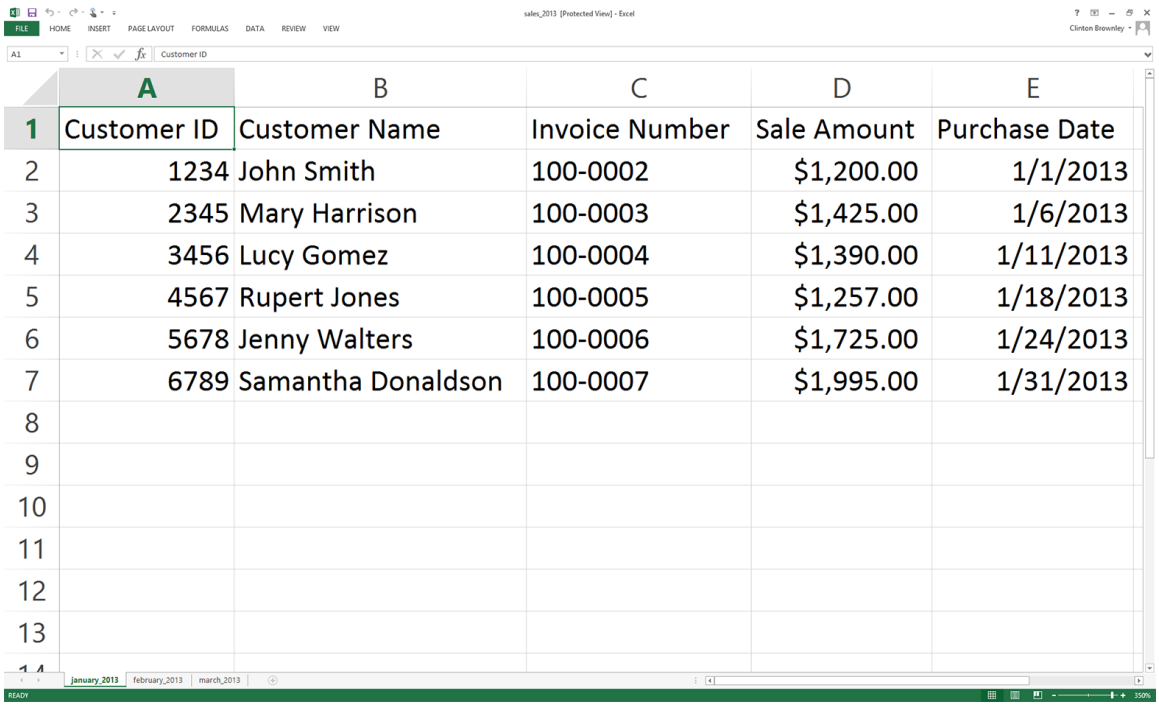
3Python 读写 Excel 文件

Microsoft Excel 文件是非常常见。我们用它来贮存客户数据，库存，员工信息。我们用它来追踪运营，销售，财务。人们用 Excel 的例子还有很多很多。因为 Excel 是如此综合的工具，学习如何用 python 处理 Excel 文件，使你能将 Python 添加入数据处理的工作流，接收别人发送的数据，与他们分享结果，以他们习惯的方式。不像 python 的 CSV 模块，没有标准的 python 模块处理 Excel 文件（如文件扩展名为.xls 或 or .xlsx）。要完成本节的例子，你要有 xlrd 和 xlwt 包。 xlrd 和 xlwt 使 python 可以在任何操作系统上处理 Excel 文件，它们强大的支持 Excel 日期。如果你安装了 Anaconda Python，你已经有这两个包了，因为它们已邦定到安装中。如果你从 Python.org 网站安装 Python，你要按附录 A 的指示下载和安装这两个包。简而言之，当我指“Excel file”时，它与“Excel workbook.”相同。Excel workbook 包含一个或多个 Excel worksheets。本章中，我交替的使用“file”和 “workbook”。我将 workbook 内的各个 worksheets 称为 worksheets。

如我们处理 CSV 文件一样，我用基础 python 运行名个例子，使你可以看到数据处理的逻辑步骤，然后用 Pandas，使你可以用更短更简洁例子，虽然有一点抽象，如果你想要拷贝和修改它来进行你的工作。

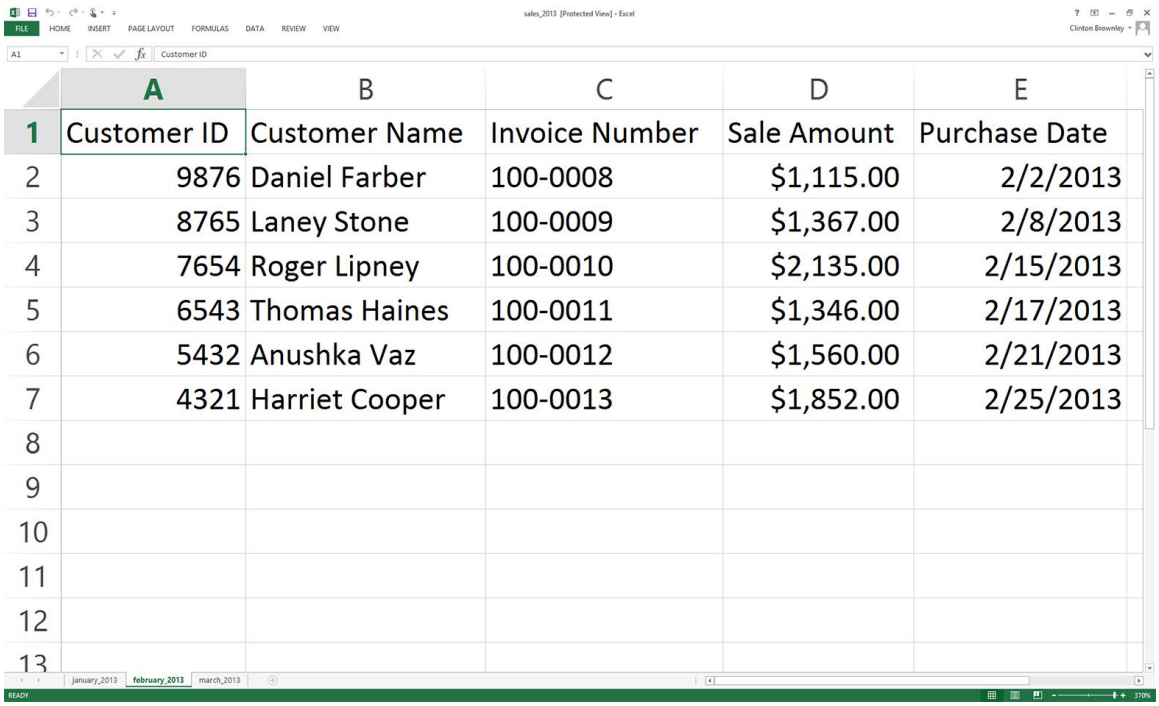
我们以创建 Excel workbook 开始本章的例子。

1. 打开 Microsoft Excel.
2. 添加三个 worksheets 到 workbook 中命名为 january_2013, february_2013, 和 march_2013. 然后按图 3-1, 3-2, 和 3-3, 添加数据。
3. 何存 workbook 为 sales_2013.xlsx.



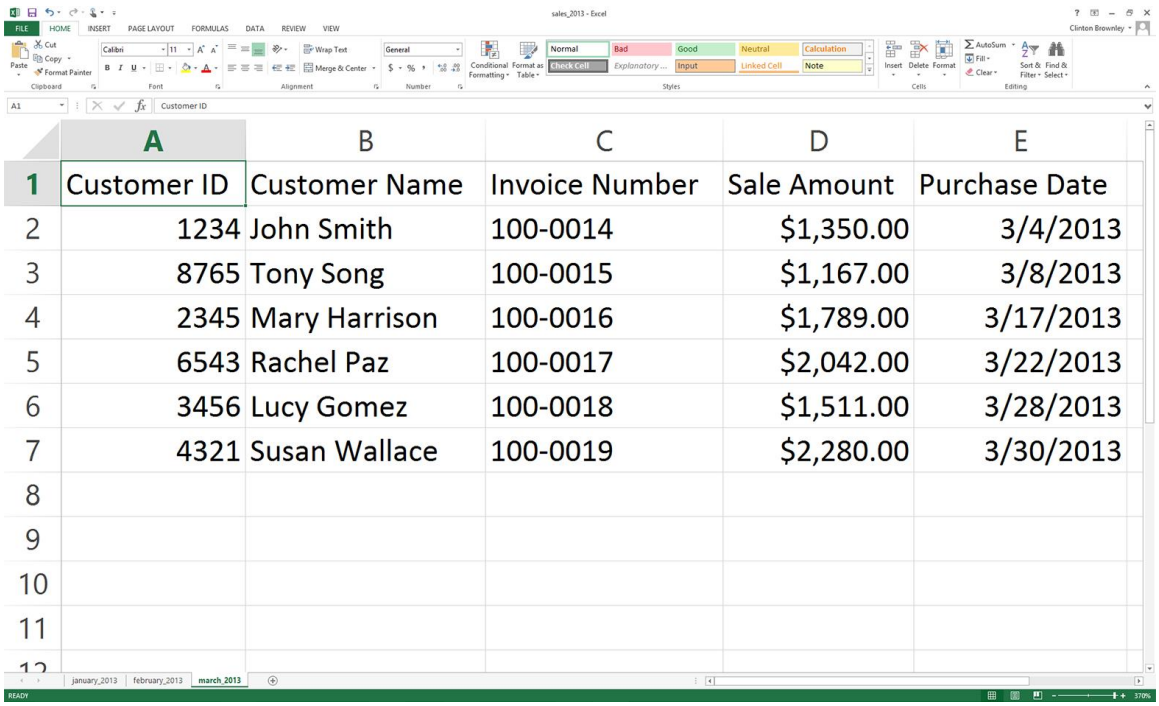
	A	B	C	D	E
1	Customer ID	Customer Name	Invoice Number	Sale Amount	Purchase Date
2	1234	John Smith	100-0002	$1,200.00	1/1/2013
3	2345	Mary Harrison	100-0003	$1,425.00	1/6/2013
4	3456	Lucy Gomez	100-0004	$1,390.00	1/11/2013
5	4567	Rupert Jones	100-0005	$1,257.00	1/18/2013
6	5678	Jenny Walters	100-0006	$1,725.00	1/24/2013
7	6789	Samantha Donaldson	100-0007	$1,995.00	1/31/2013
8					
9					
10					
11					
12					
13					

图 3-1. Worksheet 1: january_2013



	A	B	C	D	E
1	Customer ID	Customer Name	Invoice Number	Sale Amount	Purchase Date
2	9876	Daniel Farber	100-0008	$1,115.00	2/2/2013
3	8765	Laney Stone	100-0009	$1,367.00	2/8/2013
4	7654	Roger Lipney	100-0010	$2,135.00	2/15/2013
5	6543	Thomas Haines	100-0011	$1,346.00	2/17/2013
6	5432	Anushka Vaz	100-0012	$1,560.00	2/21/2013
7	4321	Harriet Cooper	100-0013	$1,852.00	2/25/2013
8					
9					
10					
11					
12					
13					

图 3-2. Worksheet 2: february_2013



	A	B	C	D	E
1	Customer ID	Customer Name	Invoice Number	Sale Amount	Purchase Date
2	1234	John Smith	100-0014	$1,350.00	3/4/2013
3	8765	Tony Song	100-0015	$1,167.00	3/8/2013
4	2345	Mary Harrison	100-0016	$1,789.00	3/17/2013
5	6543	Rachel Paz	100-0017	$2,042.00	3/22/2013
6	3456	Lucy Gomez	100-0018	$1,511.00	3/28/2013
7	4321	Susan Wallace	100-0019	$2,280.00	3/30/2013
8					
9					
10					
11					
12					

图 3-3. Worksheet 3: march_2013

探索 Excel Workbook

现在我们的 Excel workbook 包含了三个 worksheets，我们来学习一下如何用 python 处理 Excel workbook。作为备忘，我们用 xlrd 和 xlwt 包，所以确保你已安装和下载了这些包。你已经注意到 Excel 文件不同于 CSV 文件，最少有二个重要的方面。第一，不象 CSV 文件，Excel 文件不是明文文件，你无法用文本编辑器打开并观察它。你可以右击刚才创建的 Excel workbook 并用文本编辑器如 Notepad 或 TextWrangler 打开它。你看到的不是可读的数据，而是特列的符号。第二，不像 CSV 文件， Excel workbook 包含多个 worksheets。由于一个 workbook 可以包含多个 worksheets，我们要学习检查 workbook 内的多个 worksheets 不用手工打开 workbook。通过检查 workbook，我们可以检查 worksheets 的数量，每个 worksheets 数据的类型和数目，然后才处理 workbook 的数据。检查 Excel 文件有助于你保证它们含有你想要的数据，检查一致性和完整性。即理解输入文件的数目，每个文件的数据的行和列数，你可以了解工作量和文件的布局是否一致。一旦你理解了如何检查 workbook 中 worksheets,我们可以解析一个 worksheet,遍历多个 worksheets， 遍历多个 workbooks。

