的数据

Series对象常通过replace()函数进行数据替换，与loc()方法通过索引进行操作不一样，replace()指定指定value进行修改。



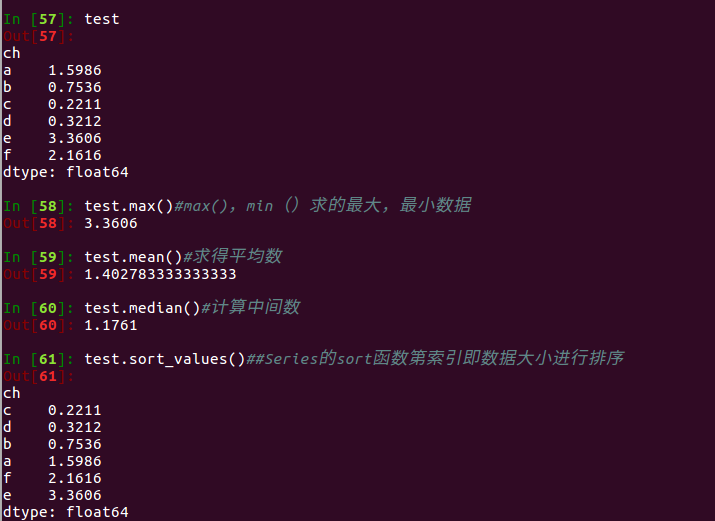
若数据中有相同的value，会一并替换。



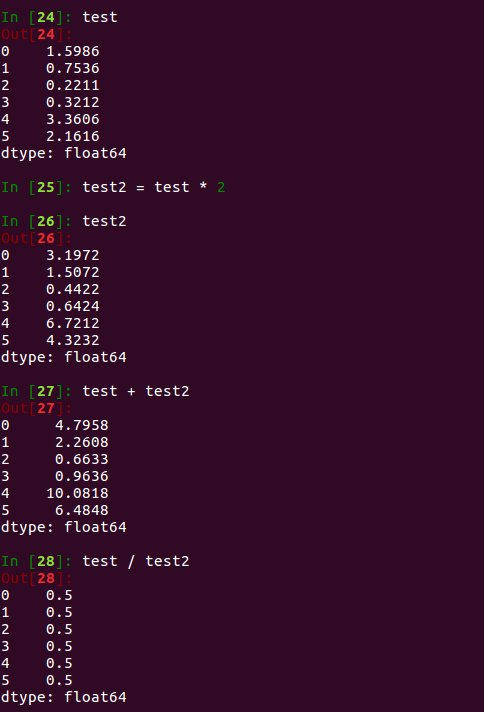
* + - 1. Series的数学运算

首先Series对象与列表类似，其内置的函数支持一些基础的数学操作如求最极值，平均值，中间值，大小排序等等 。

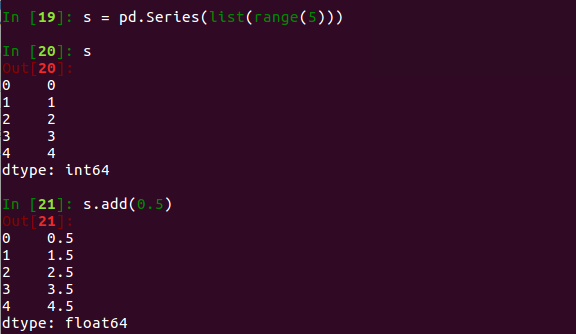
代码：



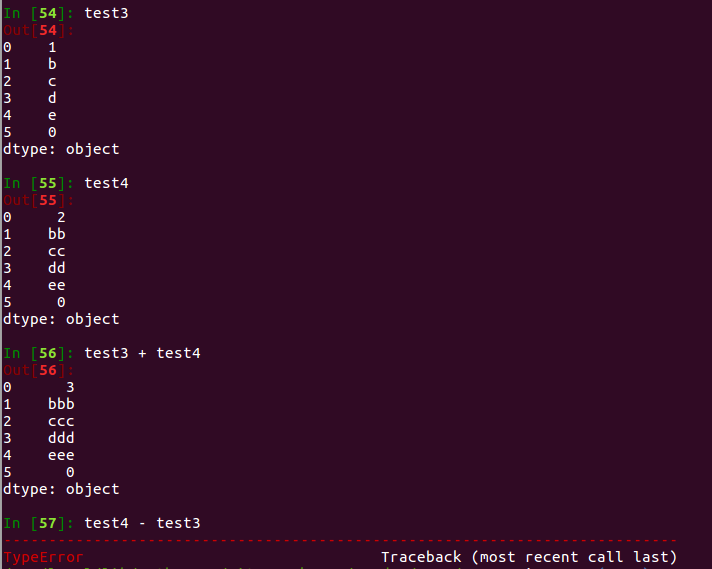
此外，Series对象之间还支持一些算术运算，包括加减乘除等。

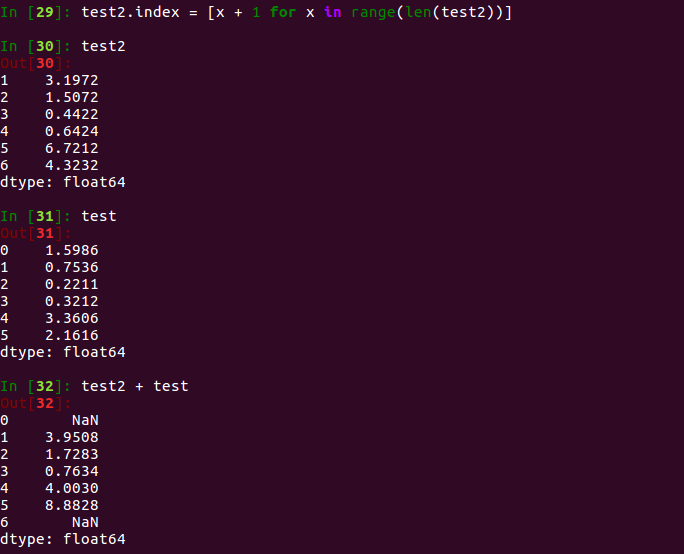


也可以使用add()函数对每个是实数型数据同时做加法。



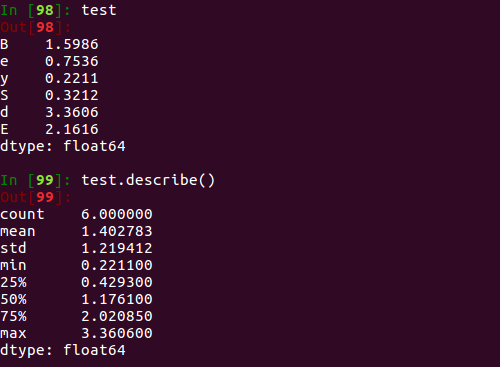
对于dtypeSeries对象索引不一致的数据做算术运算，会产生NaN做填充。（注意当Series对象之间包含有字符类型时，加法运算为对应的字符做连接操作，减法除法则不支持，乘法则只支持Series对象的数乘）。



