第 2 章

※本教材使用 Python 3.10 版本，因 Python 會不定期更新，介面可能與本教材略有不同。

從 Scratch 到 Python

學習過七、八年級的 Scratch 課程之後，相信同學們對於程式設計已經有初步的認識。本章選用 Python 文字式程式語言，藉由學習過的 Scratch 範例，引入 Python 基本語法搭配例子介紹，期望同學能夠順利地將問題從積木式語言銜接到文字式語言。除了 Python 的基本語法外，只要配合不同的套件，Python 就能變身成為不同的工具，進而解決各式各樣的問題。最後應用所學過的語法，逐步完成一個完整專題。此外，因為 Python 在各種應用上都有龐大的社群，若是需要解決某個問題的方法，只要使用搜尋引擎尋找，也可以找到各種不同的解決方法，而這就是 Python 強大的主因之一。

學習完資料型態轉換的概念後，接下來我們用「計算學期成績」的程式為例，學習關係運算符號與選擇結構的概念吧！

範例：計算學期成績

請設計一個程式，讓使用者輸入各項成績後，再將各項成績轉換為學期成績，並判斷學期成績是否及格？（其中，作業成績占 40%，測驗成績占 40%，平時成績占 20%，學期成績 60 分為及格分數。）

中英對照

保留字		變數	
if	如果	grade	成績
else	否則		
elif（else if）	否則如果		
False	假		
True	真		

保留字是文字式程式語言中，已經具有特定意義的英文單字，不可用來作為使用者自訂的變數名稱。

Scratch 積木

Python 程式碼

```
1 x = int(input('請輸入作業成績：'))      # 將輸入的字串轉換為數字存到變數 x。
2 y = int(input('請輸入測驗成績：'))      # 將輸入的字串轉換為數字存到變數 y。
3 z = int(input('請輸入平時成績：'))      # 將輸入的字串轉換為數字存到變數 z。
4 grade = x * 0.4 + y * 0.4 + z * 0.2   # 計算學期成績存到變數 grade。
5 print('學期成績是', grade)              # 呈現學期成績。
6 if (grade < 60):                      # 使用雙向選擇結構判斷結果。
7     print('不及格')                   # 當 grade<60，呈現不及格。
8 else:
9     print('及格')                     # 當 grade>=60，呈現及格。
```

小提示

程式區塊的縮排

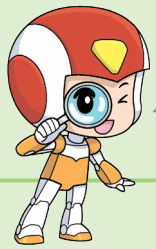
在 Python 中，自動縮排預設為四個空白字元，也可以使用一個 Tab 鍵，但須注意同一個程式只能使用同一種方式，不可同時使用四個空白字元和 Tab 鍵，否則執行會產生錯誤。

2-1 認識 Python 程式語言

▲圖 2-1 App Inventor 的介面。

本課程七、八年級都使用 Scratch 來設計程式，後續銜接的課程安排可以選擇 App Inventor 或 Python 為程式設計工具，而 Scratch 雖然簡單容易上手，但是程式設計的使用或應用上還是有些限制。同為麻省理工學院開發的 App Inventor（圖 2-1），雖然與 Scratch 皆為積木式程式設計軟體，卻具備物件導向程式設計的概念，可輕易、直覺的操作與設計，支援中文和各種手機的感測器，並提供多種網路元件，可提供網路的相關應用。學習者具備了物件導向程式設計概念後，未來轉換到文字式程式語言（例如：Python、C、Java），也可以順利無縫接軌。

本章選擇目前頗受重視的 Python，它是一種廣泛使用且功能強大的通用型程式語言，語句易讀、易懂，很適合做為第一個學習的文字式程式語言。在介紹 Python 的過程中，同時以過去學過的 Scratch 積木程式為例，對照 Python 的程式碼，讓學習者可以很快進入文字式程式語言的世界。



附錄第 260 頁有 App Inventor 的補充。

資料分析

科學運算

自 1991 年由吉多・范羅蘇姆（Guido van Rossum）研發並釋出後，陸續有許多程式設計者為 Python 提供了很多自行開發的模組，用來解決各式各樣的問題，因此學習 Python 不僅可以支援大部分的應用，也能從廣大的 Python 社群中獲得豐富的資源，其中常見的應用有資料分析、科學運算、網站開發、人工智慧、機器人控制等（圖 2-2）。

至於 Python 這個名稱的由來，在查閱過英文字典後，你會發現其中文翻譯為「蟒蛇」，然而事實上，它是取名於吉多・范羅蘇姆很喜歡的一部英國喜劇蒙提・派森的飛行馬戲團（Monty Python's Flying Circus）。

▼圖 2-2 Python 常見的應用。

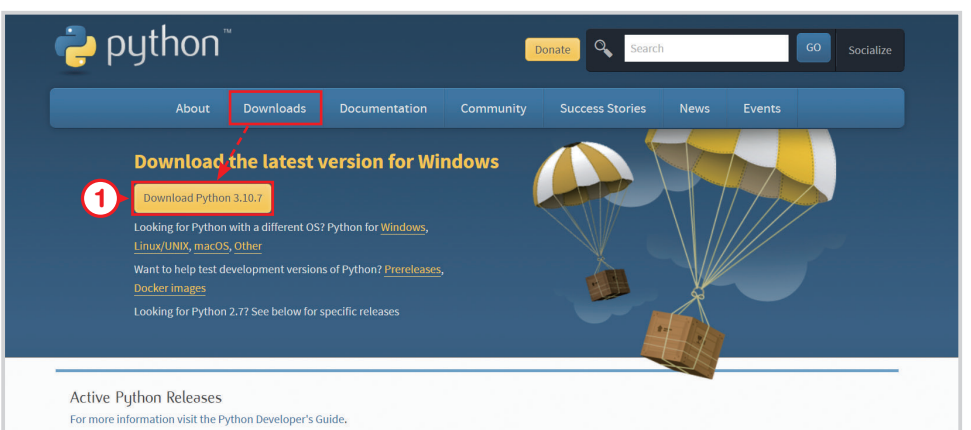
2-1-1 程式設計工具－離線版

如何開始編輯 Python 程式碼的第一步？首先，要選擇適合自己的程式設計工具，以下簡要介紹 Python 內建 IDLE 編輯器的安裝與使用介面。

① 下載與安裝

步驟 1 在 Python 網站（https://www.python.org），選取 Downloads。

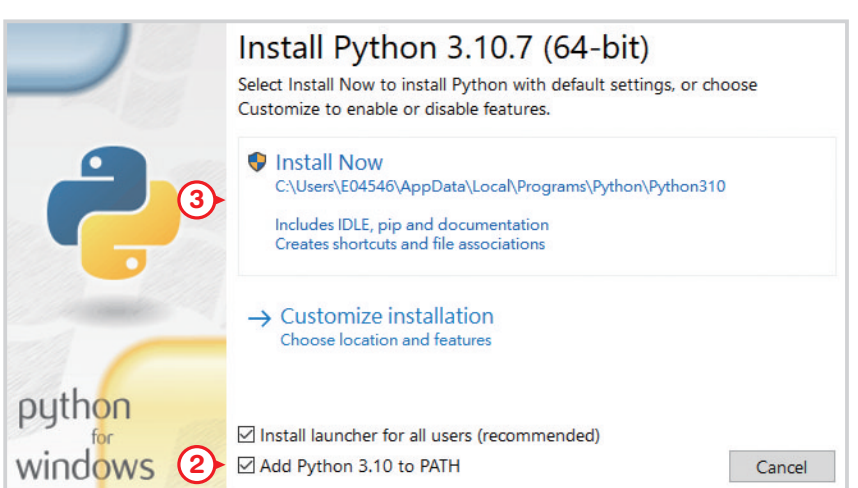
① 點選 **Download Python 3.10.7**，下載安裝檔。



步驟 2 開啟安裝檔，執行安裝介面。

② 勾選 **Add Python 3.10 to PATH**，讓安裝程式自動設定 PATH 環境變數，之後使用命令提示字元時，不管在哪一個資料夾底下都可以用 Python 指令執行程式。

