前言

辦公自動化是指利用現代化設備和技術，代替辦公人員的部分手動或重複性業務活動，優質而高效地處理辦公事務，達到對資訊的高效運用，進而提升工作效率，達到輔助決策的目的。辦公自動化通常會利用到包括 Excel、Word、PowerPoint 等工具製作報表、文稿、簡報，以及收發郵件和處理檔案等工作，雖然微軟 Office 軟體相關工具提供了程式設計介面來達成辦公自動化，但是由於其具有佔用資源大等缺點，使用應用場合較受限。

目前，在辦公自動化的研究熱潮中，如何提高工作效率也成為一個具有挑戰性的任務。Python 在辦公自動化領域的應用越來越受歡迎，它能做到大量檔案的批次生成和處理。本書以 Python 3.10 版本為基礎進行編寫，有系統地介紹以 Python 為基礎的辦公自動化技術。

本書會深入介紹 Python 在辦公自動化方面的應用：包括 Python 程式設計基礎篇、Excel 資料自動化處理篇、Word 文書自動化處理篇、簡報投影片自動化製作篇、郵件自動化處理篇、檔案自動化處理篇。

本書內容結構

第 1 篇：Python 程式設計基礎篇

第 1 章介紹 Python 軟體的特點與優勢，以及如何快速搭建 Python 3.10 版本的開發環境。

第 2 章介紹 Python 程式設計基礎，包括資料型別、基礎語法、常用進階函數和程式設計技巧。

第 3 章介紹利用 Python 進行資料的準備，包括資料的讀取、資料的索引、資料的切片、資料的刪除、資料的排序、資料的聚合、資料的透視交叉分析、資料的合併等。

第 2 篇：Excel 資料自動化處理篇

第 4 章介紹利用 Python 進行資料處理，包括重複值的處理、缺失值的處理、異常值的處理等。

第 5 章介紹利用 Python 進行資料分析，包括統計描述性分析、相關分析、線性迴歸分析。

第 6 章介紹利用 Python 進行資料視覺化，包括對比型、趨勢型、比例型、分佈型等基本圖表的繪製方法。

第 3 篇：Word 文書自動化處理篇

第 7 章介紹文書自動化處理，包括應用場景及環境搭建、Python-docx 程式庫案例示範。

第 8 章介紹利用 Python 進行文書自動化處理，包括使用 Python-docx 程式庫自動化處理對頁首頁尾、樣式、文字等進行處理。

第 9 章介紹利用 Python 製作企業營運月報 Word 版，包括使用 Python-docx 程式庫整理及清洗門市店面銷售資料、營運資料的視覺化分析、批次製作企業營運月報等。

第 4 篇：簡報投影片自動化製作篇

第 10 章介紹簡報投影片自動化製作，包括應用場景及環境搭建、Python-pptx 程式庫案例示範。

第 11 章介紹利用 Python 進行簡報投影片自動化製作，包括自動化插入文字、圖表、表格和圖案等內容。

第 12 章介紹利用 Python 製作企業營運月報簡報投影片，包括製作商品銷售分析報告、製作客戶留存分析報告。

第 5 篇：郵件自動化處理篇

第 13 章介紹利用 Python 批次發送電子郵件，包括郵件伺服器概述、發送電子郵件等。

第 14 章介紹利用 Python 來獲取電子郵件，包括獲取郵件的內容、解析郵件的內容等。

第 15 章介紹利用 Python 自動發送電商會員郵件，包括電商會員郵件行銷、提取未付費的會員資料、發送制式郵件提醒和發送制式的簡訊提醒等。

第 6 篇：檔案自動化處理篇

第 16 章介紹利用 Python 進行檔案自動化處理，包括檔案和資料夾的基本操作、檔案的解壓縮操作、顯示目錄樹下的檔案名稱、修改目錄樹下的檔案名稱、合併目錄樹下的資料檔案。

本書特色定位

1） 內容新穎，講解詳細。

本書是一本內容新穎的 Python 技術書，詳細介紹了基於 Python 的辦公自動化技術，對於初學者有很大助益。書中詳細介紹了大量辦公自動化案例，便於讀者練習和實作。

2） 由淺入深，循序漸進。

本書以案例為主線，既包括軟體應用與操作的方法和技巧，又融入了辦公自動化的案例實戰，使讀者透過對本書的學習，能夠輕鬆、快速地掌握辦公自動化技術。

3） 案例豐富，高效學習。

本書以 Python 3.10 版本為基礎進行講解，同時為了使讀者能夠快速提升辦公自動化的綜合能力，本書中的案例都盡可能地貼近實際工作情況。

本書適用讀者

本書的內容和案例適合網路、財務、顧問等相關行業的資料分析人員閱讀，可以作為學校相關專業科系學生的參考用書，也可以作為職場人員學習 Python 辦公自動化的自學用書。