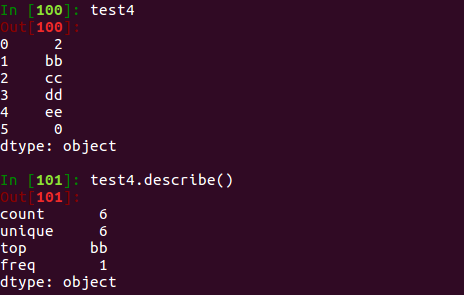


此外describle()函数会对Series对象进行汇总统计。

对于实数型数据(float64)，结果的索引将包括计数数目，平均值，标准偏差，最小值，最大值以及更低的50和更高的百分位数。默认情况下，较低的百分位是25，而较高的百分位是75.50百分位与中位数相同。



对于对象数据object（例如字符串或时间戳），结果的索引将包括count，unique，top和freq。top是最常见的价值。频率frep是最常见的频率。时间戳还包括第一个和最后一个项目。



* 1. DataFrame

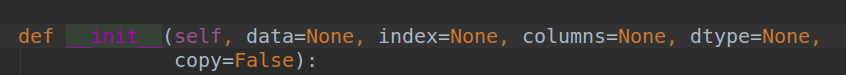
DataFrame的官方定义如下：

**DataFrame** is a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table, or a dict of Series objects. It is generally the most commonly used pandas object. Like Series, DataFrame accepts many different kinds of input.

相较于Series，DataFrame对象是一个一个表格型的二维数据结构，类似数据库中的表和一个excel表格。和Series相同，每个DataFrame对象都有一组有序的索引列（Index），每一行可以是不同的值类型(就像Series允许存储不同类型的元素一样），基本上可以DataFrame看成共享同一个索引Index的Series集合。

* + 1. DataFrame对象的构造

DataFrame 的构造方法与 Series 类似，只不过可以同时接受多条一维数据源，每一条都会成为单独的一列。DataFrame的构造函数如下：

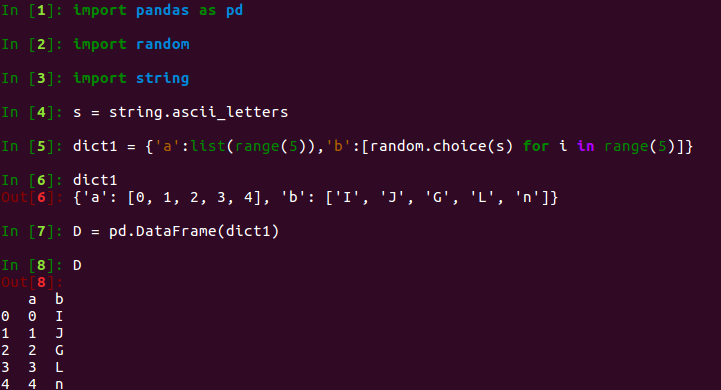


data实参类似一个字典，作为原始数据的传递参数，字典的键对应的不是index的名字，而是每一列的name属性。Index是一个列表，其组成元素为每一行DataFrame元素的行标签，与此对应，columns对应的是每一列的列标签，即name属性，当传递的data为字典且columns为空时，则将data的键列表传递给columns。若columns不为空，则columns重载(override）data字典的键列表。直接传递dtype参数的实例很少见，其究竟指代的每一行亦或是每一列的数据类型有待考却，但是任意一个DataFrame对象可直接访问其DataFrame.dtypes属性，返回一个Series对象，其中元素对应了每一二个列属性(column)的数据类型。copy参数与Series构造中的类似，当copy为真时，会在在构造函数创建了一个DataFrame对象的同时会调用该对象的数据(data)及其索引副本创建一个副本，副本的改动不影响原始对象。

DataFrame创建很灵活，其中传递的data可以是一个ndarrays或列表的字典，也可以是一个结构化的数组，甚至是一个字典列表。

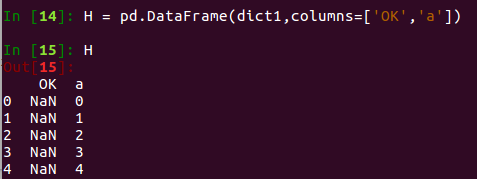
* + - 1. 通过一个字典构建DataFrame对象

代码：



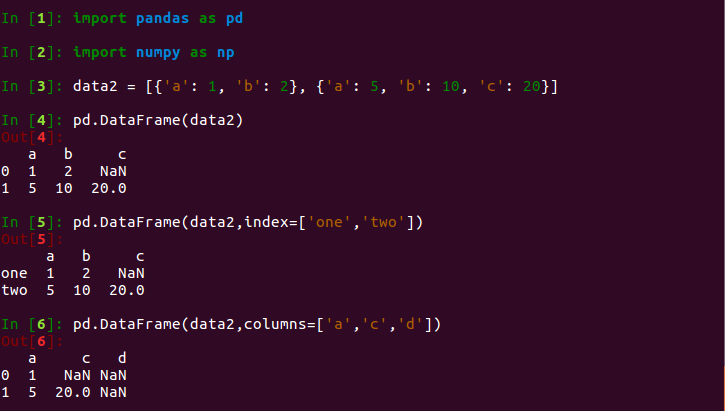
显然，在构造中，函数自动将字典的键的列表出阿迪给columns作为列属性，当columns不为None时，又会对字典传递的键的列表进行截取，即已有列属性的内容予以保留，不存在的则以NaN覆盖，效果如下：

