



第2章

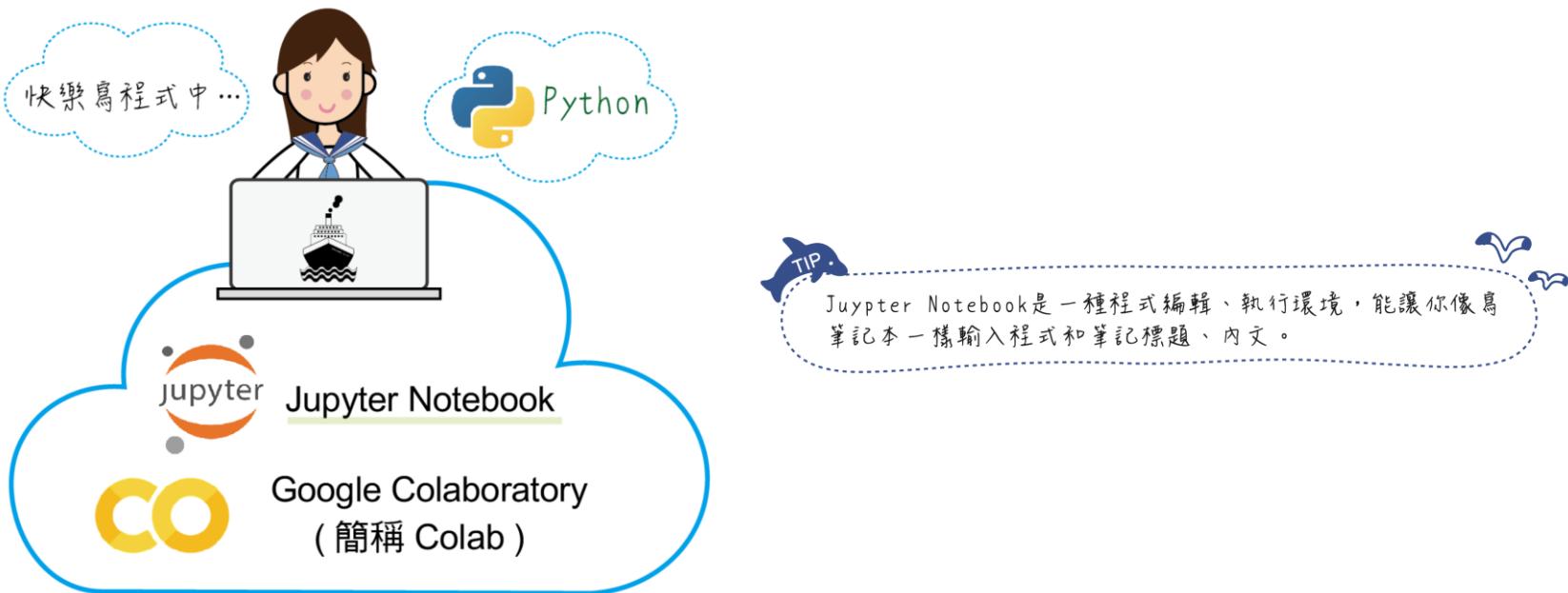
Python 資料科學實作平台： Google Colab

在這個資訊科技蓬勃發展的時代中，要如何才能在大數據 (Big Data) 和人工智慧 (AI) 等紅透半邊天的領域中，攻下一個超越別人的有利位置呢？就讓我們以 Python 為基礎，開始向目標啟航邁進吧！

