



程式語言與設計

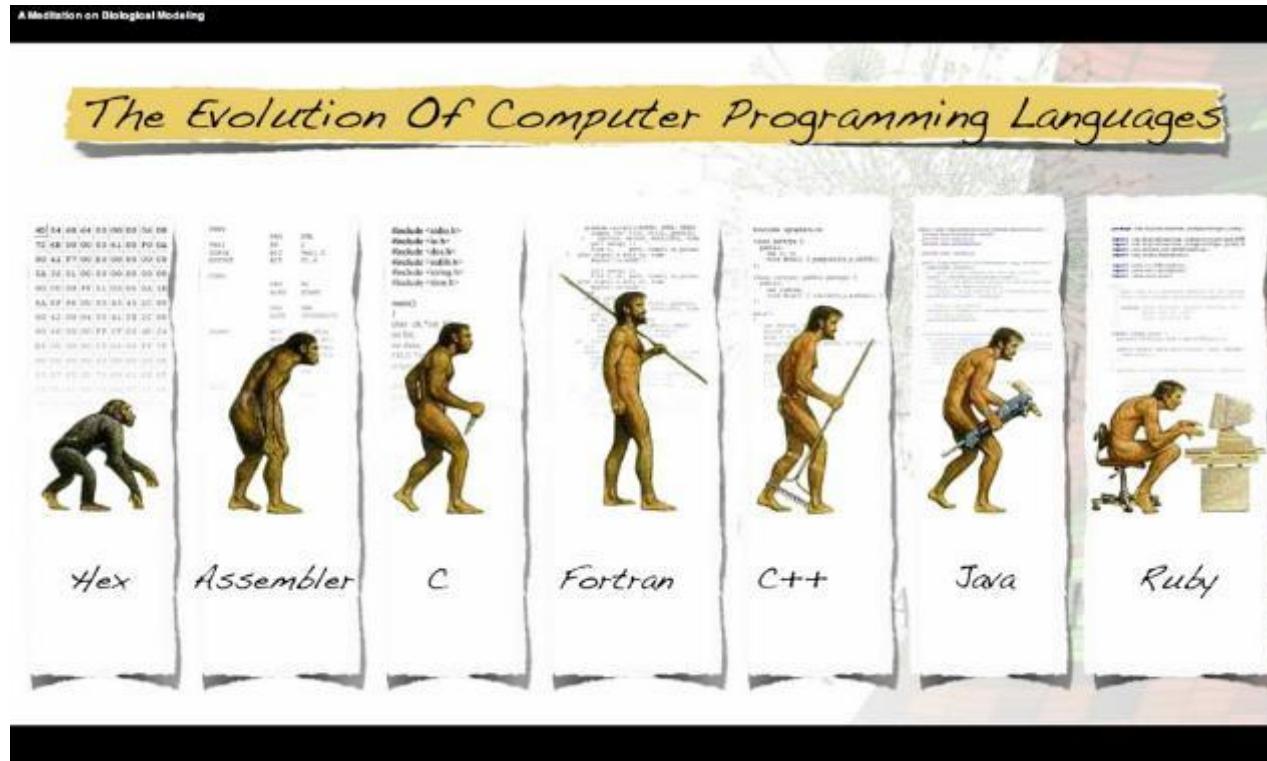
劉和師老師 part 1

2025/8/20

何謂程式語言

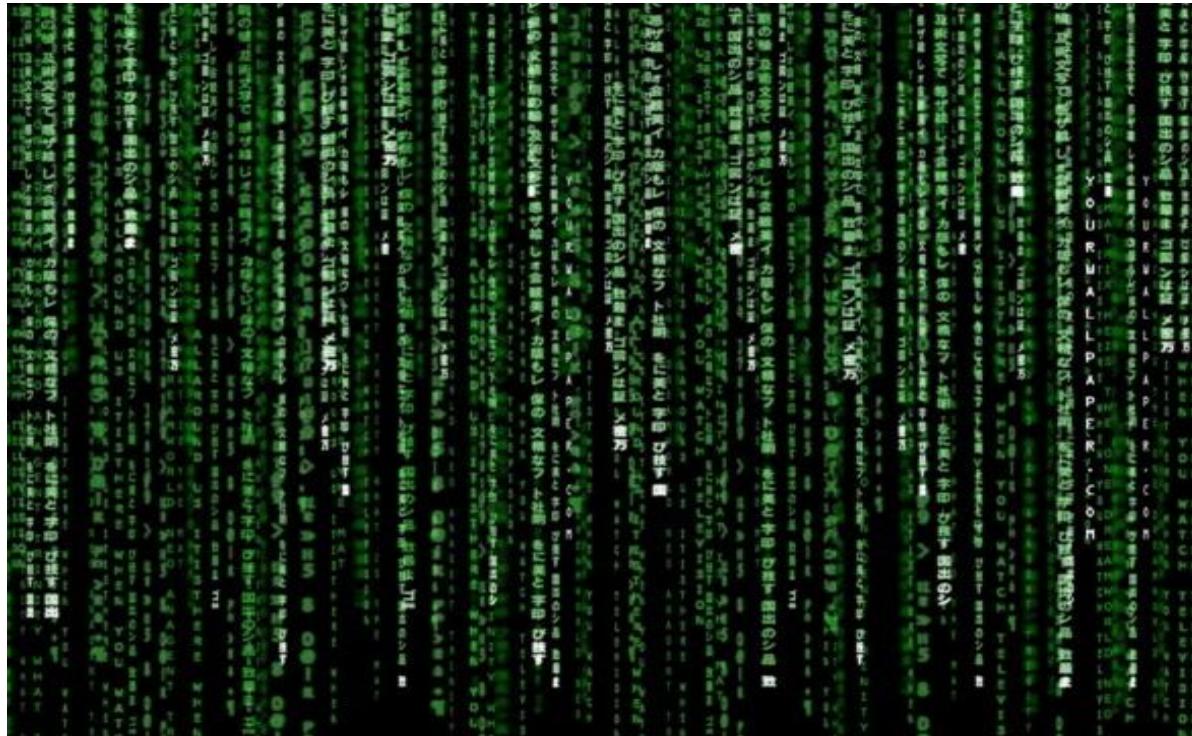
- 程式語言 (programming language)，是用來定義電腦程式的形式語言。它是一種被標準化的交流技巧，用來向電腦發出指令。

- [維基百科-Wikipedia](#)的解釋



何謂程式語言

- 程式就是指揮電腦動作的一連串指令。
- 只要用到電腦的地方，就需要程式。



(一般程式都是橫式書寫，駭客任務電影將程式以直式呈現，很有創意)

何謂程式語言

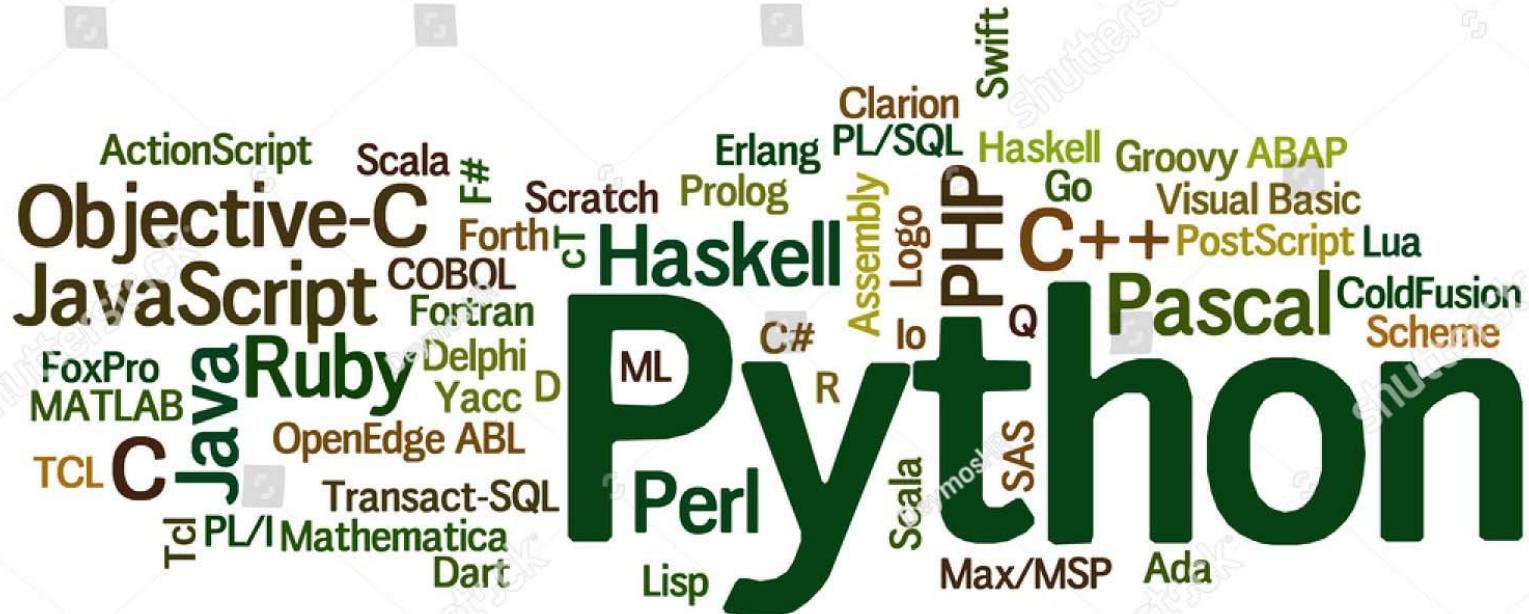
- 人的語言有中文、英文、日文等，說法不同，但都表達同一個意思。



WILLKOMMEN
歡迎
स्वागत
BIENVENIDA
WELCOME
BIENVENUE
ようこそ
добро пожаловать
ترحب
BEM-VINDO

何謂程式語言

- 程式也有多種語言，寫法也許不同，但都可以表達同一個意思。



最底層的機械語言

- 其實電腦只認得這個...
- 電腦內部是數位的世界，只認得0與1
- 這叫機械語言，但這對人類來說太困難了...

101	0111
110	1001
110	1011
110	1001
110	1001
111	0000
110	0101
110	0100
110	1001
110	0001

組合語言

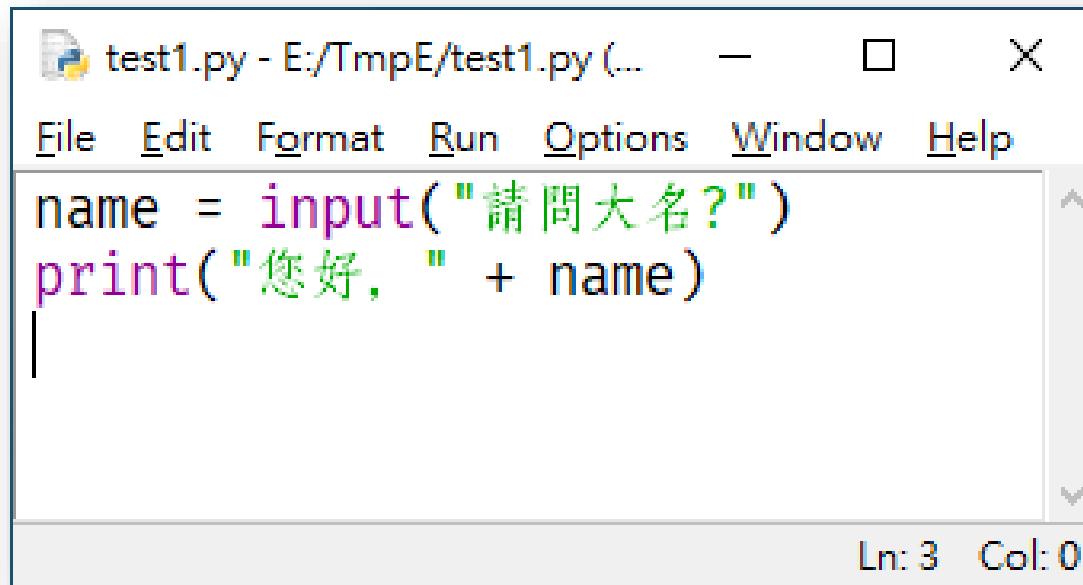
- 所以有了組合語言(Assembly)，它使用助憶碼(Mnemonics)來代替和表示特定低階機器語言的操作，硬體工程師一定要會。

- 但還是很難...

		記憶體位址	機器碼
	MOV SI, 1000H	126B : 0100	BE 00 10
	MOV DI, 2000H	126B : 0103	BF 00 20
	MOV CX, 10H	126B : 0106	B9 10 00
LOOP:	MOV AL, [SI]	126B : 0109	8A 04
	MOV [DI], AL	126B : 010B	88 05
	INC SI	126B : 010D	46
	INC DI	126B : 010E	47
	DEC CX	126B : 010F	49
	JNZ LOOP	126B : 0110	75 w
	HLT	126B : 0112	F4

高階語言

- 為了更方便與電腦溝通，能更像人與人之間在對話，所以有了高階語言。



A screenshot of a Python code editor window titled "test1.py - E:/TmpE/test1.py (...)" with the following code:

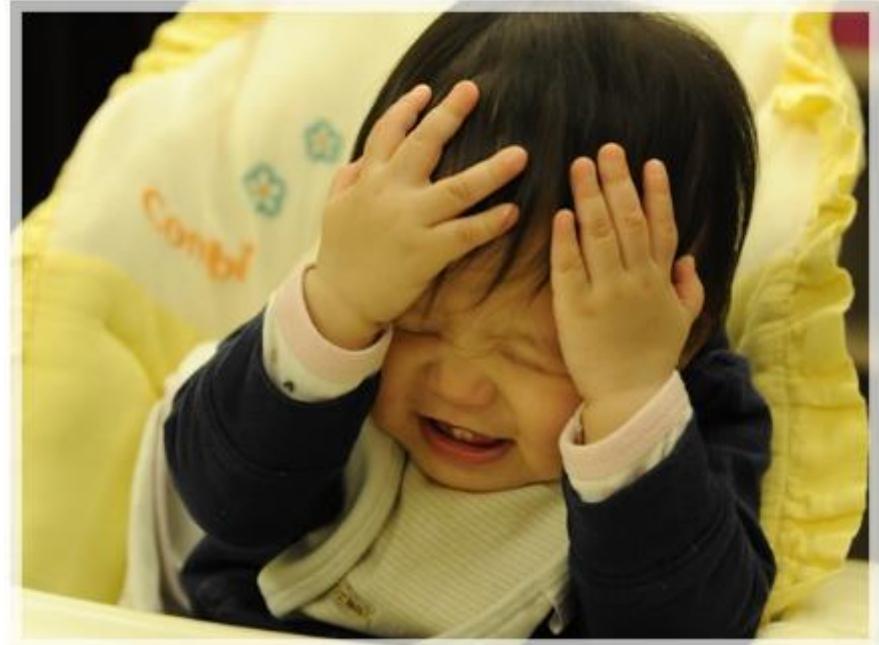
```
name = input("請問大名?")
print("您好, " + name)
```

The status bar at the bottom right shows "Ln: 3 Col: 0".

唯一的困難

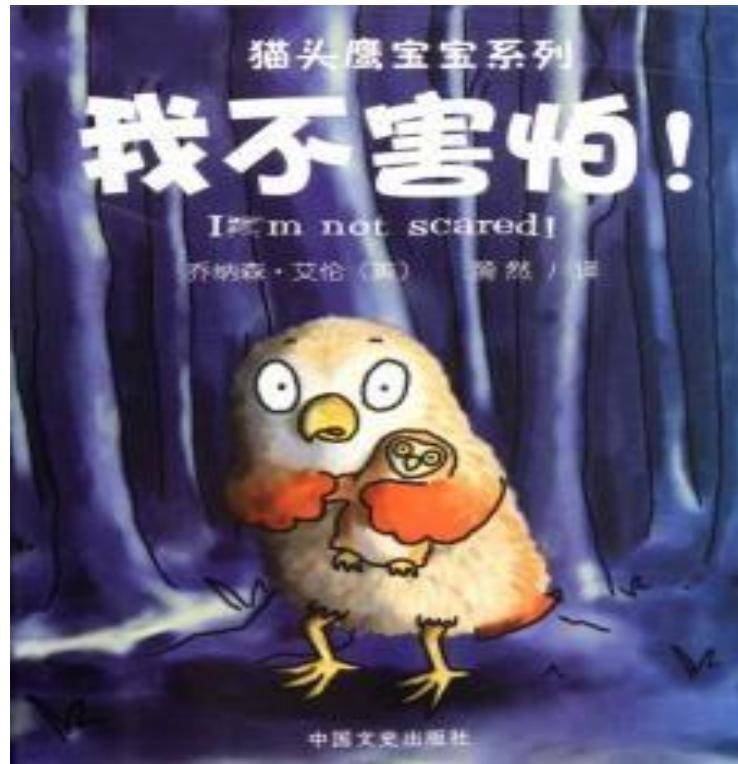
- 高階語言有很多種，應用的領域也不同，唯一相同的是，都是用...

- 英文！



我們用英文，但這不是英文課

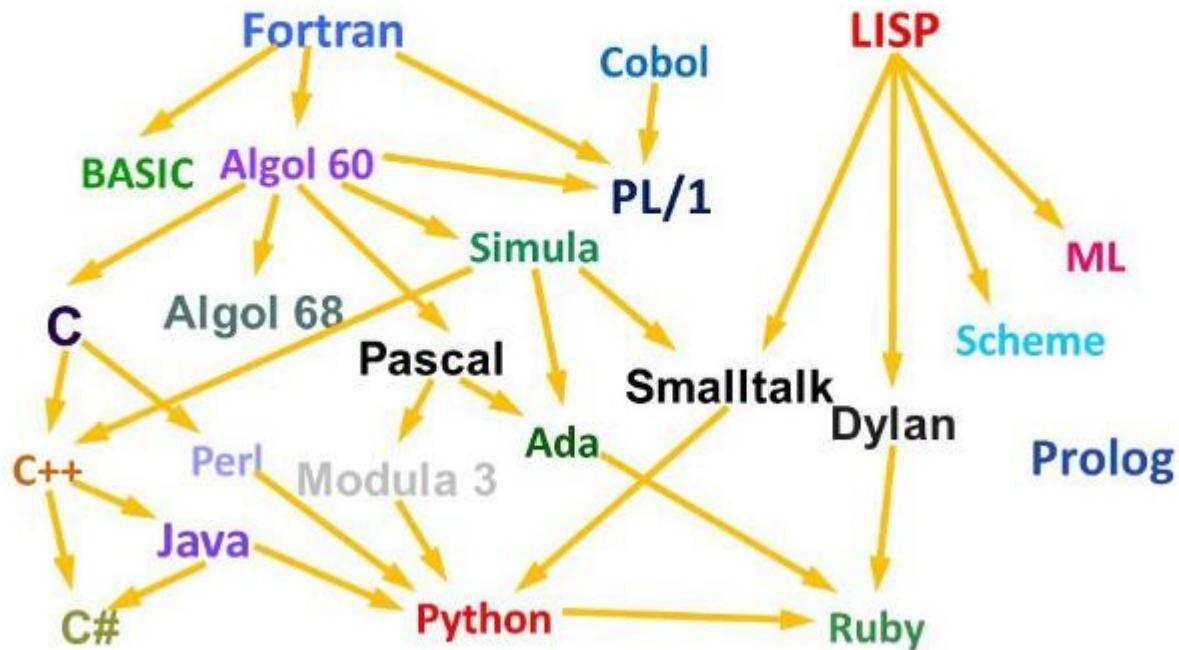
- 不用害怕，你要克服的不是英文，是心理障礙！
- 程式用到的英文很簡單，沒有甚麼過去式、未來式這些，只有簡單的命令，還有你的邏輯能力。



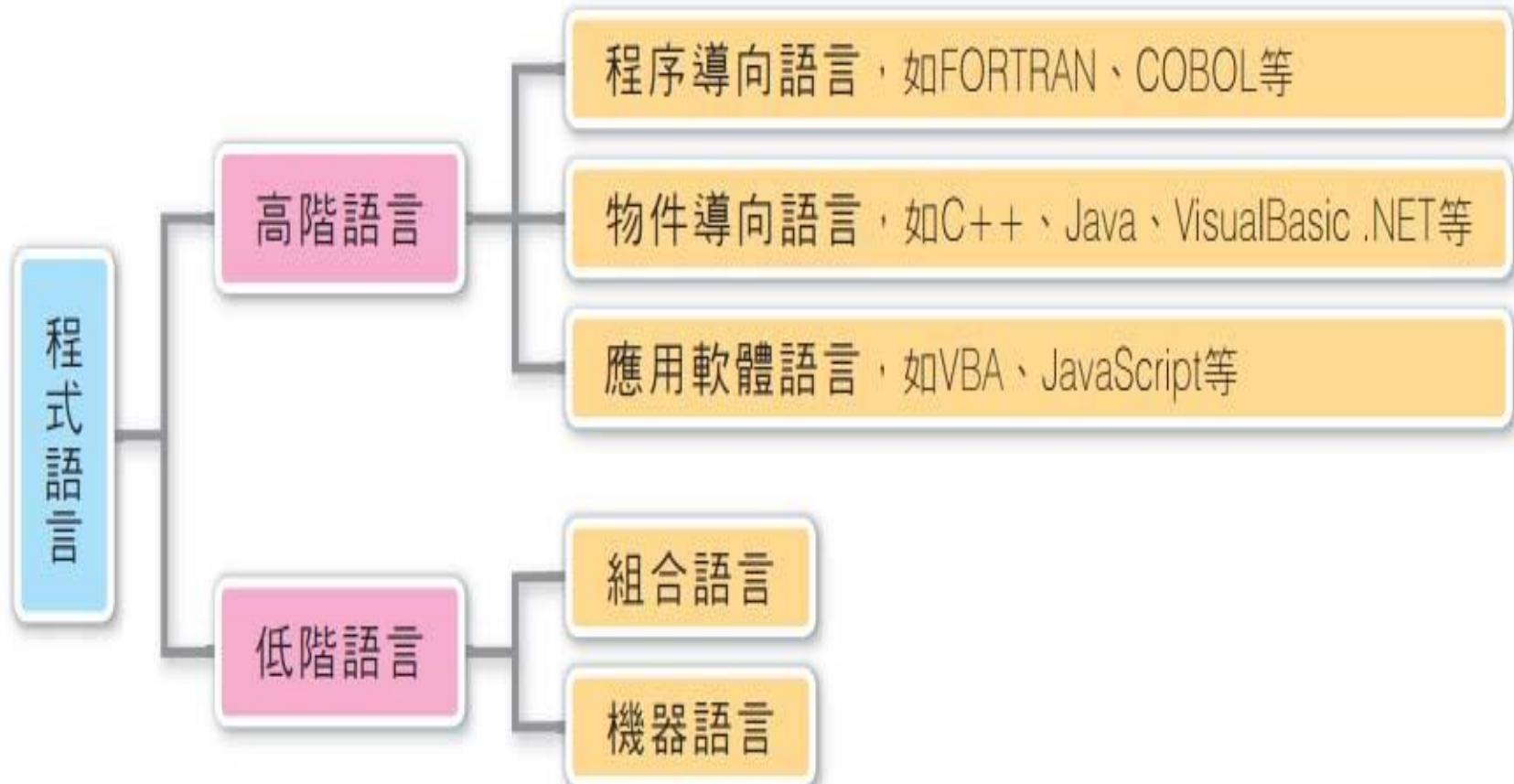
程式語言發展的歷史

A family tree of languages

Some of the 2400 + programming languages



程式語言的分類



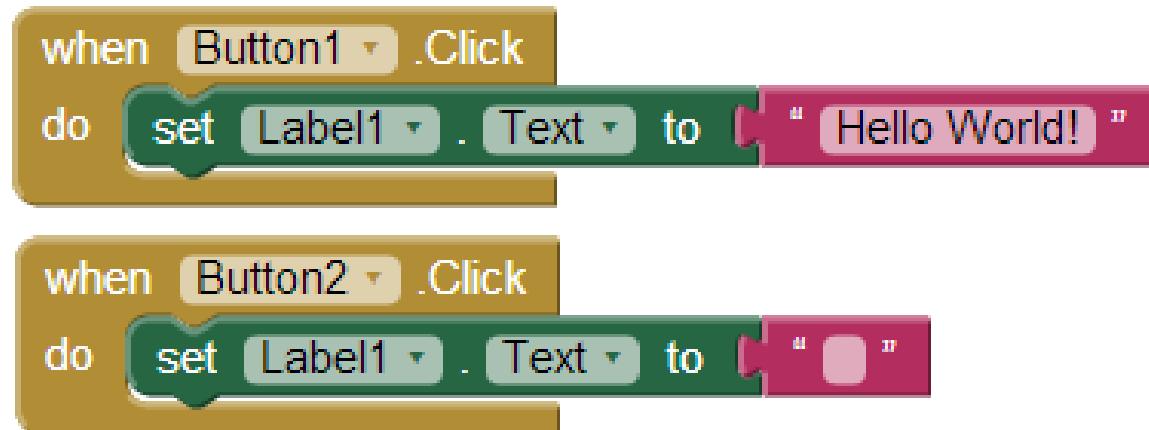
程式語言的比較

表1-1 低階語言與高階語言的比較

比較項目	機器語言	組合語言	高階語言
程式的撰寫	難 ←	→ 易	勝
維護與除錯	難 ←	→ 易	勝
可讀性	低 ←	→ 高	勝
可攜性 ^註	低 ←	→ 高	勝
執行速度	勝 快 ←	→ 慢	
佔用記憶體的空間	勝 小 ←	→ 大	

其他方式與電腦溝通

- 現在也有這種積木式語言(如App Inventor、Scratch等)，方便易學，但功能有限，尚無法做出非常複雜的東西，適合簡單快速的應用。



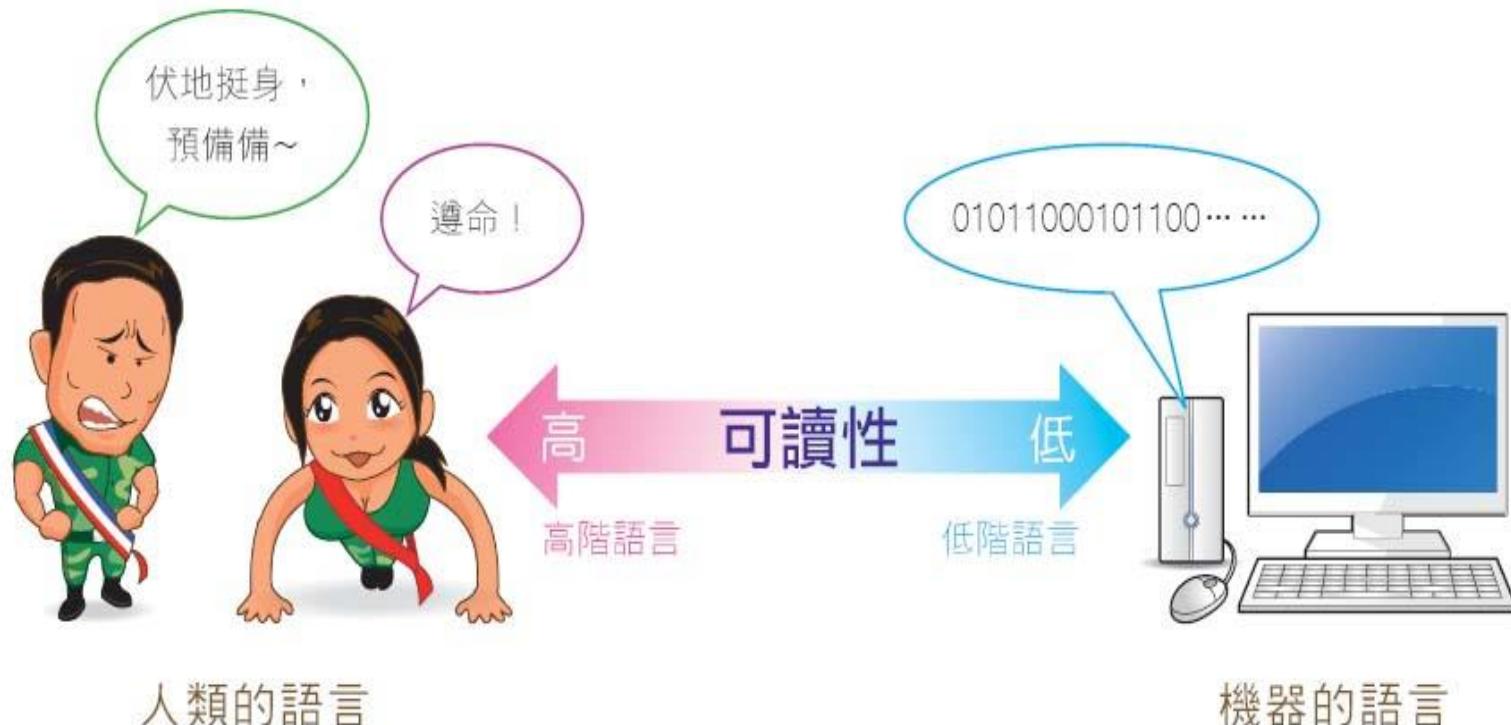
AI新世紀的到來

- 現在AI的能力都超強，可以幫我們寫程式了，但你還是要看得懂啊。



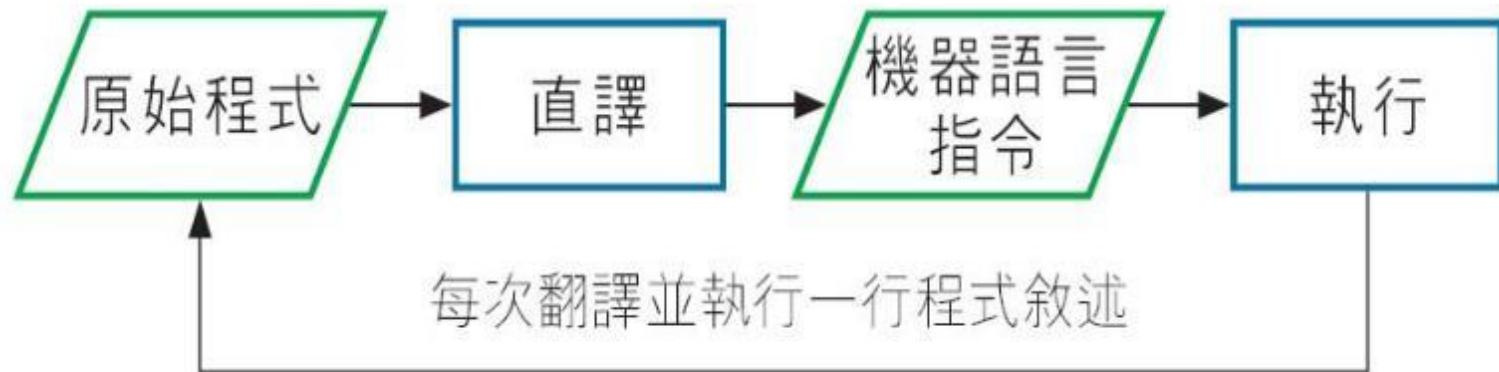
程式如何執行

- 電腦並不認得高階語言，只認得機器語言，所以不管用甚麼語言來寫程式，最後都必須轉換成010101...才行。



轉換我們寫的程式到可執行檔

- 直譯式(Interpreter)語言執行程序：

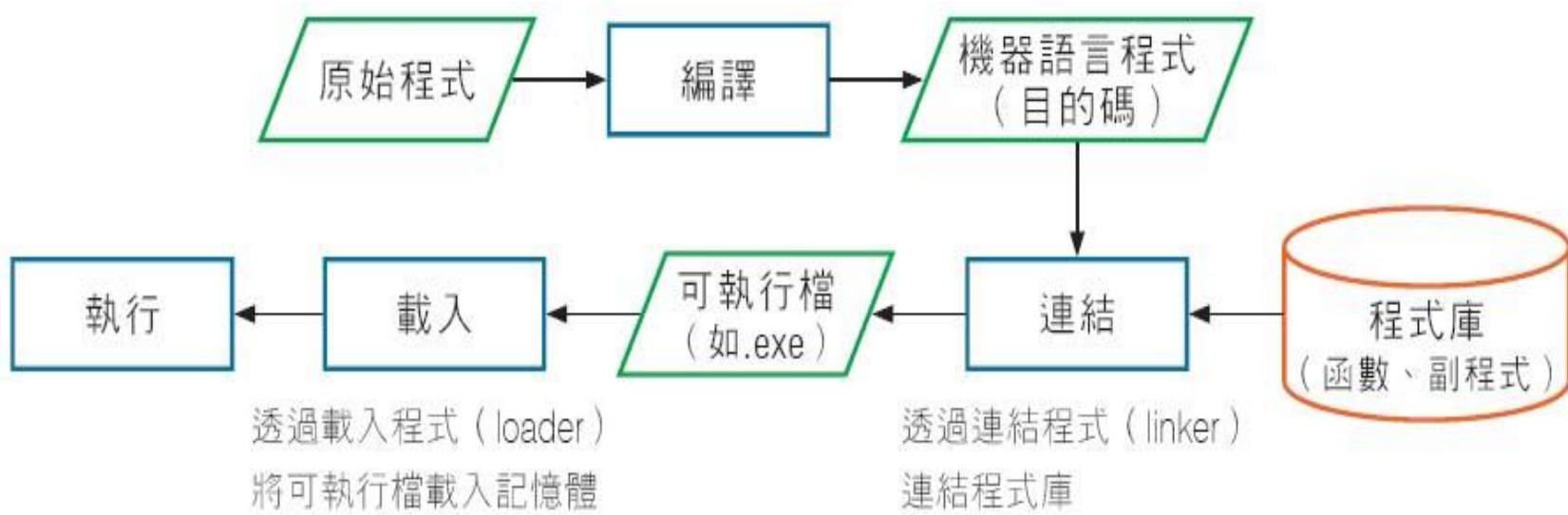


讀取下一行程式並執行，直到每一行程式都翻譯、執行完為止

- Python屬於此種執行方式。

轉換我們寫的程式到可執行檔

- 編譯式(Compiler)語言執行程序：



- 大部分高階語言屬於此種方式。

休息一下~



運算思維

- 你平時如何解決問題？
- 面對問題，你會先好好想一下嗎？
- 你的思考程序是甚麼？

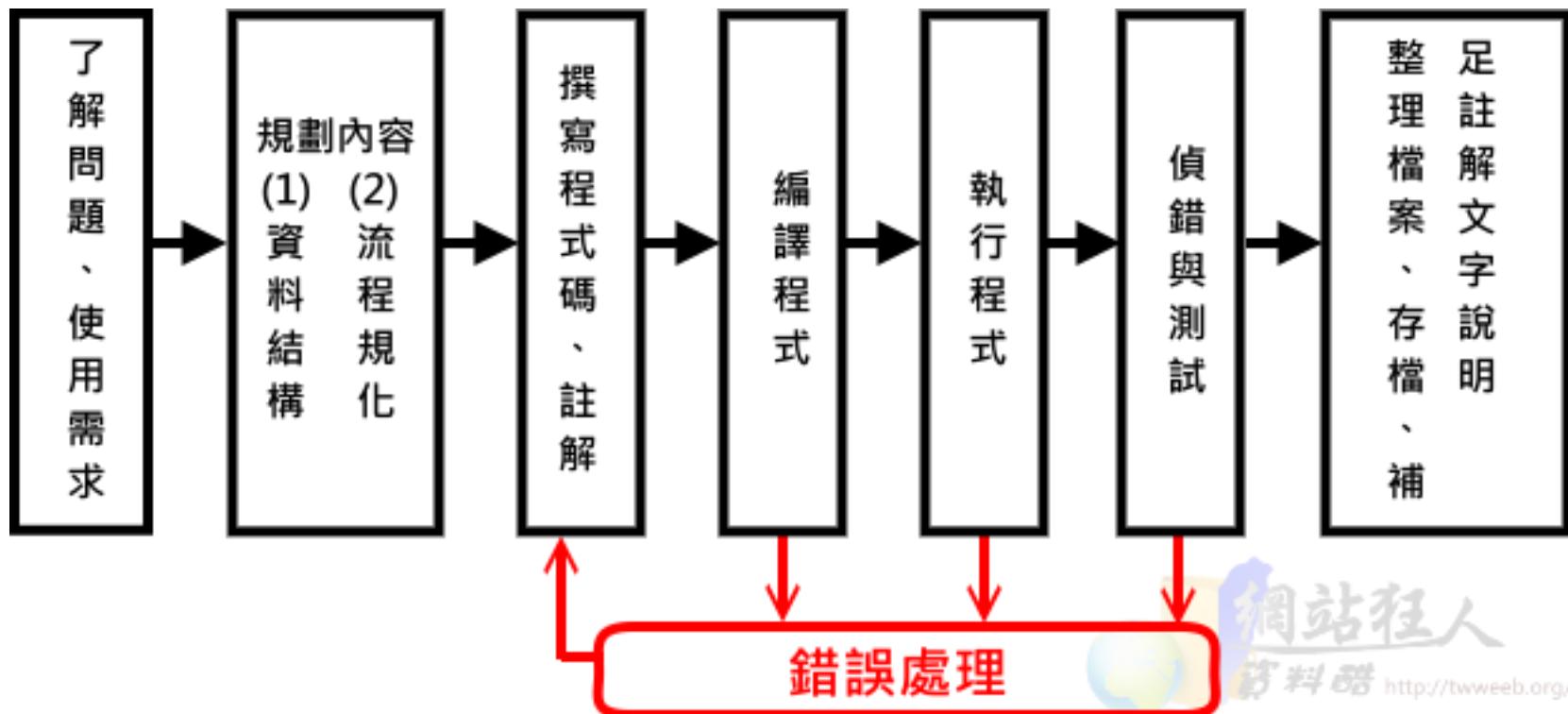


運算思維

- 運算思維是指能構思一個有條理的程序，應用各種運算方法或工具來解決問題的思維能力。
- 把你的問題分析好，再分解成一個個電腦可以幫你解決的步驟。
- 你的大腦就是一台電腦，先在大腦中跑過一遍，再去用電腦實現。

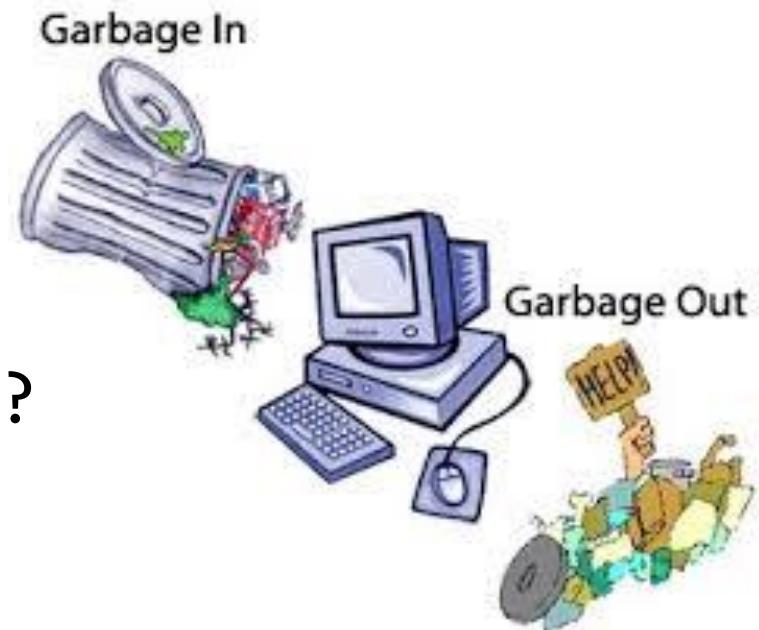
寫程式的流程

- 不是拿到問題就傻傻地坐到電腦前。
- 應該是如下圖：



先想清楚

- 垃圾進，垃圾出(Garbage In, Garbage Out，**GIGO**)。
- 先確認：
 - 你要解決甚麼問題？
 - 需要輸入甚麼？
 - 需要輸出甚麼？
 - 要用哪一種工具(語言)？



思考過程

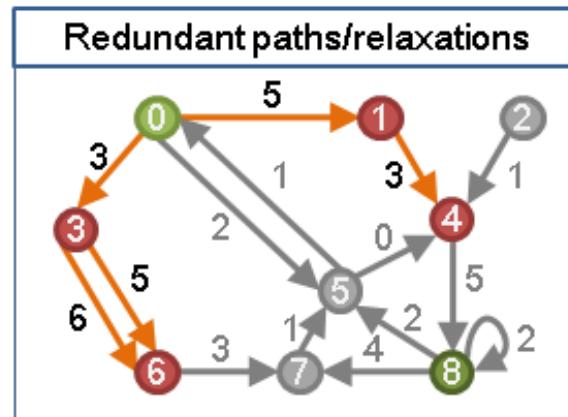
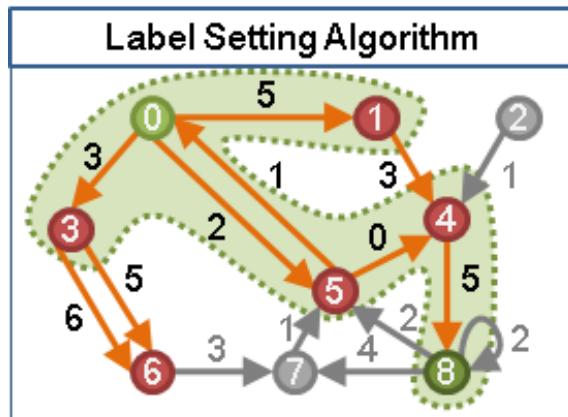
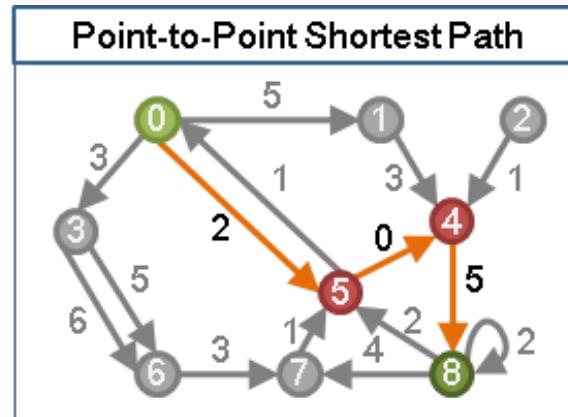
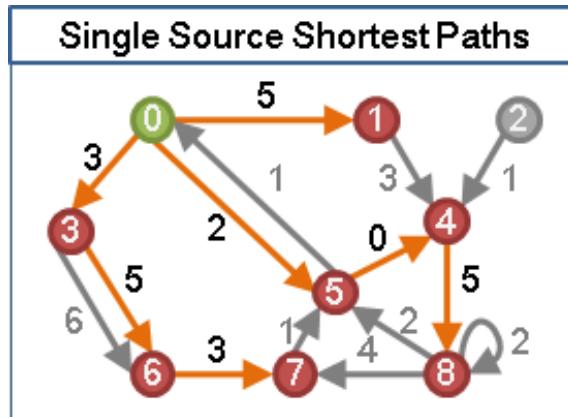
- 一開始總是沒有頭緒的，先靜下來分析你要解決的問題。
- (假設有這麼複雜...)

