第5章 文件读写

文件是计算机存储数据的重要形式，用文件组织和表达数据更加有效和灵活。文件有不同的编码和存储形式，如文本文件、图像文件、音频和视频文件、数据库文件、特定格式文件等。每个文件都有各自的文件名和属性，对文件进行操作是 Python 的重要功能。

5.1 TXT 文件读写

5.1.1 读取文件全部内容

文本文件的读写通常有 3 个基本步骤：打开文件、读写文件和关闭文件。

1. 文件打开模式

对文件访问之前，必须先打开文件，并指定被打开的文件做什么操作。在 Python 中，通过标准内置函数 open()打开一个文件并获得一个文件对象。语法格式如下。

```
1  文件句柄 = open('文件路径和名称', '操作模式')          # 文件打开语法 1
2  with open('文件路径和名称', '操作模式') as 文件句柄:    # 文件打开语法 2
```

(1) 文件读写是一个非常复杂的过程，它涉及设备(如硬盘)、通道(数据传输方式)、路径(文件存放位置)、进程(读写操作)、文件缓存(文件在内存中的存放)等复杂问题。文件句柄是简化文件读写的变量，它隐藏了设备、通道、路径、进程、缓存等复杂操作，帮助程序员关注正在处理的文件。**文件句柄用变量名表示**，通过 open()函数把它和文件连接，文件读写完成后，关闭文件就会释放文件句柄占用的资源。

(2) 文件打开或创建成功后，文件句柄就代表了打开的文件对象，这简化了程序语句。我们可以得到这个文件的各种信息(属性)。

(3) 文件可以采用相对路径，如"江南春.txt"；也可以采用绝对路径(包含完整目录的文件名)，如"d:\\test\\05\\江南春.txt"。路径前面有 r 参数是保持路径中字符串原始值的意思，也就是说不对其中的符号进行转义。路径采用\表示时，可以加上 r 参数，如 r"d:\test\05\江南春.txt"。注意，路径采用\\表示时，不要加 r 参数。

(4) 文件读取时，会同时读取行末尾包含的回车换行符号(\n)。

(5) 操作模式用于控制文件打开方式，文件打开的操作模式及参数说明如表 5-1 所示。

表 5-1 文件打开的操作模式及参数说明

操作模式	参数说明
r	仅读，待打开的文件必须存在；文件不存在时会返回异常 FileNotFoundError
w	仅写，若文件已存在，内容将先被清空；若文件不存在则创建文件，不可读

续表

操作模式	参数说明
a	仅写，若文件已存在，则在文件最后追加新内容；若文件不存在则创建文件
r+	可读，可写，可追加；待打开的文件必须存在（“+”参数说明允许读和写）
a+	读写，若文件已存在，则内容不会被清空
w+	读写，若文件已存在，则内容将先被清空
rb	仅读，读二进制文件；待打开的文件必须存在（“b”参数说明读写二进制文件）
wb	仅写，写二进制文件；若文件已存在，则内容将先被清空
ab	仅写，写二进制文件；若文件已存在，则内容不会被清空

【例 5-1】 打开当前目录下 test.txt 文件，对它进行覆盖写操作。

```
f = open("test.txt", "w")                    # 以写模式打开文件，f = 文件句柄，相对路径
```

2. 关闭文件

文件使用结束后一定要关闭，这样才能保存文件内容，释放文件占用内存。

```
文件句柄.close()                             # 文件关闭的语法格式
```

3. 文件读取函数

Python 提供了 read()、readlines()和 readline()三个文件读取函数。这三种函数都会把文件每行末尾的'\n'(换行符)也读进来，可以用 splitlines()等函数删除换行符。

(1) read()函数一次读取文件的全部内容，返回值存放在一个大字符串中。它的优点是方便、简单、速度最快；它的缺点是文件过大时，占用内存很大。

【例 5-2】 用 read()函数读取“琴诗.txt”文件中全部内容。

```
>>> f = open("d:\\test\\05\\琴诗.txt", "r")                    # 打开文件，f 为文件句柄
>>> s = f.read( )                                              # 读文件全部内容
>>> s                                                          # 输出内容为字符串
'[宋] 苏轼《琴诗》\n若言琴上有琴声，放在匣中何不鸣？\n          # \n 为换行符
若言声在指头上，何不于君指上听？\n'
```

【例 5-3】 读取“琴诗.txt”文件中全部内容，并且删除文件中的回车符。

```
>>> f = open("d:\\test\\05\\琴诗.txt", "r")                    # 打开文件，f 为文件句柄
>>> s = f.read( )                                              # 读文件全部内容
>>> s = s.splitlines()                                         # 删除字符串中的换行符
>>> s                                                          # 输出内容已转换为列表
['[宋] 苏轼《琴诗》', '若言琴上有琴声，放在匣中何不鸣？', '若言   # 输出已经删除换行符
声在指头上，何不于君指上听？']
```

(2) readlines()函数一次读取全部文件，它自动将文件内容分解成一个大的列表，该列表可以用 for 循环进行处理。它的缺点是读取大文件时会比较占内存。

【例 5-4】 用 readlines()函数读取“琴诗.txt”文件中全部内容。

```
>>> f = open("d:\\test\\05\\琴诗.txt", "r")                    # 打开文件,f 为文件句柄
>>> f.readlines( )                                             # 读文件全部内容
['[宋] 苏轼《琴诗》\n', '若言琴上有琴声,放在匣中何不                # 输出内容为列表
鸣? \n', '若言声在指头上,何不于君指上听? \n']
```

(3) readline()函数每次只读取一行,返回值是字符串。它的读取速度比 readlines()慢得多,只有在没有足够内存一次读取整个文件时,才应该使用 readline()函数。

【例 5-5】 用 readline()函数读取“琴诗.txt”文件中一行内容。

```
>>> f = open("d:\\test\\05\\琴诗.txt", "r")                    # 打开文件,f 为文件句柄
>>> f.readline( )                                              # 读文件一行内容
'[宋] 苏轼《琴诗》\n'                                          # 输出一行字符串
>>> f.close( )                                                 # 关闭文件
```

5.1.2 文件遍历

文件遍历就是读取文件中每个数据,然后对遍历结果进行某种操作。如输出遍历结果;将遍历结果赋值到某个列表;对遍历结果进行统计(如字符数,段落数等);对遍历结果进行排序;将遍历结果添加到其他文件等操作。遍历是一个非常重要的操作。

1. 文件遍历方法

用 open()语句或 with open()语句都可以实现文件遍历,它们的差别如下。

(1) 用 open()语句读取文件后,需要用 close()关闭文件;用 with open()语句读取文件结束后,语句会自动关闭文件,不需要再写 close()语句。

(2) 用 open()语句读取文件如果发生异常,语句没有任何处理功能;用 with open()语句会处理好上下文产生的异常。

(3) 用 open()语句一次只能读取一个文件;用 with open()语句一次可以读取多个文件。

【例 5-6】 文件遍历方法 1:逐行读取文件内容。

```
#E0506.py                                    #【文件遍历 1】
with open('登鹳雀楼.txt') as file_obj:       # 打开文件(相对路径,当前目录在 d:\test\05)
    content = file_obj.read()                # 循环读取文件中每一行
    print(content)                           # 输出行内容
>>>…(输出略)                                 # 程序运行结果
```

