Chapter 5：判斷式 X CAPSTONE 2

本章重點

if 條件判斷會讓程式的執行順序（控制流程）改變，通常用於決策的事件。

準備材料

可上網的電腦

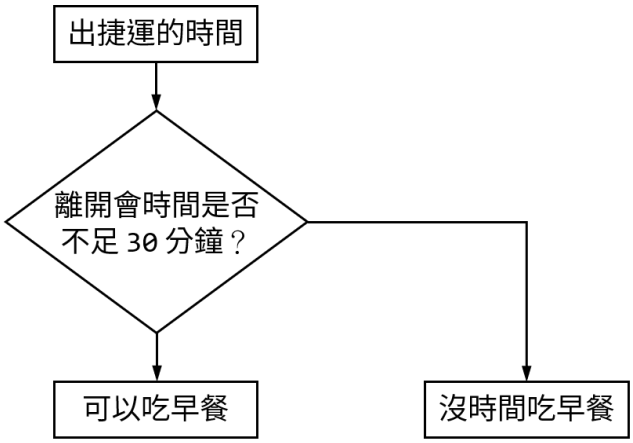
學習目標

1. 如何執行程式的決策判斷。
2. if-else 的結構。
3. CAPSTONE 2：文字冒險旅程。(請參考電子檔)

5-1. 程式的決策判斷

情境想一想

星期一早上，你的第一場會議開始時間是 8:30 AM。吃早餐的時間約 30 分鐘。現在假設走出捷運看一下時間，你如何判斷使否還有時間吃早餐呢？請使用下列流程圖來判斷。



參考答案

只要在 8:00 AM 前走出車站，就還有時間吃早餐。

平日我們做決定時，常會問一些相關的問題，回答的都是是/否、真/假、或對/錯。

例	天氣好嗎？人們回答 yes / no。電腦回答 True / False。

觀念驗證 5.1

依據下列問題，回答是 (True) 或否 (False)？

1. 你會怕黑嗎？
2. 你的口袋大小放得下你的手機嗎？
3. 你今晚會去看電影嗎？
4. 5 乘以 5 的結果是 10 嗎？
5. nibble 這個單字的長度比 google 長？

5-1-1. if 條件式

布林運算式

問句在程式中變成敘述，Python 會回覆是或不是 (True / False)。敘述可經過運算得到單一數值，稱為運算式 (expression)。

- 布林運算式 (boolean expression)：經過運算得到 True / False 的布林值 (bool)，又稱之為邏輯運算式或條件式。

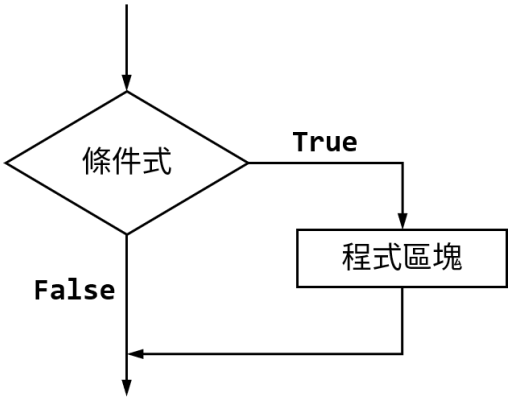
> 程式設計師的思考
>
> 電腦依據 True / False 進行判斷，也就是布林運算式到最後都可簡化成 True / False。

if 條件式語法

if 條件式，類比於口語的『 若 (if)…，則 (then)… 』。

- 若到美國，則一定出國去了。(程式能執行，邏輯正確)
- 若出國，則一定到美國去了。(程式能執行，但邏輯錯誤。Fetal Error! 致命的錯誤)

```
if 條件式:        # : 不能省略
    程式區塊        #  屬於 if 的程式區塊，要縮排 4 個空格
```



當條件式為 True 時就執行程式區塊內的敘述，否則就略過。例如：

```
a = int(input('Enter an interger: '))

if a < 1:
    a += 1
    b = a + 3

print(b)
```