代码：



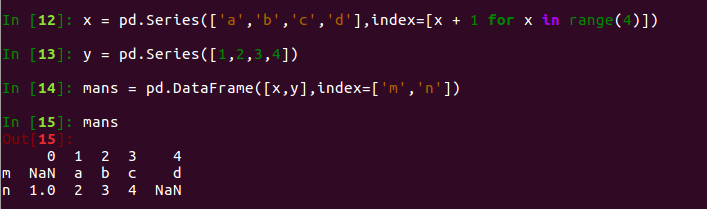
* + - 1. 通过字典列表

代码：



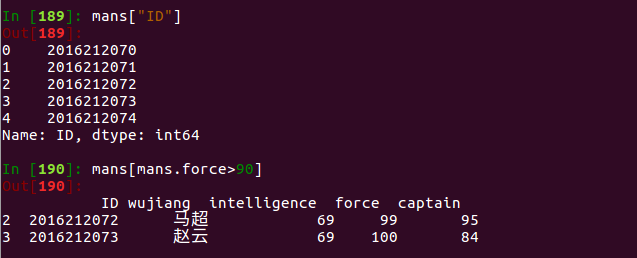
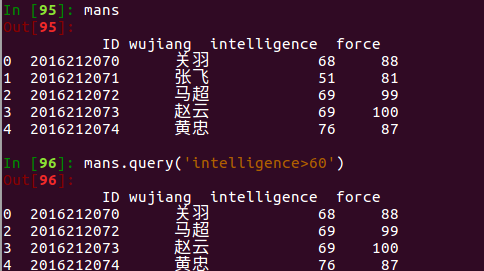
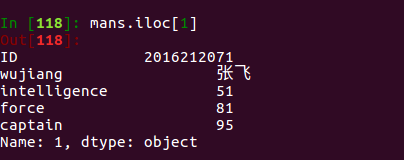
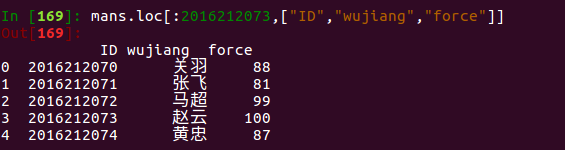
* + - 1. 通过Series列表

代码：

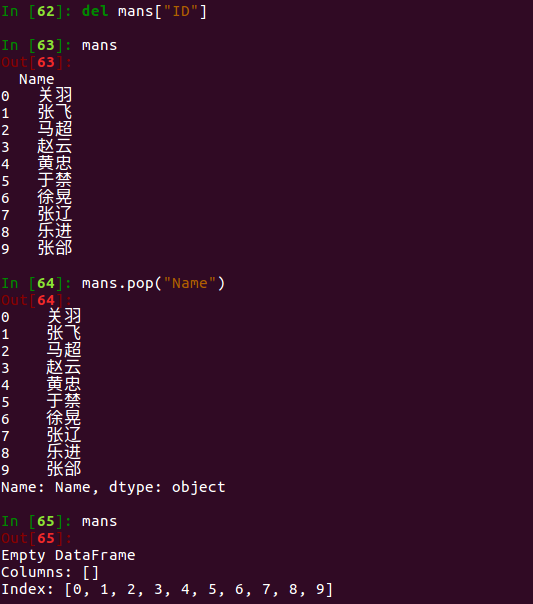
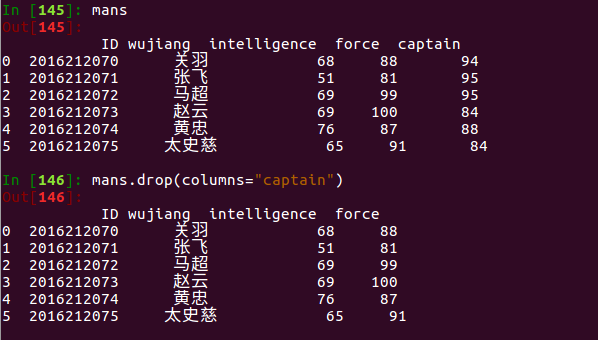


注意：当两个Series对象的索引不完全相同时，在最终的DataFrame对象中取两者的并集，空缺部分补NaN。

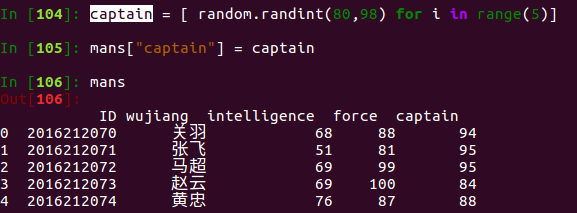
* + 1. DataFrame的基本操作
       1. DataFrame的增删改查
          1. DataFrame的查找

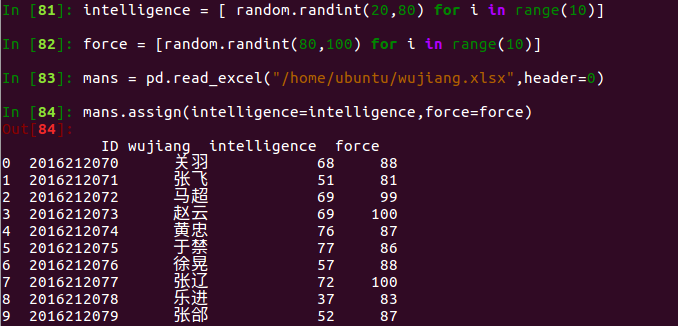
通过指定columns名称找出某一列或某几列  
  
 若想找出某一行元组，也可使用query()函数，其接受一个字符串，字符串的内容即为判别式，例如想找出智力(intelligence>60)的武将  
  
 另外还可以使用iloc和loc属性指定索引来查找具体某一编号的元组  
  
loc第一个逗号前指定索引，后一个列表指定所要查询的columns，只会切片操作。  


* + - * 1. 删除某一列属性

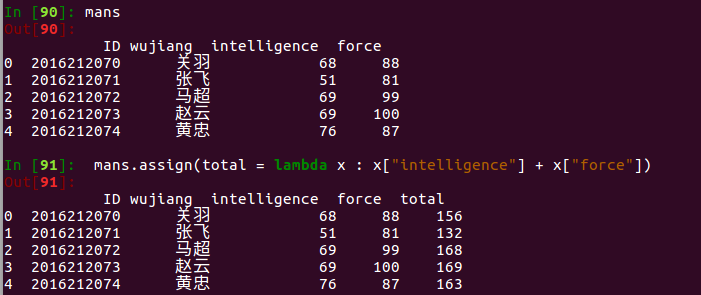
与列表的删除类似，DataFrame对象同样可以使用del和pop函数进行删除，只不过列表删除的对象为列表的一个基本元素，而DataFrame删除对象为某一列column  
  
