

第6章 文件

学习目标

- 掌握文件的打开和关闭
- 掌握读写文件的方法
- 了解 Python 的中文分词第三方库 jieba 库的使用

第2章中介绍了输入数据的 `input()` 函数和输出处理结果的 `print()` 函数，这是用户与程序交互数据的重要方式，但是这种方式输入和输出的数据都存储在计算机内存中，计算机关机后数据将丢失，无法长久保存。如果设计程序使用 `input()` 函数输入 30 个同学的考试成绩，然后计算输出平均分，每次运行程序都需要通过键盘输入每个成绩，非常麻烦，这种情况下，可以将所有考试成绩存入文件，每次运行程序只需要从文件中读一遍数据就可以了。本章将介绍如何读写文件。

6.1 文件概述

计算机中，文件是存储在某种存储器（如硬盘）上的信息的集合，操作系统是以文件为单位来管理硬盘中的数据的。一篇文章、一段视频、一个可执行程序，都可以被保存为一个文件，并赋予一个文件名。

文件名包括两部分：主文件名和扩展名，主文件名是用户自定义的，扩展名也称为后缀，表示文件的类型，例如文件名“例 5-1.py”，其中“例 5-1”是主文件名，是用户给出的，“py”是 Python 源程序文件的扩展名。

一般来说，文件可分为文本文件、视频文件、音频文件、图像文件、可执行文件等多种类别，这是从文件的功能进行分类的。从数据存储的角度来说，所有的文件本质上都是一样的，都是由一个个字节组成的，归根到底都是 01 代码。不同的文件呈现出不同的形态，有的是文本，有的是视频，这主要是文件的创建者和解释者，即使用文件的软件约定好了文件格式。例如常见的纯文本文件，扩展名为 `txt`，这种文件使用 Windows 的“记事本”程序来打开，是一段不具备 word 那种丰富格式的文字。除了纯文本文件外，图像、视频、可执行文件等一般被称作“二进制文件”。二进制文件如果用“记事本”程序打开，看到的是一片乱码。

事实上所谓“文本文件”和“二进制文件”，只是约定俗成的、从计算机用户角度出发进行的分类，并不是计算机科学的分类。因为从计算机科学的角度来看，所有的文件都是由二进制位组成的，都是二进制文件。文本文件和其他二进制文件只是格式不同而已。在 Python 读写文件时，文本文件和二进制文件需要使用不同的打开模式。

大量的文件如果不加分类放在一起，用户使用起来显然非常不便，因此计算机中引入了树形目录的管理机制，目录也叫文件夹，例如 Windows 的资源管理器，可以把文件放在不同的文件夹中，文件夹中还可以嵌套文件夹，这样使得用户可以更加方便更加清晰地管理和使用文件。

在描述文件属性时，文件所在的文件夹是文件的重要属性，文件保存的位置称为路径，有绝对路径和相对路径之分，绝对路径是从文件所在驱动器开始描述文件的保存位置，而相对路径是从当前目录开始描述文件的保存位置。

例如“pip.exe”文件存储在“C:\Users\sjfigh\venv\Scripts”文件夹中，则绝对路径为“C:\Users\sjfigh\venv\Scripts\pip.exe”，如果当前目录是“C:\Users\sjfigh\venv”，则使用相对路径描述为“\Scripts\pip.exe”。

6.2 文件的打开与关闭

读写文件之前，必须先打开文件，打开文件使用 `open()` 函数，语法格式为：

```
fp = open(文件名[, 打开模式][, 编码方式])
```

`open` 函数中常用的有三个参数，第一个参数是文件名，必选参数，包含文件的存储路径，如果没有文件的存储路径，则默认打开的文件和 `py` 文件存放在同一个文件夹下。第二个参数是打开模式，可选参数，打开模式中字符代表的含义如表 6-1 所示。第三个参数是指定的编码方式，可选参数，比如 `gbk` 或者 `utf-8`。

