Chapter

1

利用 Python 自動化工作

1 讓那些麻煩的例行公事自動化！
2 該如何構思解決問題的程式？
3 安裝 Python

Recipe 1 Chapter 1

讓那些麻煩的例行公事自動化！

Python 與工作的自動化

Python 可自動化各種工作。

Python 是非常**簡單易懂的程式設計語言**，不管是初學者還是專家都能輕鬆使用（圖 1.1）。此外，Python 可**處理的範圍非常廣泛**，可以取得或操作電腦的檔案，也能存取網路的資料，還能製作具有使用者介面的電腦應用程式，所以 Python 是能**快速讓各種工作自動化的語言**。

圖 1.1 Python

接下來，就來一起了解該怎麼利用如此方便的 Python **自動化工作**吧。只要利用 Python 撰寫程式，電腦就會代替我們完成工作。

不過，**工作**的種類有很多種，而要讓電腦進行**過於複雜的工作**時，有一些需要注意的事情。要讓「電腦進行複雜的工作」意味著「要撰寫複雜的程式」，而撰寫複雜的程式是件很困難的事，而且有時候只能用來解決「當下的工作」，無法於「下次的工作」使用，不然就是得花時間修正程式，才能於下次的工作應用。

因此，本書想要稍微換個角度思考。

那就是「不要將所有複雜的工作交給電腦處理」，而是「大部分的工作由人類負責，只有那些單純與機械化的作業請電腦幫忙」；換言之，要以「**自動化工作的輔助道具**」角度撰寫程式（圖 1.2）。

如果以「**有點單純又機械化的作業**」的角度撰寫程式，則這個程式應該就能在「這次與下次的工作」應用。簡單來說，就是請電腦負責「**有點單純又機械化的作業**」，然後人類負責「**需要動腦，而且真的很重要的工作**」。

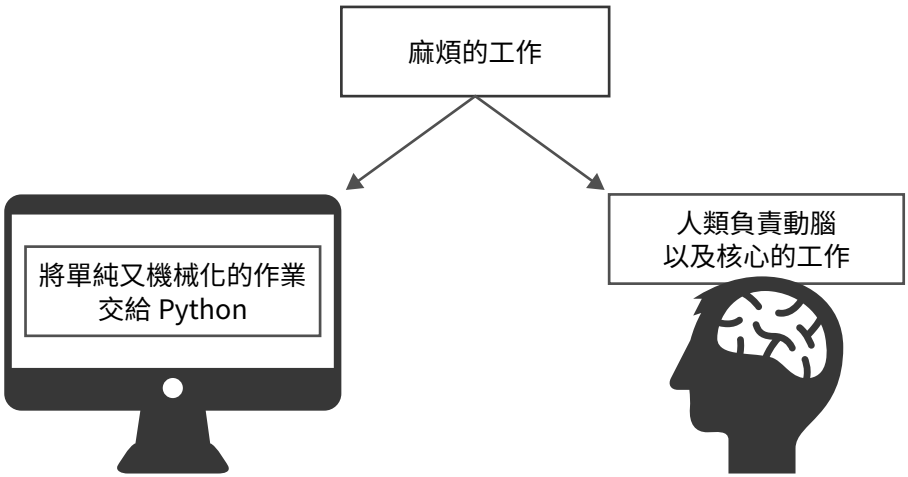

圖1.2 本書的自動化邏輯

因此，我們的目標就是讓「**有點單純又機械化的作業**」自動化。由於是簡單的作業，所以**只需要幾行程式碼**就能寫好程式的基本部分。如果撰寫了複雜的程式，然後讓電腦進行作業，這樣在傳回奇怪的結果時，恐怕會不知道「電腦到底完成了哪些部分」，反之，如果程式很簡單，就不會有這種問題。此外，**程式越簡單，就越方便自行改造**。

再者，簡單的程式也**比較容易與其他程式組合**。本書會將程式的基本部分寫成函數，**再以呼叫函數的方式執行作業**，如此一來，就能利用這些函數開發「**按個按鈕就執行作業的應用程式**」。

製作成應用程式的推薦

若問「為什麼要讓電腦幫忙我們完成作業呢？」答案當然是為了**讓工作更輕鬆**。如果不將程式檔轉換成應用程式，就必須在「想要完成某項作業」時，「先啟動 Python 環境，再載入執行該作業的程式，然後確認該程式是否能完成這項工作，再執行該程式」這類流程，但這樣一來，就沒辦法好好思考工作，工作也變得一點都不輕鬆。

