第3章

不同格式数据的读取与写入

数据往往存在于各种各样的数据库、各种各样的文件中，从外部文件读写数据是机器学习的前提，是机器学习必不可少的部分。在读写数据时可以对数据做一定的处理，为接下来使用数据集训练机器模型打好基础。

3.1 使用 csv 模块读取和写入 csv 文件

csv(comma separated values，逗号分隔值)文件是一种用来存储表格数据(数字和文本)的纯文本格式文件，文档的内容是由“,”分隔的一列列数据构成的，它可以被导入各种电子表格和数据库中。纯文本意味着该文件是一个字符序列。在 csv 文件中，数据“栏”(数据所在列，相当于数据库的字段)以逗号分隔，可允许程序通过读取文件为数据重新创建正确的栏结构(如把两个数据栏的数据组合在一起)。csv 文件由任意数目的记录组成，记录间以某种换行符分隔，一行即为数据表的一行；每条记录由字段组成，字段间的分隔符最常见的是逗号或制表符。可使用 Word、记事本、Excel 等方式打开 csv 文件。

创建 csv 文件的方法有很多，最常用的方法是用电子表格创建，如 Microsoft Excel。在 Microsoft Excel 中，选择“文件”>“另存为”命令，然后在“文件类型”下拉选择框中选择“CSV (逗号分隔)(*.csv)”，最后单击“保存”按钮即创建了一个 csv 格式的文件。

Python 的 csv 模块提供了多种读取和写入 csv 格式文件的方法。

本节基于 consumer.csv 文件，其内容为：

```
客户年龄,平均每次消费金额,平均消费周期
23,318,10
22,147,13
24,172,17
27,194,67
```

3.1.1 使用 csv.reader()读取 csv 文件

csv.reader()用来读取 csv 文件，其语法格式如下：

```
csv.reader(csvfile, dialect='excel', * * fmtparams)
```

作用：返回一个 reader 对象，这个对象是可以迭代的，有一个参数 line_num，表示当前行数。

参数说明如下。

csvfile：可以是文件(file)对象或者列表(list)对象。如果 csvfile 是文件对象，则要求该文件以 newline＝"的方式打开。

dialect：编码风格，默认为 excel 的风格，也就是用逗号分隔。dialect 方式也支持自定义，通过调用 register_dialect()方法来注册。

fmtparams：用于指定特定格式，以覆盖 dialect 中的格式。

【例 3-1】　使用 reader 读取 csv 文件(csv_reader.py)。

```
import csv
with open('consumer.csv',newline='') as csvfile:
    spamreader =csv.reader(csvfile)          #返回的是迭代类型
    for row in spamreader:
        print(', '.join(row))                #以逗号连接各字段
    csvfile.seek(0)                          #文件指针移动到文件开始
    for row in spamreader:
    print(row)
```

说明：newline 用来指定换行控制方式，可取 None、'\n'、'\r'或'\r\n'。读取时，不指定 newline，文件中的'\n'、'\r'或'\r\n'默认被转换为'\n'；写入时，不指定 newline，若换行符为各系统默认的换行符('\n'、'\r'或'\r\n')指定为 newline＝'\n'，则都替换为'\n'；若设定 newline＝''，不论读或者写，都表示不转换换行符。

将上述程序代码保存在 csv_reader.py 文件中，在 IDLE 中运行该程序文件，输出的结果如下。

```
客户年龄，平均每次消费金额，平均消费周期
23,  318,  10
22,  147,  13
24,  172,  17
27,  194,  67
['客户年龄', '平均每次消费金额', '平均消费周期']
['23',  '318',  '10']
['22',  '147',  '13']
['24',  '172',  '17']
['27',  '194',  '67']
```

3.1.2　使用 csv.writer()写入 csv 文件

csv.writer()用来写入 csv 文件，其语法格式如下：

```
csv.writer(csvfile, dialect='excel', * * fmtparams)
```

作用：返回一个 writer 对象。使用 writer 对象可将用户的数据写入该 writer 对象所对应的文件里。

参数说明如下。

csvfile：可以是文件(file)对象或者列表(list)对象。

dialect：编码风格，默认为 excel 的风格，也就是用逗号分隔。dialect 方式也支持自定义，通过调用 register_dialect()方法来注册。

fmtparams：用于指定特定格式，以覆盖 dialect 中的格式。

csv.writer()所生成的 csv.writer 文件对象支持以下写入 csv 文件的方法。

writerow(row)：写入一行数据。

writerows(rows)：写入多行数据。

【例 3-2】 使用 writer 写入 csv 文件(csv_writer.py)。

```
import csv
#写入的数据将覆盖 consumer.csv 文件
with open('consumer.csv', 'w', newline='') as csvfile:
    spamwriter =csv.writer(csvfile)              #生成 csv.writer 文件对象
    spamwriter.writerow(['55','555','55'])       #写入一行数据
    spamwriter.writerows([('35','355','35'),('18','188','18')])
with open('consumer.csv',newline='') as csvfile:   #重新打开文件
    spamreader =csv.reader(csvfile)
    for row in spamreader:                  #输出用 writer 对象的写入方法写入数据后的文件
        print(row)
```

csv_ writer.py 在 IDLE 中运行的结果如下：

```
['55', '555', '55']
['35', '355', '35']
['18', '188', '18']
```

【例 3-3】 使用 writer 向 csv 文件追加数据(csv_ writer_add.py)。

```
import csv
with open('consumer.csv', 'a+', newline='') as csvfile:
    spamwriter =csv.writer(csvfile)
    spamwriter.writerow(['55','555','55'])
    spamwriter.writerows([('35','355','35'),('18','188','18')])
with open('consumer.csv',newline='') as csvfile:   #重新打开文件
    spamreader =csv.reader(csvfile)
    for row in spamreader:                  #输出用 writer 对象的写入方法写入数据后的文件
        print(row)
```

csv_ writer_add.py 在 IDLE 中运行的结果如下：

```
['客户年龄', '平均每次消费金额', '平均消费周期']
['23', '318', '10']
```

```
['22', '147', '13']
['24', '172', '17']
['27', '194', '67']
['55', '555', '55']
['35', '355', '35']
['18', '188', '18']
```