注意：若條件 a < 1 不成立，變數 b 就不會被指派 (b = a + 3)，那麼 print(b) 就會出錯，因為 b 沒有被指定任何物件 (初始化)。

TIP 不屬於 if 的程式區塊，條件是否成立，最後還是會繼續執行 (循序執行的規則)。

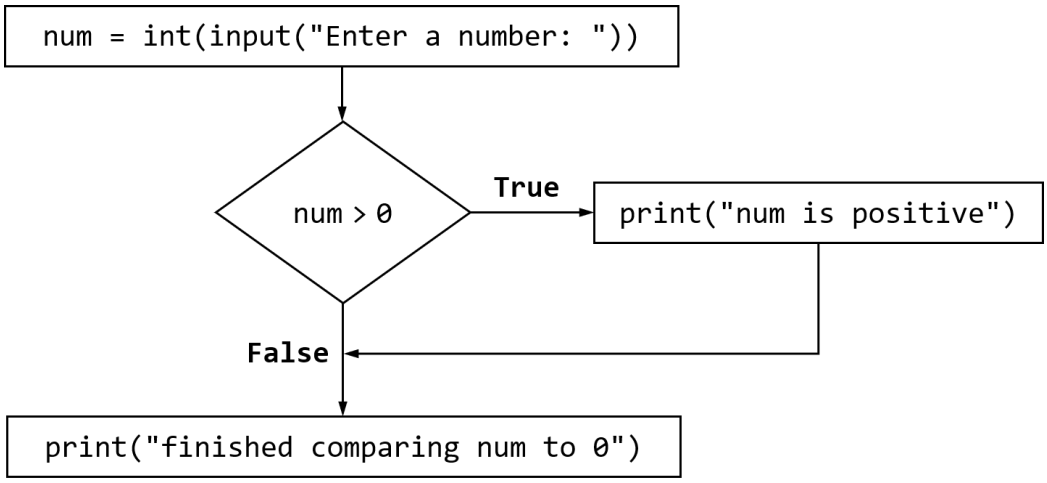
範例

一個簡單的條件式範例，判斷輸入的數值是否為正數 (只要判斷是否大於 0)。

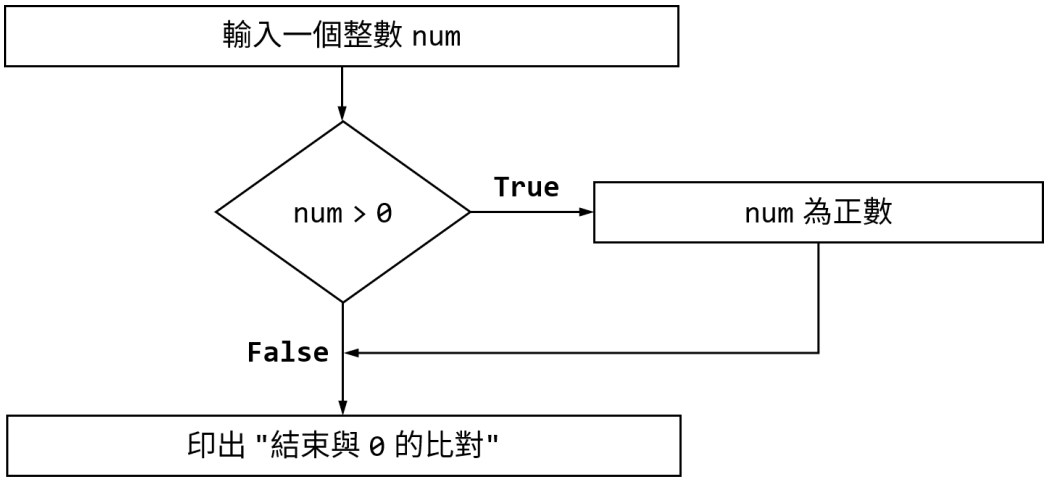
5-1 判斷輸入的數字是否為正數。

```
num = int(input('Enter a number: '))

if num > 0:
    print('num is positive')

print('finished comparing num to 0')
```

5-1 可繪製成以下流程圖：(語法版流程圖，不建議)



5-1 可繪製成以下流程圖：(虛擬碼流程圖，建議，因為大家普遍聽得懂，也好轉換到不同程式語言。)



觀念驗證 5.2

執行下列程式碼：

```
num = int(input('Enter a number: '))

if num < 10:
    print('num is less than 10')

print('Finished')
```

分別輸入下列 3 種不同數值，各會顯示何種結果？

1. 輸入 num 值 5。
2. 輸入 num 值 10。
3. 輸入 num 值 100。

觀念驗證 5.3

依據下列說明和程式碼片段，分別寫出對應的條件式 (conditional statements)：

1. 如果輸入的單字含有空格，顯示 'You did not follow directions.'

```
word = input('Tell me a word: ')
print(word)

if _____________________:
    print('You did not follow directions.')
```

2. 輸入 2 個數字，顯示加總結果，若相加大於 10000，還要另外顯示 'Wow, huge sum!'

```
num1 = int(input('One number: '))
num2 = int(input('Another number: '))
print(num1 + num2)

if _____________________:
    print('Wow, huge sum!')
```

5-1-2. 多個 if 條件式

當程式需要更多判斷，就要使用多個條件式。共有兩種結構：

- 串接的條件式：非巢狀條件 (unnested conditionals)。
- 巢狀條件式 (nested conditionals)。

串接的 if 條件式

多個 if 條件式，成串安排於程式中，每個 if 條件式都被依序執行到，就像循序執行一樣。

5-2 判斷輸入的數字 (數字只有正、負數與 0)。

```
num_a = int(input('Pick a number: '))

if num_a > 0:
    print('Your number is positive')

if num_a < 0:
    print('Your number is negative')

if num_a == 0:
    print('Your number is zero')

print('Finished!')
```

觀念驗證 5.4

依據 **5-2** 畫出流程圖，以確認你了解程式執行的順序和步驟，流程圖是了解程式所有可能執行路徑的最佳圖解。

answer.

