









In [ ]:

```
# 设定系统环境
import pandas as pd
pd.options.display.max_rows = 10 # 设定自由列表输出最多为10行
pd.__version__ # 显示当前Pandas版本号
```

# 1 获取数据

## 1.1 新建数据框

	 time	 id	 s0	 s2	 s3	 s4	 s5	 s7
1	200704	1	100北京	男	20	本科	公司普通职...	未婚
2	200704	2	100北京	男	24	高中/中专	公司普通职...	未婚
3	200704	3	200上海	男	20	高中/中专	无业/待业/...	未婚
4	200704	4	100北京	女	65	大专	企/事业管...	已婚
5	200704	5	200上海	女	40	高中/中专	公司普通职...	已婚
6	200704	6	100北京	男	50	大专	公司普通职...	已婚
7	200704	7	100北京	女	53	初中/技校...	退休	已婚
8	200704	8	300广州	男	44	高中/中专	无业/待业/...	已婚
9	200704	9	200上海	女	35	大专	公司普通职...	已婚
10	200704	10	200上海	男	21	本科	学生	未婚

Q：如何记录每个变量所携带的数据？

A：变量列就是一个有顺序的数据序列（一维数组），可以看作是一个增强版的list。  
对应了numpy中的Series格式，偷懒的话直接用list格式提供即可。  
[1,2,3,4]  
["test","train","test","train"]

Q：如何指定数据集的基本结构（变量定义）？

A：字典格式可以为每个字典元素提供名称，自然是最佳选择。  
{  
    'var1' : value,  
    'var2' : value,  
    'var3' : value,  
    'var4' : value  
}

In [ ]:

```
df1 = pd.DataFrame(  
{  
'var1' : 1.0,  
'var2' : [1,2,3,4],  
'var3' : ["test","train","test","train"],  
'var4' : 'cons'  
}  
)  
  
df1
```

**pd.DataFrame(**

data=None : 数据列表，字典格式时直接同时提供变量名

columns=None : 变量名列表

**)**

In [ ]:

```
# 以list形式按行提供数据  
df1 = pd.DataFrame(data = [[1,"test"], [2,"train"],  
                           [3,"test"],[4,"train"]],  
                   columns = [ 'var2', 'var3' ]  
)  
  
df1
```

## 1.2 读入文本格式数据文件

**pd.read\_csv(**

filepath\_or\_buffer : 要读入的文件路径

sep = ',' : 列分隔符

header = 'infer' : 指定数据中的第几行作为变量名

names = None : 自定义变量名列表

index\_col = None : 将会被用作索引的列名，多列时只能使用序号列表

usecols = None : 指定只读入某些列，使用索引列表或者名称列表均可。

[0,1,3], ["名次", "学校名称", "所在地区"]

encoding = None: 读入文件的编码方式

utf-8/GBK, 中文数据文件最好设定为utf-8

na\_values : 指定将被读入为缺失值的数值列表，默认下列数据被读入为缺失值:

' ', '#N/A', '#N/A N/A', '#NA', '-1.#IND',  
'-1.#QNAN', '-NaN', '-nan', '1.#IND', '1.#QNAN',  
'N/A', 'NA', 'NULL', 'NaN', 'n/a', 'nan', 'null'

): 读取csv格式文件，但也可通用于文本文件读取

In [ ]:

```
df4 = pd.read_csv("univ.csv", encoding="GBK")  
df2
```

`pandas.read_table()`: 更通用的文本文件读取命令

主要的区别在于默认的`sep="\t"`, 即tab符号。

In [ ]:

```
df2 = pd.read_table("univ.csv", sep=',', encoding="gbk" )
df2
```

## 1.3 读入EXCEL文件

`pd.read_excel()`

`filepath_or_buffer`: 要读入的文件路径

`sheet_name`: 要读入的表单, 字符串或者数字序号均可, 默认读入第一个

)

In [ ]:

```
df2 = pd.read_excel("高校信息.xlsx", sheet_name = 0)
df2
```

## 1.4 读入统计软件数据集

### 1.4.1 读入SAS/Stata数据文件

`pd.read_sas()`

`pd.read_stata()`

In [ ]:

```
df4 = pd.read_sas('air.sas7bdat')
df4
```

### 1.4.2 读入SPSS数据文件

安装程序包

```
# pip install savReaderWriter
# python setup.py install
```

使用程序包

```
savReaderWriter.SavReader('数据文件名').all()
```

不推荐使用的原因

本质上只能将数据文本抽取出来，不能保留原有行列框架  
丢失了值标签、变量列标签、存储格式等几乎全部附加设置  
SPSS有完善的另存数值文件格式功能，使用起来更方便