反之，如果能將程式檔轉換成應用程式，整個流程就會簡化成「**執行應用程式**」與「**按下按鈕**」，我們就能仔細思考工作的細節。

因此本書要透過下列三大方向自動化工作（圖 1.3）

1. **撰寫簡單的程式（讓機械化的作業自動化）。**
2. **函數化（容易自訂內容）。**
3. **應用程式化（不需要中斷思考就能使用）。**

這裡說的應用程式，是「**輸入一些資料，再按下按鈕就能完成作業**」的應用程式。雖然不同的工作需要不同的應用程式，但本書介紹的應用程式架構都非常簡單，所以能使用於不同的工作。只要完成一個應用程式，這個應用程式的架構也能用來開發其他的應用程式。

圖1.3 應用程式的完成圖（示意）

利用範本讓函數轉換成應用程式

因此本書預先準備了多種應用程式的範本，可利用組合「函數」的方式完成需要的應用程式。簡單來說，就是選出需要的範本、追加函數，撰寫「**按下按鈕就呼叫該函數的程式**」，就能讓程式轉換成應用程式。

若問「為什麼要用這種方法製作應用程式」，答案就是**這種方法能像製作套件般，快速開發原創的應用程式**（圖 1.4）。「**想要自動化的作業**」會隨著工作的種類而改變，本書則打算利用上述的方式開放各種應用程式。

不過，還是有可能出現本書未及介紹的作業，此時大家有可能會想要利用同一套方法自行開發「**能完成這類作業的函數**」。「**想讓自訂的函數轉換成應用程式**」的人，請務必使用本書介紹的範本自製應用程式。

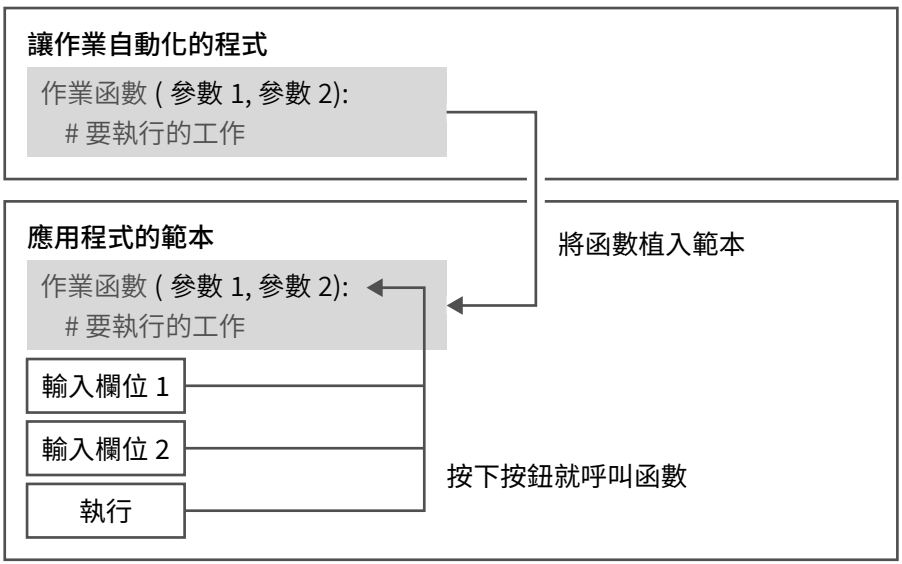

圖1.4 如同開發套件般，利用範本與函數組成應用程式

讓我們利用簡單好用的 Python，把每天的工作變得稍微輕鬆一點吧。

Recipe 2 Chapter 1

該如何構思解決問題的程式？

那麼該如何構思**代替人類執行作業的程式**呢？

第一步，可先找出執行該作業需要的功能。找到這些功能之後，就能著手開發，但如果心裡有一絲「這個功能該怎麼用？」「真的寫得出這種功能嗎？」的懷疑，很有可能會告訴自己「先做做看再說」。

這種先做做看再說的態度有時「會做出比想像中更棒的東西」，但通常都會做出不太好用的東西。為什麼會這樣呢？答案是**開發者的角度與使用者的角度不同**。

「想做出這種功能」是**開發者的想法**，而不是使用者的想法（圖 1.5）。因此，當我們想到「這種方法好像行得通」的時候，不妨停下腳步，站在**使用者的角度**重新檢視想法。