要检查 workbook 中 worksheets 的数目，名字，行数和列数，在文件编辑器中输入如下代码并将文件保存为 1excel_introspect_workbook.py:

```
1 #!/usr/bin/env python3
2 import sys
3 from xlrd import open_workbook
4 input_file = sys.argv[1]
5 workbook = open_workbook(input_file)
6 print('Number of worksheets:', workbook.nsheets)
7 for worksheet in workbook.sheets():
8 print("Worksheet name:", worksheet.name, "\tRows:",\
9 worksheet.nrows, "\tColumns:", worksheet.ncols)
```

图 3-4， 3-5， 和 3-6 展示脚本在 Anaconda Spyder, Notepad++ (Windows), and TextWrangler (macOS)中看起来的样子。

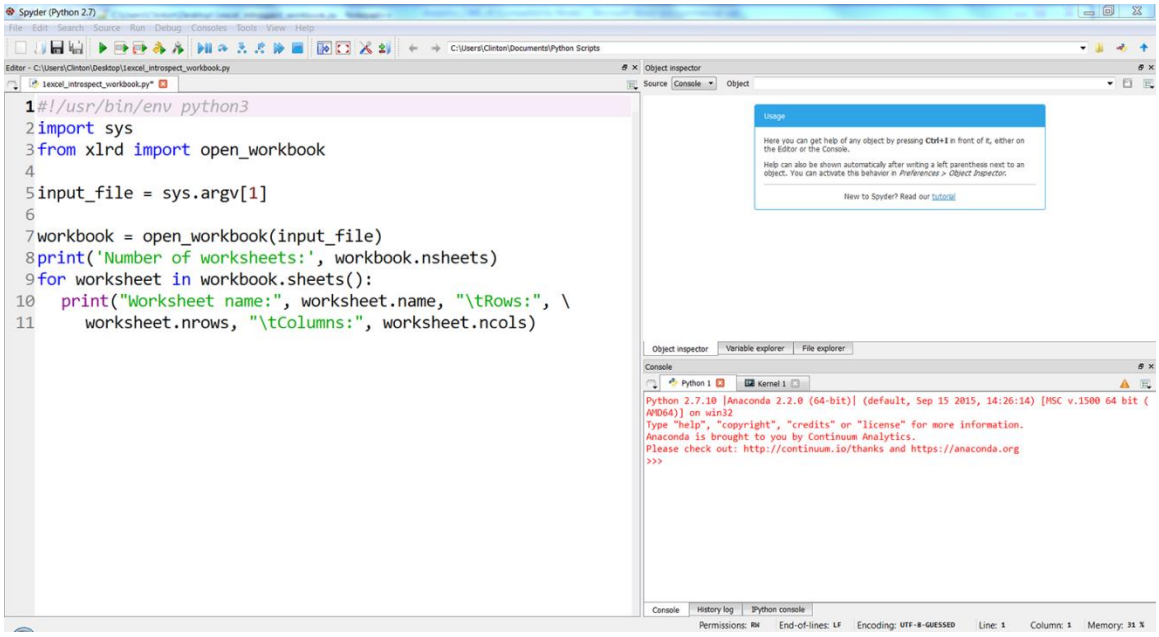


图 3-4. The Python 脚本 1excel_introspect_workbook.py 在 Anaconda Spyder 中

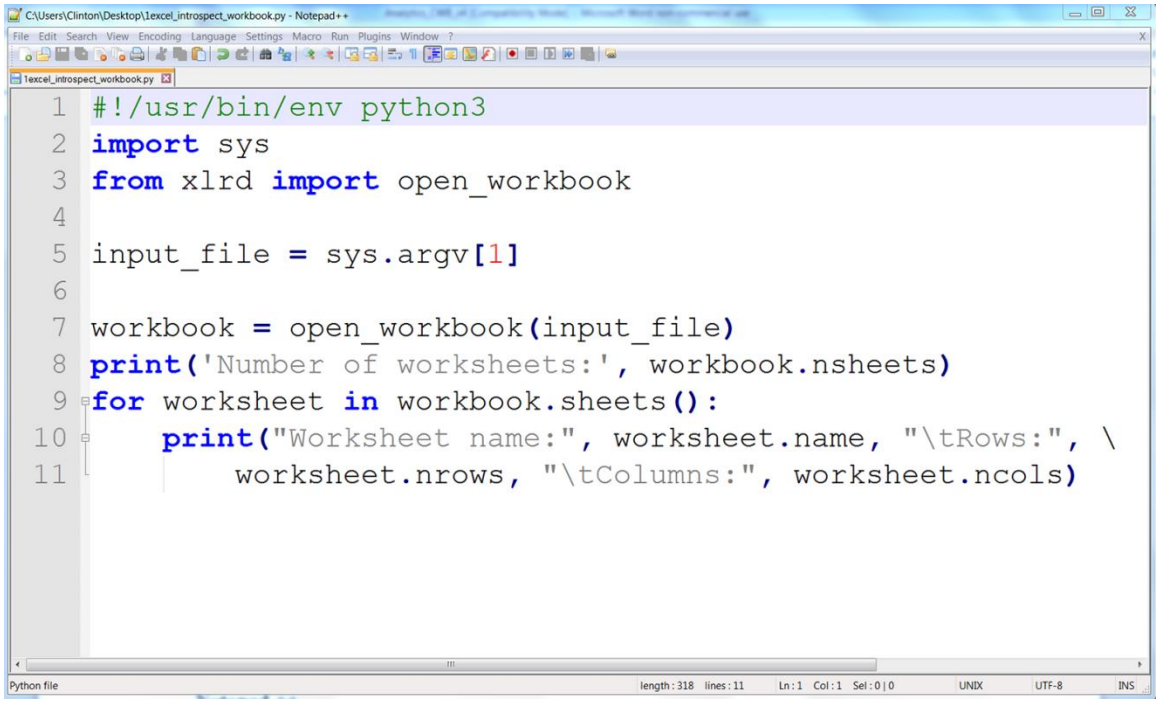


图 3-5. The Python 脚本 1excel_introspect_workbook.py 在 Notepad++ (Windows)中

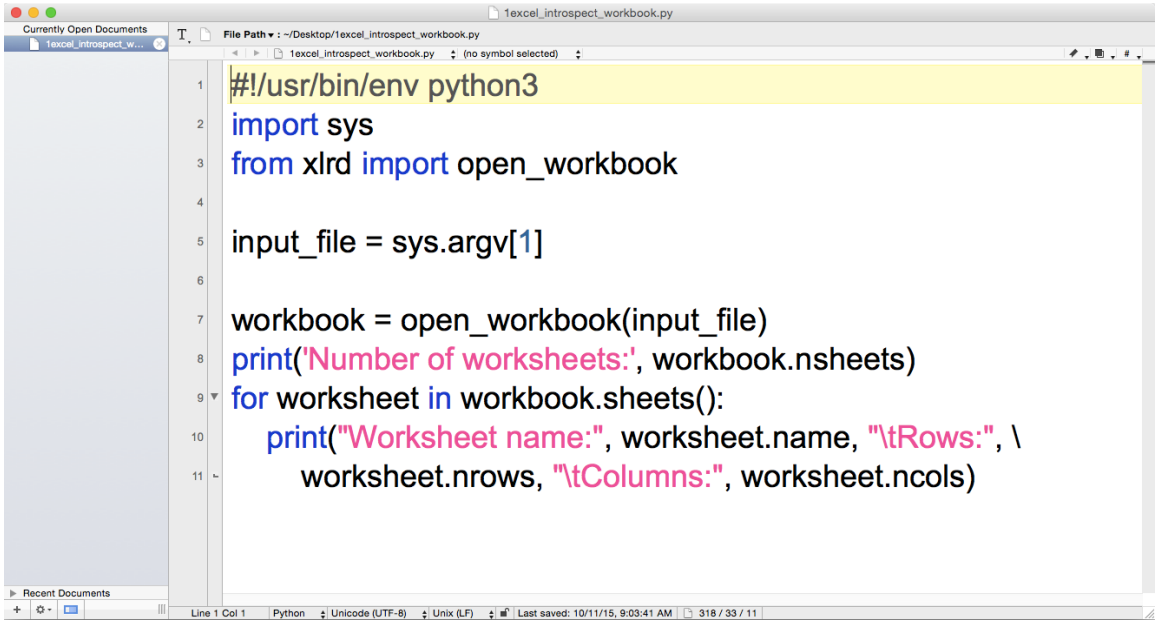


图 3-6. Python 脚本 1excel_introspect_workbook.py 在 TextWrangler (macOS)中

第 3 行导入 xlrd 模块的 open_workbook 函数以便我们可以用它来读取和解析 Excel 文件。第 7 行用 open_workbook 函数打开输入文件到对象 workbook。workbook 对象包含所有关于 workbook 的可用的信息，以便我们可以追踪 workbook 中的各个 worksheets。第 8 行打印 workbook 中的 worksheets 数目。第 9 行是 for 循环，遍历 workbook 中所有的 worksheets。Workbook 对象的 sheets 方法识别 workbook 中的所有 worksheets。第 10 行打印每个 worksheet 的名字，以及行数和列数到屏幕上。打印语句用 worksheet 对象的 name 属性识别 worksheet 的名字。相似的，它用 nrows 和 Ncols 属性识别行数和列数。

如果你在 Spyder IDE 中创建文件，则运行脚本：

1. 点击 IDE 左上角的 Run 下拉菜单 。
2. 选择“Configure”
3. 当 Run 的 Settings 窗口打开后,选择“Command line options”复选框并输入“sales_2013.xlsx” (如图 3-7)。
4. 确保“Working directory” 是你保存脚本和 Excel 文件的地方。
5. 点击 Run 按钮。

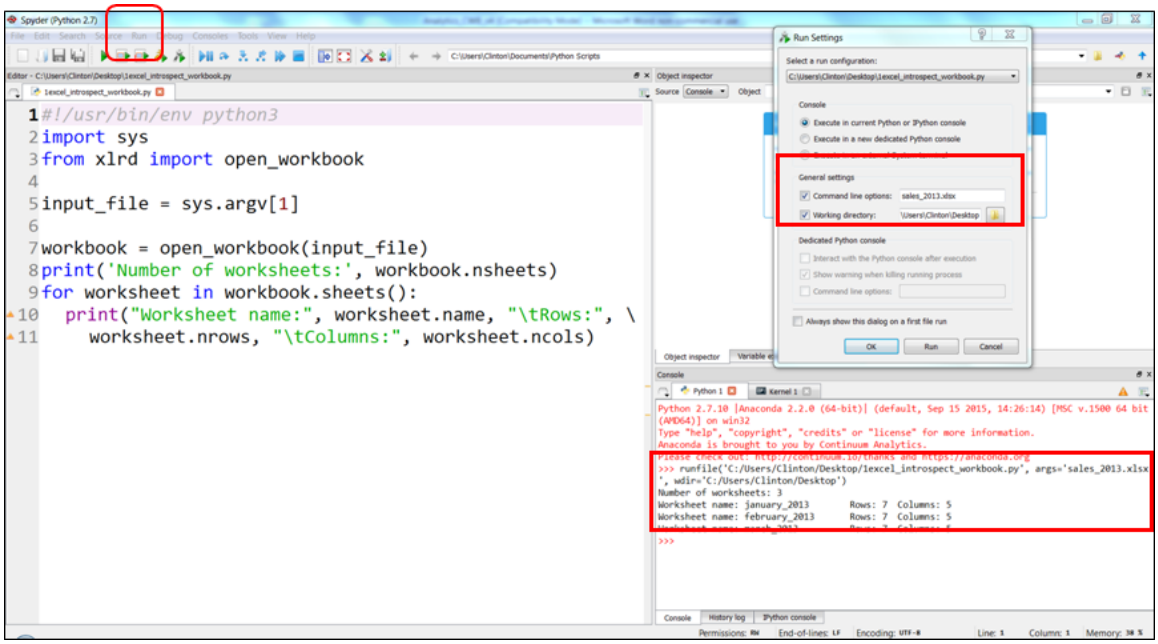


图 3-7. 在 Anaconda Spyder 中指明命令行选项

当你点击 Run 按钮时（Run Settings 窗口或 IDE 左上角的绿色 Run 按钮）你可以看到 IDE 右下方的 Python 命令行输出。图 3-7 显示 Run 下拉菜单，Run Settings 窗口的关键设置，和红色框中的输出。可选地，你可以在命令行窗口或终端窗口中运行脚本。使用以下的命令，取决于你的操作系统。

