

1-2-5 重複結構的程式實作

一、for 迴圈結構

當程式是要做重複的動作時，如要將多個變數一個一個相加計算總和，可使用「重複結構」來縮短程式碼，讓程式更加簡潔容易閱讀。

**for-in
迴圈**

for 變數 in 清單：
程式區塊

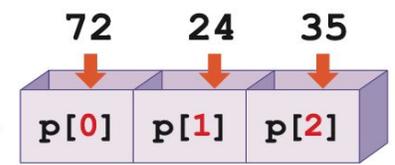
1. for 迴圈執行時會依序取出清單中的元素，當作該回合執行時的變數值
2. for 迴圈敘述的最後面要加上「:」
3. 程式區塊需「縮排」
4. 可以使用 `range()` 函式來產生指定的範圍值

假設 Eric 到超商買了三件商品，價格分別是 72 元、24 元、35 元。

● 資料：
先存放-用清單存放三件商品的價格

● 處理：
求總和-清單中有多少數字，就重複多少次的加總

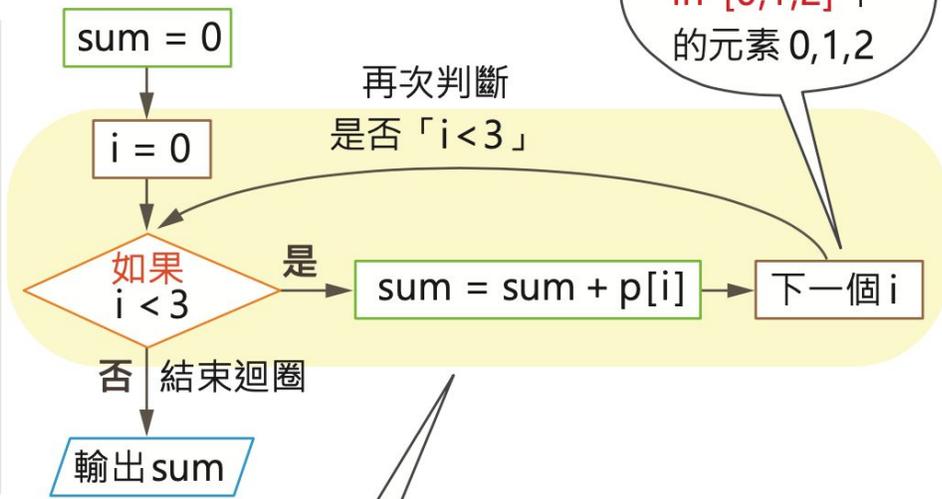
將商品價格
存入清單中



```
p = [72, 24, 35]
sum = 0
for i in [0,1,2]:
    sum = sum + p[i]
print(sum)
```

最後要加上：

for-in 迴圈



for-in 迴圈
執行 3 回合

i 初始值 = 0 sum 初始值 = 0

第 1 回合 $i = 0 \rightarrow$ 符合條件 \rightarrow $sum = 0 + 72 \rightarrow$ 下一個 $i (i = 1)$
第 2 回合 $i = 1 \rightarrow$ 符合條件 \rightarrow $sum = 0 + 72 + 24 \rightarrow$ 下一個 $i (i = 2)$
第 3 回合 $i = 2 \rightarrow$ 符合條件 \rightarrow $sum = 0 + 72 + 24 + 35$
清單元素 0, 1, 2 已用完 \rightarrow 結束迴圈

函式 range()

for-in 迴圈的變數
值除了可以用清單
來表示, 利用函式
range() 可讓程式
更加簡潔。

- range(n) : 代表整數的清單 [0, 1, ..., n-1] , 例如 : range(3) 代表清單 [0, 1, 2]
- range(m, n) : 代表整數的清單 [m, m+1, ..., n-1] , 例如 : range(1, 3) 代表清單 [1, 2]
- range(m,n,x) : x 為變化值 , 例如 : range(1, 10, 2) 代表清單 [1, 3, 5, 7, 9]

所以上圖範例的程式可以改寫成 :



```
1 p = [72, 24, 35]
2 sum = 0
3 for i in range(3):
4     sum = sum + p[i]
5 print(sum)
```