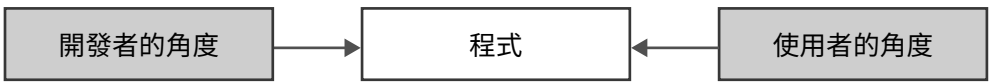

圖1.5 開發者的角度與使用者的角度

第一步要釐清目的。要以使用者的角度思考「**我在工作遇到哪些問題，希望電腦幫我完成哪些作業**」，根據使用狀況細分作業，才能從工作流程挑出**屬於人類的工作以及由電腦負責的工作**。

決定分工的部分之後，接著要站在電腦的角度思考（圖 1.6），也就是「**要完成這項作業需要哪些步驟與資料？**」從具體的作業內容思考。找出具體的作業內容之後，再思考「**Python 有哪些功能可以完成這些作業**」。

如果找到「似乎能派上用場的功能」就利用這些功能開發程式。如果沒找到，可訂立在外部函式庫尋找或是自行開發功能的計畫。

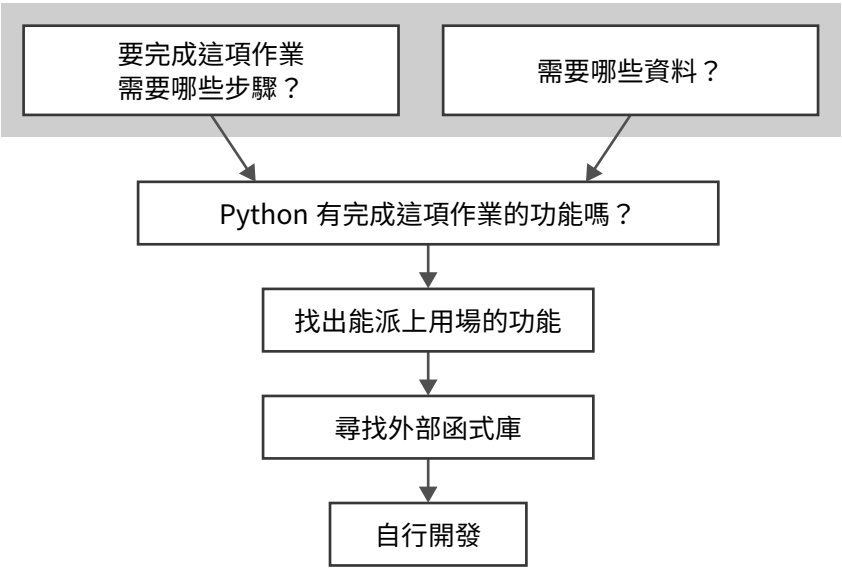

圖1.6 站在電腦角度的作業流程

建立上述的流程之後，就能開發出**人類與電腦分工清楚的程式**。

因此本書會依照下列的步驟開發程式。讓我們一起開發「方便使用的程式」吧。

1. 先搞清楚要解決什麼問題？

2. 思考要以什麼方法解決？

3. 找出解決問題需要哪些命令？

Chapter

2

基礎 Python

1 在IDLE執行Python
2 將資料放入變數再使用
3 利用if句進行判斷
4 大量資料可放入列表再重覆使用
5 在條件成立之時重覆執行處理
6 利用函數統整處理
7 函式庫是方便好用的函數集合體