Windows 系统：

```
python lexcel_introspect_workbook.py sales_2013.xlsx
```

macOS 系统：

```
chmod +x lexcel_introspect_workbook.py
./lexcel_introspect_workbook.py sales_2013.xlsx
```

你可以看到输出如图 3-8（Windows 系统）图 3-9（macOS）所示打印到屏幕。

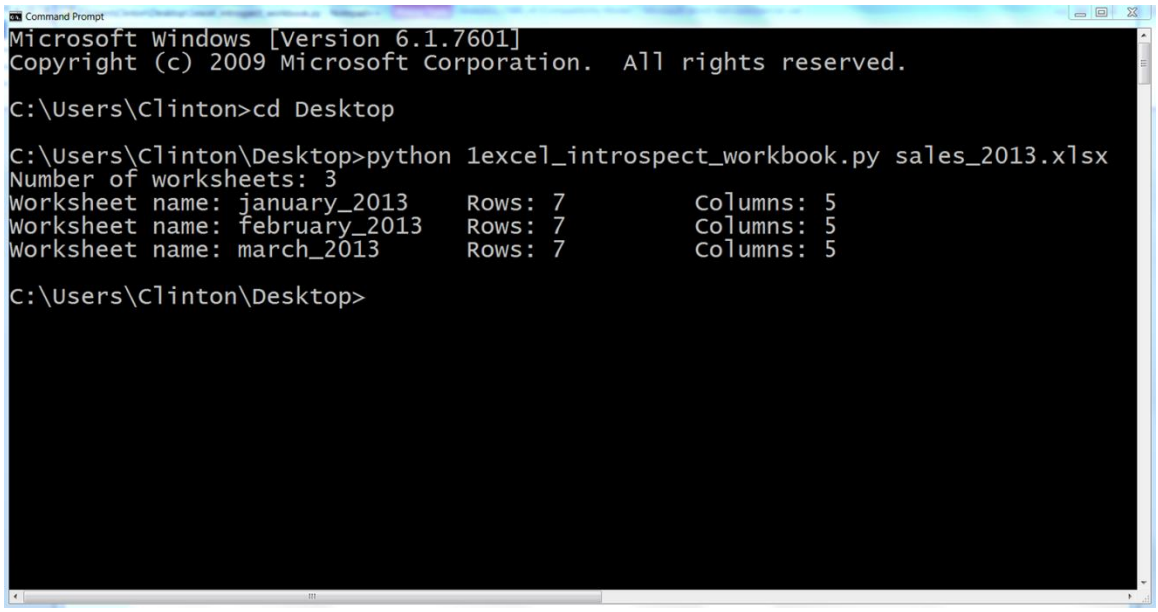


图 3-8. 脚本命令行窗口的输出（Windows 系统）

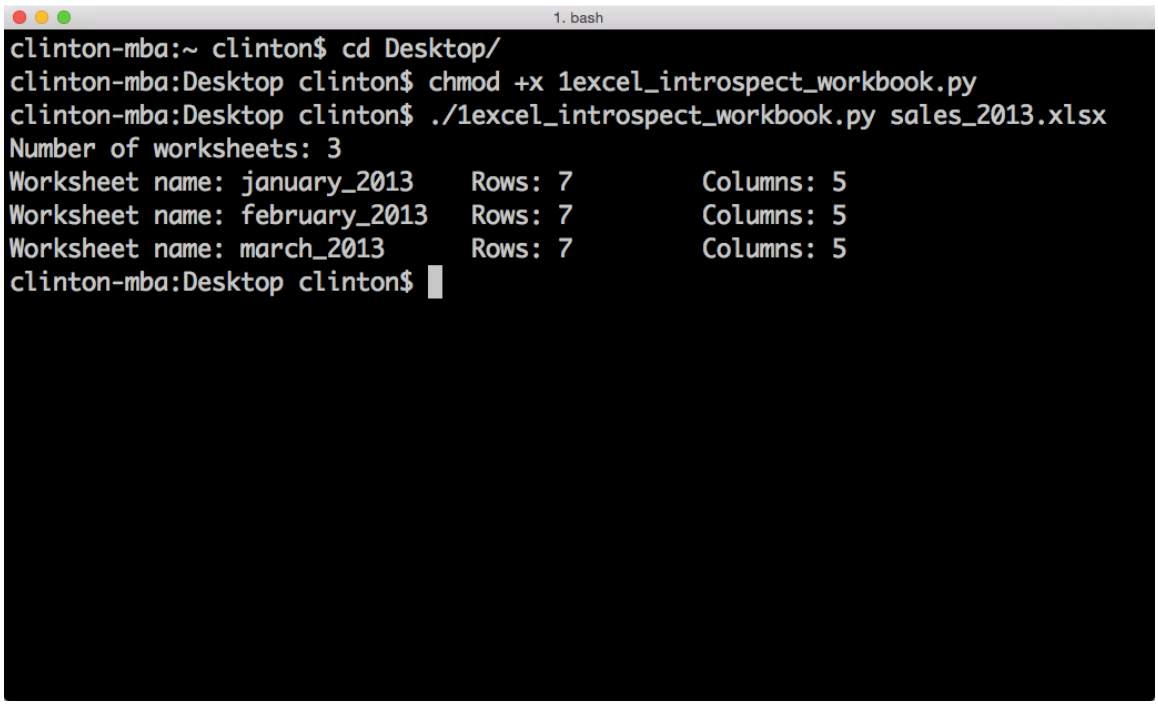


图 3-9. 脚本在终端窗口的输出（macOS）

第一行输出显示 Excel 输入文件 sales_2013.xlsx 包含三个 worksheets。下三行显示三个 worksheets 名为 january_2013，february_2013，以及 march_2013。它们也显示每个 worksheets 含 7 行，包括标题行，5 列。现在我们知道如何用 Python 来检查 Excel workbook，我们来学习下如何用不同的方法解析单个 worksheet。我们将扩展知识到遍历多个 worksheets，然后遍历多个 workbooks。

处理单个 Worksheet

虽然 Excel workbooks 可以包含多个 worksheets，有时你只希望数据在一个 worksheets 中。另外，只要你知道如解析一个 worksheet,扩展到解析多个 workbooks 就很简单。

读和写 Excel 文件

用基础 Python 的 xlrd 和 xlwt 模块

要用基础 Python 和 xlrd 和 xlwt 模块，在文本编辑器中输入如下代码并将代码保存为 2excel_parsing_and_write.py:

```
1 #!/usr/bin/env python3
2 import sys
3 from xlrd import open_workbook
4 from xlwt import Workbook
5 input_file = sys.argv[1]
6 output_file = sys.argv[2]
7 output_workbook = Workbook()
8 output_worksheet = output_workbook.add_sheet('jan_2013_output')
9 with open_workbook(input_file) as workbook:
10 worksheet = workbook.sheet_by_name('january_2013')
11 for row_index in range(worksheet.nrows):
12 for column_index in range(worksheet.ncols):
13 output_worksheet.write(row_index, column_index, \
14 worksheet.cell_value(row_index, column_index))
15 output_workbook.save(output_file)
```

第 3 行导入 imports xlrd 的 open_workbook 函数，第 4 行导入 xlwt 的 Workbook 对象。第 7 行实例 xlwt Workbook 对象以便我们可以写结果到输出文件 Excel workbook。第 8 行用 xlwt 的 add_sheet 函数添加名为 jan_2013_output 的 worksheet 到输出 workbook。第 9 行用 xlrd 的 open_workbook 函数打开输入 workbook 到 work book 对象。第 10 行用 workbook 对象的 sheet_by_name 函数来访问 名为 january_2013 的 worksheet。第 11 和第 12 行创建 for 循环遍历行和列的索引值,用 range 函数和 worksheet 对象的 nrows 和 ncols 属性,以便我们可以遍历 worksheet 的每一行和每一列。第 13 行用 xlwt 的 write 函数以及行和列索引来写每个单元格的值到输出文件的 worksheet。最后第 15 保存为关闭输出 workbook。

要运行脚本，在命令行中输入如下的命令并回车:

```
python 2excel_parsing_and_write.py sales_2013.xlsx output_files\2output.xls
```

你可以打开输出文件 2output.xls,检查结果。

你可能注意到 Purchase Date 列 column E 的日期，以数字类型出现而不是日期类型。Excel 以浮点形式保存日期和时间，日期在前时间在后。数字 1 表示 1900-Jan-1，因为从 1900-Jan-0 起过了 1 天。因此，这一列的数字代表日期，但是它们并没有格式化为日期。

Xlrd 包提供了另外的函数来格式化日期类值。下一个例子扩展前面的例子，展示如何格式化日期类值，使日期类值打印到屏幕，写入到输出文件如它们在输入文件中一样。

格式化日期 Format dates

本例基于前例，展示如何用 xlrd 维护日期格式，如它们出现在输入 Excel 文件一样。例如，如果一个 Excel worksheet 的日期为 1/19/2000，我们通常想写 1/19/2000 或另外的相关日期格式到输出文件。但是，如前例所示，用当前的代码，我们的结果是输出文件中为数字 36544.0，因为它是 1/0/1900 到 1/19/2000 的天数。要格式化日期列，在文本编辑器中输入如下代码并将文件保存为 3excel_parsing_and_write_keep_dates.py:

```
1 #!/usr/bin/env python3
2 import sys
3 from datetime import date
4 from xlrd import open_workbook, xldate_as_tuple
5 from xlwt import Workbook
6 input_file = sys.argv[1]
7 output_file = sys.argv[2]
8 output_workbook = Workbook()
9 output_worksheet = output_workbook.add_sheet('jan_2013_output')
10 with open_workbook(input_file) as workbook:
11 worksheet = workbook.sheet_by_name('january_2013')
12 for row_index in range(worksheet.nrows):
13 row_list_output = []
14 for col_index in range(worksheet.ncols):
15 if worksheet.cell_type(row_index, col_index) == 3:
16 date_cell = xldate_as_tuple(worksheet.cell_value\
17 (row_index, col_index),workbook.datemode)
18 date_cell = date(*date_cell[0:3]).strftime\
19 ('%m/%d/%Y')
20 row_list_output.append(date_cell)
21 output_worksheet.write(row_index, col_index, date_cell)
22 else:
23 non_date_cell = worksheet.cell_value\
24 (row_index,col_index)
25 row_list_output.append(non_date_cell)
26 output_worksheet.write(row_index, col_index,\
27 non_date_cell)
28 output_workbook.save(output_file)
```

第三行自 datetime 模块导入 date 函数，以便我们将值转换为日期并格式化日期。第四行自 xlrd 导入两个函数。我们用第一个函数打开前例中的 Excel workbook,所以我们关注第二个函数。xldate_as_tuple 函数，使我们转换代表日期、时间或日期时间的 Excel 数字到元组。一旦将数字

转为元组，我们便可以提取特的日期元素（如，年，月，日）并格式化元素到不同的日期格式（如， 1/1/2010 或 January 1, 2010）。第 15 行创建 if-else 语句，检查单元格的类型是否为数字 3。，如果你看过 xlrd 模块的文档，你会发现单元类型 3 意思是单元包含日期。因此，xlrd 语句检查单格是不是含日期。如果是，if 块中的代码操作这个单元格；如果不是，则 else 中的代码操作单元格。因为日期在最后一列，if 块处理最后一列。第 18 行用 worksheet 对象的 ell_value 函数和行与列的索引访问单元格的值。可选地，你可以用 cell().value 函数；两种版本的结果相同。单元格的值是 xldate_as_tuple 函数的第一个参数，它转换浮点数到表示日期的元组。workbook.datemode 参数是需要的，以便函数可以确定日期是基于 1900 还是基于 1904，并转换数字到正确的元组（一些 Mac 版本的 Excel 自 January 1, 1904 起计算日期，关于这方面的更多信息请见 Microsoft reference guide)。xldate_as_tuple 函数的结果赋值给元组变量 date_cell。这一行很长，所在分为两行，用反斜杠作为第一行的结尾（你要记得第 1 章说过反斜杠是必须的，这样 python 解释器才会将两行解释为一行)。但是你的脚本可以用一行而不需要反斜杠。

第 18 行用元组索引来访问 date_cell 元组的前三个元素（如，年，月，日元素)。然后将它们作为参数传给 date 函数，它转换值到日期对象，如第 1 章所述。接着，strftime 函数转换日期对象到字串用特定的日期格式。格式'%m/%d/%Y'指明日期类似于 March 15, 2014 应转换为 03/15/2014。格式化的日期字串再赋值给变量 date_cell。第 20 行用列表的 append 函数，追加 date_cell 中的值到输出列表 row_list_output。要看一下第 16 行和第 18 行的运行结果，可以在两个 date_cell= ...行之间增加打印语句(如 print(date_cell))。重新保存并再运行脚本看 xldate_as_tuple 函数的结果打印到屏幕。接着，去除打印语句把它放在第二个 date_cell = ...行后。重新保存和运行脚本看 date.strftime 函数的结果打印到屏幕。这些打印语句帮助你观察这两行的函数如何转换 Excel 中代表日期的数字到元组然后到格式化的日期字串。Else 块操作所有的非日期单元格。第 23 行用 worksheet 对象的 cell_value 函数以及行和列的索引访问单元格的值并赋值给 non_date_cell 变量。第 25 行用列表的 append 函数追加 non_date_cell 中的值到 row_list_output。综合一起，这两行提取各行的前四列的值并追加到 row_list_output。当行中的所有列被处理并添加到 row_list_output 后，第 26 行写 row_list_output 的值到输出文件。

要运行脚本，在命令行中输入以下并回车:

```
python 3excel_parsing_and_write_keep_dates.py sales_2013.xlsx\
output_files\3output.xls
```

你可以打开输出文件 3output.xls 检查结果。

Pandas.

