第 5 章

文件操作

文件读写是 Python 编程中的基本操作，它允许程序与外部世界进行数据交换和通信。理解如何进行文件读写操作对于许多应用程序和编程任务都至关重要。

5.1 文件的打开与关闭

在 Python 的使用中，经常需要对文件进行读写与存储操作。Python 进行文件操作通常涉及打开、读取、写入和关闭文件。

5.1.1 文件类型

用 Python 可以处理多种类型的文件，它提供了广泛的库和模块来处理各种文件格式。用户可以根据需要选择适当的库和模块，以便有效地操作和处理不同类型的文件。从数据存储方式来分，文件类型有文本文件和二进制文件。

1. 文本文件

文本文件包含文本数据，以纯文本形式存储，通常使用 ASCII 或 UTF-8 编码。由于文本文件仅包含纯文本数据，因此它们在不同操作系统之间具有很好的跨平台兼容性。这些文件通常用来存储文本文档、配置文件、日志文件等。常见的文本文件类型有以下几种。

CSV 文件：CSV（Comma-Separated Values）文件是一种常见的文本文件格式，用于存储表格数据。Python 中有各种库（如 csv 模块）可用于处理 CSV 文件。

JSON 文件：JSON（JavaScript Object Notation）文件是一种用于存储结构化数据的文本文件格式。Python 中有内置的 json 模块用于解析和生成 JSON 数据。

XML 文件：XML（eXtensible Markup Language）文件是一种用于存储和交换数据的标记语言。Python 库（如 xml.etree.ElementTree）可用于解析和生成 XML 数据。

2. 二进制文件

二进制文件包含非文本数据，如图像、音频、视频、压缩文件等。这些文件通常以二进制形式存储，可以使用二进制读写模式来处理。常见的二进制文件类型有以下几种。

Excel 文件：Excel 文件是 Microsoft Excel 电子表格文件，Python 中有库（如 pandas 和 openpyxl）可用于处理 Excel 文件。

图片文件：图片文件包括各种图像格式，如 JPEG、PNG、GIF。Python 中有库（如 Pillow）可用于处理图像文件。

音频文件：音频文件包括音频格式，如 MP3、WAV。Python 中有库（如 pydub 和 audioread）可用于处理音频文件。

视频文件：视频文件包括视频格式，如 MP4、AVI。Python 中有库（如 moviepy）可用于处理视频文件。

数据库文件：数据库文件用于存储结构化数据，如 SQLite 数据库文件（.db 文件）。Python 中有内置的 sqlite3 模块可用于操作 SQLite 数据库。

日志文件：日志文件包含应用程序的日志信息，用于调试和错误跟踪。Python 中有 logging 模块用于生成和处理日志。

5.1.2　文件的打开

1. 语法格式

文件打开的语法格式一：

```
file = open(file_path, mode, encoding=None, buffering)
    <文件操作>
```

文件打开的语法格式二：

```
with open(file_path, mode, encoding=None, buffering) as file:
    <文件操作>
```

2. 说明

使用 open()函数打开文件后，可以使用文件对象 file 来对文件进行读取、写入、关闭等操作。如果该文件无法被打开，会抛出 OSError 异常。

（1）file_path：文件的路径，可以是文件的绝对路径或相对路径。

（2）mode：打开文件的模式，具体方式如表 5-1 所示。如不指定打开文件的模式，默认会使用文本模式。

表 5-1　打开文件的模式

模式	描　　述
t	文本模式（默认）
b	二进制模式
U	通用换行模式（不推荐）
x	写模式，新建一个文件，如果该文件已存在则会报错
r	以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式
w	打开一个文件只用于写入。如果该文件已存在则打开文件，并从头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件
a	打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写到已有内容之后。如果该文件不存在，创建新文件进行写入
+	打开一个文件进行更新（可读可写）
rb	以二进制格式打开一个文件用于只读。文件指针将会放在文件的开头。这是默认模式。一般用于非文本文件，如图片等
r+	打开一个文件用于读写。文件指针将会放在文件的开头