131



任務

Bob 外婆家有一個古董的報時落地鐘，在1點的時候會敲1下鐘聲、2點敲2下、3點敲3下，以此類推到12點。

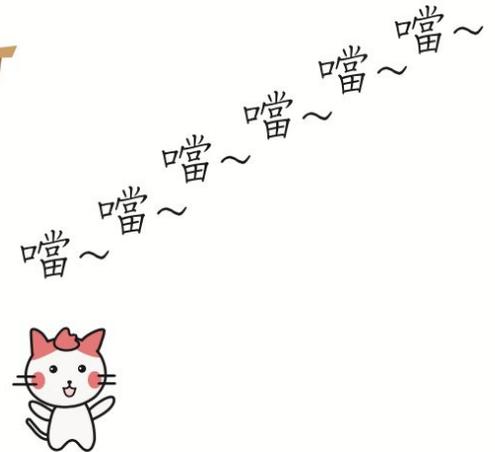
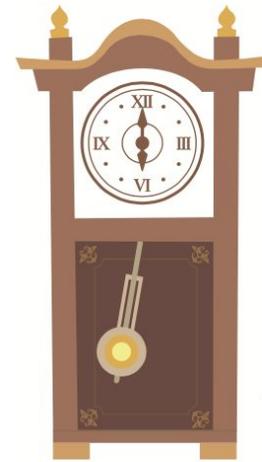
Bob 寫一程式，不使用任何數學公式，計算時鐘從1點到n點共敲了幾下。如果你是Bob，你會怎樣寫「計算 $1+2+\dots+n$ 」的程式呢？

- **輸入說明**：輸入一個正整數n

例如：10

- **輸出說明**：計算 $1+2+\dots+n$

例如：55





實例演練

加總

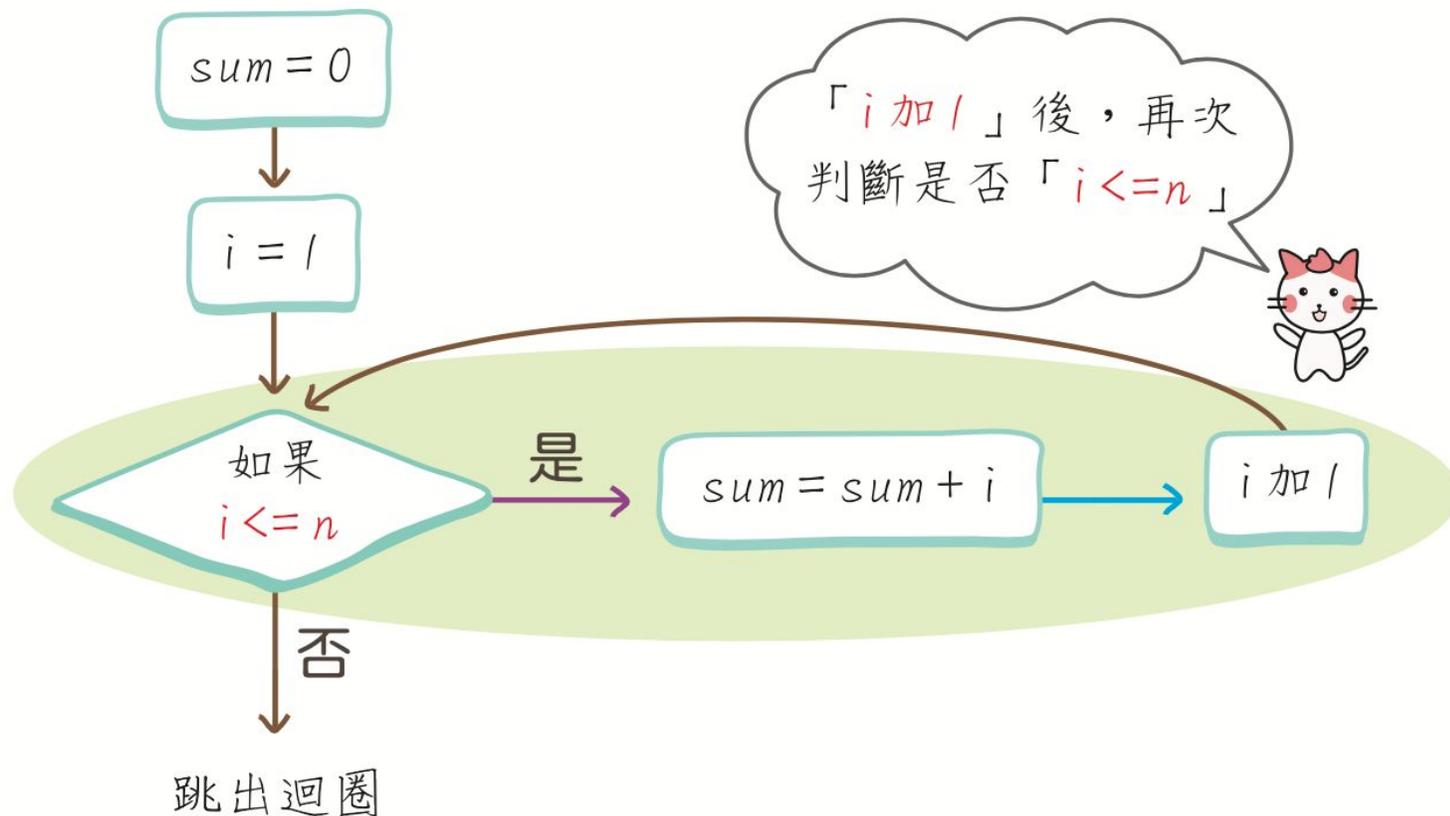
解析

```
sum = 0
```

```
for i in range(1, n+1):
```

```
    sum = sum + i
```