 使用drop()函数可以便捷删除行或列  


* + - * 1. 插入某一列属性

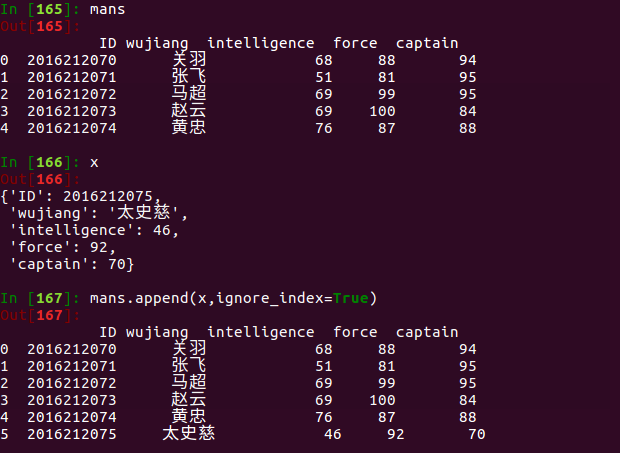
DataFrame对象可像字典一样插入新的键值对操作一般增加新的列，只不过插入的column为一个列表。  
  
 另外，DataFrame对象也使用assign()函数进行column的增加。assign()函数内传递的是一个\*\*型，可接受任意数量的关键字\*\*kwargs。\*kwargs分解为*keyword和value pairs*键。*Keyword*为新增属性列的名字，*value pairs*为对应的属性列表。assign()可接受任意数目的*keyword和value pairs*对。

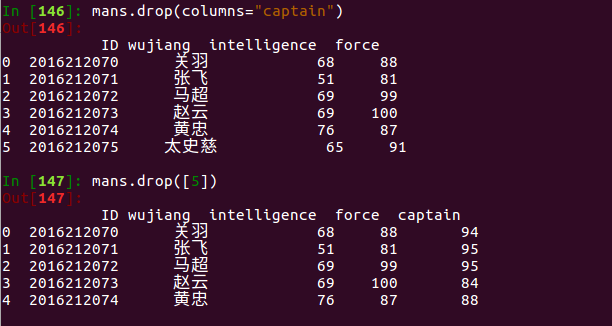


assign()内支持匿名函数lambda：

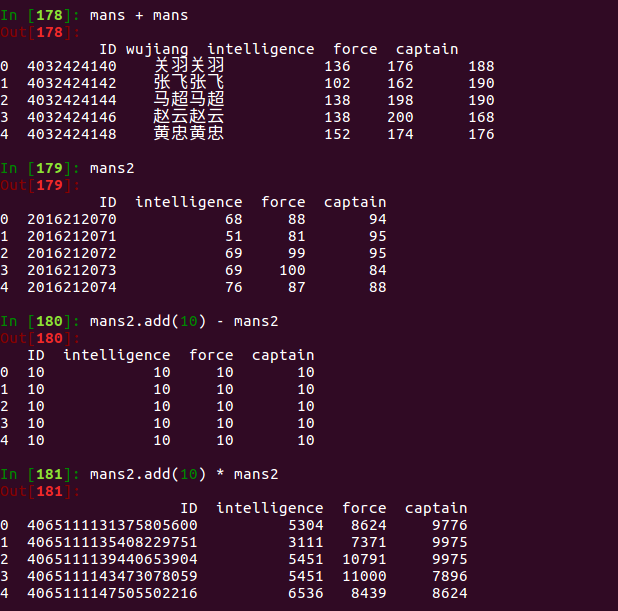
  
注意：assign()函数的操作结果返回的是我一个新对象，即旧对象的一个copy，实际要保存的话最后另做切片赋值或使用head()赋值等等。

* + - * 1. 插入或修改一个新元组

欲插入或修改某一行，可指定对象的索引loc[]来进行操作。  
  
 此外，也可通过append()函数添加一个字典参数。  


删除某一行元组则使用到drop()函数。  


* + - 1. DataFrame的算术运算

与Series对象类似，DataFrame内对象根据其类型支持不同的算术操作。字符型支持数乘，加法，实数型支持加减乘除运算。  


* 1. Pandas进阶操作
     1. 文件读写操作
        1. csv模文件处理

Csv(Comma-Separated Values)文件，也称逗号分隔值文件，是一个通常作为表格存储的纯文本文件。其在商业以及科学领域广泛应用。Csv文件的特点就是数据之间的分割符是逗号。若要将分割符为其他字符的文件转为csv文件，只需要一行简单的sed命令。

`sed -i ‘s/原有字符/目标字符/’ 目标文件`

* + - * 1. 读取csv文件

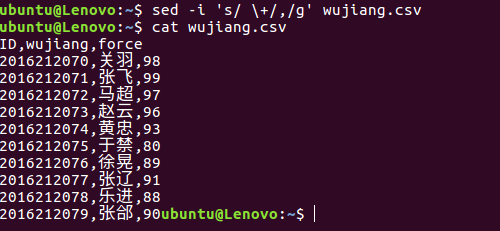
Pandas库在处理csv文件上得心应手。pandas.read\_csv其将一个csv文件读取为一个DataFrame对象。pandas.read\_csv()函数有一下几个重要参数：

* filepath\_or\_buffer : str，这个参数对应目标文件的path路径，在引用时最好使用直接路径。
* sep：str，指定分隔符，默认为逗号
* header：int or list of ints。用作列名columns的行号，当csv文件无指定columns时，传递header=0.
* index\_col : int or sequence or False，用作行索引的列编号或者列名，如果给定一个序列则有多个行索引。如果文件不规则，行尾有分隔符，则可以设定index\_col=False 来是的pandas不适用第一列作为行索引。
* norrow：int, default None，需要读取的行数。