续表

模式	描　　述
rb+	以二进制格式打开一个文件用于读写。文件指针将会放在文件的开头。一般用于非文本文件，如图片等
wb	以二进制格式打开一个文件只用于写入。如果该文件已存在则打开文件，并从头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。一般用于非文本文件，如图片等
w+	打开一个文件用于读写。如果该文件已存在则打开文件，并从头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件
wb+	以二进制格式打开一个文件用于读写。如果该文件已存在则打开文件，并从头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。一般用于非文本文件，如图片等
ab	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写到已有内容之后。如果该文件不存在，创建新文件进行写入
a+	打开一个文件用于读写。如果该文件已存在，文件指针将会放在文件的结尾。文件打开时会是追加模式。如果该文件不存在，创建新文件用于读写
ab+	以二进制格式打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。如果该文件不存在，创建新文件用于读写

（3）encoding：指定了在处理文件时所使用的编码类型，例如，encoding="utf-8"等。指定正确的 encoding 参数以确保文件中的文本数据被正确解释。

（4）buffering：缓冲参数（可选），用于指定文件的缓冲策略。通常可以省略或用默认值。

（5）如采用方式一打开文件且操作完成后，则要使用 file.close() 来关闭文件，以释放资源和确保文件完整性。

通常采用方式二（with 语句）来自动管理文件的打开和关闭，它确保在退出 with 代码块时文件会被正确关闭（即便触发异常也可以）。使用 with 相比等效的 try-finally 代码块要简短得多。

3. file 对象

用 open()函数打开文件后，就创建了文件对象 file，file 对象的部分常用方法如表 5-2 所示。

表 5-2　file 对象的部分常用方法

方　　法	描　　述
file.close()	关闭文件。关闭后文件不能再进行读写操作
file.flush()	刷新文件内部缓冲，直接把内部缓冲区的数据立刻写入文件，而不是被动的等待输出缓冲区写入
file.fileno()	返回一个整型的文件描述符，可以用在如 os 模块的 read 方法等一些底层操作上
file.isatty()	如果文件连接到一个终端设备返回 True，否则返回 False
file.next()	返回文件下一行
file.tell()	返回文件当前位置
file.truncate([size])	截取文件，截取的字节通过 size 指定，默认为当前文件位置

5.1.3　文件的关闭

文件关闭的语法格式如下：

```
f.close()
```

说明：

通过 with 语句执行完后，或调用 f.close() 关闭文件对象后，再次使用该文件对象将会失败。

5.2　文件的读写

文件打开后就可以对文件进行读取或写入操作了。根据文件类型的不同，读取文件的方式与写入也不同。

5.2.1　文件的读取

如以文本文件方式打开文件，则文件读入的是字符串；如以二进制文件方式打开，则读入的是文件字符流。file 对象常用的文件读取方法如表 5-3 所示。

表 5-3　file 对象常用的文件读取方法

方　法	描　述
file.read([size int])	从文件读取指定的字节数，返回字符串。 如果 size int 未给定或为负，则读取所有；如果为正，则读入该行前多少个字节
file.readline([size int])	读取一整行，包括行末的换行符（"\n"）。 若给定 size int>0，则读入该行前多少个字节
file.readlines([size int])	一次性读取文件所有行，包括行末的换行符（"\n"）。 每行为一个元素构成一个列表。如果 sizeint>0，则读入该行前多少个字节
file.seek(offset[, whence])	设置文件当前位置。offset=0 为文件头，offset=2 为文件尾。

【例 5-1】 打开 files 目录下的 Aspirin.txt 文件，并显示文件内容。文件内容如图 5-1 所示。

图 5-1　Aspirin.txt 文件

【任务实现】

任务分析：用 read() 方法从文件读取所有内容，方法返回的结果是字符串，可以直接打印输出。

具体程序代码和运行结果如图 5-2 所示。

```
# 以只读方式打开文件，文件编码方式为 utf-8
f=open("files/Aspirin.txt",'r',encoding="utf-8")
# 一次性读入所有内容，返回字符串
s = f.read()
print(s)
# 关闭文件
f.close()
```

（a）程序代码

```
>>>
========== RESTART: D:\MyPython\eg5-1.py ==========
Aspirin
阿司匹林
1897
https://www.bayer.com/en/products/aspirin
>>>
```

