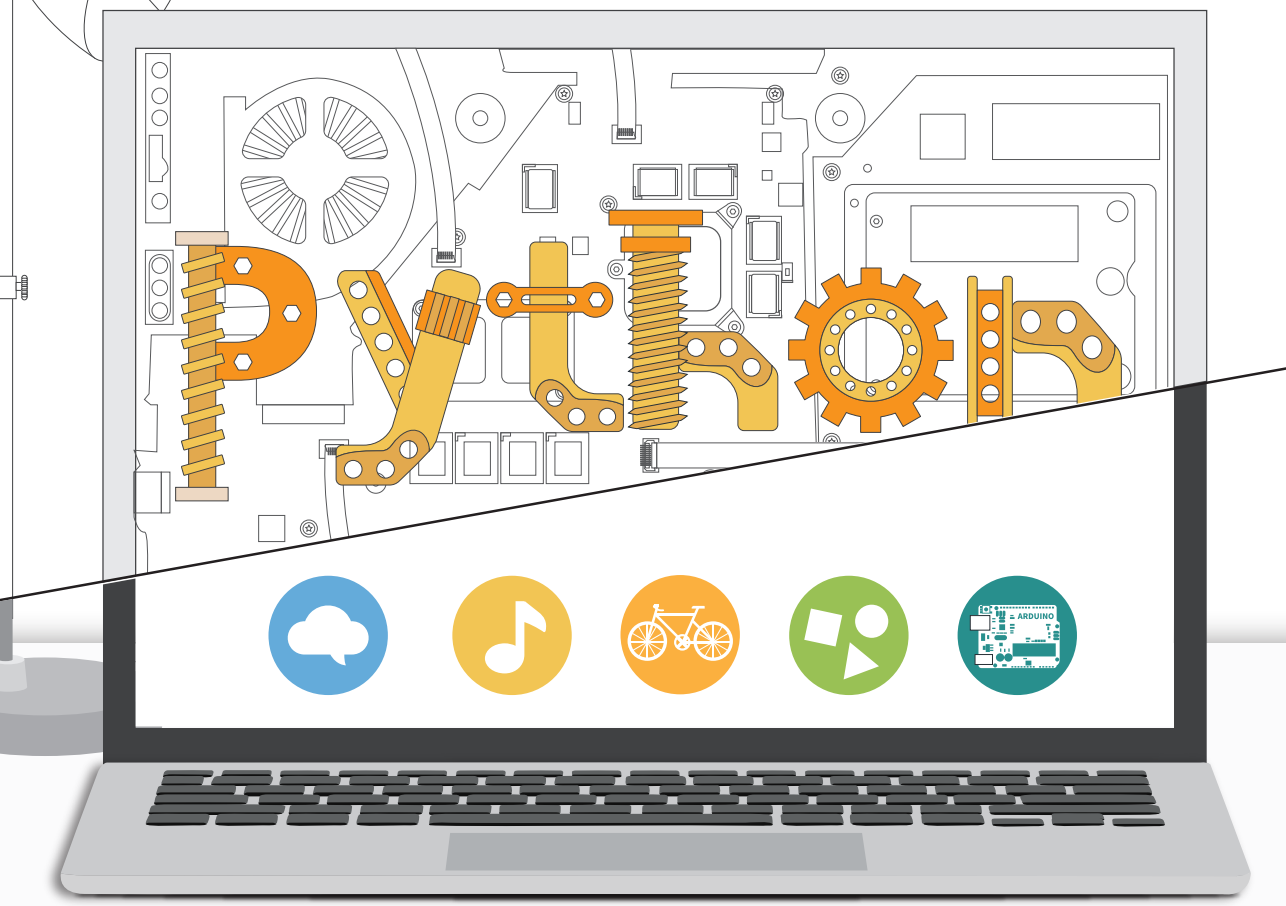
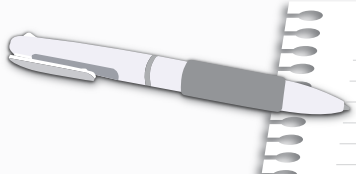
</PLUS> 推動大學{程式設計}教學
Programming Learning for University Students

Python程式設計與資料分析應用





教育部
國立臺灣師範大學/資訊工程學系

教育部 推動大學程式設計教學計畫

隨著世界各國對 STEAM 科技教育的重視，運算思維已逐漸成為現代公民皆應具備之基本能力。全面培育國民程式設計與運算思維是厚植我國資訊國力、促進數位經濟發展的重要政策方向，因此教育部於 106 年度起推動「教學創新試辦計畫」、「技專校院教學創新先導計畫」，於 107 年度納入「高等教育深耕計畫」，以經費挹注各校開設程式設計相關課程，增加學生學習程式設計之機會。

教育部在引導各大學規劃高教深耕計畫時，將大學生應具備基本程式設計能力的任務訂為各校的重點工作，並訂於 108 年度時達到 50% 學生修習過程式設計相關課程的量化目標。然而，前期研究發現大專院校實施程式設計教學時常遇到幾項問題，包含學生學習動機低落、教學教材內容過於艱澀、師資人數與經驗不足、學習環境配套有待提升及各校學生特質與需求差異大等。因此，教育部委託國立政治大學等十三所大學成立了「推動大學程式設計教學計畫」，此計畫旨在協助教育部推動高教深耕計畫中大學程式設計教育的普及，同時達到量化目標與確保教學品質。透過鼓勵各校制訂程式設計教育的推動策略與非資訊領域學生的修習課程地圖，希望能確保各校程式設計相關課程的永續發展，以有效提升學生未來參與數位經濟的基本能力。透過生活化與專業化的客製化課程設計，希望能提高學生學習動機，並從紮實的學習成果中培養學習的興趣，而教師也能從教師社群與教學回饋中，獲得更好的教學資源，提升整體教學品質。

序

在資訊全面數位化而且萬物聯網的時代，源源不絕的數據資料如大浪般襲來，唯有能分析資料、掌握內涵的人才得以站在浪頭上脫穎而出。資料分析與應用的價值，已在諸如教育 、醫療、金融、消費、製造、體育、管理等等各個領域受到重視，同時也帶動一股資料科學的風潮。人工智慧革命的來臨，號稱將消滅一半的現有工作；而資料科學家卻被哈佛商業評論譽為二十一世紀最性感的職業。擁有資料分析能力，儼然將成為各大產業競相爭取的人才。教育部「推動大學程式設計教學」計畫協助國內各大學推動程式設計教育，下設各分項計畫於不同應用領域編製與推廣教材。本書為「資料分析領域」分項計畫之教材彙編成果，以 Python 程式與其資料分析應用為主軸，共含 11 章，前半段為基礎程式概念和 Python 語言教學，後半段為資料科學導論、資料視覺化和多個資料分析專題應用。

作者

蔣宗哲

現任國立臺灣師範大學資訊工程學系副教授，臺大資工博士，主要研究領域為演化計算、多目標最佳化與資源規畫。於臺師大資工系服務十餘年，開設程式設計相關課程二十餘班次，熱愛教學與分享。教學理念是三有一無：有趣、有料、有效、無拘束。

江政杰

現任德明財經科技大學資訊科技系助理教授，臺灣師範大學資訊教育系所博士班，主要研究電腦視覺與機器學習等領域，並將相關技術開發具實用性的多媒體系統。近年嘗試將影像處理、電腦視覺及機器學習的各項技術，應用於智慧教室環境的研究開發，並參與程式設計之通識課程教材開發與教學。

蔡天怡

現任國立臺灣大學圖書資訊學系副教授，美國威斯康辛大學麥迪遜校區圖書資訊學博士，研究領域為資訊行為、資訊素養、資訊資源與服務，開設課程有資訊心理學、圖書館管理、資訊視覺化、數位圖書館等。

洪暉鈞

現任中央大學網路學習科技研究所助理教授，國立清華大學資訊系統與應用博士，專長於教育資料探勘、學習分析研究，也專精於機器學習與巨量資料系統、行動學習創新系統設計與應用。

作者

胡筱薇

現任東吳大學巨量資料管理學院副教授，國立中央大學資管系博士，專精於資料探勘與機器學習，近年來致力於巨量資料探勘以及社群網路分析應用之研究，同時也成立了資料實驗室 (Data Lab)，並透過實際的專案項目，培養巨量資料分析人才。

張幼珍

現任元智大學化學工程與材料科學學系副教授，美國馬里蘭州立大學化工博士，專精於氣膠科學、計算流體力學、微奈米粉體製備、元件與製程開發，曾任臺灣氣膠科學研究學會第一副會長、常務理事、2007 年第五屆亞洲氣膠會議秘書長。曾獲元智大學傑出教學獎、創新教學特優獎、優等獎、及佳作獎，致力推廣問題導向、創客實作等學生為中心的學習法。

林宜徵

現任安譜人工智慧音樂公司任全職首席音樂長，研發 AI 作曲技術，美國威斯康辛大學麥迪遜分校音樂藝術博士，學術專長為作曲、電子音樂、專注於音樂科技領域之研究 ，包含演算法作曲、類神經網路於自動作曲之應用、Unity 3D 與 VR 開發，作品橫跨古典器樂、電聲結合 3D 遊戲影像之多媒體作品。

目錄

Chapter 1

來一句 Python 文

/ 蔣宗哲、江政杰

學寫程式就像學一種語言，只是溝通的對象從人變成了電腦。本章將手把手帶領讀者初探寫程式是怎麼一回事，分為三小節，內容包括：環境建置與使用、資料運算（變數、資料型態和運算子）以及互動運算（內建函式之使用）。學習完本章後，讀者將在電腦上安裝好 Python 開發工具 IDLE，並可撰寫簡單的互動程式，從鍵盤輸入資料、進行簡易運作並在螢幕上顯示運算結果。

1-1 寫程式這回事

講到電腦，大家心裡想到的是甚麼呢？是很大台的超級電腦？一般桌上型電腦？還是工作使用的筆記型電腦呢？其實現代的電腦已經以各種不同的形態，廣泛的存在我們的身邊，最具有代表性的是你身上的智慧型手機了，手機內的各種應用程式幫助我們處理生活大小事，其他像平板、運動手環、穿戴裝置等，都是不同類型的電腦。

除了身邊用到的資訊裝置，現代社會中各行各業已經深刻地依賴電腦運作，尤其是近年來人工智慧的技術出現，讓資訊科技更快速地應用在社會運作的各個領域。大家或許已經發現，突然之間程式設計變得非常熱門，因為在可見的未來，人們將可透過程式設計的能力，直接控制身邊的資訊設備。透過程式設計的學習，能夠讓我們理解電腦實際運作的『思考』方式，在職場上更能夠增加跨領域的技能。

學習程式語言的目的是為了和電腦溝通，指揮電腦進行我們指定的運算流程。隨著人工智慧技術的進步，雖然人類已能使用自然語言指揮電腦 （例：Siri），但其用途和規模仍有限。寫程式，指的是我們將需要電腦執行的運算流程，依循程式語言的文法規定撰寫出來。一般來說，我們會將程式存成純文字格式的檔案，這些程式檔案被稱為程式源碼 (source code)。不同的程式語言有不同的慣用副檔名，本書要談的 Python 程式語言，通常以 .py 作為副檔名 (如圖 1-1)。

圖 1-1 Python 程式源碼及文字內容示例

所謂學寫程式，其實是在學習如何運用程式語言來描述人類的思維，以期符合機器的運作方式，藉由機器的高效運算達成人類期望完成的工作。程式語言和一般我們認知的語言相同，都會有單字、句型、文法和語意等等需要學習。不過程式語言簡單多了，通常只要學會十來個單字，三五個句型，就能完成很多常見的工作。

簡單來說，寫程式包含三個基本步驟：撰寫程式源碼、翻譯成機器碼、以及執行機器碼。後面兩個步驟通常交由軟體來負責。我們可以使用任一種文字編輯軟體（如記事本也行）寫下程式源碼，再藉由程式開發工具來負責翻譯與執行。現今的程式語言都有整合性（整合編輯、翻譯和執行）的開發工具。這裡我們使用一種簡便的工具，稱為 IDLE。

安裝 Python IDLE 開發工具

讀者連上網址 https://www.python.org/downloads/ 後，會看到一個下載的圖示，點擊圖示後，會在電腦中存下一個安裝檔。雙擊該安裝檔後，會跳出提示畫面。（此處以 Windows 安裝 Python 3.7.0 版本畫面為例，讀者若有較新的版本也無妨。）提示畫面中，請勾選最下面兩個小方塊，然後點擊 Install Now 即可。

圖 1-2 IDLE 整合開發工具下載頁面

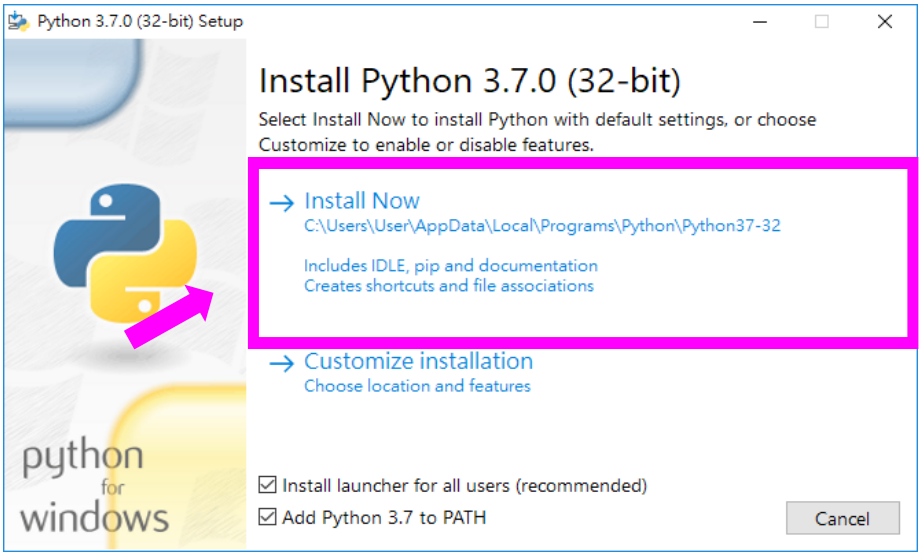


圖 1-3 IDLE 安裝畫面

安裝完畢後，找到 IDLE 軟體圖示，點擊以便執行。在 Windows 桌面的左下方有放大鏡，點擊放大鏡後，鍵入 idle 文字，應該會看到如下圖左邊的 IDLE 軟體圖示。點擊後，會開啟如下圖右邊的視窗，左上角有 Python … Shell 文字，我們稱這個視窗為 Shell 視窗。

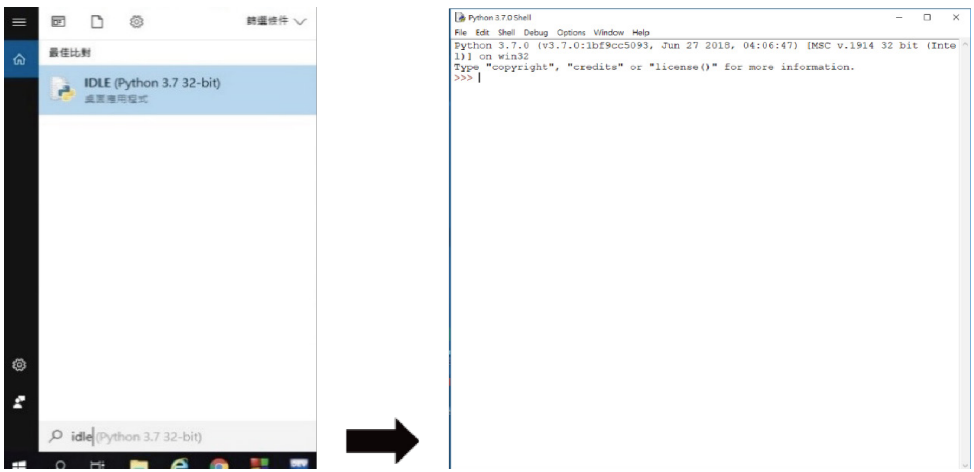


圖 1-4 Python 開發環境啟動範例

Hello World 程式

現在我們已經安裝好 Python 的程式開發環境了。前一節我們說過，程式源碼會以文字檔來儲存。我們首先開啟一個新檔案（這個動作就像平常使用文書軟體或簡報軟體一樣），請參見圖 1-5 的步驟 ❶。此時，我們會有一個新的空白視窗，我們稱此視窗為檔案視窗，在此編寫程式碼，如圖 1-5 的步驟 ❷。此處，我們先嘗試打上

```
print('Hello World')
```

這樣的文字。接著，儲存檔案，如步驟 ❸。此時，請指定一個目錄（例如桌面），然後把程式存為 ____.py，其中 ____ 可由讀者自行決定。儲存成功後，讀者應可以在指定的目錄中看到一個檔案，如步驟 ❹ 所示。接著，步驟 ❺，從功能列選擇 Run（執行程式的意思），再選擇 Run Module（或者按熱鍵 F5 一鍵搞定），此時若程式尚未存檔，會詢問是否存檔，如步驟 ❻。「確定」存檔後，便會在 Shell 視窗出現程式的執行結果了！

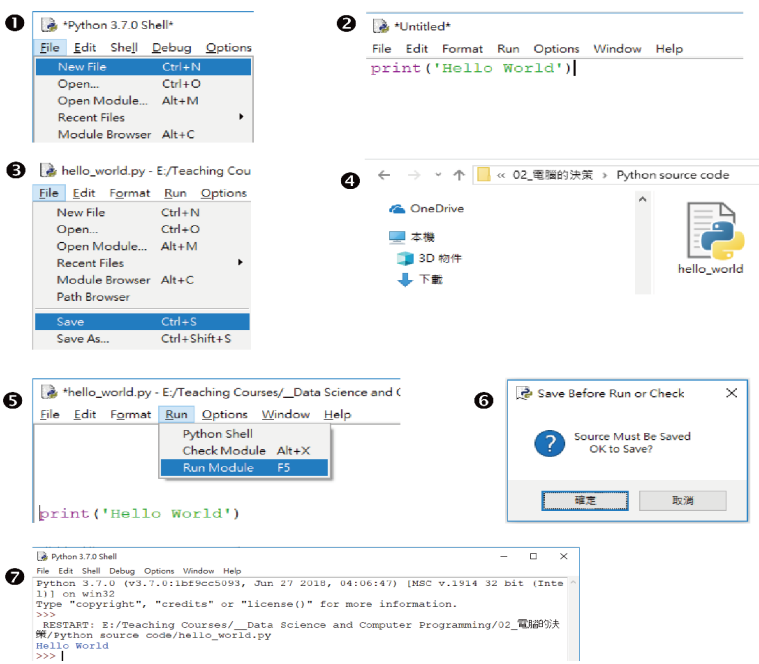


圖 1-5 Hello World 程式

隨堂練習

1.1.1 請撰寫一個程式，請寫個程式，幫你跟大家打招呼吧。

請顯示『大家好，我是 OOO』這裡的 OOO 請寫入你的名字。

執行結果：
大家好，我是 OOO

（參考解答）

```
01  print('大家好，我是 OOO')
```

1-2 資料運算

前一節讓 Python 跟大家打個招呼 say hello，引導讀者初步接觸 Python 的程式撰寫，現在我們來嘗試電腦最擅長的能力：運算。電腦最初的設計，是為了能夠快速進行繁複的科學運算，這也是電腦英文 computer 的由來。電腦面對大量繁複的運算時，不會累不會煩不會無聊不會抗議不會粗心，絕對是我們處理計算的好幫手。程式是直接控制電腦運作的媒介，所以學習程式首先會接觸的就是運算。

程式語言內可處理的運算種類很多，我們先來看看最常見的算數運算吧。表 1-1 列出六種運算範例、包括Python 的運算子符號以及運算結果。舉例來說，在 Python 程式中，乘法是以星號 (*) 來表示；百分號 (%) 代表取餘數的運算，而連續兩個星號 (**) 則代表次方運算。

表 1-1：Python 的算術運算子

算術運算	Python 程式碼	運算結果
加	`1+1`	`2`
減	`20-3`	`17`
乘	`3*7`	`21`
除	`100/8`	`12.5`
取餘數	`100%8`	`4 (100/8 = 12…4)`
次方	`4**2`	`16`

我們可以直接在 Shell 視窗打入運算式，按下 Enter 鍵便可得到計算結果。如果要作的運算很多，我們可依前一節所學，將程式碼撰寫在一個源碼檔案中，然後再按下 F5 執行，最後在 Shell 視窗查看結果。

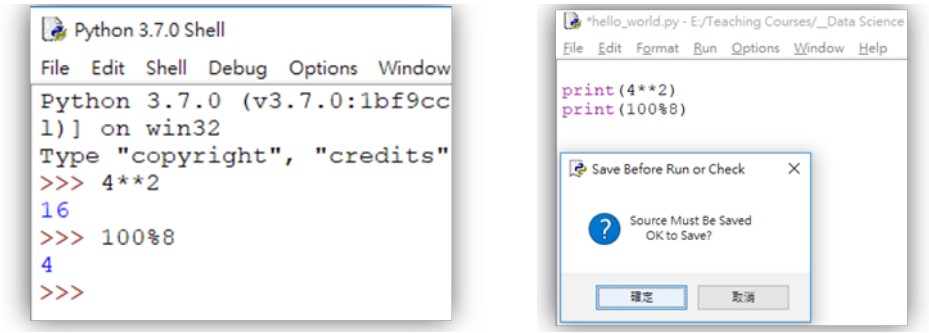


圖 1-6 撰寫 Python 程式進行算術運算

當我們同時使用多個算術運算，如 `1+10*3` 同時用到加法和乘法運算時，我們需知道運算的順序。Python 程式語言訂定的順序符合我們的直覺，表 1-1 中的六個運算子，優先權最高的是次方，接著是乘、除、餘，最後是加減。同優先權的運算，則是由左至右計算。如果要改變既有的優先權，可以使用小括號來完成。注意，小時候我們學過的中括號 [] 和大括號 {} 在 Python 程式裡另有用途，若有多層次，我們還是會持續使用小括號。

表 1-2：算術運算的優先次序

Python 程式碼	運算優先次序	運算結果
`1+10*3`	先算乘法，再算加法。	31
`20*2-4*3`	先算乘法，再算減法。	28
`3*4**2-1`	先算 4**2，再算 3*16，最後減 1。	47
`(1+2)*(3+7)`	先算加法（因為有小括號），再算乘法。	30
`((1+2)*3-1)*2`	先算 1+2，再算 3*3-1，最後乘以 2。	16

Python 的除法運算有兩種，一種是小數除法 (/)，另一種是整數除法 – 連續兩個斜線符號 (//)。整數除法較特別，計算結果是小於原除法結果的最大整數值（就是數學中的 floor 運算，一般記為 $\lfloor x \rfloor$。）另外，Python 中可將浮點數轉換成整數，以 `int()` 來操作，其中 `int` 取自整數 integer 的前三個字母。

表 1-3：Python 的除法運算

Python 程式碼	運算結果	說明
`11/2`	5.5	一般除法，得到小數結果
`11//2`	5	整數除法，得到整數結果 (較小的最靠近整數)
`int(11/2)`	5	強制轉型，得到整數結果 (無條件捨去小數部位)
`-11/2`	-5.5	一般除法，得到小數結果
`-11//2`	-6	整數除法，得到整數結果 (較小的最靠近整數)
`int(-11/2)`	-5	強制轉型，得到整數結果 (無條件捨去小數部位)

隨堂練習

1.2.1. 請使用 Python 執行下列的運算，執行前先用手計算看看，結果跟你想的是否相同？

(1) 20+1/3

(2) 20/3+1

(3) 20/(3+1)

(4) 20+1//3

(5) 20//3+1

(6) 20//(3+1)

(7) 2*17%5*2

(8) 17/2%5

(9) -17%5

(10) '12'*3

1-3 互動運算

前一節我們學會了以 Python 程式進行算術運算，但，運算的數值從哪裡來呢？我們只能計算固定的數值嗎？當然不是囉。本節我們將先介紹程式中最重要的元素 – 變數與型態，然後教大家如何讓程式獲得使用者的輸入，最後以一個簡單的互動程式作結尾。

變數

程式中的資料都存在記憶體中，現在的電腦多半有好幾 GB 的記憶體，也就是好幾十億個位元組的記憶體。一個整數大概花費 4 或 8 個位元組，所以我們的記憶體可以存放很多很多的整數資料。那麼，為了讓我們方便取用與指稱某一段記憶體中的數值，我們以一個有意義的名稱來代表，這就是變數的概念。

以下面的 Python 程式碼來說明。我們以 = 運算來生成一個變數並指派它內容。第 01 行的意思程式執行時會向電腦要求一（小）段記憶體空間，裡面存放整數值 10，並且將這段空間命名為 x。此後，當我們在程式中寫到 x 的時候，就代表著那一段記憶體。因此，程式第 02 行是利用 print() 函式來列印 x 所代表的記憶體的內容。由於第一行將 10 存入這段記憶體，所以第二行程式會列印出 10。第 03 行我們又定義了變數 y，令其代表 20。第 04 行示範 print() 可以一次印出多個變數、甚至是運算 x+y 的結果。多個列印結果之間預設以一個空白字元隔開。

```
01  x = 10
02  print(x)
03  y = 20
04  print(x, y, x+y)
```

執行結果：

```
10
10 20 30
```

資料型態

到目前為止，我們都在進行整數或小數型態的運算。在程式中，另一種常見的資料型態是文字型態。Python 中稱為字串 str 型態，其中 str 是字串 string 的前三個字母。在程式中，10 是整數型態、10.0 是小數型態，而 '10' 則是字串型態。Python 中以一對單引號或雙引號包起來的都視為字串型態。

了解資料型態對於撰寫程式十分重要，因為不同型態的資料可能允許不同的運算；即使允許同樣的運算，其運算的意義與結果也可能不同。表 1-4 比較了對整數和字串進行乘法運算與加法運算的結果。整數的部份就不贅述，字串對整數的乘法是複製的意思，如 '12'*3 的結果是 '12' 重複 3 次；字串對字串的加法是串接的意思，如 '12'+'3' 的結果是兩個字串串接起來，形成 '123'。

表 1-4：Python 的算術運算子

Python 程式碼	運算結果	說明
12*3	36	整數乘法，得到整數結果
'12'*3	'121212'	字串乘法，得到字串結果
'12'*'2'	失敗	字串不支援和字串相乘，翻譯失敗
12+3	15	整數加法，得到整數結果
'12'+3	失敗	字串不支援和整數相加，翻譯失敗
'12'+'3'	'123'	字串加法，得到字串結果

讀入資料

print() 函式可以用來輸出資料給程式的使用者看，那要如何讓使用者輸入資料呢？啊哈，輸入的英文是 input，Python 正是以 input() 函式來取得使用者的輸入資料。以下兩行程式，第 01 行以 input() 函式取得使用者輸入的文字，並以 x 指稱這段字串資料。第 02 行將 x 指稱的資料印出來。建議讀者立刻將這兩行程式鍵入檔案視窗中，按下 F5，體驗程式如何與使用者互動。按下 F5 後，Shell 視窗會有一個游標閃爍，此時讀者可嘗試輸入 198，應會看到程式回應了同樣的輸入文字 198。

```
01  x = input()
02  print(x)
```

執行結果 1：

```
198
198
```

執行結果 2：

```
Python is fun.
Python is fun.
```

當讀者按 F5 執行上述兩行程式時，可能會感到疑惑，因為 Shell 視窗中沒有任何提示訊息，讓人不知所措。我們可以在 input() 函式中增加一段提示訊息，這段訊息是文字型態，因此以一對單引號包起來。讀者可嘗試鍵入並執行下列程式碼，以了解 input() 和 print() 中列印文字的方式。

```
01  x = input('請輸入一個整數值 > ')
02  print('你輸入的是', x)
```

執行結果：

```
請輸入一個整數值 > 198
你輸入的是 198
```

相信有讀者已經迫不及待要對 input() 取得的使用者資料進行運算，但卻對運算結果感到疑惑。以下面程式為例，當使用者輸入 198，原先我們預期會得到 396 (198×2 = 396) 的結果，但是程式卻輸出 198198。聰明的讀者應該會想起前一頁我們提到字串與整數相除的運算。是的，此處 x 代表的是字串資料 '198' 而非整數資料 198。這是因為 input() 函式所帶回的資料是字串型態所致。

```
01  x = input(' 請輸入一個整數值 > ')
02  print(' 此數乘以 2 的結果是 ', x*2)
```

執行結果：

```
請輸入一個整數值 > 198
此數乘以 2 的結果是 198198
```

這該如何是好？不用擔心，稍早在表 1-3 我們曾提到 `int()` 可以將資料轉換成整數型態（同理，`str()` 可以把資料轉換成字串型態。），所以我們可以將程式修改如下。我們新增了第 02 行程式，這行的意思是取出 `x` 所代表的資料，將該資料（由字串型態）轉為整數型態，然後再令 `x` 代表這個新的資料。此例中，第 02 行會令 `x` 代表的資料由 `'198'` 變成 `198`。因此第 03 行的 `x*2` 便會進行整數運算，得到我們期待的結果。

```
01  x = input(' 請輸入一個整數值 > ')
02  x = int(x)
03  print(' 此數乘以 2 的結果是 ', x*2)
```

執行結果：

```
請輸入一個整數值 > 198
此數乘以 2 的結果是 396
```

隨堂練習

1.3.1 前一個程式跟大家打招呼後，你的名字到哪裡了呢？

延續前一個練習，請先用變數 name 放入你的名字再打招呼

請顯示『大家好，我是 OOO』這裡的 OOO 請寫入你的名字

執行結果：

```
大家好，我是 OOO
```

（參考解答）

```
01 name = 'OOO'
02 print('大家好，我是', name)
```

1.3.2 寫個程式，用變數 yy, mm, dd, day 分別記錄年月日與星期幾，然後顯示出來。

執行結果：

```
今天是 2020 年 5 月 5 日 Tuesday
```

（參考解答）

```
01 yy = 2020
02 mm = 5
03 dd = 5
04 day = 'Tuesday'
05 print('今天是', yy, '年', mm, '月', dd, '日', day)
```

1.3.3 請撰寫一個程式，首先列印出程式功能「加法程式」，接著提示使用者輸入兩個整數值，最後列印出此二整數值的和。

執行結果：

```
加法程式
請輸入第一個整數值 > 33
請輸入第二個整數值 > 44
答案是 77
```

（參考解答）

```
01 print('加法程式')
02 x = input('請輸入第一個整數值 > ')
03 y = input('請輸入第二個整數值 > ')
04 total = int(x)+int(y)
05 print('答案是', total)
```

Chapter 2

電腦的決策

/ 蔣宗哲、江政杰

有一個古早的廣告詞說「電腦也會撿土豆」，意思是電腦能根據不同情況作出不同的回應動作。本章將介紹程式語言的兩大句型之一：選擇句型 (`if`)，分為四小節，內容包括：基本二分法句型、多分法句型、巢狀句型以及邏輯運算子。學習完本章後，讀者將具備能力把日常生活中常見的流程圖轉譯成 Python 程式，並可進行資料的比較和分類。

2-1 如果句型 if...

平均的危機

前一章我們學會了撰寫 Python 程式進行簡單的數值運算，數值資料分析最常見的是計算資料的個數、極值（最大 / 小值）、平均值等等。學習程式撰寫最好的方法就是多練習寫程式，所以在本章的開始，我們先撰寫一個求平均的程式當練習，順便複習一下基本語法，以下是一個參考程式：

E2-1-1.py:

```
01  print('計算平均的程式')
02  total = int(input('請輸入總和 > '))
03  count = int(input('請輸入個數 > '))
04  print('答案是', total/count)
```

對於一個程式員，撰寫出完整的程式只是第一步，更重要的下一步是測試並確認程式運作無誤。最簡單的測試方式是輸入一些資料，觀察程式的反應是否如預期。

執行結果：測試案例 1

```
計算平均的程式
請輸入總和 > 100
請輸入個數 > 5
答案是 20.0
```

執行結果：測試案例 2

```
計算平均的程式
請輸入總和 > 42
請輸入個數 > -8
答案是 -5.25
```

執行結果：測試案例 3

```
計算平均的程式
請輸入總和 > 100
請輸入個數 > 0
Traceback (most recent call last):
     File "divide by zero.py", line 4, in <module>
       print('答案是', total/count)
   ZeroDivisionError: division by zero
```

上述三個測試案例中，案例 1 是一般人通常會測試的「正常資料」，案例 2 和案例 3 則出現異常的輸入，包含負的個數，甚至是 0 作為個數。當程式試圖計算除以 0 的結果時，程式便會出現錯誤訊息 ZeroDivisonError，代表這是一個不應該發生的運算。既然這是一個不該發生的運算，我們是否能修改程式

來避免呢？我們希望的程式流程是這樣的：

1. 使用者輸入總和與個數。
2. 判斷個數：

   2.1 如果個數為正時，計算平均。

   2.2 如果個數不為正時，提示使用者輸入錯誤。

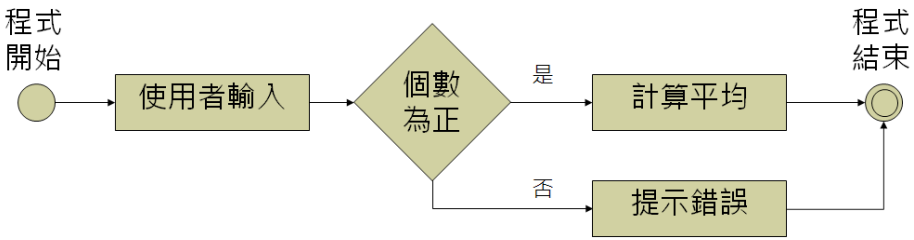


圖 2-1 程式流程圖 (平均的危機)

美好的如果

大多數的程式語言都有個「如果」的句型。

- 以中文為例：

  如果個數為正，計算並列印平均；

  否則，提示使用者輸入錯誤。

- 以英文為例：

  If the count is positive, calculate and print the average;

  otherwise, show an error message.

而在 Python 語言中，也有「如果」的句型。

在 `if` 句型中，程式會依據條件判斷式的真偽結果，選擇不同的程式片段來執行，因此 `if` 句型也常被稱為是一種「選擇控制」的句型。

語法說明：**if ... else ...**
if 條件判斷式：（別忘記冒號歐！） 　　條件成立會作的動作（可以不只一行） else: 　　條件不成立會作的動作（可以不只一行）

以下面的程式片段來舉例，如果 count>0 這個條件式是真的（或說條件式成立），也就是 count 值為正，程式便會執行第 02 行；如果條件式是假的（或說不成立），程式便執行第 04 行。

```
01  if count>0:
02      print('答案是', total/count)
03  else:
04      print('無法計算 - 個數需為正值')
```

將上述的程式片段整合至前面求取平均的程式碼中，我們便可以避開計算個數不為正時的平均值，如下程式範例：

```
01  print('計算平均的程式')
02  total = int(input('請輸入總和 > '))
03  count = int(input('請輸入個數 > '))
04  if count>0:
05      print('答案是', total/count)
06  else:
07      print('無法計算 - 個數需為正值')
```

要注意，Python 對於「如果」句型有嚴格的排版格式規定，若不符合規定，會出現 expected an indented block 的語法錯誤。簡單來說，被 if/else 控制的程式片段需要往內縮，並且要內縮一樣的空白數，常見的慣例是內縮四個空白。現今的程式編輯器都有自動內縮的功能，一般情況下，程式員不需要刻意去調整縮排。

格式錯誤範例 1

```
01  if a>0:
02  print(a, '是正值')
03  else:
04  print(a, '不是正值')
```

格式錯誤範例 2

```
01  if a>0:
02  print(a, '是正值')
03    print('這行格式錯了')
04  else:
05  print(a, '不是正值')
```

若條件不成立時不需要執行特定的運算，可以省略掉句型中的 else 部份。例如下面的程式執行到第 05 行時會判斷 school 變數的內容，如果是「臺師大」便會執行第 06 行；若 school 內容不是「臺師大」，則程式會跳過第 06 行，往下執行第 07 行。

E2-1-2.py:

```
01  print('薩莉亞餐廳的程式')
02  school = input('請輸入就讀學校 > ')
03  count = int(input('請輸入人數 > '))
04  price = count*100
05  if school == '臺師大':
06      price = price*0.9 # 師大人打九折
07  print('總價', int(price), '元')
```

執行結果 1：

```
薩莉亞餐廳的程式
請輸入就讀學校 > 臺師大
請輸入人數 > 10
總價 900 元
```

執行結果 2：

```
薩莉亞餐廳的程式
請輸入就讀學校 > 某大
請輸入人數 > 10
總價 1000 元
```

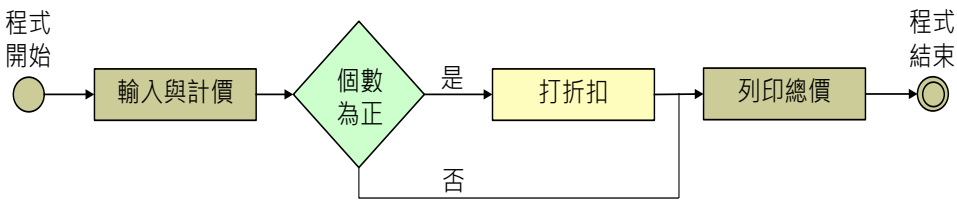
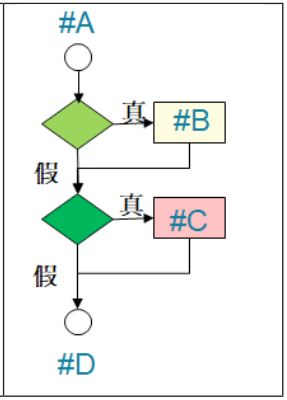


圖 2-2 程式流程圖 (省略 else)

如果句型的寓意

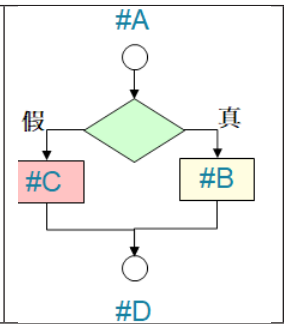
在撰寫 if 句型時，有些人會寫出類似下面的程式碼。這個程式使用了兩個獨立的 if 句子，第一個會在 count 值大於 0 時執行動作 B，第二個會在 count 值小於等於 0 時執行動作 C。雖然這個程式片段在功能面上和本頁下方使用一個 if 句子的程式片段相同，但程式可讀性卻不如下方的程式來得好。

```
01  # action A
02  if count > 0:
03      # action B
04  if count <= 0:
05      # action C
06  # action D
```



一般來說，兩個獨立 if 句子的流程圖如上面右圖，它代表著程式有四種可能的執行流程：AD、ABD、ACD 和 ABCD。本例的程式中因為兩個 if 句子的條件式有互補的關係，使得程式僅剩下 ABD 和 ACD 兩種執行流程。然而，閱讀上方程式碼的條件式並理解它們的關係，會增加閱讀程式的負擔。

```
01  # action A
02  if count > 0:
03      # action B
04  else:
05      # action C
06  # action D
```



相對而言，本頁下方的程式碼，在程式員一看到 if … else … 之後，便能立刻掌握這是一個「二分法」的選擇控制，要嘛執行動作 B，要嘛執行動作 C，沒有第三種流程了。

布林值

前面說到 if 句型會依「條件式」的真假來選擇程式片段，在 Python 程式中，所謂的真與假，可用布林 (boolean) 型態（bool） 的特定值來表示：True 代表真，False 代表假。以下列程式為例，第一個 if 句子的條件式明白寫了 True，因此程式會執行第 02 行，列印出 1 True；第二個 if 句子的條件式明白寫了 False，因此程式會執行第 09 行，列印出 2 False。

E2-1-3.py:

```
01  if True:
02      print('1 True')
03  else:
04      print('1 False')
05
06  if False:
07      print('2 True')
08  else:
09      print('2 False')
```

執行結果：

```
1 True
2 False
```

關係運算子

上面的程式只是為了舉例說明 if 句型是依條件式運算結果為 True 或 False 來選擇程式流程，我們不會真的寫出 E2-1-3 這樣的程式。（那個程式中我們已經確知程式的執行流程，可以只留下第 02 和 09 行，並刪除掉其它程式碼。）一般 if 句子的條件式會是一或多個關係運算來構成。表 2-1 列出 Python 的關係運算子 (operator) 以及運算範例。

表 2-1：Python 的關係運算子

意義	關係運算子	結果為真 True	結果為假 False
相等	==	3==3	3==4
不相等	!=	3!=4	4!=4
大於	>	4>3	3>4
大於等於	>=	4>=4	3>=4
小於	<	3<4	4<3
小於等於	<=	3<=4	5<=4

在目前我們所學過的運算子當中，關係運算子的運算優先度最低，也就是它們會最後計算：「次方」高於「乘 / 除 / 餘」高於「加 / 減」高於「關係運算」。

Python 程式碼	運算次序	運算結果
3*3==9	(3*3)==9	True
2+3>1+4	(2+3)>(1+4)	False

隨堂練習

2.1.1 請問以下程式碼的輸出為何？

```
01  x, y = 11, 20
02  print('A')
03  if 100 > 3:
04      print('B')
05  if 100 < 200:
06      print('C')
07  if x/2 > 5:
08      print('D')
09  print('E')
10  if x+y < 40:
11      print('F')
12  if x*2 == y:
13      print('G')
14  if x <= y:
15      print('H')
16  if x+3*y >= x*3+y:
17      print('I')
18  if x != y:
19      print('J')
```

解答：

執行結果：

```
A
B
C
D
E
F
H
I
J
```

2.1.2 修改「求平均」程式以達下述功能：給予使用者一次輸入錯誤（即輸入不為正的個數）的機會，當使用者連續錯誤兩次時，程式便結束。

2-2 多分法

如果不只兩種可能？

在前一節「求平均」的程式中，我們依據個數「為正值」與「不為正值」將程式流程分成兩個路線。如果我們將依據一個數值「為正」、「為 0」和「為負」分成三個執行路線，該怎麼作呢？

Python 的 `if` 句型除了 `if` 和 `else` 區塊，還有一個 `elif`（是的，就是 `else` 加 `if`）區塊。我們先看一個例子來學習它的用法：

E2-2-1.py:

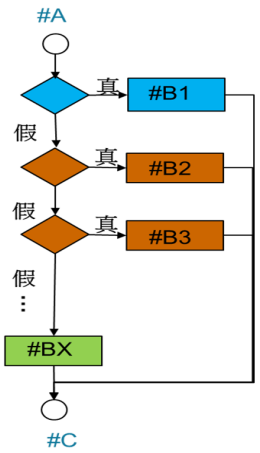
```
01  v = int(input('請輸入一個整數值 > '))
02  if v>0:
03      print('正值')
04  elif v==0:
05      print('零')
06  else:
07      print('負值')
08  print('程式結束')
```

上述程式在第 02 行會判斷 `v` 的值是否大於 `0`：如果是，會執行第 03 行，然後就執行第 08 行了。是的，請注意，當程式執行其中一個被控制的區塊，就不可能再進入其它區塊了，也就是在 `if … elif` 的語法中，我們必須謹慎安排條件判斷的執行順序，放在前面的條件一旦符合為真，在同一組 `if … elif` 後面的條件就不會被檢驗。如果 `v` 的值沒有大於 `0`，程式會檢查第 04 行的條件：如果 `v` 的值為 `0`，程式會依序執行第 05 和 08 行。如果 `v` 的值為負，則程式會在第 02 和 04 行的條件式都為假的情況下，跳到第 06 行，然後執行第 07、08 行。

執行結果 1：	執行結果 2：	執行結果 3：
請輸入一個整數值 > 3 正值 程式結束	請輸入一個整數值 > -4 負值 程式結束	請輸入一個整數值 > 0 零 程式結束

語法說明：**if ... elif ... else ...**

```
#A
if condition1:
    #B1
elif condition2:
    #B2
elif condition3:
    #B3
...
else:
    #BX
#C
```



上表描述了 if 的完整句型以及執行流程。總結來說，一個 if 的句子會以 if 作為開頭，接著有 0 個以上的 elif 區塊，最後會有 0 或 1 個 else 區塊。當句子只有 if 區塊的時候，控制的是某段程式碼的執行與否（就像一個開關，開就執行，關就不執行）；當句子有 if 和 else 區塊時，表現的是一種二分法；而再加上 elif 區塊後，便可以表現多分法。

接下來呈現兩個常見的多分法型式，E2-2-2 是單值多分法，以變數是否等於特定值來控制程式流程，用的是相等（==）關係運算子，而 E2-2-3 是區間多分法，以變數值是否落在一段區間內來控制程式碼，用的是大於等於（>=）關係運算子。

E2-2-2.py:

```
01  grade = int(input(' 你的年級 >'))
02  if grade == 1:
03      print(' 一年級 ')
04  elif grade == 2:
05      print(' 二年級 ')
06  elif grade == 3:
07      print(' 三年級 ')
08  elif grade == 4:
09      print(' 四年級 ')
10  else:
11      print(' 大大級 ')
```

雖然 if 句型不一定要有 else 的部份，但建議保留此部份作為檢查時的最後一道關卡，以便發現預期之外的變數值（如輸入 5）。

E2-2-3.py:

```
01  price = 10
02  number = int(input())
03  if number>=12:
04      price = price*number*0.9   #12 人以上
05  elif number>=6:
06      price = price*number*0.95   #6~11 人
07  elif number>=2:
08      price = price*number*0.99   #2~5 人
09  else:
10      price = price*number   #1 人
11  print(number, price)
```

注意「由上而下」的判斷順序，當程式執行第 05 行的條件式判斷時，代表 number 沒有大於等於 12，即 number 小於 12。

隨堂練習

2.2.1 多分法練習：請撰寫一個程式，使用者輸入一個分數，請依以下區間輸出等第。

分數區間	等第
90~100	A+
85~89	A
80~84	A-
77~79	B+
73~76	B
70~72	B-

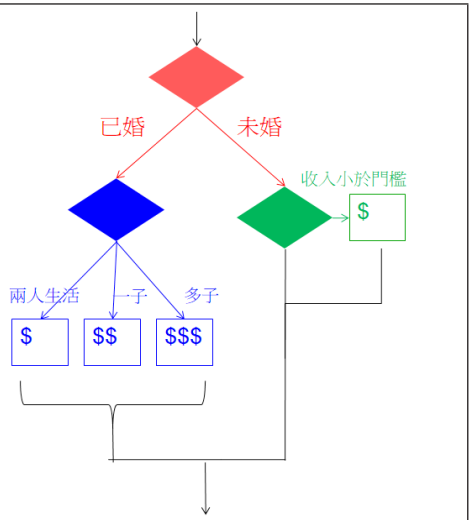
2.2.2 多分法練習：請撰寫一個程式，使用者輸入中文星期幾，程式輸出對應的英文字。

中文輸入	英文輸出
星期日	Sunday
星期一	Monday
星期二	Tuesday
星期三	Wednesday
星期四	Thursday
星期五	Friday
星期六	Saturday

2-3 巢狀控制

前一節的尾聲我們說到 if 句型可以表現出「開 / 關」、「二分法」和「多分法」這些「單層」的控制流程。事實上，在 if/elif/else 的程式區塊裡面，我們仍然可以繼續使用 if 句型，這就構成了所謂「巢狀 (nested)」控制。我們以下面的調薪程式片段來講解：

```
01  if married:
02      if num_kids == 0:
03          bonus = 1000
04      elif num_kids == 1:
05          bonus = 2000
06      else:
07          bonus = 4000
08  else:
09      if salary <= 50000:
10          bonus = 500
```



在這段程式碼中，首先第 01 行的 if 和第 08 行的 else 構成一個以婚姻狀態 (married 變數) 來控制的二分法。而這個二分法的 if 區塊（已婚狀態）中，我們再使用 if … elif … else 句型表現一個以小孩人數 (num_kids) 來控制的三分法，當小孩人數為 0、1 和 2 人以上時，分別加薪 1000、2000 和 4000 元。外層二分法的 else 區塊（未婚狀態）中，則以一個簡單的 if 句子控制，當薪資不高於 50000 元時，加薪 500 元。讀者可以對照右邊的流程圖以清楚理解左邊程式碼的執行流程。

隨堂練習

2.3.1 下面的程式碼可以判斷三個變數 A, B, C 中的最小值，請畫出其流程圖。

```
01  if A < B:
02      if A < C:
03          m = A
04      else:
05          m = C
06  else:
07      if B < C:
08          m = B
09      else:
10          m = C
11  print('最小值為',m)
```

2.3.2 這個程式共有幾種可能的執行流程（以 A/B/C/D 區塊來看）？

```
01  print('計算平均的程式')
02  total = int(input('請輸入總和 > '))
03  count = int(input('請輸入個數 > '))        #A
04  if count>0:
05      print('答案是', total/count)           #B
06  else:
07      count = int(input('個數須為正值，請輸入個數 > '))
08      if count>0:
09          print('答案是', total/count)
10      else:
11          print('無法計算 - 個數需為正值')    #D
```

解答：

編號	執行流程	測試案例
1	AC	依序輸入 100 3
2	ABC	依序輸入 100 0 3
3	ABD	依序輸入 100 0 0

2-4 邏輯運算子

在我們繼續學習更多程式知識前，我們先看看讀者是否已完全理解巢狀控制的概念，請看下列程式片段：說出三組 (month, day) 的數值可以分別讓程式執行第 03、05 和 07 行。

```
01  if month == 8:
02      if day == 15:
03          print('中秋節快樂!')
04      else:
05          print('平常日')
06  else:
07      print('平常日')
```

執行第 03 行	執行第 05 行	執行第 07 行
month: 8 day: 15	month:8 day: 13	month: 7 day: 1

建議讀者在撰寫或閱讀程式碼時，都能常作這樣的練習，有助於增強對於程式流程的掌握，也可提高測試與除錯的效率。好，現在我們進入本節主題：上述程式碼能否以更精簡、易懂的方式來撰寫呢？答案當然是可以。Python 語言中有三個邏輯運算子：而且 (and)、或者 (or) 及非 (not)，前兩個是二元運算子，運算時需要左、右兩個布林值作為運算元；而第三個是一元運算子，運算時僅需右邊一個布林值作為運算元。

邏輯運算子 and

因為 and 運算是二元運算，左右兩邊的運算元都是布林值（還記得布林值只有 False 和 True 兩種嗎？），因此 and 運算共有四種運算組合。以表 2-4-1 的第一列為例，當值 a 為 False 而值 b 為 False 時，a and b 這個邏輯運算的結果是 False。從這四種運算組合來看，我們可以得到一個簡單的規則，只有當兩個運算元皆為真 (True) 時，and 運算的結果才會為真，其餘的運算結果都是假 – 這也符合我們對「而且」這個詞的理解。

表 2-4-1: and 運算子的運算結果

a	b	a and b
False	False	False
False	True	False
True	False	False
True	True	True

邏輯運算子可以幫助我們簡化 if 句型，例如前頁的程式可改寫如下：

```
01  if month == 8 and day == 15:
02      print('中秋節快樂!')
03  else:
04      print('平常日')
```

這個程式從巢狀結構簡化成了單層結果，在閱讀時也更容易理解。建議使用者以前頁的三種 (month, day) 數值代入 and 運算，然後以 and 運算結果決定 if 句子的執行流程，並比對兩個程式片段的功能是否完全相同（答案是「完全相同」）。

邏輯運算子 or

第二個邏輯運算子是「或者」，它也是二元運算子，表 2-4-2 列出它的四種運算組合及結果。我們同樣可以歸納出一條簡單的法則，只要有一個運算元為真，or 運算的結果就為真 – 這也符合我們對「或者」這個詞的理解。

表 2-4-2: or 運算子的運算結果

a	b	a or b
False	False	False
False	True	True
True	False	True
True	True	True

下列右方程式片段展示了如何運用 or 運算來簡化左方程式的結構。

```
01 if age <= 12:
02     price *= 0.9
03 elif age >= 65:
04     price *= 0.9
05 else:
06     price -= 50
```

```
01 if age <= 12 or age >= 65:
02     price *= 0.9
03 else:
04     price -= 50
```

邏輯運算子 not

最後一個邏輯運算子是「非」，它是個一元運算子。簡單來說，它會把真變假，把假變真，如同表 2-4-3 所示。

表 2-4-3: not 運算子的運算結果

a	not a
False	True
True	False

邏輯運算子的運算優先次序低於關係運算；其中「not 運算」高於「or 運算」高於「and 運算」。下列程式片段中，第 01 行的條件式會先運算兩個相等關係運算 (==)，然後運算 not，最後再運算 and。因此，這個程式片段無法在正確的日期列印出「中秋節快樂!」。如同我們能以小括號改變算術運算的順序，例如 (2+3)*6 中先運算加法 (+) 才運算乘法 (*)，我們也能在這個程式中加上一對小括號來改正它，這裡就留給讀者去思考囉！

錯誤的範例

```
01 if not month == 8 and day == 15:
02     print('平常日')
03 else:
04     print('中秋節快樂!')
```

隨堂練習

2.4.1 請使用 Python 執行下列的運算，執行前先思考想想各邏輯運算式結果式 Ture 或是 False? 如果結果跟你想的不一樣，可以嘗試將邏輯判斷子左右兩邊的運算式分別執行看結果是甚麼。

(1) 9>0

(2) '9'>'0'

(3) '2'+'9' < '8'

(4) 2+9<8

(5) 'a'<='A'

(6) 10<5*3 and 5<3+1

(7) 10<5*3 or 5<3+1

(8) 10%3==1

(9) 10%3!=1

(10) 10 < '10'

(參考答案)

(1). True (2). True (3). True (4). False (5). False

(6). False (7). True (8). True (9). False (10). 出現錯誤

Chapter 3

成績資料庫

/ 蔣宗哲、江政杰

相對於人腦，電腦的其中一個優勢是能持續且快速地重覆運算。本章將介紹程式語言的第二個重要句型：重覆（迴圈）句型 (for)，同時也介紹重覆句型最常搭配的資料結構 -- 列表 (list)。本章分為七個小節，內容較多，但是以成績資料庫作為應用主軸，循序漸進，初學的讀者若感到些許困難，建議放慢腳步，重覆練習。學習完前三章後，讀者將具備基礎程式能力，已可處理大量、結構化的數值型態資料，進行簡易的統計分析（如求平均、極值、標準差），並能對資料排序與查詢。

3-1 列表 (list) 與迴圈

資料型態與操作

前面兩章的程式裡，我們運用到三種資料型態，分別為整數型態（int）、小數或稱浮點數型態（float）以及字串（str）型態。在程式裡，我們可以使用 type() 函式得知一個資料或變數的型態，如程式 E3-1-1 所示。

E3-1-1.py：type() 函式

```
01 print(type(3))
02 print(type(3.14))
03 print(type('NTNU'))
04 a = 3
05 print(type(a))
06 a = 3.14
07 print(type(a))
```

執行結果：

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'int'>
<class 'float'>
```

type() 函式可以取得資料的型態。

不同資料型態在電腦內部的儲存方式不同，佔用的記憶體大小也可能不同，但對一般程式員來說，最顯而易見的差異是它們支援的操作不同，即使同一種操作，其結果也不同。例如整數可以作除法，但字串就沒有除法；整數和字串都可以作乘法，但整數乘法 3*2 和字串乘法 '3'*2 就分別得到整數 6 和字串 '33' 這樣兩個不同的結果。

Python 的字串型態除了支援兩個字串相加以及字串和整數相乘，還支援一個稱作 split() 的操作。在程式中，我們會在字串的後面以一個句點再接 split() 來表示要對字串作此操作，如程式 E3-1-2 所示。

E3-1-2.py：字串的 **split()** 操作

```
01  s = 'NTNU is great!'
02  words = s.split()
03  print(type(words))
04  print(words)
```

執行結果：

```
<class 'list'>
['NTNU', 'is', 'great!']
```

這個程式中，第 01 行以變數 s 代表 'NTNU is great!' 這個字串。第 02 行以變數 words 代表字串 s 執行 split() 的結果。第 03 行列印出 words 的型態，而第 04 行列印出 words 的內容。從執行結果我們可以推論出：

1. 對字串進行 split() 操作會得到一種 list 型態的資料。
2. 對字串進行 split() 操作，會以空白字元分隔出多個字串，並以 list 型態儲存之。

⚠ 不是任何資料型態都有 split() 操作。

從實驗了解字串的 split() 操作

學習程式的過程中，我們經常可以透過實驗來幫助我們理解程式，例如我們可以用多個不同內容的字串來測驗 split() 的操作結果。在程式 E3-1-3 中，我們可以測試：

1. 字串中含有多個空白字元的 split() 結果
2. 字串中沒有任何空白字元的 split() 結果
3. 字串中含有其它看似分隔字元的 split() 結果

E3-1-3.py：字串的 **split()** 實驗

```
01 print('NTNU      is great!'.split())
02 print('abcde'.split())
03 print('123, 456, 789'.split())
04 print('123,456,789'.split())
```

執行結果：

```
['NTNU', 'is', 'great!']
['abcde']
['123,', '456,', '789']
['123,456,789']
```

從結果我們可以發現，split() 分隔出來的字串中不會含有空白字元。而如果原字串中沒有空白字元，split() 還是可以運作，只是會得到和原字串相同的結果。

取得列表中的資料

從字串中經由 split() 分離出一或多個字串，成為一個字串的列表後，我們該如何取得指定的資料呢？列表型態支援下標運算子 []，以從 0 開始的整數取用列表中的個別資料。在程式 E3-1-4 中，第 03 行直接列印整個字串列表，會以一對中括號將內容包起來。第 04 行是以下標運算子取用列表的第「一」個元素（索引值是 0，或稱為 0 號元素），然後列印出來，因此畫面列印出 2018 這個字串。第 05 行將字串列表中的第一和第二個元素相加（還記得字串相加嗎？），因此會得到 '2018'+'9'，也就是 '20189' 這個字串。第 06 行和第 05 行的差異是第 06 行把字串轉換成整數型態後才相加。

E3-1-4.py：列表的下標運算子

```
01  s = '2018 9 26'
02  num = s.split()
03  print(num)
04  print(num[0])
05  print(num[0]+num[1])
06  print(int(num[0])+int(num[1]))
```

執行結果：

```
['2018', '9', '26']
2018
20189
2027
```

從實驗了解列表的索引值

剛剛說到實驗是寫程式過程中很有趣的一部份，我們再來實驗看看列表的索引值應該如何使用。程式 E3-1-5 裡我們嘗試了幾件事：

1. 第 05 行我們使用了過大的索引值：第 03 行的結果讓我們知道 num 列表中有 3 個元素，因此列表中的三個元素應該分別是 num[0]、num[1] 和 num[2]。也就是說，num[3] 是不存在的，這也使得程式發出 IndexError 這個錯誤。
2. 第 06 行和第 07 行我們使用字串和小數作為索引值：結果發現這兩種型態不能作為列表的索引值，程式發出 TypeError 這個錯誤。
3. 第 08 和第 09 行我們使用負值作為索引值：結果發現列表的索引值可以是負的，它的結果是從列表的尾端開始取用，num[-1] 代表最後一個元素，num[-3] 代表倒數第三個元素。第 10 行和第 05 行類似，都是因為索引值超出元素個數而發生了錯誤。

E3-1-5.py：列表的下標運算子實驗

```
01  s = '2018 9 26'
02  num = s.split()
03  print(num)
04  print(num[0])
05  #print(num[3])   #IndexError
06  #print(num['first']) #TypeError
07  #print(num[1.5]) #TypeError
08  print(num[-1])
09  print(num[-3])
10  #print(num[-4])   #IndexError
```

執行結果：

```
['2018', '9', '26']
2018
26
2018
```

小結

本節中，我們學會了下列知識：

1. 字串有一種操作叫 `split()`，`split()` 會將原字串以空白分割成很多個新字串，並把這些新字串以一個列表來儲存。
2. 列表是一個可儲存多個資料的容器。（本節中我們只存放字串，事實上列表也可以存放整數和小數，每個資料可以有不同的型態。）
3. 列表可用下標運算子搭配索引值來取出其中一個資料，索引值必須是整數值，而且數值範圍必須符合列表中的資料個數。

3-2 逐一取用列表元素：for 句型

在學習 3-2 節以前，如果我們的程式要從使用者端讀入三個整數，需要呼叫三次 input()，每次讀入一個整數。現在，我們學會了字串的 split() 以及列表的相關知識，我們可以讓使用者一次輸入三個整數了！（別忘了輸入時必須以空白隔開三個整數。）

```
01  num = input('請輸入三個整數 > ').split()
02  ans = int(num[0])+int(num[1])+int(num[2])
03  print(ans)
```

上面這三行程式可以讓使用者一次輸入三個整數，然後計算三數總和，最後列印出總和。雖然比原本作三次 input() 簡潔，但 .....，如果是四個數、五個數、甚至五十個數，該怎麼辦呢？如果事先不知道有幾個數，又該怎麼辦呢？

for 句型

Python 語言有一個 for … in … 句型，可以依序取出列表中所有元素。我們先看個程式範例。在程式 E3-2-1 中，第 01 行將使用者輸入的字串 split() 成一個字串列表 num。第 03 行的 for 句型會逐一取出 num 中的 num[0], num[1], ... 直到列表結尾，每次取出一個元素，會以變數 e 代表之，然後執行第 04 行的程式碼。注意到 04 行之於 03 行的內縮格式嗎？這和前一章我們學習 if 句型的格式相同，for 句型下內縮的程式片段，會對每個列表元素執行一次。

E3-2-1.py：以 **for** 句型計算總和

```
01  num = input(' 請輸入任意個整數 > ').split()
02  ans = 0
03  for e in num:
04      ans = ans + int(e)
05  print(ans)
```

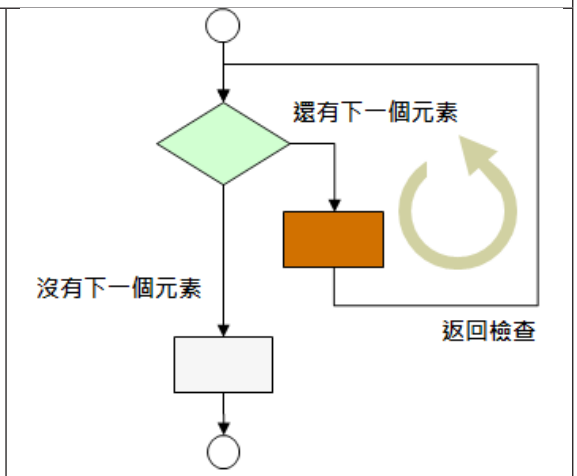
執行結果 1：	執行結果 2：
請輸入任意個整數 > 1 2 9 12	請輸入任意個整數 > 1 3 4 5 13

當使用者輸入字串 '1 2 9'，split() 會產生列表 ['1', '2', '9']，以 num 代表之。接著第 03 行會讓 e 代表 num[0]，進入第 04 行，計算 ans，得到結果整數 1。依據 for 句型的控制流程，由於 num 列表還沒有結束，所以第 03 行會再讓 e 代表 num[1]，再進入第 04 行，計算 ans 得到結果 3。同理，再讓 e 代表 num[2]，計算 ans 得到 3+9 = 12。最後，第 05 行列印出 12。

下面我們以流程圖對照程式碼來幫助理解。我們可以把 for … in … 的這一行想像成之前學過的 if 句型的條件式，如果列表裡還有元素，就會對這個元素執行 for 句型下面內縮的程式片段。和 if 句型不同的是，if 句型作完內縮片段之後就會繼續往下執行，但 for 句型在作完內縮片段後，還會再去檢查列表內還有沒有元素。也就是這個「還會再去檢查」，使得 for 句型的執行流程形成一個反覆操作的過程，在程式員的術語中，我們稱這裡有一個「迴圈」。

語法說明：**for**

```
for 變數 in 列表：
  動作 1
  動作 2
非重覆動作
```



到目前為止，我們已經學會了程式語言中最重要的兩種句型 – 選擇句型 (if) 和重覆句型 (for)。活用這兩個句型，就會作到非常多的事情。舉例來說，程式 E3-2-2 可以幫我們計算多個數字中有幾個大於等於 60，實務上的應用，可以幫老師計算有幾個學生的考試分數及格。

E3-2-2.py：以 **for** 句型和 **if** 句型計算及格人數

```
01  num = input(' 請輸入數值 > ').split()
02  ans = 0
03  for e in num:
04      if int(e)>=60:
05          ans += 1
06  print(ans)
```

執行結果：	執行結果：
請輸入數值 > 59 60 61 9 100 3	請輸入數值 > 90 80 70 60 4

程式 E3-2-3 可以幫我們計算多個數值中的偶數值的和。

E3-2-3.py：以 **for** 句型和 **if** 句型計算偶數值的和

```
01  num = input(' 請輸入數值 > ').split()
02  ans = 0
03  for e in num:
04      e = int(e)
05      if e%2==0:
06          ans += e
07  print(ans)
```

執行結果：	執行結果：
請輸入數值 > 1 2 3 4 8 14	請輸入數值 > 1 3 5 7 9 0

隨堂練習

3.2.1 運用所學，修改以下程式，使其具備以下功能：

1. 使用者在一行輸入任意個整數分數；
2. 輸出分數的個數，總和以及平均；
3. 只記錄介於 0-100 分的分數；
4. 如果沒有任何合法分數，個數、總和與平均皆為 0。

```
01  num = input(' 請輸入任意個整數 > ').split()
02  ans = 0
03  count = 0
04  for e in num:
05      ans = ans + int(e)
06      count = count + 1
07  print(' 個數 =', count, ' ，總和 = ',ans, ' ，平均 =', ans/count)
```

♦ 思考程式流程

目標	程式
在一行輸入任意個整數分數	使用 `input()` 與 `split()` 得到資料列表
輸出分數的個數，總和以及平均	使用 `for` 句型檢視列表中的資料 使用 `int()` 將字串轉換成整數值 使用兩個變數累計資料個數和總和
只記錄介於 0-100 分的分數	使用 `if` 句型判斷數值是否符合
輸出分數的個數，總和以及平均	使用 `print()` 列印結果 使用 `if` 句型判斷合法分數個數是否大於零

小結

學習完本節後，你已經能撰寫程式處理許多工作，快給自己拍拍手鼓勵一下！

能力	具體作法
向使用者取得大量數據	`input()`
以易處理的方式儲存數據	字串 `split()` 與列表 `list`
以簡潔的程式檢視數據	`for` 句型
對數據進行檢核與分類	`if` 句型，關係運算，邏輯運算
（根據分類）進行運算	算術運算
顯示結果	`print()`

3-3 自建列表與內建函式

在前一小節的隨堂練習中（什麼？你還沒寫完，那還不快去寫！），我們整合運用迴圈控制和選擇控制來計算總和與平均。在實務上，計算平均與極值、搜尋與排序，通常會透過內建函式來完成。截至目前，我們已學過 print() 和 input() 兩個函式，你應該會預期，Python 語言有一些現成的函式，例如下列程式中，我們試圖呼叫 sum() 函式。當程式執行到第 02 行時，程式發出一個 TypeError 錯誤訊息，告訴我們整數和字串無法相加。因此，我們知道 Python 語言中確實存在 sum() 函式，sum() 函式期待的是一個整數列表，而 num 是一個字串列表，因此 sum() 無法處理。

```
01  num = input('請輸入任意個整數 > ').split()
02  s = sum(num)
```

再看另一個程式，我們試圖呼叫 min() 函式。這個程式可以順利執行。我們輸入兩組測試資料，發現第一組測試資料結果符合預期，但第二組結果卻告訴我們 78 比 9 還要小，這是怎麼回事呢？

```
01  num = input('請輸入任意個整數 > ').split()
02  minimum = min(num)
03  print(minimum)
```

執行結果：	執行結果：
請輸入任意個整數 > 9 6 8 6	請輸入任意個整數 > 78 9 78

在前一個程式裡，我們意識到 num 是一個字串列表（所以 sum() 無法順利運算）。我們預期 min() 函式的功用是找出列表中的最小元素，此時因為列表中的元素是字串，所以這時候 min() 找出的是最小的字串。

這裡讓我們暫時跳開列表的處理，談一下字元與字串的大小關係。先看看比較單一字元常用的規則（事實上是基於 ASCII 碼來比較的）：

1. 數字字元由小到大為 '0', '1', '2', …, '9'。
2. 大寫英文字母字元由小到大為 'A', 'B', 'C', …, 'Z'。
3. 小寫英文字母字元由小到大為 'a', 'b', 'c', …, 'z'。
4. 任一數字字元小於任一大寫英文字母字元，任一大寫英文字母字元小於任一小寫英文字母字元。

看完上面的說明，回頭看一下隨堂練習 2.4.1 的文字比較，是不是就更清楚了？上面是單一字元的比較原則，如果是兩個字串在比較時，會依序從各自的第一個字元開始比較，如果兩個字元相異，則以這兩個字元比較大小（如 'Ant' 小於 'Bat'）；如果兩個字元相同，則比下一個字元（如 'Candy' 小於 'Cat'）。若字元兩兩相同，直到某一個字串已經結束了，則視另一個字串的狀態決定關係：如果另一個字串還沒結束，則該字串比較大（如 'Dad' 小於 'Daddy'）；如果另一個字串也結束了，則兩個字串相等。

好了，現在我們了解字串的大小關係，讓我們再看一次前頁的第二個程式：為什麼 min() 說 78 小於 9 呢？其實，因為我們傳給 min() 函式的是字串列表 ['78', '9']，所以 min() 說的是 '78' 小於 '9'，這樣一切就合理了。

學會字串的大小關係是很不錯啦，可是這沒有解決我們的問題啊 – 要怎麼樣運用內建函式 sum() 與 min() 來計算一堆數值的總和與最小值呢？答案很明白，我們需要將字串列表轉換成整數列表。喔！我知道你在想什麼。很遺憾，下面程式的第 02 行會造成 TypeError 錯誤，因為 Python 沒辦法把一個字串列表轉換成「一個」整數值。我們必須將字串列表中的字串一個一個轉成整數才行。

```
01 str = input('請輸入任意個整數 > ').split()
02 num = int(str) #TypeError
```

從字串列表建立整數列表，方法 1：append()

列表型態有一種操作叫 append()，它可以將一個元素加入到列表的尾端。程式 E3-3-1 的第 03 行先建立一個空的列表 num。接著第 04 行運用 for 句型逐一取出 str 字串列表的每一個字串，令 e 代表該字串，對 e 執行第 05 行。第 05 行中，首先將字串 e 轉換成整數，然後再以列表的 append() 操作加入列表。當第 04-05 行的整個 for 句子執行完畢，第 06 行列印出 num 的結果，我們可以看到 num 現在是一個整數列表了（對比第 02 行的輸出）。最令人興奮的是，第 07 行列印了 sum() 計算 num 列表中數值總和，這個總和就是我們希望計算的整數值 80+90+70 的正確結果。第 08 行也藉助 min() 函式求出三個整數值中的最小值了。

E3-3-1.py：列表的 **append()** 操作

```
01  s = input('請輸入任意個整數 > ').split()
02  print(s)
03  num = []
04  for e in s:
05      num.append(int(e))
06  print(num)
07  print(sum(num))
08  print(min(num))
```

執行結果：

```
請輸入任意個整數 > 80 90 70
['80', '90', '70']
[80, 90, 70]
240
70
```

從字串列表建立整數列表，方法 2：+ 運算子

除了用 append() 將單一個元素加到列表尾端，列表型態也支援兩個列表的 + 法運算，此運算會把兩個列表的元素合併起來，並產生一個新的列表（原本的左列表並不會受到影響）。

程式 E3-3-2 的第 01-02 行建立了列表 a 和 b。第 03 行令 c 代表 a+b 的結果：將 [1, 2, 3] 和 [4] 合併，形成新列表 [1, 2, 3, 4]。注意程式第 04 行列印三個列表 a、b 和 c 的結果：a 仍為 [1, 2, 3]，沒有改變。如果是使用前一種方法，以 a.append(4) 來添加元素 4 的話，a 就會有所改變，變成 [1, 2, 3, 4] 了。

E3-3-2.py：列表的 **+** 運算子

```
01 a = [1, 2, 3]
02 b = [4]
03 c = a + b
04 print(a, b, c)
```

執行結果：

```
[1, 2, 3] [4] [1, 2, 3, 4]
```

如果以 + 運算子來改寫程式 E3-3-1，第 04-05 行會寫成：

```
04 for e in s:
05     num = num + [int(e)]
```

append() 和 + 運算子有類似的功用，但還是有些差異。我們以下面兩個程式片段來比較它們。左邊的程式中，當我們執行第 03 行時，a.append(b) 是把 b 當成一個元素來加入列表 a，因此我們可以看到第 04 行列印出來的 a 當中有四個元素，最後一個是列表 [4, 5]。相比之外，右邊的程式在執行第 03 行時，是取出兩個列表內的元素（列表 a 有 3 個整數元素，列表 b 有 2 個整數元素）再組合成一個新的列表。所以第 04 行列印的 a 會有五個元素。

append()

```
01 a = [1, 2, 3]
02 b = [4, 5]
03 a.append(b)
04 print(a)
05 print(b)
```

執行結果：

```
[1, 2, 3, [4, 5]]
[4, 5]
```

+ 運算子

```
01 a = [1, 2, 3]
02 b = [4, 5]
03 a = a + b
04 print(a)
05 print(b)
```

執行結果：

```
[1, 2, 3, 4, 5]
[4, 5]
```

從字串列表建立整數列表，方法 3：列表推導 (list comprehension)

列表推導是 Python 語言建立列表的的一種具 Python 風格的語法，乍看之下有點複雜，但是習慣之後就會喜歡上它的簡潔。再者，許多 Python 程式都會用這種方法來建立列表，所以希望讀者還是能儘量了解。

列表推導式可將一個列表中的所有元素逐一取出並作運算，然後以這些運算結果建立另一新列表。程式 E3-3-3 的第 01 行建立列表 a，內含 1、12 和 43 三個整數元素。第 02 行是列表推導式，逐一取出 1、12 和 43，並以這三個數加 1 後的結果構成新的列表 [2, 13, 44]。程式 E3-3-4 也很類似，不過是以列表 a 中的三個整數的平方來構成新的列表 b，所以 b 的內容是 [1^2, 2^2, 3^2]。

語法說明：
[運算結果 for 元素 in 列表]

E3-3-3.py: 列表推導

```
01  a = [1, 12, 43]
02  b = [e+1 for e in a]
03  print(a)
04  print(b)
```

執行結果：

```
[1, 12, 43]
[2, 13, 44]
```

E3-3-4.py: 列表推導

```
01  a = [1, 2, 3]
02  b = [e**2 for e in a]
03  print(a)
04  print(b)
```

執行結果：

```
[1, 2, 3]
[1, 4, 9]
```

程式 E3-3-1 的第 03 行之後以列表推導式可改寫成程式 E3-3-5 所示。

E3-3-5.py: 列表推導

```
03  num = [int(e) for e in s]
04  print(num)
05  print(sum(num))
06  print(min(num))
```

列表推導式中也可以加入 if 條件式來篩選資料，如下面兩個程式碼所示。兩個程式目的相同，都是以列表 a 中的奇數建立新的列表 b。左邊的程式是傳統的寫法，右邊的程式則是列表推導的寫法。

```
01 a = [1, 2, 4, 7, 9]
02 b = []
03 for e in a:
04     if e%2==1:
05         b.append(e)
```

```
01 a = [1, 2, 4, 7, 9]
02 b = [e for e in a if e%2==1]
```

呼叫內建函式

這一節講了這麼多，無非是希望將原本從使用者手上取得的文字輸入轉換成數值的列表，才好運用 Python 內建的函式。表 3-3-1 條列了常用數值處理函式及使用範例。

表 3-3-1：常用的內建數值處理函式

函式名稱	簡述	函式用法
abs()	絕對值	abs(-3) 回傳 3； abs(-3.14) 回傳 3.14。
round()	四捨五入	round(3.4) 回傳 3； round(3.5) 回傳 4； round(3.44, 1) 回傳 3.4； round(3.45, 1) 回傳 3.5。
len()	資料量	len([]) 回傳 0； len([90, 3, 56]) 回傳 3。
min()	最小值	min([9, 3, 2, 1, 5]) 回傳 1； min(['bac', 'ab']) 回傳 'ab'。
max()	最大值	max([9, 3, 2, 1, 5]) 回傳 9； max(['bac', 'ab']) 回傳 'bac'。
sum()	總和	sum([]) 回傳 0； sum([5, 1, 3]) 回傳 9。
sorted()	排序	a = [8, 1, 3] b = sorted(a) print(a, b) 列印 [8, 1, 3],[1, 3, 8]。

隨堂練習

3.3.1 撰寫一個程式，讀入全班成績，過濾非法分數和離群分數，然後輸出最低分、中位數和最高分。

- 過濾條件一：只留下介於 0 到 100 分的分數。
- 過濾條件二：只留下和平均值差距三個標準差之內的分數。
- 注意：平均值和標準差之計算，必須先過濾掉非法分數。
- 平均值公式 (μ)：總和 / 個數
- 標準差公式 (SD)：$\mathrm{SD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}\mathrm{SD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$
- 思路：
  (1) 先寫下整個資料處理步驟與流程（大流程寫好後，再慢慢拆解）。
  (2) 再寫下每步驟需要使用到的程式關鍵字。
  (3) 最後再開始寫程式。
- 測試程式
  (1) 測試的時間點依你對程式的掌握度而定。
  (2) 你覺得程式有點長了，就可以測試該段程式碼的正確性。

3-4 列表切片

現在我們已經很熟悉以索引值和下標運算子取用列表中的單一元素了。如果我們想要拿出列表中的某「一段」元素，同樣可以透過下標運算子來完成所謂的「切片」(slice)。

程式 E3-4-1 的第 02 行示範了切片的基本語法：在下標運算子中以一個冒號隔開起始索引值及結束索引值。要特別注意，切片取出來的不會包括結束索引值的那個元素，只會取到前一個元素。因此第 02 行的 `data[1:3]` 只會取到 data[1] 和 data[2] 這兩個元素（不包含 data[3]）。第 03 行令 `c` 代表取出的 `data[0:2]`。第 04-05 行的目的是要表示對 `c` 列表作修改並不會修改到原本的 `data` 列表。

E3-4-1.py: 列表切片基本型式