3.1.3　使用 csv.DictReader()读取 csv 文件

把一个关系型数据库保存为 csv 文档，再用 Python 读取数据或写入新数据，在数据处理中是很常见的。很多情况下，读取 csv 数据时，往往先把 csv 文件中的数据读成字典的形式，即为读出的每条记录中的数据添加一个说明性的关键字，这样便于理解。为此，csv 库提供了能直接将 csv 文件读取为字典的函数 DictReader()，也有相应的将字典写入 csv 文件的函数 DictWriter()。

csv.DictReader()的语法格式如下：

```
csv.DictReader(csvfile, fieldnames=None, dialect='excel')
```

作用：DictReader()用于返回一个 DictReader 对象，该对象的操作方法与 reader 对象的操作方法类似，可以将读取的信息映射为字典，其关键字由可选参数 fieldnames 指定。

参数说明如下。

csvfile：可以是文件(file)对象或者列表(list)对象。

fieldnames：是一个序列，用于为输出的数据指定字典关键字，如果没有指定，则以第一行的各字段名作为字典关键字。

dialect：编码风格，默认为 excel 的风格，也就是用逗号“,”分隔。dialect 方式也支持自定义，通过调用 register_dialect()方法注册。

【例 3-4】　使用 csv.DictReader()读取 csv 文件(csv_DictReader.py)。

```
import csv
with open('consumer.csv', 'r') as csvfile:
    dict_reader =csv.DictReader(csvfile)
    for row in dict_reader:
        print(row)
```

csv_DictReader.py 在 IDLE 中运行的结果如下：

```
OrderedDict([('客户年龄', '23'), ('平均每次消费金额', '318'), ('平均消费周期', '10')])
OrderedDict([('客户年龄', '22'), ('平均每次消费金额', '147'), ('平均消费周期', '13')])
OrderedDict([('客户年龄', '24'), ('平均每次消费金额', '172'), ('平均消费周期', '17')])
OrderedDict([('客户年龄', '27'), ('平均每次消费金额', '194'), ('平均消费周期', '67')])
```

【例 3-5】　使用 csv.DictReader()读取 csv 文件，并为输出的数据指定新的字段名(csv_DictReader1.py)。

```
import csv
print_dict_name=['年龄','消费金额','消费频率']
with open('consumer.csv', 'r') as csvfile:
    dict_reader =csv.DictReader(csvfile,fieldnames=print_dict_name)
    for row in dict_reader:
        print(row)
print("\nconsumer.csv 文件内容: ")
with open('consumer.csv',newline='') as csvfile:   #重新打开文件
    spamreader =csv.reader(csvfile)
    for row in spamreader:
        print(row)
```

csv_DictReader1.py 在 IDLE 中运行的结果如下：

```
OrderedDict([('年龄', '客户年龄'), ('消费金额', '平均每次消费金额'), ('消费频率',
'平均消费周期')])
OrderedDict([('年龄', '23'), ('消费金额', '318'), ('消费频率', '10')])
OrderedDict([('年龄', '22'), ('消费金额', '147'), ('消费频率', '13')])
OrderedDict([('年龄', '24'), ('消费金额', '172'), ('消费频率', '17')])
OrderedDict([('年龄', '27'), ('消费金额', '194'), ('消费频率', '67')])
```

consumer.csv 文件内容：

```
['客户年龄', '平均每次消费金额', '平均消费周期']
['23', '318', '10']
['22', '147', '13']
['24', '172', '17']
['27', '194', '67']
```

从上述输出结果可以看出，consumer.csv 文件中第一行的数据并没发生变化。

3.1.4 使用 csv.DictWriter()写入 csv 文件

如果需要将字典形式的记录数据写入 csv 文件，则可以使用 csv.DictWriter()实现，其语法格式如下：

```
csv.DictWriter(csvfile, fieldnames, dialect='excel')
```

作用：DictWriter ()用于返回一个 DictWriter 对象，该对象的操作方法与 writer 对象的操作方法类似。参数 csvfile、fieldnames 和 dialect 的含义与 DictReader()函数中的参数类似。

【例 3-6】 使用 csv.DictWriter()写入 csv 文件(csv_DictWriter.py)。

```
import csv
dict_record =[{'客户年龄': 23, '平均每次消费金额': 318, '平均消费周期': 10}, {'客户
年龄': 22, '平均每次消费金额': 147, '平均消费周期': 13}]
keys =['客户年龄','平均每次消费金额','平均消费周期']
```

```
#在该程序文件所在目录下创建 consumer1.csv 文件
with open('consumer1.csv', 'w+',newline='') as csvfile:
    #文件头以列表的形式传入函数,列表的每个元素表示每一列的标识
dictwriter =csv.DictWriter(csvfile, fieldnames=keys)
    #若此时直接写入内容,会导致没有数据名,需先执行 writeheader()将文件头写入
    #writeheader()没有参数,因为在建立 dictwriter 时已设定了参数 fieldnames
    dictwriter.writeheader()
    for item in dict_record:
        dictwriter.writerow(item)
print("以 csv.DictReader()方式读取 consumer1.csv: ")
with open('consumer1.csv', 'r') as csvfile:
    reader =csv.DictReader(csvfile)
    for row in reader:
        print(row)
print("\n 以 csv.reader()方式读取 consumer1.csv: ")
with open('consumer1.csv',newline='') as csvfile:  #重新打开文件
    spamreader =csv.reader(csvfile)
    for row in spamreader:
        print(row)
```

csv_ DictWriter.py 在 IDLE 中运行的结果如下：

```
以 csv.DictReader()方式读取 consumer1.csv:
OrderedDict([('客户年龄', '23'), ('平均每次消费金额', '318'), ('平均消费周期', '10')])
OrderedDict([('客户年龄', '22'), ('平均每次消费金额', '147'), ('平均消费周期', '13')])
```