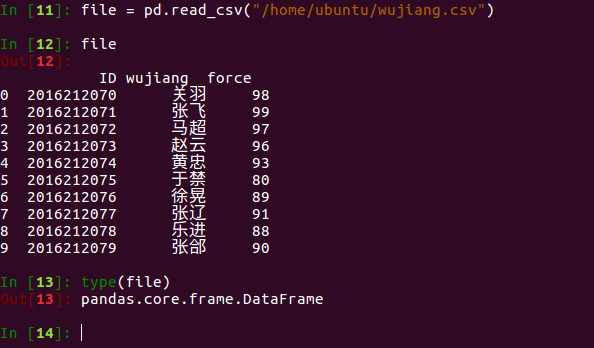
例如读取一个这样的文件



转换为csv文件



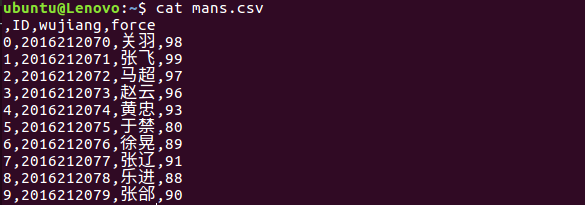
将使用read\_csv模块读取文件



* + - * 1. 写入csv文件

to\_csv()函数负责将数据写出到文本格式。其参数有：path\_or\_buf，seq(设置 seq(分隔符))、index 和 header(禁止写出行和列名)、cols(需要写出的列,以及列的顺序)，encodin(编码方式)等等。





* + - 1. Excel文件处理

Excel文件时Windows上最流行的数据处理软件，其xlsx文档广泛应用于各个领域。

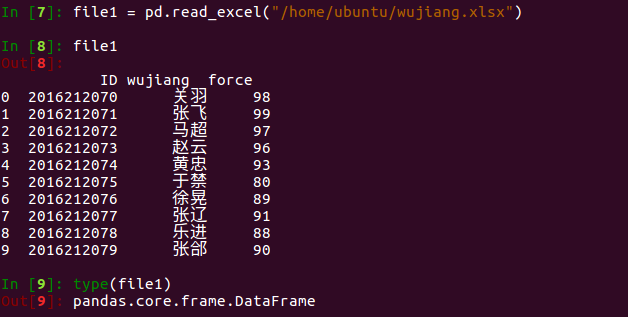
* + - * 1. 读取excel文件

Pandas.read\_excel()模块是用于读取excel文件的一种更重要手段，其中主要参数如下:

* io :string,文件的path路径
* sheet\_name : string, int, mixed list of strings/ints, or None, default 0。使用的第几个sheet表，返回多表使用sheetname=[0,1],若sheetname=None是返回全表 注意：int/string 返回的是DataFrame，而none和list返回的是dict of DataFrame。
* header：int, list of ints, default 0。指定列名行所在，header=None表示，数据不含列名。
* index\_col： int, list of ints，default None。行索引的列编号或者列名。
* names : array-like, default None。指定每列的名称。

例如读取一个这样的excel文件





显然读取的结果为一个DataFrame对象

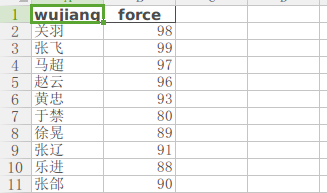
* + - * 1. 写出excel文件

存储函数为pd.DataFrame.to\_excel()，只有DataFrame写入excel,其具体参数如下：

* excel\_writer：string or ExcelWriter object File path or existing ExcelWriter 文件的直接路径
* sheet\_name : string, default ‘Sheet1’。使用的sheet表数。
* columns : sequence, 选择输出的列，默认为所有。
* header : boolean or list of string, default True。columns的名字，若无则天None。
* encoding: string, default None。编码方式。
* index : boolean, default True，行索引的名称，False表示不显示行索引。

代码：



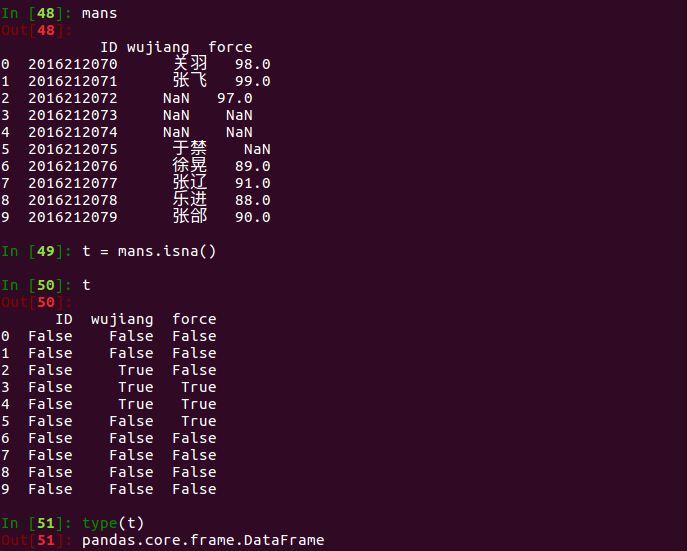


* + 1. 处理缺失数据

pandas 中 NA 的主要表现为 np.nan，表示值为空NaN或缺失missing。另外 Python 内建的 None 也会被当做 NA 处理。处理 NA 的方法有四种：dropna , fillna , isnull , notnull 。

* + - 1. isna()方法

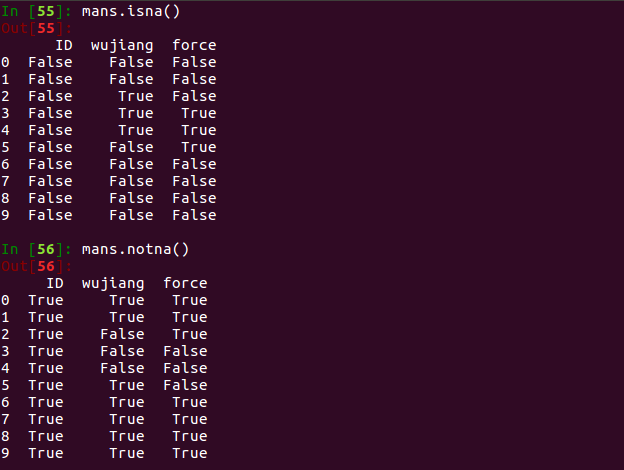
isna()方法用于检测缺失值，他通过检测DataFrame中每个元素的bool值的掩码，指示元素是否不是NaN，所以其返回结果为一个所有数值为布尔值的DataFrame类型对象。