Pandas 有一系列的命令来读取和写 Excel 文件。这里的示列代码用 pandas 来解析，将代码保存为 pandas_read_and_write_excel.py (这个脚本读取输入的 Excel 文件，打印它的内容到屏幕，写内容到输出文件):

```
#!/usr/bin/env python3
import pandas as pd
import sys
input_file = sys.argv[1]
output_file = sys.argv[2]
data_frame = pd.read_excel(input_file, sheetname='january_2013')
writer = pd.ExcelWriter(output_file)
```

```
data_frame.to_excel(writer, sheet_name='jan_13_output', index=False)
writer.save()
```

要运行脚本，在命令行中输入如下并回车:

```
python pandas_parsing_and_write_keep_dates.py sales_2013.xlsx\
output_files\pandas_output.xls
```

你可以打开输出文件 pandas_output.xls 检查结果。

现在你已理解了如何处理 Excel workbook 中的 worksheet 并保留日期格式，我们来转到过滤 worksheet 中特定的行。如我们在第 2 章一样，我们讨论如何过滤行，通过评估值是否在行中（a)满足特定的条件，(b)在感兴趣的集合内，(c）匹配特定的正则表达式。

过滤特定的行

有时 Excel worksheet 含有很多你不需要保留的行。如，你只需要行的子集，它包含特定的单词或数字，或你只需行的子集它与特定的日期相关。这些情况，你可以用 python 过滤掉你不想要的行并保留你想要的行。你可能已经熟悉了如何在 Excel 中手工过滤行，但是本章的关注点是扩展你的能力以便你可以处理很大的难于打开的 Excel 文件以及大量的 Excel worksheets 手工处理很耗时。

行中的值满足条件

基础 Python.

首先，我们来看一下如何用基础 python 过滤特定的行。我们想要选择行的子集，其中 Sale Amount 大于$1,400.00。要过滤满足这种条件的行的子集，在文本文件中输入以下代码并将代码保存为 4excel_value_meets_condition.py:

```
1 #!/usr/bin/env python3
2 import sys
3 from datetime import date
4 from xlrd import open_workbook, xldate_as_tuple
5 from xlwt import Workbook
6 input_file = sys.argv[1]
7 output_file = sys.argv[2]
8 output_workbook = Workbook()
9 output_worksheet = output_workbook.add_sheet('jan_2013_output')
10 sale_amount_column_index = 3
11 with open_workbook(input_file) as workbook:
12 worksheet = workbook.sheet_by_name('january_2013')
13 data = []
14 header = worksheet.row_values(0)
15 data.append(header)
16 for row_index in range(1,worksheet.nrows):
17 row_list = []
18 sale_amount = worksheet.cell_value\
```

```
19 (row_index, sale_amount_column_index)
20 if sale_amount > 1400.0:
21 for column_index in range(worksheet.ncols):
22 cell_value = worksheet.cell_value\
23 (row_index,column_index)
24 cell_type = worksheet.cell_type\
25 (row_index, column_index)
26 if cell_type == 3:
27 date_cell = xldate_as_tuple\
28 (cell_value,workbook.datemode)
29 date_cell = date(*date_cell[0:3])\
30 .strftime('%m/%d/%Y')
31 row_list.append(date_cell)
32 else:
33 row_list.append(cell_value)
34 if row_list:
35 data.append(row_list)
36 for list_index, output_list in enumerate(data):
37 for element_index, element in enumerate(output_list):
38 output_worksheet.write(list_index, element_index, element)
39 output_workbook.save(output_file)
```

第 13 行创建空的列表 data。我们用输入文件中我们想写入到输出文件的所有行来填充它。第 14 行提取标题行的值。因为我们想要保留标题行，并且用过滤条件检测这一行不合理，第 15 行添加标题行到 data 中。第 18 行创建变量 sale_amount，它贮存行中的 sale amount。cell_value 函数使用第 10 行定义的 sale_amount_column_index 中的数字，来定位 Sale Amount 列。因为我们想保留的行，它的 sale amount 大于 $1,400.00，我们用这个变量来检测变个条件。第 19 行创建一个 for 循环来确保我们只处理余下的行，其中 Sale Amount 列的值大于 1400.0。对于这些行，我们提取每个单元格的值到变量 cell_value ，以及单元格的类型到 cell_type。接着，我们需要检测行中的每一个值是否为日期。如果是，我们格式化值为日期。要创建值被正确格式化的行，我们在第 17 行创建空的列表 row_list，然后第 31 行和 33 行代码从行中添加日期和非日期的值到 row_list。我们为输入文件的每个日期行创建空的 row_lists。但是，我们只用一些值填 row_lists（如，Sale Amount 列大于 1400.0 行）。所以，对于输入文件的每一行，第 34 行检测 row_list 是否为空，并只添加 row_list 到 data，如果 row_list 非空。最后，第 36 和 37 行，我们遍历 data 中的列表和每个列表中的值，将它们写入输出文件。我们追加想要的行到新的列表的原因是，这样它们接收新的连续的行索引值。 那样，当我们写行到输出文件时，它们以连续行的块出现，行与行之间没有空格。要不然，我们写行到输出文件按主 for 循环处理，xlwt 的写函数用原始的输入文件的行索引并写行到输出文件，行与行之间有空白。后面，我们使用相同的方法，选择特定的列，确何我们写列到输出文件，以连续的列的块方式，列与列之间没有空白。

要运行脚本，在命令行窗口中输入如下并回车：

```
python 4excel_value_meets_condition.py sales_2013.xlsx output_files\4output.xls
```

你可以打开文件 output.xls 检查结果。

Pandas.

你可以作 pandas 过滤满足条件的行，通过你相要评估的指定列的名字并在 DataFrame 名称的方括号内指定条件。例如，下面脚本所示的条件指出我们想要保留 Sale Amount 列的值大于 1400.00 的所有行。如果你想要用多个条件，你将条件放在括号内并用(&)或 (|)符号组合它们，取决于你想要的条件逻辑。第 1 行用(&)符号，指明两个条件必须为真。第 2 行用 (|)，指明只要一个条件为真即可。要用 pandas 基于条件过滤行，输入以下的代码到文本编辑器并将文件保存为 pandas_value_meets_condition.py:

```
#!/usr/bin/env python3
import pandas as pd
import sys
input_file = sys.argv[1]
output_file = sys.argv[2]
data_frame = pd.read_excel(input_file, 'january_2013', index_col=None)
data_frame_value_meets_condition = \
data_frame[data_frame['Sale Amount'].astype(float) > 1400.0]
writer = pd.ExcelWriter(output_file)
data_frame_value_meets_condition.to_excel(writer, sheet_name='jan_13_output',\
index=False)
writer.save()
```

要运行脚本，在命令行中输入以下并回车:

```
python pandas_value_meets_condition.py sales_2013.xlsx\
output_files\pandas_output.xls
```

你可以打开输出文件 pandas_output.xls 检查结果。

行中的值在感兴趣的集合中

基础 Python.

要使用基础 Python 过滤行，其中采购日期有指定的集合内（如 01/24/2013 和 01/31/2013 的日期集）,在文本编辑器中输入以下代码并将文件保存为 5excel_value_in_set.py:

```
 1 #!/usr/bin/env python3
2 import sys
3 from datetime import date
4 from xlrd import open_workbook, xldate_as_tuple
5 from xlwt import Workbook
6 input_file = sys.argv[1]
7 output_file = sys.argv[2]
8 output_workbook = Workbook()
9 output_worksheet = output_workbook.add_sheet('jan_2013_output')
10 important_dates = ['01/24/2013', '01/31/2013']
11 purchase_date_column_index = 4
```

```
12 with open_workbook(input_file) as workbook:
13 worksheet = workbook.sheet_by_name('january_2013')
14 data = []
15 header = worksheet.row_values(0)
16 data.append(header)
17 for row_index in range(1, worksheet.nrows):
18 purchase_datetime = xldate_as_tuple(worksheet.cell_value\
19 (row_index, purchase_date_column_index)\
20 ,workbook.datemode)
21 purchase_date = date(*purchase_datetime[0:3]).strftime('%m/%d/%Y')
22 row_list = []
23 if purchase_date in important_dates:
24 for column_index in range(worksheet.ncols):
25 cell_value = worksheet.cell_value\
26 (row_index,column_index)
27 cell_type = worksheet.cell_type(row_index, column_index)
28 if cell_type == 3:
29 date_cell = xldate_as_tuple\
30 (cell_value,workbook.datemode)
31 date_cell = date(*date_cell[0:3])\
32 .strftime('%m/%d/%Y')
33 row_list.append(date_cell)
34 else:
35 row_list.append(cell_value)
36 if row_list:
37 data.append(row_list)
38 for list_index, output_list in enumerate(data):
39 for element_index, element in enumerate(output_list):
40 output_worksheet.write(list_index, element_index, element)
41 output_workbook.save(output_file)
```

脚本与基于条件过滤行相似。不同的地方在第 10, 21,和 23 行。第 10 行创建列表 important_dates 包含我们感兴趣的日期。第 21 行创建变量 purchase_date，它等于 Purchase Date 列中的值格式化为 important_dates 中的日期相同的格式。第 23 行检测行中的日期是否在 important_dates 中。如果是，我们处理这行并写入输出文件。

要运行脚本，在命令行中输入以下并回车：

```
python 5excel_value_in_set.py sales_2013.xlsx output_files\5output.xls
```

你可以打开输出文件 5output.xls 检查结果。

Pandas.

本例中，我们想要过滤行，其中 purchase date 为 01/24/2013 或 01/31/2013。Pandas 提供了 isin 函数，你可以用来检测特定值是否在某个列表中。要用 pandas 基于集合的成员来过滤行，在文本编辑器输入如下代码并将文件保存为 pandas_value_in_set.py:

```
#!/usr/bin/env python3
import pandas as pd
import sys
input_file = sys.argv[1]
output_file = sys.argv[2]
data_frame = pd.read_excel(input_file, 'january_2013', index_col=None)
important_dates = ['01/24/2013','01/31/2013']
data_frame_value_in_set = data_frame[data_frame['PurchaseDate']\
.isin(important_dates)]
writer = pd.ExcelWriter(output_file)
data_frame_value_in_set.to_excel(writer, sheet_name='jan_13_output', index=False)
writer.save()
```

在命令行中运行脚本:

```
python pandas_value_in_set.py sales_2013.xlsx output_files\pandas_output.xls
```

你可以打开输出文件 pandas_output.xls 来检测结果。

行中的值匹配特定的模式

基础 Python.