Recipe 1 Chapter 2

在 IDLE 執行 Python

接下來要說明 Python 的基本知識。第一步請先啟動 IDLE，接著會自動開啟 Shell 視窗（圖 2.1、圖 2.2）。

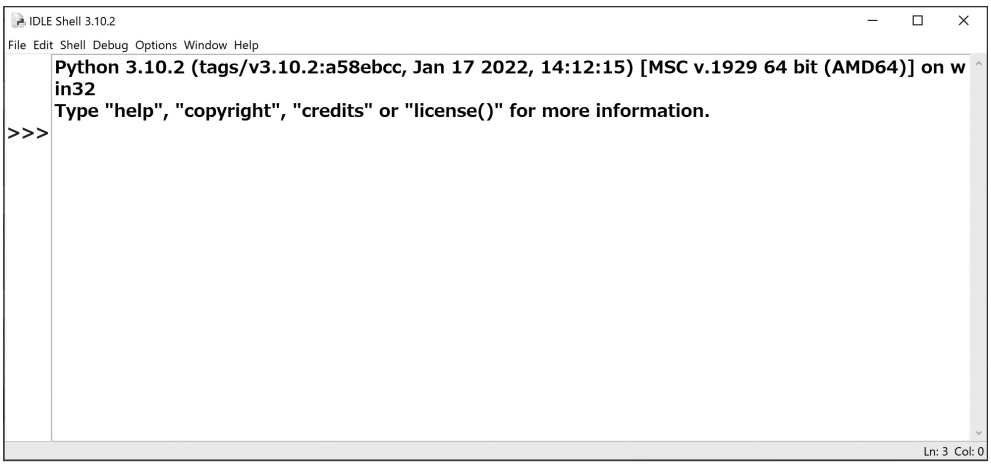

圖 2.1 Shell 視窗（Windows）

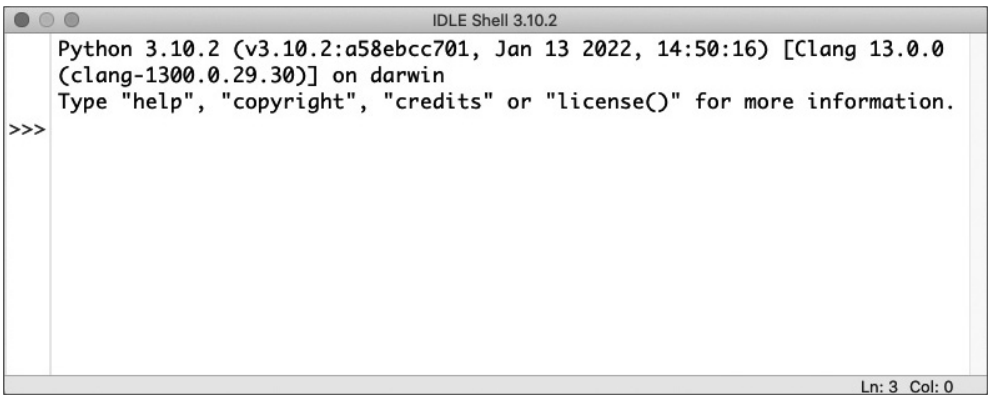

圖 2.2 Shell 視窗（macOS）

只要在這個 Shell 視窗輸入 Python 的命令就會立刻執行命令。

左側的「>>>」稱為**命令提示字元**，是等待使用者輸入命令的符號。在這個符號後面輸入 Python 的命令再按下「Enter」鍵就會立刻執行命令。

讓我們先試著執行最簡單的命令（語法 2.1），也就是**「print（值）」**。這是顯示「值」的命令句。這個括號可利用逗號間隔多個值，再一口氣顯示所有的值。

語法2.1 顯示值

```
print( 值 )
print( 值 , 值 )
```


也可以直接顯示數值或是四則運算的結果。四則運算使用的符號請參考表 2.1)，乘法與除法的符號與一般算數或數學符號有些不同，還請大家注意這點。

表2.1 四則運算的符號

符號	意義
+	加法
-	減法
*	乘法
/	除法
%	餘數
**	次方

命令可一行接著一行輸入。每次輸入之後，都會立刻顯示結果（圖 2.3）。

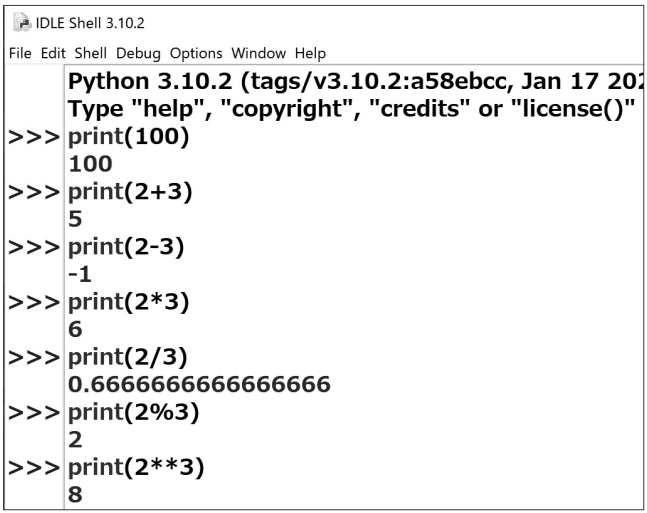

圖2.3 執行結果

※ 這章將以 Wiindows 的畫面解說。