True表示对应数值为NaN。

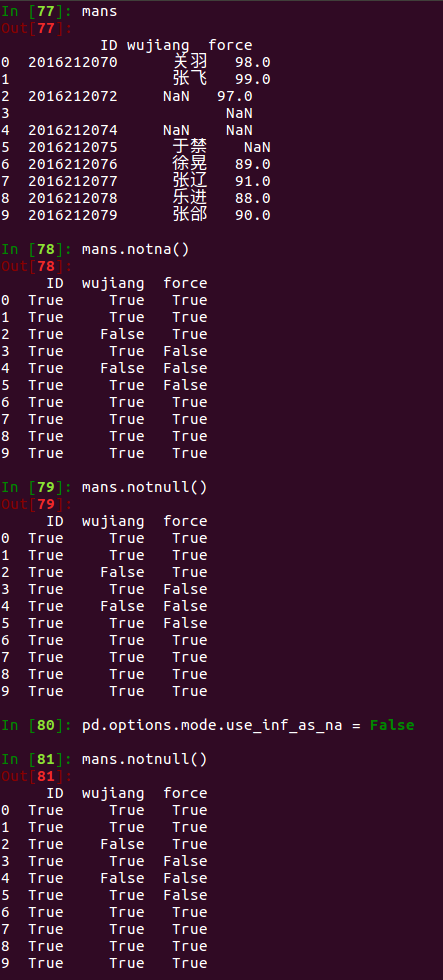
* + - 1. notna()

与isna()相反，notna()用于检测已存在的真实值(不为NaN)。

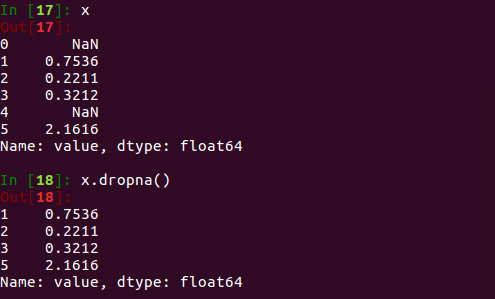


* + - 1. notnull()方法

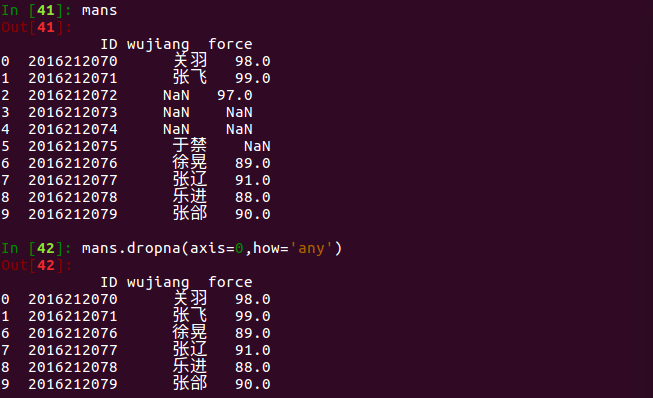
notnull()方法与notna()有所异同，相同之处在于都是用于检测和处理缺失值，并且返回一个同样规格，数据类型全为布尔型的DataFrame对象。但是notnull()对缺失值的判定更加宽广，不仅可以选择性地检测NaN(通过设置pandas.options.mode.use\_inf\_as\_na = True)，也可以检测空字符和空的numpy类型。

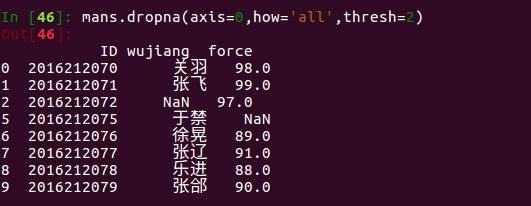


* + - 1. dropna()方法
         1. 对于Series对象，dropna返回仅包含非空数据和索引值的Series对象。



* + - * 1. 对于DataFrame对象，一个NaN的出现意味着至少丢掉一行或者一列。为此我们可以对dropna()传递额外的参数对缺失项进行特殊处理。
* axis : {0 or ‘index’, 1 or ‘columns’}, default 0。axis参数接受一个集合，集合可选0或1或‘index’或‘columns’，0或’index‘表示drop行，1或‘columns’表示drop列。
* how : {‘any’, ‘all’}, default ‘any’。How集合包含‘any’时，会将包含一个NaN项的整行都drop删除。‘all’则表示只有所有值为NaN时才drop所在行列。
* thresh: int, optional。thresh为可选项，thresh的数值表示当非NaN项数目大于thresh数值时保留整行。