由於作者能力有限，書中難免存在一些疏漏和不足，希望同行和讀者給予批評與指正。

第 3 章
利用 Python 進行資料準備

在實際的專案中，我們需要從不同的資料來源中提取資料，進行準確性檢查、轉換和合併整理，並載入資料庫，從而供應用程式分析和應用，這一過程就是資料準備。資料只有經過清洗、貼標籤、注釋和準備後，才能成為寶貴的資源。本章將詳細介紹使用 Python 進行資料準備的方法，包括資料的讀取、索引、切片、刪除、排序、聚合、交叉透視、合併等。

3.1 資料的讀取

在分析資料之前，需要準備「食材」，也就是資料，例如像商品的屬性資料、客戶的訂單資料、客戶的退單資料等。本節將介紹 Python 讀取本機離線資料、Web 線上資料、資料庫資料等各種儲存形式的資料。

3.1.1 讀取本機離線資料

1. 讀取 .txt 格式的資料

使用 Pandas 程式庫中的 read_table() 函數，Python 可以直接讀取 .txt 格式的資料，程式碼如下所示（其中文字檔的路徑是筆者電腦的路徑）：

```
import pandas as pd

data = pd.read_table('F:\Python+office-Samples\ch03\orders.txt', delimiter=',',
encoding='UTF-8')
print(data[['order_id','order_date','cust_id']])
```

在 JupyterLab 中執行上述程式碼，輸出結果如下所示。

```
             order_id order_date     cust_id
0      CN-2014-100007   2014/1/1  Cust-11980
1      CN-2014-100001   2014/1/1  Cust-12430
2      CN-2014-100002   2014/1/1  Cust-12430
3      CN-2014-100003   2014/1/1  Cust-12430
4      CN-2014-100004   2014/1/1  Cust-13405
...               ...        ...         ...
19485  CN-2020-101502  2020/6/30  Cust-18715
19486  CN-2020-101503  2020/6/30  Cust-18715
19487  CN-2020-101499  2020/6/30  Cust-19900
19488  CN-2020-101500  2020/6/30  Cust-19900
19489  CN-2020-101505  2020/6/30  Cust-21790

[19490 rows x 3 columns]
```

2. 讀取.csv 格式的資料

使用 Pandas 程式庫中的 read_csv() 函數，Python 可以直接讀取 .csv 格式的資料，程式碼如下所示：

```
#連接 CSV 資料檔案
import pandas as pd

data = pd.read_csv('F:\Python+office-Samples\ch03\orders.csv', delimiter=',',
encoding='UTF-8')
print(data[['order_id','order_date','cust_type']])
```

在 JupyterLab 中執行上述程式碼，輸出結果如下所示。

```
             order_id order_date     cust_type
0      CN-2014-100007   2014/1/1       消費者
1      CN-2014-100001   2014/1/1      小型企業
2      CN-2014-100002   2014/1/1      小型企業
3      CN-2014-100003   2014/1/1      小型企業
4      CN-2014-100004   2014/1/1       消費者
...               ...        ...        ...
19485  CN-2020-101502  2020/6/30        公司
19486  CN-2020-101503  2020/6/30        公司
19487  CN-2020-101499  2020/6/30       消費者
19488  CN-2020-101500  2020/6/30       消費者
19489  CN-2020-101505  2020/6/30       消費者

[19490 rows x 3 columns]
```

3. 讀取 Excel 檔資料

使用 Pandas 程式庫中的 read_excel() 函數，Python 可以直接讀取 Excel 檔資料，程式碼如下所示：

```
#連接 Excel 資料檔案
import pandas as pd

data = pd.read_excel('F:\Python+office-Samples\ch03\orders.xls')
print(data[['order_id','order_date','product_id']])
```

在 JupyterLab 中執行上述程式碼，輸出結果如下所示。

```
             order_id order_date     product_id
0      CN-2014-100007 2014-01-01  Prod-10003020
1      CN-2014-100001 2014-01-01  Prod-10003736
2      CN-2014-100002 2014-01-01  Prod-10000501
3      CN-2014-100003 2014-01-01  Prod-10002358
4      CN-2014-100004 2014-01-01  Prod-10004748
...               ...        ...            ...
19485  CN-2020-101502 2020-06-30  Prod-10002305
19486  CN-2020-101503 2020-06-30  Prod-10004471
19487  CN-2020-101499 2020-06-30  Prod-10000347
19488  CN-2020-101500 2020-06-30  Prod-10002353
19489  CN-2020-101505 2020-06-30  Prod-10004787
```

```
[19490 rows x 3 columns]
```