要用基础 python 过滤行，其中的客户名含特定的模式(如,以字母 J 开始)，在文件编辑器中输入如下代码并将文件保存为 6excel_value_matches_pattern.py:

```
1 #!/usr/bin/env python3
2 import re
3 import sys
4 from datetime import date
5 from xlrd import open_workbook, xldate_as_tuple
6 from xlwt import Workbook
7 input_file = sys.argv[1]
8 output_file = sys.argv[2]
9 output_workbook = Workbook()
10 output_worksheet = output_workbook.add_sheet('jan_2013_output')
11 pattern = re.compile(r'(?P<my_pattern>^J.*)')
12 customer_name_column_index = 1
13 with open_workbook(input_file) as workbook:
14 worksheet = workbook.sheet_by_name('january_2013')
15 data = []
16 header = worksheet.row_values(0)
```

```
17 data.append(header)
18 for row_index in range(1, worksheet.nrows):
19 row_list = []
20 if pattern.search(worksheet.cell_value\
21 (row_index, customer_name_column_index)):
22 for column_index in range(worksheet.ncols):
23 cell_value = worksheet.cell_value\
24 (row_index,column_index)
25 cell_type = worksheet.cell_type(row_index, column_index)
26 if cell_type == 3:
27 date_cell = xldate_as_tuple\
28 (cell_value,workbook.datemode)
29 date_cell = date(*date_cell[0:3])\
30 .strftime('%m/%d/%Y')
31 row_list.append(date_cell)
32 else:
33 row_list.append(cell_value)
34 if row_list:
35 data.append(row_list)
36 for list_index, output_list in enumerate(data):
37 for element_index, element in enumerate(output_list):
38 output_worksheet.write(list_index, element_index, element)
39 output_workbook.save(output_file)
```

第 2 行导入 re 模块以便访问模块的函数和方法。

第 11 行用 re 模块的 compile 函数创建正则表达式 pattern。如果你读，这函数的内容看起来很熟悉。r 的意思是单引号内的字串是原始字串。 ?P<my_pattern>Metacharacter 捕获匹配的子串于 <my_pattern>组，便必要时它们被打印到屏幕或写入文件。实际的模式是'^J.*'。caret 是特殊的字符，意思为“字串的开始”。所以字串要以字母 J 开始。句号(.)匹配任何除 newline 之外的符号，所以除 newline 之外的所有符号都可以接字母 J。最后，星号(*)的意思是重复前面的符号 0 或多次。综合起来，.*的组合意思是任何除 newline 之外的符号可以在字母 J 后出现任意次数。第 20 行用 re 模块的 search 方法查找 Customer Name 列的模式并检查是否有匹配的。如果有匹配的，就追加这行中的值到 row_list。第 31 行添加日期值到 row_list，第 33 行添加非日期值到 row_list。第 35 行添加 row_list 中的列表值到 data，如果列表非空。最后，第 36 行和第 37 行的两个 for 循环遍历 data 中的列表以将行写入到输出文件。

要运行脚本，在命令行中输入如下并回车：

```
python 6excel_value_matches_pattern.py sales_2013.xlsx output_files\6output.xls
```

你可以打开输出文件 6output.xls 检查结查。

Pandas.

本例，我们想要过滤行，其中客户名以字母 J 开头。Pandas 提供了多个字串和正是表达式函数，包括 startswith, endswith, match, 和 search，你可以用来识别文本中的子串和模式。要用 pandas 过滤以字母 J 开头的客户名，在文本编辑器中输入以下代码并将文件何存为 pandas_value_matches_pattern.py:

```
#!/usr/bin/env python3
import pandas as pd
import sys
input_file = sys.argv[1]
output_file = sys.argv[2]
data_frame = pd.read_excel(input_file, 'january_2013', index_col=None)
data_frame_value_matches_pattern = data_frame[data_frame['Customer Name']\
.str.startswith("J")]
writer = pd.ExcelWriter(output_file)
data_frame_value_matches_pattern.to_excel(writer, sheet_name='jan_13_output',\
index=False)
writer.save()
```

要运行脚本，在命令行中输入以下并回车:

```
python pandas_value_matches_pattern.py sales_2013.xlsx\
output_files\pandas_output.xls
```

你可以打开输出文件 pandas_output.xls 检查结果。

选择特定列

有时候 worksheet 包含多个你不想要保留的列。变种情况下，你可以用 python 选择你想要保存的列。有两种常用的方法来选择 Excel 中特定的列。

下面展示用两种方法选择列:

- 用列索引值
- 用列标题

列索引

基础 Python.

自 worksheet 中选择特定列的一种方法是用你想要保留的列的索引值。当你关注的列的索引值易于识别时，或当你处理多个输入文件，它们的列的位置相同时，使用这种方法有效。

作为例子，我们只想保存 Customer Name 和 Purchase Date 列是。要用基础 Python 保存这两列，在文本编辑器中输入以下代码并将文件保存为 7excel_column_by_index.py:

```
1 #!/usr/bin/env python3
2 import sys
3 from datetime import date
4 from xlrd import open_workbook, xldate_as_tuple
```

```
5 from xlwt import Workbook
6 input_file = sys.argv[1]
7 output_file = sys.argv[2]
8 output_workbook = Workbook()
9 output_worksheet = output_workbook.add_sheet('jan_2013_output')
10 my_columns = [1, 4]
11 with open_workbook(input_file) as workbook:
12 worksheet = workbook.sheet_by_name('january_2013')
13 data = []
14 for row_index in range(worksheet.nrows):
15 row_list = []
16 for column_index in my_columns:
17 cell_value = worksheet.cell_value(row_index,column_index)
18 cell_type = worksheet.cell_type(row_index, column_index)
19 if cell_type == 3:
20 date_cell = xldate_as_tuple\
21 (cell_value,workbook.datemode)
22 date_cell = date(*date_cell[0:3]).strftime('%m/%d/%Y')
23 row_list.append(date_cell)
24 else:
25 row_list.append(cell_value)
26 data.append(row_list)
27 for list_index, output_list in enumerate(data):
28 for element_index, element in enumerate(output_list):
29 output_worksheet.write(list_index, element_index, element)
30 output_workbook.save(output_file)
```

第 10 行创建列表变量 my_columns，它包含整数 1 和 4。变两个数代表 Customer Name 和 PurchaseDate 列的索引值。第 16 行创建 for 循环，遍历 my_columns 中的两列索引值。每次提取那列的值和为单元格类型，确定单元格中的值是否为日期类型，处理单元格的值，然后添加值到 row_list。第 26 行添加 row_list 中每个列表的值到 data。最后，第 27 行和第 28 行的两个 for 循环遍历 data 中的列表以将值写到输出文件。

要运行脚本，在命令行中输入以下并回车：

```
python 7column_column_by_index.py sales_2013.xlsx output_files\7output.xls
```

你可以打开输出文件 7output.xls 检查结果。

Pandas.

在 pandas 中有很多的方法来选择特定的列。其中一种方法是指定 DataFrame，然后在方括号内，列出你想要保留的列的索引值或列名。另一种方法是，指明 DataFrame，结合 iloc 函数。iloc 函数很有用，因为它使你同时选择特定的行和列。所以，如果你用 iloc 函数，选择列，需要加帽号和逗号在列索引值列表的前面来指示你想要保存这些列的所有行。否则，iloc 用这些索引过滤行。

要在 pandas 中基于列的索引选择列，在文本编辑器中输入以下代码并将文件保存为 pandas_column_by_index.py:

```
#!/usr/bin/env python3
import pandas as pd
import sys
input_file = sys.argv[1]
output_file = sys.argv[2]
data_frame = pd.read_excel(input_file, 'january_2013', index_col=None)
data_frame_column_by_index = data_frame.iloc[:, [1, 4]]
writer = pd.ExcelWriter(output_file)
data_frame_column_by_index.to_excel(writer, sheet_name='jan_13_output',\
index=False)
writer.save()
```

要运行脚本，在命令行中输入以下并回车:

```
python pandas_column_by_index.py sales_2013.xlsx output_files\pandas_output.xls
```

你可以打开输出文件 pandas_output.xls 检查结果。

列标题

从 worksheet 中选择列的子集的第二种方法是用列的标题。这种方法有效，当易于识别你想要保留的列的名称时。当你处理多个输入文件，且列的名字一致时，也很有帮助。

基础 Python.

要用基础 Python 选择 Customer ID 和 Purchase Date 列。在文本编辑器中输入以下代码并将文件保存为 8excel_column_by_name.py:

```
1 #!/usr/bin/env python3
2 import sys
3 from datetime import date
4 from xlrd import open_workbook, xldate_as_tuple
5 from xlwt import Workbook
6 input_file = sys.argv[1]
7 output_file = sys.argv[2]
8 output_workbook = Workbook()
9 output_worksheet = output_workbook.add_sheet('jan_2013_output')
10 my_columns = ['Customer ID', 'Purchase Date']
11 with open_workbook(input_file) as workbook:
12 worksheet = workbook.sheet_by_name('january_2013')
13 data = [my_columns]
14 header_list = worksheet.row_values(0)
15 header_index_list = []
16 for header_index in range(len(header_list)):
17 if header_list[header_index] in my_columns:
```

```
18 header_index_list.append(header_index)
19 for row_index in range(1,worksheet.nrows):
20 row_list = []
21 for column_index in header_index_list:
22 cell_value = worksheet.cell_value(row_index,column_index)
23 cell_type = worksheet.cell_type(row_index, column_index)
24 if cell_type == 3:
25 date_cell = xldate_as_tuple\
26 (cell_value,workbook.datemode)
27 date_cell = date(*date_cell[0:3]).strftime('%m/%d/%Y')
28 row_list.append(date_cell)
29 else:
30 row_list.append(cell_value)
31 data.append(row_list)
32 for list_index, output_list in enumerate(data):
33 for element_index, element in enumerate(output_list):
34 output_worksheet.write(list_index, element_index, element)
35 output_workbook.save(output_file)
```

第 10 行创建列表变量 my_columns，它包含我们想要保留的两列的名字。因为这是我们想要写入输出文件的列标题，第 13 行我们添加它们到输出列表 data 中。第 16 行创建 for 循环遍历 header_list 中列标题的索引值。第 17 行用列表索引检测列标题是否在 my_columns 里。如果是，第 18 行添加列标题的索引值到 header_index_list。第 25 行我们用这些索引值处理我们想要写入输出文件的列。第 21 行创建 for 循环遍历 header_index_list 中的列索引值。通过 header_index_list，我们只处理 my_columns 中列出的列。

要运行脚本，在命令行中输入以下并回车。

```
python 8excel_column_by_name.py sales_2013.xlsx output_files\8output.xls
```

你可以打开输出文件 8output.xls,检查结果。

Pandas.

要用 pandas 基于列标题选择特定的列,你可以列出列名,作为字串,放在 DataFrame 后的方括号内。可选地，你可以用 loc 函数。再次，如果你用 loc 函数，则你要添加帽号和逗号在列标题列表的前面以指示你想为那些列保留所有的行。要在 pandas 中基于列标题选择列，在文本文件中输入以下代码并保存为 pandas_column_by_name.py:

```
#!/usr/bin/env python3
import pandas as pd
import sys
input_file = sys.argv[1]
output_file = sys.argv[2]
data_frame = pd.read_excel(input_file, 'january_2013', index_col=None)
data_frame_column_by_name = data_frame.loc[:, ['Customer ID', 'Purchase Date']]
```

```
writer = pd.ExcelWriter(output_file)
data_frame_column_by_name.to_excel(writer, sheet_name='jan_13_output',\
index=False)
writer.save()
```

要运行脚本，在命令行中输入以下并回车：

```
python pandas_column_by_name.py sales_2013.xlsx output_files\pandas_output.xls
```

你可以打开输出文件 pandas_output.xls 检查结果。

读取 Workbook 中的所有 Worksheets

到这里，我已展示了如何处理单个 worksheet。有时候，你只需要处理一个 worksheet。这种情况，前面的例子给了你用 python 自动处理 worksheet 的思路。但是，许多情况下，你需要处理很多 worksheet，用手工处理不高效或不可能。这种情况下，python 很有用，因为它使你自动化和放大数据处理过程超越手工处理。这一节用两个例子展示如何过滤 workbook 中的所有 worksheets 的行和列。我只用一个例子来过滤行，一个例子来过滤列。因为我不想让本章写得太长（处理 workbook 中特定的 worksheets，处理多个 workbooks 还没写）。另外，你从前面的例子理解了别的方法来选择特定的行和列，你将这些别的方法整合到这些例子中就有很好的思路了。

在所有的 Worksheets 中选取特定的行

基础 Python

要用基础 python 过滤所有 worksheets 中的所有行，其中 sale amount 大于 $2,000.00，在文本文件中加入以下代码并将文件保存为 9excel_value_meets_condition _all_worksheets.py：

```
1 #!/usr/bin/env python3
2 import sys
3 from datetime import date
4 from xlrd import open_workbook, xldate_as_tuple
5 from xlwt import Workbook
6 input_file = sys.argv[1]
7 output_file = sys.argv[2]
8 output_workbook = Workbook()
9 output_worksheet = output_workbook.add_sheet('filtered_rows_all_worksheets')
10 sales_column_index = 3
11 threshold = 2000.0
12 first_worksheet = True
13 with open_workbook(input_file) as workbook:
14 data = []
15 for worksheet in workbook.sheets():
```

```
16 if first_worksheet:
17 header_row = worksheet.row_values(0)
18 data.append(header_row)
19 first_worksheet = False
20 for row_index in range(1,worksheet.nrows):
21 row_list = []
22 sale_amount = worksheet.cell_value\
23 (row_index, sales_column_index)
24 if sale_amount > threshold:
25 for column_index in range(worksheet.ncols):
26 cell_value = worksheet.cell_value\
27 (row_index,column_index)
28 cell_type = worksheet.cell_type\
29 (row_index, column_index)
30 if cell_type == 3:
31 date_cell = xldate_as_tuple\
32 (cell_value,workbook.datemode)
33 date_cell = date(*date_cell[0:3])\
34 .strftime('%m/%d/%Y')
35 row_list.append(date_cell)
36 else:
37 row_list.append(cell_value)
38 if row_list:
39 data.append(row_list)
40 for list_index, output_list in enumerate(data):
41 for element_index, element in enumerate(output_list):
42 output_worksheet.write(list_index, element_index, element)
43 output_workbook.save(output_file)
```