像這樣在 Shell 視窗輸入命令的方法通常是用來**確認命令的內容**，而一般的程式都會將一堆命令存在檔案再一口氣執行。接著就要利用這種方法撰寫程式。

將命令存成檔案的方法大致分成三個步驟。

① **新增檔案，撰寫程式。**

② **儲存檔案。**

③ **執行。**

❶「新增檔案」。

從選單點選「File」（圖 2.4 ❶）→「New File」（圖 2.4 ❷）之後，會開啟新視窗（圖 2.5）。接著要在這個視窗輸入程式。

※ 如果程式輸入視窗沒有顯示行編號（在每一行開頭的編號），可從 IDLE 的選單設定。

Windows：從選單點選「Options」→「Configure IDLE」，視窗開啟之後，從「Shell ／ Ed」索引標籤勾選「Show line numbers in new windows」選項。

macOS：從選單點選「IDLE」→「Preference」，開啟視窗之後，從「General」索引標籤勾選「Show line numbers in new windows」。

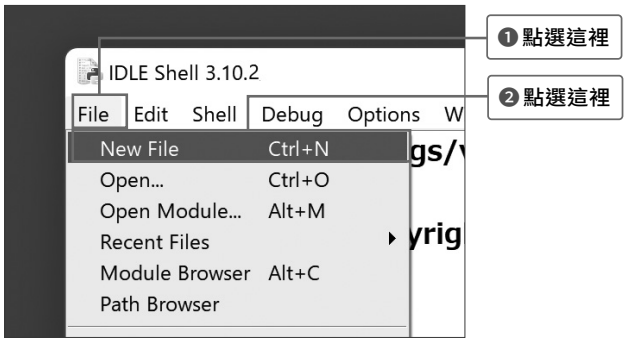

圖2.4 選單

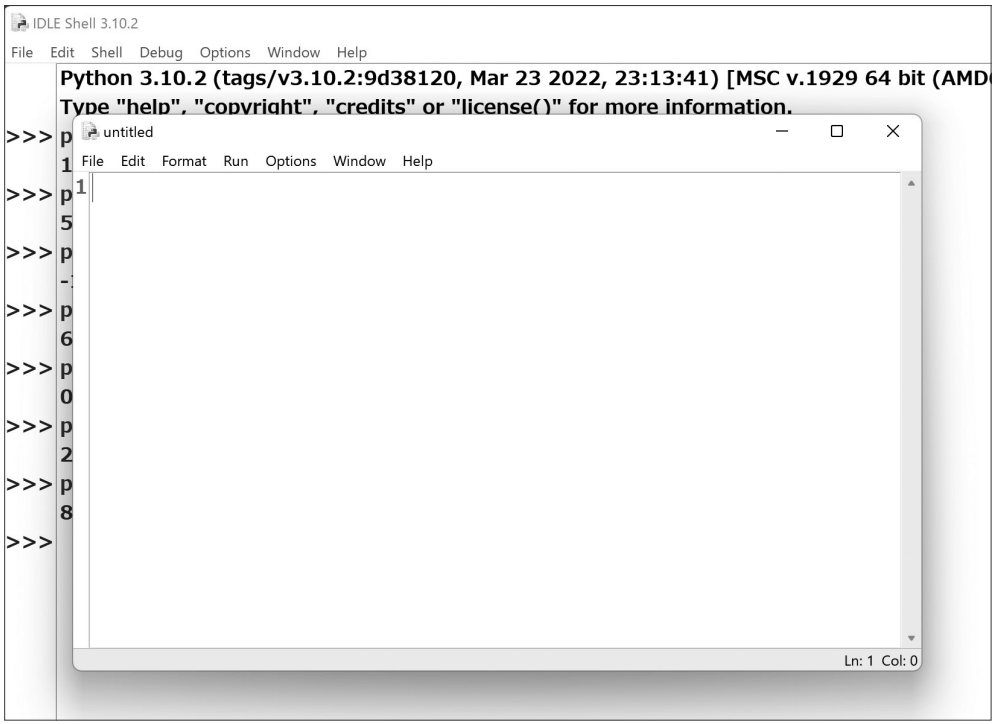

圖2.5 新視窗

接著讓我們試著輸入剛剛的命令（程式 2.1）。目前只是「在檔案撰寫命令而已」，所以輸入程式也不會顯示執行結果（圖 2.6）。

程式2.1 chap2/test2_1.py

```
001 print(100)
002 print(2+3)
003 print(2-3)
004 print(2*3)
005 print(2/3)
006 print(2%3)
007 print(2**3)
```