③ 點選 **Install Now**，開始安裝。



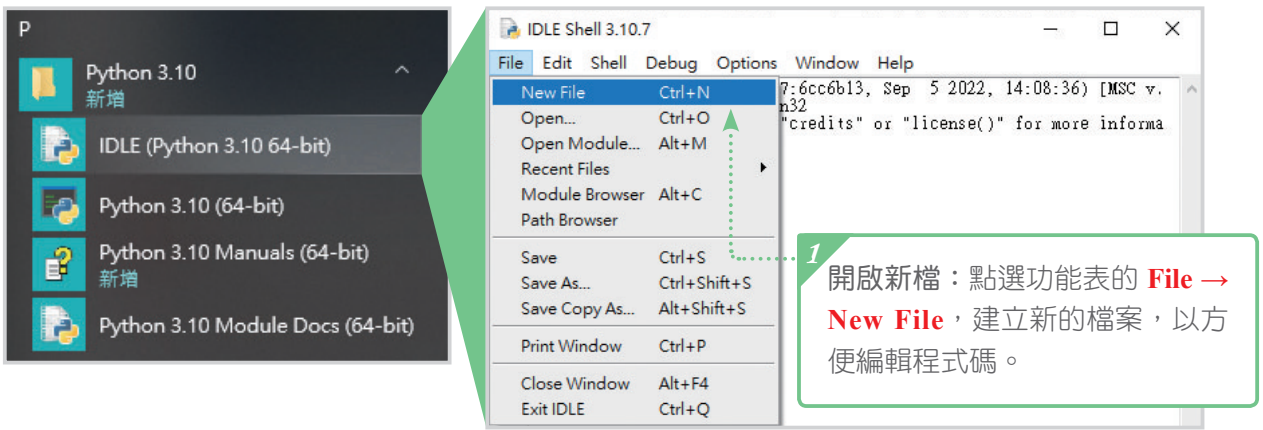
小知識

版本說明

官網上會持續推出新的 Python 版本，但因 Python 的套件很多，易與最新版本不相容，因此使用哪一種 Python 版本，應根據選用的套件是否支援決定。

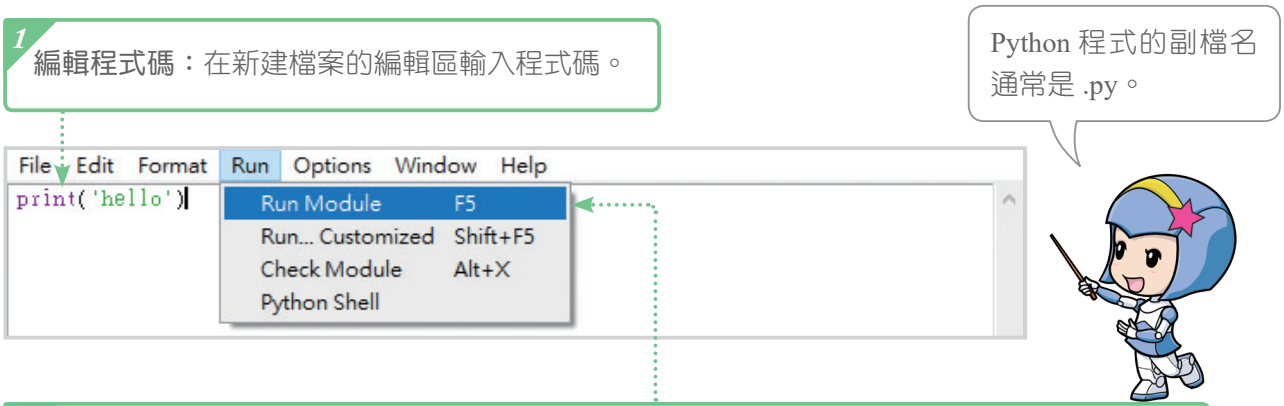
② 開啟 IDLE 編輯器

在開始功能表中，點選 Python 3.10 資料夾的 IDLE（Python 3.10 64-bit），開啟編輯器。



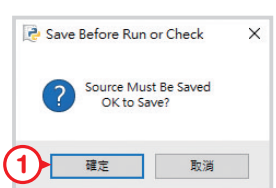
③ 編輯與執行程式碼

1 編輯程式碼：在新建檔案的編輯區輸入程式碼。

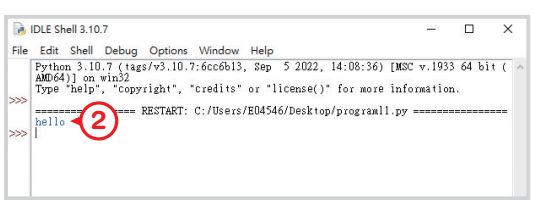


2 執行程式碼：點選功能表的 **Run → Run Module**，執行編輯區的程式碼，並顯示執行結果。

① 跳出視窗導引使用者儲存檔案，按下**確定**後，再命名檔案名稱與選擇儲存位置。



② 在原視窗的交談式介面裡，顯示程式執行結果。



2-1-2 程式設計工具－線上版

目前有許多 Python 程式設計工具，其中 Google 提供 Colaboratory（簡稱 Colab）平臺可以編輯、測試與儲存至雲端硬碟，以下簡要介紹 Colab 的介面與功能。（Colab 網址：https://colab.research.google.com）

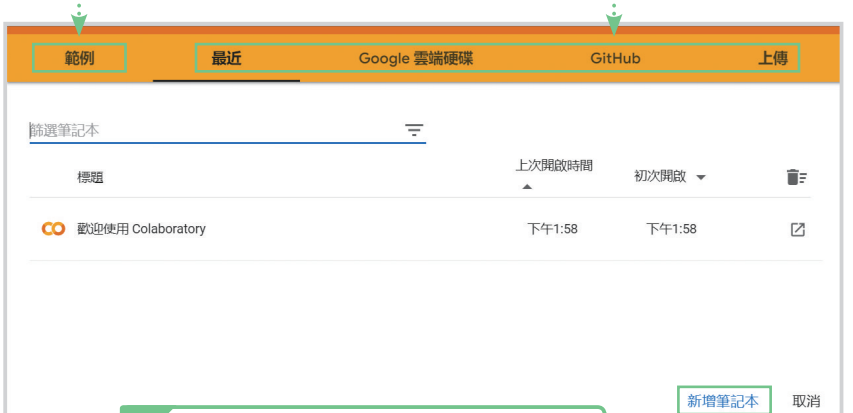
① 登入帳號與開啟筆記本

步驟 1 點選登入，再輸入 Google 帳號與密碼。

步驟 2 在自動彈出視窗中，點選欲開啟的筆記本。

1 開啟範例筆記本：使用**範例**中的筆記本。

2 開啟既有筆記本：使用**最近**、**Google 雲端硬碟**、**GitHub** 和**上傳**的筆記本。

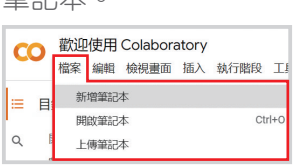


3 新增筆記本：新增空白的筆記本。

小知識

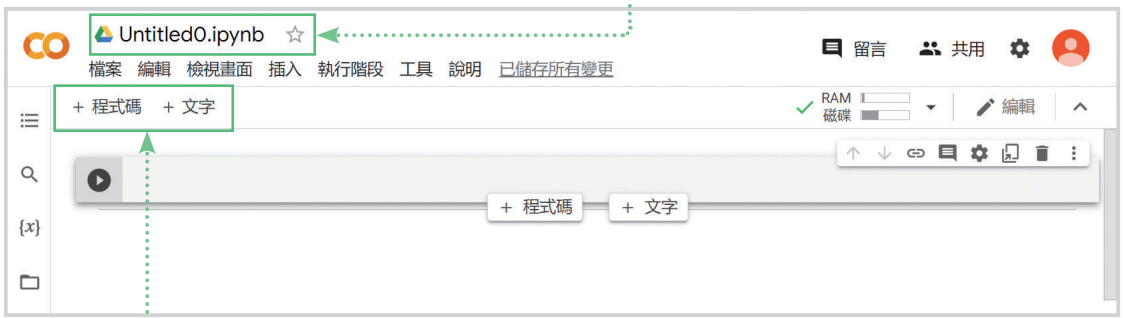
使用 Colab 筆記本的其他方式

在 Colab 首頁點選**檔案**，可新增筆記本或開啟既有筆記本。



② 編輯筆記本

1 命名筆記本：直接輸入欲命名的文字。



2 新增程式碼或文字區塊：點選功能表欲增加的儲存格類型。

(1)程式碼

(2)文字

③ 共用筆記本

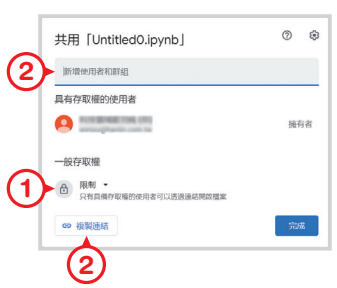
1 共用筆記本：點選功能表右上方的 共用，即可與他人共用此份筆記本。

① 設定存取權，選取**知道連結的任何人**及其**角色**（檢視者、加註者、編輯者）。

② 選擇共用的方式，有兩種如下：

(1)輸入欲共用檔案對象的 email。

(2)點選 複製連結 後，再傳給欲共用的對象。



2-2 Python 程式設計的概念

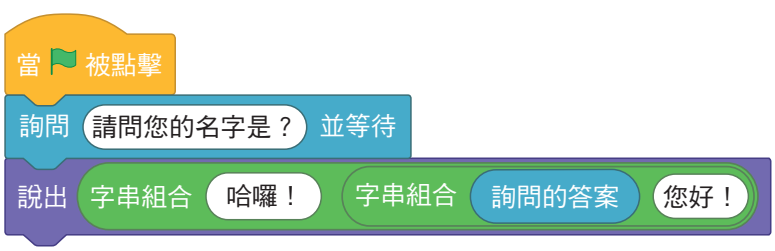
2-2-1 Python 中的基本語法

認識 Python 程式設計工具後，讓我們用「哈囉」的程式為例，認識輸入函式 input( ) 與輸出函式 print( ) 的概念，真正地進入文字式程式語言的世界吧！