```
01 data = [1, 2, 4, 8]
02 print(data[1:3])
03 c = data[0:2]
04 c[0] = 30
05 print(data, c)
```

執行結果：

```
[2, 4]
[1, 2, 4, 8] [30, 2]
```

切片的開始索引值和結束索引值都可以省略。若省略開始索引值，代表從列表的第一個元素開始，如程式 E3-4-2 第 02 行從頭開始取用三個元素。若省略結束索引值，代表取到列表的最後一個元素（含最後一個元素）為止，如第 03 行，從第三個元素開始取到最後。如果兩個都省略，則表示複製一份列表內容，如第 05 行。同樣要注意，第 05 行創造出來的 `c` 列表和原本 `data` 列表如今是兩個不同的列表了。最後，兩個索引值都可以是負值，如第 04 行從倒數第三個元素開始取用。

E3-4-2.py：列表切片省略起始或 / 和結束索引值

```
01  data = [1, 2, 4, 8]
02  print(data[:3])
03  print(data[2:])
04  print(data[-3:])
05  c = data[:]
06  c[0] = 9
07  print(data, c)
```

執行結果：

```
[1, 2, 4]
[4, 8]
[2, 4, 8]
[1, 2, 4, 8] [9, 2, 4, 8]
```

切片操作還有一個變化型式，就是指定取用的間隔。程式 E3-4-3a 中示範正值間隔，第 02 行從索引值 0 開始取出，每 2 個取一次（也就是取到 data[0]、data[2]、data[4] 和 data[6]。）第 03 行和第 02 行的差異在於從索引值 1 開始取出，所以取到的是 data[1]、data[3]、data[5] 和 data[7]。第 04 行和第 05 行指定了結束索引值，請記得只取到結束索引值的前一個元素。

程式 E3-4-3b 示範了負值間隔。由於是負值，所以取用的索引值會變小，也就是從尾向頭取用的意思。程式第 02 行是從最後 data[7] 往前逐一取用到 data[0]。這一行是倒轉列表的標準寫法。第 03 行從尾向頭每隔 2 個取一次。第 04 行有指定開始索引值和結束索引值，由索引值 2 無法遞減到索引值 6，所以得到一個空列表。第 05 行從索引值 6 遞減到索引值 2，因為不包含索引值 2 的元素，所以會逐一取出 data[6]、data[5]、data[4] 和 data[3]。第 06 行的從索引值 -1 遞減至 -5，每次減 2，取出 data[-1] 和 data[-3]（不包含 data[-5]）。

E3-4-3a.py：列表切片正值間隔

```
01 data = [11, 22, 33, 44, 55, 66, 77, 88]
02 print(data[::2])
03 print(data[1::2])
04 print(data[1:6:2])
05 print(data[1:7:2])
```

執行結果：

```
[11, 33, 55, 77]
[22, 44, 66, 88]
[22, 44, 66]
[22, 44, 66]
```

E3-4-3b.py：列表切片負值間隔

```
02 print(data[::-1])
03 print(data[::-2])
04 print(data[2:6:-1])
05 print(data[6:2:-1])
06 print(data[-1:-5:-2])
```

執行結果：

```
[88, 77, 66, 55, 44, 33, 22, 11]
[88, 66, 44, 22]
[]
[77, 66, 55, 44]
[88, 66]
```

3-5 range()、索引值走訪與搜尋

雖然我們已經學會用「for ... in 列表」句型逐一取出列表元素，也學會搭配切片取出列表中的一部份元素，不過有些時候我們會想要透過索引值來逐一取用列表元素。有了索引值，我們知道正在取用第幾個元素，也可以藉由目前元素的索引值來計算其它元素的索引值。例如我們想要比較兩個鄰近元素是否相等，若已知目前元素的索引值為 i，則前一個元素的索引值是 i-1，我們可以拿出 [i] 和 [i-1] 來比較。

以索引值走訪：range( )

呼叫 range(n) 可以幫我們產生 0, 1, 2, ..., (n-1) 的整數值，程式 E3-5-1 示範了使用 for 句型加上 range() 來逐一取出列表元素的標準寫法。第 01 行以列表推導式將使用者輸入建立成一個整數列表 data。第 02 行先以 len() 函式求得 data 的元素個數，然後傳入 range() 產生 0, 1, 2, ... 的整數值。假設使用者輸入五個整數值，則第二行 len(data) 會是 5，for 句型會讓 i 依序代表 0, 1, 2, 3, 和 4。這些值會被當作列表索引值，在第 03 行列印出 data 中的每一個元素。

E3-5-1.py: 以 range() 產生索引值並走訪列表

```
01  data = [int(e) for e in input().split()]
02  for i in range(len(data)):
03      print(data[i])
```

執行結果：

```
10 20 30 40 55
10
20
30
40
55
```

range() 函式的小括號內可以傳入一個、兩個或三個整數值，其意義和我們在前一節切片操作中的三個數值相同 – 分別代表起始值、終止值「的下一個值」、以及間隔值。程式 E3-5-2 示範了如何以 range() 函式產生連續有規律變化的整數值。print() 函式中，end=' ' 的意思是要求 print() 列印 i 後不要以預設的換行字元作結束，改以空白字元作結束，如此一來可以把所有的 i 值印在同一行。

E3-5-2.py: range() 的呼叫範例

```
01  # 列印 0, 1, 2, 3, 4
02  for i in range(5):
03      print(i, end=' ')
04  print()
05
06  # 列印 3, 4, 5, 6
07  for i in range(3, 7):
08      print(i, end=' ')
09  print()
10
11  # 列印 10, 12, ..., 20
12  for i in range(10, 21, 2):
13      print(i, end=' ')
14  print()
15
16  # 列印 -10, -9, ..., -1
17  for i in range(-10, 0, 1):
18      print(i, end=' ')
19  print()
```

執行結果：

```
0 1 2 3 4
3 4 5 6
10 12 14 16 18 20
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1
```

列表的搜尋

在列表中搜尋指定的資料是很常見的工作，Python 很貼心地提供了 in 運算子，讓我們可以輕易地判斷資料是否存在於列表中。如果想判斷不存在，加個 not 也可輕鬆搞定。

```
01 scores = [int(e) for e in input().split()]
02 if 100 in scores:
03     print('有人滿分')
```

```
01 scores = [int(e) for e in input().split()]
02 if 100 not in scores:
03     print('無人滿分')
```

執行結果 1：

```
90 100 80 70 30
有人滿分
```

執行結果 2：

```
90 60 80 70 30
無人滿分
```

如果想要計算有幾個符合查詢的值，Python 的列表型態提供了 count() 函式，可以如下面程式來使用。

```
01 scores = [int(e) for e in input().split()]
02 print('有', scores.count(100), '人滿分')
```

執行結果：

```
100 80 90 70 40 50 100 80
有 2 人滿分
```

Python 的列表型態還提供了 index() 函式，可以用來查詢某個資料在列表中的索引值。程式 E3-5-3 為一個活動報名的應用，使用者先依報名順序輸入報名者的姓名；輸入後便可查詢某人是否有報名，若有報名，則輸出他 / 她的報名順位，若無報名，則輸出查無報名資料。要注意，使用 index() 函式時，如果查詢的資料不在列表中，程式會發出錯誤訊息。因此，我們通常會先以 in 運算子確認資料存在，才呼叫 index() 函式來取得該資料的索引值。

E3-5-3.py:in 運算子與列表 index() 函式的結合應用

```
01 names = input().split()
02 query = input()
03 if query in names:
04     print(query, '的報名序號為', names.index(query)+1)
05 else:
06     print('查無報名資料')
```

執行結果 1：	執行結果 2：
請輸入報名資料 > 張三 李四 王五 趙六 請輸入欲查詢姓名 > 王五 王五 的報名序號為 3	請輸入報名資料 > 張三 李四 請輸入欲查詢姓名 > 王五 查無報名資料

range() 函式還有一個用途是以指定次數重覆執行程式片段。例如下面的程式利用第 01 行的 for 搭配 range(3) 來執行第 02 行的程式碼三次。簡單來說，range() 傳入的數值就是重覆執行的次數。此處 range(3) 所產生的 0, 1, 2 數值並不會被拿來運算，所以慣例上我們會用 _ 作為變數名稱，示意閱讀程式的人此處的 range() 純粹是為了控制迴圈次數而已。

```
01 for _ in range(3):
02     print('很重要所以說三遍')
```

執行結果：
很重要所以說三遍 很重要所以說三遍 很重要所以說三遍

3-6 分解賦值

列表有一個很有趣的分解賦值操作，讓我們可以很方便地用具名變數指稱其中的每個元素。下列程式碼的第 01 行將列表 [10, 20, 30] 的三個資料分別以 x、y 和 z 來指稱。要注意，等號左邊的變數個數和右邊的列表資料個數必須相等，否則程式就會產生 ValueError: not enough values to unpack 或 ValueError: too many values to unpack 的錯誤訊息。（讀者可以自行嘗試刪除第 01 行的 ,z 或者 ,30 來觀察錯誤訊息。）

```
01 x, y, z = [10, 20, 30]
02 print(x, y, z)
```

執行結果：

```
10 20 30
```

分解賦值常用來將輸入的資料賦予個別名稱。下面的兩段程式碼執行後有相同的結果，但第二段程式碼中我們明確地指定了三個輸入資料的變數名稱，可以提升程式的可讀性。

```
01 data = [
02 int(e) for e in input('輸入身高、體重和年齡>').split()]
03 print('你的年齡是', data[2])
```

```
01 height, width, age = [
02 int(e) for e in input('輸入身高、體重和年齡>').split()]
03 print('你的年齡是', age)
```

執行結果：

```
輸入身高、體重和年齡>180 80 20
你的年齡是 20
```

在 Python 程式中，若要交換兩個變數，標準寫法如程式 E3-6-1 第 04 行所示。事實上，這一行便用到分解賦值的操作，只不過這時候等號右邊的 y, x 會被視為一個元組 (tuple)。元組和列表相似，不過元組是不可修改的。此處我們就不繼續討論元組型態了。

E3-6-1.py: 交換兩個變數

```
01  x = 10
02  y = 20
03  print(x, y)
04  x, y = y, x
05  print(x, y)
```

執行結果：

```
10 20
20 10
```

程式 E3-6-2 結合運用了 Python 內建的 `min()` 和 `max()` 函式、列表的 `index()` 函式，以及本節所介紹的分解賦值操作，其功能為交換列表中最小值和最大值。讀者應可看出執行結果的兩行輸出中，最小值 10 和最大值 60 已互換位置。

E3-6-2.py: 交換列表中的最小與最大值

```
01  data = [40, 20, 50, 10, 60, 30]
02  print(data)
03  i = data.index(min(data))
04  j = data.index(max(data))
05  data[i], data[j] = data[j], data[i]
06  print(data)
```

執行結果：

```
[40, 20, 50, 10, 60, 30]
[40, 20, 50, 60, 10, 30]
```

3-7 多維度列表

假如今天我們的程式要儲存一個 40 人班級的期中考分數，相信讀者已經能很直覺地以列表來儲存，並能從列表中以索引值取出任一學生的分數。（當然，我們一定都記得 1 號學生的分數是存放在索引值 0，而非索引值 1，的位置。）

那麼，如果這學期不只一次考試呢？我們該如何存放，以便未來仍然可以很方便地取出第 i 號學生的第 j 次考試分數呢？由於我們有兩個索引值（學生號碼及考試次別），我們需要讓列表也同樣擁有兩個索引值 – 這就要用到本節所談的多維度列表了。

程式 E3-7-1 第 01 行建立了二維列表 scores。事實上，建立二維列表的方式與建立一維列表無異，同樣以是一對中括號把資料包起來，只不過一維列表中的資料是單一數值，而此處二維列表中的資料也是列表，讀者可以看到最外圍的中括號裡面又有三對中括號。第 02 行列印出 3 和 2，代表 scores 這個二維列表中有三個資料，而 scores 的第一個資料 scores[0] 中含有 2 個資料（即 100 和 80）。第 03 行展示了如何以兩個索引值取用二維列表中的資料，我們可以看到 scores[0][0] 表示先取出 scores 中的第一個一維列表，再取出該一維列表的第一個資料，也就是 100；依此類推，scores[1][1] 即為第二個一維列表的第二個資料 90。第 04 行展示我們可以將二維列表中取出的一維列表傳入之前介紹過的函式來處理，此處 sum() 回傳 60+50 的結果。

E3-7-1.py: 二維列表的建立與取用

```
01 scores = [[100, 80], [70, 90], [60, 50]]
02 print(len(scores), len(scores[0]))
03 print(scores[0][0], scores[1][1])
04 print(sum(scores[2]))
```

執行結果：

```
3 2
100 90
110
```

程式 E3-7-2 示範了從使用者輸入來建立二維列表的方法。首先，第 01 行建立一個空的列表。第 02 行表示第 03-04 行要重覆 3 次。第 03 行將使用者輸入以 input() 函式讀入後拆解為字串列表，然後再以列表推導式建立成整數列表，最後指定給 stu 變數。第 04 行則將 stu 這個一維整數列表附加到 scores 尾部。由於 stu 本身是個一維列表，scores 就會變成二維列表了。注意這裡使用者每次輸入的分數個數即使不同，scores 仍然能夠儲存。第 05 行列印出 scores 的內容，讀者可觀察一下 Python 以何種格式列印二維列表。程式第 06 行判斷 scores[2] 是否含有超過 1 個分數，若有，第 07 行輸出 scores[2][1]；若無，則第 09 行為 scores[2] 增加一個 0 分的資料。第 10 行再次列印 scores 的內容。

E3-7-2.py： 二維列表的建立與取用

```
01 scores = []
02 for _ in range(3):
03     stu = [int(e) for e in input().split()]
04     scores.append(stu)
05 print(scores)
06 if len(scores[2])>1:
07     print(scores[2][1])
08 else:
09     scores[2].append(0)
10 print(scores)
```

執行結果 1：

```
100 90
70
60 80
[[100, 90], [70], [60, 80]]
80
[[100, 90], [70], [60, 80]]
```

執行結果 2：

```
100 90
70
60
[[100, 90], [70], [60]]
[[100, 90], [70], [60, 0]]
```

我們經常以 for 迴圈搭配 range() 產生索引值來走訪一維列表的所有資料，由於二維列表有兩個索引值，可想而言，我們需要兩個 for 迴圈來控制這兩個索引值 – 而且，這兩個迴圈是巢狀關係。

讓我們先理解巢狀 for 迴圈的執行流程與結果。下列左邊的程式碼有兩個獨立的 for 迴圈。第一個 for 迴圈會讓程式第 02 行執行三次，列印 0、1 和 2；第二個 for 迴圈執行後列印 * 0 和 * 1，然後程式結束。右邊的程式碼則是將第二個 for 迴圈置入第一個 for 迴圈中，形成巢狀迴圈。此時，第一個（外圍）for 迴圈會讓程式第 02-04 行都執行三次。藉由這樣的巢狀迴圈，我們能創造出 (0, 0)、(0, 1)、(1, 0)、(1, 1)、(2, 0)、(2, 1) 這樣六個組合。讀者們發現了嗎？這樣的組合，恰好可以用來走訪二維列表！

兩個獨立的 for 迴圈

```
01 for i in range(3):
02     print(i)
03 for j in range(2):
04     print('*',j)
```

執行結果：

```
0
1
2
* 0
* 1
```

巢狀的 for 迴圈

```
01 for i in range(3):
02     print(i)
03     for j in range(2):
04         print('*',j)
```

執行結果：

```
0
* 0
* 1
1
* 0
* 1
2
* 0
* 1
```

程式 E3-7-3 綜合了前述幾個程式範例所講授的概念與語法。首先，第 01-04 行取得使用者輸入的三個學生的考試分數。第 06 行和第 08 行形成巢狀迴圈，第 06 行的 i 代表學生的索引值（第幾個學生），所以 range() 傳入的是二維列表 scores 的長度；而第 08 行的 j 代表分數的索引值（第幾個分數），所以 range() 傳入 scores[i] 的長度，即學生 i 的分數的個數。

第 07-10 行的目的是計算學生 i 的分數 scores[i] 中有幾個是 60 以上（通過考試），因此以第 08 行的迴圈來產生分數的索引值 j，然後依序取出 scores[i][j] 並判斷是否及格，若及格，則 pa 的值加 1。最後，第 11-12 行顯示每位學生的考試通過次數。

E3-7-3.py: 以雙層迴圈與索引值走訪二維列表

```
01 scores = []
02 for _ in range(3):
03     stu = [int(e) for e in input().split()]
04     scores.append(stu)
05
06 for i in range(len(scores)):
07     pa = 0
08     for j in range(len(scores[i])):
09         if scores[i][j]>=60:
10             pa += 1
11     print('學生', i+1, '的', len(scores[i]),
12     '次考試通過', pa, '次')
```

執行結果：

```
100 80 90
60 70 50
50 40 60 30
學生 1 的 3 次考試通過 3 次
學生 2 的 3 次考試通過 2 次
學生 3 的 4 次考試通過 1 次
```

隨堂練習

3.7.1. 撰寫一個程式，讀入五位學生的成績（成績數量不定），每位學生的成績在一行，輸出五位學生的四次成績（由高到低）和平均分數，以二維表格呈現。

- 若某位學生的成績數量少於四次，不足的成績補零分。
- 若某位學生的成績數量多於四次，只採計最高分的四次。

執行結果：範例一

```
100 90 80 70
60 50 40 30
30 30 30 30
20 30 40 50
91 92 93 94
```

輸入

```
100 90 80 70 85.0
60 50 40 30 45.0
30 30 30 30 30.0
50 40 30 20 35.0
94 93 92 91 92.5
```

輸出

執行結果：範例二

```
100 90 80 70
80 80 80 80
100 100
100 0 10 20 90 90 100
```

輸入

```
100 90 80 70 85.0
80 80 80 80 80.0
100 100 0 0 50.0
100 100 90 90 95.0
0 0 0 0 0.0
```

輸出

（參考答案）

```
01  NumStudens = 5
02  NumCountedScores = 4
03
04  data = []
05  for _ in range(NumStudens):
06      data.append([int(e) for e in input().split()])
07
08  for stu in data: # 對每位學生成績 stu
09      # 將 stu 由小到大排序，反轉，取出前 NCS 個
10      stu = sorted(stu)[::-1][:NumCountedScores]
11      # 不足 NCS 個分數補 0 分
12      for _ in range(NumCountedScores-len(stu)):
13          stu.append(0)
14      # 列印分數與平均
15      for sc in stu:
16          print(sc, end=' ')
17      print(sum(stu)/len(stu))
18  print()
```

排序的部份也可以利用 sorted() 的參數 reverse 來由大到小排序，sorted(stu, reverse=True)。
另外，如果不要保留原始資料的順序，也可以直接排序，stu.sort(reverse=True)。

Chapter 4

函式

/ 蔣宗哲、江政杰

隨著程式碼的規模越來越大，我們的程式功能也越來越豐富；但同時也變得越來越難維護和除錯。本章將介紹程式語言中的重要元件 -- 函式，撰寫函式是邁向中型程式的必要過程，本章分為五節，內容包括：函式的基本概念與語法、函式與變數的可用點、預設引數和關鍵字引數、函式作為引數以及程式模組。本章以臺北市自行車竊案資料為分析標的，學習完本章後，讀者將能對多欄位的文字檔資料進行統計、查詢和排序等分析。

4-1 函式初探

到目前為止，我們在不知不覺中學會了使用許多 Python 的內建函式，例如關於輸入與輸出的 input() 與 print() 函式以及關於統計的 sum() 與 min() 函式。函式可以幫忙程式員用簡短、易懂且有效率的方式來撰寫程式。這一章就讓我們來學習如何撰寫自己的函式。

什麼是函式？

函式將一段程式碼包裝成一個單元，程式員呼叫函式以執行該段程式碼。該段程式碼會共同完成一個明白、特定的功能。Python 的函式可說是由三個部份來組成：

1. 名稱：以一個清楚易懂的名稱來代表該段程式碼的功能，例如 print。
2. 參數列：接收使用該函式的程式員所提供的資料，例如 print() 函式接收要列印的資料內容。
3. 主體：用以實際完成功能的程式碼。主體中還可以回傳計算後的結果。

讓我們以程式 E4-1-1 來說明。程式第 01 行以 def 開頭，即代表要開始撰寫一個函式，len 代表函式名稱，小括號裡面的 data 稱為參數，會接受呼叫 len() 函式時傳遞的資料，最後還有一個冒號。這一行中，粗體的文字是固定的語法格式，只有函式名稱以及小括號內的內容是可以變動的。接下去的第 02-05 行內縮的程式碼，就是 len() 這個函式的主體。第 02-04 行的內容讀者應該已很熟悉，其作用為走訪 data 並以 ans 累計 data 中的資料個數。第 05 行是一個在函式內才能執行的 return 敘述，表示在此結束函式並將 ans 回傳給函式的呼叫者。程式第 06 行沒有內縮了，表示它不是函式的一部份，此行建立了列表 d。第 07 行呼叫 print() 函式來列印資料，我們把 len(d) 作為要列印的資料，此處便是呼叫了 len() 函式，並將 d 傳遞給 len() 函式的 data 變數。len() 函式執行其主體後，會回傳列表的資料個數 5，然後 print() 便會列印出 5。

E4-1-1.py: 函式的定義與呼叫

```
01  def len(data):
02      ans = 0
03      for e in data:
04          ans += 1
05      return ans
06  d = [1, 2, 10, 4, 88]
07  print(len(d))
```

執行結果：

```
5
```

讀者第一次閱讀此程式可能會感到比較吃力，不用擔心。繼續閱讀本章的過程中，我們會反覆說明與理解函式的各個部份，便會越來越理解了。

和 if、for 句型一樣，函式裡的程式碼要內縮。

撰寫函式的三種型式

前一個段落我們以完整的型式來介紹函式，此處讓我們再以三個範例來示範函式的三種型式。

- 基本型（無傳入傳出）：最簡單的函式型式是空參數列，也沒有回傳資料，如下列程式碼第 01–02 行定義了函式 hello()，其功能是列印固定的文字 Hello Python。當程式員呼叫了函式 hello()，便會列印 Hello Python。

```
01  def hello():
02      print('Hello Python')
03  hello()
```

執行結果：

```
Hello Python
```

- 傳入資料：當函式的參數列（即小括號內）不為空白時，代表呼叫函式的程式員可以傳入資料以客製化函式的行為表現。例如下列程式碼第 01–02 行定義了函式 echo()，以參數 word 接受一個函式外傳入的資料，然後列印 Hello 加上傳入的資料。當程式第 03 行呼叫 echo('NTNU') 時，傳入字串 'NTNU'，使得該次呼叫列印 Hello NTNU；第 04 行呼叫 echo('Python')，傳入字串 'Python'，使得該次呼叫列印 Hello Python。由此我們知道，當函式具有參數列時，可以增加函式的彈性。

```
01 def echo(word):
02     print('Hello ' + word)
03 echo('NTNU')
04 echo('Python')
```

執行結果：

```
Hello NTNU
Hello Python
```

- 傳入傳出：最完整的函式型態就是有傳入資料，也有傳出資料，如下列程式範例。第 01–02 行定義函式 sum3()，參數列有三個參數，可以接收函式外部傳入三個資料。函式會傳回三個資料相加的結果。因此，第 03 行呼叫 sum3() 時傳入 10、9 和 8，此時 sum3() 的 a、b 和 c 會依序指稱 10、9 和 8。所以第 02 行回傳了三個整數相加的結果 27，使得 print() 列印出 27。第 04 行傳入三個字串，此時 sum3() 回傳三個字串相加的結果並由 print() 印出。

```
01 def sum3(a, b, c):
02     return a+b+c
03 print(sum3(10, 9, 8))
04 print(sum3('Hi ', 'Python', '!'))
```

執行結果：

```
27
Hi Python!
```

函式的進入與返回

在程式碼中，如果出現了函式定義，程式的流程會有什麼改變呢？首先，沒有被呼叫的函式就不會有作用。例如下面的程式碼中，第 03-04 行定義了函式 f()；然而，整個五行的程式碼中，並沒有任一處呼叫 f()。因此，整個程式執行完畢，不會執行第 04 行的 print() 敘述。

```
01  print('第一行')
02  print('第二行')
03  def f():
04      print('沒人用我，就不會印出這行')
05  print('第三行')
```

執行結果：

```
第一行
第二行
第三行
```

當函式被呼叫，程式會進入到函式內部去執行其程式碼；執行完函式的程式碼或者執行了 return 敘述後，會結束函式並回到呼叫點繼續執行。以下列程式為例:首先會執行第 01 行。接下來會跳過第 02-04 行的 echo() 函式定義。執行第 05 行時，將呼叫 echo() 函式，並傳入字串 '1'。接著程式流程便會走到第 03 行，然後是第 04 行，結束 echo() 函式後回到第 05 行，繼續往下執行第 06 行。第 07 行再一次呼叫 echo()，這次傳入字串 '2'，程式再一次走到第 03 與 04 行。這次結束後，將會返回到第 07 行，然後往下執行第 08 行。

E4-1-2.py: 函式的執行流程

```
01  print(' 開始 ')
02  def echo(word):
03      print(' 現在在 echo() 函式裡面 ')
04      print('Hello ' + word)
05  echo('1')
06  print('------')
07  echo('2')
08  print(' 結束 ')
```

執行結果：

```
開始
現在在 echo() 函式裡面
Hello 1
------
現在在 echo() 函式裡面
Hello 2
結束
```

return 的用途

本節的最後，讓我們來談一下 return 敘述。首先，return 代表要結束函式並返回呼叫點，所以 return 敘述一定要放在函式定義中。在函式中，只要執行到 return 敘述，就會立刻結束函式。以下面的程式為例，第 01-04 行定義了函式 f()，而第 05 行呼叫了 f()。呼叫後，程式將執行第 02 行，接著是第 03 行。由於第 03 行是 return 敘述，所以函式在這裡結束，也就不會執行到第 04 行。（本範例純粹是為了說明 return 會結束函式，正常情況下我們不會寫出永遠執行不到的程式碼。）

```
01  def f():
02      print(' 進來 f()')
03      return
04      print(' 這一行跑不到 ')
05  f()
```

執行結果：

```
進來 f()
```

除了單純地結束函式，return 敘述也可以回傳資料，以下我們將看三個範例。首先，程式 E4-1-3 示範如何傳回單一個數值。第 01-05 行定義函式 avg()，參數列有一個參數 data 接受一個資料，從第 02 與 03 行來看，可以看出 data 預計是要接受一個數值列表。若列表的長度不為零，return 回傳列表中數值的平均；否則，return 回傳 0。第 06 行呼叫 avg() 時傳入空列表，因此 avg() 將執行第 05 行回傳 0，致使 print() 列印出 0。第 07 行呼叫 avg() 時傳入列表 [1, 3, 5]，因此 avg() 將執行第 03 行，呼叫 sum() 和 len()，並將 [1, 3, 5] 傳入這兩個函式。我們知道 sum() 和 len() 的功能是計算列表中數值總和以及數值個數，因此知道 sum() 將回傳 9 而 len() 將回傳 3 到第 03 行。接著，第 03 行再將 9/3 回傳到第 07 行，致使 print() 列印出 3.0。

E4-1-3.py: 以 **return** 傳回一個數值。

```
01  def avg(data):
02      if len(data)!=0:
03          return sum(data)/len(data)
04      else:
05          return 0
06  print(avg([]))
07  print(avg([1, 3, 5]))
```

執行結果：

```
0
3.0
```

return 也可以傳回多個值（以元組 tuple 型態回傳）。例如程式 E4-1-4 的第 02 行就回傳 max(data) 和 min(data) 兩個值。第 03 行呼叫 max_min() 時傳入列表 [3, 4, 1, 2]，即 max_min() 的 data 代表 [3, 4, 1, 2]；因此第 02 行的 max() 回傳 4 而 min() 傳回 1，而 return 敘述就會回傳 4 和 1 到第 03 行。第 03 行使用分解賦值的語法，讓 M 指稱 4 而 m 指稱 1。最後，第 04 行列印出 4 和 1。

E4-1-4.py: 以 return 傳回多個數值

```
01  def max_min(data):
02      return max(data), min(data)
03  M, m = max_min([3, 4, 1, 2])
04  print(M, m)
```

執行結果：

```
4 1
```

最後，讓我們看看程式 E4-1-5 示範回傳列表。第 01-02 行定義函式 remove_zeros()，第 02 行使用第 3-3 節曾講過的列表推導式從 data 中取出不為 0 的值建立一個新的列表，然後以 return 回傳。第 03 行呼叫 remove_zeros()，傳入列表 [3, 2, 0, 3, 9, 0] 給第 01 行的 data。第 02 行回傳 [3, 2, 3, 9] 給第 03 行的 result。第 04 行再呼叫 print() 函式列印 result。

E4-1-5.py: 以 return 傳回列表

```
01  def remove_zeros(data):
02      return [e for e in data if e!=0]
03  result = remove_zeros([3, 2, 0, 3, 9, 0])
04  print(result)
```

執行結果：

```
[3, 2, 3, 9]
```

如果函式 f() 沒有回傳值，a = f() 會令 a 的值為 None。
試試這行：print(print())。

隨堂練習

4.1.1 自行車竊案統計

- **自本書附件取得** TaipeiCityBikeCrime.txt 以及 E4-1-6.py。
- 程式 E4-1-6 中的 GetCrimeData() 函式可以傳入一個檔案名稱，並將檔案從第二行起讀成二維陣列傳回。
    - 請基於程式 E4-1-6，新增 GetYearCount() 函式，並在 main() 函式中呼叫 GetYearCount() 以查詢某一年度（如民國 106 年）的自行車竊案數。
    - 讀者可以撰寫更多的函式來查詢特定月份（如 8 月份）與特定行政區（如松山區）的自行車竊案數。

E4-1-6.py：隨堂練習 **4.1.1**

```
01  def GetCrimeData(filename):
02      data = []
03      with open(filename, encoding='UTF-8') as f:
04          f.readline()
05          for line in f:
06              data.append(line.split())
07      return data
08  def GetYear(record):
09      return int(record[2][:3])
10  def main():
11      data = GetCrimeData('TaipeiCityBikeCrime.txt')
12      print('第一筆竊案發生在', GetYear(data[0]) , '年')
13      qyear = int(input('請輸入欲查詢年份'))
14      print(qyear, '年總案件數=', GetYearCount(data, qyear))
15
16  main()
```

4-2 函式與變數的可用點

函式的可用點

在我們開始撰寫自己的函式之後，可能會發生無法呼叫函式的錯誤。例如執行下面左方的程式時，會出現 NameError: name 'f' is not defined，該錯誤訊息表示 f 這個名稱尚未定義。會發生這個錯誤，是因為程式由上往下執行到呼叫點，也就是第 01 行時，尚未看過函式 f() 的定義，因此不知道有 f() 的存在。把呼叫 f() 的程式碼移到第 03 行，如右方程式所示，在由上往下執行到第 03 行這個呼叫點前已經看過 f() 的定義，就可以成功呼叫 f() 了。

```
01 f()
02 def f():
03     print('f()')
```

發生 NameError

```
01 def f():
02     print('f()')
03 f()
```

輸出 f()

事實上，Python 的「見過」與否是以執行流程來判定，不是單純以呼叫點和函式定義的位置前後來判定。以下面的程式為例，程式由上而下一直到第 08 行才真正開始執行，此處呼叫函式 f()，隨即跳到第 02 行，接著第 03 行呼叫 g()。雖然第 03 行的呼叫寫在第 05-06 行的函式 g() 定義之前，但由於程式已經一路走到第 08 行才跳回第 02 行，所以已經看過 g() 的定義了。因此，第 03 行可以成功呼叫 g()。

```
01 def f():
02     print('f()')
03     g()
04
05 def g():
06     print('g()')
07
08 f()
```

Python 程式中常見一種組織函式的方式，是定義一個主函式（可稱為 main()，但也可以定成其它名稱），將程式所有要作的任務寫在主函式中，然後把呼叫主函式的敘述寫在程式碼的最下面一行。如此一來，在呼叫主函式之前必定會看過所有函式定義，也就不必擔心會發生無法呼叫函式的情況了。

程式 E4-2-1 便是示範這種常見的組織方式。儘管 main() 函式定義中呼叫 f()（第 02 行）和呼叫 g()（第 04 行）的敘述都在 f() 和 g() 的定義之前，但因為 main() 函式是在第 13 行呼叫，因此在程式從第 13 行跳到第 02 行時，已經看過第 06–08 行的 f() 定義以及第 10–11 行的 g() 定義，是以第 02 行和第 04 行的呼叫都會成功。

E4-2-1.py: 定義主函式來呼叫其它函式

```
01  def main():
02      f()
03      print('.')
04      g()
05
06  def f():
07      print('f()')
08      g()
09
10  def g():
11      print('g()')
12
13  main()
```

執行結果：

```
f()
g()
.
g()
```

有些語言要求 define higher，而 Python 要求 define earlier。

變數的可用點

程式中不只是呼叫函式可能發生 NameError，使用變數也可能發生。例如下面兩個程式片段，左方程式第 01 行先定義了變數 b，第 02 行再使用 b 定義 a，沒有問題。右方程式第 01 行便想要用變數 b 來定義 a，但此時仍未定義過 b，因此就造成了 NameError。

01 02	b = 3 a = b	01 02	a = b b = 3
沒有錯誤		NameError: name 'b' is not defined.	

變數定義與否，會和程式流程有關。例如下面的程式碼中，第 01 行 if 的條件式為 True，因此執行第 02 行定義了變數 a，沒有執行第 04 行定義變數 b。這便造成第 06 行發生 b 未定義的 NameError。

```
01 if True:
02     a = 3
03 else:
04     b = 4
05 print(a)
06 print(b)
```

定義在函式外的變數稱為全域變數，函式內部可以取用全域變數。例如下面兩個程式片段，函式 f() 中都可以順利取用到全域變數 i 來列印。

01
02
03
04
	i = 9
	def f():
	    print(i)
	f()
		01
		02
		03
		04
			def f():
			    print(i)
			i = 9
			f()

列印 9		列印 9	

讀者可能對上述右方的程式感到疑惑，事實上，這個程式之所以可行的道理和前面我們談到函式是否可用的道理相同；我們看的是程式執行的過程，而不是程式碼的位置。上述右方的程式雖然第 02 行 print(i) 在第 03 行 i = 9 之前，但是程式執行的流程是先執行第 03 行的定義，接著第 04 行呼叫 f()，然後才作到第 02 行的 print(i)。所以，在執行 print(i) 時，變數 i 已經定義過了。

下面再用一個程式說明「流程」而非「位置」決定結果。第 01 行定義變數 i 為 9，緊接著第 04 行將 i 改為 8，然後第 05 行呼叫 f()，程式跳到第 03 行執行 print(i)，因此列印的結果為 8。不要因為第 03 行放在第 01 行之後、第 04 行之前，而誤以為 print(i) 會印出 9 喔。

```
01  i = 9
02  def f():
03      print(i)
04  i = 8
05  f()
```

定義在函式內的變數稱為區域變數，之所以稱為區域，是因為區域變數只能在它的定義式所在的函式區域使用。下列三個程式片段中，第 04 行 print(fi) 取用 fi 都會造成 NameError。

```
01  def f():
02      fi = 9
03
04  print(fi)
```

```
01  def f(fi):
02      print(fi)
03  f(9)
04  print(fi)
```

```
01  def f():
02      fi = 9
03  def g():
04      print(fi)
05  g()
```

與其將「區域變數的存取範圍僅在所屬函式中」視為一種限制，不如將它視為一種保護機制。因為有這樣的保護，當區域變數的數值有誤而需要對程式除錯時，我們可以專注於該函式定義的程式碼就好，不需要把整個程式檔的每一行都檢查一次。

學到這裡，我們讓讀者來看看下面這個程式。這個程式定義了 GetPrice() 函式，其作用是根據物品的數量和單價來計算總價，並且在數量為 10 個以上的時候將總價打九折。請讀者思考看看，這個程式最後會輸出什麼呢？

```
01  def GetPrice():
02      price = num*unit_price
03      if num>=10:
04          price *= 0.9
05
06  price = 0
07  num = 10
08  unit_price = 8
09  GetPrice()
10  print(price)
```

答案是 0，您答對了嗎？關鍵在於這個程式定義了兩個 price 變數。第 02 行定義了 GetPrice() 函式內的區域變數 price，而第 06 行定義了全域變數 price。GetPrice() 函式內操作的都是區域變數 price（所以也沒有改變全域變數 price 所指稱的值），而第 10 行取用到的則是全域變數 price，這個變數自始至終都是 0。

雖然有方法可以在函式內部修改全域變數，我們建議透過參數列將數值傳遞進函式內，再以 return 敘述將運算結果傳回呼叫端。這樣的作法可以在閱讀程式的時候更清楚理解資料的流動。依據這樣的建議，我們將上述程式改寫如程式 E4-2-2。首先，我們在第 01 行 GetPrice() 函式的參數列中增加了兩個參數 n 和 p 來接受外部傳入的數量和單價。函式內我們定義了區域變數 price 來計算總價，在第 05 行將 price 傳回。函式之外，我們在第 09 行呼叫 GetPrice() 時傳入 num 和 unit_price。這樣讓閱讀程式的人一眼就知道 GetPrice() 的運算結果會和 num 與 unit_price 有關。這一行定義了全域變數 price 並以 GetPrice() 的回傳值來設定它。程式第 10 行列印全域變數 price 的值。

E4-2-2.py: 將資料透過參數列傳入，透過 **return** 敘述回傳

```
01 def GetPrice(n, p):
02     price = n*p
03     if n>=10:
04         price *= 0.9
05     return price
06
07 num = 10
08 unit_price = 8
09 price = GetPrice(num, unit_price)
10 print(price)
```

最後讓我們再看一個對比的例子。下面程式碼定義了 Adjust() 函式，其作用是計算以分數 score 開根號乘以 10。左方的程式碼「誤以為」能直接修改全域變數，但其實第 01 行的 score 參數是屬於 Adjust() 函式中的區域變數，第 02 行修改的是這個區域變數，而不是第 04 行定義的全域變數。因此，

第 05 行呼叫完 Adjust() 後，全域變數的值仍為 36。右方的程式碼則是利用函式的回傳值修改了全域變數，達到目的。

```
01 def Adjust(score):
02     score = score**0.5*10
03
04 score = 36
05 Adjust(score)
06 print(score)
```

列印 36

```
01 def Adjust(score):
02     return score**0.5*10
03
04 score = 36
05 score = Adjust(score)
06 print(score)
```

列印 60.0

隨堂練習

4.2.1 任務：自行車竊案統計 (II)

- 前一節的隨堂練習中，我們撰寫了程式查詢特定年份、月份和行政區的竊案總數。
- 由於只能查詢單一行政區的竊案數，如果想要掌握所有行政區的竊案數（例如想知道哪個行政區的竊案數最多），就必須先知道所有行政區的名稱並多次呼叫函式，這相當麻煩。
- 這個練習希望讀者能撰寫出一個函式 `GetPlaceStat()`，一次就能統計所有行政區的竊案數。

E4-2-3.py：隨堂練習 **4.2.1**

```
01  def GetCrimeData(filename):
02      data = []
03      with open(filename, encoding='UTF-8') as f:
04          f.readline()
05          for line in f:
06              data.append(line.split())
07      return data
08
09  def GetPlace(record):
10      return record[4][3:6]
11
12  def GetPlaceStat(data):
13
14  def main():
15      data = GetCrimeData('TaipeiCityBikeCrime.txt')
16      results = GetPlaceStat(data)
17      print(results)
18
19  main()
```

4-3 預設引數與關鍵字引數

前一節我們看了好幾個將資料傳遞到函式的範例，相信讀者已經逐漸熟悉資料傳遞的過程：在呼叫函式時放置引數（被傳遞者），引數被複製到參數（接受者），然後函式內部再使用參數來運算。本節我們將介紹兩個傳遞資料的進階語法：預設引數值和關鍵字引數。

預設引數值

在呼叫函式時，如果傳入的引數個數不足（比參數個數少），程式執行時便會發生錯誤。例如下面的程式片段，在執行到 05 行的時候，會產生 TypeError 錯誤，錯誤訊息指出缺少了一個需要的引數給參數 b。

```
01  def f(a, b):
02      print(a, b)
03
04  f(1, 2)
05  f(1)
```

執行結果：

```
1 2
TypeError: f() missing 1 required positional argument: 'b'.
```

所謂「預設引數值」，指的是在參數列為參數設定預設值；當呼叫函式者未提供足夠的引數值時，便會以預設值作為引數。以下列程式為例，第 01 行未設定參數 a 的預設值，但設定了參數 b 的預設值。在第 03 行呼叫者傳遞了兩個引數值 1 和 2，因此參數 a 和 b 分別指稱了 1 和 2 這兩個整數值，在畫面上會輸出 1 2。然而，在第 04 行呼叫者只傳遞了一個引數值 1，此時引數和參數會由左往右配對，因此參數 a 指稱整數 1，而參數 b 沒有引數，將使用預設值，也就是 0，畫面上輸出了 1 0。第 05 行呼叫沒有傳遞任何引數，由於 a 並沒有預設值，因此產生了錯誤。

```
01  def f(a, b = 0):
02      print(a, b)
03  f(1, 2)
04  f(1)
05  f()
```

```
執行結果：
1 2
1 0
TypeError: f() missing 1 required positional argument: 'a'
```

在設定預設引數值時，要注意只能由右往左設定，也就是不能發生左邊的參數有預設值但右邊的參數沒有的情況。例如下面的程式碼會造成語法錯誤（Nondefault argument follows default argument）。

```
01  def f(a = 0, b):
02      print(a, b)
```

關鍵字引數

當我們傳遞多個引數時，引數和參數的配對是依位置次序由左往右配對；當引數數量越來越多時，就可能發生引數位置錯誤，但如果程式還是能執行，程式員就不一定會發現（需要仔細查看執行結果才會發現）。以程式 E4-3-1 為例，第 01-02 行定義了函式 `f()`。第 03 行呼叫 `f()`，將 180 傳給 `height`，78 傳給 `weight`，程式也順利印出「身高 180 體重 78」。但若如第 04 行的呼叫，不小心將兩個引數顛倒了，就會導致「身高 78 體重 180」這樣不合理的狀況。

關鍵字引數可以幫助我們避免上述傳遞順序不慎而引起的錯誤。例如第 05 行，我們可以明確指定參數 `weight` 為 60，參數 `height` 為 170。注意，此時的順序不會影響，不管先寫 `weight=60` 還是 `height=170` 都可以。我們也可以一部份的引數是依位置，一部份的引數是以關鍵字方式來傳遞，如第 06 行所示。但關鍵字引數只能出現在位置引數之後，所以第 07 行會產生錯誤「positional argument follows keyword argument」。

E4-3-1.py: 使用關鍵字引數的範例

```
01  def f(height, weight):
02      print('身高', height, '體重', weight)
03  f(180, 78)
04  f(78, 180)
05  f(weight=60, height=170)
06  f(90, weight=15)
07  f(height=3, 5)   # 關鍵字引數只能出現在位置引數之後
08
09  def g(a, b, c):
10      print(a, b, c)
11  g(1, 2, 3)
12  g(1, c=3, b=2)
```

執行結果：

```
身高 180 體重 78
身高 78 體重 180
身高 170 體重 60
身高 90 體重 15
1 2 3
1 2 3
```

程式 E4-3-2 示範了運用預設引數值的多種呼叫函式的可能性，讀者們可以先不要看下面的執行結果，自行推論出結果，再對照是否正確。

E4-3-2.py: 使用預設引數值的範例

```
01  def f(a=1, b=2, c=3):
02      print(a, b, c)
03  f(9, 8)
04  f(9)
05  f(a=9)
06  f(b=8)
07  f(c=7, a=9)
```

執行結果：

```
9 8 3
9 2 3
9 2 3
1 8 3
9 2 7
```

過去我們在呼叫 Python 內建函式時，其實就大量依賴了預設引數值。例如 print() 函式的參數 sep 的預設值是空白字元 ' '，而參數 end 的預設值是換行字元 '\n'。在程式 E4-3-3 中，第 01 行的呼叫未傳遞引數給 sep 和 end，因此採用預設值，該呼叫會列印 1 2 3 並換到下一行。第 02 行的呼叫傳遞了 ',' 給 sep，但 end 仍是預設值 '\n'，該呼叫會列印 1,2,3 並換到下一行。第 03 行則是傳遞 'X' 給 end，但 sep 是預設值 ' '，因此列印 1 2 3X，沒有換行。第 04 行則同時指定了 sep 和 end，因此會印出 1@2@3.。函式 sorted() 有一個參數 reverse，預設為 False。當我們沒有傳遞引數給 reverse，sorted() 將資料由小到大排序；若我們將 reverse 設為 True，則 sorted() 會將資料由大到小排序。

E4-3-3.py： 內建函式的關鍵字引數呼叫以及預設引數值

```
01 print(1, 2, 3)
02 print(1, 2, 3, sep=',')
03 print(1, 2, 3, end='X')
04 print(1, 2, 3, sep='@', end='.')
05 print()
06 print(sorted([9, 1, 4]))
07 print(sorted([9, 1, 4], reverse=True))
```

執行結果：

```
1 2 3
1,2,3
1 2 3X1@2@3.
[1, 4, 9]
[9, 4, 1]
```

4-4 函式作為引數

透過將程式碼需操作的資料設為參數，然後由呼叫函式的程式員將資料以引數傳入，可以提升程式碼的彈性與重用性。例如我們可以傳入各種不同的資料給 print() 去列印，傳入各個不同的列表給 sorted() 去排序。本節我們將介紹一個有點酷的概念，就是把函式作為引數來傳遞給另一個引數。是不是有點難想像？沒關係，我們先從把函式當成變數來學起。

函式如變數使用

在 Python 中，函式可以像個變數來使用。程式 E4-4-1 的第 01-02 和 03-04 行分別定義了函式 f() 和 g()。第 05 行我們令變數 ff 指稱函式 f，因此第 06 行我們可以將 ff 當成一個函式來呼叫 ff(4)，此時等同於呼叫 f(4)。第 07 行建立了一個列表 funcs，其中有兩個元素 f 和 g。第 08 行的 for 句型會依序從 funcs 中取出 f 和 g，令 e 指稱它們，然後透過 e(7) 來呼叫 f(7) 和 g(7)。

E4-4-1.py: 函式如變數的使用範例

```
01 def f(i):
02     print('f(', i, ')', sep='')
03 def g(j):
04     print('g(', j, ')', sep='')
05 ff = f
06 ff(4)
07 funcs = [f, g]
08 for e in funcs:
09     e(7)
```

執行結果：

```
f(4)
f(7)
g(7)
```

將函式作為引數

讓我們作一個更有意義的範例：老師以兩種調分函式處理成績。程式 E4-4-2 中第 01-09 行定義了三個函式 sqrt10()、add10() 和 process()。函式 sqrt10() 會接受一個數值，然後回傳該數值開根號乘以 10 的結果，函式 add10() 會接受一個數值，然後回傳該數值加 10 的結果。函式 process() 有兩個參數，data 代表一個數值列表而 f 代表一個函式。第 07 行設定索引值 i，而第 08 行則將 data[i] 設為 f(data[i]) 回傳的結果。這裡的重點是，函式 f 是可以由呼叫 process() 的程式員來決定的，不需要固定寫死在程式碼中。因此，當第 11 行傳入 sqrt10，process() 便將所有數值都開根號乘以 10，而當第 12 行傳入 add10，process() 便將所有數值都加 10。未來如果還有新的調分函式，process() 仍然可以重覆使用，豈不妙哉！

E4-4-2.py: 函式作為引數的範例

```
01 def sqrt10(s):
02     return int((s**0.5)*10)
03 def add10(s):
04     return s+10
05
06 def process(data, f):
07     for i in range(len(data)):
08         data[i] = f(data[i])
09     return data
10
11 print(process([36, 49, 64], sqrt10))
12 print(process([36, 49, 64], add10))
```

執行結果：

```
[60, 70, 80]
[46, 59, 74]
```

許多 Python 內建函式都將關鍵的操作開放成為參數，讓呼叫者有更大的彈性與操控權。以下我們就以 max() 和 sorted() 為例來說明能傳入函式有多麼方便。

內建函式 max() 的函式參數 key

內建函式 max() 有一個參數 key，在 max() 內部會被呼叫作為比較資料的基準。以程式 E4-4-3 為例，第 01-04 行定義了兩個函式，last() 會接受一個數值並回傳該數的個位數，而 last2sum() 會接受一個數值並回傳該數最後兩位數字的和。

第 07 行沒有傳入任何函式，此時 max() 會以資料的原值來比較，並回傳 data 列表中的最大值，即為 61。第 08 行傳入 last 給 key，此時 max() 會以每個資料傳入 last() 的結果來比較，因此最後會回傳個位數最大的資料，也就是 25（data 中五個整數的個位數分別為 2、4、2、1 和 5。）。第 09 行傳入 last2sum 給 key，此時 max() 會回傳十位數和個位數相加最大的值，結果為 44（其兩位數字相加為 8，大於另外四個數的十位和個位數的和。）

E4-4-3.py: 傳入自訂函式作為 max() 的 key

```
01  def last(v):
02      return v%10
03  def last2sum(v):
04      return v//10 + v%10
05
06  data = [32, 44, 52, 61, 25]
07  print(max(data))
08  print(max(data, key=last))
09  print(max(data, key=last2sum))
```

執行結果：

```
61
25
44
```

程式 E4-4-4 是一個處理多維度資料的範例。已知學生資料是由姓名、身高和體重三個資料構成的列表，函式 GetHeight() 回傳列表的第二項 v[1]，即為身高；而函式 GetWeight() 回傳列表的第三項 v[2]，即為體重。

第 07 行將 GetHeight 傳給 max() 的 key，因此 max() 會以每個列表的第二項（身高）來比較，並回傳該項最大的列表，也就是回傳身高最高的學

生資料，其結果為 ['李四', 190, 80]（190 大於 180 和 175）。而第 08 行將 GetWeight 傳給 max() 的 key，使得 max() 回傳體重最重的學生資料，結果為 ['王五', 175, 90]（90 大於 75 和 80）。

E4-4-4.py: 指定列表欄位作為 max() 的 key

```
01 def GetHeight(v):
02     return v[1]
03 def GetWeight(v):
04     return v[2]
05
06 data = [['張三', 180, 75], ['李四', 190, 80], ['王五', 175, 90]]
07 print(max(data, key=GetHeight))
08 print(max(data, key=GetWeight))
```

執行結果：

```
['李四', 190, 80]
['王五', 175, 90]
```

內建函式 sorted() 的函式參數 key

函式 sorted() 和 max() 一樣，也有一個參數叫 key，作用也類似，決定 sorted() 排序時會以什麼數值來排序。程式 E4-4-5 的第 03 行沒有傳入函式給 sorted() 的 key，因此以原值來排序，得到由小到大的結果 [3, 65, 74, 82, 91, 100]。第 04 行傳入 last 作為 key，因此 sorted() 會以個位數來排序，使得結果變成 [100, 91, 82, 3, 74, 65]（其個位數是 0, 1, 2, 3, 4, 5）。

E4-4-5.py: 傳入自訂函式作為 sorted() 的 key

```
01 def last(v):
02     return v%10
03 print(sorted([100, 91, 82, 3, 74, 65]))
04 print(sorted([100, 91, 82, 3, 74, 65], key=last))
```

執行結果：

```
[3, 65, 74, 82, 91, 100]
[100, 91, 82, 3, 74, 65]
```

程式 E4-4-6 的目的是要示範 sorted() 在排序多維度資料時，預設（第 04 行的呼叫）會先比較第一項，若第一項相同，才會再比第二項，依此類推。第 05 行傳入 mykey，因此會以第二項來排序（由小到大為 82、88、90、95、100）。

E4-4-6.py: 指定列表欄位以 sorted() 進行排序

```
01 def mykey(v):
02     return v[1]
03 data =[[1, 100], [4, 90], [3, 88], [2, 95], [4,82]]
04 print(sorted(data))
05 print(sorted(data, key=mykey))
```

執行結果：

```
[[1, 100], [2, 95], [3, 88], [4, 82], [4, 90]]
[[4, 82], [3, 88], [4, 90], [2, 95], [1, 100]]
```

程式 E4-4-7 示範了如何以兩個以上的資料來決定排序。函式 mykey() 回傳了列表的第二項和第三項，因此第 07 行將 mykey 傳入 sorted() 的 key 後，sorted() 便會先比列表的第二項，若相同再比第三項。由列印的結果可以看出，排序結果確實先依第二個項目（身高）排序，若遇到身高相同（趙六和王五都是 175），再依第三個項目（體重）排序。

E4-4-7.py: 指定多個列表欄位以 sorted() 進行排序

```
01 def mykey(v):
02     return v[1], v[2]
03 data = [['張三', 180, 75],
04         ['李四', 190, 100],
05         ['王五', 175, 90],
06         ['趙六', 175, 85]]
07 print(sorted(data, key=mykey))
```

執行結果：

```
[['趙六', 175, 85], ['王五', 175, 90], ['張三', 180, 75],
['李四', 190, 100]]
```

Lambda

將函式作為引數傳遞給另一個函式可大大擴展了函式的彈性，但撰寫程式碼還是有一點點不方便，因為我們需要額外再寫一個完整的函式定義（例如程式 E4-4-7 我們要寫出第 01-02 行）。Python 提供了所謂 lambda 的一行無具名函式，讓我們可以更直觀地將函式作為引數。

舉例來說，程式 E4-4-3 可利用 lambda 簡寫成下列程式。對照之下，我們可以了解 `lambda x: x%10` 的意思就如同一個函式藉由參數 `x` 接收傳入的資料，然後回傳 `x%10` 的結果。我們可以把 lambda 想成是這個無具名函式暫時的名字，lambda 和冒號之間放的是參數（名稱可以自訂，不一定要是 `x`），而冒號後面的運算就是回傳的結果。就文法上，要注意這裡不需要像函式用小括號把參數包起來，也不需要寫 `return`。

```
01 data = [32, 44, 52, 61, 25]
02 print(max(data, key=lambda x: x%10))
03 print(max(data, key=lambda x: x//10 + x%10))
```

程式 E4-4-4 和 E4-4-6 可使用 lambda 簡寫如下：

```
01 data = [['張三', 180, 75], ['李四', 190, 80], ['王五', 175, 90]]
02 print(max(data, key=lambda x: x[1]))
03 print(max(data, key=lambda x: x[2]))
```

```
01 data = [[1, 100], [4, 90], [3, 88], [2, 95], [4,82]]
02 print(sorted(data, key=lambda x: x[1]))
```

隨堂練習

4.4.1 任務：自行車竊案統計 (III)

- 前一節的隨堂練習我們希望讀者撰寫出一個函式 GetPlaceStat()，一次就能統計所有行政區的竊案數。
- 如果我們要修改上述函式，改為統計所有月份的竊案數，應該如何作？如何改為統計各年份？是否可透過一個函式來控制被統計的欄位？
- 修改前一節的函式為 GetStat()，除了有一個參數接收資料，新增一個參數接受欄位函式。

```
01  def GetStat(data, keyf):
02      results = []
03      for record in data:
04          key = keyf(record)
05          [略]
06      return results
07
08  def main():
09      data = GetCrimeData('TaipeiCityBikeCrime.txt')
10      print(GetStat(data, GetPlace))
11      print(GetStat(data, GetYear))
```

4.4.2 任務：自行車竊案統計 (IV)

- 前一小題的函式 GetStat() 會回傳各指定欄位的竊案次數，其回傳資料是一個二維列表，其中每個一維列表的第一個項目是欄位，第二個項目是竊案數。
- 參照程式 E4-4-6，運用 sorted() 函式，搭配傳入適當的函式，回答下 列問題：
  1. 竊案數前三多的行政區？
  2. 竊案數前三多的月份？
  3. 竊案數前三多的時刻？
- 嘗試改寫程式，以 lambda 來傳入函式。

4-5 程式模組

你都怎麼組織管理你的資料呢？大家都很熟悉將資料分門別類放在不同檔案甚或資料夾（目錄）中吧！當你撰寫的程式碼越來越多，把它們分門別類放在不同檔案中，通常也有助於管理。將程式碼分拆在不同檔案的好處包括了：

1. 將性質相近的函式放在同個檔案，方便找尋和分享。
2. 將程式碼分散在多個檔案，方便修改、測試與抽換。
3. 將程式碼分散在多個檔案，方便團隊工作，團隊中的每個人可以各自專注在自己的程式檔案，不必擔心修改到別人的程式碼。

本章前面四節我們已經學習到如何撰寫函式和呼叫函式，當函式和呼叫的程式碼不在同一個程式檔案時，又該如何處理呢？讓我們透過一個範例來學習。假設我想要計算一個列表的平均值，但我不知道該如何撰寫程式碼。我的好友大明說他手邊有一個這樣的函式可以分享給我。他傳了一個檔案 *lib1.py*，裡面是他寫好的 `mean()` 函式程式碼。此時，我需要在我的程式碼檔案 *main1.py* 中寫一行導入程式的程式碼，如程式 main1 的第 01 行所示。有了這一行之後，我就可以在程式中呼叫 lib1 中任何的函式了。例如 main1 程式第 02 行我以 `lib1.mean()` 來呼叫。

main1.py

```
01  import lib1
02  print(lib1.mean([3, 5, 8]))
```

lib1.py

```
01  def mean(data, prec=2):
02      if data:
03          return round(sum(data)/len(data),prec)
04      else:
05          return 0
```

我們將一個 *XYZ.py* 檔案稱為一個模組 XYZ，import XYZ 的作用就是帶入該模組的程式碼。如果沒有以 import 帶入、忘了寫模組名稱、或是寫錯模組名稱（如以下三個範例），都會導致錯誤喔！

```
01  print(lib1.mean([3, 5, 8]))
```
```
NameError: name 'lib1' is not defined.
```

```
01  print(mean([3, 5, 8]))
```
```
NameError: name 'mean' is not defined.
```

```
01  import libb
02  print(mean([3, 5, 8]))
```
```
ModuleNotFoundError: No module named 'libb'
```

import 敘述有兩種變化，我們先看第一種：加上 from。下列的範例中，程式 main2 第 02 行的意思是從 lib2 模組中帶入函式 f 和 g。有了這一行，之後便可以直接以函式 f() 和 g() 來呼叫，不必再寫出模組 lib2 的名字了。第 05 行的 lib2. 是可寫可不寫，但第 06 行的 lib2. 就必須寫了。

lib2.py

```
01  def f():
02      print('f()')
03  def g():
04      print('g()')
05  def h():
06      print('h()')
```

main2.py

```
01  import lib2
02  from lib2 import f, g
03  f()
04  g()
05  lib2.f()
06  lib2.h()
```

執行結果：main2.py

```
f()
g()
f()
h()
```

下面程式碼的 01 行示範一個偷懶的寫法，import * 的意思就是把所有該模組的函式一口氣帶入。這種寫法雖然方便，但也增加了兩個模組間函式或變數名稱相同而造成的名稱衝突問題；因此，在使用時要特別留心。

lib2.py

```
01 def f():
02     print('f()')
03 def g():
04     print('g()')
```

main2b.py

```
01 from lib import *
02 f()
03 g()
```

執行結果：main2b.py

```
f()
g()
```

接續前一種 import 敘述的變化，我們還可以用 as 來指定模組或函式的代稱。通常我們會用來縮短名稱或解決名稱衝突的問題。以下第一個範例中我們以 lib 代稱原有的模組名稱 this_is_a_lib_with_very_long_name。

```
01 import this_is_a_lib_with_very_long_name as lib
02 lib.f()
03 lib.g()
```

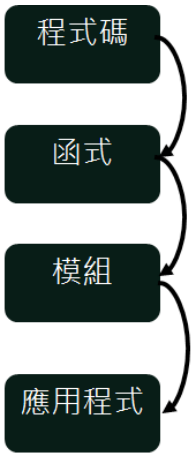
這個用法也適用於函式名稱，如下例。

```
01 from lib import a_func_with_a_very_long_name as f
02 f()
```

如果我的程式碼中已有一個 sort() 函式，我可以帶入另一個模組的 sort() 並以另一個名稱 FastSort() 來指稱它。

```
01 from lib import sort as FastSort
02 FastSort()
```

小結



當你發現有些程式碼經常被使用或是想要隱藏實作細節，便可將完成一特定功能的程式碼片段包裝成一個函式。該函式的參數代表使用者能控制此函式的彈性與權力。當你發現多個函式具有相似性、相關性或相互輔助性時，便可將這些函式歸納在一個模組（.py 檔案）。當你手邊有許多好用的模組，你就可以輕鬆地取用其中需要的函式，組成出你想要的應用程式。

讀者可能覺得「我才剛開始學寫程式，有可能寫出那麼多的程式碼，還把它們分成多個檔案嗎？」也許目前讀者們所寫的程式規模確實還不會大到需要區分模組，但是，讀者有很大的機會去使用它人的模組，例如下一章談到的圖表繪製主題，就需要帶入 matplotlib 套件的 pyplot 模組。因此，本節所介紹的 import 敘述還是很有用的喔！

Chapter 5

圖表繪製

/ 蔣宗哲、江政杰

俗話說，「一圖勝千言」。當分析過程中產生的資料趨於龐大繁雜，資料科學家們會尋求更適當的方式來呈現，其中各式圖表就是值得考慮的好工具了。本章將講述如何運用 matplotlib 套件來繪製折線圖、散佈圖和長條圖，並接續前一章的臺北市自行車竊案分析應用，展示各種圖表的應用時機。學習完本章後，讀者將可撰寫程式繪製最常見的三種圖表，設定圖表細節並存成圖片檔案以備後續編輯使用。

5-1 matplotlib 套件安裝

俗諺有云，「一圖勝千言」(A picture is worth a thousand words.)，以圖表方式呈現資料，讓人能夠一目瞭然迅速理解資料所訴說的意涵。很多時候，我們希望能以圖表型式來呈現資料分析的結果。這一章我們即將學習以 Python 程式繪製折線圖、散佈圖以及長條圖這三種圖表。

Python 語言有數種繪圖套件，這裡我們使用的是 matplotlib 套件。這一節示範的是以 Windows 作業系統安裝 matplotlib 的流程。其它作業系統的安裝方式可以參考網路上的教學文章。

準備工作：安裝 matplotlib 套件

開啟命令提示字元視窗：點擊左下角搜尋，輸入 cmd 後按 Enter 鍵。接著，在命令提示字元視窗中鍵入 pip install matplotlib，然後按 Enter 鍵。

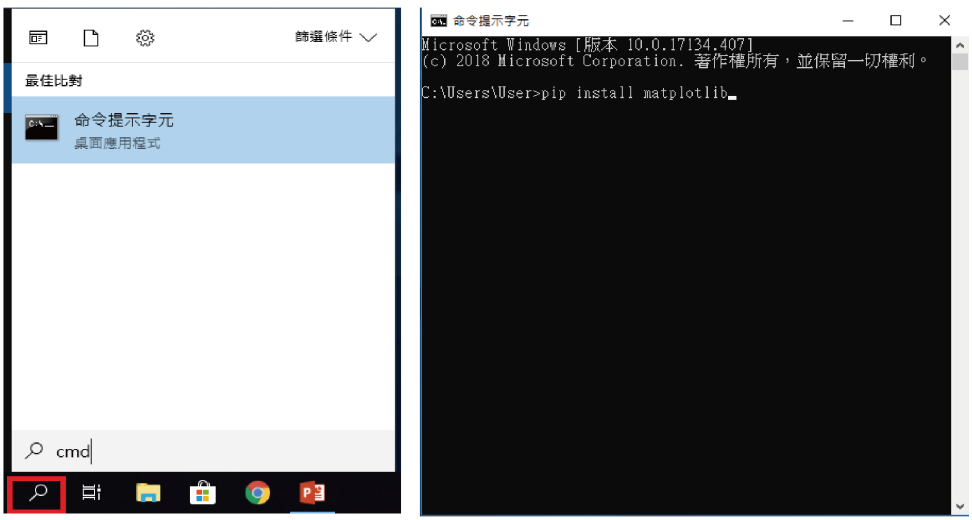


圖 5-1：安裝 matplotlib 套件

若成功安裝，會出現如圖 5-2 的畫面。

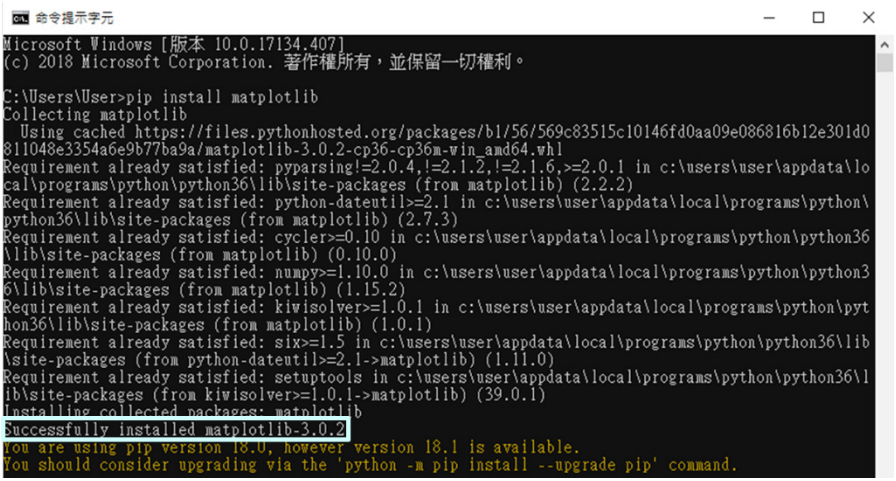


圖 5-2：matplotlib 套件安裝成功

準備工作：中文字顯示問題

matplotlib 套件預設狀態無法顯示中文字型，圖表上顯示中文字時會出現亂碼，因此需要額外進行設定。此部份流程較為繁瑣，建議讀者上網查詢符合己身系統的教學文章。

5-2 折線圖

立馬體驗：plot() 與 show()

請先安裝好 matplotlib 套件，我們來實際體驗一下如何繪製圖表。程式 E5-2-1 的第 **01** 行帶入 matplotlib 套件的 pyplot 模組，並以縮寫 plt 代表之。程式第 **02** 行呼叫 plt 模組中的 plot() 函式，傳入一個表示資料數值的列表。第 **03** 行呼叫 plt 模組中的 show() 函式，顯示繪製結果，如圖 5-3 所示。

E5-2-1.py:

```
01 import matplotlib.pyplot as plt
02 plt.plot([2, 0, 1, 8])
03 plt.show()
```

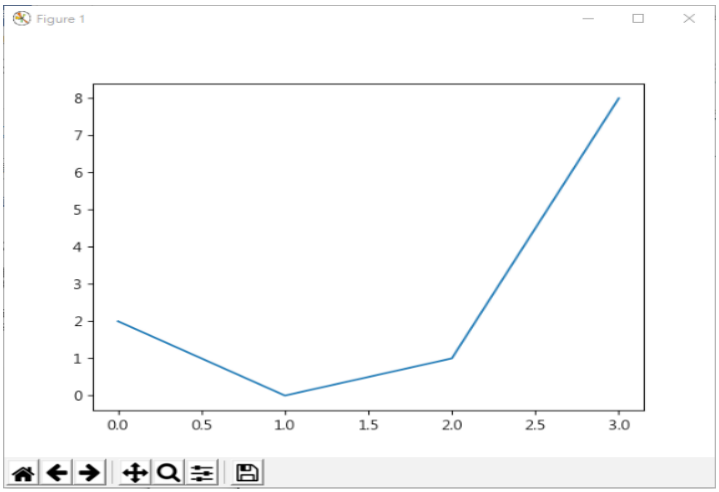


圖 5-3：程式 **E5-2-1.py** 之繪製結果

當我們只傳入一個列表時，這個列表代表的是 y 軸的資料值，而 x 軸預設為 0, 1, 2, …。因此圖 5-3 中的第一個資料點是 (0, 2)，第二個資料點是 (1, 0)，以此類推。如果我們只是想觀察一組有序資料的趨勢，只要三行程式碼就能輕鬆完成。

當我們同時擁有 x 和 y 軸的資料值時，則需分別將兩個列表傳入 plot() 函式，如程式 E5-2-2 所示。注意，若 x 和 y 的長度不同，會發生 ValueError 的錯誤。圖 5-4 為繪製結果，由四個資料點 (1, 2)、(3, 0)、(9, 1) 和 (10, 8) 構成折線。

E5-2-2.py：繪製一條折線

```
01  import matplotlib.pyplot as plt
02  x = [1, 3, 9, 10]
03  y = [2, 0, 1, 8]
04  plt.plot(x, y)
05  plt.show()
```

我們當然也可以在同一張圖上繪製多條折線，只需呼叫 plot() 數次，分別傳入資料即可，如程式 E5-2-3 所示。

E5-2-3.py：繪製兩條折線

```
01  import matplotlib.pyplot as plt
02  y1 = [2, 0, 1, 8]
03  y2 = [1, 3, 9, 10]
04  plt.plot(y1)
05  plt.plot(y2)
06  plt.show()
```

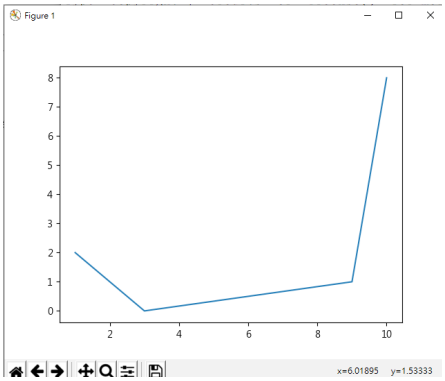


圖 5-4：**E5-2-2.py** 之繪製結果

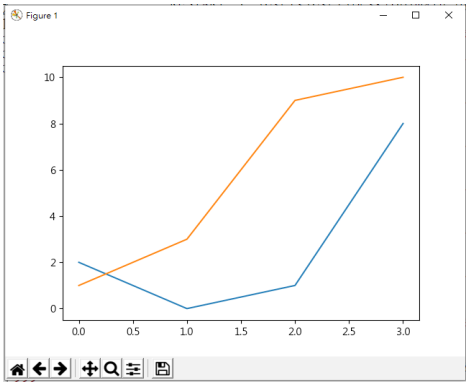


圖 5-5：**E5-2-3.py** 之繪製結果

plot() 的常用參數

plot() 函式具備有非常多樣的參數設定，讓程式員可以根據自己的需求，繪製出各種變化的圖表。我們先來看線條的顏色怎麼設定。plot() 函式繪製線條時有預設的線條色彩順序，也就是你不必指定線條的顏色，每次呼叫 plot() 函式時，Python 會自動給予線條的顏色。當然，我們也可以自行指定色彩，只需透過 color 參數來指定。表 5-2-1 說明三種指定 color 的方式：以色彩字串、紅綠藍三原色浮點數或紅綠藍三原色十六進位來表示，越接近 00 代表那一項的原色量越少，越接近 FF 代表那一項的原色量越大。

表 5-2-1：plot() 的線條色彩 (color) 參數

參數名稱	可用值
color	字串：blue、green、red、cyan、magenta、yellow、 black、white 三原色浮點數： (1.0, 0.2, 0.3) 三原色十六進位： #00FF00

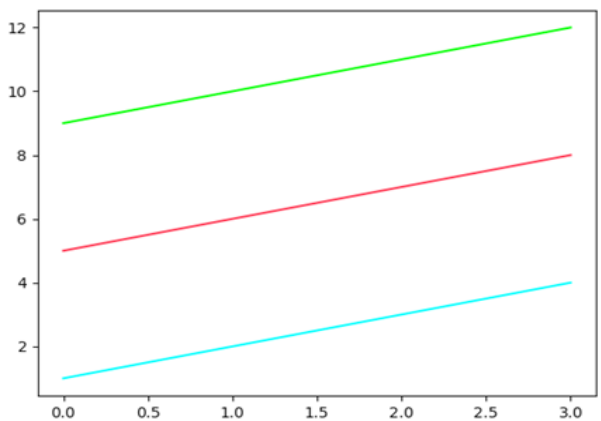


圖 5-6：程式 **E5-2-4.py** 之繪製結果

線條的樣式及寬度則分別以 linestyle 與 linewidth 表示。程式 E5-2-4 以不同的顏色、樣式和寬度組合繪製四條折線。

表 5-2-2：plot() 的線條樣式 (linestyle) 與寬度 (linewidth) 參數

參數名稱	可用值
linestyle	字串：-、--、-.、:
linewidth	通常在 1-20 之間。

E5-2-4.py：以四種顏色、樣式和寬度繪製四條折線

```
01 import matplotlib.pyplot as plt
02 plt.plot([13, 14, 15, 16], color='green', linestyle=':', linewidth=20)
03 plt.plot([9, 10, 11, 12], color='red', linestyle='-.', linewidth=10)
04 plt.plot([5, 6, 7, 8], color='blue', linestyle='--', linewidth=5)
05 plt.plot([1, 2, 3, 4])
```

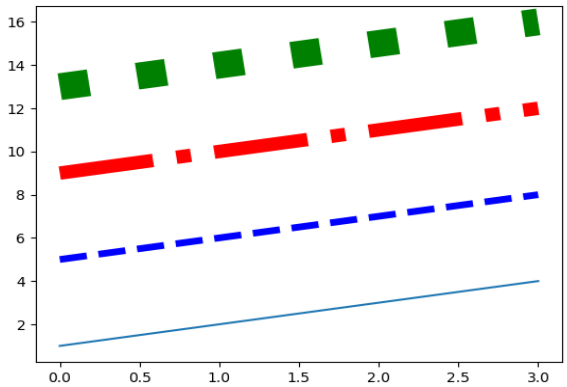


圖 5-7：程式 **E5-2-4.py** 之繪製結果

為了清楚標記資料點，我們可以指定 marker 和 markersize 兩個參數。

表 5-2-3：plot() 的資料點符號樣式 (marker) 及大小 (markersize) 參數

參數名稱	可用值
marker	字串如 .,ov^<>1234sp*hH+xDd|_ 中的單一字元
markersize	通常在 1-10 之間。

E5-2-5.py：以兩種資料點符號繪製兩條折線

```
01 import matplotlib.pyplot as plt
02 plt.plot([5, 6, 7, 8], color='red', marker='v', markersize = 5)
03 plt.plot([1, 2, 3, 4], color='blue', marker='o', markersize = 10)
04 plt.show()
```

E5-2-6.py：多種資料點符號圖示

```
01 import matplotlib.pyplot as plt
02 mstr='.,ov^<>1234sp*hH+xDd|_'
03 for i in range(len(mstr)):
04     plt.plot([i], marker=mstr[i], markersize=10)
05 plt.show()
```

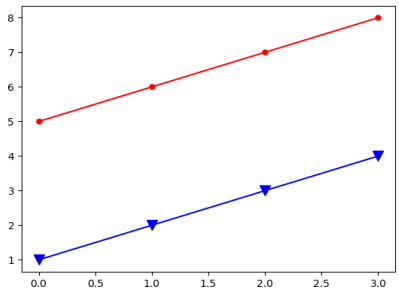


圖 5-8：**E5-2-5.py** 之繪製結果

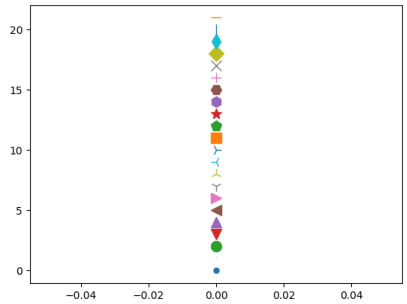


圖 5-9：**E5-2-6.py** 之繪製結果

程式 **E5-2-7** 示範了一種簡易的設定方法，可以將線條顏色、資料點符號以及線條樣式以一個字串傳入 `plot()`。程式第 02 行的字串 `'bo-'` 代表藍色 (`'b'`)、圓型符號 (`'o'`) 和實線 (`'-'`)。程式第 03 行的字串 `'--g'` 代表虛線 (`'--'`) 和綠色 (`'g'`)。

E5-2-7.py：以簡易字串繪製折線

```
01 import matplotlib.pyplot as plt
02 plt.plot([2, 0, 1, 8], 'bo-')
03 plt.plot([4, 5, 6], [1, 4, 9], '--g')
04 plt.show()
```

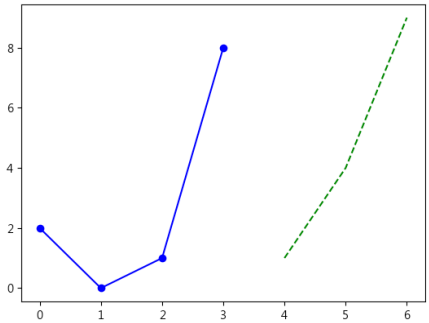


圖 5-10：E5-2-7.py 之繪製結果

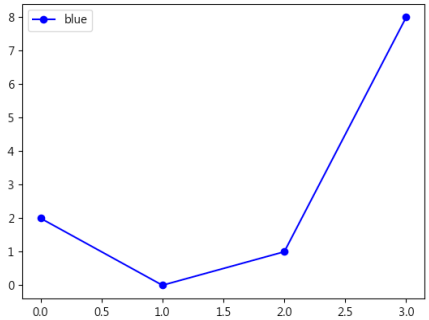


圖 5-11：E5-2-8.py 之繪製結果

最後介紹 `plot()` 函式的 `label` 參數，指定該參數可以設定線條的標籤。如程式 **E5-2-8** 的第 02 行，我們指定 `label` 為 `'blue'`，這使得該折線的標籤文字為 `blue`，如圖 5-11 左上角所示。注意，程式第 03 行需要呼叫 `legend()` 函式，否則標籤將不會顯示出來。

E5-2-8.py：為折線添加標籤文字