3.1.2 讀取 Web 線上資料

Python 可以讀取 Web 線上資料，這裡選取的資料集是 UCI 上的紅酒資料集，該資料集是對義大利同一地區種植的葡萄酒進行化學分析的結果，這些葡萄酒來自 3 個不同的品種，分析確定了 3 種葡萄酒中每種葡萄酒含有的 13 種成分的數量。不同種類的酒品，它的成分也會有所不同，透過對這些成分的分析就可以對不同的特定的葡萄酒進行分類分析，原始資料集共有 178 個樣本數、3 種資料類別，每個樣本有 13 個屬性。

Python 讀取紅酒線上資料集的程式碼如下所示：

```
#匯入相關程式庫
import numpy as np
import pandas as pd
import urllib.request

url = 'http://archive.ics.uci.edu//ml//machine-learning-databases//wine//wine.data'

raw_data = urllib.request.urlopen(url)
dataset_raw = np.loadtxt(raw_data, delimiter=",")
df = pd.DataFrame(dataset_raw)
print(df.head())
```

在 JupyterLab 中執行上述程式碼，輸出結果如下所示。

3.1.3　讀取常用資料庫中的資料

1. 讀取 MySQL 資料庫中的資料

Python 可以直接讀取 MySQL 資料庫中的資料，連接之前需要安裝 pymysql 程式庫。例如，統計匯總資料庫 orders 資料表中 2020 年不同類型商品的銷售額和利潤額，程式碼如下所示：

```
#連接 MySQL 資料庫
import pandas as pd
import pymysql

#讀取 MySQL 資料庫中的資料
conn = pymysql.connect(host='192.168.93.207',port=3306,user='root',password='
Wren_2014',db='sales'charset='utf8')
sql_num = "SELECT category,ROUND(SUM(sales/10000),2) as
sales,ROUND(SUM(profit/10000),2) as profit FROM orders where dt=2020 GROUP BY
category"
data = pd.read_sql(sql_num,conn)
print(data)
```

在 JupyterLab 中執行上述程式碼，輸出結果如下所示。

```
     category   sales   profit
0     辦公用品    79.13    5.65
1       技術      78.35    4.11
2       傢俱      87.51    4.54
```

2. 讀取 SQL Server 資料庫中的資料

Python 可以直接讀取 SQL Server 資料庫中的資料，連接之前需要安裝 pymssql 庫。例如，查詢資料庫 orders 表中 2020 年利潤額在 400 元以上的所有訂單，程式碼如下所示：

```
#連接 SQL Server 資料庫
import pandas as pd
import pymssql

#讀取 SQL Server 資料庫中的資料
conn = pymssql.connect(host='192.168.93.207',user='sa',password='Wren2014',
database='sales',charset='utf8')
sql_num = "SELECT order_id,sales,profit FROM orders where dt=2020 and profit>400"
data = pd.read_sql(sql_num,conn)
print(data)
```

在 JupyterLab 中執行上述程式碼，輸出結果如下所示。

```
          order_id     sales    profit
0   CN-2020-100004   10514.03   472.33
1   CN-2020-100085    7341.60   479.14
2   CN-2020-100115    6668.90   472.64
3   CN-2020-100113   10326.40   408.29
4   CN-2020-100148    5556.60   486.06
...
56  CN-2020-101326   11486.16   420.28
57  CN-2020-101365    7188.30   406.57
58  CN-2020-101370    8346.74   408.02
59  CN-2020-101471    7982.10   407.52
60  CN-2020-101509    6919.08   468.66

[61 rows x 3 columns]
```

NOTE

以上是以安裝了 MySQL 及 SQL Server 資料庫的情況下才能執行，且伺服器主機位置、使用者帳號、密碼會依讀者安裝指定而不相同，而資料庫中要有個 'sales' 資料表，存放的字元編碼是 'utf8'。上述範例僅為示範之用，讓讀者了解其連接語法。

3.2 資料的索引

索引是對資料中一欄或多欄的值進行排序的一種結構，使用索引可以快速存取資料中的特定資訊。本節將會介紹 Python 如何建立索引、重構索引、調整索引等，假設使用的資料為「2020 年兩個學期學生考試成績」。