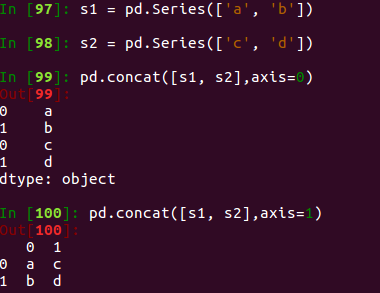


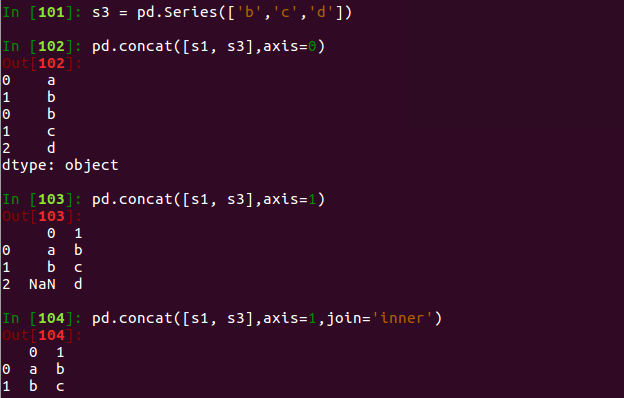
* + 1. Pandas高阶操作
       1. 连接、合并

Pandas提供了多种操作将Series，DataFrame对象通过索引以及关系代数逻辑组合在一起。

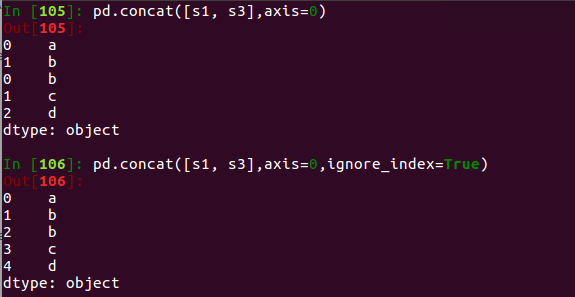
* + - * 1. concat()方法

concat()函数严格按照轴对DataFrame对象进行串联或并联操作。串联，即concat()中axis参数为0或’index’，即多个Series或DataFrame对象按照行索引连接，类似append()函数从一个对象中添加其他对象。并联，即concat()中参数为1或’columns’，具体操作类似与数据库中的多表连接，多个DataFrame对象按照列并在一起，根据join参数又分为内联和外联两种，当join=‘inner’时，两个DataFrame对象连接得到共有columns组成的新DataFrame对象，当join=‘outer’(即默认情况下)，会将生成两个DataFrame对象并联的columns下的新DataFrame对象，具体操作见下：





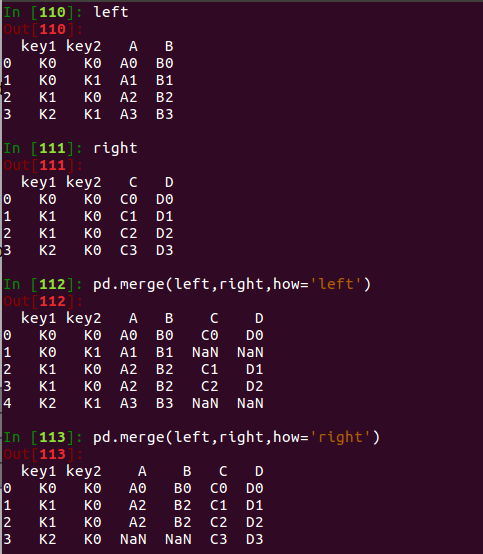
此外，还有参数ignore\_index(默认为False)，当其为True时，表示连接后重新编排行索引，这主要是考虑到axis=0时，连接中有些索引会重复(如上图），造成不必要的影响。

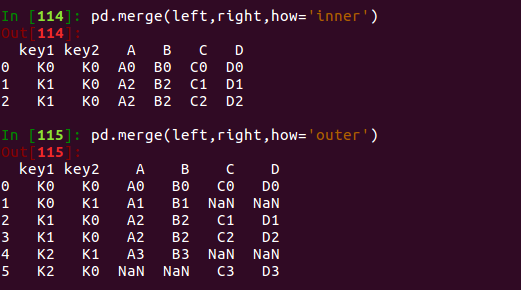


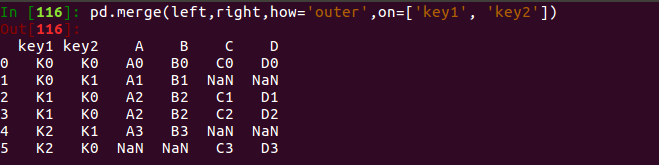
* + - * 1. merga()合并函数

merge()函数作为连接操作类似数据库语言外连接操作中的join。其中主要参数如下：

* left : DataFrame 左边的DataFrame对象
* right : DataFrame 右边的DataFrame对象
* how : {‘left’, ‘right’, ‘outer’, ‘inner’}, default ‘inner’。how参数表示数据融合的方式。outer和inner的区别和上述concat()中join参数类似，left和数据库中的左链接相似，即保留左边DataFrame对象，以此为基础合并新的DataFrame对象。right则与类似SQL中的右外连接，与left正好相反。
* on:label or list。用来对齐的那一列的名字，用到这个参数的时候一定要保证左表和右表用来对齐的那一列都有相同的列名。如果on=None并且不合并索引，则默认为连个DataFrame对象columns的交集。

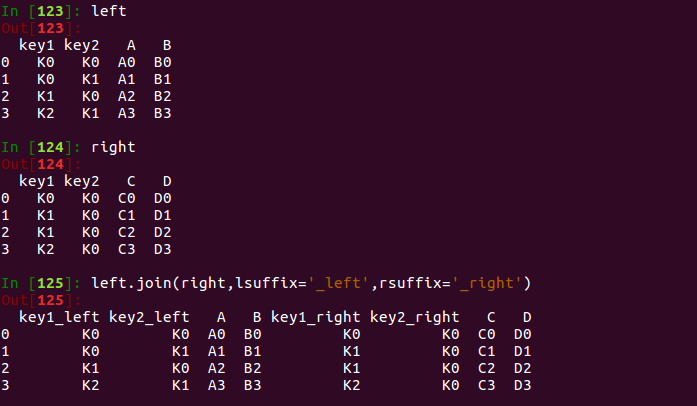






* + - * 1. join()

join()通过索引与其他DataFrame对象连接。其操作较为简答，故不过多说明。



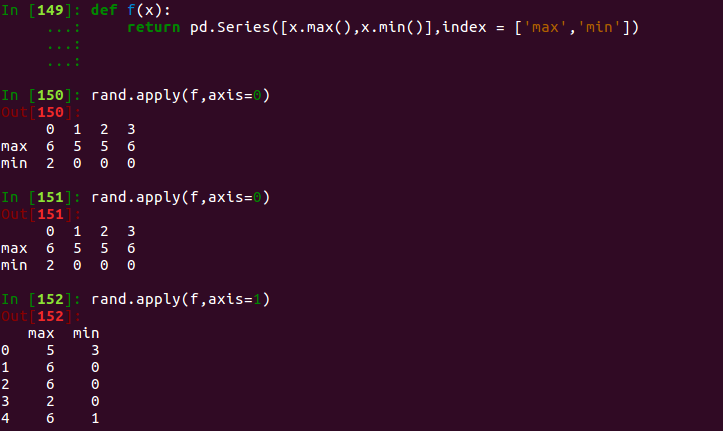
其中，lsuffix参数表示对左边的DataFrame对象中的重复列名通过添加后缀的方式重命名，rsuffix则对右边的DataFrame对象进行同样的操作。