2-1

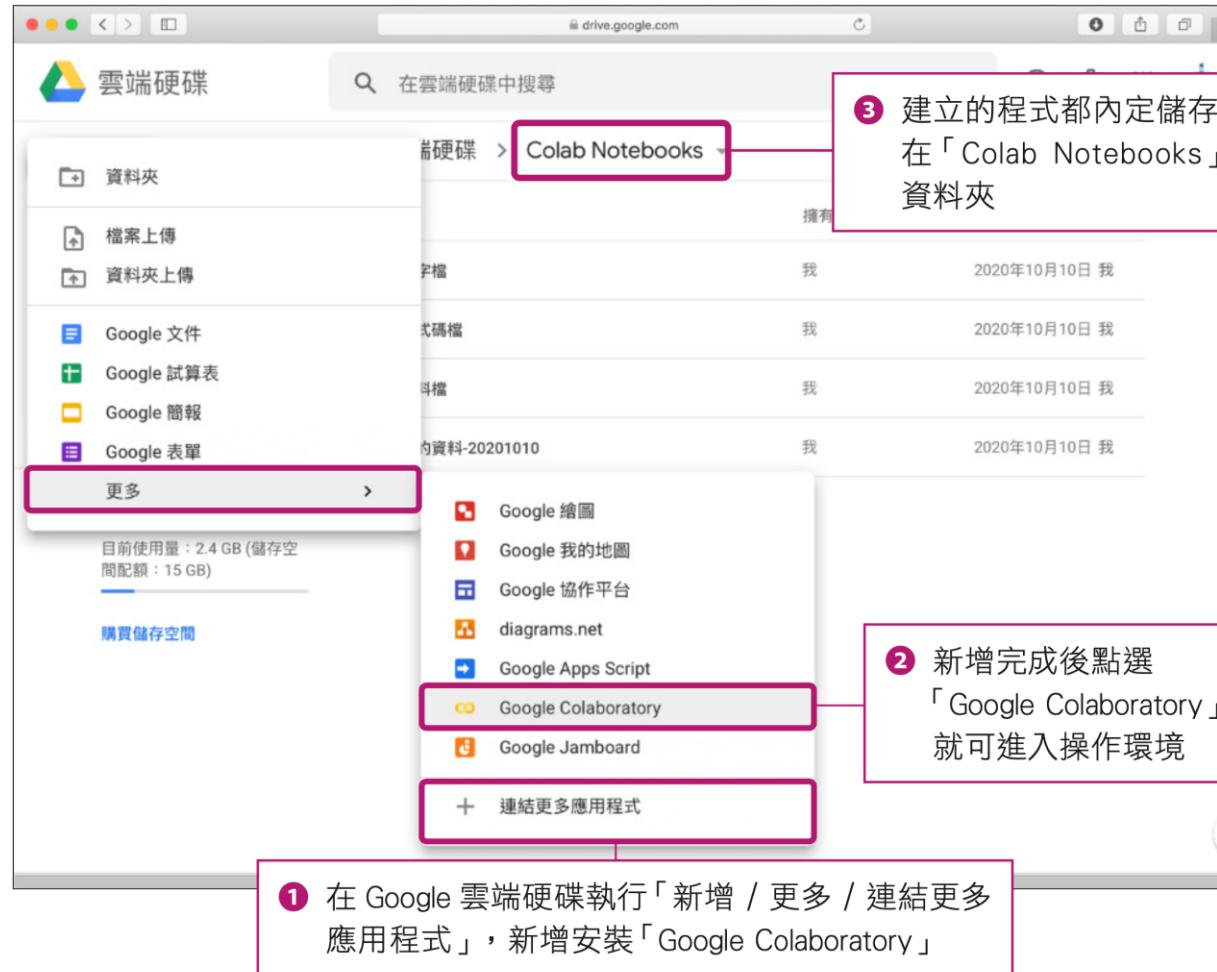
Python 線上程式開發平台 — Colab

- Python 是一種容易入門的程式設計語言，語法簡單好寫，背後社群強大，在數據分析領域深受學界及業界的喜愛
- 不必安裝即可直接在雲端上使用的「互動式線上程式開發平台」—Colab來編寫程式



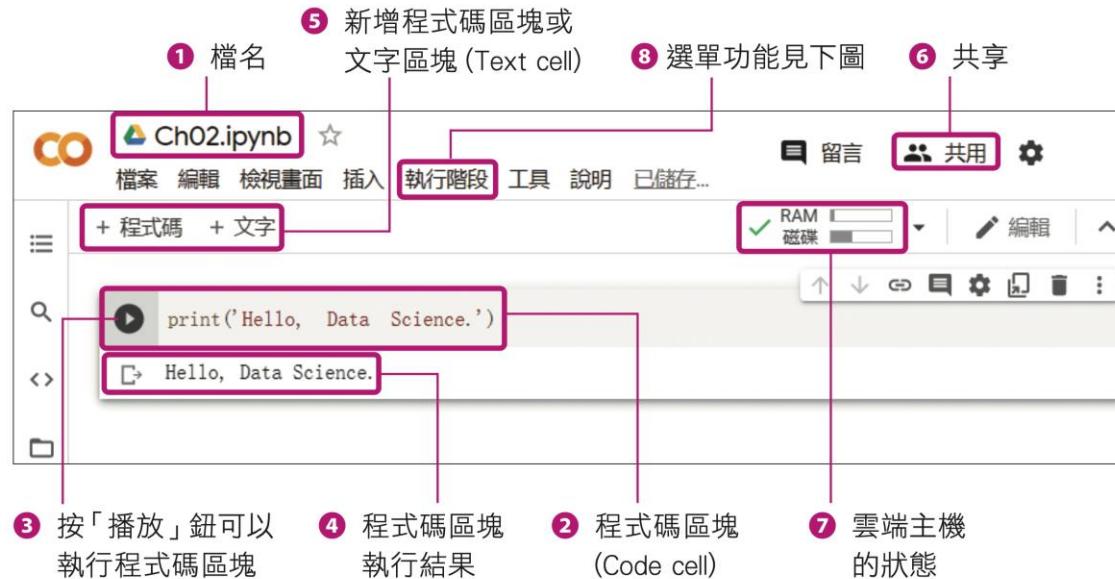
▲ 利用「互動式線上程式開發平台」- Colab 來編寫 Python 程式

外掛 Colab 應用程式



▲ 在 Google 雲端硬碟中外掛 Colab 應用程式

Colab 環境介紹



全部執行	Ctrl+F9
執行上方的儲存格	Ctrl+F8
執行聚焦的儲存格	Ctrl+Enter
執行選取範圍	Ctrl+Shift+Enter
執行下方的儲存格	Ctrl+F10
中斷執行	Ctrl+M I
重新啟動執行階段	Ctrl+M .
重新啟動並執行所有儲存格	
恢復原廠設定的執行階段	

- ⑧ 清除所有變數並重新啟動



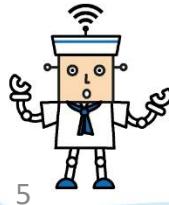
2-2-1 輸出函式

`print()` 函式可用來將資料輸出到螢幕。

`print()`
輸出函式

`print(項目1, 項目2, ...)`

1. 不同項目之間用「,」分隔
2. 如果要輸出文字時則加上雙引號「" "」或單引號「' '」
3. 項目可以是常數，也可以是
變數





print() 函式

EX2-2.1.ipynb

01

在程式碼區塊 (Code Cell) 中輸入以下的程式碼。

+ 程式碼 + 文字

```
[ ] 1 n = 'Python'  
     2 print('Hello,',n)
```

按 Enter 可在程式區塊內斷行，接著就可輸入下一行敘述

若想顯示行數，可透過 Colab 的「工具」選單
開啟「設定 / 編輯器」內的「顯示行數」按鈕



print() 函式

EX2-2.1.ipynb

02

輸入完程式將滑鼠游標移至程式區塊的上面邊緣處，按畫面上的 **+ 文字** 插入一個**文字區塊** (Text Cell)，雙按後可加上說明文字
註 2。完成後按 **▶** 看看程式執行結果。

程式檔名

CO Untitled0.ipynb

檔案 編輯 檢視畫面 插入 執行階段

+ 程式碼 + 文字 連線

寫程式了!!哈囉!Python.