(示意圖)

$$\begin{aligned}
 \bar{y} &= \frac{\alpha}{2} \tilde{x} + \frac{q}{2} \in \frac{x}{\alpha} \Rightarrow \frac{dy}{dx} = \frac{1}{2} \sqrt{\text{non}}^3 / \frac{x}{y} = \sqrt{P7} \text{ to the 4th power} \quad \boxed{43/23 \text{ Dhamma}} \\
 \frac{7}{8} &= (B)^{-7/10} \div \sqrt{4} \pi \text{ add bear } \frac{1}{2} \sqrt{481^2 / 2} \geq n \geq 3 + \frac{\text{Whiskerbert}}{90,000} \\
 9 + \frac{1}{2} &= y + 11 \times 22 + 8? \quad \langle \tilde{a} \tilde{x} \rangle! \frac{1}{6} < \text{DIAMETER} \quad \frac{2+2=5=\pi}{2} \\
 \bar{z} N &\geq \text{beard } \sum_{n=0}^{\infty} -57 \quad \text{erino} + \frac{1}{2} \text{ folliculus } \frac{3}{224} \left(\frac{n}{2}\right)^8 / 10 (\sqrt{95}) \\
 (47)^{-6/32} &= 14261 (135-n) 8 \quad \text{wise} + \text{acre} = \left(\frac{a}{10}\right) 2 \sqrt{48}(1) + \text{red heart} \quad \left(\frac{1}{R} + \frac{1}{92}\right) \frac{22}{203} \left\{ \frac{40}{87} \right\} 1 \sqrt{85} / 125 \left(\frac{65}{227}\right) \frac{68}{913} = 4 \\
 &= \left(\frac{1}{R} + \frac{1}{92}\right) \frac{22}{203} \left\{ \frac{40}{87} \right\} 1 \sqrt{85} / 125 \left(\frac{65}{227}\right) \frac{68}{913} = 4 \\
) z &\geq 7/108 \quad \frac{75^\circ}{B} < \frac{626}{7} > (C) \quad 148/199 P=55 \left(\frac{6}{275}\right) + \frac{\pi}{z \text{ axis}} \phi \sqrt{319} \pi \quad \frac{4500}{\pi} + 2 \\
 \frac{2}{3} b &= f + (n-1) = f + \frac{1}{mac} = f + (x-22) = \text{far} (?) \quad z+y=c \leq z \quad 9 + \text{de } 11^{\text{th}} \text{ Pow} \\
 \frac{2}{3} \sqrt{a} &= \frac{nR}{a} \text{ when } \frac{K1e}{hr} \quad \frac{48000}{(c/z)} \left\{ \frac{45}{95^\circ} \right\} \sqrt{9500} (4) + \left(\frac{9}{10}\right) = (\sqrt{9361} \left(-\frac{2}{3}\right)) \\
 &= 857 + \left(\frac{8}{99}\right)^0 - \frac{47}{85} \sqrt{9/10} \div \left(\frac{c}{z}\right) \frac{48000}{\text{Q}} \left\{ \frac{45}{95^\circ} \right\} 24909 \text{ stachio } 1/10 \text{ of axis } \phi \\
 \frac{7}{18} \cdot 33^\circ/n &= \frac{89}{9000} \text{ pores} \quad x \text{ will } 4 \\
 \sqrt{(9/1000)} &< 4 + \frac{925}{9983} \left(\frac{\text{eldest}}{\text{mother}} \right) + \frac{\pi}{148} \quad 9/10 (-) \quad 1/10 \geq \left(\frac{\text{eldest}}{\text{mother}} \right) 45/100 \left(\frac{4}{3}\right) + \sqrt{95619}
 \end{aligned}$$

思考過程

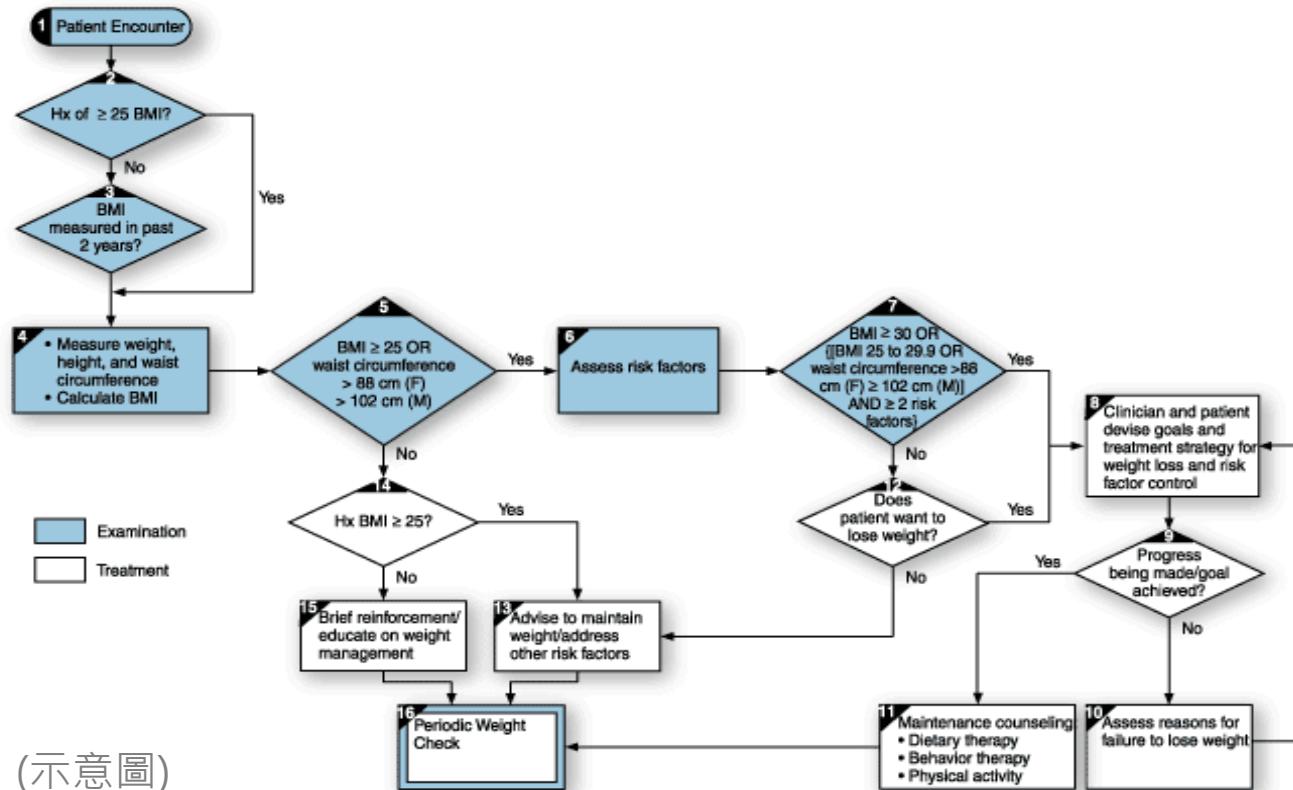
- 抽象化和模型化，找出規則、規律、區塊等等。



(示意圖)

思考過程

- 規劃成電腦可執行的步驟。



* This algorithm applies only to the assessment for overweight and obesity and subsequent decisions based on that assessment. It does not include any initial overall assessment for cardiovascular risk factors or diseases that are indicated.

思考過程

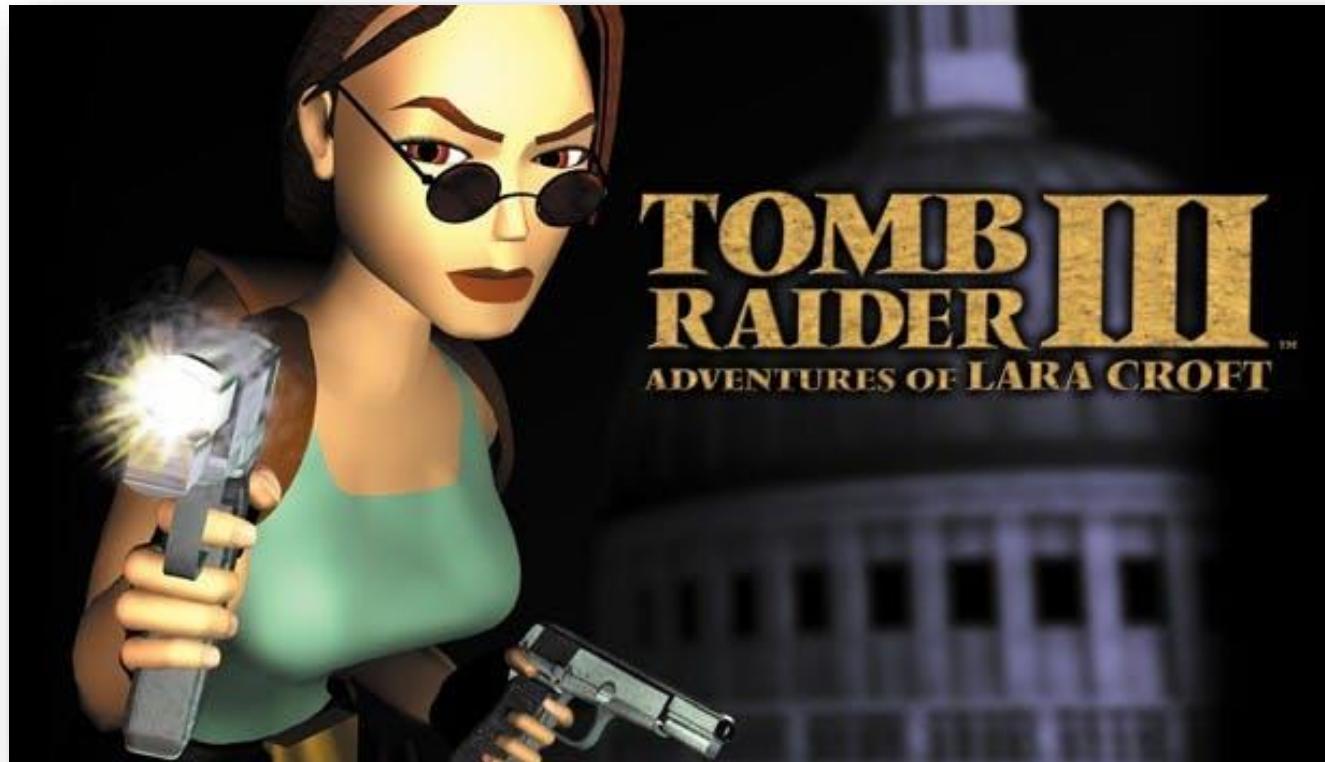
- 寫程式(Coding)

```
1  def selection(Array,i,N):
2      index = i
3      k = i+1
4      while k < N:
5          if Array[k] < Array[index]:
6              index = k
7              k += 1
8      if index != i:
9          Array[i],Array[index] = Array[index],Array[i]
10
11 #main
12 A = []
13 N = int(input("請輸入陣列大小:"))
14
15 for i in range(N):
16     x = int(input("請輸入第"+str(i)+"個數字:"))
17     A.append(x)
18
19 for i in range(N):
20     print(A)
21     selection(A,i,N)
22     print()
23
24 print(A)
```

(示意圖)

思考過程

- 欣賞及確認執行成果。

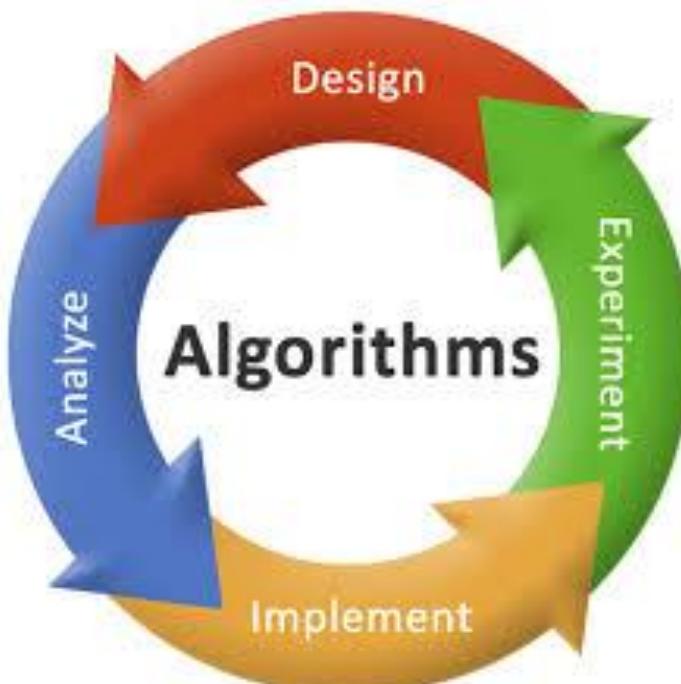


(示意圖)

- (要知道所有的遊戲都是程式寫出來的唷~)

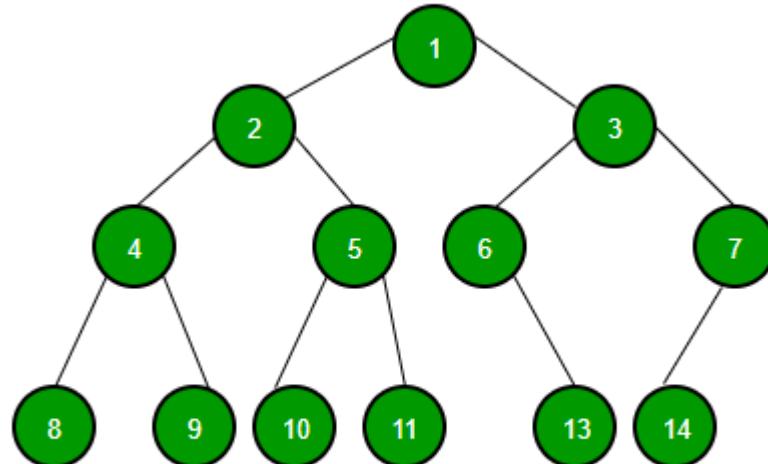
演算法(Algorithms)

- 把問題抽象化和模型化，找出合理有效的解決步驟。
- 就是解決問題的方法啦~



演算法(Algorithms)

- 面對不同類型的問題，使用適當的資料結構(Data Structure)及演算法可以讓問題簡單很多。
- 基本的資料結構例如：
 - 陣列(Array)
 - 鏈結串列(Link List)
 - 堆疊(Stack)
 - 佇列(Queue)
 - 樹(Tree)
 - 圖(Graph)



演算法(Algorithms)

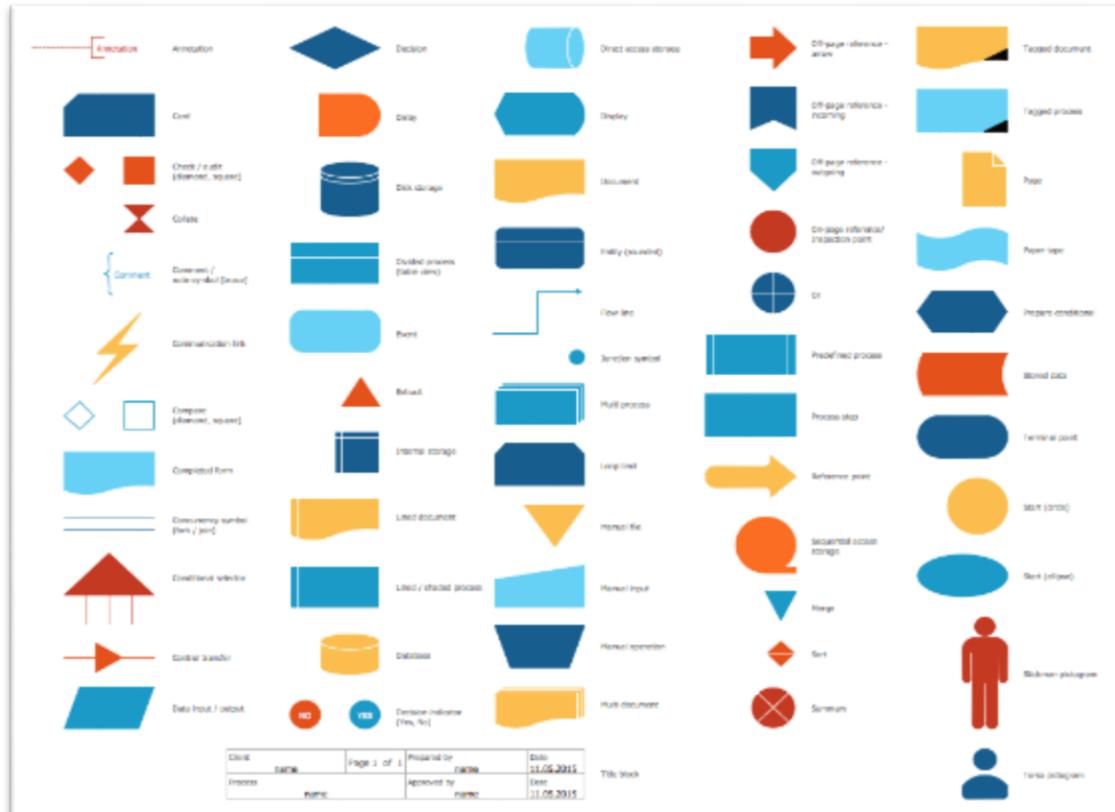
- 一個好的演算法要求：
 - **輸入**：有零個或一個以上輸入。
 - **輸出**：應有一個或一個以上輸出。
 - **明確性**：演算法的描述必須無歧義，以保證演算法的實際執行結果是精確地符合要求或期望，通常要求實際執行結果是確定的。
 - **有限性**：必須在有限個步驟內完成任務。
 - **有效性**：又稱可行性。能夠實現，。

"Good job!"



演算法的工具

- 流程圖(Flow Chart)是個不錯的方式，用簡單的圖形來代表執行的步驟。



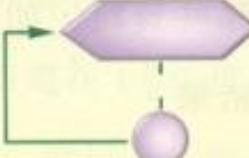
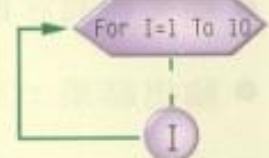
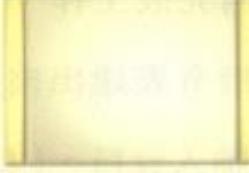
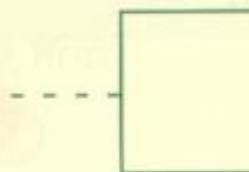
常用流程圖

- 圖例：

名稱	符號	意義	範例
1.起止符號		表示程式的開始或結束	開始 結束
2.流程符號		表示流程進行的方向	
3.輸入/輸出符號		表示資料之輸入或結果的輸出	輸入 A、B 之值 顯示出總和
4.處理符號		表示執行或處理某些工作	$S=S+A$ $A\$= "XYZ"$
5.決策判斷符號		表示對某一個條件做判斷	否 是
6.連接符號		用於： 1.轉接到另一頁 2.避免流線交叉 3.避免流線太長	

常用流程圖

- 圖例：

名稱	符號	意義	範例
7.迴圈符號		表示程式迴圈控制變數初值及終值的假設	
8.副程式 符號		表示一群程式步驟或流程，用以說明副程式或其他流程的組合	利息處理 副程式
9.報表符號		表示以列表機印出報表文件	學生成績單
10.註解符號		表示對某一流程加以註解	S = 總和 A = 加數

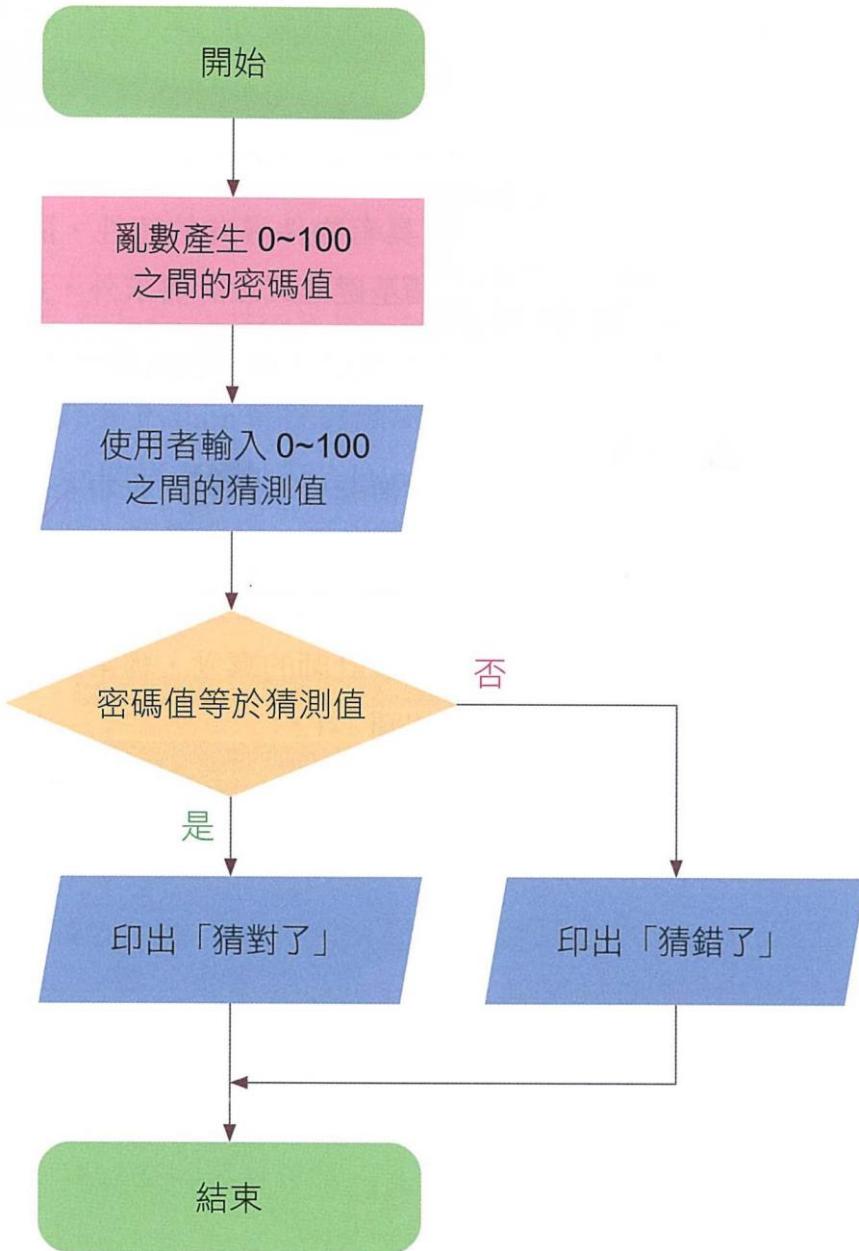
常用流程圖

- 對於簡單的問題，其實記住這四個也就夠了。

名稱	符號圖形	意義
開始或結束符號		表示流程的開始或結束。
流程符號		表示程式流程進行的方向。
程序處理符號		表示要進行的處理工作。
輸入或輸出符號		表示資料輸入或結果輸出。
決策判斷符號		根據條件式來判斷程式進行方向。

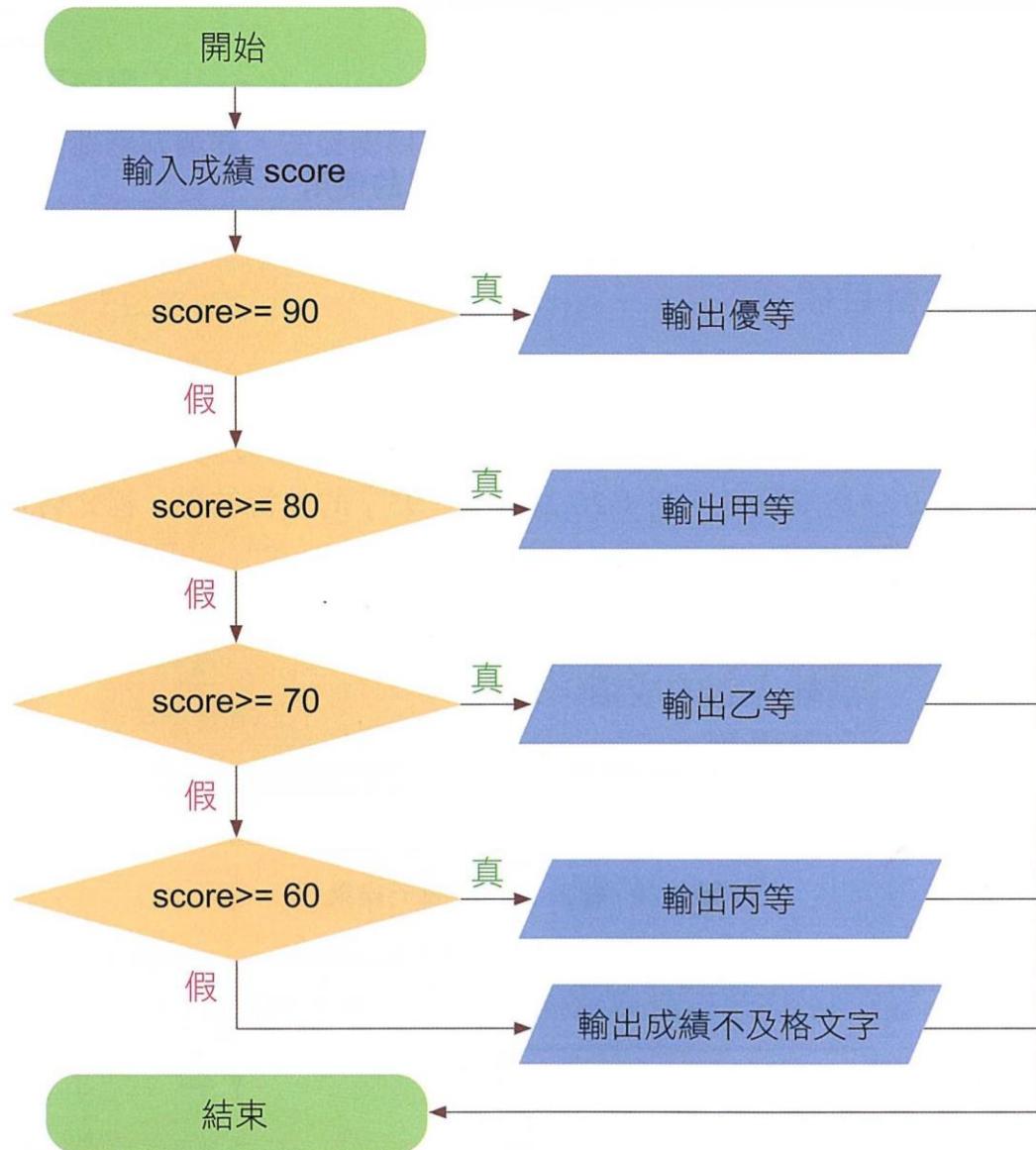
流程圖範例

一個猜密碼的 流程圖



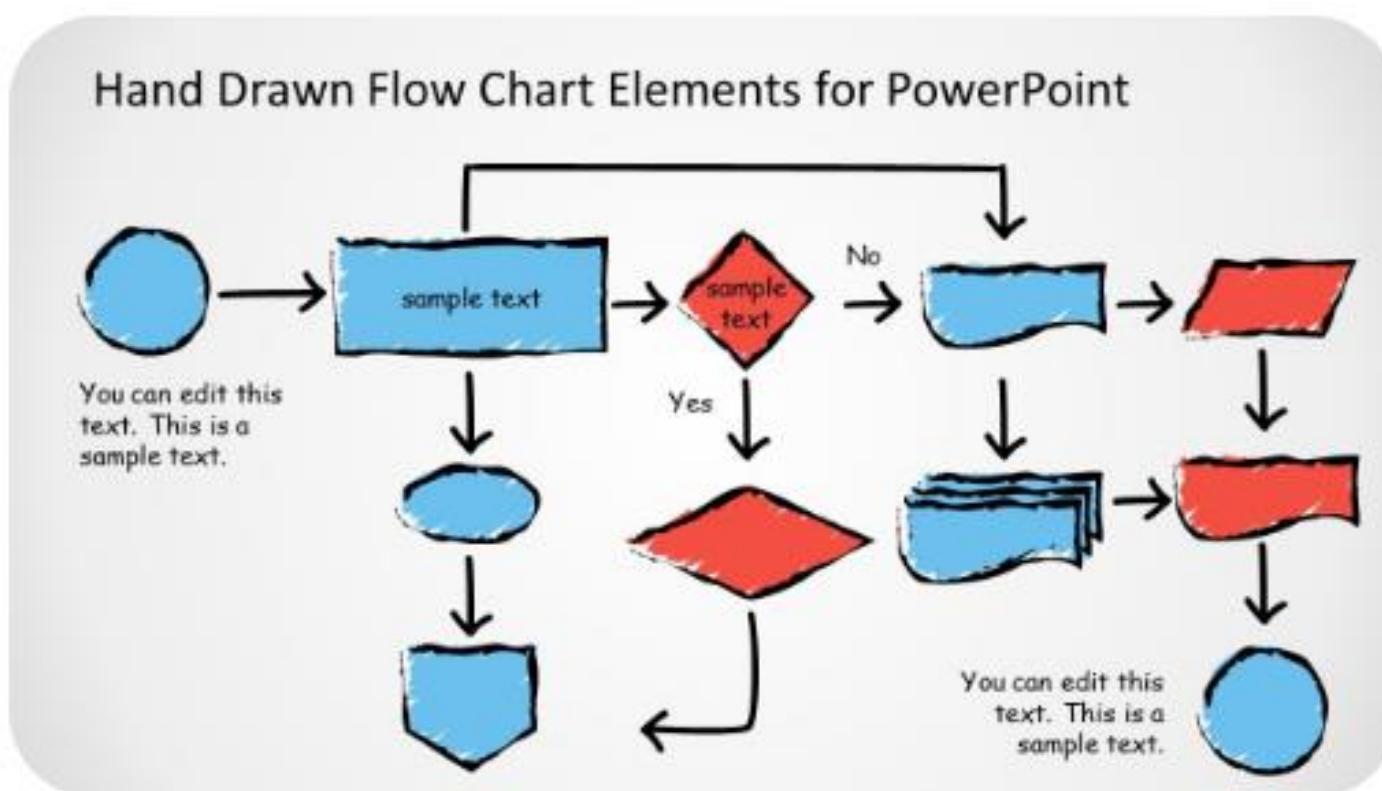
流程圖範例

將成績轉換成等第的流程圖



畫流程圖的工具

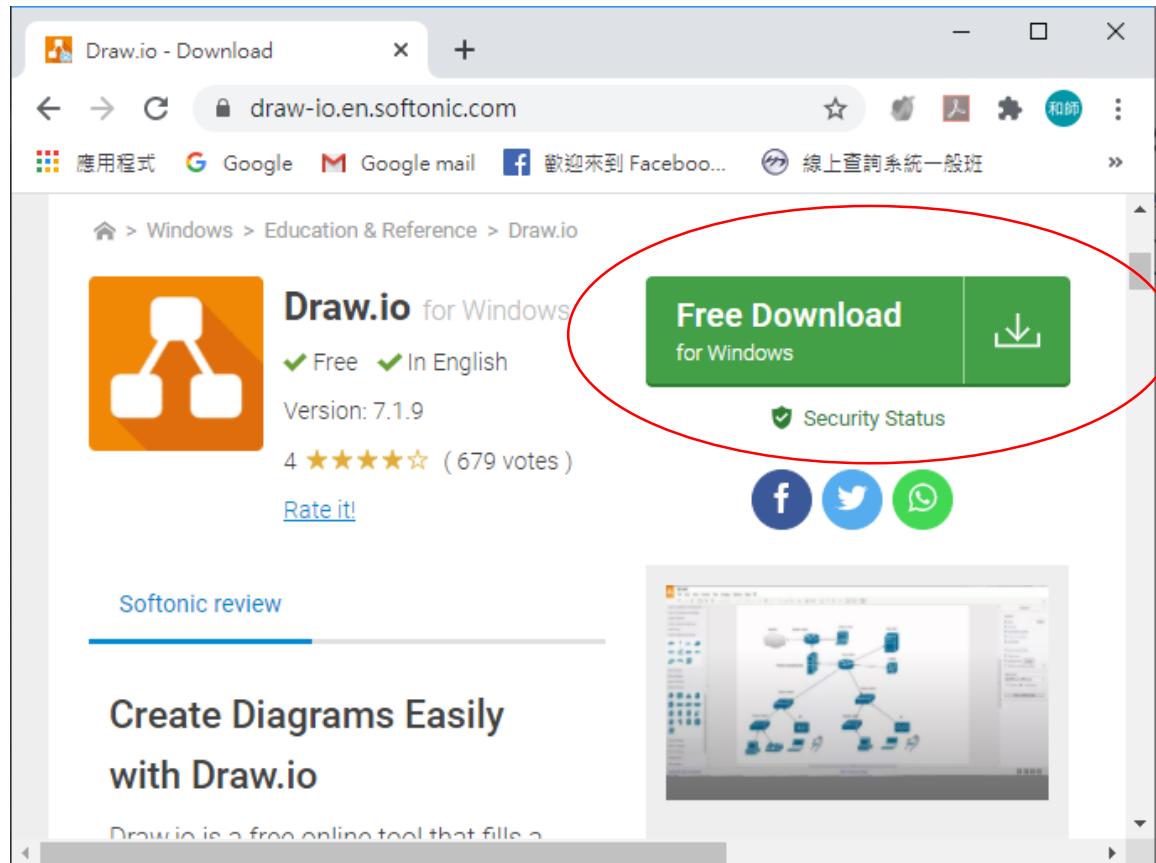
- 問題不複雜的話用手繪就行了。



(這位外國老兄其實畫得不錯)

