**pandas**

<https://www.pypandas.cn/docs/getting_started/10min.html#%E7%94%9F%E6%88%90%E5%AF%B9%E8%B1%A1>

<https://juejin.im/post/6844903850592174094>

<https://juejin.im/post/6844903815485849613>

<https://juejin.im/search?query=pandas&type=all>

**1.介绍**

pandas是python中的数据处理包，类型有2种，DataFrame（二维数据）、Serise（一维数组）。

建好一个python脚本， 引入pandas方法，常常和numpy搭配使用，如下所示：

|  |
| --- |
| import pandas as pd  import numpy as np |

说明：as是对pandas的重命名（类似小名），以后就使用pd即可；numpy类似。

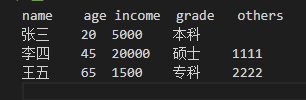
**2.数据保存读取和保存**

数据分析常见的数据保存在txt、csv、excel、数据库里面（这里介绍mysql），利用pandas读取和保存这些数据，具体其他格式可以参考。

**2.1读取和保存txt/csv文件**

【这两种放在一起讲，使用的方法是一样】

首先看txt/csv文件是否有表头、每个字段的分割方式。

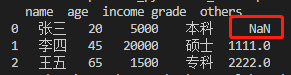


**方式一：有表头、分割方式\t**

假如有表头【name、age、income、grade、others】，分割方式是\t，那么读取的方式，需要设置sep，【如果不设置sep，则默认以逗号“，”分割数据】如下：

|  |
| --- |
| # 读取  df = pd.read\_csv(“1.txt”, sep=”\t”)  df = pd.read\_csv(“1.csv”, sep=”\t”)  print(df.head()) # 打印读取的数据head()表示显示前10行，head(n)，这个n可自定义打印多少行  # 保存  df.to\_csv("1.txt", sep="\t", index=False, header=False) #保存文档，sep=”\t”保存数据以\t分割，index=False不需要索引，header=False不需要表头  index：从0开始，可以理解为序号。 |

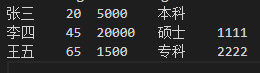
打印结果：



通过打印函数可以看到读取txt/csv文件的数据，其中我们可以看到，数据是带有表头的。并且发现，如果数据存在空值，该空值的地方会显示“NaN”，这里先不考虑，后面统一讲解。

**方式二：没有表头、分割方式\t**

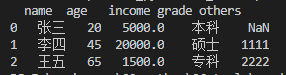
假如你的数据是这样的：



如果不添加表头，则读取数据，则认为第一行为表头，此时需要你添加表头，添加表头的方式如下，我们发现如果有空值在某一列，一定要设置成str类型的，不会报错。

|  |
| --- |
| # 读取  df = pd.read\_csv("1.txt",  sep="\t",  names=["name", "age", "income", "grade", "others"],  dtype={"name":np.str,  "age":np.int32,  "income":np.float,  "grade":np.str,  "others":np.str})  print(df.head()) |

得到结果如下：



**2.2读取excel文件**

|  |
| --- |
| # 读取  df = pd.read\_excel("1.xlsx") # 这里不需要使用分割符，excel默认\t  print(df.head())  # 保存 |

**2.3读取mysql数据库**

|  |
| --- |
| # 读取  import pymysql  conn = pymysql.connect(  host = "localhost",  user = "user",  passwd = "123456",  db = "testdb",  port = 3306,  charset='utf8'  )  sql = "select \* from testdb.tb\_info"  df = pd.read\_sql(sql, con=conn)  print(df.head())  # 保存成txt/csv/excel均可，也可以重新保存回mysql |

【备注】如果连接mysql数据库，建议使用mysql数据库连接池。防止连接过程中因为网络问题出现连接断开的情况发生。

**这里特别注意，**如果读取大数据量进行统计的时候，如果每列数据独立，则可以分块读取数据。

例如：

|  |
| --- |
| data = pd.read\_csv(“1.csv”, sep=”\t”, chunksize=100000)  for df in data:  process(df) # 处理每一块的数据，这样减少大内存使用 |

**3.DataFrame和Series**

DataFrame是二维数组，而Serise是一位数组

以上述例子为例：

df是二维数组，一位数组怎么取呢，很简单，利用**字段名**即可，例如：

|  |
| --- |
| name = df[“**name**”] # 单列    #### 多列  n = [“name”, ”age”] # 建议定义一个变量防止报错  sub\_df = df[n] |

**4.基本函数**

|  |
| --- |
| df.shape # 二维数组的行列    df.info() # 字段详细信息，可以浏览每个字段的属性、列数、字段类型、实体个数等。    df.describe() # 可以查看每个数据字段统计信息，例如：个数、均值、方差、最大/最小值、百分占比等。    df.count() #统计每列数据个数，非NaN值的数量    df.head() # 查看头部，head(n)，n可以取任意值  df.tail() # 查看尾部，tail(n)，n可以取任意值  df.index 查看索引  df.columns 查看列名  df.to\_numpy() 转成numpy格式的数据    df.sort\_index(axis=1, ascending=False) # 索引排序：按照轴**排序**  df.sort\_values(by=[“列名”]) # 值排序：按照某个列名进行**排序**，常用。  df.sum() # 每列数据求和    df.cumsum() # 每列累积值    df.min() # 每列最小值    df.max() # 每列最大值    假如数据如下：    df.idxmin(axis=0) # **每列**最小索引值，如果**没有axis**参数，默认为列    第一列（col1）最小值是第1个数；  第二列（col2）最小值是第3个数；  第三列（col3）最小值是第2个数。  df.idxmin(axis=1) # **每行**最小索引值    第一行最小的数是第一列；  第二行最小的数是第三列；  第三行最小的数是第二列。  df.idxmax(axis=0) # **每列**最大索引值  df.idxmax(axis=1) # **每行**最大索引值 |

**5.分片取数据**

|  |
| --- |
| # 取前2行数据  df[:2]    # 取前2列数据  df.loc[:, [“name”, “age”]]    【以上2种方式结合，可以取明确的数据】  等价于： df.at[0, “name”] |

**6. append函数**

df可以看成list，支持append方法，如果索引想从新编号，则设置

df.append({“字段名1”: xxx，”字段名2”:xxx}, ignore\_index=False)

ignore\_index=False即可

**7.条件取数**

|  |
| --- |
| df = df[df.age>40] 查看年龄大于40岁的取法  或者： df=df[df[“age”]>40]    df[df.name.isin(["张三", "李四"])] # 通过isin()筛选数据 |

**8.生成时间序列**

|  |
| --- |
| s1 = pd.Series([1, 2, 3, 4, 5, 6], index=pd.date\_range('20130102', periods=6)) |

**9.赋值**

DataFrame和Series有一个很好的性质，就是可以对某一列进行操作。

|  |
| --- |
| # DataFrame    # Series一列+1 |

**10.apply的高级用法**

不管是DataFrame和Series都可以使用apply来处理某一列或者一行的数据。

|  |
| --- |
| def comput(row):  return row.age+1  df[“new\_age”] = df.apply(comput) |

**11.缺失值处理**