In [3]:
```
import numpy as np
import pandas as pd
```

第一部分--关于CSV文件

一、什么是CSV

逗号分隔值（Comma-Separated Values，CSV），其文件以纯文本形式存储表格数据（数字和文本），文件的每一行都是一个数据记录。每个记录由一个或多个字段组成，用逗号分隔。使用逗号作为字段分隔符是此文件格式的名称的来源，因为分隔字符也可以不是逗号，有时也称为字符分隔值。

CSV广泛用于不同体系结构的应用程序之间交换数据表格信息，解决不兼容数据格式的互通问题，一般按照传输双方既定标准进行格式定义，而其本身并无明确格式标准。

CSV用逗号分隔字段的基本思想是清楚的，但是当字段数据也可能包含逗号或者甚至嵌入换行符时，该想法变得复杂。 CSV实现可能无法处理这些字段数据，或者可能会使用引号来包围字段。引用并不能解决所有问题：有些字段可能需要嵌入引号，因此CSV实现可能包含转义字符或转义序列。

RFC 4180提出了MIME类型（”text/csv”）对于CSV格式的标准，可以作为一般使用的常用定义，满足大多数实现似乎遵循的格式。

二、CSV的格式规范（RFC 4180）

下面每种介绍的格式，可以拷贝到txt文本中（CRLF用回车代替，
这里显示写出来是为了区分有没有回车符），然后改后缀名为csv打开看看是不是相应的表格格式

- 1. 每一行记录位于一个单独的行上，用回车换行符CRLF(也就是\r\n)分割

aaa,bbb,ccc CRLF zzz,yyy,xxx CRLF

- 1. 文件中的最后一行记录可以有结尾回车换行符，也可以没有

aaa,bbb,ccc CRLF
zzz,yyy,xxx

- 1. 第一行可以存在一个可选的标题头，格式和普通记录行的格式一样
  标题头要包含文件记录字段对应的名称，应该有和记录字段一样的数量

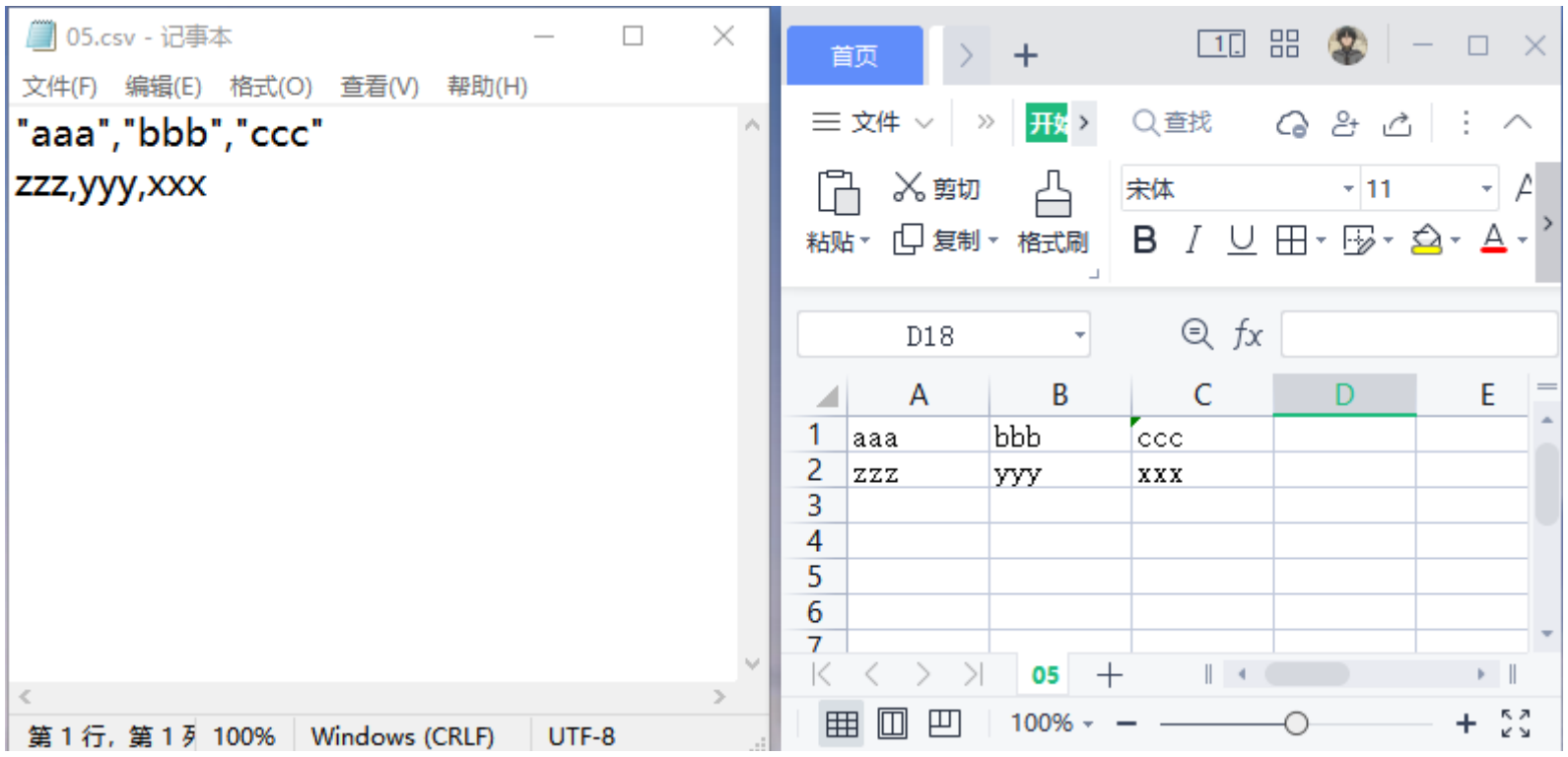
column_name1,column_name2,column_name3 CRLF
aaa,bbb,ccc CRLF
zzz,yyy,xxx CRLF