```
01 import matplotlib.pyplot as plt
02 plt.plot([2, 0, 1, 8], 'bo-', label='blue')
03 plt.legend()
04 plt.show()
```

5-3 圖表常用功能

設定標題文字

pyplot 模組有幾個函式可以用來設定圖表標題以及兩軸標題，如表 5-3-1 所示。這三個函式第一個參數代表標題文字，隨後附帶其它設定的參數，例如字型大小 (fontsize) 與文字位置 (loc) 等等。

表 5-3-1：設定標題文字的函式

函式名稱	說明	附帶參數
title()	圖表標題	fontsize 字型大小 loc 文字位置：left center right
xlabel()	X 軸標題	fontsize 字型大小
ylabel()	Y 軸標題	fontsize 字型大小

E5-3-1.py：設定圖表標題及軸標題文字

```
01 import matplotlib.pyplot as plt
02 plt.plot([2, 0, 1, 8], 'bo-')
03 plt.title('This is title', fontsize=24, loc='left')
04 plt.xlabel('x-axis caption', fontsize=15)
05 plt.ylabel('y-axis caption', fontsize=10)
06 plt.show()
```

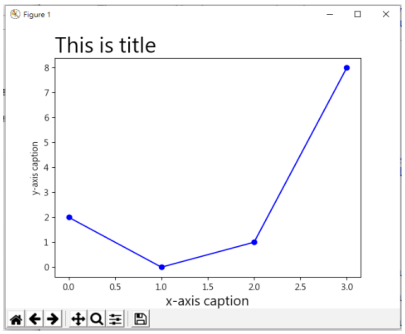


圖 5-12：程式 E5-3-1.py 之繪製結果

設定刻度範圍與刻度值

plot() 函式繪圖時會依資料數值的範圍自動設定刻度範圍和刻度值，如果想要自訂刻度的細節，可以使用表 5-3-2 的幾個函式來達成。xlim() 和 ylim() 函式用來設定刻度範圍，例如程式 E5-3-2 的第 03 行設定 x 軸的刻度範圍是 [-5, 5]，而第 04 行設定 y 軸的刻度範圍是 [0, 20]。xticks() 和 yticks() 函式用來設定刻度值，例如第 05 行設定 x 軸的刻度值只顯示了 -5、0、1、2 和 5，而第 06 行設定 y 軸的刻度值只顯示了 0、2、4、6 和 8。

表 5-3-2：設定刻度的函式

函式名稱	說明
xlim()	x 軸邊界，傳入最小和最大值。(反過來傳的話，圖也會反過來畫。)
ylim()	y 軸邊界，傳入最小和最大值。(反過來傳的話，圖也會反過來畫。)
xticks()	x 軸刻度，傳入 x 軸要顯示的數值。
yticks()	y 軸刻度，傳入 y 軸要顯示的數值。

E5-3-2.py：設定圖表刻度範圍與刻度值

```
01 import matplotlib.pyplot as plt
02 plt.plot([2, 0, 1, 8], 'bo-')
03 plt.xlim(-5, 5)
04 plt.ylim(0, 20)
05 plt.xticks([-5, 0, 1, 2, 5])
06 plt.yticks(range(0, 10, 2))
07 plt.show()
```

一般來說，傳入 xlim() 和 ylim() 兩個數值依序是最小值和最大值。如果傳入的第一個值比第二個值大，該軸的範圍就會由大到小。例如我們若把程式 E5-3-2 的第 03 行寫成 plt.xlim(5,-5)，繪製結果就會如圖 5-13b 所示。

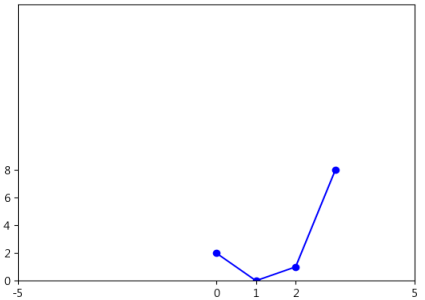


圖 5-13a：程式 5-3-2 之繪製結果

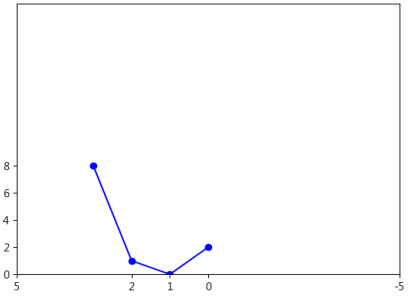


圖 5-13b：程式 5-3-2 之繪製結果 II

儲存圖檔

很多時候我們希望把繪製出來的圖表拿到其它地方使用，例如放入文書報告或簡報中，甚至於儲存下來以備未來使用。pyplot 模組中的 `savefig()` 函式這時就派上用場了。

程式 **E5-3-3** 示範了如何用程式繪製出 $y=x^0$、$y=x^1$、$y=x^2$ 和 $y=x^3$ 四個函式圖形並加以存檔。程式第 02 行有一個 `for` 句型，可以看出 `i` 的值會依序以 0、1、2 和 3 來執行以下第 03-08 行。第 03 行設定 `x` 是從 0 到 9 的整數，`y` 則是 `x` 中的每個值 `e` 作 `i` 次方的結果所構成的列表。第 06 行顯示圖表標題，第 07 行繪製圖表，而第 08 行就會將該圖表儲存成名為 0.png、1.png、2.png 和 3.png 的檔案。要特別注意的程式第 05 行呼叫了 `clf()` 函式。這個函式會清除目前圖表上的資料（`clf` 就是 clear figure 的縮寫）。如果省略了這一行，後續繪製的折線就會不斷疊加在前面折線上。（讀者可以自行嘗試看看。）

表 5-3-3：儲存圖檔的函式

函式名稱	說明
`savefig()`	傳入圖檔檔名，支援 png, pdf, eps, pgf, ps, raw, rgba, svg, svgz 格式。

E5-3-3.py：繪製四個函式並存檔

```
01  import matplotlib.pyplot as plt
02  for i in range(4):
03      x = range(10)
04      y = [e**i for e in x]
05      plt.clf()
06      plt.title('y = x**' + str(i))
07      plt.plot(x, y)
08      plt.savefig(str(i) + '.png')
```

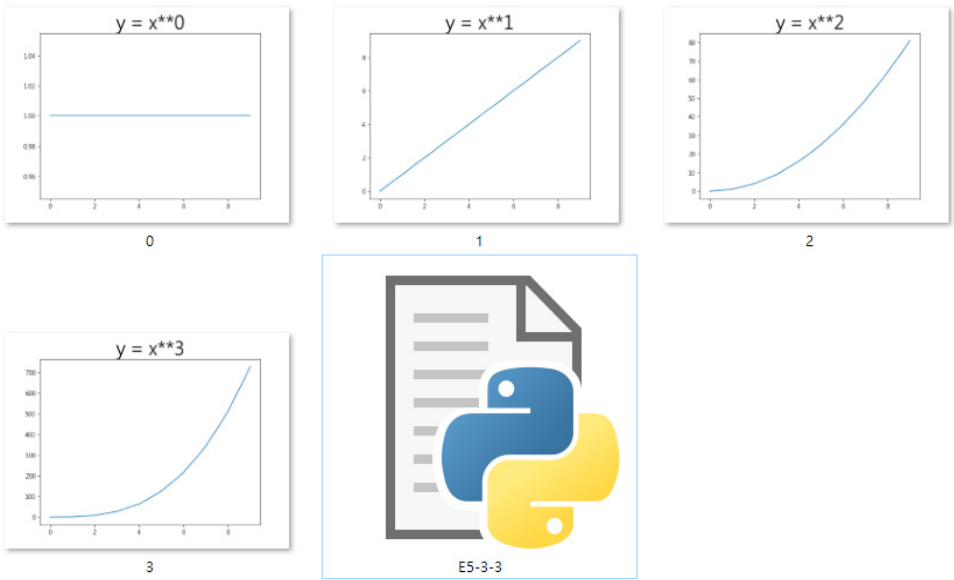


圖 5-14：程式 **E5-3-3.py** 之執行成果

執行程式 **E5-3-3** 後，我們會在程式檔的同一個目錄下看到四個圖檔，如圖 5-14 所示。學會存檔後，我們可以更方便地把圖表儲存下來，以作更廣泛的運用囉！

隨堂練習

5.3.1 請使用自行車竊盜資料檔，繪製如下的折線圖，用以觀察各行政區在不同時刻的竊案數變化。

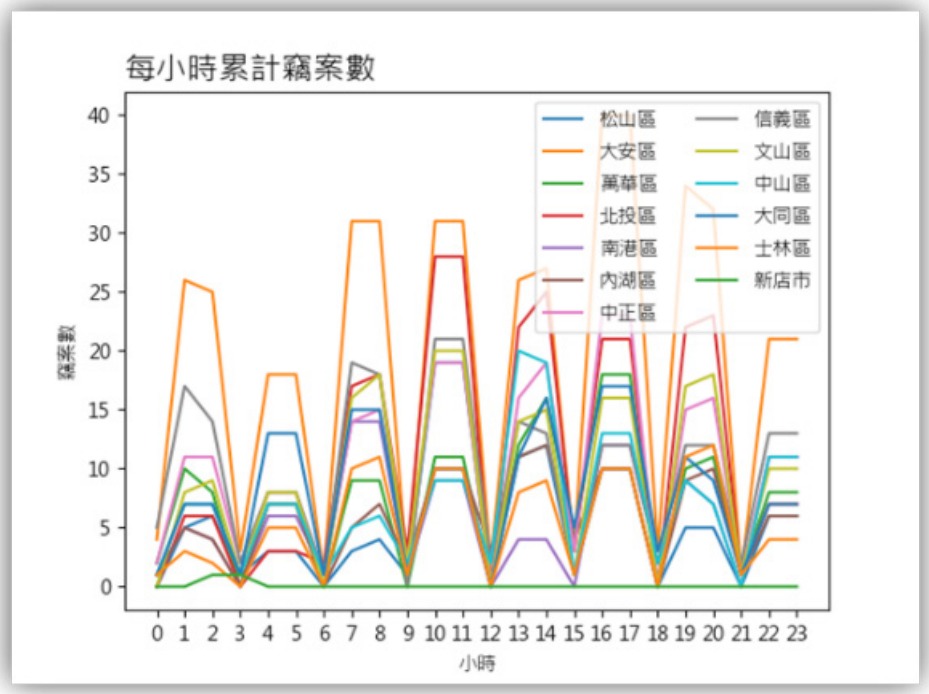


圖 5-15：自行車竊案之折線圖範例

5-4 散佈圖

折線圖常用來觀察有序數列的變化趨勢，而散佈圖可用於觀察兩個維度資料的關係。前一節我們學習了用 plot() 函式繪製折線圖，事實上，plot() 函式也可以繪製散佈圖，只要呼叫時指定資料點符號樣式但不要指定線條樣式就可以了。程式 **E5-4-1** 的第 04 行以綠色 (g) 倒三角形 (v) 繪製十個資料點 (0, 10)、(1, 11)、……、(9, 19)。第 05 行以藍色 (b) 和星號 (*) 繪製十個資料點 (10, 0)、(11, 1)、……、(19, 9)。

E5-4-1.py：繪製兩組資料形成的散佈圖

```
01 import matplotlib.pyplot as plt
02 x = range(10)
03 y = range(10, 20)
04 plt.plot(x, y, 'gv')
05 plt.plot(y, x, 'b*', markersize=20)
06 plt.show()
```

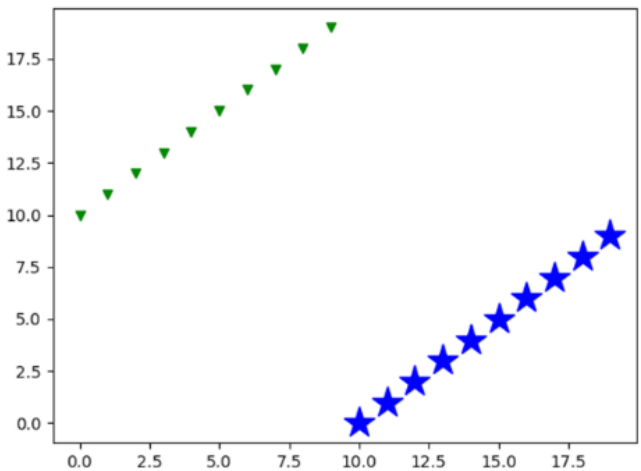


圖 5-16：程式 **E5-4-1.py** 繪製散佈圖之結果

散佈圖函式：scatter()

pyplot 模組中有一個專門用來繪製散佈圖的函式 scatter()。（散佈圖的英文即為 scatter plot。）呼叫 scatter() 時，基本需要傳入的是 x 和 y 軸的資料，可以是單值（如 scatter(2, 4)），也可以是一組值（如 scatter([1, 2], [3, 4])），x 和 y 的值個數需相同。表 5-4-1 列出此函式的常用參數，而程式 E5-4-2 示範了如何呼叫 scatter()。第 04 行設定資料點符號為菱形 ('d')，標籤文字為 'default'。未指定的符號相關屬性（如符號大小與顏色等等）都會有預設值。程式第 05 行設定符號大小 200，顏色為紅色，邊框寬度 3，邊框顏色為黑色，標籤文字為 'red'。圖 5-16 呈現程式 E5-4-2 的執行結果。

表 5-4-1：散佈圖函式 scatter() 的常用參數

參數名稱	可用值
x / y	資料點的 x / y 值
marker	字串 .,ov^<>1234sp*hH+xDd|_ 中的單一字元
s / c	符號大小 / 符號顏色
linewidths	符號邊框寬度
edgecolors	符號邊框顏色
label	符號標籤

E5-4-2.py：設定散佈圖的符號屬性

```
01 import matplotlib.pyplot as plt
02 x = range(10)
03 y = range(10, 20)
04 plt.scatter(x, y, marker='d', label='default')
05 plt.scatter(y, x, s=200, c='red', linewidths=3,
06             edgecolors='black', label='red')
07 plt.legend()
08 plt.show()
```

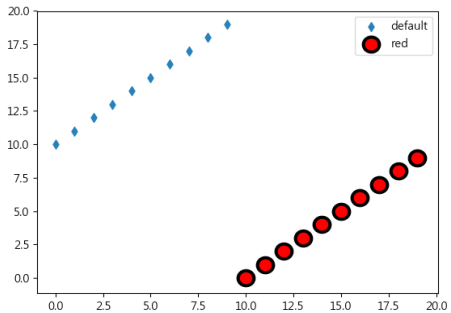


圖 5-17：程式 5-4-2 繪製之結果

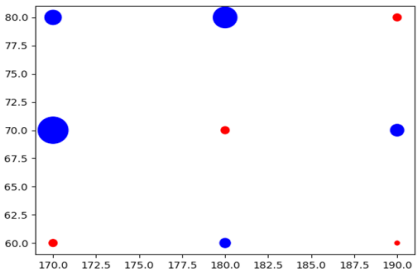


圖 5-18：程式 5-4-3 繪製之結果

在程式 E5-4-2 中我們使用了 s 和 c 參數來指定大小和顏色。作為一個專門的散佈圖繪製函式，scatter() 可以指定個別資料點符號的大小與顏色。只要我們傳入多個 s 和 c 的值，scatter() 便會以傳入的值來設定。例如 scatter([1, 2], [3, 4], c=['blue', 'red'], s=[10, 20]) 所繪製的兩個資料分別為座標 (1, 3)、大小為 10 的藍色符號以及座標 (2, 4)、大小為 20 的紅色符號。程式 E5-4-3 繪製九組 (身高，體重) 的人數的散佈圖。第 03 行的 h_w_c 的每一個元素代表身高（如 170）、體重（如 80）和人數（如 12）。程式第 08 行依人數決定符號大小，第 09 行設定超過 3 人的符號為藍色。

E5-4-3.py：繪製 (身高，體重) 的人數散佈圖

```
01 import matplotlib.pyplot as plt
02 h_w_c = [[170, 80, 12], [180, 80, 25], [190, 80, 3],
03           [170, 70, 40], [180, 70, 3], [190, 70, 8],
04           [170, 60, 3], [180, 60, 5], [190, 60, 1]]
05 x = [e[0] for e in h_w_c]       # 以身高作為 x 軸
06 y = [e[1] for e in h_w_c]       # 以體重作為 y 軸
07 s = [e[2]*20 for e in h_w_c]   # 以人數作為符號大小
08 c = ['blue' if e[2]>3 else 'red' for e in h_w_c ]
09 plt.scatter(x, y, s, c)
10 plt.show()
```

隨堂練習

5.4.1 請使用自行車竊盜資料檔，繪製如下的散佈圖，用以觀察竊案數在各月份及各時刻的分佈情形。

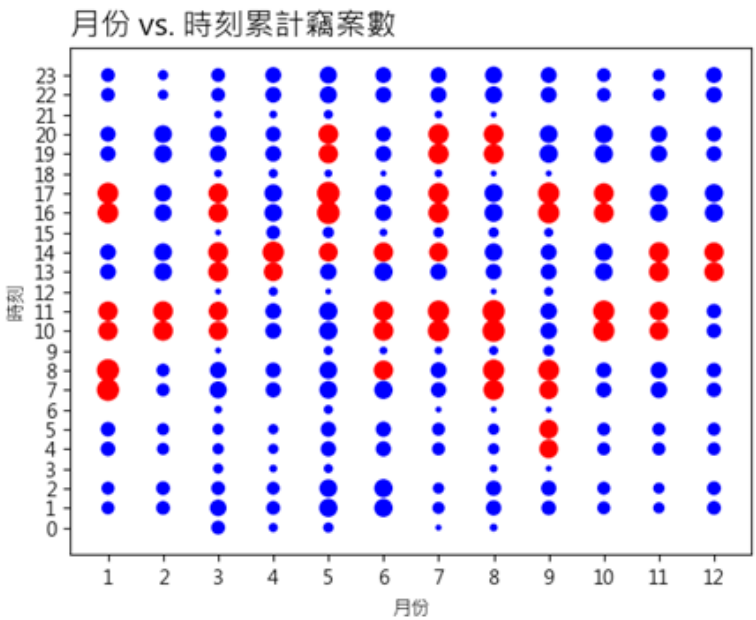


圖 5-19：自行車竊案之散佈圖範例

5-5 長條圖

本章最後一節要介紹的是長條圖，長條圖常用來比較多組資料的數值大小。

長條圖函式：bar()

呼叫 bar() 函式時，基本傳入的是 x 和 y 軸（y 在長條圖中特別稱為 height）的資料。此外，也可以指定長條的寬度與顏色。

表 5-5-1：長條圖函式 bar() 的常用參數

參數名稱	可用值與範例
x/height	x / y 軸座標值
width	長條寬度（預設 0.8）
color	長條顏色
label	標籤

程式 E5-5-1 呼叫了 bar() 函式兩次。第 02 行在 x 軸位置 0、2 和 4 處分別繪製高度為 10、15 和 8 的長條，標籤文字為 blue。第 03 行在 x 軸位置 1、3 和 5 處分別繪製高度為 7、12 和 11 的長條，寬度為 0.2，顏色為粉紅色 (pink)，標籤文字為 orange。圖 5-20 為此程式繪製之結果。

E5-5-1.py：繪製兩組資料形成的長條圖

```
01  import matplotlib.pyplot as plt
02  plt.bar([0, 2, 4], [10, 15, 8], label='blue')
03  plt.bar([1, 3, 5], [7, 12, 11], width=0.2,
04           color='pink', label='orange')
05  plt.legend()
06  plt.show()
```

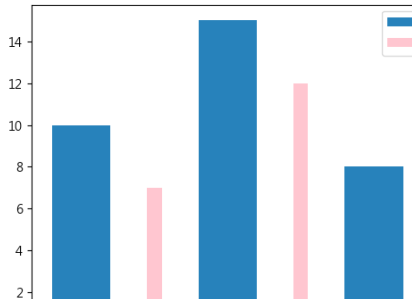


圖 5-20：程式 5-5-1 繪製之結果

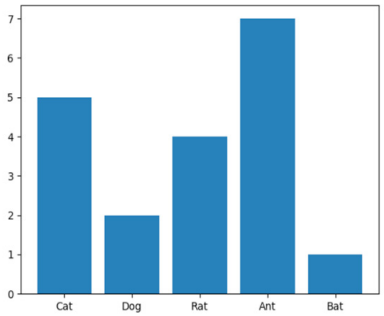


圖 5-21：程式 5-5-2 繪製之結果

在本章 5-3 節我們曾介紹 `xticks()` 可以用來指定 x 軸要顯示的刻度，例如 `xticks([1, 3, 5])` 代表 x 軸只會顯示刻度 1、3 和 5。在繪製長條圖時，x 軸經常需要顯示每個長條代表的意義（文字）而不是刻度的數值。呼叫 `xticks()` 時可以傳入第二組資料來代表在指定刻度顯示的文字，例如 `xticks([1, 3, 5], ['a', 'b', 'c'])` 會在刻度 1、3 和 5 處分別顯示文字 a、b 和 c。

程式 E5-5-2 中第 02 行為欲繪製成長條圖的資料，共有五組資料。第 04 行在 x 軸位置 0、1、2、3 和 4 處繪製高度為 5、2、4、7 和 1 的長條。第 05 行在相同的 x 軸位置顯示了 Cat、Dog、Rat、Ant 和 Bat 的文字。圖 5-21 為本程式繪製的結果。

E5-5-2.py：設定長條圖 **x** 軸的文字

```
01 import matplotlib.pyplot as plt
02 data = [['Cat', 5], ['Dog', 2], ['Rat', 4], ['Ant', 7], ['Bat', 1]]
03 plt.bar(range(len(data)), [e[1] for e in data])
04 plt.xticks(range(len(data)), [e[0] for e in data])
05 plt.show()
```

隨堂練習

5.5.1 請使用自行車竊盜資料檔，繪製如下的長條圖，用以比較各行政區的竊案數。

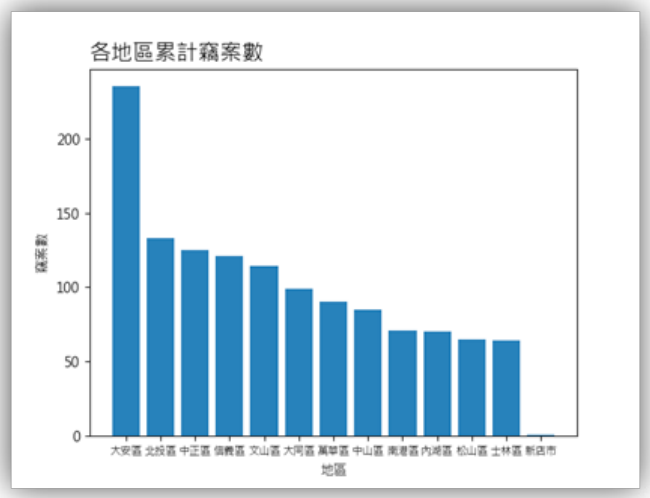


圖 5-22：自行車竊案之長條圖範例

Chapter 6

字典資料結構

/ 蔣宗哲、江政杰

第三章我們曾經學習過列表 (list) 資料結構，本章我們將講述第二種資料結構－字典 (dictionary) 和它的衍生結構 Counter，它可以建立兩個資料間的對應關係，常用於查詢和計數；舉例來說，可以記錄中英文的對照表以及計算一堆字詞的出現次數。雖然這些功能也能透過列表完成，不過字典在程式簡潔度和執行效率上都有其優勢。學習完本章後，讀者將能快速完成資料儲存、查詢、計數以及排序應用。

6-1 以字串作鍵值的查詢

回顧：列表的查詢

到目前為止，我們已經相當熟悉列表資料結構，它可以容納多個（不同型態）的資料（甚至是另一個列表），而且可以透過索引值來取用其中的資料。若有一個列表 x，從索引值 i 與列表資料 x[i] 的配對來看，我們可以說列表維護了一種整數值 0, 1, 2, ... 和資料的對應關係。

以程式 E6-1-1.py 為例，我們首先建立一個可存放 100 個資料的列表 scores。接著，詢問使用者全班人數，以 n 代表。程式第 03-04 行會讀入 n 個學生成績，以學生座號作為索引值，列表 scores 的 score[i] 代表第 i 號學生的成績。輸入完成後，第 05-09 行程式可以讓使用者輸入學生座號，然後查詢該生的成績。

E6-1-1.py: 以列表記錄座號與分數關係並查詢

```
01 scores = [0]*100
02 n = int(input('全班人數>'))
03 for i in range(n):
04     scores[i+1] = int(input('學生'+str(i+1)+'分數>'))
05 stu_id = int(input('你想知道幾號學生的分數>'))
06 if 1<=stu_id<=n:
07     print(scores[stu_id], '分')
08 else:
09     print('查無此人')
```

執行結果：

```
全班人數>5
學生1分數>100
學生2分數>90
學生3分數>80
學生4分數>95
學生5分數>88
你想知道幾號學生的分數>4
95 分
```

上述範例建立了學生座號與學生成績的對應關係，由於座號數字連續，所以把成績儲存在列表中以供後續查詢是一個不錯的作法。現在，讓我們來思考另一個查詢資料的例子。

假設我們想要以國際標準書號 ISBN 來查詢書名，若以列表來建立 ISBN 和書名的關係，雖然 ISBN 是一個整數值，可以作為索引值，但 ISBN 有 13 個數字，要建立一個可容納 10^{13} 個資料的列表會需要很大的記憶體空間；再者，圖書資料庫中不一定會存放所有的書籍資料，所以即使造出這麼大的列表，當中也可能有很多空間是沒有使用到而浪費的。

程式 E6-1-2.py 示範了以列表來完成 ISBN 查詢書名的作法，這裡我們假設只記錄 978986 開頭的書籍資料，所以只用了 ISBN 的後 7 碼。已知 9789863123798 的書號是對應到「Python 程式設計超入門」這本書，因此程式第 02 行以 books 列表的 books[3123798] 代表該書書名。

E6-1-2.py: 以列表記錄書號與書名關係並查詢

```
01  books = ['']*(10**7)
02  books[3123798] = 'Python 程式設計超入門'
03  books[4342549] = 'Python 程式設計實例入門'
04  isbn = int(input('輸入 ISBN >')) - 9789860000000
05  if books[isbn] != '':
06      print(books[isbn])
07  else:
08      print('查無此書')
```

執行結果：

```
輸入 ISBN >9789863123798
Python 程式設計超入門
```

如果我們要以書名反查 ISBN，以列表能作得到嗎？下面我們示範兩種可能的作法，兩種方法都以二維列表來存放書名及 ISBN。

程式 E6-1-3 將書名及 ISBN 兩兩存放在一個列表，然後再把這些兩個元素的列表存成一個更大的二維列表，如程式第 06-07 行所示。在函式 GetBook() 中，第 02 行依序檢查二維列表 books 中的每個一維列表 e，如果發現列表 e 的第一個元素（即書名資料）符合欲查詢的書名 name，即回傳該一維列表 e，此時程式第 11 行的條件式會判定為真，第 12 行便會列印出列表的第二個元素（即 ISBN）；如果 books 中沒有符合的，此函式的回傳值為 None，此時程式第 11 行的條件式會判斷為假，執行第 14 行，列印「查無此書」。

E6-1-3.py：書名反查 **ISBN**，方法一

```
01  def GetBook(books, name):
02      for e in books:
03          if e[0] == name:
04              return e
05
06  books = [['Python 程式設計超入門', 9789863123798],
07           ['Python 程式設計實例入門', 9789864342549]]
08
09  name = input('輸入書名>')
10  b = GetBook(books, name)
11  if b:
12      print(b[1])
13  else:
14      print('查無此書')
```

執行結果：

```
輸入書名>Python 程式設計實例入門
9789864342549
```

程式 E6-1-4 將所有書名存在一個一維列表，這些書籍的 ISBN 「依序」存在另一個一維列表，然後以這兩個列表形成一個二維列表，如第 06-07 行所示。函式 GetBook() 的第 03 行呼叫了列表的內建方法 index()，找出書名在 books 中的索引值並以 ind 表示之。第 04 行就回傳書名與書號。由於 books 的第一個元素是書名列表，第二個元素是書號列表，因此程式第 04 行以 books[1][ind] 從 books[1] 中取得和欲查詢書名 name 同索引值 ind 的書號。

E6-1-4.py：書名反查 **ISBN**，方法二

```
01  def GetBook(books, name):
02      if name in books[0]:
03          ind = books[0].index(name)
04          return name, books[1][ind]
05
06  books = [['Python 程式設計超入門', 'Python 程式設計實例入門'],
07           [9789863123798, 9789864342549]]
08
09  name = input('輸入書名 >')
10  b = GetBook(books, name)
11  if b:
12      print(b[1])
13  else:
14      print('查無此書')
```

執行結果：

```
輸入書名 >Python 程式設計實例入門
9789864342549
```

從前面的例子中，我們可以看到列表的侷限性，雖然列表在資料儲存時非常適合搭配迴圈的處理，但是還是會像上面書名反查書號的例子般碰到些許麻煩。本章將介紹 Python 的另一種利器：字典，讓我們在資料處理時更加方便。

6-2 字典 (dictionary) 資料結構

這一節裡我們來介紹一種新的資料結構，稱為 dictionary，中文是字典的意思。如同列表可以記錄整數索引值和資料的對應關係，字典也可以記錄一對鍵 (key) 與值 (value) 的對應關係，而且字典的鍵值不僅可以是整數，還可以是小數或字串。哇！那不就可以解決前一節我們遇到的以書名反查書號的問題了嗎，真是太好了！

在實際解決這個問題前，我們先以下列程式快速理解字典的用法。程式的第 01 行代表以 rec 作為一個空的字典。（還記得嗎？如果是一對中括號，那代表的是一個空的列表，請見程式 E3-3-1.py。）程式第 02-04 行分別建立起 'milk' 與 100、123 與 4、及 '臺灣' 與 'Taiwan' 的對應關係，其中每組關係的第一個資料（如 'milk'）稱為「鍵」，第二個（如 100）稱為「值」。在建立對應關係後，我們便可以用下標運算子取得「鍵」對應的「值」，如程式第 05 行會取得 100 然後列印出來。如同列表的索引值可以是變數，字典的鍵也可以是變數，如程式第 07-08 行所示。注意，字典中查無鍵值，會發生 KeyError 的錯誤。

E6-2-1.py: 字典的建立與查詢

```
01  rec = {}
02  rec['milk'] = 100
03  rec[123] = 4
04  rec['臺灣'] = 'Taiwan'
05  print(rec['milk'])
06  print(1+rec[123])
07  place = '臺灣'
08  print(rec[place])
```

執行結果：

```
100
5
Taiwan
```

來一個符合「字典」這個名稱的例子，程式 E6-2-2 實現一個「英翻中」的程式，其中第 01-04 行建立一個空字典，然後記錄三個英文單字與中文翻譯的對應關係。第 05 行令 query 代表使用者輸入的字串。第 06 行判斷字典中是否存在 query 代表的字串，如果有，便列印出對應的中文翻譯；否則，列印「未收錄此字」。

這個程式有兩個地方值得留意，第一是為了避免鍵不存在所引發的 KeyError，我們必須先確定鍵存在才使用下標運算子去取用對應的值；第二，字典和列表一樣，都可以用 in 查詢，不過列表是查詢某個值存不存在，但字典是查詢某個鍵存不存在。

E6-2-2.py: 簡易英翻中字典範例

```
01 mydict = {}
02 mydict['book'] = '書'
03 mydict['computer'] = '電腦'
04 mydict['program'] = '程式'
05 query = input('英翻中>')
06 if query in mydict:
07     print('解釋:', mydict[query])
08 else:
09     print('未收錄此字')
```

執行結果 1：	執行結果 2：
英翻中 >book 解釋： 書	英翻中 >csie 未收錄此字

更多字典的初始化與記錄方法

除了像前面兩個程式先建立空字典再以下標運算子逐一建立「鍵」-「值」關係外，底下我們再介紹幾種直接初始化字典的語法。讀者可以參考使用。其中方法 B 的程式片段很輕鬆地完成前一節中以書名反查書號的要求。

方法 **A**：以大括號初始化字典

```
01 eng = 'apple'
02 chinese = '蘋果'
03 mydict = {'book':'書', 'computer':'電腦', 'program':'程式',
04            eng: chinese}
05 print(mydict['book'])
```

方法 **B**：以 n×2 的列表初始化字典

```
01 books = [['Python 程式設計超入門', 9789863123798],
02          ['Python 程式設計實例入門', 9789864342549]]
03 mydict = dict(books)
04 print(mydict['Python 程式設計超入門'])
```

字典中每個鍵只能出現一次，對應一個值。記錄鍵值後，無法修改鍵的內容，但可以修改其對應值的內容。如程式 E6-2-3 在第 06 行把 'computer' 對應的中文翻譯由第 03 行設定的 '電腦' 改為 '計算機'。

E6-2-3.py：字典內的鍵 - 值關係可以更改

```
01 mydict = {}
02 mydict['book'] = '書'
03 mydict['computer'] = '電腦'
04 mydict['program'] = '程式'
05 query = input('英翻中>')
06 mydict['computer'] = '計算機'
07 if query in mydict:
08     print('解釋:', mydict[query])
09 else:
10     print('未收錄此字')
```

執行結果：

```
英翻中>computer
解釋: 計算機
```

當我們分別擁有鍵的列表與值的列表時，我們可以經由一個迴圈來建立它們兩兩的關係，如程式 E6-2-4 所示。程式第 01 行是數字一到九的中文國字，第 02 行是阿拉伯數字。第 04-05 行依序走訪這兩個列表，然後建立起一和 1、二和 2、三和 3，直到九和 9 的對應關係。程式第 06-08 行便可將使用者輸入的一串中文字轉變成阿拉伯數字。（注意：此處我們省略判斷鍵是否存在，以便讓程式較為精簡。）

E6-2-4.py：從中文轉換成阿拉伯數字

```
01  chinese = '一二三四五六七八九'
02  digit = '123456789'
03  mydict = {}
04  for i in range(len(chinese)):
05      mydict[ chinese[i] ] = digit[i]
06  query = input('輸入由一、二、..... 九的中文字 >')
07  for c in query:
08      print(mydict[c], end='')
```

執行結果：

```
輸入由一、二、..... 九的中文字 > 九九八一
9981
```

有個 zip() 函式，有興趣的讀者可以試試它有什麼妙用。

多維字典

如同列表中的內容可以是列表（因而形成多維列表），字典的值也可以是字典或是列表。程式 E6-2-5 示範了以列表作為字典的值（第 02、03 行），因此第 06 行的 scores['張三'] 會得到列表 [100, 90, 92]，而 scores['張三'][0] 便會得到 100。

E6-2-5.py：以列表作為字典的值

```
01  scores = {}
02  scores['張三'] = [100, 90, 92]
03  scores['李四'] = []
04  scores['李四'] += [85]
05  scores['李四'] += [75]
06  print(scores['張三'][0])
07  print(scores['李四'][1])
```

執行結果：

```
100
75
```

程式 E6-2-6 示範了以字典作為字典的值，第 02 行令 students['張三'] 是一個空字典，第 03-05 行將三組鍵值記錄到 students['張三'] 中。因此，如果第 06 行使用者輸入 '張三'，第 07 會判斷出 students 中有 '張三' 這個鍵，然後第 08 行會列印出 students['張三'] 對應的內容，也就是存有三組鍵值關係的字典。

E6-2-6.py：以字典作為字典的值

```
01 students = {}
02 students['張三'] = {}
03 students['張三']['出生地'] = '台北'
04 students['張三']['體重'] = 70
05 students['張三']['考試分數'] = [100, 80, 85]
06 stu = input('學生姓名>')
07 if stu in students:
08     print(students[stu])
```

執行結果：

```
學生姓名>張三
{'出生地': '台北', '體重': 70, '考試分數': [100, 80, 85]}
```

字典應用：計數

字典有個常見應用是計算每一個資料出現的次數，我們把想要計次的資料作為字典的鍵，把次數作為字典的值。程式 E6-2-7 是一個計票程式範例，第 01 行 votes 列表代表得票者的姓名，第 02 行建立空字典 counts。第 03-07 行走訪 votes 中所有姓名，若姓名首次出現，會在字典中建立該姓名對應的值為 1（代表得到 1 票）；若姓名已出現過，則會將該姓名對應的票數加 1。第 08-12 行則為查詢的程式碼。

E6-2-7.py：計票程式範例一

```
01 votes = ['張三', '李四', '張三', '張三', '李四']
02 counts = {}
03 for v in votes:
04     if v not in counts:
05         counts[v] = 1
06     else:
07         counts[v] = counts[v]+1
08 query = input('候選人姓名>')
09 if query in counts:
10     print('得票數:', counts[query])
11 else:
12     print('查無此人')
```

執行結果：

```
候選人姓名>張三
得票數: 3
```

程式 E6-2-7 需要第 04 行的判斷，否則若鍵值不存在，直接執行第 07 行會產生錯誤。如果覺得多加這個判斷很麻煩，字典型態有提供 get() 函式，可以設定鍵不存在時的回傳值。如下列程式碼所示。程式執行第 02 行時，mydict 尚無 '張三' 這個鍵，此時 get('張三', 0) 便會回傳 0。

```
01 mydict = {}
02 print(mydict.get('張三', 0))
03 mydict['張三'] = 4
04 print(mydict.get('張三', 0))
```

執行結果：

```
0
4
```

若以 get() 函式來改寫程式 E6-2-7，可精簡成為程式 E6-2-8。主要差異為程式 E6-2-7 的第 04-07 行縮短為程式 E6-2-8 的第 04 行。

E6-2-8.py：計票程式範例二

```
01 votes = ['張三', '李四', '張三', '張三', '李四']
02 counts = {}
03 for v in votes:
04     counts[v] = counts.get(v, 0) + 1
05 query = input('候選人姓名>')
06 if query in counts:
07     print('得票數:', counts[query])
08 else:
09     print('查無此人')
```

執行結果：

```
候選人姓名>張三
得票數: 3
```

隨堂練習

6.2.1　請嘗試以字典重寫 4.2.1 的隨堂練習

6-3 走訪字典內容

以字典來計數真的很方便，對吧！那我們該怎麼取出所有計數結果呢？之前我們只學到以一個特定的鍵來查詢其對應的值，本節我們來看看如何以 for 句型來走訪字典中的資料。（是不是覺得似曾相識？在學習列表時，一開始我們也學習以下標運算子取得列表中的單一筆資料，後來再學習以 for 句型走訪整個列表。）

E6-3-1.py：走訪字典

```
01 mydict = {'book':'書', 'computer':'電腦', 'program':'程式'}
02 for e in mydict:
03     print(1, e)
04 for e in mydict.keys():
05     print(2, e)
06 for e in mydict.items():
07     print(3, e)
08 for e in mydict.values():
09     print(4, e)
```

執行結果：

```
1 book
1 computer
1 program
2 book
2 computer
2 program
3 ('book', '書')
3 ('computer', '電腦')
3 ('program', '程式')
4 書
4 電腦
4 程式
```

程式 E6-3-1 有四個 for 句子，其中第 02 行 in mydict 和第 04 行 in mydict.keys() 的意義相同，都是走訪字典中的所有鍵。第 06 行的 mydict.items() 代表同時取出 (鍵 , 值)，而第 08 行的 mydict.values() 則表示僅取出值。程式在列印時加上數字 1、2、3 和 4 是為了讓讀者更容易看出程式碼與列印結果的對應關係。

學會走訪後，我們將計票程式 E6-2-8 修改如 E6-3-2，該程式在計票後會列印出每位候選人的得票數。

E6-3-2.py：計票並列印所有結果

```
01 votes = ['張三', '李四', '張三', '張三', '李四']
02 counts = {}
03 for v in votes:
04     counts[v] = counts.get(v, 0) + 1
05 for e in counts:
06     print(e, '得了', counts[e], '票')
```

執行結果：

```
張三 得了 3 票
李四 得了 2 票
```

以 for 句型走訪字典，走訪順序如同字典新增鍵值的順序。以程式 E6-3-1 為例，鍵的順序為 'book'、'computer'、'program'，以程式 E6-3-2 為例，鍵的順序為 '張三'、'李四'。以下兩個程式將示範如何以鍵或值的大小關係排序後走訪。程式 E6-3-3 的第 04 行以 keys() 取出字典 books 中所有的鍵，然後以 sorted() 函式排序（預設由小到大），最後以 for 句型走訪排序後的鍵。因此，原本第 01-03 行建立字典時，三本書並未依書號排序（9789864342549 在 9789863123798 後但在 9789862487488 前），但經過第 04 行的處理再走訪，第 05 行的列印結果已由小到大排序。

E6-3-3.py：以鍵排序後走訪字典

```
01 books = {9789863123798: 'Python 程式設計超入門',
02          9789864342549: 'Python 程式設計實例入門',
03          9789862487488: '書呆與阿宅'}
04 for e in sorted(books.keys()):
05     print(e, books[e])
```

執行結果：

```
9789862487488 書呆與阿宅
9789863123798 Python 程式設計超入門
9789864342549 Python 程式設計實例入門
```

以值排序走訪比較複雜一點，我們雖然能以 values() 函式取出字典中所有的值，但我們沒辦法從值反查鍵。因此，我們的作法是以 items() 同時取出鍵 - 值對，然後自訂函式來訂定比較欄位為值，再以 sorted() 函式來排序。

程式 E6-3-4 中，第 07 行先取出 counts.items()，在程式 E6-3-1 中，我們已看到這會是所有鍵 - 值的資料，每一對鍵 - 值以元組的方式表示，[0] 為鍵而 [1] 為值。以 sorted() 函式對 counts.items() 排序時，每次會比較兩對鍵 - 值，預設的情況會先比鍵再比值。因為此處我們要以值來比較大小，我們另訂一個函式 cmp()，該函式傳回傳入物 x 的第二個資料 x[1]，也就是鍵 - 值中的值。在 sorted() 中指定 reverse=True 代表由大到小排序。

E6-3-4.py：以值排序後走訪字典

```
01 votes = ['王五', '張三', '李四', '張三', '張三', '李四']
02 counts = {}
03 for v in votes:
04     counts[v] = counts.get(v, 0) + 1
05 def cmp(x):
06     return x[1]
07 for e in sorted(counts.items(), key=cmp,
08 reverse=True):
       print(e[0], '得了', e[1], '票')
```

執行結果：

```
張三 得了 3 票
李四 得了 2 票
王五 得了 1 票
```

程式的結果會以票數由高到低列印出來。如果不特別處理，如下列程式碼，列印的結果會依照新增鍵的順序。

```
votes = ['王五', '張三', '李四', '張三', '張三', '李四']
counts = {}
for v in votes:
    counts[v] = counts.get(v, 0) + 1
for e in counts:
    print(e, '得了', counts[e], '票')
```

執行結果：

```
王五 得了 1 票
張三 得了 3 票
李四 得了 2 票
```

大家可以發現，列表與字典的使用，有點像又有點不一樣。建議大家可以多熟悉列表與字典的操作，搭配迴圈的運作，可以解決非常多樣的困難問題。

6-4 Counter 資料結構

說到計數，字典已經夠方便了，不過，還有更方便的！登登！讓我們來看看 Counter。Counter 是一種特別的字典，它可以拿需計數的資料來初始化，並在初始化後就得到每個資料的次數。對於不存在的鍵，它有預設值 0；它可以取出次數最高的項目，而且它還支援計數器的運算。

使用前，我們需從 collections 模組中匯入 Counter 型態，如下列程式第 01 行所示。第 02 行我們以字串 'abcdabcaa' 初始化 Counter，並以 c 指稱之。從第 03 行列印的結果，我們可以看到 c 已經算好每個字母的次數。Counter 也是一個字典，因此我們可以像使用字典一樣，以鍵查值，如第 04 行所示。

```
01 from collections import Counter
02 c = Counter('abcdabcaa')
03 print(c)
04 print(c['a'])
```

執行結果：

```
Counter({'a': 4, 'b': 2, 'c': 2, 'd': 1})
4
```

運用 Counter，我們可以把程式 E6-3-2 改寫成 E6-4-1。

E6-4-1.py：計票並列印所有結果：以 **Counter** 實作

```
01 from collections import Counter
02 votes = ['張三', '李四', '張三', '張三', '李四']
03 counts = Counter(votes)
04 print(counts)
05 for e in counts:
06     print(e, '得了', counts[e], '票')
```

執行結果：

```
Counter({'張三': 3, '李四': 2})
張三 得了 3 票
李四 得了 2 票
```

當然，我們也可以使用字典原有的初始化語法來初始化 Counter。

```
01 from collections import Counter
02 c = Counter({'張三':3, '李四':2})
03 print(c)
04 print(c['張三'])
```

執行結果：

```
Counter({'張三': 3, '李四': 2})
3
```

下列程式左邊使用 Counter，右邊使用字典。當鍵不存在時，Counter 會傳回 0 值，但字典會造成 KeyError。

```
01 from collections import Counter
02 c = Counter()
03 print(c['a'])
```

執行結果：

```
0
```

```
01 d = dict()
02 print(d['a'])
```

執行結果：

```
KeyError: 'a'
```

Counter 有個 most_common(n) 函式可以傳回次數最高的前 n 個鍵 - 值對，如程式 E6-4-2 所示。若次數相同，會依建立的鍵順序。若未傳入 n 值或 n 值過大，會傳回所有的鍵 - 值對。

E6-4-2.py：Counter 的 most_common() 函式

```
01 from collections import Counter
02 c = Counter('acbcdabcaab')
03 print(c.most_common(1))
04 print(c.most_common(2))
05 print(c.most_common())
```

執行結果：

```
[('a', 4)]
[('a', 4), ('c', 3)]
[('a', 4), ('c', 3), ('b', 3), ('d', 1)]
```

Counter 還可以支援一些運算，如表 6-4-1 所示。程式 E6-4-3 可以更清楚了解運算的結果。第 02 和 03 行建立兩個 Counter，a 和 b，並在第 04 和 05 行列印其內容。第 06 行列印相加的結果，舉例來說，鍵 'c' 對應的次數為 3+4=7。第 07 行列印相減的結果，舉例來說，鍵 'c' 對應的次數為 3–4=-1，

因為次數不為正，所以就不再有 'c' 這個鍵了。第 08 行列印交集的結果，鍵 'c' 對應的次數為 3 和 4 中的較小值，因此變成了 3。第 09 行列印聯集的結果，鍵 'c' 對應的次數為 3 和 4 中的較大值，因此變成了 4。

表 6-4-1: Counter 的運算

運算	運算式	運算結果
加法	x+y	將次數相加。
減法	x-y	x 次數減 y 的次數，刪除次數非正的鍵值。
交集	x&y	取兩者次數的較小值。
聯集	x|y	取兩者次數的較大值。

E6-4-3.py：Counter 的常見運算

```
01 from collections import Counter
02 x = Counter('abccdabcaab')
03 y = Counter('abccccd')
04 print('  x', x)
05 print('  y', y)
06 print('x+y', x+y)
07 print('x-y', x-y)
08 print('x&y', x&y)
09 print('x|y', x|y)
```

執行結果：