第 10 行创建 sales_column_index 这量存贮 Sale Amount 列的索引值。相似的，第 11 行创建 threshold 存贮我们关注的 sale amount。我们将 Sale Amount 列的值与这个 threshold 值比较,确定哪些行写入输出文件。第 15 行创建 for 循环用来遍历 workbook 中的所有 worksheets。它用 workbook 对象的 sheets 属性来列出 workbook 的所有的 worksheets。第 16 行对于第一个 worksheet 为真，所以对于第一个 worksheet，我们提取标题行，添加到 data，然后设置 first_worksheet 为假。代码继续，处理余下的行数据，其中行的 sale amount 大于 threshold 值。对于后面的所有 worksheets，first_worksheet 为假，所以脚本进行到第 20 行处理每个 worksheet 的数据行。你知道它处理数据行，而不是列标题行，因为 range 函数从 1 开始而不是 0 开始。

要运行脚本，在命令行中输入以下并回车：

```
python 9excel_value_meets_condition_all_worksheets.py sales_2013.xlsx\
output_files\9output.xls
```

你找开输出文件 9output.xls 检查结果。

Pandas

Pandas 允许你一次性读 workbook 内的所有 worksheets，通过指定 read_excel 函数的 sheetname=None。Pandas 读取 worksheets 到 DataFrames 构成的字典中，key 为 worksheet 的名字,value 为 DataFrame 中 worksheet 的数据。所以你可以评估所有的 workbook 中的数据，通过遍历字典的键值。当你过滤每个 DataFrame 的特定行时，结果为新的过滤过的 DataFrame，所以你可以创建过滤过的 DataFrame 的列表，然后拼接它们到最终的 DataFrame。本例，我们想要过滤所有 worksheets 的所有行，其中 sale amount 大于 $2,000.00。要用 pandas 过滤这些行，在文本编辑器中输入如下代码并保存为 pandas_value_meets_condition_all_worksheets.py:

```
#!/usr/bin/env python3
import pandas as pd
import sys
input_file = sys.argv[1]
output_file = sys.argv[2]
data_frame = pd.read_excel(input_file, sheetname=None, index_col=None)
row_output = []
for worksheet_name, data in data_frame.items():
row_output.append(data[data['Sale Amount'].astype(float) > 2000.0])
filtered_rows = pd.concat(row_output, axis=0, ignore_index=True)
writer = pd.ExcelWriter(output_file)
filtered_rows.to_excel(writer, sheet_name='sale_amount_gt2000', index=False)
writer.save()
```

要运行脚本，在命令行中输入如下并回车:

```
python pandas_value_meets_condition_all_worksheets.py sales_2013.xlsx\
output_files\pandas_output.xls
```

你可以打开输出文件 pandas_output.xls 检查结果。

选择所有 Worksheets 中的特定列

有时 Excel workbook 包含多个 worksheets 且每个 worksheets 有多个你不想要的列。这些情况，你可以用 Python 读所有的 worksheets，过滤出你不想要的列，保留你想要的列。我们前面学习过，有最少两种方法来从 worksheet 选择列的子集--通过索引值或列标题。下面的例子展示如何从 workbook 的所有 worksheets 选择特定的列，使用列标题。

基础 Python

要用基础 python 从所有的 worksheets 中选择 Customer Name 和 Sale Amount 列，在文本编辑器中输入以下代码并保存为 10excel_column_by_name_all_worksheets.py:

```
1 #!/usr/bin/env python3
2 import sys
```

```
3 from datetime import date
4 from xlrd import open_workbook, xldate_as_tuple
5 from xlwt import Workbook
6 input_file = sys.argv[1]
7 output_file = sys.argv[2]
8 output_workbook = Workbook()
9 output_worksheet = output_workbook.add_sheet('selected_columns_all_worksheets')
10 my_columns = ['Customer Name', 'Sale Amount']
11 first_worksheet = True
12 with open_workbook(input_file) as workbook:
13 data = [my_columns]
14 index_of_cols_to_keep = []
15 for worksheet in workbook.sheets():
16 if first_worksheet:
17 header = worksheet.row_values(0)
18 for column_index in range(len(header)):
19 if header[column_index] in my_columns:
20 index_of_cols_to_keep.append(column_index)
21 first_worksheet = False
22 for row_index in range(1, worksheet.nrows):
23 row_list = []
24 for column_index in index_of_cols_to_keep:
25 cell_value = worksheet.cell_value\
26 (row_index, column_index)
27 cell_type = worksheet.cell_type(row_index, column_index)
28 if cell_type == 3:
29 date_cell = xldate_as_tuple\
30 (cell_value,workbook.datemode)
31 date_cell = date(*date_cell[0:3])\
32 .strftime('%m/%d/%Y')
33 row_list.append(date_cell)
34 else:
35 row_list.append(cell_value)
36 data.append(row_list)
37 for list_index, output_list in enumerate(data):
38 for element_index, element in enumerate(output_list):
39 output_worksheet.write(list_index, element_index, element)
40 output_workbook.save(output_file)
```

第 10 行创建立列表量 my_columns，它包含我们想要保留的两列的名字。第 13 行将 my_columns 作为 data 的第一个列表的值，因为它们是我们想要写入到输出文件的列标题。第 14 行创建一个空的列表 index_of_cols_to_keep，它包含 Customer Name 和 Sale Amount 列的索引。第 16 行检测

我们是否处理第一个 worksheet。如果是，我们识别 Customer Name 和 Sale Amount 列的索引值并添加到 index_of_cols_to_keep。然后我们设置 first_worksheet 为假。继续代码，处理余下的数据行，用第 24 行处理 Customer Name 和 Sale Amount 列的值。对于后面的 worksheets，worksheets 为假，所以脚本进行到第 22 行处理每个 worksheets 的数据行。对于这些 worksheets，我们只处理 index_of_cols_to_keep 列出的索引值的列。如果这些列中的值是日期，我们格式化它为日期。组合一行数据后我们想写到输出文件，第 36 行我们添加值列表到 data。

要运行脚本，在命令行中输入如下并回车:

```
python 10excel_column_by_name_all_worksheets.py sales_2013.xlsx\
output_files\10output.xls
```

你可以找开输出文件 10output.xls，检查结果。

Pandas

再一次我们用 pandas 将所有的 worksheets 写入字典，使用 read_excel 函数。然后用 loc 从每个 worksheet 选择特定的列，创建过滤后的 DataFrames 的列表，拼接这些 DataFrames 到一个 DataFrame。本例中，我们想选择所有 worksheets 的 Customer Name 和 Sale Amount 列。

要用 pandas 选择这些列，在文本编辑器输入如下代码并保存为 pandas_column_by_name_all_worksheets.py:

```
#!/usr/bin/env python3
import pandas as pd
import sys
input_file = sys.argv[1]
output_file = sys.argv[2]
data_frame = pd.read_excel(input_file, sheetname=None, index_col=None)
column_output = []
for worksheet_name, data in data_frame.items():
column_output.append(data.loc[:, ['Customer Name', 'Sale Amount']])
selected_columns = pd.concat(column_output, axis=0, ignore_index=True)
writer = pd.ExcelWriter(output_file)
selected_columns.to_excel(writer, sheet_name='selected_columns_all_worksheets',\
index=False)
writer.save()
```

要运行脚本，在命令行中输入如下并回车:

```
python pandas_column_by_name_all_worksheets.py sales_2013.xlsx\
output_files\pandas_output.xls
```

你可以找开输出文件 pandas_output.xls 检查结果。

读 Excel Workbook 的 Worksheets 集合

本章早期展示了如何从一个 worksheet 过滤特定的行和列。前面展示了如何从 workbook 的所有 worksheet 过滤特定的行和列。但是，有时候，你只需要要处理一个 workbook 的 worksheets

子集。例如，你的 workbook 有许多 worksheets，而你只要处理其中 20 个。变种情况，你可以用 workbook 的 sheet_by_index 或 sheet_by_name 函数，来处理 worksheets 子集。本节的例子展示如何从 workbook 的 worksheets 子集过滤特定的行和列。我只举了一个例子，因为到这里你可以整合前面的其它的过滤和选择操作到这个例子。

从 Worksheets 集合过滤特定的行

基础 Python

本例中，我们想从第一个和第二个 worksheets 过滤行和列，其中 sale amount 大于 $1,900.00。要用基础 python 从第一个和第二个个 worksheets 选择行和列的子集，
在文本编辑器输入如下代码并保存为 1excel_value_meets_condition_set_of_worksheets.py:

```
1 #!/usr/bin/env python3
2 import sys
3 from datetime import date
4 from xlrd import open_workbook, xldate_as_tuple
5 from xlwt import Workbook
6 input_file = sys.argv[1]
7 output_file = sys.argv[2]
8 output_workbook = Workbook()
9 output_worksheet = output_workbook.add_sheet('set_of_worksheets')
10 my_sheets = [0,1]
11 threshold = 1900.0
12 sales_column_index = 3
13 first_worksheet = True
14 with open_workbook(input_file) as workbook:
15 data = []
16 for sheet_index in range(workbook.nsheets):
17 if sheet_index in my_sheets:
18 worksheet = workbook.sheet_by_index(sheet_index)
19 if first_worksheet:
20 header_row = worksheet.row_values(0)
21 data.append(header_row)
22 first_worksheet = False
23 for row_index in range(1,worksheet.nrows):
24 row_list = []
25 sale_amount = worksheet.cell_value\
26 (row_index, sales_column_index)
27 if sale_amount > threshold:
28 for column_index in range(worksheet.ncols):
29 cell_value = worksheet.cell_value\
30 (row_index,column_index)
```

```
31 cell_type = worksheet.cell_type\
32 (row_index, column_index)
33 if cell_type == 3:
34 date_cell = xldate_as_tuple\
35 (cell_value,workbook.datemode)
36 date_cell = date(*date_cell[0:3])\
37 .strftime('%m/%d/%Y')
38 row_list.append(date_cell)
39 else:
40 row_list.append(cell_value)
41 if row_list:
42 data.append(row_list)
43 for list_index, output_list in enumerate(data):
44 for element_index, element in enumerate(output_list):
45 output_worksheet.write(list_index, element_index, element)
46 output_workbook.save(output_file)
```

第 10 行创建变量 my_sheets，它包含两个整数代表 我们想要处理的 worksheets 的索引。第 16 行创建 workbook 的所有 worksheets 的索引，并用 for 循环来遍历索引值。第 17 行检查 for 循环中考虑的索引是否是 my_sheets 中的值。这个检查确保我们只处理我们想要的 worksheets。因为我们遍历 worksheet 索引值，第 18 行我们要用 workbook 的 sheet_by_index 函数，给合索引值来访问当前的 worksheet。对于第一个我们要处理的 worksheet，第 19 行为真，所以我们添加标题行到 data 然后设置 first_worksheet 为假。然后我们处理余下的行用早期的例子相似的方法。对于第二个和后面的我们想要处理的 worksheets，脚本进行到第 23 行处理 worksheets 的数据行。

要运行脚本，在命令行中输入如下并回车:python
11excel_value_meets_condition_set_of_worksheets.py sales_2013.xlsx\
output_files\11output.xls

你可以找开输出文件 11output.xls，检查结果。

Pandas

Pandas 使选择 workbook 中的 worksheets 子集变得容易。你只要在 read_excel 函数中指明 worksheets 的名字或索引的列表。本例中，我们创建索引值列表 named my_sheets 然后在 read_excel 中设置 sheetname 等于 my_sheets。要用 pandas 选择 worksheets 子集，
在文本编辑器输入如下代码并保存为pandas_value_meets_condition_set_of_worksheets.py:

```
#!/usr/bin/env python3
import pandas as pd
import sys
input_file = sys.argv[1]
output_file = sys.argv[2]
my_sheets = [0,1]
```

```
threshold = 1900.0
data_frame = pd.read_excel(input_file, sheetname=my_sheets, index_col=None)
row_list = []
for worksheet_name, data in data_frame.items():
row_list.append(data[data['Sale Amount'].astype(float) > threshold])
filtered_rows = pd.concat(row_list, axis=0, ignore_index=True)
writer = pd.ExcelWriter(output_file)
filtered_rows.to_excel(writer, sheet_name='set_of_worksheets', index=False)
writer.save()
```