<> 1 n = 'Python'
2 print('Hello,', n)

執行程式

說明文字

程式碼

執行結果



print() 函式

EX2-2.1.ipynb

03

在「程式檔名」方塊中可修改成不同的檔案名稱（如：EX2-2.1.ipynb）。

更改檔名

EX2-2.1.ipynb

檔案 編輯 檢視畫面 插入 執行階段

+ 程式碼 + 文字

寫程式了!!哈囉!Python.

```
1 n = 'Python'  
2 print('Hello,',n)
```

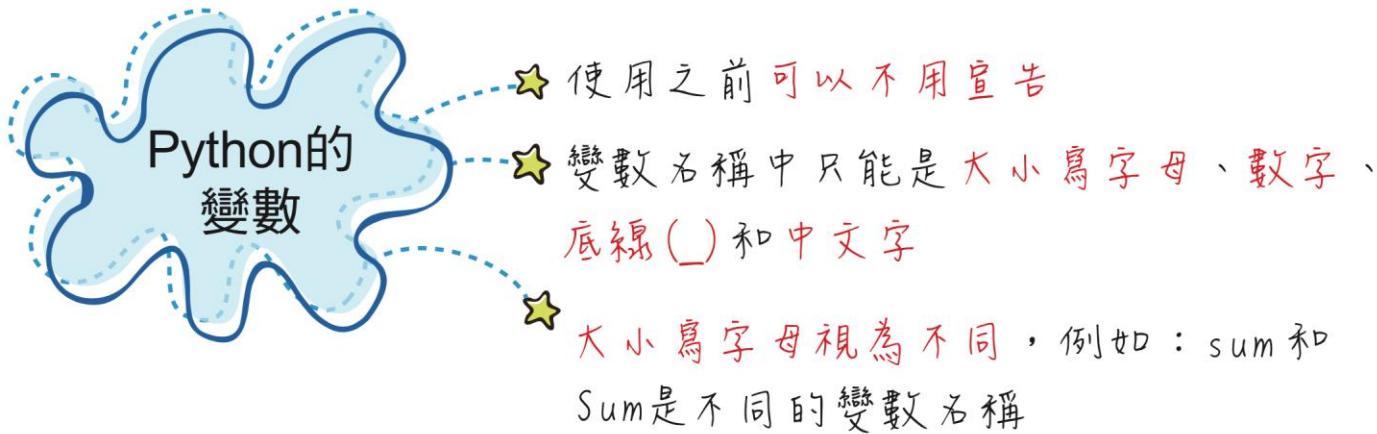
>Hello, Python

The screenshot shows the Google Colab interface. At the top, there's a '更改檔名' (Change Name) button with a downward arrow. Below it, the file name 'EX2-2.1.ipynb' is highlighted with a red rectangle. The menu bar includes '檔案' (File), '編輯' (Edit), '檢視畫面' (View), '插入' (Insert), and '執行階段' (Runtime). On the left, there are icons for '程式碼' (Code), '文字' (Text), and a search bar. The main workspace contains the text '寫程式了!!哈囉!Python.' followed by a code cell with two lines of Python code: '1 n = 'Python'' and '2 print('Hello,',n)'. The output of the code is 'Hello, Python'.



2-2-2 變數

程式執行時會將資料暫存於**變數** (Variables) 中，變數就和數學中的 x, y, z 等代數很相似。例如在 $y = 2x$ 程式敘述中，當 $x = 10$ 就會得到 $y = 20$ ； $x = 20$ 時則會得到 $y = 40$ 。因為 x 的數值可以不停的變化，所以就將之稱為變數。





設定變數值

EX2-2.2.ipynb

練習在程式中建立數值、字串等變數，完成後以 print() 函式印出其內容。



```
1 n = 'Python'  
2 m = 2099  
3 print('Hello,', n, m, '.')
```

```
⇒ Hello, Python 2099 .
```



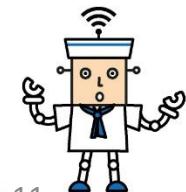
2-2-3 輸入函式

執行程式的過程中，如果需要使用者從鍵盤輸入資料時，可以使用 `input()` 函式。

`input()`
輸入函式

字串變數 = `input('提示文字')`

- 
1. '提示文字' 可以省略
 2. 輸入的資料會以字串型別設定到字串變數中





實作 input() 函式

EX2-2.3.ipynb

利用 input() 函式讓使用者輸入資料，完成後再印出結果。



```
1 n = input('請輸入文字：')
2 print('Hello,',n)
```

有提示文字



請輸入文字：機器學習
Hello, 機器學習



```
1 n = input()
2 print('Hello,',n)
```

無提示文字

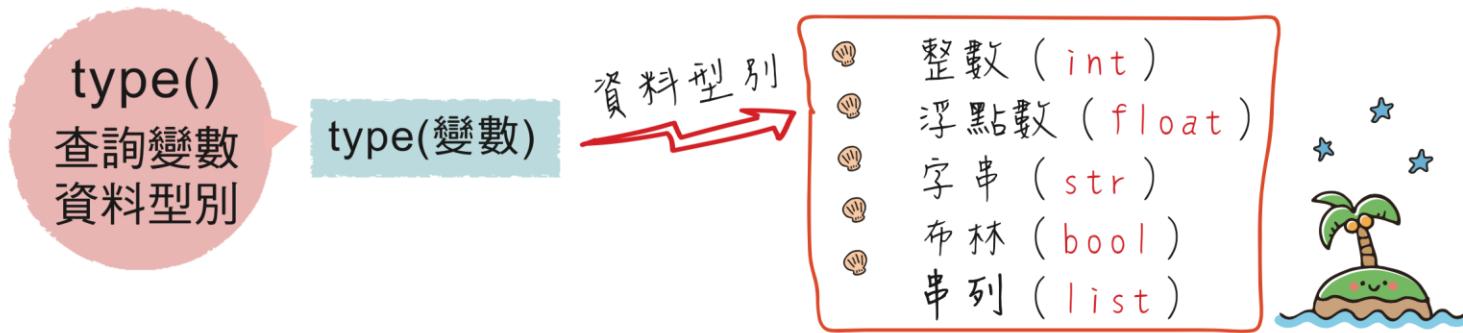


機器學習
Hello, 機器學習



2-2-4 資料型別

Python 提供多種不同的資料型別，例如：[整數](#) (int)、[浮點數](#) (float)、[字串](#) (str)、[布林](#) (bool)、[串列](#) (list) 等。變數在使用前不用宣告，Python 會自動依其內容值給定合適的型別，方便使用且極具彈性。若要查詢變數的資料型別可以用 type() 函式。





實作 資料型別

EX2-2.4.ipynb

01

建立整數、浮點數、字串、布林四種型別的變數，完成後印出所有變數的內容。



```
1 a = 4
2 b = 3.14
3 c = 'Alice'
4 d = True
5 print(a,b,c,d)
```

```
→ 4 3.14 Alice True
```



