01

第 1 章

使用 Python 读写 Excel 文件

Excel 是办公人员经常使用的软件，主要用来做数据的统计与分析。

虽然 Excel 的功能已经很强大了，但有时我们可能会遇上需从数百或更多电子表格中依条件复制一些数据到其他表格，或是从数百或更多数据表中搜寻符合特定条件的数据。使用 Excel 只能按部就班一步一步完成，但使用 Python 却可以轻松完成这类工作，本书就是讲解如何使用 Python 更加轻松地操作工作表、工作簿，实现高效办公。

1-1 前期准备工作

本书的讲解是基于读者已经有 Python 基础，如果读者不熟悉 Python，建议可以阅读笔者所著的 Python 书籍《Python 王者归来（增强版）》。

1-2 使用 Python 操作 Excel 的模块说明

本章内容需要使用外挂模块 openpyxl，读者可提前下载此模块，下载指令如下：

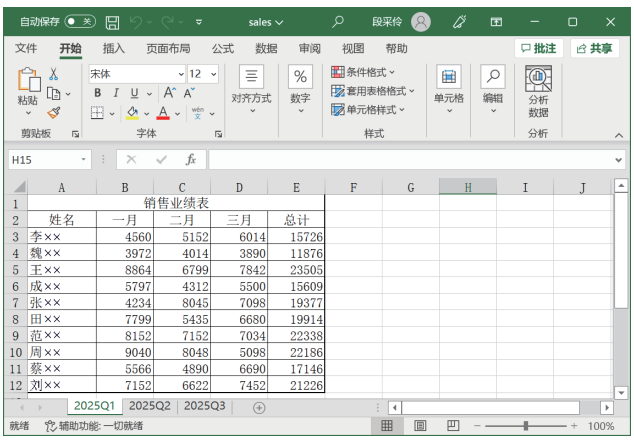
```
pip install openpyxl
```

程序导入方法如下：

```
import openpyxl
```

1-3 认识 Excel 窗口

下列是 Microsoft Excel 窗口。



Python玩转Excel：轻松实现高效办公 3校 正文.indd 2 2023/9/13 19:20:13

Microsoft Excel 文件的扩展名是 xlsx，下列是一些基本名词。

工作簿 (workbook)：Excel 的文件又称工作簿。

工作表 (worksheet)：一个工作簿是由不同数量的工作表组成，若以上图为例，是由 2025Q1、2025Q2、2025Q3 这 3 个工作表所组成，其中 2025Q1 底色是白色，表示这是目前工作表 (active sheet)。

列 (column)：工作表的列名是 A、B 等。

行 (row)：工作表的行名称是 1、2 等。

单元格 (cell)：工作表内的每一个格子称单元格，用 (列名 , 行名) 代表。

1-4 读取 Excel 文件

在本书 ch1 文件夹有 sales.xlsx，本节主要以此文件为实例解说。

1-4-1 开启文件

当我们导入 openpyxl 模块后，可以使用 openpyxl.load_workbook( ) 方法开启 Excel 文件。此函数语法如下：

```
wb = openpyxl.load_workbook(filename, read_only=FALSE,
    data_only=False, keep_vba=KEEP_VBA)
```

上述函数参数意义如下：

- filename：所读取的 Excel 文件名。
- read_only：设定是否只读，也就是只能读取不可编辑，预设是 False，表示所开启的 Excel 文件可以读写，如果设为 True 表示所开启的 Excel 文件只能读取无法更改。
- data_only：设定含公式的单元格是否具有公式功能，默认是 False，表示含公式的单元格仍具有公式功能。
- keep_vba：保存 Excel VBA 的内容，预设是保存。

上述函数可以回传工作簿对象 (也可称 Excel 文件对象)wb，本章将用 wb 变量代表 workbook 工作簿文件对象，当然读者也可以使用其他名称。

程序实例 ch1_1.py：开启 sales.xlsx 文件，然后列出回传 Excel 工作簿文件对象的文件类型。

```
1  # ch1_1.py
2  import openpyxl
3
4  fn = 'sales.xlsx'
5  wb = openpyxl.load_workbook(fn)      # wb是Excel 文件对象
6  print(type(wb))
```

执行结果

```
========== RESTART: D:\Python_Excel\ch1\ch1_1.py ==========
<class 'openpyxl.workbook.workbook.Workbook'>
>>>
```