```
  x Counter({'a': 4, 'b': 3, 'c': 3, 'd': 1})
  y Counter({'c': 4, 'a': 1, 'b': 1, 'd': 1})
x+y Counter({'c': 7, 'a': 5, 'b': 4, 'd': 2})
x-y Counter({'a': 3, 'b': 2})
x&y Counter({'c': 3, 'a': 1, 'b': 1, 'd': 1})
x|y Counter({'a': 4, 'c': 4, 'b': 3, 'd': 1})
```

Chapter 7

數據與地圖資料視覺化概論

/ 蔡天怡

資料視覺化是資料分析過程中十分重要的環節，透過視覺化圖表呈現分析結果，可以簡潔清晰的表達資料中蘊藏的意義與趨勢。本章介紹資料視覺化的基本概念及意涵，並介紹 Tableau 視覺化工具。本章共有四個小節，包括：資料視覺化的基本概念與原則、資料視覺化工具選介與準備工作、以政府開放統計資料談數據資料視覺化，以及地圖資料視覺化。學習完本章後，讀者將了解資料視覺化的重要原則，並能以 Tableau 工具繪製長條圖與地理圖表。

7-1 資料視覺化的基本概念與原則

什麼是資料視覺化？

你可曾在閱讀充滿統計數據的文字或欄位複雜的表格時感到暈頭轉向？對於資訊超載的情況感到認知無法負荷，甚至感到焦慮？若你曾有上述感受，資料視覺化或許有助化解上述情形。

「資料視覺化」是用視覺呈現的方式為資料說出它的故事。顧名思義，這是種將資料轉化成視覺形式的歷程，即透過圖像和／或顏色等視覺要素來呈現資料，而其主要目的在透過圖像傳遞資訊，以達成更有效而精準的溝通。

將資料轉化為視覺呈現的過程中，勢必包含資料的處理與分析，方得有效呈現其內涵。因此，資料視覺化與資料分析密不可分，這包含了人文社會科學理解資料的各種分析方式及許多領域皆常使用的統計分析；如何透過適當的工具將資料批次清理與轉換，則與電腦科學息息相關，而其視覺呈現中的圖像規劃更與藝術設計等領域相關。也就是說，資料視覺化的相關領域可能橫跨了人文藝術、社會科學與理工等學門，是相當跨領域的學問。

根據 Stephen Few（2012），資料視覺化的重要性可從視覺是人類非常重要的感官談起，人們非常仰賴視覺感知解讀所見事物。一般說來，人們平時感知週遭事物時，大約 70% 來自視覺；30% 來自其他感官。因此，掌握視覺線索（visual cues）如何影響感知是相當重要的。

圖 7-1 是以《100 Things Every Designer Needs to Know About People》書中實例為基礎所作的。根據該書作者 Susan Weinschenk，在沒有特殊提示的情況下，一般人閱讀圖中左右兩側的文字時，會依照顏色此一視覺線索所引導的方向來理解詞彙的意涵。因此，絕大多數的人會順著顏色指示的方向，將左圖讀作「STOP TALK」、「QUIET NOW」；將右圖讀作「STOP QUIET」、「TALK

NOW」。你是否也受到視覺線索的影響了呢？本節在介紹過資料視覺化的功能與原則後，將探討資料視覺化原則並進一步說明視覺線索的運用。

STOP	TALK
QUIET	NOW

STOP	TALK
QUIET	NOW

圖 7-1 視覺線索可能影響閱讀方向

資料視覺化的主要功能

資料視覺化的核心價值包括：幫助人們思考和溝通、使人得以更有效地了解問題、更清楚而精確地說故事。藉由提供更多資訊，以克服人們認知記憶能力上的限制，以協助我們發掘未曾注意的資訊、協助找出各種關係、協助解決問題。

理想的分析與呈現方式使人得以作更多有意義的比較、更大量的比較（greater quantity）、更多面向的比較（more dimensions），以及更多觀點的比較（multiple perspectives）。

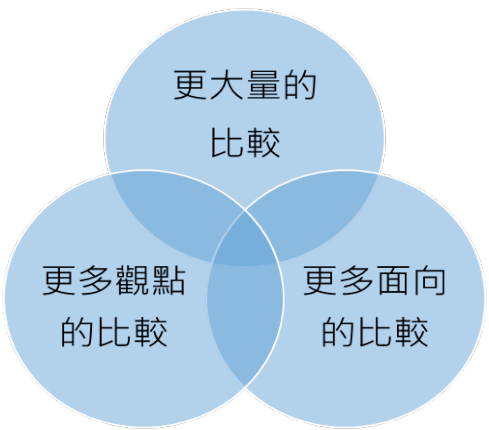


圖 7-2 資料視覺化有助達成理想的資料分析與呈現

其應用層面亦非常廣泛，從各種科學資料視覺化到新聞資料視覺化等等，從不同的學術領域到日常生活，舉凡涉及資料分析與處理的範疇，皆可應用資料視覺化。

資料視覺化的基本原則

視覺呈現的故事邏輯安排與文章的段落安排類似，行文往往須依特定邏輯與架構安排，讓文章容易閱讀；視覺呈現同樣須有其敘事的故事邏輯，讓讀者或受眾容易理解這些圖像所欲傳達的意涵，也才能發揮資料視覺化之功能，達到有效溝通的目標。

至於視覺呈現的故事邏輯如何安排？為了幫助合理地理解與思考，視覺呈現通常會先提供概覽（overview），再進一步提供局部和篩選過的資訊（zoom and filter）。最後，如果是載體可提供互動設計（如：網站上的視覺呈現），則可依閱聽人之需求提供細節（details-on-demand）。

這個安排與報導或學術文章撰寫的段落安排相似，尤其在學術寫作中，研究結果的呈現通常也由研究發現的概況展開（如：整體分布情形、各項描述性統計結果），再進一步依照研究問題所關注的個別議題，針對相關細節作進一步對照與討論等。最後，若有特別值得討論卻未必落在原先預設研究問題之重要發現，也會在此進一步說明。這種從整體概況到具體細節（from general to specific）的脈絡安排，是資料視覺化在規劃圖像順序與關係的重要觀念，也是讓圖文架構一致又簡單易懂的好作法。

就視覺呈現的設計而言，Edward Tufte 在《Beautiful Evidence》一書中提及資料視覺化的基本原則，以下簡述各項原則的要點：

1. 原則一：呈現比較關係

將資料視覺化時，記得不斷詢問自己：「這可以和什麼作比較？」由於理想的資料視覺呈現旨在協助讀者「消化」整理資訊，使圖像豐含大量、多面向、多觀點的資訊，因此，盡可能呈現多重的比較關係，方能充分發揮資料視覺化之效益。

2. 原則二：呈現因果關係

在相關理論或文獻的基礎上，可盡量嘗試將可能相關或具因果關係的變項

納入圖中，讓視覺化成果得以用來解釋關聯，更系統性地了解事物的架構，深化視覺呈現成果。因此，領域先備知識的掌握對於作圖亦相當重要。若能根據相關文獻與理論提出假設，或基於自己的相關知識，則更能掌握哪些欄位變項放入圖中，可能有助釐清因果關係。

3. 原則三：呈現多變量的資料

由於現實世界往往是多變量的（兩個變項以上），作圖時，一般希望盡可能納入豐富的相關資訊，以呈現事物較全面的樣貌。再者，資料視覺化旨在協助讀者「消化」大量而多樣龐雜的資訊，因此，在有限的空間中，設法以簡潔清楚的方式呈現較多變量的資料，是理想的視覺呈現之重要原則之一。

4. 原則四：整合證據

Edward Tufte 也提醒在進行資料視覺化時，切勿過度仰賴工具進行資料分析，整合文字、數字、圖表等各種呈現方式，才能發揮資料視覺化的最大效益。有些時候加上簡單的註解說明，仍然很重要。在各種視覺化工具如此便利的環境下，我們的確更應審慎看待軟體所提供之功能，並加以彈性運用，千萬別讓軟體限縮了我們的視覺呈現方式。

談到整合證據的作法，有一著名的視覺化案例可參考：

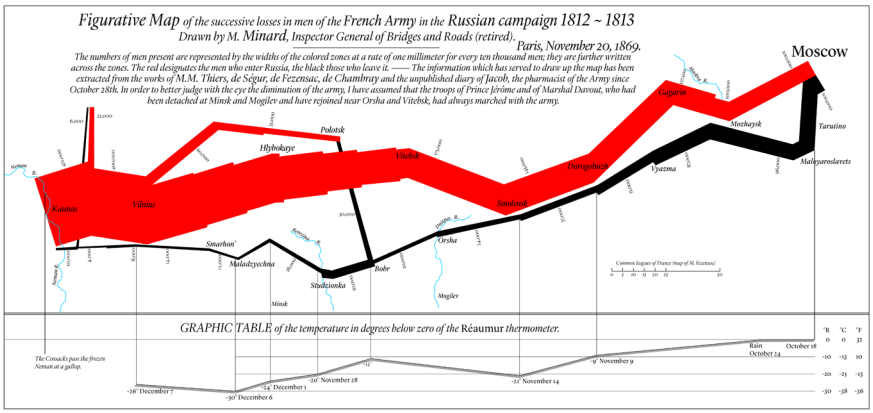


圖 7-3 Charles Minard's depiction of Napoleon's ill-fated march on Moscow

資料來源：https://en.wikipedia.org/wiki/File:Minard_Update.png（This file is licensed under the Creative Commons Attribution-Share Alike 4.0 International license）

該圖描繪拿破崙 1812 年征俄失敗的情形，結合了豐富的資訊（即「證據」）與呈現方式，將多方資訊，如：軍隊人數（傷亡情形）、地理位置、行軍距離與方向、氣溫、日期等，全數整合在二維視覺空間中。

5. 原則五：以合適的文字、尺度、資料來源等描述並紀錄各項證據

視覺化的呈現必須如實地說出完整的故事，切勿扭曲事實或刻意操弄。合適的說明與資料來源等描述有助協助理解視覺呈現，例如：圖例說明是相當重要的元素。而合適的尺度更是如實呈現資料的重要關鍵。若以不合適的尺度呈現長條圖，可能讓數值相近的類別看似差異甚大，或讓數值差異甚大的類別看似所差無幾，都可能有操弄視覺成果之嫌。例如：圖 7-4 以相同數據作圖，但在不同尺度下，左圖看似餐廳 A 和 B 的評價落差甚大，而右圖看似餐廳 A 和 B 的評價相近。

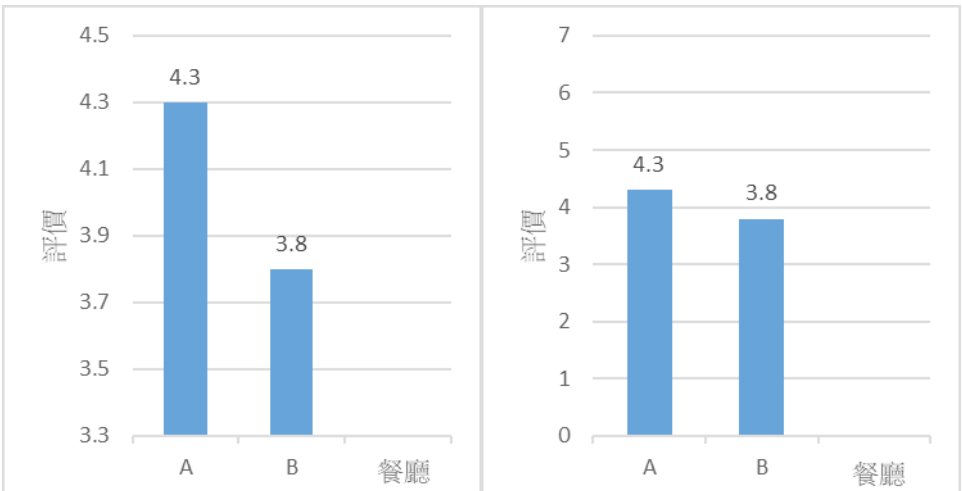


圖 7-4 不合適的尺度將操弄視覺化成果

6. 原則六：內容至上

資料分析與呈現之成敗最終仰賴資訊內容的品質、相關程度及完整性。即使視覺成果做得再華麗、再精美，倘若其內容空洞、不完整、品質不佳或缺乏連貫性，該資料視覺化也稱不上成功之作。一切還是得回歸根本的內容品質控管，才能做出好的視覺成果。

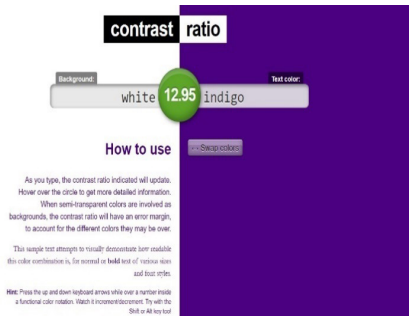
視覺線索（visual cues）的運用

除了前述原則，掌握視覺線索（visual cues）也是相當重要的。在資料視覺化的範疇中，所謂的「視覺線索」是指人們用以理解圖像呈現的要素。根據 Nathan Yau 在《Data Points》書中的，常見的視覺線索包括：位置、長度、角度、方向、面積、體積、顏色、色調等。一般說來，人們通常只能較精確地感知某些視覺線索（如：位置、長度、角度、方向），而較無法精確感知其他類型的視覺線索（如：面積／體積、顏色／色調）。因此，將資料視覺化時，我們通常要盡量以人們能較精確感知的視覺線索為主要「提示」，避免單獨使用人們較難精確感知的視覺線索。

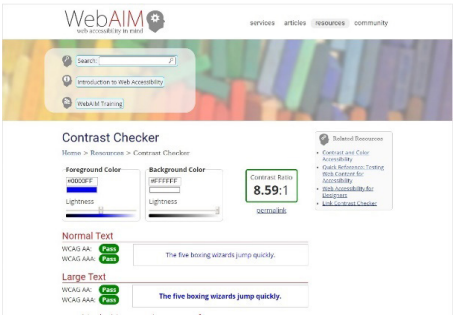
此觀念與「選擇合適的呈現方式」一節之內容可對照理解。許多時候，我們在選擇合適的視覺呈現時，會設法避免讓所選的圖像必須單獨仰賴唯一一種視覺線索，尤其是前述人們難以精確感知的線索。舉例而言，圓餅圖的主要視覺線索仰賴角度，但加上顏色或色調，可提供多重線索；泡泡圖的主要視覺線索是面積，但許多時候也會加上色調，提供多重線索。前述兩例中，圓餅圖較為單純，且其所仰賴的視覺線索是人們較易精確感知的角度，因此，即使以黑白或單色呈現，去除色彩之線索，對於人們理解此圖像的影響不是太大。相形之下，泡泡圖所仰賴的面積與色調都是人們不易精確感知的線索，繪製這類圖像時，提供多重線索益形重要。

以下將進一步探討人們無法精確感知的視覺線索，並簡要說明運用這些線索時特別須注意之事項。

色彩的運用是視覺線索運用中，非常重要的一環。姑且不論人們較難精確感知顏色，根據線上教育百科，男性紅綠色盲人口約佔全人口的 8% 、女性約佔 0.5%，且人們對色彩的感知與其所在文化密切相關。因此，我們在規劃視覺呈現時，須慎選顏色。以下選色與配色工具，可供參考：

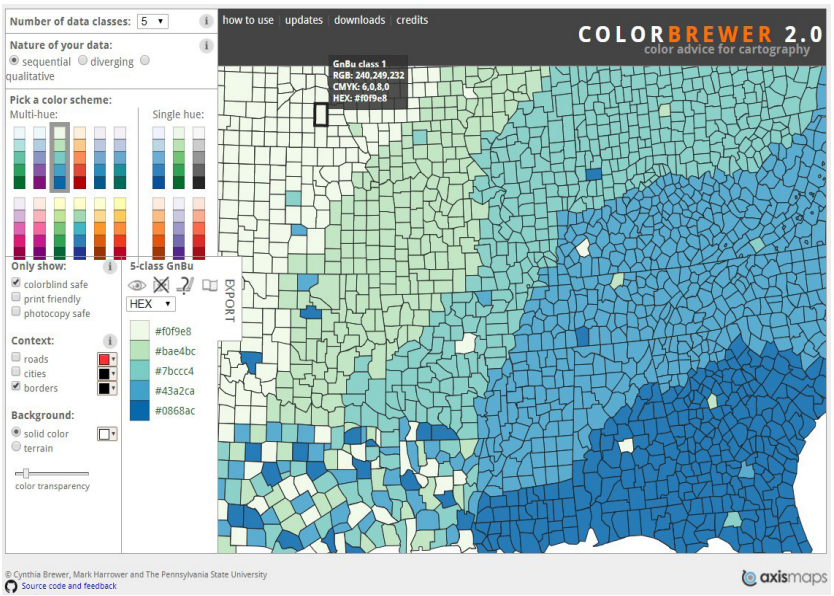


Contract ratio
https://contrast-ratio.com/

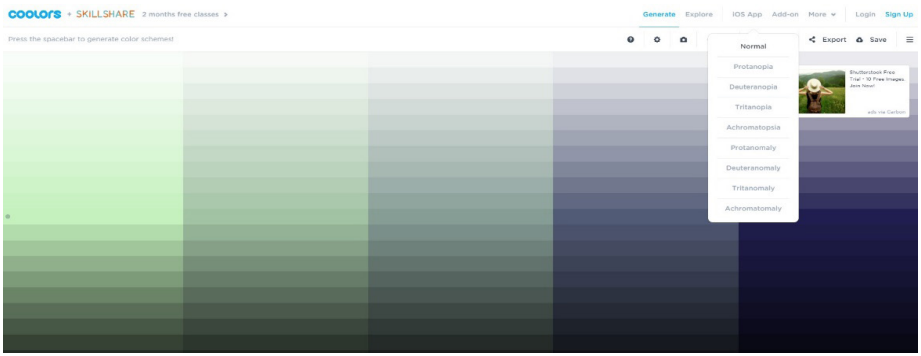


Color Contrast Checker by WebAIM
https://webaim.org/resources/contrastchecker/

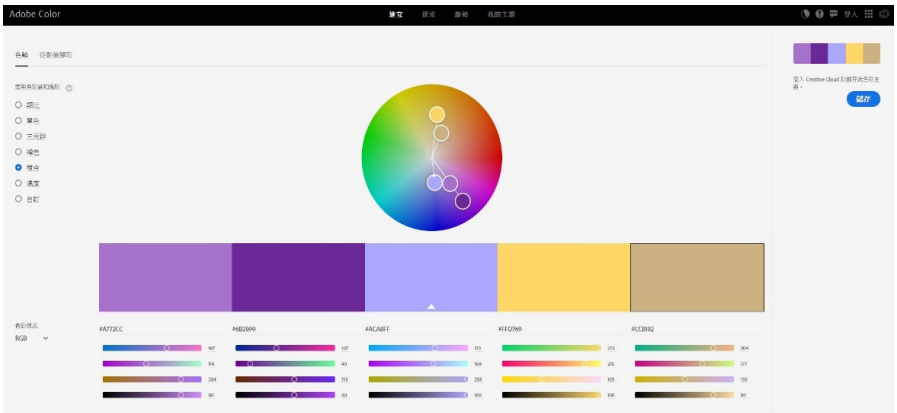
圖 7-5 對比色檢查工具舉隅



Color Brewer https://colorbrewer2.org/



Coolors https://coolors.co/



Adobe Color CC
https://color.adobe.com/

圖 7-6 選色工具舉隅

透過類似 Color Blind Awareness（http://www.colourblindawareness.org/）這類網站的模擬（如：http://www.colourblindawareness.org/colour-blindness/colour-blindness-experience-it/），我們可了解不同類型的色盲人士所見的色彩與一般人有何不同。然而，若難以避開不利色盲人士辨識之顏色，則務必加上其他視覺線索，或同時以色調深淺加強顏色所提供之線索，避免顏色成為單一線索。

圖 7-7 色盲人士所見顏色模擬網站

針對人們難以精確感知的另一項線索 — 面積與體積，特別提醒大家慎用立體圖。立體圖雖然可能有助呈現複雜的向度，但在資料視覺化講求有效準確溝通的脈絡下，非必要時，建議避免使用立體圖，因為體積也是一項人們較難以精確感知的視覺線索。換言之，除非所欲呈現的維度多到無法透過二維空間呈現，再考慮以二維的平面空間來呈現三維以上的立體圖。當然，若跳脫嚴謹的資料視覺化前提，當你認為讀者無法準確感知立體圖形大小等並不影響你所要傳遞的資訊時，立體圖仍然可使用，一切端看作圖的首要目標為何。

此外，使人分心的背景、顏色與裝飾等皆無益溝通。以圖 7-8、7-9 為例，其用色亦不符合前述原則，其背景與立體長條圖皆對讀者理解圖中資訊無太大助益。尤其此二圖所欲呈現的資訊僅包含兩個向度，以單純的二維平面圖來呈現已綽綽有餘。因此，就資料視覺化原則而言，此二圖有諸多不甚合理之處。其中，圖 7-8 的資料呈現方式可參考本章第四節地圖的作法。

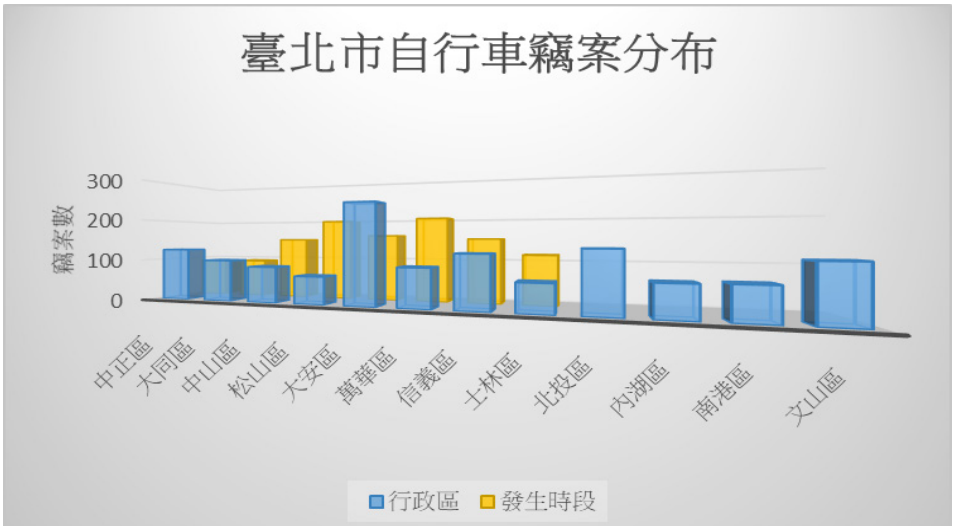


圖 7-8 人們較難精確掌握立體圖

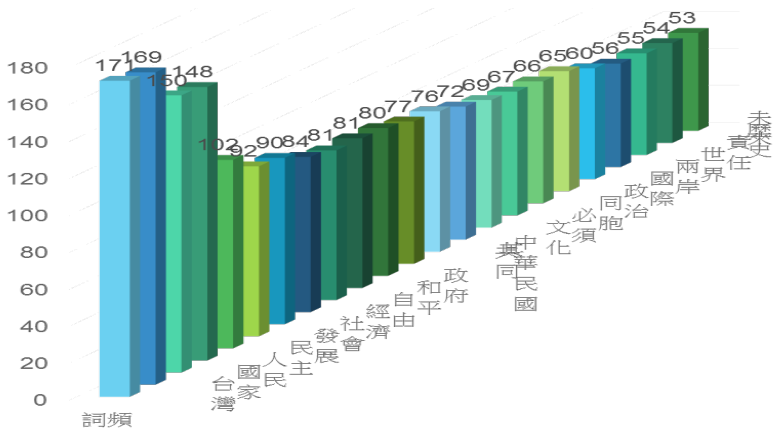


圖 7-9 人們較難精確掌握立體圖

最後，項目順序的重要性也是值得一提的原則。順序的安排可說是一項隱藏的重要視覺線索，順序本身雖非視覺元素，但卻深深影響了我們的視覺理解。例如：只要將簡單的長條圖稍加排序，無論是由高至低或依照年代時序排列，都能讓人更快速理解圖中的資訊，找到最大、最小值或高、低點，有助進行更快速的比較與類聚。總之，清楚掌握自己究竟想要傳遞哪些資訊是最根本的。根據所欲傳達的主軸，對圖表中項目加以排列，將大幅提升該圖表的易讀性。

介紹了這麼多資料視覺化原則與相關叮嚀，有些時候作圖實在難以面面俱到，但有了這層認識與理解，未來在進行資料視覺化時，可更彈性地運用各項原則，並在當下情境中，作出最理想的取捨。例如：倘若為了使用某些代表特定商標的顏色而犧牲了符合色盲人士的需求，則必須同時運用更明顯的色調對比，並搭配其他多重視覺線索來作圖。而平時看新聞或閱讀到包含各種圖表的內容時，可隨時練習用資料視覺化相關原則加以檢視與評論，如此將有助將相關原則內化並自然地運用至自己的視覺呈現中。

選擇適合的呈現方式

當我們要選擇適合的視覺呈現方式時，除了要了解用來作圖的欄位變項之屬性（如：類別變項、連續變項）之外，更要掌握作圖的主要目的。其實，若以簡潔易懂與有效溝通為主要訴求，國民中小學數學課所學的基本概念對於選擇適合的視覺呈現已相當充足。針對變項數量與屬性，選擇長條圖、直方圖、折線圖等呈現，可清楚表達所欲傳遞的資訊內容。

美國天主教大學行銷學系的 Andrew Abela 教授自 2009 年以來，製作了幾份協助選擇視覺化呈現方式的決策圖。此決策圖簡明易懂，以下列出視覺化的主要目標及較合適的視覺化呈現方式，供選擇視覺呈現方式之參考：

- 項目比較 → 長條圖
- 歷時比較 → 折線圖、長條圖、雷達圖
- 變數間的關係 → 散布圖、泡泡圖
- 分布狀況 → 直方圖、曲線圖、散布圖
- 靜態比例分布 → 圓餅圖、瀑布圖、100% 堆疊長條圖
- 歷時變化情形 →100% 堆疊長條圖、堆疊長條圖、堆疊面積圖

上述各種圖形樣式可參考 Andrew Abela 在 The Extreme Presentation

Method 網站上，根據 Gene Zelazny 的《Saying It With Charts》一書所繪製的「選擇好圖表」（Choosing a good chart）之內容（https://extremepresentation.typepad.com/files/choosing-a-good-chart-09.pdf）。而 Juice Analytics 也據此製作了線上圖表選擇工具並提供微軟 Excel 與 PowerPoint 格式的範例案下載（http://labs.juiceanalytics.com/chartchooser/index.html；圖 7-10）。

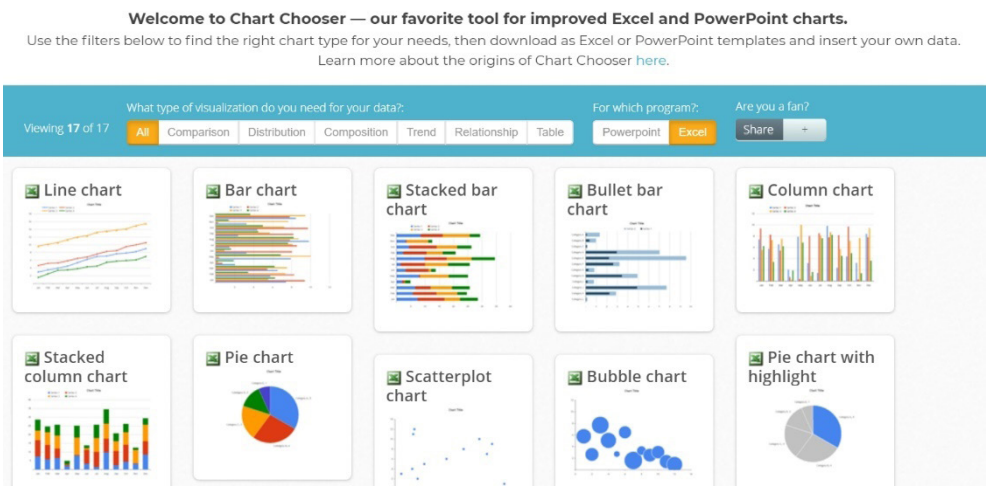


圖 7-10 Chart Chooser

小結

資料視覺化旨在協助我們有效溝通與傳遞資訊，透過各種視覺呈現，協助讀者與受眾減輕認知負荷、消化大量龐雜的原始資料，讓更豐富且有意義的訊息，更快速有效地傳遞。

讀完本節，你應該已掌握資料視覺化的核心精神。當你手邊拿到一份資料，就大膽依循前述資料視覺化原則，適當地運用視覺線索，選擇合適的視覺呈現方式，莫忘初衷，好好地繼續跟隨本章腳步，為資料說出它的故事吧！

7-2 資料視覺化工具選介與準備工作

資料視覺化工具選介

「工欲善其事，必先利其器」。若你透過 Google 搜尋引擎查找資料視覺化工具，檢索結果多得嚇人，各種推薦文論及的工具也是五花八門、不勝枚舉。當我們要進行資料視覺化，究竟該如何選擇工具呢？

研究資料視覺化常用的工具時常與資料型態與資料分析方法息息相關。因此，選擇資料視覺化工具時，首先要了解即將用來視覺化的資料型態（如：文字字串、數值、地理資訊）與主要預計用以呈現的方式為何（如：長條圖、網絡圖、地圖）。一般說來，廣為人知的資料分析與處理工具，如：Excel 試算表、SPSS 統計分析軟體，皆包含製作圖表的功能，可供製作常見的各類型圖表。在需求單純的情境中，這些資料分析與處理的軟體，不失為資料視覺化的簡便工具。

而若你打算製作某些特定形式的圖像時，亦可選擇專門針對此類型的軟體。例如：欲製作社會網絡圖時，往往就可直接選擇專門的軟體來作圖。而較常見的社會網絡圖工具，則有以 Microsoft Excel 插件方式使用的免費開源社會網絡分析軟體 NodeXL（http://nodexl.codeplex.com/）、UCINet、Gephi、Pajek 等等。

又如果，你想要選擇可支援多種呈現方式的工具，則可考慮專門視覺化的軟體或平臺，例如：Google 推出容易上手的 Google Data Studio、臺灣資料視覺化團隊自行開發的線上圖表製作平臺 PlotDB，以及本章第三、四節實作部分所欲介紹的資料視覺化軟體 —Tableau 等等。

Tableau 簡介

Tableau 是互動式的資料視覺化軟體，曾獲得諸多獎項，如：Best Overall in Data Visualization、Best of 2005 for Data Analysis、2008 Best Business Intelligence Solution。其主要產品包括：Tableau Public、Tableau Desktop、Tableau Online 和 Tableau Server 等。其中，Tableau Public 為免費版，功能較有限，且須仰賴雲端空間；Tableau Desktop 功能較完整，可離線操作，亦可將成果儲存在雲端。相形之下，若須自動更新資料則可選擇使用 Tableau Online 或 Tableau Server。

而當我們要選擇適合的 Tableau 版本時，可根據其可支援匯入的檔案格式、視覺化成果的分享方式，以及安全性等面向來衡量。下表綜整 Ryan Sleeper（2018）在《Practical Tableau》一書中的說明，可作為軟體版本選擇時的參考。

	Tableau Public	Tableau Desktop	Tableau Online	Tableau Server
資料匯入支援格式	Excel, text	Excel, text, Access, 統計檔、Tableau 檔	已發布在 Tableau Online 上且已授權之 workbooks	已發布在 Tableau Server 上且已授權之 workbooks
成果分享方式	雲端	離線或雲端	雲端 第三方 off-premise	自家就地部署 on-premise 或雲端
自動更新資料	無法	無法	可	可
安全性	仰賴雲端稍有風險，但可限制他人下載所發布至雲端的成果	視電腦及伺服器本身之安全性而定	視第三方安全性而定	視所在伺服器安全性而定

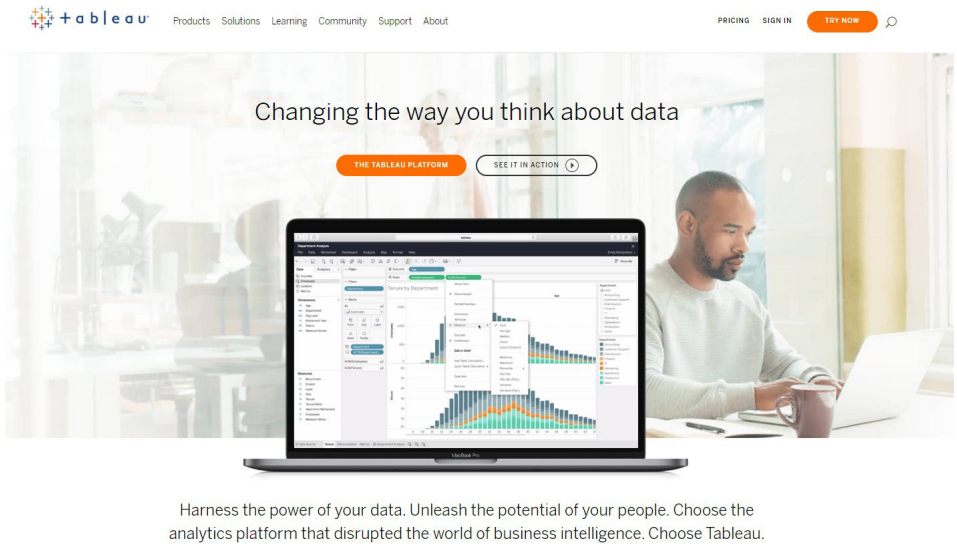


圖 7-11 Tableau 官方網站（https://www.tableau.com/）

而 Tableau 官方所提供的學習資源堪稱豐富，參考相關案例可見 Tableau Gallery（https://public.tableau.com/en-us/s/gallery） 或 Viz of the Day（https://public.tableau.com/en-us/s/gallery?qt-overview_gallery=1）。

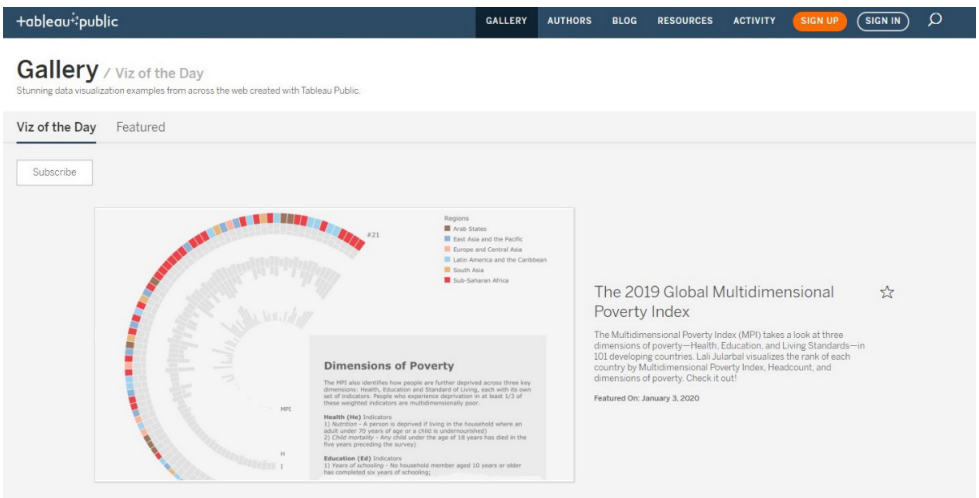


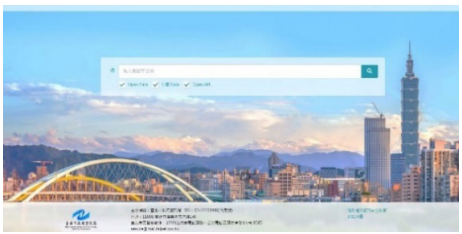
圖 7-12　Tableau Gallery 提供豐富的案例

Tableau 的官方網站也包含了許多學習資源。初學者可根據自己的需求，多加利用 Tableau Public 官方網站上的各項學習資源，透過學習影片（How-to Videos: https://public.tableau.com/s/resources）循序漸近地從零開始學習如何操作此軟體，運用官網提供的各種「範例資料」作練習；同時，參考官網整理的「社群資源」，掌握國際間資料視覺化相關社群所舉辦各種視覺化競賽或其他活動、Podcasts，以及社群媒體等。當然，Tableau 的官方網站亦提供軟體相關支援（Tableau Help），分別針對 Tableau 旗下不同產品作說明。此外，Tableau 還有其知識庫（https://www.tableau.com/support/knowledgebase）與論壇社群（https://community.tableau.com/），都是學習歷程中遭遇問題的好幫手。無論知識庫或論壇社群皆提供搜尋功能，可直接將目前操作上的問題以關鍵字進行搜尋，查找相關解決辦法，大多數的問題都可在此得到解答。只是對中文使用者而言，以中文查找通常可查得的結果可能不如英文豐富。雖然相信隨著 Tableau 的中文使用者成長，中文資源也將日漸豐富，以目前的情況來看，建議若以中文關鍵字難以獲得解答，可嘗試改以英文關鍵字搜尋；閱讀英文不大費力的話，則可直接以英文關鍵字進行搜尋。

最後，除了運用免費的 Tableau Public 或購買 Tableau Desktop 等版本外，Tableau 亦針對學校師生提供免費授權方式。教師若開設相關課程，於開設該學期可透過 Tableau for Teaching，填妥學校單位以及課程相關資料（https://www.tableau.com/academic/teaching），申請教學授權版的 Tableau Desktop；學生在學期間亦可透過 Tableau for Students 檢附在學證明等資料（https://www.tableau.com/academic/students），申請為期一年的學生版。本章相關介紹係以 2019 年下半年至 2020 年初筆者掌握的情況來說明。

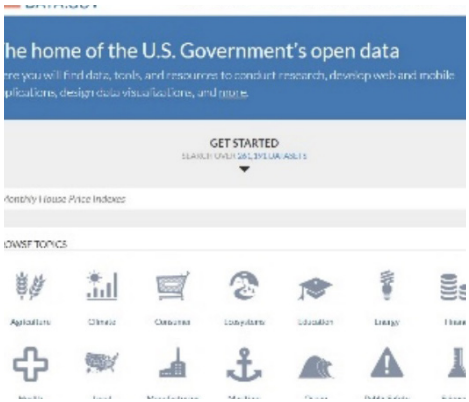
資料分析與視覺化的基本步驟

一般說來，研究資料視覺化的流程涉及研究資料的整個處理過程，包括：資料蒐集、資料清理與準備、資料分析（包含統計檢定），以及資料呈現（即視覺化）。其中，資料分析與資料呈現有時可來回循環地進行，尤其若你處理的議題或研究愈具探索性質，愈容易在二階段間往返徘徊。因為隨著資料視覺化的探索，可能讓你對資料分析產生更多新的想法；隨著資料分析的深化，可能也讓你對資料視覺化產生更多新的想法。



臺北市資料大平臺
https://data.taipei/

政府開放資料平臺
https://data.gov.tw/



美國政府開放資料平臺
https://www.data.gov/

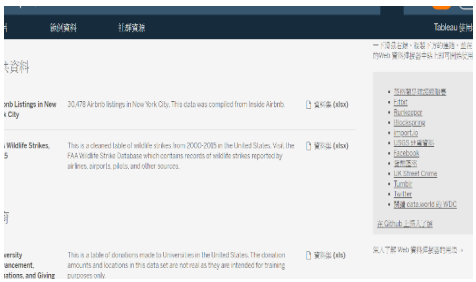


Tableau 範例資料集
https://public.tableau.com/s/resources

圖 7-13 政府開放資料與 Tableau 範例資料來源

就資料蒐集而言，研究資料的來源可能是既有的資料，像是政府開放資料或相關軟體的範例資料（如：Tableau 範例資料集）等等（圖 7-13），也可能是自行蒐集（如：問卷調查所得結果）。許多時候，由於單一的資料集未必能滿足當前所欲探討問題的資料需求，因而必須設法蒐集並整併來自不同資料來源的資料，以解答目前所欲探究的問題。例如：當你想製作一張人口密度與自行車竊案的關係圖，以了解人口密度高之處是否較易發生竊案時，可能需要從不同政府部門的統計資料中，取得人口密度與自行車竊案數等資料。然而，不同來源的資料無法任意整併（例如：不同範圍或不同方法的調查無法直接合併比較），如欲嚴謹地蒐集資料，多半須要具備研究方法相關知識與能力，以利判斷如何結合不同來源的資料，以探討相關議題。

完成資料的蒐集後，就進入資料視覺化品質決勝的關鍵階段了。資料清理與準備是資料視覺化非常重要的一環，也是相當費心與費時的階段。資料清理工作做得愈嚴謹，資料的品質也愈理想。而資料分析則是影響資料視覺化深度與廣度的重要環節，倘若能以適合的資料分析方式處理資料，則可強化討論的深度與廣度。然而，資料分析涉及各種學科知識背景與研究方法的知能，且各種不同的資料分析方式皆各有其學問，顯然已超出本章範圍，故對資料分析感興趣的讀者可再鑽研自己感興趣領域的研究方法與資料分析相關書籍。

Information is Beautiful 官方網站指出，繪製資訊圖表的過程當中，大約 80% 的時間都在蒐集與整理資料，只有剩下 20% 的時間花費在規劃與實際將資料轉化為視覺呈現。因此，以下針對資料清理進一步說明。

資料清理

所謂「資料清理」是指將原始資料轉換為可用於資料分析 / 統計分析 / 視覺化的形式，也就是將從資料來源而來而無法直接分析的原始資料（Raw Data）轉化為資料可直接用於分析的乾淨資料（Clean/Tidy Data）之過程。一般說來，可分析的資料結構是以每個變項（variable）為一欄、每個觀察值（observation）為一列的方式呈現的，許多時候可能需要合併、分割等處理（例如：以識別碼合併多個表格，將相同的項目合併、將原本列在「其他」中的項目歸類）。由於資料清理的過程繁複，且如欲改變清理資料的策略，則可能需要退回先前的步驟，故詳實地記錄每個資料清理的步驟是非常重要的工作，在團隊合作中尤然。

若未適當清理資料，則資料分析 / 視覺化的結果無法減輕讀者認知負荷，或無法有效呈現重要發現，甚至可能混淆視聽。圖 7-14 呈現了學生就讀學系分布概況，然而，該圖即為未適當清理資料即將資料視覺化之實例。在資料清理過程中，應先將相同的項目進行合併，才能看出不同學系的分布情形。此圖在製作前並未清理資料，故相同學系的人數散落於不同項目中，讀者須自行將這些相同的項目加總，才能得知特定學系的人數。也就是說，清理資料時，應先合併相同的類目，即合併「資訊管理」、「資管」與「資訊管理學系」、合併「資工」、「資訊工程」、「資工系」與「資訊工程學系」，以及合併「圖書資訊」、「圖書資訊學系」與「圖資」，才能確實呈現出資訊管理學系、資訊工程學系，以及圖書資訊學系的人數分布概況。

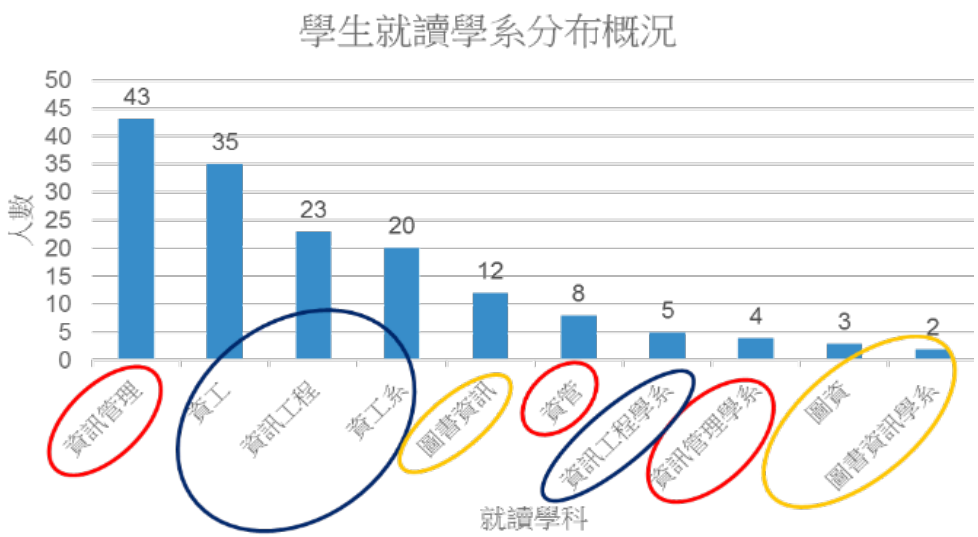


圖 7-14 未適當清理資料則相同的項目散落各項，難以有效溝通

而常見的資料清理工作可能包括：處理遺漏值、檢查拼字與更正錯漏字等細節、檢查並更正資料格式（如：統一日期、電話號碼等格式）、核對範圍（如：確認年齡須為正值、成績須落在一定範圍區間內）、重整變項中之類別（如：將列在「其他」的項目歸入既有類別或另行歸類），以及合併或新增變項等。而 Information is Beautiful 網站也提供了清理後的資料範例，可供參考：http://www.informationisbeautiful.net/data/。

小結

資料視覺化工具不勝枚舉，充分掌握資料特性與資料分析與呈現的主要目標，才能幫助自己選擇適合的視覺化工具。而資料視覺化的主要步驟包括：資料蒐集、資料清理與準備、資料分析，以及資料呈現。其中，資料呈現之前的各項準備工作相當費時費力，約佔整個資料視覺化歷程所花費時間的 80%。資料蒐集時，可善用各種政府開放資料，而資料清理的工作務必要嚴謹以對，才能維護資料品質，做出有意義且有效的視覺化成果。

7-3 以政府開放統計資料談數據資料視覺化

本節將帶領 Tableau 新手一步步學習如何運用此軟體進行資料視覺化，先從 Tableau 的初始介面開始談起，並以政府開放統計資料為例，說明如何使用 Tableau 簡易製圖，再進一步介紹 Tableau 的函式語法。

Tableau 初始介面

當你進入 Tableau 軟體，首先，將看到圖 7-15 之初始介面。它與一般人常用的微軟文書處理軟體的初始介面不大相同，然而，其初始介面相當單純，主要分為資料連結、開啟檔案與探索資源三大區塊。

其中，最左上角帶有 Tableau logo 的小花圖示是快速切換資料與視覺化工作表的快速切換鍵，待資料匯入後，如果退回此介面，可直接點擊此快速鍵。而左半部的主要區塊則提供連結與匯入資料的重要功能，可直接點選所欲匯入的資料檔案類型，即可找到資料位置並匯入 Tableau 中。例如：對新手而言，最常見的就是連結 Excel 試算表，故可直接點選 Microsoft Excel 即可匯入資料。

乾淨俐落的中間區塊提供了一項主要功能 — 開啟舊檔，當你開始使用此軟體之後，近期使用的檔案亦將出現在此區塊，直接點擊欲開啟的檔案即可。中間下半部則可開啟 Tableau 提供的範例檔案。至於右側探索區塊則可讓使用者連結到 Tableau 官方網站的各項教學資源，包括：教學影片、每日精選作品等。

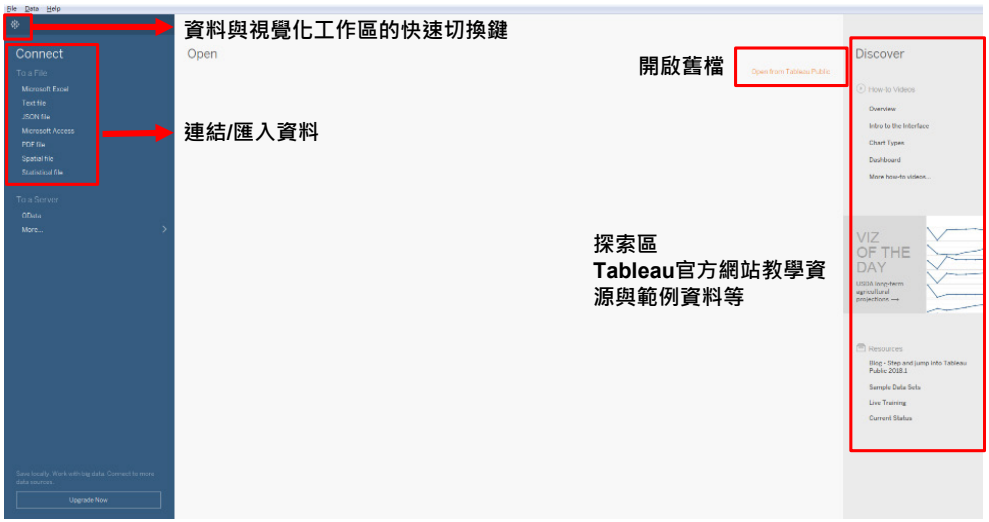


圖 7-15 Tableau 初始介面

Tableau 工作介面

Tableau 工作介面除了與一般軟體相同、置於頂端的工具列之外，大致可分為四區，由左至右，依序包括：資料欄位變項與資料分析區、視覺設定區、工作區，以及圖形選擇區（圖 7-16）。當資料匯入之後，最左側將出現 匯入的欄位變項，我們即可利用這些欄位變項來作圖。而當我們將變項置於中間偏上的欄與行 （列）時，Tableau 就會自動幫我們繪製圖形，而此圖形將出現在圖中空白的工作區。透過最右側的圖形選擇及中間偏左的視覺設定區，我們可進一步調整圖形樣式，並根據當前的需求，進一步將 Tableau 自動產生的圖像客製化。

圖 7-16 Tableau 工作介面

Tableau 的基本操作流程

由前述介面說明，我們其實已大致掌握了 Tableau 的基本操作流程了。一般說來，簡易的製圖不外乎進入初始畫面並連結 / 匯入資料，並確保資料型態設定正確（例如：連續變項設為數值、類別變項設為字串、地理資訊設為地理角色）。爾後，再進入工作分頁，準備開始進行視覺化的工作。

在準備資料的過程中，有時也將視需求，運用 Tableau 函式語法新增所需欄位變項等（例如：若資料僅包含學生的單科成績或每學期的成績，可運用公式計算出總平均）。接下來，當所欲用以製圖的欄位變項都準備妥當，就可開始繪圖了！

繪圖時，首先要選擇欲用以繪圖之欄位變項，將之拖曳到欄與行（列）。此時，Tableau 會自動根據所選的欄位變項類型與屬性來判定要幫你繪製什麼圖形。因此，若未確保欄位變項類型與屬性設定正確，Tableau 將難以畫出我們預想的圖。如果對 Tableau 自動繪製的圖形不滿意，則可透過圖形選擇區，直接點選合適的圖形。接著，即可開始進行客製化調整，增加標籤說明、修改顏色等等。最後，再將視覺化成果匯出，就大功告成了。

以下透過實作練習，讓大家可以了解前述流程的確切進行方式並掌握 Tableau 之各項基本功能。

Tableau 實作技巧

接下來，我們將運用美國教育統計開放資料 Integrated Postsecondary Education Data System (IPEDS)，介紹 Tableau 的基本功能。

1. 匯入資料

首先，請進入 Tableau 初始介面，透過連結 Microsoft Excel 的功能，開啟你存放本節欲用來練習的 IPEDS 課堂練習檔案的位置。由於此試算表檔案包含四個分頁，我們僅需要包含實際資料的 Data 分頁，因此，只要以滑鼠左鍵點選欲用來作圖的 Data 分頁不放，並拖曳至右側資料區上半部的空白處，再放開滑鼠，即可匯入該分頁之資料（如圖 7-17）。若資料檔案較大，得稍候片刻，待資料區的下半部出現資料表及其內容，確保所有資料匯入完成。

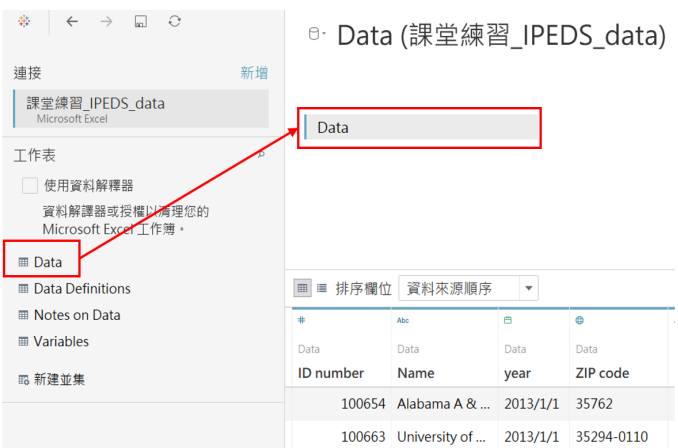


圖 7-17 將檔案匯入 Tableau

此外，雖然本節僅介紹 Tableau 的基本功能，但在此順帶一提，Tableau 也提供同時匯入不同分頁的功能，因此，如果我們想要一次匯入同一試算表的多個分頁，則可將不同的分頁循前述步驟，分別拖曳至右側資料區空白處，並選擇一特定欄位（如：ID）將不同分頁的資料加以連結合併。

匯入資料的過程中，Tableau 會自動判定每個欄位變項的型態與屬性，在每個欄位變項上方，以藍色和綠色分別標示該欄位屬於維度（dimensions）或量值（measures），並以 Abc、＃等符號標示該欄位變項是文字字串或數字等（圖 7-18）。其中，設定成量值的欄位變項可用於一般運算，而設成日期者可以年、月、日等單位處理；設定地理角色者可用以繪製地圖。

此時，我們要確保每個欄位的資料型態設定正確（如：類別變項設為字串、連續變項設為數字），以利 Tableau 自動作圖時，為我們選擇合適的呈現方式。

排序欄位 資料來源順序

# Data ID number	Abc Data Name	Data year	Data ZIP code	Abc Data Highest de...	Data County name
100654	Alabama A & ...	2013/1/1	35762	Doctor's degr...	Madison Cou...
100663	University of ...	2013/1/1	35294-0110	Doctor's degr...	Jefferson Cou...
100690	Amridge Univ...	2013/1/1	36117-3553	Doctor's degr...	Montgomery ...
100706	University of ...	2013/1/1	35899	Doctor's degr...	Madison Cou...

圖 7-18 確保資料型態設定正確

2. 簡易製圖：以並列長條圖的製作為例

最簡單的製圖方式，就是直接在連結檔資料後，進入工作表（worksheet），將欲用來繪圖的欄位變項拖曳至工作區的欄與行（列），再選擇圖形呈現方式（如：簡單長條圖、並列的長條圖、泡泡圖）。

一般說來，我們在做製圖的規劃時，可將 Tableau 自動判定成維度的類別變項設想成自變項，將 Tableau 自動判定成量值的連續變項設想成依變項。在這個練習中，我們若想繪製一張並列的長條圖，藉此比較不同學校類型的全職與兼職學生註冊人數，則可直接將欲用以繪圖的欄位變項拖曳至欄與行（列）。此例可將學校分類 [Carnegie Classification] 拖曳至行（列）（Columns）、全職學生註冊人數 [Full-time enrollment] 與兼職學生註冊人數 [Part-time enrollment] 拖曳至欄（Rows），如圖 7-19。

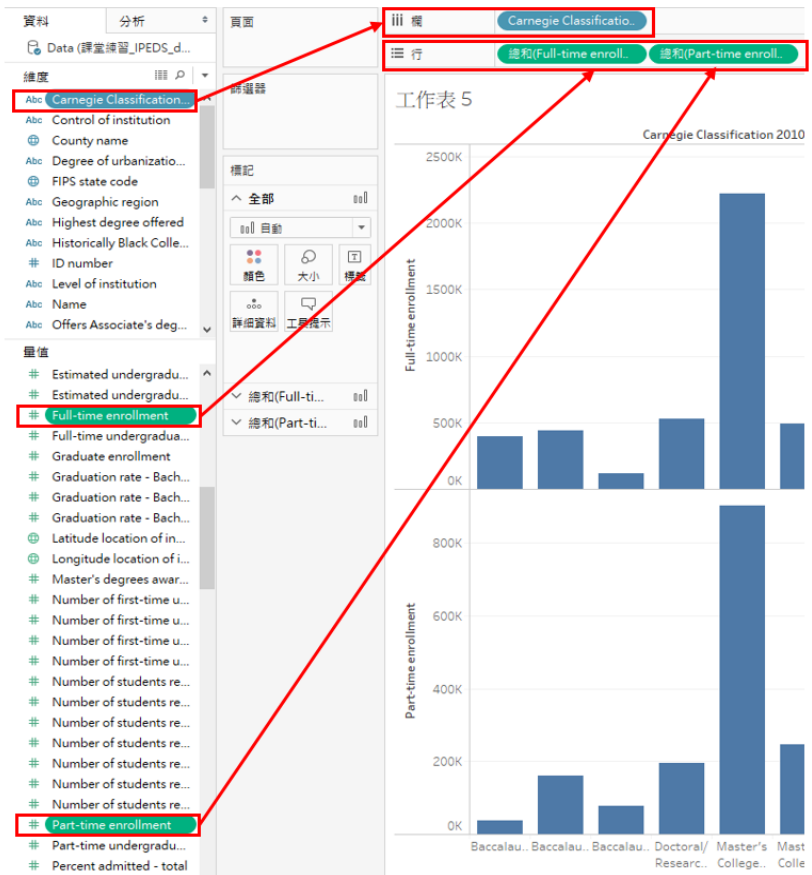


圖 7-19 將欄位變項拖曳至欄與行（列）

接著，透過最右側的圖書選擇功能「顯示」（Show Me），選擇合適的視覺化呈現方式。此例選擇的是「並列的長條圖」 。如此即可初步完成一張並列的長條圖。

3. 簡易客製化

3.1 修改顏色

如果你想進一步作些客製化調整，最常見的可能是修改顏色。在 Tableau 中，只要在標記區點擊「顏色」、「編輯顏色」，即可針對長條圖修改顏色，亦可在此標記區調整長條圖的不透明度與邊界等。

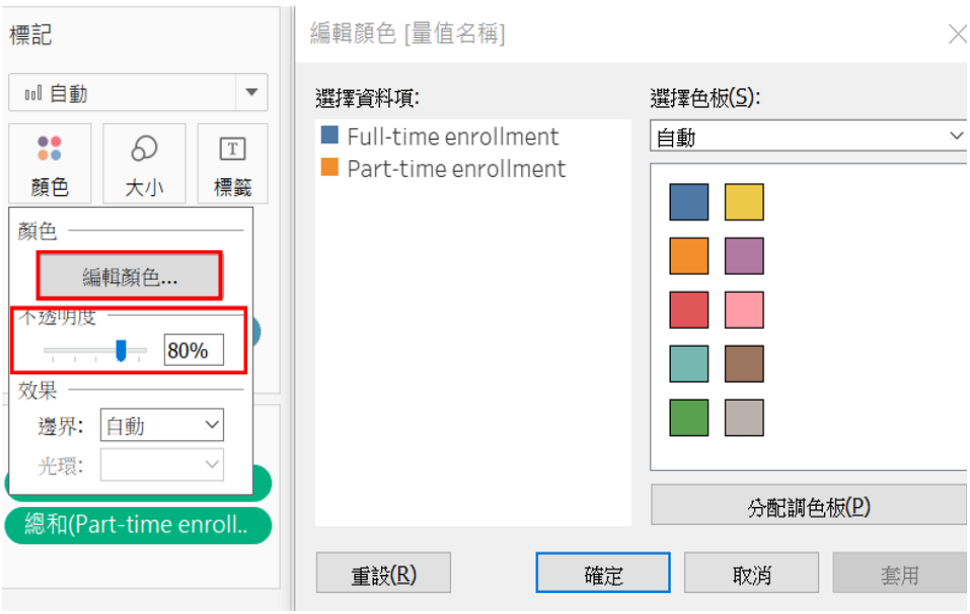


圖 7-20 修改顏色

3.2 新增標籤

若你想要為長條圖增加標籤（labels），以說明每個數值（此例是全職與兼職學生的註冊人數），則可在標記區點擊「標籤」，再勾選「顯示標記標籤」即可（如圖 7-21）。

圖 7-21 增加標籤說明

3.3 過濾篩選

此外，Tableau 還提供了過濾篩選部分資料的重要功能。就此例而言，大學分類項目略嫌太多，可能看了眼花撩亂，如僅欲呈現部分大學類型的資料，則可點選大學分類之變項（ [Carnegie Classification] ）或在工作區右側的空白處按右鍵，選擇「篩選器」，並選擇欲用來篩選之欄位變項（此處為 [Carnegie Classification] ）。接下來，再根據你所要選擇的項目作勾選即可（圖 7-22）。

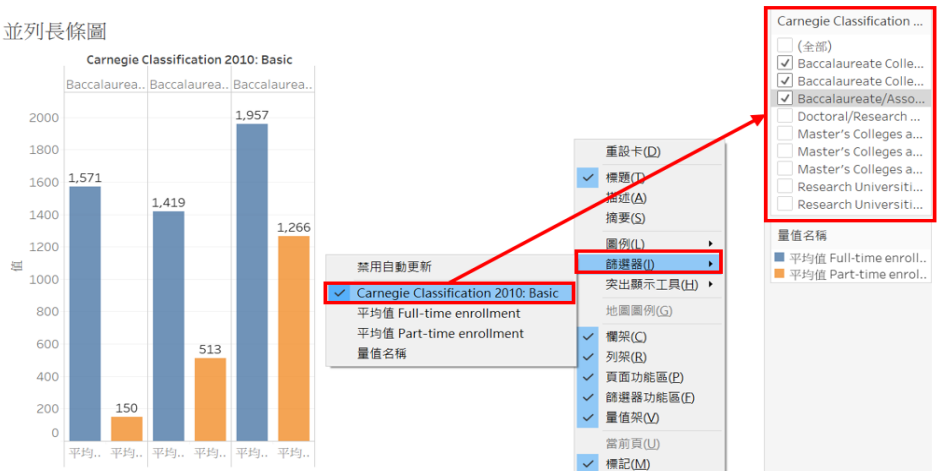


圖 7-22 過濾篩選部分資料

3.4 調整欄位變項的計算方式

最後，Tableau 還提供了一項可直接在視覺呈現中調整變項計算方式之強大功能。就本例而言，我們目前所製作的長條圖是以 Tableau 的預設值 — 總和來呈現的，然而，若我們認為應以平均數來呈現，方可看出各類型學校平均的招生人數情形，則可直接點選已位於度量值區塊中的欄位變項右緣的小三角箭頭，將其度量中的總和改成平均值（圖 7-23）。而在此並列長條圖中，用來對照比較的兩個欄位變項（即全職學生註冊人數 [Full-time enrollment] 與兼職學生註冊人數 [Part-time enrollment] ），二者須修改成相同的計算方式，其對照比較才有意義。所以，此處二者的計算方式須一併由總和改成平均值。

在此順帶提醒，製圖時，須留意 Tableau 預設之數值呈現方式，若其預設值不合理或不合意，則可隨時將其計算方式調整為總和、平均數、中位數、最大值、最小值等。這是一項相當方便而實用的功能。

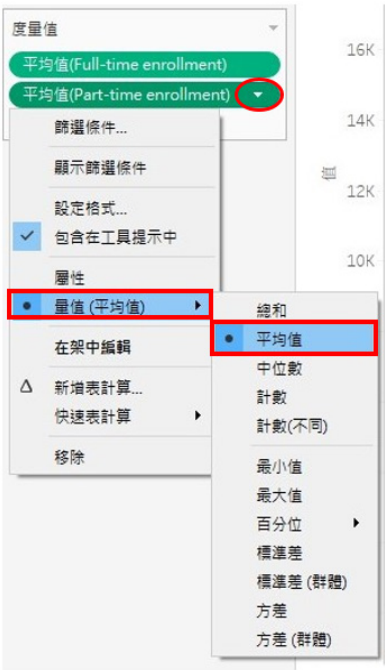


圖 7-23 調整欄位變項的計算方式

4. 管理技巧與匯出視覺化成果

4.1 檢視原始資料

在 Tableau 中，如欲檢視原始資料，可直接退回「資料來源」之頁籤，亦可由工具列的「資料」、「資料 (資料檔案名稱)」、「檢視資料」來對照匯入的資料內容（圖 7-24）。直接退回「資料來源」頁籤固然方便，但有時欲同時檢視視覺呈現與資料，則可運用後者方法來檢視資料。

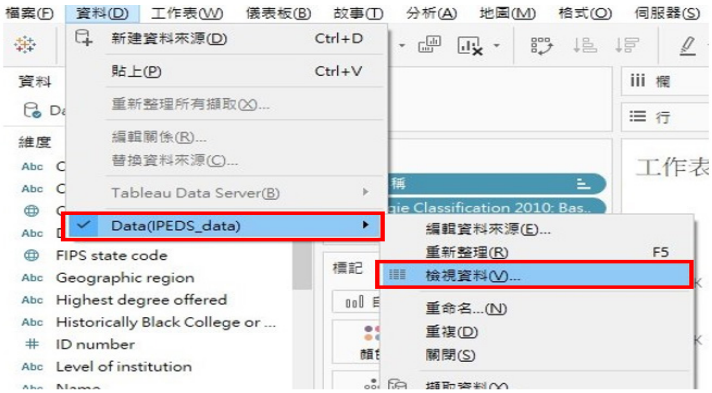


圖 7-24 檢視資料

4.2 管理分頁

Tableau 的工作頁籤分為三種形式：工作表、儀表板與故事。最基本的視覺呈現工作皆是由工作表展開。一般皆在工作表製圖，待所有圖片製作完成後，再視需求，將完成的工作表拼貼整併至儀表板，或將圖片以連續性的故事形式呈現。若要管理分頁，最簡便的方式就是直接於分頁標籤上按右鍵，即可「新建工作表」、「新建儀表板」、「新建故事」、「重複」（複製整個工作表）或「刪除」當前的分頁；如欲修改當前的分頁名稱，則可透過「重新命名」作調整（圖 7-25）。

圖 7-25 管理工作表

倘若你要保留當前的工作表，但想清除工作表中的內容，則可點選工具列的「刪除圖示」，即可透過選擇「清除工作表」來清除內容（圖 7-26）。Tableau 不會再次詢問你是否確定要刪除等，因此，操作類似的動作，須格外謹慎。萬一不慎清除工作表，可盡快點選工具列的復原，或用 Ctrl+z 來退回至上一步。

圖 7-26 一鍵清除工作表內容

此外，使用者亦可將 Tableau 的分頁標上不同顏色，加以分類管理。當你在一份檔案中的分頁較多時，不失為一項方便的管理功能。與前述管理分頁的作法相似，只要在分頁標籤上按右鍵，即可將分頁標示成不同顏色（Highlight worksheet），以便管理（圖 7-27）。

圖 7-27 將工作表標上顏色，以便管理

4.3 儲存、分享與匯出工作簿

最後，當你完成視覺化任務後，往往須要存檔或分享。在 Tableau 中，儲存整個工作簿就如同一般軟體，透過「檔案」、「儲存」或「另存新檔」即可儲存工作簿。然而，直接儲存下來的工作簿，只能透過 Tableau 軟體讀取，如果你希望將工作簿分享給非 Tableau 使用者，則須透過「檔案」、「匯出已封裝工作簿（Export Packaged Workbook）」，將檔案存成封裝的工作簿。

而當你希望將特定的工作表匯出，通常可透過以下兩種方式將工作表匯出為 PDF 檔或圖檔。不過，若欲匯出 PDF 檔，得從「檔案」、「列印為 PDF」來執行；若僅欲儲存圖檔，則須從「工作表」、「導出」、「影像」來執行。

4.3.1 匯出 **PDF**

從「檔案」和「列印為 PDF」來匯出 PDF 檔時，可選擇欲匯出的範圍是整個工作簿（所有的分頁）抑或目前使用中的工作表，亦可指定紙張尺寸（圖 7-28）。若你所作的圖片尺寸大於列表機預設紙張尺寸的大小（如：超出 A4 或 letter size 的大小），則可考慮將紙張設定為「未指定」。因為當工作表中的圖片超過紙張尺寸時，匯出的 PDF 檔會自動將圖片切割成多頁，如此一來，就難以檢視或列印完整的圖片了。因此，許多時候，可考慮直接將紙張尺寸設定為「未指定」，待實際有列印需求時，再透過讀取 PDF 的軟體來設定縮放比例即可。

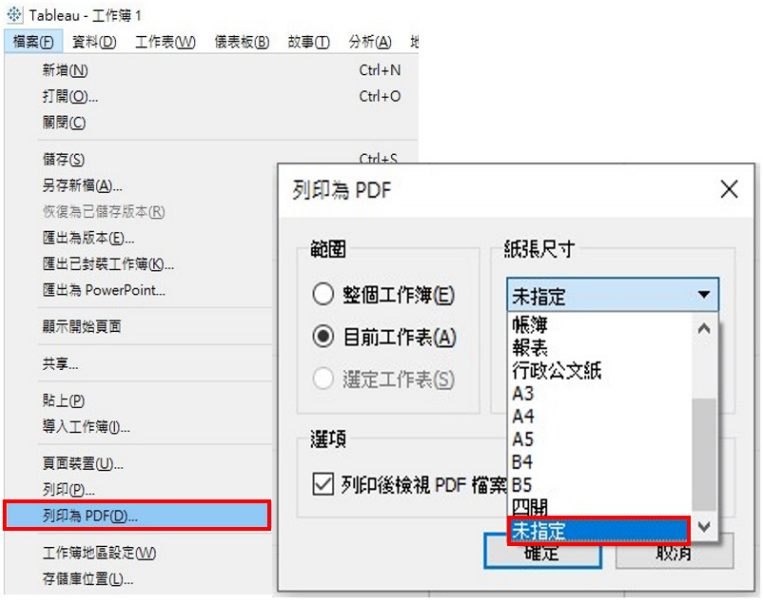


圖 7-28 匯出 PDF 檔

4.3.2 匯出圖檔

在 Tableau 中，匯出圖檔與其他儲存分享功能不同，並非透過「檔案」功能來執行，而須從「工作表」、「導出」、「影像」來執行。這點可能令不熟 Tableau 的人感到有些困惑。因為 Tableau 將主要的視覺呈現工作分為工作表、儀表板和故事三大類，所有的工作皆以此為據，所以，當我們只針對其中一個工作表的內容進行任務時，必須直接透過工具列的「工作表」來操作。

圖 7-29 呈現了匯出圖檔的步驟。當你選擇導出影像後，匯出圖像的視窗將會跳出，供你選擇匯出的圖像是否包含標題與描述。最後，在存檔時，可選擇圖片的檔案格式（.jpg、.png、.bmp 或 .emf）。

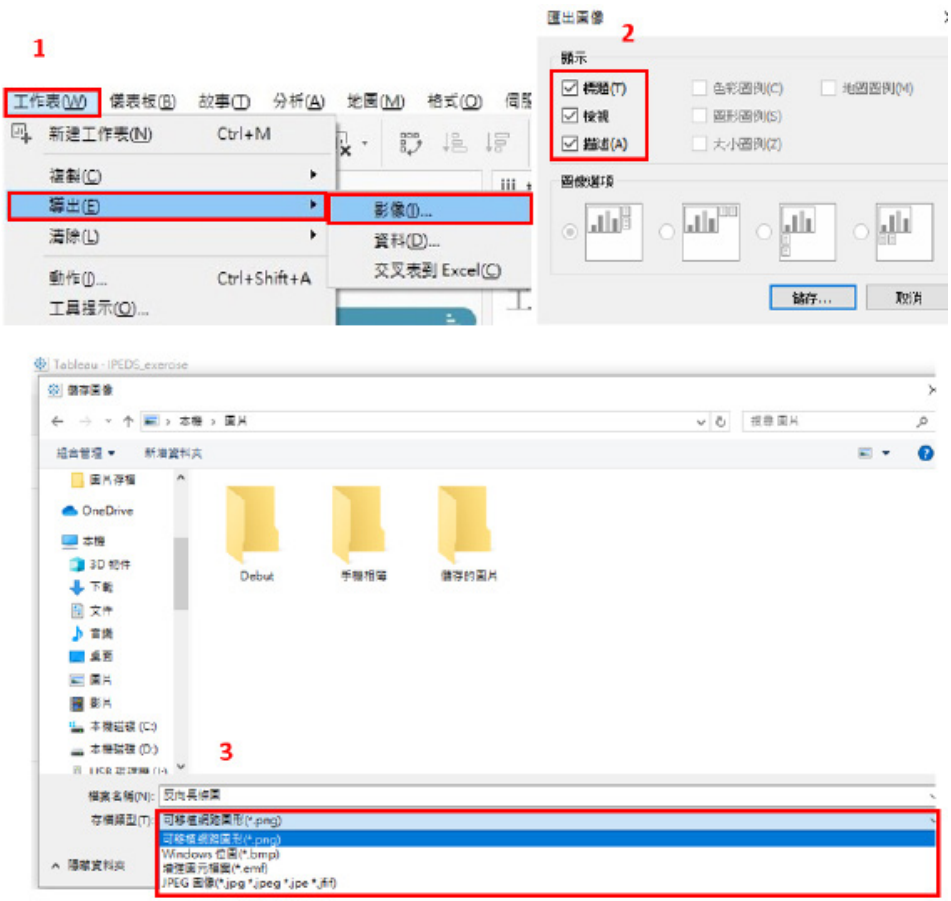


圖 7-29 匯出圖檔

反向長條圖的製作

接下來，我們將在同一個檔案的新分頁中，運用前述 IPEDS 教育統計資料，繪製不同類型大學收受學生數與註冊人數之長條圖，藉此說明反向長條圖的製作方式。

1. 簡易製圖

先前製作並列長條圖時，我們用的學校分類是美國教育統計常用的卡內基高等教育分類 [Carnegie Classification]；現在我們將運用最高授予學位 [Highest

degree offered] 之欄位變項來探討學校類型。當然，你也可以嘗試運用 IPEDS 檔案中的其他有趣變項來製圖。

首先，將收受人數 [Admission total] 和註冊人數 [Enrolled total] 拖曳至欄，並將最高授予學位 [Highest degree offered] 拖曳至行（列）（圖 7-30 步驟 1）。如此，Tableau 已為我們自動產生兩個並排的長條圖。

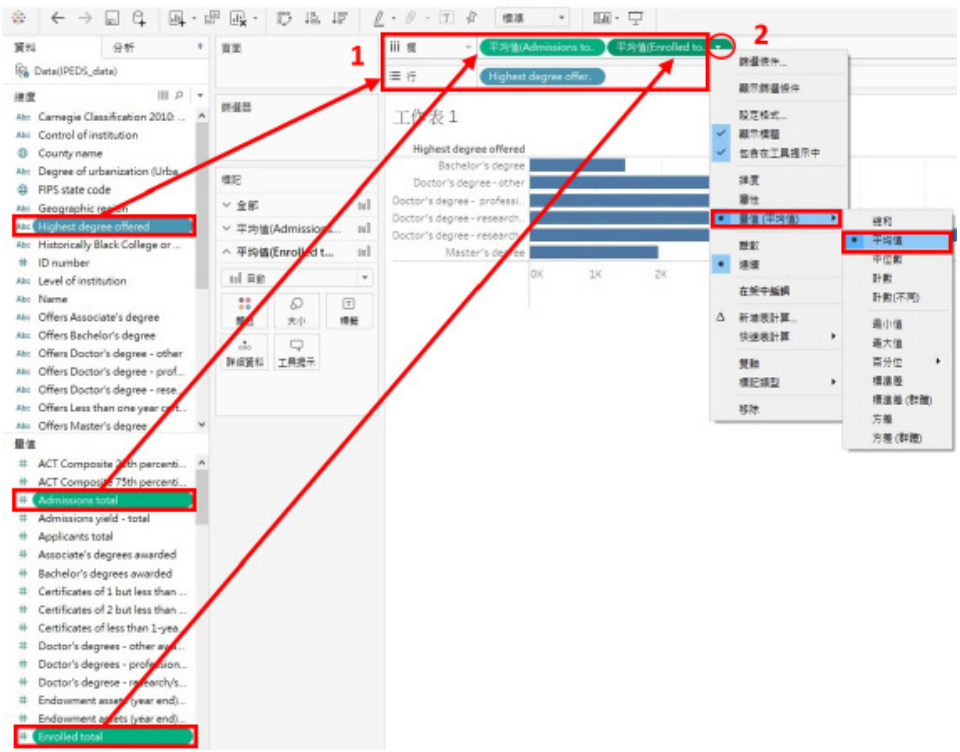


圖 7-30 繪製長條圖

2. 調整欄位變項的計算方式

Tableau 同樣預設以收受人數及註冊人數的總和來繪圖，但我們希望以平均數來呈現不同類型學校的人數情形。此時，我們可運用先前介紹並列長條圖製作的方法，先對此長條圖進行資料呈現上的調整，將收受人數 [Admission total] 和註冊人數 [Enrolled total] 改以平均數呈現（圖 7-30 步驟 2）。

3. 反向軸之調整

接著，我們欲將左側長條圖轉向，讓兩個並排的長條圖分別朝左右反向展開，因而須調整左圖軸的方向。欲調整軸的方向，可直接在左半圖之軸上按右鍵，並選擇「編輯軸（Edit Axis）」（圖 7-31）。此時，編輯軸的視窗將跳出。我們只要在編輯軸視窗中，勾選「倒序」即可。如此一來，左側軸向以倒序（即反向）排列，即可快速完成一反向長條圖。

圖 7-31 設定反向軸

4. 新增參照標準與排序

Tableau 在資料分析方面提供了簡單易用的功能，方便我們為圖像加上參照標準（如：常數線、平均線、四分位差線）。倘若我們欲在剛才繪製的反向長條圖中加上四分位差線作為參照，則可在資料區選擇分析（Analytics）頁籤，再點選欲加到圖中的參照標準，將之拖曳至圖表工作區。

在我們拖曳的過程中，滑鼠移動到工作區時，會出現圖 7-32 之方框，供我們選擇參照樣式，即參照標準將繪製到整個圖表、區段，或儲存格上。此處，我們可選擇「表」，讓四分位差繪製到整個長條圖上。

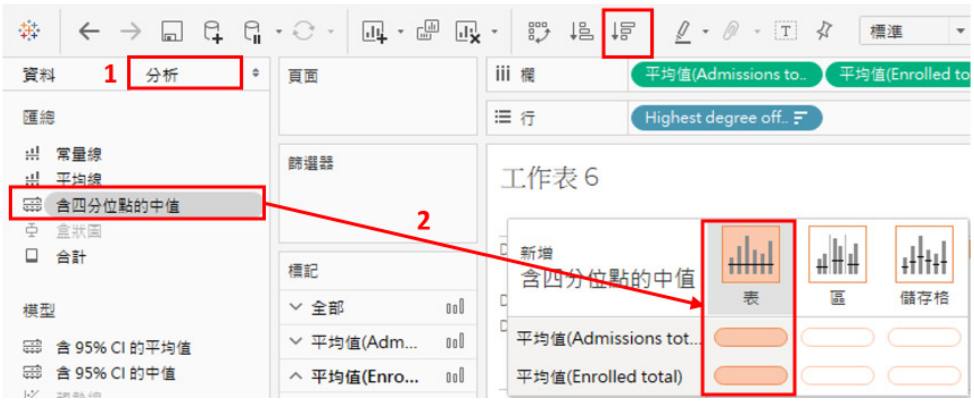


圖 7-32 新增參照標準與排序

此外，我們亦可利用工具列的排序鍵（圖 7-32），進一步針對長條圖進行排序。完成後，即可得到排序整齊並加上參照標準的反向長條圖了（圖 7-33）。

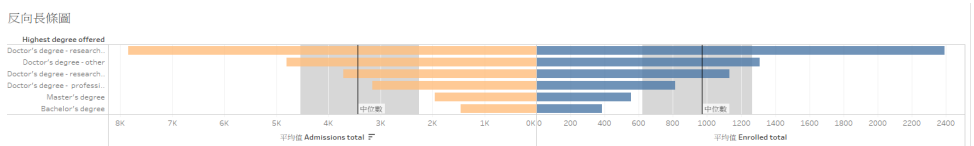


圖 7-33 初步完成反向長條圖

5. 調整尺規刻度

最後，由於此圖左右的尺規刻度不同，可能造成誤導，也難以快速比較左右兩側的數值多寡，因此，我們將進一步將兩軸之尺度調整成相同的，以利比較。作法上，我們可先在數值較大的一軸上透過右鍵點選「編輯軸」，檢視其最大值為何，並複製此數值，以便將另一側的軸也設定為相同的數值。

此例中，我們先在左側軸上，透過右鍵「編輯軸」的功能，檢視其範圍之最大值為何，複製此數值後，再到右側軸上，同樣透過右鍵「編輯軸」的功能，將此數值貼至其範圍的最大值。由於預設的範圍是自動產生的，故我們要將之改為「固定」，並將前述最大值複製到固定結束的空格中（圖 7-34）。

圖 7-34 調整尺度範圍

如此，則可得圖 7-35 之反向長條圖，以便了解不同類型學校的收受人數與註冊人數之落差。

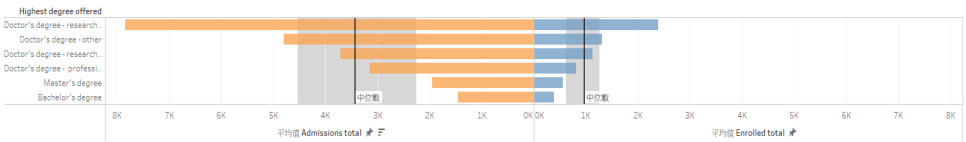


圖 7-35 調整尺度範圍後的反向長條圖

Tableau 函式語法簡介

雖然 Tableau 提供了許多簡單易用的功能，若能運用其函式語法，可大大提升其功能性。以下將簡介 Tableau 函式語法的運作：

在 Tableau 中，函式語法可進行簡單的四則運算（例如：計算總和、平均數），亦可用來取代部分字串、計算日期，或依條件計數等等。無論撰寫何種函式語法，皆是利用新增欄位變項來操作，因此，接下來將先介紹「建立計算欄位」之功能。

我們可以透過點選資料區中，維度或量值右方的的小箭頭，並選取建立計算欄位（Create Calculated Field），亦可透過工具列的「分析」、建立計算欄位來新增變項（圖 7-36）。

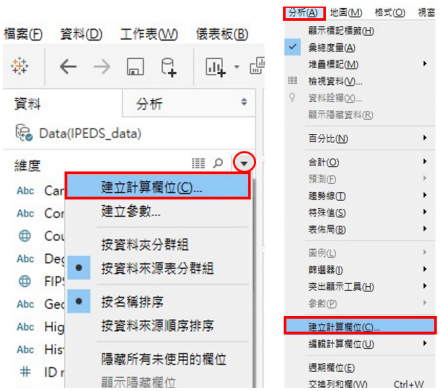


圖 7-36 建立計算欄位的兩種常見方式

接下來，就可為此欄位變項命名，並寫出計算公式或函式語法了。以前述 IPEDS 檔案為例，我們可先試著透過簡單的運算來練習運用此功能。若將註冊總人數 [Enrolled total] 之八成來估算理想的留校率（ideal retention rate），我們可在欄位變項名稱中，輸入「ideal retention rate」，並在計算區寫上算式，以註冊人數乘以 0.8。在 Tableau 中，欄位變項須以方括號 [] 表示，四則運算則可直接以加（+）、減（-）、乘（*）、除（/）表示。故此例中的算式為：[Enrolled total]*0.8（圖 7-37）。按下「確定」，即可新增一計算欄位。

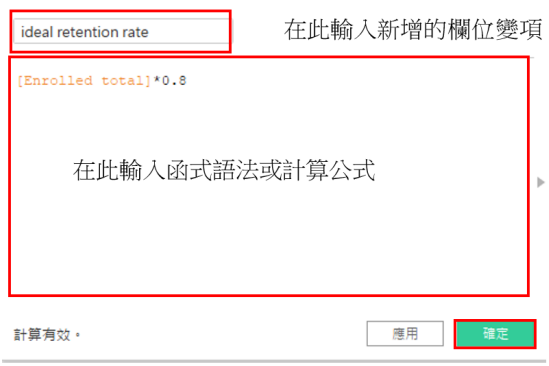


圖 7-37 建立計算欄位

Tableau 函式語法基本概念

如前所述，在 Tableau 中，欄位變項以方括號表示，例如：[年齡]；當表述值包含字串時，通常須加雙引號（“”），特定表述值加單引號（''），視所選函式而定（參見後段函式語法說明）。

除了可如前述直接列出算式，針對欄位變項進行四則運算外（例如：[國語]+[數學]+[英文]），亦可利用各種類型的函式語法進行其他運算。簡單的函式語法表述方式多為函數名稱加括號，括號內則須放入變項或條件等，多數情況可與四則運算搭配使用。例如：SUM([人數])、SUM([人數])/COUNTD([ID]) 等等。以下將分別簡介 Tableau 主要的函式類型：

1. 常用數字函式語法簡介

數字函式語法可供回傳絕對值、最大值、最小值等，以下分別說明用以回傳絕對值的 ABS、用來取整數值的 ROUND，以及回傳最大值和最小值的 MAX 和 MIN。

語法說明：ABS(number) 或 ABS([欄位變項])
回傳絕對值，可用來回傳特定數值或特定變項之絕對值 例如：ABS(-2) = 2 或 ABS([歷時變化]) = [變化幅度]

ABS 函數可用於數值或欄位變項。舉例而言，上述函式 ABS([歷時變化]) 可用在了解歷時變化的幅度。也就是說，當我們想要了解某項產品的銷售金額或學生成績的變化幅度，但不在意其變化是增是減時，可利用「歷時變化」之欄位變項，針對每項產品，計算其銷售金額在某段時間的變化（即成長幾元或減少幾元），或針對每位學生，計算其成績的變化（即進步幾分或退步幾分），再運用 ABS 來回傳此歷時變化正、負值的絕對值，以得到其變化幅度。

語法說明：ROUND(number, [decimals])
回傳捨去至指定的小數位數或整數值。 例如：ROUND([特定變項])

ROUND 函數用於取整數或取特定位數的小數，若將其函式省略 [decimals]，則表示取整數。舉例而言，若我們希望將學期成績取至整數位，則可透過 ROUND([學期成績]) 來捨棄整數之後的小數。

語法說明：MAX(number, number) 與 MIN(number, number)
回傳較大或較小值。 例如：MAX(4,7) MIN([變項 A],[變項 B])

而 MAX 與 MIN 函數則是用來回傳較大或較小值。其中，MAX 與 MIN 只能比較兩個相同類型的變項，如：兩者皆為數值。上述實例中，MAX(4,7) 將回傳 7。而假設我們欲找出班上學生的國語和英文成績中較低的成績時，可利用 MIN([國語], [英文]) 的函式語法，找出國語和英文兩欄位中較低的成績。

2. 常用字串函式語法簡介

字串函式可供找出是否包含特定用字、取代部分文字等，有助批次處理文字字串資料。以下分別說明可用來找出是否包含特定用字的 CONTAINS，以及用來取代部分文字的 REPLACE。

語法說明：CONTAINS(string, substring)
若字串 string 中包含 substring，則回傳 true 例如：CONTAINS('Librarians', 'Lib') = true

上述實例中，CONTAINS 語法協助我們找出 Librarians 一字中，是否包含 Lib，得到結果是 true，表示 Librarians 一字包含 Lib。此例看似簡單，但當我們的整份資料包含大量的文字字串時，即可發揮此語法之效用。

語法說明：REPLACE(string, substring, replacement)
在 string 中查找 substring，若查得，則將其取代為 replacement；若未查得，則加以保留 例如：REPLACE('1st Edition', '1st', '2nd') = '2nd Edition'

上述實例中，REPLACE 函數可幫助我們確認 1st Edition 中是否包含 1st，若包含 1st，則將 1st 取代為 2nd。當我們發現資料中錯誤或需要調整之處，我們可運用 REPLACE 的方式，批次找出所有問題，並加以取代。就本例而言，我們發現原來資料中的 1st Edition 應為 2nd Edition，即可以此函式來處理。

3. 常用日期函式語法簡介

日期函式可供截取部分日期、計算日期／時間區間等。以下介紹兩個常見的日期函式語法。

語法說明：DATENAME(指定日期之部分， #yyyy-mm-dd# 或特定日期變項 , [start_of_week]
以字串的形式回傳該日期之部分，可指定 year、month、week、day 或 hour。 Tableau 預設每週由週日起算，若欲修改可自行設定。 例如：DATENAME('year', #2019-01-15#) = '2019' 例如：DATENAME('month', #2019-01-15#) = 'January' 例如：計算日期變項 [Date] 為該月的第幾週 DATENAME('week', [Date], 'monday') - DATEPART('week',DATETRUNC('month',[Date]),'monday') + 1

DATENAME 函數是用來截取部分日期。上述實例中，第一、二例說明了當我們僅希望截取該日期的年份或月份時，可以透過此語法，去除不必要的細節。而第三例則是當我們想要了解該日期是該月第幾週的計算方式。

語法說明：DATEDIFF(date_part, date1, date2, [start_of_week])
計算兩日期或時間之差 例如：以週起始日（此例設定為週一起算），計算兩日期相差幾週 DATEDIFF('week', #2019-01-20#, #2019-01-22#, 'monday')= 1 其中，2019-01-20 為週日，2019-01-22 為週二，以週一起算相隔一週 若以 2019-01-22 週二與 2019-01-25 週五來計算，則將回傳 0 例如：計算兩日期變項 date1 與 date2 相差幾天 DATEDIFF('day', [date1], [date2]) 例如：計算兩時間變項 hour1 與 hour2 相差幾小時 DATEDIFF('hour', [hour1], [hour2])

DATEDIFF 函數是用來計算兩日期或時間之差。假設我們欲知某事件的始末日期相差幾週、相差幾天，甚或相差幾小時，則可運用此函式語法來計算。

4. 常用邏輯函式語法簡介

邏輯函式可供設定條件，針對特定變項新增變項以重新歸類等。以下介紹兩個常見的邏輯函式語法。

語法說明：CASE 針對特定欄位變項設定條件，並回傳相對應之值

```
CASE [ 欄位變項 ]
WHEN 'value1' THEN 'return1'
WHEN 'value2' THEN 'return2' ...
END
```

文字才須加上雙引號

例如：欲將 [Type] 中的 A 和 B 歸為第 1 類、C 和 D 歸為第 2 類

```
CASE [Type]
WHEN 'A' THEN 1
WHEN 'B' THEN 1
WHEN 'C' THEN 2
WHEN 'D' THEN 2
END
```

CASE 是資料處理中相當常用的。當我們欲針對特定欄位變項中的內容（值）重新歸類時，CASE 就是最好的幫手。舉例而言，當我們欲將年度變項之 2011、2012、2013、2014、2015 年歸入「2011-2015 年」一類，2016、2017、2018、2019、2020 年歸入「2016-2020 年」一類，則可套用上述實例的語法。

語法說明：IF

針對一系列表述，當表述成立時，則回傳 THEN 之後的值

```
IF <expr> THEN <then>
[ELSEIF <expr2> THEN <then2>...]
[ELSE <else>]
END
```

例如：欲以學校收受學生與實際註冊人數來衡量其表現

```
IF [Enrolled total] = [Admissions total] THEN 'great'
ELSEIF ([Admissions total]-[Enrolled total]) <
[Admissions total]*.6 THEN 'fair'
ELSE 'poor'
END
```

雖然 CASE 和 IF 某種程度都可協助我們「歸類」，但 CASE 是針對某個特定欄位變項中的值來重新歸類（前述實例是將單一欄位變項中的內容進行合併，以重新分類），而 IF 則是針對幾個欄位變項的關係，設定的條件敘述來分級歸類。在這個實例中，我們係以本單元的 IPEDS 教育統計資料來設想，當我們希望透過一所學校收受學生的人數與其實際註冊人數的情況來衡量該校表現時，

可運用上述實例的 IF 語法，將學校表現分級。此函式語法的表述係將學校表現粗分為三級：1. 註冊人數等於受收人數之學校，視為表現優異者；2. 收受人數與註冊人數相差不及收受人數六成者，視為普通；3. 其他則視為不理想。

小結

本節介紹了如何運用 Tableau 進行資料視覺化，以及 Tableau 的基本程式語法。一般說來，將資料匯入 Tableau 後，要記得確認資料格式設定正確（尤其是地理資訊），以利 Tableau 較準確地為我們選擇合適的圖形呈現方式。在工作表中，只要將欲用來繪圖的欄位變項置於欄與行（列），並選擇合適的圖形呈現方式，即可簡易製圖。

除了進一步客製化顏色、標籤等，善用 Tableau 的過濾篩選及相關分析功能，皆有助提升資料探索與呈現的深度。運用過濾篩選功能將有助聚焦特定項目之比較；運用平均數或四分位差等參照功能，則有助理解數值的意涵。而無論是提供比較或幫助理解，皆是資料視覺化的核心價值。

Tableau 的函式語法主要包括：數字、字串、日期與邏輯四大類。其中，數字函式可回傳絕對值、取整數、取較大或較小值等；字串函式可尋找與取代部分文字等；日期函式可截取部分日期或計算日期／時間等；邏輯函式則可協助將欄位變項重新歸類或依設定條件進行分級等。若可進一步運用函式語法，則可更深入地探討相關問題，運用新增的運算欄位，做更多、更有意義的視覺呈現。下節（第四節）將進一步以 CASE 的應用，介紹製作拼貼地圖（small multiple maps）的作法。

7-4 地圖資料視覺化

本節將介紹地圖資料視覺化基本技巧，我們將以臺北市政府開放資料的自行車竊案資料為例，首先，將說明單一地圖的製作方式，接著，再說明拼貼地圖（small multiple maps）的製作方式。

單一地圖的製作

請參考第三節的匯入檔案的作法，將「自行車竊案 .xlsx」檔案匯入 Tableau 中。第三節曾強調匯入資料時，須確保每個欄位格式設定正確。由於我們將製作地圖，地理資訊是否設定正確是非常重要的步驟。

以自行車竊案的課堂練習檔案為例，郵遞區號、經度、緯度、縣市和行政區皆須設定為地理角色，其中，由於臺灣的地理角色層級和美國不同，對應上難免有些出入。我們可將縣市設定為「城市」、行政區設為「郡 / 縣」。不過，無論你將縣市與行政區設定為地理角色中的哪個層級，最重要的是先確保這些地理資訊設定為地理角色（圖 7-38）。如此才能讓 Tableau 判定我們接下來可以繪製地圖。

圖 7-38 確保資料格式設定正確

繪製地圖

地理角色設定完成後，我們就可以進入工作表，開始繪製地圖了。在這個練習中，我們欲繪製一張臺北市自行車竊案地圖，並以顏色來表示竊案數的多寡。

雖然繪製地圖的方式很多。此例中，我們先快速雙擊地理變項 [郵遞區號]，讓 Tableau 先快速為我們產生底圖。接著，將 [記錄數] 拖曳至標記區的顏色（圖 7-39）。

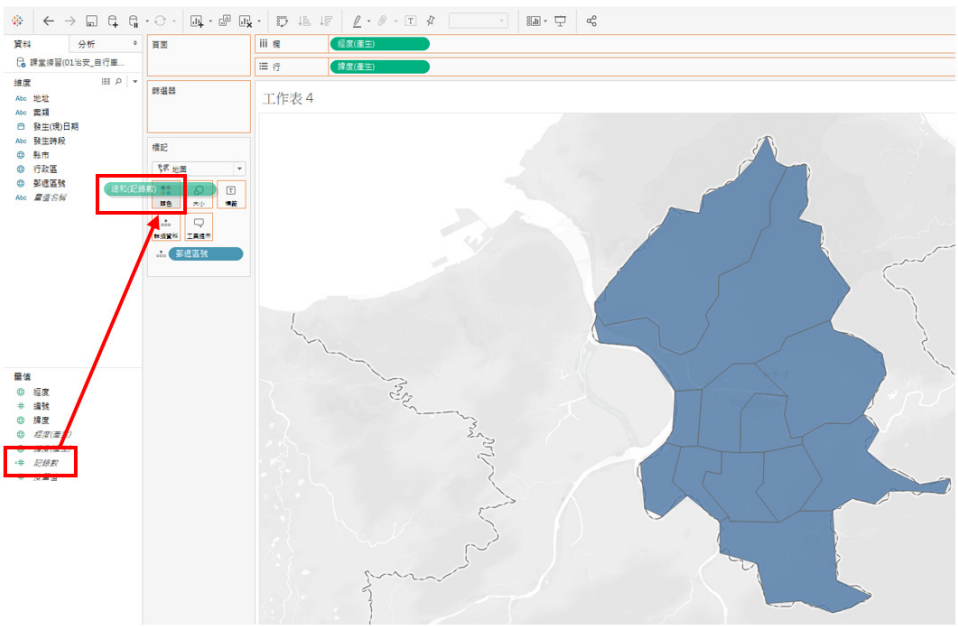


圖 7-39 以郵遞區號快速繪製的底圖

此時，若 Tableau 沒有為我們選擇填色地圖，而是以點狀呈現各行政區位置時，則可參考第三節選擇圖形呈現方式的作法，運用「顯示」（Show Me）功能，點選我們欲繪製的地圖類型。如此，我們便可快速完成一張簡單的地圖了。

然而，我們可能希望進一步客製化地圖的顏色與邊界，甚至加上標籤說明。這些客製化的作法亦如第三節所述，我們可在標記區中，設定顏色及邊界。若要取消邊界，可在標記區的「顏色」中，將「邊界」由「自動」改為「無」（圖 7-40）。

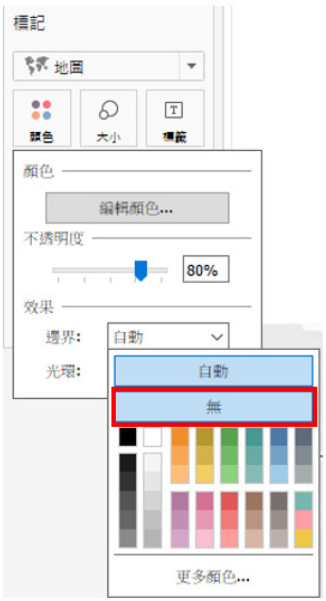


圖 7-40 去除行政區的邊界

若我們要新增地圖標籤，說明行政區的位置分布，則可將「行政區」拖曳至標籤。然而，此時，若 Tableau 無法辨識部分地區，則會出現圖 7-41 的現象。原本完整的臺北市地圖，因為加上 Tableau 無法辨識的行政區標籤而缺了幾塊。此時，則須透過「編輯位置」的方式，協助 Tableau 辨識這些行政區。

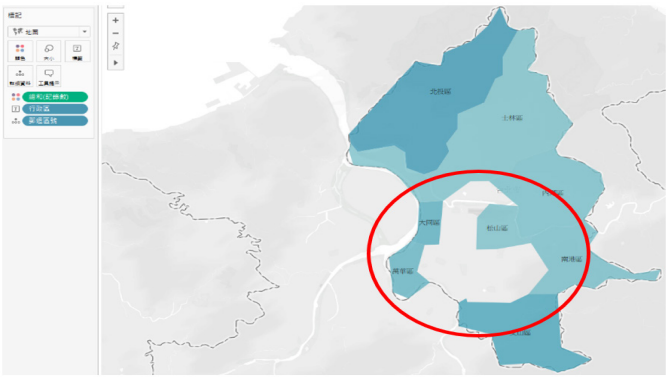


圖 7-41 Tableau 無法辨識部分行政區

我們可透過工具列上的「地圖」，透過「編輯位置」的功能，將「州 / 省」設定為「Taipei City」，確保 Tableau 可辨識地理行政區，以解決前述問題（圖 7-42）。

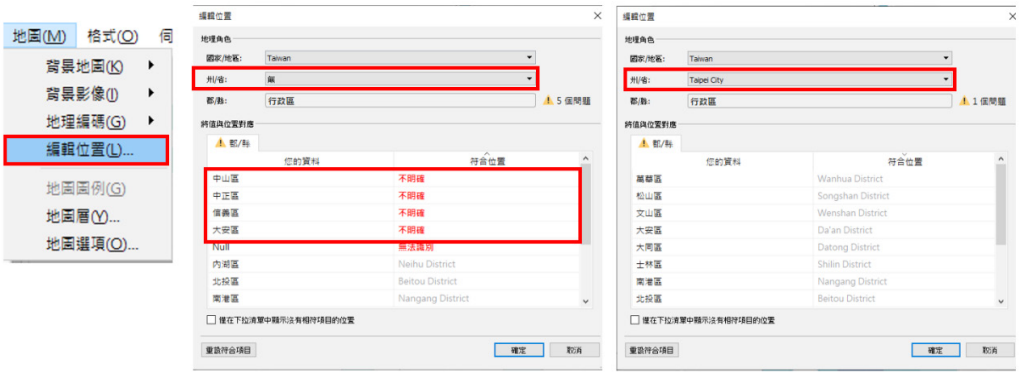


圖 7-42 編輯位置

由此看出，Tableau 將臺北市視為「州 / 省」。因此，若在設定資料時，直接設定為「州 / 省」，則在加入標籤時，就不會出現圖 7-41 的現象了。然而，編輯地理位置是運用 Tableau 繪製地圖的一項重要功能，雖然 Tableau 版本不斷更新進步，對於臺灣地圖的辨識也愈來愈理想，但繪製地圖難免遭遇 Tableau 無法辨識地理位置的問題，故前段刻意如此安排，讓大家掌握因應的作法。

如此便完成一簡單的臺北市自行車竊案地圖了（圖 7-43）。

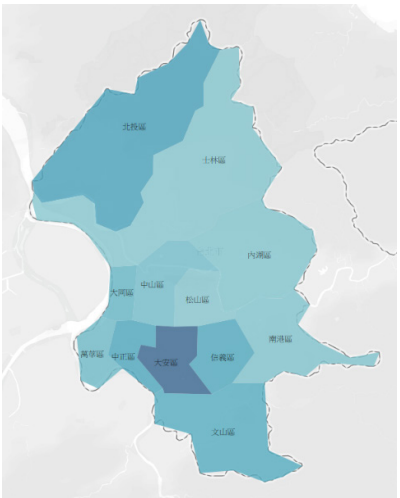


圖 7-43 完成單一地圖

Small Multiple Maps 的製作

第二節談到資料視覺化的基本原則時，我們提到資料視覺化往往希望盡量納入更多變項 / 面向的資料，也提到在非必要時，避免使用立體圖。然而，我們要如何在二維的平面空間中呈現納入許多變項呢？ Small multiples 將許多小圖拼貼成一大圖的技巧運用往往可以幫助我們達成上述的目標。因此，Small multiples 是資料視覺化中，常用的技巧之一。

Small multiples 的中文譯名不一，有人將之譯成「組圖」，有人譯成「分割圖（表）」，也有人譯成「倍製圖」。由於這些名稱似乎不如 small multiples 來得貼切，故本節姑且延用原文，並以較直白的「拼貼」來描述其作法。接下來，就讓我們一起來製作 small multiple maps 吧。

以 Tableau 製作 small multiple maps 的兩種常見作法，一是在每個分頁中做一張地圖，再新增一儀表板（dashboard），並將前述各圖拼貼至儀表板；一是利用單一分頁製作並排列所有的小地圖。本節將運用較簡單的方式，在單一的工作表分頁中排列小地圖。

排列小地圖的常見方式也有兩種：一是運用既有的欄位變項，以 A 變項為欄、B 變項為列；另一是新增用來指定小地圖排列位置的欄與列變項，直接指定每張小地圖要排在第幾欄、第幾列的位置。由於後者的函式語法相形複雜些，且直接運用既有變項更滿足資料視覺化納入更多變項的原則，因此，本節針對第一種方式作簡介。

繪製 small multiple maps 的準備工作

假設我們欲針對年度與最常發生的 5 個時段來繪製 5X5 的小地圖，由於既有的 [發生 (現) 日期] 已包含年度資訊，故僅須新增 [好發時段] 變項，以指定 5 個好發時段，並用以此二欄位變項來排列地圖（圖 7-44）。

	2014	2015	2016	2017	2018
好發時段 1					
好發時段 2					
好發時段 3					
好發時段 4					
好發時段 5					

圖 7-44 地圖的排列示意圖

此時，我們可利用邏輯函式語法來設定五個好發時段。而為了找出哪五個時段為好發時段，我們亦可充分運用資料視覺化探索資料的方式，透過長條圖找出最常發生的五個時段。圖 7-45 是在新的分頁中，快速製作一張以發生時段為欄、記錄數為行（列）的長條圖，並加以排序。

圖 7-45　運用長條圖找出五個好發時段

CASE 函式語法的應用

在維度欄位資料區按右鍵，選擇「建立計算欄位」，並輸入函式語法（圖 7-46）。

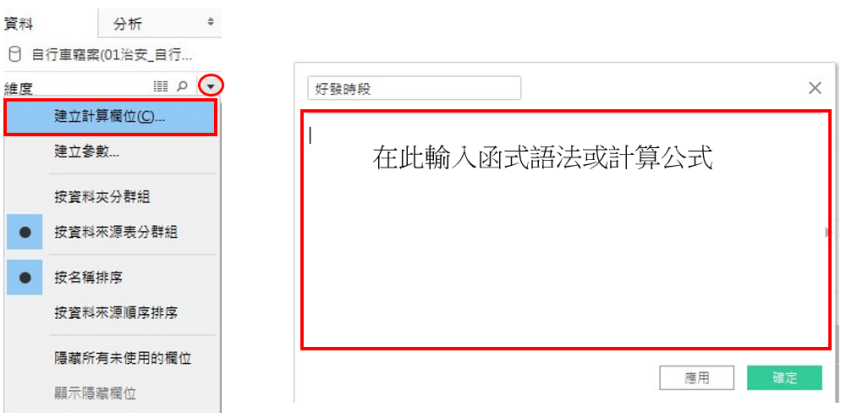


圖 7-46 建立計算欄位

在新欄位名稱輸入「好發時段」，並利用 Case 函數，針對 [發生時段] 設定欲排入前五的好發時段，函式語法內容如圖 7-47，指定當發生時段為「16~18」時，則設定為好發時段的「Top1 16~18」; 發生時段為「10~12」，則設定為好發時段「Top 2 10~12」，依此類推，將圖 7-45 的前五名時段設定完成。CASE 函式完成時，其語法必須記得加上 END 作結。

圖 7-47 運用 CASE 函式新增「好發時段」

繪製 Small Multiple Maps

目前我們已備妥要用來排列小地圖的欄位了，接下來，我們直接在已繪製好的地圖工作表中，將 [發生 (現) 日期] 拖曳至欄，[好發時段] 拖曳至行（列）。若要保留第四節前半段所欲繪製的單一地圖，則可複製工作表（可參考第三節管理分頁的說明，在工作表頁籤上按右鍵，選擇「重複」即可），在新的工作表中進行上述步驟，即可快速繪製出 small multiple maps。

然而，由於竊案並非時刻發生且有些年度的資料可能未必完整，我們以 small multiple maps 打散資料後，部分小地圖的案例更顯稀少，沒有案例的地圖就包含了許多空白。為了讓 small multiple maps 更有效呈現竊案發生數較高的年度和時段，我們可以運用過濾篩選功能，過濾欄與行（列）中的缺值（Null），以排除較不明顯的好發時段與資料過於不全之內容。

由於 2013 的資料較不全，我們可點選 [發生 (現) 日期)] 欄位變項的小箭頭，選擇「篩選條件 ...」，並取消勾選 2013；同樣地，我們可點選 [好發時段] 變項的小箭頭，選擇「篩選條件 ...」，並取消勾選 Null（圖 7-48）。

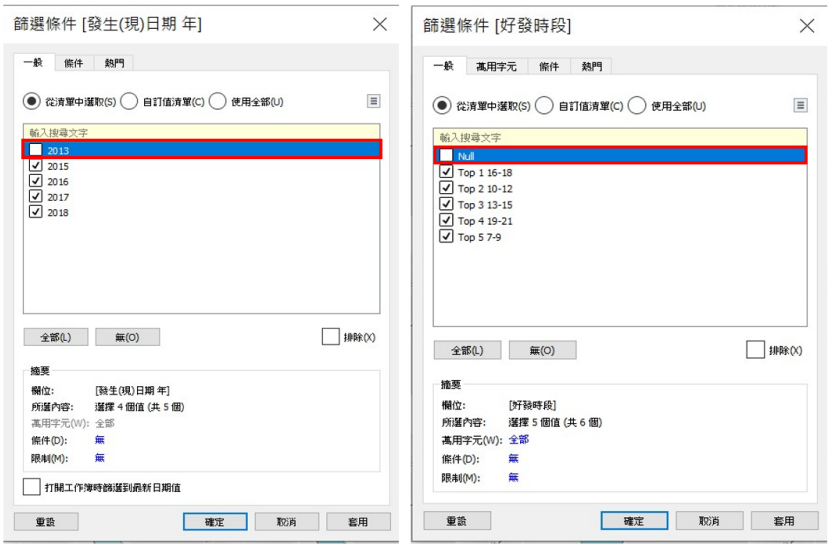


圖 7-48 過濾排除資料不全的部分

如此，我們就初步完成了臺北市自行車竊案的 small multiple maps 了（圖 7-49）。

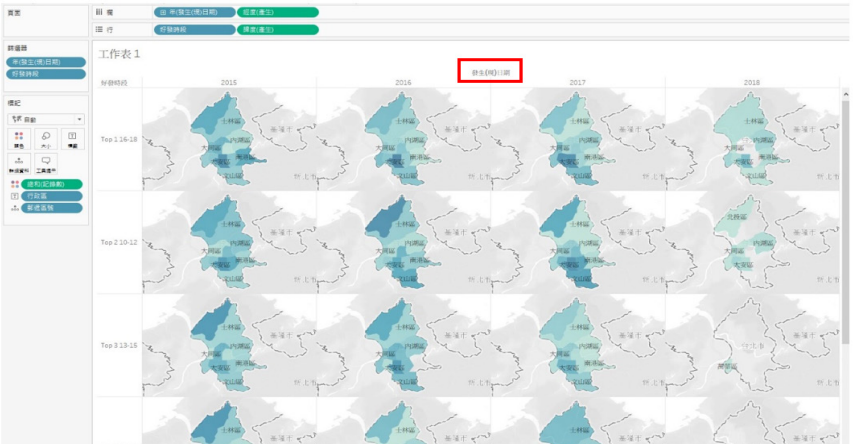


圖 7-49 初步完成 small multiple maps

地圖繪製小技巧

1. 隱藏欄位標籤

如果覺得圖 7-49 的「發生 (現) 日期」顯得多餘，我們可在欄與行（列）標籤上按右鍵，選擇「隱藏欄位標籤」（圖 7-50）。

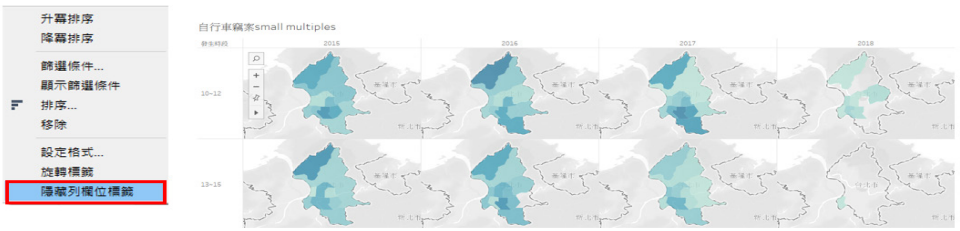


圖 7-50 隱藏 Small Multiple Maps 欄與行（列）標籤

2. 新增註解說明文字

我們亦可針對每張小圖作註解說明。只要在小地圖上按右鍵，選擇「新增註解」、「區域」，即可針對此區塊增加文字說明（圖 7-51），亦可設定說明文字的字型、顏色等格式。

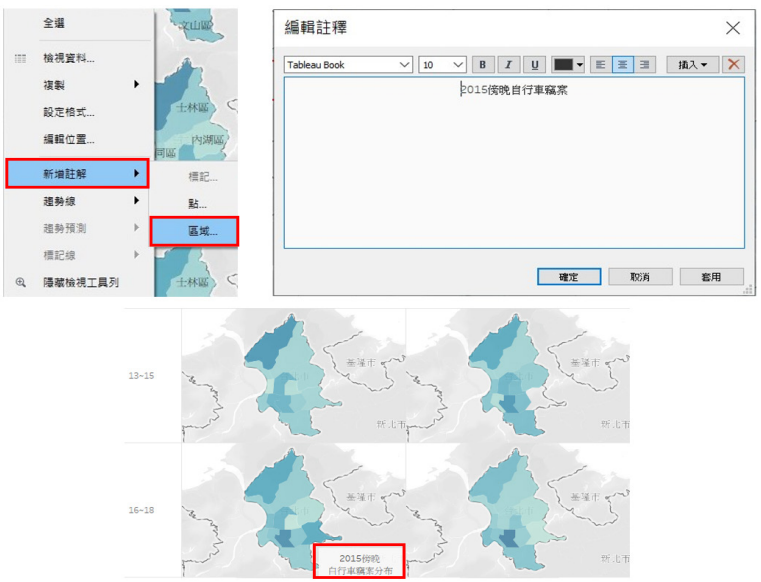


圖 7-51 新增註解說明文字

3. 調整地圖大小與位置

若要調整地圖的大小，可將滑鼠移至地圖上，此時，左上角的工具列即出現（圖 7-52），即運用＋、－作放大、縮小等調整。若要調整地圖位置，則可透過＋、－下方小箭頭的工具列，點選平移功能（即左數第二個垂直雙向的十字圖示  ）來移動地圖。若要截取部分地圖，則可利用最右側的三個圖示  ，選擇矩形、圓形或其他區塊，再作保留或刪除。

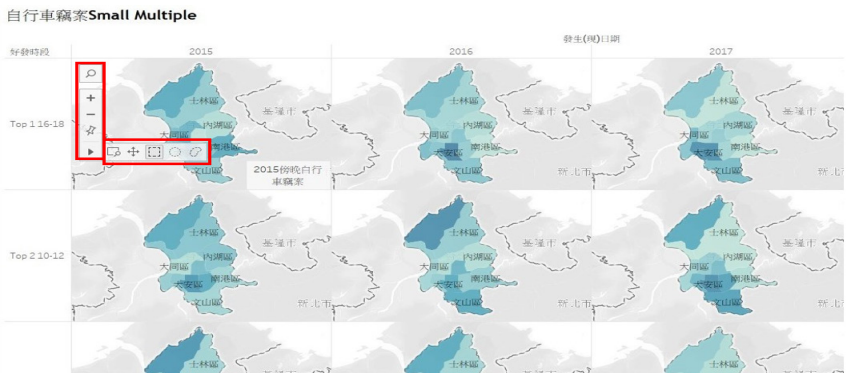


圖 7-52 調整地圖大小比例

4. 新增比例尺

若要新增比例尺，則可透過工具列的「地圖」，開啟「地圖選項」，再勾選「顯示地圖比例尺」即可（圖 7-53）。

圖 7-53 新增比例尺

5. 設定地圖層

最後，若要修改目前的地圖層，亦可透過工具列的「地圖」開啟「地圖層」選項（圖 7-54），即可調整圖層與地圖樣式。地圖層的視窗會開啟在原本資料區的位置，其中，背景樣式深淺有三種選擇，亦可選擇街道圖或衛星圖；地圖層則有許多選項可勾選，主要包括：土地覆蓋情形是否顯示地形或海岸線等，各種邊界是否顯示、各種地理行政區標示名稱等等。完成設定後，點選右上角的叉號，即可關閉此功能。

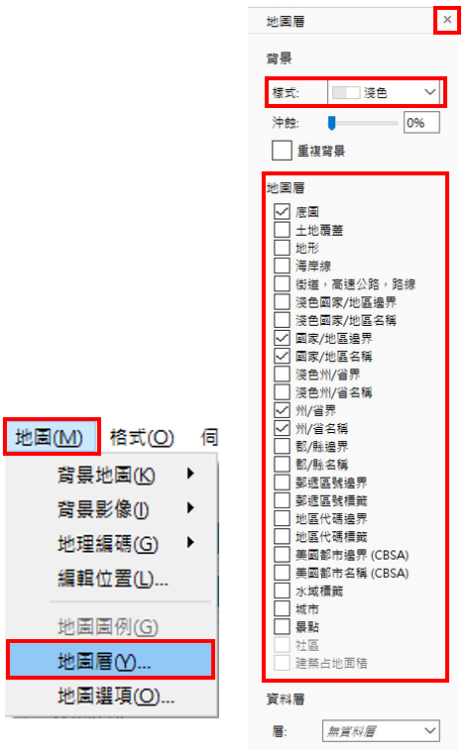


圖 7-54 地圖層的設定

製作互動式地圖

在 Tableau 中可以製作互動圖。本節最後將介紹互動圖的兩種簡單作法：一是運用「顯示篩選條件」的功能，一是運用「頁面」的功能。

第一種方法是針對欲用來互動瀏覽的欄位變項，點選其右側小箭頭，再點選「顯示篩選條件」，右側將出現篩選條件，即可供勾選互動瀏覽。圖 7-55 是針對一張以年度 [發生 (現) 日期] 和 [發生時段] 製作的臺北市自行車竊案 small multiple maps，並依 [發生時段] 建立互動瀏覽的作法。

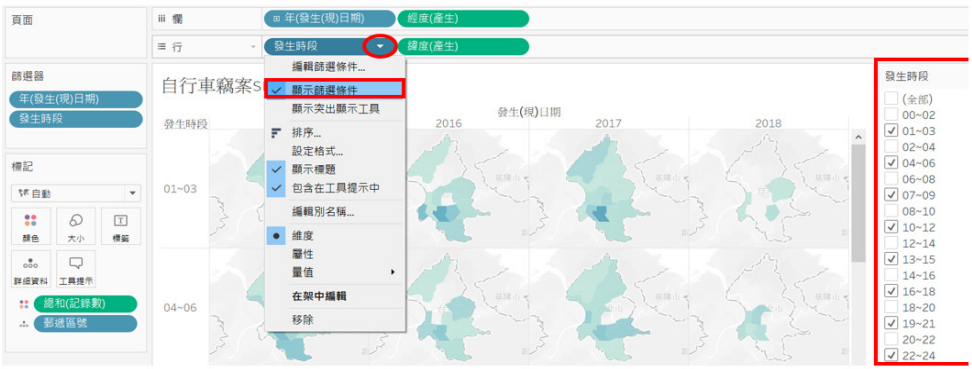


圖 7-55 運用篩選條件製作互動圖

第二種方法則是利用工作區左上方的「頁面」功能來製作互動式地圖。我們若將 [發生 (現) 日期] 拖曳至「頁面」，即可手動分年瀏覽檢視或自動播放歷年變化。然而，本例已將年度分成小地圖呈現，依年瀏覽意義較不大，故我們可在臺北市自行車竊案的單一地圖中，嘗試這項功能。圖 7-56 是將 [發生 (現) 日期] 拖曳至「頁面」後的互動圖。

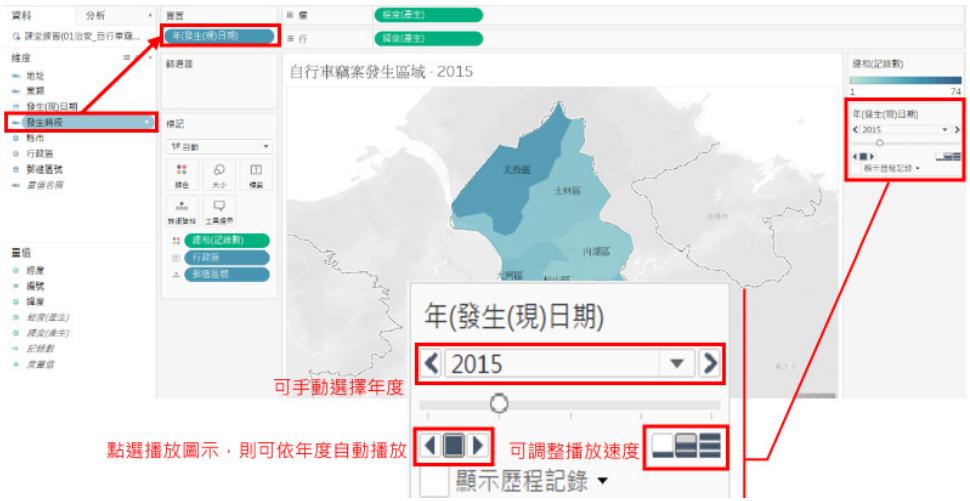


圖 7-56 運用頁面功能製作互動圖

此互動圖中，若手動點選年度（圖 7-56 的 2015）旁的左右箭頭，則可一一檢視不同年度的地圖，或以下拉式選單直接檢視特定年度。若要自動播放，則可點選左下方的左右播放按鍵，讓 Tableau 自動播放。其中，點選向右播放鍵，Tableau 將從 2015 播放至 2018 年；點選向左播放，則將倒序從 2018 年播放至 2015 年。而右下側的三個按鍵代表播放速度，一條線的按鍵是最慢的速度，可自行調整。

最後，在 Tableau 中，快速調整日期呈現單位的小技巧也非常值得一提。就讓我們透過微調先前製作的 small multiple maps，說明調整日期呈現單位的方法。

若我們欲將原本 small multiple maps 中的以「年」為單位呈現的 [發生 (現) 日期] 改以「季」為單位來呈現，並用「年」來互動瀏覽，可以點選 [發生 (現) 日期] 右側的小箭頭，並選擇以「季」呈現（圖 7-57）。如此就可快速讓 small multiple maps 改為以第一至四季為欄、以五大好發時段為列來呈現了。接著，我們將 [發生 (現) 日期] 再次拖曳至「頁面」，即可完成以「季」為 small multiples 底圖並支援分「年」瀏覽的互動圖了。

圖 7-57 調整日期呈現單位並製作互動圖

小結

本節介紹了單一地圖與拼貼地圖的製作，也應用了 CASE 函式語法，根據自行車竊案的發生時段，新增一欄位來描述其好發時段。地圖的製作須留意的是欲用來繪製地圖的地理資訊是否設定為地理角色，而倘若 Tableau 無法辨識地理位置，則須透過「編輯位置」的功能來解決此問題。此外，本節也介紹了幾項地圖製作的技巧與互動圖的製作方式等。其中，互動圖的製作與日期的呈現單位亦可應用於各類型的圖形呈現中，皆是 Tableau 相當重要的功能。

雖然本章三、四節著眼於 Tableau 此一資料視覺化軟體，Tableau 也確有其強大功能，然而，沒有任何一項工具是資料視覺化的唯一選擇。藉由介紹此功能豐富的視覺化軟體，希望讀者認識資料視覺化之基本作法，相信學會使用一套工具之後，即可大致掌握資料視覺化的基本操作流程與觀念，未來運用其它工具也能更快速地上手。

參考資料

1. 色盲。教育百科。檢自 http://163.28.84.216/Entry/WikiContent?title=%E8%89%B2%E7%9B%B2&search=%E8%89%B2%E7%9B%B2

2. Abela, A. (2009). Choosing a good chart. Retrieved from http://extremepresentation.typepad.com/files/choosing-a-good-chart-09.pdf

3. Color Blind Awareness. Retrieved from http://www.colourblindawareness.org/

4. Few, S. (2009). Now you see it: Simple visualization techniques for quantitative analysis. Oakland, Calif: Analytics Press.

5. Few, S. (2012). Show me the numbers: designing tables and graphs to enlighten (2nd ed.). Burlingame, Calif: Analytics Press.

6. Kirk, A. (2016). Data visualisation: a handbook for data driven design. Los Angeles: Sage Publications.

7. Sleeper, R. (2018). Practical Tableau: 100 tips, tutorials, and strategies from a Tableau zen master. Sebastopol, CA: O'Reilly Media.

8. Tableau Learning. Retrieved from https://www.tableau.com/learn

9. Tufte, E. (2006). Beautiful Evidence. Graphics Press.

10. Weinschenk, S. (2011). 100 things every designer needs to know about people. Berkeley, CA: New Riders.

11. Yau, N. (2013). Data Points: Visualization That Means Something, Wiley.

Chapter 8

資料科學導論

/ 洪暉鈞

因應人工智慧的發展，Python 成為資料科學中的首選工具，在各應用領域都有大量方便實用的套件。本章介紹 Python 中兩大重要套件 numpy 與 pandas，本章共有兩小節，包括：運用 array 處理一維陣列與多維矩陣運算以及運用 DataFrame 處理結構化具列索引與欄標的二維資料集及檔案輸出入。學習完本章後，讀者將學會讀取檔案資料成資料表，處理遺漏值與重複值，以及對資料表的合併、分割與轉置等操作。

8-1 科學運算 NumPy & pandas

在 Python 在處理資料科學相關運用的時候，有兩個重要的套件，一個是 NumPy，一個是 pandas。本小節將依序介紹 NumPy 以及 pandas 的重要功能與使用情境。

NumPy

NumPy 是 Python 中有關數值計算最重要的套件，其他許多提供科學計算的套件都是基於 NumPy 的陣列作為基礎。NumPy 的資料結構可以處理一維或多維矩陣運算，稱為陣列 (array) 物件。陣列可以是一維或多維度的，我們可以針對這種陣列資料統一執行一些數學運算，所以陣列之中的每一個元素都必須是相同型態，也讓程式執行更有效率，更適合用於數學運算與資料較龐大時的運算。

程式 E8-1-1 將建立一維跟二維的陣列，並且示範陣列常見的屬性，像是陣列的 .ndim 會回傳該陣列的維度；.shape 與 .size 則會回傳陣列的列行數與資料個數。

E8-1-1.py