以 csv.reader()方式读取 consumer1.csv：

```
['客户年龄', '平均每次消费金额', '平均消费周期']
['23', '318', '10']
['22', '147', '13']
```

3.2　使用 python-docx 模块处理 Word 文档

Python 可以利用 python-docx 模块处理 Word 文档，处理方式是面向对象的。也就是说 python-docx 模块会把 Word 文档、文档中的段落、段落中的文本内容都看作对象，对对象进行处理就是对 Word 文档的内容进行处理。python-docx 模块的 3 种类型对象如下。

(1) Document 对象，表示一个 Word 文档。

(2) Paragraph 对象，表示 Word 文档中的一个段落。

(3) Paragraph 对象的 text 属性对象，表示段落中的文本内容。

通过“pip install python-docx”进行 python-docx 库的安装。通过“import docx”导入 python-docx 库。

3.2.1 创建与保存 Word 文档

使用 docx 的 Document()可以新建或打开 Word 文档并返回 Document 对象,若 Document()中指定了 Word 文档路径,则是打开文档;若没有指定路径,即 Document()中是空白的,则是新建的 Word 文档。

用 Document 对象的 save(path)方法保存文档,其中参数 path 是保存的文件路径。

【例 3-7】 创建与保存 Word 文档。

```
from docx import Document
test =Document()                                    #创建一个新的 Word 文档对象
test.save(r'D:\Python\testWord.docx')          #将新建的文档保存为 testWord.docx
```

运行上述程序代码,在 D 盘的 Python 文件夹下会创建一个名为 testWord.docx 的文档。

3.2.2 读取 Word 文档

先创建一个“D:\Python\word.docx”文档,并在其中输入如下内容。

书中有日月山河↵
有生活的琐碎苟且↵
也有梦里的诗与远方↵
有醇香的酒↵
有动人的故事↵
但凡尘世间的一切人事↵
都在书里化作一朵文字的花↵
你若用心品↵
芬香自然来↵

【例 3-8】 读取“D:\Python\word.docx”文档。

```
import docx
file=docx.Document(r"D:\Python\word.docx")          #创建文档对象
print("段落数:"+str(len(file.paragraphs)))          #段落数为 9,每个回车隔一段
#输出每一段的内容
for para in file.paragraphs:
    print(para.text)
#输出段落编号及段落内容
for i in range(len(file.paragraphs)):
    print("第"+str(i)+"段的内容是: "+file.paragraphs[i].text)
```