範例：哈囉

請設計一個程式，讓使用者輸入名字後，電腦會將名字呈現在畫面上與使用者打招呼。

Scratch 積木



概念 input( ) 函式

input( ) 函式是讓使用者由鍵盤輸入資料，通常會用變數儲存該資料，語法如下：

```
變數 = input('提示字串')
```

程式中會出現各種符號，以下將分別說明各種符號的作用：

1. 等號 = 的作用：將等號右邊的數值儲存到等號左邊的變數。
2. 小括號 ( ) 的作用：放入想要儲存的內容。
3. 單引號 ' ' 的作用：' ' 中的內容會被程式視為字串，通常會在字串前後成對出現，字串的表示也可用雙引號 " "。

例如

《程式碼》

```
x = input('請輸入一個數字：')      # 字串後方可輸入資料。
```

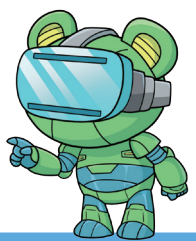
《執行結果》

```
請輸入一個數字：
```

中英對照

函式名稱	
input	輸入
print	輸出

變數	
name	名字



號後面的文字代表程式註解，Python 會忽略它不去解譯，程式註解通常是使用自然語言來描述。

Python 程式碼

```
1 name = input('請問您的名字是？')    # 將輸入的名字存到變數 name。
2 print('哈囉！', name, '您好！')     # 呈現打招呼與名字。
```

概念 print( ) 函式

print( ) 函式是將資料進行輸出，可以一次輸出多個項目，項目之間以「,」隔開，語法如下：

```
print(項目 1, 項目 2, ...)
```

例如

《程式碼》

```
name = '阿顯'
print('顯示一段文字')      # 輸出' '中的內容。
print('顯示', name)        # 輸出' '中和變數 name 的內容。
```

《執行結果》

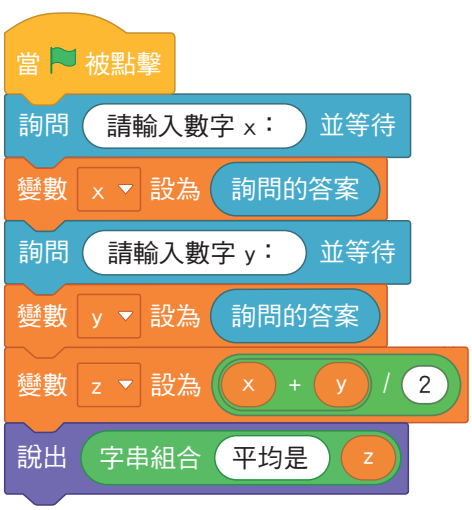
```
顯示一段文字
顯示 阿顯
```

學習完 input( ) 與 print( ) 函式後，接下來我們用「求平均數」的程式為例，學習變數與資料型態、資料型態轉換與算術運算符號的概念吧！

範例：求平均數

請設計一個程式，讓使用者輸入兩個數字後，再呈現兩個數字的平均值。

Scratch 積木



概念 變數與資料型態

在程式中使用變數來儲存資料，語法如下：

```
變數名稱 = 變數值
```

Python 會根據變數值設定適合的資料型態（如右表），對於不同性質的資料，須決定使用哪種資料型態來儲存，這也會影響到電腦要配置多大的記憶體空間給這個變數使用。

中英對照

函式名稱	
int（integer）	整數
float	浮點數
bool（boolean）	布林值
str（string）	字串

變數	
score	成績
average	平均值
flag	旗子

Python 程式碼

```
1 x = int(input('請輸入數字 x：'))   # 將輸入的字串轉換為數字存到變數 x。
2 y = int(input('請輸入數字 y：'))   # 將輸入的字串轉換為數字存到變數 y。
3 z = (x + y) / 2                   # 計算平均數存到變數 z。
4 print('平均是', z)                 # 呈現平均數。
```

使用 input( ) 函式輸入數字，會以字串的資料型態儲存，因此必須先用 int( ) 函式強制轉換為整數，才能使用加號進行運算，否則就會把輸入的數字視為字串相加，而不是數值相加。

Python 對應的資料型態

變數資料型態	變數值	範例
整數（int）	80、63、18…	score = 80（自動設定變數為整數。）
浮點數（float）	80.8、12.9、3.14…	average = 80.8（自動設定變數為浮點數。）
布林值（bool）	True、False	flag = True （自動設定變數為布林值。）
字串（str）	John、你好、Apple…	name = 'John'（自動設定變數為字串。）

概念 資料型態轉換

資料型態相同的變數才能進行運算。對於某些資料型態，Python 能夠進行簡單的自動轉換。如果是整數與浮點數運算，系統會先將整數轉換成浮點數再運算，而最後的運算結果仍為浮點數。

例如

《程式碼》

```
a = 8 + 6.3                    # 整數＋浮點數。
print(a)
```

《執行結果》

```
14.3
```

如果系統無法自動進行資料型態轉換，就需使用下列的強制資料轉換函式。

函式	使用時機
int( )	強制轉換為整數。
float( )	強制轉換為浮點數。
bool( )	強制轉換為布林值。
str( )	強制轉換為字串。

舉例來說，如果對整數與字串做加法運算，會產生錯誤，另提供其他數值資料型態的運算例子。

例如

《程式碼》

```
b = 56 + '23'                  # 整數＋字串。
print(b)
```

《執行結果》

```
資料型態錯誤，無法運算。
```

《程式碼》

```
c = 56 + int('23')             # 整數＋整數。
print(c)
```

《執行結果》

```
79
```

《程式碼》

```
d = 56 + float('23.6')         # 整數＋浮點數。
print(d)
```

《執行結果》

```
79.6
```

《程式碼》

```
e = 56 + str(23.6)             # 整數＋字串。
print(e)
```

《執行結果》

```
資料型態錯誤，無法運算。
```

加號不僅可用於數值運算，也可用於字串組合，但同學們必須謹記「只有資料型態相同的變數才能進行運算」。因此，使用時需注意其所應用的資料型態。

例如

《程式碼》	《執行結果》
`f = 'hello' + 34` `# 字串＋整數。` `print(f)`	資料型態錯誤，無法運算。
`g = 'hello' + '34'` `# 字串＋字串。` `print(g)`	`hello34`
`h = 'hello' + str(34)` `# 字串＋字串。` `print(h)`	`hello34`

概念 算術運算符號

Python 中，除了相加之外，還有其他可使用的運算符號如下表所示。

算術運算符號	意義	範例	運算結果
+	加法	8 + 5	13
-	減法	8 - 5	3
*	乘法	8 * 5	40
/	除法	8 / 5	1.6
%	除法取餘數	8 % 5	3
//	除法取商數	8 // 5	1
**	次方	8 ** 2	64

例如

《程式碼》	《執行結果》
`print(15 % 7)` `# 15÷7=2…1。`	1
`print(30 // 8)` `# 30÷8=3…6。`	3

學習完資料型態轉換的概念後，接下來我們用「計算學期成績」的程式為例，學習關係運算符號與選擇結構的概念吧！

範例：計算學期成績

請設計一個程式，讓使用者輸入各項成績後，再將各項成績轉換為學期成績，並判斷學期成績是否及格？（其中，作業成績占 40%，測驗成績占 40%，平時成績占 20%，學期成績 60 分為及格分數。）

Scratch 積木

中英對照

保留字	
if	如果
else	否則
elif（else if）	否則如果
False	假
True	真

變數	
grade	成績

Python 程式碼

```
1 x = int(input('請輸入作業成績：'))      # 將輸入的字串轉換為數字存到變數 x。
2 y = int(input('請輸入測驗成績：'))      # 將輸入的字串轉換為數字存到變數 y。
3 z = int(input('請輸入平時成績：'))      # 將輸入的字串轉換為數字存到變數 z。
4 grade = x * 0.4 + y * 0.4 + z * 0.2   # 計算學期成績存到變數 grade。
5 print('學期成績是', grade)              # 呈現學期成績。
6 if (grade < 60):                      # 使用雙向選擇結構判斷結果。
7     print('不及格')                   # 當 grade<60，呈現不及格。
8 else:
9     print('及格')                     # 當 grade>=60，呈現及格。
```

小提示

程式區塊的縮排

在 Python 中，自動縮排預設為四個空白字元，也可以使用一個 Tab 鍵，但須注意同一個程式只能使用同一種方式，不可同時使用四個空白字元和 Tab 鍵，否則執行會產生錯誤。

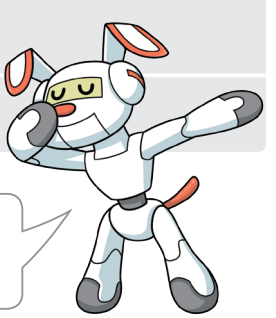
概念 關係運算符號

Python 中，可使用的關係運算符號如下表所示。

關係運算符號	意義	範例	運算結果
==	相等	8 == 5	False
		8 == 8	True
!=	不相等	8 != 5	True
		8 != 8	False
>	大於	8 > 5	True
		2 > 6	False
<	小於	8 < 5	False
		2 < 6	True
>=	大於或等於	8 >= 5	True
		2 >= 6	False
<=	小於或等於	8 <= 5	False
		2 <= 5	True