- 1. 在标题头行和普通行每行记录中，会存在一个或多个由半角逗号(,)分隔的字段
  整个文件中每行应包含相同数量的字段，空格也是字段的一部分，不应被忽略
  每一行记录最后一个字段后不能跟逗号（通常用逗号分隔，也有其他字符分隔的CSV，需事先约定）

aaa,bbb,ccc

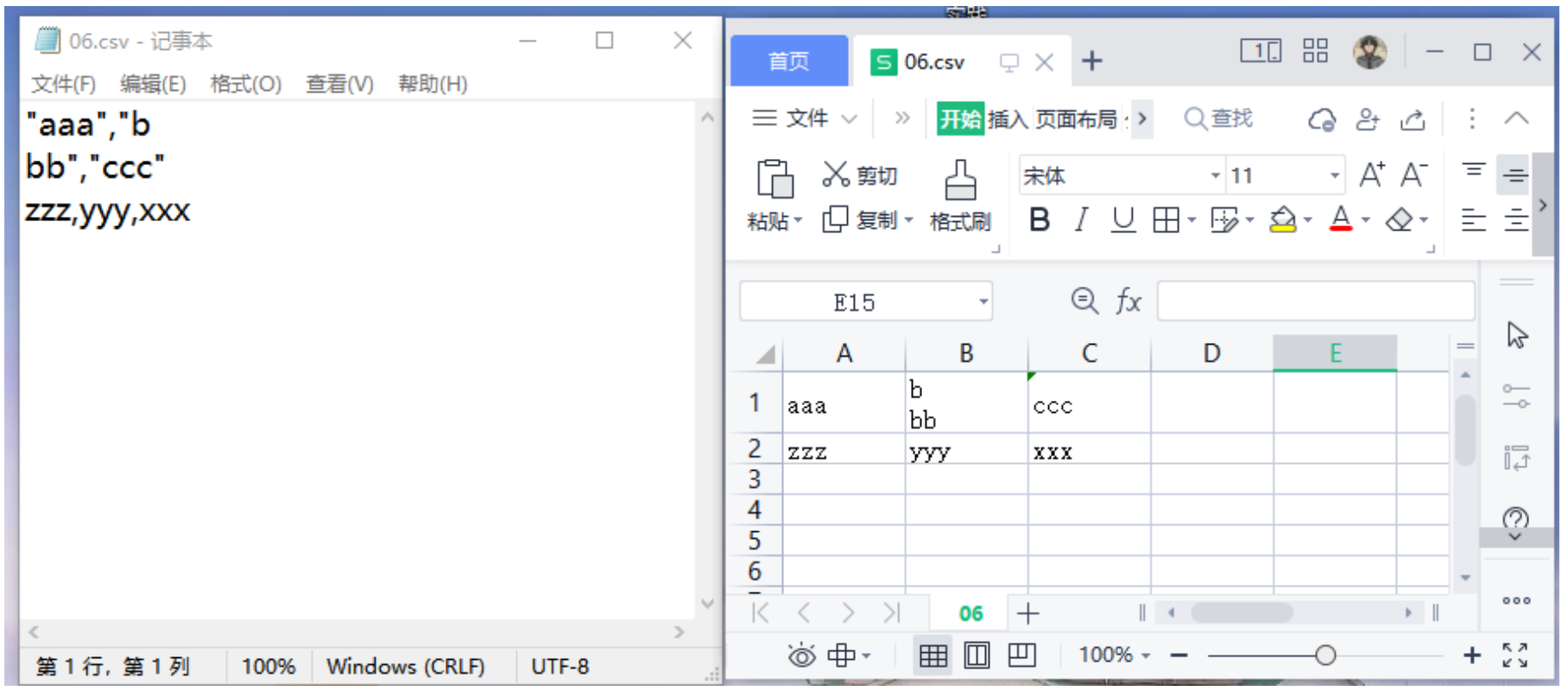
- 1. 每个字段可用也可不用半角双引号(“)括起来
  不过有些程序，如Microsoft的Excel就根本不用双引号
  如果字段没有用引号括起来，那么该字段内部不能出现双引号字符