巢狀 if 條件式

只有在第 1 層條件式成立 (True) 時，才會執行第 2 層條件式的程式寫法，叫做巢狀條件式 (nested conditionals)。

- 利用巢狀條件式進行比較複雜的判斷。
- 透過流程圖，從上而下了解這個程式的主要流程。

情境想一想

有些事情是在某種條件成立之下，才需要再做下一步的條件判斷。例如：你在賣場看到很多不同品牌的麥片，這時候應該想想家中的麥片是否吃完了？然後再考慮買哪一種麥片？

5-3 巢狀條件式 VS 非巢狀條件式。

```
# 巢狀條件式 (5-3a.py)
num_a = int(input('Number? '))
num_b = int(input('Number? '))

if num_a < 0:
    print('num_a is negative')
    if num_b < 0:
        print('num_b is negative')

print('Finished')
```

第一個條件成立，才會執行第二個條件

```
# 非巢狀條件式 (5-3b.py)
num_a = int(input('Number? '))
num_b = int(input('Number? '))

if num_a < 0:
    print('num_a is negative')

if num_b < 0:
    print('num_b is negative')

print('Finished')
```

第一個條件是否成立，都會執行第二個條件

num_a	num_b	巢狀條件式	非巢狀條件式
-9	5	num_a: is negative Finished	num_a: is negative Finished
9	5	Finished	Finished
-9	-5	num_a: is negative num_b is negative Finished	num_a: is negative num_b is negative Finished
9	-5	Finished	num_b is negative Finished

巢狀 if 條件式：範例 (適合的情境，才需要巢狀 if)

先使用文字虛擬碼

超市入口處，一個惱人的巧克力優惠活動：

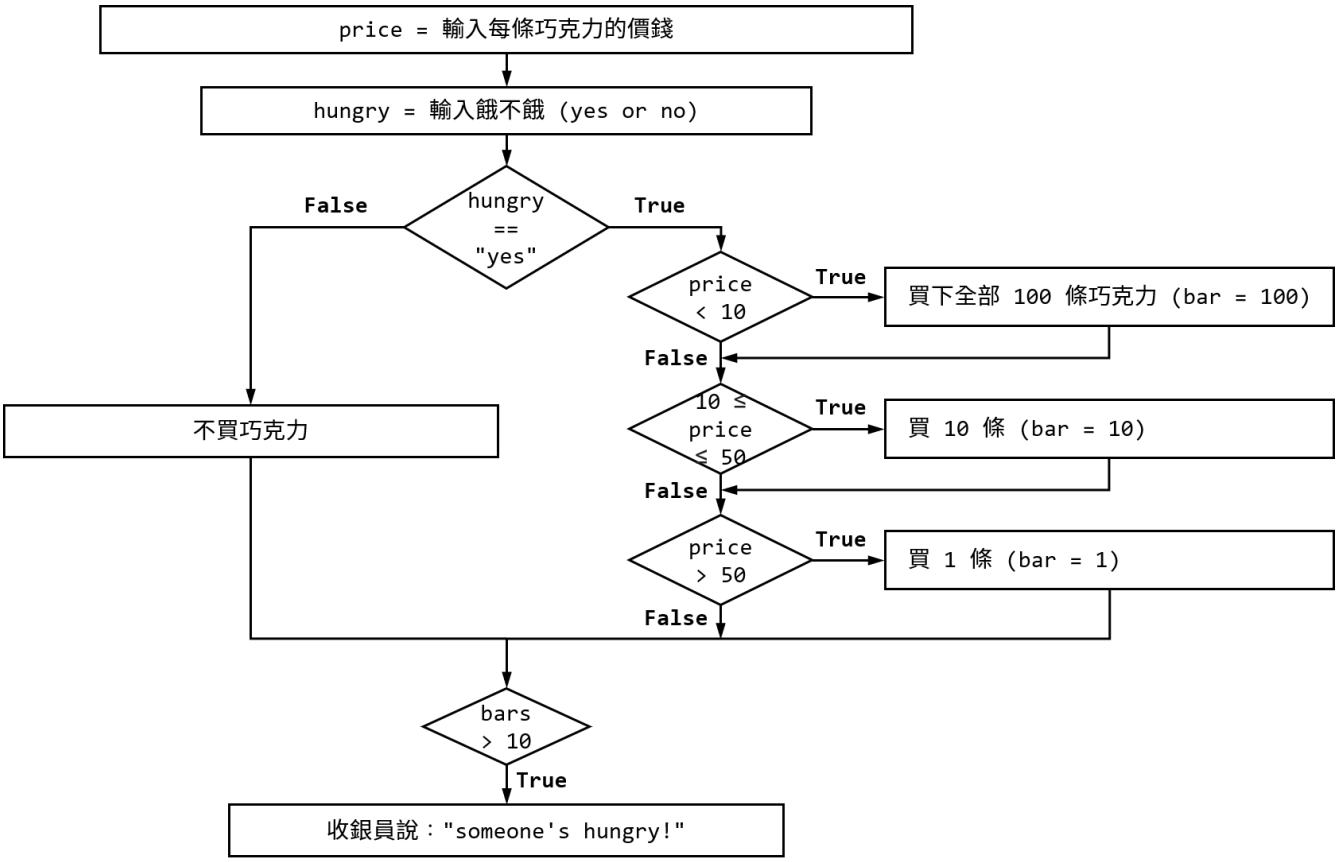
- 每一條巧克力的價錢？
- 你覺得餓嗎？
- 覺得餓，且巧克力的價錢低於 10 元，你會把全部的巧克力 (100 條) 買下來。
- 覺得餓，且巧克力售價在 10 ~ 50 元間，則買 10 條。
- 覺得餓，且巧克力售價超過 50 元，買 1 條。
- 如果不餓，就不買巧克力。
- 依據你購買的數量，收銀員在結帳時會對你說不同的話。