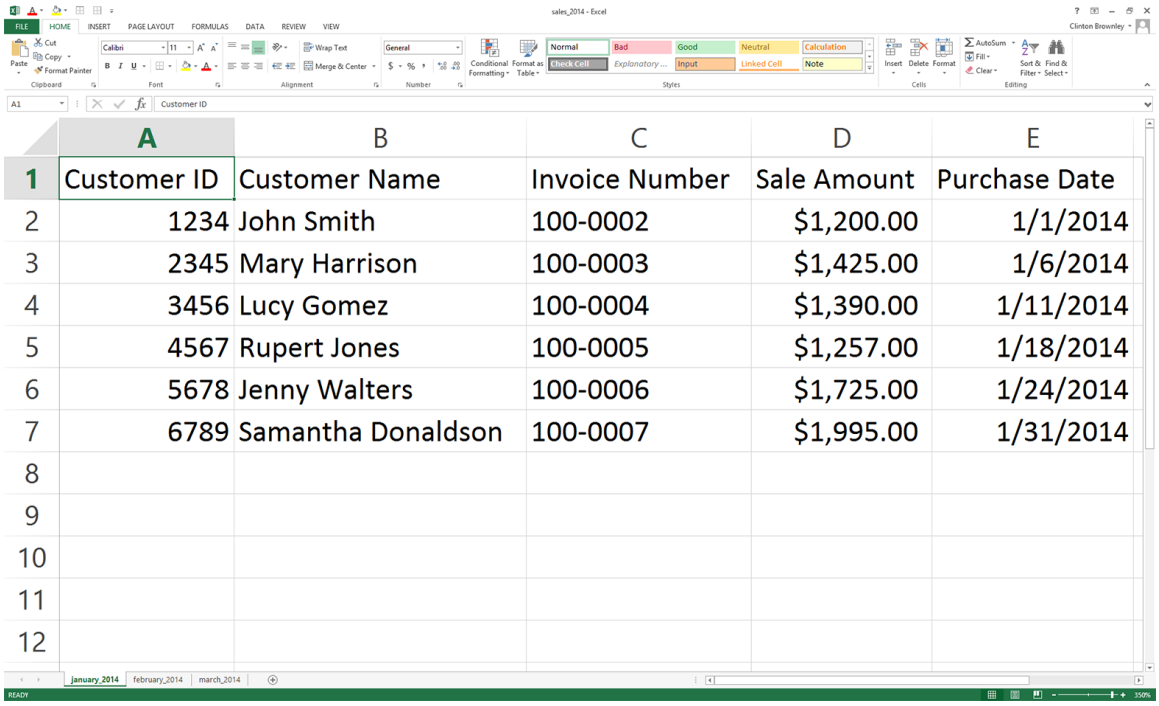
要运行脚本，在命令行中输入如下并回车:

```
python pandas_value_meets_condition_set_of_worksheets.py\
sales_2013.xlsx output_files\pandas_output.xls
```

你可以找开输出文件 pandas_output.xls,检查结果。

处理多个 Workbooks

本章前面我们展示了如何从一个 workbook 的一个 worksheet，所有 worksheets，worksheets 子集过滤特定的行和列。这些技术对于处理一个 workbook 非常有用。但是，有时候，你要处理许多的 workbooks。变种情况，python 很有用，因为它使你自动化和放大你的数据处理过程，远远超越手工处理。 这一节再次介绍 Python 的内置模块 glob，我们在第 2 章讲过,并构建一些本章早期的例子来展示如何处理多个 workbooks。为了处理多个 workbooks，我们要创建多个 workbooks。我们再创建两个 workbooks 来处理，总共三个 workbooks。
但是要记信，这里的技术可以放大到你的计算机可以处理的许多文件。
开如:
1. 打开已有的 sales_2013.xlsx.
创建第二个 workbook:
2. 改为已有的三个 worksheets 名字为 january_2014, february_2014, 和 march_2014.
3.在这三个 worksheets 里,改变 Purchase Date 列的年份到 2014。
每个 worksheet 有六个数据行，所以你要处理 18 行变化（6 行 * 3 worksheets)。除了改变年份，你不需要其它的改变。
4. 保存第二个 workbook 为 sales_2014.xlsx。
图 3-10 展示了 january_2014 看起来的样子。



	A	B	C	D	E
1	Customer ID	Customer Name	Invoice Number	Sale Amount	Purchase Date
2	1234	John Smith	100-0002	$1,200.00	1/1/2014
3	2345	Mary Harrison	100-0003	$1,425.00	1/6/2014
4	3456	Lucy Gomez	100-0004	$1,390.00	1/11/2014
5	4567	Rupert Jones	100-0005	$1,257.00	1/18/2014
6	5678	Jenny Walters	100-0006	$1,725.00	1/24/2014
7	6789	Samantha Donaldson	100-0007	$1,995.00	1/31/2014
8					
9					
10					
11					
12					

图 3-10. 通过改变日期从第一个 workbook 创建第二个 workbook

现在创建第三个:

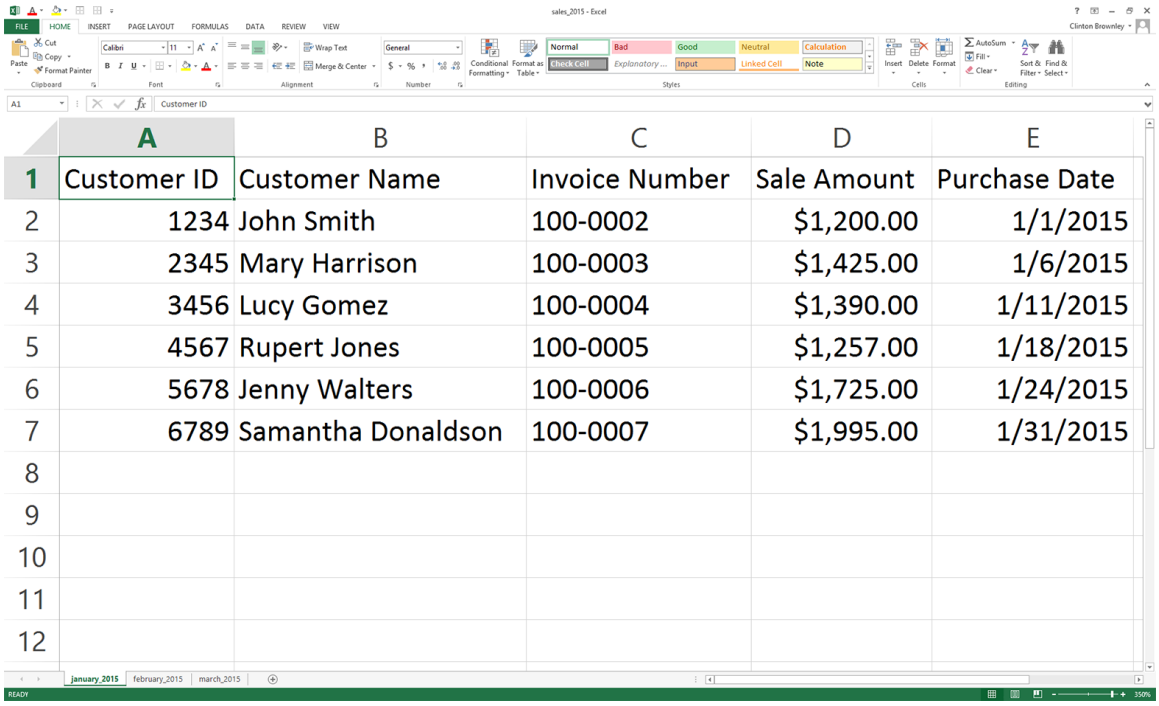
5. 改变三个 worksheets 名字为 january_2015, february_2015, and march_2015.

6. 在这三个 worksheets 里,改变 Purchase Date 列的年份为 2015.

每个 worksheet 有六个数据行,所以你要改变 18 行 (6 行 *3 个 worksheets)。除了改变年份,你不需要作其它的改变。

7. 将第三个 workbook 保存为 sales_2015.xlsx.

图 3-11 展示 january_2015 看起来的样子。



	A	B	C	D	E
1	Customer ID	Customer Name	Invoice Number	Sale Amount	Purchase Date
2	1234	John Smith	100-0002	$1,200.00	1/1/2015
3	2345	Mary Harrison	100-0003	$1,425.00	1/6/2015
4	3456	Lucy Gomez	100-0004	$1,390.00	1/11/2015
5	4567	Rupert Jones	100-0005	$1,257.00	1/18/2015
6	5678	Jenny Walters	100-0006	$1,725.00	1/24/2015
7	6789	Samantha Donaldson	100-0007	$1,995.00	1/31/2015
8					
9					
10					
11					
12					

图 3-11. 通过改变日期从第二个 workbook 创建第三个 workbook

计算 Workbooks 数目以及每个 Workbooks 的行数和列数

有时，你想知道你要处理的 workbooks 的内容;有时你还没有创建 workbooks 所以不知道内容。不像 CSV 文件，Excel workbooks 可以包含多个，所以如果你不熟悉 workbooks，那么你在开始处理它们之前要获得一些关于它们的描述信息。

要计算目录中 workbooks 的数目，每个 workbook 的 worksheets 数目，每个 worksheets 的行数和列数， 在文本编辑器输入如下代码并保存为 12excel_introspect_all_workbooks.py:

```
#!/usr/bin/env python3
import glob
import os
import sys
from xlrd import open_workbook
input_directory = sys.argv[1]
workbook_counter = 0
for input_file in glob.glob(os.path.join(input_directory, '*.xls*')):
workbook = open_workbook(input_file)
print('Workbook: %s' % os.path.basename(input_file))
print('Number of worksheets: %d' % workbook.nsheets)
for worksheet in workbook.sheets():
print('Worksheet name:', worksheet.name, '\tRows:',\
worksheet.nrows, '\tColumns:', worksheet.ncols)
```

```
15 workbook_counter += 1
16 print('Number of Excel workbooks: %d' % (workbook_counter))
```

第 2 行和第 3 行导入 Python 的内置 glob 和 os 模块,以便我们可以用它们的函数来识别和解析我们想要处理的文件路径。第 8 行用 Python 的内置 glob 和 os 模块来创建我们想要处理的输入文件列表并用 for 循环来遍历输入文件的列表。这一行允许我们遍历我们想要处理的所有的 workbooks。第 10 行到第 14 行打印每个 workbook 的信息到屏幕中。第 10 行打印 workbook 的名字,第 11 行打印 workbook 中 worksheets 名字。第 13 行和第 14 行打印 workbook 中 worksheets 名字以及每个 worksheets 的行数和列数。

要运行脚本，在命令行中输入如下并回车:

```
python 12excel_introspect_all_workbooks.py "C:\Users\Clinton\Desktop"
```

你可以看到如图 3-12 所示的打印输出到你的屏幕。

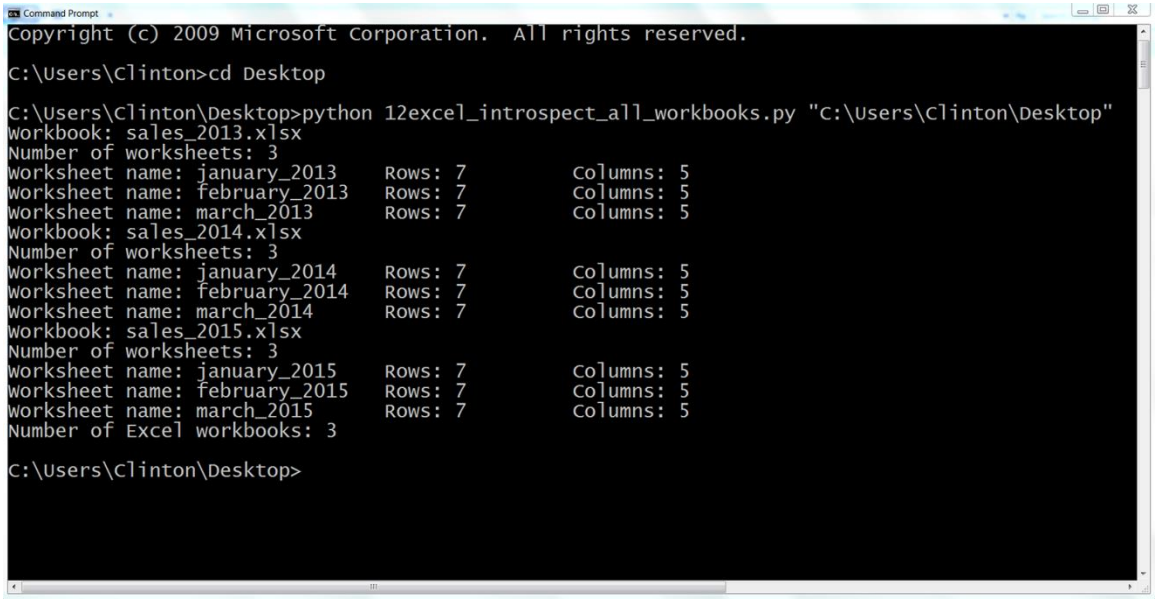


图 3-12. Python 脚本处理多个 workbooks 的输出

输出显示脚本处理了三个 workbooks。它也显示了三个 workbooks 的名字 (如,sales_2013.xlsx)，每个 workbook 中 worksheets 的名字 (如，january_2013)，以及每个 worksheet 的行数和列数 (如，7 行和 5 列)。