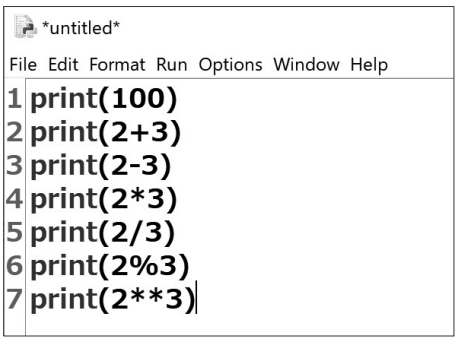

圖2.6 輸入的程式碼

❷ 輸入完畢之後，立刻儲存檔案。

從選單點選「File」（圖 2.7 ❶）→「Save」（圖 2.7 ❷），替檔案命名之後，點選「存檔」。比方說，將檔案儲存為「test2_1」（圖 2.8 ❶、❷）。如果作業系統是 Windows，可選擇儲存在「文件」；如果是 macOS，也可以選擇「文件」這類資料夾儲存檔案。

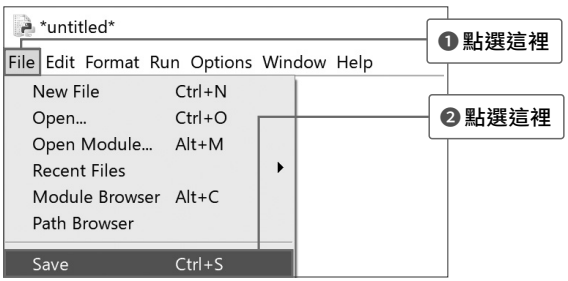

圖2.7 選單

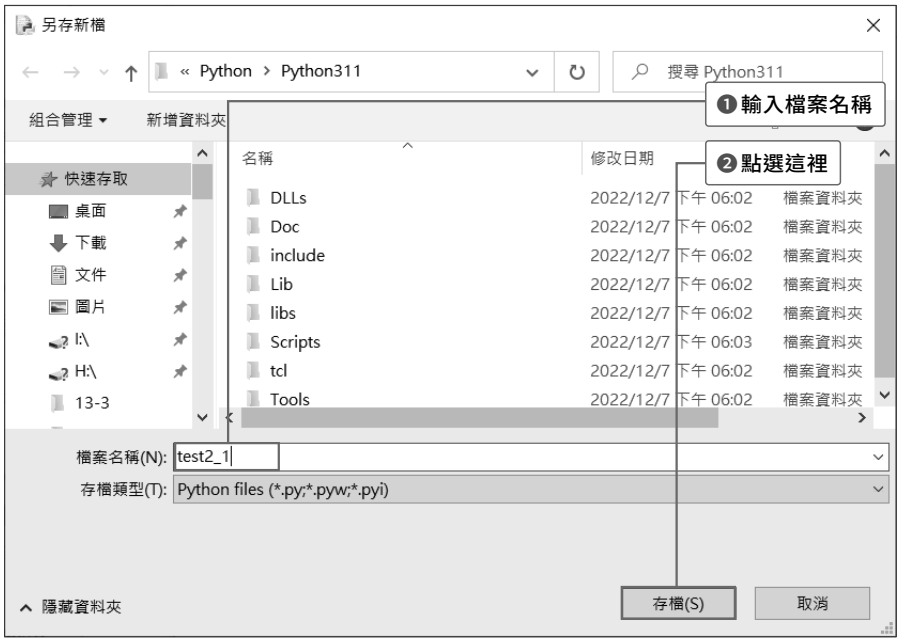

圖 2.8 另存新檔對話框

此外，Python 檔案的副檔名為「.py」，所以輸入「test2_1」就會自動以「test2_1.py」這種帶有副檔名的檔案名稱儲存。

❸ 執行這個程式

要執行程式可從選單點選「Run」（圖 2.9 ❶）→「Run Module」（圖 2.9 ❷），如此一來就會執行程式（圖 2.10）。

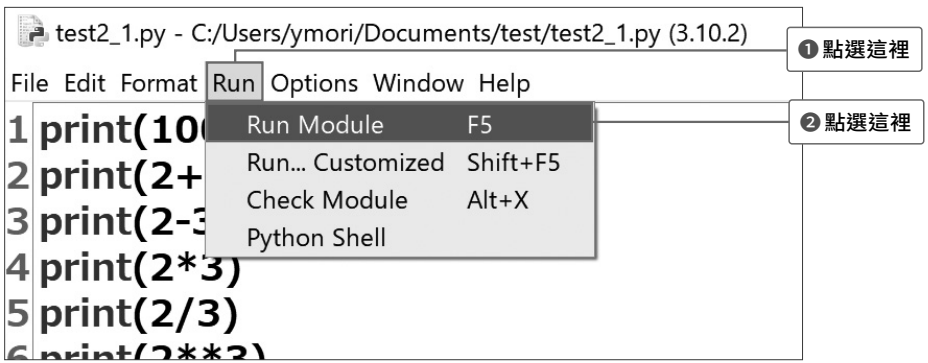