改寫的文字虛擬碼 (比較像程式結構，順便設定合適的變數名稱)

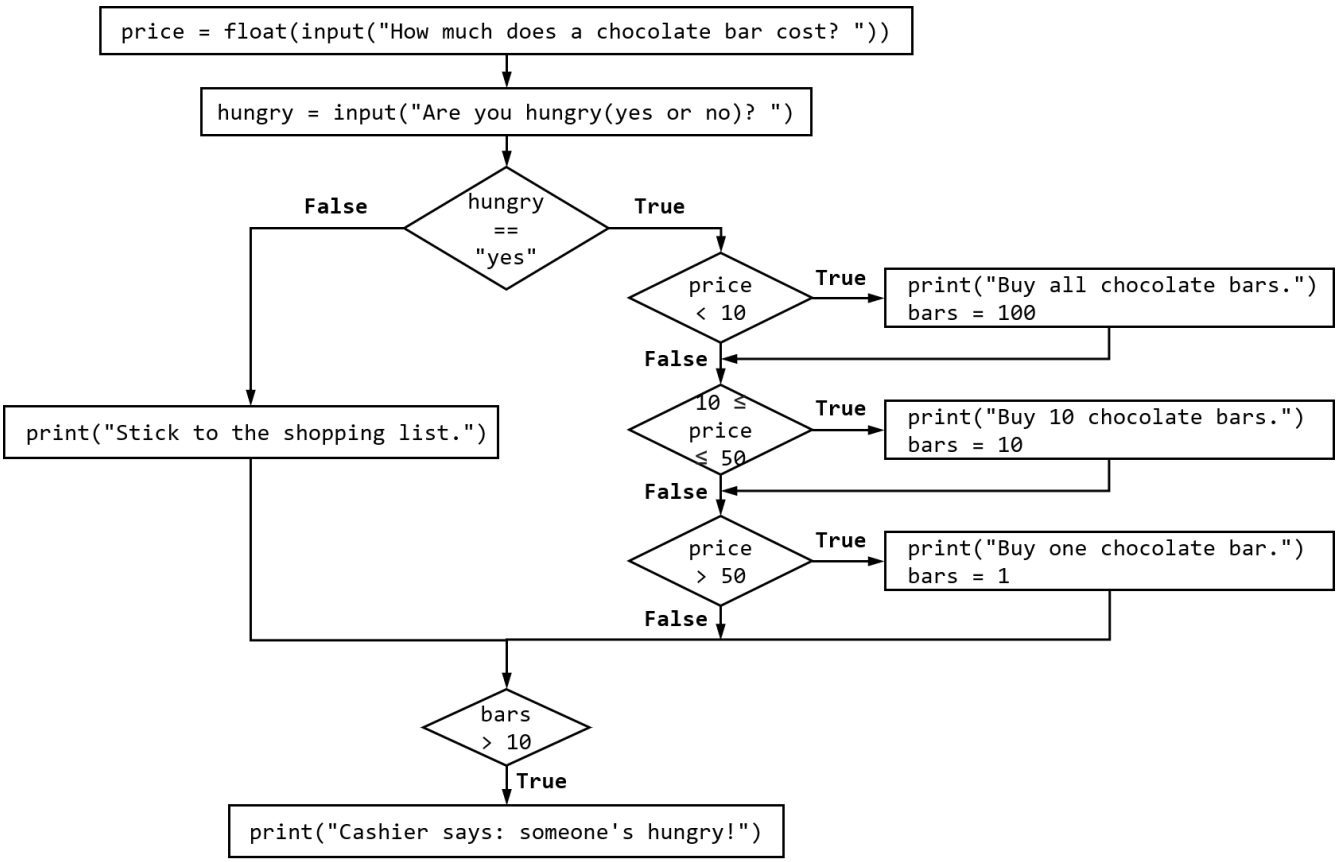
超市入口處，一個惱人的巧克力優惠活動：

- 輸入每一條巧克力的價錢 (price)
- 輸入餓不餓 (hungry) (提醒使用者輸入 yes or no)
- 如果餓 (使用者輸入 yes)
    - ✓ 如果售價低於 10 元
        - 則把全部的巧克力 (100 條) 買下來
    - ✓ 如果售價在 10 ~ 50 元間
        - 則買 10 條
    - ✓ 如果售價超過 50 元
        - 則買 1 條
- 如果不餓 (使用者輸入 no)
    - ✓ 則不買巧克力
- 如果買超過 10 條 (bar)
    - ✓ 則收銀員說：'someone's hungry!'

虛擬碼的流程圖（建議使用）



Python 語法的流程圖（不建議使用）



5-4 巧克力購買的條件判斷。

```
price = float(input('How much does a chocolate bar cost? '))
hungry = input('Are you hungry (yes or no)? ')
bars = 0

if hungry == 'yes':
    if price < 10:
        print('Buy all chocolate bars.')
        bars = 100
    if 10 <= price <= 50:
        print('Buy 10 chocolate bars.')
        bars = 10
    if price > 50:
        print('Buy only one chocolate bar.')
        bars = 1

if hungry == 'no':
    print('Stick to the shopping list.')

if bars > 10:
    print('Cashier says: someone's hungry!')
```

5-1-3. 多條件的邏輯運算式

多條件的邏輯運算式：將兩個問題寫進程式中，該怎麼安排呢？

- 直接把兩個問題合併（使用 and 邏輯運算子）。
- 以巢狀條件式語法（適用外層條件成立時，再執行內層的條件判斷）。