运行上述程序代码,得到的输出结果如图 3-1 所示。

3.2.3 写入 Word 文档

【例 3-9】 向 Word 文档写入文字。

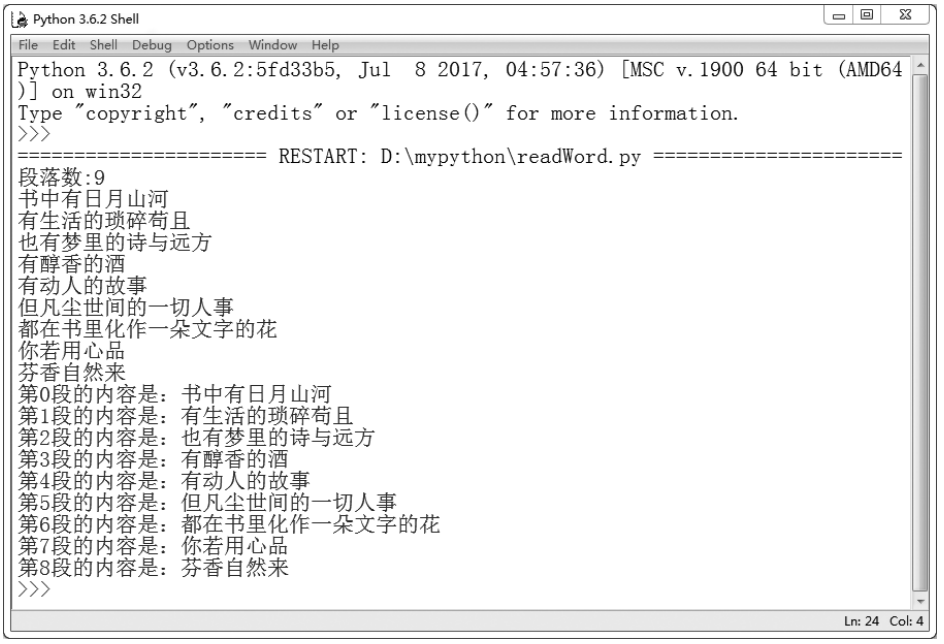


图 3-1　得到的输出结果

```
from docx import Document
from docx.shared import Pt
from docx.shared import Inches
from docx.oxml.ns import qn
document =Document()                                    #创建文档
document.add_heading('卜算子咏梅',0)                    #加入 0 级标题
document.add_heading('作者:[宋] 陆游',1)                #加入 1 级标题
document.add_heading('诗的正文',1)                      #加入 1 级标题
#插入段落,段落是 Word 文档最基本的对象之一
paragraph =document.add_paragraph('',style=None)
#给段落追加文字
for x in ['驿外断桥边,寂寞开无主。','已是黄昏独自愁,更著风和雨。','无意苦争春,一任群芳妒。','零落成泥碾作尘,只有香如故。']:
    paragraph.add_run(x+'\n',style="Heading 2 Char")
#添加图像,此处用到图像"咏梅.png"
document.add_picture(r'D:\Python\咏梅.png', width=Inches(5),height =
Inches(3))
document.add_heading('【简析】',1)                      #加入 1 级标题
paragraph1 =document.add_paragraph('',style=None) #插入段落
#给段落追加文字和设置字符样式
run1 =paragraph1.add_run('陆游一生酷爱梅花,写有大量歌咏梅花的诗,歌颂梅花傲霜雪,凌寒风,不畏强暴,不羡富贵的高贵品格。')
run1.font.size =Pt(18)                                  #设置字号
#设置中文字体
```

```
run1.font.name=u'隶书'
r =run1._element
r.rPr.rFonts.set(qn('w:eastAsia'), u'隶书')
document.add_heading('陆游生平',1)                    #加入 1 级标题
paragraph2 =document.add_paragraph('',style=None) #插入段落
#给段落追加文字
run2 =paragraph2.add_run(u'陆游(1125—1210 年),字务观,号放翁,汉族,越州山阴(今绍
兴)人,南宋文学家、史学家、爱国诗人。')
run2.font.size =Pt(16)#设置字号
run2.italic =True                                     #设置斜体
run2.bold =True                                       #设置粗体
document.save('writeWord.docx')                       #保存文件
```

运行上述程序代码,得到的 writeWord.docx 文档的内容如图 3-2 所示。

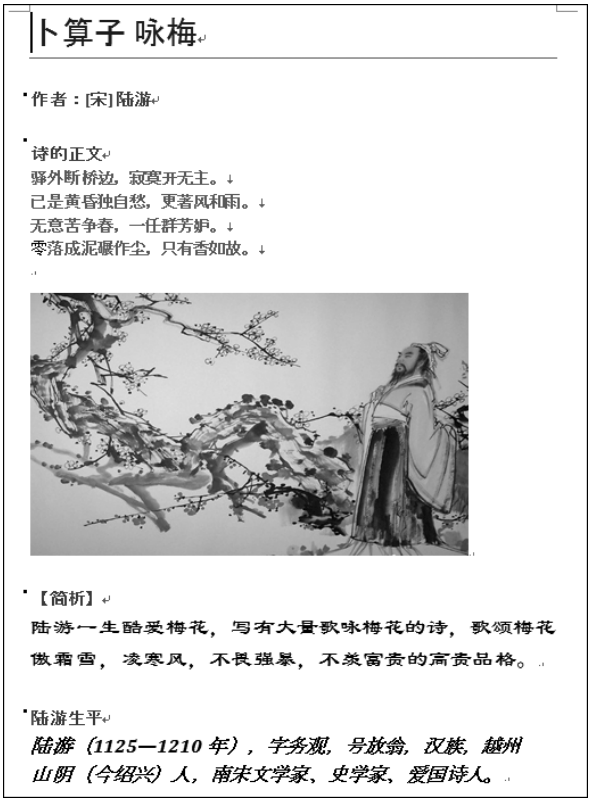


图 3-2 writeWord.docx 文档的内容

3.3 Excel 的文件读与写

一个以.xlsx 为扩展名的 Excel 文件打开后叫工作簿 workbook。每个工作簿可以包括多张表格 worksheet,正在操作的这张表格被认为是活跃的 active sheet。每张表格有

行和列，行号为 1、2、3…，列号为 A、B、C…。在某一个特定行和特定列的小格子叫单元格 cell。

Python 读写 Excel 的方式有很多，不同的模块在读写的语法格式上稍有区别，下面介绍用 xlrd 和 xlwt 库进行 Excel 文件的读与写。首先是安装第三方模块 xlrd 和 xlwt，输入命令“pip install xlrd”和“pip install xlwt”进行模块安装。

3.3.1　利用 xlrd 模块读 Excel 文件

1. 打开工作簿文件

```
>>>import xlrd
#打开工作簿文件 2016—2017 总成绩排名.xlsx,其部分内容如图 3-3 所示
>>>book =xlrd.open_workbook(r'G:\上课课件\Python 数学建模方法与实践\2016—2017 总成绩排名.xlsx')                    #创建 workbook 对象,即工作簿对象
```

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	课程/环节[类别][学分]		[0403103]操作系统	[1324102]软件工程	[1326120]基于项目的软件系统实训（ssh）	[1501101]毛泽东思想和中国特色社会主义理论体系概论2	[1901100]大学生就业指导	[0414101]计算机组成原理	[0721200]现代企业管理	[1303100]数据库原理	[1313100]计算机网络	[1324107]Java Web 框架技术	[1326125]软件开发综合实训（jsp）2	[1501100]毛泽东思想和中国特色社会主义理论体系概论1	德育学分1	德育学分2	总成绩	个人签名
2			[必修4.0]	[必修3.0]	[环节3.0]	[必修3.0]	[必修1.0]	[必修4.0]	[必修1.0]	[必修3.0]	[必修3.0]	[必修3.0]	[环节2.0]	[必修3.0]	[2.0]	[2.0]		
3	541413440244	武彦华	83	87	80	81	70	89	90	86	90	84	80	90	100	100	3202	
4	541413440132	刘文静	82	92	70	81	90	89	80	92	91	85	80	80	100	100	3187	
5	541413440126	李贤	87	91	80	83	90	88	80	87	88	87	80	80	88	95	3184	
6	5.41413E+11	贾云雷	83	85	80	82	80	82	90	81	90	85	80	90	100	100	3169	
7	541413440226	李雪杰	85	85	80	77	80	80	90	76	83	85	80	80	100	98	3084	
8	541413440250	颜梦林	79	89	70	80	80	84	90	86	83	87	80	90	80	85	3067	
9	541413440136	祁朋	76	86	70	75	80	88	80	90	90	85	90	70	82	98	3054	
10	541413440103	陈念	85	85	80	69	90	84	90	71	81	71	90	80	100	100	3047	
11	541413440123	李宁	77	90	70	81	80	83	80	79	83	77	80	80	98	100	3036	
12	541413440111	谷永慧	80	92	70	72	80	80	90	80	85	77	80	80	91	100	3020	
13	541413440222	李曼玉	80	83	70	85	80	71	90	87	80	76	80	80	100	100	3017	
14	541413440147	徐丹丹	81	90	80	66	90	87	80	76	88	87	60	70	93	91	3001	
15	541413440209	付雪华	78	80	80	91	80	69	90	69	76	82	80	80	100	100	2992	
16	541413440108	丰浩	77	82	80	83	90	80	70	75	83	66	80	80	98	100	2991	
17	541413440101	曹艳	81	88	80	70	80	76	80	77	84	73	80	80	93	100	2990	
18	541413440134	马越	78	80	60	64	90	87	80	87	91	78	80	80	90	100	2990	
19	541413440156	张孝帅	80	88	60	83	90	74	90	77	84	70	90	80	86	100	2974	
20	541413440214	何亚云	78	85	80	76	80	69	90	69	80	76	80	90	100	93	2969	