例如

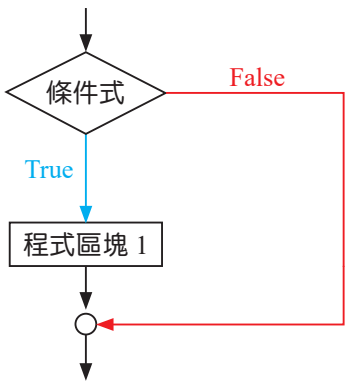
《程式碼》		《執行結果》
`print(3 + 6 == 2 + 5)`	`# 9==7，結果不成立。`	`False`
`print(7 - 2 != 5 + 4)`	`# 5!=9，結果成立。`	`True`
`print(4 + 5 > 3 + 7)`	`# 9>10，結果不成立。`	`False`
`print(50 < 100)`	`# 50<100，結果成立。`	`True`
`print(10 >= 30)`	`# 10>=30，結果不成立。`	`False`
`print(1 + 3 <= 6 - 2)`	`# 4<=4，結果成立。`	`True`



概念 單向選擇結構

在日常生活中，我們經常會遇到要作決策的情況，例如：如果下雨了，就帶把傘出門。程式的執行也是類似的情況，根據條件式運算結果，執行不同的程式區塊。最簡單流程控制是單向選擇結構，語法如下：

```
if 條件式:
    程式區塊
```



if 敘述後面接著條件式，條件式的運算結果是布林值，當條件式為 True 時，就會執行程式區塊的敘述；當條件式為 False 時，不會執行程式區塊。在條件式之後必須有冒號，底下的程式區塊必須縮排。

例如

《程式碼》

```
a = 10
b = 8
if (a > b):
    print('變數 a 比較大')  # 10>8，結果成立。
```

《執行結果》

```
變數 a 比較大
```

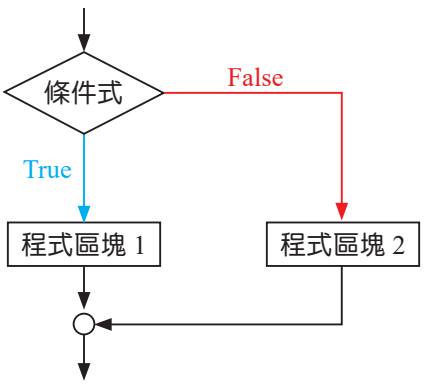
```
c = int(input('請評分餐點 1～5 分：'))
if (c >= 3):
    print('滿意')          # 假設 1 是 4>=3，結果成立。
                           # 假設 2 是 3>=3，結果成立。
                           # 假設 3 是 1>=3，結果不成立。
```

```
假設 1
請評分餐點 1～5 分：4
滿意

假設 2
請評分餐點 1～5 分：3
滿意

假設 3
請評分餐點 1～5 分：1
```

概念 雙向選擇結構

如果我們希望遇到情況為 True 和 False 時，分別做不同的事，就需要使用雙向選擇結構，語法如下：

```
if 條件式:
    程式區塊 1
else:
    程式區塊 2
```



當條件式為 True 時，就執行程式區塊 1，否則就執行 else 裡面的程式區塊 2。

例如

《程式碼》

```
d = int(input('請輸入一個數字：'))
if (d % 2 == 0):
    print('偶數')      # 假設 1 是 88%2==0，結果成立。
else:
    print('奇數')      # 假設 2 是 55%2==0，結果不成立。
```

《執行結果》

```
假設 1
請輸入一個數字：88
偶數

假設 2
請輸入一個數字：55
奇數
```

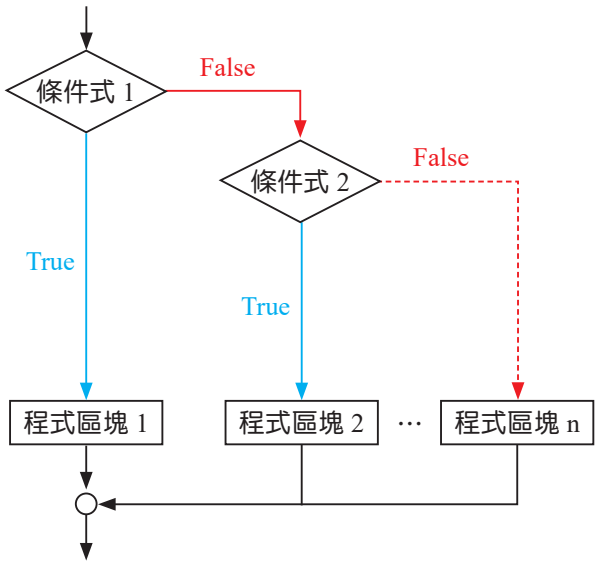
```
e = int(input('請評分餐點 1～5 分：'))
if (e >= 3):
    print('滿意')       # 假設 1 是 5>=3，結果成立。
                        # 假設 2 是 3>=3，結果成立。
else:
    print('不滿意')     # 假設 3 是 2>=3，結果不成立。
```

```
假設 1
請評分餐點 1～5 分：5
滿意

假設 2
請評分餐點 1～5 分：3
滿意

假設 3
請評分餐點 1～5 分：2
不滿意
```

概念 多向選擇結構

如果發生的情況不只兩種，就可以透過使用 elif 做更進一步的選擇，語法如下：

```
if 條件式 1:
    程式區塊 1
elif 條件式 2:
    程式區塊 2
elif 條件式 3:
        ⋮
else:
    程式區塊 n
```



當其中一個條件式為 True 時，就執行相對應的程式區塊，並且當所有條件式皆為 False 時，就執行 else 裡面的程式區塊。

例如

《程式碼》

```
f = int(input('請評分餐點 1～5 分：'))
if (f == 5):
    print('非常滿意')    # 假設 1 是 5==5，結果成立。
elif (f >= 3):
    print('滿意')        # 假設 2 是 3>=3，結果成立。
else:
    print('不滿意')      # 假設 3 是 1>=3，結果不成立。
```

《執行結果》

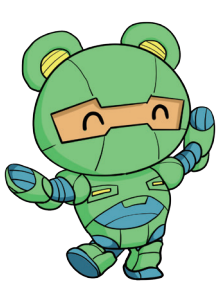
```
假設 1
請評分餐點 1～5 分：5
非常滿意

假設 2
請評分餐點 1～5 分：3
滿意

假設 3
請評分餐點 1～5 分：1
不滿意
```

學習完關係運算符號與選擇結構的概念後，接下來我們用「累加計算」的程式為例，學習串列、range( ) 函式與 for 迴圈的概念吧！

範例：累加計算

請設計一個程式，讓使用者輸入數字 n 後，再計算 1 ＋ 2 ＋ 3 ＋…＋ n 的值。

Scratch 積木



附錄還有邏輯運算和 while 迴圈的補充。

中英對照

變數 / 串列名稱	
sum	總和
list	串列

保留字	
for	對於
in	在…之內

函式名稱	
range	範圍

Python 程式碼

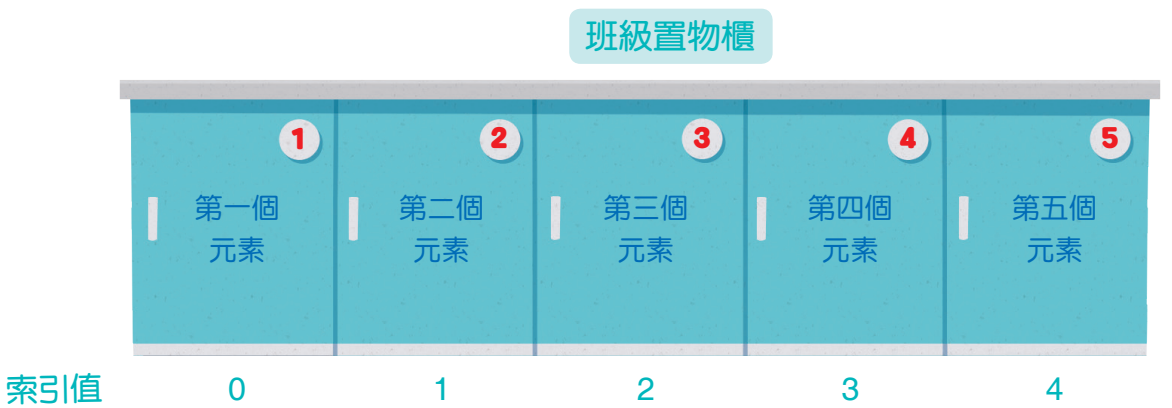
```
1 sum = 0                                    # 將變數 sum 初始值設為 0。
2 n = int(input('請輸入數字 n：'))             # 將輸入的字串轉換為數字存到變數 n。
3 for i in range(1, n + 1):                  # 執行計次式迴圈，累加變數 i 到總和。
4     sum = sum + i
5 print('1 + 2 +...+', n, '=', sum)          # 呈現總和。
```

概念 串列

程式中的資料通常使用變數來儲存，但是對於大量的資料則可以用串列（list）簡化變數的需求量。每一個串列擁有一個識別名稱，串列中的每一個資料稱為元素，每一個元素相當於一個變數，要存取串列中特定元素，是以元素在串列中的位置做為索引值，語法如下：

```
串列名稱 = [元素 1, 元素 2, ...]
```

串列中的元素資料型態可以相同，也可以不相同。要注意的是 Python 的串列索引值從 0 開始，第一個元素的索引值是 0，第二個元素的索引值是 1，第三個元素的索引值是 2，以此類推。此外，索引值也可以是負數，代表從串列最後算起第幾個元素，例如：最後一個元素的索引值即為－1。