【例 5-7】 文件遍历方法 2:一次全部读入文件内容到列表,再逐行遍历列表。

```
#E0507.py                                    #【文件遍历 2】
f = open('d:\\test\\05\\登鹳雀楼.txt', 'r')   # 打开文件(绝对路径)
L = f.readline()                             # 将文件内容一次全部读入列表 L
while L:                                     # 循环输出列表 L 中的内容
    print(L, end = '')                       # 参数 end = ''为不换行输出
    L = f.readline()                         # 读取列表中行的内容
f.close()                                    # 关闭文件
>>>…(输出略)                                 # 程序运行结果
```

【例 5-8】 文件遍历方法 3：循环读取文件内容到列表，再输出列表。

```
1  #E0508.py                                  #【文件遍历 3】
2  for myList in open("登鹳雀楼.txt"):        # 打开文件，循环输出列表内容（相对路径）
3      print(myList)                          # 输出列表内容
   >>>…（输出略）                             # 程序运行结果
```

【例 5-9】 文件遍历方法 4：文件内容一次全部读入列表，再逐行遍历列表内容。

```
1  #E0509.py                                  #【文件遍历 4】
2  f = open("d:\\test\\05\\登鹳雀楼.txt", "r")  # 打开文件，前面 r = 路径，后面"r" = 读操作
3  L = f.readlines()                          # 读取文件全部内容到列表
4  for myList in L:                           # 循环读取列表中的行
5      print(myList, end = '')                # 输出行内容（没有 end = ''参数会多输出一些空行）
6  f.close()                                  # 关闭文件
   >>>…（输出略）                             # 程序运行结果
```

【例 5-10】 文件遍历方法 5：一次读取两个文件全部内容到列表，再输出列表。

```
1  #E0510.py                                                    #【文件遍历 5】
2  with open("金庸名言 1.txt", 'r') as f1, open("金庸名言 2.txt", 'r') as f2:   # 读入两个文件
3      print(f1.read())                                         # 打印第一个文件
4      print(f2.read())                                         # 打印第两个文件
   >>>                                                          # 程序运行结果
   侠之大者，为国为民。
   ——金庸《射雕英雄传》
   只要有人的地方就有恩怨，有恩怨就会有江湖，人就是江湖。
   ——金庸《笑傲江湖》
```

2. 用 if 语句判断文件是否结束

【例 5-11】 用 if 语句判断文件是否结束。

```
1  #E0511.py                                  #【判断文件是否结束】
2  filename = "d:\\test\\05\\登鹳雀楼.txt"     # 路径变量赋值
3  file = open(filename, 'r')                 # 读取文件全部内容，file 为文件句柄，r 为读取模式
4  done = 0                                   # 初始化循环终止变量
5  while not done:                            # 循环读取文件行
6      Line = file.readline()                 # 读取文件行
7      if (Line != ''):                       # 如果 Line 不等于空，则文件没有结束
8          print(Line)                        # 输出行内容
9      else:
10         done = 1                           # 退出循环标识
11 file.close()                               # 关闭文件
   >>>…（输出略）                             # 程序运行结果
```

程序第 7 行，if (Line !=''）为判断第 6 行 readline()语句读到的内容是否为空，行内容为空意味着文件结束。如果 readline()语句读到一个空行，也会判断为文件结束吗？事实上空行并不会返回空值，因为空行的末尾至少还有一个回车符(\n)。所以，即使文件中包

含空行，读入行的内容也不为空，这说明 if (Line !='')语句判断是正确的。

5.1.3 读取文件指定行

1. 文件指针

文件打开后，对文件的读写有一个读取指针(元素索引号)，从文件中读入内容时，读取指针不断向文件尾部移动，直到文件结束位置。Python 提供了两个读写文件指针位置相关的函数 tell()和 seek()。它们的语法格式如下。

```
1  文件对象名.tell()                          # 获取当前文件操作指针的位置
2  文件对象名.seek(偏移量[, 偏移位置])          # 改变当前文件操作指针的位置
```

参数“偏移量”表示要移动的字节数，若为正数则向文件尾部移动，若负数则向文件头部移动。

参数“偏移位置”值为 0 表示文件开头，值为 1 表示当前位置，值为 2 表示文件结尾。

【例 5-12】 获取文件指针位置。

```
1  >>> f = open('d:\\test\\05\\琴诗.txt')        # 打开文件
2  >>> f.tell()                                 # 获得当前文件读取指针
   0
3  >>> f.seek(10)                               # 将文件指针向后移动 10 字节
   10
4  >>> f.seek(0, 2)                             # 将文件读取指针移动到文件尾部
   85                                           # 文件大小为 85 字节
```

注意，tell()方法中，size 代表字节数，这里数字、英文字符、回车符(包括空行回车符)占 1 字节，而汉字和中文标点符号占 2 字节。

2. 读取文件指定行

【例 5-13】 文件“成绩 utf8.txt”内容如下所示。

```
学号,姓名,班级,古文,诗词,平均
1,宝玉,01,70,85,0
2,黛玉,01,85,90,0
3,晴雯,02,40,65,0
4,袭人,02,20,60,0
```

用 readlines()函数读出文件到列表，从列表对指定数据进行切片。

```
1  #E0513.py                                          #【读文件指定行】
2  fileName = 'd:\\test\\05\\成绩 utf8.txt'           # 路径赋值(绝对路径)
3  with open(fileName, 'r', encoding = 'UTF - 8') as f:   # 打开文件(文件编码为 UTF - 8)
4      lines = f.readlines()                          # 读取全部文件到列表 lines
5      for i in lines[1:3]:                           # 列表切片，循环读取 1、2 行
6          print(i.strip('\n'))                       # 函数 strip()删除字符串两端的空格
   >>>                                                # 程序运行结果
   1,宝玉,01,70,85,0
   2,黛玉,01,85,90,0
```

程序第 5 行，for i in lines[1:3]语句中，[1:3]为列表切片索引号位置，其中，0 行是表头不读取，列表第 3 行不包含，因此语句功能为循环读取列表 1、2 行。

程序扩展：如果希望对文件数据隔一行读一行时，只需要修改程序第 5 行中列表索引号即可，如“for i in lines[1:4:2]:”，该语句表示读取列表 1～4 行，步长为 2(即读 1、3 行)。

【例 5-14】 用标准模块 linecache 中的 getline()函数读出文件指定行。

```
1  #E0514.py                                          #【读文件指定行】
2  import linecache                                   # 导入标准模块 - 行读取
3  s = linecache.getline('d:\\test\\05\\成绩.txt', 1)  # 读取文件第 1 行
4  print('第 1 行:', s)
   >>>第 1 行: 学号,姓名,班级,古文,诗词,平均            # 程序运行结果
```

【例 5-15】 统计文件的行数。

```
1  #E0515.py                                          #【统计文件行数】
2  filepath = 'd:\\test\\05\\成绩.txt'                 # 文件路径赋值
3  count = 0                                          # 计数器初始化
4  for index, line in enumerate(open(filepath,'rb')):  # 循环读取文件行
5      count += 1                                     # 行数累加
6  print('文件行数为:', count)
   >>>文件行数为: 5                                    # 程序运行结果
```