* + - 1. apply()函数

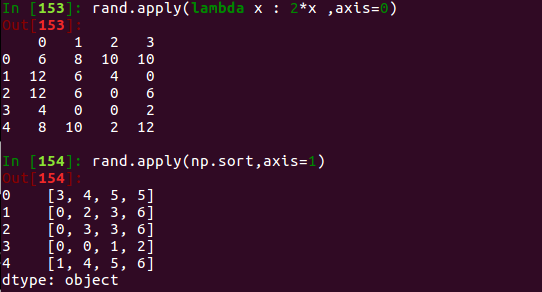
apply()是DataFrame中的一个模块，其根据axis参数的值选择性地对DataFrame对象中的索引(index)或列(columns)进行函数func应用。apply()中主要参数如下

* func : function 。对行或列进行操作的函数
* axis：{0 or ‘index’, 1 or ‘columns’}, default 0 。0 or ‘index’表示对以索引为轴进行操作，1 or ‘columns’表示以列为轴进行函数操作。
* raw : bool, default False。当row为False时，将DataFrame对象按行或列拆解为Series作为参数传递到func中。当row为True时，则以ndarray为参数传递给func。

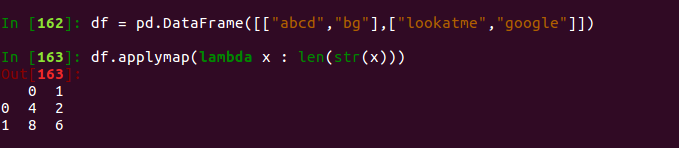
代码：



此外，func函数亦可为匿名函数

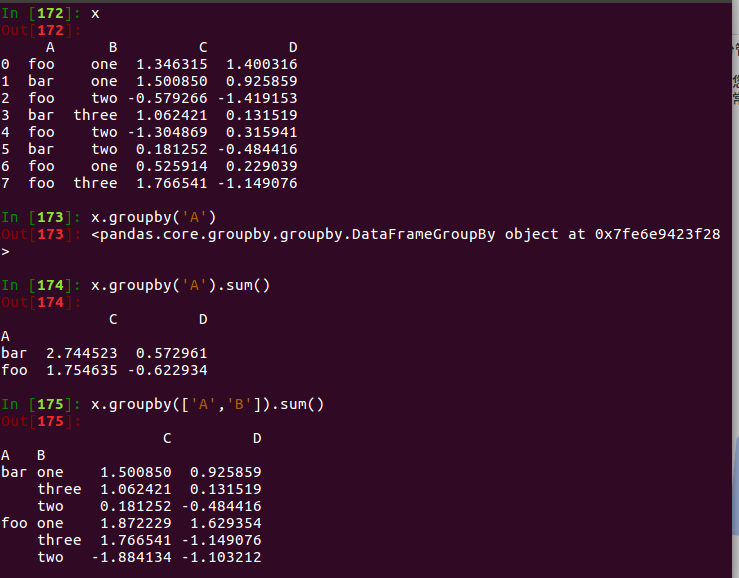


此外，类似的还有applymap()，其中的函数func的参数为DataFrame对象中的每一个元素，其返回对象为与DataFrame同规模的一个新的DataFrame对象。该新对象中的每一个元素为原始对象对应的函数对函数func的应用结果。



* + - 1. Groupby分组

Pandas中groupby功能使得我们能轻松的对数据集进行切片，切块等操作。groupby()内的一个重要参数。by参数形式多样，其决定了分组的根据。如果by为一个函数，这个函数将对对象索引上的每一个值调用。当by为一个label标签时，则按照这个标签对数据进行划分。



Groupby操作类似于SQL语言中的分组，其通过by这个分组依据进行划分得到一个groupby对象，最后应用sum()或mean()过滤数据得到一个分组DataFrame对象。

1. Sklearn学习

由于前段时间参加互联网+的一个比赛加上临近期末各种报告繁多，Sklearn模块这块没能完成。由此简要谈谈对这方面的理解，只是稍微看了点资料。Sklearn是python中针对机器学习编写的一个第三方模块，其对于常见的一些机器学习方法进行了封装。所以在实际分析中可以直接调用该模块中的方法进行各种分类，回归以及聚类任务，而不需要用到我们实验二中我们编写的那些繁冗的代码。

在对某个数据集进行分析时，针对庞大的数据，我们首先需要从数据集中提取出有效的数据，一方面是对数据集中一些缺省的数据如何处理，如果直接删除未免太浪费，当然也可以像上面pandas学习中针对missing data的方式根据缺省程度决定是否保留，再一个有些数据集的属性特征太多，如何提取选出对分类影响最大的那部分特征，又要采取什么何时度量单位，算法进行比较从而选出不冗余的属性。做完这些，我们算是成功提炼了一个有效的训练集，接下来按理要进行训练模型，但是考虑到数据集还是庞大，构成的特征矩阵过大，导致计算时间过长。因此在这个特征矩阵熵我们还需要采取一些数据手段进行优化，其中一个思路就是将矩阵的维度递归式的降解，具体的方法我想sklearn或者其他数学库中有，我们应该比较选出最合适的方法。然后是训练模型算法的选择了，常有的算法主要分聚类，分类，回归几大类，首先根据训练的任务，到底是监督学习还是无监督学习进行一个初步选择，然后在各大算法之间进行比较，看那个模型的准确率高，泛化性能好，之后对于学习器我们还需要采用手段取强化，提炼。就像集成学习中AdaBoost 算法那样，通过对已生成的学习器包括数据集进行动态调整，一次次地递归，不断纠错，加强模型的泛化能力，然后或是选择最终的生成器，亦或是生成的各个生成器加权结合，得到强化的模型。这些便是我对于sklearn那块要解决的问题的一点粗浅看法，我相信还有更优秀的算法以及优化方式，而这些也是我将来要学习的目标。