例如

《程式碼》

```
list1 = [1, 2, 3, 4, 5]              # 元素皆為整數。
list2 = ['John', 'Marry', 'Tom']     # 元素皆為字串。
list3 = ['Hellen', 18, True]         # 元素包含不同資料型態。
print(list1[0])
print(list2[1])
print(list3[-1])
```

《執行結果》

```
1
Marry
True
```

概念 range( ) 函式

計次式迴圈中經常使用 range( ) 函式來建立整數循序串列，語法如下：

```
range(起始值, 終止值, 累加值)
```

產生的串列是由起始值開始（預設值為 0），每次遞增累加值（預設值為 1），直到終止值的前一項。舉例來說，range(1 , 4) 會產生一個從 1 開始，小於 4（不包含 4）的整數串列，因此回傳的串列是 [1 , 2 , 3]，而若要產生從 1 開始，4 結束的整數串列，則要寫成 range(1 , 5)。

例如

《程式碼》

```
list1 = range(6)          # list1 的內容是 [0, 1, 2, 3, 4, 5]。
list2 = range(1, 6)       # list2 的內容是 [1, 2, 3, 4, 5]。
list3 = range(1, 6, 2)    # list3 的內容是 [1, 3, 5]。
list4 = range(6, 1, -1)   # list4 的內容是 [6, 5, 4, 3, 2]。
```

概念 for 迴圈

for 迴圈是一種處理重複步驟的語法，它會從一個串列中逐一取出元素，然後指定給迴圈變數，因為串列中的元素個數，即是重複迴圈中程式碼執行的次數，所以稱為計次式迴圈。

for 和 in 之間的迴圈變數可以自己定義變數名稱，而 in 後面接著一個序列，在這個例子中，我們使用了串列，它是序列的其中一種。for 迴圈會依序從序列中取得元素，並且將元素指定給前面自訂的迴圈變數，再執行迴圈裡的內容，直到序列每一個元素都被取過為止，語法如下：

```
for 迴圈變數 in 串列：
    程式區塊
```

舉例來說，有一個數字串列 [1 , 2 , 3 , 4 , 5]，將此串列放在 in 的後方，並使用 i 來做為迴圈的變數。在程式區塊部分，為了讓 Python 了解這個區塊是屬於迴圈的內容，所以要向內縮排。程式區塊內可以寫入我們希望程式完成的函式，例如：若想輸出每一個迴圈中的迴圈變數值，就可以寫成 print(i)。

例如

《程式碼》

```
for i in [1, 2, 3, 4]:
    print(i)
```

《執行結果》

```
1
2
3
4
```

另外，因為這個串列是有規則的數列，所以可以用 range( ) 函式來簡化程式碼。

例如

《程式碼》

```
for i in range(1, 5):
    print(i)
```

《執行結果》

```
1
2
3
4
```

練習

程式執行後，讓使用者輸入 n 後，再計算出 1 × 2 × … × n 的值。請參考右方的 Scratch 積木，依照程式流程，完成 Python 程式碼。

2-2-2 Python 中的繪圖模組

在七年級的 Scratch 單元中，我們曾經使用畫筆積木來繪製有規律的圖案，同樣地，這些圖案也可以用 Python 的 turtle 繪圖模組來實作。這個模組之所以命名為 Turtle，概念是源自於 Seymour Papert 和 Wally Feurzig 在 1966 年所創造，用來引導孩子們學習程式設計的海龜繪圖 Logo 程式語言。

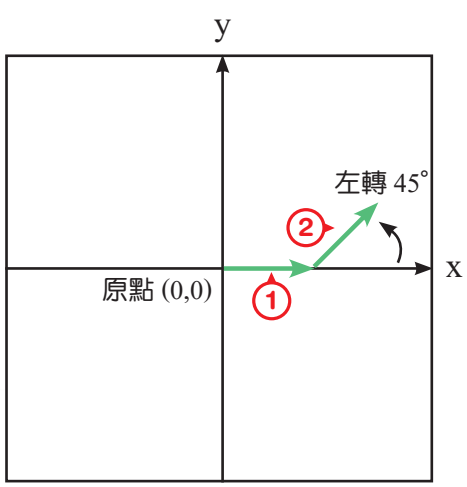
一個 Python 模組是一個檔案，內含 Python 程式指令，可以提供其他 Python 程式匯入使用。只要匯入模組，就可以使用模組裡事先定義的函式或變數。例如：匯入 turtle 模組後，就可以使用 Screen( ) 函式來產生畫布，或是使用 Turtle( ) 函式產生一個海龜進行繪圖，這兩個函式都是在 turtle 模組裡定義的，因此讓我們從 turtle 的繪圖坐標開始學習使用 Python 繪圖的做法吧！

turtle 繪圖坐標

turtle 的繪圖坐標是不是跟 Scratch 相似呢？讓我們來複習一下方向。

① 向前畫出一條線時，從原點開始繪製線段。

② 朝左邊 45 度畫出一條線時，會先調整向左轉 45 度後，再向前繪製線段。



小提示

Colab 繪圖模組

本章程式碼皆以 IDLE 進行編輯，如果使用 Colab 的繪圖模組，在程式語法中，會與 IDLE 有所差別。

① 安裝 Colab 的繪圖模組套件，執行：

```
!pip3 install ColabTurtle
```

② 接著引用 turtle 模組：

```
import ColabTurtle.Turtle as 變數
```

③ 產生畫布：

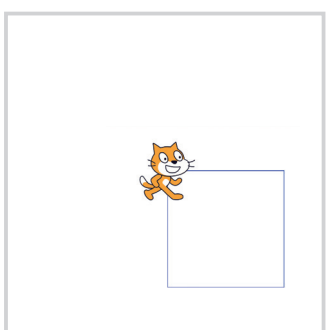
```
變數.initializeTurtle()
```

七年級的課程中，我們曾經使用 Scratch 畫出一個正方形。現在讓我們用「畫正方形」的程式為例，說明如何用 Python 畫出正方形的圖案吧！

範例：畫正方形

請設計一個程式，讓使用者畫出一個邊長為 150 個單位的正方形。

Scratch 積木

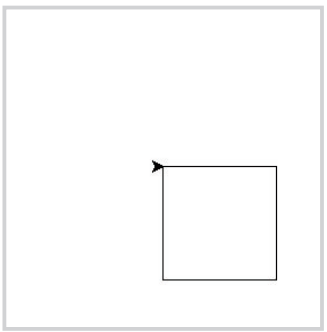


中英對照

函式名稱	
turtle	海龜
screen	視窗
shape	形狀

函式名稱	
forward	往前
right	右邊
exitonclick	點擊關閉

Python 程式碼



```
1  import turtle                    # 匯入 turtle 模組，以使用 Turtle 函式。
2  window = turtle.Screen()         # 產生畫布以進行畫圖。
3  john = turtle.Turtle()           # 建立一個海龜 Turtle，命名為變數 john。
4  john.forward(150)                # 往前走 150 個單位。
5  john.right(90)                   # 右轉 90 度。
6  john.forward(150)                # 往前走 150 個單位。
7  john.right(90)                   # 右轉 90 度。
8  john.forward(150)                # 往前走 150 個單位。
9  john.right(90)                   # 右轉 90 度。
10 john.forward(150)                # 往前走 150 個單位。
11 john.right(90)                   # 右轉 90 度。
12 window.exitonclick()             # 等待使用者關閉視窗。
```

概念 turtle.Turtle( ) 及 turtle.Screen( ) 函式

Python 是一個物件導向的程式語言，在使用物件的屬性或方法時，會使用「.」的語法，解釋為「的」，而屬性與方法的差別，在於呼叫方法時會加上 ( )，有時候 ( ) 中還會帶有參數，一同傳入給方法進行運算。

舉例來說，若要使用 turtle 模組裡的 Turtle( ) 方法，函式可以寫成 turtle.Turtle( )，解釋為「在 turtle 模組中，取用名為 Turtle 的方法」。這個方法會傳回一個用來畫圖的物件（就像是畫筆），它的名稱是海龜，然後讓 john 這個人具有海龜畫圖的能力（名稱為 john 的變數），也可以修改變數 john 為 tom 或 mary，只要符合變數要求即可。

接著，執行 window = turtle.Screen( ) 這行程式碼後，window 這個變數就會儲存一個畫布物件，並且遵循 turtle 模組事先定義的屬性與方法，語法如下：

例如

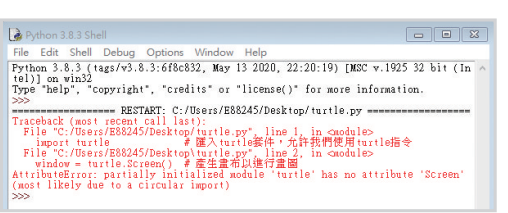
《程式碼》

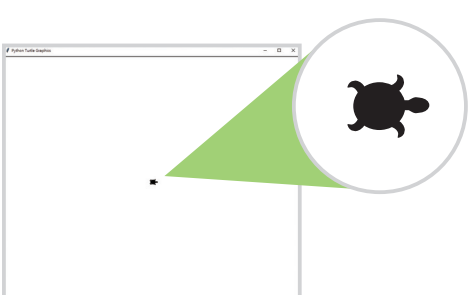
```
import turtle                    # 匯入 turtle 模組，以使用 Turtle 函式。
window = turtle.Screen()         # 產生畫布以進行畫圖。
john = turtle.Turtle()           # 建立一個海龜 Turtle，命名為變數 john。
```