i 的範圍 $1 \sim n$





實例演練

加總

解析

```

sum = 0
for i in range(1, n+1):
    sum = sum + i

```

i 的範圍 1~*n*



迴圈共執行 *n* 回合

<i>i</i>	sum=0	<i>i</i> 初始值=1
1	sum=0+1	<i>i</i> =1 → 符合條件 → sum=sum+1 → <i>i</i> 加1 (<i>i</i> =2)
2	sum=0+1+2	<i>i</i> =2 → 符合條件 → sum=sum+2 → <i>i</i> 加1 (<i>i</i> =3)
3	sum=0+1+2+3	<i>i</i> =3 → 符合條件 → sum=sum+3 → <i>i</i> 加1 (<i>i</i> =4)
⋮	⋮	⋮
⋮	⋮	⋮
⋮	⋮	⋮
<i>n</i>	sum=0+1+2+(<i>n</i> -1)+ <i>n</i>	<i>i</i> = <i>n</i> → 符合條件 → sum=sum+(<i>n</i> -1) → <i>i</i> 加1 (<i>i</i> = <i>n</i> +1)
<i>n</i> +1	跳出迴圈	<i>i</i> = <i>n</i> +1 → 不符合條件 → 跳出迴圈



解析

1. 將問題拆解為幾個小問題

輸入正整數 n \rightarrow 每次加進來一個數, 採用 `for i in range(1, n+1)`, 把每次的 i 值加進去 \rightarrow 輸出總和值

2. 以 `input` 讀取最大的正整數, 配合 `int` 轉換為整數後, 存到變數 n

3. 利用 `for` 迴圈計算 $\text{sum} = 1+2+\dots+n$

```
sum=0
```

```
for i in range(1, n+1)
```

```
    sum = sum + i
```