表 6-1 文件打开模式字符含义表

字符	含义
"r"	读文件，默认选项
"w"	写文件，首先清空文件
"x"	独占创建文件，如果文件已经存在，则失败
"a"	打开写入文件，如果文件存在，则追加到文件的末尾
"b"	二进制模式
"t"	文本模式，默认选项
"+"	打开磁盘文件进行更新（读写）

Python 打开文件的方式分为二进制文件和文本文件两种，如果打开参数中含有“b”，则表示以二进制方式打开，这种方式返回的是未经解码的二进制字节，如果打开参数中有“t”或者没有“b”，则表示以文本方式打开文件，这种方式返回的是经过解码的字符串，可以使用平台依赖的编码方式，也可以在 `open()` 的第三个参数中给出。

例如，打开模式“r+”、“w+”、“a+”表示在原功能基础上增加同时读写功能，“rb+”、“wb+”、“ab+”表示以二进制读写模式打开。

关闭文件使用 `close()`，文件使用完毕后必须关闭，因为文件对象会占用操作系统的资源，并且操作系统同一时间能打开的文件数量也是有限的。

6.3 读文件

`read(n)` 方法是常用的读文件的语句，参数 `n` 表示从文件当前位置读取 `n` 个字节的内容，如果省略 `n` 或者 `n=-1` 的话，则读取到文件结束。Python 把内容读到内存，用一个字符串对象表示。`read(n)` 方法的语法格式为：

```
<file>.read(n)
```

【例 6-1】设计程序完成如下功能，D 盘根目录下的 `file1.txt` 文件中存有某班 20 位同学的英语考试成绩，如图 6-1 所示，统计并显示 90~100、80~89、70~79、60~69 以及 60 分以下各分数段的同学的人数。



图 6-1 文本文件 `file1`

【解析】从图中可以看到若干名同学的成绩使用逗号分隔，可以使用字符串的 `split()` 方法

来进行拆分。另外，每个分数段和相应的人数存在对应关系，适合使用字典数据类型。打开文件的语句 `f1=open("d:\\file1.txt","r")`，D 盘根目录使用的是“\\”，也可以写作 `f1=open(r"d:\file1.txt","r")`，加了 r 之后就可以使用“\”了。

【参考代码】

ex6-1.py

```
1 f1=open("d:\\file1.txt","r")
2 s=f1.read()
3 x=s.split(",")
4 y={"90~100":0,"80~89":0,"70~79":0,"60~69":0,"60 以下":0}
5 for i in x:
6     if 90<=eval(i)<=100:
7         y["90~100"]+=1
8     if 80<=eval(i)<=89:
9         y["80~89"]+=1
10    if 70<=eval(i)<=79:
11        y["70~79"]+=1
12    if 60<=eval(i)<=69:
13        y["60~69"]+=1
14    if eval(i)<60:
15        y["60 以下"]+=1
16 print(y)
17 f1.close()
```

【运行结果】

```
{'90~100': 3, '80~89': 6, '70~79': 3, '60~69': 4, '60 以下': 4}
```

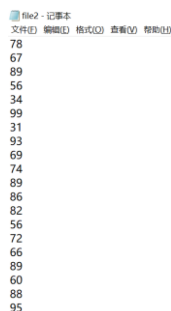
第二个常用的读文件的方法是 `readline(n)` 方法，如果给出参数 `n`，则读取文件的一行的前 `n` 个字节的内容，如果省略 `n` 或者 `n=-1` 的话，则读取文件的一行。`readline(n)` 方法的语法格式为：

`<file>.readline(n)`

第三个常用的读文件的方法是 `readlines(n)` 方法，如果给出参数 `n`，则读取文件的前 `n` 行，如果省略 `n` 或者 `n=-1` 的话，则读取文件的所有行，以每行为元素形成一个列表。`readlines(n)` 方法的语法格式为：