小提示

不可使用的檔名

檔案名稱不可以是 turtle.py，否則會和內建的 turtle 模組名稱互相衝突，並出現右方的錯誤提醒。





設定海龜外型

在繪圖時使用的形狀，可以設定成海龜（turtle）、圓形（circle）、方形（square）和箭頭（arrow）造型，如延續上面程式碼，設定為海龜造型，語法如下：

```
john.shape('turtle')
```

概念 forward( ) 及 right( ) 函式

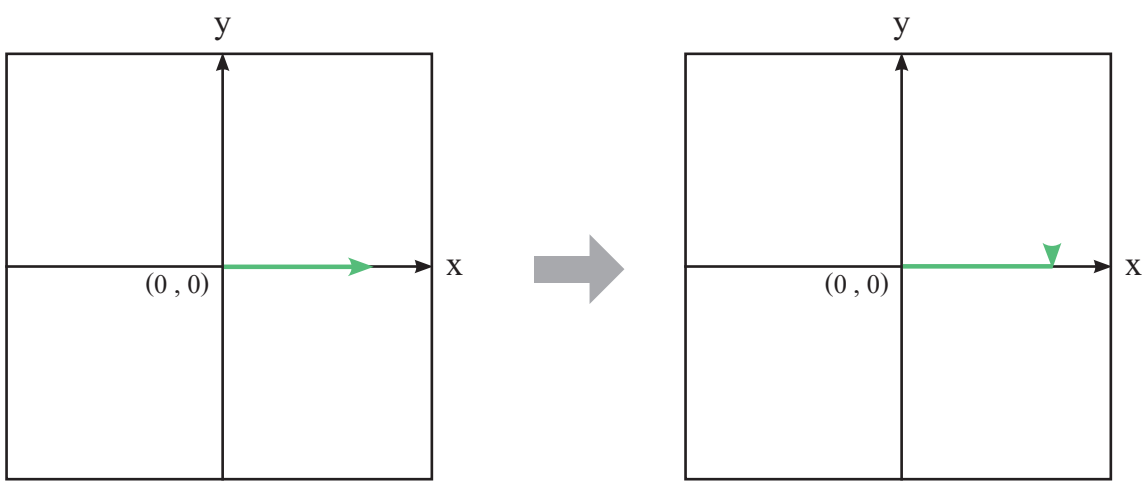
當海龜物件命名為 john，執行 john.forward(100) 函式時，將朝向面對的方向（預設為 x 軸正向）前進 100 點，並隨著移動畫出一條線段，再執行 john.right(90) 函式，就會在原地右轉 90 度，語法如下：

例如

《程式碼》

```
import turtle                    # 匯入 turtle 模組，以使用 Turtle 函式。
window = turtle.Screen()         # 產生畫布以進行畫圖。
john = turtle.Turtle()           # 建立一個海龜 Turtle，命名為變數 john。
john.forward(100)                # 往前走 100 個單位。
john.right(90)                   # 原地右轉 90 度。
```

往前走 100 個單位　　原地右轉 90 度



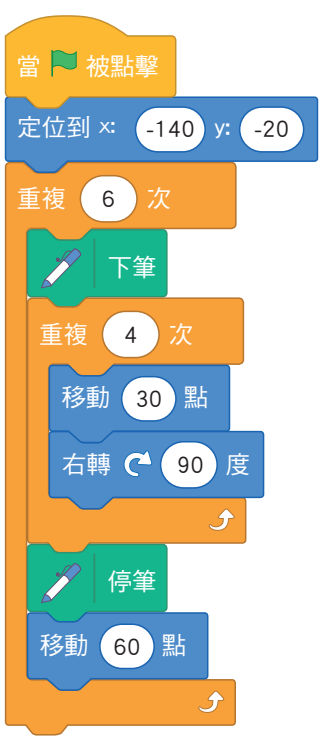
因此，只要配合 forward( ) 和 right( ) 函式，我們就可以使用 Python，畫出一個正方形的圖案。

學習完匯入 turtle 模組、產生畫布、前進與轉彎的概念後，接下來我們用「畫平行排列的正方形」的程式為例，學習畫布大小、提筆、定位與 for 迴圈的概念吧！

範例：畫平行排列的正方形

請設計一個程式，讓使用者畫出六個平行排列的正方形。

Scratch 積木



中英對照

函式名稱	
setup	建立
penup	提筆
goto	前往
pendown	下筆

Python 程式碼

```
1  import turtle                    # 匯入 turtle 模組，以使用 Turtle 函式。
2  window = turtle.Screen()         # 產生畫布以進行畫圖。
3  window.setup(480, 360)           # 設定畫布大小為寬 480 點，高 360 點。
4  john = turtle.Turtle()           # 建立一個海龜 Turtle，命名為變數 john。
5  john.penup()                     # 提筆。
6  john.goto(-140, -20)             # 定位到 (-140, -20) 的位置。
7  for i in range(6):               # 執行計次式迴圈，重複 6 次。
8      john.pendown()               # 下筆。
9      for j in range(4):           # 執行計次式迴圈，重複 4 次。
10         john.forward(30)         # 往前走 30 個單位。
11         john.right(90)           # 右轉 90 度。
12     john.penup()                 # 提筆。
13     john.forward(60)             # 往前走 60 個單位。
14 window.exitonclick()             # 等待使用者關閉視窗。
```

概念 window.setup( ) 函式

呼叫設定畫布大小的函式，寫成 window.setup(480 , 360)，解釋為「將畫布的大小設定為寬 480 點，高 360 點」。

例如

《程式碼》

```
import turtle                    # 匯入 turtle 模組，以使用 Turtle 函式。
window = turtle.Screen()         # 產生畫布以進行畫圖。
window.setup(480, 360)           # 設定畫布大小為寬 480 點，高 360 點。
```

概念 goto( ) 函式

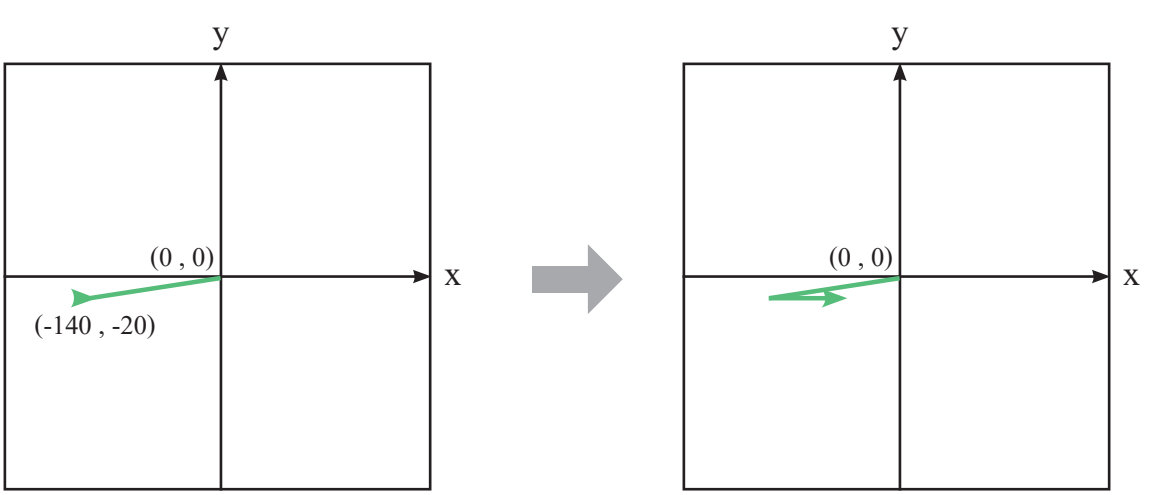
goto( ) 是指定負責畫圖的變數移動到畫布的指定位置，而在畫布正中間的坐標是 (0 , 0)，以下程式碼有指定起點，觀察看看畫出來的線會出現在什麼地方？

例如

《程式碼》

```
import turtle                    # 匯入 turtle 模組，以使用 Turtle 函式。
window = turtle.Screen()         # 產生畫布以進行畫圖。
john = turtle.Turtle()           # 建立一個海龜 Turtle，命名為變數 john。
john.goto(-140, -20)             # 定位到 (-140, -20) 的位置。
john.forward(90)                 # 往前走 90 個單位。
```

定位到 (-140 , -20)　　往前走 90 個單位



概念 penup( ) 及 pendown( ) 函式

在沒有 penup( ) 的情況下，只要用 turtle 物件的變數移動，就會跟著它移動的路徑畫出東西。pendown( ) 則是提醒這個變數，只要移動就是要畫出東西來。