5-5 多條件運算式 VS 巢狀條件式。

```
# 多條件運算式 (5-5a.py)
num_a = int(input('Number? '))
num_b = int(input('Number? '))

if num_a < 0 and num_b < 0:
    print('both negative')
```

```
# 巢狀條件式 (5-5b.py)
num_a = int(input('Number? '))
num_b = int(input('Number? '))

if num_a < 0:
    if num_b < 0:
        print('both negative')
```

5-1-4. 運算子的優先順序

Python 與數學相同，都是先乘除後加減。在布林運算式當中，使用比較運算子與邏輯運算子時，就要注意運算的先後順序，下表說明 Python 運算子的優先順序：

1. 上方的運算子優先於下方的運算子。例如：() 優先於 **。
2. 同一區塊內的運算子，優先順序相同。例如：* 和 / 順序相同。
3. 當優先順序相同時，運算式裡的計算順序是由左而右計算。

運算子	表示
()	括號
**	指數運算
*	乘法運算
/	除法運算
//	除法取商的整數
%	餘數運算
+	加法運算
-	減法運算
==	比較運算，判斷是否相同
!=	比較運算，是否不同
>	比較運算，大於
>=	比較運算，大於等於
<	比較運算，小於
<=	比較運算，小於等於
is	判斷所儲存的物件，是否和另一個變數儲存的物件相同
is not	判斷所儲存的物件，是否和另一個變數儲存的物件不同
in	判斷所儲存的物件，是否在另一個物件裡
not in	判斷所儲存的物件，是否不在另一個物件裡
not	邏輯運算 not
and	邏輯運算 and
or	邏輯運算 or

觀念驗證 5.5

參考運算子優先順序，計算下列運算式的結果。

```
1. 3 < 2 ** 3 and 3 == 3
2. 0 !=4 or (3/3 == 1 and (5 + 1)/3 ==2)
3. 'a' in 'code' or 'b' in 'Python' and
   len('program') == 7
4. 3 + 2 > 5 / 2 and 7 *8 != 6 + 7
5. x = 13
   y = 7
   2 * x**3 * y**2 + 9 * x**2 * y + 3 * x + 15
```

