**Pandas 知识**

**一、数据结构**

**1、Series：一维度序列**

* 包含数组所有特性，同时含有索引。
  + 通过 print('title:\n', series) 来给 series 加 title
  + 没有具体索引内容就会自动从 0 开始添加索引
* 通过 series.value 和 series.index 分别提取值和索引
  + 通过 matplotlib 可画 series 图表：plt.plot(series)
* 创建 Series：pd.Series(data, index, dtype, copy)
  + data 为 array, list 或字典。index 为 list。
  + 可以 range 来为 data 和 index 赋值。
  + index 在 Series 创建之后，可通过 series.index = ['a', 'b', ......] 来修改全部索引。不能通过 series.index[n] = 其它值来修改部分索引。
  + 如括号里只有一个数，则值都会显示此数。
  + 如括号里只有 range，则索引和值都会从 0 到最后一个数排列。
* 提取元素：
  + series['索引1'] 输出值1。
  + series[['索引1', '索引2']] 输出含「索引1」、「索引2」、「值1」、「值2 」的 series。
  + series['索引1' : '索引2']、series['索引1':]、series[:'索引2']
  + 提取时要直接写数字（代表索引），如 series[[1, 2]]。不可用字符表达数字，如 series[['1', '2']]。
  + series[series == 1] 输出所有值等于 1 的索引和值。series[series == 1].index[0] 输出第一个值为 1 的索引。
* 运算：Series 相加时，根据索引来匹配值并相加。所有的索引都会在结果中出现，如没有匹配的索引，则显示为 NaN。

**2、DataFrame：二维表单**

* 既有行索引，也有列索引
  + 通过 print('title:\n', series) 来给 series 加 title
* 创建 DataFrame：
  + pd.DataFrame(array, index, columns)：data 为 array, list 或字典。index、column 为 list。
    - 默认字典索引为列索引，字典中值须为 list。如要同时赋值给行列索引，要字典里套字典，则字典索引为行索引，字典值为字典。
  + 基于现有列表创建：df[['列索引1', '列索引2', ......]]
* 提取元素：
  + df[df[n 或 'name'] == a]，n 或 'name' 为列索引名，和 R 中表达相似。
  + df['name'][0 : 20]：提取 name 列中头部数据
  + df['feature\_names']：提取所有列名
  + df.values：提取所有的值
* 查看
  + df.head()
  + df.columns
  + df['列'].value\_counts()：可对行列或 Series 进行不同值所占数量的统计。
* 运算：DataFrame 可相加。

**二、数据处理**

**1、重建索引**

* Series
  + index 在 Series 创建之后，可通过 series.index = ['a', 'b', ......] 来修改全部索引。
  + 也可通过 series = series.reindex(index = ['a', 'b', ......])。
    - index 为默认值，可不写「index =」。
  + 不能通过 series.index[n] = 其它值来修改部分索引。
* DataFrame
  + df = df.reindex(index = ['a', 'b', ......], columns = ['a', 'b', ......])
  + 如只有一个参数，则自动修改行索引

**2、删除和添加**

* 删除
  + Series：series.drop('索引1') 删除「索引1」所在行。
  + DataFrame
    - df.drop('行索引1')
    - df.drop('列索引1', axis =1 或 axis = 'column' )
    - 参数 inplace = True 为默认，表示删除后直接输出已删除的列表
  + .dropna() 函数可通用（inplace 参数也可设定）
* 添加：df.shift(n) 添加 n 行 NaN 值（可填充为 0）。

**3、特定值的选取和过滤（包含末端值）**

* 值选取见课件中图。



* 值填充
  + 替换值
    - 通过 df > 2 来判断各元素是否大于 2。
    - 通过 df[df == 1]] = 2 来把所有等于 1 的元素的值换成 2。如赋的值是 None，则会被替换成 NaN。
  + 计算值
    - 不同的 Series 在索引相同的情况下，值可以相互进行运算。如 series\_1/series\_2。
    - 如有缺失值需要填补，可用如下方法
      * 添加参数 fill\_value = 0（必须用运算函数，而非运算符）。
      * .fillna(n)：n 为要填充的值。
    - 具体运算函数见课件。