畫流程圖的工具

- 如果需要精美或較複雜的應用，Draw.io是個不錯又免費的畫流程圖的工具軟體。

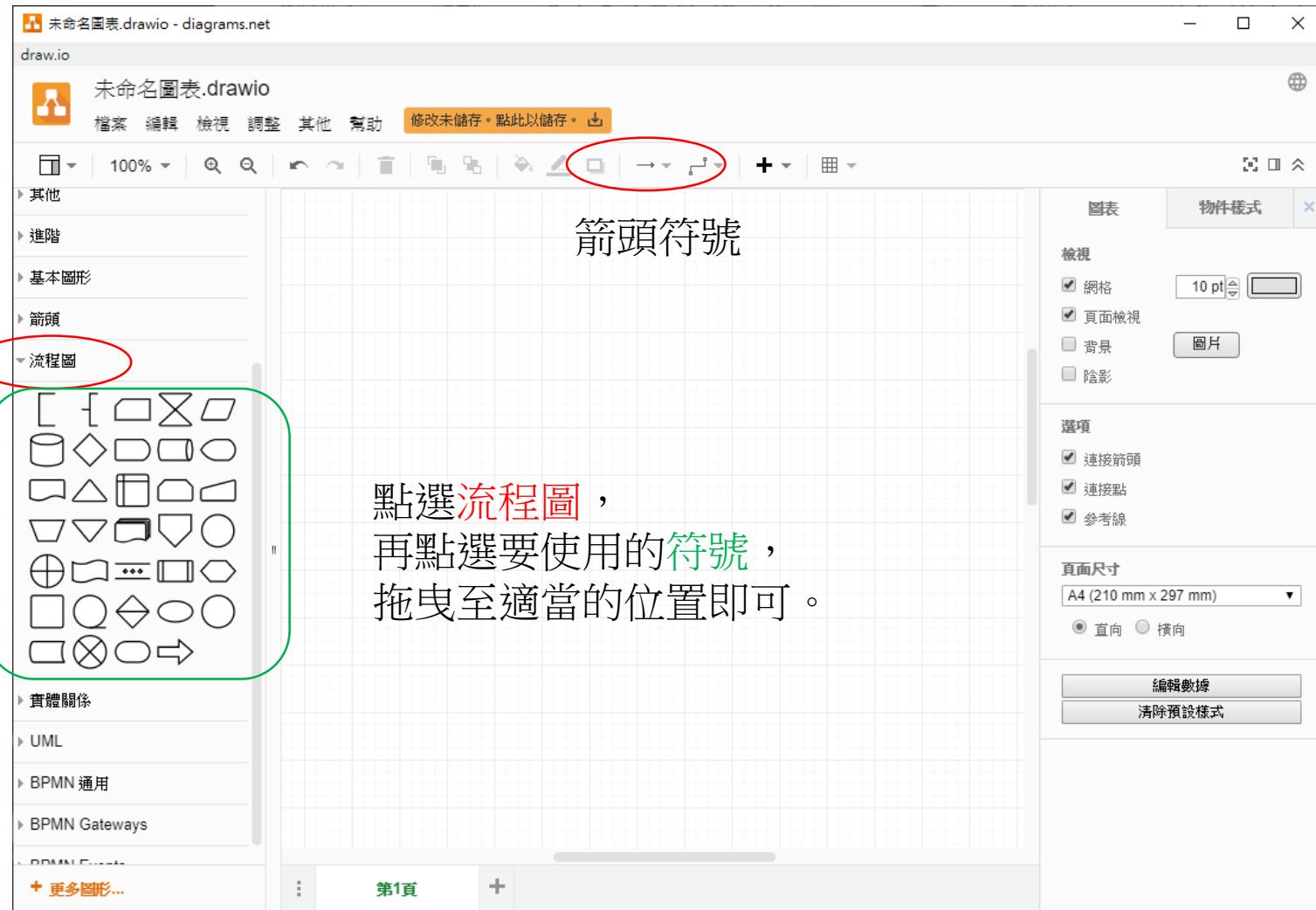


畫流程圖的工具

- 安裝後開啟程式選「流程圖」→「新增」

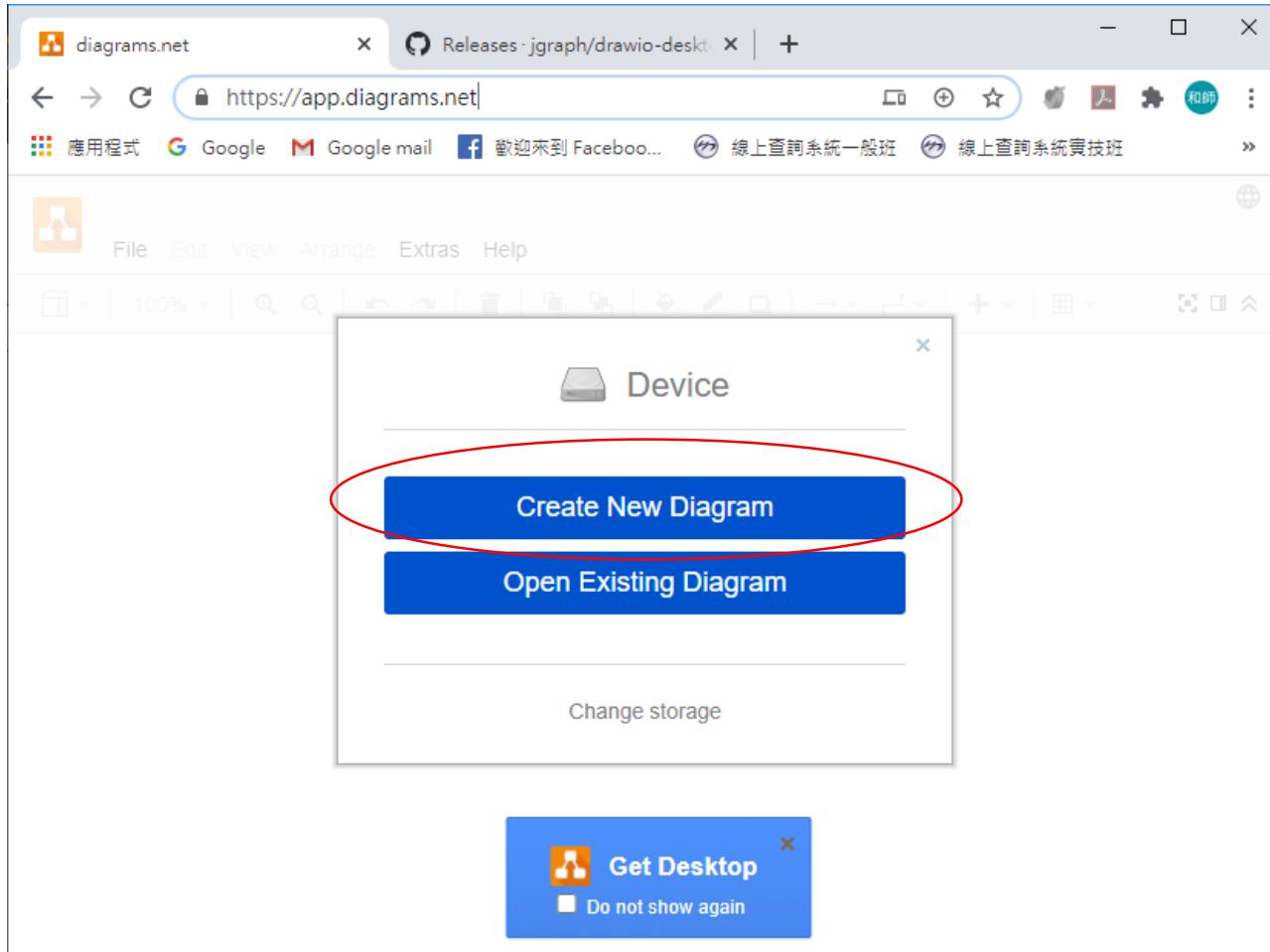


畫流程圖的工具



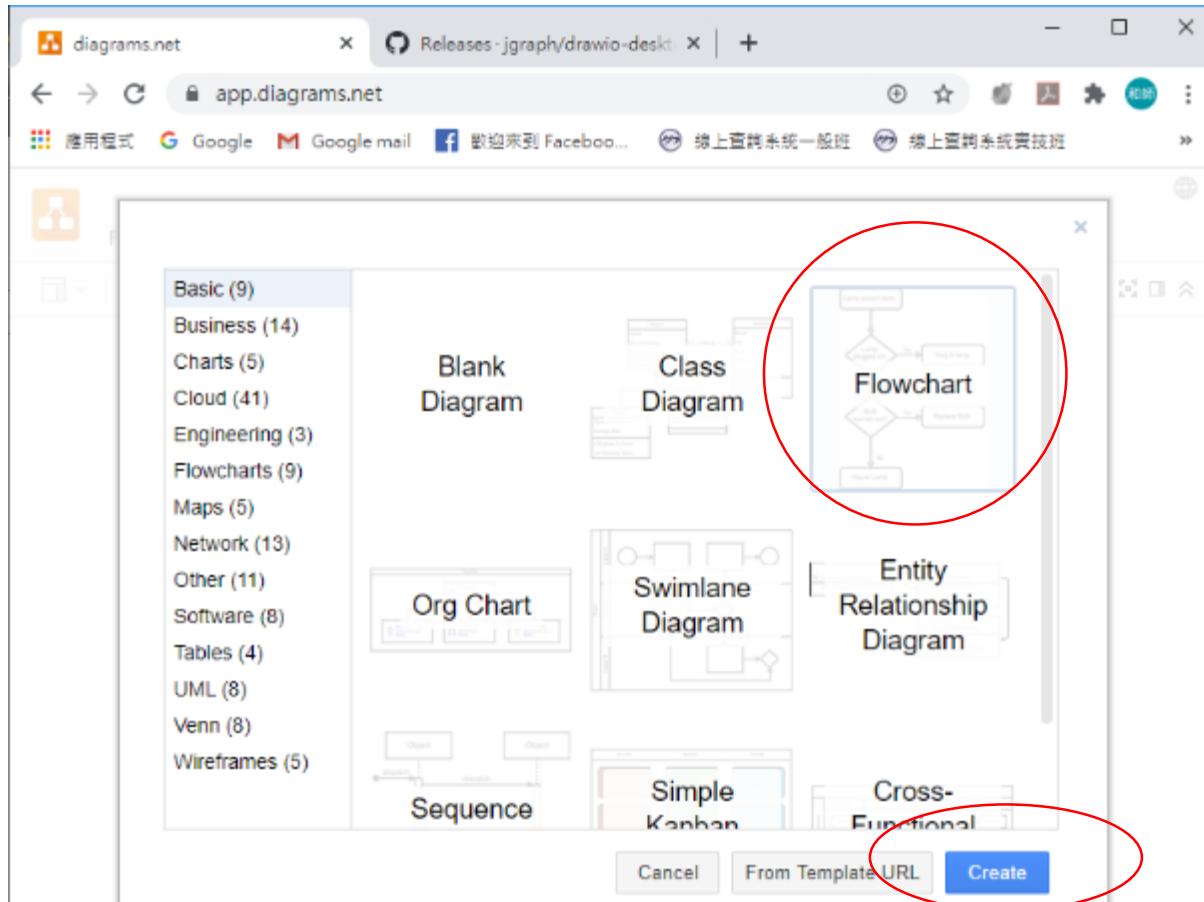
畫流程圖的工具

- 也可用線上版，直接在瀏覽器裡使用。



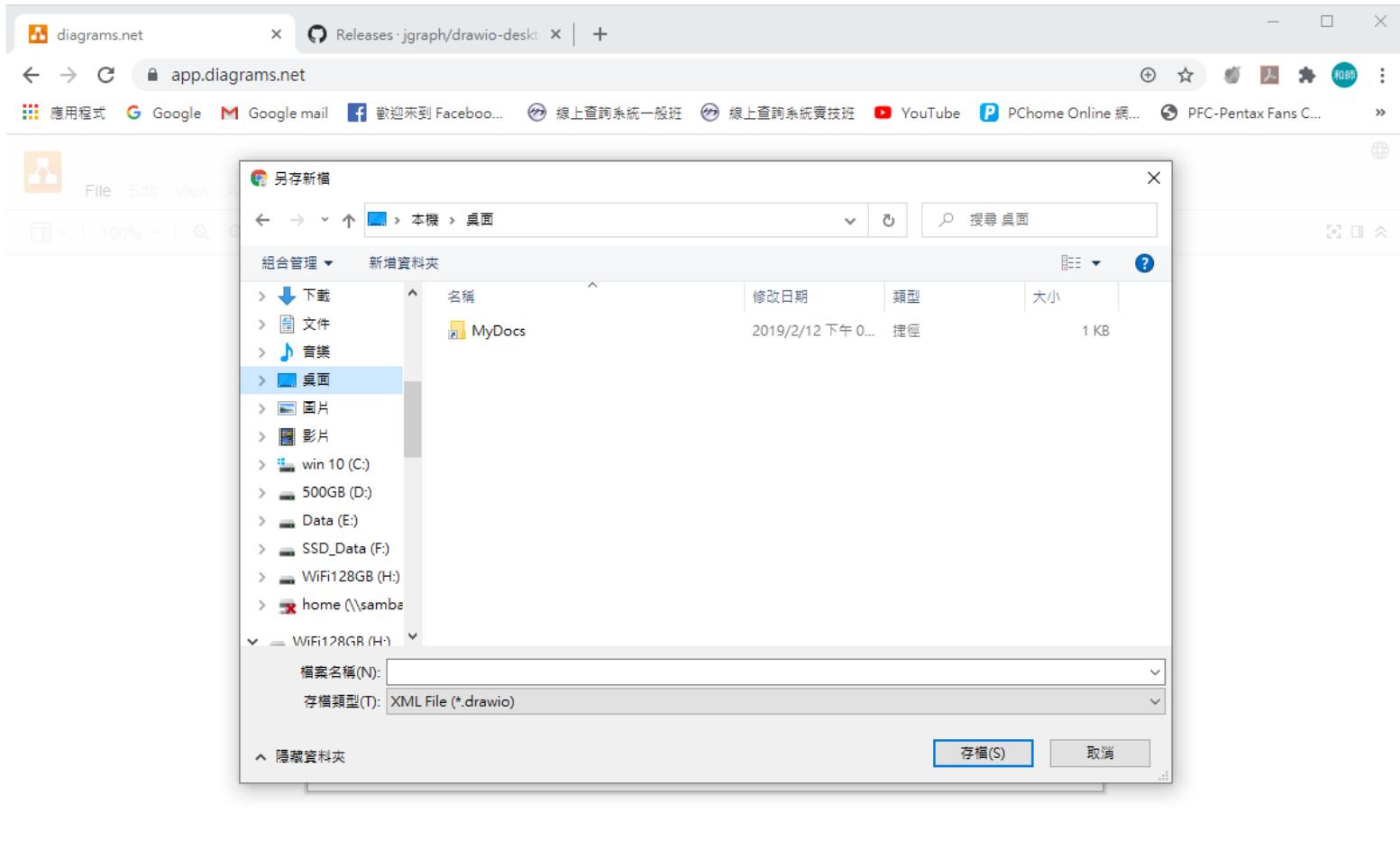
畫流程圖的工具

- 點選Flowchart → Create



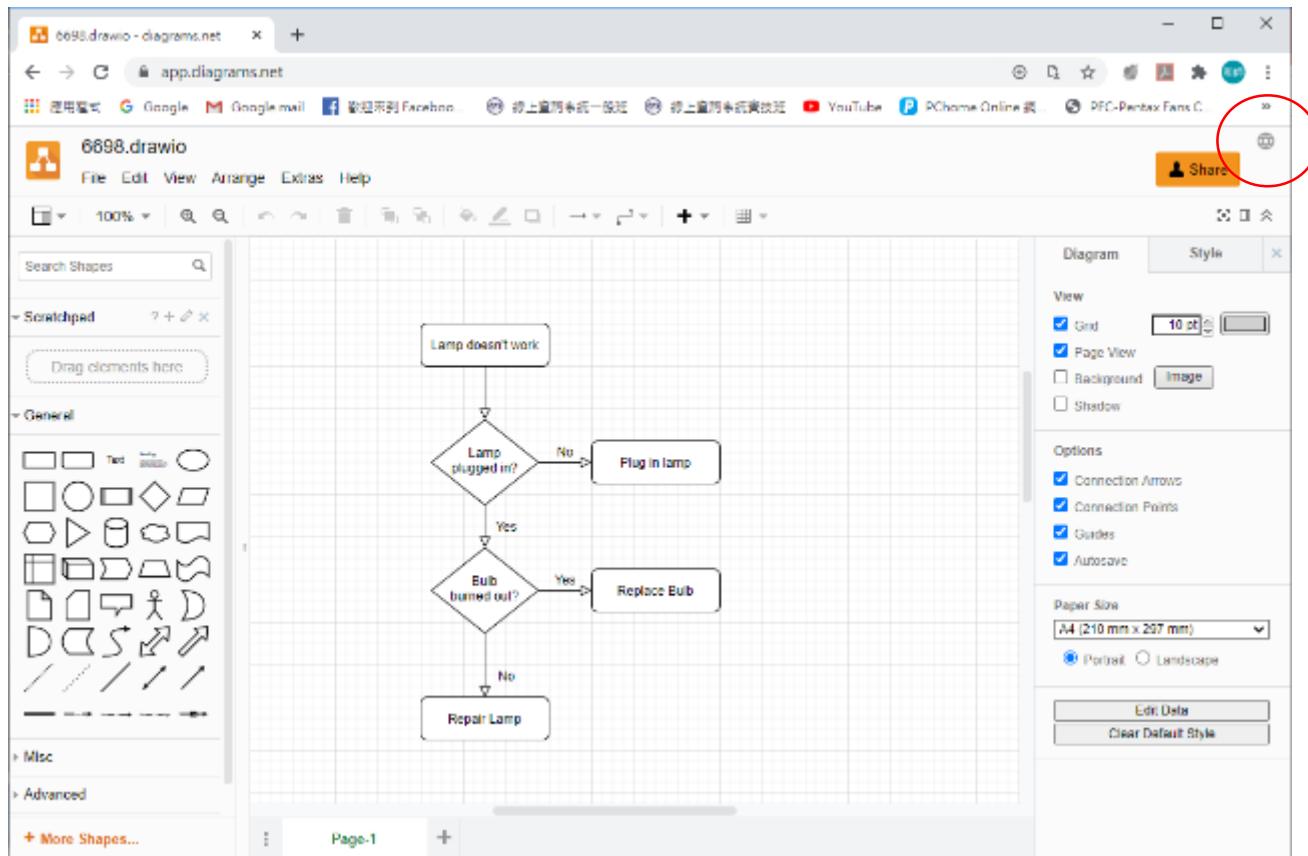
畫流程圖的工具

- 會先問你要存在哪兒



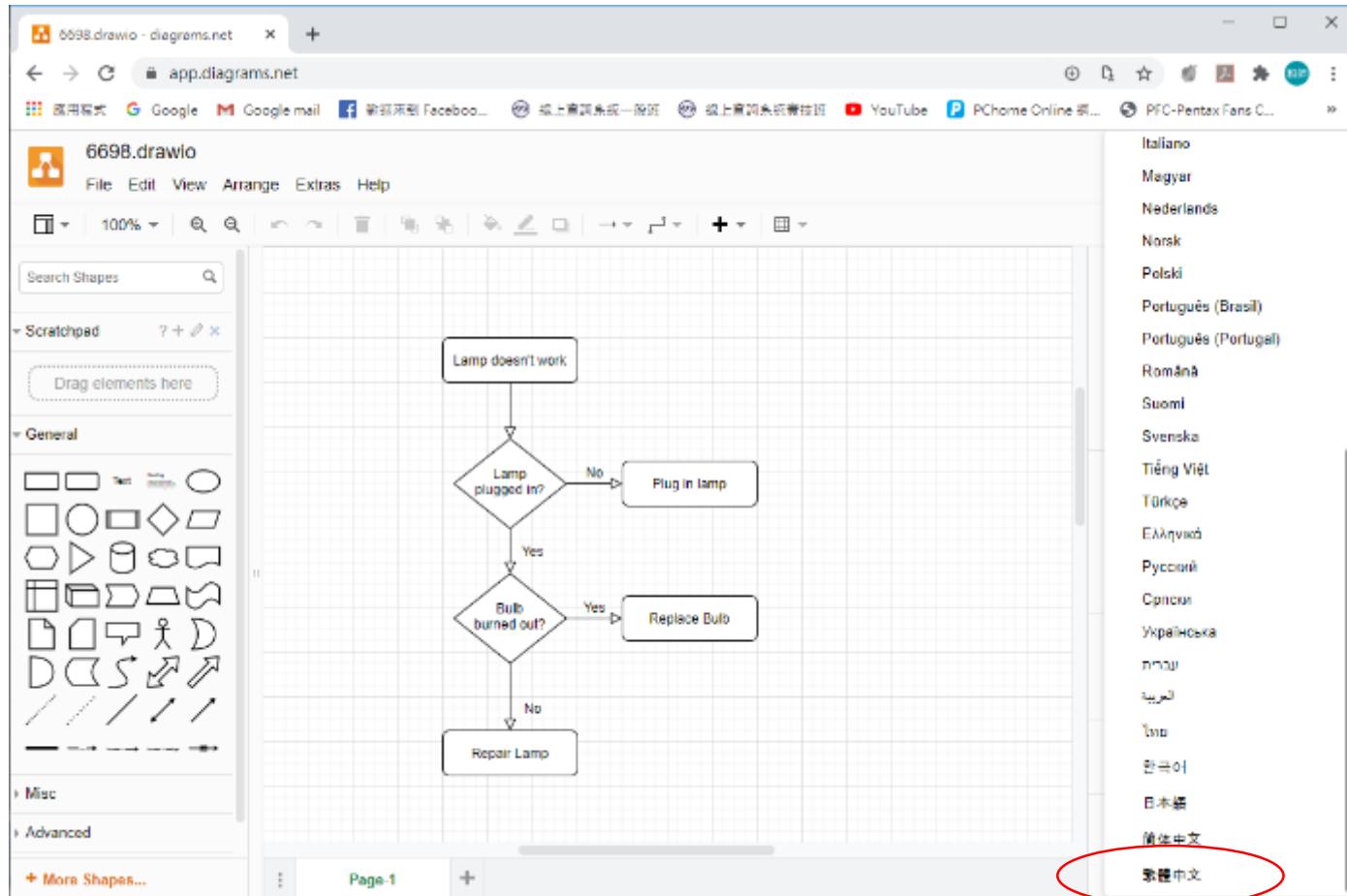
畫流程圖的工具

- 開始使用～（右上方地球符號可以更換界面語言）



畫流程圖的工具

- 可以選繁體中文(需要重新載入頁面)



想想看

- 颱風天到了，請寫一個防颱準備的流程圖



- (你會不會把門窗封住了才想到要去買泡麵吧?)

這些益智遊戲可以多玩玩

- 過河問題：3個人和3個鬼要過河，可是只有一隻小船，一次只能載兩個，且在河兩邊鬼的數量不能比人多，否則就會把人給吃了，怎麼樣才能安全的過河呢？



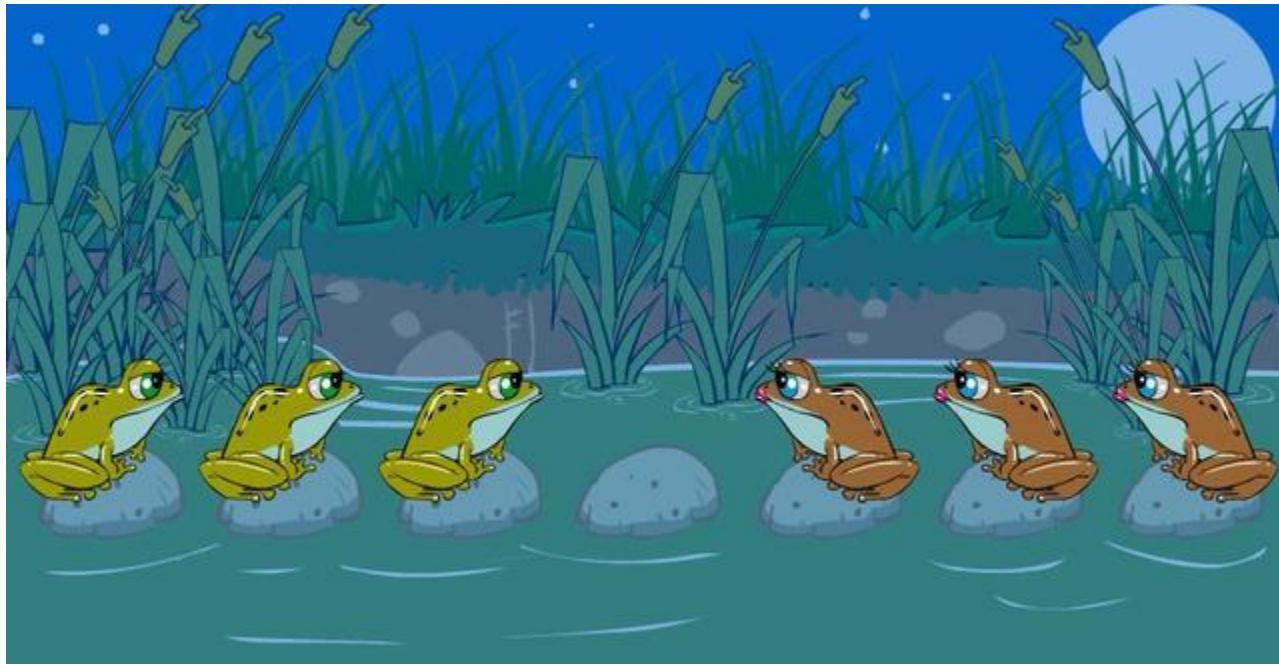
這些益智遊戲可以多玩玩

- 比較複雜的過河遊戲：一個男人帶著兩個小男孩，一個女人帶著兩個小女孩，一個警察帶著一個小偷，要過河。可是只有一條小船，一次只能載兩個人，孩子和小偷不會划船。如果男人不在則女人會教訓小男孩，如果女人不在男人則會教訓小女孩，如果警察不在小偷則會傷害這些人。怎麼才能安全過河呢？



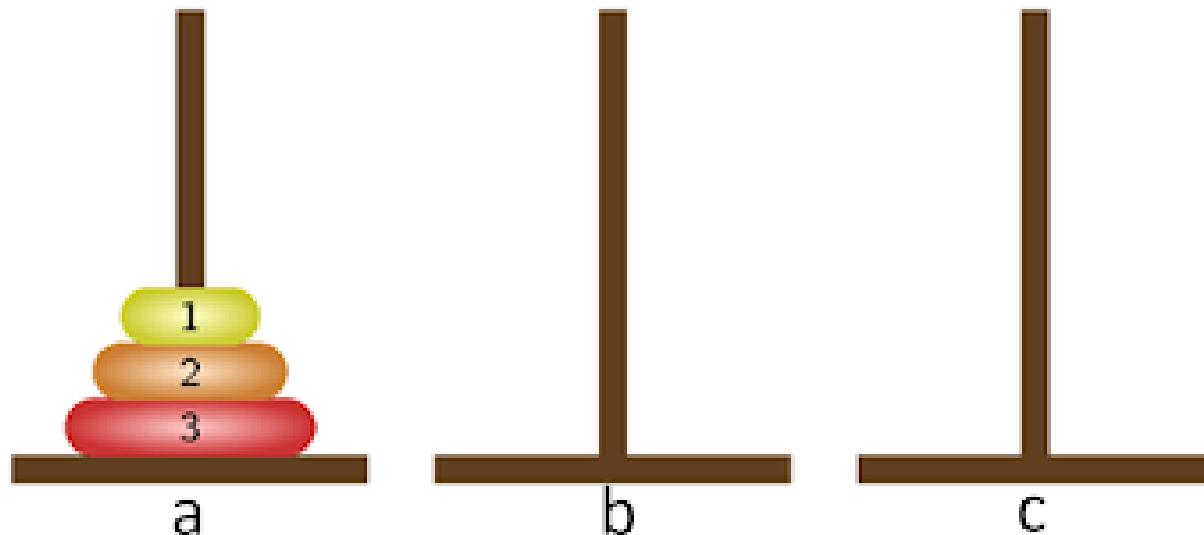
這些益智遊戲可以多玩玩

- 有六隻青蛙要過河，河的左邊有三隻，河的右邊有三隻，中間一塊空白石頭。它們互不相讓。怎麼樣才能過河呢？
- 規則：可向前跳不可向後，每次只能躍過1隻青蛙。



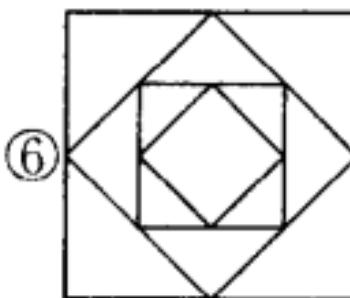
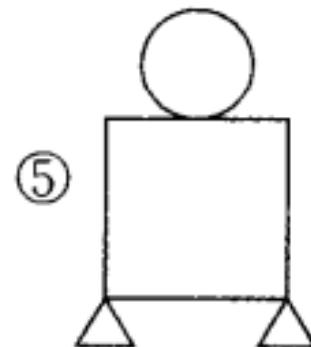
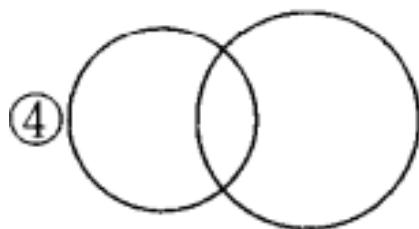
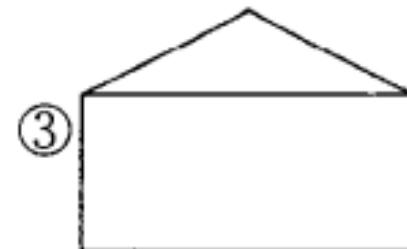
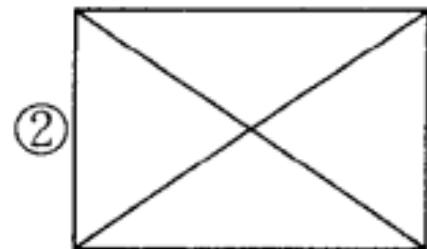
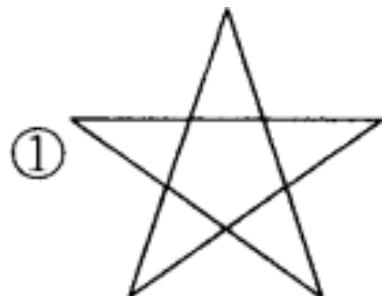
這些益智遊戲可以多玩玩

- 河內塔：將左邊的圈圈移動到最右邊。
 - 規則一：一次只能移動一個。
 - 規則二：任何時候不能發生上面的圈圈比下面的圈圈大。



這些益智遊戲可以多玩玩

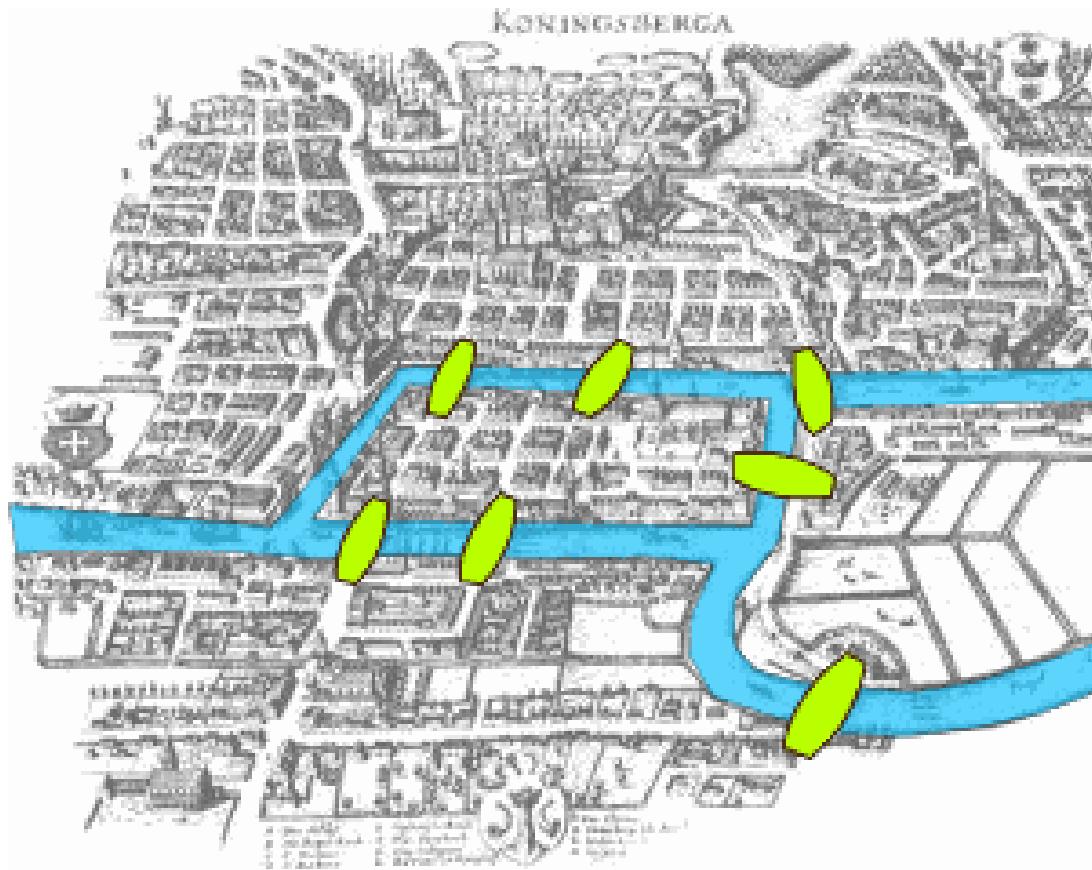
- 一筆畫問題：一筆畫走完所有線段，不能重複。



- 技巧：只要頂點引出的線為單數的頂點個數不大於2，這個幾何圖形就可以一筆畫。

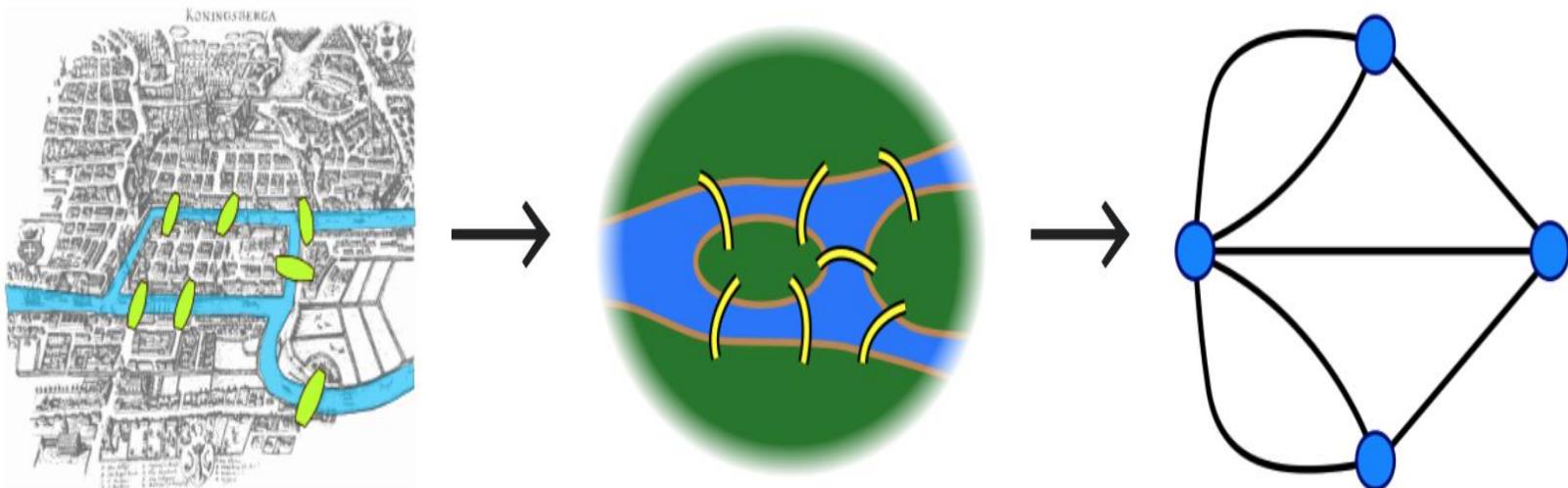
這些益智遊戲可以多玩玩

- 柯尼斯堡的七橋問題：要如何才能從某一塊土地開始，將每座橋恰好經過一次，然後回到原地？



這些益智遊戲可以多玩玩

- 柯尼斯堡的七橋問題：要如何才能從某一塊土地開始，將每座橋恰好經過一次，然後回到原地？
- (模組化、抽象化)



練習一下(畫出流程圖)

- 有兩串香蕉，比較它們的重量，
然後輸出哪一個比較重？



- 輸入三個數值a，b，c，比較它們
的大小，輸出最大的那個。



練習一下(畫出流程圖)

- 請寫一個流程圖，狀況如下：
- 今天晚餐要約小美吃飯，如果身上有\$1000就吃大餐，如果只有\$500就吃牛肉麵，如果只有\$300就吃夜市，都不夠的話取消約會(慘...)。



多練習就會了

- 流程圖是你解決問題的想法及過程，沒有標準答案，只要能解決問題就可以。
- 不過**效率**就有關係了，要練習讓你的思考更清晰，步驟更簡潔。



休息一下~

加油~ ~



Python簡介

- Python 是一種泛用性的動態物件導向程式語言。
- Python 是一種直譯式語言。
- 它可以在命令列模式下操作，也可以寫成 Script 檔。
- 附檔名為 **.py**。
- Python 可以執行在 Windows、Mac OS X、Linux 等常見的作業系統平台和其它較少使用的作業系統上，也可以在 Java 和 .Net 環境中執行。
- 透過 Jython，Python 能與 Java 合作愉快；
透過 IronPython，Python 能與 .Net 合作愉快。

Python簡介

- Python的設計哲學強調程式碼的可讀性和簡潔的語法，尤其是使用 空格縮排劃分程式碼塊。

在Python語言裡整齊的縮排很重要而且是必須的
(很重要，唸三遍!)

- Python這個單字是一種蟒蛇，所以這個語言的Logo就是這樣來的。



Python簡介

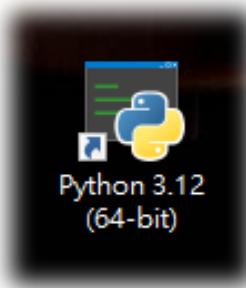
- Python常應用的領域：



(圖：又上財經學院)

Python簡介

- Python官網 <https://www.python.org/>
- 目前是3.13.5版(2025年8月)，下載安裝後可於命令列模式下執行，提示字元為 >>>
- 若須測試簡單的指令或執行已寫好的程式，可用此模式。



圖示
(3.12版)

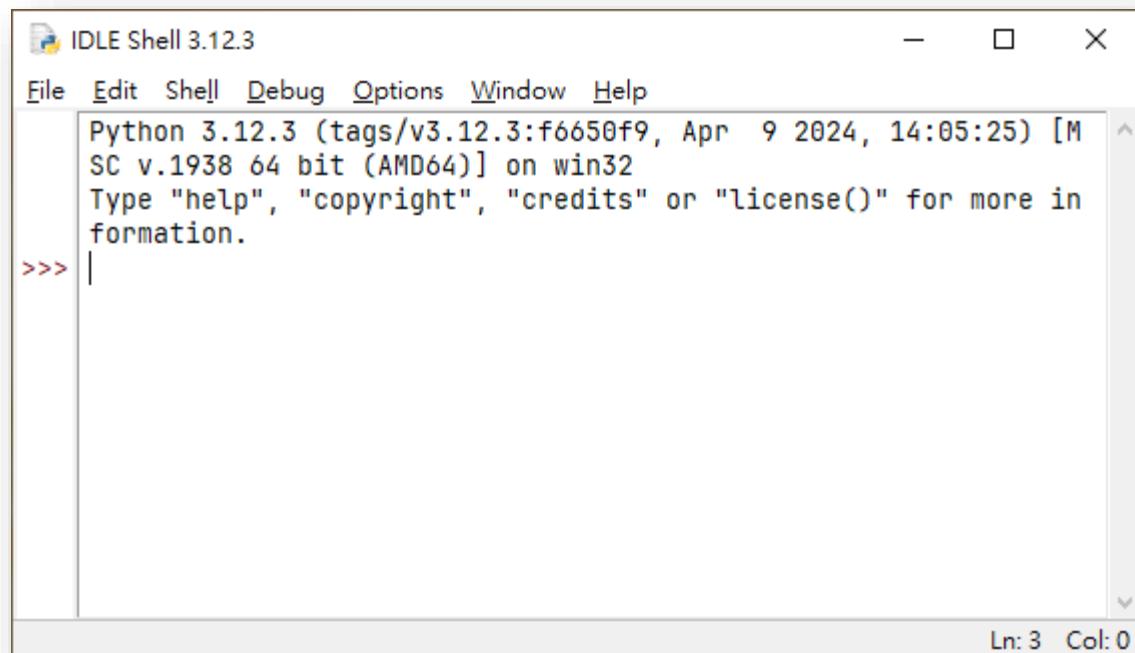
A screenshot of a Windows terminal window titled "Python 3.12 (64-bit)". The window displays the Python 3.12.3 copyright information and a command prompt (>>>). The terminal has a black background with white text and a vertical scroll bar on the right side.

Python簡介

- 安裝後也有一個執行環境叫IDLE，不用再找IDE了，之後我們都用這個環境來練習簡單的指令和小程式。



圖示
(3.12版)

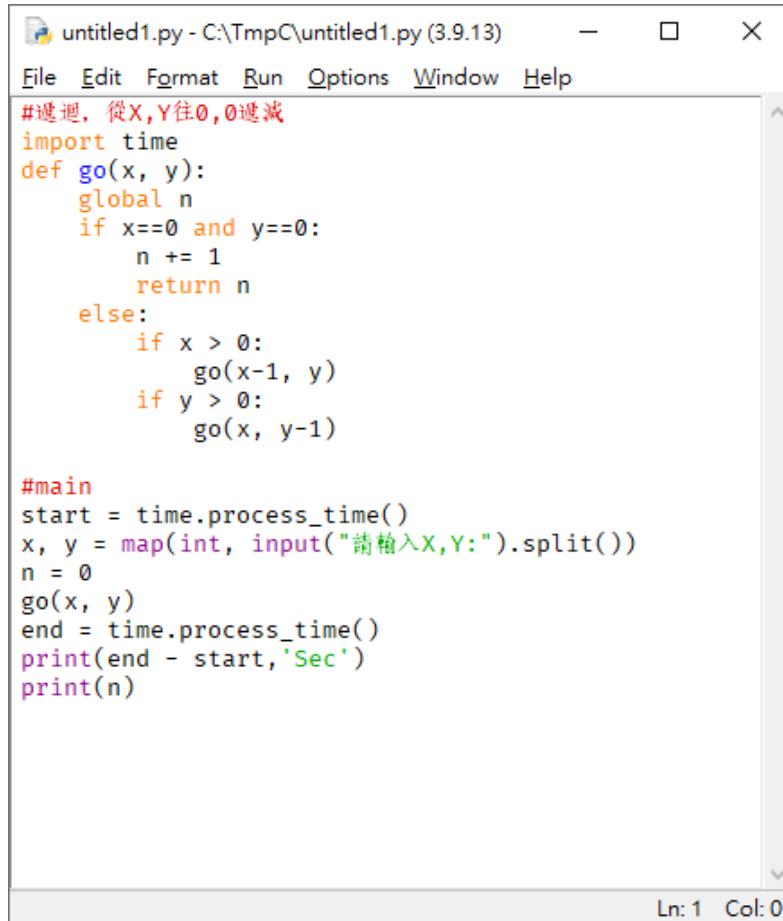


```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
=>
```

- IDE(Integrated Development Environment；整合發展環境)

Python簡介

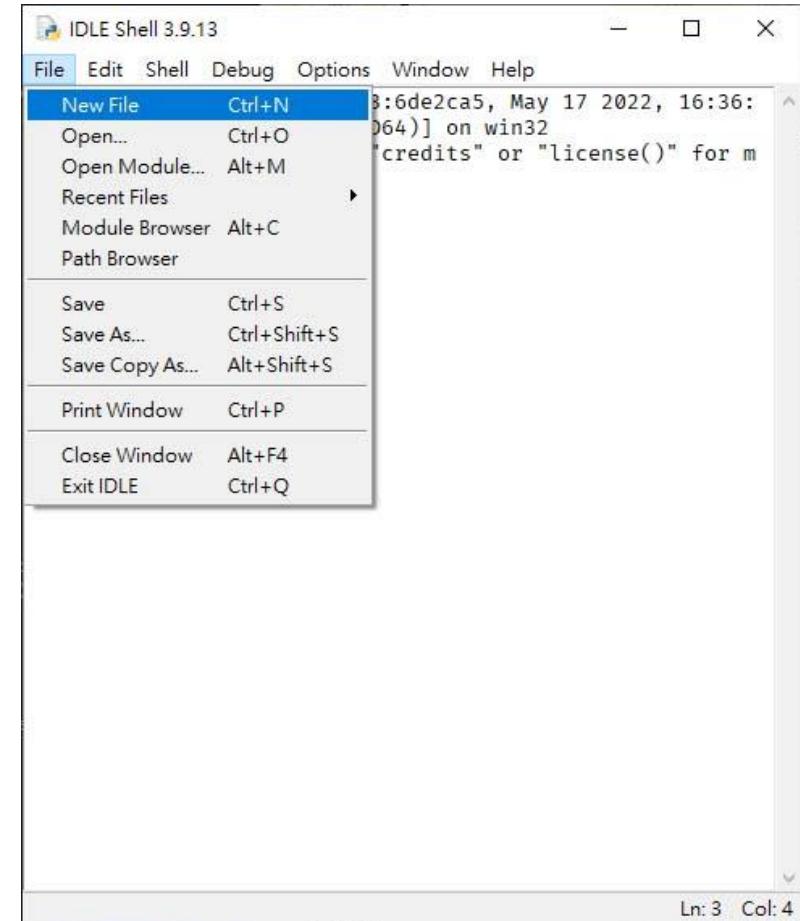
- 可以開啟Script編寫視窗來發展程式。



```
#遞迴，從X,Y往0,0遞減
import time
def go(x, y):
    global n
    if x==0 and y==0:
        n += 1
        return n
    else:
        if x > 0:
            go(x-1, y)
        if y > 0:
            go(x, y-1)

#main
start = time.process_time()
x, y = map(int, input("請輸入X,Y:").split())
n = 0
go(x, y)
end = time.process_time()
print(end - start, 'Sec')
print(n)
```

Ln: 1 Col: 0



Python簡介

- 還有一個常用的套件：Anaconda，安裝後有一個編輯器叫Spyder，很好用，還內建有超過1000種的Data Science Packages可使用，要import套件時就不用怕找不到還要去下載了。
- 網址：<https://www.anaconda.com/>



Anaconda這個字
也是一種蟒蛇

Python簡介

- Spyder編輯器和有名的Visual Studio一樣有自動提示與即時顯示錯誤的功能，能減少程式的錯誤。



The image shows the Spyder Python IDE interface. On the left, there's a code editor window titled 'temp.py' containing Python code for connecting to a MySQL database and executing a query. On the right, there's an 'IPython Console' window showing the Python and IPython versions, and an 'Usage' help panel.

```
1 import pymysql
2 db_info = {'host' : 'localhost', 'port' : 3306,
3            'user' : 'root', 'password' : '12345678',
4            'db' : '我的商店', 'charset' : 'utf8'}
5 db = pymysql.connect(**db_info)
6 cursor = db.cursor()
7
8 query = '''SELECT DISTINCT A.商品ID, B.商品名稱,
9             FROM 販賣資料 A, 商品清單 B, 負責人清單 C
10            WHERE A.商品ID=B.商品ID AND A.負責人ID =
11                  AND (處理日 >= '2000-04-01' AND 處理日 <
12                  ORDER BY A.商品ID'''
13 cursor.execute(query)
14
15 data = cursor.fetchall()
16 for row in data:
17     print(row[0],row[1],row[2])
18
19 db.close()
20
```

IPython Console History Terminal
internal (Python 3.8.10) Completions: internal ✓ LSP: Python Line 20, Col 1 UTF-8-GUESSED CRLF RW Mem 64%

Python簡介

- 還有一個蠻專業的IDE叫PyCharm，它有免費版和收費版，非常好用，它會有許多提示訊息，讓你避免錯誤，如果你遵照它的提示，可以寫出很"正統"的Python程式。
- PyCharm是一個跨平台開發環境，擁有 Windows、macOS和Linux版本。
- 官方網站：<https://www.jetbrains.com/pycharm/>

A screenshot of the PyCharm IDE interface. On the left, there's a project tree showing a 'PyCharmMiscProject' folder containing a 'venv library root' and a 'script.py' file. The main window shows the code for 'script.py':

```
# This is a sample Python script.  
# Press Shift+F10 to execute it or replace it with your code.  
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.  
  
def print_hi(name):  
    # Use a breakpoint in the code line below to debug your script.  
    print(f'Hi, {name}') # Press Ctrl+F8 to toggle the breakpoint.  
  
# Press the green button in the gutter to run the script.  
if __name__ == '__main__':  
    print_hi('PyCharm')  
  
# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

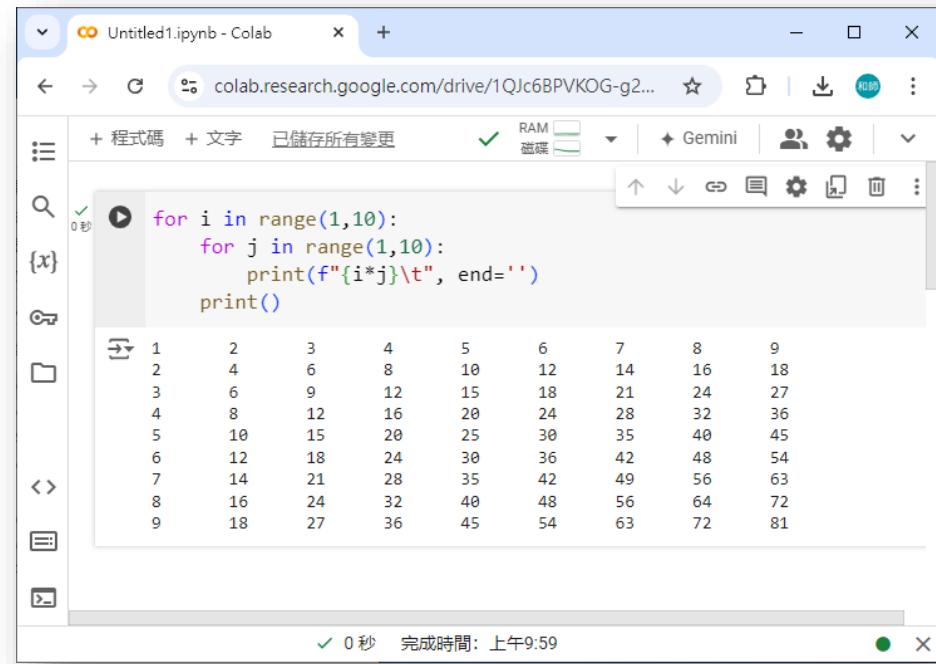
Below the code editor is a terminal window titled 'Run' showing the output of running the script:

```
C:\Users\orion\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\orion\PyCharmMiscProject\script.py  
Hi, PyCharm  
Process finished with exit code 0
```

The status bar at the bottom indicates the project name 'PyCharmMiscProject', the file 'script.py', indexing status, and system information like '17:1 CR LF UTF-8 4 spaces Python 3.11 (PyCharmMiscProject)'.

Python簡介

- 如果你什麼都不想安裝，也可以直接在線上練習。
- 先註冊一個google帳號，然後進入
<https://colab.research.google.com/>
- 我們會在上機時說明如何使用。



The screenshot shows a Google Colab notebook titled "Untitled1.ipynb - Colab". The code cell contains the following Python code:

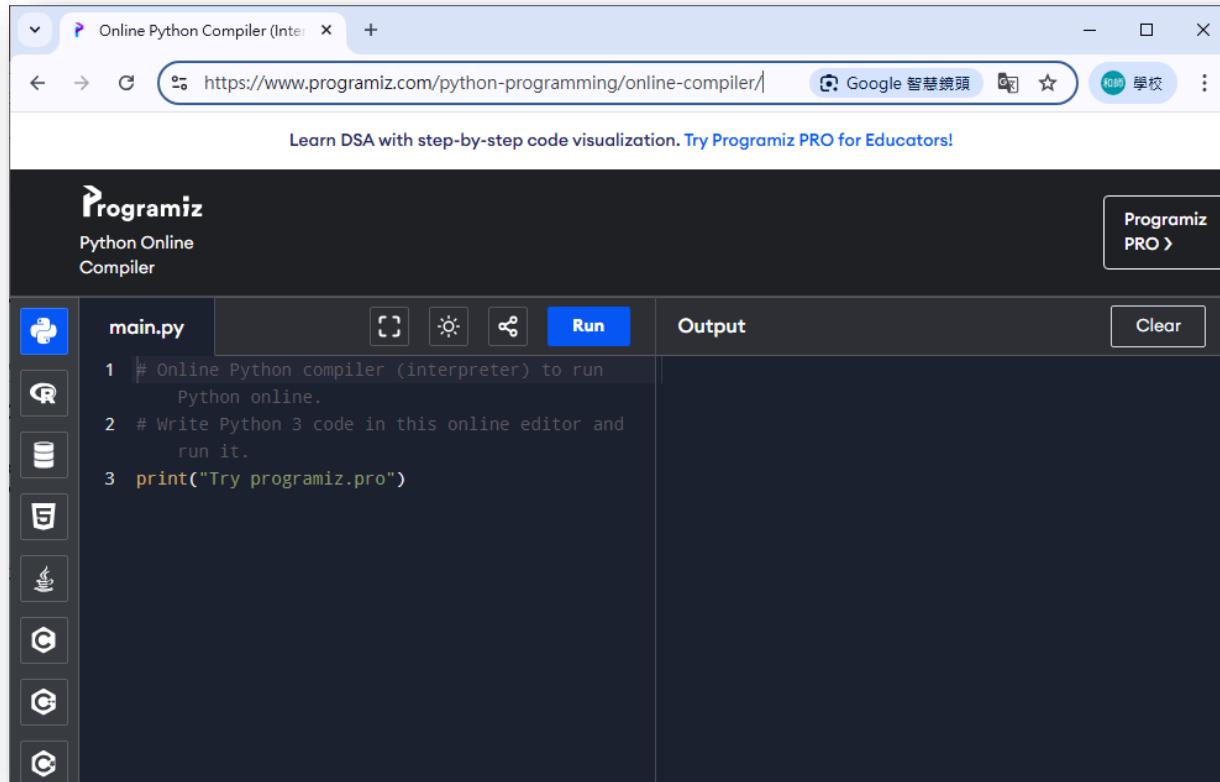
```
for i in range(1,10):
    for j in range(1,10):
        print(f"{i*j}\t", end='')
    print()
```

The output of the code is a 10x10 multiplication table:

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

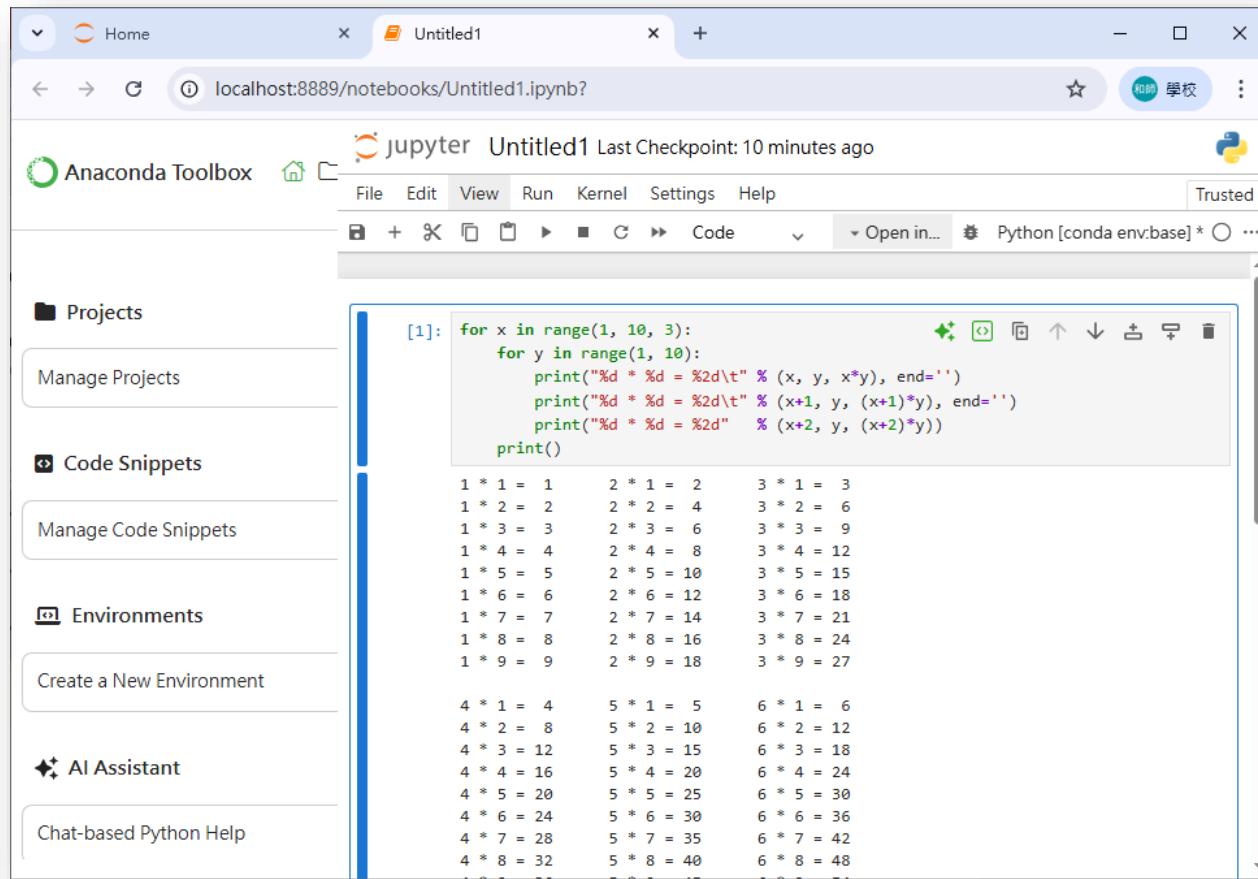
Python簡介

- 網路上也有許多讓你練習Python的網站，你不必安裝任何軟體，直接上網就可以練習了。
- 例如：Programiz網站。



Python簡介

- 或者Anaconda也有附一個Jupyter Notebook，讓你在線上有自己專屬的空間來練習Python。



The screenshot shows the Anaconda Toolbox interface with a Jupyter Notebook window open. The notebook cell [1:] contains the following Python code:

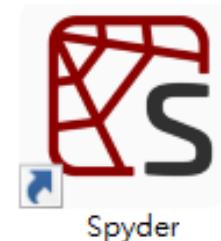
```
for x in range(1, 10, 3):
    for y in range(1, 10):
        print("%d * %d = %2d\t" % (x, y, x*y), end=' ')
        print("%d * %d = %2d\t" % (x+1, y, (x+1)*y), end=' ')
        print("%d * %d = %2d\t" % (x+2, y, (x+2)*y))
    print()
```

The output of the code is displayed below the code cell, showing multiplication tables for 1, 2, 3, 4, 5, 6, 7, 8, and 9. The output is formatted as three columns of multiplication results.

	1 * 1 = 1	2 * 1 = 2	3 * 1 = 3
1 * 2 = 2	2 * 2 = 4	3 * 2 = 6	
1 * 3 = 3	2 * 3 = 6	3 * 3 = 9	
1 * 4 = 4	2 * 4 = 8	3 * 4 = 12	
1 * 5 = 5	2 * 5 = 10	3 * 5 = 15	
1 * 6 = 6	2 * 6 = 12	3 * 6 = 18	
1 * 7 = 7	2 * 7 = 14	3 * 7 = 21	
1 * 8 = 8	2 * 8 = 16	3 * 8 = 24	
1 * 9 = 9	2 * 9 = 18	3 * 9 = 27	
4 * 1 = 4	5 * 1 = 5	6 * 1 = 6	
4 * 2 = 8	5 * 2 = 10	6 * 2 = 12	
4 * 3 = 12	5 * 3 = 15	6 * 3 = 18	
4 * 4 = 16	5 * 4 = 20	6 * 4 = 24	
4 * 5 = 20	5 * 5 = 25	6 * 5 = 30	
4 * 6 = 24	5 * 6 = 30	6 * 6 = 36	
4 * 7 = 28	5 * 7 = 35	6 * 7 = 42	
4 * 8 = 32	5 * 8 = 40	6 * 8 = 48	

Python簡介

- 對於初學者來說，IDLE最簡單，PyCharm功能最強但最複雜，如果按使用簡單到複雜排序如下：
 - IDLE（熟悉Python基本語法）。
 - Colab / Jupyter Notebook（開始做互動式練習與數據處理）。
 - Spyder（熟悉專案結構與變數管理）。
 - PyCharm（進入專案開發與專業開發流程）。



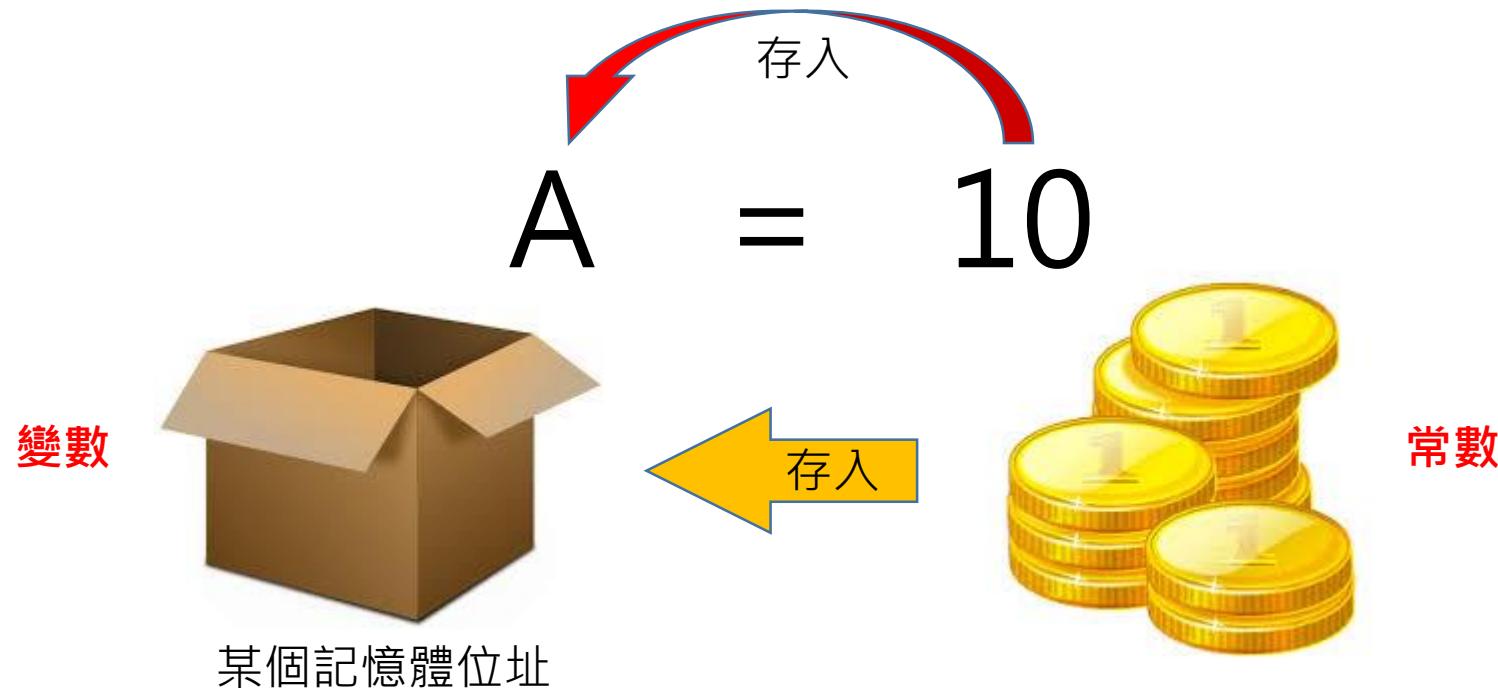
休息一下~

- 工欲善其事，必先利其器，請回家後把環境安裝好，不能只靠每週一次電腦教室的環境。
- 想學會寫程式光是靠聽老師說是不可能學會的，要自己把範例都親自輸入跑一遍，然後習題都要做出來才行。

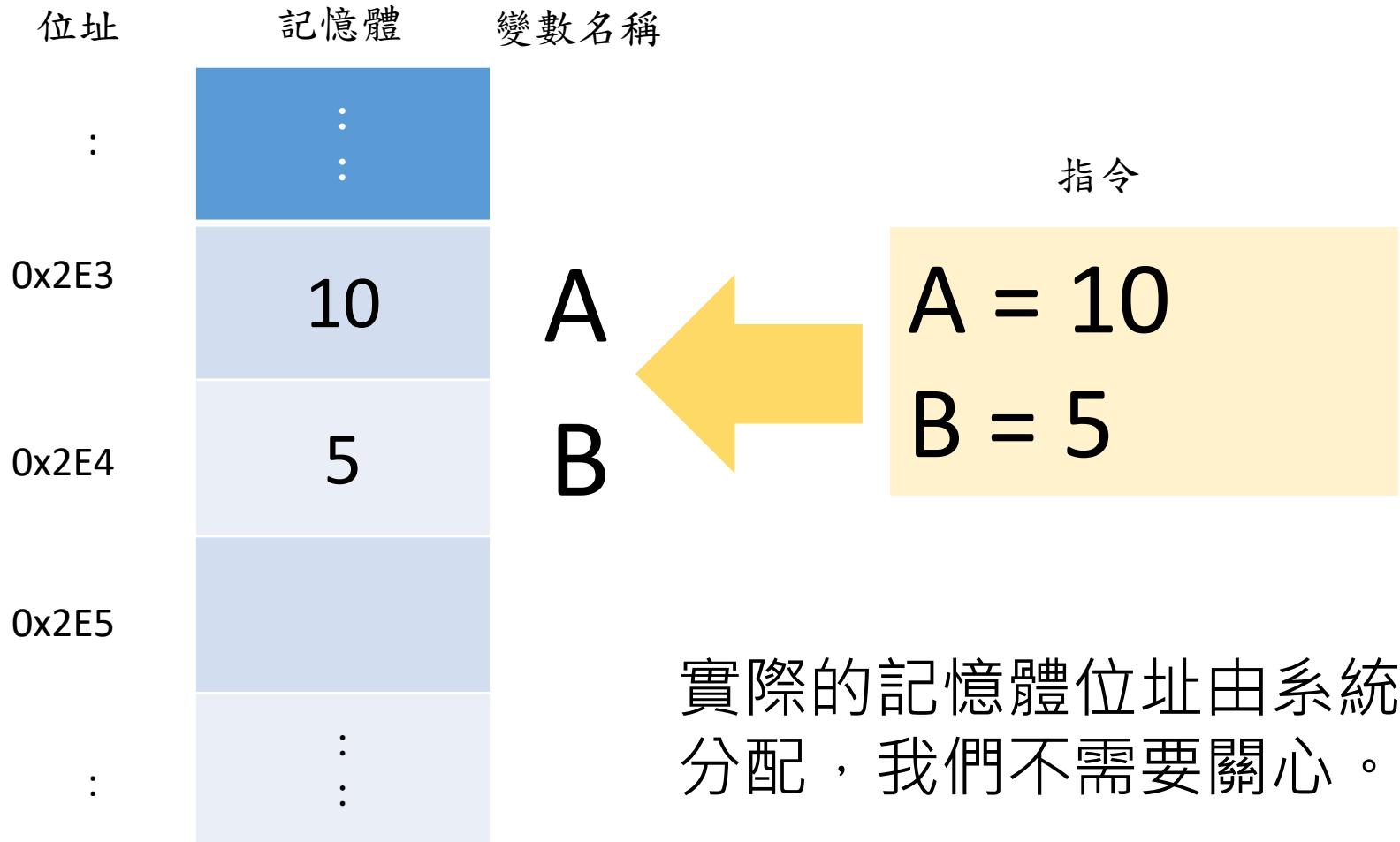


常數與變數

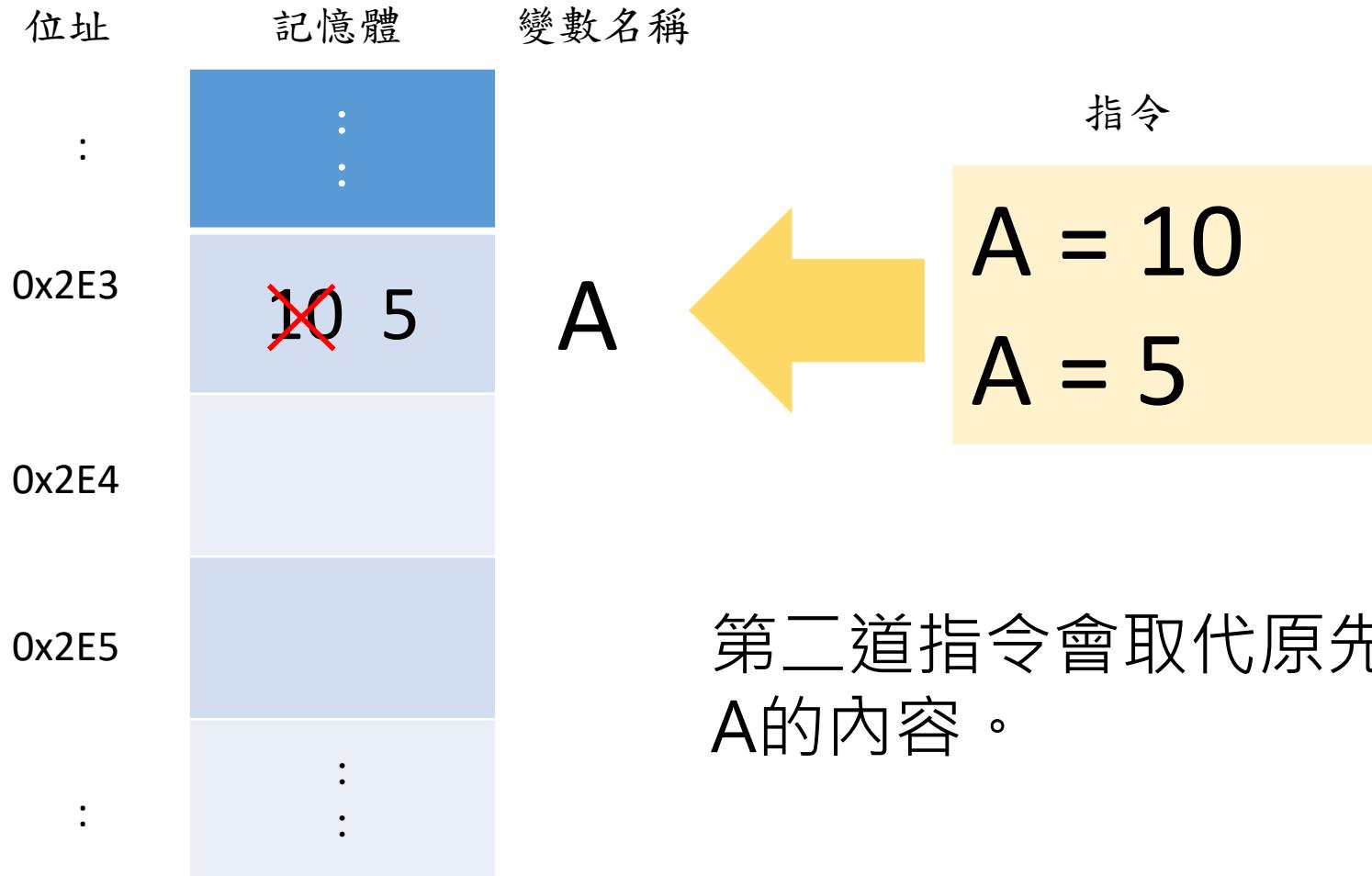
- 常數：程式執行時的值。
- 變數：代表一個可存放資料的記憶體位置，是程式中不可或缺的部分。



常數與變數

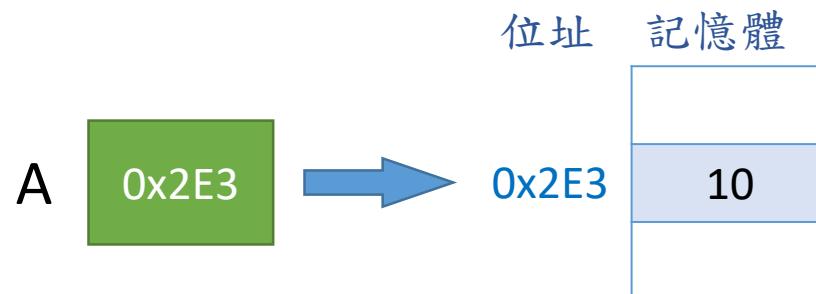


常數與變數



常數與變數

- 以上所述只是一個概念，實際上的情況在 Python 會複雜一些，它是採用「物件」的方式來儲存資料。
- 對 Python 而言，變數是採用 **參照 (reference)** 位址的概念，也就是變數 A 存放的是一個指向常數 10 的位址的值，而不是直接存放 10。
- 我們以後會介紹物件，現在你只要有前頁的概念就行了。



變數命名規則

- 變數是要自己命名的，規則如下：
 - 只能由字母、數字或底線組成，不可以用標點符號或特殊字元。
 - 須以英文字母或底線開頭。
 - 不能使用關鍵字命名。
 - 同一個可視範圍不能有同名的變數。
 - 英文有大小寫之分，也就是cat與Cat是不同的變數。
 - 可以使用中文命名，但不建議。
 - 長度沒有限制，但太長了自己也不方便。



Python保留字

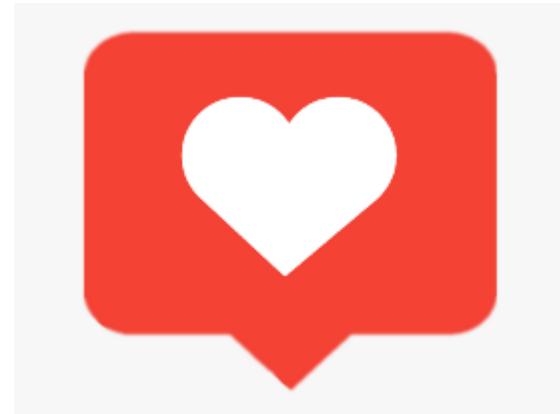
- 保留字又稱關鍵字，是有特殊意義的字，自己為變數取名時不可以使用這些特殊的字。

變數-python 保留字

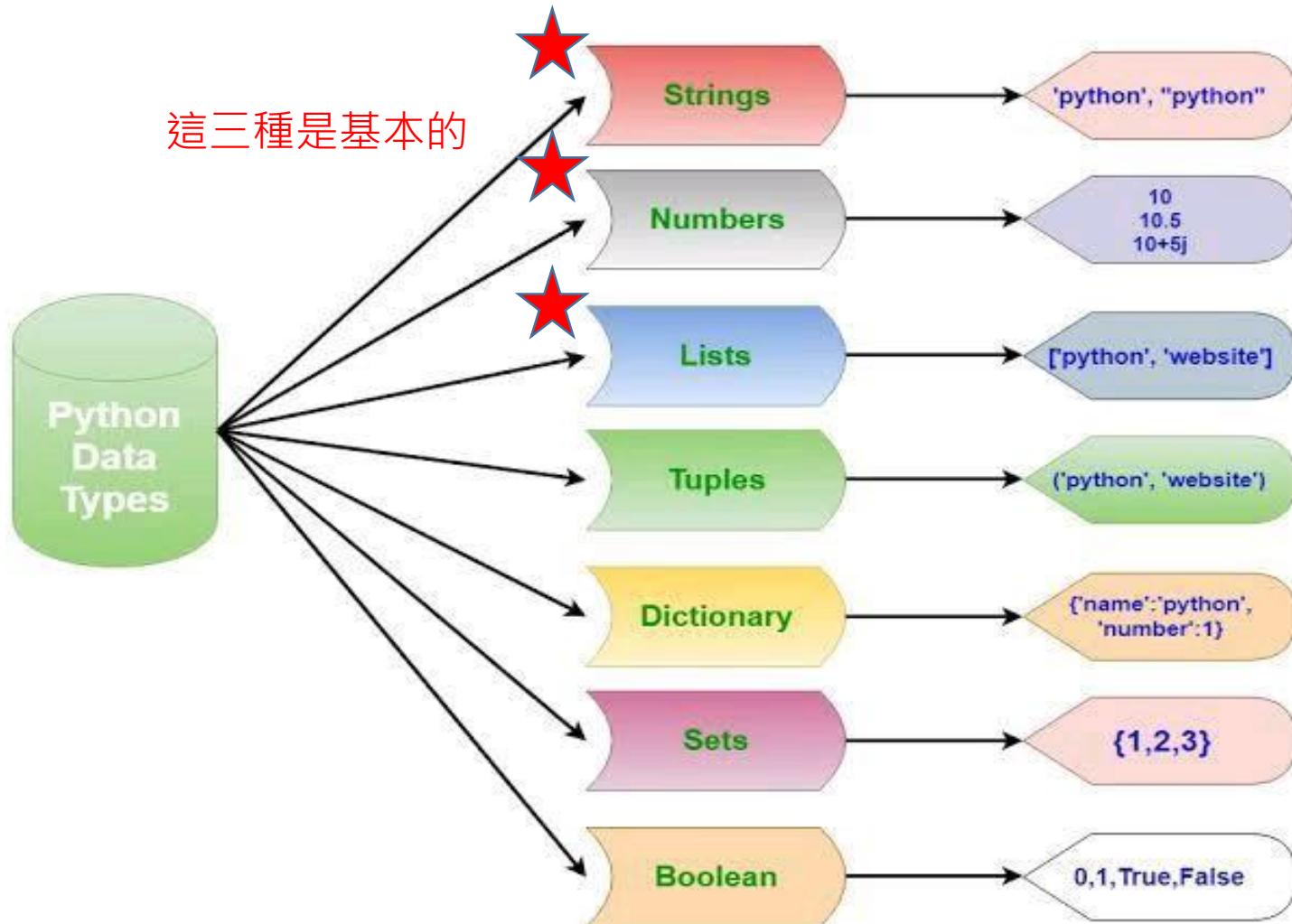
False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Python 變數不須宣告

- 許多程式語言都是需要先行宣告變數才可以存取的，目的是讓作業系統為我們的程式在執行時保留適當的記憶體空間。
- Python所有變數皆不需先行宣告，直接指派使用就行了，但你自己必須要弄清楚變數裡現在存了什麼類型的資料。

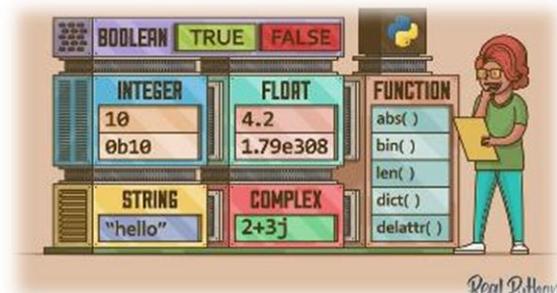


Python的資料型態



Python的資料型態

- Python的主要資料型態有：
 - 基本資料型態：
 - **number**：數值
 - 整數(**integer**)、長整數(**long**)
 - 浮點數(**float**)、複數(**complex**)
 - **string**：字串
 - **boolean**：布林
 - 群集資料型態：
 - **list**：清單(陣列)
 - **tuple**：元組
 - **dictionary**：字典
 - **set**：集合



基本資料型態

- **number**(數值)包含：
 - 浮點數(**float**)：
 - 例：3.1415, 6.02E13 (64位元，倍精度)
 - 整數(**integer**)：
 - 例：100 (十進制)
0o324 (八進位數)
0xA140 (十六進位數)
0b10011100 (二進位數)
 - 複數(**complex**)：
 - 例：1-5j (兩個浮點數)

6.02¹³
在程式中要寫成這樣

基本資料型態

- **string**(字串)：一個或一串文字。
 - 單引號包圍
 - 例：`'this is a "single-quoted"`
`string, \ndon\ 't you like it? '`
 - 雙引號包圍
 - 例：`"this is a 'double-quoted' string"`
 - 三引號包圍(三個單引號)
 - 例：`'''this is a triply-quoted string`
字串中可以換行，不用加跳脫字元，
也可以用'單引號'和"雙引號" '''
- (也可用來當多行的註解)

基本資料型態

- **boolean**(布林)：
 - 布林是一個人的名字，他發展了一套專門對真真假假做運算的學問，叫布林代數。
 - 布林值只有「True」和「False」兩種，只有0或空值(Null)視為假，其餘皆視為真。
- 例： a = **True** (a裡面存了一個"真"這個值)
b = **False** (b裡面存了一個"假"這個值)
(不須用引號框起來，大小寫不要拼錯)

資料型別轉換

- 有時我們需要將資料型別進行轉換，以配合我們的需求，例如將數值轉成字串或字串轉成數值等。
- 常用的型別轉換：
 - `str(x)` #將變數x的內容轉換成字串
 - `int(x)` #將變數x的內容轉換成整數
 - `float(x)` #將變數x的內容轉換成浮點數
 - `complex(x)` #將變數x的內容轉換為複數
- 為何需要轉換？看看下面的例子：

數值0在電腦中值為 00000000
文字0在電腦中值為 00110000
- 0和'0'在螢幕上看一樣，但存入電腦值是不同的。

群集資料型態

- **list**(清單) :

- 也稱陣列，由中括號加逗號組成。

Ex: `m = [1, 5, 7, -3, -1, 0]`

- 清單中可以包含多種形態的資料。

Ex: `k = [3.14, 'm', 10, 'hello']`

浮點數

字元

整數

字串

群集資料型態

- **tuple**(元組)：

- 和清單相似，是一些量的有序集合。和清單最大的不同是不能改變，可以作為**dictionary**的鍵、函式回傳、多變數給值...等。
- 以小括號加逗號表示，小括弧視情況可省略。

例： t1 = (-1, 3.5, 'hello')

t2 = 1, 2, 3

x, y, z = 4, 5, 6

群集資料型態

- **dictionary**(字典)：

- 鍵(key)和值(value)的對照表-->「關連式記憶」。
- 鍵必需是不可變的型態 (如：常數、字串、Tuple等)。
- 例如：

```
prices = {'apple':7.5, 'orange':4.5, 'banana':2}
```

price 包含

Key	Value
'apple'	7.5
'orange'	4.5
'banana'	2

群集資料型態

- **set(集合)：**
 - 集合內的元素不可以重複，若有重複會自動剔除。
 - 當我們只關心資料是否出現在其中，而不關心其位置時可使用。
- 例：`s = set('糖糖 葫蘆葫蘆 糖糖糖 葫蘆蘆蘆')`
(實際上只會存'糖',' ','葫','蘆')
- 我們後面會再詳細介紹這些群集資料型態。

群集資料型態

型態	符號	特性說明	應用場景
list	[]	有序，具備索引，內容與長度可變，通過索引進行查找。	可以使用索引，簡單的資料集合。
tuple	()	有序，將多樣的物件集合到一起，不能修改，通過索引進行查找。	把一些資料當作一個整體去使用，不能修改。
dict	{ }	是一組 <u>鍵</u> 和 <u>值</u> 的組合，通過 <u>鍵</u> 進行查找，沒有順序， <u>鍵</u> 不能重複。	使用 <u>鍵</u> <u>值</u> 和 <u>值</u> 進行關聯的資料。
set	()或 { }	無序，元素只出現一次，自動去除重複。	資料只出現一次，只關心資料是否出現，不關心其順序與位置。

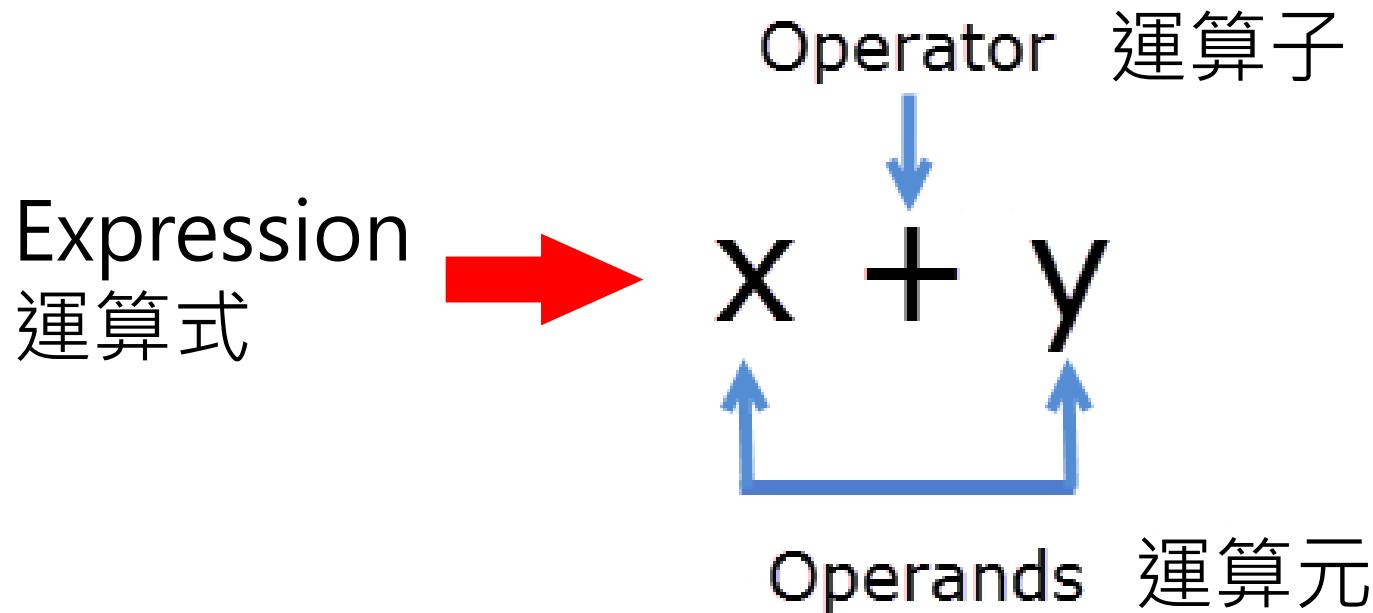
休息一下~



3png.com

基本運算式

- 運算子(Operator)與運算元(Operand)



運算符號

- 正負號：

符號	意義	範例
+	正號	+ a
-	負號	- a

- 算術運算子

運算子	功能	範例
+	加	a + b
-	減	a - b
*	乘	a * b
/	除	a / b
%	取餘數	a % b
**	指數	a ** b
//	整除(向下取整數)	a // b

運算符號

- 邏輯運算子：

運算子	功能	範例
and	且	a and b
or	或	a or b
not	非	not a

- 真值表：

and		
a	b	a and b
0	0	0
0	1	0
1	0	0
1	1	1

or		
a	b	a or b
0	0	0
0	1	1
1	0	1
1	1	1

not	
a	not a
0	1
1	0

0：代表假
1：代表真

運算符號

- 關係運算子：

運算子	功能	範例
<code>==</code>	相等	<code>a == b</code>
<code>!=</code>	不相等	<code>a != b</code>
<code>></code>	大於	<code>a > b</code>
<code>>=</code>	大於等於	<code>a >= b</code>
<code><</code>	小於	<code>a < b</code>
<code><=</code>	小於等於	<code>a <= b</code>

- 成員運算子：

運算子	功能	範例
<code>in</code>	如果在指定的序列中找到值返回 True，否则返回 False。	<code>a in list</code>
<code>not in</code>	如果在指定的序列中没有找到值返回 True，否则返回 False。	<code>a not in list</code>

運算符號

- 位元運算子：

運算子	功能	範例
&	and	$a \& b$
	or	$a b$
^	xor	$a ^ b$
~	not	$\sim a$
<<	位元左移	$a << 2$
>>	位元右移	$a >> 2$

- 位元運算是對運算子的"二進制"每個Bit去做運算，大家平常可能比較少用到，請複習一下數位科技概論。

運算符號

- 指派運算子：

運算子	功能	範例
=	指派	$a = b$
+ =	相加同時指派	$a += b$
- =	相減同時指派	$a -= b$
* =	相乘同時指派	$a *= b$
/ =	相除同時指派	$a /= b$
% =	取餘數同時指派	$a \% = b$
** =	取指數同時指派	$a ** = b$

- 即 $a = a + b$ 可簡寫成 $a += b$

(把a的值取出來和b相加後，再將結果存回a。就是將b累加到a裡面去的意思)

運算符號的優先順序

- 運算子是有先乘除後加減這類的優先順序的。

優先權	運算子	說明
1	()	括弧
2	**	指數
3	not、-	邏輯運算子 not、負號
4	*、/、%、//	算術運算子的乘法、除法、餘數、商數
5	+、-	算術運算子的加法和減法
6	>、>=、<、<=	關係運算子的大於、大於等於、小於和小於等於
7	$==$ 、 $!=$	關係運算子的等於和不等於
8	and、or	邏輯運算子的 and、or
9	=	指定運算子

運算符號

- **+** 與 ***** 也可用於字串(string)，例如
`b = "a" + "b" #字串連接，b的值是"ab"`
`c = "ab" * 2 #字串重複2次，c的值是"abab"`
- **=** 也可以用平行指派(parallel assignment)

例如：

`a, b, c = 11, 22, 33`

a 會等於 11
b 會等於 22
c 會等於 33

`a = b = c = 1`

a、b、c都等於 1

- 可適度的使用括符()來區分優先順序。

註解

- 好的程式需要有好的說明，這些穿插在程式中的說明稱為註解。
- 註解是給人看的，電腦會予以忽略。
- 單行註解：
 使用#符號，#之後的文字皆視為註解。
- 多行註解：
 使用'''...'''(連續三個單引號)框住的段落，可以多行。

註解

- 註解範例：
 - 單行註解：用 `#` 開頭的部分至行尾

`# This is a comment`，這是註解.....

`print("Hello") #This is a comment...`

- 多行註解：連續三個單引號`'''`框住的範圍

`'''`

這是 \$%隨便%^&(\$^%

多行 !@\$你^&%^

註解 ^&*%要說什麼@#&

`'''`

運算式寫法

- 在任何程式語言都不接受什麼上標、下標、改大小等等一般文書處理的功能，只能單純的寫成純文字，那數學式子怎麼寫成程式的表示方式呢？
- 利用各種運算子和括符()來解決。

• 例： $x = 2\frac{1}{y}$ 要寫成 $x = 2*(1/y)$

$y = ax+b$ 要寫成 $y = a*x+b$

$x = a^2 + b^2$ 要寫成 $x = a**2 + b**2$

運算式寫法

- 練習：

$$x = 2^{\frac{1}{2}} \rightarrow x = 2^{**}(1/2) \text{ 或 } x = 2^{**}0.5$$

$$x = \sqrt{a^2 + b^2} \rightarrow x = (a^{**2} + b^{**2})^{**0.5}$$

$$x = \frac{a+b}{c-d} \rightarrow x = (a+b)/(c-d)$$

$$c = (f-32)\frac{5}{9} \rightarrow c = (f-32)*5/9$$

運算式寫法

- 自己練習：

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \rightarrow$$

$$n = ax^2 + bx + c \rightarrow$$

$$y = 1 + \frac{x}{2} + \frac{x^2}{3} + \frac{x^3}{4} \rightarrow$$

$$z = \sqrt[3]{a + \frac{x}{y}} \rightarrow$$

基本輸出指令

- 輸出函數print()：輸出訊息到螢幕。
 - Ex: `print("Hello World!")`
輸出Hello World!並自動加上換行。
 - Ex: `print(a, b, end='')`
輸出變數a和b的內容，依end指定的字元當結尾，不會自動加上換行。
 - Ex: `print(a, '\t', b)`
輸出變數a的內容，然後一個跳格的跳脫字元，再輸出變數b的內容，然後換行。
 - Ex: `print(a + b * c)`
也可以直接印出運算的結果，變數a、b、c必須已經存在且存放了數值。

跳脫字元

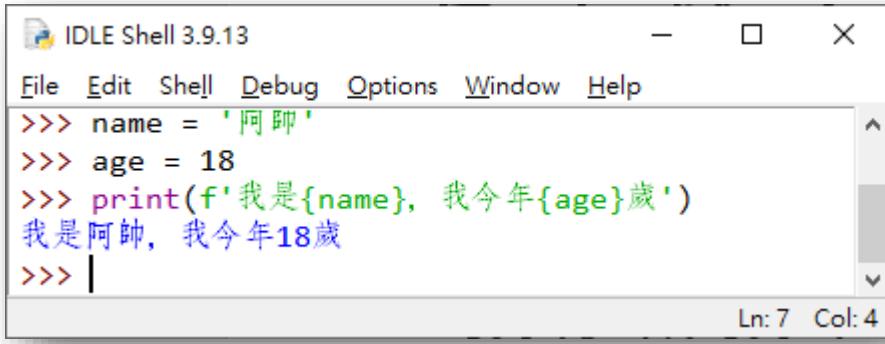
- 對輸出有影響的特殊字串組合(跳脫字元)：

跳脫字元	說明
\\"	印出反斜線「\」
\''	印出單引號「'」
\\""	印出雙引號「"」
\n	換行字元
\a	發出「嗶」聲
\t	水平跳格字元 (Tab)
\b	倒退一格字元 (Backspace)
\f	跳頁字元

基本輸出指令

- 如果要輸出的資料包含複雜的變數和字串，還可以使用以下的方式：
- 在要輸出的字串前加一個 **f**，然後將變數放在 { } 之間即可。

```
name = '阿帥'  
age = 18  
print(f"我是{name}，我今年{age}歲")
```



The screenshot shows the Python IDLE Shell interface. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The command history window displays three commands:
>>> name = '阿帥'
>>> age = 18
>>> print(f"我是{name}，我今年{age}歲")
The final command's output is shown below it: "我是阿帥，我今年18歲". The status bar at the bottom right indicates Ln: 7 Col: 4.

基本輸出指令

- 如果需要精確的控制輸出格式，可用%格式化字串。
- Ex：

```
print("在%s中，PI約為%5.2f" % ("Python", 3.14159))
```

要輸出的字串

用%分隔輸出
字串與參數

要輸出的參數，如
果一個以上要用()括起來。

- 輸出結果：



基本輸出指令

- % 格式化字元：

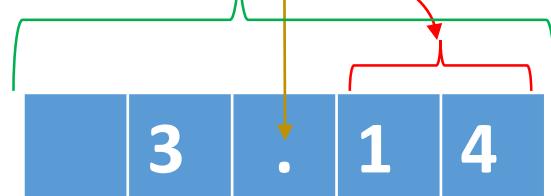
- 對應不同的資料型態要使用各自的格式字元。

- 前例用到的

- `%d`：格式化整數輸出。
- `%f`：格式化浮點數輸出。
- `%x`：格式化 16 進位整數輸出。
- `%X`：格式化大寫 16 進位整數輸出。
- `%o`：格式化 8 進位整數輸出。
- `%s`：格式化字串輸出。
- `%e`：格式化科學記號 e 的輸出。
- `%E`：格式化科學記號大寫 E 的輸出。

基本輸出指令

- %格式化的控制方式：
- Ex : **%5.2f** 表示總共留5格，小數佔用2格。
(小數點固定占用1格)



- Ex : **%10d** 表示總共留10格，向**右**靠齊。
- Ex : **%-10d** 表示總共留10格，向**左**靠齊。

基本輸入指令

- 輸入函數 `input()`：自鍵盤取得使用者輸入。
- Ex: `tel = input("請輸入你的電話：")`
取得使用者輸入的字串後存入變數 `tel`。
- Ex: `age = input("你幾歲？")`
取得使用者輸入的字串存入變數 `age`。

基本輸入指令

- 所有輸入皆視為字串，若需數值須用函式轉換。
- Ex: `age = int(input("你幾歲?"))`
使用者輸入的字串轉成整數存入變數age。
- Ex: `BMI = float(input("你的BMI?"))`
使用者輸入的字串轉成浮點數存入變數BMI。
- Ex: `num = eval(input("請輸入數字:"))`
不論是整數或浮點數都會自動轉型成可以做計算的數值。

基本輸入指令

休息一下~

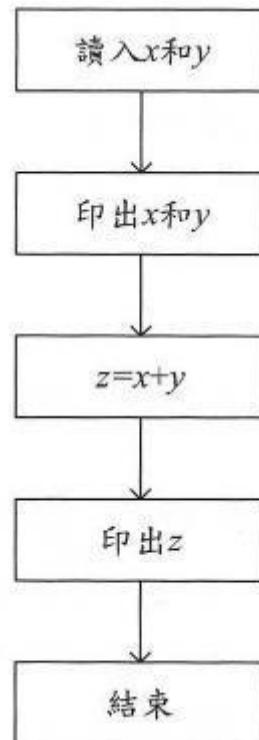


簡單的程式

- 整數加法：

假設我們有兩個整數 x 和 y ，要求輸出 $x+y$ 的結果。

- 流程圖：



簡單的程式

- 參考程式：

```
x = int(input("請輸入數值x: "))      #請求輸入，存到x
y = int(input("請輸入數值y: "))      #請求輸入，存到y
print("x =", x)                      #印出"x ="和x的內容
print("y =", y)                      #印出"y ="和y的內容

z = x + y                            #執行運算

print("z =", z)                      #印出"z ="和z的內容
```

- 說明：

- 1. 深紅色的部分稱"字串"，會忠實地顯示在結果畫面(螢幕)上。
- 2. x、y、z稱"變數"，print()時會以它們的值呈現。
- 3. input()是請求鍵盤輸入，print()是印出資訊。
- 4. int()是將輸入的字串轉成整數(鍵盤輸入的都視為"字串")。

簡單的程式

- 執行情形：

The image shows two windows from the Python IDLE environment. On the left is the 'test.py' script editor window, containing the following code:

```
x = int(input("請輸入數值x: "))
y = int(input("請輸入數值y: "))
print("x value:", x)
print("y value:", y)

z = x + y

print("z value:", z)
```

A red arrow points from the bottom of the script editor towards the 'Run Module F5' button in the menu bar. Another red arrow points from the bottom of the menu bar towards the right-hand shell window.

The right window is the 'IDLE Shell 3.9.13' window, displaying the output of running the script. It shows the following interaction:

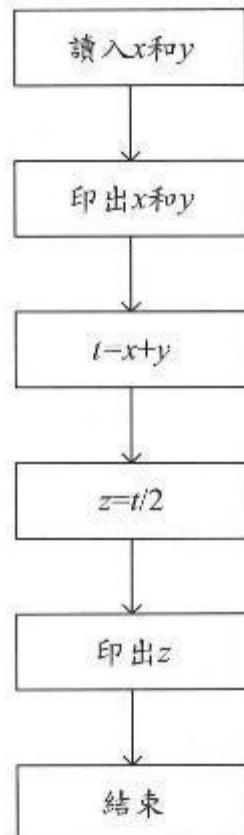
```
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:42
) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=====
RESTART: F:\TmpF\test.py
=====
請輸入數值x: 14
請輸入數值y: 53
x value: 14
y value: 53
z value: 67
>>>
```

簡單的程式

- 計算 $(x+y)/2$ ：

假設我們有兩個整數 x 和 y ，要求輸出 $(x+y)/2$ 的結果。

- 流程圖：



簡單的程式

- 參考程式：

```
x = int(input("請輸入數值x: "))
y = int(input("請輸入數值y: "))
print("x =", x)
print("y =", y)

t = x + y
z = t / 2

print("z =", z)
```

- 說明：

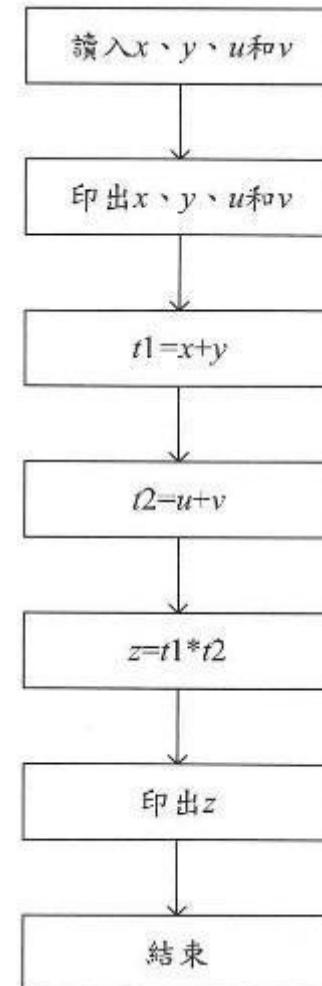
- 執行時空白行會予以忽略，空行是為了程式美觀或易於閱讀。

簡單的程式

- 計算 $(x+y)*(u+v)$ ：

假設我們有四個整數 x, y, u 和 v ，
要求 $(x+y)*(u+v)$ 。

- 流程圖：



簡單的程式

- 參考程式：

```
x = int(input("請輸入數值x: "))
y = int(input("請輸入數值y: "))
u = int(input("請輸入數值u: "))
v = int(input("請輸入數值v: "))

print("x =", x)
print("y =", y)
print("u =", u)
print("v =", v)

t1 = x + y
t2 = u + v
z = t1 * t2

print("z =", z)
```

簡單的程式

- 浮點數字加法：
 - 用 `float()` 取代 `int()` 即可。

```
x = float(input("請輸入數值x: "))
y = float(input("請輸入數值y: "))
print("x =", x)
print("y =", y)

z = x + y

print("z =", z)
```

- 再次強調：鍵盤輸入的東西一律視為字串，如果你要做計算，要轉換成你要的型別，例如整數或浮點數。

簡單的程式

- 若一次要輸入多個相同型態的資料，可用下面的寫法：