4. 以 `print` 輸出計算結果



實例演練

加總

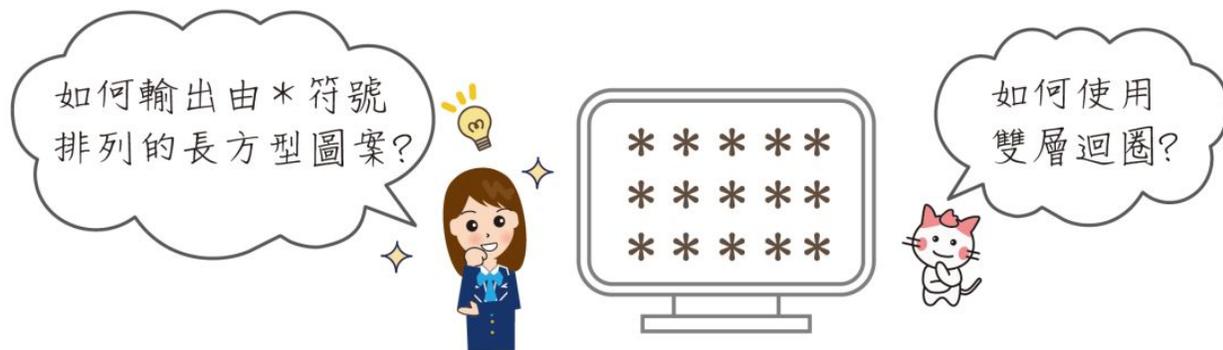
實作

程式檔名：ch1-4- 加總

```
ch1-4- 加總.ipynb - Colaborator +
colab.research.google.com/drive/1VC-HKcM0P-nMR4WRTfr97N1J9yrFelI5
+ 程式碼 + 文字
連線
1 x = input()
2 n = int(x)
3 sum = 0
4 for i in range(1, n+1):
5     sum = sum + i
6
7 print(sum)
10
55
```



二、雙層for迴圈結構



如果想輸出一個由星號構成的長方形(如右圖)

你想怎麼做呢?「先用迴圈重複印出星號5次,

即『*****』,把這樣的動作重複3次。

不過,後面這個重複3次的動作,似乎也可以用迴圈來做啊?」

搶先回答的Alice陷入思考。

於是老師給了以下的暗示:「這種情況下,可以使用雙層迴圈,

也就是迴圈內還有迴圈,如果內層迴圈的控制變數為1,2,3,4,5時,

每次輸出『*』,最終輸出『*****』。外層迴圈控制變數為1,2,3時,

每次輸出『*****』,最終會輸出什麼呢?」

二、雙層for迴圈結構

雙層for 迴圈結構

外迴圈
for 變數 in 清單： ← 外迴圈 (第1層)
內迴圈
for 變數 in 清單： ← 內迴圈 (第2層)
程式區塊