1-4-2 取得工作表 worksheet 名称

延续前一小节，有了工作簿 wb 对象后，可以使用下列方式获得工作簿的相关信息。

Python玩转Excel：轻松实现高效办公 3校 正文.indd 3　2023/9/13 19:20:13

wb.sheetnames：所有工作表。

wb.active：目前工作表。

wb.active.title：目前工作表名称。

程序实例 ch1_2.py：列出 sales.xlsx 工作簿文件所有的工作表、目前工作表和目前工作表的名称。

```
# ch1_2.py
import openpyxl

fn = 'sales.xlsx'
wb = openpyxl.load_workbook(fn)      # wb是Excel 文件对象
print("所有工作表      = ", wb.sheetnames)
print("目前工作表      = ", wb.active)
print("目前工作表名称 = ", wb.active.title)
```

执行结果

```
========== RESTART: D:\Python_Excel\ch1\ch1_2.py ==========
所有工作表      = ['2025Q1', '2025Q2', '2025Q3']
目前工作表      = <Worksheet "2025Q1">
目前工作表名称 = 2025Q1
```

上述程序获得了所有的工作表、目前工作表和目前工作表名称。其实在开启 Excel 文件后，最左边的工作表是预设的工作表，如下所示：

所以程序实例 ch1_2.py 的第 6 行和第 7 行的输出，才是上述执行结果。

程序实例 ch1_2_1.py：使用循环列出 sales.xlsx 工作簿所有的工作表名称。

```
# ch1_2_1.py
import openpyxl

fn = 'sales.xlsx'
wb = openpyxl.load_workbook(fn)      # wb是Excel 文件对象
print("所有工作表      = ", wb.sheetnames)
for sheet in wb.sheetnames:
    print("工作表名称 = ", sheet)
```

执行结果

```
========== RESTART: D:\Python_Excel\ch1\ch1_2_1.py ==========
所有工作表      = ['2025Q1', '2025Q2', '2025Q3']
工作表名称 = 2025Q1
工作表名称 = 2025Q2
工作表名称 = 2025Q3
```

1-5 切换工作表对象

1-5-1 直接使用工作表名称

对于 wb 目前工作簿对象而言，可以使用下列语法切换工作表对象。

```
ws = wb[sheetname]    # 假设工作表对象名称是 ws
```

以目前实例而言，可以使用列表的参数，在此可以称索引，切换特定的工作表对象，下列指令可以切换获得 2025Q3 工作表对象。

```
ws = wb['2025Q3']
```


Python玩转Excel：轻松实现高效办公 3校 正文.indd 4 2023/9/13 19:20:13

此例 ws 是工作表对象，可以使用 title 属性列出实际工作表名称。

程序实例 ch1_3.py：重新设计 ch1_2.py，将 title 属性应用在 ws 对象，取得特定工作表对象 wb['2025Q3'] 的名称。

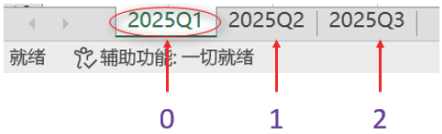
```
# ch1_3.py
import openpyxl

fn = 'sales.xlsx'
wb = openpyxl.load_workbook(fn)
print("预设的工作表名称 = ", wb.active.title)
ws = wb['2025Q3']      # 设定特定工作表的名称
print("特定工作表的名称 = ", ws.title)
```

执行结果

```
================== RESTART: D:\Python_Excel\ch1\ch1_3.py ==================
预设的工作表名称 =  2025Q1
特定工作表的名称 =  2025Q3
```

1-5-2　使用 worksheets[n] 切换工作表

openpyxl 模块的 wb 对象，也允许使用 worksheets[n] 属性切换工作表，此 n 代表工作表编号，此编号是从 0 开始编号，概念如下：



程序实例 ch1_4.py：使用 worksheets[n] 切换显示工作表名称。

```
# ch1_4.py
import openpyxl

fn = 'sales.xlsx'
wb = openpyxl.load_workbook(fn)
print("预设的工作表名称 = ", wb.active.title)
ws0 = wb.worksheets[0]
ws1 = wb.worksheets[1]
ws2 = wb.worksheets[2]
print("特定工作表的名称 = ", ws0.title)
print("特定工作表的名称 = ", ws1.title)
print("特定工作表的名称 = ", ws2.title)
```