`<file>.readlines(n)`

【例 6-2】设计程序完成如下功能，D 盘根目录下的 `file2.txt` 文件中存有某班 20 位同学的英语考试成绩，如图 6-2 所示，统计并显示 90~100、80~89、70~79、60~69 以及 60 分以下各分数段的同学的人数。



```
file2 - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
78
67
89
56
34
99
31
93
69
74
89
86
82
56
72
66
89
60
88
95
```

图 6-2 文本文件 file2

【解析】例 6-2 与例 6-1 的唯一的不同在于文本文件存储成绩的格式不同，例 6-1 中成绩之间用逗号分隔，例 6-2 中成绩按行分隔。

【参考代码 a】

ex6-2a.py

```
1 f2=open("d:\\file2.txt","r")
2 y={"90~100":0,"80~89":0,"70~79":0,"60~69":0,"60 以下":0}
3 try:
4     while True:
5         s=f2.readline()
6         if s=="":
7             break
8         else:
9             if 90<=eval(s)<=100:
10                 y["90~100"]+=1
11             if 80<=eval(s)<=89:
12                 y["80~89"]+=1
13             if 70<=eval(s)<=79:
14                 y["70~79"]+=1
15             if 60<=eval(s)<=69:
16                 y["60~69"]+=1
17             if eval(s)<60:
18                 y["60 以下"]+=1
19 finally:
20     f2.close()
21 print(y)
```

【参考代码 b】

ex6-2b.py

```
1 f2=open("d:\\file2.txt","r")
2 y={"90~100":0,"80~89":0,"70~79":0,"60~69":0,"60 以下":0}
3 try:
4     s=f2.readlines()
5     for i in s:
6         if 90<=eval(i)<=100:
7             y["90~100"]+=1
8         if 80<=eval(i)<=89:
9             y["80~89"]+=1
10        if 70<=eval(i)<=79:
11            y["70~79"]+=1
12        if 60<=eval(i)<=69:
13            y["60~69"]+=1
14        if eval(i)<60:
15            y["60 以下"]+=1
16 finally:
17     f2.close()
```

18 print(y)

表 6-2 read()、readline() 和 readlines() 对比表

方法	参数	返回值	特点
read(n)	n 表示读取文件中的字节数, 省略 n 或者 n=-1 表示读文件中的所有字节	字符串	如果文件非常大, 尤其是大于内存时, 无法使用
readline(n)	省略 n 或者 n=-1 表示读文件中的一行, 如果给出 n, 读出一行的前 n 个字节	字符串	保持当前行的内存, 比 readlines 慢得多
readlines(n)	省略 n 或者 n=-1 表示一次性读取整个文件, 以每行元素形成一个列表, 如果给出 n, 读入前 n 行	列表	自动将文件内容分析成一个行的列表

6.4 写文件

写文件有两个方法, 一个是 write(str) 方法, 用于向文件中写入指定字符串 str, 在文件关闭前或缓冲区刷新前, 字符串内容存储在缓冲区中, 这时在文件中是看不到写入的内容的。write(str) 方法的语法格式为:

```
<file>.write(str)
```

另外一个方法是 writelines(strlist) 方法, 用于向文件中写入一个序列的字符串 strlist, 这一序列字符串可以由迭代对象产生的, 如一个字符串列表。writelines(strlist) 方法的语法格式为:

```
<file>.writelines(strlist)
```

【例 6-3】设计程序完成如下功能, 已知列表 x 中存有若干整数, x=[7, 15, 11, 24], 将其中的素数存入 D 盘根目录下的 file3.txt 文件中。

【解析】首先将列表中的每一个数取出, 然后使用第 3 章中介绍的方法判断一个数是否是素数, 是素数的话则写入文件中。写入文件后, file3.txt 内容如图 6-3 所示。

【参考代码 a】

ex6-3a.py

```

1  x=[7, 15, 11, 24]
2  f1=open("d:\\file3.txt", "w")
3  for i in x:
4      flag=True
5      for j in range(2, i):
6          if i%j==0:
7              flag=False
8              break
9      if flag==True:
10         f1.write(str(i)+"\n")
11  f1.close()