注意：實際寫程式不會有這些惱人的運算式，考試才有。

結論

分支較多時，務必要畫出流程圖來幫助理解。

1. if 條件式內有 4 個空格縮排的程式，表示同屬於 if 的區塊。
2. 可以依照程式邏輯來選擇使用巢狀或非巢狀條件式。
3. 串接（非巢狀）條件式是循序執行的條件式。
4. 巢狀條件式一個條件式（外層），內含另一個條件式（內層）。
5. 流程圖可以讓程式的條件判斷及處理步驟圖形化。
6. 執行運算式時，要注意運算子的優先順序。

練習 5-1

1. 建立一個變數，該變數可以是整數或字串。如果該變數是整數，則顯示 'I'm a numbers person'，如果該變數是字串，則顯示 'I'm a words person'。

2. 輸入的字串中包含空格，則顯示 'This string has spaces'。

3. 指派一個 0~99 之間的整數給變數，當作神秘數字，然後畫面顯示 'Guess my number!(0~99)'，並讓使用者輸入一個數字。如果輸入數字比神秘數字小，則顯示 'Too low'；如果比神秘數字大，則顯示 'Too high'；如果猜中數字則顯示 'You got!'。

4. 請撰寫一個程式讓使用者輸入一個整數，然後顯示該整數的絕對值。

5-2. 進階條件式：if-else 的結構

情境想一想

高速公路有車速限制，太快、太慢都不行，最內側為超車道，不得低於最高速限，但也不能超速 10 公里以上，若車速低於最高限速 10 公里以下，則必須行駛最外側車道，車速最慢不得低於高速公路最低速限。試將高速公路車速與車道的規定，整理成容易識別的表格。

參考答案

假設最高速限：時速 110KM，最低速限：時速 80KM，則其車速與車道的規定如下表：

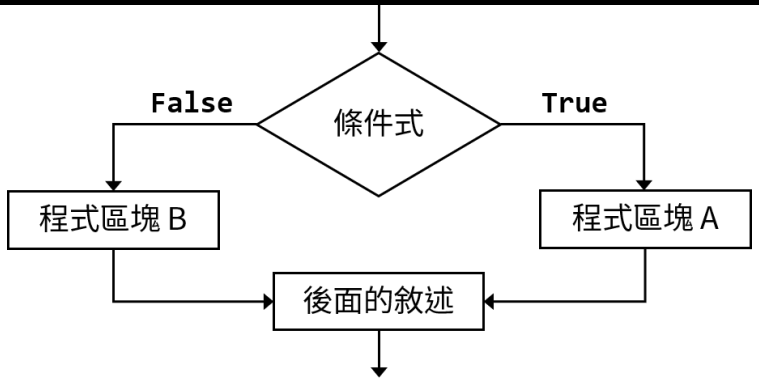
車速	說明
車速 < 80	低於速限，請加速
80 ≤ 車速 < 100	慢速車，請行駛外側車道
100 < 車速 < 110	除最內側，其他車道都可行駛
110 ≤ 車速 ≤ 120	可行駛所有車道
120 < 車速	已超速，請減速

5-2-1. 多重選擇條件式 (if-elif-else)

if-else 語法

if-else：類比於口語的『 如果 (if)…，則 (then)…，否則 (else)… 』。

```
if 條件式:
   程式區塊 A
else:
   程式區塊 B
```



5-6 使用餘數運算，判斷輸入的整數，是奇數還是偶數。

```
Num = int(input('please pick a number: '))

if Num % 2 == 0:
    print('Even Number')
else:
    print('Odd Number')
```

- 一個整數除以 2 ，不是 0 就是 1，適合用 if-else (一定有一個程式區塊會被執行)。

程式設計師的思考

程式實現各種想法，當程式遇到理論，不足就要搜尋並確認理論正確。例如：數學概念。

- 整數：有正整數 (自然數 1, 2, 3, …)，與負整數 (-1, -2, -3, …) 與 0。
- 正、負數：大於 0 稱為正數，小於 0 稱為負數，0 非正非負 (0 不屬於正負數)。