JAVA技术14，测试技术14，软件会计14，软件开发14，移动互联网14，JAVA技术15，测试技术15，软件会计15，软件开发15，移动互联网15

图 3-3　2016—2017 总成绩排名.xlsx 文件的部分内容

2. 获取表格 sheet

通过工作簿对象获取表格 sheet 的常用方法如下。

(1) 获取所有 sheet 名字：book.sheet_names()。

(2) 获取 sheet 数量：book.nsheets。

(3) 获取所有 sheet 对象：book.sheets()。

(4) 通过 sheet 名查找：book.sheet_by_name("test")。

(5) 通过索引查找：book.sheet_by_index(3)。

```
>>>book.sheet_names()                    #获取工作簿 book 的所有表格的名字
['Java 技术 14', '测试技术 14', '软件会计 14', '软件开发 14', '移动互联网 14', 'Java 技术 15', '测试技术 15', '软件会计 15', '软件开发 15', '移动互联网 15', 'Java 技术 16', '软件测试 16', '移动互联网 16', '软件开发 16']
```

```
>>>book.nsheets                                    #获取工作簿 book 的表格数量
14
>>>book.sheet_by_name("测试技术 14")               #通过 sheet 名查找表格
<xlrd.sheet.Sheet object at 0x0000000003224198>
>>>book.sheet_by_index(1)                         #通过索引查找表格
<xlrd.sheet.Sheet object at 0x0000000003224198>
```

3. 获取表格 sheet 的汇总数据

```
>>>sheet1 =book.sheets()[0]                       #获取第 1 个表格,创建表格对象 sheet1
```

表格对象的常用属性如下。

(1) 获取 sheet 名：sheet1.name。

(2) 获取总行数：sheet1.nrows。

(3) 获取总列数：sheet1.ncols。

```
>>>sheet1.name                                    #获取 sheet 名
'Java 技术 14'
>>>sheet1.nrows                                   #获取总行数
113
>>>sheet1.ncols                                   #获取总列数
22
```

4. 单元格批量读取

(1) 行操作。

```
• sheet1.row_values(3)          #获取表格对象 sheet1 第 4 行的所有内容,合并单元格
• sheet1.row(3)                                  #获取第 4 行各单元格值的类型和内容
• sheet1.row_types(3)                            #获取第 4 行各单元格数据的类型
>>>sheet1.row_values(3)                         #获取第 4 行内容
[2.0, '541413440132', '刘文静', 82.0, 92.0, 70.0, 81.0, 90.0, 89.0, 80.0, 92.0,
91.0, 85.0, 80.0, 80.0, 100.0, 100.0, 3187.0, '', '', '', '']
>>>sheet1.row(3)                                 #获取第 4 行各单元格值的类型和内容
[number:2.0, text:'541413440132', text:'刘文静', number:82.0, number:92.0,
number:70.0, number:81.0, number:90.0, number:89.0, number:80.0, number:92.0,
number:91.0, number:85.0, number:80.0, number:80.0, number:100.0, number:
100.0, number:3187.0, empty:'', empty:'', empty:'', empty:'']
>>>sheet1.row_types(3)                           #获取第 4 行各单元格数据的类型
array('B', [2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0])
```

数据类型说明如下。

- 空：0；
- 字符串：1；
- 数字：2；

- 日期：3；
- 布尔：4；
- error：5。

(2) 表操作。

- sheet1.row_values(0, 6, 10)　　#取第 1 行,第 6~ 10 列(不含第 10 列)
- sheet1.col_values(0, 0, 5)　　#取第 1 列,第 0~ 5 行(不含第 5 行)

```
>>>sheet1.row_values(0, 6, 10)
['[1501101]毛泽东思想和中国特色社会主义理论体系概论 2', '[1901100]大学生就业指导',
'[0414101]计算机组成原理', '[0721200]现代企业管理']
>>>sheet1.col_values(2, 0, 5)　　#取第 3 列,第 0~ 5 行(不含第 5 行)
['', '', '武彦华', '刘文静', '李贤']
```

5. 获取单元格值

(1) sheet1.cell_value(2，2)：取第 3 行、第 3 列的值。
(2) sheet1.cell(2，2).value：取第 3 行、第 3 列的值。
(3) sheet1.row(2)[2].value：取第 3 行、第 3 列的值。

```
>>>sheet1.cell_value(2, 2)
'武彦华'
>>>sheet1.cell(2, 2).value
'武彦华'
>>>sheet1.row(2)[2].value
'武彦华'
```