程序第 4 行，函数 enumerate()具有枚举功能，它可以遍历一个可迭代对象(如列表、字符串)，并且将其组成一个索引序列，利用它可以同时获得索引和值。

5.1.4 向文件写入数据

1. 覆盖写入文件

Python 提供了 2 个文件写入函数，语法格式如下。

```
1  文件句柄.write(["单字符串"])          # 语法 1:向文件写入一个字符串或字节流
2  文件句柄.writelines(行字符串)         # 语法 2:将多个元素的字符串写入文件
```

函数 write()是将字符串写入一个打开的文件。注意，首先，这里的字符串可以是二进制数据，而不仅仅是文字；其次，write()函数不会在字符串结尾添加换行符(\n)。

【例 5-16】 用 write()函数将字符串内容写入名为“诗歌 1.txt”的文件。

```
1  #E0516.py                                          #【写入新文件 1】
2  str1 = '白日依山尽,黄河入海流。\n 欲穷千里目,更上一层楼。\n'  # \n 为换行符
3  f = open('d:\\test\\05\\诗歌 1.txt', 'w')            # 以写模式打开文件
4  f.write(str1)                                      # 字符串内容写入文件
5  f.close()                                          # 关闭文件
6  print('写入成功,保存在 d:\\test\\05\\诗歌 1.txt')
   >>>写入成功,保存在 d:\test\05\诗歌 1.txt             # 程序运行结果
```

程序第 2 行，由于 write()函数不会在字符串结尾自动添加换行符(\n)，因此字符串中必须根据需要人为加入换行符。

程序第 3 行，如果这个文件已经存在，那么源文件内容将会被新内容覆盖。

【例 5-17】 用 writelines()函数将内容写入名为“登鹳雀楼 out.txt”的文件。

```
#E0517.py                                                    #【写入文件 2】
s = ['白日依山尽,', '黄河入海流。', '欲穷千里目,', '更上一层楼。']    # 定义列表
f = open('d:\\test\\05\\登鹳雀楼 out.txt', 'w')               # 写模式打开文件
f.writelines(s)                                              # 列表写入文件
f.close( )                                                   # 关闭文件
print('列表写入成功。')
>>>列表写入成功。                                             # 程序运行结果
```

2. 追加写入文件

【例 5-18】 将字符串内容追加写入“诗歌 1.txt”文件的结尾。

```
#E0518.py                                                    #【写入文件 3】
f = open('d:\\test\\05\\诗歌 1.txt', 'a + ')                  # 以追加模式打开已存在的文件
f.write('——王之涣《登鹳雀楼》\n')                             # 字符串内容追加写入文件末尾
f.close()                                                    # 关闭文件
print('追加写入成功。')
>>>追加写入成功。                                             # 程序运行结果
```

5.1.5 文件属性检查

Python 提供了许多常用的文件和目录操作方法，用户利用它们可以方便地重命名、删除文件和创建、删除、更改目录等。这主要是通过 Python 内置 os 模块来进行。

在进行文件操作前，先检查文件或目录是否存在，不然会使程序出错。有三种检查文件或目录是否存在的模块：os 模块、try 语句、pathlib 模块。

1. 使用 os.access()函数检查文件属性

使用 os.access()函数可以判断文件的读写操作。语法格式如下。

```
os.access(路径，操作模式)                                    # 文件属性检查的语法格式
```

函数返回值为 True 或者 False。

【例 5-19】 利用 access()函数检查文件各种读写属性。

```
>>> import os                                                # 导入标准模块 - 系统
>>> os.access("d:\\test\\05\\登鹳雀楼.txt", os.F_OK)         # 参数 os.F_OK 检查文件路径是否存在
True
>>> os.access("d:\\test\\05\\登鹳雀楼.txt", os.R_OK)         # 参数 os.R_OK 检查文件是否可读
True
>>> os.access("d:\\test\\05\\登鹳雀楼.txt", os.W_OK)         # 参数 os.W_OK 检查文件是否可写
True
>>> os.access("d:\\test\\05\\登鹳雀楼.txt", os.X_OK)         # 参数 os.X_OK 检查文件是否可加载
True
```

2. 使用 try 语句捕获程序异常

如果文件不存在或者文件不能读写，Python 解释器会抛出一个程序异常，可以使用 try 语句来捕获这个异常。

【例 5-20】 利用 try 语句捕捉程序异常。

```
#E0520.py                                    #【捕获程序异常】
try:                                         # 异常捕捉开始
    f = open("d:\\test\\05\\登鹳雀楼 temp.txt")  # 访问不存在的文件：登鹳雀楼 temp.txt
    for line in f.readlines():               # 循环读取文件行
        print(line)                          # 输出文件行
    f.close()                                # 关闭文件
except IOError:                              # 程序若正常则跳过本语句，若异常则执行下面语句
    print("文件不可读写!")                    # 文件异常提示信息或其他处理语句
>>>文件不可读写!                             # 程序运行结果
```

用 try 语句处理程序异常时，程序简单优雅，而且不需要引入其他模块。

3. 使用 pathlib 模块检查文件路径

可以使用 pathlib 模块检查文件路径和创建文件路径。

【例 5-21】 获取当前文件路径。

```
>>> from pathlib import Path                 # 导入标准模块 - 路径
>>> p = Path("d:\\test\\05\\登鹳雀楼.txt")    # 获取文件位置
>>> print(p)                                 # 输出文件位置
d:\\test\\05\\登鹳雀楼.txt
```

【例 5-22】 获取当前目录路径。

```
>>> from pathlib import Path                 # 导入标准模块 - 路径
>>> path = Path.cwd()                        # 获取目录位置
>>> print(path)                              # 输出目录位置
D:\\test
```

【例 5-23】 查看当前路径下的文件。

```
#E0523.py
from pathlib import Path                     # 导入标准模块 - 路径
p = Path.cwd()                               # 导入路径标准模块 pathlib
pys = p.glob('*.py')                         # 获取当前路径，查找符合规则的文件名
for py in pys:                               # 读取扩展名为.py 的文件
    print(py)
>>>…(输出略)                                 # 程序运行结果
```

程序第 4 行，函数 glob('*.py')表示查找文件扩展名为.py 的文件，* 号表示所有文件主名。glob()函数可以使用 * 、?、[ ]这三个匹配符。

5.2 CSV 文件读写

5.2.1 CSV 文件格式

1. CSV 格式文件概述

CSV(逗号分隔值)文件以纯文本格式存储数据,CSV 文件由任意数量的记录组成。每个记录为一行,行尾是换行符;每个记录由一个或多个字段组成,字段之间最常见的分隔符有逗号、空格等。CSV 文件广泛用于不同系统平台之间的数据交换,它主要解决数据格式不兼容的问题。

CSV 文件是一个纯文本文件,所有数据都是字符串。CSV 文件并不是表格,但是可以用 Excel 打开。CSV 文件无法生成和保存公式,CSV 文件不能指定字体颜色,没有多个工作表,不能嵌入图像和图表。CSV 文件采用的字符集有 ASCII、Unicode、EBCDIC、GB 2312 等。

2. CSV 格式文件规范

CSV 文件并不存在通用的格式标准,国际因特网工程小组(IEIT)在 RFC 4180(因特网标准文件)中提出的一些 CSV 格式文件的基础性描述,但是没有指定文件使用的字符编码格式,采用 7 位 ASCII 码是最基本的通用编码。目前大多数 CSV 文件遵循 RFC 4180 标准提出的基本要求,它们有以下规则。