执行结果

```
================== RESTART: D:\Python_Excel\ch1\ch1_4.py ==================
预设的工作表名称 =  2025Q1
特定工作表的名称 =  2025Q1
特定工作表的名称 =  2025Q2
特定工作表的名称 =  2025Q3
```

1-6　写入 Excel 文件

openpyxl 模块也提供了方法可以让我们写入 Excel 文件。

Python玩转Excel：轻松实现高效办公 3校 正文.indd 5　　2023/9/13 19:20:14

1-6-1 建立空白工作簿

openpyxl.Workbook( ) 可以建立空白的工作簿对象，也可想成 Excel 文件对象，此函数的语法如下：

```
wb = openpyxl.Workbook(write_only=False)
```

上述函数回传 wb 工作簿对象，默认所建立的文件对象是可擦写，如果想要设为只写模式，可以加上 write_only=True 参数。

1-6-2 存储 Excel 文件

save( ) 方法可以存储 Excel 工作簿文件，这个方法需由 Excel 文件对象启动，先前我们是使用 wb 当作文件对象的变量，这时可以使用 active 获得目前工作表对象，概念如下：

```
ws = wb.active                          # 获得目前工作表对象
```

有了 ws 工作表对象，可以使用 title 属性获得或是设定工作表名称，如下所示：

```
ws.title                                # 目前工作表名称
```

假设想要将目前工作表名称改为“My sheet”，可以使用下列指令：

```
ws.title = 'My sheet'
```

要存储目前工作簿文件可以使用下列语法：

```
wb.save( 文件名 )                       # 可以存储指定文件名的文件
```

或是：

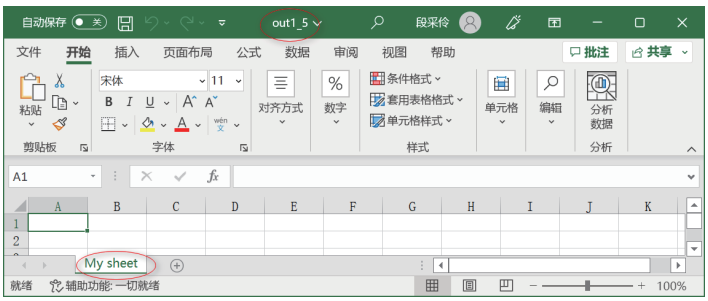
```
wb.save(filename = 文件名 )
```

程序实例 ch1_5.py：建立一个空白的 Excel 文件，列出预设的工作表名称，然后将预设工作表名称改为“My sheet”，最后用 out1_5.xlsx 名称存储此文件。

```
1  # ch1_5.py
2  import openpyxl
3
4  wb = openpyxl.Workbook()                # 建立空白的工作簿
5  ws = wb.active                          # 获得目前工作表
6  print("目前工作表名称 = ", ws.title)     # 打印目前工作表
7  ws.title = 'My sheet'                   # 更改目前工作表名称
8  print("新工作表名称   = ", ws.title)     # 打印新的目前工作表
9  wb.save('out1_5.xlsx')                  # 将工作簿存储
```

执行结果 下列是执行结果与 out1_5.xlsx 的结果。

```
========== RESTART: D:\Python_Excel\ch1\ch1_5.py ==========
目前工作表名称 =  Sheet
新工作表名称   =  My sheet
```

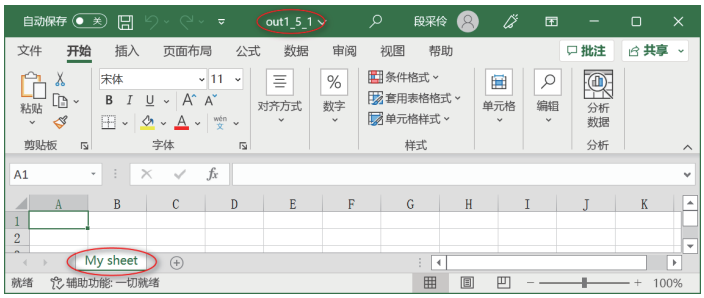


Python玩转Excel：轻松实现高效办公 3校 正文.indd 6 2023/9/13 19:20:14

程序实例 ch1_5_1.py：重新设计 ch1_5.py，使用另一种方式建立与存储工作簿文件 out1_5_1.xlsx。

```
# ch1_5_1.py
import openpyxl

wb = openpyxl.Workbook()                    # 建立空白的工作簿
ws = wb.active                              # 获得目前工作表
print("目前工作表名称 = ", ws.title)          # 打印目前工作表
ws.title = 'My sheet'                       # 更改目前工作表名称
print("新工作表名称    = ", ws.title)         # 打印新的目前工作表
fn = 'out1_5_1.xlsx'
wb.save(filename=fn)                        # 将工作簿存储
```