資料型別

EX2-2.4.ipynb

02

利用 type() 函式看看這些變數各是屬於何種資料型別。



1 type(a)



int



1 type(b)



float



1 type(c)



str



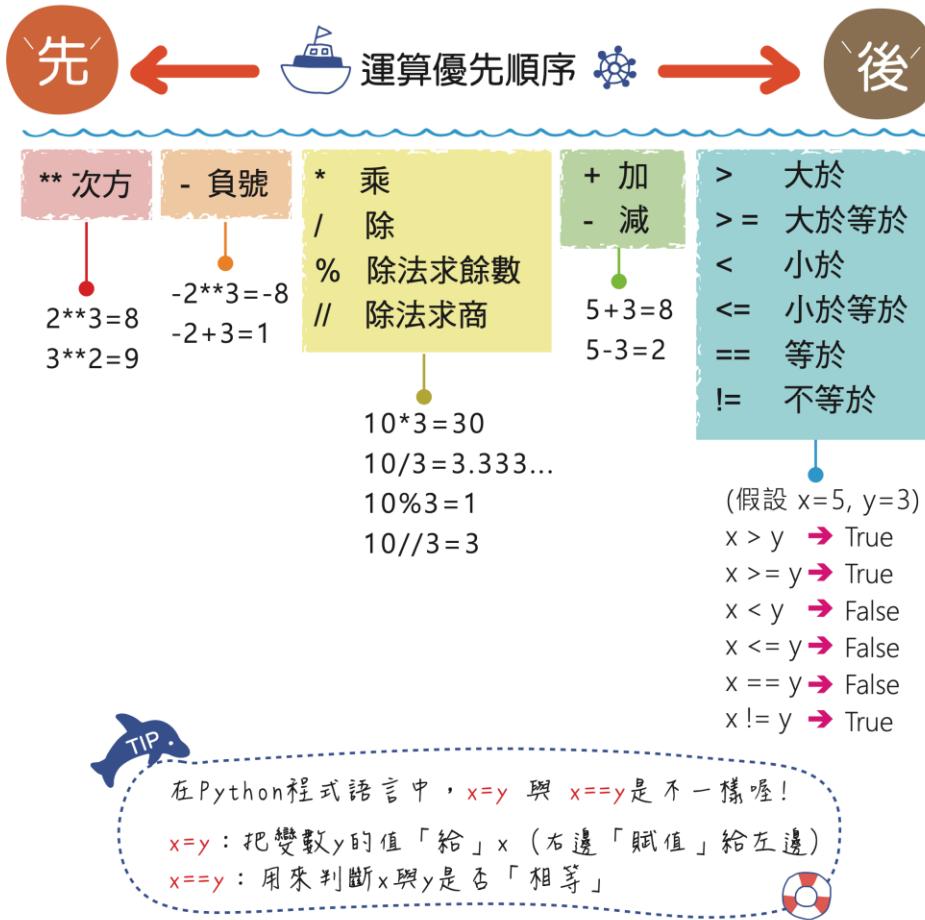
1 type(d)



bool



2-2-5 資料運算





資料運算

EX2-2.5a.ipynb

01

練習在程式中使用 + , - , *, / , ** 做數值運算。



```
1 a = 3  
2 b = 2  
3 print(a+b, a-b, a*b, a/b, a**b)
```

```
→ 5 1 6 1.5 9
```



資料運算

EX2-2.5a.ipynb

02

練習字串的連結 (+) 和重複 (*) 運算。

```
1 s1 = '資料'  
2 s2 = '分析'  
3 s3 = s2 + s1      #字串連結 (+)：將不同字串連結成一個字串  
4 print(s3)  
5 s4 = '讚!' * 3    #字串重複 (*)：產生重複的字串  
6 print(s4)
```

☛ 分析資料
讚!讚!讚!



在 Python 一行程式碼中，

「#」之後的文句是註解。

* 註解通常用來說明程式碼的用途

* 註解內可以寫任意的文句

* 註解不會被電腦執行





資料運算

EX2-2.5a.ipynb

03

練習兩個數值的比較運算。



```
1 x = 60  
2 y = 92  
3 print(y>x, y==x, y!=x)
```



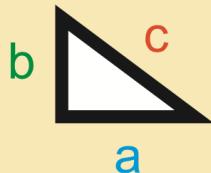
True False True



綜合練習～畢氏定理

EX2-2.5b.ipynb

畢式定理



$$a^2 + b^2 = c^2$$

$$\rightarrow c = \sqrt{a^2 + b^2}$$

$$\rightarrow c = (a**2 + b**2)**0.5$$

開根號就是 0.5 次方喔！



```
1 a = 3  
2 b = 4  
3 c = (a**2 + b**2) ** 0.5  
4 print(c)
```

5.0



綜合練習～成績處理

EX2-2.5c.ipynb

練習讓使用者在程式執行時輸入三項成績並自動計算總成績，完成後列印計算結果。

```
▶ 1 c = int(input('請輸入國文分數:'))
2 e = int(input('請輸入英文分數:'))
3 m = int(input('請輸入數學分數:'))
4 s = c + e + m
5 print('總分為:',s)
```

```
⇨ 請輸入國文分數:92
    請輸入英文分數:88
    請輸入數學分數:82
    總分為: 262
```



- `input()`函式可以讓使用者輸入資料，若輸入為「數值」會視為「文字」。
- 文字轉成數值函式：`int()`或`float()`，其中`int()`只能轉換成整數（如：`'99'`），而`float()`則可以轉換包含小數的數值（如：`'99.9'`）。





2-2-6 串列 (list)

想用變數來記錄班上所有同學的姓名，通常會考慮使用**串列** (list) 的資料型別，它就像其他程式語言的**陣列** (Array)，操作上相當簡易和方便！

一維 串列

串列名稱 = [元素0, 元素1, ...]

c = ['國文', '英文', '數學']

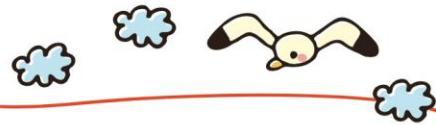
c[0] c[1] c[2]