二、雙層for迴圈結構

內迴圈每次print一個*時，
不換列，記得要加上end=''
外迴圈則是要用print()換列



```
for i in range(1,4):  
    for j in range(1,6):  
        print('*',end='')  
    print()
```

外迴圈 (outer loop) is indicated by a red bracket on the left, encompassing the entire code block.
內迴圈 (inner loop) is indicated by a blue bracket on the left, encompassing the inner for loop and the print statement.
內迴圈 (inner loop) is also indicated by a blue arrow pointing to the print statement.
外迴圈 (outer loop) is indicated by a red arrow pointing to the final print statement.

內迴圈

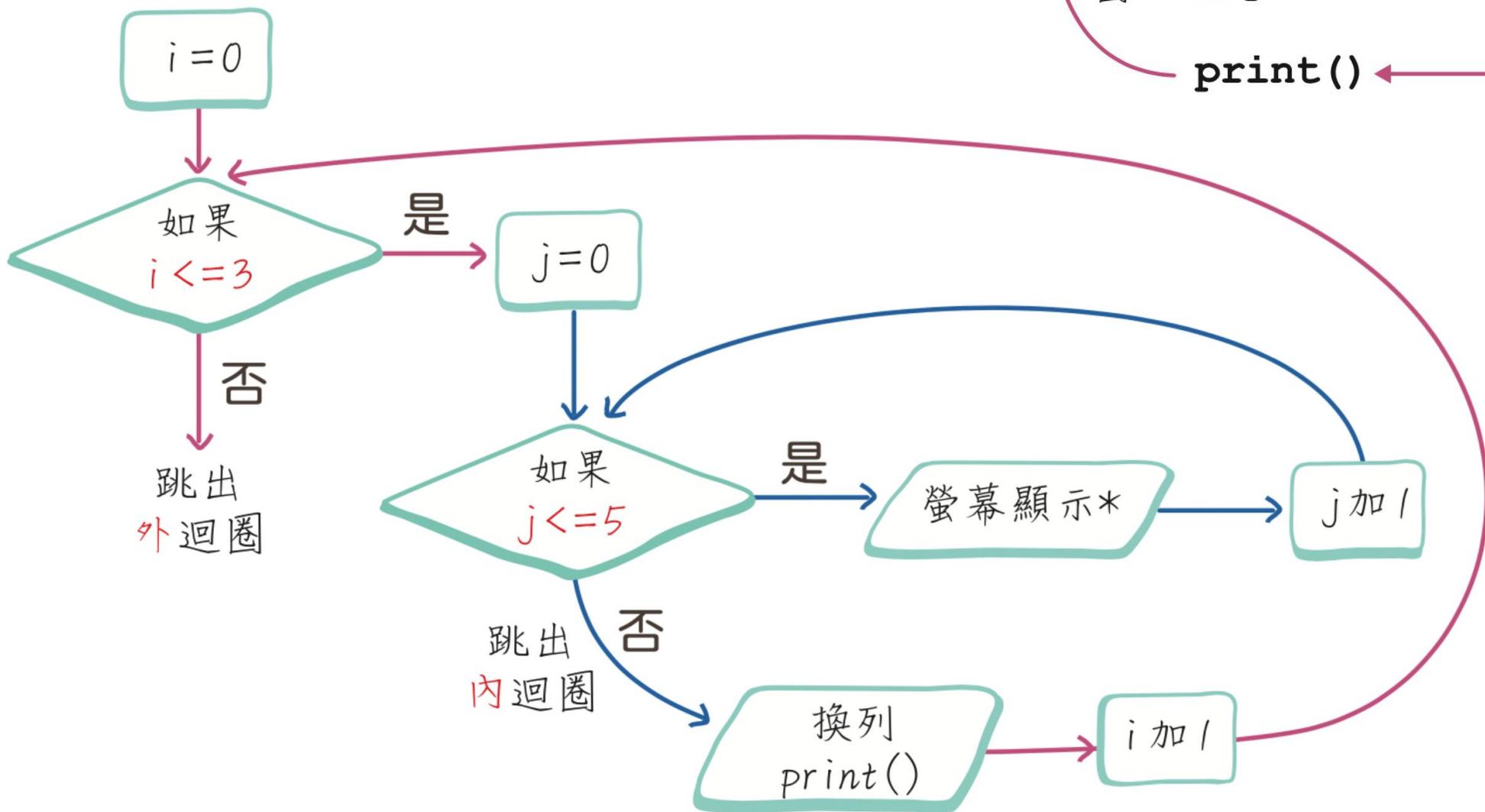
	j=1	j=2	j=3	j=4	j=5	
i=1	*	*	*	*	*	換列
i=2	*	*	*	*	*	換列
i=3	*	*	*	*	*	