```

【参考代码 b】

ex6-3b.py

```

1  x=[7, 15, 11, 24]

```

```

2   y=[]
3   fl=open("d:\\file3.txt","w")
4   for i in x:
5       flag=True
6       for j in range(2,i):
7           if i%j==0:
8               flag=False
9               break
10      if flag==True:
11          y.append(str(i)+"\n")
12  fl.writelines(y)
13  fl.close()

```

【运行结果】

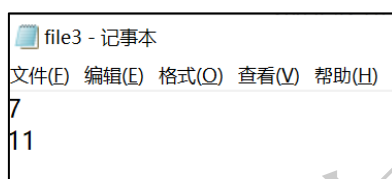


图 6-3 例 6-3 运行结果图

【例 6-4】设计程序完成如下功能，已知 D 盘根目录下存放有 file4a.txt 和 file4b.txt，文件内容如图 6-4 所示，将两个文件的内容合并存入 file4a.txt。

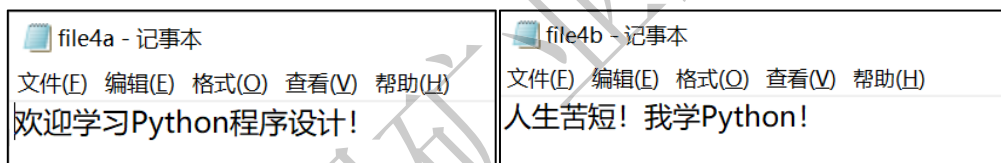


图 6-4 例 6-4 中 file4a.txt 和 file4b.txt

【解析】题中要求将两个文件的内容合并存入 file4a.txt，其含义是读取 file4b.txt 的内容，并追加在 file4a.txt 文件的后面，所以对于 file4b.txt 是读文件，对于 file4a.txt 是追加式的写文件。

【参考代码】

ex6-4.py

```

1   fl=open("d:\\file4a.txt","a")
2   f2=open("d:\\file4b.txt","r")
3   x=f2.read()
4   fl.write(x)
5   fl.close()
6   f2.close()

```

【运行结果】

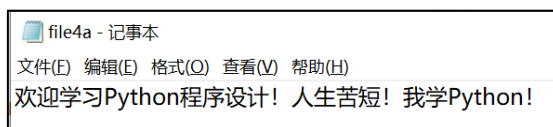


图 6-5 例 6-4 运行结果图

之前的例子都是从头到尾按顺序读写文件，读写完毕，文件读写位置将顺序移动，在某

些情况下，如果需要在文件中任意移动读写位置，可以使用 `seek()` 方法，`seek()` 的一般语法格式为：

```
<file>.seek(offset, whence)
```

`Offset` 为相对于所指示位置的字节偏移量，`whence` 为可选参数，`whence=0` 表示相对于文件开始位置，`offset=1` 表示相对于当前读写位置，`offset=2` 表示相对于文件结尾位置。

【例 6-5】设计程序完成如下功能，已知 D 盘根目录下存放有 `file5.txt`，文件内容如图 6-6(a) 所示，在每行字符串前加上序号，修改后文件内容如图 6-6(b) 所示。