```
01  import numpy as np      # 使用 numpy 採用 np 簡寫
02  array1 = np.array([1,2,3])   # 一維的 array
03  array2 = np.array([[1,2,3], [4,5,6]]) # 二維的 array
04  print(array2)
05  print('維度:', array2.ndim)
06  print('列行:', array2.shape)
07  print('個數:', array2.size)
```

執行結果：

```
[[1 2 3]
 [4 5 6]]
維度: 2
行列: (2, 3)
個數: 6
```

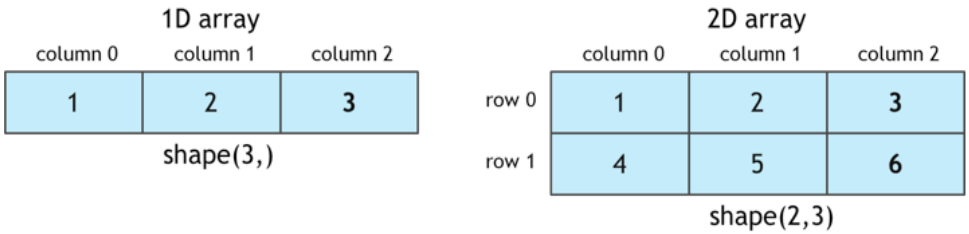


圖 8-1 numpy array

程式 E8-1-2 介紹建立一維陣列的不同方法，除了用 np.array 加上自定義的 list 以外，亦可以用 np.arange()、np.zeros()、np.ones() 等方式批次產生一維的 array。

E8-1-2.py

```
01 import numpy as np
02 a = np.array([3,6,9,12,15,18])
03 b = np.arange(6)   # 建立一個 size 為 6 的陣列，數據為 0~5
04 c = np.zeros(6) # 數據全為 0，size 6
05 d = np.ones(6, dtype = np.int)# 數據為 1，size 6，型態整數
06 e = np.random.random(6)#size 為 6 的陣列，隨機亂數範圍為 [0.0,1.0]
07 print(a,b,c,d,e, sep='\n')
```

執行結果：

```
[ 3  6  9 12 15 18]
[0 1 2 3 4 5]
[ 0.  0.  0.  0.  0.  0.]
[1 1 1 1 1 1]
[ 0.9038974   0.19673665  0.8186001   0.13724558  0.05995684  0.9762709 ]
```

當有兩個陣列需要一起運算，如果這兩個陣列的個數相同，可以省略掉迴圈的使用而讓陣列每個元素進行相同的運算。例如程式 E8-1-3 中，我們希望能產生陣列 b 的內容是 0-5 共六個數字，然後陣列 b 的內容減掉陣列 a。此時 b-a 就會把這兩個陣列內六個對應的元素相減，產生新的陣列結果。不限定減法，加乘除或其他運算也可以，要特別注意的是這樣的運算必須是針對相同元素數量的陣列才可以。

E8-1-3.py

```
01  import numpy as np
02  a = np.array([3,6,9,12,15,18])
03  b = np.arange(6)
04  f = b-a
05  print(f)
06  print(f<-9)
07  print(f==-9)
```

執行結果：

```
[ -3  -5  -7  -9 -11 -13]
[False False False False  True  True]
[False False False  True False False]
```

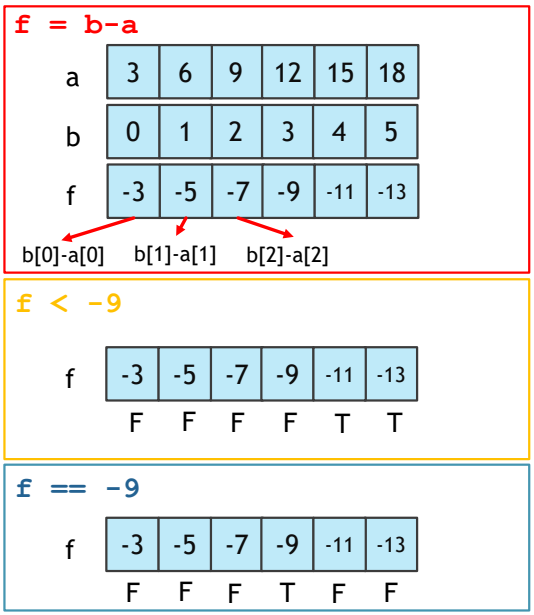



圖 8-2 一維 array 運算

如同一維陣列，我們也可以產生多維陣列。下面的例子就是我們產生不同的 2X3 的二維陣列，並且進行二維陣列中行、列的運算加總。

E8-1-4.py

```
01  import numpy as np
02  a = np.array([[3,6,9],[12,15,18]])
03  b = np.arange(6).reshape(2,3)
04  c = np.zeros((2,3))   # 數據全為 0
05  d = np.ones((2,3), dtype = np.int)   # 數據為 1，6 個
06  e = np.random.random((2,3))   # 亂數，6 個
07  print(a,b,c,d,e, sep='\n')
08  print(np.sum(a))
09  print(np.sum(a, axis=1))
```

執行結果：

```
[[ 3  6  9]
 [12 15 18]]
[[0 1 2]
 [3 4 5]]
[[ 0.  0.  0.]
 [ 0.  0.  0.]]
[[1 1 1]
 [1 1 1]]
[[ 0.59331307  0.94254093  0.06424767]
 [ 0.92768566  0.19110781  0.56701524]]
63
[18 45]
```

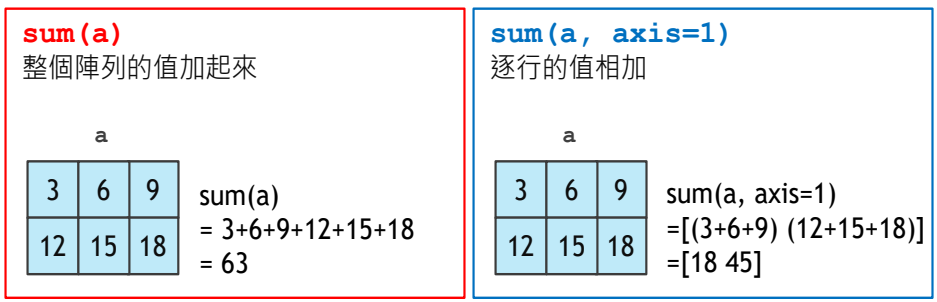


圖 8-3 二維 array 的 sum 運算

如同一維陣列，二維陣列也可以進行相對位置的運算 (加、減、乘、除)。

E8-1-5.py

```
01  import numpy as np
02  a=np.array([[2,4,6],[8,10,12]])
03  b=np.arange(6).reshape((2,3))
04  print(a)
05  print(b)
06  print(a-b)
07  print(a*b)
```

執行結果：

```
[[ 2  4  6]
 [ 8 10 12]]
[[0 1 2]
 [3 4 5]]
[[2 3 4]
 [5 6 7]]
[[ 0  4 12]
 [24 40 60]]
```

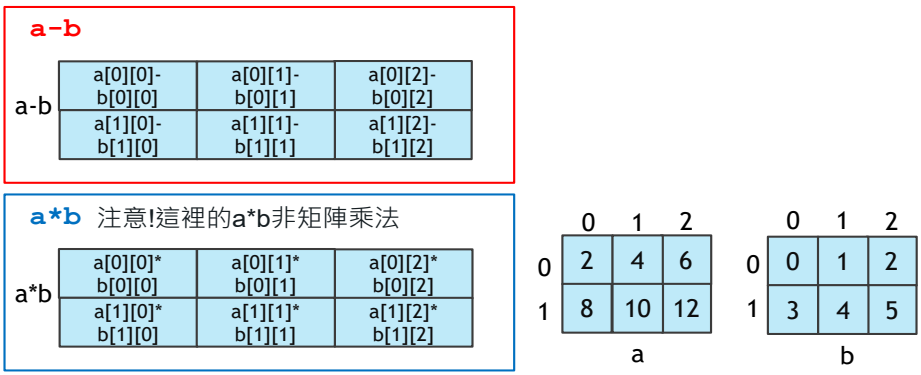


圖 8-4 二維矩陣減法與對應項乘法

而如果是要進行兩個矩陣之間的矩陣乘法，則須使用 np.dot() 來進行矩陣乘法。另外因為矩陣乘法的兩個矩陣行列數需要相反，所以下面的例子我們先將第二個矩陣進行轉置 (*transpose*)，將 2×3 的矩陣變成 3×2。

E8-1-6.py

```
01  import numpy as np
02  a=np.array([[2,4,6],[8,10,12]])
03  b=np.arange(6).reshape((2,3))
04  print(a)
05  print(b.T) # 轉置
06  print(np.dot(a,b.T))
```

執行結果：

```
[[ 2  4  6]
 [ 8 10 12]]
[[0 3]
 [1 4]
 [2 5]]
[[ 16  52]
 [ 34 124]]
```

			0	3
			1	4
			2	5
2	4	6	16	52
8	10	12	34	124

圖 8-5 二維矩陣轉置與矩陣乘法

兩個 array 之間除了運算，還可以進行合併成一個 array，以下範例就是透過 np.vstack() 以及 np.hstack() 進行垂直與水平的合併。

E8-1-7.py

```
01  import numpy as np
02  A = np.array([2,4,6])
03  B = np.array([8,10,12])
04  print(np.vstack((A,B)))   #vertical
05  print(np.hstack((A,B)))   #horizontal
```

執行結果：

```
[[ 2  4  6]
 [ 8 10 12]]
[2 4 6 8 10 12]
```

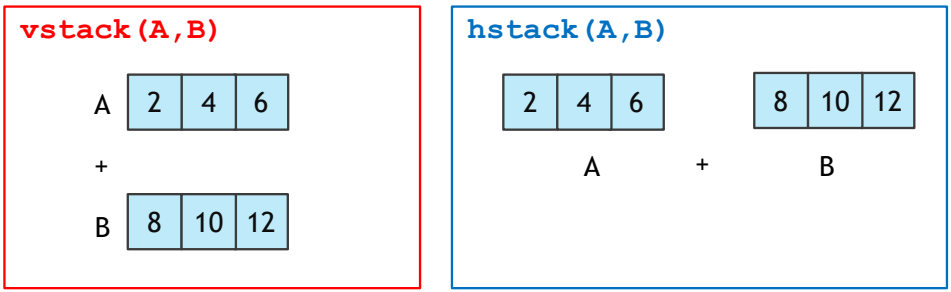


圖 8-6 array 垂直與水平的合併

同理，我們也可以將一個多維的 array，垂直或水平平均分隔成 2 個以上的 array，如下列程式所示：

E8-1-8.py

```
01  import numpy as np
02  a = np.arange(12).reshape((3,4))
03  print(a)
04  print(np.vsplit(a,3))
05  print(np.hsplit(a,2))
```

執行結果：

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
[array([[0, 1, 2, 3]]), array([[4, 5, 6, 7]]), array([[ 8,  9, 10, 11]])]
[array([[0, 1],
       [4, 5],
       [8, 9]]), array([[ 2,  3],
       [ 6,  7],
       [10, 11]])]
```

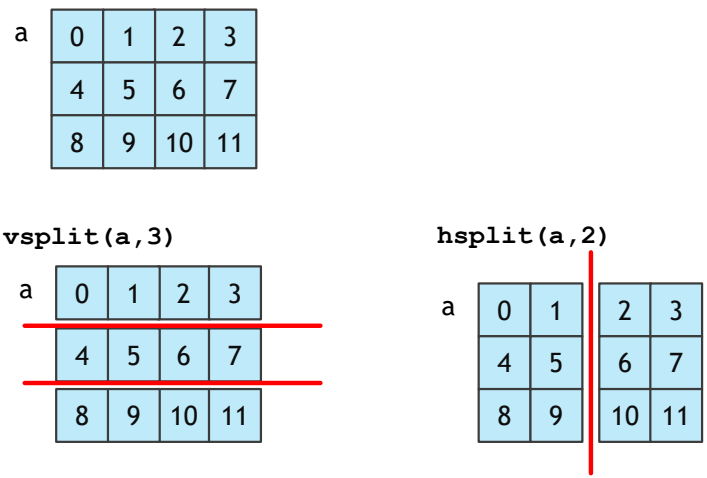


圖 8-7 array 切割

Pandas

pandas 是基於 numpy 的資料分析工具，能夠快速的處理結構化資料的大量資料結構和函數。接下來會跟大家介紹 pandas 之中常用的兩個資料結構：Series 跟 DataFrame。

Series 是由一組資料和一組索引組成，你可以視 Series 為有帶索引值的一維 array，以下示範建立一個 Series，可以看出他除了含有不同資料型態的內容，還帶有從 0 開始的索引值。

E8-1-9.py

```
01  import pandas as pd
02  import numpy as np
03  s = pd.Series([1,'abc','6',np.nan,44,1])
04  print(s)
```

執行結果：

```
0  1
1  abc
2  6
3  NaN
4  44
5  1
dtype: object
```

想一想：如果是 numpy 的 array 呢？

接下來我們來示範 pandas 的另外一個重要實用的資料結構 DataFrame。這是一個表格型的資料結構。有欄跟列的索引。DataFrame 可以用來處理結構化 (Table like) 的資料，有列索引與欄標籤的二維資料集，可以透過字典資料結構或是 array 來建立，但也可以利用外部的資料來讀取後來建立，例如： CSV 檔案、資料庫等等。

以下我們建立一個 DataFrame 的資料集，資料是隨機 (random)7×3 筆亂數。

E8-1-10.py

```
01  import pandas as pd
02  import numpy as np
03  # 生成方式一：用 array 的矩陣 (1)
04  df = pd.DataFrame(np.random.randn(7,3))
05  print(df)
```

執行結果：

```
          0         1         2
0  1.238052 -1.034129  0.222476
1 -0.457043 -1.252934  0.955214
2 -0.476281 -1.509945  0.573486
3  1.046944  0.143649 -0.138385
4 -1.268997  0.033301 -0.989308
5 -0.572510 -0.950910 -0.086820
6  0.250895 -0.889284 -0.149622
```

在我們建立好二維的資料表後，我們進一步在這已建立好的 DataFrame 加上列索引與欄標籤。

E8-1-11.py

```
01  import pandas as pd
02  import numpy as np
03  # 生成方式一：用 array 的矩陣 (2)
04  eat = np.random.randint(10,size=(7,3))*5+50
05  dates = pd.date_range('20170812',periods=7)
06  df0 = pd.DataFrame(eat)
07  # 加上欄位
08  df1 = pd.DataFrame(eat, index=dates, columns=['早餐','午餐','晚餐'])
09  print(df1)
```

執行結果：

```
            早餐 午餐 晚餐
2017-08-12  65  75  65
2017-08-13  65  55  70
2017-08-14  70  55  80
2017-08-15  70  85  95
2017-08-16  65  85  60
2017-08-17  60  80  60
2017-08-18  95  95  50
```

Pandas 的 DataFrame 很方便的一個功能是可以透過索引值來選擇或搜尋數據，以下示範兩種方式：

- df1[' 欄位名稱 '] 取得該欄位底下的值
- df1[起始索引 : 結束索引] 取得起始索引列到結束索引列前的資料

E8-1-11.py

```
10  print(df1[' 午餐 '])
11  print(df1[0:3])
```

執行結果：

```
2017-08-12    75
2017-08-13    55
2017-08-14    55
2017-08-15    85
2017-08-16    85
2017-08-17    80
2017-08-18    95
Freq: D, Name: 午餐 , dtype: int32
             早餐  午餐 晚餐
2017-08-12  65    75   65
2017-08-13  65    55   70
2017-08-14  70    55   80
```

DataFrame 也可透過指定的位置來選擇數據，如 loc、iloc、ix。

E8-1-11.py

```
12  print(df1.loc['20170812'])
13  print(df1.loc[:,[' 早餐 ',' 晚餐 ']])
14
15  print(df1.iloc[3,1])
16  print(df1.iloc[3:5,1:3])
17
18  print(df1[df1. 午餐 >80])
```

執行結果：

```
早餐  65
午餐  75
晚餐  65
Name: 2017-08-12 00:00:00, dtype: int32
             早餐 晚餐
2017-08-12   65   65
2017-08-13   65   70
2017-08-14   70   80
2017-08-15   70   95
2017-08-16   65   60
2017-08-17   60   60
2017-08-18   95   50
85
             午餐 晚餐
2017-08-15   85   95
2017-08-16   85   60
             早餐 午餐 晚餐
2017-08-15   70   85   95
2017-08-16   65   85   60
2017-08-18   95   95   50
```

以下示範另外一種建立 DataFrame 的方式，是由字典開始：

E8-1-12.py

```
01  import pandas as pd
02  import numpy as np
03  # 生成方式二，字典方式
04  df2 = pd.DataFrame({
05  '小數':pd.Series (1,index=list(range(4)),dtype='float32'),
06  '整數':np.array([3] * 4,dtype='int32'),
07  '時間':pd.Timestamp('20170812'),
08  '類別資料':pd.Categorical(['test','train','test','train']),})
09
10  #dtype 指定資料格式
```

想一想：字典的 key 代表甚麼？

接下來介紹 DataFrame 常用的三個基本的屬性 dtypes、index、describe，可藉由這三個屬性了解 DataFrame 的基本資料與狀態。

1. dtypes 查看資料型態。
2. index 查看資料集的索引。
3. describe() 查看數字資料描述。

E8-1-12.py

```
11  #DataFrame 的屬性
12  print(df2)
13  print(df2.dtypes)
14  print(df2.index)
15  print()
16  print(df2.columns), print(df2.values)
17  print()
18  print(df2.describe())
```

執行結果：

```
    小數 整數        時間      類別資料
0  1.0    3 2017-08-12    test
1  1.0    3 2017-08-12   train
2  1.0    3 2017-08-12    test
3  1.0    3 2017-08-12   train
小數        float32
整數          int32
時間    datetime64[ns]
類別資料      category
dtype: object
Int64Index([0, 1, 2, 3], dtype='int64')

Index([' 小數', ' 整數','時間','類別資料 '], dtype='object')
[[1.0 3 Timestamp('2017-08-12 00:00:00') 'test']
 [1.0 3 Timestamp('2017-08-12 00:00:00') 'train']
 [1.0 3 Timestamp('2017-08-12 00:00:00') 'test']
 [1.0 3 Timestamp('2017-08-12 00:00:00') 'train']]

       小數  整數
count  4.0  4.0
mean   1.0  3.0
std    0.0  0.0
min    1.0  3.0
25%    1.0  3.0
50%    1.0  3.0
75%    1.0  3.0
max    1.0  3.0
```

在我們建立完 DataFrame 並且查詢了基本的資料型態後，我們亦可以針對 DataFrame 進行資料的轉置以及排序資料。

E8-1-12.py

```
19  print(df2.T)
20  print(df2.sort_index(axis=1, ascending=False))
21  print(df2.sort_values(by=' 類別資料 '))
```

執行結果：

```
0          ...                                3
小數                    1          ...                          1
整數           3          ...                          3
時間   2017-08-12 00:00:00       ...       2017-08-1200:00:00
類別資料       test           ...                        train

[4 rows x 4 columns]
  類別資料     時間 整數  小數
0    test 2017-08-12    3  1.0
1   train 2017-08-12    3  1.0
2    test 2017-08-12    3  1.0
3   train 2017-08-12    3  1.0
  小數 整數     時間  類別資料
0  1.0    3 2017-08-12    test
2  1.0    3 2017-08-12    test
1  1.0    3 2017-08-12   train
3  1.0    3 2017-08-12   train
```

而對於已建立好的 DataFrame，我們可以進一步新增輸入或是修改現有的值，這時候需要利用 loc、iloc 來進行修改，判斷的方式亦同。

E8-1-13.py

```
01 import numpy as np
02 import pandas as pd
03 eat = np.random.randint(10,size=(7,3))*5+50
04 dates = pd.date_range('20170812',periods=7)
05 df1 = pd.DataFrame(eat, index=dates, columns=['早餐','午餐','晚餐']
06 df1.iloc[2,2] = 95
07 df1.loc['20170818','晚餐'] = 60
08 df1.晚餐[df1.早餐>80] = 40
09 df1.loc['20170814','午餐'] = np.nan
10 print(df1)
```

執行結果：

```
            早餐  午餐  晚餐
2017-08-12  95  65.0  40
2017-08-13  70  60.0  95
2017-08-14  55   NaN  95
2017-08-15  85  75.0  40
2017-08-16  75  80.0  75
2017-08-17  80  85.0  75
2017-08-18  70  90.0  60
```

用 Pandas 來讀取或儲存檔案也是非常方便的，可以將檔案的內容存取成 DataFrame，亦可以非常方便的將處理結果儲存成新的 csv 檔。

E8-1-14.py

```
01 import pandas as pd
02 data = pd.read_csv('XXX.csv')
03 print(data)
04 data.to_csv('student.csv')
```

說明：針對不同的格式檔案的讀取與儲存，請參照 http://pandas.pydata.org/pandas-docs/stable/io.html

當有來自不同資料來源的時候，可以使用 pd.concat() 橫向或縱向合併兩個 DataFrame。

E8-1-15.py

```
01  import pandas as pd
02  import numpy as np
03
04  # 定義資料集
05  df1 = pd.DataFrame(np.ones((3,4))*0,
06        columns=['a','b','c','d'])
07  df2 = pd.DataFrame(np.ones((3,4))*1,
08        columns=['a','b','c','d'])
09  df3 = pd.DataFrame(np.ones((3,4))*2,
10        columns=['a','b','c','d'])
11
12  #concat 縱向合併
13  res = pd.concat([df1, df2, df3], axis=0, ignore_index=True)
14  print(res)
```

	a	b	c	d
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0
6	2.0	2.0	2.0	2.0
7	2.0	2.0	2.0	2.0
8	2.0	2.0	2.0	2.0

圖 8-8 使用 concat() 合併 DataFrame

使用 pd.concat() 合併資料時，可以選擇設定 axis=0 為直向合併，ignore_index=True 可以忽略合併時舊有的索引值欄位，而改採用自動產生的索引值。而 concat 的 join 屬性有兩種模式：inner 交集以及預設的 outer 聯集。

E8-1-16.py

```
01 import pandas as pd
02 import numpy as np
03
04 df1 = pd.DataFrame(np.ones((3,4))*0,
05       columns=['a','b','c','d'], index=[1,2,3])
06 df2 = pd.DataFrame(np.ones((3,4))*1,
07       columns=['b','c','d','e'], index=[2,3,4])
08 res = pd.concat([df1, df2], axis=0, join='outer',
09       ignore_index=True )
10 print(res)
```

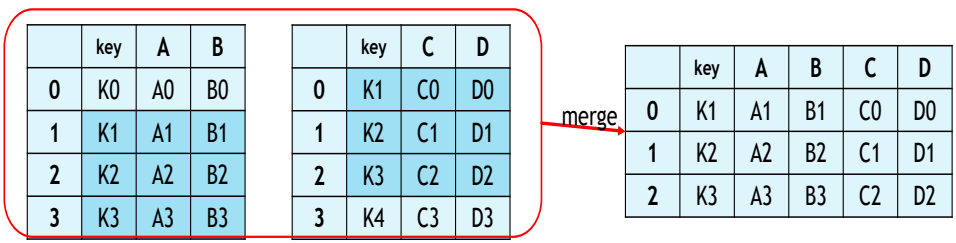
	a	b	c	d	e
0	0.0	0.0	0.0	0.0	NaN
1	0.0	0.0	0.0	0.0	NaN
2	0.0	0.0	0.0	0.0	NaN
3	NaN	1.0	1.0	1.0	1.0
4	NaN	1.0	1.0	1.0	1.0
5	NaN	1.0	1.0	1.0	1.0

圖 8-9 使用 concat() 合併 Dataframe

合併兩個 DataFrame 時，如果需要針對特定的索引合併，這時候就需要使用 merge() 進行合併，可使用 on= 指定要索引的欄位，並且使用 how= 指定 inner、outer、right 或 left 模式。

E8-1-17.py

```
01 import pandas as pd
02
03 left = pd.DataFrame(
04 {'key': ['K0', 'K1', 'K2', 'K3'],
05  'A': ['A0', 'A1', 'A2', 'A3'],
06  'B': ['B0', 'B1', 'B2', 'B3']})
07
08 right = pd.DataFrame(
09 {'key': ['K1', 'K2', 'K3', 'K4'],
10 'C': ['C0', 'C1', 'C2', 'C3'],
11 'D': ['D0', 'D1', 'D2', 'D3']})
12
13 res = pd.merge(left, right, on='key')
14 print (res)
```



	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

	key	C	D
0	K1	C0	D0
1	K2	C1	D1
2	K3	C2	D2
3	K4	C3	D3

	key	A	B	C	D
0	K1	A1	B1	C0	D0
1	K2	A2	B2	C1	D1
2	K3	A3	B3	C2	D2

圖 8-10 使用 merge() 合併 Dataframe

E8-1-18.py：

```
01  import pandas as pd
02
03  # 定義資料集並列印出
04  left = pd.DataFrame(
05  {'key1': ['K0', 'K1', 'K1', 'K2'],
06   'key2': ['K0', 'K1', 'K0', 'K1'],
07   'A': ['A0', 'A1', 'A2', 'A3'],
08   'B': ['B0', 'B1', 'B2', 'B3']})
09
10  right = pd.DataFrame(
11  {'key1': ['K0', 'K1', 'K1', 'K2'],
12    'key2': ['K0', 'K0', 'K0', 'K0'],
13    'C': ['C0', 'C1', 'C2', 'C3'],
14    'D': ['D0', 'D1', 'D2', 'D3']})
15
16  res = pd.merge(left, right, on=['key1', 'key2'], how='inner')
17  print (res)
```

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

圖 8-11 使用 merge() 合併 Dataframe

小結

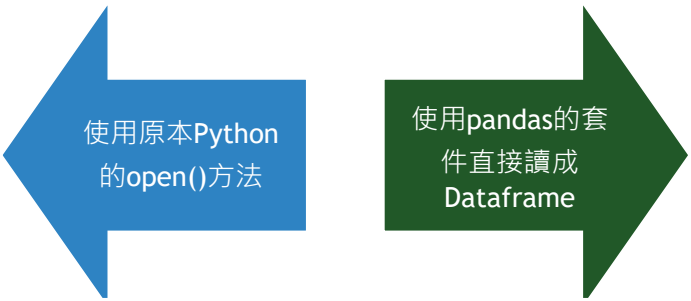
透過本小節的內容，我們已經可以熟悉 Python 中有關科學運算裡的兩個重要套件：NumPy 與 pandas。我們可以透過 NumPy 的 array 處理一維陣列以及多維矩陣運算；亦可利用 pandas 的 DataFrame 處理結構化具列索引與欄標籤的二維資料集，並且進一步針對資料表進行常見的新增、刪除、修改、合併等動作，以 Python 有效率地處理資料相關問題。

8-2 資料預處理實作

在處理大量資料時，首先的問題是大量資料怎麼進入程式的陣列、Series 或 DataFrame ？最常見的是從資料庫讀取資料，或者透過 CSV 等檔案匯入資料，也可以使用網路爬蟲到特定網站抓取需要的資料。抓取外部資料的來源很多樣，每一種方法都需要有相當篇幅的介紹，這邊我們先介紹最簡單的方法：檔案的匯入與匯出。在取得資料後，請先忍耐忍耐，還沒辦法立刻進行資料分析喔，因為仔細檢查外部資料時很容易發生各種問題，例如不同來源的資料格式不同、資料欄位有遺漏值或重複值，有時候我們還需要先將相關的資料合併，或者是進行交集與聯集的處理等。這一節我們就來看看資料在開始分析之前常見的各種前處理工作。

檔案匯入匯出

這裡會介紹兩種讀檔案的方式，可以用 open() 的方法讀取文字檔，也可以使用 pandas 的套件直接從檔案讀取成 DataFrame。



首先介紹第一種檔案開啟的方式，是使用 Python 的 open() 打開檔案。指令格式是 f = open(' 檔案 ', ' 模式 ')，其中模式有三種情形：

1. r 讀取 (檔案需存在)
2. w 新建檔案寫入 (檔案可不存在，若存在則清空)
3. a 資料附加到舊檔案後面 (游標指在 EOF)

在檔案開啟之後，可使用 read() 針對內容進行一次全部讀取。除此之外還有其他的讀取模式：

1. f.read(size)

   size 可以指定要讀進來的字串長度。

2. f.readline()

   讀取一行。

3. f.readlines()

   讀取成一個列表，每一行為列表中的一項。

在檔案寫入的部分，一樣是使用 open() 打開，並且使用 write() 寫入：

- f = open('檔案', 'a')
- f.write('要添加的內容….')
- f.close()

第二種常用的方式是使用 pandas 的套件直接讀成 DataFrame，如果你希望資料讀入程式後，資料以 DataFrame 的形式進行後續的分析處理，那麼建議你以這裡介紹的方式匯入與匯出資料，程式如下：

- import pandas as pd
- df = pd.read_csv('開啟的檔名.csv')
- df.to_csv('要存檔的檔名.csv')

中文要注意轉碼問題，有可能要設定編碼，如：

```
df = pd.read_csv('檔名.csv', encoding='big5')
```

E8-2-1.py：

```
01 import pandas as pd
02 df1 = pd.read_('chap8.csv',encoding='big5')
03 print(df1[:5])
```

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

圖 8-12 讀取 csv 檔

使用第二種方式直接讀成 DataFrame，如果要讀取的檔案是其他格式如 excel、sql、sas 等不同格式，可參考下表的方式與指令。

表 8-1: 讀取檔案格式

Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google Big Query	read_gbq	to_gbq

處理遺漏值

在我們讀取資料成 DataFrame 之後，我們可以先判斷是否有缺失數據，下列程式示範如何判斷是否有遺漏值，以及遺漏值的分布情形。

首先，我們可以用 .isnull() 去判斷 DataFrame 是否有遺漏值，有遺

漏值的資料會回傳 False。

E8-2-1.py：

```
03  # 判斷是否有缺失數據 NaN, 為 True 表示缺失數據：
04  print(df1.isnull())
```

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

圖 8-13 csv 檔原內容

	ID	dept	gender	test01	test02	test03
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	True	False
3	False	False	False	True	False	False
4	False	False	False	False	False	False

圖 8-14 判斷是否有缺失數據

再者，我們可以使用 sum() 進行加總，這樣可以呈現不同欄位的遺漏值總數。

E8-2-1.py：

```
05  print(df1.isnull().sum())
```

	ID	dept	gender	test01	test02	test03
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	True	False
3	False	False	False	True	False	False
4	False	False	False	False	False	False

```
ID          0
dept        0
gender      0
test01      1
test02      1
test03      0
dtype: int64
```

圖 8-15 顯示遺漏值分布情形

當我們遇到遺漏值的時候，我們會有很多種處理的方法，例如針對遺漏值的欄位進行填值或刪除，以下我們針對不同處理方法一一介紹。

- 處理方法一：填入指定值

在 DataFrame 中使用 .fillna()，針對遺漏的欄位直接填補指定的值。

E8-2-1.py：

```
06  df2 = df1.fillna(value=0)
```

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	0.0	85
3	4	牙醫學系(106)	m	0.0	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

圖 8-16 填入指定值

- 處理方法二：刪除遺漏值

可使用 .dropna() 將遺漏欄位的相關筆數直接刪除。其中還可定義條件或針對特別的欄位刪除。

1. 將 NaN 值刪除

E8-2-1.py：

```
07  df3 = df1.dropna()
```

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

ID=3,4被刪除了

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	5	牙體技術學系(104)	f	95.0	85.0	80

圖 8-17 將 NaN 值刪除

2. 參數

E8-2-1.py：

```
08  # 其他用法
09  df1.dropna(subset=['test02']) # 針對特定欄位名稱
10  df1.dropna(axis=0) #0: 刪除有 NaN 值的列; 1: 刪除有 NaN 值的欄
11  df1.dropna(how='any') #'any': 存在 NaN 就 drop 掉;
12                        #'all': 全是 NaN 才 drop
13  df1.dropna(thresh=3) # 至少要有 thresh 個非 NaN 值
14  # 其他用法
15  df3 = df1.dropna(subset=['test02'],axis=0,how='any')
16  # 針對 test02 列存在 NaN 就 drop 掉
```

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

ID=3被刪除了

			gender	test01	test02	test03
			m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	4	牙醫學系(106)	m	NaN	95.0	90
3	5	牙體技術學系(104)	f	95.0	85.0	80

圖 8-18 將 NaN 值刪除

- 處理方法三：使用其他值進行插補

可使用其他具有代表性的數值如平均值、中位數等針對遺漏欄位進行插補。下面範例則是使用 sci-kit learn 的套件進行插補，插補時可選擇平均值 mean, 中位數 median 或眾數 most_frequent。

E8-2-1.py：

```
17 from sklearn.preprocessing import Imputer
18 # strategy='mean', 'median', 'most_frequent'
19 imr = Imputer(missing_values='NaN', strategy=mean, axis=0)
20 imr = imr.fit(df1)
21 imputed_data = imr.transform(df1.values)
22 print(imputed_data)
```

	ID	test01	test02	test03
0	1	95.0	85.0	95
1	2	95.0	95.0	95
2	3	95.0	NaN	85
3	4	NaN	95.0	90
4	5	95.0	85.0	80

	ID	test01	test02	test03
0	1	95.0	85.0	95
1	2	95.0	95.0	95
2	3	95.0	90.0	85
3	4	95.0	95.0	90
4	5	95.0	85.0	80

圖 8-19 使用其他值補值

移除重複值

在我們讀取資料成 DataFrame 之後，除了處理遺漏值的問題，常常我們也會遇到重複的資料這樣的問題。當我們有遇到重複值的問題，我們可以用 duplicated() 去判斷 DataFrame 是否有重複值，有重複值的資料會回傳 True。

而通常針對重複值的處理，一般我們會選擇把重複的值刪除，而刪除的方式也是非常簡單，還可以針對特定欄位進行重複值的比對進行刪除，亦可加上欄位名稱與保留順序。例如：df.drop_duplicates(['col1'], keep='last') 即是針對 col1 這個欄位進行重複值的判斷，當有重複值的時候，我們用 keep='last' 只保留最後一筆重複的資料。

```
01  df.drop_duplicates()
```

E8-2-2.py

```
01  import pandas as pd
02  df = pd.DataFrame(
03       {'col1':['A','A','A','A','B','B','C','C'],
04       'col2':[1,2,2,3,3,4,5,5]})
05  df_d = df.drop_duplicates(['col1'], keep='last')
06  # 指定欄位 col1 重複就移除，保留最後一個
07  print(df_d)
```

	col1	col2
0	A	1
1	A	2
2	A	2
3	A	3
4	B	3
5	B	4
6	C	5
7	C	5

	col1	col2
3	A	3
5	B	4
7	C	5

圖 8-20 移除重複值

資料對應 \ 取代 \ 分箱

最後再補充一下，我們進行資料處理的時候，可能需要針對部分的欄位資料批次的處理，這時候可能就會需要用到資料的對應或取代等處理。

我們先說明資料的對應，在進行資料對應的時候，必須新建一個字典，然後用字典裡的 value 取代原本的 key (原本資料表的值)。在建立字典之後，可以在特定欄位用 map() 執行。例如：下面的例子就是將性別欄位 (女、男) 變成 (0、1)。

E8-2-3.py

```
01  import pandas as pd
02  df = pd.DataFrame({'gender':['male','male','female',
03                       'male','female','female']})
04
05  gender_to_boolean = {'female':0,'male':1} # 字典
06  df['code'] = df['gender'].map(gender_to_boolean)
07  print(df)
```

	gender
0	male
1	male
2	female
3	male
4	female
5	female

	gender	code
0	male	1
1	male	1
2	female	0
3	male	1
4	female	0
5	female	0

圖 8-21 資料對應

跟對應很相似的另外一種方法是取代。DataFrame 取代的方法是 df.replace(原本的值 , 新的值)。下面的例子就是針對 col2 欄位，用數字 0 來取代字串 '-' 。

E8-2-4.py

```
01  import pandas as pd
02  df = pd.DataFrame({
03       'col1':['c01','c02','c03','c04','c05'],
04       'col2':[65,'NULL','-','-',78],
05       'col3':[321,34,'NULL','-',34]})
06  df['col2'] = df['col2'].replace('-',0)
07  print(df)
```

	col1	col2	col3
0	c01	65	321
1	c02	NULL	34
2	c03	-	NULL
3	c04	-	-
4	c05	78	34

	col1	col2	col3
0	c01	65	321
1	c02	NULL	34
2	c03	0	NULL
3	c04	0	-
4	c05	78	34

圖 8-22 資料取代

df.replace() 取代的方式也可以用字典，然後用字典裡的 value 取代原本的 key (原本資料表的值)。

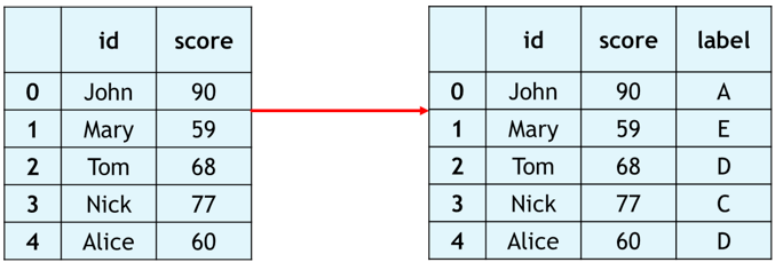
```
01  df.replace({'NULL':0 , '-':-1})
```

另外一個在處理資料常會用到的是分箱，使用分箱的方法可以使連續的類別數值資料變成類別資料。接下來示範如何將 0 到 100 分的分數轉換成 A、B、C、D 的等級制。

一開始我們建立兩個列表，第一個 bins (分箱的間隔點列表) 跟第二個 labels (各區間對應的標籤)，然後使用 .cut() 方法進行分箱。

E8-2-5.py

```
01 import pandas as pd
02 df = pd.DataFrame({
03      'id':['John','Mary','Tom','Nick','Alice'],
04      'score':[90,59,68,77,60]})
05 bins = [0, 60, 70, 80, 90, 100]
06 labels = ['E','D','C','B','A']
07 df['label'] = pd.cut(df['score'],bins, right=False, labels=labels)
08 print(df)
```



	id	score
0	John	90
1	Mary	59
2	Tom	68
3	Nick	77
4	Alice	60

	id	score	label
0	John	90	A
1	Mary	59	E
2	Tom	68	D
3	Nick	77	C
4	Alice	60	D

圖 8-23 資料分箱

小結

本小節示範了從檔案匯入成 Python 可讀取的格式、遺漏值和重複值處理、資料表聯集等實際資料實作，這些都是建立統計學或機器學習模型時非常重要的前置作業。趕快拿起手邊的資料一起動手做做看吧！

Chapter 9

社群資料分析

/ 胡筱薇

現代幾乎人人都有自己喜愛的社群媒體，社群媒體中累積的大量資料，是許多資料科學家熱衷分析的標的物之一。本章將帶領讀者從無到有完整地體驗 Facebook 社群資料的分析流程，搭配資料分析演算法和相關套件（如 seaborn、jieba 和 sklearn），進行資料分析與視覺化。本章共有五個小節，包括：環境安裝、認識 Facebook API、EDA 探索式資料分析、文字探勘和資料建模。學習完本章後，讀者將學會抓取 Facebook 社群資料並以演算法分析、以圖表呈現分析結果、以及運用線性迴歸模型預測。

9-1 環境安裝：Python 程式語言

經過前面章節的練習，大家應該對 Python 有相當認識，也完成不少程式練習吧！！甚麼，你覺得前面使用 IDLE 的介面寫程式很麻煩？安裝 numpy 或 pandas 等套件實在很頭昏？既然大家都花不少力氣終於入門 Python，那我們就來介紹進階的工具：Anaconda，協助大家開發複雜的程式工具，也能夠玩玩 Python 提供的各種神奇功能。

本章將以 Python 的進階應用為介紹主題，基本上是互相獨立的單元，大家可以照順序慢慢閱讀慢慢練習，也可以挑自己有興趣的主題跳著嘗試，萬一碰到有連續性的程式範例而看不懂時，再回頭慢慢研讀整個主題就可以，總之就是依照自己喜歡的步調練習更深入的 Python 程式吧。

Anaconda

首先，我們需要先安裝 Anaconda 軟體來啟動 Python 環境！ Anaconda 為一款集合許多工具的軟件包，同時也包含了不同的編譯器 (Jupyter notebook、Spyder、RStudio 等)，編譯器為撰寫程式語言的地方，而我們在以下將會選用 Jupyter notebook 編譯器作為撰寫 Python 的環境。

官方下載網址：https://www.anaconda.com/download/

圖 9-1 Anaconda Navigator

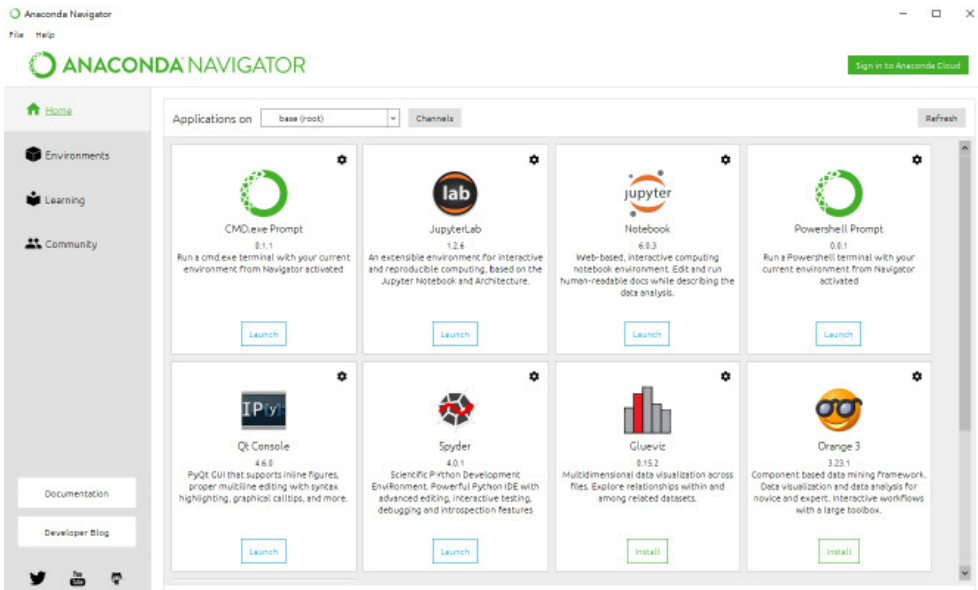


圖 9-2 Anaconda Navigator 開啟後畫面

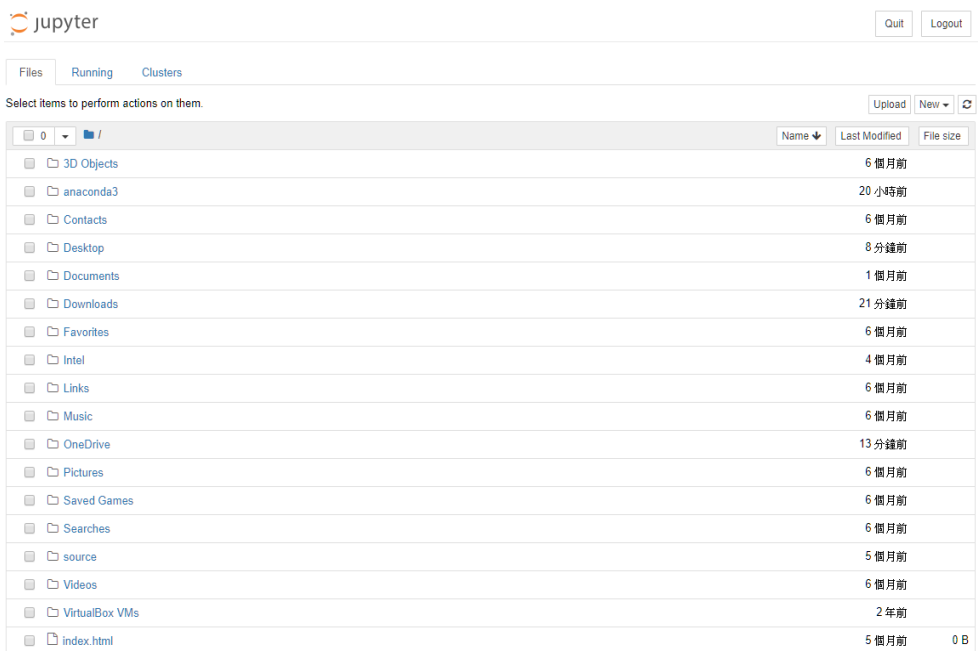


圖 9-3 Jupyter Notebook

安裝完成後點選 Anaconda-Navigator 開啟軟體 (圖 9-1)，接著在界面中找尋 Jupyter Notebook 點擊下方 Launch(圖 9-2)，最後在網頁瀏覽器中跳出此畫面 (圖 9-3) 即代表安裝成功，下一小節將一一介紹如何使用 Jupyter notebook 撰寫 Python 程式。

Jupyter Notebook

新增一個 Notebook

圖 9-4 新增一個 Notebook

Notebook 也就是撰寫 Python 的檔案，而畫面 (圖 9-3) 中的所有資料夾或檔案是真實存在於此台電腦中的資料，因此我們可以自由選擇今天寫的程式要存放於哪個資料夾當中。而新增一個 Notebook 的方式為點擊右上角的 New 選單，接著點選 Python 3 即可 (圖 9-4)。

重新命名

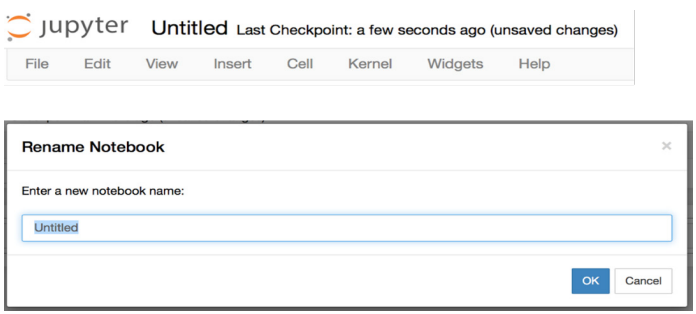


圖 9-5 重新命名

成功新增一個 Notebook 檔案後記得先幫檔案重新命名，方便日後找尋。重新命名的方式可以直接從電腦中的檔案位置按右鍵重新命名，也可以在 Notebook 當中點選最上方的 Untitled 字眼進行更改 (圖 9-5)。

撰寫程式 & 程式執行

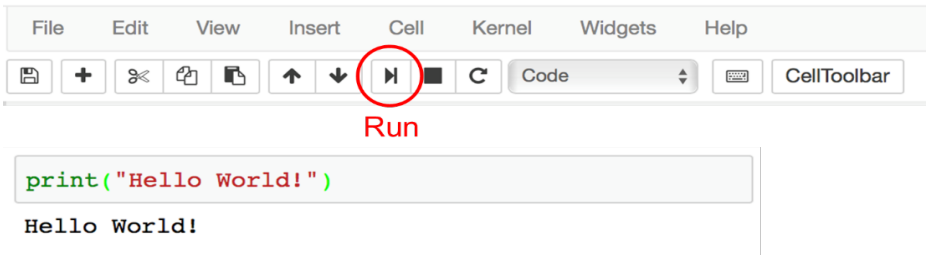


圖 9-6

接著我們在 Notebook 中間會看到有一個方框的區塊，此區塊即用來撰寫 Python 程式的地方，我們可以在此位置試著寫入 print(“Hello World!”) 字眼。撰寫完畢後會發現什麼事都還沒發生，這是因為我們尚未給予執行的指令，執行程式的方式共有三種，其一為點擊 (圖 9-6) 中紅色圈選的區塊，或是可以使用兩種快捷鍵 (Ctrl + Enter or Shift + Enter) 也可以成功執行，執行完成後會看到方框底下成功打印出 Hello World! 的字眼！

Cell 介紹

圖 9-7 Cell 介紹

剛剛我們寫入程式的方框又可稱之為 Cell，在 Notebook 檔案中我們可以自由新增很多個 Cell 來撰寫不同的程式 (點選上排 + 號即可)，而每一個 Cell 會獨立儲存各自執行的結果，這也是 Jupyter notebook 獨特的功能之一。我們可以試著新增第二個 Cell 執行 (圖 9-7) 當中的內容。

註解介紹

```
# 這是註解
# 我的第一支程式
print("Hello World!")
```

```
Hello World!
```

圖 9-8 註解測試

接著來介紹註解的功能，註解的特性為並非是程式語言，因此不論撰寫的內容為何皆不會被執行。而註解的目的在於方便使用者記錄此段內容為何，或是供他人閱讀時不用花太多時間即可快速掌握我們撰寫程式的內容，也就是寫筆記的概念。而註解撰寫的方式如 (圖 9-8)，在撰寫的內容前面加上 # 字號，可以發現字體顏色統一改變為一致的顏色，同時執行後可以看到下方只打印出 Hello World! 的字眼而沒有把註解的內容也打印出來。

Markdown 介紹

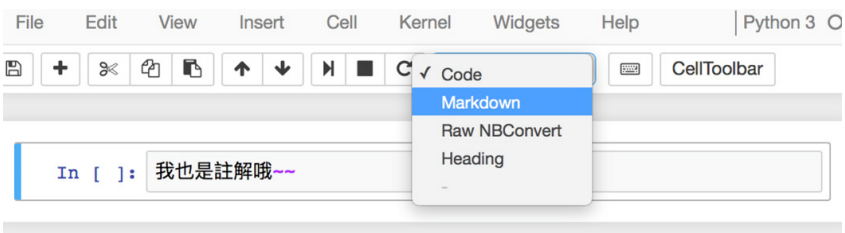


圖 9-9 Markdown

圖 9-10 Markdown

在這邊我們再來介紹另外兩種的註解方式，同時也是 Jupyter notebook 的獨有功能之一，也就是 Markdown 與 Heading。兩者的特色為不需要在撰寫程式的 Cell 中即可完成註解的撰寫，整體排版更加簡潔，類似於 Word 文件的排版格式。使用方式為 (圖 9-9) 中點擊上方 Code 的下拉式選單 (圖 9-11)，改選為 Markdown 或 Heading，接著於 Cell 當中打印想要撰寫的文字內容在執行，即可成功 (圖 9-10 & 9-12)。

Heading 介紹

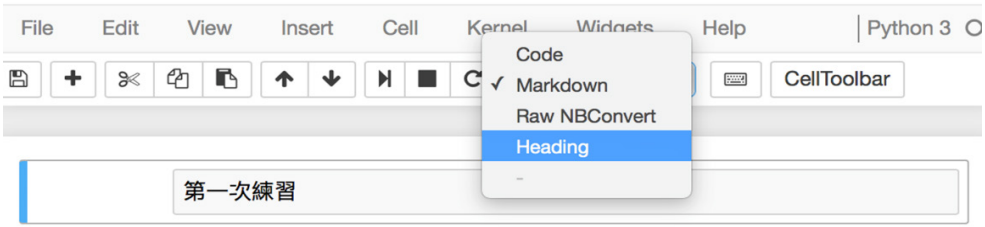


圖 9-11 Heading

圖 9-12 Heading

儲存檔案

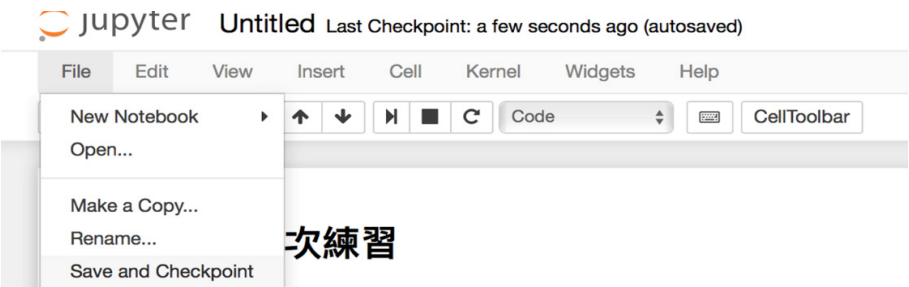


圖 9-13 儲存檔案

最後來教學如何儲存檔案，可以點選左上方 Flie 下拉式選單當中的 Save and Checkpoint(圖 9-13)，或是使用快捷鍵 (Ctrl + s) 即可成功儲存檔案。在這邊建議各位讀者要有一邊撰寫程式一邊儲存檔案的好習慣，減少全部都撰寫完成後才儲存檔案，以免中間 Anaconda 或是電腦當機，而造成撰寫的內容都消失的情況發生。

檔案位置

```
In [3]: import os
        os.getcwd()
Out[3]: '/Users/andy/Documents/python'
```

圖 9-14

另外如果真的找尋不到 Notebook 在電腦中的存放位置時，可以執行圖 9-14 的程式碼，此段程式會告訴我們 Notebook 的電腦存放路徑。

下載 Python 第三方套件

我們在前面有提到之所以選用 Python 有一原因為其具有豐富的函式庫套件，而這些額外的函式庫套件可以協助我們進行網路爬蟲、資料分析、機器學習、人工智慧、網站架設、遊戲開發等等的實務應用，接下來將會教學如何在 OS X 以及 Windows 的環境中下載其他的函式庫。

OS X 環境 (Mac 用戶)

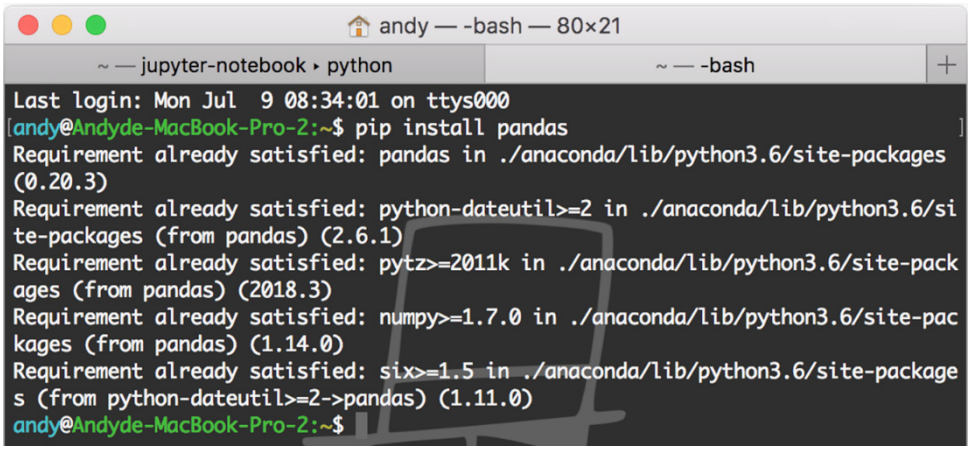


圖 9-15 pip install (OS X 環境)

Mac 用戶的話可以直接打開應用程式當中的終端機軟體，接著輸入 pip install 套件名稱，出現如圖 9-15 當中的畫面即為安裝完成。

Windows 環境

圖 9-16 pip install (Windows 環境)

Windows 用戶則在應用程式中搜尋 Anaconda Prompt，點選右鍵以系統管理員身份執行，接著輸入 pip install 套件名稱，出現如 (圖 9-16) 當中的畫面即為安裝完成。

OS X + Windows 環境

```
!pip install pandas
```

```
Requirement already satisfied: pandas in /Users/andy/anacon
da/lib/python3.6/site-packages (0.20.3)
Requirement already satisfied: python-dateutil>=2 in /Users
/andy/anaconda/lib/python3.6/site-packages (from pandas) (2
.6.1)
Requirement already satisfied: pytz>=2011k in /Users/andy/a
naconda/lib/python3.6/site-packages (from pandas) (2018.3)
Requirement already satisfied: numpy>=1.7.0 in /Users/andy/
anaconda/lib/python3.6/site-packages (from pandas) (1.14.0)
Requirement already satisfied: six>=1.5 in /Users/andy/anac
onda/lib/python3.6/site-packages (from python-dateutil>=2->
pandas) (1.11.0)
```

圖 9-17 pip install (OS X + Windows 環境)

除了使用各自的終端機安裝第三方套件的方法外，也可以直接在 Notebook 當中進行下載，在 Cell 當中首先打 ! 符號，後面輸入 pip install 套件名稱在執行，同樣可以下載成功 (圖 9-17)。範例中安裝的為 pandas 套件，此一套件廣泛用於資料分析當中，協助清整資料以及進行統計性分析，後續我們也會使用此一套件，因此建議讀者可以照著上述的方法先進行安裝。

小結

在此一章節中，藉由我們的帶領，相信各位都已成功安裝好 Anaconda，以及開啟 Jupyter notebook 試著執行看看 Python 程式了。在後面的章節中我們將會帶領各位進行大量的實作，中間不但會使用到大量的第三方函式庫，同時也會使用到許多較複雜的 Python 程式 (如 Json)，因此建議各位讀者若是之前從來沒有程式語言基礎，可以先稍微了解一下的基礎 Python 教學，這樣在後續的章節中才不會覺得太吃力呦！

9-2 資料取得：認識 Facebook API

本章節將會為各位介紹什麼是 API 以及如何透過 API 取得 Facebook 粉絲專頁的資料，中間資料爬取的過程皆會透過 Python 程式來完成。Facebook 作為社群網站的龍頭之一想必大家都已經耳熟能詳、且每天都必定會點開來看些消息吧！因此我們認為選用 Facebook 粉絲專頁的資料作為後續分析的範例是再適合不過的了。

API 介紹

API 全名為 Application Programming Interface，中文名稱又可稱為 " 應用程式介面 "；維基百科給予 API 的定義為「與網際網路相連的端系統提供的一個應用程式介面是軟體系統不同組成部分銜接的約定。」如果單看此定義有點太過於學術，並不好理解，因此我們用一個自動販賣機的例子來舉例說明。假設今天我們口渴了想要一瓶可樂，而我們需要透過前方面板上的按鈕來告訴販賣機我們今天想要的飲料是可樂，當點選完成以及付完款後可樂就會從下方跑出來給我們了。在取得可樂的過程中我們可以把 Facebook 的資料當作是販賣機當中的飲料，而中間的面板按鈕就是 API，最後付完款就可以取得 Facebook 的資料了，因此整體流程如下：

1. 想要一瓶可樂 (想要的資料)
2. 投幣按下按鈕 (送出請求)
3. 販賣機掉出可樂 (取得資料)

想必大家一定會很好奇，欸奇怪為什麼我只是想要 Facebook 的資料而已需要這麼麻煩？原因在於任何的社群媒體 (如：FB, IG, Twitter...) 都掌握了相當多關於個人隱私的資料，當這些資料隨意流出時會產生相當多的麻煩與風暴 (如 :2018 年 Facebook 劍橋流出資料事件)。這些社群媒體公司有必要控管哪些資料是可以開放給大眾使用，而又有哪些資料是需要保護使用者不會造成個資外流的。因此 API 就是我們與社群媒體公司中間接洽的窗口，透過 API 才可以取得我們想要的資料。

當然在使用 API 時有可能發生一種情況，以自動販賣機的例子來說：明明知道超商有賣水蜜桃口味的可樂，可是販賣機裡卻沒有看到，所以我們就知道販賣機裡沒有賣的飲料等同於 API 不開放、不讓你取得的資料。甚至有些公司的 API 可能也會有每月取得資料多寡的上限，就像販賣機的飲料賣完一樣，或是超過此上限就需要付費才能取得。雖然不開放的資料無法透過 API 來取得，但其實還是有其他透過網路爬蟲的技巧可以取得，不過這就是另外一個議題了，有興趣的讀者可以在去搜尋相關文章內容。

Facebook API

今天我們想要取得的是 Facebook 粉絲專頁的資料，因此就要透過 Facebook API 這個管道來取得資料；目前僅開放粉絲專頁、公開社團以及些許好友的公開資料，非好友的使用者資料因涉及隱私權問題並無開放。此外因為 2018 年的劍橋事件造成現階段只有該粉絲專頁的管理者能夠取得自己名下管理的粉專資料而已，因此若讀者並非是任何粉專的管理者，建議可以先隨意建立一個粉絲專頁，並且簡單發些文和留言，再試著使用以下的方法看能不能成功取得這些資料呦。