(1) 回车换行符。

【例 5-24】 每一个记录都位于一个单独行,用回车换行符 CRLF(即\r\n)分隔。

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

(2) 结尾回车换行符。

【例 5-25】 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
```

(3) 标题头。

【例 5-26】 第 1 行可以有一个可选的标题头,格式和普通记录行格式相同。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。

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

(4) 字段分隔。

【例 5-27】 标题行和记录行中,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。每一行记录最后一个字段后面不能跟逗号。注意,字段之间一般用逗号分隔,也有用其他字符(如空格)分隔的 CSV。

```
aaa,bbb,ccc
```


(5) 字段双引号。

【例 5-28】 每个字段之间可用或不用半角双引号(")括起来(注意,Excel 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
```

【例 5-29】 字段中如果包含回车换行符、双引号或者逗号,该字段要用双引号括起来。

```
aaa, b CRLF
bb,"c,cc" CRLF
zzz,yyy,xxx
```

【例 5-30】 如果用双引号括字段,字段内双引号前必须加一个双引号进行转义。

```
""aaa","b""bb","ccc"
```

3. CSV 格式文件应用案例

【例 5-31】 二手汽车价格如表 5-2 所示,将表格内容按 CSV 格式存储。

表 5-2 二手汽车价格表

出厂日期	制造商	型号	说明	价格
2015	福特	SUV2015 款	ac, abs, moon	30000.00
2016	雪佛兰	前卫"扩展版"		49000.00
2017	雪佛兰	前卫"扩展版, 大型"		50000.00
2016	Jeep	大切诺基	低价急待出售! air, moon roof, loaded	47990.00

将表 5-2 内的数据以 CSV 格式表示。

```
出厂日期,制造商,型号,说明,价格
2015,福特,SUV2015 款,"ac, abs, moon",30000.00
2016,雪佛兰,"前卫""扩展版""",",49000.00
2017,雪佛兰,"前卫""扩展版, 大型""",",50000.00
2016,Jeep,大切诺基,"低价急待出售!
air, moon roof, loaded",47990.00
```

5.2.2 CSV 文件读取

1. 用标准模块读取数据

Python 标准库支持 CSV 文件的操作,读取 CSV 文件函数的语法格式如下。

```
csv.reader(csvfile, dialect = 'excel', ** fmtparams)          # CSV 文件读取的语法格式
```

参数 csvfile 为 CSV 文件或者列表对象。

参数 dialect 为指定 CSV 格式，dialect='excel'表示 CSV 文件格式与 Excel 格式相同。

参数 ** fmtparams 为关键字参数，用于设置特殊的 CSV 文件格式(如空格分隔等)。

返回值 csv.reader 是一个可迭代对象(如列表)。

【例 5-32】 “梁山 108 将 gbk.csv”文件内容如图 5-1 所示，读取和输出文件表头。

	A	B	C	D	E	F	G	H
1	座次	星宿	诨名	姓名	初登场回数	入山时回数	梁山泊职位	
2	1	天魁星	及时雨、呼	宋江	第18回	第41回	总督兵马大元帅	
3	2	天罡星	玉麒麟	卢俊义	第61回	第67回	总督兵马副元帅	
4	3	天机星	智多星	吴用	第14回	第20回	掌管机密正军师	
5	4	天闲星	入云龙	公孙胜	第15回	第20回	掌管机密副军师	
6	5	天勇星	大刀	关胜	第63回	第64回	马军五虎将之首兼左军	
7	6	天雄星	豹子头	林冲	第7回	第12回	马军五虎将之二兼右军	

图 5-1　梁山 108 将 gbk.csv 文件内容(片段)

```
1  # E0532.py                                                    # 【CSV 文件读第 1 行】
2  import csv                                                   # 导入标准模块
3  with open("d:\\test\\05\\梁山 108 将 gbk.csv") as f:          # 打开文件循环读取
4      reader = csv.reader(f)                                   # 创建读取对象
5      head_row = next(reader)                                  # 读文件第 1 行数据
6      print(head_row)
   >>>                                                          # 程序运行结果
   ['座次', '星宿', '诨名', '姓名', '初登场回数', '入山时回数', '梁山泊职位']
```

程序第 3 行，没有指明 CSV 文件编码时，如果文件中有字符串，则采用 GBK 编码。

程序第 4 行，从 CSV 文件读出的数据都是字符串。

【例 5-33】 “梁山 108 将 gbk.csv”文件内容如图 5-1 所示，读取 CSV 文件中第 3 列对应的所有数值，并且打印输出。

```
1  # E0533.py                                                    # 【CSV 文件读第 3 列】
2  import csv                                                   # 导入标准模块 - CSV 读写
3  with open("d:\\test\\05\\梁山 108 将 gbk.csv") as f:          # 打开文件，绝对路径
4      reader = csv.reader(f)                                   # 创建读取对象
5      column = [row[3] for row in reader]                      # 循环读文件第 3 列数据
6      print(column)
   >>> ['姓名', '宋江', '卢俊义', '吴用', …(输出略)            # 程序运行结果
```

程序第 5 行，这种方法需要事先知道列序号，如已知“姓名”在第 3 列。

2. 用 pandas 包读取数据

【例 5-34】 “梁山 108 将 utf8.csv”文件内容如图 5-1 所示，利用 pandas 软件包(参见 11.1 节)读取和输出文件中“诨名”和“姓名”两列数据的前 5 行。

```
1  >>> import pandas as pd                                      # 导入第三方包 - 数据分析
2  >>> data1 = pd.read_csv('d:\\test\\05\\梁山 108 将 utf8.csv', usecols = ['诨名', '姓名'], nrows = 5)
3  >>> print(data1)
                诨名    姓名
   0  及时雨、呼保义、孝义黑三郎    宋江
   1              玉麒麟    卢俊义…(输出略)
```

程序第 2 行，pd.read_csv()是 pandas 读取 CSV 文件数据的函数；参数 usecols=['诨名', '姓名']表示只读取文件中的这两列；参数 nrows=5 表示读取前 5 行的记录。

注意：pandas 软件包路径前不能有 r 参数；其次文件默认编码为 UTF-8。

【例 5-35】 “梁山 108 将 gbk.csv”文件内容如图 5-1 所示，读取 CSV 文件中“姓名”列对应数据，并打印输出。

```
1   #E0535.py                                              #【CSV 读指定列】
2   import csv                                             # 导入标准模块 - CSV 文件读写
3   import pandas as pd                                    # 导入第三方包 - 数据分析
4
5   filename = "d:\\test\\05\\梁山 108 将 gbk.csv"          # 文件为 GBK 编码(绝对路径)
6   list1 = []
7   with open(filename, 'r') as file:                      # 打开文件
8       reader = csv.DictReader(file)                      # 读取 CSV 文件到列表
9       column = [row['姓名'] for row in reader]           # 循环读取列表的"姓名"列
10      print(column)
    >>>['宋江', '卢俊义', '吴用', '公孙胜', …(输出略)      # 程序运行结果
```

程序第 8 行，pandas 包的 csv.DictReader()与 reader()函数类似，接收一个可迭代对象，返回的每一个列都放在一个字典的值内，字典的键则是列标题。