打印一些你要处理的文件的信息是有帮助的，当你不了解这些文件时。了解文件的数目，每个文件的行数和列数，可以知道工作量的大小，以及知道文件的布局是否一致。

从多个 Workbooks 拼接数据

基础 Python

要用 python 从多个 workbooks 的所有 worksheets 垂直的拼接到一个输出文件，在文本编辑器输入如下代码并保存为 13excel_ concat_data_from_multiple_workbooks.py:

```
1 #!/usr/bin/env python3
```

```
2 import glob
3 import os
4 import sys
5 from datetime import date
6 from xlrd import open_workbook, xldate_as_tuple
7 from xlwt import Workbook
8 input_folder = sys.argv[1]
9 output_file = sys.argv[2]
10 output_workbook = Workbook()
11 output_worksheet = output_workbook.add_sheet('all_data_all_workbooks')
12 data = []
13 first_worksheet = True
14 for input_file in glob.glob(os.path.join(input_folder, '*.xls*')):
15 print os.path.basename(input_file)
16 with open_workbook(input_file) as workbook:
17 for worksheet in workbook.sheets():
18 if first_worksheet:
19 header_row = worksheet.row_values(0)
20 data.append(header_row)
21 first_worksheet = False
22 for row_index in range(1,worksheet.nrows):
23 row_list = []
24 for column_index in range(worksheet.ncols):
25 cell_value = worksheet.cell_value\
26 (row_index,column_index)
27 cell_type = worksheet.cell_type\
28 (row_index, column_index)
29 if cell_type == 3:
30 date_cell = xldate_as_tuple\
31 (cell_value,workbook.datemode)
32 date_cell = date(*date_cell[0:3])\
33 .strftime('%m/%d/%Y')
34 row_list.append(date_cell)
35 else:
36 row_list.append(cell_value)
37 data.append(row_list)
38 for list_index, output_list in enumerate(data):
39 for element_index, element in enumerate(output_list):
40 output_worksheet.write(list_index, element_index, element)
41 output_workbook.save(output_file)
```

第13行创建Boolean变量first_worksheet ,用于区别第一个 worksheet与其它 worksheet。

对于我们要处理的第一个 worksheet，第 18 行是真，所以我们添加标题行到 data，然后设置 first_worksheet 为假。对于第一个 worksheet 余下的行以及其它 worksheets 的行，我们跳过标题行并开始处理数据行。我们知道我们从第二行开始，因为第 22 行的 range 函数从 1 开始而不是从 0 开始。

要运行脚本，在命令行中输入如下并回车:python 13excel_
concat_data_from_multiple_workbooks.py "C:\Users\Clinton\Desktop"\
output_files\13output.xls

你可以找开输出文件 13output.xls 检查结果。

Pandas

Pandas 提供了 concat 函数来拼接 DataFrames。如果你要垂直的拼接 DataFrames，则用 axis=0。如果你要水平的拼接它们则用 axis=1。可选地，如果你要基于关键列拼接 DataFrames，pandas 的 merge 函数提供了 SQL join - like 操作（如果不合理，不用担心，我们在后面讨论数据库操作）。要垂直地拼接多个 workbooks 的所有 worksheets 到一个输出文件，在文本编辑器输入如下代码并保存为 pandas_concat_data_from_multiple_workbooks.py:

```
#!/usr/bin/env python3
import pandas as pd
import glob
import os
import sys
input_path = sys.argv[1]
output_file = sys.argv[2]
all_workbooks = glob.glob(os.path.join(input_path,'*.xls*'))
data_frames = []
for workbook in all_workbooks:
all_worksheets = pd.read_excel(workbook, sheetname=None, index_col=None)
for worksheet_name, data in all_worksheets.items():
data_frames.append(data)
all_data_concatenated = pd.concat(data_frames, axis=0, ignore_index=True)
writer = pd.ExcelWriter(output_file)
all_data_concatenated.to_excel(writer, sheet_name='all_data_all_workbooks',\
index=False)
writer.save()
```

要运行脚本，在命令行中输入如下并回车:python pandas_concat_data_from_multiple_workbooks.py
"C:\Users\Clinton\Desktop"\
output_files\pandas_output.xls

你可以找开输出文件 pandas_output.xls，检查结果。

按 Workbook 和 Worksheet 求和与平均值

基础 Python

要用基础 Python 为多个 workbook 计算 worksheet- 和 workbook-水平的统计量，在文本编辑器输入如下代码并保存为 14excel_sum_average_multiple_workbooks.py:

```
#!/usr/bin/env python3
import glob
import os
import sys
from datetime import date
from xlrd import open_workbook, xldate_as_tuple
from xlwt import Workbook
input_folder = sys.argv[1]
output_file = sys.argv[2]
output_workbook = Workbook()
output_worksheet = output_workbook.add_sheet('sums_and_averages')
all_data = []
sales_column_index = 3
header = ['workbook', 'worksheet', 'worksheet_total', 'worksheet_average',\
'workbook_total', 'workbook_average']
all_data.append(header)
for input_file in glob.glob(os.path.join(input_folder, '*.xls*')):
with open_workbook(input_file) as workbook:
list_of_totals = []
list_of_numbers = []
workbook_output = []
for worksheet in workbook.sheets():
total_sales = 0
number_of_sales = 0
worksheet_list = []
worksheet_list.append(os.path.basename(input_file))
worksheet_list.append(worksheet.name)
for row_index in range(1,worksheet.nrows):
try:
total_sales += float(str(worksheet.cell_value\
(row_index,sales_column_index))\
.strip('$').replace(',',''))
number_of_sales += 1.
except:
total_sales += 0.
number_of_sales += 0.
average_sales = '%.2f' % (total_sales / number_of_sales)
worksheet_list.append(total_sales)
```

```
39 worksheet_list.append(float(average_sales))
40 list_of_totals.append(total_sales)
41 list_of_numbers.append(float(number_of_sales))
42 workbook_output.append(worksheet_list)
43 workbook_total = sum(list_of_totals)
44 workbook_average = sum(list_of_totals)/sum(list_of_numbers)
45 for list_element in workbook_output:
46 list_element.append(workbook_total)
47 list_element.append(workbook_average)
48 all_data.extend(workbook_output)
49
50 for list_index, output_list in enumerate(all_data):
51 for element_index, element in enumerate(output_list):
52 output_worksheet.write(list_index, element_index, element)
53 output_workbook.save(output_file)
```

第 12 行创建空的列表 all_data 来贮存我们想要写入到输出文件的所有行。第 13 行创建 sales_column_index 变量来贮存 Sale Amount 列索引值。第 14 行为输出文件创建列标题的列表。第 16 行添加这个列表值到 all_data。第 19，20，21 行我们创建三个列表。list_of_totals 包含一个 workbook 的三个 worksheets 的总的 sale amounts。list_of_numbers 包含 sale amounts 数值用于计算 workbook 中所有 worksheets 的总 sale amounts。第三个列表，workbook_output，包含我们要写入到输出文件的所有输出列表。 第 25 行，我们创建列表 worksheet_list，贮存我们要保留的 worksheet 的所有信息。第 26 行和第 27 行，我们添加 workbook 的名字和 worksheet 的名字到 worksheet_list。相似地，第 38 行和 39 行，我们添加总的以及平均的 sale amounts 到 worksheet_list。第 42 行，我们添加 worksheet_list 到 workbook_output 来贮存 workbook 水平的信息。第 40 行和第 41 行我们添加 worksheet 总的 sale amounts 以及 sale amounts 数目到 list_of_totals 和 list_of_numbers。以便贮存所有 worksheets 的这些值。第 43 行和第 44 行用列表计算 workbook 的总的 sale amount 以及平均 sale amount。第 45 到 47 行，我们遍历 workbook_output 的列表（每个 workbook 有三个列表，因为每个 workbook 有三个 worksheets）并添加 workbook 水平的总 sale amounts 和平均 sale amounts 到每个列表。只要我们有了我们想要保留的信息（如三个列表，每个 worksheet 一个），我们扩展列表到 all_data。我们用扩展而不是 append，以便 workbook_output 的列表成为 all_data 的独立元素。即是说，处理了三个 workbooks 后，all_data 是 9 个元素的列表，其中每个元素是列表。如果我们不是用 append，all_data 中就只有三个元素，每个都是列表的列表。

要运行脚本，在命令行中输入如下并回车：

```
python 14excel_sum_average_multiple_workbooks.py "C:\Users\Clinton\Desktop"\
output_files\14output.xls
```

你可以找开输出文件 14output.xls 检查结果。

Pandas

Pandas 遍历多个 workbooks 并为每个 workbook 计算 worksheet 和 workbook 水平的统计量相对直接。本例，我们计算 workbook 中每个 worksheets 的统计量并拼接结果到 DataFrame。然后我们计算 workbook 水平的统计量，转换为 DataFrame，合并两个 DataFrame 到列表。一旦所有 workbook 水平的 DataFrames 都在列表中，我们拼接它们到一个 DataFrame 并写入输出文件。要从多个 workbooks 计算 worksheet 和 workbook 水平的统计量，在文本编辑器输入如下代码并保存为 pandas_sum_average_multiple_workbooks.py:

```
#!/usr/bin/env python3
import pandas as pd
import glob
import os
import sys
input_path = sys.argv[1]
output_file = sys.argv[2]
all_workbooks = glob.glob(os.path.join(input_path,'*.xls*'))
data_frames = []
for workbook in all_workbooks:
all_worksheets = pd.read_excel(workbook, sheetname=None, index_col=None)
workbook_total_sales = []
workbook_number_of_sales = []
worksheet_data_frames = []
worksheets_data_frame = None
workbook_data_frame = None
for worksheet_name, data in all_worksheets.items():
total_sales = pd.DataFrame([float(str(value).strip('$').replace(\
',',''))
for value in data.loc[:, 'Sale Amount']]).sum()
number_of_sales = len(data.loc[:, 'Sale Amount'])
average_sales = pd.DataFrame(total_sales / number_of_sales)
workbook_total_sales.append(total_sales)
workbook_number_of_sales.append(number_of_sales)
data = {'workbook': os.path.basename(workbook),
'worksheet': worksheet_name,
'worksheet_total': total_sales,
'worksheet_average': average_sales}
worksheet_data_frames.append(pd.DataFrame(data, \
columns=['workbook', 'worksheet', \
'worksheet_total', 'worksheet_average']))
worksheets_data_frame = pd.concat(\
worksheet_data_frames, axis=0, ignore_index=True)
workbook_total = pd.DataFrame(workbook_total_sales).sum()
```

```
workbook_total_number_of_sales = pd.DataFrame(\
workbook_number_of_sales).sum()
workbook_average = pd.DataFrame(\
workbook_total / workbook_total_number_of_sales)
workbook_stats = {'workbook': os.path.basename(workbook),
'workbook_total': workbook_total,
'workbook_average': workbook_average}
workbook_stats = pd.DataFrame(workbook_stats, columns=\
['workbook', 'workbook_total', 'workbook_average'])
workbook_data_frame = pd.merge(worksheets_data_frame, workbook_stats, \
on='workbook', how='left')
data_frames.append(workbook_data_frame)
all_data_concatenated = pd.concat(data_frames, axis=0, ignore_index=True)
writer = pd.ExcelWriter(output_file)
all_data_concatenated.to_excel(writer, sheet_name='sums_and_averages', \
index=False)
writer.save()
```

要运行脚本，在命令行中输入如下并回车:

```
python pandas_sum_average_multiple_workbooks.py "C:\Users\Clinton\Desktop"\
output_files\pandas_output.xls
```

你可以找开输出文件 pandas_output.xls 检查结果。

这一章我们讨论了很多。我们讨论了如何读和解析 Excel workbook，导航 Excel worksheet 的行，导航 Excel worksheet 的列，处理多个 Excel worksheet，处理多个 Excel workbooks，为每个 workbooks 和每个 worksheets 计算统计量。按照本章的例子，你已写了 14 个 Python 脚本。最好的部分是你已可以导航和处理 Excel 文件，商务中最常见的文件类型。而且，许多商务分支贮存数据于 Excel workbooks，你已有很多工具来处理这些 workbooks 的数据，而不管 workbooks 的数目，大小，每个 workbook 中有多少个 worksheets。你已有用计算机自动和放大处理 Excel workbooks 中数据的能力。接下来我们讨论数据库。因为数据库是常见的数据贮存方式，了解如何访问数据库中的数据是非常重要的。如果你知道了如何访问数据，你便可以一行一行的处理它，就像你处理 CSV 和 Excel 文件一样。

完成了这两章的例子，你可以准备处理数据库了。