國文	英文	數學
----	----	----

s = [84, 92, 88]

s[0] s[1] s[2]

84	92	88
----	----	----

- 
- 元素用中括號「[]」括起來
 - 元素之間用「,」分隔
 - 元素的索引編號從「0」開始
 - 元素可以是不同的資料型別
 - 串列名稱[元素索引]表示某個元素的值
 - 例▶ c[0]='國文'，s[2]=88。
 - Python支援三維以上的串列



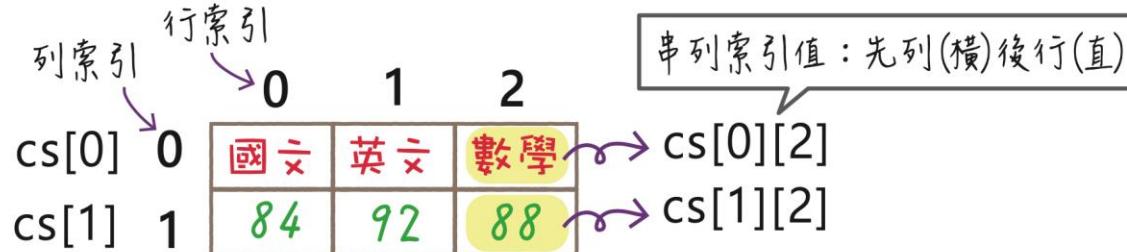


2-2-6 串列 (list)

二維
串列

串列名稱 = [[第0個一維串列] , [第1個一維串列] , ...]

`cs = [['國文', '英文', '數學'], [84, 92, 88]]`



1. 串列名稱[串列索引]表示某個串列索引中的所有元素

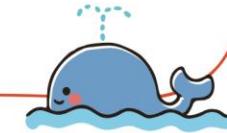
例 ➤ `cs[0]=['國文', '英文', '數學']`

`cs[1]=[84, 92, 88]`

2. 串列名稱[列索引編號][行索引編號]表示某個元素的值

例 ➤ `cs[0][2]='數學'`

`cs[1][2]=88`





串列的應用

EX2-2.6.ipynb

01

建立包含多個資料的一維串列，完成後印出整個串列以及某一個元素。

```
▶ 1 c = ['國文', '英文', '數學']
  2 s = [84, 92, 88]
  3 print(c)
  4 print(s)
  5 print(c[2], s[2])
```

```
⇨ ['國文', '英文', '數學']
    [84, 92, 88]
    數學 88
```



串列的應用

EX2-2.6.ipynb

02

建立包含多個資料的二維串列，完成後印出整個串列以及某一個元素。



```
1 cs = [['國文', '英文', '數學'], [84, 92, 88]]  
2 print(cs)  
3 print(cs[0][2], cs[1][2])
```

```
→ [['國文', '英文', '數學'], [84, 92, 88]]  
數學 88
```



2-2-7 條件判斷

- 條件判斷能夠根據條件的成立與否，決定要執行哪些程式碼，這項功能經常會出現在日常應用中。條件判斷依不同的使用狀況可分為
 - 單向
 - 雙向
 - 多向





實作

條件判斷 (單向) - 猜數字遊戲

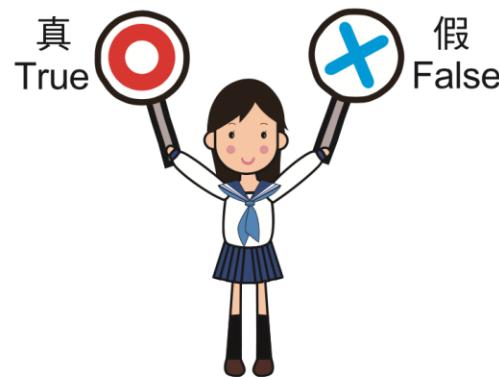
EX2-2.7a.ipynb

讓使用者輸入一個數字，使用 if 來判斷並印出與答案的關係是「猜中」、「太大」或「太小」。

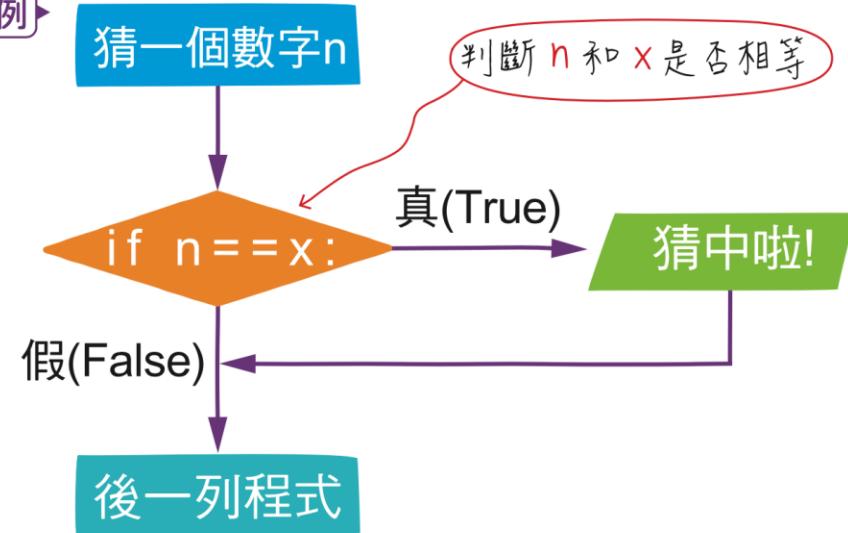
if
條件判斷
(單向)

if 條件式：
程式區塊

1. 條件成立時執行程式區塊，不成立時則不執行程式區塊而繼續往下執行
2. if敘述最後面要加上「:」
3. 程式區塊需「縮排」



例





條件判斷 (單向) - 猜數字遊戲

EX2-2.7a.ipynb

注意這兩行！變數賦值是用 =，而判斷是否
「相等」是用 ==，兩者不同不可混用喔！

```
1 x = 60
2 n = int(input('猜一個數字：'))
3 if n == x :
4     print('猜中啦！')
5     print('Good job')
6 if n > x :
7     print('太大！')
8     print('答案是：', x, '才對！')
9 if n < x :
10    print('太小！')
```