**4、Series 和 DataFrame 之间的运算**

* 默认情况下会将 Series 的索引和 DataFrame 的对应的列索引匹配，然后将 Series 的每行值和 DataFrame 每列里的每一个值进行运算。
  + 如果有列找不到对应的 Series 索引，则仅针对有相同名称的 Series 索引和列索引进行上述运算。其余值会以 NaN 填充。
* 如需按行索引对应，则一定要用运算函数（参数 axis = 'index'）。按行索引对应后，DataFrame 中的每列中的所有值都会其和所在行的 Series 值进行相应运算。不然输出所有值为 NaN 的列表。

**5、排序和排名**

* 排序
  + 按 index 排序：df.sort\_index()
    - 默认按 index 从大到小排（axis = 0 或 ascending = True），要调整则改成 df.sort\_index(axis = 1 或 ascending = False)
  + 按 value 排序：df.sort\_values()
    - 要按某一列里值的大小排序，则用参数 by = '列名'。
    - df = df.sort\_values(by = ['a', 'b'], ascending = [True, False]) 可对多列进行排序，但 df 结构被破坏。只有同为 True 或 False 时才可以。
    - 任何缺失默认值 NaN 都会排到末尾行。
* 排名
  + series.rank()：将所有索引按值的大小来排名，并打分。默认按升序排名。
    - 参数 method = 'first' 表示不可有相同分数。分数相同时，先出现的小一分。（此参数其它值见课件图片）
    - 设置参数 ascending = False 实现降序排名

**6、数据表合并**

* merge

df1 = DataFrame({'name':['ZhangFei', 'GuanYu', 'a', 'b', 'c'], 'data1':range(1,6)})

df2 = DataFrame({'name':['ZhangFei', 'GuanYu', 'A', 'B', 'C'], 'data2':range(1,6)})

df3 = pd.merge(df1, df2, on='name') # 基于 name 这列进行连接

* 内、外、左、右连接
  + inner 内连接：df3 = pd.merge(df1, df2, on = '列名', how='inner')
  + left 左连接：df3 = pd.merge(df1, df2, on = '列名', how='left')
  + right 右连接：df3 = pd.merge(df1, df2, on = '列名', how='right')
  + outer 外连接：df3 = pd.merge(df1, df2, on = '列名', how='outer')
    - 如两个表的公共列值相同但列名不同，可将「on」变成 left\_on = ['表一中列名'], right\_on = ['表二中列名']。其中列名如果没有可用数字代替。多个列名可用逗号隔开。

**7、Pandas 中的 groupby 使用：作用是进行数据的分组以及分组后地组内运算。**

* 应用

import numpy as np

import pandas as pd

data = pd.read\_csv('heros2.csv', encoding='gbk') # 文件中有中文，所以采用 gbk 编码读取

result = data.groupby('role').agg([np.sum, np.mean]) # 也可用 group('role').sum().mean()

print(result)