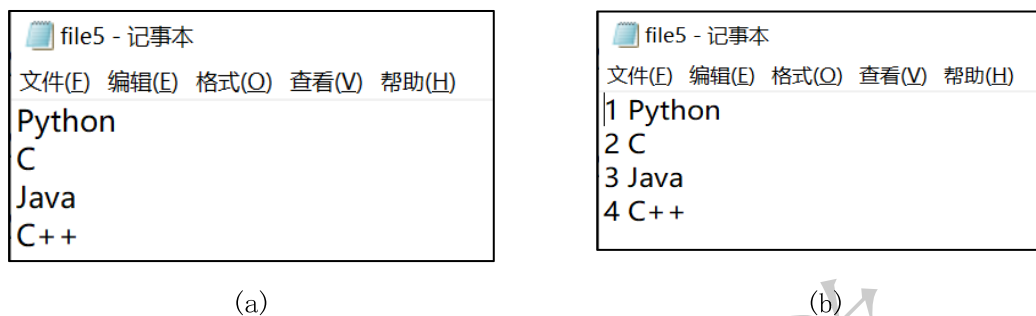


图 6-6 例 6-5 文件内容图

【参考代码】

ex6-5.py

```
1 f=open("d:\\file5.txt", "r+")
2 x=f.readlines()
3 for i in range(0, len(x)):
4     x[i]=str(i+1)+" "+x[i]
5 f.seek(0)
6 f.writelines(x)
7 f.close()
```

【解析】本程序中，首先需要读出文件的内容，然后给每行内容前加上序号，需要注意的是，读文件结束时，当前读写位置在文件结尾处，接下来在文件开始处写文件，所以使用了第 5 行 `f.seek(0)` 将读写位置移到文件开始处，最后就可以写文件了。

6.5 实例《西游记》词频统计

【实例功能】《西游记》是我国古代四大文学名著之一，师徒四人去西天取经，经历九九八十一难，终于取得真经的故事深入人心，那么在西游记中，师徒四人唐僧、悟空、八戒和沙僧这四个词哪个出现的最多呢？我们用 Python 程序来统计一下吧，看看和你想象的一样不一样。

【实例代码】

实例 6.py

```
1 import jieba
2 shitu=["唐僧","悟空","八戒","沙僧"]
3 xyj=open("西游记.txt", "r", encoding="gbk")
4 txt=xyj.read()
5 words=jieba.lcut(txt)
6 tj={}
```

```
7     for i in shitu:
8         tj[i]=words.count(i)
9     print(tj)
10    xyj.close()
```

【实例运行】

```
Building prefix dict from the default dictionary ...
Dumping model to file cache C:\Users\sjfgh\AppData\Local\Temp\jieba.cache
Loading model cost 0.790 seconds.
Prefix dict has been built succesfully.
{'唐僧': 802, '悟空': 379, '八戒': 1677, '沙僧': 721}
```

程序运行结果显示八戒出现的最多,比出现第二多的唐僧多一倍呢,是不是有点出人意料呢。这里需要说明的是,小说中每个人物可能都有若干个称呼,比如“唐僧”可能被称为“师傅”、“圣僧”等,本实例程序没有考虑这个问题。

【解析】

在实例6中,我们统计了《西游记》文本中四个词的出现次数,是在对这个“西游记.txt”文件的读操作的基础上进行的统计分析。

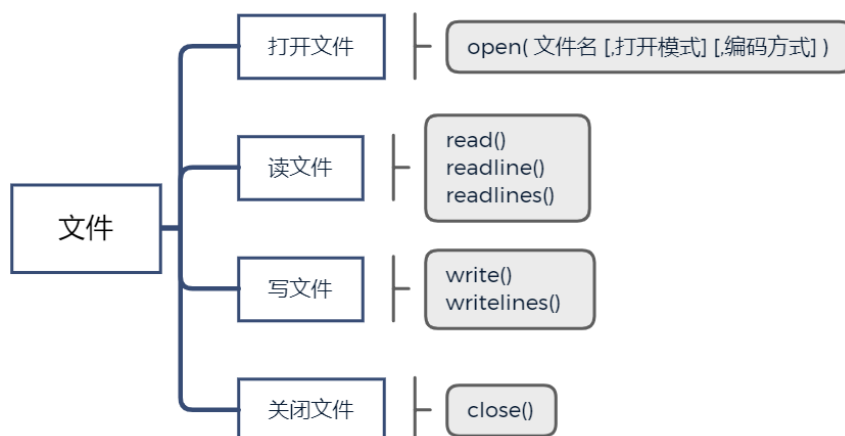
读写文件是最常用的 I/O 操作,第3行 `xyj=open("西游记.txt","r",encoding="gbk")` 是读写文件的第一步打开文件,第4行 `txt=xyj.read()` 是读文件,最后一行 `xyj.close()` 是关闭文件。