設定 3 個
條件式

```
C> 猜一個數字：70
    太大！
    答案是： 60 才對！
```



實作 條件判斷 (雙向)

EX2-2.7b.ipynb

使用 if-else 判斷成績是否及格，成績大於等於 60 時印出「及格」，否則就印出「不及格」。

if-else
條件判斷
(雙向)

if 條件式：
 程式區塊 1
else：
 程式區塊 2

1. 條件成立時執行程式區塊1，條件不成立時則執行程式區塊2
2. if和else要對齊，且最後面都要加上「:」
3. 程式區塊皆需「縮排」並對齊



例▶



成績 n

判斷成績是否 ≥ 60 分

真(True)

假(False)

及格

不及格

後一列程式



```
1 n = 86
2 if n >= 60:
3     print('及格')
4 else:
5     print('不及格')
```

撰寫條件判斷(雙向)

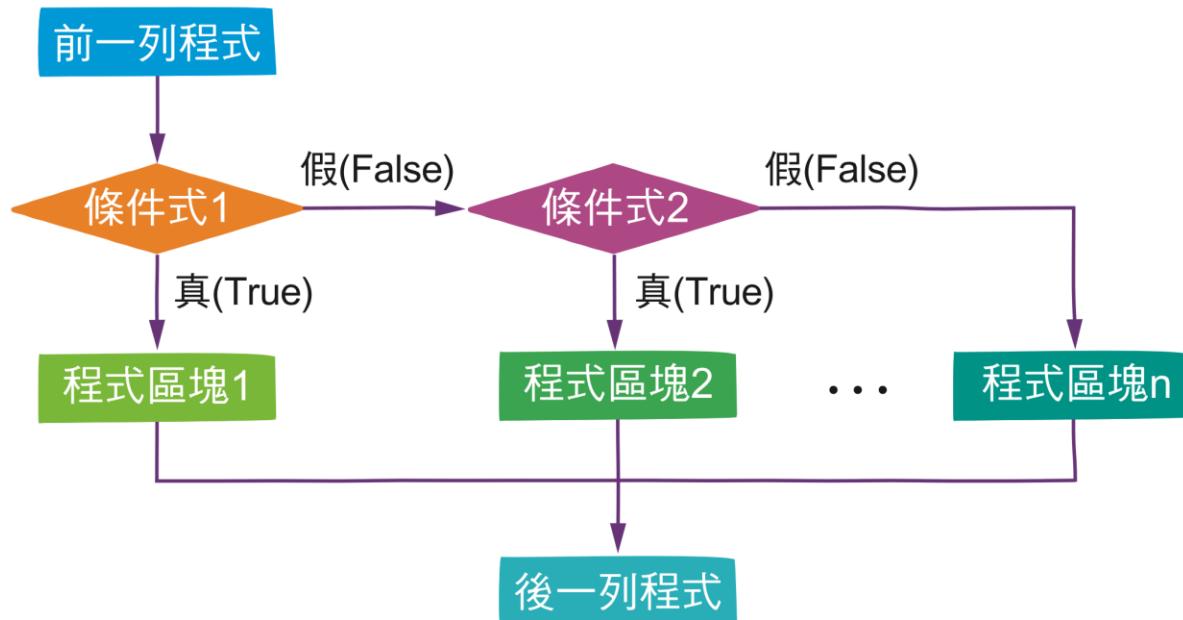
→ 及格



if-elif-else 條件判斷 (多向)

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

- 1. 條件式1成立時執行程式區塊1，否則當條件式2成立時執行程式區塊2，一次判斷一個條件式，皆不成立時則執行程式區塊n
- 2. if、elif 和 else都要對齊，且最後面都要加上「:」
- 3. 程式區塊皆需「縮排」並對齊

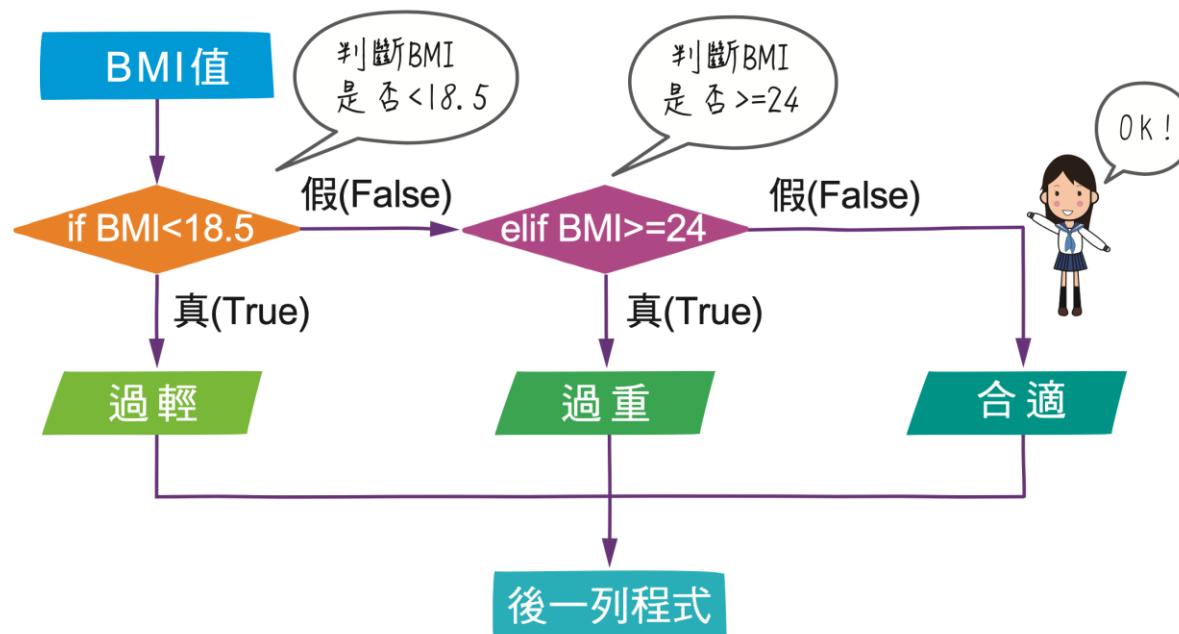




- 寫一個計算 BMI 的程式，判斷體重是「合適」、「過重」、「過輕」

公式： $BMI = \frac{\text{體重 (公斤)}}{\text{身高}^2 (\text{公尺}^2)}$

判斷條件是：若 $BMI < 18.5$ 為「過輕」， $18.5 \leq BMI < 24$ 為「合適」，否則就算「過重」。





```
1 h = float(input('身高(m):'))
2 w = float(input('體重(Kg):'))
3 BMI = w / (h**2)
4 print('BMI:',BMI)
5 if BMI < 18.5:
6     print('過輕')
7 elif BMI >=24:
8     print('過重')
9 else:
10    print('合適')
```