"aaa","bbb","ccc" CRLF
zzz,yyy,xxx



- 1. 字段中若包含回车换行符、双引号或者逗号，该字段需要用双引号括起来

"aaa","b CRLF
bb","ccc" CRLF
zzz,yyy,xxx



- 1. 如果用双引号括字段，那么出现在字段内的双引号前必须加一个双引号进行转义

三、关于CSV文件解析

上面说过，CSV并不是一种单一的、定义明确的格式（尽管RFC 4180有一个被通常使用的定义）。因此在实践中，术语“CSV”泛指具有以下特征的任何文件：

- 纯文本，使用某个字符集，比如ASCII、Unicode、EBCDIC或GB2312；
- 由记录组成（典型的是每行一条记录）；
- 每条记录被分隔符分隔为字段（典型分隔符有逗号、分号或制表符；有时分隔符可以包括可选的空格）；
- 每条记录都有同样的字段序列。
  所以，在常规的约束条件下，存在着许多CSV变体，故CSV文件并不完全互通，如使用约定好的其他分隔符、转义规则等。
  因此，实际使用CSV需要数据交换双方约定规则（其实大体思路不变，一些细节），在进行CSV文件读写就免不了进行文件的解析。

正如CSV并不明确的格式，CSV文件的解析同样没有标准方法，一般可以自己实现读写，网上也有很多种不同语言的实现版本。例如opencsv、csvreader等。

四、关于文本文件与二进制文件

参考链接

简单来说就是这么回事，计算机中数据都是以二进制保存的

- 文本文件：
  如果采用ASCII字符集，会一次性读入8bit（1byte）转换成16进制的数字，这个数字就是文件的存储形式，
  然后根据ASCII字符集，将其对应到128字符，假如包含汉字，按照ASCII解码就会出现乱码，这时候改为utf-8即可。
  文本文件举例：txt，csv，c，cpp，py，html，word，excel等等
- 二进制文件：
  一次性可以读入不同bit的二进制，转换成数字，这就是直接基于值编码
  二进制文件举例：图片，视频，音频等等

两者的区别不是物理层面的，而是逻辑上的，一个是基于字符编码，一个是基于值编码

关于图片（二进制文件）：
单色位图： 每个像素最多可以表示2种颜色，只需要使用长度为1的二进制位来表示，因此每个像素占1/8byte
16色位图： 每个像素最多可以表示16种颜色，所以只需要长度为4的二进制表示，因此每个像素占1/2byte
24位位图：即RGB三原色位图 每个像素占3个byte
256色位图： 每个像素最多可以表示256中颜色，所以只需要长度是8的二级制位表示就可以了，因此每个像素占1byte

第二部分--从csv到DataFrame：pandas.read_csv()

这个方法参数加起来一共有40多个，这里主要介绍最基本的几种

In [15]:
```
header=pd.Index(['体力值','种族值','战斗力'],name='属性')
list1=['男','男','男','男','女','女','女']
list2=['百里守约','项羽','猪八戒','蔡徐坤','上官婉儿','小乔','蔡文姬']
index_col=pd.MultiIndex.from_arrays([list1,list2],names=['性别','姓名'])
df=pd.DataFrame(data=np.random.randint(1,100,[7,3]),index=index_col,columns=header)
df.to_csv('C:\\Users\\Amadeus\\Desktop\\01.csv')
```

一、路径，分隔符，编码

说明：

- 路径地址这里填的本地地址，windows系统下是反斜杠，但是python字符串里反斜杠表示转义，
  所以地址复制过来不能直接用，前面加个r取消转义或者里面反斜杠前加个转义反斜杠
- 分隔符默认就是逗号，一般不需要填
- 当文件中包含中文字符时，采用utf-8编码，当然，这里也不用填，默认的编码貌似也支持中文

In [18]:
```
df1=pd.read_csv(r'C:\Users\Amadeus\Desktop\01.csv',sep=',',encoding='utf-8')
df1
```