程序第 9 行，这种方法不需要事先知道列号，直接按表头名称“姓名”输出。

5.2.3 CSV 文件写入

1. 创建 CSV 文件

【例 5-36】 创建一个“学生 gbk.csv”文件。

```
1   #E0536.py                                              #【CSV 文件创建】
2   import csv                                             # 导入标准模块 - CSV 读写
3
4   csvPath = 'd:\\test\\05\\学生 gbk.csv'                  # 设置 CSV 文件保存路径
5   f = open(csvPath, 'w', encoding = 'gbk', newline = '') #【1.创建文件对象】
6   csv_writer = csv.writer(f)                             #【2.构建写入对象】
7   csv_writer.writerow(['姓名', "年龄", "性别"])           #【3.构建表头标签】
8   csv_writer.writerow(['贾宝玉', '18', '男'])            #【4.写入行内容】
9   csv_writer.writerow(['林黛玉', '16', '女'])
10  csv_writer.writerow(['薛宝钗', '18', '女'])
11  f.close()                                              #【5.关闭文件】
12  print('文件创建成功!')
    >>>文件创建成功!                                       # 程序运行结果
```

程序第 5 行，参数 newline=''解决写入数据时写 CSV 文件出现多余空行的问题。

2. 向 CSV 文件追加数据

【例 5-37】 “成绩 gbk.csv”文件如图 5-2 所示，在文件尾部写入一行新数据。

	A	B	C	D	E	F
1	学号	姓名	班级	古文	诗词	平均
2	1	宝玉	1	70	85	0
3	2	黛玉	1	85	90	0
4	3	晴雯	2	40	65	0
5	4	袭人	2	30	60	0

图 5-2 “成绩 gbk.csv”文件内容

```
1  #E0537.py                                                    #【CSV 文件写入数据】
2  import csv                                                   # 导入标准模块 - CSV 读写
3  with open("d:\\test\\05\\成绩 gbk.csv", 'a') as f:           # 打开文件,添加模式
4      row = ['5', '薛蟠', '01', '20', '60', '0']               # 插入行赋值
5      write = csv.writer(f)                                    # 创建写入对象
6      write.writerow(row)                                      # 在文件尾写入一行数据
7      print("写入完成!")
   >>>写入完成!                                                 # 程序运行结果
```

3. 两个 CSV 文件行内容合并

【例 5-38】 “成绩 source.csv”文件存放所有源数据(见图 5-2),另一个文件“成绩 update.csv”存放更新数据(见图 5-3)。两个文件表头相同,将“成绩 update.csv”文件内容添加到“成绩 source.csv”文件中。

	A	B	C	D	E	F
1	学号	姓名	班级	古文	诗词	平均
2	6	史湘云	1	80	80	0
3	7	刘姥姥	2	0	30	0

图 5-3 “成绩 update.csv”文件内容

```
1  #E0538.py                                                    #【CSV 文件行合并】
2  import csv                                                   # 导入标准模块 - CSV 读写
3  import pandas as pd                                          # 导入第三方包 - 数据分析
4
5  reader = csv.DictReader(open('d:\\test\\05\\成绩 update.csv'))  # 读取"成绩 update.csv"文件,相对路径
6  header = reader.fieldnames                                   # 获取表头标签信息
7  with open('d:\\test\\05\\成绩 source.csv', 'a') as csv_file:  # 以追加模式打开"成绩 source.csv"文件
8      writer = csv.DictWriter(csv_file, fieldnames = header)   # 批量写入新内容
9      writer.writerows(reader)                                 # 内容写入文件
10 print("写入完成!")
   >>>写入完成!                                                 # 程序运行结果
```

程序第 5 行,DictReader()为 pandas 的读取 CSV 文件函数。

程序第 8 行,DictWriter()为 pandas 的写入 CSV 文件函数。

4. 两个 CSV 文件列内容合并

【例 5-39】 源文件“成绩 gbk.csv”如图 5-4 所示,扩展文件“成绩 expand.csv”内容如图 5-5 所示。将两个文件进行列合并,合并后内容保存为“成绩 out.csv”文件。

	A	B	C	D	E	F
1	学号	姓名	班级	古文	诗词	平均
2	1	宝玉	1	70	85	0
3	2	黛玉	1	85	90	0
4	3	晴雯	2	40	65	0
5	4	袭人	2	30	60	0

图 5-4　源文件“成绩 gbk.csv”内容

	A	B	C
1	姓名	性别	年龄
2	宝玉	男	18
3	黛玉	女	16
4	晴雯	女	15
5	袭人	女	20

图 5-5　扩展文件“成绩 expand.csv”内容

从图 5-4 与图 5-5 可以发现，源文件中“姓名”一列与扩展文件中“姓名”一列的属性相同，内容相同，可以以这一列为主键，把扩展文件中的“性别”“年龄”这两列的数据添加到源文件中。如果扩展文件缺少某些行，则空着，最后源文件的行数不变。文件合并后内容如图 5-6 所示。

	A	B	C	D	E	F	G	H
1	学号	姓名	班级	古文	诗词	平均	性别	年龄
2	1	宝玉	1	70	85	0	男	18
3	2	黛玉	1	85	90	0	女	16
4	3	晴雯	2	40	65	0	女	15
5	4	袭人	2	30	60	0	女	20

图 5-6　列合并后的“成绩 out.csv”文件内容

```
1  #E0539.py                                                          #【CSV 文件列合并】
2  import pandas as pd                                                # 导入第三方包 - 数据分析
3
4  df1 = pd.read_csv('d:\\test\\05\\成绩 gbk.csv', encoding = 'gbk')          #【1.读入主文件】
5  df2 = pd.read_csv('d:\\test\\05\\成绩 expand.csv', encoding = 'gbk')       #【2.读入扩展文件】
6  outfile = pd.merge(df1, df2, how = 'left', left_on = '姓名', right_on = '姓名') #【3.设置合并主键】
7  outfile.to_csv('d:\\test\\05\\成绩 out.csv', index = False, encoding = 'gbk') #【4.保存 CSV 文件】
8  print("文件合并完成!")
   >>>文件合并完成!                                                    # 程序运行结果
```

程序第 4 行，df1＝pd.read_csv()为读入 CSV 文件全部内容，返回值 df1 为 pandas 中的 DataFrame 二维表格结构，即源文件“成绩 gbk.csv”为 5 行 6 列，df1 也为 5 行 6 列。

程序第 6 行，pd.merge(df1, df2, how＝'left', left_on＝'姓名', right_on＝'姓名')中，参数 how＝'left'表示两表拼接时以左侧为主键；参数 left_on＝'姓名'表示左侧主键名称为“姓名”；参数 right_on＝'姓名'表示右边的关联键名称也为“姓名”。

5. 解决 CSV 文件乱码问题

【例 5-40】 用 UTF-8 编码写 CSV 文件后，如果用 Excel 打开时会显示乱码。

```
1  #E0540.py                                                          #【CSV - 文件乱码】
2  import csv                                                         # 导入标准模块
3
4  file = open('d:\\test\\05\\csv 乱码.csv', 'w', encoding = 'utf - 8', newline = '')  # 创建文件
5  wr = csv.writer(file)                                              # 创建写入对象
6  wr.writerow(['姓名', '成绩'])                                       # 写入表头列名
7  wr.writerow(['宝玉', 85])                                          # 写入数据行
8  wr.writerow(['黛玉', 90])
9  wr.writerow(['宝钗', 88])
10 file.close()
```

```
11  print('文件写入成功。')                                    # 关闭文件句柄
    >>>文件写入成功。                                          # 程序运行结果
```