圖 2.9 執行程式

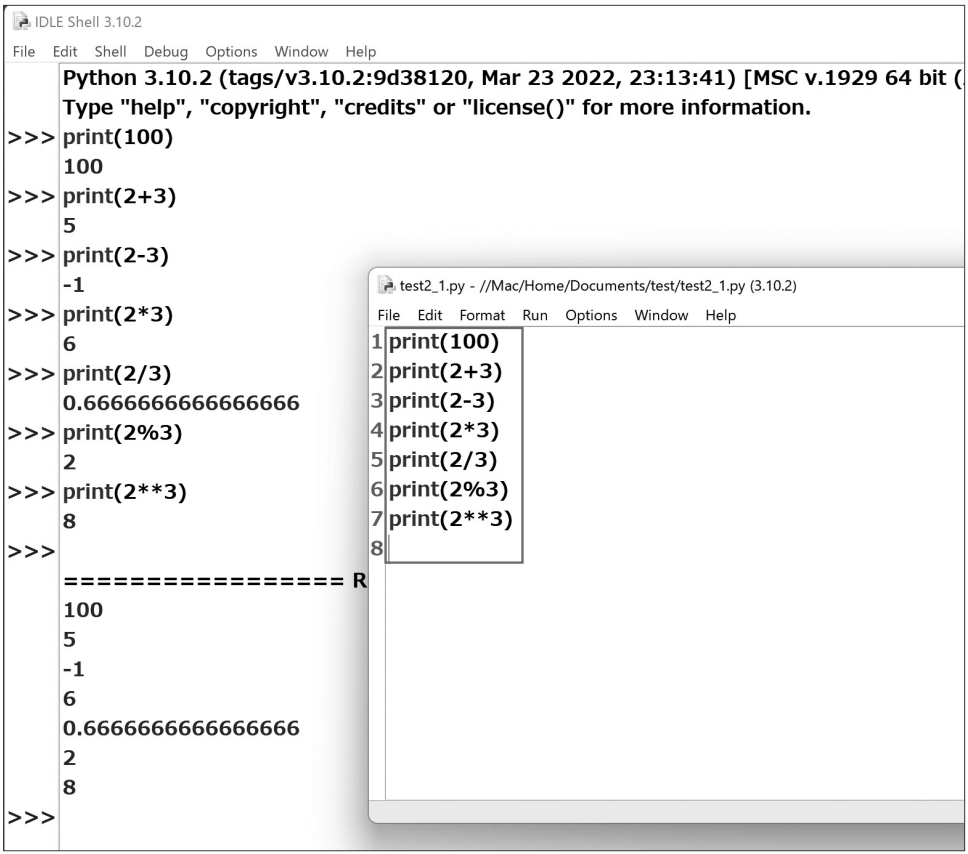

圖2.10 執行結果

執行結果會顯示於剛剛顯示執行結果的「Shell 視窗」，不會在撰寫程式的視窗顯示。

Recipe 2 Chapter 2

將資料放入變數再使用

在程式設計的世界裡，通常會將資料放進**「箱子」**再使用，而這個箱子就稱為**「變數」**。

要建立變數只需輸入**「變數名稱 = 值」**（語法 2.2）。這個語法的意思是**「請建立這個變數名稱的箱子，再將值放進這個箱子」**。

語法2.2 建立變數的方法

```
變數名稱 = 值
```

這個變數名稱可隨意自訂，但通常只會使用半形的英文字母。此外，「print」這種已經被當成 Python 命令使用的單字（保留字），不能用來替變數命名。

接著讓我們試著撰寫建立變數，計算資料的程式吧。從選單點選「File」→「New File」，新增檔案，再輸入程式 2.2 的程式碼。

程式2.2 chap2/test2_2.py

```
001 a = 2
002 b = 3
003 c = a + b
004 print(a, b, c)
```