In [ ]:

```
import savReaderWriter as savRW
spssfile = savRW.SavReader('air.sav', ioUtf8 = True).all()
spssfile
```

## 1.5 读入数据库文件

### 1.5.1 配置所需的程序包和驱动

所需程序包：

SQLAlchemy engine: 几乎可以使用任何常见的DB格式和操作命令  
DBAPI2 connection: 只对sqlite3有较完整的支持

驱动配置: [SQLAlchemy docs \(http://docs.sqlalchemy.org/en/latest/dialects/index.html\)](http://docs.sqlalchemy.org/en/latest/dialects/index.html)

针对每一种数据库格式，都需要配置好可用的驱动程序  
如何选择合适的驱动程序？

In [ ]:

```
# conn.txt格式: 用户名:密码@数据库服务地址:端口/库名
conndf = pd.read_csv("c:/conn.txt", header = None)
connstr = conndf.iloc[0,0]
type(connstr)
```

In [ ]:

```
from sqlalchemy import create_engine
eng = create_engine('mysql+pymysql://' + connstr + '?charset=utf8')
type(eng)
```

### 1.5.2 读入数据表

pd.read\_sql(

sql : 需要执行的SQL语句/要读入的表名称  
con : SQLAlchemy连接引擎名称  
index\_col = None : 将被用作索引的列名称  
columns = None : 当提供表名称时，需要读入的列名称list

)

read\_sql\_query()

read\_sql\_table()

In [ ]:

```
tbl = pd.read_sql("select code, name from basic", con = eng)
tbl
```

In [ ]:

```
tbl = pd.read_sql("select count(*) from basic", con = eng)
tbl
```

## 1.6 实战：读入北京PM2.5数据

数据来源：<http://www.stateair.net/web/historical/1/1.html> (<http://www.stateair.net/web/historical/1/1.html>)

要求：PM25子目录中已经下载了北京从2008年至今的美领馆版本PM2.5数据，请将这些数据文件分别读入为不同的数据框，名称可以为bj2008、bj2009等。

提示：

每个文件的前三行为注释，第四行为变量名，因此读入时需要跳开前三行  
多个文件分别读入为不同的数据框，可以自行编制一个函数完成该重复任务  
数据文件的依次读取可以手工写入文件名称，也可以使用`os.walk()`方法遍历得到

## 2 保存数据

### 2.1 保存数据至外部文件

`df.to_csv()`

`filepath_or_buffer`: 要保存的文件路径  
`sep = ','`: 列分隔符  
`columns`: 需要导出的变量列表  
`header = True`: 指定导出数据的新变量名，可直接提供list  
`index = True`: 是否导出索引  
`mode = 'w'`: Python写模式，读写方式: `r`, `r+`, `w`, `w+`, `a`, `a+`  
`encoding = 'utf-8'`: 默认导出的文件编码格式

)

In [ ]:

```
df2.to_csv('temp.txt', columns = ['名次', '总分'], header = ['名次2', '总分2'], index = F
```

`df.to_excel()`

`filepath_or_buffer`: 要读入的文件路径  
`sheet_name = 'Sheet1'`: 要保存的表单名称

)

In [ ]:

```
df2.to_excel('temp.xlsx', index = False, sheet_name = 'data')
```

df.to\_stata()

## 2.2 保存数据至数据库

df.to\_sql(

name : 将要存储数据的表名称  
con : SQLAlchemy引擎/DBAPI2连接引擎名称  
if\_exists = 'fail' : 指定表已经存在时的处理方式  
    fail : 不做任何处理（不插入新数据）  
    replace : 删除原表并重建新表  
    append : 在原表后插入新数据  
index = True : 是否导出索引

)

In [ ]:

```
'''  
apptbl.to_sql(name="jt_histrec", con=eng,  
              if_exists='append', index=False)  
'''
```

## 2.3 实战：保存北京PM2.5数据为数据文件

要求：尝试将PM2.5数据保存为csv、EXCEL等格式，并使用各种不同的参数设置。

# 3 变量列的基本操作

## 3.1 对数据作简单浏览

In [ ]:

```
print(df2)
```

In [ ]:

```
# 数据框的基本信息  
df2.info()
```

In [ ]:

```
# 浏览前几条记录  
df2.head(10)
```

Q: 课程学习遇到不懂怎么办？

- ✧ 本课程提供额外福利：QQ 群供学员交流心得，群号：  
630030855，可直接扫下方的二维码进入。
- ✧ 老师有空时也会参与讨论，但请不要把你的工作问题直接让老师解决。
- ✧ 老师鼓励学员多思考、多动手，通过自己努力解决问题，这样能发现乐趣、有成就感、成长快。