觀念驗證 6.5

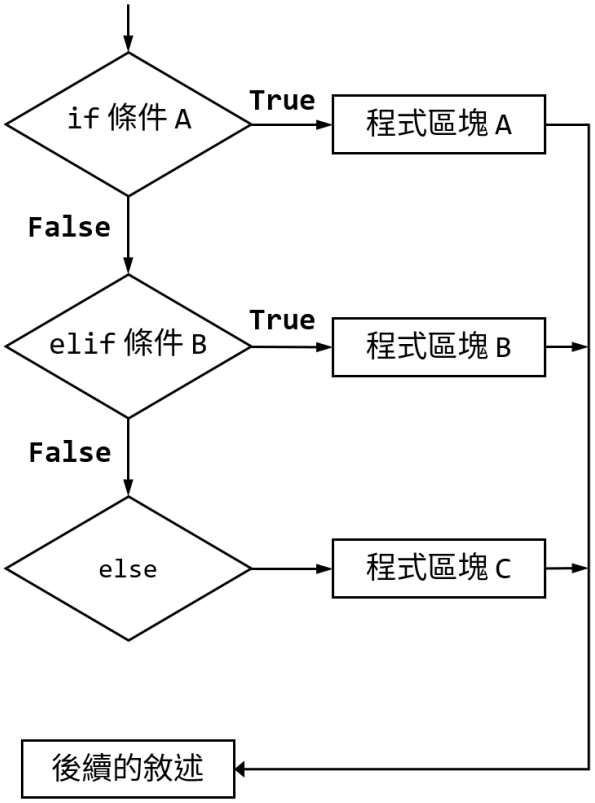
請將 **5-6** 改為判斷正負數 (Num > 0 或 Num < 0)，並試著執行看看，若輸入 0 會發生什麼狀況？

answer.

if-elif-else 語法

if-elif-else：『 如果…，則 A，否則如果…，就 B，否則就 C 』。

```
if 條件 x:
   程式區塊 A
elif 條件 y:
   程式區塊 B
else:
   程式區塊 C
```



依序確認 `if-elif-else` 中的條件，只要某一個條件成立，就會執行該程式區塊，其他條件就不會再進行判斷。例如：**5-7**。

5-7 判斷數字為正數、負數或 0。

```
num = int(input('Enter a number: '))

if num > 0:
    print('num is positive')
elif num < 0:
    print('num is negative')
else:
    print('num is zero')
```

觀念驗證 5.6

執行 **5-7**，輸入不同的 num 值：-3、0、2、1，畫面分別顯示什麼資訊呢？

answer.

觀念驗證 5.7

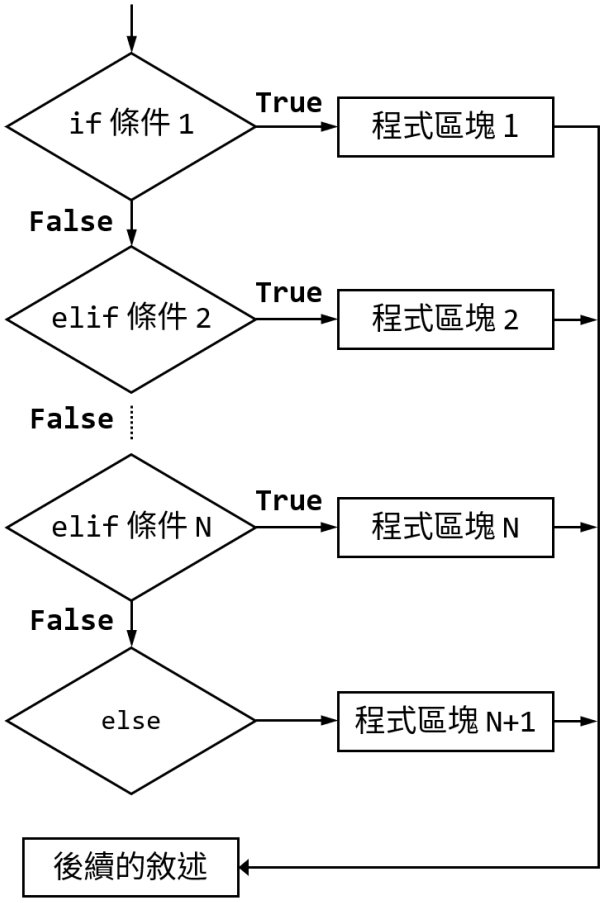
請將 **5-7** 畫成流程圖。

answer.

多個 elif 的情形

依序確認 if-elif-else 中的條件，只要某一個條件成立，就會離開整個 if-elif-else。

- 若所有條件都是 False，則執行 else 的程式區塊。



觀念驗證 5.8

請確認下列程式碼：

if-elif-else 敘述

```
num = int(input('Enter a number: '))
if num < 6:
    print('num is less than 6')
elif num < 10:
    print('num is less than 10')
elif num > 3:
    print('num is greater than 3')
else:
    print('No relation found.')
print('Finished.')
```

if 敘述

```
num = int(input('Enter a number: '))
if num < 6:
    print('num is less than 6')
if num < 10:
    print('num is less than 10')
if num > 3:
    print('num is greater than 3')
print('Finished.')
```