* groupby 默认自动合并掉一列中的重复项
* 参数 as\_index 默认等于 True，公共列会被作为索引。设置成 False 则不会。

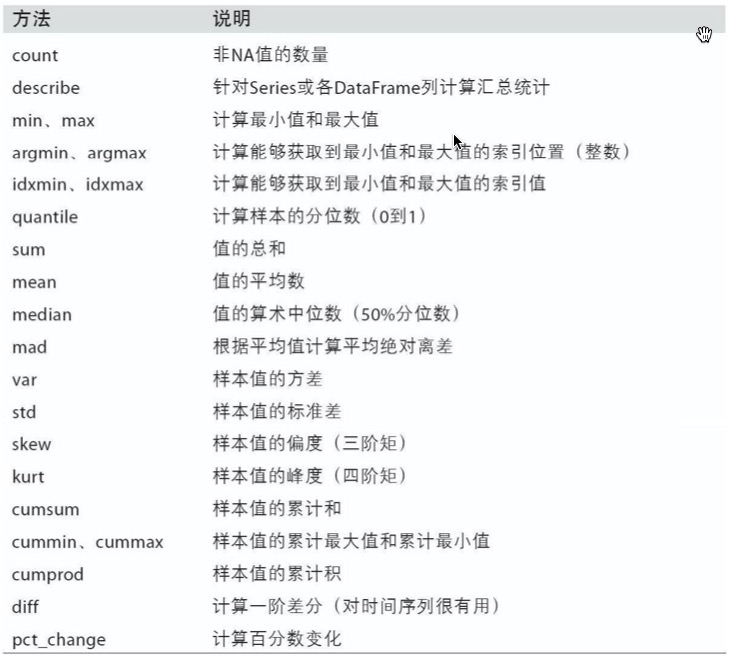
**8、其它数据处理**

* 重命名列名 columns：df2.rename(columns={'Chinese': '语文', 'English': 'Yingyu'}, inplace = True)
* 去掉重复的值：df = df.drop\_duplicates()
* 更改数据格式
  + df2['Chinese'].astype('str')
  + df2['Chinese'].astype(np.int64)
* 去掉数据间的空格
  + df2['Chinese']=df2['Chinese'].map(str.strip) # 删除左右两边空格
* 大小写转换
  + df2.columns = df2.columns.str.upper() # 全部大写
  + df2.columns = df2.columns.str.lower() # 全部小写
  + df2.columns = df2.columns.str.title() # 首字母大写
* 查找空值：df.isnull()
* DataFrame.diff()函数：用来将数据进行某种移动之后与原数据进行比较得出的差异数据。
* df.T：行列对换（行列名也相应对换）
* DataFrame.shift(period, freq, axis, fill\_value)：可以把数据移动指定的位数。
  + periods = -1 往上移动或往左移动；periods =1 往下移动或往右移动。移动后原有行或列默认由 NaN 填充。
  + freq： DateOffset、timedelta、time rule string，可选参数，默认值为 None，只适用于时间序列。如果这个参数存在，那么会按照参数值移动时间索引，而数据值没有发生变化。



**三、函数**

**1、基本统计函数 df.f()**



**2、用 apply 函数进行数据清洗：apply 函数是 Pandas 中自由度非常高的函数，使用频率高。**

* 比如对 name 列的数值都进行大写转化

df['name'] = df['name'].apply(str.upper)

* 也可以定义个函数，在 apply 中进行使用

def double\_df(x):

      return 2\*x

df1[u'语文'] = df1[u'语文'].apply(double\_df)

* apply 函数的结果会带上每个值的行名或列名。
  + 如要构建新的一列，想要将列名和值一同放到新列单元格，需要 apply(lambda x: ' '.join(str(x)), axis = 1)

\* apply 和 map 的区别是什么？

* + apply 用在 dataframe 上，用于对 row 或者 column 进行计算（默认按列）。
  + applymap 用于 dataframe 上，是针对每个元素的操作
  + map 是 python 自带的，用于 series 上，是针对每个元素的操作

**3、相关性分析：DataFrame.corr(method = 'pearson', min\_periods = 1)**

* method 参数
  + pearson，衡量两个数据集合是否在一条线上面，针对线性数据的相关系数计算，对于非线性数据有误差。
    - 标准化数据（注意 data.std(ddof = 0)）。
    - 每个对应的值两两相乘，取这些乘积的平均数作为最后结果。
  + kendall，反映分类变量相关性的指标，通常用于评分数据一致性水平研究，比如评委打分，数据排名等。
  + spearman：非线性的，非正太分布的数据的相关系数。
* 案例：构造一元二次方程，y=2x\*x+1 非线性关系

def compute(x):

  return 2\*x\*x+1

x=[i for i in range(100)]

y=[compute(i) for i in x]

data = DataFrame({'x':x,'y':y})

print(data.corr())

print(data.corr(method='spearman')) # 查看pearson系数

print(data.corr(method='kendall'))

          x         y

x  1.000000  0.967644

y  0.967644  1.000000

     x    y

x  1.0  1.0

y  1.0  1.0

     x    y

x  1.0  1.0

y  1.0  1.0