撰寫條件判斷(多向)

```
↳ 身高(m):1.72
    體重(Kg):72
    BMI: 24.337479718766904
    過重
```



2-2-8 重複執行

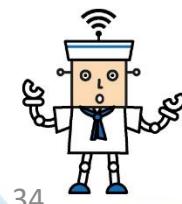
靈活運用**重複執行**可以大幅縮短重複的程式碼，讓程式更加簡潔與容易閱讀，`for` 迴圈就是一種常用的重複執行。

for 迴圈（單層）

for
迴圈
(單層)

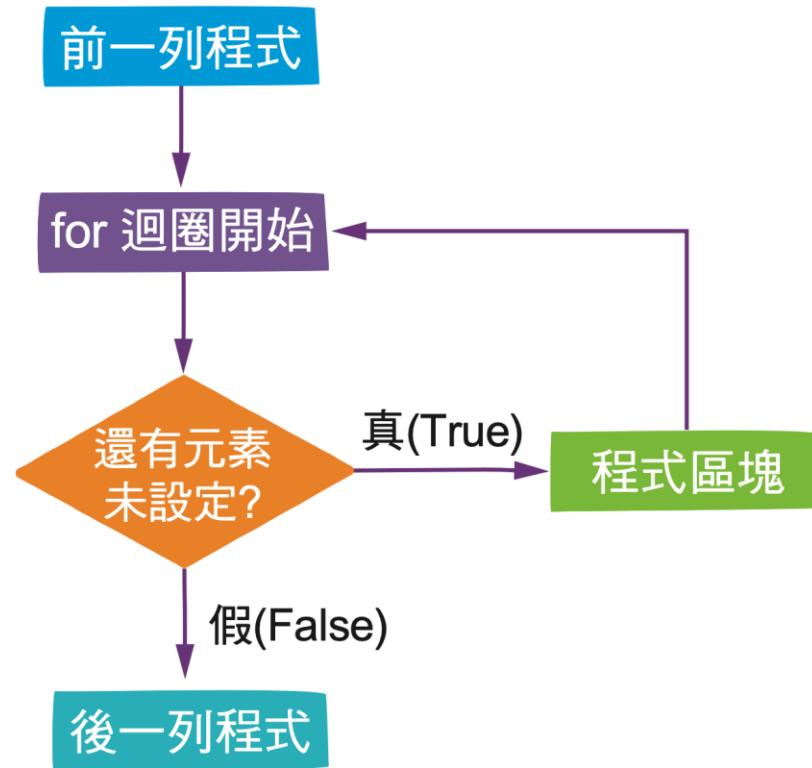
for 變數 in 串列：
 程式區塊

1. `for` 迴圈執行時會依序取出串列中的元素，當作該回合執行時的變數值
2. `for` 迴圈敘述的最後面要加上「`:`」，程式區塊需縮排
3. 可以使用`range()`函式來產生指定的範圍值(後述)





2-2-8 重複執行





重複執行 (單層 for)

EX2-2.8a.ipynb

建立一個串列包含寶可夢的 4 隻 (或以上) 寶物名字，完成後印出串列中每 1 隻寶物的名字。



```
1 寶可夢 = ['妙蛙種子', '妙娃草', '噴火龍', '綠毛蟲']
2 for 抓寶 in 寶可夢:
    3     print(抓寶)
```

⇨ 妙蛙種子
妙娃草
噴火龍
綠毛蟲

寶可夢 = ['妙娃種子', '妙娃草', '噴火龍', '綠毛蟲']



妙娃種子
妙娃草
噴火龍
綠毛蟲

b

第 0 次迴圈 → 抓寶 = '妙娃種子' → 印出 妙娃種子
第 1 次迴圈 → 抓寶 = '妙娃草' → 印出 妙娃草
第 2 次迴圈 → 抓寶 = '噴火龍' → 印出 噴火龍
第 3 次迴圈 → 抓寶 = '綠毛蟲' → 印出 綠毛蟲



重複執行 (單層 for)

EX2-2.8b.ipynb

建立包含考試科目和成績的二個串列，完成後分別使用串列及 range() 函式搭配 for 迴圈印出每個元素的值。

```
▶ 1 t = ['國文', '英文', '數學', '資訊科技']
  2 s = [84, 92, 88, 95]
  3 for i in [0,1,2,3]:
  4     print(t[i])
  5 for i in range(4):
  6     print(s[i])
```

```
⇨ 國文  
英文  
數學  
資訊科技  
84  
92  
88  
95
```

```
t = ['國文', '英文', '數學', '資訊科技']      s = [84, 92, 88, 95]
```

t[0]	國文	s[0]	84
t[1]	英文	s[1]	92
t[2]	數學	s[2]	88
t[3]	資訊科技	s[3]	95



TIP :

使用 `range()` 函式來產生指定的範圍值：

1. `range(n)`：產生 $0 \sim n-1$ 的連續整數

例 → `range(10)` 會產生 $0 \sim 9$ 的連續整數

2. `range(起始值, 終止值, 間隔值)`：產生起始值 ~ (終止值 - 1) 之間，
且有間隔的整數

例 → `range(1, 9, 2)` 會產生 $1, 3, 5, 7$ 這 4 個整數





重複執行 (雙層 for)

EX2-2.8c.ipynb

建立包含科目名稱和成績的二維串列 cs，完成後利用雙層 for 迴圈依次印出 cs 二維串列中每個元素的值。



```
1 cs = [['國文', '英文', '數學', '資訊科技'], [84, 92, 88, 95]]  
2 for i in [0,1]:  
3     for j in [0,1,2,3]:  
4         print(cs[i][j])
```