执行结果　与 ch1_5.py 相同，可是此程序建立了 out1_5_1.xlsx 工作簿文件。



1-6-3　复制 Excel 文件

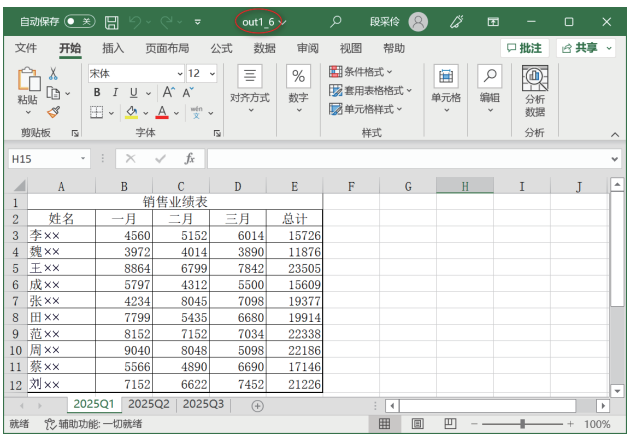
我们可以开启文件，然后用以新名称存储文件的方式复制 Excel 文件。

程序实例 ch1_6.py：将 sales.xlsx 复制一份至 out1_6.xlsx。

```
# ch1_6.py
import openpyxl

fn = 'sales.xlsx'
wb = openpyxl.load_workbook(fn)  # 开启sales.xlsx工作簿
wb.save('out1_6.xlsx')           # 将工作簿存储至out1_6.xlsx
print("复制完成")
```

执行结果　可以在目前工作文件夹看到所建立的 out1_6.xlsx 文件，文件内容与 sales.xlsx 相同。

```
========== RESTART: D:\Python_Excel\ch1\ch1_6.py ==========
复制完成
```



Python玩转Excel：轻松实现高效办公 3校 正文.indd 7　　2023/9/13 19:20:14

1-7 关闭文件

先前实例笔者使用 wb 当作工作簿对象，对于未来不再使用的工作簿对象，可以使用 close( ) 函数关闭此工作簿对象，执行 close( ) 函数后，可以将此对象所占据的内存归还系统，语法如下：

```
wb.close( )
```

如果没有执行此函数，程序也不会错，因为本书程序大多较短小，所以笔者大都省略此函数。

程序实例 ch1_6_1.py：增加 close( ) 函数，重新设计 ch1_6.py 程序。

```
1  # ch1_6_1.py
2  import openpyxl
3
4  fn = 'sales.xlsx'
5  wb = openpyxl.load_workbook(fn)  # 开启sales.xlsx工作簿
6  wb.save('out1_6_1.xlsx')         # 工作簿存储至out1_6_1.xlsx
7  print("复制完成")
8  wb.close()
```

执行结果 可以在目前工作文件夹看到所建立的 out1_6_1.xlsx 文件，文件内容与 sales.xlsx 相同。

```
================== RESTART: D:\Python_Excel\ch1\ch1_6_1.py ==================
复制完成
```

1-8 找出目前文件夹中的 Excel 文件

Python 内有一个模块 glob 可用于列出特定工作文件夹的内容 (不含子文件夹)，当导入这个模块后可以使用 glob( ) 方法获得特定工作目录的内容，这个方法最大特色是可以使用通配符“*”，例如：可用“*.xlsx”获得所有 Excel 文件。“?”可以是任意字符、“[abc]”必须是 abc 字符。更多应用可参考下列实例。