步驟 1：進入 **Facebook API** 官網：**https://developers.facebook.com**

圖 9-18 Facebook API 官網

步驟 2：點擊右上角更多的下拉式選單，找到 " 工具 " 後點擊進入頁面。

圖 9-19 找到 " 工具 " 選項

步驟 3：在開發人員工具頁面中，找尋圖形 **API** 測試工具，點擊進入頁面。

圖 9-20 找尋圖形 API 測試工具

步驟 4：進入圖形 **API** 測試工具後，點擊右側的 **Generate Access Token** 。

圖 9-21 Generate Access Token

步驟 5：點擊取得用戶存取權杖。

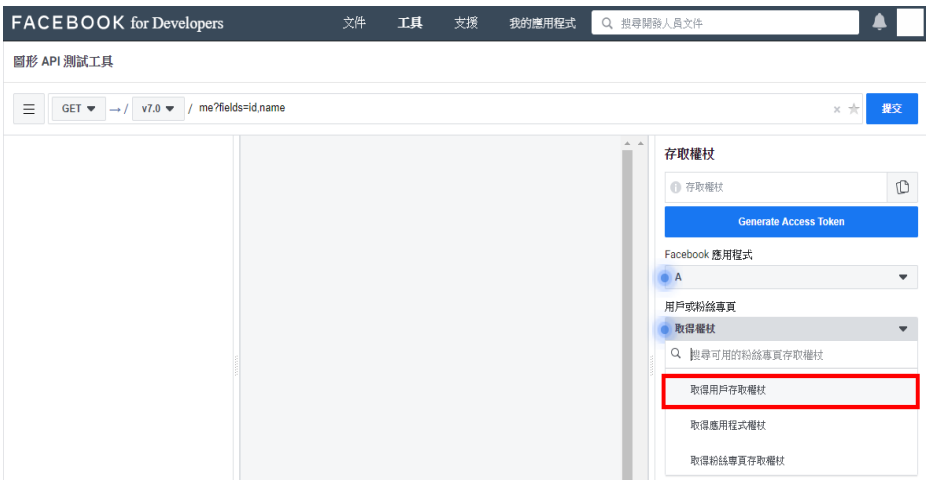


圖 9-22 取得用戶權杖

步驟 6：此頁面能設定我們能夠取得什麼樣資料的權限，現在我們要抓取的是粉絲專頁的資料，因此將中間「活動、社團和粉絲專頁」的權限全部打開。

圖 9-23 設定權限

步驟 7：設定完成權限後，回到 **Facebook** 的粉絲專頁網頁，並且複製上方的網址列。

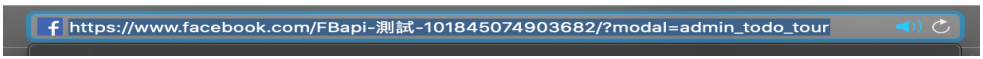


圖 9-24 複製網址列

步驟 8：照著圖 **9-25** 中的指示貼上複製好的網址列，按下右方的提交，此時可以看見畫面下方出現了粉絲專頁的名稱以及該粉絲專頁的專屬 **id** 。

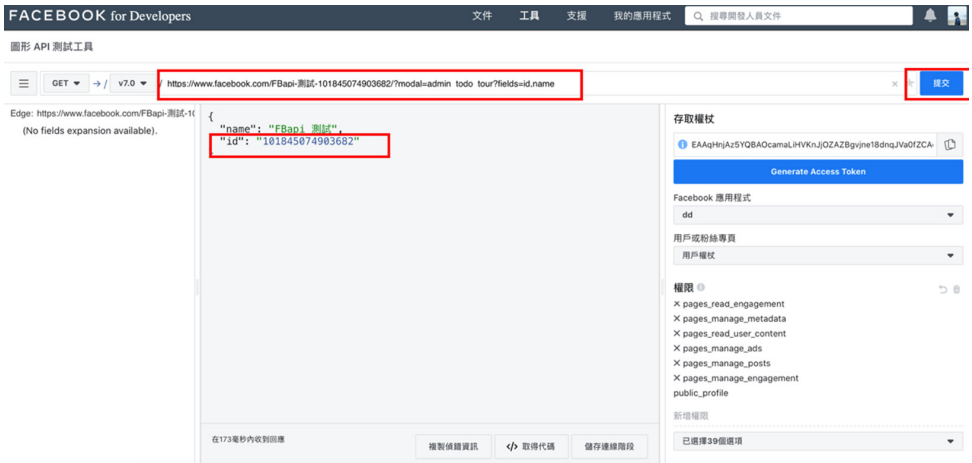


圖 9-25 取得粉絲專頁的 id

步驟 9：將出現的粉絲專頁 **id** 貼回剛剛貼上網址列的地方，重新按下提交後可以發現左邊的白框出現了搜尋欄位。

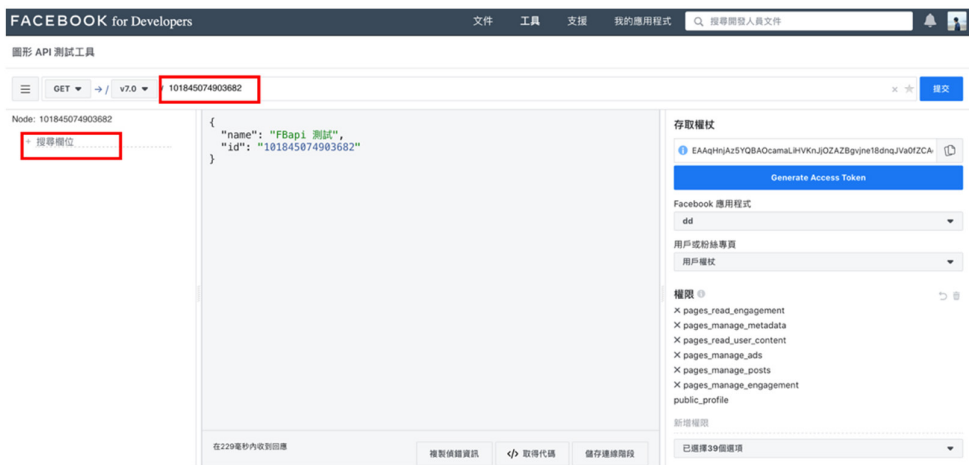


圖 9-26 出現搜尋欄位

步驟 10：在左方的搜尋欄位中找尋 **posts** 並點擊。

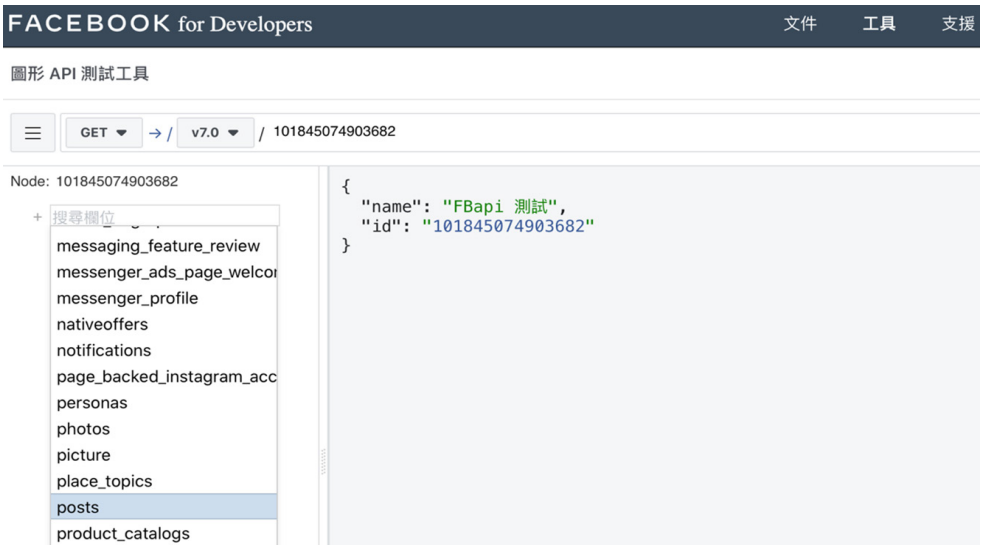


圖 9-27 點選

步驟 11：點擊提交後可以發現中間畫面出現了該粉絲專頁最新的貼文內容！

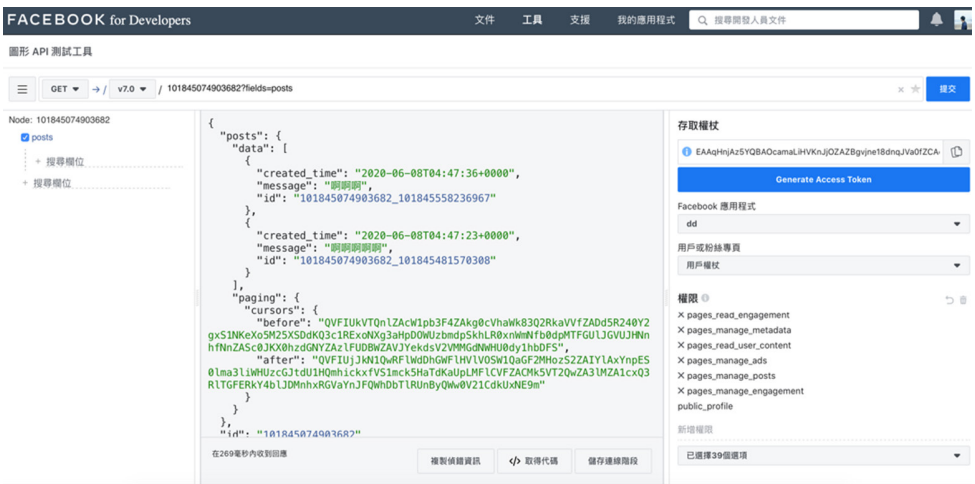


圖 9-28 貼文內容

步驟 12：剛剛我們所做的動作也就是透過 **API** 請求取得 **posts** 貼文內容的資料，而我們可以繼續新增其他想要的資訊 **(** 如：**Likes** 按讚數量等 **)**。

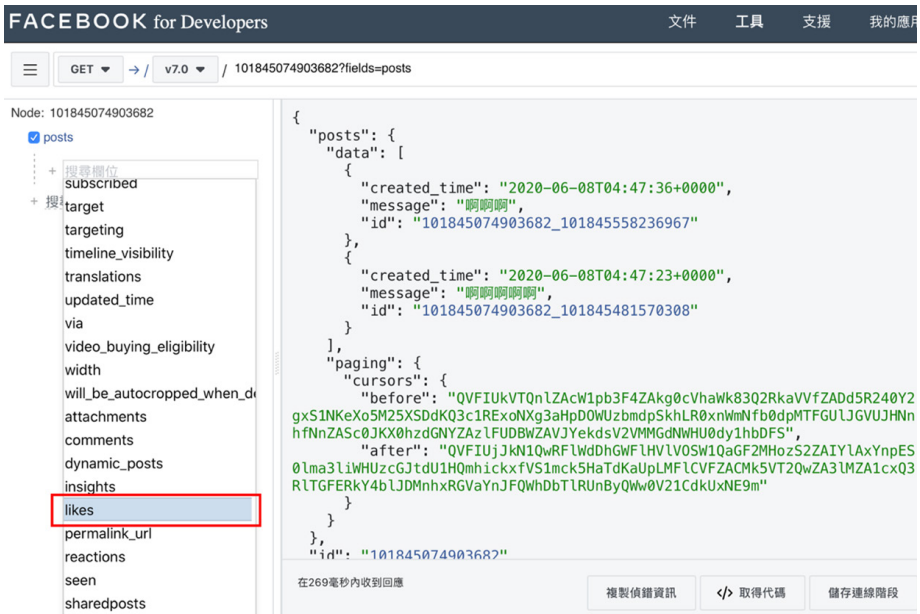


圖 9-29 可以繼續新增其他的資訊

步驟 13：點擊圖 **9-30** 畫面中的小箭頭將整個網址展開。

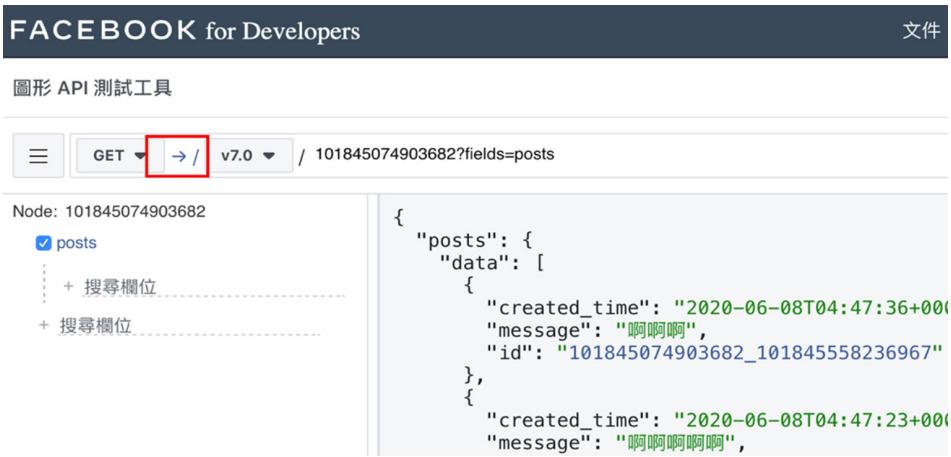


圖 9-30 網址展開

步驟 14：最後我們將此網址列複製起來，後續將透過 **Python** 程式來抓取資料。

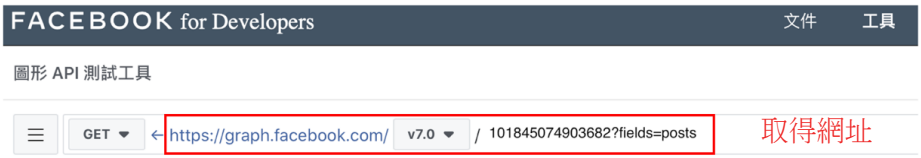


圖 9-31 複製網址

Python 串接 API

接下來在此小節中，我們將教各位如何透過 Python 程式串接 Facebook API 以及成功將粉絲專頁的資料存取成 Excel 檔案輸出。

匯入套件

首先先匯入相關套件，尚未安裝過的記得安裝 Json 與 requests 套件

```
01 import json
02 import requests
```

串接 Facebook API

此時我們需要剛剛圖形 API 測試工具頁面中的存取權杖、網址以及粉專 ID (圖 9-32)。

```
token = 存取權杖
res = requests.get( 搜尋欄位的網址 )
```

圖 9-32

依照相關位置將資訊複製至各字串當中，token 存放的是存取權杖，res 存放的則是請求過後取得的資料。

```
03  token =
04  'EAAVBPxZBVDb8BAIk78UfZBuID04j8VrrPT29IOOVUB8Dmo5
05  WZCa7uFIRlhpGjiW1ZBmiKLwS769xukxNUZAzoNri57UrL5Eld
06  9ZBkY9xlre6WgYXWl49QHK0qiBZBAG5e35bUTWVGhEPDhoxv5t
07  behZBeDybBZBO6Qr9K3C28yfQrQtHnmuFv8QQZCeDws22ZBB1N
08  cZD'
09
10  res = requests.get('https://graph.facebook.com/v2.9/{}/
11  posts&access_token={}'.format(325767030937170, token))
```

實際上因為需要取得文章按讚數、分享數、留言等相關資料，因此 requests 的網址會是非常長的一段，建議各位讀者可以直接複製後，按下執行送出請求。

https://graph.facebook.com/v2.9/{}/posts?fields=id,message,type,created_time,link,shares,comments.limit(0).summary(true),likes.limit(0).summary(total_count).as(reaction_like),reactions.type(LOVE).limit(0).summary(total_count).as(reactions_love),reactions.type(WOW).limit(0).summary(total_count).as(reactions_wow),reactions.type(HAHA).limit(0).summary(total_count).as(reactions_haha),reactions.type(SAD).limit(0).summary(total_count).as(reactions_sad),reactions.type(ANGRY).limit(0).summary(total_count).as(reactions_angry)&access_token={}

使用 json 讀取其格式

透過 API 抓回來的資料是使用 Json 的方式進行存取，因此我們使用 json 套件進行解碼，並存入 fanpage 變數當中。

```
12 fanpage = json.loads(res.text)
13 print(fanpage)
```

執行結果：

```
{'data': [{'id': '325767030937170_1050616578452208' ,
'message': '【徵才】\n 原友蘭教授 誠徵 # 專任全職助理 \n\n➡ 工作
地點：東海大學 \n 工作時間：週一到週五，# 沒有打卡，但是每週會需要固
定一天報告進度 \n➡ 工作條件：英文能書信溝通、統計分析、報告撰寫，
會處理 #GoogleAnalysis 資料或是 R 語言的優先考慮 \n➡ 工作福利：
\n* 有 3-4 次出國機會（印尼與菲律賓）\n* 包含住宿與機票 \n➡ 薪資
待遇：\n* 32000-34000 之間 + 勞健保 + 一個月年終 \n\n➡ 其他事項：
\n* 有碩士學會就可申請在東海兼課 \n* 會幫忙申請教師證 \n\n➡ 聯繫
方式：\n* yoyoyuan@go.thu.edu.tw\n* 0910-659134', 'type':
'status', 'created_time': '2019-11-06T02:55:08+0000',
'shares': {'count': 1}, 'comments': {'data':
```

文章相關資料位在鍵值為 data 的名稱內。

```
14 fanpage['data']
```

執行結果：

```
[{'comments': {'data': [],
   'summary': {'can_comment': True, 'order': 'ranked',
'total_count': 0}},
  'created_time': '2019-11-06T02:55:08+0000',
  'id': '325767030937170_1050616578452208',
  'message': '【徵才】\n 原友蘭教授 誠徵 # 專任全職助理 \n\n➡ 工作地點：東海大學 \n 工作時間：週一到週五，# 沒有打卡，但是每週會需要固定一天報告進度 \n➡ 工作條件：英文能書信溝通、統計分析、報告撰寫，會處理 #GoogleAnalysis 資料或是 R 語言的優先考慮 \n➡ 工作福利：\n* 有 3-4 次出國機會（印尼與菲律賓）\n* 包含住宿與機票 \n➡ 薪資待遇：\n* 32000-34000 之間 + 勞健保 + 一個月年終 \n\n➡ 其他事項：\n* 有碩士學會就可申請在東海兼課 \n* 會幫忙申請教師證 \n\n➡ 聯繫方式：\n* yoyoyuan@go.thu.edu.tw\n* 0910-659134',
```

我們可以發現透過一次的 API 會抓回最新的 25 筆文章內容。

```
15 len(fanpage['data'])
```

執行結果：

```
25
```

查看第一筆資料

可以發現裡面存放文章內容、貼文時間、按讚數量等等資訊。

```
16 fanpage['data'][0]
```

執行結果：

```
{'comments': {'data': [],
  'summary': {'can_comment': True, 'order': 'ranked',
'total_count': 0}},
 'created_time': '2019-11-06T02:55:08+0000',
 'id': '325767030937170_1050616578452208',
 'message': '【徵才】\n 原友蘭教授 誠徵 # 專任全職助理 \n\n➡ 工作地點：東海大學 \n 工作時間：週一到週五，# 沒有打卡，但是每週會需要固定一天報告進度 \n➡ 工作條件：英文能書信溝通、統計分析、報告撰寫，會處理 #GoogleAnalysis 資料或是 R 語言的優先考慮 \n➡ 工作福利：\n* 有 3-4 次出國機會（印尼與菲律賓）\n* 包含住宿與機票 \n➡ 薪資待遇：\n* 32000-34000 之間 + 勞健保 + 一個月年終 \n\n➡ 其他事項：\n* 有碩士學會就可申請在東海兼課 \n* 會幫忙申請教師證 \n\n➡ 聯繫方式：\n* yoyoyuan@go.thu.edu.tw\n* 0910-659134',
```

第一筆資料發文時間：

```
17 | fanpage['data'][0]['created_time']
```

執行結果：

```
'2019-11-06T02:55:08+0000'
```

第一筆資料文章 ID：

```
18 | fanpage['data'][0]['id']
```

執行結果：

```
'325767030937170_1050616578452208'
```

第一筆資料文章內容：

```
19 | fanpage['data'][0]['message']
```

執行結果：

```
'【徵才】\n 原友蘭教授 誠徵 # 專任全職助理 \n\n➡ 工作地點：東海大學 \n 工作時間：週一到週五，# 沒有打卡，但是每週會需要固定一天報告進度 \n➡ 工作條件：英文能書信溝通、統計分析、報告撰寫，會處理 #GoogleAnalysis 資料或是 R 語言的優先考慮 \n➡ 工作福利：\n* 有 3-4 次出國機會（印尼與菲律賓）\n* 包含住宿與機票 \n➡ 薪資待遇：\n* 32000-34000 之間 + 勞健保 + 一個月年終 \n\n➡ 其他事項：\n* 有碩士學會就可申請在東海兼課 \n* 會幫忙申請教師證 \n\n➡ 聯繫方式：\n* yoyoyuan@go.thu.edu.tw\n* 0910-659134'
```

第一筆資料不同的按讚數量 (例 : 讚 , 驚訝 , 愛心 , 笑 , 生氣 , 難過)

```
20 print(fanpage['data'][0]['reaction_like']['summary']['total_count'])
21 print(fanpage['data'][0]['reactions_wow']['summary']['total_count'])
22 print(fanpage['data'][0]['reactions_love']['summary']['total_count'])
23 print(fanpage['data'][0]['reactions_haha']['summary']['total_count'])
24 print(fanpage['data'][0]['reactions_angry']['summary']['total_count'])
25 print(fanpage['data'][0]['reactions_sad']['summary']['total_count'])
```

執行結果：

```
12
0
0
0
0
0
```

第一筆資料分享數量 :

```
26 fanpage['data'][0]['shares']['count']
```

執行結果：

```
1
```

第一筆資料留言數量 :

```
27 fanpage['data'][0]['comments']['summary']['total_count']
```

執行結果：

```
2
```

現在我們得知了所有與文章相關資訊的存放位置了，而我們一共有 25 筆文章資料，因此我們需要透過迴圈的方式將資料一筆筆取出，再存入不同的串列當中。

把資訊存入 list 裡

首先建立各個資訊的空串列變數，在透過 `for` 迴圈將每一筆資料相對應的資訊放入串列當中。第 43 行那邊加入了例外處理，原因是當分享數量是 0 時它並不會出現在 Json 檔裡，因此需要我們手動給予否則程式會出錯。

```
time, ID, context, like, wow, love,
 haha, angry, sad, share, comment = [], [],
 [], [], [], [], [], [], [], [], []

for i in fanpage['data']:
    time.append(i['created_time'])
    ID.append(i['id'])
    context.append(i['message'])

    like.append(i['reaction_like']['summary']['total_count'])
    wow.append(i['reactions_wow']['summary']['total_count'])
    love.append(i['reactions_love']['summary']['total_count'])
    haha.append(i['reactions_haha']['summary']['total_count'])
    angry.append(i['reactions_angry']['summary'['total_count'])
    sad.append(i['reactions_sad']['summary']['total_count'])
    try:
        share.append(i['shares']['count'])
    except:
        share.append(0)
    comment.append(i['comments']['summary']['total_count'])
```


25 筆資料以外的資料呢？

我們透過一次的 API 請求只會回傳 25 筆資料而已，原因是若一次就想把上千上萬篇的資料取得容易造成伺服器塞車或當機現象。而若要抓取存有下一個 25 筆資料的 Json 需要再次透過 API 請求，而換頁資訊位在鍵值為 paging 和 next 的欄位裡。

```
fanpage['paging']['next']
```

執行結果：

```
'https://graph.facebook.com/v2.9/325767030937170/posts?access_
token=EAAVBPxZBVDb8BAJib0SAxWnOtZCPBjXVnuPtUx0iVMYdq0GlcRrh
xwU5sZCGv3idx0ObuWgW3UVWzHXTpk3ZBB4gi3aIldxrgUdyyRYoZCQIvl9
ZCAoZB7jXnZA8cT74gQZAXEMHj3XpK7rZCZAUeSLySdOpZCHIQNi2B
ab8Dh4HZAr0zqkYWARIYo0LpDzNms3dIQGUY746LfRfDsgZD
ZD&fields=id%2Cmessage%2Ctype%2Ccreated_time%2Clink%2Cshares%2Ccomments.
limit%280%29.summary%28true%29%2Clikes.limit%280%29.summary%28to
tal_count%29.as%28reaction_like%29%2Creactions.type%28LOVE%29.limit%280%29.
summary%28total_count%29.as%28reactions_love%29%2Creactions.type%28WOW%29.
limit%280%29.summary%28total_count%29.as%28reactions_wow%29%2Creactions.
type%28HAHA%29.limit%280%29.summary%28total_count%29.
```

抓取 25 筆以外的資料

上述的執行結果已經列出了接下來需要請求的網址列了，因此可以透過前面同樣的方法再次取得資料。不過一次一次的抓取 25 筆資料的速度實在是太慢了，在這邊我們可以透過 while 迴圈幫助我們預先設定好總共想抓取的文章數量有多少，再一次全部抓取回來。如下面程式所述，page 為存放爬取頁數的變數，當每次抓取後 page 會自動 +1，而當 page 大於等於 5 時抓取會自動停止，也就是共抓取 100 筆文章資料。

```
page=2
url = fanpage['paging']['next']
while page < 5:
    res = requests.get(url)
    fanpage2 = json.loads(res.text)
    for i in fanpage2['data']:
        time.append(i['created_time'])
        ID.append(i['id'])
        context.append(i['message'])

        like.append(i['reaction_like']['summary']['total_count'])
        wow.append(i['reactions_wow']['summary']['total_count'])
        love.append(i['reactions_love']['summary']['total_count'])
        haha.append(i['reactions_haha']['summary']['total_count'])
        angry.append(i['reactions_angry']['summary']['total_count'])
        sad.append(i['reactions_sad']['summary']['total_count'])
        try:
            share.append(i['shares']['count'])
        except:
            share.append(0)
        comment.append(i['comments']['summary']['total_count'])
    url = fanpage2['paging']['next']
    page += 1
```

轉成 DataFrame 格式

成功抓取 100 筆的文章資料後，使用 zip 函數將不同的 list 合併。

```
information = list(zip(time, ID, context, like,
wow, love, haha, angry, sad, share, comment))
```

並且匯入 pandas 套件將其轉成 DataFrame 格式。

```
74 import pandas as pd
75 df = pd.DataFrame(information,
76 columns=['time','id','context','like','wow',
77 'love','haha','angry','sad','share','comment'])
```

透過 DataFrame 的操作我們可以輕易將抓取的資訊轉成表格內容存放。

```
78 df.head()
```

	time	id	context	like	wow	love	haha	angry	sad	share	comment
0	2018-11-06T02:55:08+0000	325767030937170_1050616578452208	【徵才】\n原友蘭教授 誠徵 #專任全職助理\n\n➡ 工作地點：東海大學\n工作時間週一...	12	0	0	0	0	0	1	0
1	2018-10-26T15:30:42+0000	325767030937170_1045065325674000	【東吳巨資碩士甄試報名】\n網路蓬勃發展的時代，KOL的人數也逐漸上升，\n但究竟要如何經營...	15	0	0	0	0	0	0	0
2	2018-10-24T04:50:17+0000	325767030937170_1043766419137224	【東吳巨資碩士甄試報名】\n想要了解時下深度學習、AI究竟是什麼嗎？\n就是現在！東吳巨資碩...	36	0	2	3	0	0	8	9
3	2018-10-19T16:07:39+0000	325767030937170_1041456549368211	【東吳巨資碩士甄試報名】\n還在煩惱報名步驟繁瑣嗎？\n還在擔心是不是少準備了什麼東西嗎？\...	13	0	0	0	0	0	2	0
4	2018-10-18T11:00:01+0000	325767030937170_1040710716109461	【東吳巨資碩士甄試報名】\n現在是數據化的時代，你卻還一片徬徨嗎？\n想要學以致用，習得一些...	50	0	1	0	0	0	11	0

圖 9-33

連同第一次換頁前爬的共爬取了 4 頁，100 筆資料。

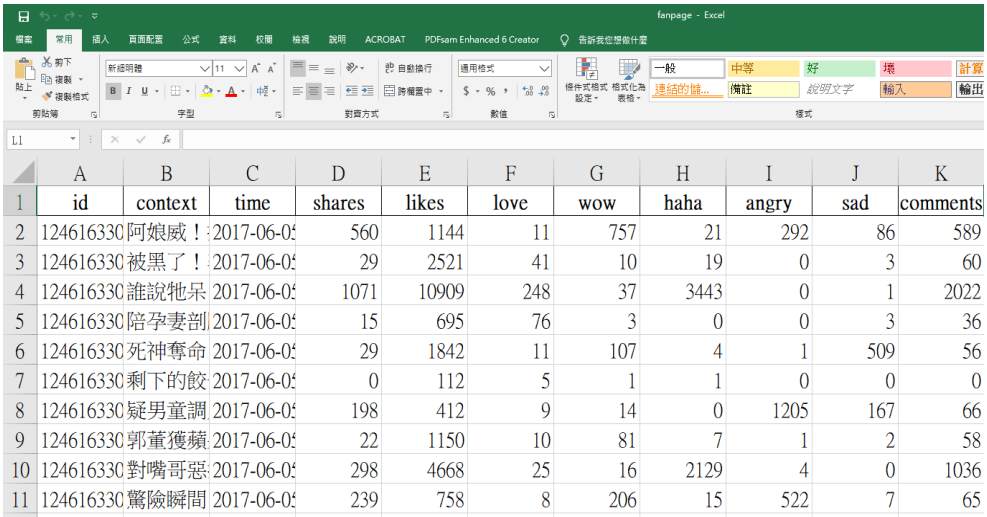
```
79 df.info()
```

執行結果：

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 11 columns):
time        100 non-null object
id          100 non-null object
context     100 non-null object
like        100 non-null int64
wow         100 non-null int64
love        100 non-null int64
haha        100 non-null int64
angry       100 non-null int64
sad         100 non-null int64
share       100 non-null int64
comment     100 non-null int64
dtypes: int64(8), object(3)
memory usage: 8.7+ KB
```

最後將此 DataFrame 匯出成 Excel 檔案，完成粉絲專頁的資料搜尋步驟。

```
80 df.to_excel('粉絲專頁資料.xlsx', index=False)
```



	A	B	C	D	E	F	G	H	I	J	K
1	id	context	time	shares	likes	love	wow	haha	angry	sad	comments
2	124616330	阿娘威！	2017-06-0	560	1144	11	757	21	292	86	589
3	124616330	被黑了！	2017-06-0	29	2521	41	10	19	0	3	60
4	124616330	誰說牠呆	2017-06-0	1071	10909	248	37	3443	0	1	2022
5	124616330	陪孕妻剖	2017-06-0	15	695	76	3	0	0	3	36
6	124616330	死神奪命	2017-06-0	29	1842	11	107	4	1	509	56
7	124616330	剩下的餃	2017-06-0	0	112	5	1	1	0	0	0
8	124616330	疑男童調	2017-06-0	198	412	9	14	0	1205	167	66
9	124616330	郭董獲蘋	2017-06-0	22	1150	10	81	7	1	2	58
10	124616330	對嘴哥惡	2017-06-0	298	4668	25	16	2129	4	0	1036
11	124616330	驚險瞬間	2017-06-0	239	758	8	206	15	522	7	65

圖 9-34

小結

在此一節中我們向各位介紹了何謂 API、如何使用 Facebook API 以及透過 Python 程式串接取得資料，中間使用到了大量的 Python 語法進行資料的清理以及邏輯運算，當然這只是其中一種取得 FB 粉專資料的寫法而已，網路上搜尋也可以找到不同的程式寫法，也鼓勵大家可以多瀏覽參考不同的寫法，或許可以得到不同的經驗值呢。若看完此節覺得有點吃力實屬正常，這已經是相當實用以及高階的應用範圍了，甚至很多公司會專門請人幫忙爬取甚至分析相關的粉絲專頁網站，因此還是建議新手可以多多練習 Python 程式的基本功 (如：迴圈、串列的處理、Json 的處理等等)，倘若看完此章節還跟得上的讀者相信對於 Python 已經有一定程度的了解及經驗了呢！那就趕緊跟著我們進入下一章節吧。

9-3 簡單料理：EDA 探索式資料分析

本章節將會為各位讀者示範一些使用 Python 清理資料的方式，包含處理遺失值、加減乘除計算、轉換時間格式等，所有過程皆會使用到 pandas 這個第三方的函式庫。清理完資料後接著會進行一些簡單的描述性統計以及使用視覺化圖表的方式呈現數據，過程中會使用到另外兩款函式庫 matplotlib 以及 seaborn，還沒安裝的讀者建議可以先行進行安裝。在這部分使用到的範例資料是東森新聞 FB 粉絲專頁在 2017 年左右的 10,000 篇文章資料，建議各位可以先與我們用同一份資料實作，之後再改用自己的其他資料試試看。

資料前處理

匯入套件與資料

首先匯入 pandas 套件以及匯入粉專資料。

```
01 import pandas as pd
02 df = pd.read_excel('fanpage.xlsx')
```

查看資料 1

此份資料的欄位也就是上一章節使用 API 所抓取的欄位資訊。

```
03 df.head(2)
```

	id	context	time	shares	likes	love	wow	haha	angry	sad	comments
0	1246163309068800_1560501197318299	阿娘威！披羊皮的狼？竟大口嚼小雞\n#要打統編：小編真的是快嚇死了...😩😩\n\n影片來源...	2017-06-05T03:09:40+0000	560	1144	11	757	21	292	86	589
1	1246163309068800_1560454417322977	被黑了！李毓芬演唱「大落拍」 網友卻意外發現「亮點」\n#條紋編：這一段應該是昨天的亮點表演...	2017-06-05T03:00:00+0000	29	2521	41	10	19	0	3	60

圖 9-35

查看資料 2

整份資料共有 9,975 筆資料，且發現 context 欄位有遺失值。

```
04 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9975 entries, 0 to 9974
Data columns (total 11 columns):
id          9975 non-null object
context     9968 non-null object
time        9975 non-null object
shares      9975 non-null int64
likes       9975 non-null int64
love        9975 non-null int64
wow         9975 non-null int64
haha        9975 non-null int64
angry       9975 non-null int64
sad         9975 non-null int64
comments    9975 non-null int64
dtypes: int64(8), object(3)
memory usage: 857.3+ KB
```

圖 9-36 查看資料 info()

處理遺失值

context 內容為空值的填入字串 " 無 "，若不先處理遺失值後續的統計分析將會出錯。

```
05 df['context'] = df['context'].fillna('無')
```

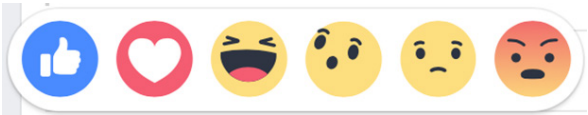
再次查看資料可以發現成功去除空值，每個欄位都有 9,975 筆資料。

```
06 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9975 entries, 0 to 9974
Data columns (total 11 columns):
id          9975 non-null object
context     9975 non-null object
time        9975 non-null object
shares      9975 non-null int64
likes       9975 non-null int64
love        9975 non-null int64
wow         9975 non-null int64
haha        9975 non-null int64
angry       9975 non-null int64
sad         9975 non-null int64
comments    9975 non-null int64
dtypes: int64(8), object(3)
memory usage: 857.3+ KB
```

圖 9-37 查看資料 info()

整理資料



計算按讚數量加總，共有下面六種指標，新增欄位 likes_count，為六種按讚指標的加總。

```
07 df['likes_count'] =
08 df['likes']+df['love']+df['wow']+df['haha']+df['angry']+df['sad']
```

再次查看資料確實成功新增了 likes_count 的欄位。

```
09 df.head(1)
```

	id	context	time	shares	likes	love	wow	haha	angry	sad	comments	likes_count
0	1246 1633 0906 800_ 1560 5011 9731 8299	阿娘威！披羊皮的狼？竟大口嚼小雞\n#要打統編：小編真的是快嚇死了... 😱😱\n\n影片來源...	2017 -06- 05T0 3:09 :40+ 0000	560	1144	11	757	21	292	86	589	2311

圖 9-38 查看資料 head()

整理時間資料

查看原先的時間格式，這是 FB 存放時間的資料格式，要先進行些許轉換後續分析才好使用。

```
10 df['time'][0]
```

```
執行結果：
'2017-06-05T03::09:40+0000'
```

去除多餘的時間符號以及去除秒數後四位。

```
11  df['time'] = df['time'].str.replace('T','').str.replace('-','')
12  df['time'] = df['time'].str.replace(':','').str.split('+').str[0]
13  df['time'][0]
```

執行結果：
```
'20170605030940'
```

轉換成 pandas 中的時間格式 datetime。

```
14  df['time'] = pd.to_datetime(df['time'],format='%Y%m%d%H%M%S')
15  df['time'][0]
```

執行結果：
```
Timestamp('2017-06-05 03:09:40')
```

FB 預設的時區為中原標準時間，也就是倫敦的時區，因此需要額外再加 8 小時才是台灣的時區。

```
16  import datetime
17  df['time'] = df['time']+datetime.timedelta(hours = 8)
18  df['time'][0]
```

執行結果：
```
Timestamp('2017-06-05 11:09:40')
```

新增欄位 hour 存放小時，且轉為 object 資料型態。

```
19  df['hour'] = df['time'].dt.hour
20  df['hour'] = df['hour'].astype('object')
21  df['hour'][0]
```

執行結果：
```
11
```

新增欄位 weekday 存放星期 (預設為數字 0~6)，數字從 0~6 代表星期一至日，而我們將其取代成英文字串較好觀看。

```
22  df['weekday'] = df['time'].dt.weekday
23  df['weekday'] = df['weekday'].replace([0, 1, 2, 3, 4, 5, 6],
24  ['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'])
25  df['weekday'][0]
```

執行結果：
```
'Monday'
```

再次查看資料，成功新增了 hour 與 weekday 兩個欄位。

```
26  df.head(1)
```

	id	context	time	shares	likes	love	wow	haha	angry	sad	comments	likes_count	hour	weekday
0	1246163309060800_1560501197318299	阿娘威！披羊皮的狼？竟大口嚼小雞\n#要打統編：小編真的是快嚇死了...😀😀\n\n影片來源...	2017-06-05 11:09:40	560	1144	11	757	21	292	86	589	2311	11	Monday

圖 9-39 查看資料 head()

取出小編名稱

新聞的粉專都會有個奇特的現象，在每篇文章中都會有個 tag 來標示是哪一位小編所發的文章。這是一個很有趣的現象，我們也特別把它單獨取出來，後續分析看看有沒有什麼有趣的發現。

圖 9-40 取出小編名稱

每篇貼文的最後都會出現 #XX 編，因此加以切開取出，存入變數 curator 的 list 當中。迴圈中使用 if 判斷式是因為不是所有的 # 字號都是小編的名稱，因此要先確定其為小編名稱才加入 list 當中。

```
27 curator = []
28 for i in df['context'].str.split('#').str[1].str.split('：').str[0]:
29     try:
30         if i[-1] == '編':
31             curator.append(i)
32         else:
33             curator.append(None)
34     except:
35         curator.append(None)
```

在 dataframe 當中新增欄位 curator 加入小編名稱。

```
36 df['curator'] = pd.DataFrame(curator)
```

查看資料後確實成功取出小編名。

```
37 df.head(1)
```

	id	context	time	shares	likes	love	wow	haha	angry	sad	comments	likes_count	curator
0	1246163309068000_1560501197318299	阿娘威！披羊皮的狼？竟大口嚼小雞\n#要打統編：小編真的是快嚇死了...😀😀\n\n影片來源...	2017-06-05 11:09:40	560	1144	11	757	21	292	86	589	2311	要打統編

圖 9-41 查看資料 head()

描述性統計

匯入套件

matplotlib 和 seaborn 皆為 Python 的繪圖套件 。

```
38 import seaborn as sns
39 import matplotlib.pyplot as plt
40 from matplotlib.font_manager import FontProperties
```

設定中文字型

繪圖套件本身並不支援中文，因此我們需要事先設定中文的路徑。

OS X 環境 (Mac 用戶)

```
41 font = FontProperties(fname=r'/System/Library/Fonts/STHeiti Light.ttc')
```

Windows 環境

```
41 font = FontProperties(fname=r'C:/windows/Fonts/msjh.ttc')
```

開始進行基本統計計算

連續型資料統計指標：表格之中包含數量、平均、標準差、四分位距以及最大最小值。

```
42 df.describe()
```

	shares	likes	love	wow	haha	angry	sad	comments	likes_count
count	9975.000000	9975.000000	9975.000000	9975.000000	9975.000000	9975.000000	9975.000000	9975.000000	9975.000000
mean	277.510877	3125.289925	115.573634	118.844411	262.752381	147.826065	111.721704	585.611830	3882.008120
std	1171.178025	6770.009492	500.730071	323.255901	1105.812423	942.289195	967.558397	2545.403304	8278.568139
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	6.000000	384.000000	7.000000	8.000000	3.000000	0.000000	0.000000	10.000000	465.000000
50%	40.000000	1017.000000	15.000000	28.000000	11.000000	1.000000	1.000000	52.000000	1246.000000
75%	160.000000	2795.500000	43.000000	95.000000	77.000000	10.000000	8.000000	254.000000	3515.000000
max	39140.000000	144475.000000	11394.000000	8474.000000	30265.000000	30751.000000	48997.000000	64206.000000	156850.000000

圖 9-42 連續型資料統計指標

小編出現次數排名：可以發現 B 編是最常發文的一位小編。

```
43 df['curator'].value_counts().head(11)
```

```
B編          410
內編          397
條紋編         383
悠悠編         299
哩厝編         276
惡魔在身編       276
M編          275
哈姆編         266
周二編         266
閃編          265
西瓜挖大編       248
```

圖 9-43 小編出現次數排名

計算小編數量：發現東森新聞在這段期間內共有 86 位小編！

```
44  len(df['curator'].value_counts().index)
```

```
執行結果：
86
```

針對小編進行進一步分析

以圖表方式呈現發文數量前 10 名的小編。

```
45  sns.countplot(data=df, x='curator',
46      order=df['curator'].value_counts().iloc[:10].index)
47  plt.xticks(fontproperties=font, size=10)
48  plt.title('小編發文數量', fontproperties=font, size=12)
49  plt.show()
```

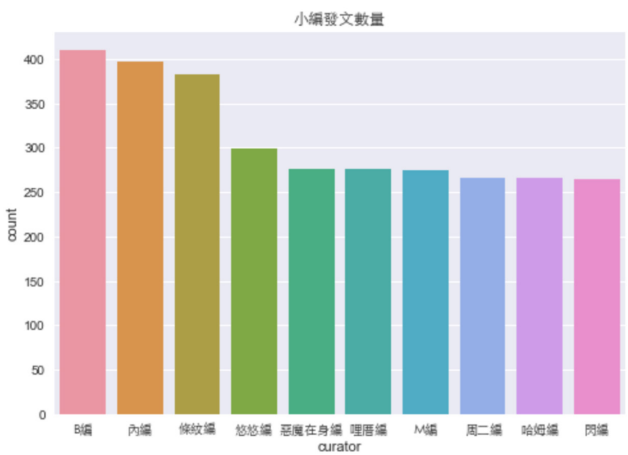


圖 9-44 小編發文數量

以圖表方式呈現發文按讚數量平均前 10 名的小編 (須先計算每一位小編各自的平均按讚數量)。

```
likes_avg = []
for i in df['curator'].value_counts().index:
    likes_avg.append([i,(df[df['curator']==i])['likes_count'].
mean()])

df2 = pd.DataFrame(likes_avg, columns=['curator','mean_likes'])

df3 = df2.sort_values('mean_likes',ascending=False).head(10)
sns.barplot(x='curator', y='mean_likes', data=df3)
plt.xticks(fontproperties=font, size=10)
plt.title(' 按讚數量前 10 名小編 ', fontproperties=font, size=12)
plt.show()
```

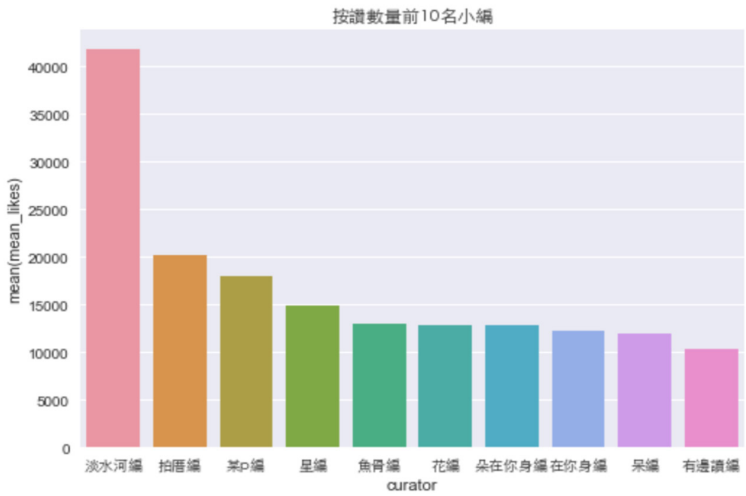


圖 9-45 按讚數量前 10 名小編

時間因子所造成的影響

以圖表方式呈現各個發文時間的統計

```
61  sns.countplot(data=df, x='hour')
62  plt.xticks(fontproperties=font, size=10)
63  plt.title('發文時間統計', fontproperties=font, size=12)
64  plt.show()
```

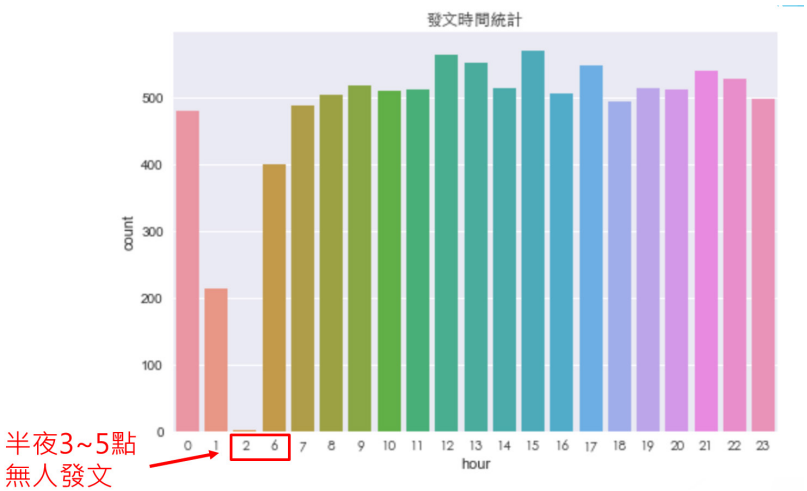


圖 9-46 發文時間的統計

以圖表方式呈現各個時段發文的按讚成效 (須先計算每個時段的按讚平均)。

```
likes_avg_hour = []
for i in df['hour'].value_counts().index:
    likes_avg_hour.append([i,(df[df['hour']==i])['likes_count'].mean()])
df4 = pd.DataFrame(likes_avg_hour, columns=['hour','mean_likes'])
sns.barplot(x='hour', y='mean_likes', data=df4)
plt.xticks(fontproperties=font, size=10)
plt.title(' 各時段發文按讚成效 ', fontproperties=font, size=12)
plt.show()
```

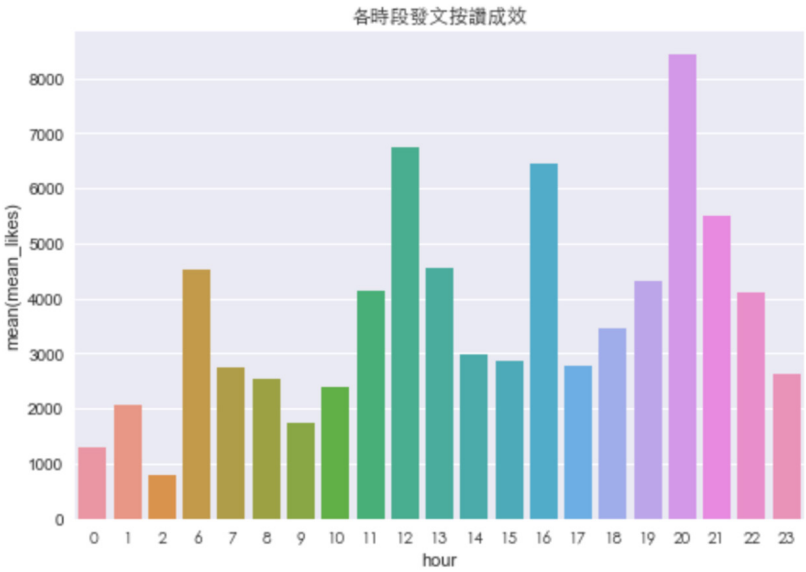


圖 9-47 各個時段發文的按讚成效

以圖表方式呈現各個發文星期的統計。

```
73  sns.countplot(data=df, x='weekday')
74  plt.xticks(fontproperties=font, size=10)
75  plt.title('發文星期統計', fontproperties=font, size=12)
76  plt.show()
```

以圖表方式呈現各個星期發文的按讚成效(須先計算每個星期的按讚平均)。

```
77  likes_avg_week = []
78  for i in df['weekday'].value_counts().index:
79      likes_avg_week.append([i,(df[df['weekday']==i])['likes_count'].mean()])
80  df4 = pd.DataFrame(likes_avg_week, columns=['weekday','mean_likes'])
81  sns.barplot(x='weekday', y='mean_likes', data=df4)
82  plt.xticks(fontproperties=font,size=10)
83  plt.title('各星期發文按讚成效',fontproperties=font,size=12)
84  plt.show()
```

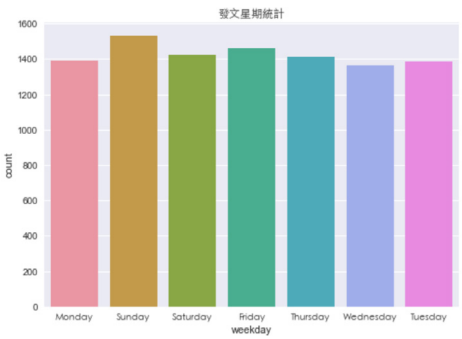
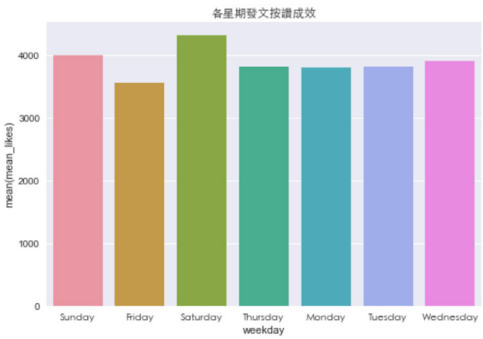


圖 9-48 發文星期的統計圖



9-49 各個星期發文的按讚成效

使用熱點圖 (Heatmap) 看兩兩之間的關係

各個星期 vs. 各個時間點的發文數量。

```
85  df6 = pd.crosstab(df['hour'], df['weekday'])
86  sns.heatmap(df6, annot=True,cmap='Oranges')
87  plt.show()
```

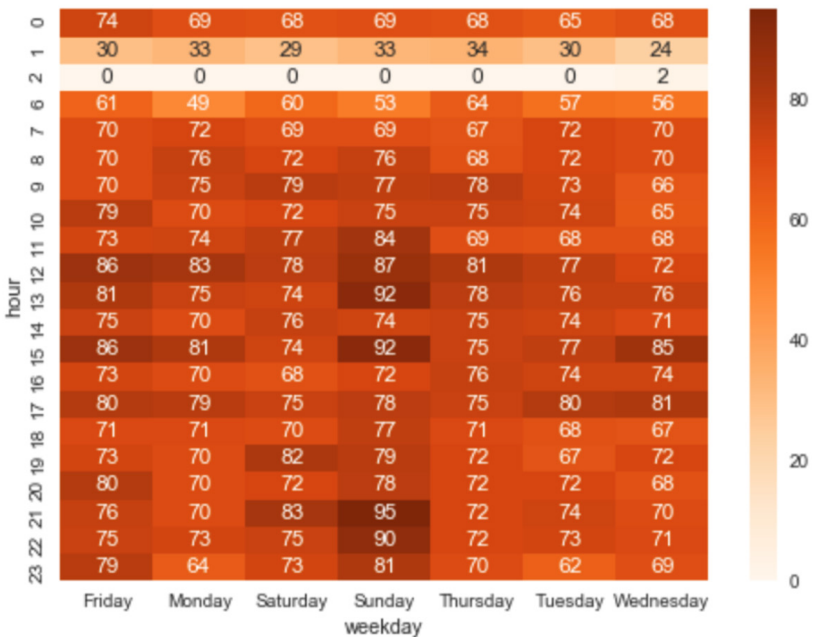


圖 9-50 熱點圖查看關係

各個星期 vs. 各個時間點的按讚數量。

```
88  df7 = pd.pivot_table(df, index=['hour'],
89      columns=['weekday'], values=['likes_count'])
90  sns.heatmap(df7, annot=True, cmap='Oranges')
91  plt.show()
```

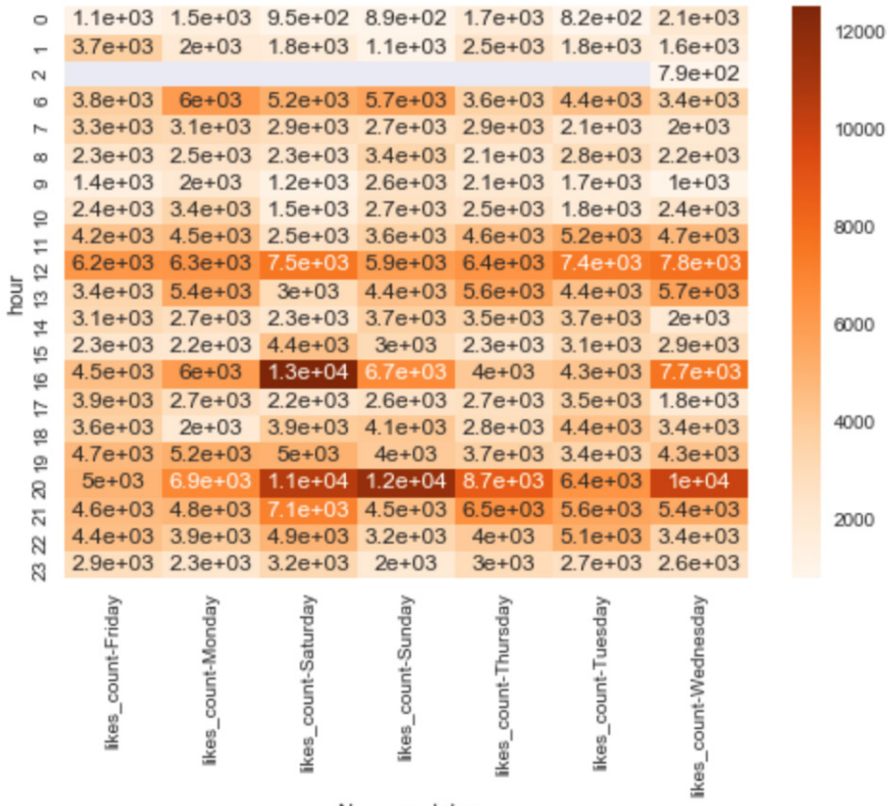


圖 9-51 熱點圖查看關係

匯出檔案 :

最後匯出經過整理好的資料

```
92 df.to_excel('fanpage_clean.xlsx', index=False)
```

小結

我們在本節中示範了 Python 清理資料的方法以及使用圖表呈現一些統計性分析，當然各位讀者也可以針對各自有興趣的欄位畫圖查看有無有趣的發現，我們在這邊只列出幾張統計性圖表而已。以上述的分析為例，可以發現常常發文的小編顯然按讚的迴響並不高，反而只要淡水阿編一發文就可以收到很大的按讚數量，當然這也與發文次數的多寡有關係，不過藉由這種分析就可以很容易的查看不同變量間的關係。此外從時間與按讚數量的關係可以發現在中午十二點、下午四點以及晚上八點能收到最大的按讚數量，推測原因可能是中午午休時間、快下班的時間以及吃飽飯後是最多人會打開 FB 查看訊息的時間點，因此藉由這些簡單的統計圖表，其實就可以帶給東森新聞粉專的小編很多不同的資訊，利用這些資訊來調整發文的方式等等。甚至更深入一點可以針對留言與分享數量進行分析，或是發文的議題為何等等，這些部分就留給讀者繼續思考囉！

9-4 資料工程：文字探勘 Text mining

本節將會為各位讀者介紹什麼是文字探勘以及如何使用 Python 進行一些簡易的實作。文字探勘簡而言之就是針對文字的資料進行處理與分析，而我們使用的資料同樣是上一章節的東森新聞粉專資料；接著將從文字的斷字斷詞開始帶領實作，同時帶入兩款處理文字常見的演算法 (TFIDF, Word2Vec)，以下將會一一詳細介紹。

什麼是文字探勘

文字探勘其實是自然語言處理 : (Natural Language Process, NLP) 的其中一項範疇。自然語言處理的研究歷史相當久遠，就維基百科顯示最早可以追溯到 1950 年代，其所需具備的領域知識更是廣闊，包含計算機科學、統計學、人工智慧、語言學等等。不過現階段自然語言處理的應用其實已經融合在我們的日常生活當中囉！像是我們常常會使用的 Google 翻譯、手機的輸入法選字等等 (圖 9-52)

圖 9-52 Google 翻譯、手機的輸入法

1. 文本朗讀（Text to speech）/語音合成（Speech synthesis）
2. 語音識別（Speech recognition）
3. 自動分詞（word segmentation）
4. 詞性標註（Part-of-speech tagging）
5. 句法分析（Parsing）
6. 自然語言生成（Natural language generation）
7. 文本分類（Text categorization）
8. 信息檢索（Information retrieval）
9. 信息抽取（Information extraction）
10. 文字校對（Text-proofing）
11. 問答系統（Question answering）
12. 機器翻譯（Machine translation）
13. 自動摘要（Automatic summarization）
14. 文字蘊涵（Textual entailment）

圖 9-53 自然語言處理應用範疇

自然語言處理的應用範疇就像圖 (圖 9-53) 非常廣闊 (參閱維基百科 https://zh.wikipedia.org/wiki/ 自然語言處理)，而我們今天所說的文字探勘 (text mining) 也被稱為文本挖掘、文字採礦、智慧型分析等，其為自然語言處理的其中一部分領域。學術上的表達也就是從非結構化的文字資訊中，萃取出重要的資訊或知識；若白話一點說明則是試著從大量的文章中找出隱含的寶藏、有用的資訊。

- 結構化資料
  --> Excel, csv, json 檔...
- 非結構化資料
  --> 文字檔，圖片檔，影音檔...

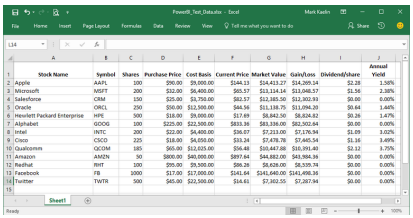


圖 9-54 結構化和非結構化資料

文字探勘進行的方式

現在我們已經稍微理解什麼是文字探勘了，接下來我們來談談究竟電腦 (機器) 是怎麼閱讀人類的文字呢？首先第一步要先理解的是文字的斷字斷詞，處理文字前一定要先能分辨一個句子或一篇文章中的所有單字

以英文為例，單字與單字中間有空格分開，只要透過程式將空白處分別切開，即可取得單獨的單字。那中文的處理方式呢？現在我們有以下例句，他應該是由四個不同的單詞所組成的，但該怎麼將他們分開？

例 : 我來自台灣東吳大學

我 / 來自 / 台灣 / 東吳大學

解決方式就是我們必須擁有一份中文字詞的字典，而這份字典裡需要有中文單字、權重與詞性，若權重越高代表優先切開的字詞。為什麼會需要有權重呢？我們來看以下例子：

例 : XX 是開發中國家

1. XX / 是 / 開發中國家
2. XX / 是 / 開發中 / 國家
3. XX / 是 / 開發 / 中國 / 家

開發中國家這五個字詞可以是單純的一個字，也可以是由開發中、國家兩個字所組成，甚至可以是開發、中國與家三個字所組成，這就是中文在斷字斷詞中最困難的地方，因此我們需要有權重的機制幫助我們決定優先切開的字詞為何。這份字典的範例如圖 (圖 9-55) 所示：

圖 9-55 字典範例

當成功斷完中文的字詞，事情並沒有這麼簡單就結束了，還必須要去除停止詞 (stopwords) 才行。什麼是停止詞呢？也就是在中文當中沒有意義的字詞，例如我、你、它、當、的等等，以「這堂課是文字探勘」為例，正確的切出字詞應該是 (這堂、課、文字探勘) 才對。因此我們同樣需要一份停止詞的字典來告訴我們什麼樣的詞是沒有意義的 (圖 9-56)

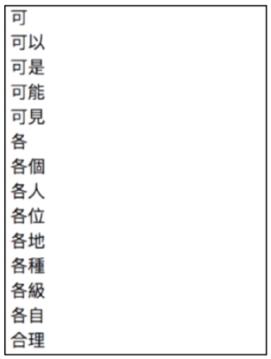


圖 9-56 停止詞字典

這些字典都已經有人整理過並公開在網路上開放下載，當然大家整理的字典都會有些許不同，甚至有些專有名詞並不會被收錄在這種通用型的字典裡面，因此當今天如果我們要處理的文字是醫學期刊的文章，那字典就需要另外再找，甚至是自己建立了！

文字探勘套件介紹

NLTK 套件

```
01 | import nltk
```

nltk 是 Python 非常大的自然語言處理套件庫，包含的套件非常多，若全部下載將會非常花時間，因此我們個別下載需要用的就好。

```
02 | nltk.download()
```

跳出新的下載視窗並點選 All Packages

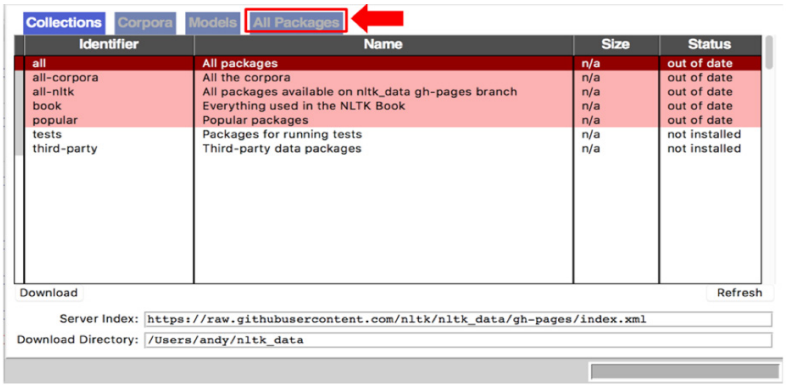


圖 9-57 點選 All Packages

往下滑找到 stopwords，點選左下角 Downloads 即下載完成。

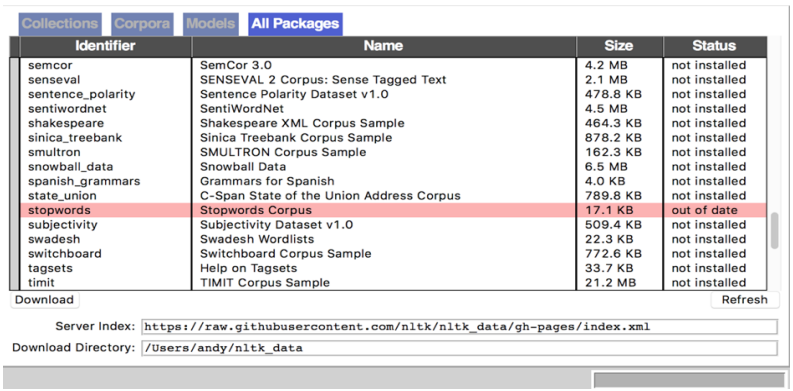


圖 9-58 找到 stopwords

Jieba 套件介紹

Jieba 是 Python 中處理中文斷字斷詞的套件，其他還有像是 Stanford CoreNLP 以及中研院的 CKIP 等等，在這邊選用 Jieba 是因為其安裝方便快速且使用起來非常簡單，那我們就直接來看程式吧！

匯入 Jieba 套件以及兩份下載好的字典

```
03 import jieba
04 from nltk.corpus import stopwords
05 jieba.set_dictionary('dict.txt.big.txt')
06 stop=stopwords.words('stop_words2.txt')
```


在 Jieba 當中有三種斷字斷詞的模式

全模式：會把所有認為可能是單字的詞都切出來。

```
07 word = jieba.cut('我來到台灣台北東吳大學', cut_all=True)
08 print('/'.join(word))
```

執行結果：
我 / 來到 / 台 / 灣 / 台北 / 北東 / 東吳 / 東吳大學 / 大學

精確模式：只會切出最有可能在這個句子中的單字。

```
09 word = jieba.cut('我來到台灣台北東吳大學', cut_all=False)
10 print('/'.join(word))
```

執行結果：
我 / 來到 / 台灣 / 台北 / 東吳大學

PS：若無設定 cut_all 參數，默認為精確模式。

搜尋模式：切出有可能會被搜尋的字詞。

```
11 word = jieba.cut_for_search('我來到台灣台北東吳大學')
12 print('/'.join(word))
```

執行結果：

```
我 / 來到 / 台灣 / 台北 / 東吳 / 大學 / 東吳大學
```

各位可以試著到網路上找一篇長篇文章，使用精確模式看切出來的字詞效果如何。就此篇文章而言可以發現有些棒球的專業術語並未成功切出。

```
13 word = jieba.cut_for_search("""目前暫居龍頭的中信兄弟今
14 天在主場迎戰富邦悍將，1 局下黃衫軍精神領袖彭政閔就擊出 3 分砲，
15 助隊在開賽就攻下 3 比 0 領先。8 月 6 日是洽洽的 40 歲生日，在年紀
16 越來越大的情況下，近年不斷傳出他可能從球員身分退役的消息，雖已
17 接近不惑之年，彭政閔本季目前為止打擊率仍有 3 成 76，還看得到年
18 輕時期「4 割男」的影子。彭政閔本季至今天賽前 229 打數擊出 72 支
19 安打包含 4 發全壘打、貢獻 36 打點、打擊率 3 成 76，上回洽洽開轟是
20 6 月 14 日面對富邦悍將投手張耿豪所擊出。今天同樣是面對富邦，苦
21 主則換成前隊友五鐸 (Bryan Woodall)。( 中時電子報 )""")
22 print('/'.join(word))
```

執行結果：

```
目前 / 暫居 / 龍頭 / 的 / 中信 / 兄弟 / 今天 / 在 / 主場 / 迎戰 / 富邦 / 悍將 / ，
/1/ 局下 / 黃 / 衫 / 軍 / 精神領袖 / 彭政閔 / 就 / 擊出 /3/ 分 / 砲 / ，/ 助隊 / 在 / 開賽
/ 就 / 攻下 /3/ 比 /0/ 領先 / 。/8/ 月 /6/ 日 / 是 / 洽洽 / 的 /40/ 歲 / 生日 / ，/ 在 /
年紀 / 越來越 / 大 / 的 / 情況 / 下 / ，/ 近年 / 不斷 / 傳出 / 他 / 可能 / 從 / 球員 / 身
分 / 退役 / 的 / 消息 / ，/ 雖 / 已 / 接近 / 不惑之年 / ，/ 彭政閔 / 本季 / 目前 / 為止 /
打擊率 / 仍 / 有 /3/ 成 /76/ ，/ 還看 / 得到 / 年輕 / 時期 / 「/4/ 割 / 男 / 」/ 的 / 影子 / 。
/ 彭政閔 / 本季 / 至 / 今天 / 賽前 /229/ 打數 / 擊出 /72/ 支安打 / 包含 /4/ 發 / 全壘打 / 、
/ 貢獻 /36/ 打點 / 、/ 打擊率 /3/ 成 /76/ ，/ 上 / 回洽 / 洽開 / 轟 / 是 /6/ 月 /14/ 日 /
面對 / 富邦 / 悍將 / 投手 / 張耿豪 / 所 / 擊出 / 。/ 今天 / 同樣 / 是 / 面對 / 富邦 / ，/
苦主 / 則 / 換成 / 前 / 隊友 / 五鐸 / (/Bryan/ /Woodall/) / 。/ (/ 中時 / 電子報 /)
```

加入自定義字典

Jieba 除了有內建辭庫外，也可以自行加入沒有在字典的字詞，也就是有不同的語料庫。因此我們打開記事本，輸入剛剛文章中沒有切好的詞 (圖 9-60)。

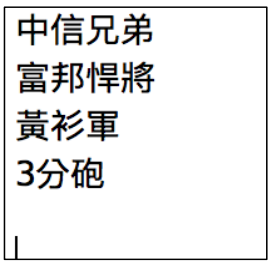


圖 9-60 沒有切好的字

接著重新載入自定義的字典

```
23  jieba.load_userdict('addword.txt')
```

重跑一次剛剛的程式，就可以發現正確切開了新加入的字詞。

```
24  word = jieba.cut_for_search("""目前暫居龍頭的中信兄弟今
25  天在主場迎戰富邦悍將，1局下黃衫軍精神領袖彭政閔就擊出3分砲，
26  助隊在開賽就攻下3比0領先。8月6日是洽洽的40歲生日，在年紀
27  越來越大的情況下，近年不斷傳出他可能從球員身分退役的消息，雖已
28  接近不惑之年，彭政閔本季目前為止打擊率仍有3成76，還看得到年輕
29  時期「4割男」的影子。彭政閔本季至今天賽前229打數擊出72支安
30  打包含4發全壘打、貢獻36打點、打擊率3成76，上回洽洽開轟是6
31  月14日面對富邦悍將投手張耿豪所擊出。今天同樣是面對富邦，苦主
32  則換成前隊友五鐸(Bryan Woodall)。(中時電子報)""")
33  print('/'.join(word))
```

執行結果：

目前/暫居/龍頭/的/中信兄弟/今天/在/主場/迎戰/富邦悍將/，/1/局下/黃衫軍/精神領袖/彭政閔/就/擊出/3/分砲/，/助隊/在/開賽/就/攻下/3/比/0/領先/。/8/月/6/日/是/洽洽/的/40/歲/生日/，/在/年紀/越來越/大/的/情況/下/，/近年/不斷/傳出/他/可能/從/球員/身分/退役/的/消息/，/雖/已/接近/不惑之年/，/彭政閔/本季/目前/為止/打擊率/仍/有/3/成/76/，/還看/得到/年輕/時期/「/4/割/男/」/的/影子/。/彭政閔/本季/至/今天/賽前/229/打數/擊出/72/支安打/包含/4/發/全壘打/、/貢獻/36/打點/、/打擊率/3/成/76/，/上/回洽/洽開/轟/是/6/月/14/日/面對/富邦悍將/投手/張耿豪/所/擊出/。/今天/同樣/是/面對/富邦/，/苦主/則/換成/前/隊友/五鐸/(/Bryan/ /Woodall/)/。/(/中時/電子報/)

去除停止詞

除了將中文字詞切開外，第二步要做的就是去除停止詞。

- 去除前：

```
34  word = jieba.cut('目前暫居龍頭的中信兄弟今天在主場迎戰富邦悍將')
35  print('/'.join(word))
```

執行結果：

目前/暫居/龍頭/的/中信兄弟/今天/在/主場/迎戰/富邦悍將

- 去除後：

```
word = jieba.cut('目前暫居龍頭的中信兄弟今天在主場迎戰富邦悍將')
stoptext = ''
for words in word:
    if words not in stop:
        stoptext += '/'+words
print(stoptext)
```

執行結果：

```
/暫居/龍頭/中信兄弟/主場/迎戰/富邦悍將
```

分析粉絲專頁資料

我們已經介紹完如何處理中文的斷字斷詞以及去除停止詞了，那接下來就來實際跑跑看真實的資料吧！首先匯入 pandas 套件與粉專資料，其中 `fanpage_clean.xlsx` 是前一節創造的成果。

```
import pandas as pd
df = pd.read_excel('fanpage_clean.xlsx')
df.head()
```

	id	message
0	124616330906800_1560501197318299	阿娘威！披羊皮的狼？竟大口嚼小雞 #要打統編：小編真的是快嚇死了...😂😂 影片來源：騰訊視頻 #草食性 #羊
1	124616330906800_1560454417322977	被黑了！李毓芬演唱「大落拍」網友卻意外發現「亮點」 #條紋編：這一段應該是昨天的亮點表演之一吧～ #李毓芬 #落拍 #唱歌
2	124616330906800_1559870414048044	誰說牠呆？心機月月調虎離山 網友讚影帝 #樂無編：最萌心機鬼～(*ˊ∀ˋ)~❤ 影片來源：秒拍 #萌 #心機 #調虎離山計

圖 9-64 粉專資料

因為後面使用到的 TFIDF 與 Word2Vec 演算法所需要的字詞格式並不同，因此在這邊將匯入的資料做斷字斷詞，分成兩部分 (`text`, `text2`)。第一層 `for` 迴圈處理的是每一篇文章中的 Jieba 斷詞，第二層 `for` 迴圈處理的是判斷每一個字是否為停止詞，最後將結果存入 `jieba_text` 的列表當中。

```
jieba_text = []
for index in range(len(df)):
    words = jieba.cut(str(df['context'][index]))
    text, text2 = [], ''
    for word in words:
        if word not in stop:
            text.append(word)
            text2 += ' '+word
    jieba_text.append([text,text2,len(text)])
```

圖 9-65 將匯入的資料做斷字斷詞

結果呈現，放回原本的 Dataframe 裡面。

```
df['jieba_text'] = pd.DataFrame(jieba_text)[0]
df['jieba_text2'] = pd.DataFrame(jieba_text)[1]
df['jieba_count'] = pd.DataFrame(jieba_text)[2]
df.head(3)
```

	id	message	jieba_text	jieba_count	jieba_text2
0	124616330906800_1560501197318299	阿娘威！披羊皮的狼？竟大口嚼小雞\n#要打統編：小編真的是快嚇死了...😂😂\n\n影片來源...	[阿娘, 威, 披, 羊皮, 狼, 竟大口, 嚼, 小雞, \n, #, 統編, 小編, 真...	30	阿娘 威 披 羊皮 狼 竟大口 嚼 小雞 \n # 統編 小編 真的 快 嚇死 ... 😂...
1	124616330906800_1560454417322977	被黑了！李毓芬演唱「大落拍」網友卻意外發現「亮點」\n#條紋編：這一段應該是昨天的亮點表演...	[黑, 李毓芬, 演唱, 「, 大落, 拍, 」, , 網友, 卻, 意外, 發現, 「,...	37	黑 李毓芬 演唱 「 大落 拍 」 網友 卻 意外 發現 「 亮點 」 \n # 條紋...
2	124616330906800_1559870414048044	誰說牠呆？心機月月調虎離山 網友讚影帝\n#樂無編：最萌心機鬼～(*´∀`)~♥\n\n影片...	[誰, 說, 牠, 呆, 心機, 月, 月, 調虎離山, , 網友, 讚, 影帝, \n,...	43	誰 說 牠 呆 心機 月 月 調虎離山 網友 讚 影帝 \n # 樂無編 最萌 心機 ...

圖 9-66 結果呈現

TF-IDF 演算法

TF-IDF

接下來我們將實作兩種處理文字的演算法，首先是 TF-IDF。TF-IDF 想要達成的目標是找出一段文字或一篇文章中有哪些字詞是較重要的。它其實是 TF 和 IDF 的結合，公式如下看起來很複雜，不過我們將其分開來介紹。

Term Frequency (TF)

TF 計算的是一個字詞 t 在一篇文章 d 中所出現的總次數 $tf_{t,d}$，並取 log 降維，但文字出現的越多不代表越重要，有可能是不重要的字詞　如 : 我、你、它等等 (若沒事先去除停止詞的話)。

$$\text{TF} = \begin{cases} 1+\log_{10} tf_{t,d} & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Inverse Document Frequency (IDF)

IDF 計算的是 [總文件數 / 文字出現在文件內的次數]，並取 log。也就是把雖然出現頻率很高但卻是不重要的字詞降低權重。

$$\text{IDF} = \log_{10} \frac{N}{df_t}$$

Term	df_t	IDF (N=1000000)
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

圖 9-67 Inverse Document Frequency

所以 TF-IDF 就是將 TF 與 IDF 兩個相乘 (其中一種算法，其實還有很多衍伸型)。

$$\text{TF} - \text{IDF} = \text{TF} \times \text{IDF} = (1+\log(tf_{t,d})) \times \log(N / df_t)$$

如果還是看不太懂公式，下圖表達應該較好理解 (圖 9-68)。

Word	出現總次數	出現文件數	
ferrari	10422	17	←較高的稀有性 (高資訊量)
insurance	10440	3997	

圖 9-68 TF-IDF

如果已經稍微理解TF-IDF是什麼了的話，那我們就進入Python的實作吧！

首先我們需要 scikit-learn 這個套件，安裝後匯入其中的 TF-IDF 模組：

```
from sklearn import feature_extraction
from sklearn.feature_extraction.text import TfidfVectorizer
```

接著將文字轉換成矩陣模式，使用到的文字格式是剛剛處理好以空白切開每個字詞的字串格式。

```
vectorizer = TfidfVectorizer()
tfidf = vectorizer.fit_transform(df['jieba_text2'])
```

計算 TF-IDF，我們顯示每篇文章中的前五個關鍵字。

```
words = []
words2 = vectorizer.get_feature_names()
for i in tqdm_notebook(range(len(df['jieba_text2']))):
    print('======= Post'+str(i+1)+' =======')
    temp_array = tfidf[i,:].toarray()
    for l in temp_array:
        print([(words2[x],l[x]) for x in (l*-1).argsort()][:5])
```

執行結果：

```
======= Post1 =======
[('草食性', 0.3886164663330653), ('竟大口', 0.3886164663330653),
('小雞', 0.37205588933099115), ('羊皮', 0.37205588933099115),
('阿娘', 0.31216568557942365)]
======= Post2 =======
[('李毓芬', 0.5393296772291432), ('亮點', 0.47428487993368196),
('大落', 0.28166788829793676), ('昨天', 0.2611485393335714), ('
演唱', 0.2611485393335714)]
======= Post3 =======
[('心機', 0.6875944679814824), ('調虎離山', 0.5748856520127635),
('影帝', 0.25649296680540506), ('最萌', 0.22919815599382748),
('樂無編', 0.2181098701502329)]
```

將執行的結果存到變數 words 裡面。

```
words = []
words2 = vectorizer.get_feature_names()
for i in tqdm_notebook(range(len(df['jieba_text2']))):

    temp_array = tfidf[i,:].toarray()
    for l in temp_array:
        words.append([(words2[x],l[x]) for x in (l*-1).argsort()][:5])
```

轉成 DataFrame 形式。

```
# 每篇文章的前五關鍵詞
df2 = pd.DataFrame(words,columns=['Keyword1','Keyword2',
                    'Keyword3','Keyword4','Keyword5'])
df2.head()
```

與原本的資料合併。