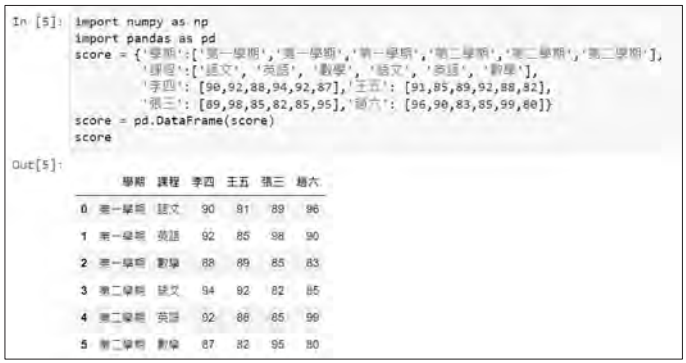
3.2.1 set_index() 函數：建立索引

在建立索引之前，先建立一個由 4 名學生考試成績構成的資料集，程式碼如下所示：

```
import numpy as np
import pandas as pd
score = {'學期':['第一學期','第一學期','第一學期','第二學期','第二學期','第二學期
'],'課程':['語文', '英語', '數學', '語文', '英語', '數學'],
'李四': [90,92,88,94,92,87],'王五': [91,85,89,92,88,82],
'張三': [89,98,85,82,85,95],'趙六': [96,90,83,85,99,80]}
score = pd.DataFrame(score)
score
```

執行上述程式碼，建立的資料集如下所示。

```
      學期   課程  李四  王五  張三  趙六
0   第一學期   語文    90    91    89    96
1   第一學期   英語    92    85    98    90
2   第一學期   數學    88    89    85    83
3   第二學期   語文    94    92    82    85
4   第二學期   英語    92    88    85    99
5   第二學期   數學    87    82    95    80
```



使用 index（索引）可以查看所有資料集，預設是從 0 開始步長為 1 的數值索引，程式碼如下所示：

```
score.index
```

程式碼輸出結果如下所示。

```
RangeIndex(start=0, stop=6, step=1)
```

set_index() 函數可以將其一欄轉換為列索引，程式碼如下：

```
score1 = score.set_index(['課程'])
score1
```

程式碼輸出結果如下所示。

```
課程       學期  李四  王五  張三  趙六
語文   第一學期    90    91    89    96
英語   第一學期    92    85    98    90
數學   第一學期    88    89    85    83
語文   第二學期    94    92    82    85
英語   第二學期    92    88    85    99
數學   第二學期    87    82    95    80
```

set_index() 函數還可以將其多欄轉換為列索引，程式碼如下：

```
score1 = score.set_index(['學期','課程'])
score1
```

程式碼輸出結果如下所示。

```
    學期   課程   李四  王五   張三   趙六
第一學期   語文    90   91     89    96
           英語    92   85     98    90
           數學    88   89     85    83
第二學期   語文    94   92     82    85
           英語    92   88     85    99
           數學    87   82     95    80
```

在預設情況下，索引欄的欄位會從資料集中移除，但是透過設定 drop 參數也可以將其保留下來，程式碼如下：

```
score.set_index(['學期','課程'],drop=False)
```

程式碼輸出結果如下所示。

```
    學期   課程       學期   課程  李四  王五  張三  趙六
第一學期   語文   第一學期   語文    90    91    89    96
           英語   第一學期   英語    92    85    98    90
           數學   第一學期   數學    88    89    85    83
第二學期   語文   第二學期   語文    94    92    82    85
           英語   第二學期   英語    92    88    85    99
           數學   第二學期   數學    87    82    95    80
```

3.2.2 unstack() 函數：重構索引

reset_index() 函數的功能與 set_index() 函數的功能相反，層次化索引的級別會被轉移到資料集中的欄裡面，程式碼如下所示：

```
score1.reset_index()
```

程式碼輸出結果如下所示。

```
      學期   課程  李四  王五  張三  趙六
0   第一學期   語文    90    91    89    96
1   第一學期   英語    92    85    98    90
2   第一學期   數學    88    89    85    83
3   第二學期   語文    94    92    82    85
```

```
4   第二學期   英語    92    88    85   99
5   第二學期   數學    87    82    95   80
```

可以透過 unstack() 函數對資料集進行重構，其功能類似於 pivot() 函數的功能，不同之處在於，unstack() 函數是針對索引或標籤的，即將欄索引轉成最內層的列索引；而 pivot() 函數則是針對欄的值，即指定某欄的值作為列索引，程式碼如下所示：

```
score1.unstack()
```