程序实例 ch1_7.py：方法 1 是列出所有特定文件夹的文件，方法 2 是列出目前文件夹中的 Excel 文件，方法 3 是列出 out1 开头的所有文件，方法 4 是使用“?”字符列出 out1_ 开头的文件 (注：out1_ 后面只限一个字符)。

```
1  # ch1_7.py
2  import glob
3
4  print("方法1:列出\\Python\\ch1文件夹的所有Excel文件")
5  for file in glob.glob('D:\\Python_Excel\\ch1\*.xlsx'):
6      print(file)
7
8  print("方法2:列出目前文件夹的Excel文件")
9  for file in glob.glob('*.xlsx'):
10     print(file)
11
12 print("方法3:列出目前文件夹out1开头的Excel文件")
13 for file in glob.glob('out1*.xlsx'):
14     print(file)
15
16 print("方法4:列出目前文件夹out1_开头的Excel文件")
17 for file in glob.glob('out1_?.xlsx'):
18     print(file)
```


Python玩转Excel：轻松实现高效办公 3校 正文.indd 8 2023/9/13 19:20:14

执行结果

```
==================== RESTART: D:\Python_Excel\ch1\ch1_7.py ====================
方法1:列出\Python\ch1文件夹的所有Excel文件
D:\Python_Excel\ch1\out1_5.xlsx
D:\Python_Excel\ch1\out1_6.xlsx
D:\Python_Excel\ch1\sales.xlsx
D:\Python_Excel\ch1\~$sales.xlsx
方法2:列出目前文件夹的Excel文件
out1_5.xlsx
out1_6.xlsx
sales.xlsx
~$sales.xlsx
方法3:列出目前文件夹out1开头的Excel文件
out1_5.xlsx
out1_6.xlsx
方法4:列出目前文件夹out1_开头的Excel文件
out1_5.xlsx
out1_6.xlsx
```

1-9 找出目前文件夹所有 out 开头的 Excel 文件

为了更有效率操作 Excel，可能我们会想要一次下载多个 Excel 文件，可以参考下列实例。

程序实例 ch1_8.py：下载目前文件夹内所有 out1 开头的 Excel 文件，同时列出这些文件的工作表。

```
1  # ch1_8.py
2  import glob
3  import openpyxl
4
5  files = glob.glob('out1*.xlsx')
6  for file in files:
7      wb = openpyxl.load_workbook(file)
8      print(f'下载 {file} 成功')
9      print(f'{file} = {wb.sheetnames}')
```

执行结果

```
==================== RESTART: D:\Python_Excel\ch1\ch1_8.py ====================
下载 out1_5.xlsx 成功
out1_5.xlsx = ['My sheet']
下载 out1_6.xlsx 成功
out1_6.xlsx = ['2025Q1', '2025Q2', '2025Q3']
```

1-10 复制所有 out1 开头的文件

在操作文件夹时，可能会想要将所有特定的 Excel 文件全部复制一份，这时可以使用复制下载，然后更改文件名。

程序实例 ch1_9.py：将所有 out1 开头的 Excel 文件名前面增加 new 字符串。

```
1  # ch1_9.py
2  import glob
3  import openpyxl
4
5  files = glob.glob('out1*.xlsx')
6  for file in files:
7      wb = openpyxl.load_workbook(file)
8      newfile = 'new' + file
9      wb.save(newfile)
10 newfiles = glob.glob('new*.xlsx')
11 print("输出复制结果")
12 for newf in newfiles:
13     print(newf)
```


Python玩转Excel：轻松实现高效办公 3校 正文.indd 9　　2023/9/13 19:20:14

执行结果

```
==================== RESTART: D:\Python_Excel\ch1\ch1_9.py ====================
输出复制结果
newout1_5.xlsx
newout1_6.xlsx
```

程序实例 ch1_10.py：将所有 out 开头的 Excel 文件，另外复制一份为 new 取代 out 开头。

```
# ch1_10.py
import glob
import openpyxl

files = glob.glob('out1*.xlsx')
for file in files:
    wb = openpyxl.load_workbook(file)
    newfile = file.replace('out','new')
    wb.save(newfile)
newfiles = glob.glob('new1*.xlsx')
print("输出复制结果")
for newf in newfiles:
    print(newf)
```

执行结果

```
==================== RESTART: D:\Python_Excel\ch1\ch1_10.py ====================
输出复制结果
new1_5.xlsx
new1_6.xlsx
```

1-11 输入关键词查找工作簿

1-11-1 目前工作文件夹

在使用 Excel 时，也可以用关键词搜寻工作簿。

程序实例 ch1_11.py：搜寻目前工作文件夹内文件名含 out 的工作簿。

```
# ch1_11.py
import glob

key = input('请输入关键词 : ')
keyword = '*' + key + '*.xlsx'  # 组成关键词的字符串
files = glob.glob(keyword)
for fn in files:
    print(fn)
```