```
df3 = pd.concat([df, df2], axis=1)
df3.head(1)
```

	id	message	jieba_message	Keyword1	Keyword2	Keyword3	
0	124616330906800_1560501197318299	阿娘威！披羊皮的狼？竟大口嚼小雞\n#要打統編：小編真的是快嚇死了...😊😊\n\n影片來源...	/阿娘/威/披/羊皮/狼/竟大口/嚼/小雞/\n/統編/小編/真的/嚇死/\n/\n/\n/...	(草食性, 0.4165732670138584 4)	(竟大口, 0.4165732670138584 4)	(羊皮, 0.3988213335189045)	(小雞, 0.398821

圖 9-69 顯示結果

評估 TF-IDF 的結果好壞有一種方法：「若我們只看這五個或是額外延伸至十個關鍵字是否有辦法猜出原本的文章大致內容呢？」就從目前的結果而言，可以發現還是有斷字斷詞並未處理乾淨的狀況發生。處理中文字詞最難的地方往往是在斷字斷詞身上，也有很多研究針對如何讓電腦自行加入自定義的字詞等，或是也可以嘗試用用看別種的中文斷詞方法，也許最後的結果也會不同，這邊就留給有興趣的讀者繼續研究囉！

Word2Vec 演算法

接下來我們來介紹另一種處理文字的演算法 Word2Vec。還記得我們前面有提到究竟電腦 (機器) 是怎麼閱讀人類的文字嗎？關鍵就在於將文字向量化 (Word Embedding) ！電腦終究只看得懂數字符號而已，因此需要有方法將文字資料轉換成數學向量模式，其中一種方式就是透過 Word2Vec 演算法 (Word to Vector)。

Word2Vec 概念

我們可以想像把所有的文字字詞轉換成向量投射到高維度的空間中，在這個高維度的空間中越接近的文字其相似度應該會越高，且類似的文字應該會有相同的距離，如下左圖 (圖 9-70) (MAN 與 WOMAN 的距離等同於 KING 與 QUEEN 的距離)。此外文字的單複數也同樣適用於這個規則，如下右圖 (圖 9-70)(KING 與 QUEEN 的複數應該擁有相同的空間距離。)

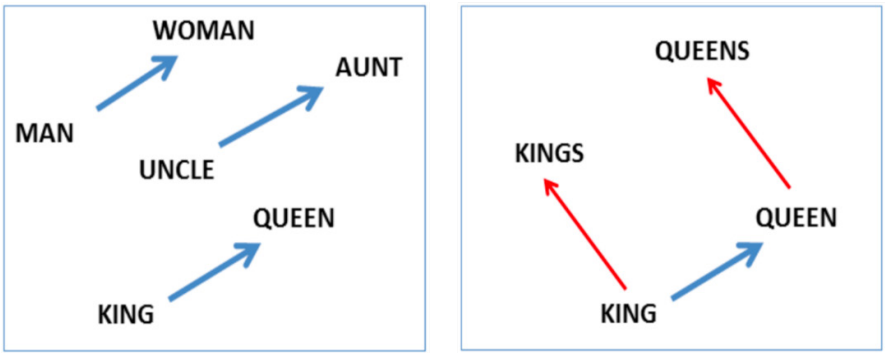


圖 9-70

Word2Vec 在做的就是找尋字與字之間的關聯性，以及讓同概念的詞其距離能夠越接近越好。每一個字詞都是在空間中的一個向量，其維度相對也是屬於高維的向量。又如同下圖的範例 (圖 9-71)，國家之於其首都大致都是同樣的距離，而國家字詞都在左邊，城市之詞都在右邊，Word2Vec 就是在實現這種想法。

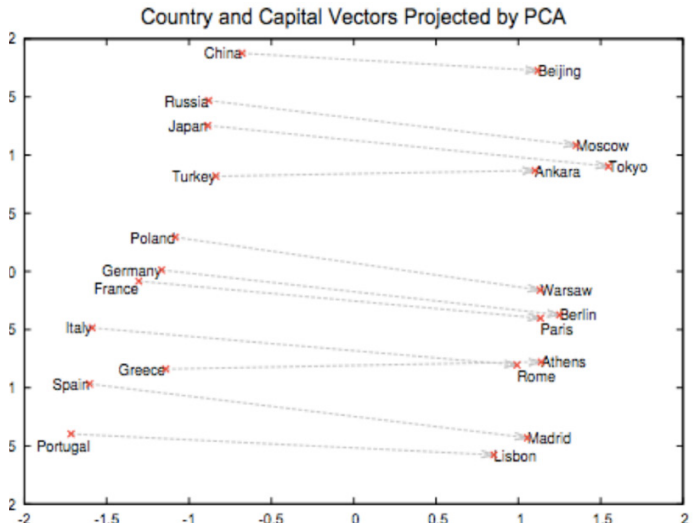


圖 9-71

Word2Vec 計算距離的方法是向量間的餘弦 (cos) 值，也就是夾角角度。

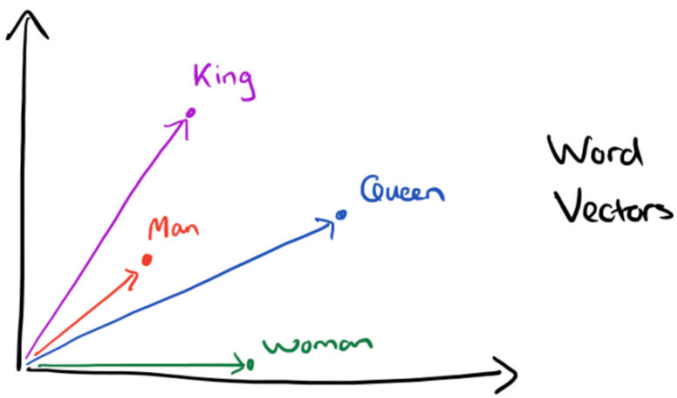


圖 9-72

若對 Word2Vec 有興趣，想要了解其詳細演算過程，可以再從網路上找尋其它教材，已有很多網友整理與介紹，在這邊就不一一詳細講解囉，我們就直接帶領各位進行 Python 的實作吧。

首先，我們會需要 genism 這個套件，安裝完成後匯入 Word2Vec 模組。

```
41 from gensim.models.word2vec import Word2Vec
```

接著建立 Word2Vec 模型，使用到的文字格式是先前處理好以串列 list 儲存每個字詞的格式，執行完這行，模型就建立完成囉！

```
42 model = Word2Vec(df['jieba_text'])
```

另外建立一個函式來顯示不同字詞前 10 個相關的字詞有哪些，for 迴圈依序處理每一個要搜尋的字詞，wv.most_similar 則是會判斷最接近此字詞的前 n 個字並存入 similar_words 變數當中，最後以 Dataframe 方式呈現。倘若字詞並未出現在模型當中，則會顯示 not found in Word2Vec model! 字詞

```
43 def most_similar(w2v_model, words, topn=10):
44     similar_df = pd.DataFrame()
45     for word in words:
46         try:
47             similar_words = pd.DataFrame(
48             w2v_model.wv.most_similar(word, topn=topn),
49             columns=[word, 'cos'])
50 
51             similar_df = pd.concat([similar_df, similar_words], axis=1)
52         except:
53             print(word, 'not found in Word2Vec model!')
54     return similar_df
```

查看其結果，以這 9 個字詞來看各自最接近的前 10 字詞有哪些：

```
55 most_similar(model, ['新聞','體育','娛樂','政治',
56 'XDD','小編','夜市','明星','女孩'])
```

	新聞	cos	娛樂	cos	政治	cos	XDD	cos	小編	cos	夜市	cos	明星	cos	女孩	cos
0	直播	0.967324	S	0.998950	地方	0.994920	哩厝編	0.994200	好	0.984673	環島	0.999220	技巧	0.999714	噴發	0.998853
1	中心	0.965926	櫻花	0.998900	社會	0.994582	狗狗	0.991577	萌死	0.978975	飯店	0.999144	髮	0.999616	爸爸	0.998728
2	綜合	0.956663	起司	0.998894	T	0.990361	柯基	0.991063	曉	0.977118	高空	0.999091	最美	0.999566	Q	0.998684
3	育樂中心	0.943752	役	0.998769	NCC	0.990037	XDDD	0.990727	想	0.972329	餐廳	0.999069	安娜	0.999490	抱	0.998549
4	愛玩	0.932010	害怕	0.998734	壯壯	0.989895	誰准	0.990455	人	0.971244	團	0.999040	茶	0.999484	洋蔥	0.998500
5	吳念樺	0.928818	飛行	0.998684	禽流感	0.987717	好笑	0.989693	寶寶	0.970022	鍋貼	0.998937	V	0.999439	心	0.998497
6	朱姵慈	0.928699	手工	0.998654	肇逃	0.987193	愛	0.988566	吃	0.963738	驚悚	0.998870	倍	0.999332	床	0.998367
7	台	0.917264	抓到	0.998643	侵權	0.986938	養樂多	0.988533	真的	0.962447	巴黎	0.998865	父	0.999247	有趣	0.998241
8	提醒您	0.909181	性別	0.998492	毒品	0.986920	牠	0.988344	Jay	0.962223	跑車	0.998857	編髮	0.999242	瞬間	0.998178
9	李欣	0.907590	約會	0.998433	巨蛋	0.986402	Baby	0.988283	長	0.959666	板車	0.998834	馬尾	0.999203	挑戰	0.998119

圖 9-73 最接近的前 10 字詞

我們在建立模型時可以修改幾個參數

1. `size` 代表詞向量的維度大小

2. `iter` 代表訓練的次數多寡

修改後會發現跟原先跑出的結果也不盡相同

```
57  model_d250 = Word2Vec(df['jieba_text'], size=250, iter=10)
58  most_similar(model_d250, ['新聞','體育','娛樂','政治',
59  'XDD','小編','夜市','明星','女孩'])
```

```
體育 not found in Word2Vec model!
```

	新聞	cos	娛樂	cos	政治	cos	XDD	cos	小編	cos	夜市	cos	明星	cos	女孩	cos
0	Nick	0.908591	Keigo	0.995358	監獄	0.978106	愛	0.963409	好	0.927528	逢甲	0.979087	原味	0.994482	掛在	0.968459
1	節目	0.908183	Youtube	0.991555	香港	0.972059	牠	0.961356	想	0.902362	溫泉	0.971208	跟風	0.994479	汪汪	0.964284
2	直播	0.898731	邦	0.990808	美國	0.972043	可愛	0.958527	感覺	0.901491	步道	0.968835	上身	0.994263	卡哇伊	0.956831
3	愛玩	0.891666	紙牌	0.990400	青瓦台	0.971166	xD	0.957824	真的	0.870075	牛肉	0.967882	出奇	0.994263	這招	0.956034
4	房趣	0.887777	く	0.989025	谷	0.967853	XDDDD	0.948542	人	0.869911	殺手	0.967501	出招	0.993517	主人	0.955768
5	調查	0.821492	媒合	0.988894	汙	0.967761	嘗試	0.943750	媽媽	0.849948	沖繩	0.966578	境界	0.993509	表情	0.955542
6	中心	0.813243	ι	0.988487	中部	0.966782	der	0.940438	厲害	0.846849	品牌	0.966445	完美	0.993505	挑戰	0.953859
7	挑	0.810430	Mihara	0.988282	政府	0.966702	嘴編	0.937272	耶	0.843780	駁火	0.966388	放閃	0.993127	小	0.953715
8	車	0.797249	咖哩	0.988211	跨界	0.964899	脆嘀叔	0.933207	好吃	0.841185	商圈	0.965920	材料	0.993045	隻	0.952203
9	馨	0.788445	錕	0.988182	車站	0.964738	萌死	0.930417	統編	0.840771	麻辣鍋	0.965484	電眼	0.992904	肚臍	0.951553

圖 9-74 修改後結果

Word2Vec 屬於神經網路的一環，這類型的演算法都很仰賴大量的資料進行訓練，越多的資料對於其結果會越準確，不過同時也需要有好的設備才容易進行訓練，此外也同樣需要準確的斷字斷詞結果才行。

儲存模型

當資料量很大時避免每次都要重複訓練模型，可以先儲存起來

```
52 model.save('word2vec.model')
```

當下次還想使用時可以直接匯入模型，再進行其他的分析

```
01 from gensim.models.word2vec import Word2Vec
02 model = Word2Vec.load('word2vec.model')
```


小結

我們在本節中介紹了文字探勘的一些基本概念，以及處理中文斷字斷詞的流程，另外也介紹了兩個與文字相關的演算法。從兩者最後的結果當中我們可以得知在進行這類型的演算時有幾個重要的先決條件，其一為需要擁有準確的斷字斷詞前處理，其二為大量的資料對於訓練的結果影響也很巨大，這一點與下一節的機器學習相同。不過在這邊我們希望各位讀者能夠吸收到的是如何使用 Python 程式來實現這些分析方法，當要套用回自己的資料時是可以輕易跑出結果的，同時也希望各位讀者能夠有能力解讀跑出來的結果所代表的意義。有時候我們並非是高深的統計學家，甚至可能也沒學過微積分和線性代數等等課程，因此若要理解演算法背後的運算方式是有些難度的。但若我們至少能夠理解演算法的概念，以及修改的參數對於結果會造成哪些影響，最重要的是能夠解釋運算結果所呈現的意義，就大部分的應用層級而言已經是相當足夠的了。若讀者並非是數學背景出生的，建議可以朝這個方向前進呦！

9-5 進階分析：資料建模 Data modeling

本節將為各位讀者介紹什麼是機器學習、機器學習有哪些演算法、如何透過 Python 建立機器學習模型進行預測、以及可以透過哪些指標來評估模型的好壞。我想近幾年各位對於大數據這一詞一定不陌生，不論是從哪個領域當中我們都可以發現大數據的蹤跡，像是商業行為、醫療、交通、犯罪、工廠等等。大數據的背後與本章節所提及的機器學習息息相關，綜觀而言，機器學習其實是 2、30 年前就已經出現的技術，近年來電腦運算的速度不斷進步、成本也逐漸下降，不論是企業、學校、乃至於個人現階段都有能力使用具有高速運算能力的電子產品，同時使用機器學習技術的門檻也逐漸降低，我們並不需要是資訊背景、統計背景的專家才可以使用，只要有能力撰寫 Python、R 等程式語言都可以輕易實現，不過當然若能理解機器學習背後的運算邏輯會更有幫助。現在我們就直接進入我們的最後一節：資料建模吧！

機器學習的出現

從圖 9-75 中我們可以發現早在 1950 年代就開始出現人工智慧 AI 的想法了，那時候因為戰爭的影響因此各國都很仰賴電腦能幫助他們進行作戰，而到了 1980 年代逐漸出現了所謂的專家系統，也就是在那陣子開始出現機器學習的概念，希望電腦可以習得人類的相關知識來簡化人類的負擔，而到了 2012 年因為一場電腦視覺競賽的結果造就了現階段最夯的深度學習出現。

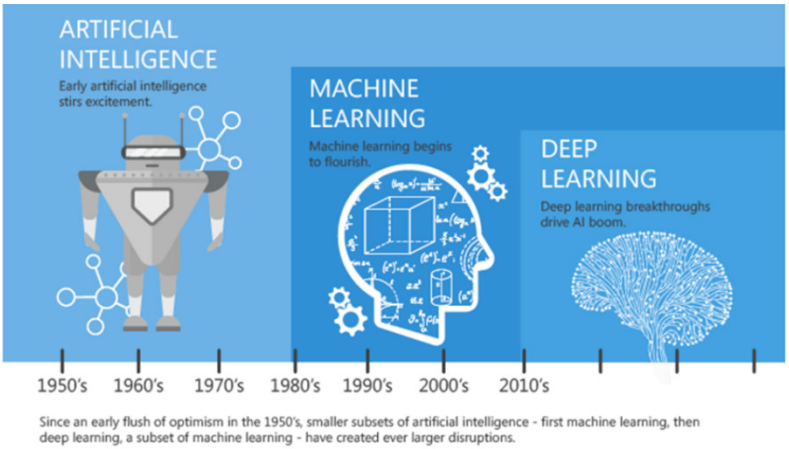


圖 9-75 機器學習發展 (圖片來源 : Kapil Tandon (2016) AI & Machine Learning: The evolution, differences and connections)

機器學習的概念

其實我們不用把機器學習想的太複雜，簡單透過一句話表達就是從過往的資料和經驗中讓電腦學習並找到其中運行的規則。這些過往的資料我們稱之為「訓練資料 (training data)」，訓練資料中會包含兩部分「特徵 (feature)」和「目標 (label)」。如圖 (圖 9-76) 我們想要預測瓶中是哪一種酒，我們可以使用酒的顏色與酒精濃度當作特徵，建立出模型後再進行預測，而酒的種類就是所謂的目標。

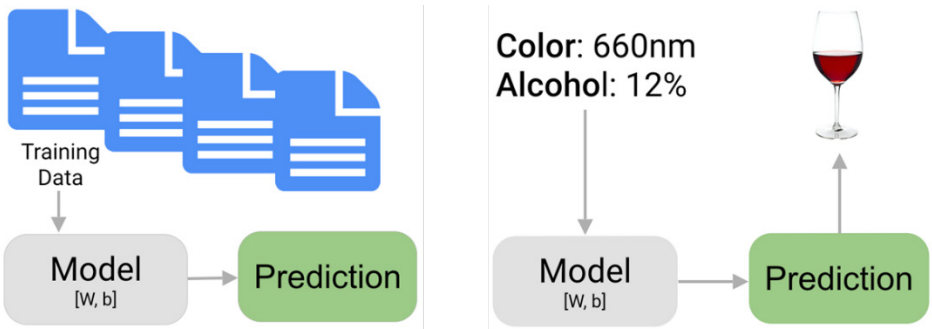


圖 9-76 機器學習的概念

機器學習的分類

機器學習可以分為監督式學習與非監督式學習。監督式學習包含所謂的分類模型與迴歸模型，這類的模型在訓練時能夠事先得知正確的答案為何；例如我們想要建立預測酒的品種模型，我們會事先知道每一種酒的顏色與酒精濃度，也就是在有已知答案的情況下讓電腦學習其中的規則為何。非監督式學習屬於在沒有正確的答案下所建立的模型，稱之為分群模型，例如我們現在有上百筆電影的資料，我們想將他們分類但卻又不知道該怎麼進行分類，因此透過分群演算法電腦會自動幫我們分成預先設定好的群體數量 (如 : 分成四類)，分完後再由人去看這些群體所代表的個別意義為何，可能其中一群都是動作片，其他群可能是恐怖片、動畫片等等。在後續的教學我們會先以監督式學習當中的迴歸演算法當作示範。

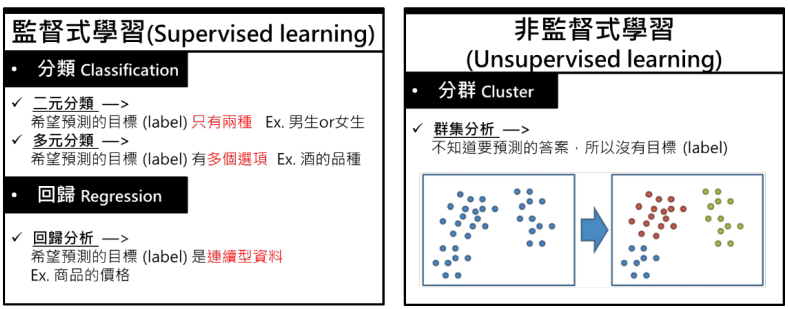


圖 9-77 監督式與非監督式學習

機器學習架構圖

每一種機器學習方法都分別對應有不同的演算模型，像是分類模型有 SVM、NB、KNN 模型等，迴歸模型有 LR、SVR、GPR 等，分群模型則有 K-Means、Hierarchical 等。每一種模型都有其各自的優缺點以及適合使用的資料，至於每一種資料適合哪一種演算法除了需事先了解背後的演算邏輯外，還有各自的建模經驗所帶來了經驗法則，因此各位讀者若對於這塊有興趣的話，不妨多參考網路上所分享的分析經驗，以及試著多使用看看不同的資料實際建立模型會更有所收穫呦。

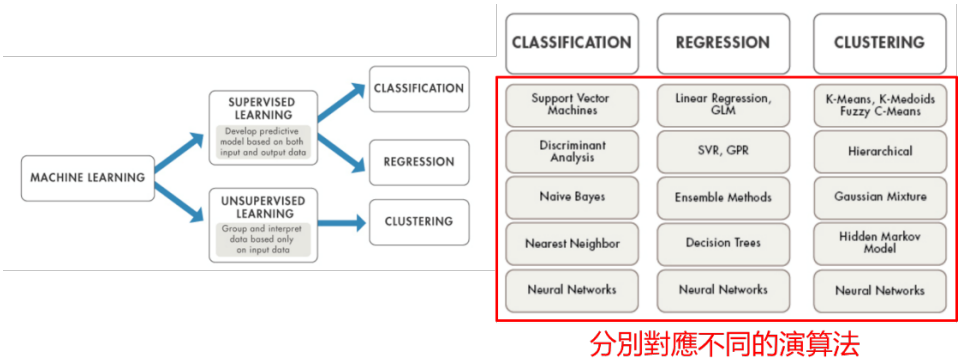


圖 9-78 不同的演算法

機器學習的流程

前面提及了我們在建立模型時需要擁有訓練資料 (training data)，而此資料裡面又會分為特徵及目標。在建立模型時通常只會使用資料的 70% 進行訓練，剩下的 30% 則會當作測試資料 (testing data)，也就是用來驗證模型的好壞。

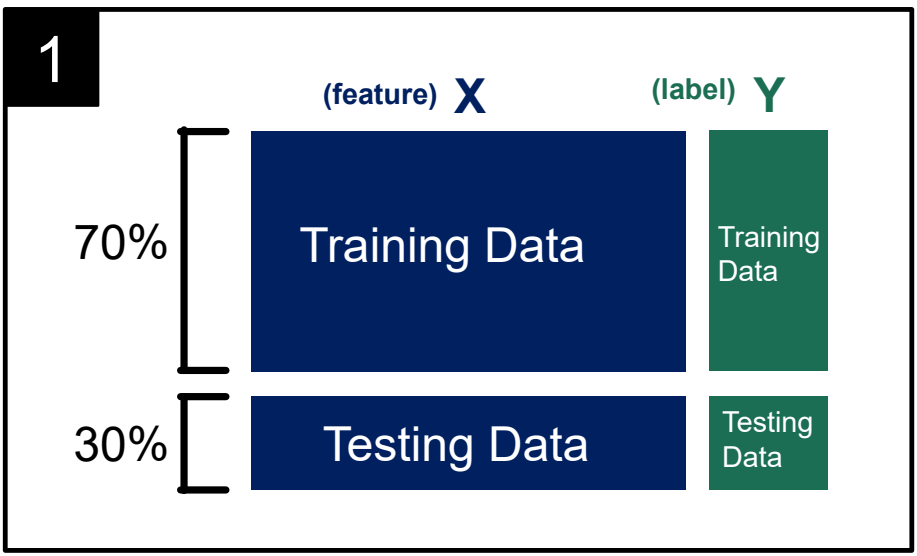


圖 9-79 70% 訓練，30% 測試

將 70% 的訓練資料用於建立機器學習模型 :

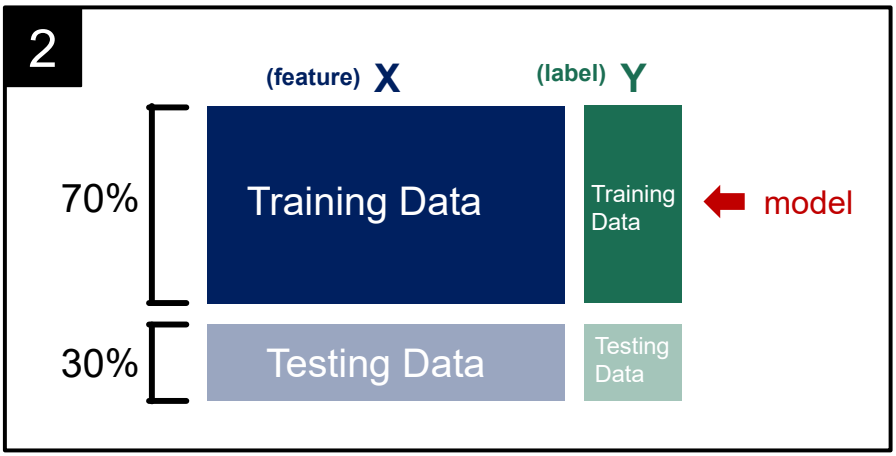


圖 9-80 70% 的訓練資料

當我們訓練好模型，接著就會拿 30% 的測試資料預測看看會得到什麼樣的結果 :

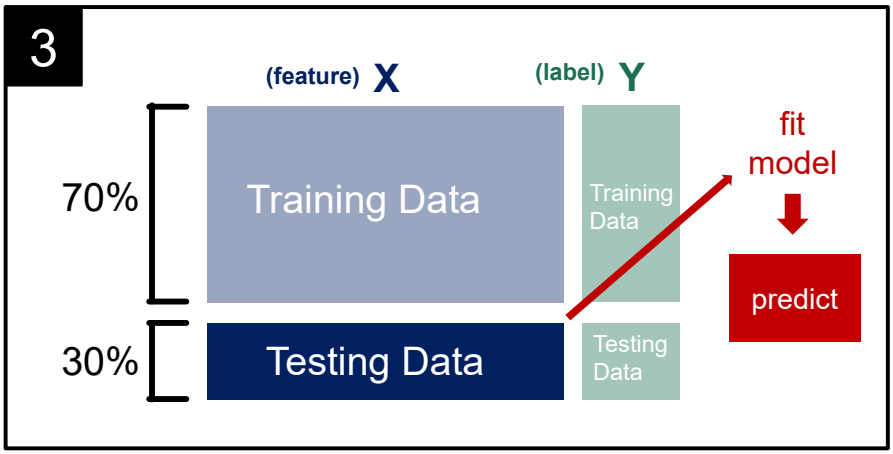


圖 9-81 30% 的測試資料

最後預測完成就會將具有真實答案的測試資料與模型所預測出的結果進行比對，來看看模型的效果是否具有準確預測能力。

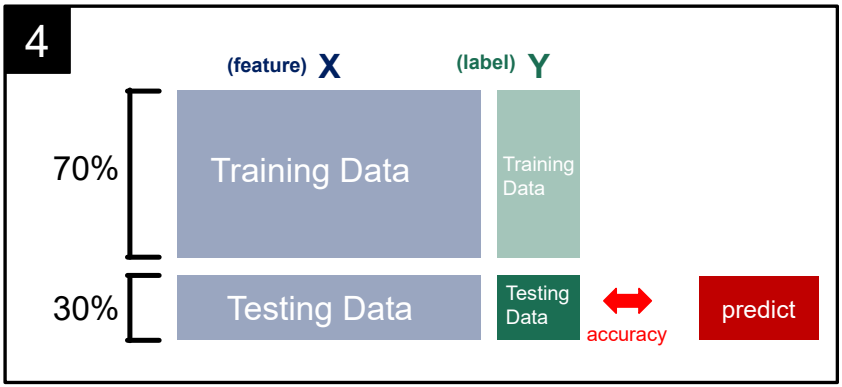


圖 9-82 預測的結果進行比對

以上就是建立機器學習的大致流程，現在已經學會了那我們就開始使用 Python 來進行實踐吧！不過在建立模型前還是要先經過資料前處理的步驟才行。

資料前處理

匯入套件與資料

首先匯入 pandas 套件以及 9-3 節經過整理的粉專資料。

```
01 import pandas as pd
02 df = pd.read_excel('fanpage_clean.xlsx')
```

查看資料：

```
03 df.head()
```

	id	message
0	124616330906800_1560501197318299	阿娘威!披羊皮的狼?竟大口嚼小雞\n#要打統編:小編真的是快嚇死了... \n\n影片來源..
1	124616330906800_1560454417322977	被黑了!李毓芬演唱「大落拍」網友卻意外發現「亮點」\n#條紋編:這一段應該是昨天的亮點表演..
2	124616330906800_1559870414048044	誰說牠呆?心機月月調虎離山網友讚影帝\n#124616330906800_1559870414048044樂無編:最萌心機鬼~(*^Y)~y\n\n影片..

圖 9-83 查看資料 head()

將資料中的特徵 X 與目標 Y 分開

我們將會使用分享數、留言數、小編、發文時間以及發文星期來試著預測看看文章的按讚數量多寡，因此前面的資料就會是特徵 X，而按讚數量則是目標 Y。

```
04 X = df[['shares','comments','curator','hour','weekday']]
05 y = df['likes_count']
```

檢查資料

接著檢查特徵資料當中有無遺失值，發現小編資料並非每筆文章當中都有。

```
06 X.info()
```

```
<class 'pandas.core. frame. DataFrame'>
Range Index: 9975 entries, 0 to 9974
Data columns (total 5 columns):
shares 9975 non-null int64
comments 9975 non-null int64
curator 7232 non-null object
hour 9975 non-null int64
weekday 9975 non-null object
dtypes: int64(3), object(2)
memory usage: 389.7+ KB
```

圖 9-84 檢查資料

處理遺失值

因此將具有空值的小編填入字串未知：

```
07 X['curator'] = X['curator'].fillna('未知')
```

再次檢查，發現已成功填入空值：

```
08 X.info()
```

```
<class 'pandas.core. frame. DataFrame'>
Range Index: 9975 entries, 0 to 9974
Data columns (total 5 columns):
shares 9975 non-null int64
comments 9975 non-null int64
curator 9975 non-null object
hour 9975 non-null int64
weekday 9975 non-null object
dtypes: int64(3), object(2)
memory usage: 389.7+ KB
```

圖 9-85 處理遺失值

資料標準化

接著將連續型的資料進行標準化，用意為將資料的範圍壓縮在一定的範圍內，可以減少資料上運算的複雜度。我們的特徵中屬於連續型資料的有分享數量以及留言數量兩個欄位。

```
09 from sklearn import preprocessing
10 standard = preprocessing.StandardScaler()
11 X[['shares','comments']] =
12     standard.fit_transform(X[['shares','comments']])
```

其餘的欄位是類別型的資料，我們將其轉換為 Dummy 變數，Dummy 變數也就是我們將不同的小編名稱轉換為電腦看得懂的 0,1 數字，否則電腦將無法讀懂 B 編、淡水阿編這種特殊字眼。

```
13 X_1 = pd.get_dummies(X)
```

轉換後的結果

```
14 X_1.head()
```

	shares	comments	hour	curator_BG編	Curator_B編	curator_M編	curator_七條編	curator_人間四月編	curator_什麼編	curator_傻編	...	curator_高光編
0	0.241213	0.001331	11	0	0	0	0	0	0	0	...	0
1	-0.212199	-0.206505	11	0	0	0	0	0	0	0	...	0
2	0.677548	0.564335	10	0	0	0	0	0	0	0	...	0
3	-0.224154	-0.215934	10	0	0	0	0	0	0	0	...	0
4	-0.212199	-0.208076	10	0	0	0	0	0	0	0	...	0

圖 9-86 轉換後的結果

Training & Testing data

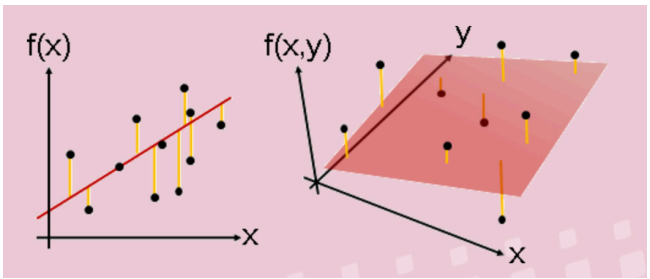
將資料分成 training data 以及 testing data，不一定要 70% 比 30% 的比例切，也有些人分成 80% 與 20% 等。透過 `test_size` 參數可以協助設定切分的比例。

```
15 from sklearn.model_selection import train_test_split
16 X_train, X_test, y_train, y_test =
17     train_test_split(X_1, y, test_size = 0.3, random_state = 2)
```

預測模型：Linear Regression

線性迴歸介紹

我們來介紹第一個要使用到的機器學習模型 — 線性迴歸。相信各位在國高中時應該都有學過迴歸函數，也就是找一個函數，盡量能夠使其符合手邊的一堆數據。當資料是屬於二維度的，此迴歸函式就是一直線 (y = ax+b)，當資料是屬於三維度的，則迴歸函數會變成是平面的空間，資料維度越大，迴歸的函式就會越複雜。



- 二維：直線
- 三維：平面
- 多維：超平面

圖 9-87 線性迴歸

簡單線性迴歸公式如下，也就是我們熟悉的 y = ax + b，希望每一個資料點距離此條線的誤差加總能夠最小化。

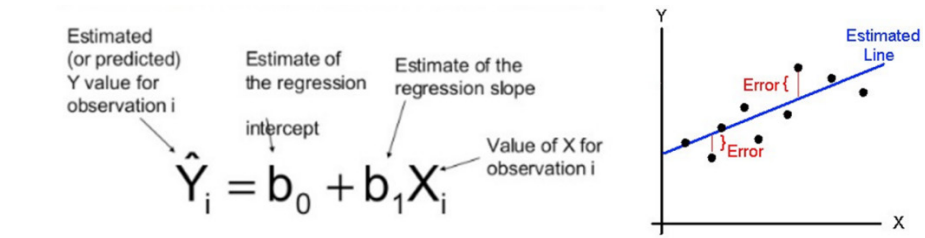


圖 9-88 線性迴歸公式

複迴歸：當我們的資料特徵值不只一項時，就會使用到複迴歸，公式中的每一個 X 都代表一個資料特徵，例如酒精顏色、酒精濃度等，每一個 X 都會搭配一個權重 β(Beta) 來代表此特徵的重要性高低。而我們所要建立的也就是複

迴歸公式，接下來就開始進入 Python 程式吧。

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \varepsilon_i$$

圖 9-89 複迴歸

匯入迴歸模型套件

從 sklearn 套件中匯入線性迴歸模型，並且將此模型存入 lm 變數當中。

```
18 from sklearn.linear_model import LinearRegression
19 lm = LinearRegression()
```

接著進行模型訓練，使用到的是前面所切分的 70% 訓練資料

```
20 lm.fit(X_train.values, y_train.values)
```

執行結果：

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1
, normalize=False)
```

有沒有發現建立模型居然只要一行程式就建立完畢了！真的相當簡單。

查看模型

迴歸模型的截距項，也就是公式中的 β_0。

```
21 print(lm.intercept_)
```

執行結果：

```
3620.626025474225
```

每一個特徵變相的迴歸係數，也就是公式中的 β_1~ β_n。

```
22 print(lm.coef_)
```

執行結果：

```
[ 3.00462596e+03   4.72062851e+03   2.01721953e+01  -2.28276444e+03
  2.07920813e+03   1.48858548e+02  -1.46823617e+03  -2.81825555e+03
  3.51436775e+03  -1.33455026e+03   4.27693492e+02   3.67418636e+02
 -1.60739471e+03  -3.07750639e+02  -1.51851326e+03   1.48574584e+03]
```

進行預測

建立好模型後，我們接著來預測看看 30% 的測試資料。

```
23 y_predict = lm.predict(X_test)
```

將預測按讚數與真實按讚數列出前五筆作比較，可以發現沒有很準確。

```
24 pd.DataFrame(list(zip(y_test.values, y_predict)),
25                columns=['Measured','Predicted']).head()
```

	Measured	Predicted
0	6938	221.857708
1	860	2083.246286
2	128	273.179627
3	4851	5543.968663
4	491	12672.784547

圖 9-90 前五筆資料

將按讚數量的分佈透過圖形呈現，可以發現大多都聚集在20,000個讚以下。

```
26 import matplotlib.pyplot as plt
27 plt.scatter(y_test.values,y_predict,s=2)
28 plt.plot([y_test.values.min(), y_test.values.max()],
29          [y_test.values.min(), y_test.values.max()], 'k--', lw=2)
30 plt.ylabel('Predicted')
31 plt.xlabel('Measured')
32 plt.show()
```

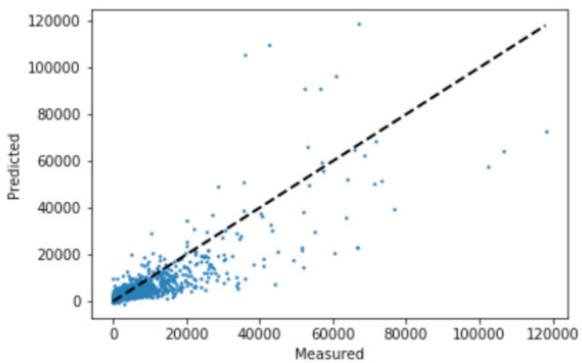


圖 9-91 按讚數量

檢驗迴歸模型

我們的資料總共有將近一萬筆，不太可能一筆一筆的確認其預測的是否準確，因此可以透過一些統計指標來協助我們評估模型的準確性。

均方誤差 MSE，也就是所有的按讚數量與真實按讚數量之間的誤差總和 :

```
33 from sklearn.metrics import mean_squared_error
34 mse = mean_squared_error(y_test.values, y_predict)
35 print('MSE : ',mse)
```

執行結果：

```
MSE : 23446608.538704727
```

均方根誤差 RMSE，將 MSE 開根號 :

```
36 from math import sqrt
37 rms = sqrt(mean_squared_error(y_test.values, y_predict))
38 print('RMSE : ',rms)
```

執行結果：

```
RMSE : 4842.169817210537
```

判定係數 R-square 與調整後的判定係數 Adjusted R-square：判定係數用於評估所使用的特徵對於目標 Y 的解釋能力有多少比例，當特徵不只一項時查看 Adjusted R-square 較為準確。因此從以下結果中可以得知分享數、留言數、小編、發文時間以及發文星期在線性迴歸模型當中只能解釋目標 Y 的 69% 而已。

```
39 R_2 = lm.score(X_train, y_train)
40 print('R-squared  : ',R_2)
```

執行結果：

```
R-squared: 0.6991904295741282
```

```
41 adj_R_2 = R_2 - (1 - R_2) *
42           (X_train.shape[1] / (X_train.shape[0] - X_train.shape[1] - 1))
43 print('Adjusted R-squared : ',adj_R_2)
```

執行結果：

```
Adjusted R-squared:  0.6949518287125203
```

現在我們已經成功建立出第一個機器學習模型了，不過我們並不容易透過這些指標來得知究竟模型好不好，因為沒有其他比較的基準，因此現在我們就來使用另一項迴歸模型，隨機森林樹 (Random Forest Regression) 來比較看看模型的好壞吧！

Random Forest Regression

隨機森林樹迴歸介紹

此模型是基於決策樹 (decision tree) 的樹狀模型，當我們資料中具有較多的類別特徵，且這些類別特徵有可能對於目標 Y 會有顯著影響時非常適合使用。從第三節的 EDA 分析可以發現 PO 文的小編以及發文的時間確實會影響到按讚的數量，因此可以設想採用此模型效果有機會更好。

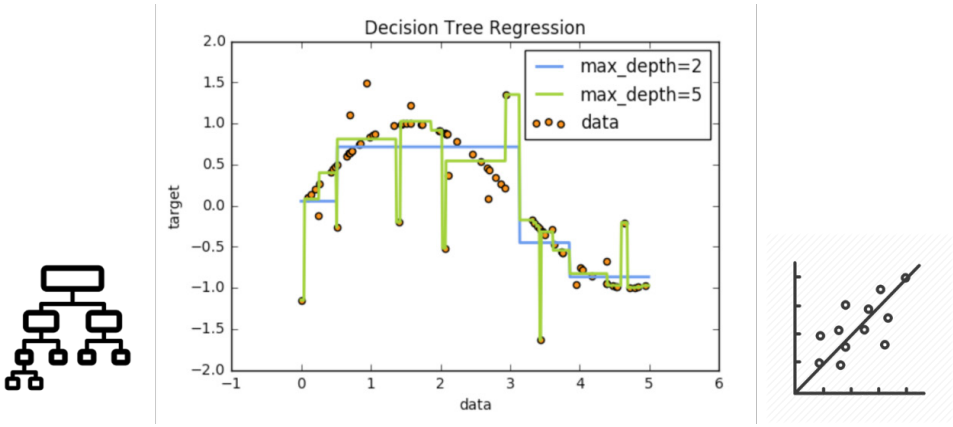


圖 9-92 隨機森林樹迴歸

匯入樹迴歸模型套件

從 sklearn 套件中匯入隨機森林樹迴歸，並且將此模型存入 rfr 變數當中。

```
44 from sklearn.ensemble import RandomForestRegressor
45 rfr = RandomForestRegressor()
```

接著進行模型訓練，使用到的是前面所切分的 70% 訓練資料。

```
46  rfr.fit(X_train.values, y_train.values)
```

執行結果：

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto',
                      max_leaf_nodes=None, max_samples=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=100,
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)
```


進行預測

建立好模型後，我們接著來預測看看 30% 的測試資料：

```
47  y_predict_rfr = rfr.predict(X_test)
```

將預測按讚數與真實按讚數列出前五筆作比較，可以發現好像兩者之間有較為接近的感覺囉！

```
48  pd.DataFrame(list(zip(y_test.values, y_predict_rfr)),
49               columns=['Measured','Predicted']).head()
```

	Measured	Predicted
0	6938	3221.5
1	860	755.8
2	128	292.6
3	4851	5626.1
4	491	730.9

圖 9-93 前五筆資料

檢驗迴歸模型

接著我們同樣透過一些統計的指標來評估此模型的準確程度：

均方誤差 MSE

```
50 from sklearn.metrics import mean_squared_error
51 mse = mean_squared_error(y_test.values, y_predict_rfr)
52 print('MSE : ',mse)
```

執行結果：

```
MSE :17156586.352579575
```

均方根誤差 RMSE

```
53 from math import sqrt
54 rms = sqrt(mean_squared_error(y_test.values, y_predict_rfr))
55 print('RMSE : ',rms)
```

執行結果：

```
MSE :4142.050983821852
```

判定係數 R-square

```
54 R_2 = rfr.score(X_train, y_train)
55 print('R-squared  : ',R_2)
```

執行結果：

```
R-squared : 0.9591639892499428
```

調整後的判定係數 Adjusted R-square

```
56 adj_R_2 = R_2 - (1 - R_2) *
57           (X_train.shape[1] / (X_train.shape[0]- X_train.shape[1] - 1))
58 print('Adjusted R-squared : ',adj_R_2)
```

執行結果：

```
Adjusted R-squared: 0.9585885835203154
```

我們可以從結果中發現不論是 MSE、RMSE 或是判定係數都比線性迴歸模型來的優秀，甚至判定係數高達 94%，這也就是前面所提及的，當我們對於所要分析的資料具有一定的了解，以及懂得不同演算法背後的運算邏輯時，在選用模型上確實可以收到不錯的效果，當然這並非代表隨機森林樹就是最適合此份資料的演算法。相信加入更多的特徵 X，以及進行更多的資料前處理後會有更好的預測結果，我們將這部分稱之為資料特徵工程 (feature engineering)。因

此，想要擁有好的預測結果，除了選對模型之外，最重要的就是挑選特徵變項。特徵變項並不是越多越好，而是要能找到最能解釋目標 Y 的幾項特徵才行，這部分就要透過第三節的 EDA 分析來看特徵與目標之間的關係是否有顯著影響，此外就是依靠我們本身的領域知識 (domain knowledge) 了，當我們對資料越加熟悉，越容易得知什麼樣的特徵對於目標會有影響。

小結

我們在本章介紹了機器學習的一些基本概念、以及如何透過 Python 程式成功實踐。相信各位讀者讀完此章後會發現建立機器學習的程式相當簡單，但中間所需具備的領域知識卻是相對高深且複雜的，沒有一定的實務分析經驗並不容易建立出一個好的機器學習模型。此外我們也只介紹了機器學習當中的迴歸模型，尚未介紹分類模型與分群模型，雖然使用 Python 程式同樣相當的簡單與簡短即可成功，不過這兩種模型又各有指標用於評估模型的好壞，因此要成為一名機器學習工程師，其實是需要許多時間磨練經驗的！不過各位讀者也不用太擔心這條路途會非常遙遠，網路上有許多的教材甚至影片，坊間也有不少的書籍。也不用擔心沒有資料可以練習分析。在這邊相當推薦 Kaggle 這個網站，他是一個資料分析競賽的網站，裡面有各式各樣的數據集資料，像是鐵達尼號、電影、旅館、交通等等，即使比賽已經結束了仍然能下載其資料來進行分析，此外每個比賽都會有許多高手分享他們的分析方法以及建模過程，因此相當推薦有興趣的讀者參考及學習。

總結

我們的教學就告一段落了，所有內容都是相當的實用的，從最初的資料爬取到資料分析、建立模型等等，將中間的所有過程整理起來會是相當完整的一份分析報告書呢！我們希望藉由此種方式可以讓各位讀者了解什麼叫做資料分析，在資料分析的過程當中會是以什麼樣的方式在進行。即使讀者對於資料科學領域並未有太大的興趣，但至少也習得了一套完整的資料分析流程。現階段各行各業都在想盡辦法跟上這一波大數據的浪潮，相信各位在未來不論是與數

據分析師交談亦或是與公司內部的 IT 人員討論時，可以更有自信地表達自己的看法，希望本章可以讓各位讀者具有知識上或是技術上的收穫。

參考資料

1. API 介紹：https://cola.workxplay.net/what-is-an-api/
2. 圖形 API 測試工具的官方文件說明：
   https://developers.facebook.com/docs/graph-api/using-graph-api/#fieldexpansion
3. 斷字斷詞字典下載位置：https://github.com/fxsjy/jieba/raw/master/extra_dict/dict.txt.big
4. 停止詞字典下載位置：https://github.com/chdd/weibo/blob/master/stopwords/ 中文停用词库 .txt
5. 圖形 API 參考資料：https://developers.facebook.com/docs/graph-api/reference
6. word2vec 簡介：
   https://medium.com/pyladies-taiwan/ 自然語言處理入門 -word2vec 小實作 -f8832d9677c8
7. word2vec 參數
   (1) 官方文件：https://radimrehurek.com/gensim/models/word2vec.html
   (2) 中文參考：https://www.kaggle.com/jerrykuo7727/word2vec

Chapter 10

Arduino 開發板在資料擷取之應用

/ 張幼珍

號稱萬物聯網的現代，生活周遭遍佈各式各樣的感測器，日以繼夜地搜集環境中的資訊。本章以溫濕度感測器資料分析為應用，介紹 Arduino UNO 開發板和 DHT11 溫濕度感測器硬體及連接方式，示範如何利用 pySerial 模組讀取溫濕度感測器數據並以 matplotlib 繪製折線圖。學習完本章後，讀者將對開發板、感測器與程式之軟體整合有初步認識。

前言

近年來 Python 在程式開發領域大受各領域 (科學、工程、藝術等) 人士們的歡迎，背後有很多的原因在支持，其中一種是嵌入式系統的應用。傳統的程式設計需要有電腦來執行程式結果，隨著硬體技術的發展，現在我們可以使用樹莓派 (Raspberry Pi) 或是 Arduino 等嵌入式系統的環境，搭配合適的感測器，設計各種貼近生活的智慧化資訊系統。

本章的主題將以 Arduino 開發板為操作平台，使用 Python 程式來截取 Arduino 與 DHT11 所取得的數據溫濕度。Arduino 開發板上搭配微處理器與控制器，雖然其規格仍遠不及樹莓派這樣的小型微電腦，基本上仍可視為一個近乎完整的微電腦系統。我們可以在 Arduino 平台上根據自己的需求搭配不同的感測器，然後撰寫程式控制與讀取這些感測器的資訊，並利用平台上的微處理器進行資料處理與分析。從前面的章節，大家應該就猜得出來，我們在 Arduino 上撰寫 Python 程式的主要工作，就是負責取得資料後的資料處理、統計分析、科學繪圖等，如果能再結合控制元件如繼電器 (relay)、伺服馬達 (servo)、步進馬達等，就可以做智慧化、自動化系統設計與開發，譬如，氣溫高到某個程度，自動啟動噴霧系統來降溫；濕度監控到某個濕度後，就自動切斷噴霧器的電源。因此前面各章節所學習的程式技巧，在本章大多是可以延續使用的。

本章是建立在 Arduino 系統上，將會同時接觸開發板平台的軟硬體與 Python 程式撰寫的軟體環境，這是本章與其他章節最大的差異處。因此我們將從下一節的軟硬體需求與環境設置開始，逐步介紹如何上傳 Arduino 開發板的程式以及使用 Python 程式來進行資料處理的工作。特別要注意的是，Arduino 與各項感應器的硬體生產廠商不只一家，各家的接腳標示與接腳數目有時也不同，品質與價格也有很大差異，標示或接腳數目不同時，都需要從購買時包裝上或元件上找到產品編號，然後用產品編號透過 Google 關鍵字搜尋，找尋一些外觀上相同的感測器接線圖 (wiring diagram) 來進行比較，找出正確接線方式。

如果讀者手邊的硬體版本與書中描述的有所差異，書中的各種說明仍是值得參考，但是在實際運用時，碰到書中沒有提及的問題時，請發揮創客精神上網利用關鍵字搜尋來找尋網路相關資料來找到正確的用法。

對初學者而言，各種感測器、控制元件如繼電器等在網際網路上都有現成的 Arduino 程式範例可以免費下載，初學者只要學會如何閱讀與修改、合併 Arduino 程式碼，然後再學習如何寫出可以跟 Arduino 往來溝通的 Python 程式，取得資料後就可以分析繪圖或控制系統上的其他元件。

10-1 軟硬體基本需求

軟體安裝步驟

本章以 Windows 10 系統和 Python 3.6 版本來進行 Arduino UNO/DHT11 溫濕度感測的開發專案的示範，安裝的 pySerial 模組也是針對 Python 3.6 版本。因此，建議讀者也安裝 Python 3.6 來學習，以免出現版本不同造成的諸多不相容問題。

表 10-1: 需求模組介紹 :

模組名稱	功能說明	安裝指令
pySerial	連接電腦序列埠 (序列埠) 的 Python 套件	pip install pySerial

要安裝 pySerial，請先啟動 Windows 的命令提示字元後，如圖 10-1 輸入命令「pip install pyserial」，pip 會自動上網去搜尋、下載與安裝模組套件 (package)。

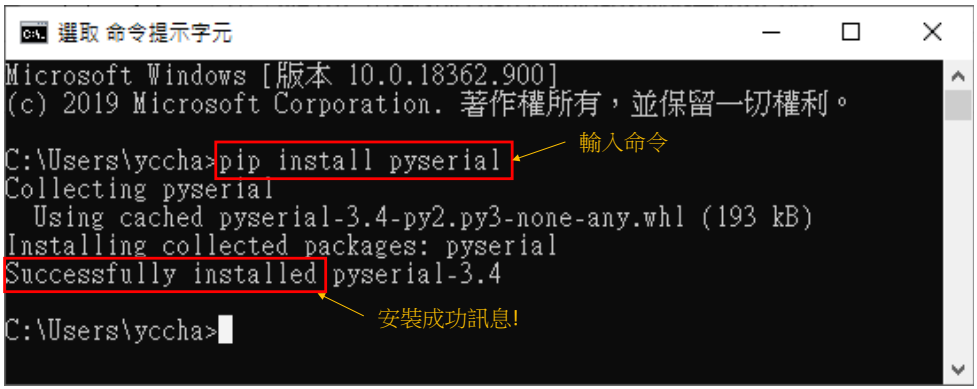


圖 10-1 輸入 pip install pySerial 以安裝 pySerial 套件

硬體組裝步驟

Arduino 開發板有多種，Arduino UNO 只是其中一個基礎的開發板，不同的開發板有不同的規格。我們可以利用 Arduino 官網提供的免費軟體來幫助電腦辨識是哪一塊 Arduino 開發板並告訴電腦使用哪一個序列埠 (COM port) 來連接，幫助初學者克服序列埠 (COM port) 設定問題。

首先請上網搜尋 Arduino Software 官網，進入 Arduino 官網 (https://www.Arduino.cc/)，由上方選單選取 SOFTWARE 下的 DOWNLOADS，進入 Download the Arduino IDE 頁面後，請選取右邊第一個安裝檔選項 :Windows Installer, for Windows 7 and up，即會進入 Contribute to the Arduino Software 頁面。在此頁面右下方按下 just download 這個連結即可免費下載最新的 Arduino IDE 安裝執行檔，譬如，Arduino-1.8.13-windows.exe。執行此檔案直到出現安裝成功訊息。下載 IDE 並安裝，開始安裝 Arduino UNO。

Arduino UNO board: Arduino UNO 是最基礎的 Arduino 開發板，我們可以透過網路商店購買，也可以到電子材料行購買平價 Arduino UNO 開發板，外觀如圖 10-2。以 Arduino UNO 板為例，一般電子材料行店面售價 (台幣約 300 多)，是網路賣家價格 (約台幣 180 元) 的兩倍左右。感測器以 DHT11 為例，店家價格也大約是網路賣場價格的 2 倍左右，如果購買的數量大，鄰近的網路賣場是較經濟的選擇，7 日的退換貨時間也比較充裕，網路賣家也可參考已賣出數量，數量高買家多的，暗示買家比較過價格後的選擇，此外也可能暗示貨品品質較可信賴。

USB 2.0 Cable Type A/B: USB 是 Universal Serial Bus 的英文縮寫，中文譯名為通用序列匯流排 (圖 10-3)。USB 已廣泛使用於連接電腦與外部裝置，我們將透過 USB 將 Arduino UNO 連接到電腦的 USB 插槽中。

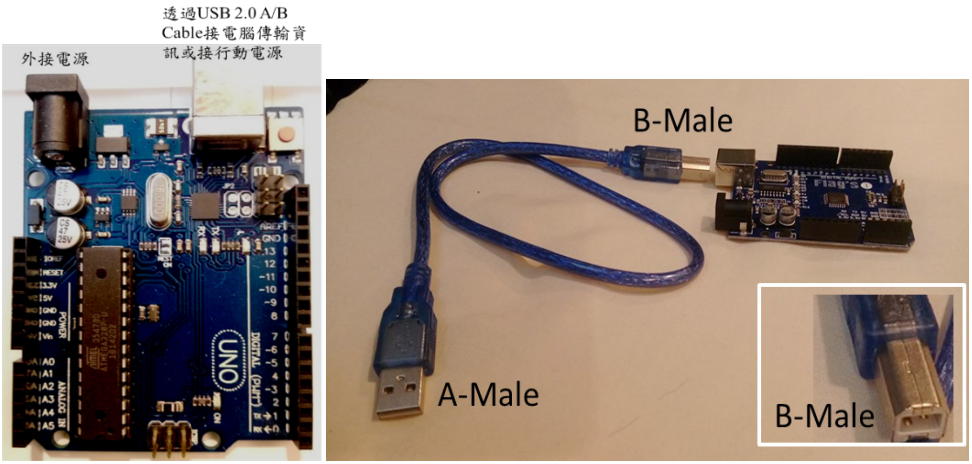


圖 10-2 (a)Arduino UNO board (b)USB 2.0 Cable Type A/B

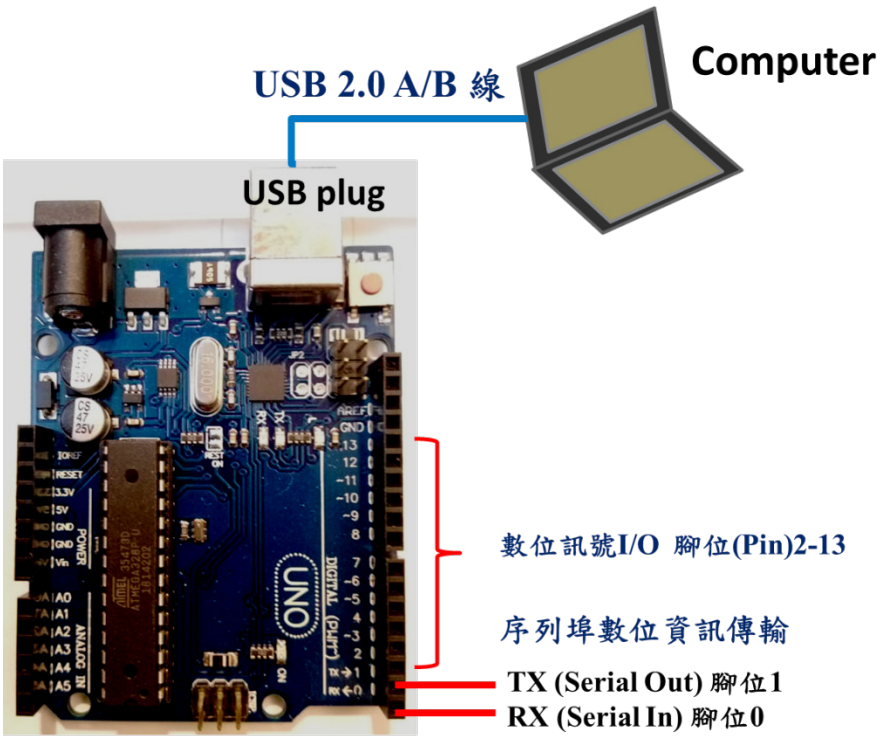
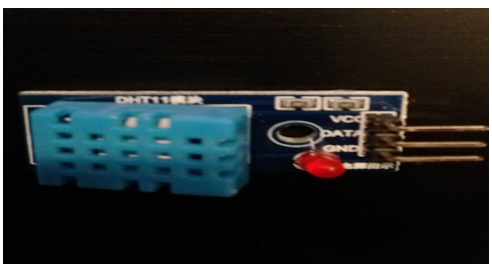


圖 10-3 序列埠訊號傳輸 (Arduino to PC Serial Communication)

DHT11 溫濕度感測器：DHT11(圖 10-4) 是以電阻感應濕度的高低，以 NTC 感應溫度，再用數位方式輸出。我們使用公 - 母 (male-female) 杜邦線 (Dupont wires or Jumper wires) 將 DHT11 與 Arduino UNO 連接如圖 10-5。電子材料行或網路上買的 DHT11 接腳標示各有不同方式，有的標示如圖中的 VCC, Data, GND, 有的標示 +,OUT,-。+ 即 VCC，要用 female-male 杜邦線去銜接 Arduino UNO 的 5V 腳位，- 即 GND 要接到 Arduino UNO 的 GND 接地腳位，OUT 即 Data 要接到 Arduino UNO 的 Digital pin 腳如 D2-D13 任一個。特別注意的是實際腳位是多少，Arduino 程式中也要改成這個腳位。

經驗上，由於 Arduino 零件價格低廉，買來的開發板、USB 線、杜邦線、感測器等，都有可能買到出廠時即是瑕疵品，因此，如果接線後測試不成功，除重複檢查本書提醒的諸多細節，建議手邊準備一組確定可以正常運作的開發板與感測器等，用這套硬體來逐一測試懷疑是瑕疵品的開發板、USB 線、杜邦線、感測器等。購買後盡量在收到貨品的幾天內完功能成測試地確認，如確為瑕疵品，還可於七天內跟店家換貨。瑕疵品可能是所有接線、程式都正確，但量測到的溫溼度測值都是零，也可能是溫度或濕度遠高於實際值，譬如實際乾燥的環境裡濕度測值總是維持在 90% 以上。



VCC: 3-5VDC 電源輸入 DATA: 數位訊號傳輸
GND: 接地
RH Range: 20-90%
RH Accuracy: ±5% RH.
Temperature Range: 0-50 °C

圖 10-4 DHT11 溫溼度感測器

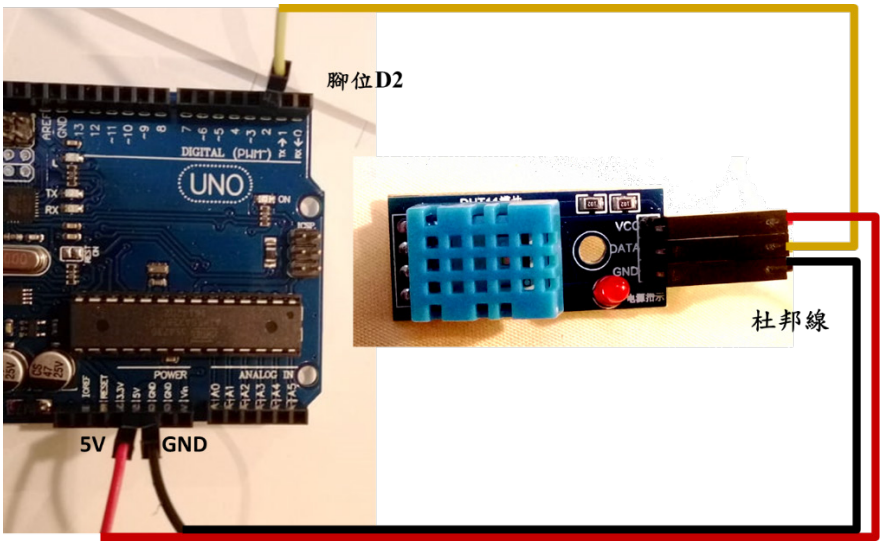


圖 10-5 Arduino UNO 與 DHT11 溫濕度感測器接線圖

圖 10-6 是 Arduino UNO+ 溫度感測器 DTH11 的完成圖，我們開始將安裝到電腦上，請依照下面程序進行。

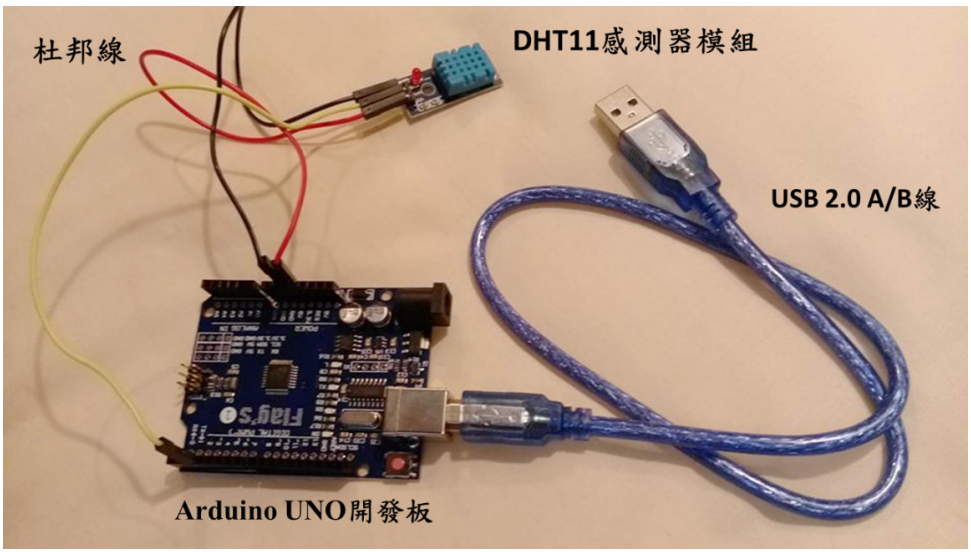


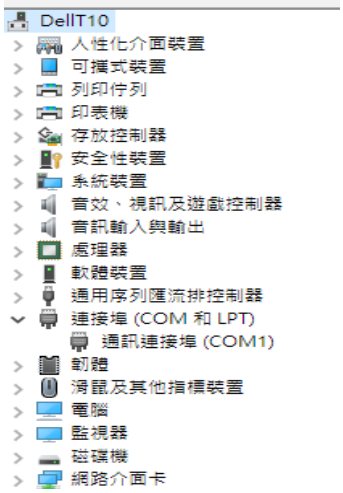
圖 10-6 Arduino UNO+ 溫度感測器 DTH11 接線完成圖。

1. 將滑鼠移動到 Windows 視窗左下角 Start(開始) 鍵上，按右鍵出現的選單中 (圖 10-7 (a))。

2. 將滑鼠移到裝置管理員後出現的視窗按滑鼠左鍵，出現一個使用者帳戶控制視窗，詢問您是否要允許此應用程式變更您的裝置，選擇「是」。

3. 點選連接埠 (COM 和 LPT)，出現如圖 10-7(b) 畫面。如果連接埠 (COM 和 LPT) 中沒有出現 Arduino 佔用的序列埠，而是出現其它裝置下有偵測到一個 USB2.0-Serial 的裝置，這表示電腦無法判讀這塊 Arduino 板，必須從網路上以關鍵字搜尋下載 CH340 驅動程式。首先找到可以下載這個程式的網頁，參考相關網頁說明，並於下載後解壓縮，然後執行檔案夾中 SETUP.EXE 檔。安裝後，其它裝置裡的 USB2.0-Serial 就會自動消失，而在連接埠 (COM 和 LPT) 中出現 USB-SERIAL CH340 的裝置就是剛插入的 Arduino 開發板，讀出它所佔用的序列埠，在 Arduino IDE 中工具下選取開發板使用的序列埠。

(a) Windows 視窗左下角 Start(開始) 鍵上按右鍵出現的選單

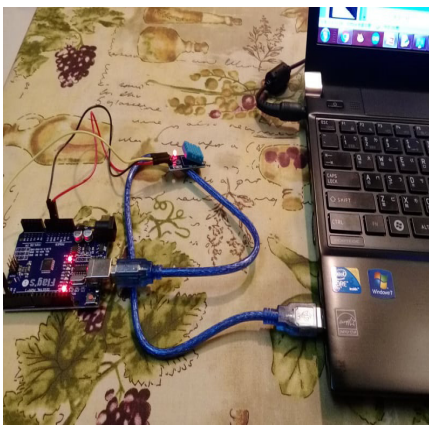


(b) Arduino UNO 尚未接電腦前，裝置管理員下的連接埠 (COM 和 LPT) 內容

圖 10-7 啟動裝置管理員

4. 將 Arduino UNO cable USB 插頭插入電腦 USB 插槽中，如下圖 10-8(a)，Arduino UNO 板和 DHT11 的 LED 燈都亮起 (坊間有些 DHT11 感測器沒有 LED 燈)。經驗上，網路上或電子材料行購買到的 DHT11 品質各異，且接腳接線錯誤也容易燒壞，初學者建議仔細檢查接線兩遍以上再去接電。如果突然聞到臭味一般是燒壞的徵兆之一，另一個徵兆是 DHT11 發燙，任一現象都有可能是 DHT11 燒壞或接線錯誤的問題，請立即將 Arduino 接到電腦上的 USB 插頭取出斷電，重新檢查。

5. 裝置管理員會自動掃描更新新增的裝置，掃描後連接埠出現 Arduino UNO(COM8)，如圖 10-8(b)。不同的電腦可能會接在不同序列埠，請記下 Arduino UNO 接在你的電腦上哪一個序列埠，稍後設定 Arduino IDE 軟體時需要設定讓軟體去這個序列埠找硬體。這章後面內容包括 Arduino IDE 的設定與 Python 程式碼，會持續使用 COM8 示範。讀者自行參考本章實作時，請務必將相關序列埠設定與程式碼的序列埠位置改成您的序列埠號碼，才能順利執行。



(a) 透過 USB 連接

(b) 裝置管理員會自動掃描更新新增的裝置，連接埠出現 Arduino UNO(COM8)

圖 10-8 將 Arduino UNO 透過 USB 與電腦連接

Arduino IDE 軟體安裝與設定

首次安裝 Arduino IDE 開啟後 (圖 10-9)，在上面選單選擇工具，滑鼠下移到開發板，選擇 Arduino/Genuino Uno 後，紅色框標示開發板 : 後面出現 "Arduino/Genuino Uno" 。前面我們已經將 Arduino UNO 安裝到 COM8(圖 10-8(b))，因此我們透過 Arduino IDE，應該就可以看到系統已經連接到開發板以及序列埠，如圖 10-9 與圖 10-10。經驗上，許多初學者每次將開發板 USB 接上電腦後，很容易忘記先去裝置管理員中檢查 Arduino 板接上哪個序列埠，更不記得去 Arduino IDE 的工具下選擇正確的開發板與序列埠，而直接上傳 Arduino 程式，由於沒有正確設定開發板與序列埠，Arduino IDE 執行程式上傳時，會因不知道開發板種類和序列埠位置而無法完成上傳出現錯誤訊息。因此，請務必養成每次將開發板 USB 接頭插入電腦時，立即到裝置管理員檢查開發板接到哪個序列埠，然後直接去 Arduino IDE 的工具下選擇正確的開發板名稱與序列埠。

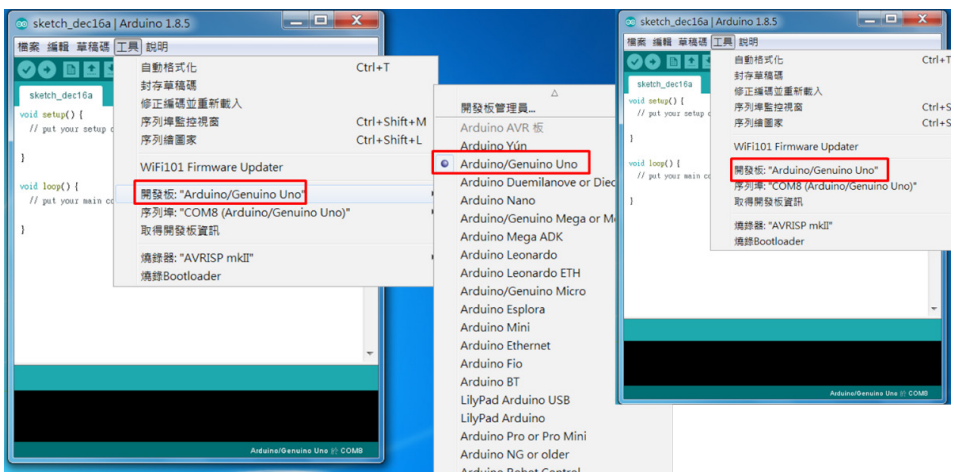


圖 10-9 Arduino IDE 選單 / 工具 / 選取開發板

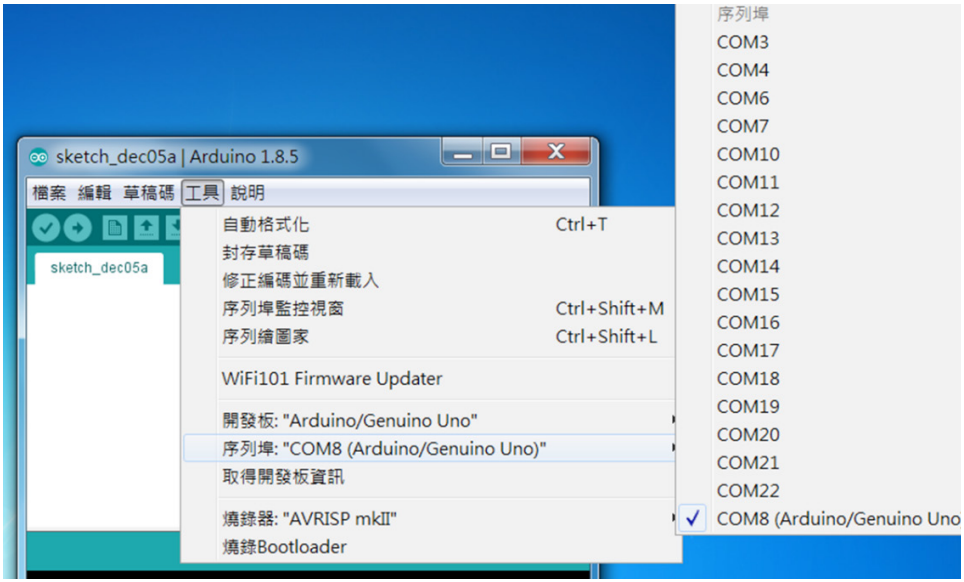


圖 10-10 Arduino IDE 選單 / 工具 / 選取序列埠

設定開發版和序列埠後，在 Arduino IDE 上方選單選擇草稿碼後出現的選單中選擇管理程式庫，之後出現右方的程式庫管理員，在上方鍵入 DHT11 後按 Enter 鍵，即會出現含有 DHT11 的程式庫，選擇 EduIntro，右邊按下安裝，安裝成功後，在 EduIntro 旁出現 INSTALLED 字樣，表示已成功安裝在您的電腦上，即可按下右下方關閉鍵。

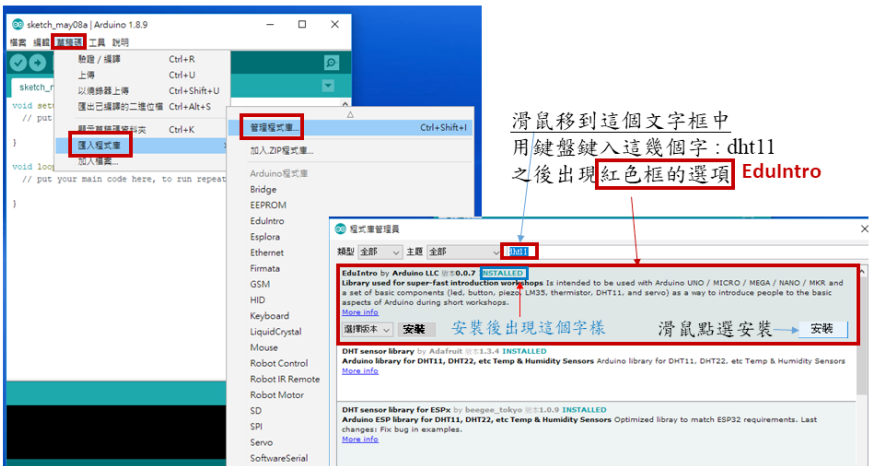


圖 10-11 Arduino IDE 中匯入 EduIntro 的程式庫

其次，選取上方選單的檔案，在檔案下出現的選單中選擇範例，在範例的選單中選擇 EduIntro->by-topic->DHT11，即可打開 DHT11 的範例程式。

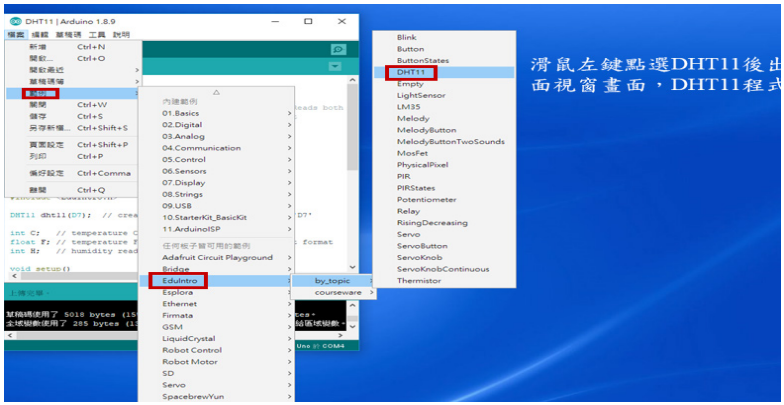


圖 10-12 Aduino IDE 下尋找 EduIntro 中 DHT11 的程式範例

下圖是開啟後的範例程式，在圖左程式中螢光色標示的那一行程式，// 後面說明這個範例程式是 DHT11 上的 Data 或 OUT 腳位是接到 Arduino UNO 板 Digital 這一側的 D7 位置。圖右是將 D7 改為 Arduino 板上實際接的腳位 D2。

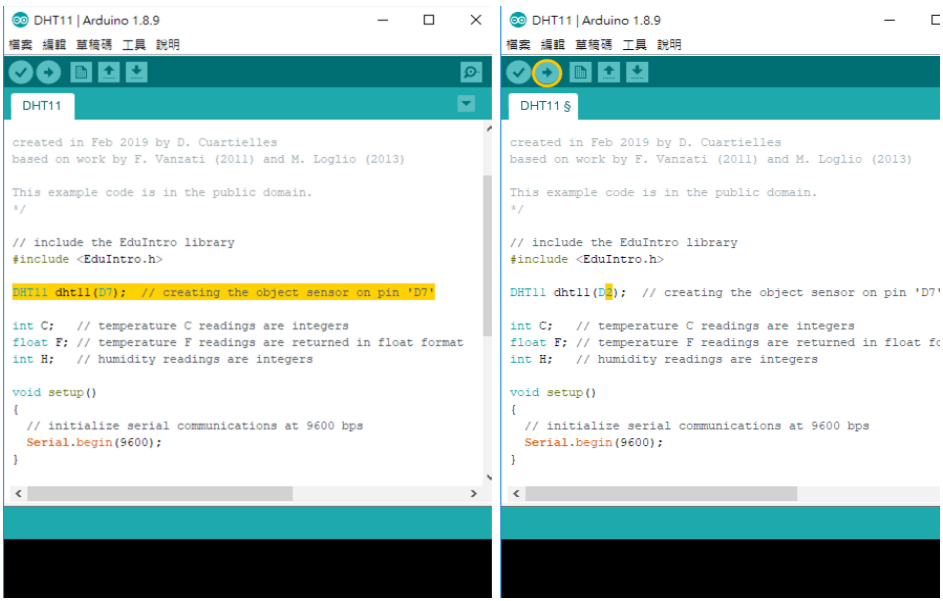


圖 10-13 開啟 EduIntro 下的 DHT11 程式範例並修改數位 Data 傳輸的腳位

改完後，即可參考圖 10-14 按下上面的圓形箭頭按鍵將程式上傳到 Arduino 開發板的記憶區。經驗上，許多初學者以為每次使用這套開發板進行溫溼度感測，都需要重新上傳程式，事實上，如果程式上傳開發板後測試成功，之後就可以用個人的行動充電電池來供電給開發板使用 (開發板的 USB 插頭插入行充即可)，如果順利執行沒有錯誤，就不需要修改程式再上傳；如果需要更改腳位或傳輸速率或其他程式碼，每次修改後都請按下圓形箭頭按鍵上傳程式到 Arduino 開發板記憶區。此外，也請注意圖右中 Serial.begin(9600) 的 9600 是溫濕度資料的傳輸速率，之後，不管是在 Arduino IDE 的監控視窗或是 Python 程式中，都必須使用相同的傳輸速率來傳輸溫溼度資訊，否則會出現無法辨識的亂碼，影響後續的資料分析與輸出。

由於本章示範使用的 DHT 程式範例只是網路上可以找到的眾多現成 DHT11 程式範例之一，不同 Arduino 程式範例，輸出給 Python 程式讀取的資料字串不同，當字串不同時，如果需要做字串分割 (split)，必須依 Arduino 實際產生的字串來決定，譬如，Arduino 產生的字串，溫溼度以逗號分隔，字串分割就用 `split(',')`，如果字串中的溫溼度是以 \t 分隔，就應以 `split('\t')` 來分割字串。如果可以進一步去讀 Arduino 程式碼並修改其溫溼度字串輸出使用的分隔符號，就可以選擇你希望用的分隔符號。

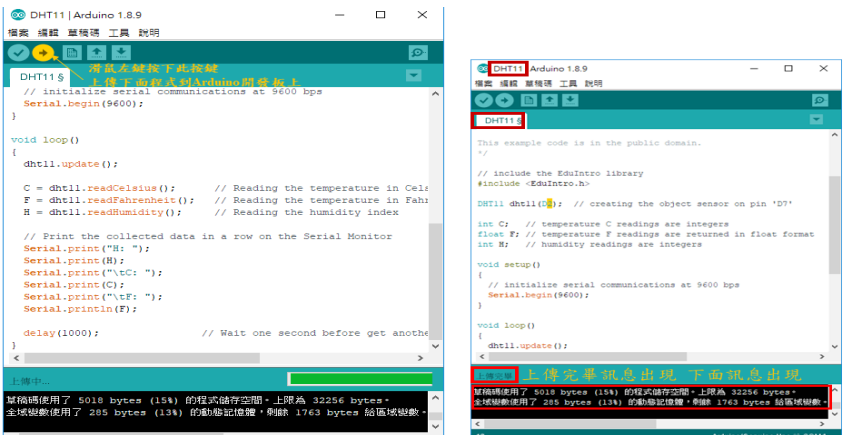


圖 10-14 Arduino IDE 中上傳 DHT11 修改檔與上傳成功的訊息

DHT11 程式上傳成功後，即可參考圖 10-15 左去 Arduino IDE 上方選單選擇工具，於工具下方出現的選單中選擇序列埠監控視窗，圖右即開啟後的監控視窗，這裡請注意右下方的 Data 傳輸速率要選擇與剛上傳到 Arduino 開發板的 DHT11 程式中相同的速率 9600 bps(baud rate)。

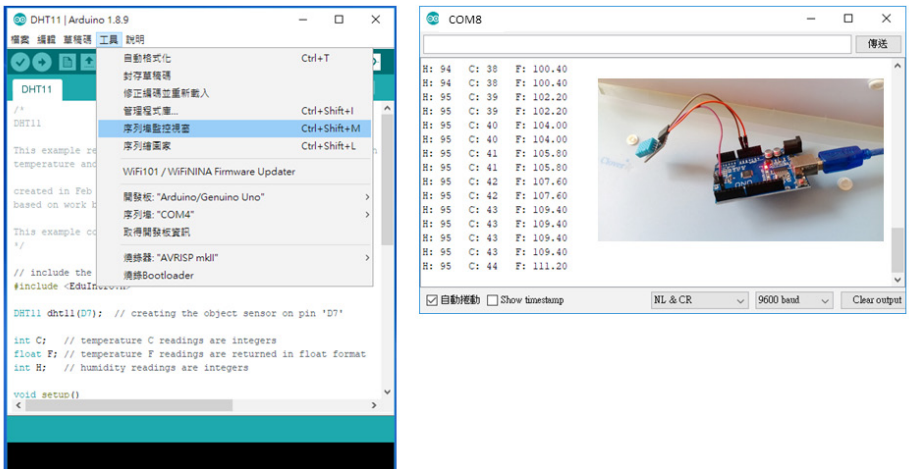


圖 10-15 開啟 Arduino IDE 下的序列埠監控視窗檢查開發板接收到的溫溼度數據

由於序列埠監控視窗開啟時會獨佔開發板目前使用的序列埠 (COM port)，而 Python 程式執行時也需要獨佔開發板使用的序列埠 (COM port)，兩者無法同時獨佔。因此，一個好習慣是序列埠監控視窗確認有得到溫溼度訊息，感測器正常運作後即可按下右上方 X 將其關掉，釋出序列埠 (COM port) 給 Python 程式使用。如果 Arduino 的序列埠監控視窗使用 (圖 10-16 右) 中，序列埠 (COM port) 會被 Arduino 序列埠視窗獨佔，此時如果在 Python Shell 下鍵入並執行 `ser.open()` 程式碼 (圖 10-16 左)，由於 Arduino 序列埠視窗仍獨佔這個序列埠 (COM port) 未釋出給其他程式使用，就會出現圖左 Python Shell 下序列埠 (COM port) 存取被拒的錯誤訊息。

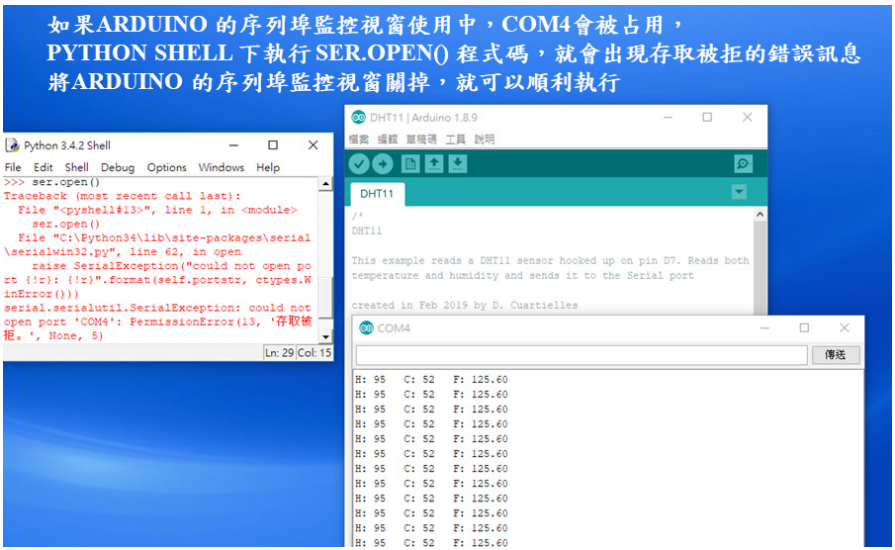


圖 10-16 Arduino 監控視窗獨佔 Arduino COM port 導致 Python 無法開啟序列埠

這時只需將 Arduino 序列埠監控視窗關掉 (Arduino IDE 不須關掉)，再次如圖 10-17 左 Python Shell 下鍵入並執行 `ser.open()` 與後續的程式碼，就可以順利執行得到一組溫濕度值。

圖 10-17 關閉 Arduino IDE 的序列埠監控視窗就可以讓 Python 開啟 Arduino 的 COM port

反之，如果 Python 開始獨佔 Arduino 開發板的序列埠 (COM port) 後，此時想再打上傳修改程式到 Arduino UNO 板或開啟序列埠監控視窗，Arduino IDE 視窗下就會出現橘色的開啟序列埠錯誤的訊息。

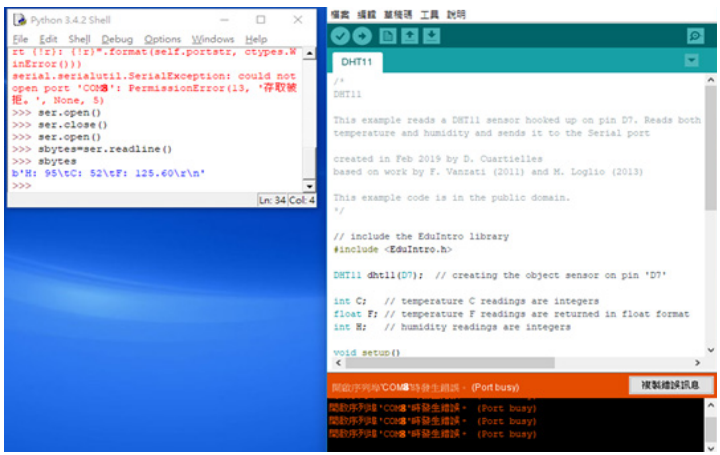


圖 10-18 Python Shell 下開啟獨佔序列埠會導致 Arduino 無法獨佔序列埠

此時，只需在圖左最後一行 >>> 鍵入 `ser.close()` 或直接按 Python Shell 右上角 X 鍵結束程式執行，即可釋出這個序列埠給 Arduino 使用。

10-2 Arduino 開發板在資料擷取之應用：pySerial 訊號處理

pySerial 套件下程式指令和語法

pySerial 套件提供裝置透過序列埠連接電腦的函式，這一節中我們將介紹基本的 pySerial 指令，讀者請跟著我們提供的範例，搭配 Arduino UNO 連接電腦後實際測試。

當 Arduino UNO 連接好後，首先可以測試序列埠是否順利連接，使用的指令是 seial.Serial()，如下面的程式範例：

E10-2-1.py: 開啟序列埠及狀態檢查