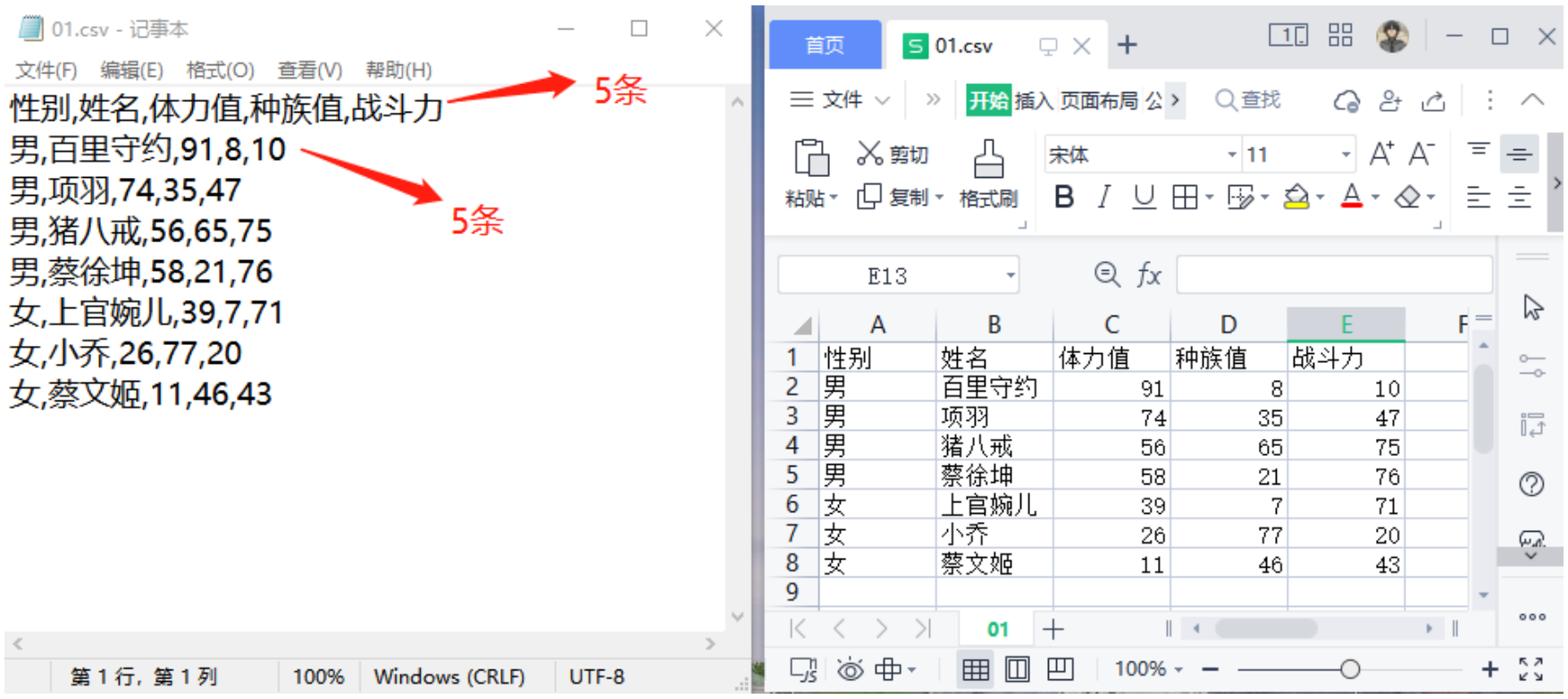
Out[18]:

	性别	姓名	体力值	种族值	战斗力
0	男	百里守约	91	8	10
1	男	项羽	74	35	47
2	男	猪八戒	56	65	75
3	男	蔡徐坤	58	21	76
4	女	上官婉儿	39	7	71
5	女	小乔	26	77	20
6	女	蔡文姬	11	46	43