执行结果

```
==================== RESTART: D:\Python_Excel\ch1\ch1_11.py ====================
请输入关键词 : out
newout1_5.xlsx
newout1_6.xlsx
out1_5.xlsx
out1_6.xlsx
```

1-11-2 搜寻特定文件夹

前一小节是搜寻目前工作文件夹内含 out 字符串的工作簿，读者也可以搜寻其他工作文件夹的

Python玩转Excel：轻松实现高效办公 3校 正文.indd 10 2023/9/13 19:20:15

工作簿，只要增加文件夹名称即可。

程序实例 ch1_12.py：可以参考下列实例。

```
# ch1_12.py
import glob

mydir = input('请输入指定文件夹 : ')
key = input('请输入关键词 : ')
keyword = mydir + '*' + key + '*.xlsx'
files = glob.glob(keyword)
for fn in files:
    print(fn)
```

执行结果

```
=================== RESTART: D:\Python_Excel\ch1\ch1_12.py ===================
请输入指定文件夹 : D:/Python_Excel/ch1/
请输入关键词 : out
D:/Python_Excel/ch1\newout1_5.xlsx
D:/Python_Excel/ch1\newout1_6.xlsx
D:/Python_Excel/ch1\out1_5.xlsx
D:/Python_Excel/ch1\out1_6.xlsx
```

上述输入是使用“/”区隔子文件夹，也可以使用“\”区隔子文件夹，可以参考下列执行结果。

```
=================== RESTART: D:\Python_Excel\ch1\ch1_12.py ===================
请输入指定文件夹 : D:\Python_Excel\ch1\
请输入关键字 : out
D:\Python_Excel\ch1\newout1_5.xlsx
D:\Python_Excel\ch1\newout1_6.xlsx
D:\Python_Excel\ch1\out1_5.xlsx
D:\Python_Excel\ch1\out1_6.xlsx
```

1-11-3　使用 os.walk( ) 遍历所有文件夹下的文件

Python 的 os 模块有 os.walk( ) 方法可以遍历指定文件夹下所有的子文件夹，有了这个概念我们就可以使用 os.walk( ) 方法找特定工作簿文件。这个方法每次执行循环时将回传 3 个值：

（1）目前工作文件夹名称 (dirName)。

（2）目前工作文件夹下的子文件夹列表 (sub_dirNames)。

（3）目前工作文件夹下的文件列表 (fileNames)。

下列是语法格式：

```
for dirName, sub_dirNames, fileNames in os.walk( 文件夹路径 ):
    程序区块
```

程序实例 ch1_13.py：输入指定文件夹与文件名关键词，这个程序会输出所有文件夹下相符的工作簿。

```
# ch1_13.py
import glob
import os

mydir = input('请输入指定文件夹 : ')
key = input('请输入关键词 : ')
for dirName, sub_dirNames, fileNames in os.walk(mydir):
    print(f"目前文件夹名称 : {dirName}")
    keyword = dirName + '\*' + key + '*.xlsx'
    files = glob.glob(keyword)
    for fn in files:
        print(fn)
```


Python玩转Excel：轻松实现高效办公 3校 正文.indd 11　2023/9/13 19:20:15

执行结果

```
========== RESTART: D:\Python_Excel\ch1\ch1_13.py ==========
请输入指定文件夹 : D:\Python_Excel\
请输入关键词 : out
目前文件夹名称 : D:\Python_Excel\
目前文件夹名称 : D:\Python_Excel\ch1
D:\Python_Excel\ch1\newout1_5.xlsx
D:\Python_Excel\ch1\newout1_6.xlsx
D:\Python_Excel\ch1\out1_5.xlsx
D:\Python_Excel\ch1\out1_6.xlsx
目前文件夹名称 : D:\Python_Excel\ch10
D:\Python_Excel\ch10\out10_1.xlsx
D:\Python_Excel\ch10\out10_2.xlsx
```

注 当读者执行此文件时，由于许多文件夹下皆有 *out*.xlsx 文件，所以可以看到更多搜寻结果。

Python玩转Excel：轻松实现高效办公 3校 正文.indd 12 2023/9/13 19:20:15