→ 國文
英文
數學
資訊科技
84
92
88
95

for 迴圈 (雙層)

for
迴圈
(雙層)

for 變數1 in 串列 : ← 外圈
for 變數2 in 串列 : ← 內圈
 程式區塊

外圈每執行一次，
內圈要重複執行
程式區塊至結束後，
再跳到外圈繼續執行。

```
for i in[0,1]: ← i外圈  
    for j in[0,1,2,3]: ← j內圈
```

外圈 i 執行第 1 次 ($i=0$)，
內圈 j 會執行 4 次 ($j=0 \sim 3$) 後，
再跳到外圈 i 繼續執行
第 2 次 ($i=1, j=0 \sim 3$)，共會執
行 8 次 (8 次的順序如下)

前 4 次

i 外圈 j 內圈
 $i = 0, j = 0 \sim 3$



後 4 次

$i = 1, j = 0 \sim 3$

第 1 次 → 第 2 次 → 第 3 次 → 第 4 次

$i \quad j$
cs[0][0] cs[0][1] cs[0][2] cs[0][3]

國文	英文	數學	資訊科技
84	92	88	95

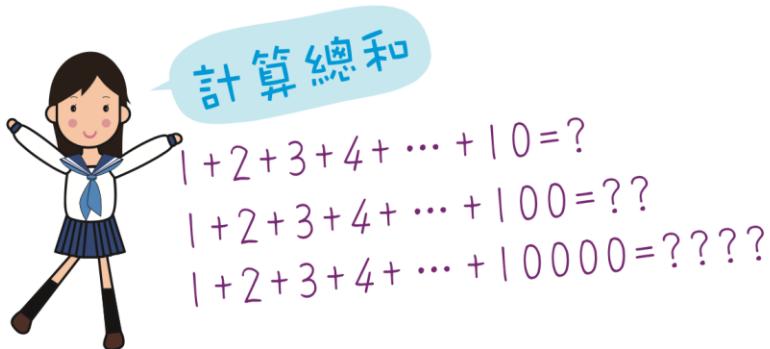
$i \quad j$
cs[1][0] cs[1][1] cs[1][2] cs[1][3]

第 5 次 → 第 6 次 → 第 7 次 → 第 8 次



重複執行 (for 求總和)

EX2-2.8d.ipynb



利用 for 迴圈、range() 函式
計算 1~10 的總和。



```
1 n = 10
2 s = 0
3 for i in range(1,n+1):
4     s = s + i
5 print(s)
```

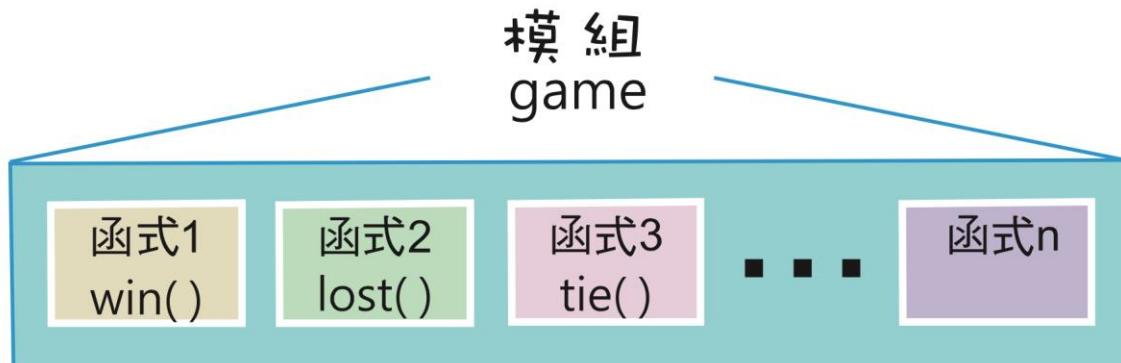
⇒ 55



2-2-9 從模組匯入更多函式

在這小節中，我們將介紹利用匯入模組的方式來使用更多的函式。Python 提供許多模組可以使用，**模組**其實就是一個 Python 程式檔案 (.py)，通常包含許多事先寫好的函式，在程式中只要匯入模組就可直接使用。

在程式中可以用 `import` 來匯入模組，例如：有一個模組名稱為 game，其中包含 `win()`、`lost()` 及 `tie()` 等函式：



匯入此模組有以下幾種方法：

1. 匯入整個模組：`import game`
2. 只匯入模組中的特定函式：`from game import win`
3. 匯入並給定模組別名：`import game as g`



匯入亂數模組

EX2-2.9a.ipynb

練習使用 random 模組裡的 randint() 函式，產生一個介於 1~10 整數亂數（包含 1 和 10）。

```
1 import random as rd  
2 x = rd.randint(1,10)  
3 print(x)
```

先匯入 random 模組，
取別名為 rd

8

以「模組名稱.函式
名稱」來使用



randint(m, n) 函式

產生 m~n 的整數亂數 (包含 m、n)





設計一個好玩的「終極密碼戰」程式，動作要求如下：

- (1) 使用 random 模組裡的 randint() 函式產生一個介於 1~10 整數亂數做為密碼。
- (2) 由玩家猜密碼值。
- (3) 玩家只能猜三次。
- (4) 猜錯時，提示太大或太小。
- (5) 猜中時就立即停止。



實作 終極密碼戰

EX2-2.9b.ipynb

```
1 import random as rd
2 x = rd.randint(1,10)
3 for i in range(3):
4     s = int(input('猜數字1-10:'))
5     if x == s:
6         print('猜中')
7         break ←
8     elif x > s:
9         print('大一點')
10    elif x < s:
11        print('小一點')
12 print('遊戲結束!')
```

見底下 TIP

☞ 猜數字1-10:5

小一點

猜數字1-10:3

小一點

猜數字1-10:2

小一點

遊戲結束!



break : 跳出迴圈

1. 在 for 迴圈內執行到 break 時，程式會立即跳出迴圈外，至 for 迴圈的下一行執行。以上例來說當第 5 行結果為 True 時，執行到第 7 行 break 時就會跳出 for 迴圈，接著執行第 12 行的 print()。

2. break 通常會搭配 if-else 使用。