將程式儲存為 test2_2.py 這個檔案之後，從選單點選「Run」→「Run Module」執行看看。

執行結果

```
2 3 5
```

這次將 2 放入變數 a，再將 3 放入變數 b，然後將 a+b 的計算結果放入變數 c，最後再依序顯示上述流程的結果。

Python 除了可以操作數值，還可以操作**各種資料**，而這就叫做**「資料類型」**，其中包含「整數類型、浮點數類型、字串類型、布林類型」等類型。

整數類型又稱為**「int」類型**，可用來計算東西的個數或是調查東西的順序。「int」就是 integer（整數）的縮寫。

浮點數類型（小數點）又稱為**「float（浮點）」**，通常用來代表現實世界的重量或是長度。所謂的「float」就是代表浮點數的 floating point number 簡稱。

字串類型又稱為**「str」**，可用來代表字串。「str」就是 string（字串）的簡稱。

布林類型又稱為**「bool」**，主要在電腦進行判斷的時候使用。正確的時候會是 True，錯誤的時候會是 False 這種值。

雖然資料的類型有很多，但 Python 都**以相同的語法操作**，所以建立變數的時候，不管是哪種類型都可利用**「變數名稱 = 值」**建立變數，因為 Python 會根據值的資料類型，自動準備**「適合存放這個資料的箱子」**。

資料類型不同的時候，其實得在電腦內部的記憶體進行不同的處理，但 Python 會幫我們省去這類麻煩。

讓我們試著建立各種資料類型的變數吧（程式 2.3）。

程式2.3 chap2/test2_3.py

```
001  a = 123
002  b = 123.4
```

```
003 c = "abc"
004 d = True
005 print(a, b, c, d)
```

執行之後，就會顯示各種資料。

執行結果

```
123 123.4 abc True
```


能以相同的方式操作各種資料類型是件很方便的事，但我們得提醒自己「現在操作的是哪種資料」。比方說，「123」與「"123"」看起來是一樣的資料，但資料類型卻不一樣。

比方說，在撰寫讓這個數字「乘以 2 倍」的程式時（程式 2.4），就會得到不同的結果。

程式2.4 chap2/test2_4.py

```
001 a = 123
002 b = "123"
003 print(a*2)
004 print(b*2)
```

執行結果

```
246
123123
```

執行之後，第 1 個結果會是「246」，第 2 個結果會是「123123」。

這就是資料類型不同的結果。整數類型的「123」*2，就是乘以 2 倍，會是「246」；但是字串類型的「"123"」*2，卻是「字串重覆」的「"123123"」。

所以在撰寫程式的時候，一定要記得「現在操作的資料」。比方說，**「從鍵盤輸入數值時」**或是**「從外部檔案載入數值時」**，Python 都會先以「字串類型」的方式載入數值，這是因為來自鍵盤或外部檔案的資料有可能會是「非數值的字串」。

記得「**從外部檔案輸入的資料**會先當成字串類型操作」這點非常重要。輸入的數值會先被當成「"123"」這種字串操作，所以無法進行計算。如果想要當成「123」計算，就得先**「轉換資料類型」**。這種**「轉換資料類型」**可利用簡單的命令完成，也就是**「變數 = 資料類型（值）」**的命令（語法 2.3）。

語法2.3　轉換資料類型

變數 = float（值）	轉換為浮點數類型
變數 = int（值）	轉換為整數類型
變數 = str（值）	轉換為字串類型
變數 = bool（值）	轉換為布林類型

換言之，只需撰寫**「想轉換的資料類型（值）」**，就能將資料轉換成需要的資料類型。讓我們試著利用這個命令修正程式 2.4 的第 2 行程式吧（程式 2.5）。

程式2.5　chap2/test2_5.py

```
001  a = 123
002  b = int("123")
003  print(a*2)
004  print(b*2)
```

執行結果

```
246
246
```

執行之後，就能正確計算字串類型的資料。這就是常在需要正確操作人類輸入的數值時使用的轉換處理。