二、不设置列索引的3种情况

注：如果不对索引进行设置，那么情况绝对不止以下3种，所以如果情况复杂，还是需要一些参数设置的，在后面会介绍

- （1）.如下图，第一行的数据条数和后面的一致，那么会将第一行作为默认列索引，下面作为data，行索引采用默认数字

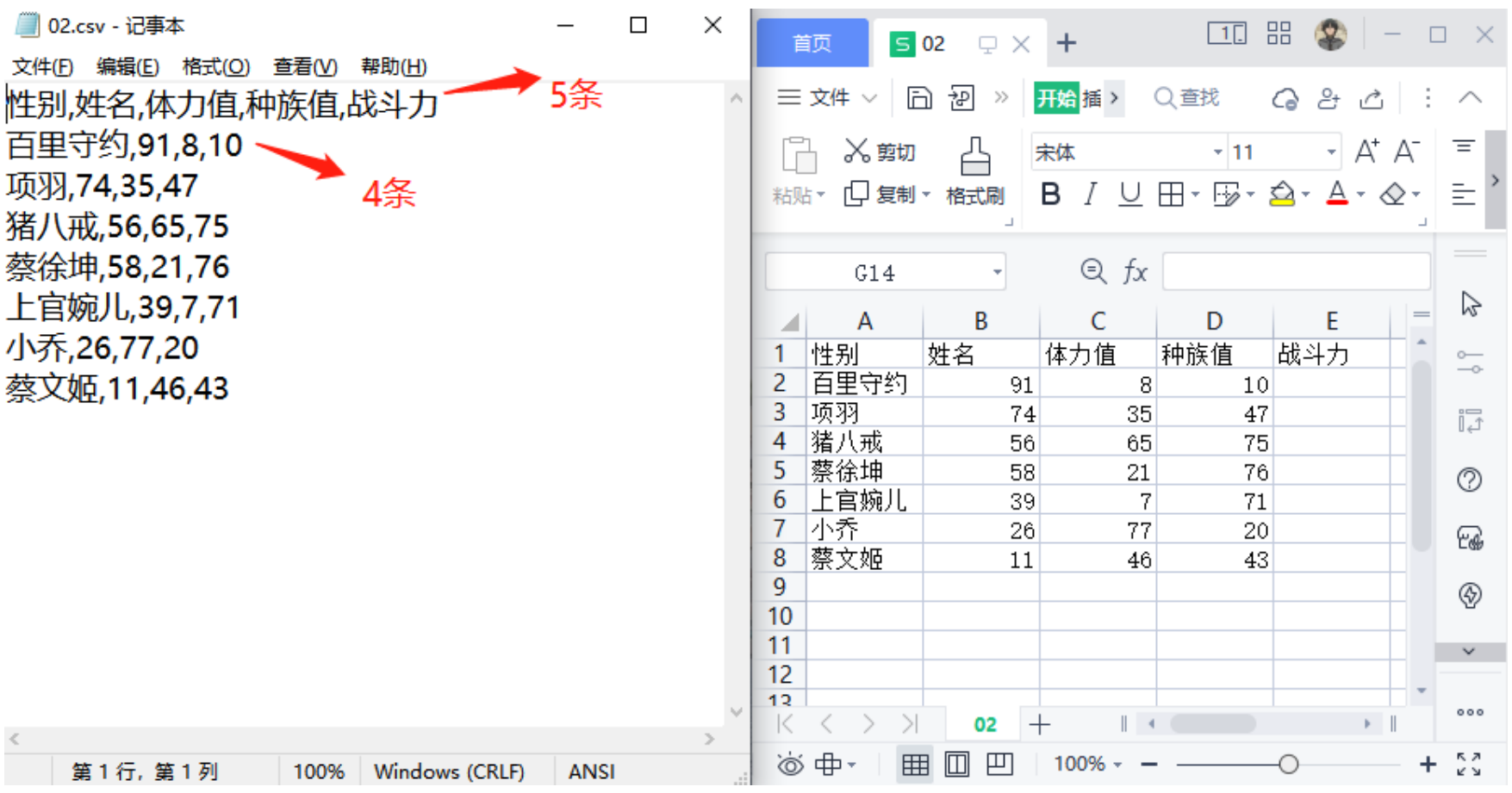


In [19]:
```
pd.read_csv(r'C:\Users\Amadeus\Desktop\01.csv',sep=',',encoding='utf-8')
```

Out[19]:

	性别	姓名	体力值	种族值	战斗力
0	男	百里守约	91	8	10
1	男	项羽	74	35	47
2	男	猪八戒	56	65	75
3	男	蔡徐坤	58	21	76
4	女	上官婉儿	39	7	71
5	女	小乔	26	77	20
6	女	蔡文姬	11	46	43