```
x, y = map(int, input("請輸入身高，體重：").split())
```

④ 依序分配給變數

① input敘述

③ 轉成指定的型態，
此為整數

② 將輸入依指定的符號
分開，此處為空白

- split()會把輸入依指定的符號分開。
- map()會把一個個輸入轉換成指定的型態，再依序分配給變數。

簡單的程式

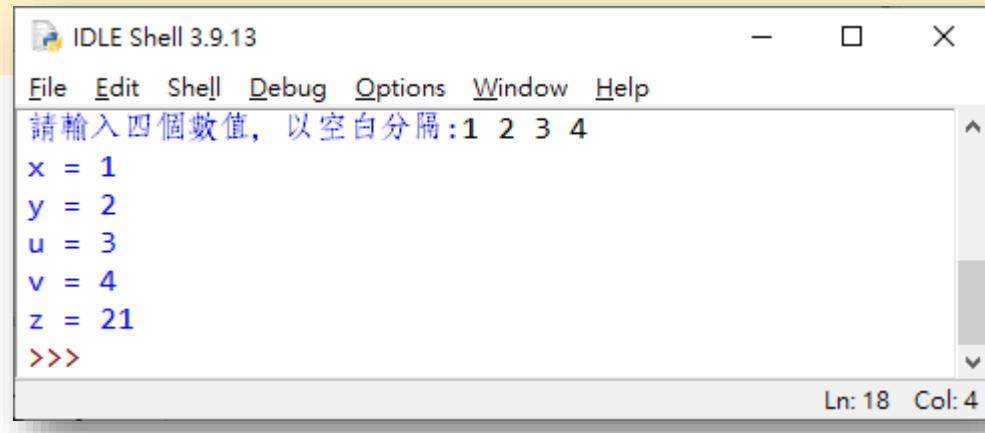
- 所以前例可改寫成如下：

```
#一次輸入四個數值
x,y,u,v = map(int,input("請輸入四個數值，以空白分隔:").split())

#只使用一個print指令
print(f"x = {x}\ny = {y}\nu = {u}\nv = {v}")

z = (x + y) * (u + v)

print("z =", z)
```



簡單的程式

- 另外input()裡面也可以是空的，只是沒有提示訊息容易造成使用者困擾，程式應盡量將訊息表達清楚。
- 例如：

The screenshot shows the Python IDLE environment. On the left, a script editor window titled "1-5.py" contains the following code:

```
x=int(input())
y=int(input())

z = x + y

print(z)
```

A callout bubble from the first line of the script says: "input()裡沒有任何提示訊息" (There is no prompt message in the input()).

In the center, an "IDLE Shell 3.9.13*" window is open. It displays the Python interpreter's prompt and some initial information:

```
Python 3.9.13 (tags/v3.9.13/f13b6d4b652168d4ca06516d49d09cf6, Dec 13 2022, 10:45:50) [MSC v.1932 64 bit (AMD64)] on win32
Type "help", "pyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:/TmpF/test1.py ======
```

A callout bubble from the shell window says: "程式已經在執行了，卻沒有任何提示要使用者做什麼，這樣很不優" (The program is already running, but there is no prompt for the user to do anything, which is very poor).

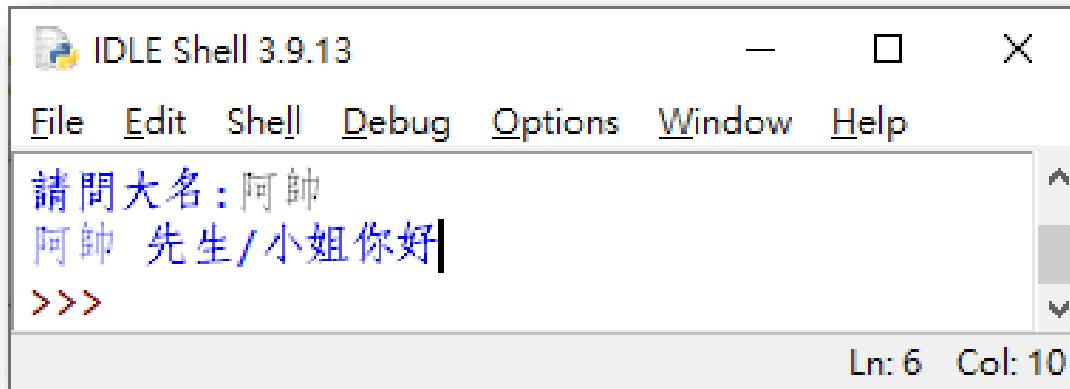
簡單的程式

- 輸入文字：

如果要輸入文字，簡單的用 `input()` 即可。

```
name = input("請問大名:")
print(name, "先生/小姐你好", )
```

- 執行結果：



簡單的程式

- 計算二次方 $a^2 + b^2$:

```
a = int(input("請輸入 a = "))  
b = int(input("請輸入 b = "))  
c = pow(a, 2) + pow(b, 2)  
print("c = ", c)
```

或

```
a = int(input("請輸入 a = "))  
b = int(input("請輸入 b = "))  
c = a**2 + b**2  
print("c = ", c)
```

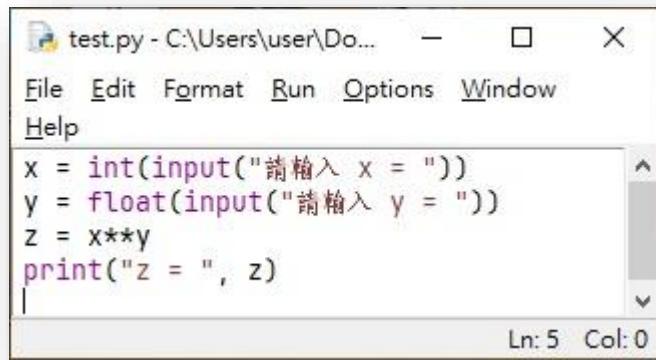
- 以上結果是相同的。

- 計算次方可以用函數pow()或運算符號 **。
- 例如計算 x^y ，可寫成pow(x,y)或 $x^{**}y$ 。

簡單的程式

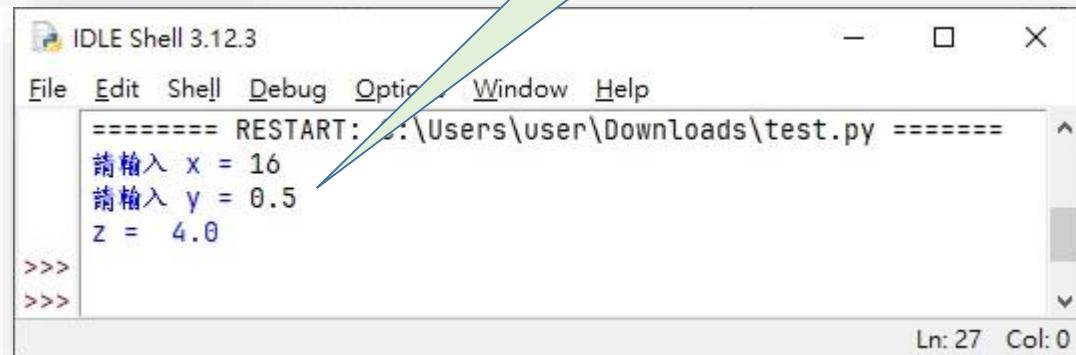
- 小於1的次方：

計算 x^y ，但 y 為介於 0 到 1 之間的浮點數。



```
test.py - C:\Users\user\Do...  —  X
File Edit Format Run Options Window
Help
x = int(input("請輸入 x = "))
y = float(input("請輸入 y = "))
z = x**y
print("z = ", z)
Ln: 5 Col: 0
```

輸入浮點數



```
IDLE Shell 3.12.3
File Edit Shell Debug Options Window Help
===== RESTART: C:\Users\user\Downloads\test.py =====
請輸入 x = 16
請輸入 y = 0.5
z = 4.0
>>>
>>>
Ln: 27 Col: 0
```

簡單的程式

- 計算 $\log(x)$ ：

需要引入math套件

```
import math  
x = int(input("請輸入 x： "))  
y = math.log(x)  
print("y = ", y)
```

- 在這裡 $\log(x)$ 是以 $e = 2.718$ 為底。
- 某些特別的函數需要另外引入，我們用**import**指令，這裡是引入數學函數，所以用：**import math**，然後用**math.log()**函數。

你可以想成是"math的log()"

簡單的程式

- 計算平方根 $y = \sqrt{x}$ ：
 - 1. 以import方式，使用sqrt()函數開根號。

```
import math
x = int(input("請輸入x= "))
y = math.sqrt(x)
print("y = ", y)
```

- 2. 不使用引入，直接用次方運算符號。

```
x = int(input("請輸入 x = "))
y = x**0.5
print("y = ", y)
```

- 說明：開根號是 $\frac{1}{2}$ 次方，即0.5次方。

簡單的程式

- 計算 $y = \sin(x)$ ：

```
import math
x = float(input("請輸入 x = "))
y = math.sin(x * math.pi)
print("y = ", y)
```

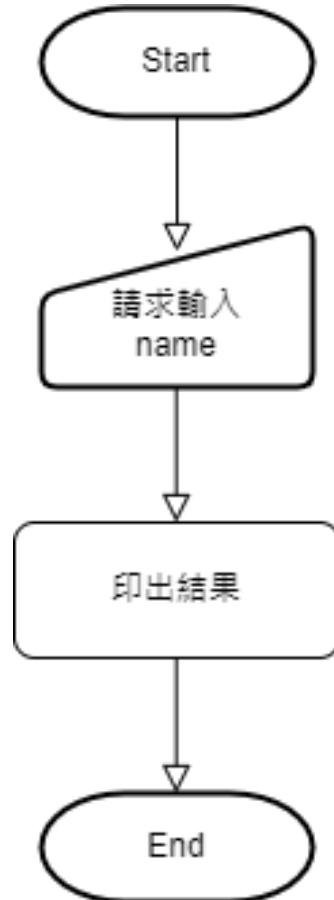
- `math.sin()`、`math.pi`都是`math`提供的函數。
- Python的 π 精確到小數點後15位。
(約為3.141592653589793)

練習題

- 3-1. 請使用者輸入它的暱稱，然後向他問好。例如使用者輸入"阿美"，你要印出"你好，阿美"。先畫出流程圖。
- 3-2. 寫一程式，輸入 a, b, c, d ，計算 $\frac{(a+b)}{(c-d)} \times d$ 。先畫出流程圖。

練習題

- 3-1流程圖：



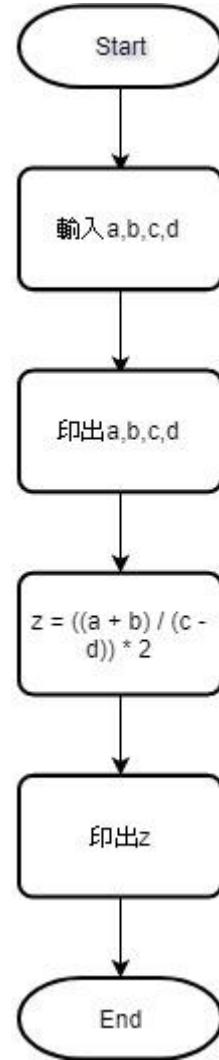
練習題

- 3-1 參考程式：

```
#請求輸入  
name = input("請問您的大名:")  
  
#印出結果  
print("你好，", name)
```

練習題

- 3-2流程圖：



練習題

- 3-2參考程式：

```
a = float(input("請輸入數值a: "))
b = float(input("請輸入數值b: "))
c = float(input("請輸入數值c: "))
d = float(input("請輸入數值d: "))

print("a =", a)
print("b =", b)
print("c =", c)
print("d =", d)

z = ((a + b) / (c - d)) * d

print("((a + b) / (c - d)) * d =", z)
```

- 誰能把它簡化一些？

練習題

- 3-3. 輸入民國幾年，印出其西元年。(提示：
將民國幾年加上1911，即可得到其對應的
是西元幾年)
- 3-4. 華氏溫度F可以表示成 $F=C*9/5+32$ 。現在
給你攝氏C的溫度，請問它是華氏幾度？
- 3-5. 紿你一個矩形的長和寬，請問它的面積是
多少？

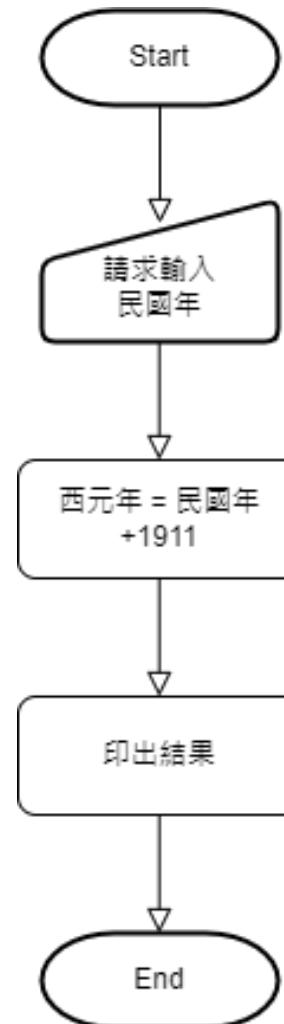
練習題

- 3-3參考程式：

```
year = int(input("請輸入民國年:"))
```

```
AD = year + 1911
```

```
print(f"民國{year}年是西元{AD}年")
```



練習題

- 3-4參考程式：

```
C = int(input("請輸入攝氏度："))

print(C * 9 / 5 + 32)
```

$$\begin{aligned}{}^{\circ}\text{F} &= \frac{9}{5}({}^{\circ}\text{C}) + 32 \\ {}^{\circ}\text{C} &= \frac{5}{9}({}^{\circ}\text{F} - 32)\end{aligned}$$

練習題

- 3-5參考程式：

```
w = float(input("請輸入矩形長:"))
h = float(input("請輸入矩形寬:"))

area = w * h

print("面積 = ", area)
```

休息一下~

- 電腦之所以稱為 "**Computer**" 是因為它當初發明的目的就是為了幫人類計算，以後你會發現 "程式" 這個東西很多時候都在做計算的工作。



Python重要規則

- 在繼續之前你一定要記得：
- Python要求嚴格的縮排
 - 同一區段該縮排要縮排，要確實對齊。
 - 一般以二或四個空白，也可以用Tab縮排，但建議不要空白和Tab混用。

選擇 / 判斷結構

- 如果在程式中遇到要做選擇 / 判斷時，就要使用 **if** 指令。



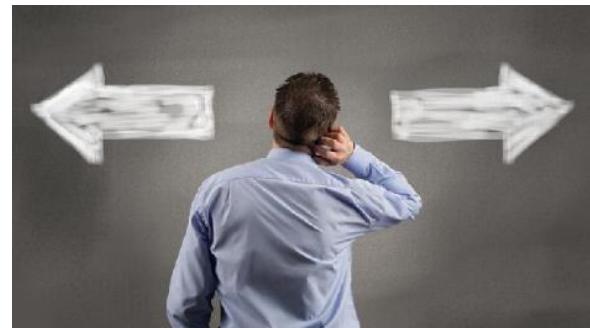
選擇 / 判斷結構

- 依照我們給的條件來判斷要不要執行某一段程式。
- if 寫法有三種：

if ... (單選)

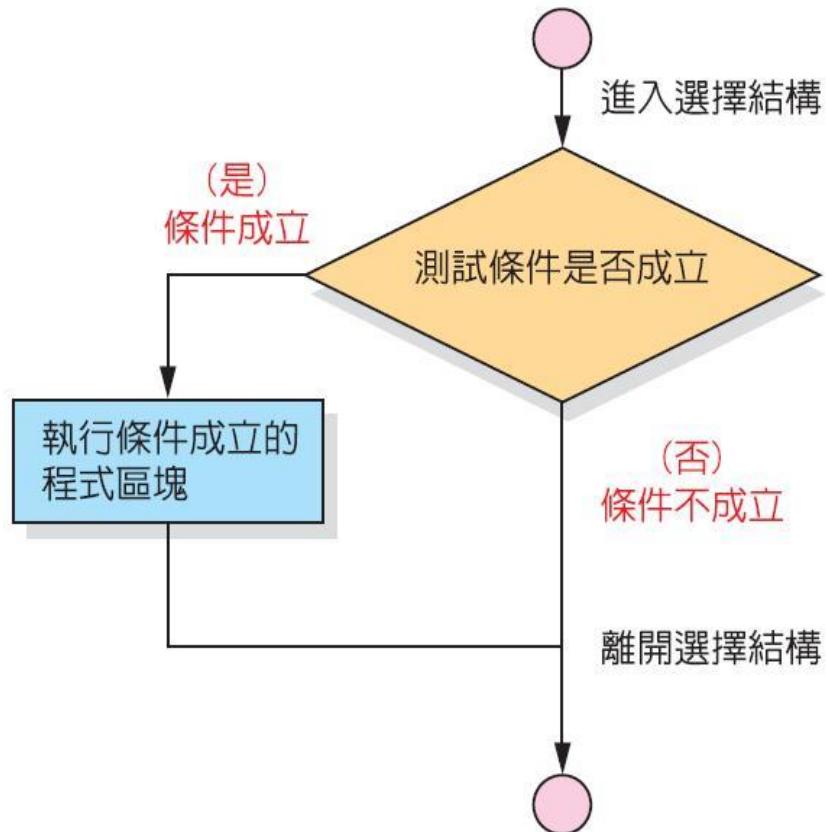
if ... else ... (二選一)

if ... elif ... else ... (多重選擇)



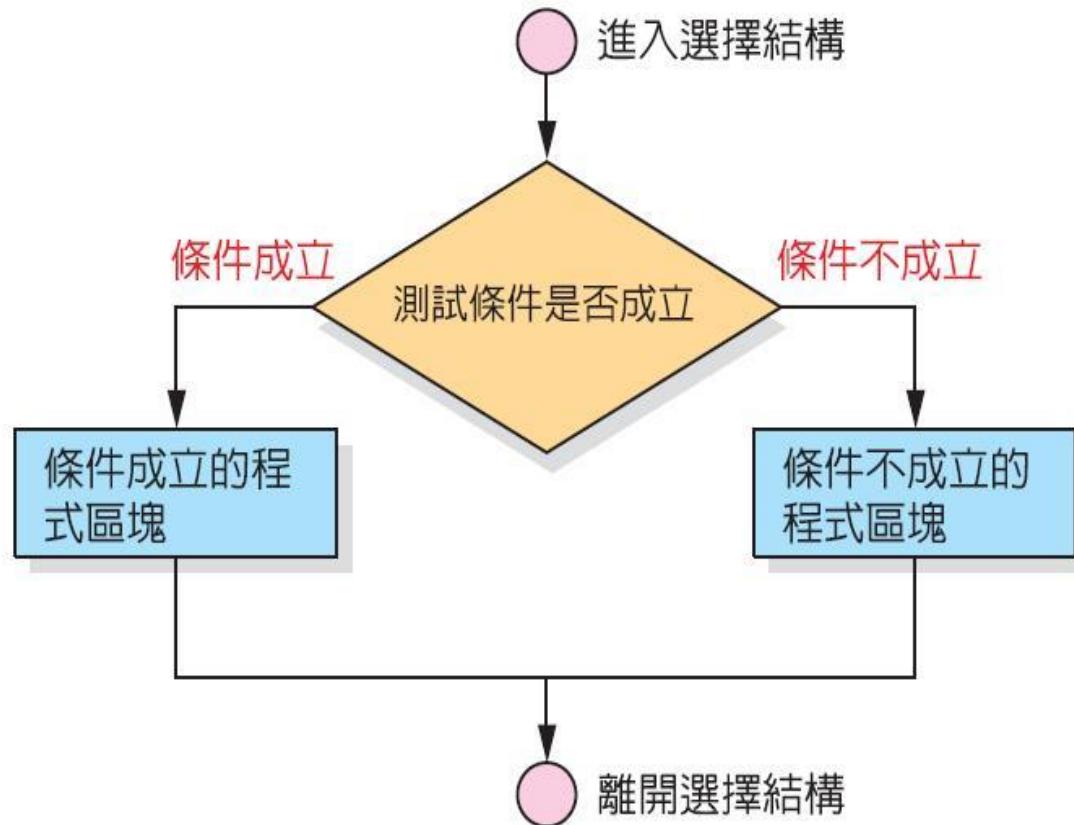
選擇 / 判斷結構

- **if** 指令是：「如果怎樣怎樣(測試條件)，你就如何如何(程式區塊)」。



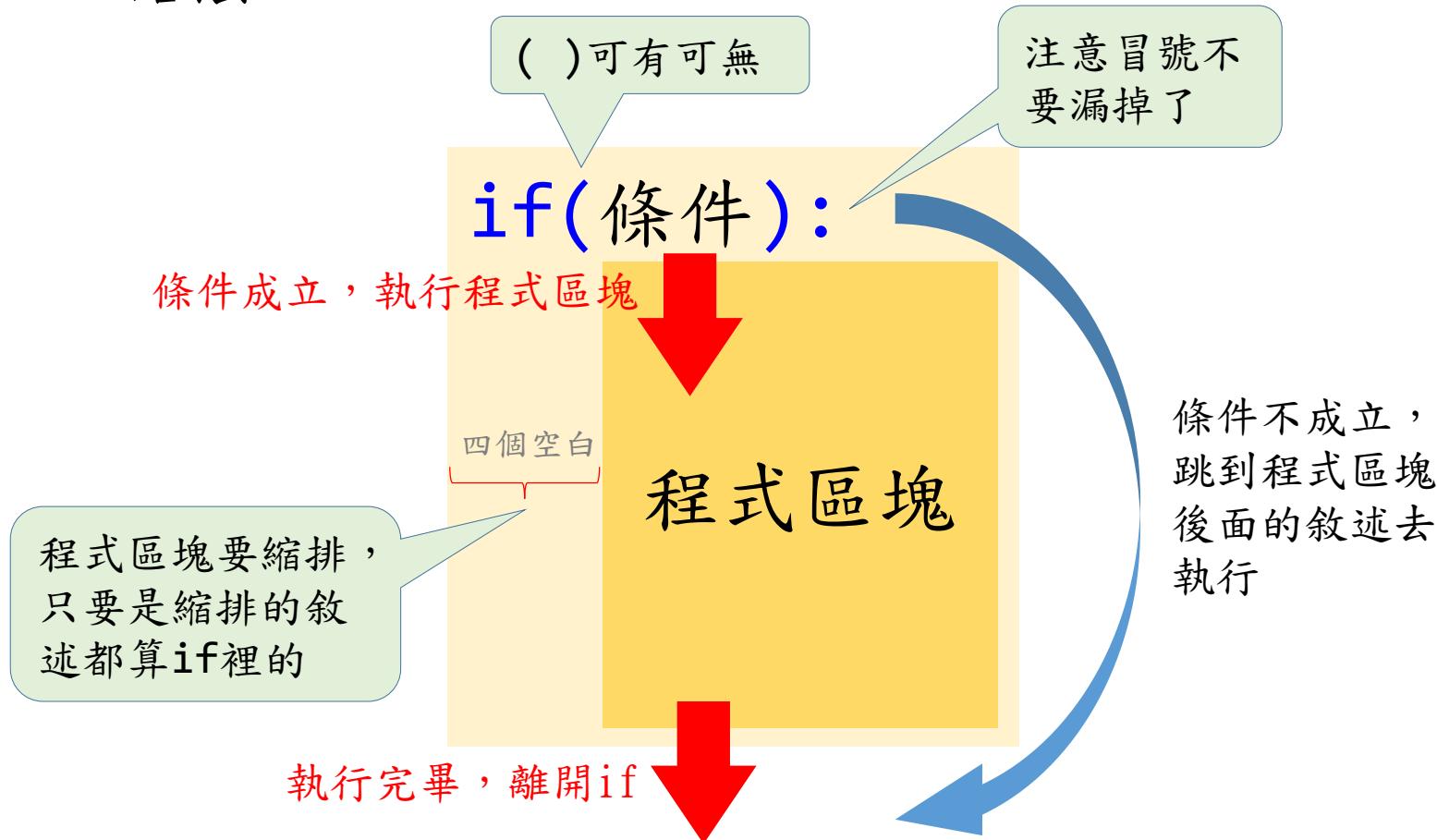
選擇 / 判斷結構

- `if...else...` 是：「如果怎樣怎樣，你就這樣這樣，否則就那樣那樣」。



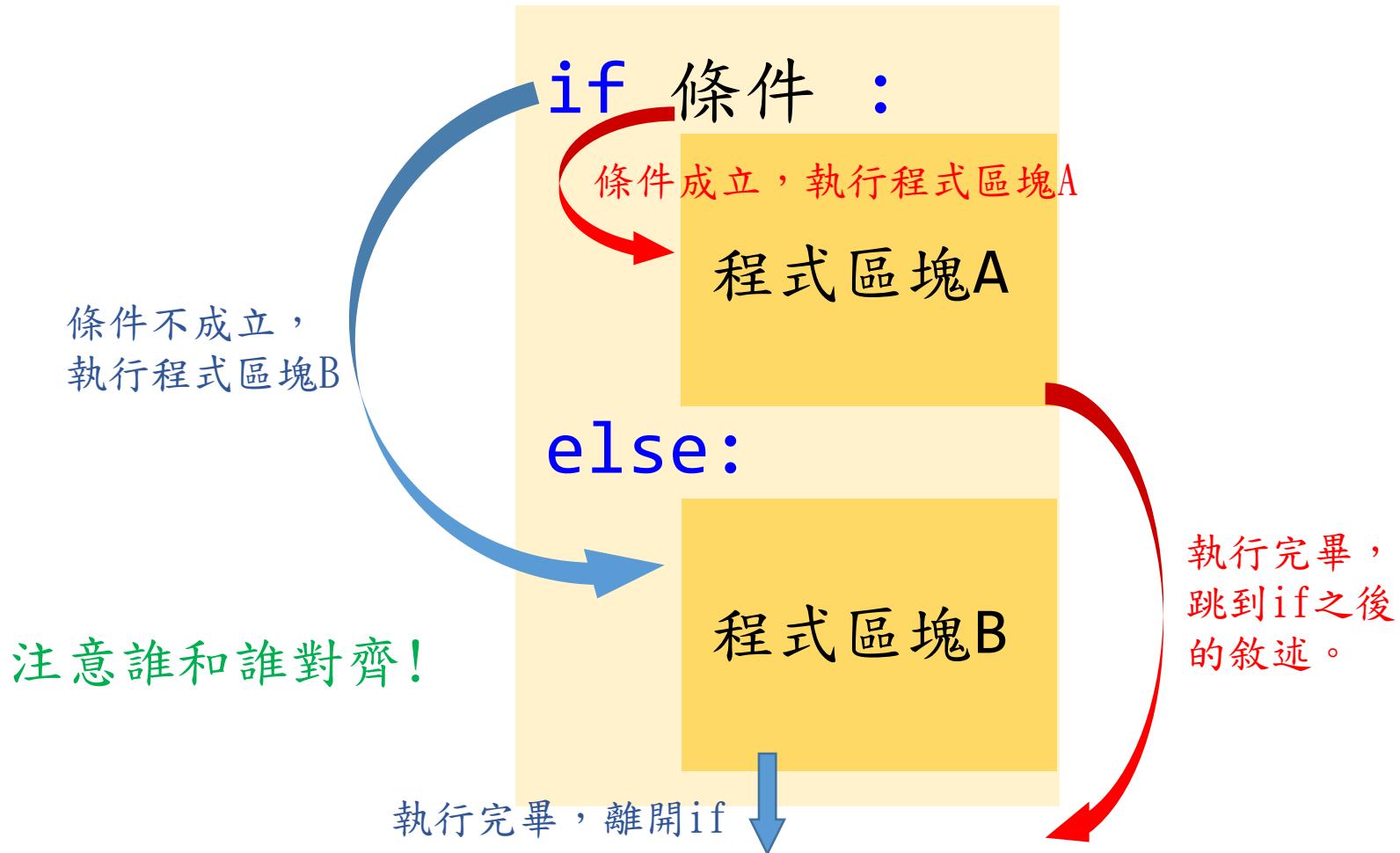
選擇 / 判斷結構

- **if**語法：



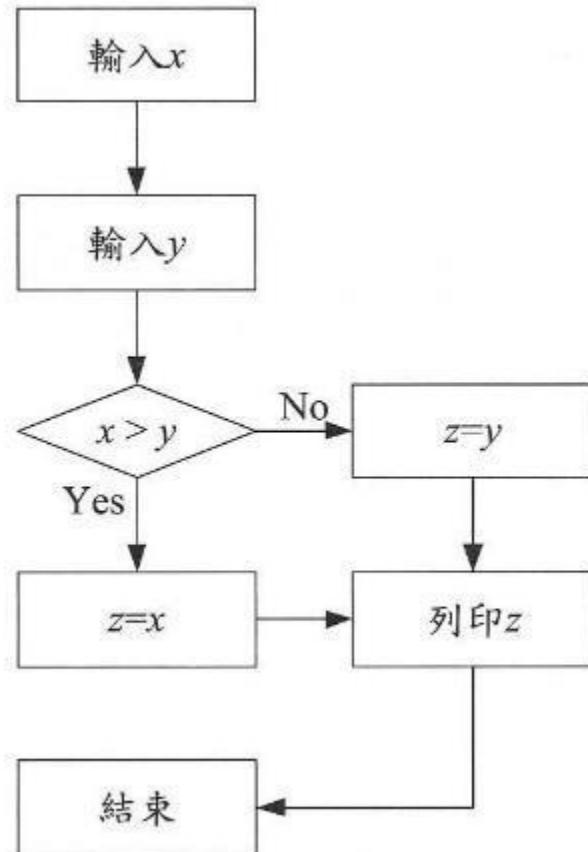
選擇 / 判斷結構

- if...else語法：



選擇 / 判斷結構

- 讀入兩個數 x 和 y ，選出大的那一個。
- 流程圖：



選擇 / 判斷結構

- 參考程式：

```
x, y = map(int, input("請輸入數值x和y: ").split())
print("x = ", x)
print("y = ", y)

if x > y:
    z = x
else:
    z = y

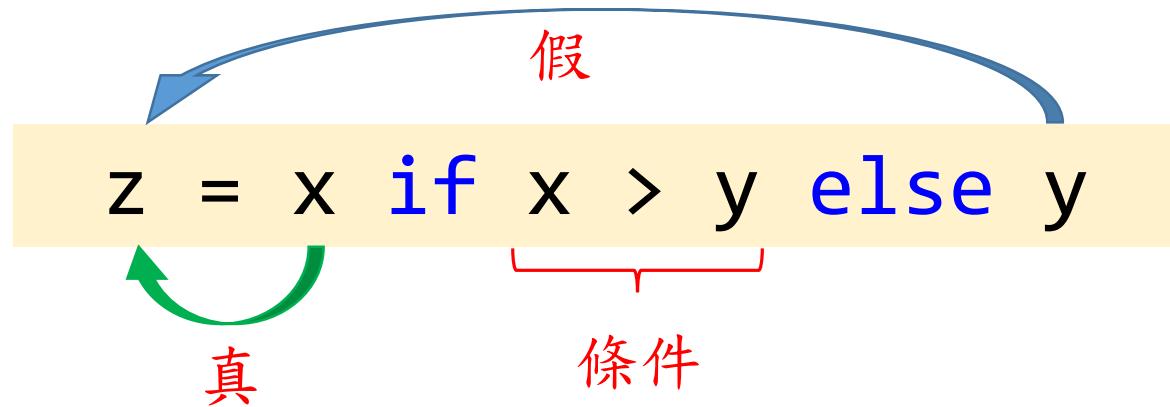
print("Max = ", z)
```

注意冒號及縮排！
if和else要對齊

- 再次強調：程式中縮排的空格數目必須相同，一般用2或4格空白，不要和Tab混用。

選擇 / 判斷結構

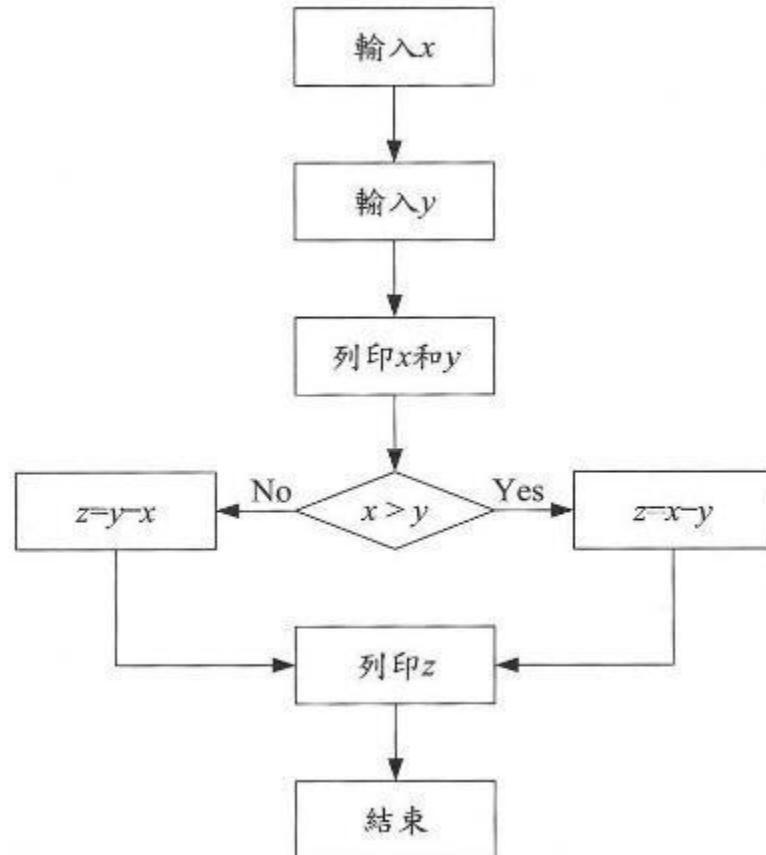
- 一行式的寫法(`else`不可省略)：



- 說明：
 - 如果判斷式很簡單如前例，可以採用這個「很 Python」的寫法。
 - 意思是：將`x`指定給`z`，如果條件為真的話。否則將`y`指定給`z`。

選擇 / 判斷結構

- 讀入 x 和 y ，如果 $x>y$ ，則回傳 $x-y$ ，否則回傳 $y-x$ 。
- 流程圖：



選擇 / 判斷結構

- 參考程式：

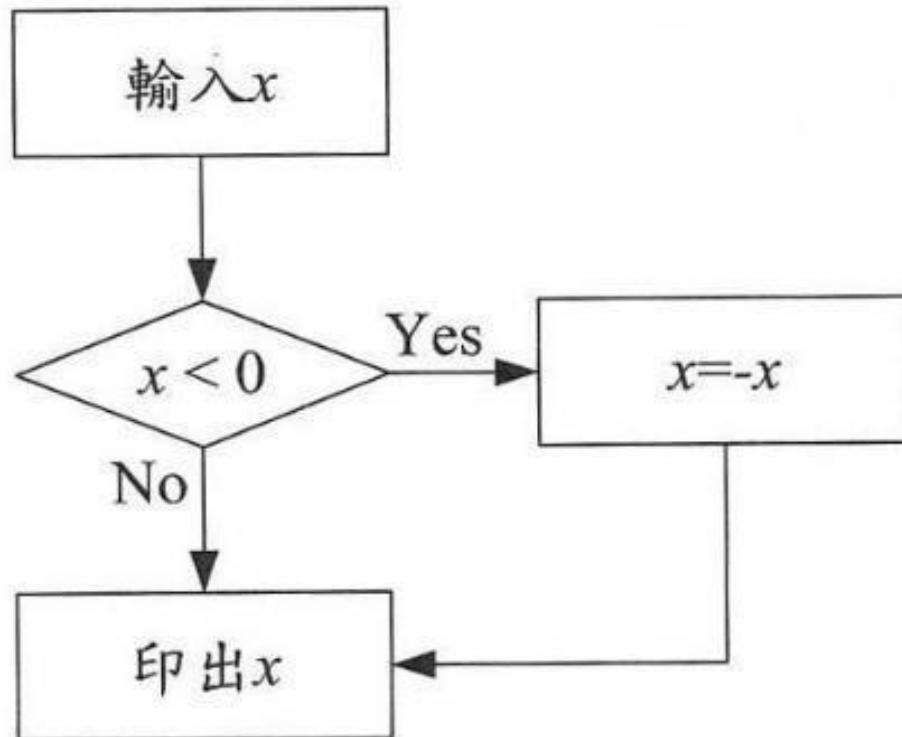
```
x, y = map(int, input("請輸入數值x和y: ").split())
print("x = ", x)
print("y = ", y)

z = (x - y) if x > y else (y - x)

print("z = ", z)
```

選擇 / 判斷結構

- 讀入一數字，如果是負數，就將它轉成正數。
- 流程圖：



選擇 / 判斷結構

- 參考程式：

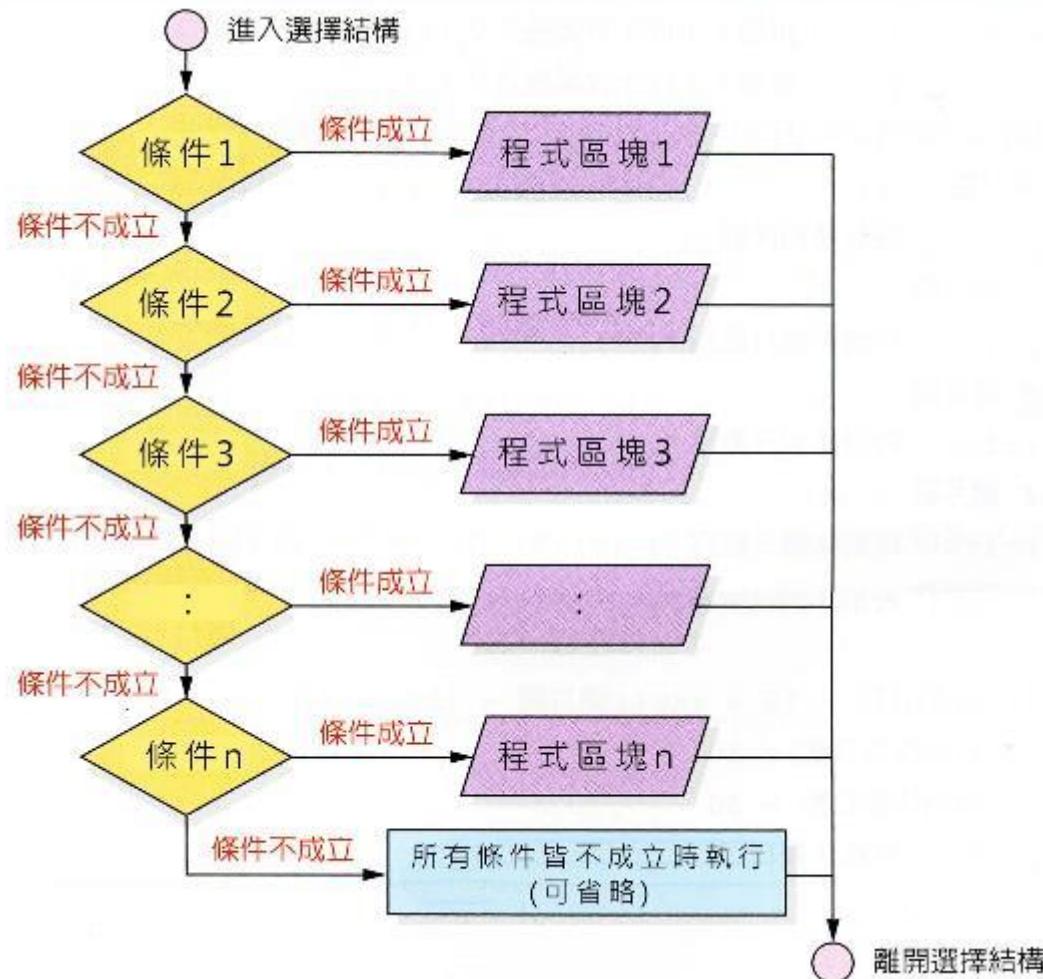
```
x = int(input("請輸入數值x: "))

print("x =", x if x > 0 else -x)
```

- 對print()指令來說，它最後會收到x或-x來輸出。

選擇 / 判斷結構

- 多重選擇：



選擇 / 判斷結構

- 語法：

哪個條件成立就執行那個區塊。
`elif`的個數沒有限制。

若以上條件都不成立，就執行`else`區塊。

若也沒有`else`區塊，那就什麼都不執行。

`else`區塊可省略。

`if 條件1 :`

 程式區塊1

`elif 條件2 :`

 程式區塊2

`elif 條件3 :`

 程式區塊3

 : :

`elif 條件n :`

 程式區塊n

`else:`

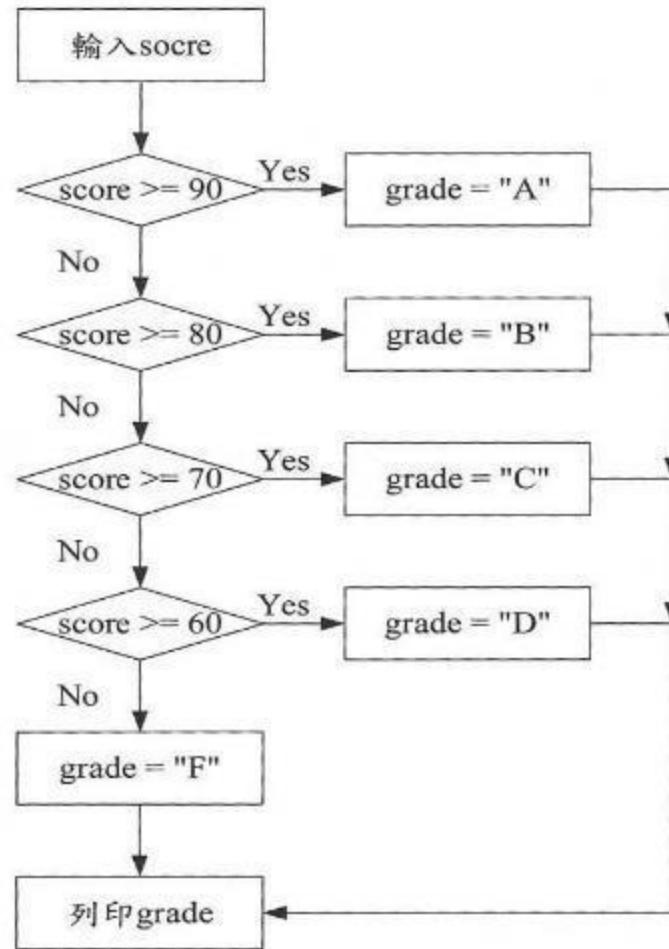
 程式區塊

執行完區塊後離開`if`。

選擇 / 判斷結構

- 輸入一個學生的分數，輸出該成績級別，分數級別如下：

分數	級別
90分以上	A
80~89分	B
70~79分	C
60~69分	D
60分以下	F



- 流程圖如右：

選擇 / 判斷結構

- 參考程式：

```
print("輸入學生成績，將會輸出對應的成績級別")
score = eval(input("輸入成績(0~100): "))
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "F"
print("此成績級別為 ", grade)
```

選擇 / 判斷結構

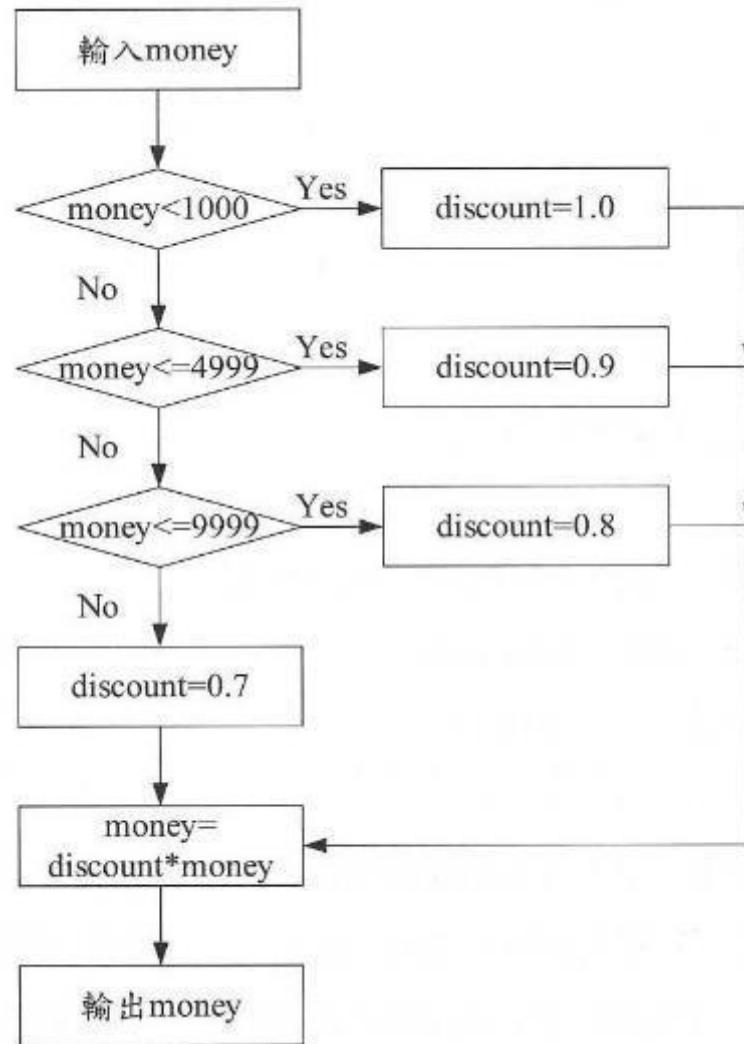
- 某商店在大特價，根據購買的商品總金額決定打折的折扣，總金額越高折扣越多，折扣表如下：

商品總金額	折扣
未滿1000	無折扣
1000 ~ 4999	9折
5000 ~ 9999	8折
高於10000	7折

- 小數點四捨五入，印出折扣後的金額。

選擇 / 判斷結構

- 流程圖：



選擇 / 判斷結構

- 參考程式：