外迴圈 (outer loop) is indicated by a red bracket on the left, encompassing the rows i=1, i=2, and i=3.
內迴圈 (inner loop) is indicated by blue arrows pointing from left to right across each row.

二、雙層for迴圈結構

```
for i in range(1,4):  
    for j in range(1,6):  
        print('*',end='') ← 內迴圈  
    print() ← 外迴圈
```

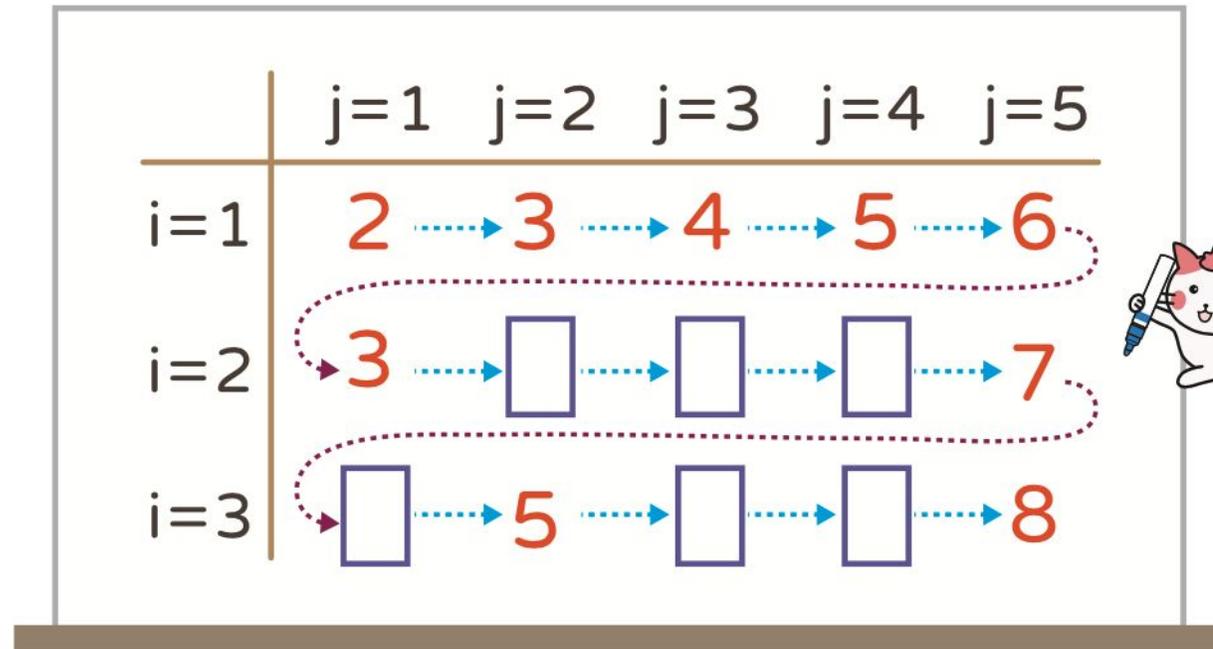
外迴圈
內迴圈



二、雙層for迴圈結構

沿著圖中由左而右的藍色虛線走，就能看出內迴圈的 j 值變化，由上而下的紫色虛線則意謂著外迴圈的 i 值變化。

```
for i in range(1,4):  
    for j in range(1,6):  
        print(i+j, end=' ')  
    print()
```



外迴圈
換一行!