程序第 4 行，本语句写 CSV 文件后，用 Windows 自带的“记事本”程序打开 CSV 文件，则 CSV 文件显示正常，如图 5-7 所示；如果用 Excel 打开刚刚写入的文件，就会发现文件内容是乱码，如图 5-8 所示，这说明 Excel 的默认编码存在问题。

UTF-8 编码分为两种：一种是不带 BOM(字节顺序编码)的标准形式；另一种是带 BOM 的微软 Excel 格式。UTF-8 以字节为编码单元，它没有字节序的问题，因此它并不需要 BOM。但是 Excel 的 UTF-8 文件默认带 BOM 格式，即 utf-8-sig(UTF-8 with BOM)。写入 CSV 文件时，定义 encoding＝'utf-8-sig'就可以解决 Excel 乱码问题。将 E0540.py 程序第 4 行修改为以下形式，就可以解决 Excel 乱码问题(结果见图 5-9)。

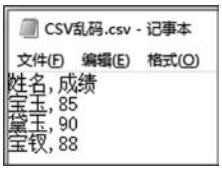


图 5-7　记事本打开正常

图 5-8　Excel 打开乱码

图 5-9　Excel 打开正常

```
4  file = open('d:\\test\\05\\csv 乱码解决.csv', 'w', encoding = 'utf-8-sig', newline = '')
```

5.3　Excel 文件读写

5.3.1　Excel 模块操作函数

1. 第三方软件包基本功能

处理 Excel 文件的第三方软件包有 xlrd/xlwt、xlutils、openpyxl、xlsxwriter、pandas、win32com 等。由于设计目的不同，每个模块通常都着重于某一方面功能，各有所长。

xlrd 和 xlwt 是应用非常广泛的 Excel 读写第三方包，它可以工作在任何平台，这意味着可以在 Linux 下读取 Excel 文件。xlrd 和 xlwt 是两个相对独立的模块，它们主要是针对 Office 2003 或更早版本的 xls 文件格式。xlrd 软件包可读取 xls 或 xlsx 文件；xlwt 软件包只能写入 xls 文件，除了最基本的写入数据和公式，xlwt 提供的功能非常少。

2. xlrd 和 xlwt 的安装和导入

xlrd 和 xlwt 是 Python 的第三方包，它们的安装方法如下。

```
1  > pip install xlrd - i https://pypi.tuna.tsinghua.edu.cn/simple   # 清华大学镜像网站安装 xlrd
2  > pip install xlwt - i https://pypi.tuna.tsinghua.edu.cn/simple   # 清华大学镜像网站安装 xlwt
```

可以通过 help(xlrd)命令查看 xlrd 帮助信息，它会输出 xlrd 包中的一些模块，以及一些成员变量、常量、函数等。

3. Excel 基本概念

Excel 中工作簿(book)和工作表(sheet)的区别：一个工作簿就是一个独立的文件，一个工作簿里可以有一个或者多个工作表，工作簿是工作表的集合。

每个工作表都有行和列，行以数字 1 开始，列以字母 A 开始。一个工作表由单元格(cell)组成，单元格可以存储数字和字符串。单元格以“行号”和“列号”进行定位。

注意：读取 Excel 数据时，sheet 号、行号、列号都是从索引 0 开始的。

4. Excel 操作函数

xlrd 软件包中，Excel 文件的常用操作语句案例如表 5-3 所示。

表 5-3　xlrd 软件包中 Excel 文件的常用操作语句案例

操作语句应用案例	说　明
import xlrd	导入模块，只能读不能写，可读 xlsx、xls 文件
data = xlrd.open_workbook('路径和文件名.xlsx')	打开 Excel 文件读取数据
table = data.sheets()[0]	获取一个工作表，通过索引顺序获取
table = data.sheet_by_index(0)	获取一个工作表，通过索引顺序获取
table = data.sheet_by_name('Sheet1')	获取一个工作表，通过工作表名称获取
name = table.name	获取工作表名称
nrows = table.nrows	获取行数
nclos = table.ncols	获取列数
row_value= table.row_values(i)	获取某行整行的值
col_value = table.col_values(i)	获取某列整列的值
cell_A1 = table.cell(0, 0).value	获取第 1 行第 1 列单元格数据，方法 1
cell_C4 = table.cell_value(2, 3)	获取第 3 行第 4 列单元格数据，方法 2
cell_A1 = table.row(0)[0].value	获取第 1 行第 1 列单元格索引号，方法 1
cell_A2 = table.col(1)[0].value	获取第 2 行第 1 列单元格索引号，方法 2
for i in range(nrows): 　print(table.row_values(i))	循环获取行的数据
for j in range(nclol): 　print(table.col_values(j))	循环获取列的数据
rows = sheet.get_rows() for row in rows: 　print(row[0].value)	遍历每行数据，输出第 1 列数据

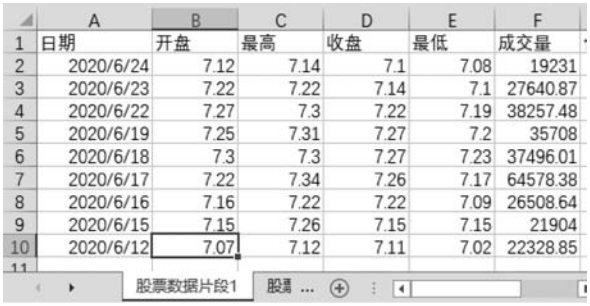
5.3.2　Excel 文件内容读取

1. 打开 Excel 工作簿

```
open_workbook()                                    # 打开 Excel 工作簿
```

函数返回的是一个 book 对象，通过 book 对象可以获得一个 sheet 工作表。

【例 5-41】 “股票数据片段.xlsx”文件有两个工作表(股票数据片段 1、股票年报数据片段 2)，内容如图 5-10、图 5-11 所示，输出其中所有 sheet 的名称。



	A	B	C	D	E	F
1	日期	开盘	最高	收盘	最低	成交量
2	2020/6/24	7.12	7.14	7.1	7.08	19231
3	2020/6/23	7.22	7.22	7.14	7.1	27640.87
4	2020/6/22	7.27	7.3	7.22	7.19	38257.48
5	2020/6/19	7.25	7.31	7.27	7.2	35708
6	2020/6/18	7.3	7.3	7.27	7.23	37496.01
7	2020/6/17	7.22	7.34	7.26	7.17	64578.38
8	2020/6/16	7.16	7.22	7.22	7.09	26508.64
9	2020/6/15	7.15	7.26	7.15	7.15	21904
10	2020/6/12	7.07	7.12	7.11	7.02	22328.85
11						

图 5-10 “股票数据片段.xlsx”中 sheet1 内容

	A	B	C	D	E	F
1	股票代码	股票名称	每股收益	增长率	bvps	净资产收益e
2	600589	广东榕泰	-0.76	-445.45		-18.53
3	603087	甘李药业	3.23	24.71	15.16	23.88
4	300841	康华生物	4.15	10.96	12.77	40.49
5	300842	帝科股份	0.94	27.03	5.37	19.24
6	300838	浙江力诺	0.66	-5.71	3.78	18.94
7	300837	浙矿股份	1.28	29.29	5.87	24.54
8	605288	凯迪股份	5.76	-17.36	19.22	35.27
9	300836	佰奥智能	1.74	27.01	8.09	24.1
10						