```
print("輸入總金額，將會依據金額打折，並輸出打折後的金額")
money = int(input("請輸入購買總金額："))

if money < 1000:
    print("抱歉，沒有折扣唷!")
    discount = 1.0
elif money < 4999:
    print("享有9折優惠唷!")
    discount = 0.9
elif money < 9999:
    print("享有8折優惠唷!")
    discount = 0.8
else:
    print("享有7折優惠唷!")
    discount = 0.7

money = int(round(money * discount, 0))
print("打折後總金額為：", money)
```

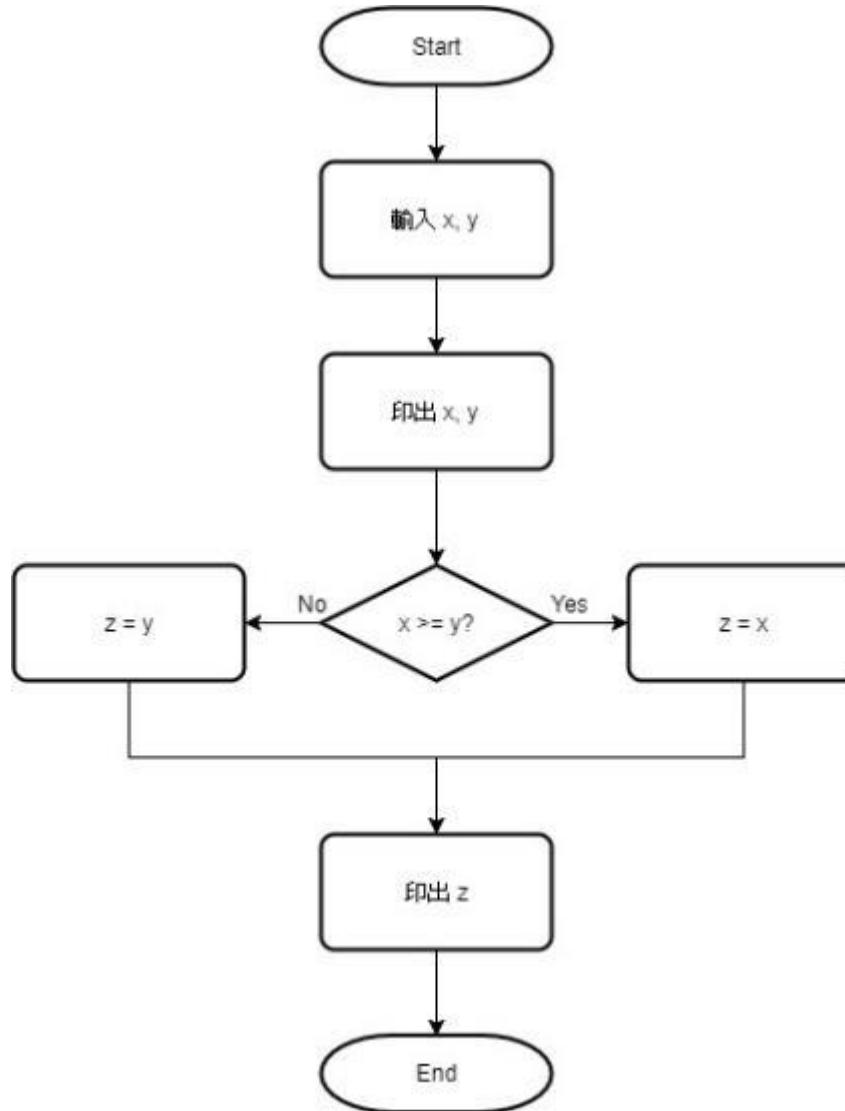
函數**round(n, m)**
將n值在小數點第m
位四捨五入

練習題

- 以下各題請先畫出流程圖。
- 4-1. 寫一程式，輸入 x 和 y ，如果 $x \geq y$ ，則列印 x ，否則列印 y 。
- 4-2. 寫一程式，輸入 x 和 y ，如果 x 和 y 都是正數，令 $z=1$ ，如兩者均為負數，令 $z=-1$ ，否則令 $z=0$ 。
- 4-3. 如果一個整數是2的倍數，則它就是一個偶數，否則為奇數。給你一個整數，請你判斷它是奇數還是偶數。若輸入的整數為奇數，則輸出 ODD，若為偶數則輸出 EVEN。

練習題

- 4-1流程圖



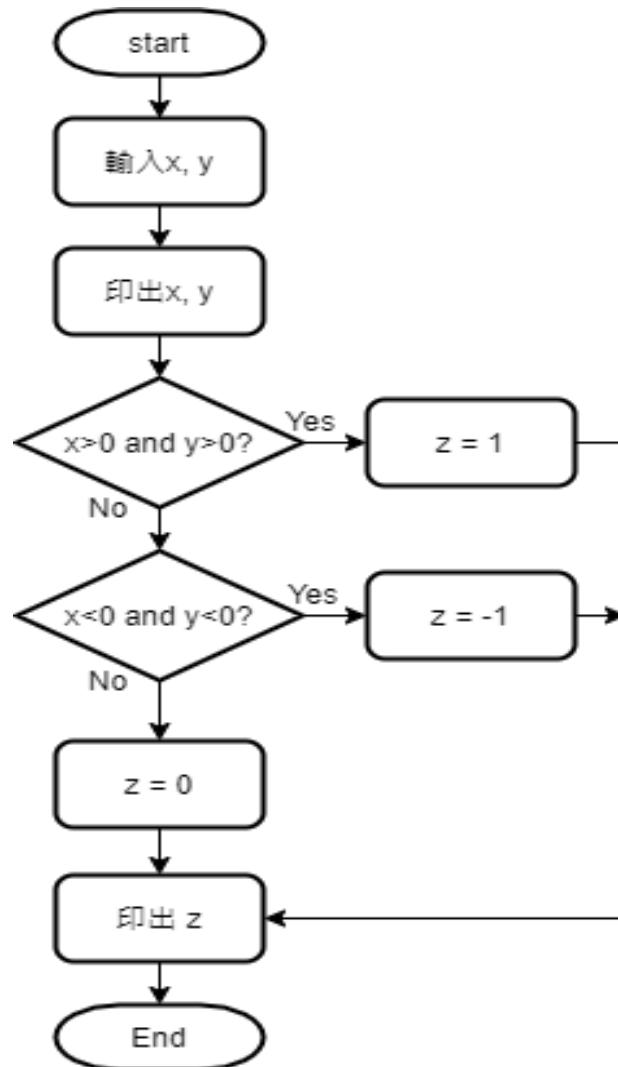
練習題

- 4-1 參考程式：

```
x, y = map(int, input("請輸入數值x和y: ")).split()  
print("x = ", x, "\ny = ", y)  
  
z = x if x >= y else y  
  
print("z = ", z)
```

練習題

- 4-2流程圖：



練習題

- 4-2參考程式：

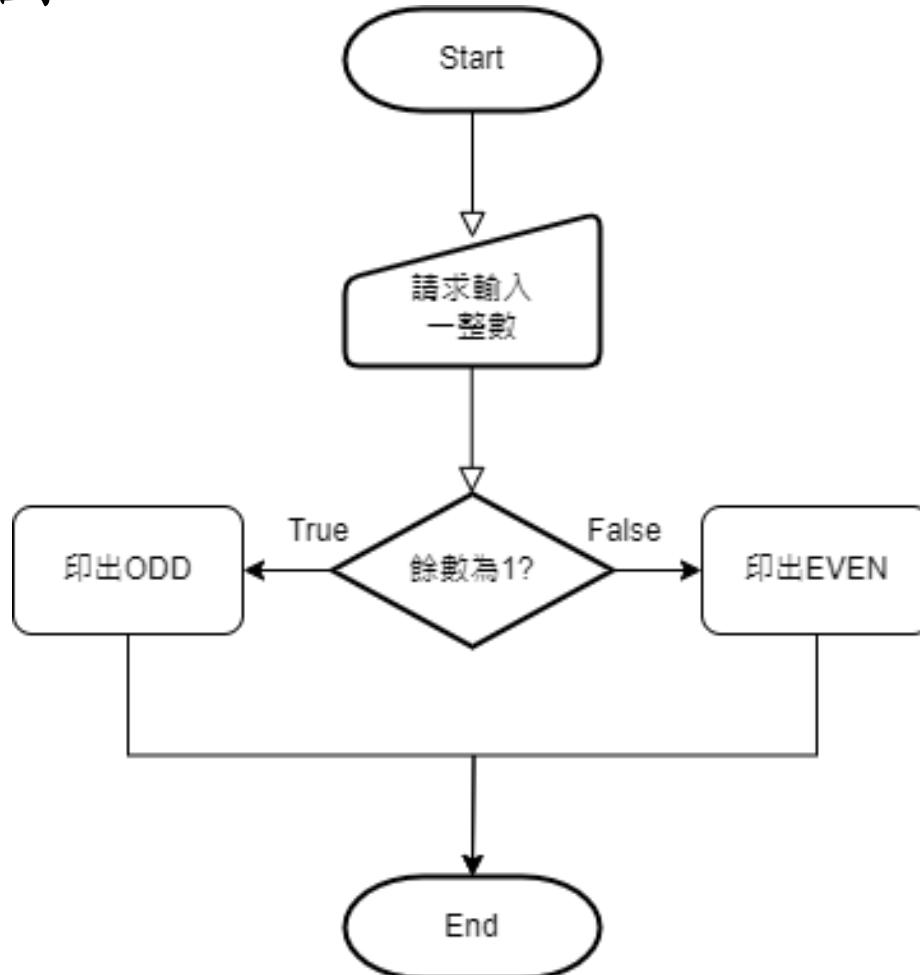
```
x, y = map(int, input("請輸入數值x和y: ").split())
print("x = ", x, "\ny = ", y)

if x > 0 and y > 0:
    z = 1
elif x < 0 and y < 0:
    z = -1
else:
    z = 0

print("z = ", z)
```

練習題

- 4-3流程圖：



練習題

- 4-3參考程式：

```
N = int(input("請輸入一個整數："))
```

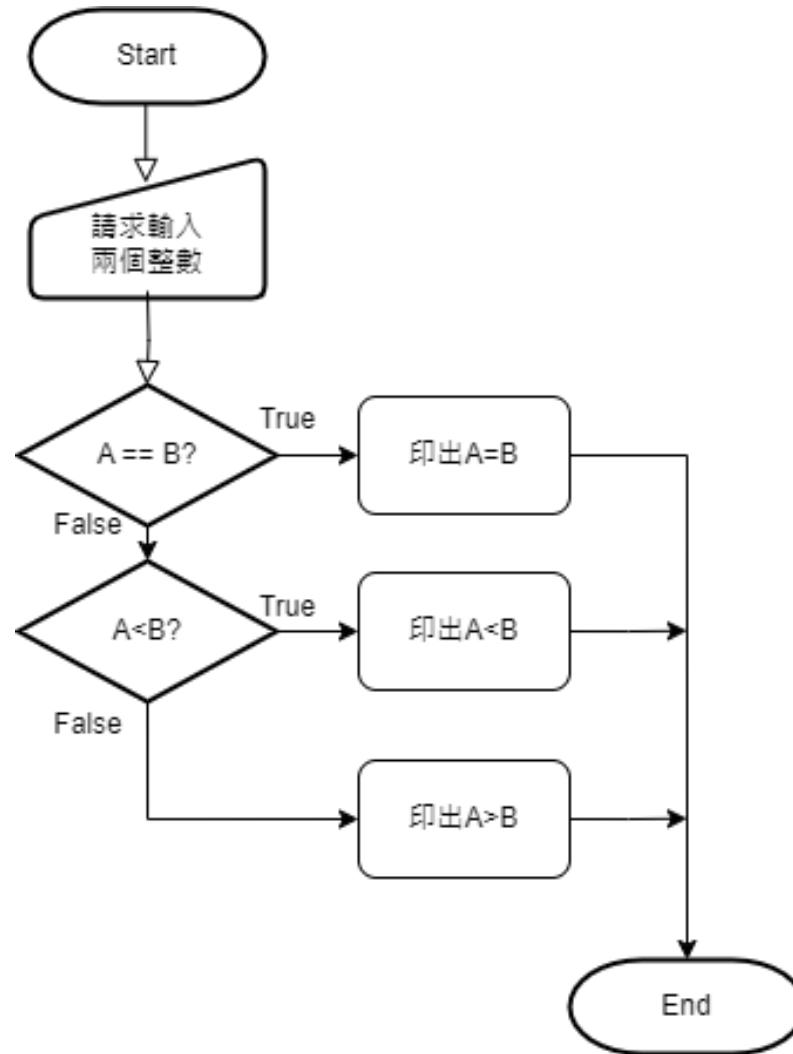
```
#整數除2的餘數只會是0或1，  
#可代表真或假回傳給if判斷用  
if N % 2:  
    print(f"{N}是奇數")  
else:  
    print(f"{N}是偶數")
```

練習題

- 4-4. 紿你兩個整數，請輸出它們之間的大小關係。如果A大於B，印出"A>B"，若A小於B，印出"A<B"，如果A等於B，印出"A=B"。
- 4-5. 寫一程式，輸入x和y，如果 $x \geq y$ ，令 $z=x^2$ ，否則令 $z=y^2$ 。

練習題

- 4-4流程圖：



練習題

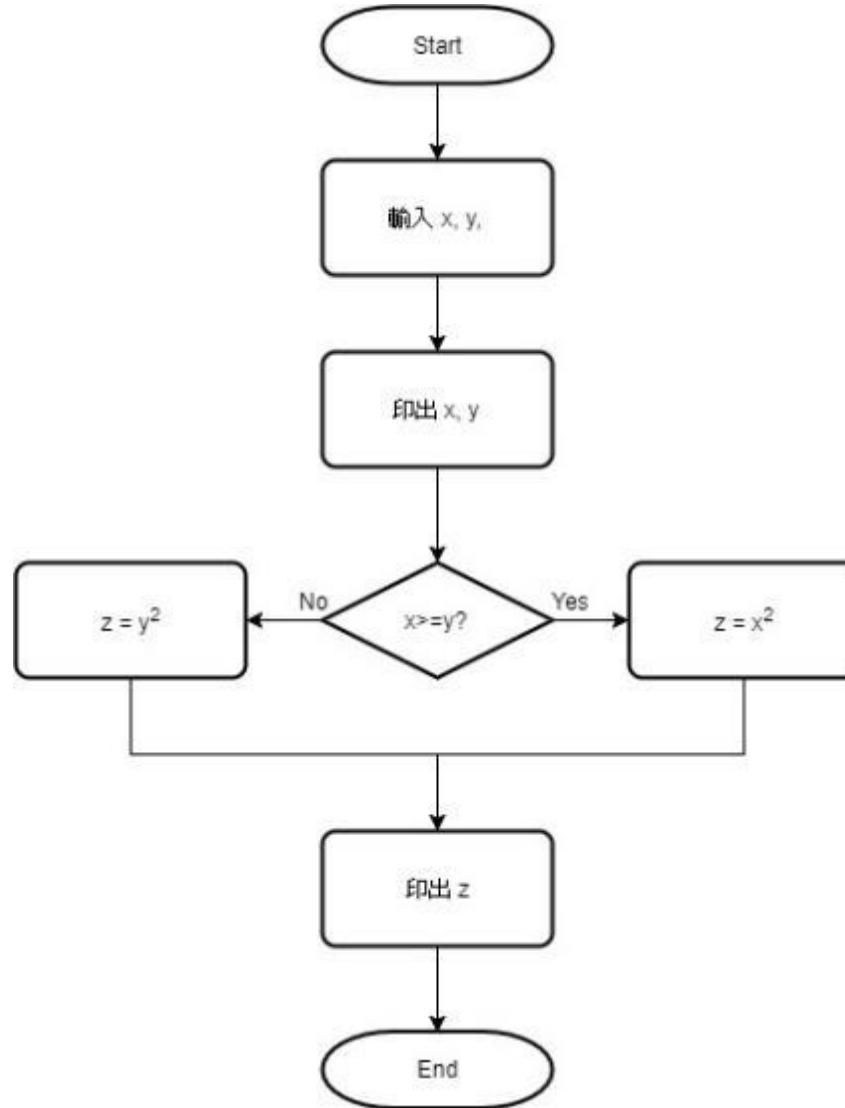
- 4-4參考程式：

```
a, b = map(int, input("請輸入兩個數值：").split())

if a == b:
    print(a, "=", b)
elif a < b:
    print(a, "<", b)
else:
    print(a, ">", b)
```

練習題

- 4-5流程圖：



練習題

- 4-5參考程式：

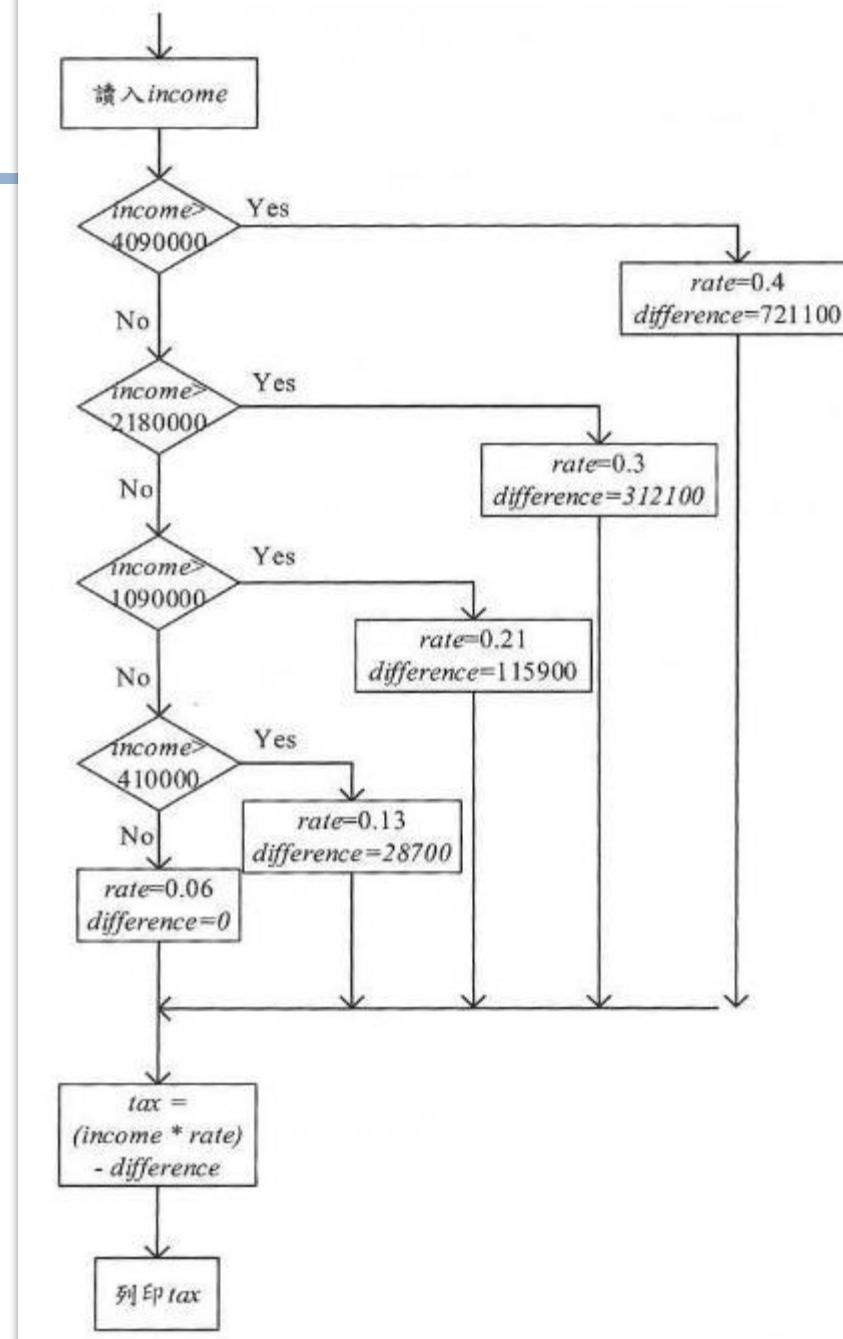
```
x, y = map(int, input("請輸入數值x和y: ").split())
print("x = ", x, "\ny = ", y)

if x >= y:
    z = x**2
else:
    z = y**2

print("z = ", z)
```

練習題

- 4-6. 依照以下的流程圖，寫一程式。



練習題

- 4-6參考程式：

```
income = int(input("Please input your income: "))
if income > 4090000:
    rate = 0.4
    difference = 721100
elif income > 2180000:
    rate = 0.3
    difference = 312100
elif income > 1090000:
    rate = 0.21
    difference = 115900
elif income > 410000:
    rate = 0.13
    difference = 28700
else:
    rate = 0.06
    difference = 0
tax = (income * rate) - difference
print("tax = ", tax)
```

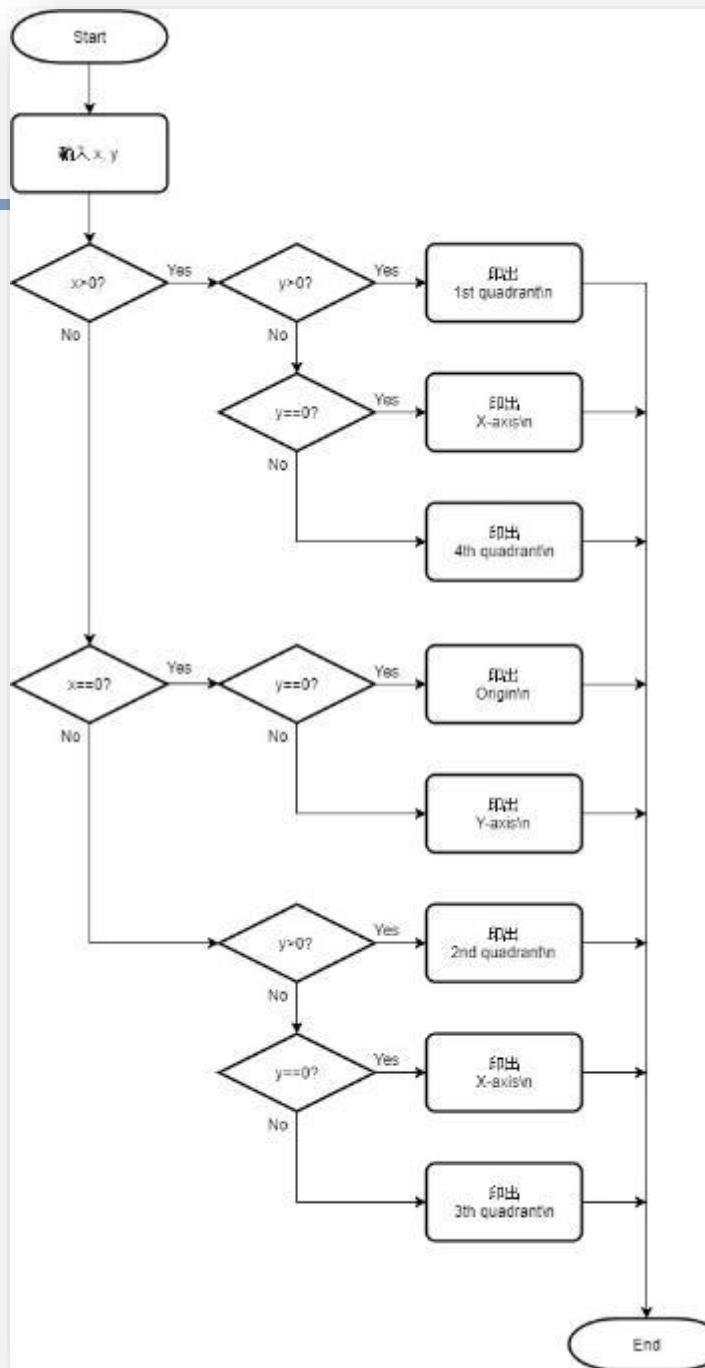
練習題

- 4-7. 將以下程式的流程圖畫出來。

```
x=int(input("Enter x: "))
y=int(input("Enter y: "))
if x > 0:
    if y > 0:
        print("1st quadrant\n")
    elif y == 0:
        print("X-axis\n")
    else:
        print("4th quadrant\n")
elif x == 0:
    if y == 0:
        print("Origin\n")
    else:
        print("Y-axis\n")
else:
    if y > 0:
        print("2nd quadrant\n")
    elif y == 0:
        print("X-axis\n")
    else:
        print("3rd quadrant\n")
```

練習題

• 4-7. 流程圖



休息一下~



3png.com

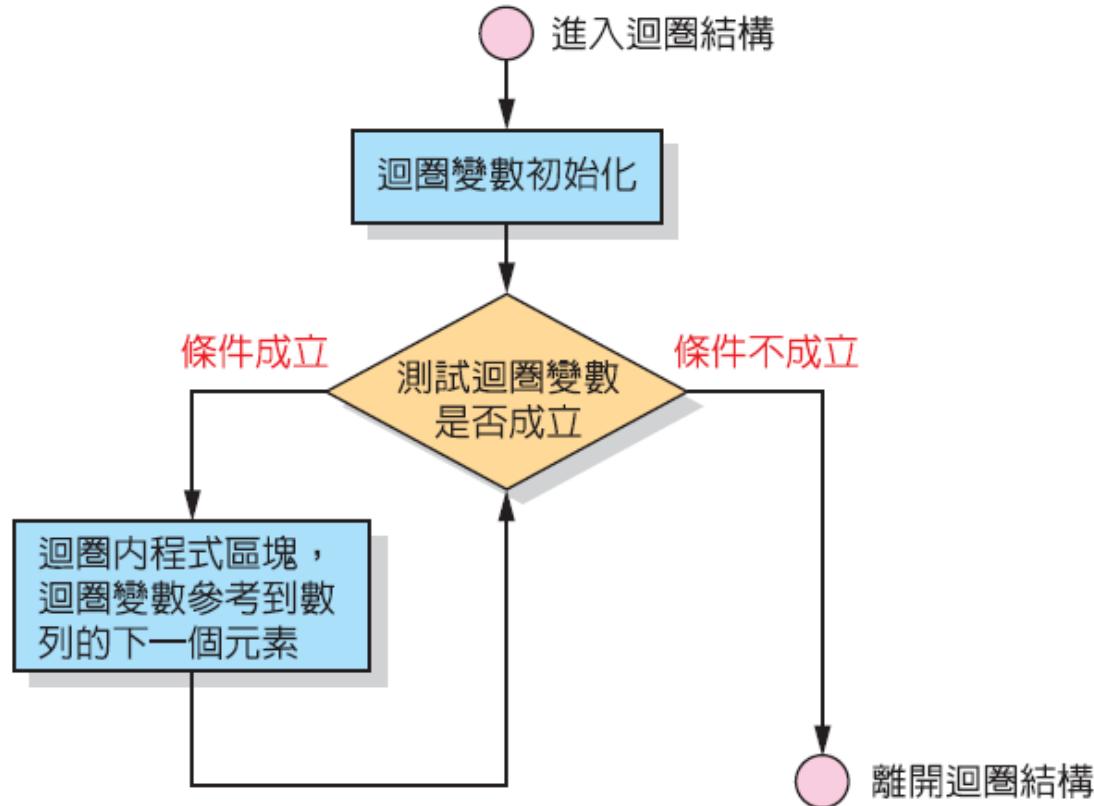
迴圈 / 重複結構

- 如果在程式中遇到要**重複同樣動作**的指令，就要使用 **for** 或 **while** 迴圈指令。



for迴圈指令

- **for**是：如果參照的**迴圈變數**的值或清單的元素在範圍內，就重複執行區塊。



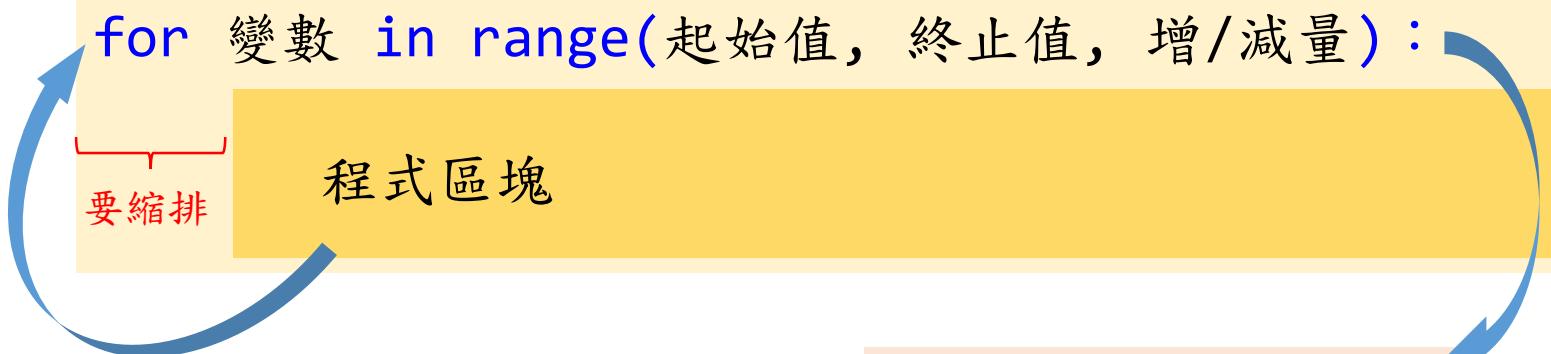
for迴圈指令

- **for**語法：

可省略，若省略
則內定為0起始

不可省略，若設
為n，則終值為
 $n-1$

每次增/減多少，
可省略，若省略
則內定為+1



變數的值加上增/減量後
在終止值之內則重複程
式區塊。

變數的值加上增/減量後等
於或超出終止值，離開for
迴圈，執行之後的指令。

註：`range()`是一個函式，可單獨使用。

for迴圈指令

- for語法：

清單可以是字串、list、tuple、dict等

for 變數 in 清單：

要縮排

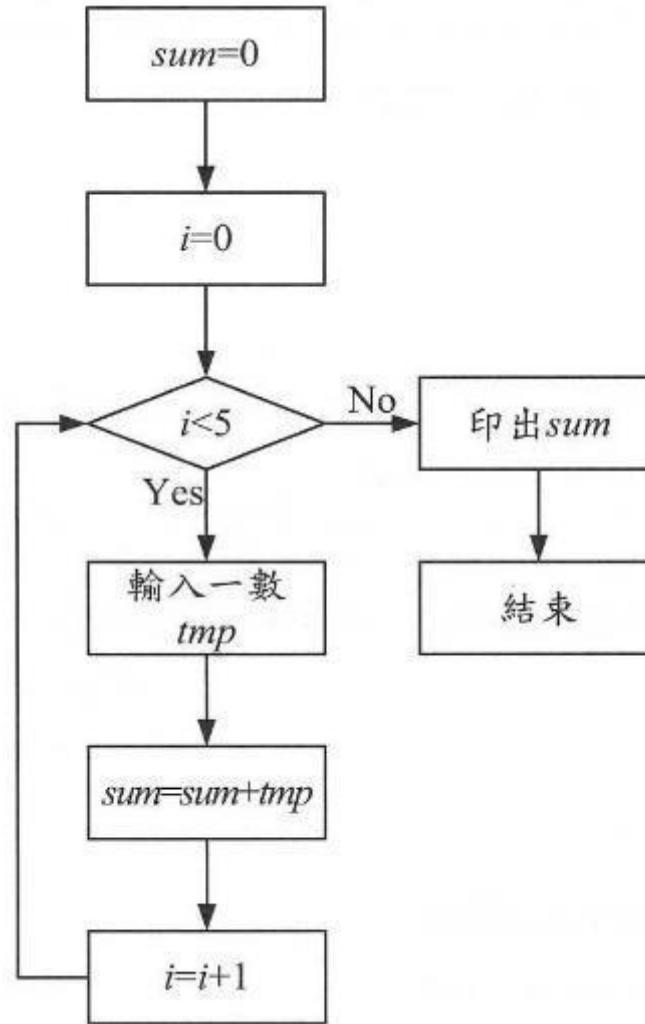
程式區塊

依序取出清單的值存入變數，直到取完為止。

清單內所有元素依序取完後離開迴圈。

for迴圈指令

- 讀入五個整數，求這五個整數的和。
- 流程圖：



for迴圈指令

- 參考程式：
 - 假設依序輸入1,3,5,7,9

```
sum = 0

for i in range(5):
    sum += int(input("請輸入數值: "))

print("sum = ", sum)
```

執行情形

i	輸入	sum
0	1	1
1	3	4
2	5	9
3	7	16
4	9	25

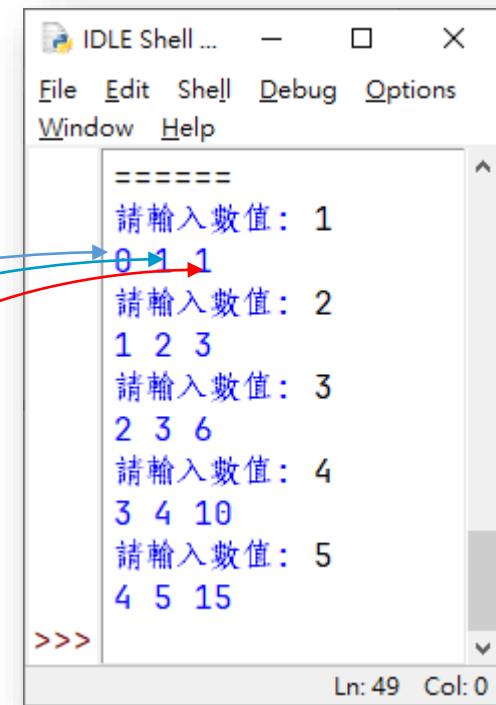
- i會從0開始，每次加1，直到i=4為止（共5次），i在這個範圍內for迴圈內的敘述就要重複執行。

for迴圈指令

- 可以將迴圈執行過程印出來看看各變數內容的變化。
 - 假設依序輸入1,2,3,4,5

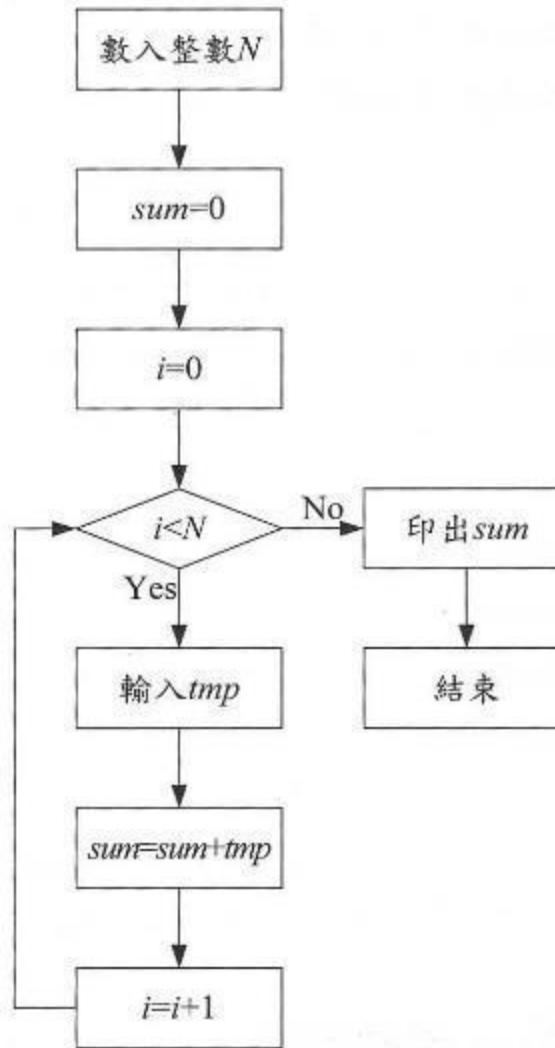
```
sum = 0

for i in range(5):
    n = int(input("請輸入數值: "))
    sum += n
    print(i, n, sum)
```



for迴圈指令

- 求N個整數的和，
N由使用者輸入。
- 流程圖：



for迴圈指令

- 參考程式：

```
N = int(input("請輸入數值個數N: "))
sum = 0

for i in range(N):
    sum += int(input("請輸入數值: "))

print("sum = ", sum)
```

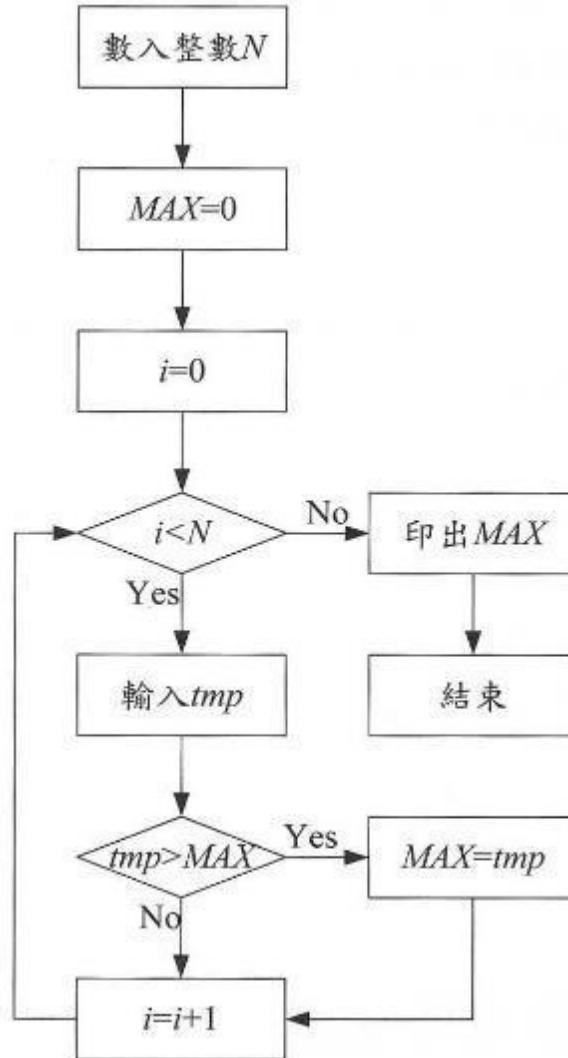
- 迴圈重複的次數由輸入的N決定，從0到N-1。

for迴圈指令

- 輸入N個正整數，求其中的最大值。
 - 先假設最大值MAX等於0，然後逐一檢查讀入的每一個數字，如果比MAX還大，就令MAX等於這個數字，如果沒有比MAX大，就不做任何事情，最後MAX一定會是這一串數字中的最大值。
- 這個例子在之後介紹List(清單)時會更簡單。

for迴圈指令

- 流程圖：



for迴圈指令

- 參考程式：

```
N = int(input("請輸入數值個數N: "))
MAX = 0
```

```
for i in range(N):
    tmp = int(input("請輸入數值: "))
    if tmp > MAX:
        MAX = tmp
```

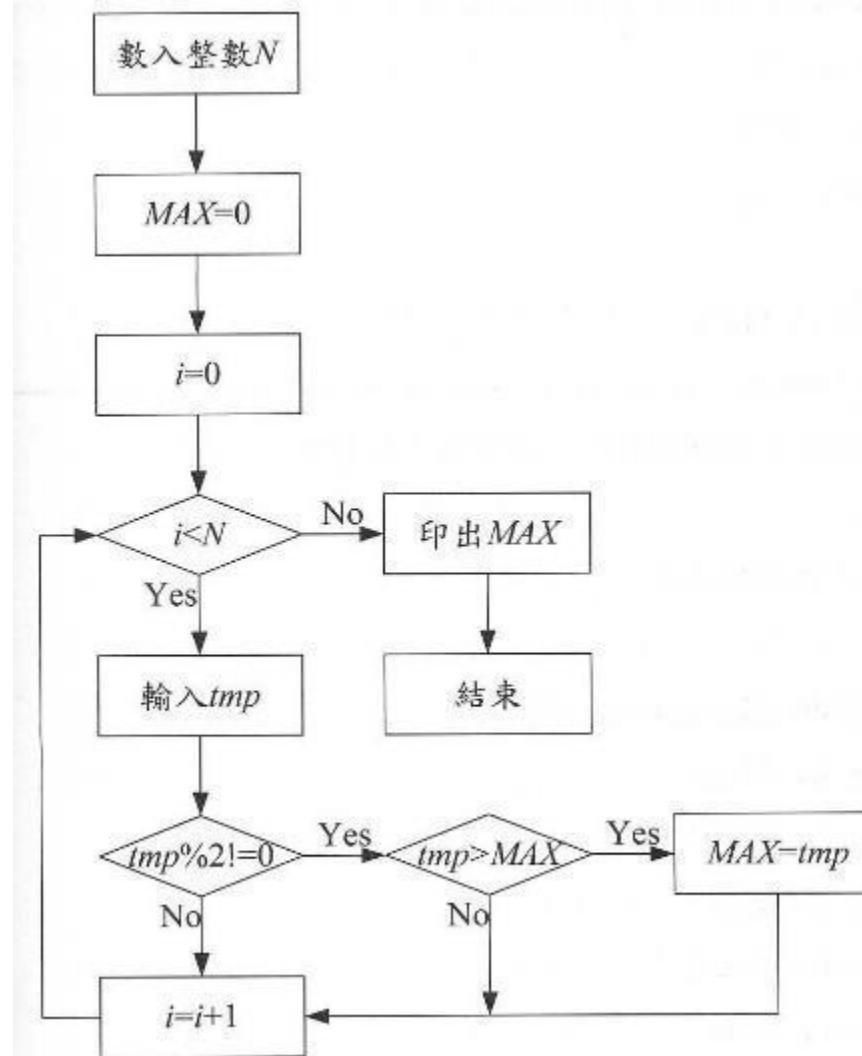
```
print("MAX = ", MAX)
```

for迴圈指令

- 求最大奇數值。
 - 先假設最大值MAX等於0，然後逐一檢查讀入的每一個數字，如果是奇數而且比MAX還大，就令MAX等於這個數字，如果不是，就不做任何事情，最後MAX一定會是這一串數字中的奇數最大值。
 - 可以用 $\%$ (取餘數)運算，將n除以2之後若餘0，為偶數，若餘1則為奇數。

for迴圈指令

- 流程圖：



for迴圈指令

- 參考程式：

```
N = int(input("請輸入數值個數N: "))
MAX = 0
```

```
for i in range(N):
    tmp = int(input("請輸入數值: "))
    if tmp % 2 != 0:
        if tmp > MAX:
            MAX = tmp
```

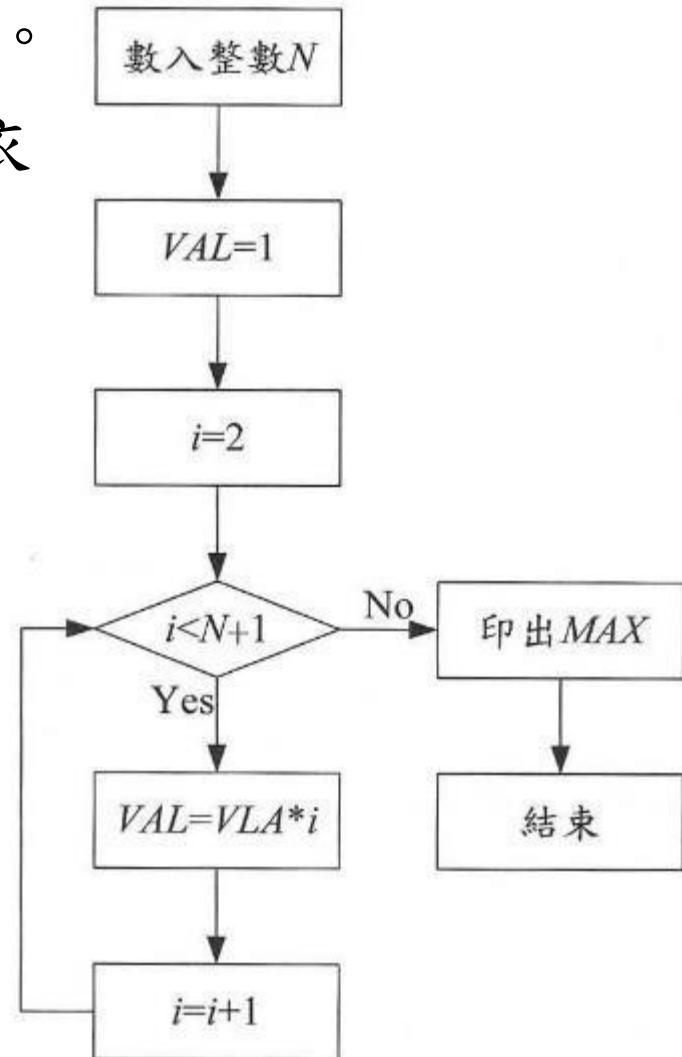
```
print("MAX = ", MAX)
```

for迴圈指令

- 求N階乘，即求 $1 \times 2 \times 3 \times \dots \times N$ 。
- 請求輸入N，讓for迴圈自2依序乘到N，所以要寫成：

```
for i in range(2, N+1)
```

- 流程圖：



for迴圈指令

- 參考程式：

```
N = int(input("請輸入數值N: "))
VAL = 1
```

```
for i in range(2, N+1):
    VAL *= i
```

```
print("VAL = ", VAL)
```

- $VAL = VAL * i$ 可寫成 $VAL *= i$ 。
- 注意迴圈若要執行到N，要寫成 $N+1$ ，否則只執行到 $N-1$ 。

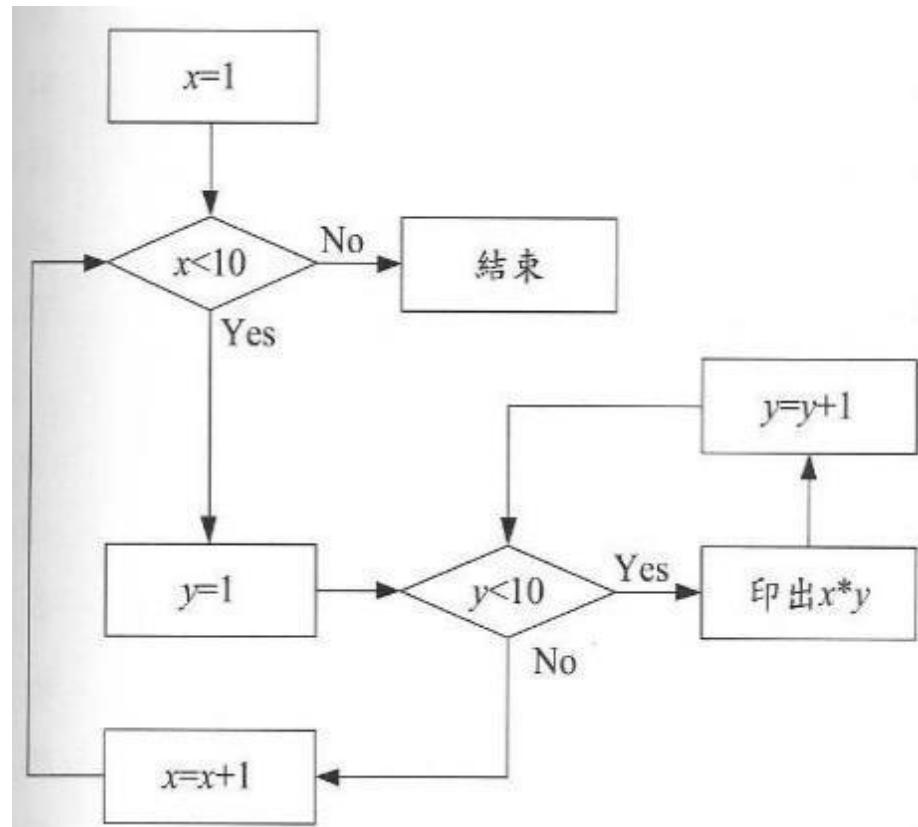
巢狀迴圈(Nested loop)

- 巢狀迴圈就是迴圈裏面還有迴圈，一層套一層。
- 巢狀迴圈的層次沒有限制，但層次越多，效率越差，通常三層、四層就很多了，如果還需要更多，那可能要重新檢討一下是不是有其它方法(演算法)可以解決。
- 每一層的變數名稱切記不可相同，否則會相互干擾，無法如你想的正常執行，除非你有什麼特殊目的啦。



巢狀迴圈(Nested loop)

- 經典問題：印出一個九九乘法表。
- 利用兩層迴圈(巢狀迴圈)。
- 要記得迴圈的層數沒有限制，但越多效率越差。
- 流程圖如右：



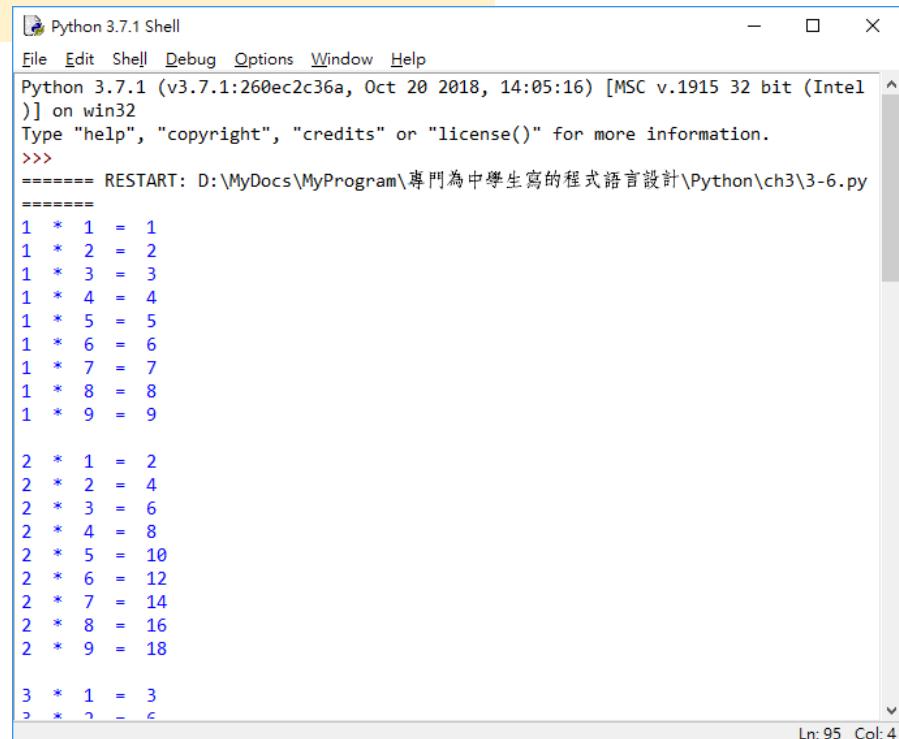
巢狀迴圈(Nested loop)

- 參考程式：

```
for x in range(1, 10):
    for y in range(1, 10):
        print(f"{x} * {y} = {x*y}")
    print()
```

- 內外層變數不可用相同
名稱，注意縮排。

- 執行結果：



The screenshot shows the Python 3.7.1 Shell window. The title bar says "Python 3.7.1 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main area displays the output of a script named "ch3\3-6.py". The output shows the multiplication table from 1 to 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
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
```

At the bottom right of the shell window, it says "Ln: 95 Col: 4".