- （2）.如下图，第一行的条目比下面的多了一行，这种第一行还是作为列索引，下面的作为数据，不足的部分补NAN

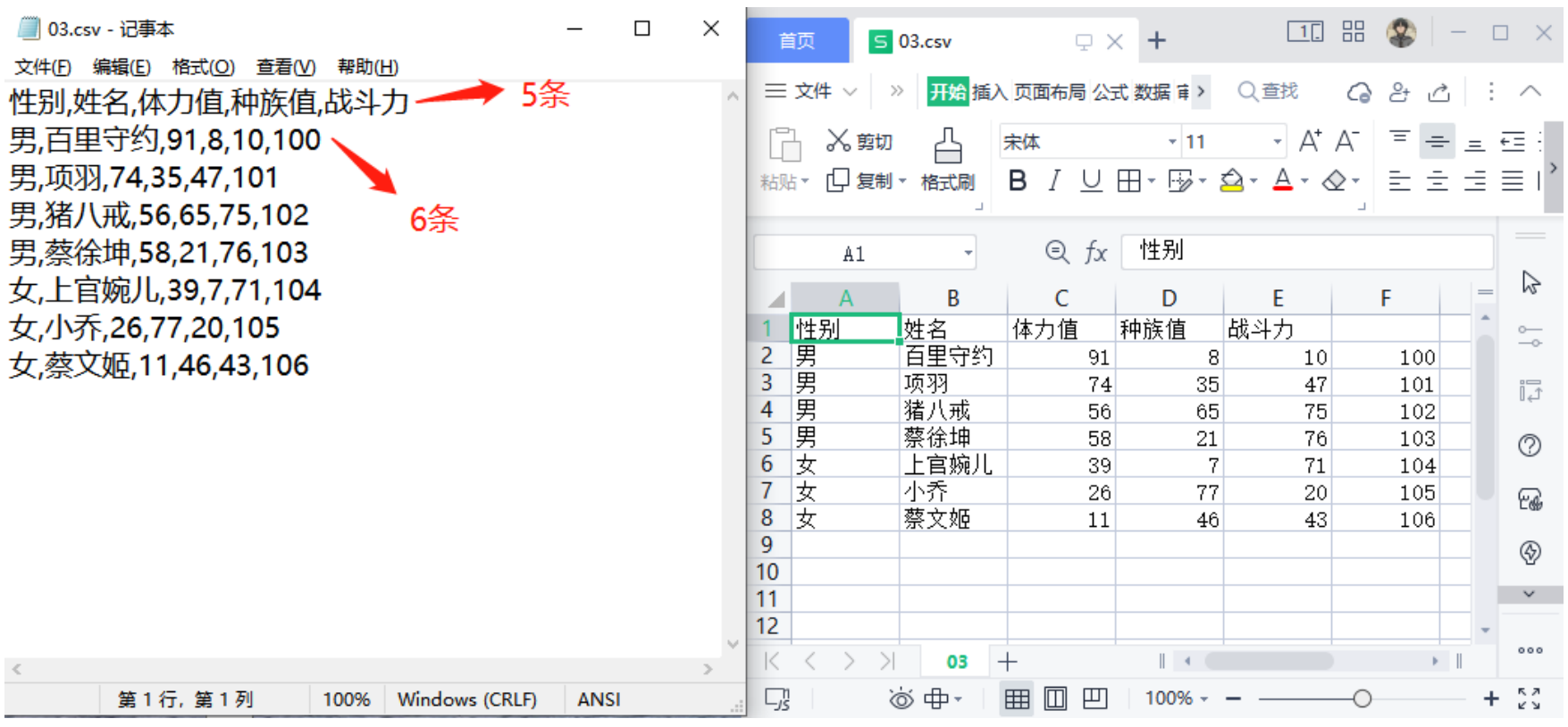


In [22]:
```
pd.read_csv(r'C:\Users\Amadeus\Desktop\02.csv',sep=',',encoding='gbk')
```

Out[22]:

	性别	姓名	体力值	种族值	战斗力
0	百里守约	91	8	10	NaN
1	项羽	74	35	47	NaN
2	猪八戒	56	65	75	NaN
3	蔡徐坤	58	21	76	NaN
4	上官婉儿	39	7	71	NaN
5	小乔	26	77	20	NaN
6	蔡文姬	11	46	43	NaN