3.3.2　利用 xlwt 模块写 Excel 文件

目前,xlwt 支持 xls 格式的 Excel,还不支持 xlsx 格式的 Excel。
利用 xlwt 模块写 Excel 文件的过程如下。
(1) 新建 Excel 工作簿。
(2) 添加 sheet 工作表。
(3) 写入内容。
(4) 保存文件。

```
>>>import xlwt
#创建一个工作簿 workbook 并设置编码
>>>workbook =xlwt.Workbook(encoding ='utf-8')
#在工作簿中添加新的工作表,若不给名字,就是默认的名字,这里的名字是 chengji
>>>worksheet =workbook.add_sheet('chengji')
>>>worksheet.write(0,0,'Student ID')　　#向第一个单元格写入 Student ID
#上面的语句是按照独立的单个单元格写入的
#下面按照行写入,因为有的时候需要按照行写入
>>>row2=worksheet.row(1)　　#在第 2 行创建一个行对象
```

```
>>>row2.write(0,'541513440106')                    #向第 2 行第 1 列写入"541513440106"
#保存工作簿 workbook,会在当前目录生成一个 Students.xls 文件
>>>workbook.save('Students.xls')
```

3.4 pandas 读写不同格式的数据

从外部文件读写数据是数据分析处理的前提,是数据处理必不可少的部分。在读写数据时可以对数据做一定的处理,为接下来对数据做进一步分析打好基础。pandas 常用的读写不同格式文件的函数见表 3-1。

表 3-1 pandas 常用的读写不同格式文件的函数

读取函数	写 入 函 数	描 述
read_csv()	to_csv()	读写 csv 格式的数据
read_table()		读取普通分隔符分割的数据
read_excel()	to_excel()	读写 excel 格式的数据
read_json()	to_json()	读写 json 格式的数据
read_html()	to_html()	读写 html 格式的数据
read_sql()	to_sql()	读写数据库中的数据

下面主要介绍常见的 csv 文件的读写、txt 文件的读取、Excel 文件的读写。

3.4.1 读写 csv 文件

1. 读取 csv 文件中的数据

在介绍读写 csv 格式的文件之前,先在 Python 的工作目录下创建一个短小的 csv 文件,将其保存为 student.csv。文件内容如下。

```
Name,Math,Physics,Chemistry
WangLi,93,88,90
ZhangHua,97,86,92
LiMing,84,72,77
ZhouBin,97,94,80
```

这个文件以逗号作为分隔符,可使用 pandas 的 read_csv()函数读取它的内容,返回 DataFrame 格式的文件。

```
>>>csvframe =pd.read_csv('student.csv')      #从 csv 中读取数据
>>>type(csvframe)
<class 'pandas.core.frame.DataFrame'>
>>>csvframe
```

```
      Name  Math  Physics  Chemistry
0    WangLi    93       88         90
1  ZhangHua    97       86         92
2    LiMing    84       72         77
3   ZhouBin    97       94         80
```

csv 文件中的数据为列表数据，位于不同列的元素用逗号隔开，csv 文件被视作文本文件，也可以使用 pandas 的 read_table()函数读取，但需要指定分隔符。

```
>>>pd.read_table('student.csv',sep=',')
      Name  Math  Physics  Chemistry
0    WangLi    93       88         90
1  ZhangHua    97       86         92
2    LiMing    84       72         77
3   ZhouBin    97       94         80
```

pd.read_csv()函数的语法格式如下。

```
pd.read_csv(filepath_or_buffer, sep=',', header='infer', names=None, index_
col=None, usecols=None)
```

作用：读取 csv(逗号分割)文件到 DataFrame 对象。

参数说明如下。

filepath_or_buffer：拟要读取的文件的路径，可以是本地文件，也可以是 http、ftp、s3 文件。

sep：其类型是 str，默认为','，用来指定分隔符。如果不指定 sep 参数，则会尝试使用逗号分隔。csv 文件中的分隔符一般为逗号分隔符。

header：其类型中 int 或 int 型列表指定第几行作为列名，默认为 0(第 1 行)。如果第 1 行不是列名，是内容，可以设置 header＝None，以便不把第 1 行当作列名。header 参数可以是一个列表，例如[0, 2]，这个列表表示将文件中的这些行作为列标题(这样，每一列将有多个标题)，介于中间的行将被忽略(例如本例中的第 2 行；本例数据中，行号为 0、2 的行将被作为多级标题出现，行号为 1 的行将被丢弃，dataframe 的数据从行号为 3 的行开始)。

names：用于结果的列名列表，对各列重命名，即添加表头。如果数据有表头，但想用新的表头，可以设置 header＝0、names＝['a','b']实现新表头的定制。

index_col ：其类型为 int 或序列类型，默认为 None，用作行索引的列编号或者列名，可使用 index_col＝[0, 1]指定文件中的第 1 和 2 列为行索引。

usecols：其类型是列表，默认 None，返回一个数据子集，即选取某几列，不读取整个文件的内容，有助于加快速度和降低内存，如 usecols＝[1,2]或 usercols＝['a','b']。为＃指定 csv 文件中的行号为 0、2 的行为列标题

```
>>>csvframe =pd.read_csv('student.csv',header=[0,2])
>>>csvframe
```

```
        Name  Math  Physics  Chemistry
    ZhangHua    97       86         92
0     LiMing    84       72         77
1    ZhouBin    97       94         80
>>>pd.read_csv('student.csv',usecols=[1,2])     #读取第 2 列和第 3 列
    Math  Physics
0    93       88
1    97       86
2    84       72
3    97       94
#设置 header=0, names=['name','maths','physical','chemistry']实现表头定制
>>> pd.read_csv('student.csv',header=0,names=['name','maths','physical',
'chemistry'])
        name  maths  physical  chemistry
0     WangLi     93        88         90
1   ZhangHua     97        86         92
2     LiMing     84        72         77
3    ZhouBin     97        94         80
>>>pd.read_csv('student.csv',index_col=[0,1])  #指定前两列作为行索引
                Physics  Chemistry
Name       Math
WangLi      93       88         90
ZhangHua    97       86         92
LiMing      84       72         77
ZhouBin     97       94         80
```