巢狀迴圈(Nested loop)

- 前一個寫法太醜了，請將程式改成如下輸出：
 - (注意個位數對齊)

The screenshot shows the IDLE Shell interface with the title bar "IDLE Shell 3.9.13". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays a 3x3 grid of multiplication tables for numbers 1 through 9. Each row contains three columns of multiplication equations, with the results aligned by their last digit.

Row	Column 1	Column 2	Column 3
1	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
2	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
3	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

Ln: 67 Col: 4

巢狀迴圈(Nested loop)

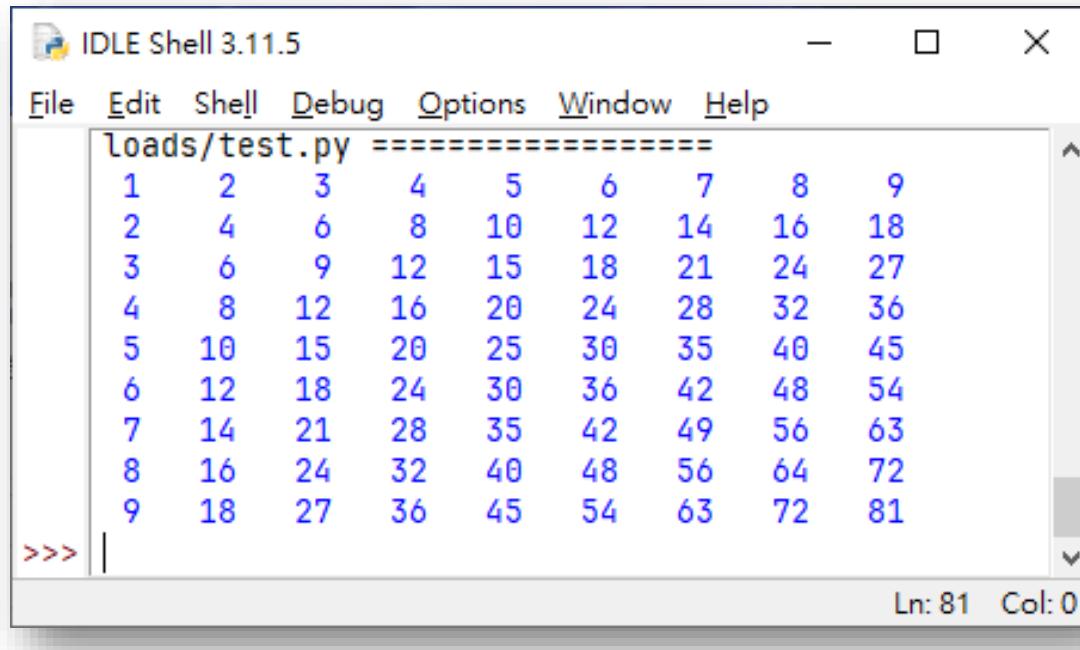
- 參考程式：

```
for x in range(1, 10, 3):
    for y in range(1, 10):
        print("%d * %d = %2d\t" % (x, y, x*y), end=' ')
        print("%d * %d = %2d\t" % (x+1, y, (x+1)*y), end=' ')
        print("%d * %d = %2d" % (x+2, y, (x+2)*y))
print()
```

- "\t"稱跳脫字元，是Tab的意思。
- 常用的還有"\n"，是換行的意思。
- print()函數內定結尾會自動加上換行，加上end=''參數後就會依我們加入的文字當結尾。

巢狀迴圈(Nested loop)

- 再簡化一點吧，請將程式改成如下輸出：



The screenshot shows the Python IDLE Shell interface with the title "IDLE Shell 3.11.5". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The code editor window displays a script named "loads/test.py" containing the following code:

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

The command line at the bottom shows ">>> |". The status bar at the bottom right indicates "Ln: 81 Col: 0".

巢狀迴圈(Nested loop)

- 參考程式：

```
for i in range(1, 10):  
    for j in range(1, 10):  
        print("%2d    " % (i * j), end=' ')  
    print()
```

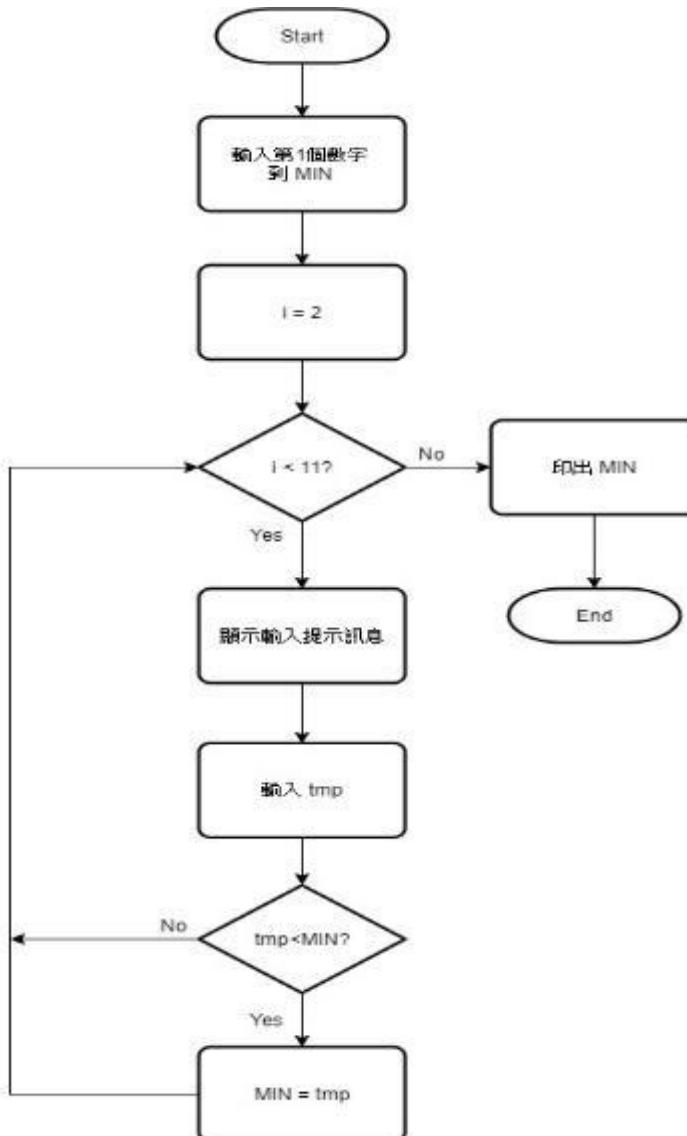
- 是不是很簡單？
- 印度小朋友要背 19×19 的乘法表，你可以幫他們產生一個嗎？

練習題

- 5-1. 寫一程式，輸入10個整數，求其最小值。
- 5-2. 寫一程式，輸入N個整數，求其最小值。
- 5-3. 寫一程式，輸入10個整數，列出其中所有大於12的數字。
- 5-4. 寫一程式，輸入10個整數，列出其中所有大於12的數字的總和。

練習題

- 5-1流程圖：



練習題

- 5-1 參考程式：

```
MIN = int(input("請輸入第 1 個數值："))

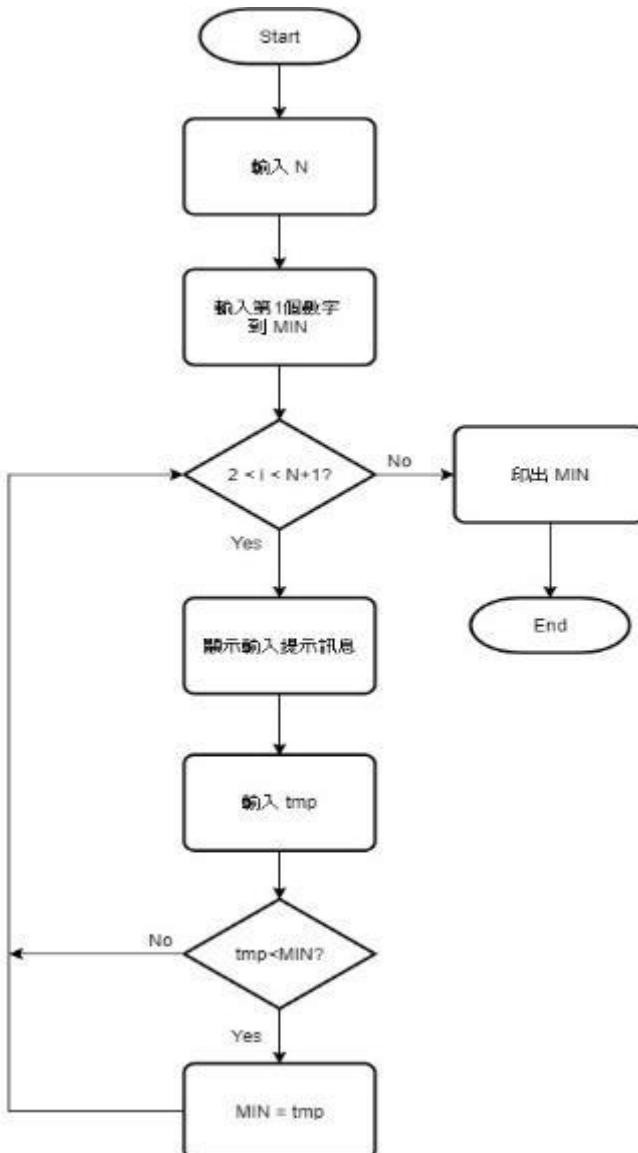
for i in range(2, 11):
    print("請輸入第", i, "個數值：", end="")
    tmp = int(input())
    if tmp < MIN:
        MIN = tmp

print("MIN = ", MIN)
```

- 先將第一個輸入的值當成最小的，再依序跟後續輸入的做比較。

練習題

- 5-2流程圖：



練習題

- 5-2參考程式：

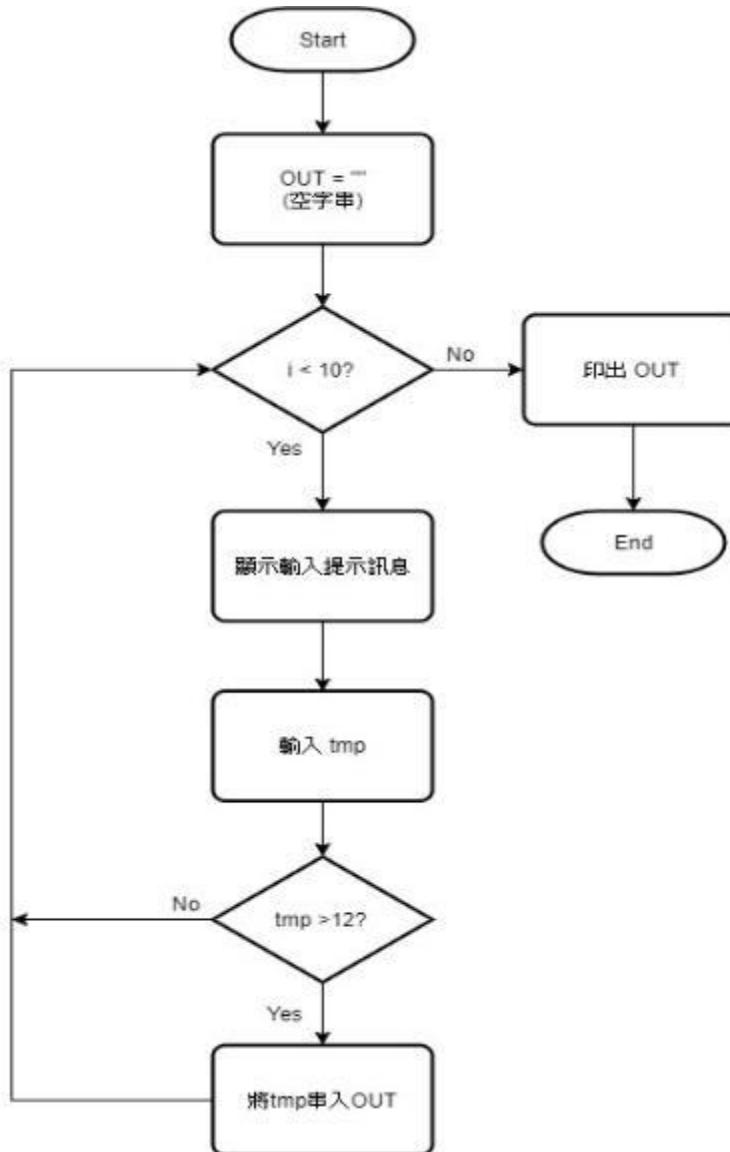
```
N = int(input("請問有幾個數值: "))
MIN = int(input("請輸入數值 1 : "))

for i in range(2, N+1):
    print("請輸入數值", i, ":", end="")
    tmp = int(input())
    if tmp < MIN:
        MIN = tmp

print("MIN = ", MIN)
```

練習題

- 5-3流程圖：



練習題

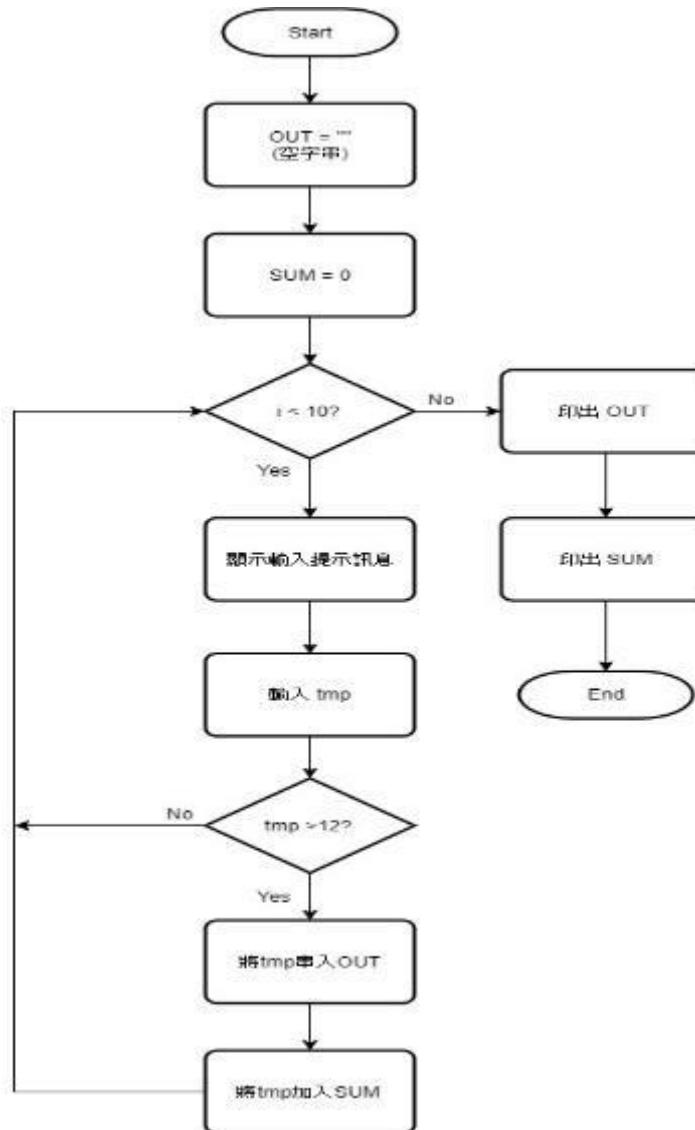
- 5-3參考程式：

```
OUT = ""
for i in range(10):
    print("請輸入數值", i+1, ":", end="")
    tmp = int(input())
    if tmp > 12:
        OUT = OUT + str(tmp) + " "
print("比12大的數有：", OUT)
```

- 將符合大於12的數字轉成文字串接存入OUT，最後將OUT印出來。

練習題

- 5-4流程圖：



練習題

- 5-4參考程式：

```
OUT = ""
SUM = 0

for i in range(10):
    print("請輸入數值", i+1, ":", end="")
    tmp = int(input())
    if tmp > 12:
        OUT = OUT + str(tmp) + " "
        SUM += tmp

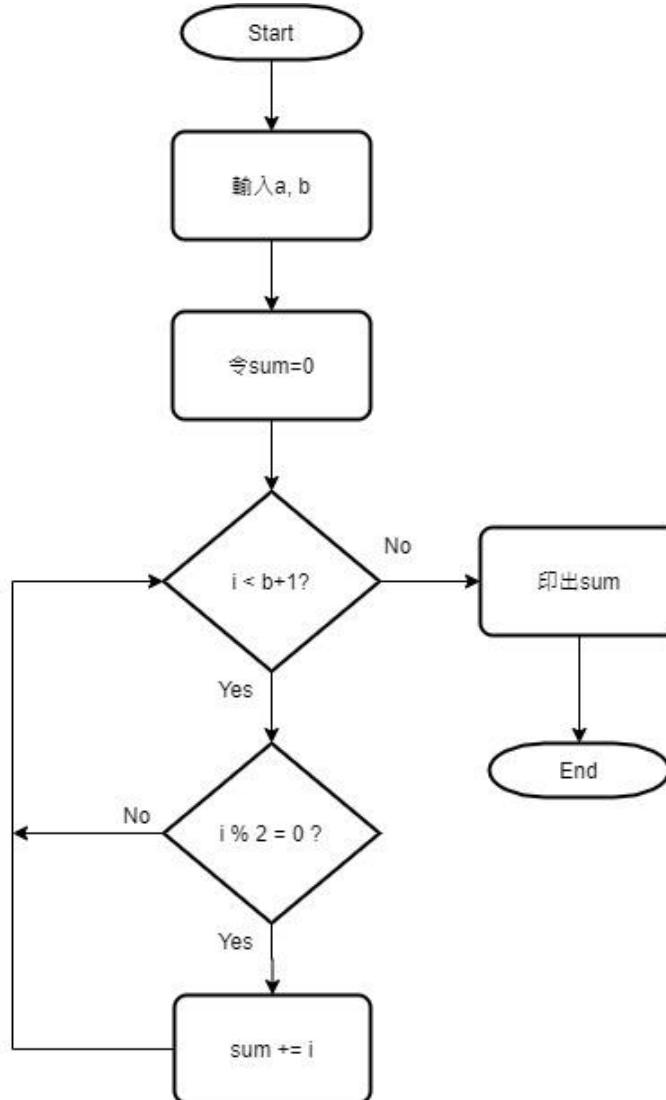
print("比12大的數：", OUT)
print("總和 = ", SUM)
```

練習題

- 5-5. 寫一程式，輸入一個數字範圍，求這段數字範圍間所有偶數的和。例如輸入5, 20，則為 $6+8+10+12+14+16+18+20$ ，答案是104。
- 5-6. 寫一程式，輸入N個數字，求其所有正數之平方的加總。例如輸入1, -2, 3, -4, 5五個數字，得到：
$$1^2 + \cancel{-2} + 3^2 + \cancel{-4} + 5^2 = 1 + 9 + 25 = 35$$
- 5-7. 寫一程式，輸入N個數字，其中有些是負數，將這些負數轉換成正數，例如-7會被轉換成7。

練習題

- 5-5. 流程圖：



練習題

- 5-5. 參考程式：

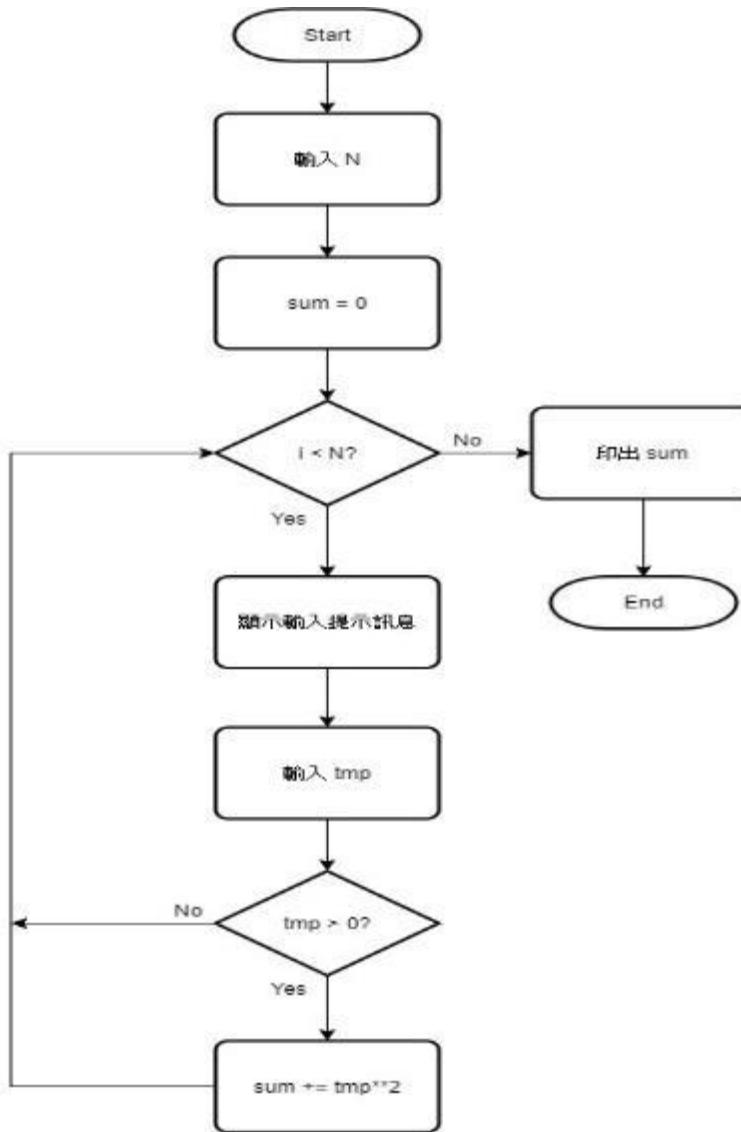
```
a, b = map(int, input("請輸入數字範圍:").split())
sum = 0

for i in range(a, b+1):      #迴圈要包含a和b
    if i % 2 == 0:
        sum += i

print("總和=", sum)
```

練習題

- 5-6. 流程圖：



練習題

- 5-6. 參考程式：

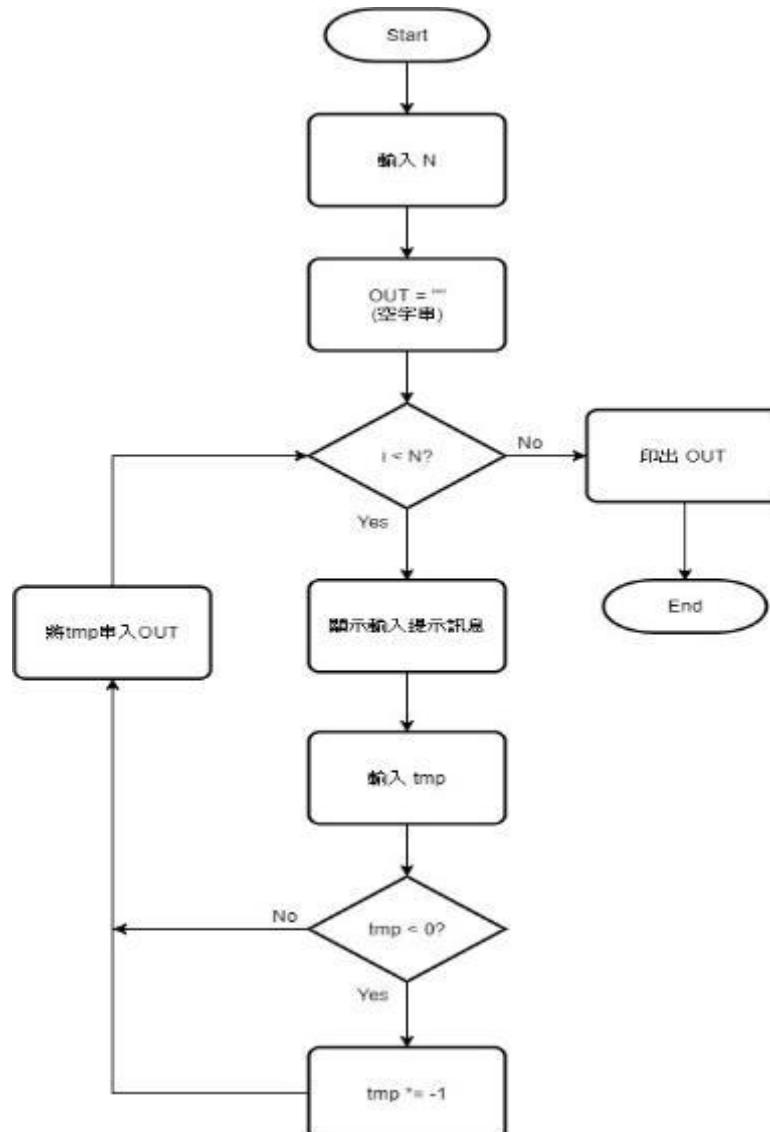
```
N = int(input("請輸入數值個數N: "))
sum = 0

for i in range(N):
    print("請輸入數值", i+1, ":", end="")
    tmp = int(input())
    if tmp > 0:
        sum += tmp**2

print("Sum = ", sum)
```

練習題

• 5-7. 流程圖



練習題

- 5-7. 參考程式：

```
N = int(input("請輸入數值個數N: "))
OUT = ""

for i in range(N):
    print("請輸入第", i, "個數值: ", end="")
    x = int(input())
    tmp = int(input())
    if tmp < 0:
        tmp *= -1
    OUT = OUT + str(tmp) + " "

print("結果 =", OUT)
```

練習題

- 5-8. 由於地球公轉一周大約是365.242199日，所以如果以一年365日來記算，每四年就會多出0.968796天，於是有人提出每四年要多出一天，這就是閏年。但是因為每四年多出來的並非完整的一天，所以累積到100年的時候，就不須要再多這一天了。而同樣的道理，到了400年，又會多出一天來。目前閏年的規則如下(西元紀年)：
 - 如果這一年不是4的倍數，則它是平年。
 - 如果這一年是4的倍數，但不是100的倍數，則它是閏年。
 - 如果這一年是100的倍數，但不是400的倍數，則它是平年。
 - 如果這一年是400的倍數，則它是閏年。
- 現在給你一個年份，請你判斷它是不是閏年。
- 5-9. 印出某段時間內的閏年，例如1911 ~ 2025年之間所有的閏年。

練習題

- 5-8. 參考程式：

```
year = int(input("請輸入年份:"))

if (year % 4) == 0 \
    and (year % 100) !=0 \
    or (year % 400) == 0:

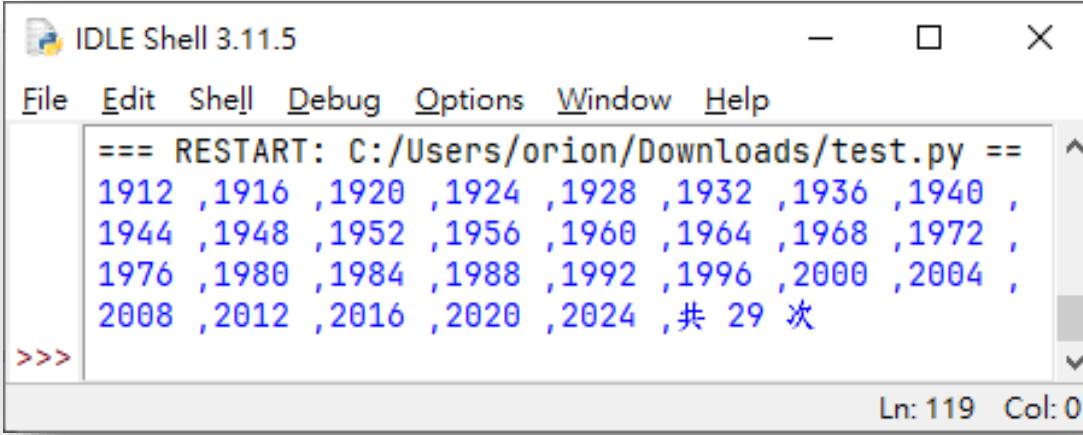
    print("YES")
else:
    print("NO")
```

當指令太長需要折斷換行時，要在行尾加上一個"\\"，這樣編譯器知道你這行還指令沒結束。

練習題

- 5-9. 參考程式：

```
x = 0
for y in range(1911, 2026):
    if y % 400 == 0 or (y % 4 == 0 and y % 100 != 0):
        x += 1
        print(y, ',', end = '')
print('共', x, '次')
```



The screenshot shows the IDLE Shell window with the title "IDLE Shell 3.11.5". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following text:

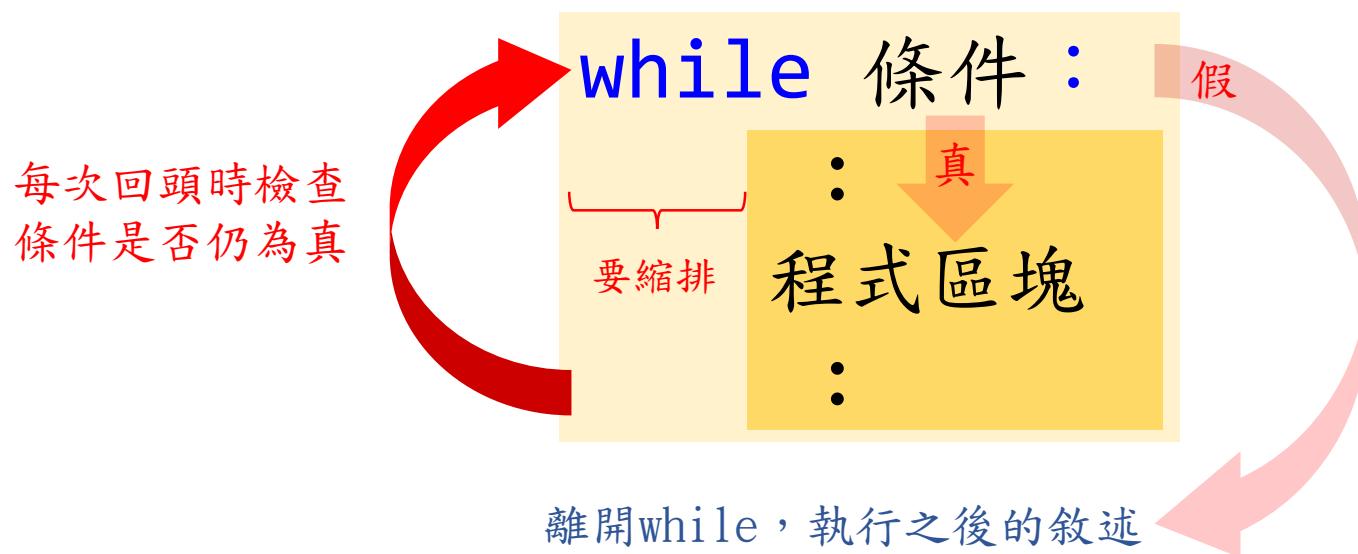
```
File Edit Shell Debug Options Window Help
==== RESTART: C:/Users/orion/Downloads/test.py ====
1912 ,1916 ,1920 ,1924 ,1928 ,1932 ,1936 ,1940 ,
1944 ,1948 ,1952 ,1956 ,1960 ,1964 ,1968 ,1972 ,
1976 ,1980 ,1984 ,1988 ,1992 ,1996 ,2000 ,2004 ,
2008 ,2012 ,2016 ,2020 ,2024 ,共 29 次
>>> Ln: 119 Col: 0
```

休息一下~



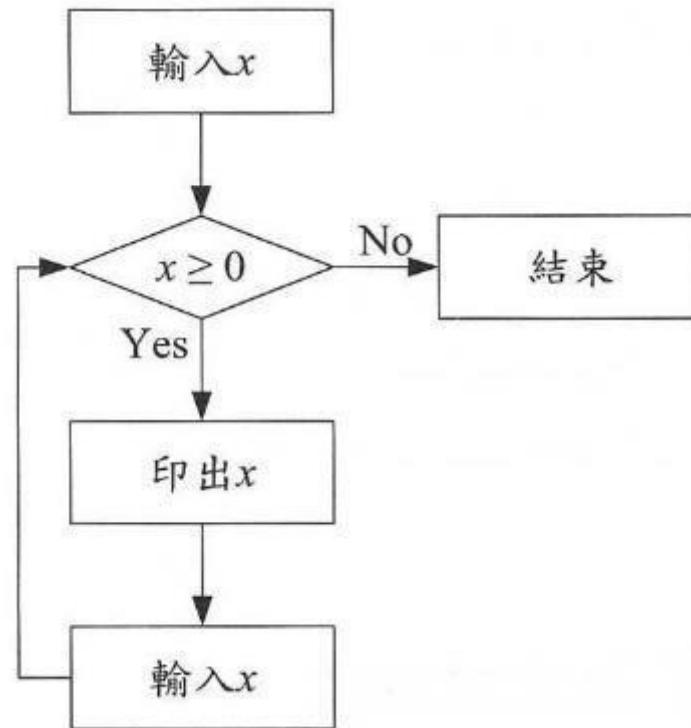
while迴圈指令

- 通常for圈都用在確定要重複執行多少次的情況，若重複的次數不確定，我們可以用while迴圈。
- while以"真"、"假"來做為終止條件。
- 當條件為真時執行迴圈，為假時結束迴圈。



while迴圈指令

- 讀入一連串的數字，但不知道有多少個數字。設定一個特殊的數字，一旦讀到這個數字就終止迴圈。
- 假設輸入的都是正數，一旦讀到負數就終止。
- 流程圖：



while迴圈指令

- 參考程式：

```
x = int(input("請輸入數值: "))
while (x >= 0):  
    print("x = ", x)
    x = int(input("請輸入數值: "))
```

條件

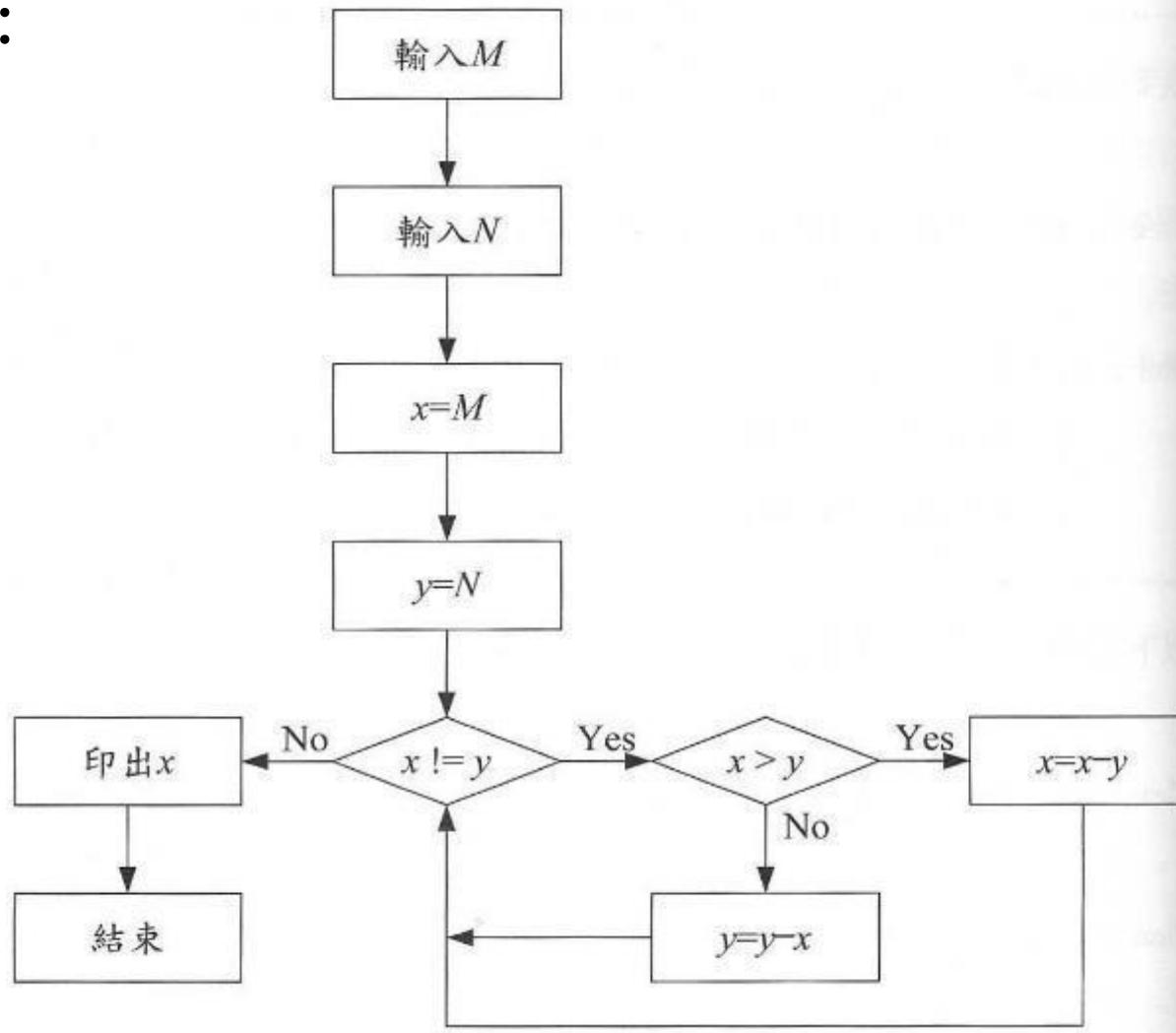
- 當 $x \geq 0$ 成立時，就執行迴圈內敘述，不成立時就終止迴圈。
- while(條件)**或**while 條件**都可以，即括符可加可不加。

while迴圈指令

- 求最大公約數(Greatest Common Divisor，簡寫G.C.D，或稱最大公因數)。
- 指能夠整除多個整數的最大正整數。而多個整數不能都為零。例如8和12的最大公因數為4。
- 用輾轉相除法：兩數相除，取餘數重複進行相除，直到餘數為0時，前一個除數即為最大公約數。
- (可別遇到數學就懶得理解喔，電腦程式很多時候都是在處理數學問題，或在處理數學問題的路上😊😊😊。)

while迴圈指令

- 流程圖：



while迴圈指令

- 參考程式：

```
M = int(input("請輸入數值M: "))
N = int(input("請輸入數值N: "))
x = M
y = N
```

```
while (x != y):
    if (x > y):
        x = x - y
    else:
        y = y - x
```

```
print("x = ", x)
```

while迴圈指令

- 參考程式二：

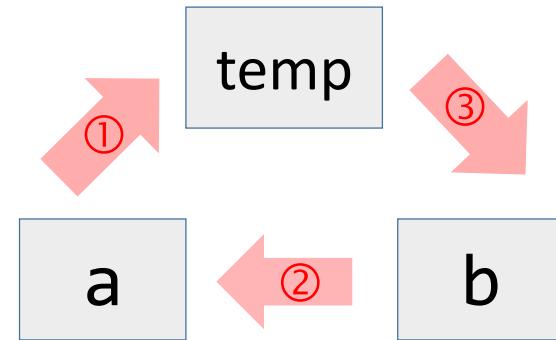
```
a = int(input("請輸入數值a: "))
b = int(input("請輸入數值b: "))
# 如果 a < b，把a和b對調
if a < b:
    a, b = b, a

r = a % b
# While迴圈會計算到r=0才會停止
while r != 0:
    a = b
    b = r
    r = a % b
print("GCD是 " + str(b))
```

while迴圈指令

- 關於兩變數交換內容的方式：
- 其他語言都須借助第三個變數當中介，例如要交換a,b的內容，則需寫成：

```
① temp = a  
② a = b  
③ b = temp
```