（b）运行结果

图 5-2　ccmu.txt 文件读取并显示

【例 5-2】 打开 files 目录下的 bj.csv 文件，将各个城区名称存到列表中。文件内容如图 5-3 所示。

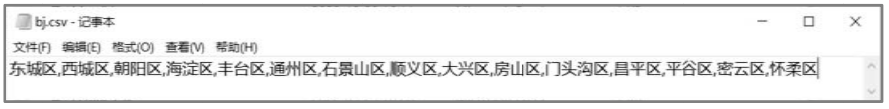

图 5-3　ccmu1.csv 文件

【任务实现】

任务分析：CSV 文件是一种常见的文本文件格式，用于存储表格数据，其中数据以逗号（或其他分隔符）分隔列，如图 5-3 所示。一般双击打开 CSV 文件时，系统会自动用 Excel 软件打开，但可以用记事本打开看原始文件内容。

因为要将文件内容中的各个城区名称存到列表中，所以可以用 read() 方法从文件读取所有内容。读取的字符串以“,”为分隔符分割出各个元素组成一个列表。

具体程序代码如图 5-4 所示。

```
# 打开文件
with open("files/bj.csv",'r',encoding="utf-8") as f:
    s = f.read() # 读取文件内容为字符串

# 以“,”为分隔符分割出元素组成一个列表
ls = s.split(',')
# 关闭文件
f.close()
```

图 5-4　bj.csv 文件读取并存储为列表

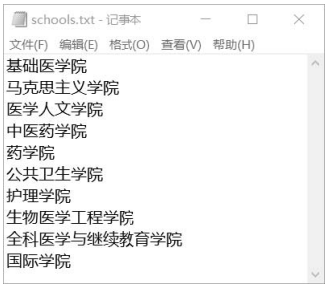

图 5-5　schools.txt 文件

【例 5-3】 打开 files 文件夹下的 schools.txt 文件，将各个学院名称存到列表中。文件部分内容如图 5-5 所示。

【任务实现】

任务分析：因为要将文件内容中的各个学院名称存到列表中， 所以可以用 readline()方法从文件逐行读取内容。此外，由于 readline()方法返回结果包括行末的换行符（'\n'），因此需要使用 strip()方法来去除末尾的换行符再存入列表中。

具体程序代码如图 5-6 所示。

```
# 打开文件
with open('files/schools.txt', 'r', encoding="utf-8") as file:
    lines = []                          # 创建一个空列表来存储每行内容
    while True:
        line = file.readline()          # 读取一行
        if not line:
            break                       # 如果读取到文件末尾，跳出循环
        lines.append(line.strip())      # 用 strip()方法去除换行符并将行添加到列表中
```

图 5-6　用 readline()方法读取 ccmu_school.txt 文件并存储为列表

【例 5-4】 用 readlines() 方法来实现例 5-3。

任务分析：readlines() 方法一次读取所有内容并存到列表中，但每一行行末的换行符（'\n'）都保存下来了。所以，可以在读取之后再各个列表元素用 strip()方法去除末尾的换行符再存入新的列表中。具体程序代码如图 5-7 所示。

```
# 打开文件
with open('files/schools.txt', 'r', encoding="utf-8") as file:
    lns = file.readlines()

# 创建一个空列表来存储每行内容
lines = []
# 逐行/元素 用 strip()方法去除换行符并添加到新列表中
for line in lns:
    lines.append(line.strip())
```

图 5-7　用 readlines()方法读取 school.txt 文件并存储为列表

5.2.2　文件的写入

file 对象常用的文件写入如表 5-4 所示。

表 5-4　file 对象常用的文件写入方法

方　　法	描　　述
file.write(str)	将字符串写入文件，返回的是写入的字符长度
file.writelines(sequence)	向文件写入一个序列字符串列表，如果需要换行则要自己加入每行的换行符

【例 5-5】 在 bj.csv 文件末尾中，添加一个“延庆区”。

【任务实现】

任务分析：因为要添加数据，所以使用“a”模式来打开文件，然后使用 write() 方法将新数据添加到文件的末尾。具体程序代码如图 5-8 所示。

```
with open('files/bj.csv', 'a', encoding="utf-8") as file:
    str = ',延庆区'
    file.write(str)
```

图 5-8　文件添加数据

【例 5-6】 2023 年 8 月 19 日中国医师节主题：“勇担健康使命，铸就时代新功”。将这个主题写入一个新文件，保存名为 eg5-6.txt，文件内容如图 5-9 所示。