- （3）.如下图，第一行比下面的少一条，会把第一列作为行索引

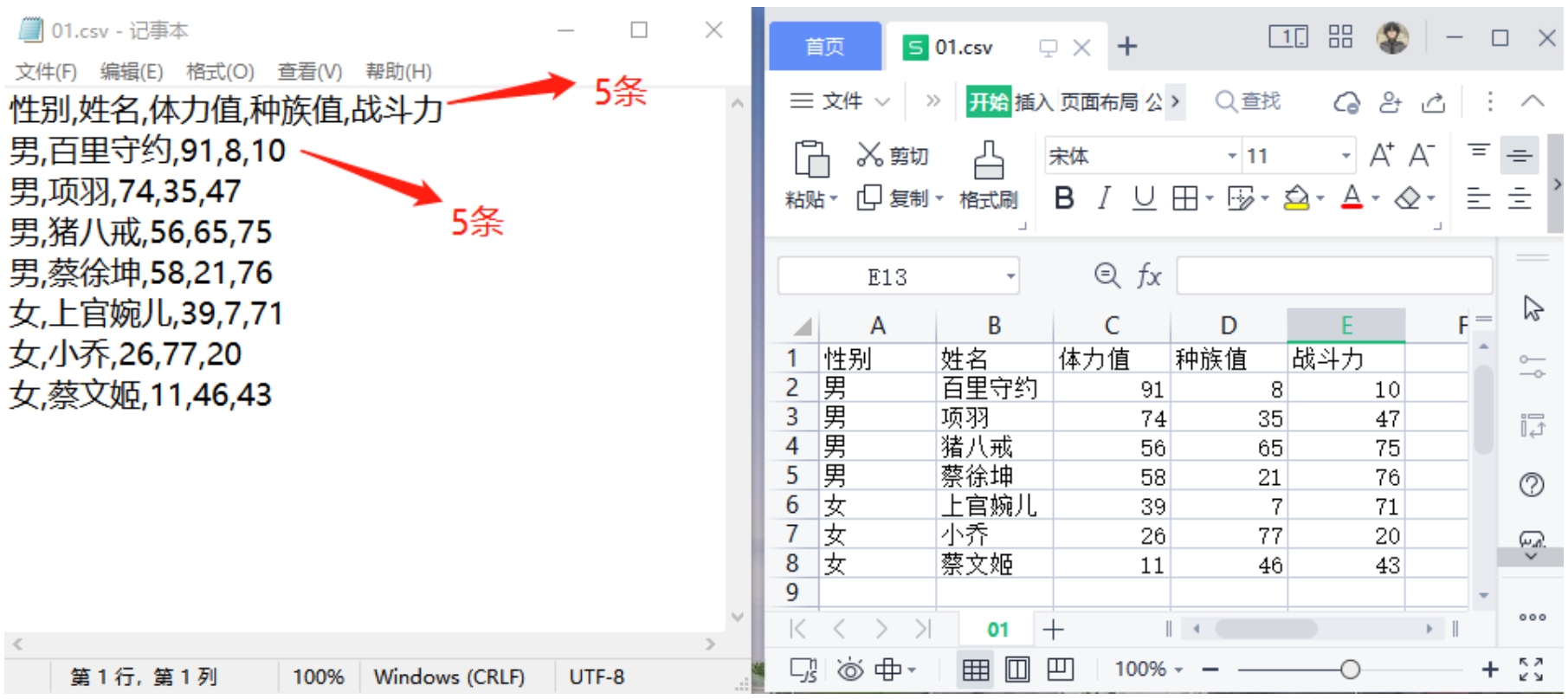


In [23]:
```
pd.read_csv(r'C:\Users\Amadeus\Desktop\03.csv',sep=',',encoding='gbk')
```

Out[23]:

	性别	姓名	体力值	种族值	战斗力
男	百里守约	91	8	10	100
男	项羽	74	35	47	101
男	猪八戒	56	65	75	102
男	蔡徐坤	58	21	76	103
女	上官婉儿	39	7	71	104
女	小乔	26	77	20	105
女	蔡文姬	11	46	43	106

四、使用header与names设置表头（列索引）



- （1）.仅设置header

header可以是：数字，数字列表，标签，标签列表

In [25]:
```
pd.read_csv(r'C:\Users\Amadeus\Desktop\01.csv',sep=',',header=1)
```

Out[25]:

	男	百里守约	91	8	10
0	男	项羽	74	35	47
1	男	猪八戒	56	65	75
2	男	蔡徐坤	58	21	76
3	女	上官婉儿	39	7	71
4	女	小乔	26	77	20
5	女	蔡文姬	11	46	43

(2).仅设置names，使用自定义列索引，一般用于没有表头的情况

names可以是：标签列表

In [28]:
```
pd.read_csv(r'C:\Users\Amadeus\Desktop\01.csv',sep=',',names=['第1列','第2列','第3列','第4列','第5列'])
```

Out[28]:

	第1列	第2列	第3列	第4列	第5列
0	性别	姓名	体力值	种族值	战斗力
1	男	百里守约	91	8	10
2	男	项羽	74	35	47
3	男	猪八戒	56	65	75
4	男	蔡徐坤	58	21	76
5	女	上官婉儿	39	7	71
6	女	小乔	26	77	20
7	女	蔡文姬	11	46	43