图 5-11 “股票数据片段.xlsx”中 sheet2 内容

```
1  #E0541.py                                          #【读取 Excel 工作表名称】
2  import xlrd                                        # 导入第三方包 - Excel 读入
3  excelPath = "d:\\test\\05\\股票数据片段.xlsx"      # 获取一个工作簿 book 对象
4  book = xlrd.open_workbook(excelPath, "r")          # 获取一个工作表 sheet 对象
5  sheets = book.sheets()                             # 遍历每一个工作表 sheet 名称
6  for sheet in sheets:                               # 输出工作表名称
7      print(sheet.name)
```

```
>>>                                                   # 程序运行结果
股票数据片段 1
股票年报数据片段 2
```

2. 读取 Excel 单元格内数据

```
1  sheet.cell(行号, 列号)                             # 读取 Excel 单元格内数据(行列均从 0 起)
2  sheet.cell_type(行号, 列号)                        # 读取 Excel 单元格内数据类型(行列均从 0 起)
```

【例 5-42】 “股票数据片段.xlsx”文件中，sheet1 内容如图 5-10 所示，sheet2 内容如图 5-11 所示。读取和输出两个工作表中第 4 行第 2 列单元格中的数据。

```
1  #E0542.py                                          #【读取单元格内容】
2  import xlrd                                        # 导入第三方包 - Excel 读入
3  excelPath = "d:\\test\\05\\股票数据片段.xlsx"      # 获取一个 book 对象
4  book = xlrd.open_workbook(excelPath)               # 获取一个 sheet 对象
5  sheets = book.sheets()                             # 遍历每一个 sheet
6  for sheet in sheets:                               # 读取 sheet1 和 sheet2 第 4 行第 2 列内容
7      print(sheet.cell_value(3, 1))
```

```
>>>                                          # 程序运行结果
7.22 康华生物                                # 注意,输出为竖行
```


【例 5-43】 “股票数据片段.xlsx”文件有两个工作表(股票数据片段 1、股票年报数据片段 2),内容如图 5-10、图 5-11 所示,输出 Excel 工作表相关数据。

```
#E0543.py                                                    #【读取 Excel 综合案例】
import xlrd                                                  # 导入第三方包 - Excel 读取
from datetime import date, datetime                          # 导入标准模块 - 日期时间

def read_excel():                                            #【定义 Excel 读取函数】
    workbook = xlrd.open_workbook('d:\\test\\05\\股票数据片段.xlsx')  # 打开文件
    print('全部工作表:', workbook.sheet_names())             # 获取所有工作表 sheet
    sheet2_name = workbook.sheet_names()[1]
    sheet2 = workbook.sheet_by_index(1)                      # 根据 sheet 索引获取 sheet 内容
    sheet2 = workbook.sheet_by_name('股票年报数据片段 2')    # 工作表名称赋值
    print('当前工作表:', sheet2.name, sheet2.nrows, '行', sheet2.ncols, '列')  # sheet 名/行/列
    rows = sheet2.row_values(3)                              # 获取第 4 行整行的值
    cols = sheet2.col_values(3)                              # 获取第 4 列行整列的值
    print('第 4 行全部内容:', rows)
    print('第 4 列全部内容:', cols)
    print('单元格第 2 行 2 列:', sheet2.cell(1, 1).value)   # 获取单元格第 2 行第 2 列内容
    print('单元格第 3 行 2 列:', sheet2.cell_value(2, 1))   # 获取单元格第 3 行第 2 列内容
    print('单元格数据类型:', sheet2.cell(1, 0).ctype)       # 获取单元格第 2 行第 0 列数据类型

if __name__ == '__main__':
    read_excel()                                             # 调用 Excel 读取函数
```

```
>>>…(输出略)                                 # 程序运行结果
```

5.3.3 Excel 文件写入数据

1. Excel 写操作语法

第三方软件包 xlwt 中,Excel 文件的常用操作语句应用案例如表 5-4 所示。

表 5-4 xlwt 软件包中 Excel 文件的常用操作语句应用案例

操作语句应用案例	说　明
import xlwt	导入写入库,只能写不能读,仅支持 xls 文件
book = xlwt.Workbook()	新建一个 Excel 文件
sheet = book.add_sheet('红楼梦_sheet')	添加一个名称为“红楼梦”的工作表 sheet
sheet.write(row, col, s)	在单元格中写入数据
book.save('test.xls')	保存到当前目录下,注意,只能保存为 xls 文件

2. 创建 Excel 文件并写入数据

xlrd 和 xlwt 是两个相对独立的模块,xlwt 只提供写入方法,无法读取 Excel 中的数据,因此对已经写入的数据无法修改。除了最基本的写入数据和公式,xlwt 提供的功能非常少,而且 Excel 文件只能保存为 xls 格式。

【例 5-44】 创建一个新文件“红楼梦 excel_out.xls”，并写入相关数据。

```
#E0544.py                                          #【写入 Excel 数据】
import xlwt                                        # 导入第三方包 - Excel 写入(只能写不能读)

stus = [['姓名', '年龄', '性别', '分数'],          # 定义工作表数据(列表嵌套)
        ['宝玉', 20, '男', 90],
        ['黛玉', 18, '女', 95],
        ['宝钗', 18, '女', 88],
        ['晴雯', 16, '女', 80]
        ]
book = xlwt.Workbook()                             # 新建一个 Excel 文件
sheet = book.add_sheet('红楼梦_sheet')             # 添加一个工作表 sheet 页
row = 0
for stu in stus:                                   # 行循环控制
    col = 0
    for s in stu:                                  # 列循环控制
        sheet.write(row, col, s)                   # 在单元格写入数据
        col += 1
    row += 1
book.save('d:\\test\\05\\红楼梦 excel_out.xls')    # 保存 Excel 文件
print("Excel 文件写入完成!")
```

```
>>> Excel 文件写入完成!                           # 程序运行结果如图 5-12 所示
```

	A	B	C	D	E
1	姓名	年龄	性别	分数	
2	宝玉	20	男	90	
3	黛玉	18	女	95	
4	宝钗	18	女	88	
5	晴雯	16	女	80	
6					

图 5-12 创建的“红楼梦 excel_out.xls”文件内容

5.4 其他文件读写

5.4.1 二进制文件读写

1. 二进制文件的解码和编码

文件在硬盘里以二进制格式存储，文件读到内存后需要转换为我们能看懂的数据格式。如果文件按照 GBK 格式的二进制字节码存储，而程序按照 UTF-8 编码格式读取，就会产生乱码问题。在 Python 程序设计中，我们需要指定用什么模式打开文件(字符串或二进制字节码)。如果以二进制字节码读取文件，还需要指定读取数据时采用什么编码格式(ASCII、UTF-8、GBK 等)，如果不指定读取编码格式，Python3 默认编码为 UTF-8。

(1) 读二进制文件的解码操作 decode()。二进制文件按照字节流的方式读写。从二进制文件中读取数据时，应先进行解码操作，decode()函数可以将字符串由其他编码转换为 UTF-8 字节码，语法如下。

```
字符串变量名.decode([encoding = "UTF-8"][,errors = "strict"])   # 二进制解码的语法格式
```