實例演練

星號直角三角形

任務

Alice 最近迷上了編織，她上網找了一下各國民族服飾，意外發現這些民族服飾有一個共同的特色，就是會使用多個相同的圖案來排列組成美麗的花紋。

Alice 打算寫一個程式設計一個編織花紋，一個星號代表一個圖案。假設指定高度 n 個星號，底長 n 個星號。

- 輸入說明：輸入一個正整數 n
例如：5
- 輸出說明：
例如：右圖星號排列方式



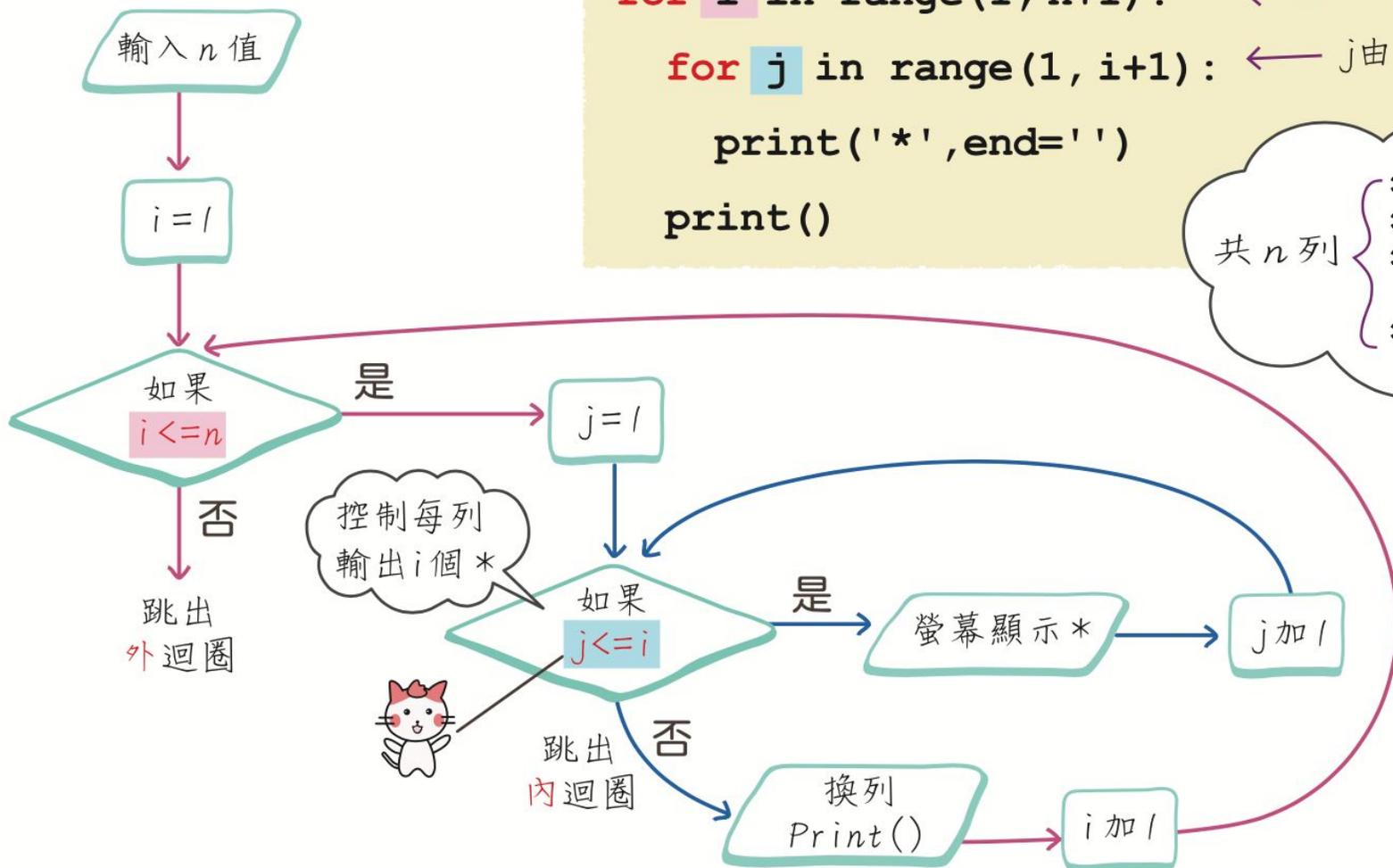
```
*  
**  
***  
****  
*****
```



實例演練

星號直角三角形

解析



```
for i in range(1, n+1):  
    for j in range(1, i+1):  
        print('*', end='')  
    print()
```

← i 由 1 ~ n, 共 n 列
← j 由 1 ~ i, 每列輸出 i 個 *

共 n 列

```
*  
**  
*** 每列 i 個 *  
⋮  
****...
```





解析

1. 將問題拆解為幾個小問題

輸入正整數 → 「共有幾列，每列輸出幾個 *」，採用雙層迴圈

2. 以 **input** 讀取正整數，再配合 **int** 轉換為整數後，存到變數 **n**

3. 共有 **n** 列，以外層迴圈 **for i in range(1,n+1)** 控制，
i 由 1 到 n 每列輸出 i 個 *，以內層迴圈 **for j in range(1,i+1)** 控制，
j 由 1 到 i

4. 內層迴圈每次 **print** 一個 * 時，不換列，因此 **print** 中要加上「**end=""**」

5. 每次內迴圈結束後，利用 **print()** 換列



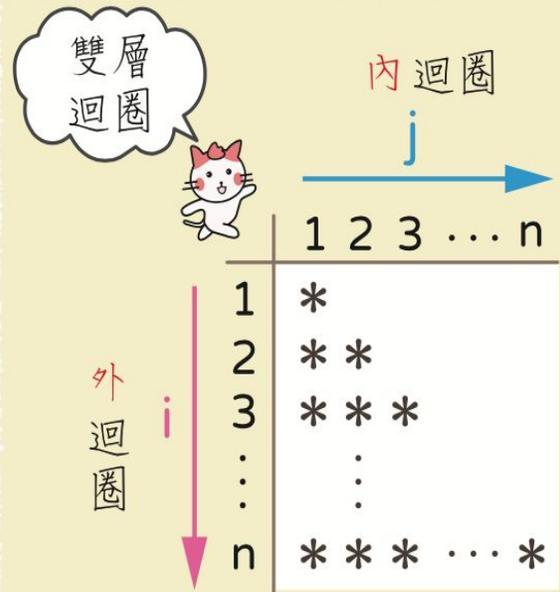
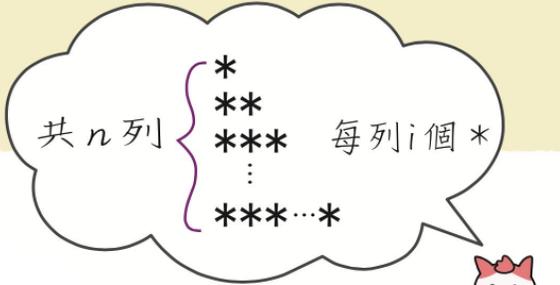
實例演練

星號直角三角形

```

for i in range(1, n+1):
    for j in range(1, i+1):
        print('*', end='')
    print()

```



```

for i in range(1, n+1):

```

$i=1$ 時，輸出 1 個 * (利用 $j=1$)

$i=2$ 時，輸出 2 個 * (利用 $j=1, 2$)

$i=3$ 時，輸出 3 個 * (利用 $j=1, 2, 3$)

...