在这个例子中我们用到了一个第三方库 `jieba`, `jieba` 是优秀的中文分词的第三方库,中文不同于英文,英文每个单词间都有空格,中文词与词之间紧密相连,没有分隔,所以需要第三方库 `jieba` 来完成分词的功能,这个库需要额外安装,类似地,在命令行窗口输入 `pip install jieba`,就可以完成安装了。实例6的代码的第5行 `words=jieba.lcut(txt)` 就使用了 `jieba` 库中的 `lcut` 方法将存有西游记所有文本的变量 `txt` 进行分词,返回的结果 `words` 是分词生成的列表,列表的每一个元素都是一个词。

比如说,使用 `lcut` 方法对“我爱北京天安门。”进行分词,结果就是 `['我', '爱', '北京', '天安门', '。']`。

```
>>> import jieba
>>> x=jieba.lcut("我爱北京天安门。")
>>> x
['我', '爱', '北京', '天安门', '。']
```


6.6 本章小结



课后习题

一、选择题

- 使用 `open()` 打开文件，下列选项中，_____不是合法的文件打开模式。
A. a B. r+ C. wb+ D. c
- `open()` 的默认文件打开方式是_____。
A. r B. r+ C. w D. w+
- 下列文件打开方式中，_____不能对打开的文件进行写操作。
A. w B. wt C. r D. a
- 下列方法中，_____不是 Python 对文件的读操作方法。
A. `read()` B. `readline()` C. `readtext()` D. `readlines()`
- 以下函数中，用于文件定位的函数是_____。
A. `read()` B. `seek()` C. `write()` D. `open()`

二、填空题

- 已知 C 盘根目录下有一个文本文件 `in.txt`，如果程序中需要读出此文件内容，则打开文件语句是_____。
 - 补全下列程序代码，完成以下功能，用户输入文件路径，以文本文件方式读入文件内容并逐行打印。例如 C 盘根目录下有一个文本文件 `in.txt`，则用户输入 “`c:\in.txt`”。
- ```

fname=input("请输入要打开的文件：")
fo=open(_____, "r")
for line in _____:
 print(line)
fo.close()

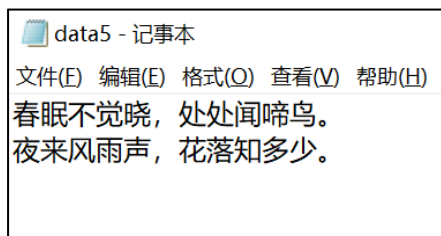
```

### 三、编程题

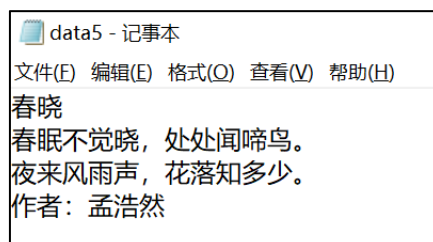
- 设计程序完成如下功能，D 盘根目录下的 `data1.txt` 文件中存有某班若干位同学的英语考试成绩，以逗号隔开，读取文件，计算并输出总分和平均分。
- 设计程序完成如下功能，D 盘根目录下的 `data2.txt` 文件中存有若干个随机的整数，以逗号隔开，输入一个整数，判断并输出这个数是否存在于文件中。
- 设计程序完成如下功能，把字符串 “Hello world!” 写入 D 盘根目录下的 `data3.txt` 文

件中。

- 设计程序完成如下功能，已知列表  $x$  中存有若干整数， $x=[6, 27, 33, 21, 14, 9]$ ，将其中的最大值和最小值存入 D 盘根目录下的 data4.txt 文件中。
- 设计程序完成如下功能，D 盘根目录下的 data5.txt 文件中是一首古诗，如图 a 所示，改写文件，在古诗前加上诗的名字“春晓”，在古诗后加上“作者：孟浩然”，如图 b 所示。



a



b

中国矿业大学