若有 penup( ) 卻沒有 pendown( ) 的情況下，只會跟著 turtle 物件的變數移動，但不會畫出任何東西。

例如

《程式碼》

```
import turtle                    # 匯入 turtle 模組，以使用 Turtle 函式。
window = turtle.Screen()         # 產生畫布以進行畫圖。
john = turtle.Turtle()           # 建立一個海龜 Turtle，命名為變數 john。
john.penup()                     # 提筆。
john.forward(90)                 # 往前走 90 個單位。
```

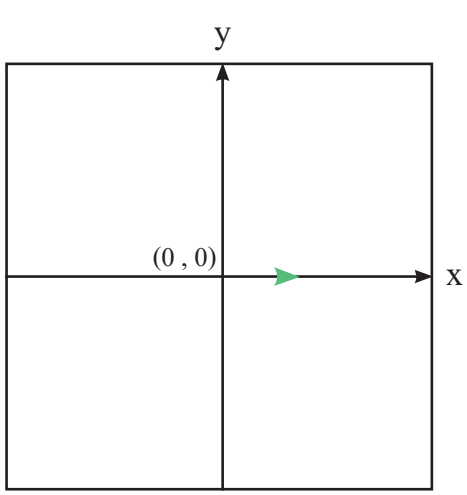
因此，使用 penup( ) 時，記得要使用 pendown( ) 才會開始繪圖。延續上方程式碼，補齊下筆的動作吧！

例如

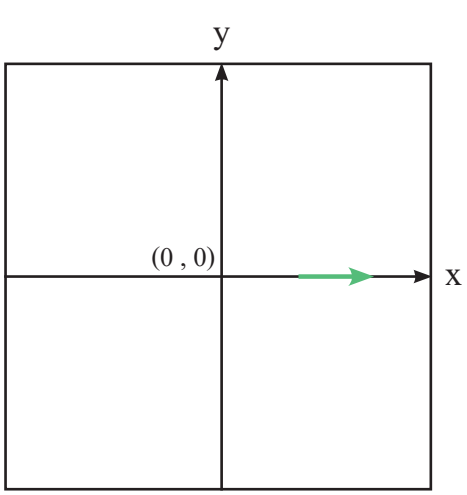
《程式碼》

```
john.pendown()                   # 下筆。
john.forward(90)                 # 往前走 90 個單位。
```

提筆後往前走 90 個單位



下筆後往前走 90 個單位



概念 for 迴圈的使用

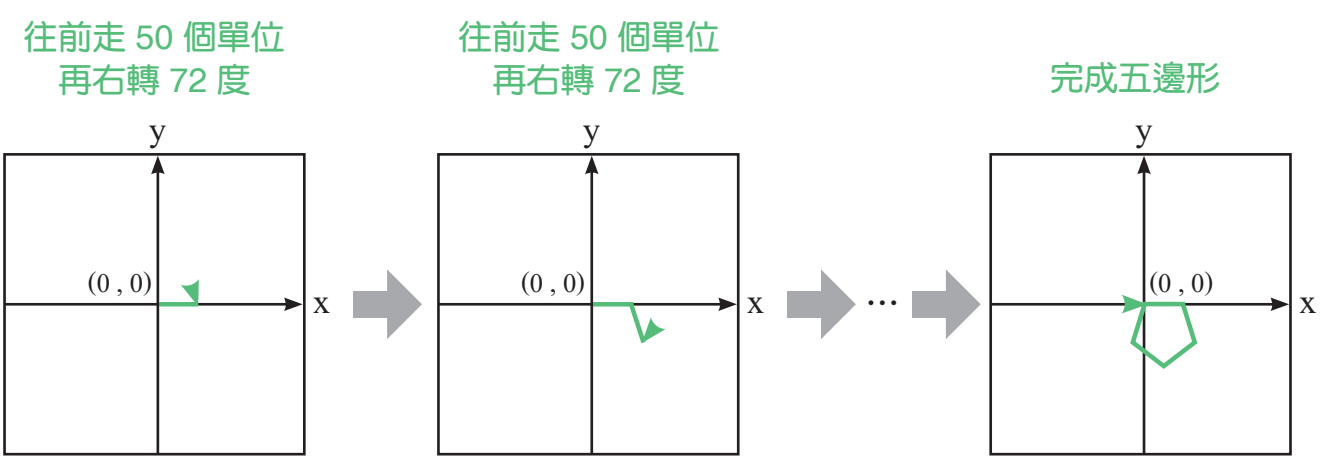
在 Python 中，串列是有序的物件集合，具有索引特性，功能類似我們在八年級中學到的陣列（在 Scratch 中稱為清單）。同樣地，串列的長度也可以變動。若要建立串列，可以使用 [ ]，至於串列中的每個元素，則是使用逗號來區隔。

迴圈變數 i 是用來控制迴圈執行的次數，但其實我們並沒有直接在迴圈中使用到這個變數。此外，若要創造出包含 0 , 1 , 2 , 3 這 4 個數字的串列，可以使用 range( ) 這個 Python 的內建函式來產生有規則的整數串列，例如：使用 range(4) 函式會產生一個從 0 開始，小於 4 的整數串列，因此它會回傳 [0 , 1 , 2 , 3] 這個串列。了解這些細節後，我們可以使用 range( ) 函式撰寫畫出一個五邊形的程式碼，語法如下：

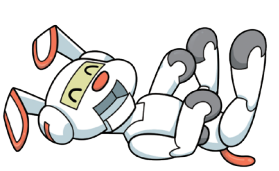
例如

《程式碼》

```
import turtle                          # 匯入 turtle 模組，以使用 Turtle 函式。
window = turtle.Screen()               # 產生畫布以進行畫圖。
john = turtle.Turtle()                 # 建立一個海龜 Turtle，命名為變數 john。
for i in range(5):                     # 執行計次式迴圈，重複 5 次。
    john.forward(50)                   # 往前走 50 個單位。
    john.right(72)                     # 右轉 72 度。
```



比較前一個範例畫正方形的程式碼，使用 for 迴圈，就可以簡化完成五邊形程式碼了！



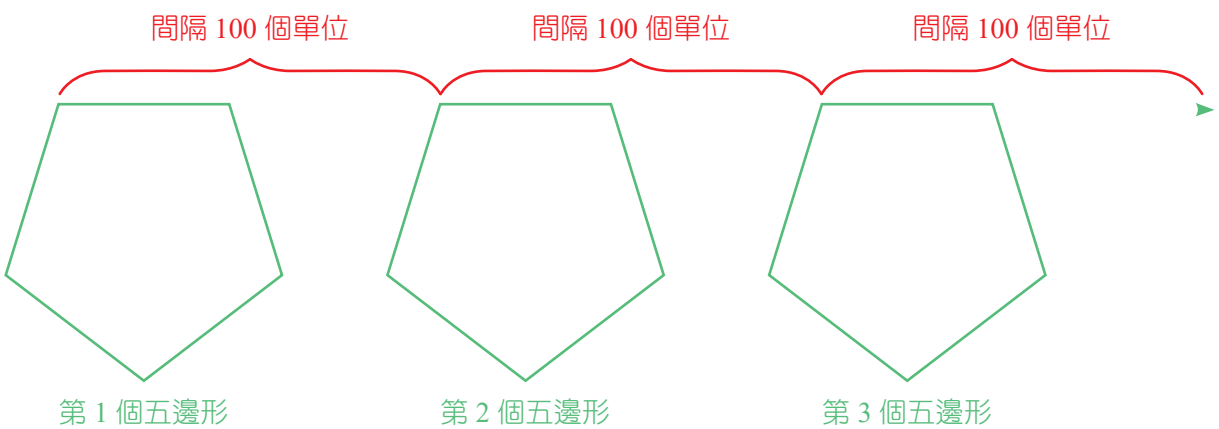
如果想將五邊形重複畫出 3 次，且每個五邊形都需要固定間隔的距離，這個時候我們可以使用雙迴圈。程式碼裡會有兩個 for 迴圈，為了清楚辨別，可以先拆分為兩個部分：一個部分為「畫一個五邊形」的動作，另一個部分則是從 for i in range(3) 開始，將「下筆、畫一個五邊形、提筆、往前移動 100 點」重複執行 3 次的動作，語法如下：

例如

《程式碼》

```
for i in range(3):                  # 執行計次式迴圈，重複 3 次。
    john.pendown()                  # 下筆。
    for j in range(5):              # 執行計次式迴圈，重複 5 次。
        john.forward(30)            # 往前走 30 個單位。
        john.right(72)              # 右轉 72 度。
    john.penup()                    # 提筆。
    john.forward(100)               # 往前走 100 個單位。
```