2. 向 csv 文件写入数据

把 DataFrame 对象中的数据写入 csv 文件,要用到 to_csv()函数,其语法格式如下。

```
DataFrame.to_csv(path_or_buf=None, sep=',', na_rep='',columns=None, header=
True, index=True)
```

作用:以逗号为分隔符将 DataFrame 对象中的数据写入 csv 文件中。

参数说明如下。

filepath_or_buffer:拟要写入的文件的路径或对象。

sep:默认字符为',',用来指定输出文件的字段分隔符。

na_rep:字符串,默认为'',缺失数据表示,即把空字段替换为 na_rep 所指定的值。

columns:指定要写入文件的列。

header:是否保存列名,默认为 True,保存。如果给定字符串列表,则将其作为列名的别名。

index:是否保存行索引,默认为 True,保存。

```
>>>import pandas as pd
```

```
>>>date_range =pd.date_range(start="20180801", periods=4)
>>>df=pd.DataFrame({'book':[12,13,15,22],'box':[3,8,13,18],'pen': [5,7,12,
15]},index=date_range)
>>>df
            book  box  pen
2018-08-01   12    3    5
2018-08-02   13    8    7
2018-08-03   15   13   12
2018-08-04   22   18   15
>>>df.to_csv('bbp.csv')            #把 df 中的数据写入默认工作目录下的 bbp.csv 文件
```

生成的 bbp.csv 文件的内容如下：

```
,book,box,pen
2018-08-01,12,3,5
2018-08-02,13,8,7
2018-08-03,15,13,12
2018-08-04,22,18,15
```

由上述例子可知，把 df 中的数据写入文件时，行索引和列名称连同数据一起写入，使用 index 和 header 选项，把它们的值设置为 False，可取消这一默认行为。

```
>>>df.to_csv('bbp1.csv',index=False,header=False)
```

生成的 bbp1.csv 文件的内容如下：

```
12,3,5
13,8,7
15,13,12
22,18,15
#写入时,为行索引指定列标签名
>>>df.to_csv("bbp2.csv",index_label="index_label")
```

bbp2.csv 文件内容：

```
index_label,book,box,pen
2018-08-01,12,3,5
2018-08-02,13,8,7
2018-08-03,15,13,12
2018-08-04,22,18,15
```

3.4.2　读取 txt 文件

txt 文件是一种常见的文本文件，可以把一些数据保存在 txt 文件里，用的时候再读取出来。pandas 的函数 read_table()可读取 txt 文件。

pd.read_table 函数的语法格式如下。

```
pandas.read_table(filepath_or_buffer, sep='\t', header='infer', names=None,
```

```
index_col=None, skiprows=None, nrows=None, delim_whitespace=False)
```

作用：读取以'\t'分隔的文件，返回 DataFrame 对象。

参数说明如下。

sep：其类型是 str，用来指定分隔符，默认为制表符，可以是正则表达式。

index_col：指定行索引。

skiprows：用来指定读取时要排除的行。

nrows：从文件中要读取的行数。

delim_whitespace：delim_whitespace=True 表示用空格分隔每行。

首先在工作目录下创建名为 1.txt 的文本文件，其内容如下：

```
C  Python  Java
1     4       5
3     3       4
4     2       3
2     1       1
>>>pd.read_table('1.txt')                          #读取 1.txt 文本文件
   C  Python  Java
0  1     4       5
1  3     3       4
2  4     2       3
3  2     1       1
```

从上面的读取结果可以看出，文件读取后所显示的数据不整齐。读取文本文件时可以通过用 sep 参数指定正则表达式来匹配空格或制表符，即用通配符“\s*”，其中“\s”匹配空格或制表符，星号“*”表示这些字符可能有多个。

```
>>>pd.read_table('1.txt',sep='\s*')
    C  Python  Java
0   1     4       5
1   3     3       4
2   4     2       3
3   2     1       1
```

如上所示，得到了整齐的 DataFrame 对象，所有元素均处在和列索引对应的位置上。

当文件较大时，可以一次读取文件的一部分，这时须明确指明要读取的行号，要用到 nrows 和 skiprows 参数选项，skiprows 指定读取时要排除的行，nrows 指定从起始行开始向后读取多少行。

```
>>>pd.read_table('1.txt',sep='\s*',skiprows=[1],nrows=2)
   C  Python  Java
0  3     3       4
1  4     2       3
```

在接下来这个例子中，2.txt 文件中数字和字母杂糅在一起，需要从中抽取数字部分。2.txt 文件的内容如下：

```
0BEGIN11NEXT22A32
1BEGIN12NEXT23A33
2BEGIN13NEXT23A34
```

2.txt 文件显然没有表头，用 read_table 读取时需要将 header 选项设置为 None。

```
>>>pd.read_table('2.txt',sep='\D*',header=None)
   0   1   2   3
0  0  11  22  32
1  1  12  23  33
2  2  13  23  34
```

3.4.3　读写 Excel 文件

在数据分析处理中，用 Excel 文件存放列表形式的数据也非常常见，为此 pandas 提供了 read_excel()函数来读取 Excel 文件，用 to_excel()函数向 Excel 文件写入数据。

1. 读取 Excel 文件中的数据

pandas.read_excel()函数的语法格式如下。

```
pandas.read_excel(io,sheet_name=0,header=0,names=None,index_col=None,
usecols=None,skiprows=None,skip_footer=0)
```