程式碼輸出結果如下所示。

```
          李四                王五                張三                趙六
課程      數學  英語  語文  數學  英語  語文  數學  英語  語文  數學  英語  語文
  學期
第一學期    88    92    90    89    85    91    85    98    89    83    90    96
第二學期    87    92    94    82    88    92    95    85    82    80    99    85
```

此外，stack() 函數是 unstack() 函數的逆運算，程式碼如下：

```
score1.unstack().stack()
```

程式碼輸出結果如下所示。

```
    學期   課程   李四   王五   張三   趙六
第一學期   數學     88     89     85     83
          英語     92     85     98     90
          語文     90     91     89     96
第二學期   數學     87     82     95     80
          英語     92     88     85     99
          語文     94     92     82     85
```

3.2.3　swaplevel() 函數：調整索引

有時，可能需要調整索引的順序，swaplevel() 函數接受兩個級別編號或名稱，並返回一個互換了級別的新物件。例如，對學期和課程的索引級別進行調整，程式碼如下所示：

```
score1.swaplevel('學期','課程')
```

程式碼輸出結果如下所示。

```
課程      學期  李四  王五  張三  趙六
語文    第一學期   90   91   89   96
英語    第一學期   92   85   98   90
數學    第一學期   88   89   85   83
語文    第二學期   94   92   82   85
英語    第二學期   92   88   85   99
數學    第二學期   87   82   95   80
```

sort_index() 函數可以對資料進行排序，參數 level 設定需要排序的欄。需要注意的是，這裡的欄包含索引欄，第 1 欄是 0（「學期」欄），第 2 欄是 1（「課程」欄），程式碼如下所示：

```
score1.sort_index(level=1)
```

程式碼輸出結果如下所示。

```
    學期   課程  李四  王五  張三  趙六
第一學期   數學   88   89   85   83
第二學期   數學   87   82   95   80
第一學期   英語   92   85   98   90
第二學期   英語   92   88   85   99
第一學期   語文   90   91   89   96
第二學期   語文   94   92   82   85
```

3.3 資料的切片

在解決各種實際問題的過程中，常常會遇到從某個物件中提取部分資料的情況，切片操作可以完成這個任務。本節將會介紹 Python 如何提取一欄或多欄資料、一列或多列資料、指定區域的資料等，假設使用的資料為「2020 年第二學期學生考試成績」。

3.3.1 提取一欄或多欄資料

在介紹資料切片之前，需要建立一個由 4 名學生學習成績構成的資料集，程式碼如下所示：

```
import numpy as np
import pandas as pd
score = {'李四': [90,91,87,92,95,85],'王五': [91,85,89,92,88,82],'張三':
[89,98,85,82,85,95],'趙六': [96,90,83,85,99,80]}
score = pd.DataFrame(score, index=['數學','語文','英語','物理','化學','生物'])
score
```

執行上述程式碼，建立的資料集如下所示。

```
      李四  王五  張三  趙六
數學    90    91    89    96
語文    91    85    98    90
英語    87    89    85    83
物理    92    92    82    85
化學    95    88    85    99
生物    85    82    95    80
```

可以提取某一欄資料，程式碼如下所示：

```
score['王五']
```

程式碼輸出結果如下所示。

```
數學    91
語文    85
英語    89
物理    92
化學    88
生物    82
Name: 王五, dtype: int64
```

可以提取某幾欄連續和不連續的資料，如提取兩欄資料，程式碼如下：

```
score[['王五','趙六']]
```

程式碼輸出結果如下所示。

```
      王五  趙六
數學    91    96
語文    85    90
英語    89    83
物理    92    85
化學    88    99
生物    82    80
```

3.3.2　提取一列或多列資料

可以使用 loc() 函數和 iloc() 函數獲取特定列的資料。其中，iloc() 函數是透過列號獲取資料的，而 loc() 函數則是透過列標籤索引資料的。例如，提取第 2 列資料，程式碼如下所示：

```
score.iloc[1]
```

程式碼輸出結果如下所示。

```
李四    91
王五    85
張三    98
趙六    90
Name: 語文, dtype: int64
```

也可以提取幾列資料，需要注意的是，列號也是從 0 開始的，區間是左閉右開。例如，提取第 3 列～第 5 列的資料，程式碼如下所示：

```
score.iloc[2:5]
```