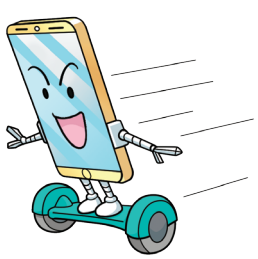
畫一個五邊形



在這裡要特別注意兩個地方：

⑴因為有兩個不同的迴圈，所以迴圈變數不能混用，一個是 i、一個是 j。

⑵縮排的位置關係到該行程式碼是屬於哪一個迴圈的區塊。



2-3 Python 程式設計的應用

經過前面的課程，同學們已經熟悉了基本的輸入、輸出、迴圈及繪圖的功能，接下來我們要統整學習過的觀念來完成一個程式。

範例：你想畫什麼，我來畫給你看

請設計一個程式，有三個選項讓使用者選擇要畫三角形、六邊形或五角星星的圖形。然後再讓使用者輸入想要的圖形數量，畫出對應的圖形結果。

問題分析

我們可以將這個程式拆解為幾個部分如下：

①如何匯入 turtle 模組，並定位起始位置？

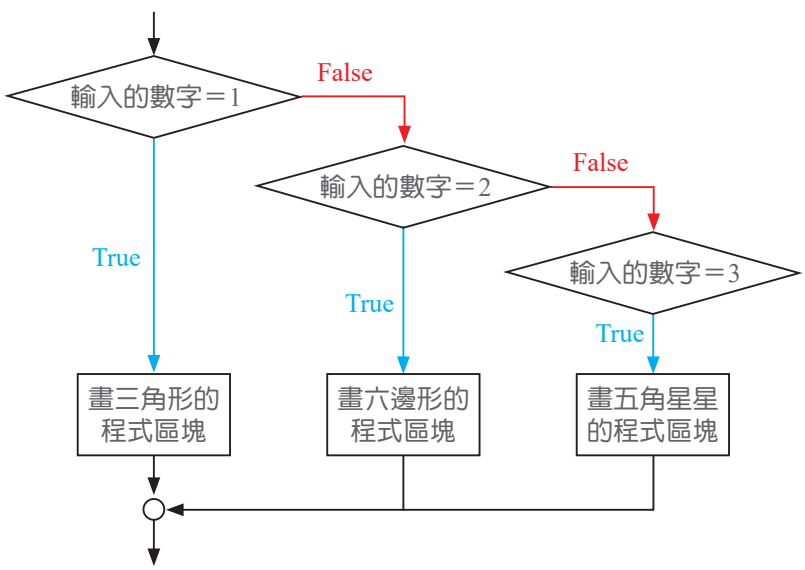
1. 執行時，如何設定起始位置，並考量畫完圖形時，不會超出畫面？

②如何產生選單讓使用者輸入數字，決定要畫的圖形與數量？

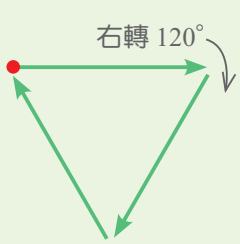
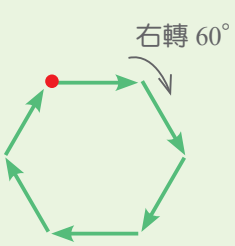
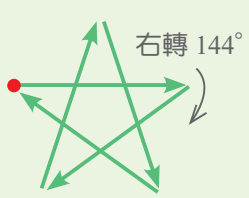
1. 執行時，如何製作選單？

編號	1	2	3
圖形	三角形	六邊形	五角星星

③如何判斷輸入的數字，分別代表何種圖形？



④如何使用迴圈畫出三角形、六邊形和五角星星？

編號	1	2	3
圖形（● 為起點）			

⑤如何使用迴圈達成重複畫圖形的結果？

解題步驟

問題拆解 1 如何匯入 turtle 模組，並定位起始位置？

步驟 1 匯入 turtle 模組，並定位畫筆的起始位置。

1. 匯入 turtle 模組，並提筆。
2. 請同學想想看，設定**起始位置**時，虛線框要填入什麼？

```
import turtle
t = turtle.Turtle()
t.penup()
t.goto(        )
```

問題拆解 2 如何產生選單讓使用者輸入數字，決定要畫的圖形與數量？

步驟 2 製作選單，並詢問使用者想畫的圖形與數量。

1. 設定圖形變數 **draw_what** 與數量變數 **draw_times**。
2. 請同學想想看，虛線框要填入什麼？

```
draw_what = (input('輸入想畫的圖形(1.三角形 2.六邊形 3.五角星星)：'))
draw_times = (input('你想畫幾個這樣的圖形：'))
```

問題拆解 3 如何判斷輸入的數字，分別代表何種圖形？

步驟 3 讓程式判斷輸入的數字所代表的圖形。

1. 當輸入的數字為 **1**，對應的圖形為**三角形**，以此類推。
2. 請同學想想看，虛線框要填入什麼？

```
if draw_what == 1:
    畫三角形的程式區塊
elif draw_what ==  :
    畫六邊形的程式區塊

    畫五角星星的程式區塊
```

畫圖形的程式區塊，步驟 4 再完成撰寫。

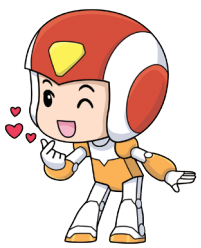
問題拆解 4 如何使用迴圈畫出三角形、六邊形和五角星星？

步驟 4 撰寫畫三角形、六邊形和五角星星的程式區塊。

1. 請同學想想看，考量畫布大小，畫出一個三角形時，虛線框要填入什麼？
2. 請完成畫六邊形和五角星星的程式區塊。

```
t.pendown()                  #畫三角形的程式區塊。
for j in range(3):           #設定三角形邊數。
    t.forward(   )           #設定三角形邊長。
    t.right(   )             #設定三角形角度。
t.penup()
```

想一想，繪製六邊形和五角星星，旋轉角度分別是多少呢？



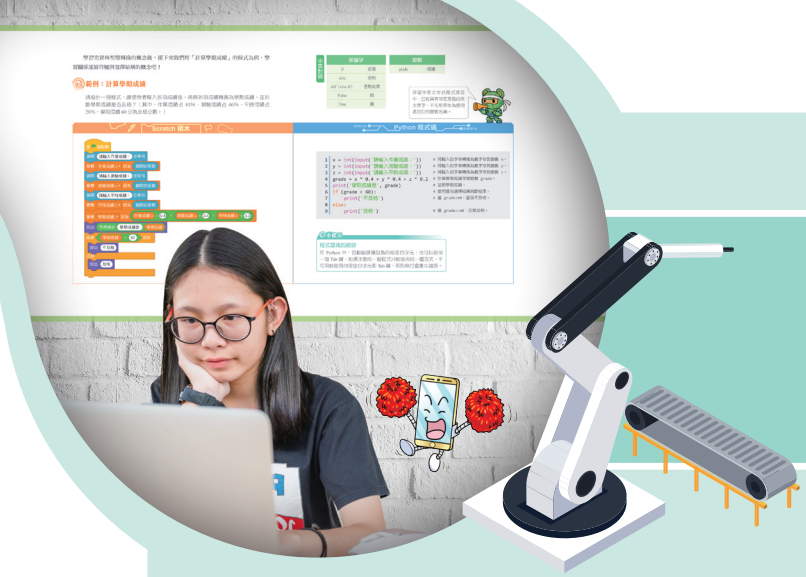
問題拆解 5 如何使用迴圈達成重複畫圖形的結果？

步驟 5 結合前面兩個步驟，畫出指定數量的圖形。

1. 當畫出一個三角形後，產生固定的間隔，再重複畫三角形時，才不會重疊。
2. 請同學想想看，執行**計次式迴圈**時，虛線框內要填入什麼？
3. 請同學想想看，設定畫三角形的**間隔距離**時，虛線框要填入什麼？
4. 請完成畫六邊形和五角星星的間隔距離設定。

```
for i in range(          ):      #執行計次式迴圈。
    if draw_what == 1:
        畫三角形的程式區塊
        t.forward(   )           #設定間隔距離。
```

請完成整個專題程式，接著想想看，如果使用者輸入 1 到 3 以外的數字，程式會有什麼反應？



重點回顧

本章選用 Python 來學習文字式程式語言，並導入學習過的 Scratch 範例，讓同學能夠將問題從積木式語言銜接到文字式語言。Python 是一種廣泛使用且功能強大的通用型程式語言，很適合做為第一個用來學習的文字式程式語言。下表為本章學習過的概念、函式及語法：

概念		語法
輸入		變數 = input('提示字串')
輸出		print(項目 1 , 項目 2, ...)
選擇結構	單向選擇結構	if 條件式: 　　程式區塊

	雙向選擇結構	if 條件式:
		　　程式區塊 1
		else:
		　　程式區塊 2

	多向選擇結構	if 條件式 1:
		　　程式區塊 1
		elif 條件式 2:
		　　程式區塊 2
		elif 條件式 3:
		　　⋮
		else:
		　　程式區塊 n

重複結構	產生有規律串列	range(起始值, 終止值, 累加值)
	計次式迴圈	for 迴圈變數 in 串列： 　　程式區塊
變數		變數名稱 = 變數值
串列		串列名稱 = [元素 1, 元素 2, ...]

資料型態	轉換函式	使用時機
整數	int()	強制轉換為整數。
浮點數	float()	強制轉換為浮點數。
布林值	bool()	強制轉換為布林值。
字串	str()	強制轉換為字串。

符號類型	符號	意義
算術運算符號	+	加法
	-	減法
	*	乘法
	/	除法
	%	除法取餘數
	//	除法取商數
	**	次方

符號類型	符號	意義
關係運算符號	==	相等
	!=	不相等
	>	大於
	<	小於
	>=	大於或等於
	<=	小於或等於

繪圖模組	語法
匯入 turtle 模組	import turtle
建立 Turtle	變數 = turtle.Turtle()
產生畫布	turtle.Screen()
往前	forward()
右轉	right()
畫布大小	window.setup()
提筆	penup()
下筆	pendown()
定位	goto()