作用：读取 Excel 文件中的数据，返回一个 DataFrame 对象。

参数说明如下。

io：Excel 文件路径，是一个字符串。

sheet_name：返回指定的 sheet(表)，如果将 sheet_name 指定为 None，则返回全表；如果需要返回多个表，可以将 sheet_name 指定为一个列表，例如['sheet1', 'sheet2']；可以根据 sheet 的名字字符串或索引指定所要选取的 sheet，例如[0,1, 'Sheet5']将返回第一、第二和第五个表；默认返回第一个表。

header：指定作为列名的行，默认为 0，即取第 1 行，数据为列名行以下的数据；若数据不含列名，则设定 header = None。

names：指定所生成的 DataFrame 对象的列的名字，传入一个 list 数据。

index_col：指定某列为行索引。

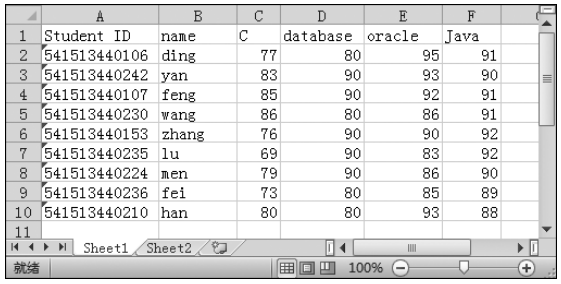
usecols：通过名字或索引值读取指定的列。

skiprows：省略指定行数的数据。

skip_footer：int，默认值为 0，读取数据时省略最后的 skip_footer 行。

首先在工作目录下创建名为 chengji.xlsx 的 Excel 文件，Sheet1 的内容见表 3-2。

表 3-2 Sheet1 的内容

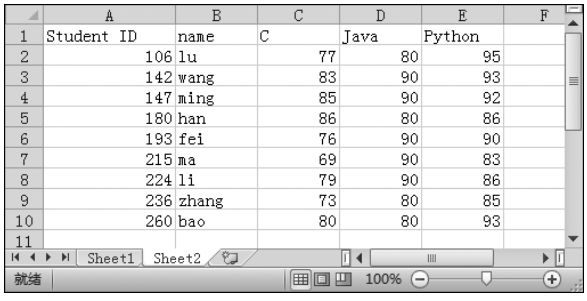


	A	B	C	D	E	F
1	Student ID	name	C	database	oracle	Java
2	541513440106	ding	77	80	95	91
3	541513440242	yan	83	90	93	90
4	541513440107	feng	85	90	92	91
5	541513440230	wang	86	80	86	91
6	541513440153	zhang	76	90	90	92
7	541513440235	lu	69	90	83	92
8	541513440224	men	79	90	86	90
9	541513440236	fei	73	80	85	89
10	541513440210	han	80	80	93	88
11						

Sheet1 Sheet2 就绪 100%

Sheet2 的内容见表 3-3。

表 3-3 Sheet2 的内容



	A	B	C	D	E	F
1	Student ID	name	C	Java	Python	
2	106	lu	77	80	95	
3	142	wang	83	90	93	
4	147	ming	85	90	92	
5	180	han	86	80	86	
6	193	fei	76	90	90	
7	215	ma	69	90	83	
8	224	li	79	90	86	
9	236	zhang	73	80	85	
10	260	bao	80	80	93	
11						

Sheet1 Sheet2 就绪 100%

接下来通过 pandas 的 read_excel()方法读取 chengji.xlsx 文件。

```
>>>pd.read_excel('chengji.xlsx')
     Student ID    name   C   database   oracle   Java
0   541513440106   ding  77      80         95      91
1   541513440242    yan  83      90         93      90
2   541513440107   feng  85      90         92      91
3   541513440230   wang  86      80         86      91
4   541513440153  zhang  76      90         90      92
5   541513440235     lu  69      90         83      92
6   541513440224    men  79      90         86      90
7   541513440236    fei  73      80         85      89
8   541513440210    han  80      80         93      88
#将 chengji.xlsx 的列名作为所生成的 DataFrame 对象的第 1 行数据,并重新生成索引
>>>pd.read_excel('chengji.xlsx',header=None)
              0     1    2       3         4
0     Student ID   name   C   database   oracle
1   541513440106   ding  77      80         95
2   541513440242    yan  83      90         93
3   541513440107   feng  85      90         92
4   541513440230   wang  86      80         86
5   541513440153  zhang  76      90         90
```

```
6  541513440235         lu    69    90
7  541513440224  men    79    90    86
8  541513440236  fei    73    80    85
9  541513440210  han    80    80    93
#skiprows 指定读取数据时要忽略的行,这里忽略第 1、2、3 行
>>>pd.read_excel('chengji.xlsx',skiprows =[1,2,3])
      Student ID   name   C   database  oracle  Java
0   541513440230   wang  86      80        86      91
1   541513440153  zhang  76      90        90      92
2   541513440235     lu   69      90        83      92
3   541513440224    men   79      90        86      90
4   541513440236    fei   73      80        85      89
5   541513440210    han   80      80        93      88
#skip_footer=4,表示读取数据时忽略最后 4 行
>>>pd.read_excel('chengji.xlsx',skip_footer=4)
                      Student ID  name   C database oracle Java
    0 541513440106          ding 77   80             95            91
    1 541513440242           yan 83   90             93            90
    2 541513440107          feng 85   90             92            91
    3 541513440230          wang 86   80             86            91
4 541513440153 zhang 76          90   90             92
#index_col="Student ID"表示指定 Student ID 为行索引
>>>pd.read_excel('chengji.xlsx',skip_footer=4,index_col="Student ID")
               name   C   database  oracle  Java
 Student ID
541513440106   ding  77      80        95      91
541513440242    yan  83      90        93      90
541513440107   feng  85      90        92      91
541513440230   wang  86      80        86      91
541513440153  zhang  76      90        90      92
#names 参数用来重新命名列名称
>>>pd.read_excel('chengji.xlsx',skip_footer=5,names=["a","b","c","d","e",
"f"])
               a      b   c   d   e   f
0   541513440106  ding  77  80  95  91
1   541513440242   yan  83  90  93  90
2   541513440107  feng  85  90  92  91
3   541513440230  wang  86  80  86  91
#sheet_name=[0,1]表示同时读取 Sheet1 和 Sheet2
>>>pd.read_excel('chengji.xlsx',skip_footer=5,sheet_name=[0,1])
```