程式碼輸出結果如下所示。

```
     李四  王五  張三  趙六
英語   87   89   85   83
物理   92   92   82   85
化學   95   88   85   99
```

如果不指定 iloc() 函數的列索引的初始值，則預設從 0 開始，即第 1 列，程式碼如下所示：

```
score.iloc[:3]
```

程式碼輸出結果如下所示。

```
     李四  王五  張三  趙六
數學   90   91   89   96
語文   91   85   98   90
英語   87   89   85   83
```

3.3.3 提取指定區域的資料

使用 iloc() 函數還可以提取指定區域的資料。例如，提取第 3 列～第 5 列的資料、第 2 欄～第 4 欄的資料，程式碼如下所示：

```
score.iloc[2:5,1:3]
```

程式碼輸出結果如下所示。

```
      王五   張三
英語    89     85
物理    92     82
化學    88     85
```

此外，如果不指定區域中欄索引的初始值，那麼從第 1 欄開始，程式碼如下：

```
score.iloc[2:5,:3]
```

程式碼輸出結果如下所示。

```
      李四   王五   張三
英語    87     89     85
物理    92     92     82
化學    95     88     85
```

同理，如果不指定欄索引的結束值，那麼提取後面的所有欄。

3.4　資料的刪除

Pandas 程式庫之中有 3 個用來刪除資料的函數：drop()、drop_duplicates()、dropna()。其中 drop() 函數用於刪除列或欄中的資料，drop_duplicates() 函數用於刪除重復資料，dropna() 函數用於刪除空值。本節將會介紹如何刪除一列或多列資料、一欄或多欄資料、指定的列表物件等，假設使用的資料為「2020 年第二學期學生考試成績」。

3.4.1　刪除一列或多列資料

在介紹如何用 Pandas 程式庫刪除資料之前，還是建立一個關於 4 名學生學習成績的資料集，程式碼如下所示：

```
import numpy as np
import pandas as pd
score = {'李四': [90,91,87,92,95,85],'王五': [91,85,89,92,88,82],'張三':
[89,98,85,82,85,95],'趙六': [96,90,83,85,99,80]}
score = pd.DataFrame(score, index=['數學','語文','英語','物理','化學','生物'])
score
```

執行上述程式碼，建立的資料集如下所示。

```
      李四  王五  張三  趙六
數學    90    91    89    96
語文    91    85    98    90
英語    87    89    85    83
物理    92    92    82    85
化學    95    88    85    99
生物    85    82    95    80
```

drop() 函數預設是刪除列資料，參數是列索引。例如，刪除一列資料，程式碼如下所示：

```
score.drop('化學')
```

程式碼輸出結果如下所示。

```
      李四  王五  張三  趙六
數學    90    91    89    96
語文    91    85    98    90
英語    87    89    85    83
物理    92    92    82    85
生物    85    82    95    80
```

還可以刪除幾列連續和不連續的資料。例如，刪除化學和生物的考試成績，程式碼如下所示：

```
score.drop(['化學','生物'])
```

程式碼輸出結果如下所示。

```
      李四  王五  張三  趙六
數學    90    91    89    96
語文    91    85    98    90
英語    87    89    85    83
物理    92    92    82    85
```

3.4.2 刪除一欄或多欄資料

對於欄資料的刪除，可以透過設定參數 axis=1 來達成（如果不設定參數 axis，則 drop() 函數的參數預設為 axis=0，即對行進行操作）。例如，刪除 1 名學生的考試成績，程式碼如下所示：

```
score.drop('李四',axis=1)
```

程式碼輸出結果如下所示。

```
      王五  張三  趙六
數學    91    89    96
語文    85    98    90
英語    89    85    83
物理    92    82    85
化學    88    85    99
生物    82    95    80
```

也可以透過設定 axis='columns' 達成刪除欄資料。例如，刪除兩名學生的考試成績，程式碼如下所示：

```
score.drop(['李四','趙六'],axis='columns')
```

程式碼輸出結果如下所示。

```
      王五  張三
數學    91    89
語文    85    98
英語    89    85
物理    92    82
化學    88    85
生物    82    95
```

此外，drop() 函數的可選參數 inplace，預設值為 False，即不改變原陣列，如果將其值設定為 True，則原始陣列直接會被修改，程式碼如下所示：

```
score.drop('張三',axis=1,inplace=True)
score
```

程式碼輸出結果如下所示。

```
      李四  王五  趙六
數學    90    91    96
語文    91    85    90
英語    87    89    83
物理    92    92    85
化學    95    88    99
生物    85    82    80
```