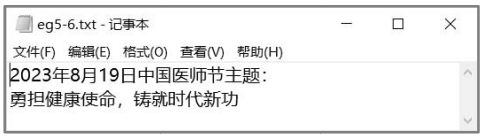

图 5-9 新文件内容

【任务实现】

任务分析：如果该文件不存在，则创建新文件；如果该文件已存在，则覆盖以前生成的文件。所以使用“w”模式来打开文件。另外，由于分两行写入，则需要自己加入每行的换行符“\n”。具体程序代码如图 5-10 所示。

【说明】

由于没有打开文件时，没有指定编码方法，新建文件的编码方式为 ANSI。ANSI 编码通常采用的编码是“encoding='cp1252'”，是特定于 Windows 操作系统的，而在跨平台和国际化环境中，更推荐使用 Unicode 编码，如 UTF-8 或 UTF-16，以处理不同语言和字符集的文本。

```
ls = ['2023年8月19日中国医师节主题：\n', '勇担健康使命，铸就时代新功']
with open('files/output/eg5-6.txt', 'w') as file:
    file.writelines(ls)
```

图 5-10 新建文件写入数据

【例 5-7】 将例 3-14 中的 5 位测试者的基本数据（见表 3-1）写入文件中，保存名为 eg5-7.csv，文件内容如图 5-11 所示。

图 5-11 新数据文件内容

【任务实现】

任务分析：使用“w”模式来打开文件。由于数据内容为二维列表，所以采用循环方式将数据逐行写入文件中。具体程序代码如图 5-12 所示。

```
# 构建数据集列表： 性别，体重，身高，年龄，平日活动强度
data =[ [202301,0, 55.5, 165, 18, 2],  [202302,1, 80.5, 185, 18, 4],
       [202303,0, 50.2, 161.3, 23, 1],[202304,1, 72.3, 175.4, 19, 3],
       [202305,1, 76.6, 178.5, 20, 5] ]
# 打开文件
with open('files/output/eg5-7.csv', 'w') as file:
    for row in data:
        # 将子列表转换为逗号分隔的字符串
        row_str = ','.join([str(num) for num in row])
        file.write(row_str + '\n')  # 写入每个子列表字符串并添加换行符
```

图 5-12 创建新数据文件

5.3　文件夹的操作

在 Python 的文件使用中，经常需要对文件夹进行创建、删除、检查存在性、获取文件夹内容等操作。可使用 Python 标准库中的 os 模块来实现这些功能。

5.3.1　os 模块

os 全称为 Operating System，os 模块是 Python 标准库中的一个核心模块。os 模块提供了与操作系统交互的各种方法，例如，创建、删除、移动文件或目录，获取当前工作目录，运行命令行程序等。在用到 os 模块相应的方法之前，要先导入该库，即“import os”。

1. 目录及文件操作

os 模块中包含了一些目录及文件操作相关的方法，常用的方法如表 5-5 所示。

表 5-5　常用的目录及文件操作方法

方　　法	描　　述
os.getcwd()	返回当前工作目录
os.chdir(path)	改变当前工作目录
os.listdir(path)	返回 path 文件夹中包含的文件或文件夹名字的列表
os.mkdir(path)	创建单层目录
os.makedirs(path)	递归创建多层目录
os.rmdir(path)	删除空目录
os.removedirs(path)	递归删除多层空目录
os.remove(path)	删除文件
os.rename(src, dst)	重命名文件或目录
os.walk(path, topdown =True)	遍历目录树，返回一个三元组，包括当前文件夹的路径（root）、当前文件夹中的子文件夹列表（dirs）和当前文件夹中的文件列表（files）。 其中，若 topdown=True，遍历顺序是自顶向下，先返回根文件夹，然后是子文件夹；若 topdown=False，则自底向上，先返回叶子文件夹，然后是根文件夹

2. 文件路径操作

os.path 是 os 模块的一个子模块，用于路径操作，如查询文件属性，检查文件是否存在以及执行各种路径相关的操作。常用的方法如表 5-6 所示。

表 5-6　常用的文件路径方法

方　　法	描　　述
os.path.join(path, *paths)	接路径的各个部分拼接成一个完整的路径
os.path.exists(path)	检查文件或目录是否存在
os.path.isdir(path)	检查指定路径是否是目录
os.path.isfile(path)	检查指定路径是否是文件
os.path.abspath(path)	返回指定路径的绝对路径
os.path.dirname(path)	返回路径中的目录部分