(3).header和names都设置

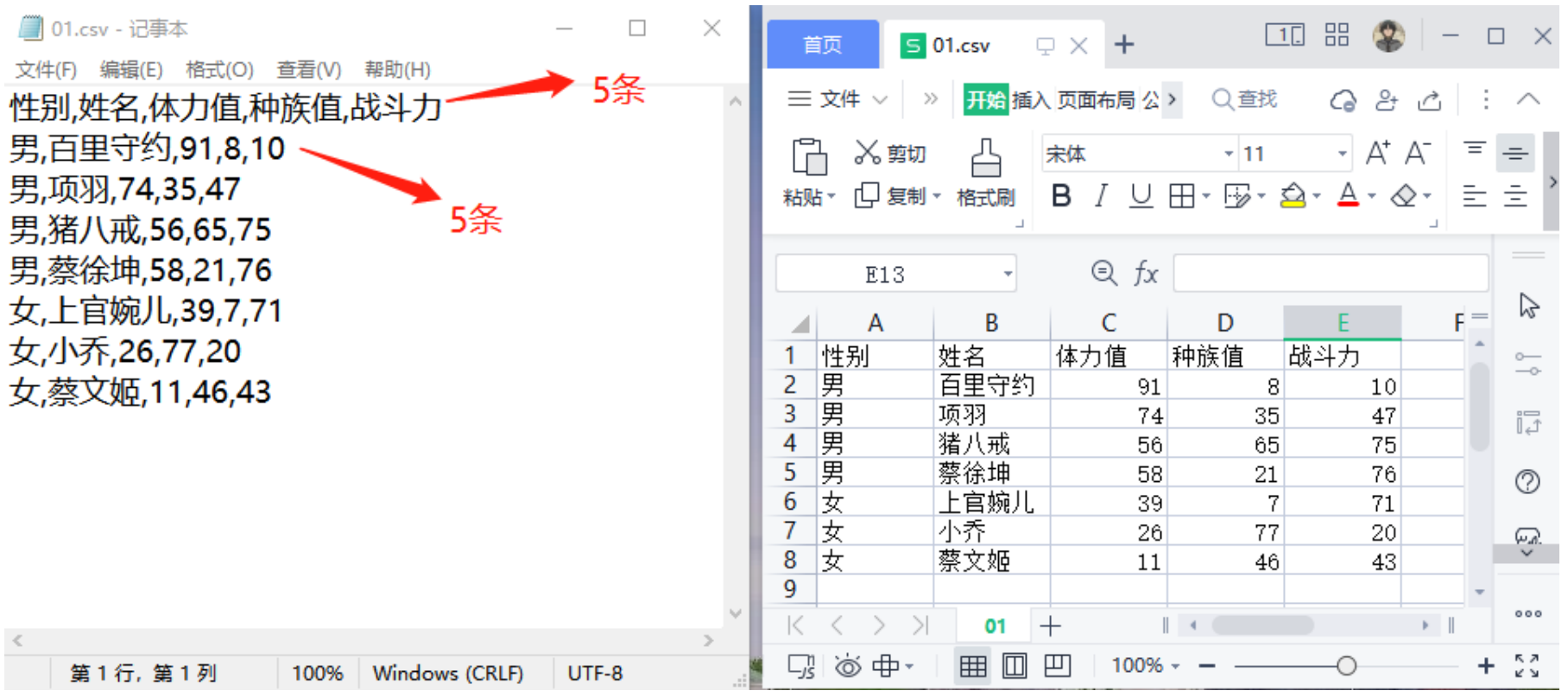
先将某一行设置为表头，再用names覆盖，相当于把那行数据抹除了,一般用于替换原来的表头

In [29]:
```
pd.read_csv(r'C:\Users\Amadeus\Desktop\01.csv',sep=',',header=0,names=['第1列','第2列','第3列','第4列','第5列'])
```

Out[29]:

	第1列	第2列	第3列	第4列	第5列
0	男	百里守约	91	8	10
1	男	项羽	74	35	47
2	男	猪八戒	56	65	75
3	男	蔡徐坤	58	21	76
4	女	上官婉儿	39	7	71
5	女	小乔	26	77	20
6	女	蔡文姬	11	46	43

五、使用index_col设置行标签



- index_col可以是：数字，数字列表，标签，标签列表

In [30]:
```
pd.read_csv(r'C:\Users\Amadeus\Desktop\01.csv',sep=',',header=0,index_col=0)
```

Out[30]:

性别	姓名	体力值	种族值	战斗力
男	百里守约	91	8	10
男	项羽	74	35	47
男	猪八戒	56	65	75
男	蔡徐坤	58	21	76
女	上官婉儿	39	7	71
女	小乔	26	77	20
女	蔡文姬	11	46	43

In [31]:
```
pd.read_csv(r'C:\Users\Amadeus\Desktop\01.csv',sep=',',header=0,index_col=['性别','姓名'])
```

Out[31]:

性别	姓名	体力值	种族值	战斗力
男	百里守约	91	8	10
	项羽	74	35	47
	猪八戒	56	65	75
	蔡徐坤	58	21	76
女	上官婉儿	39	7	71
	小乔	26	77	20
	蔡文姬	11	46	43

六、其他常用参数

与行相关的：

- skiprows=5: 设置从第5行开始读取数据（如果数据开始有一大堆说明的行需要跳过）
- skiprows=[0,2,5]： 设置跳过0，2，5行

与列相关的：

- usecols=['种族值','战斗力']: 指定需要读入的列
- usecols=[3,4]： 同上

与值相关的：

- na_values=55： 把数据_data中的55全变成NAN
- na_values={'战斗力':55}： 仅仅把战斗力那一列中的55替换为NAN