```
01 # 匯入 serial 模組
02 import serial
03 #serial.Serial 括弧內可以設定要開啟的序列埠與傳輸方式
04 ser=serial.Serial('COM8')
05 # 螢幕輸出 ser.name
06 print(ser.name)
07 #ser.is_open 回傳 COM8 是否是開啟的布林值
08 print(ser.is_open)
09 #ser.readline() 回傳 COM8 有 \n 結尾的一行溫濕度資訊到變數 s 中
10 s=ser.readline()
11 print(s)
12 ser.close()
```

執行結果：

```
COM8
True
b'H: 61\tC: 22\tF: 72.00\r\n'
```

ser.name 設定好 COM8，ser.is_open() 回傳 True，即 COM8 是開啟的。連接埠順利連接後，接著是透過 readline() 函式讀取感測器所偵測的數值。

螢幕上輸出溫濕度值 61% 和溫度值 22C。這裡的輸出字串前面出現 b 的符號，代表資料以 bytes 的方式呈現。bytes 的方式會把字串內每個字元全部列出，因此後面會包含 \r 與 \n 這種特殊符號。我們希望能將 bytes 的字串轉成 Python 常用的 string 格式，需要經過資料轉譯工作。後面以一個 Python 程式範例，列出常見的資料轉譯指令供大家參考使用。實務上，bytes 字串可以轉成 ASCII，也可以直接當字串來處理，因此程式寫法不只一種，讀者可以自行實驗將 bytes 字串以字串方式來處理，擷取需要的部份字串，讀者也可以上網找其他類似 Python 程式範例來比較觀摩學習，然後再依實際需要去寫 Python 程式。這裡先示範將 bytes 字串轉成 ASCII 再進行後續資料處理，再示範 bytes 字串不做轉譯，直接以字串方式做後續資料處裡。

Python Example 1

當溫濕度感測器安裝完畢後，請想像一下此裝置的運作：感測器一直讀取外界空氣的溫濕度資訊，當 Python 程式要求讀取溫濕度資訊時，Arduino UNO 就會把當下感測到的溫濕度傳回給程式。因此，我們在撰寫程式必須根據應用系統的需求，決定顯示溫濕度的時機。

在這個程式範例中，我們直接用一個迴圈，『不間斷』的要求 Arduino UNO 回傳溫濕度資訊。請注意這個『不間斷』的意思不是真的完全沒有間斷，當每個迴圈的 `readline()` 函式要求傳回溫濕度資訊時，Arduino UNO 就取出感測器的溫濕度值；至於這些溫濕度值有多麼『不間斷』，其實是看每次迴圈中呼叫 `readline()` 函式的間隔多久。

E10-2-2.py

```
01 import serial
02 import time
03 ser=serial.Serial()
04 #ser.baudrate=9600
05 ser.port='COM8'
06 ser.open()
07 print('Serial Port opened:',ser.name)
08 for i in range(10):
09     sbytes=ser.readline()
10     sstrs=sbytes.decode('ASCII').strip()
11     DTH=sstrs.split('\t')
12     t=time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())
13     print(str(i+1).rjust(2),t,'Temp='+DTH[1][-2:]+chr(176)+'C',
14           'RH='+DTH[0][-2:]+'%')
15 ser.close()
```

執行結果：

```
Serial port opened: COM8
 1 2018-12-16 20:01:40 Temp=23°C RH=64%
 2 2018-12-16 20:01:41 Temp=23°C RH=63%
 3 2018-12-16 20:01:43 Temp=23°C RH=64%
 4 2018-12-16 20:01:45 Temp=23°C RH=63%
 5 2018-12-16 20:01:48 Temp=23°C RH=63%
 6 2018-12-16 20:01:50 Temp=23°C RH=63%
 7 2018-12-16 20:01:52 Temp=23°C RH=63%
 8 2018-12-16 20:01:54 Temp=23°C RH=63%
 9 2018-12-16 20:01:56 Temp=23°C RH=63%
10 2018-12-16 20:01:58 Temp=23°C RH=64%
```

```
09 sbytes =ser.readline()
```

從機器讀取的 bytes 的資料型態為一行 b 字母起首， \n 結尾的資料，09 這行將 bytes 資料指向新變數 sbytes 的記憶空間，sbytes 的內容為 b'H: 64\tC: 23\tF: 72.00\r\n'。

```
10 sstrs=sbytes.decode('ASCII').strip()
```

將 sbytes 轉譯前後空格都刪去的字串指定給新變數 sstrs，sstrs 的內容為 'H: 64\tC: 23\tF: 72.00\r\n'。

```
11 DTH=sstrs.split('\t')
```

將 sstrs 字串資料以 split('\t')，將溫度和濕度以 list 新變數 DTH 儲存，DTH[0] 的內容為 'H: 64'，DTH[1] 的內容為 ' C: 23'。

```
12 t:time.strftime('%Y-%m-%d%H:%M:%S',time.localtime())
```

'2018-12-16 20:01:40'，利用 time.strftime() 將讀取時間轉存成引號內輸出格式。

```
13 'Temp='+DTH[1][-2:]+chr(176)+'C'
```

利用 chr(176) 將 ° 符號加在溫度值和 C 中間，從 'Temp=' 變成 'Temp=23°C'。

```
14 'RH='+DTH[0][-2:]+'%'
```

DTH[0][-2:] 是 '64'，加上前後的字串，結果變成 'RH=64%'

Python Example 2

本範例同樣讀取溫度與濕度資訊，但是在資料處理時不另外將 byte 字串轉譯成 ASCII 字串。在第四行程式的 COM8 後的 9600 是溫溼度資訊的傳輸速率或波特率，這個速率需與上傳開發板的 DHT11 程式中的波特率相同。如果希望改變成其他速率 (如 19200, 38400 等)，則 Arduino 和 Python 兩邊程式中的波特率必須相同，否則會出現亂碼而導致 Python 程式錯誤。

E10-2-3.py

```
01  import serial                    # 匯入 serial 模組
02  import time
03  # Winsows 系統使用 COM8 以波特率 9600 連線序列埠並命名為
04  ser
05  ser=serial.Serial('COM8',9600)
06
07
08  if ser.is_open:
09    print(ser.name)          # 螢幕上顯示使用的 序列埠
10    for i in range(10):   #for 迴圈 , i = 0,1,...,9
11      #讀一個 '\n' 結尾的一行並以字串 (string) 資料型態儲存到變數 s 中
12      s=str(ser.readline())
13      t=time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())
14  #將變數 s 儲存的字串指標 [5:7] 的部份字串 ( 即濕度 ) 加上攝氏的 '%符號
15  #將變數 s 儲存的字串指標 [12:14] 的部份字串 ( 即溫度 ) 加上相對濕度的 'C符號
16      print(str(i+1).rjust(2),t,'Temp='+s[12:14]+chr(176)+'C',
              'RH='+s[5:7]+'%')
    ser.close()
```

執行結果：

```
 1 2020-11-20 17:19:37 Temp=27°C RH=55%
 2 2020-11-20 17:19:38 Temp=27°C RH=55%
 3 2020-11-20 17:19:39 Temp=27°C RH=55%
 4 2020-11-20 17:19:40 Temp=27°C RH=55%
 5 2020-11-20 17:19:41 Temp=27°C RH=55%
 6 2020-11-20 17:19:42 Temp=27°C RH=55%
 7 2020-11-20 17:19:43 Temp=27°C RH=55%
 8 2020-11-20 17:19:44 Temp=27°C RH=55%
 9 2020-11-20 17:19:45 Temp=27°C RH=55%
10 2020-11-20 17:19:46 Temp=27°C RH=55%
```

10-3 Arduino 開發板在資料擷取之應用 matplotlib 圖表繪製

matplotlib

matplotlib 是 Python 常被使用的繪圖套件，我們在第五章已經花一整章的篇幅介紹。在 Arduino UNO 取得感測器資訊後，通常會使用 matplotlib 將資料以圖表的方式繪出，讓使用者透過圖表閱讀資料才能夠更清楚。第五章已經有提供 matplotlib 許多指令與範例，這裡我們仍然列出一些常用的 matplotlib 函式與繪圖方法，供讀者們參考。

subplot()

測試 matplotlib 模組之語法 - subplot 子畫面切割範例

E10-3-1.py

```
01  import matplotlib.pyplot as plt
02  plt.subplot(2,2,1)
03  plt.subplot(2,2,2)
04  plt.subplot(2,2,3)
05  plt.subplot(2,2,4)
06  plt.show()
```

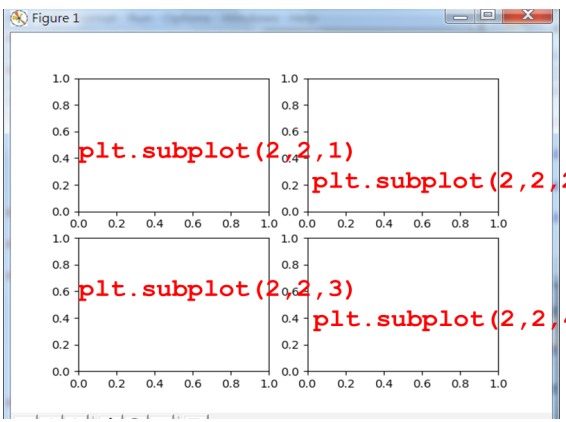


圖 10-19 以兩列、兩行放置四張圖的 matplotlib 語法

測試 matplotlib 模組之語法 - grid 顯示網格

E10-3-2.py

```
01  import matplotlib.pyplot as plt
02  plt.subplot(1,1,1)
03  plt.ylim(15,40)
04  plt.title('Real Time DHT11 Data')
05  plt.grid(True)
06  plt.show()
```

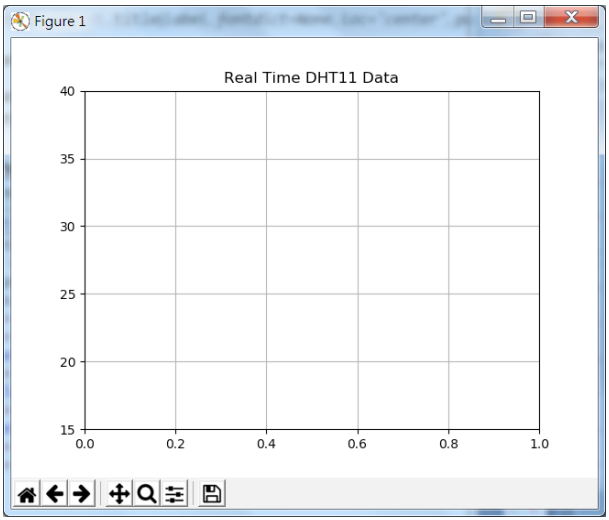


圖 10-20 matplotlib 圖中顯示網格 (grid)

主程式 – import

主程式需要匯入 pySerial、matplotlib 與 drawnow 這三個第三方模組 (third party modules)，第三方模組都必須透過命令提示字元視窗下如前面 pySerial 安裝時示範，用 pip install { 模組名稱 } 命令進行安裝，譬如，以 pip install matplotlib 安裝 matplotlib, 以 pip install drawnow 來安裝 drawnow。請先完成這兩個第三方模組的安裝再執行下面的 Python 程式。請特別注意，Python 程式中匯入名稱為 serial, 但安裝模組時命令是 pip install pySerial，經驗上，少數人會誤以為 Python 程式匯入的是 serial 而在命令提示字元視窗鍵入 pip install serial 這樣的錯誤安裝命令，導致沒有安裝正確的模組 (pySerial)，執行 Python

程式時出現找不到這個模組而無法順利匯入的訊息！這時必須在命令提示字元下先鍵入 pip uninstall serial 來解除錯誤安裝的模組，再鍵入 pip install pySerial 來安裝正確的模組。

在命令提示字元視窗下安裝 matplotlib:

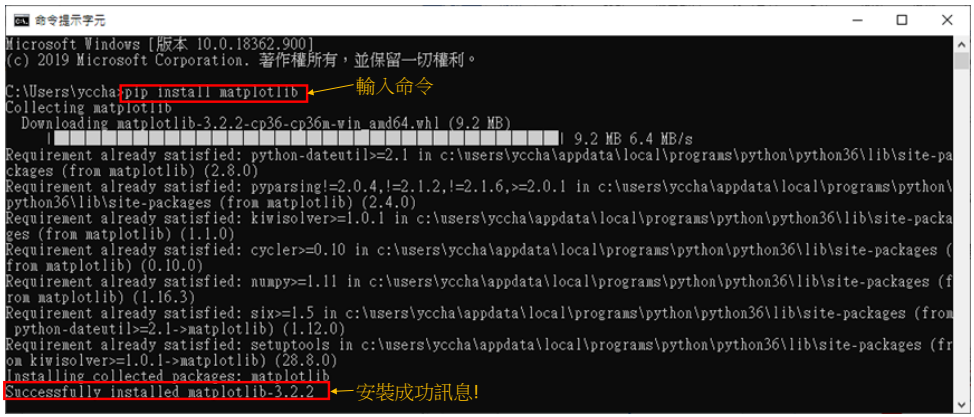


圖 10-21 命令提示字元視窗下安裝 matplotlib 並執行成功

繼續在命令提示字元視窗下安裝 drawnow:

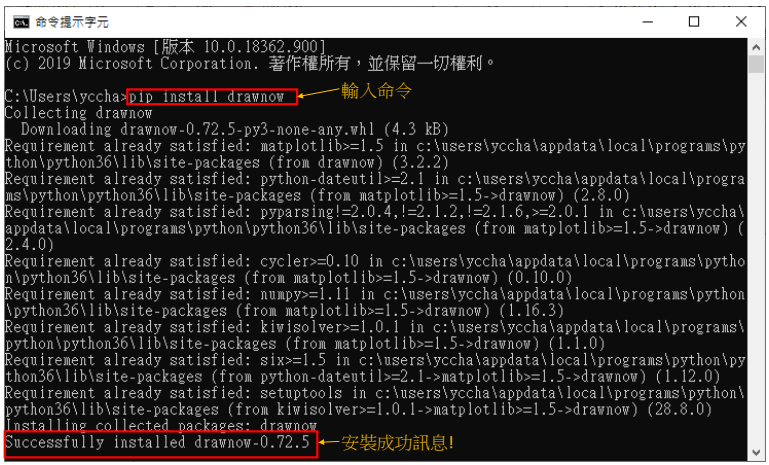


圖 10-22 命令提示字元視窗下安裝 drawnow 並執行成功

此外，也有少數人，因安裝命令的拼字錯誤而導致無法安裝，因此，如出現無法安裝錯誤，請仔細檢查是否有拚字錯誤。

圖 10-23 pip install 安裝錯誤與可能原因

E-10-3-3.py

主程式 - **import**

```
01  import serial  # 輸入 serial Library  # 序列埠
02  # 輸入 matplotlib library
03  import matplotlib.pyplot as plt
04  from drawnow import *
05  # 匯入第三方模組 drawnow，需要請讀者先行安裝，程式才能順利執行。
```

主程式 - CreatePlot() 繪圖函式

主程式 - **CreatePlot()** 繪圖函式

```
06  def CreatePlot():
07      # 設定上圖的高度、寬度
08      plt.subplot(2,1,1)
09      # 設定 y 軸的上下限值
10      plt.ylim(15,40)
11      # 上圖上方處加上標題
12      plt.title('Real Time DHT11 Data')
13      # 讓上圖中呈現灰色網格
14      plt.grid(True)
15      # 設定上圖 y 軸標題 ( label )
16      plt.ylabel('Temp C')
17      # 將溫度數據畫出來
18      plt.plot(tempC,'b^-', label='Degree C')
19      # 畫出符號說明 ( legend )
20      plt.legend(loc='upper center')
21      # 設定下圖的高度、寬度
22      plt.subplot(2,1,2)
23      plt.grid(True)
24      # 設定下圖軸上下限
25      plt.ylim(15,100)
26      # 將濕度結果畫出來
27      plt.plot(humidity, 'g*-',label='Humidity (%)')
28      # 設定下圖 y 軸標題 ( label )
29      plt.ylabel('Humidity (%)')
30      #to stop autoscale y axis
31      plt.ticklabel_format(useOffset=False)
32      plt.legend(loc='upper center')
```

主程式 - 設定

主程式 - 設定

```
33  # 設定 Serial port 為 Arduino 板接到的序列埠同時設定 Baudrate
34  Arduino = serial.Serial('COM8', 9600)
35  # 指定使用 matplotlib 的互動模式來繪製動態數據
36  plt.ion()
37  tempC = []
38  humidity = []
39  count=0
```

主程式 - 主迴圈 (從 Arduino 讀值後繪圖輸出)

主程式 - 主迴圈 (從 **Arduino** 讀值後繪圖輸出)

```
40  #While loop 不停的循迴
41  while True:
42      # 等待直到有數據
43      while (Arduino.inWaiting()==0):
44          # 沒有數據就繼續等
45          pass
46      #[ 有數據 ] 從 serial port 讀取數據
47      ArduinoString = Arduino.readline()
48      sstrs=ArduinoString.decode('ASCII').strip()
49      # 將數據拆解成一個列表
50      dataArray = sstrs.split('\t')
51      # 將第一個部分字串轉換成數字
52      temp = eval(dataArray[1][-2:])
53      # 將第二個部分字串轉換成數字
54      hum = eval(dataArray[0][-2:])
55      # 藉 append 將每個 temp 數據存到列表 tempC
56      tempC.append(temp)
57      # 藉 append 將每個 hum 數據存到列表 humidity
58      humidity.append(hum)
59      # 由於溫濕度數據會不斷更新呈現最新的 20 個溫溼度數據，
60      # 因此需要透過第三方模組 drawnow 所提供的功能來更新
61      # (update) 呈現 CreatePlot 函式執行產生的新圖檔。
62      drawnow(CreatePlot)
63      # Pause for interval seconds.
64      plt.pause(.000001)
65      count=count+1
66      # 假如數據多於 20 筆只取最後 20 筆
67      if(count>20):
68          #pop out first element of tempC
69          tempC.pop(0)
70          #pop out first element of humidity
71          humidity.pop(0)
72
73
```

主程式 - 結果

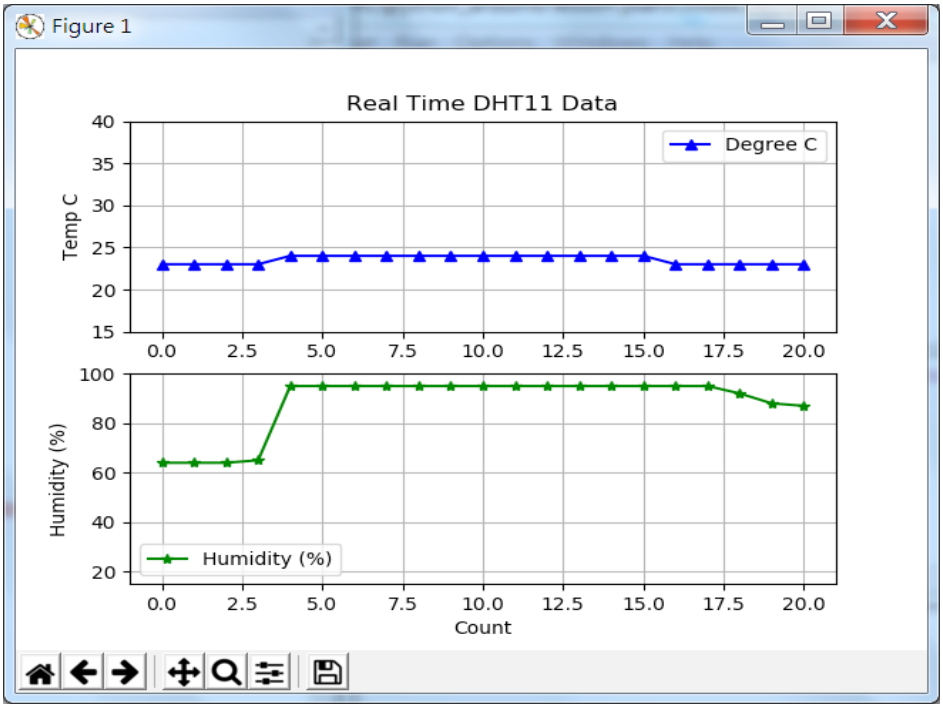


圖 10-24 程式範例執行後產生最新 20 組溫濕度數據的圖示

由於 Python 程式執行後，會開啟兩個視窗，一個是 Python Shell 視窗，一個是圖 10-24 這個視窗，如果需要結束程式，請直接關閉 Python Shell 視窗，圖 10-24 視窗就會自動跟著關閉。否則僅關閉圖 10-24，沒有關閉這個程式執行後開啟的 Python Shell 視窗，這個程式的迴圈仍會繼續執行 drawnow 更新圖檔並再度開啟圖 10-24 視窗。此外，如果這個程式的 Python Shell 沒有關閉，程式仍會獨佔這個 Arduino 板的 COM port ，導致您要執行的其他程式無法獨佔這個 COM port 。因此，請務必關閉這個 Python 程式執行時開啟的 Python Shell 視窗，方能釋出 COM port 給其他程式如 Arduino IDE 使用。關閉 Python Shell 右上角 X 鍵，會出現 Kill? 視窗，詢問 Your program is still running! Do you want to kill it? 按下確定鍵即可停止程式關閉溫濕度圖示互動視窗。

Chapter11

玩聲音，數位音訊的無限可能

/ 林宜徵

程式與音樂是一個乍聽沒有關聯，但實際上已結合許久的應用領域。本章將介紹數位音訊原理、以程式處理音訊，以及運用圖像式工具編輯 MIDI 訊號和製作簡單合成器。本章共有四小節，包括 MIDI 訊號讀取分析、PureData 音樂編輯程式與 MIDI 創作、音訊原理與頻譜檢視和 PureData 合成器製作。學習完本章後，讀者將能撰寫程式製作 wav 音訊，也能以 PureData 工具進行簡易音樂編輯與創作。

緒論

數位音樂一般來說包含兩種資料型態，一為音訊（如 .wav 或 .aiff 等），一為 MIDI 音訊。若要做任何數位相關的音樂處理，無論是創作或分析，皆無法離開此二種資料型態的範疇。本章分別介紹兩種音樂數位訊號（音波與 MIDI），每一主題段落內之內容又區分為：(一) 用基礎程式語言和套件讀取基本數位訊號的格式和訊息；(二) 以高階的圖像化音樂編輯程式指導如何編排和應用這些數位訊號。本章內容主軸從音樂資料的分析到運用高階的圖像化程式實作較為複雜的練習作業。若無高階的圖像化程式輔助，僅用一般程式語言處理的話，音樂的製作過程會相當冗長費時與無法掌握實作成果，因此在業界，採用數位音樂工作站或圖像化程式是最普遍的做法。以下是本章後續內容的簡要清單。一般音樂製作人在創作編曲時多半仰賴 MIDI 訊號的編輯，MIDI 相對屬於需音樂背景知識方能掌握之訊號；而音訊處理則是屬於聲響工程或合成器開發之工程人員所熟悉的領域，需工程數學背景知識方能掌握。讀者僅需知道此二種格式的特性以及能以程式讀取或以軟體作基礎編輯運用即可，不需詳記相關算式或訊號格式。

11-1 簡介 MIDI 與 MIDI 訊號讀取分析

自從 MIDI 訊號問世後，音樂產業中音樂的製作流程可說是受到極大的衝擊和影響，過去寫譜、印譜、排練、演出、錄音的音樂生產流程，轉變為 MIDI 訊號作曲、編曲、部分聲部（如主奏樂器）錄棚、混音（於電腦上透過音訊部分頻率音量的調整平衡）、最後過帶輸出。流程上大幅減少對於演奏者錄製音樂的需求，影響產業甚鉅！

音樂資料類型概述與基本 MIDI 訊號讀取分析

- Digital Audio Workstation 簡介
- 簡介 MIDI 與 MIDI 訊號讀取分析

古代創作音樂的方式

- 作曲家創作（可經過出版商出版）
- 演奏家練習
- 演奏家演出

現代數位音樂創作音樂製作方式與流程

- 作曲（主旋律）
- 編曲（配器）、伴奏等 loop
- 錄棚（主唱、真實樂器演奏）
- 混音
- 過帶（輸出）
- 出版發行
- 上述流程除了錄棚外，其餘可由 Digital Audio Workstation (DAW) 完成。

聲音編輯的兩種基本方式：MIDI 與音訊

音樂數位介面（Musical Instrument Digital Interface，簡稱 MIDI）是一個工業標準的電子通訊協定。MIDI 並未帶有任何音訊訊息，僅有音符開始、結束之時間點、力度、音色、拍號等資料。

- 簡介 MIDI 與 MIDI 訊號讀取分析

上網搜尋 : MIDI to Text online converter

試試看：http://flashmusicgames.com/midi/mid2txt.php

上網搜尋任一 MIDI sequencer

試試看：https://onlinesequencer.net/import

Play with Chrome Music Lab:

試試看：https://musiclab.chromeexperiments.com/Song-Maker/

- Music21

Music21 為一款由美國麻省理工學院製作之處理 MIDI 訊號之套件，其有完整功能可協助分析或編輯音樂 MIDI 資料。

Music21 實作方式

1. 打開 Google Colab(須註冊 Google）

https://colab.research.google.com/notebooks/welcome.ipynb

2. 在 Google Colab 內新增一個 Python3 之記事本。

圖 11-1 新增記事本

將以下程式碼複製貼上，然後按左側三角執行。

E11-2-1.py

```
01 !pip3 install music21
02 from music21 import *
03 environment.set('autoDownload','allow' )
04 s = converter.parseURL('http://yuan.yocjh.kh.edu.tw/midi/td.mid')
05 s.show('text')
```

圖 11-2

程式碼解說

安裝 Music 21 之環境。

```
01  !pip3 install music21
```

把 Music21 載入使用。

```
02  from music21 import *
```

Music21 自動下載 URL 資料設定。

```
03  environment.set('autoDownload','allow' )
```

自動下載 URL 的 MIDI 檔案並轉成 Music21 之格式。

```
04  s = converter.parseURL('http://yuan.yocjh.kh.edu.tw/midi/td.mid')
```

以文字顯現 MIDI 檔案之內容。

```
05  s.show('text')
```

Digital Audio Workstation (DAW)

課後可自行嘗試：下載任一免費或試用版本之 DAW: Garage Band, Pro Tools, Reaper, Cubase 等，在其官網或 youtube 觀看教學影片，並自行嘗試使用 piano roll 作點旋律。

小結

運用 Colab 雲端服務，可以讓創建 Python 環境簡單一致化，不用擔心每台電腦程式環境不一致的問題。

讀者僅需知道如何在參考 Music21（或其他類似第三方套件）的官方說明下能正確讀取 MIDI 檔案即可，不需熟記套件的指令。

透過此節，讀者先從運用他人的 MIDI 讀取網站，到實際運用第三方套件撰寫程式碼讀取 MIDI，了解 MIDI 訊號提供的內容後，再用編曲軟體以 MIDI 訊號編曲，將更對 MIDI 訊號更熟悉且對加強學習樂趣。讀者可以嘗試同時輸入兩三首樂曲的旋律，或是用 piano roll 畫上簡單的圖形（也可運用不同的力度在 DAW 上為圖形『著色』），聽聽看，會很有趣的！

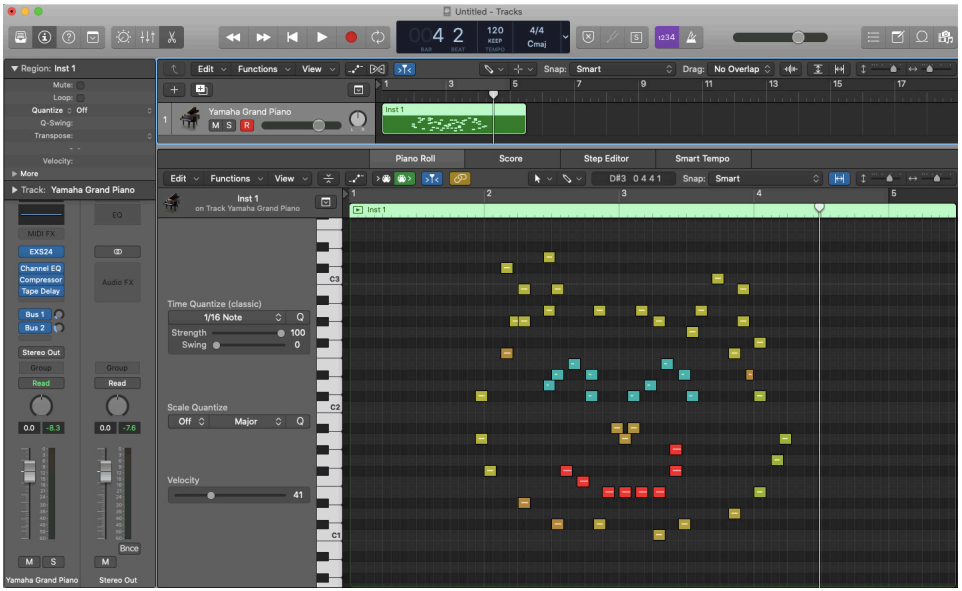


圖 11-3 用 piano roll 畫圖著色做音樂。

隨堂練習

11.1.1 . 請問以下對 MIDI 訊號之敘述，何者為非？

(A) 是一個工業標準的電子通訊協定。

(B) 不需經過 DAW 等軟體的音源設定，讀取後可直接由電腦產生聲響，作為跨平台使用，非常方便。

(C) 有音符開始、結束之時間點、力度、音色、拍號等資料。

(D) 在 DAW 上作曲，用此種訊號創作很方便。

參考解答：B，MIDI 訊號和音訊最大的差異就是 MIDI 不帶有任何音訊在裡面，因此在未設定音源和過帶的狀況下， MIDI 訊號無法產生任何聲音。

11.1.2 . 以下關於 Digital Audio Workstation 的敘述何者為非？

(A) 可用來錄音。

(B) 可用來創作。

(C) 僅能讀取 MIDI 檔案，無法讀取聲音訊號。

(D) 有了 DAW 後，音樂市場對於演奏者的需求減少了。

參考解答：C，DAW 可用來處理 MIDI 和音訊檔案格式，且此二種檔案皆能再其上做效果器或混音等處理。

11-2Pure Data（免費版）圖像化音樂編輯程式與 MIDI 創作

Pure Data 歷史、下載

Pure Data（或稱作 PD）是米勒・帕克特在 90 年代為創造交互的計算機音樂和多媒體作品而開發的視覺化程式設計語言。雖然帕克特是 PD 的主要作者，但是它是一個多數開發者一起開發新擴展的開放原始碼項目。它以一個類似於 BSD 許可證下發行，可運行在 GNU/Linux、Mac OS X、iOS、Android 和 Windows。Max/MSP 屬於 Pure Data 商業化的版本，讀者可自行斟酌是否採用。

Pure Data 下載

請至 https://puredata.info/ 下載對應 OS 之版本→下載後解壓縮→打開解壓縮後的檔案。

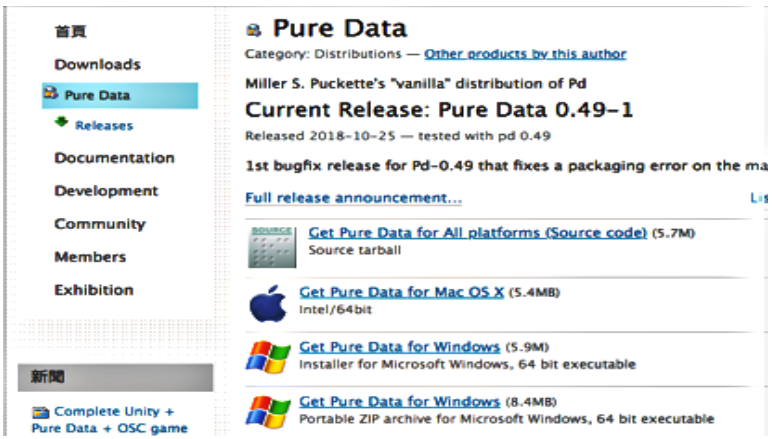


圖 11-4

PD 資料型態介紹

1. PD 有自己獨特的資料型態，包括： Object、Message、 Number、 Symbol 和 Comment。
2. 可用右邊的快捷鍵建立各種資料型態之物件。
3. 將這些物件連接起來，行程 Patch，便可執行許多數學運算、MIDI、甚至音訊等運算。

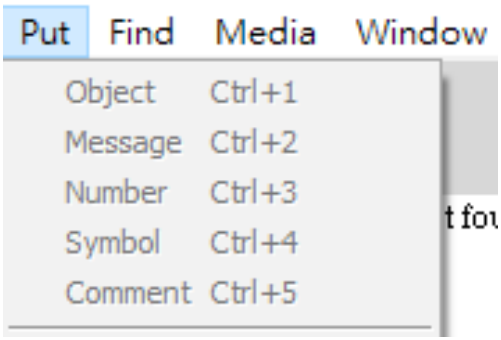


圖 11-5

Object

最主要資料型態，屬於傳統程式語言之 function 類別，本身具有邏輯運算的功能，當輸入正確的 API （與參數）時，會出現帶有粗黑短線條之黑色框，粗黑短線條為訊號輸入和輸出之接口，上端接口為輸入端，下為輸出端 。所有 Object List 可查詢：http://blazicek.net/list_of_pure_data_objects.html。

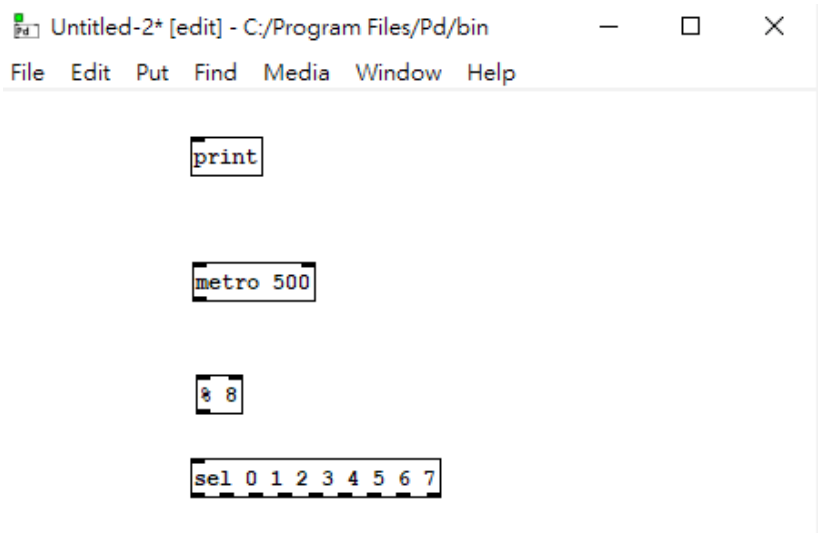


圖 11-6

Message

Message 屬於靜態字串類別，旗標狀，左側上下各有粗線接口，上為輸入端、下為輸出端。

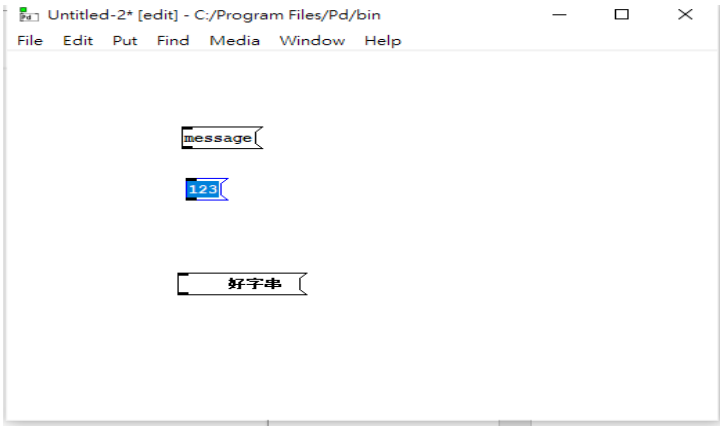


圖 11-7

Number

Number 屬於數字類別，可有負數、浮點值，右上方有缺角之框格狀，左側上下各有粗線接口，上為輸入端，下為輸出端，在 edit mode 中只能呈現預設值 0，切換到 play mode 時（crl+e or cmd+e），以滑鼠左鍵點按於該數字框格上下拖曳後，可更改數字。

以 message 輸入連接浮點數字後，在 edit mode 點按 message 框格後，number 即變成為可拖曳更改之浮點值（圖中最右）

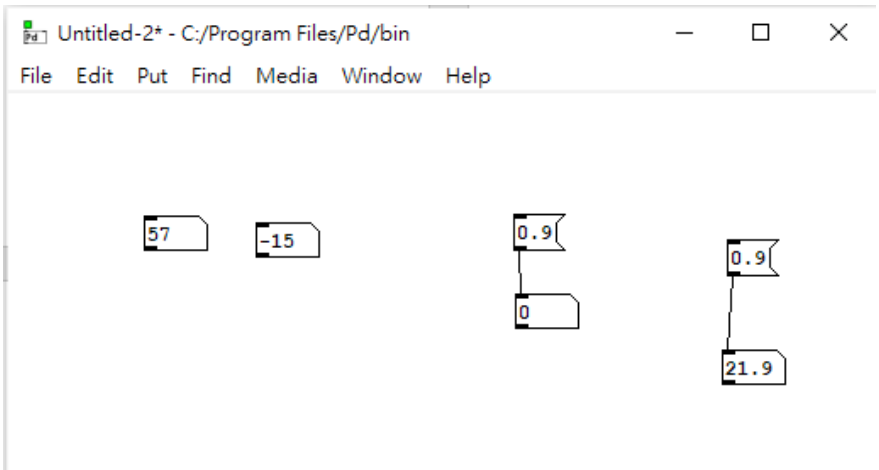


圖 11-8

Comment

1.Comment 可用來加註文字。

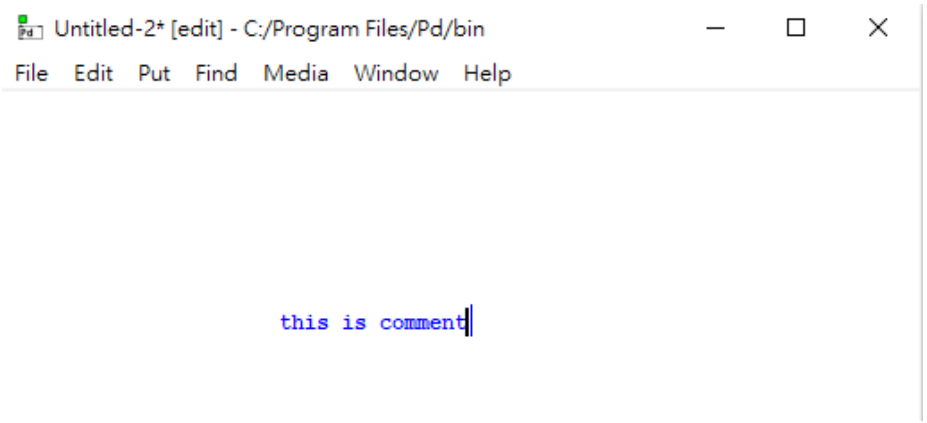


圖 11-9

Symbol

屬於 String，具有被快取（cached）或實際上永久儲存於 PD 的雜湊表中，可以被附上一些資料。PD 中有許多資訊是靠 Symbol 做辨認的，因此 Symbol 是一種非常重要的資料型態。一般來說初學 PD 較少使用 Symbol ，因此在此不多詳述，如有興趣了解請見以下資料連結： https://puredata.info/community/pdwiki/symbol

Help 功能

僅須將滑鼠以右鍵點按上述物件便能看到關於該物件的 *Help* 解釋。

PD 實作（旋律產生器）

在 Log 視窗勾選右上方的 DSP 來開啟音訊。依照下方圖示產生各種物件，並以滑鼠左鍵點按個物件的輸出或輸入端，以產生線條連結個物件。

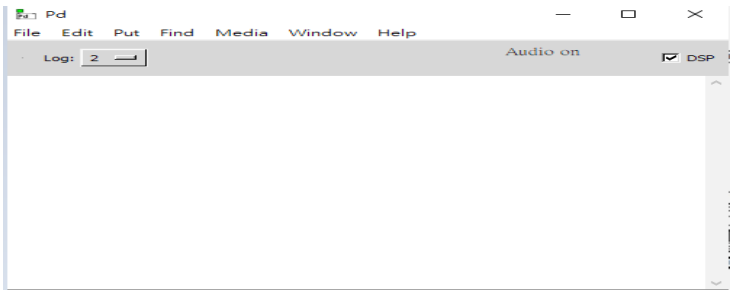


圖 11-10

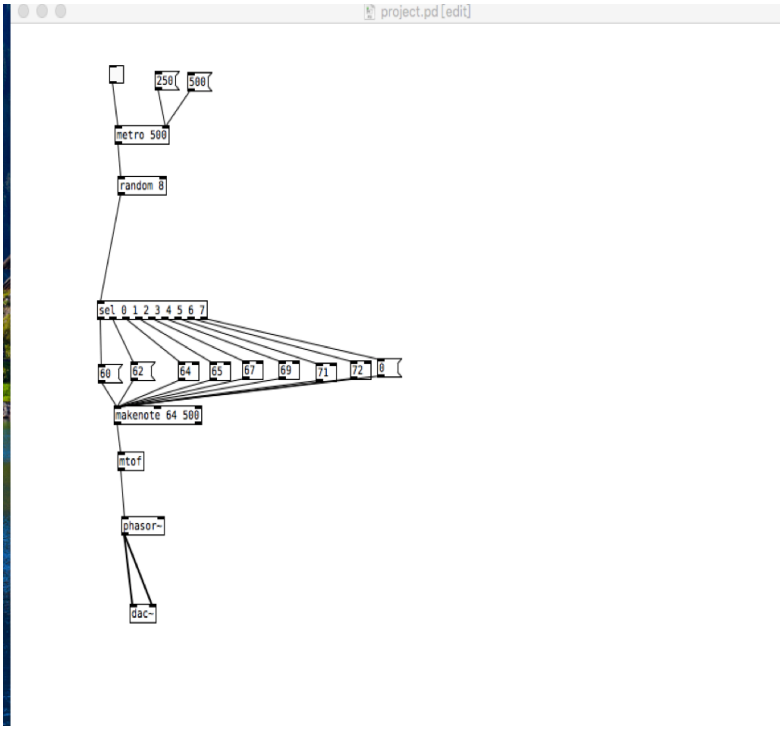


圖 11-11

小結

透過圖像式程式，同學們製作了第一個音樂的 AI 小工具，當我們撰寫程式，讓電腦產生類似人類的思考或判斷，就是人工智慧的範疇，AI 和我們的生活關係越來越密切，但終究是一個工具，如何設計工具和使用工具，最終也是由人的智慧去掌握的！

隨堂練習

11.2.1 . 請問以下關於 Pure Data 的敘述，何者為是？

(A) 是一種為創造交互的計算機音樂和多媒體作品而開發的視覺化程式設計語言。

(B) 為米勒・帕克特獨立開發的軟體 。

(C) 不支援 Mac OS。

(D) 無法處理 MIDI 訊號 。

參考解答：A

11.2.2. 請問以下 PD 的資料型態，何者為非？

(A) Object

(B) Message

(C) Comment

(D) Function

參考解答：D，PD 有自己獨特的資料型態，包括 : Object, Message, Number, Symbol, 和 Comment 。

11.2.3. 請問對於 PD 的操作方式，何者為是？

(A) 當需要使用註解時，可使用旗標狀的 Message 資料型態。

(B) 在 edit mode 下即可執行並更改 Number 資料的數值。

(C) 如要產生音訊，須在 Log 視窗勾選右上方的 DSP 來開啟音訊。

(D) Pure Data 為一款付費軟體 。

參考解答：C，初學者常忽略須在 Log 視窗勾選右上方的 DSP 方能開啟音訊

11.2.4. 請用 PD 撰寫一個程式，能同時產生二個旋律。

1. 二組旋律皆為五聲音階組成之旋律，而其中一組的音階高於 第二組一個八度。

2. 請用 orc~ 物件取代 phasor~ 物件，使音色較為悅耳。

3. 調整二個旋律的音量，使一組較大一組較小聲。

4. 利用音量為 0，設計一個二個旋律結束的開關。

5. 想想看還可加入什麼變化？

（提示：將課堂的練習再新增另一組，並結合在一起。五聲音階由 CDEGA 組成。）

參考解答：見附檔 project.pd

11-3 數位音訊原理 (基本音波撰寫) 與頻譜檢視

數位聲音訊號原理

聲音是一種波動，當演奏樂器、拍打一扇門或者敲擊桌面時，聲音的振動會引起介質 —— 空氣分子有節奏的振動，使周圍的空氣產生疏密變化，形成疏密相間的縱波，這就產生了聲波，這種現象會一直延續到振動消失為止。

一般的聲音總是包含一定的頻率範圍。人耳可以聽到的聲音的頻率範圍在 20 到 2 萬赫茲（Hz）之間（每秒震動循環次數）。高於這個範圍的波動稱為超音波，而低於這一範圍的稱為次聲波。

類比音訊與數位音訊

圖像有類比圖像（例如傳統底片式相片）與數位圖像的分別，類比圖像所有的線條為連續性，當一條斜線無限放大，還是斜線，但數位則是不連續性，當一條斜線被放大解讀，則會看到不連續之階梯狀線條。

類比音訊就比如日常生活的聲音，所有音波是連續的，但數位音訊則是指採取某些不連續的點以儲存入電腦中。

數位音訊採樣頻率

要將音波資料儲存入電腦等數位裝置，需要將一段音波的細分成許多點，將各點的聲壓數值依序儲存起來。細分成多少點，稱之為取樣頻率，有鑒於人類可聽頻率最高為兩萬赫茲，一個波循環，至少需有兩個取樣點，因此能聽頻率最高所需的取樣為 2 萬 乘以 2(取樣點）＝ 4 萬取樣點，依電腦的位元數，則為建議為每秒至少需取 44100 個取樣點以上，方能將人類能聽頻率全數收錄。

現行 CD 採樣頻率為 44100 Hz，其他如搭配影像或高音值聲音作品可將採樣頻率提高到 48000 ~ 192000 HZ 之間。採樣頻率越高，所需儲存空間則越多。

頻譜檢視與分析

用不同樂器演奏同樣的音高，聲音頻率雖然相同，但由於波形不同，形成不同的音色。各式各樣的波形，只要是循環波，便可以被分解為不同頻率、不同強度的正弦波的疊加。這種變換（或分解）的過程，稱為傅立葉變換。

透過傅立葉轉換成頻譜，我們可看到一個循環音波被分解成由低至高頻的多個正弦波之分布情形，以理解音色的特性。例如：聲音渾厚者的歌聲與聲音纖細者的歌聲，雖唱同樣音高，頻譜分析下，聲音渾厚者之低頻分佈會較多，聲音纖細者低頻分佈會較少。

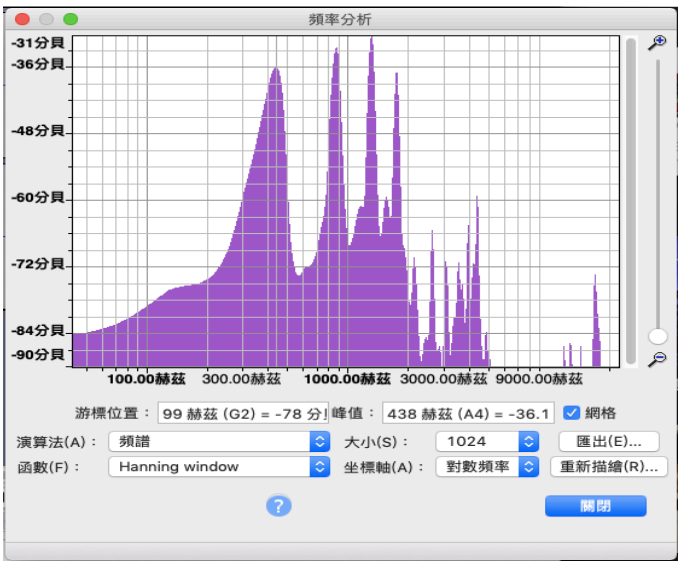


圖 11-12 頻譜範例

此為筆者人聲哼唱 標準音 A 音（440 Hz) 的頻譜分析，可見頻譜分佈並非僅於 440 赫茲，還有 440 的泛音與其他非泛音之雜訊頻率（沙啞聲）。越清亮渾厚的好歌喉產生的聲音，頻譜分佈會分佈在泛音列之上，非泛音列之雜訊頻率會較少。

下載 Audacity 與分析頻譜實作

Audacity 是一套免費的音頻分析與編輯軟體，可致其官方網站下載：https://www.audacityteam.org/download/。下載安裝並執行此軟體後，點按錄音鍵，便可用電腦的麥克風開始錄音，請哼唱一段小曲。

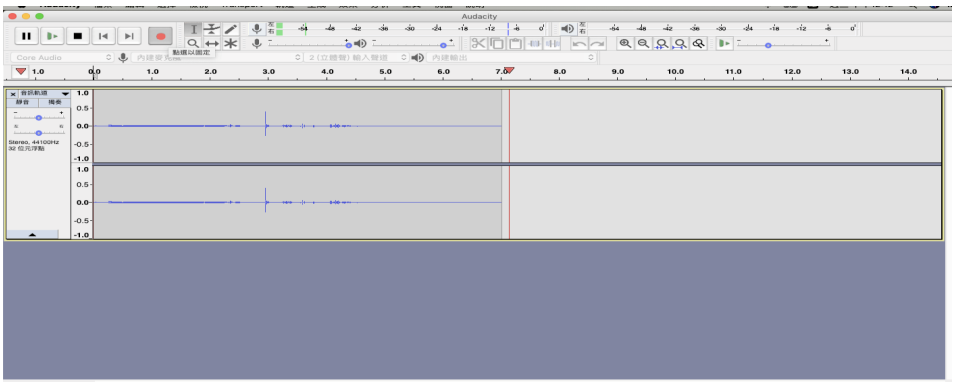


圖 11-13

點按上方紅色錄音鈕開始錄音，結束錄音請按停止鈕。

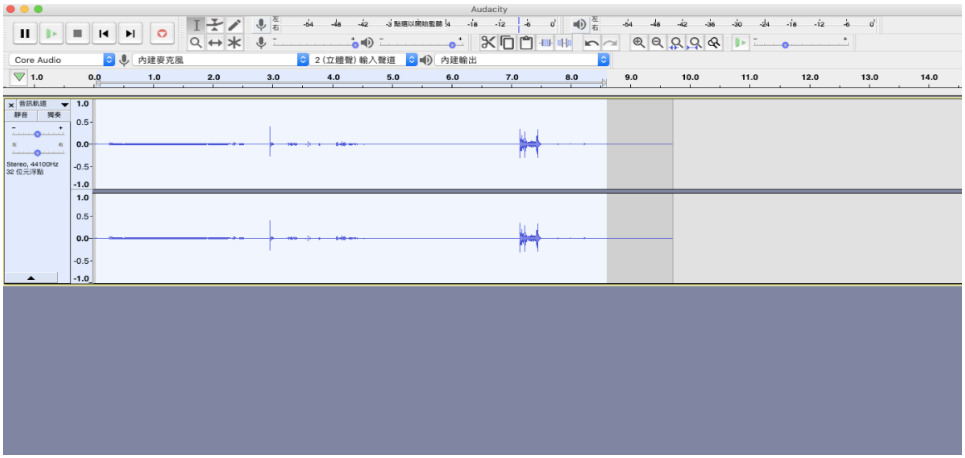


圖 11-14

點按停止鈕後，以滑鼠長按左鍵一邊選取想要分析的段落，被選取之段落會反白。選取後，於上方工具列中點按『分析 —> 描繪頻譜』。

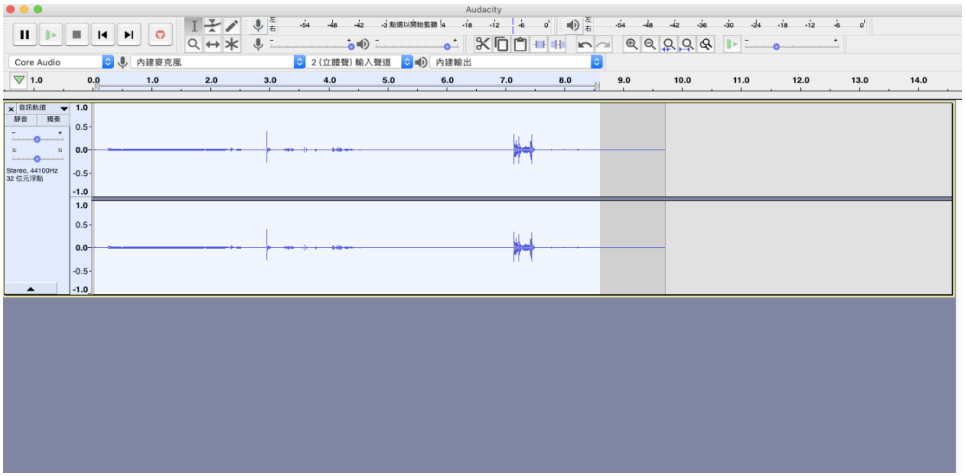


圖 11-16

完成之頻譜分析。

基本 sine wave 音訊程式撰寫

了解音訊原理後，接下來實際用 Python 來寫一個 sine wave 音頻訊號。

打開 Google Colab (須註冊 Google 帳號）。

https://colab.research.google.com/notebooks/welcome.ipynb

在 Google Colab 內新增一個 Python3 之記事本。

將以下程式碼複製貼上。

E11-3-1.py

```
01  import numpy as np
02  from scipy.io.wavfile import write
03  from scipy.io.wavfile import read
04  from google.colab import files
05  #sample rate per seconds
06  sps = 44100
07  #frequency
08  freq_hz = 440.0
09  # duration in seconds
10  duration_s = 4.0
11  amplitude = 0.3
12  each_sample_number = np.arange(duration_s * sps)
13  waveform = np.sin(2*np.pi * each_sample_number * freq_hz / sps)
14  waveform_volume = amplitude * waveform
15  waveform_intergers = np.int16(waveform_volume * 32767)
16  #32767 為振幅定值
17  write('sinewave.wav', sps, waveform_intergers)
18  read('sinewave.wav')
19  files.download('sinewave.wav')
```

01 行至 04 行為環境設定：scipy 套件內有一個 wavfile 可執行 wav 檔案的讀寫。

Sine wave 的數學式為：y(t) = A * sin(2*pi*f*t + phi)

先設定基本參數，sps （sample rate per seconds） = 44100；

欲產生的 sine wave 頻率 freq_hz = 440.0，時長 duration_s = 4.0（四 秒），振幅值 (A) 為 0-1，在此設定 0.3，amplitude = 0.3。

由於在數位處理時，必須給電腦每一個取樣的時間值 (t)， 因此對於每一次波的循環 t = 1/sps 為時 4 秒的 sine wave 將會有 4 * sps 的取樣點 (each_sample_number)，此外之後方便運算，將這些取樣點以 np.arange 方式計算。

套上公式：

```
waveform = np.sin(2*np.pi * each_sample_number * freq_hz /sps)
```

執行程式碼，便會自動下載一個 sinewave.wav 的檔案。

小結

正弦波（sine wave）是聲音的最基本單位，透過正弦波的組合，可組合成各種聲音，而這也是數位音訊的基本原理。

透過這節，讀者了解正弦波和聲音的關係，也能運用程式撰寫基本的正弦波，並轉成音訊檔。這些音訊的計算，屬於音訊處理的領域，為工程領域，並廣泛運用於聲響工程，如音響、合成器、聲音或音樂特徵擷取等專業之中。雖然音訊處理屬於艱深的工程領域，但讀者藉由本節課，可以對此數位聲響的運作原理有基本概念，進而對於音訊的多種格式，如 wav, mp3, aiff 等或是耳機或揚聲器響應頻率等基本生活相關知識，也能更了解與應用。

隨堂練習

11.3.1. 請問以下關於人耳可聽到的頻率，何者為非？

(A) 人耳可聽到的頻率平均為 20~20000Hz 之間 。

(B) 人耳對於各種中音頻的敏銳度較高音頻和低音頻為優。

(C) 人耳可以同時聽到多種音頻，並藉由頻率的差異辨認方向。

(D) 人耳可聽到的頻率範圍會隨年紀增加而遞增。

參考解答：D，範圍應隨年紀增加而遞減。

11.3.2. 請問以下關於數位音訊的敘述何者為是？

(A) 為連續的訊號。

(B) 揚聲器可以直接接收和讀取數位音訊。

(C) 依據人耳可聽範圍，數位音訊的採樣頻率至少應為 40000Hz 以上。

(D) 聲波最基礎的單位是方形波。

參考解答：C

11.3.3. 請問以下關於聲波的敘述何者為非？

(A) 不同樂器演奏相同音高，音色不同的原因是在於其波型不同。

(B) 一把好的木製樂器通常高頻的振幅（力度）較大，低頻較小。

(C) 當我們觀察聲波的圖形時， 須透過傅立葉轉換，方能計算出頻譜。

(D) 泛音是指與基音呈現倍率的頻率。

參考解答：B，通常好的木製樂器，通常低頻較為渾厚，聲音較具共鳴。

11.3.4. 請撰寫一個程式能產生白噪音（注意音量不要太大聲，振幅建議為 0.1，且揚聲器或耳機須先調整到最小聲，以免傷耳與損害到揚聲設備）提示：若把各種隨機的頻率疊加在一起，便能產生白噪音的效果。

E11-3-2.py

```
01  import numpy as np
02  from scipy.io.wavfile import write
03  from scipy.io.wavfile import read
04  from google.colab import files
05  import random
07  #sample rate per seconds
08  sps = 44100
10  # duration in seconds
11  duration_s = 4.0
13  waveform = np.arange (duration_s * sps)
14  each_sample_number = np.arange (duration_s * sps)
16  i = 0
17  while i <=500:
18    #frequency
19    freq_hz = 40000.0 * random.random()
20    waveform += np.sin(2*np.pi * each_sample_number * freq_hz / sps)
21    i+=1
22  amplitude = 0.1
23  waveform_volume = amplitude * waveform
24  waveform_intergers = np.int16(waveform_volume * 32767) #32767 為振幅定值
25  write('whitenoise.wav', sps, waveform_intergers)
26  read('whitenoise.wav')
27  files.download('whitenoise.wav')
```

請試著找出生活中的各種聲音，用頻譜分析後，找找看哪些聲音和白噪音類似？（提示：如風聲、海浪聲）

11-4 Pure Data 合成器製作

音訊疊加式合成器

一般日常聽到的器樂演奏之聲音，含有具循環性樂音頻率和非循環性的噪音頻率（例如擦弦、氣聲、或敲擊物件的噪音）所組成。可由正弦波的組合來模擬循環性樂音頻率的部分。循環性的樂音通常為某音的基音頻率以及該基因的倍數頻率（又稱泛音）所疊加合成，音訊上僅要將各取樣點的數值相加，便可製作。

PD 實作（音訊疊加式合成器）

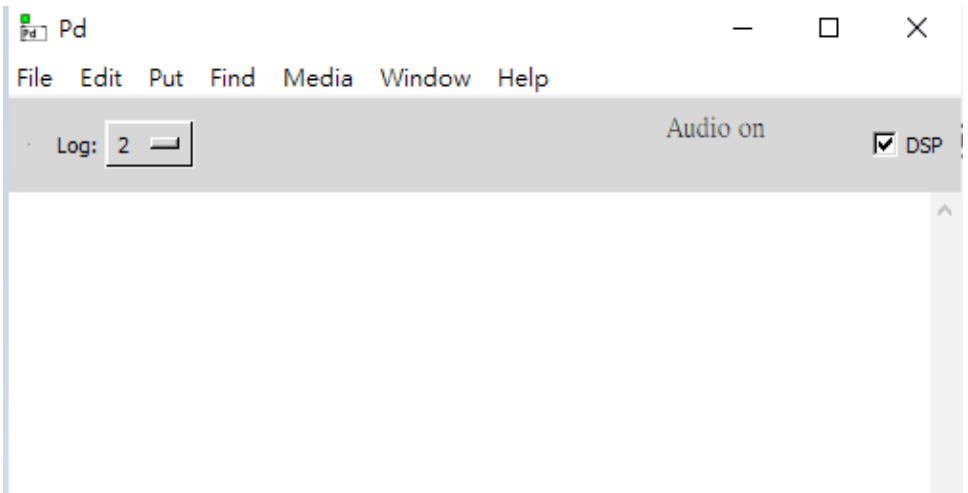


圖 11-18

在 *Log* 視窗勾選右上方的 *DSP* 來開啟音訊，依照上方圖示產生各種物件，並以滑鼠左鍵點按個物件的輸出或輸入端，以產生線條連結個物件。

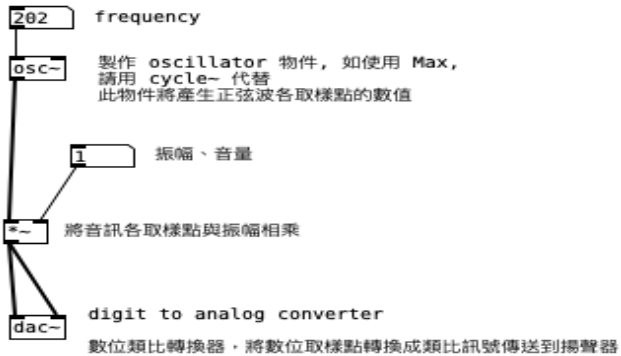


圖 11-19 單一正弦波

首先依照圖示製作一個單一的正弦波，運用 crtl + e 或是 comment + e 切換編輯 / 執行模式，在執行模式中，可用滑鼠左鍵拖曳調整 frequency 和振幅的數值，frequency 須在人耳可聽範圍方能被聽見，震幅則須介於 0~1 之間的浮點數。

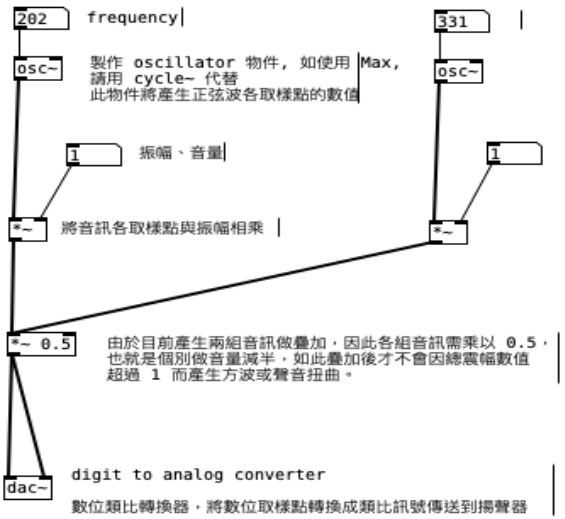


圖 11-20 第二組正弦波

可用滑鼠圈選欲重製的物件，按 crtl + c 以及 crtl + v 複製貼上，並拖曳到要放置的位置，以產生另一組正弦波。

由於兩組正弦波之振幅最高數值震幅為 1 的情況下，疊加後振幅為 2 ，因此需再將疊加後的振幅減半（乘以 0.5）使最高振幅不超過 1。

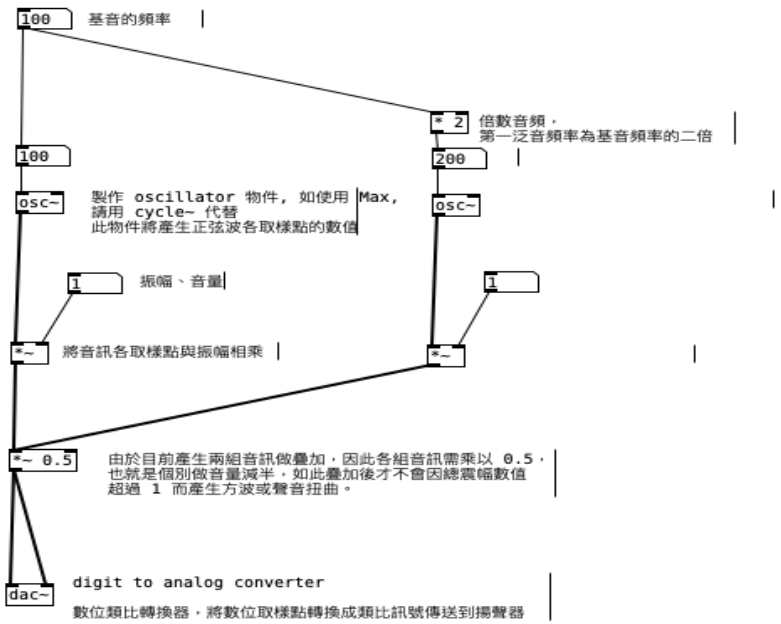


圖 11-21 使第二個正弦波與第一個正弦波為倍數關係

在上方加入一個數字物件作為基音的頻率，並連至第一個正弦波的頻率數字物件上（則第一個正弦波之頻率數字物件僅會重現基音頻率的數值）。

第二個正弦波上方加入相乘物件，在此因為要做第一泛音，其頻率為基音之兩倍，故乘以二，並將基音頻率物件連接到此相乘物件。

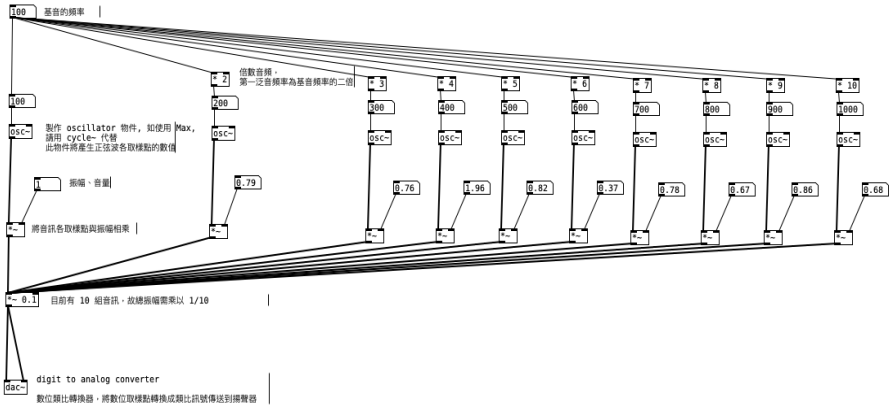


圖 11-22 製作其他九組泛音正弦波

- . 繼續製作其他九組泛音正弦波，並調整倍數值。
- 由於目前有 10 組正弦波，故將總振幅數值改乘以 0.1 (1/10)。
- 在執行模式下，對各組振幅數值之物件，按住 shift 鍵，並以滑鼠左鍵點按上下拖曳，便能產生浮點值（小數點），滑鼠的位置偏右一點，可調整小數點的位數。

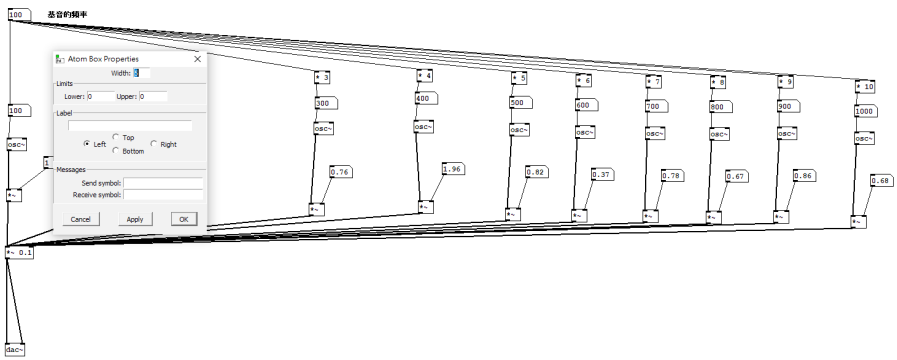


圖 11-23 設定數字物件的範圍

- 以滑鼠右鍵點按振幅數字物件，便會出現對話方塊，在方塊中設定數字物件的值範圍，在此設定為 1，如此在執行模式下，更易操縱。

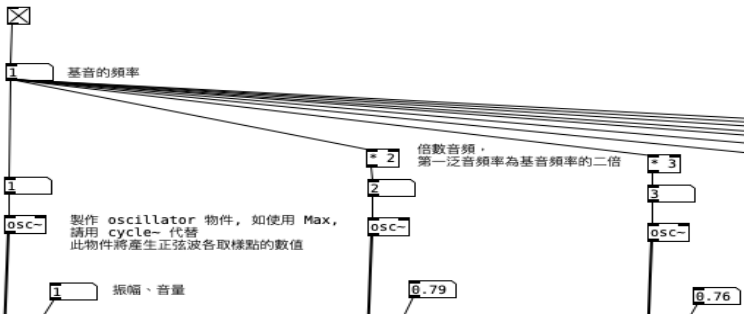


圖 11-24 總開關設置

- 在基音頻率上方新增一物件，並打上 toggle，此時便會自動產生方塊狀的開關物件。
- 將開關物件連接到基音頻率。
- 在執行模式下，可開啟開關（空格狀）或關閉開關（有 X 字狀）。
- 調整基音頻率，此時所有正弦波之數值將被連動，產生樂音。
- 透過調整各正弦波之振幅值，以改變音色，基音的振幅需高過其他泛音之振幅，如此，此總音波感知頻率才會維持在基音。

小結

由於音訊處理以 Python 程式語言撰寫相當不容易，且合成器製作與應用需要能即時播放的系統，因此許多數位音樂家或工程師在設計合成器時，會選擇 PD 或是 Max/MSP 的圖像式程式（但程式本身是以 C 語言為基礎作成的）創作，如此便可以越過訊號處理的計算知識直接進行聲響上的變化與選擇。除了圖像式語言，也有接近傳統程式語言的聲響程式，例如 SuperCollider, CSound

或 ChucK，而 Max/MSP 也能銜接 Pro Tools 等 DAW，對於有興趣鑽研數位音樂的讀者可深入研究這些程式工具。

參考資料

1. Google Colab 官方網站與使用說明：https://colab.research.google.com/
2. Music21 官方網站與說明：http://web.mit.edu/music21/
3. Pure Data 的操作方式，可參考以下連結：https://tuftsdev.github.io/MusicAppsOnTheIpad/readings/reading1.pdf
4. Pure Data 官方網站：https://puredata.info/
5. Audacity 官方網站與說明：https://www.audacityteam.org/
6. 資料引用：https://zh.wikipedia.org/wiki/%E5%A3%B0%E9%9F%B3

書 名：Python 程式設計與資料分析應用

企劃編輯 : 江家榕

文字編輯 : 王碩英、吳培弘、謝東霖

美術編輯 : 劉冠伶

出版日期 ：民國 109 年 12 月版

（非賣品）