$i=n$ 時，輸出 n 個 * (利用 $j=1, 2, 3, \dots, n$)

```

for j in range(1, i+1):

```



實例演練

星號直角三角形

實作

程式檔名：ch1-5- 星號三角形

```
1 x = input()
2 n = int(x)
3
4 for i in range(1, n+1):
5     for j in range(1, i+1):
6         print('*',end=' ')
7     print()
```

5
*
**

實力打卡

1. 請問右方程式碼的執行結果為何？

解

2. 請寫一個求階乘的程式，輸入一個正整數 n ，求出 $n!$ 。

解

```
sum = 0
for i in range(5, 21, 5):
    sum = sum + i
print(sum)
```

實力打卡

The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL `colab.research.google.com/drive/1PPrWnhR-98Jm3ZOMQ-qE5q`. The notebook title is `ch1-實力打卡-階乘.ipynb - Colab`. The interface includes a left sidebar with icons for menu, search, variables, files, and code execution. The main area contains a code cell with the following Python code:

```
1 x = input()  
2 n = int(x)  
3 prod = 1  
4 for i in range(1, n+1):  
5     prod = prod * i  
6  
7 print(prod)
```

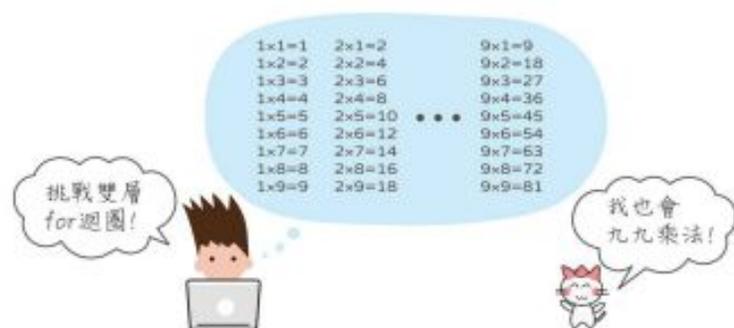
Below the code cell, the output is displayed as:

```
3  
6
```

進階挑戰題：

學習單

九九乘法表



九九乘法表是許多人在初學程式設計的時候，會用來自我挑戰的例子，就讓我們來試試看吧！

1. 先寫一程式，讓它可以輸出如右圖的結果（提示：可使用迴圈）
2. 改寫程式，讓它可以輸出左下圖的結果（提示：可以使用雙層迴圈）
3. 請進一步挑戰右下圖，修改上面的程式碼，讓它輸出九列，而且「1*1=1」之後，採用「跳位」後，才出現右邊的「1*2=2」

※ 按下 **Tab** 鍵時，游標會移到「下一個位置」，故稱為跳位鍵

```
1*1=1
1*2=2
1*3=3
1*4=4
1*5=5
1*6=6
1*7=7
1*8=8
1*9=9
```

```
1*1=1
1*2=2
1*3=3
1*4=4
1*5=5
1*6=6
1*7=7
1*8=8
1*9=9
2*1=2
2*2=4
```

```
9*6=54
9*7=63
9*8=72
9*9=81
```

```
1*1=1 1*2=2 1*3=3 1*4=4 1*5=5 1*6=6 1*7=7 1*8=8 1*9=9
2*1=2 2*2=4 2*3=6 2*4=8 2*5=10 2*6=12 2*7=14 2*8=16 2*9=18
3*1=3 3*2=6 3*3=9 3*4=12 3*5=15 3*6=18 3*7=21 3*8=24 3*9=27
4*1=4 4*2=8 4*3=12 4*4=16 4*5=20 4*6=24 4*7=28 4*8=32 4*9=36
5*1=5 5*2=10 5*3=15 5*4=20 5*5=25 5*6=30 5*7=35 5*8=40 5*9=45
6*1=6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36 6*7=42 6*8=48 6*9=54
7*1=7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49 7*8=56 7*9=63
8*1=8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64 8*9=72
9*1=9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
```

動動腦想一想

提示

1. 輸出「外層迴圈的變數 i 」，「符號 $*$ 」，「內層迴圈的變數 j 」，「符號 $=$ 」，以及「 i 與 j 的乘積」。
2. 加上跳位鍵「Tab」的用法跟換列的用法相似：
換列：`cout << endl;`
跳位：`cout << "\t";`