续表

方　法	描　述
os.path.basename(path)	返回路径中的文件名部分
os.path.split(path)	返回目录和文件名的元组

5.3.2 文件夹操作应用

【例 5-8】 显示文件夹下所有的文件夹或文件名。

【任务实现】具体程序代码如图 5-13 所示。

```
import os

# 指定文件夹的路径
folder_path = 'D:/MyPython/files'
# 获取文件夹下所有的文件夹或文件名
file_list = os.listdir(folder_path)

# 遍历文件列表并打印文件名
for file in file_list:
    print(file)
```

图 5-13　显示文件夹下所有的文件夹或文件名

【例 5-9】 批量创建文件夹。给 schools.txt 文件中的名称，创建相应的文件夹。

【任务实现】 具体程序代码如图 5-14 所示。

```
import os

# 指定根文件夹路径
root_folder = "D:/MyPython/files/output/eg5-9"

# 打开文件
with open('files/schools.txt', 'r', encoding="utf-8") as file:
    # 逐行读取文件内容
    for line in file:
        folder_name = line.strip()          # 去除行尾的换行符
         # 构建新文件夹完整路径
        folder_path = os.path.join(root_folder, folder_name)
        if not os.path.exists(folder_path):  # 如果该文件夹不存在，则创建文件夹
            os.makedirs(folder_path)
```

图 5-14　批量创建文件夹

【例 5-10】 批量删除某文件夹中空的子文件夹。

【任务实现】只删除空文件夹，所以要先判断是否为空文件夹。具体程序代码如图 5-15 所示。

【思考题】

程序中为什么是“topdown=False”，而不是“topdown=True”？

```
import os

# 自定义函数
def delete_empty_folders(folder_path):
    for root, dirs, files in os.walk(folder_path, topdown=False):
        for dir in dirs:
            dir_path = os.path.join(root, dir)
            if not os.listdir(dir_path):
                os.rmdir(dir_path)
                print(f"已删除空文件夹: {dir_path}")

# 指定文件夹路径
folder_path = "D:/MyPython/files/output"

# 调用函数删除空文件夹
delete_empty_folders(folder_path)
```

图 5-15　批量删除空的子文件夹

【例 5-11】 将例 4-7 所计算实现的输出数据按 CSV 格式保存到 output 文件夹下的“eg5-11.csv”文件中，文件内容如图 5-16 所示。

```
编号,性别,体重,身高,年龄,平日活动强度,基础代谢率,每日能量消耗
202301,0,55.5,165,18,2,1335.25,1835.96875
202302,1,80.5,185,18,4,1876.25,3236.53125
202303,0,50.2,161.3,23,1,1234.125,1480.95
202304,1,72.3,175.4,19,3,1729.25,2680.3375
202305,1,76.6,178.5,20,5,1786.625,3394.5874999999996
```

图 5-16　新 CSV 文件内容

【任务实现】

具体程序代码如图 5-17 所示。

```
# 自定义函数，函数参数 5 个：性别、体重、身高、年龄、平日活动强度，
# 函数返回结果 2 个：基础代谢率 BMR、每日能量消耗 TDEE
def CalBMR(gender, weight, height, age, atype=1):
    BMR = 0
    TDEE = 0
    # 先按男性公式，计算基础代谢率
    BMR = 10 * weight + 6.25 * height - 5 * age + 5
    # 如果是女生，则在计算数据上做对应调整
    if gender == 0:
        BMR = BMR – 166
    # 计算每日能量消耗
    match atype:
        case 1:  # 久坐或基本不运动
            TDEE = 1.2 * BMR
        case 2:  # 轻度活动
            TDEE = 1.375 * BMR
        case 3:  # 中度活动
            TDEE = 1.55 * BMR
        case 4:  # 积极活动
            TDEE = 1.725 * BMR
        case 5:  # 高强度活动
            TDEE = 1.9 * BMR
        case _:  # 其他
            TDEE = 0
```

图 5-17　计算数据并保存