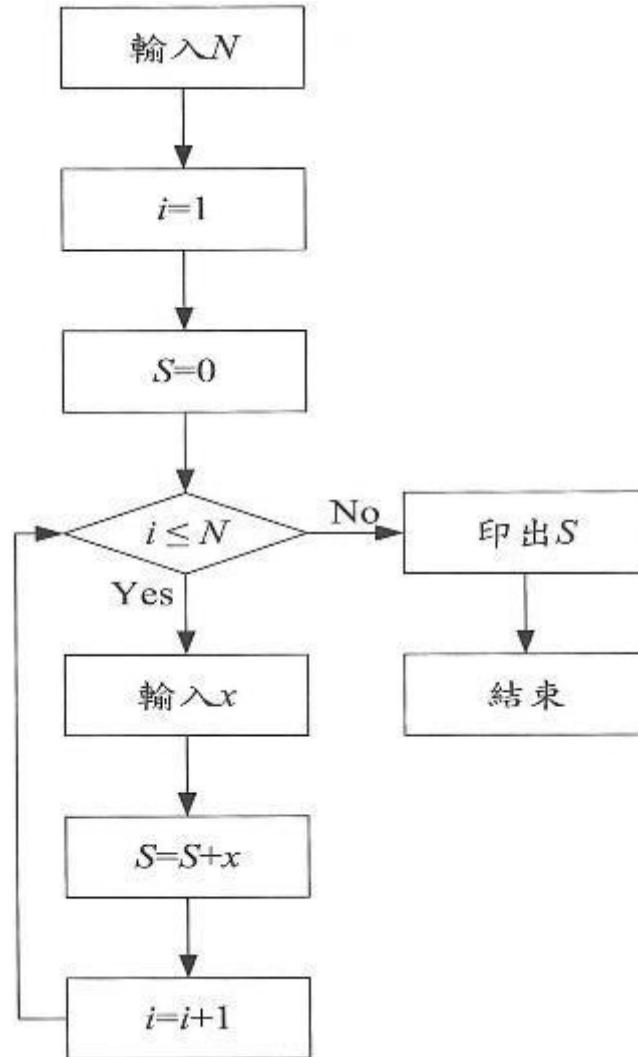
- 在python語言只需要一行即可。

```
a , b = b , a
```



while迴圈指令

- 計算N個數字的和。
- 檢查*i*有沒有超過N，超過就不做了。
- 流程圖：



while迴圈指令

- 參考程式：

```
N = int(input("請問有幾個數值要加總: "))
i = 1
S = 0
```

```
while i <= N:
    print("請輸入第", i, "個數值: ", end="")
    x = int(input())
    S = S + x
    i = i + 1
```

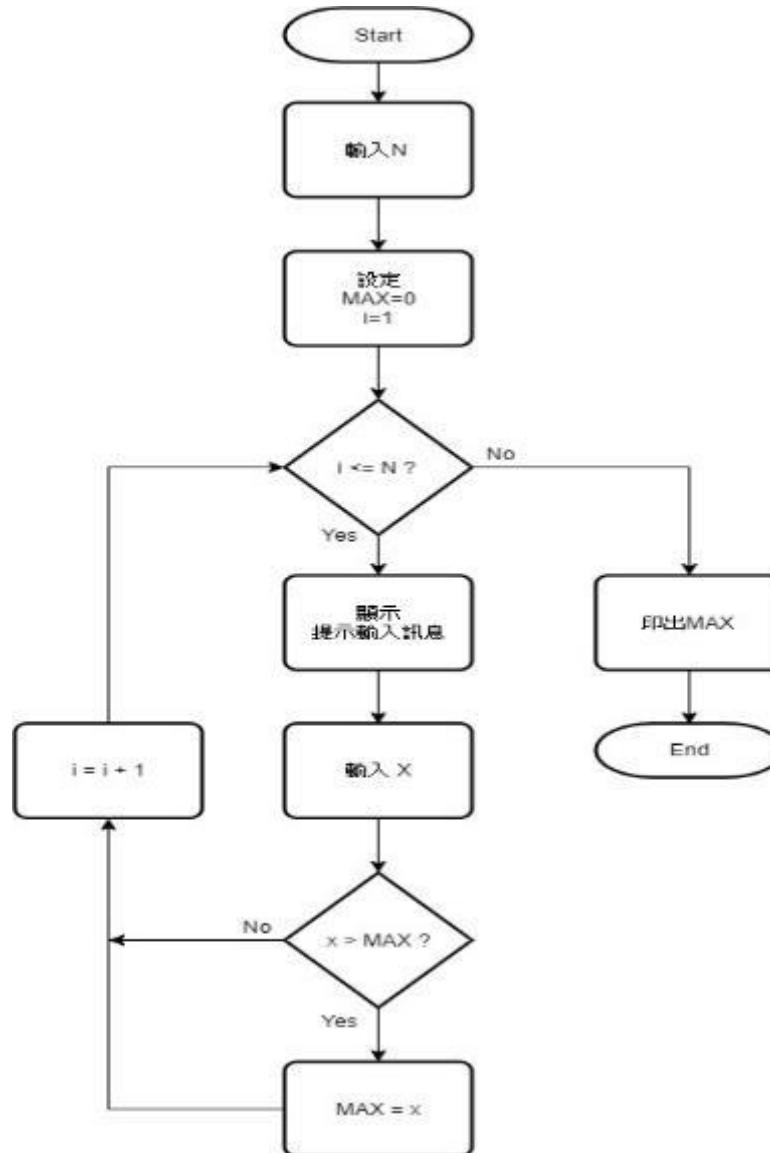
```
print("結果 = ", S)
```

練習題

- 5-8. 利用while寫一程式求N個數字的最大值。
- 5-9. 利用while寫一程式求一個等差級數數字的和，一共有N個數字，程式應該輸入最小的起始值以及數字間的差(即從起始值開始，間格差，共N個 的總和)。
- 5-10. 利用while寫一程式，讀入N個數字，然後找出所有小於13的數，再求這些數字的和。

練習題

• 5-8. 流程圖



練習題

- 5-8. 參考程式：

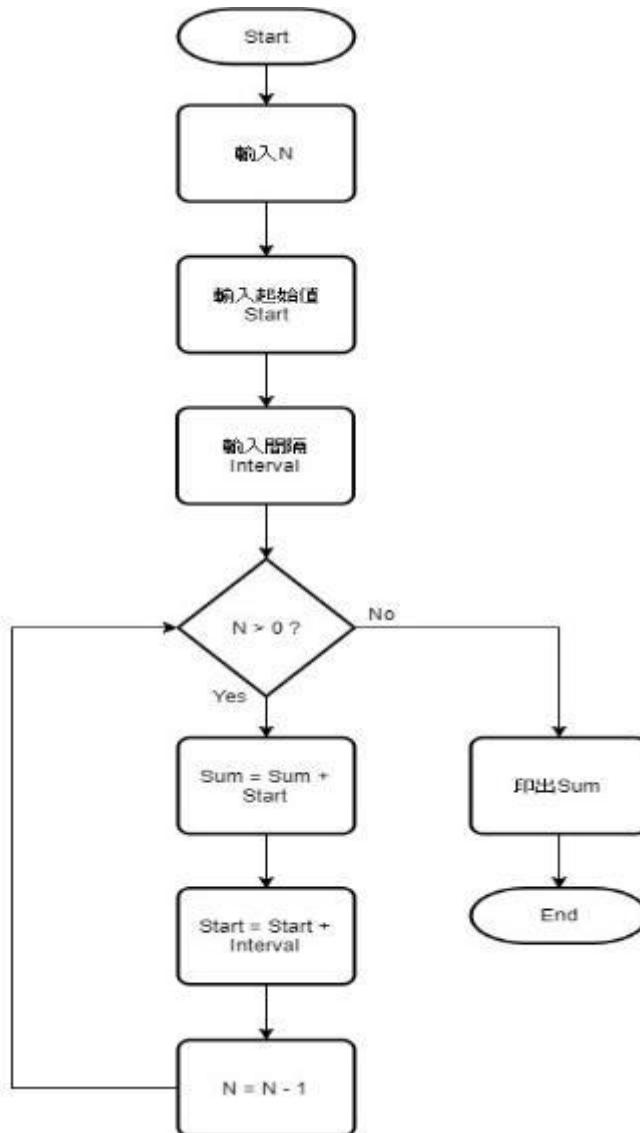
```
N = int(input("請問有多少個數字? "))
MAX = 0
i = 1

while i <= N:
    print("請輸入數值", i, ":", end="")
    x = int(input())
    if x > MAX:
        MAX = x
    i = i + 1

print("MAX = ", MAX)
```

練習題

- 5-9. 流程圖：



練習題

- 5-9. 參考程式：

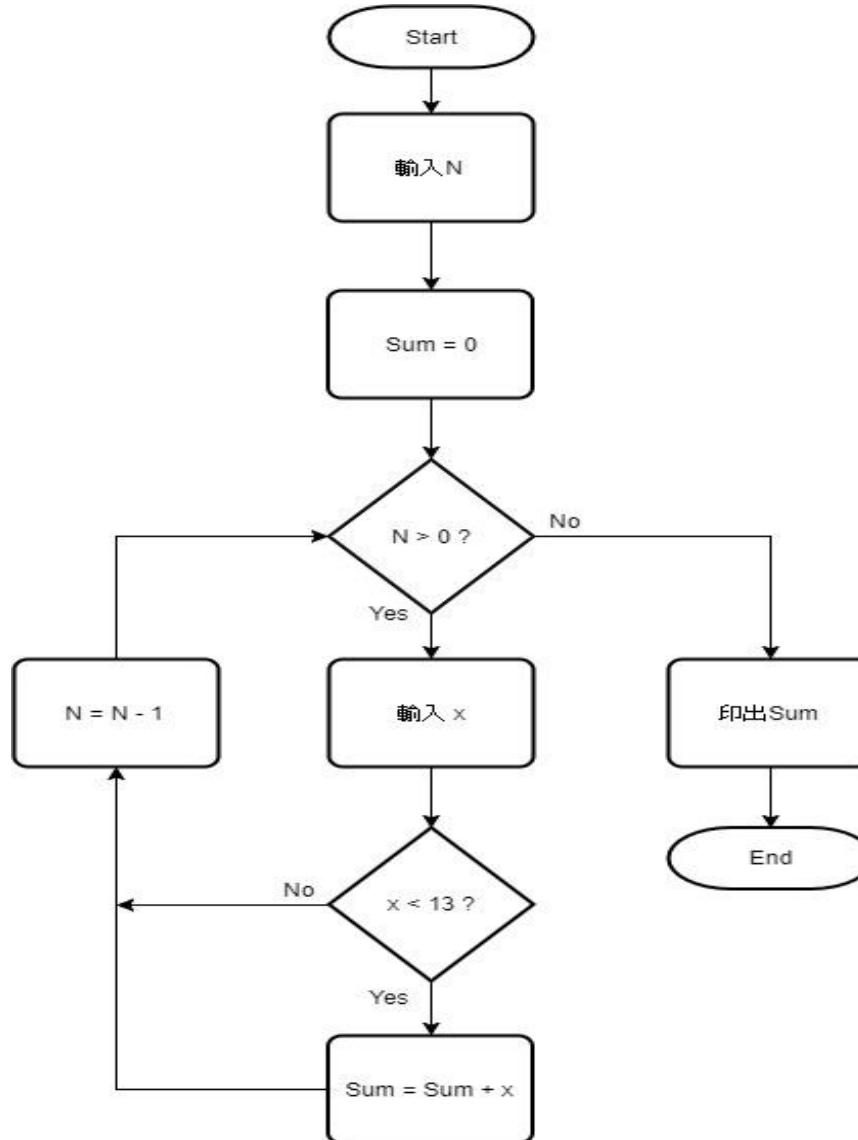
```
N = int(input("請輸入N: "))
Start = int(input("請輸入起始數值: "))
Interval = int(input("請輸入間格差: "))
Sum = 0

while N > 0:
    Sum += Start
    Start += Interval
    N -= 1

print("總和 = ", Sum)
```

練習題

- 5-10. 流程圖：



練習題

- 5-10. 參考程式：

```
N = int(input("請問有幾個數字: "))
Sum = 0

while N > 0:
    x = int(input("請輸入數值: "))
    if x < 13:
        Sum += x
    N -= 1

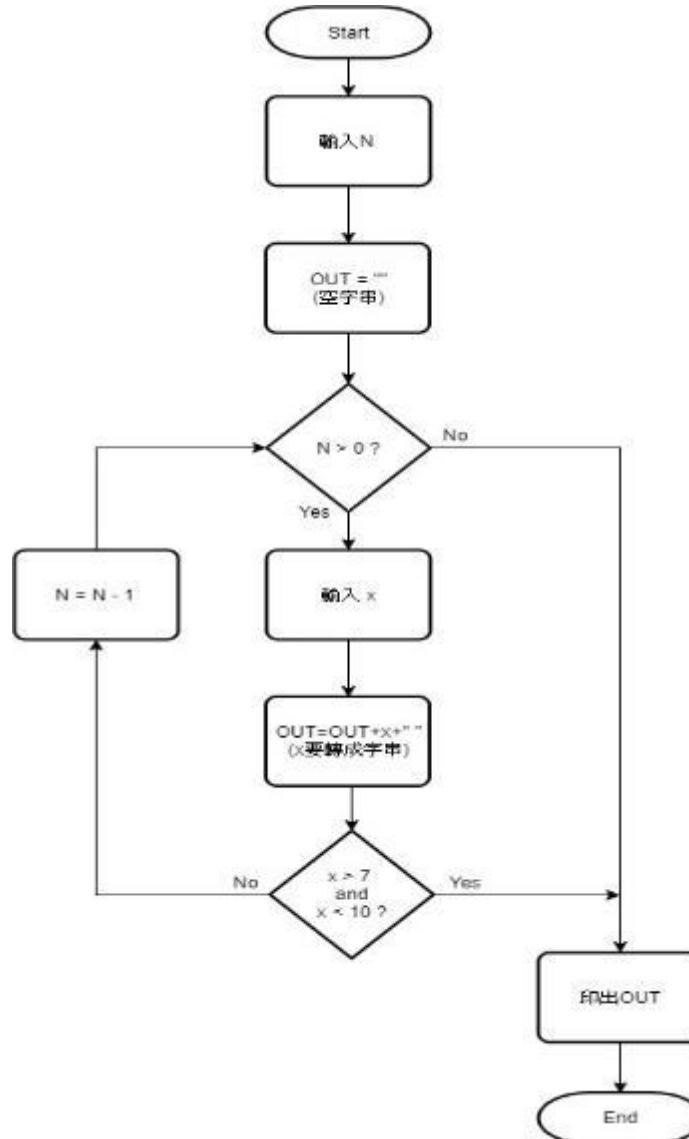
print("小於13的數字總和= ", Sum)
```

練習題

- 5-11. 利用while寫一程式，讀入N個數字，找到第一個大於7而小於10的數字就停止，而且列印出這個數字。
- 5-12. 細菌是非常細小的，除非到達一定的量，否則我們不容易發現它的存在。現在有一種X菌，每隔一分鐘就會成長為原本的3倍，若它一開始只有 1個，請問你要經過幾分鐘才會繁殖到10個(含)以上呢？。

練習題

• 5-11. 流程圖：



練習題

- 5-11. 參考程式：

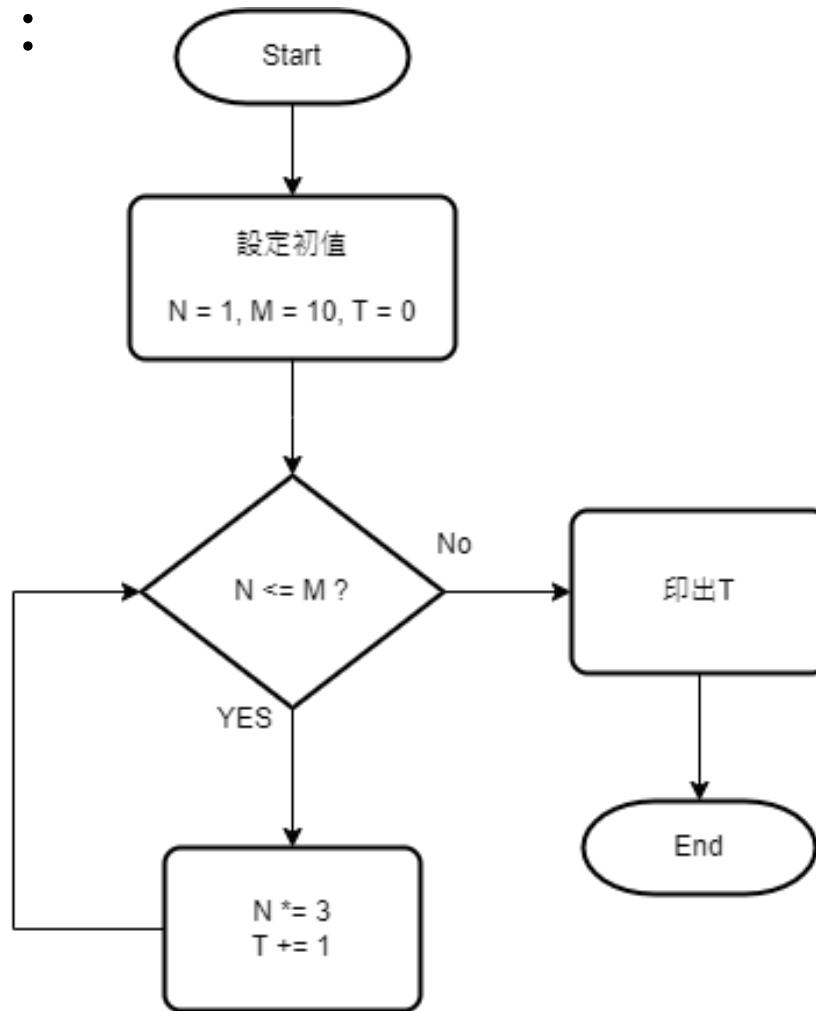
```
N = int(input("請問有幾個數字： "))

while N > 0:
    x = int(input("請輸入數值： "))
    if x > 7 and x < 10:
        break      #此指令會強迫終止迴圈，稍後介紹
    N -= 1

print("停止在： " , x)
```

練習題

- 5-12. 流程圖：



練習題

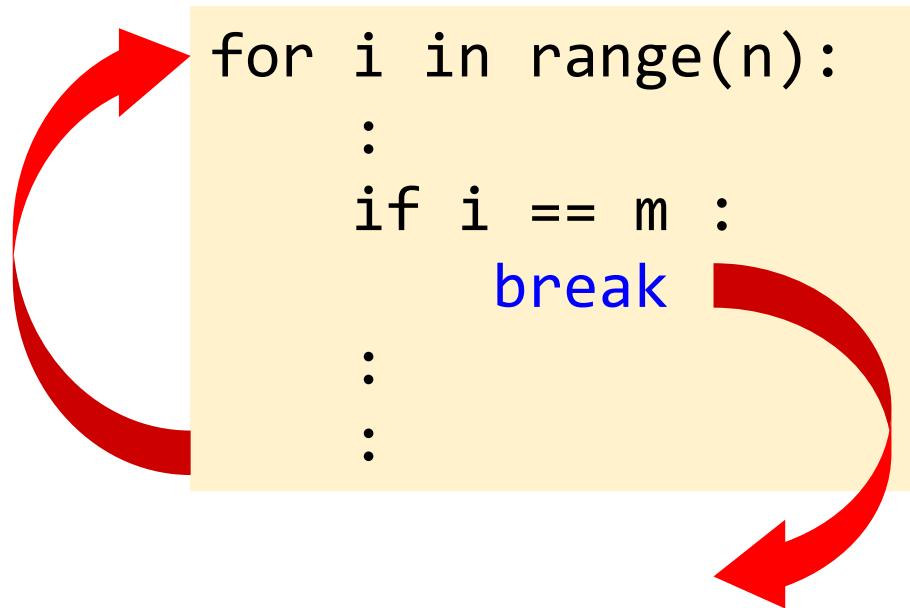
- 5-12. 參考程式

```
N = 1      #一開始的X菌數  
M = 10     #要到達(或超過)的菌數  
T = 0      #所需時間
```

```
while N <= M:  
    N *= 3  
    T += 1  
print(f"需{T}分鐘")
```

迴圈補充說明

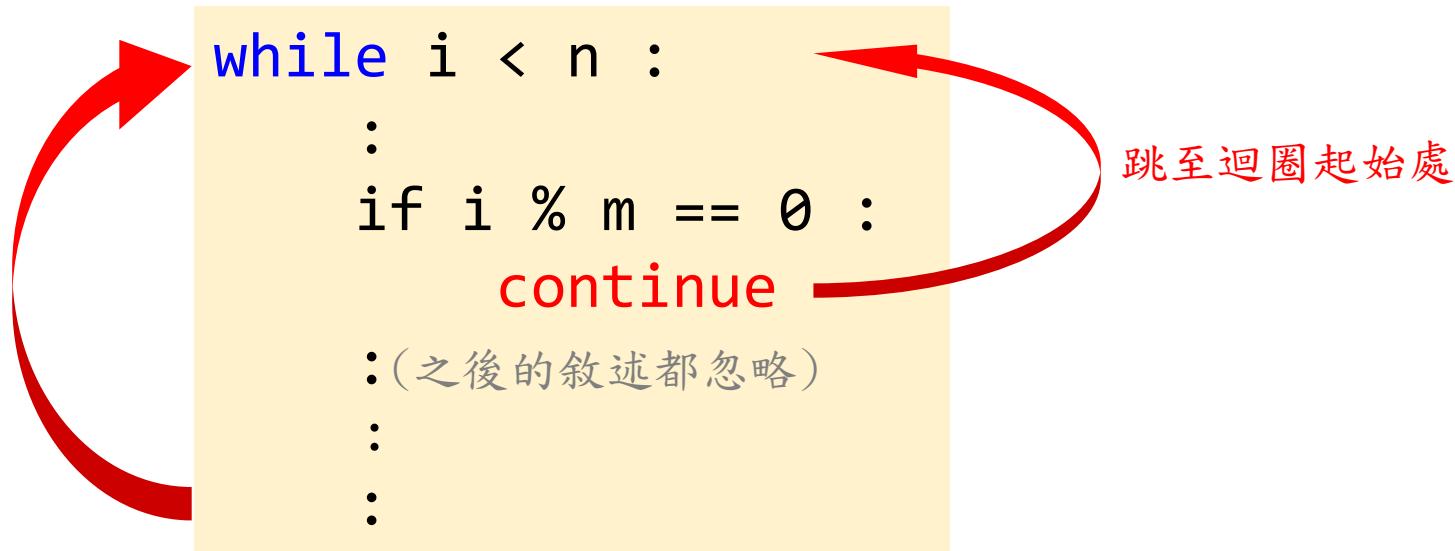
- **break**敘述：強制終止迴圈，會直接跳到迴圈之後的敘述去執行。



跳出迴圈，執行迴圈外的下一道敘述，
注意一次只跳出一層迴圈。

迴圈補充說明

- **continue**敘述：強制回到迴圈初始的地方，忽略之後的敘述。



- **break**和**continue**指令皆可用於**for**或**while**裏。

無盡迴圈

- 當while的條件永遠為真時，稱無盡迴圈，會一直執行下去，我們必須在迴圈中設一個跳出迴圈的條件，否則會執行到當機為止。
 - 聽起來「條件永遠為真」好像怪怪的，但其實還蠻常用到無盡迴圈的。

會一直執行
迴圈

while True:

2

if 某些條件 :

break

2

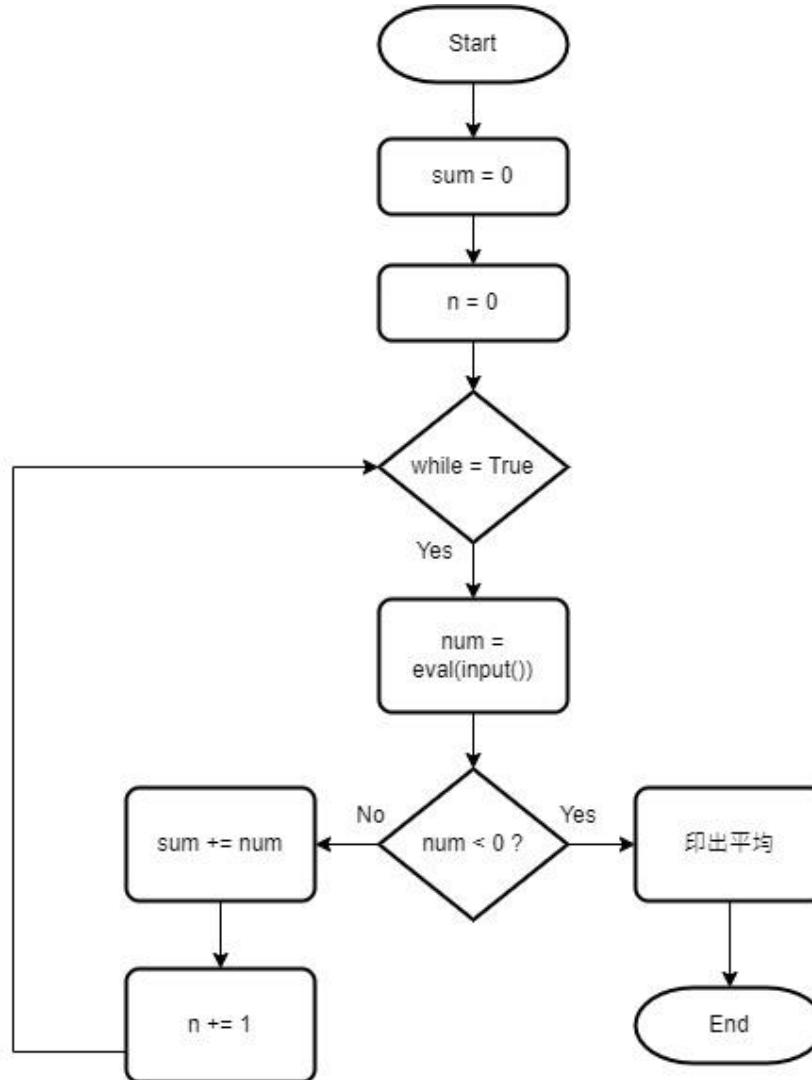
直到“某些條件”
成立才跳離迴圈

練習題

- 請使用無盡迴圈來做。
- 5-13. 老師請小明將同學的國文成績輸入到電腦累加起來並計算平均，但小明不知道有多少成績要輸入，所以設定如果輸入負數時就停止，然後印出平均分數。
- 5-14. 電腦裡有一個祕密數字，介於1~100之間，請使用者猜是多少，如果輸入的數字比答案大，就回應"太大了"，如果輸入的數字比答案小，就回應"太小了"，直到猜對為止。
(秘密數字自己指定，之後介紹亂數時會請電腦出題)

練習題

- 5-13. 流程圖：



練習題

- 5-13. 參考程式：

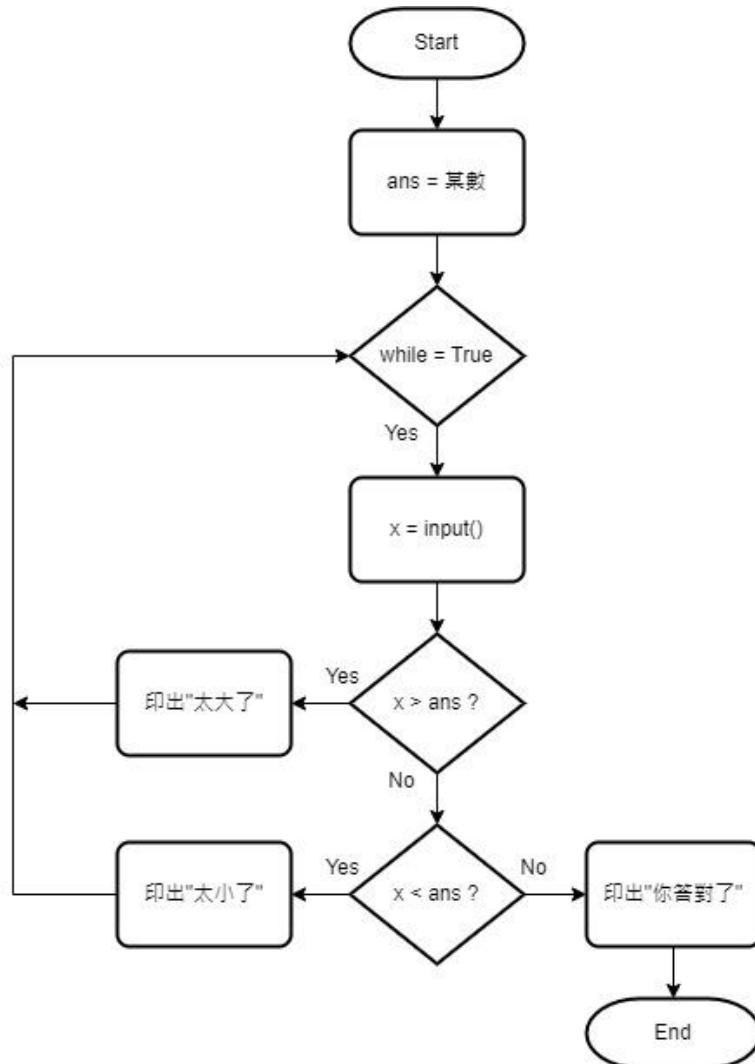
```
sum = 0 #記錄總成績用
n = 0    #紀錄輸入了幾個資料

while True: #無盡迴圈
    num = eval(input("請輸入成績:"))
    if num < 0: #離開迴圈的條件
        break
    sum += num
    n += 1

print("總平均=%5.1f" % (sum/n)) #顯示到小數一位
```

練習題

• 5-14. 流程



練習題

- 5-14. 參考程式：

```
ans = 36 #秘密數字，自己決定

while True:
    x = int(input("請輸入數字："))
    if x > ans:
        print("太大了")
    elif x < ans:
        print("太小了")
    else:
        print("答對了")
        break #除非答對了，否則迴圈會一直執行
```

綜合練習

- 請輸出下列圖形。

1.

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

2.

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

3.

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

綜合練習

- 參考程式：
- 1.

```
for i in range(1, 7):
    print('*' * i)
```
- 2.

```
for i in range(1, 7):
    print(' ' * (7-i), '*' * i)
```
- 3

```
s = 5
for i in range(1, 12, 2):
    print(' ' * s, '*' * i)
    s -= 1
```

綜合練習

- 一個數等於它所有因數的合(不包括它本身)，這種數我們稱它為「完全數」。

例如：

$$6 = 1 + 2 + 3$$

$$28 = 1 + 2 + 4 + 7 + 14$$

- 請寫一程式，找出1 ~ 10000之間所有的完全數。

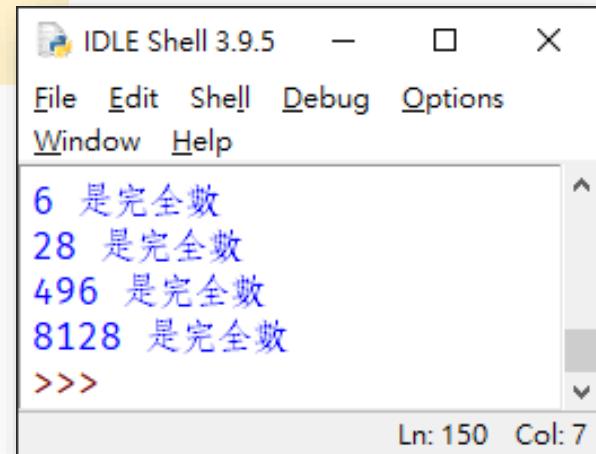
10個の完全数

6
28
496
8128
33550336
8589869056
137438691328
2305843008139952128
2658455991569831744654692615953842176
191561942608236107294793378084303638130997321548169216

綜合練習

- 參考程式：

```
for i in range(1, 10000):
    check = 0
    for j in range(1, i):
        if(i % j == 0):
            check += j
    if(check == i):
        print(i, "是完全數")
```



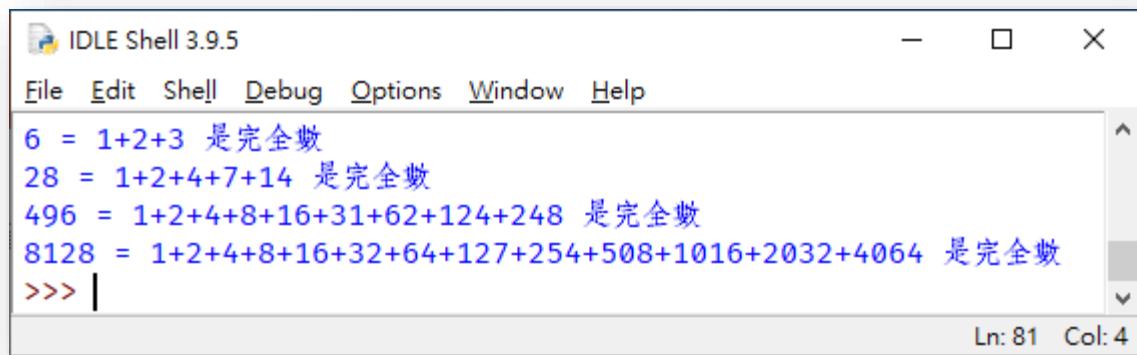
The screenshot shows the IDLE Shell 3.9.5 interface. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the output of the Python script, which prints five perfect numbers: 6, 28, 496, 8128, and their respective descriptions as perfect numbers.

```
6 是完全數
28 是完全數
496 是完全數
8128 是完全數
>>>
```

綜合練習

- 參考程式(詳細顯示版)：

```
for i in range(1, 10000):
    check = 0
    s = ''      #用來記錄所有因數
    for j in range(1, i):
        if(i % j == 0):
            check += j
            s += str(j) + '+'
    if(check == i):
        s = s[:len(s)-1]    #去掉最後多的一個 '+'
    print(i, "=", s, "是完全數")
```



綜合練習

- 韓信（前230年—前196年）是楚漢相爭期間為劉邦拚江山的名將。有一次戰役他帶著1500名士兵戰鬥，約有四百多名士兵死亡。當他們回到總部時，需要知道有多少士兵倖存下來。
- 然後他命令士兵3名一排，最後一排剩2名士兵。
然後他命令他們5名一排，最後一排剩3名士兵。
然後他命令他們7名一排，最後一排剩2名士兵。
- 請問有多少士兵倖存下來？



韓信點兵

劉邦：『卿部下有多少兵卒？』

韓信：『兵不知其數，三三數之剩二，五五數之剩三，七七數之剩二。』

綜合練習

- 參考程式：

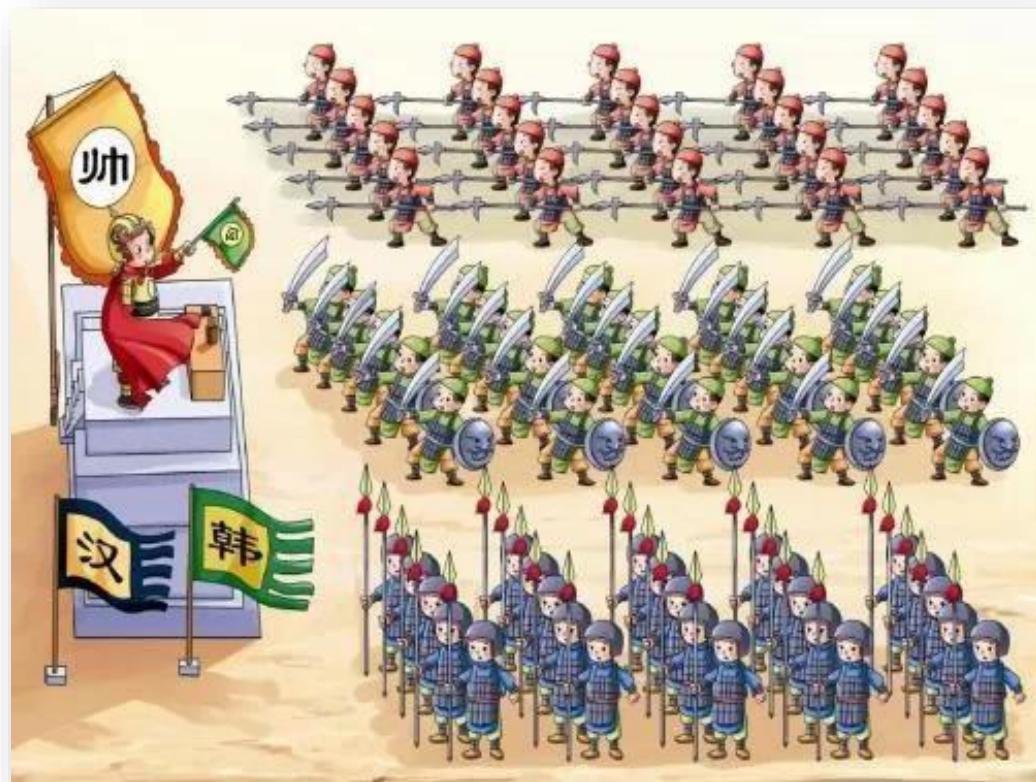
```
for i in range(1000, 1100):
    if(i%3==2 and i%5==3 and i%7==2):
        print(i)
```

- 說明：韓信已知損失兵力約400~500人，所以1500人剩下應該是1000~1100人之間，所以只要找這之間符合的數字即可。
- 答案是1073人。



綜合練習

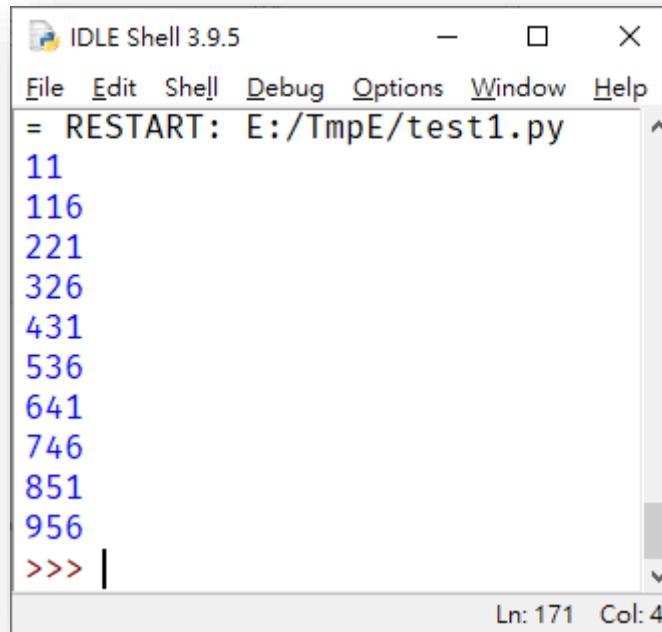
- 類似題：找出1 ~ 1000之間所有滿足「三個三個一數餘2、五個五個一數餘1、七個七個一數餘4」的數字。



綜合練習

- 2. 參考程式：

```
for i in range(1, 1000):
    if(i%3==2 and i%5==1 and i%7==4):
        print(i)
```



The screenshot shows the IDLE Shell 3.9.5 interface. The title bar reads "IDLE Shell 3.9.5". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the command "RESTART: E:/TmpE/test1.py" followed by a list of numbers: 11, 116, 221, 326, 431, 536, 641, 746, 851, and 956. At the bottom, there is a status bar with "Ln: 171 Col: 4".

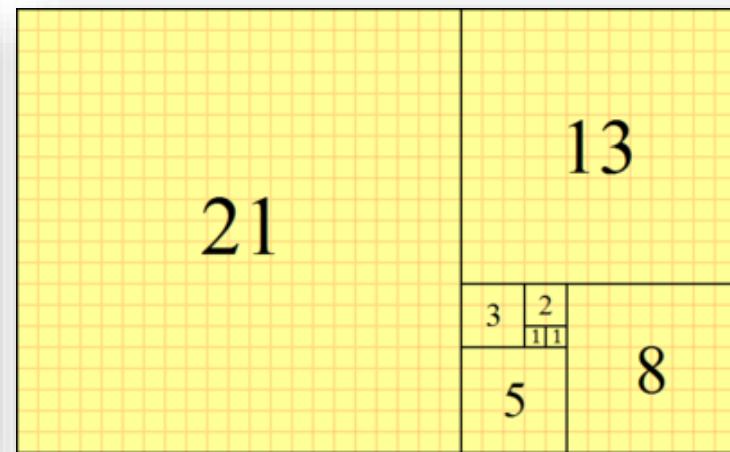
綜合練習

- 費波那西數(義大利語:Successione di Fibonacci)，又稱費氏數列、黃金分割數列。它隱藏在大自然中，如果你注意就會觀察到。它的原則是每一個數字都是前兩個數字的相加。

$$F_\theta = \theta$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}$$



- 請寫一程式計算數列的第n個數字是什麼。(0不算)

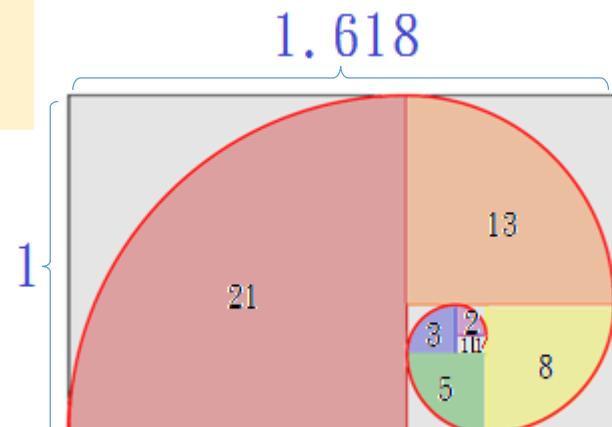
綜合練習

- 參考程式：

```
f1 = 1 #第一及第二個直接指定
f2 = 1
n = int(input("請輸入 n : "))
for i in range(3, n+1):
    fn = f1 + f2
    f1, f2 = f2, fn
print("第", n, "個費氏級數是", fn)
```



```
IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
===== RESTART: E:\TmpE\test1.py =====
請輸入n: 8
第 8 個費氏級數是 21
>>> |
```



1, 1, 2, 3, 5, 8, 13, 21, 34, ...

綜合練習

- 質數判斷：對於一個正整數N來說，如果它的因數只有1和N本身，沒有其他的因數，則我們稱它為「質數」。現在給你一個正整數，請你判斷它是不是質數。
- 也就是除了1和N本身之外，沒有任何數可以整除N，則這個數就稱為質數。
- 質數在密碼學中有很大的用處。

1~100之間的質數

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

綜合練習

- 參考程式：

```
N = int(input("請輸入一個整數:"))
if N < 2 or N % 2 == 0: #排除2以下及所有偶數
    s = "這不是質數"
elif N == 2: #2是質數
    s = "這是質數"
else:
    s = "這是質數"
    i = 3
    while i <= int(N**0.5):
        if N % i == 0:
            s = "這不是質數"
            break
        i += 2
print(s)
```

關於AI

- 雖然現在AI可以幫你產生程式碼了，但你得看得懂啊，加油~

和師 請用PYTHON寫一個判斷某數是否為質數的程式

當然，我可以幫你寫一個判斷某數是否為質數的Python程式。以下是一個簡單的實現：

```
python
def is_prime(number):
    if number <= 1:
        return False

    for i in range(2, int(number ** 0.5) + 1):
        if number % i == 0:
            return False

    return True
```

這個函數名為`is_prime`，接受一個整數`number`作為參數，並返回一個布爾值，表示該數是否為質數。

這個函數的運作原理如下：

- 首先，如果`number`小於等於1，則直接返回`False`，因為1及以下的數不被視為質數。
- 接下來，使用一個迴圈從2開始迭代到`number`的平方根(`int(number ** 0.5) + 1`)。
 - 在迴圈中，檢查`number`是否可以被當前的迭代值整除，如果可以，則`number`不是質數，返回`False`。
- 如果迴圈結束後都沒有找到可以整除的數，返回`True`。

Send a message. ➤

下課~

資料來源：

1. 本講義部分圖片取自網路。
2. 部分取自「程式語言與設計」課本(碁峰、科友)。
3. 專門為中學生寫的程式語言設計(使用Python)，李家同。