参数 encoding 为可选参数，表示要使用的编码，默认编码为 UTF-8。

参数 errors 为可选参数，表示设置错误处理方案。默认为 strict，含义是编码错误时，引起一个 UnicodeError 异常提示。

(2) 写二进制文件的编码操作 encode()。数据在写入二进制文件之前，应先进行编码操作，encode()函数的作用是将字符串由 UTF-8 字节码转换为其他编码。

```
字符串变量名.encode([encoding = "UTF-8"][,errors = "strict"])   # 二进制编码的语法格式
```

其参数与上面的解码函数相同。

【例 5-45】 字符串的编码和解码。

```
>>> a = '中国'                                    # 定义字符串
>>> a_utf8 = a.encode('UTF - 8')                  # 将字符串编码为 UTF - 8 字节码
>>> a_utf8
b'\xe4\xb8\xad\xe5\x9b\xbd'
>>> a_unicode = a_utf8.decode('UTF - 8')          # 将 UTF - 8 字节码解码为字符串
>>> a_unicode
'中国'
>>> b_gbk = a_unicode.encode('GBK')               # 将 UTF - 8 字节码转换为 GBK 字节码
>>> b_gbk
b'\xd6\xd0\xb9\xfa'
>>> b_unicode = b_gbk.decode('GBK')               # 将 GBK 字节码转换回字符串
>>> b_unicode
'中国'
```

2. 读取二进制文件

步骤 1：读取二进制文件时，使用 open()函数打开文件，打开模式选择二进制数据读取模式"rb"，使读取的数据流是二进制。

步骤 2：由于纯二进制文件内部不含任何数据结构信息，因此需要自定义二进制数据读取时的字节数。如数据类型是 float32 型时，对应数据的字节数就是 4B。

步骤 3：使用循环语句块逐个读入 n 个字节数据。

【例 5-46】 读取二进制数据库文件“梁山 108 将.db”。

```
#E0546.py                                                  #【读取二进制文件】
import struct                                              # 导入标准模块 - 工具函数

with open("d:\\test\\05\\梁山 108 将.db", mode = "rb") as f:   # rb 表示以二进制形式打开文件
    f.seek(3)                                              # 移至指定字节位置
    a = f.read(16)                                         # 读入 16 字节
    print(type(a))                                         # 打印 a 类型
    print(len(a))                                          # 打印 a 内字节数
    print(a)                                               # 打印 a 内数据，以十六进制数显示
    val_tuple = struct.unpack("< 4H2I", a)                 # 解析一个数据
    print(val_tuple)
    val_list = list(val_tuple)                             # 将元组转为 list
    print(val_list)                                        # 关闭文件
```

```
>>>                                                    # 程序运行结果
16
b'ite format 3\x00\x10\x00\x01'
(29801, 8293, 28518, 28018, 857764961, 16781312)
[29801, 8293, 28518, 28018, 857764961, 16781312]
```

程序第 6 行,16 字节为 4 个 unsigned short 数据和 2 个 unsigned int 数据,字节排列为小端字节序(Little Endian),返回数据类型为元组。

程序第 10 行,如果解析一个数据,则应当读取与数据存储空间大小一致的字节数目,unpack 仍然返回元组数据类型。

3. 写二进制文件

【例 5-47】 将字符串写入二进制文件。

```
1  #E0547.py                                          #【写二进制文件】
2  f = open('d:\\test\\05\\test1.dat', 'wb')          # 以二进制覆盖写模式创建文件
3  s1 = 'Python 你好!'                                # 设置字符串内容
4  s2 = s1.encode('GBK')                              # 字符串转换为 GBK 字节码
5  f.write(s2)                                        # 把字节码写入文件
6  f.close()                                          # 关闭文件
7  print('源字符串:', s1)
8  print('GBK 字节码:', s2)
```

```
>>>                                                   # 程序运行结果
源字符串: Python 你好!
GBK 字节码: b'Python\xc4\xe3\xba\xc3\xa3\xa1'
```

此外,Python 提供了 pickle、struct、shutil 等标准模块进行二进制文件读写操作。

5.4.2 JSON 文件读写

1. JSON 文件数据类型

JSON(JavaScript 对象符号)是一种轻量级的数据交换格式。JSON 是 ECMAScript(欧洲计算机协会)制定的一个规范,它采用独立于编程语言的文本格式来存储和表示数据。JSON 采用文本格式。网址在向页面 JavaScript 传输数据时,JSON 是最常用的数据格式之一。Python 与 JSON 数据类型存在一些细小的差别。

2. 读取 JSON 文件中的数据

【例 5-48】 “工资.json”文件内容如下所示。

```
1  [
2    {
3      "姓名": "关羽",
4      "年龄": 30,
5      "职业": "将军",
6      "工资": 10000
7    },
8     {
```

```
    "姓名": "周仓",
    "年龄": 25,
    "职业": "副官",
    "工资": 5000
  },
  {
    "姓名": "糜芳",
    "年龄": 40,
    "职业": "文书",
    "工资": 3000
  }
]
```

【例 5-49】 方法 1：读取并输出“工资.json”文件内容。

```
#E0549.py                                          #【读取 Json 文件】
import json                                        # 导入标准模块 - Json 文件读写
f = open('d:\\test\\05\\工资.json')                 # 将数据导入文件句柄 f
data = json.load(f)                                # 用 loads()方法将字符串转换为列表
for dict_data in data:                             # 遍历 data,打印列表中的每一个字典
    print(dict_data)
```

```
>>>                                                # 程序运行结果
{'姓名': '关羽', '年龄': 30, '职业': '将军', '工资': 10000}
{'姓名': '周仓', '年龄': 25, '职业': '副官', '工资': 5000}
{'姓名': '糜芳', '年龄': 40, '职业': '文书', '工资': 3000}
```

【例 5-50】 方法 2：读取“工资.json”文件中某一键对应的值。

```
#E0550.py                                          #【读取 Json 文件的值】
import json                                        # 导入标准模块 - Json 读写
f = open('d:\\test\\05\\工资.json')                 # 将数据导入文件句柄 f
data = json.load(f)                                # 用 loads()方法将字符串转换为列表
for dict_data in data:                             # 遍历 data,输出列表中每一个字典
    print(dict_data['姓名'], '工资 = ' + str(dict_data['工资']))
```

```
>>>                                                # 程序运行结果
关羽 工资 = 10000
周仓 工资 = 5000
糜芳 工资 = 3000
```

习　题　5

5-1　简要说明文本文件读写的步骤。

5-2　Python 提供了哪些读取文本文件内容的函数？

5-3　open()语句或 with open()语句都可以实现文件读写，它们有哪些差别？

5-4　简要说明 CSV 格式文件的特征。

5-5　简要说明 xlrd 和 xlwt 软件包的特征。

5-6　编程：检查 d:\\test\\05\\test_no.txt 文件是否存在。

5-7　编程：数据文件 test.txt 内容如下所示，把数据读入到矩阵中。

```
1 2 2.5
3 4 4
7 8 7
```

注：每个数据以空格分开。

5-8　编程：读取并打印“鸢尾花数据集.csv”。

5-9　编程：读取并打印“鸢尾花数据集.csv”数据集中第 2 行。

5-10　编程：将表 5-5 中的数据保存到“成绩.csv”文件。

表 5-5　题 5-10 表

学号	姓名	性别	班级	古文	诗词
100001	宝玉	男	1 班	85	70
100002	黛玉	女	2 班	88	85