如果 num 值如下，請確認兩種條件式架構的運算結果為何？

num	if-else-if	if
20		
9		
5		
0		
20		

5-2-2. 巢狀 if-elif-else

同巢狀 `if` 一樣，巢狀 `if-elif-else` 每一層區塊利用 4 的倍數來縮排。

例如：**5-8** 猜數字，可以利用巢狀 `if-elif-else` 來做更多的判斷與結果。

5-8 猜數字，與正負數的判斷。

```
num_a = int(input('pick a number: '))
num_b = int(input('pick another number: '))
lucky_num = 7

if num_a == num_b:
    print('You enter the same number')
else:
    if num_a > 0 and num_b > 0:
        print('both numbers are positive')
    elif num_a < 0 and num_b < 0:
        print('both numbers are negative')
    else:
        print('numbers have opposite sign')

if num_a == lucky_num or num_b == lucky_num:
    print('you also guessed my lucky number!')
else:
    print('I have a secret number in mind…')
```

程式設計師的思考

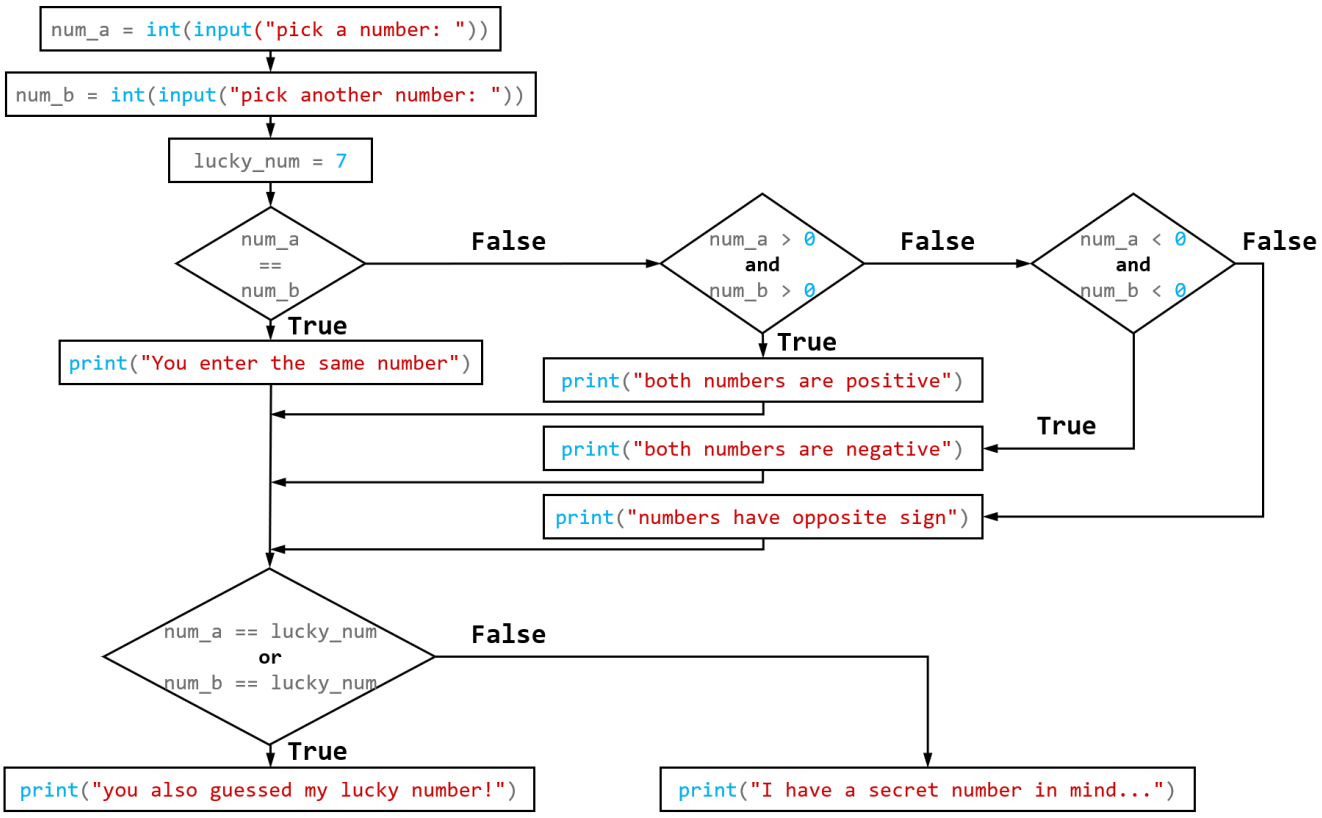
要能寫出可讀性高的程式，使用有意義的變數名稱，會比直接把運算式寫在敘述裡要好很多。例如：

```
if (x**2 - x + 1 == 0) or x + y**3 + x**2 == 0)
```

改寫成下列比較好：

```
xeq = x**2 - x + 1
xy_eq = x + y**3 + x**2
if x_eq == 0 or xy_eq == 0
```

說明 **5-8** 的流程圖：



5-9 改寫 **5-8**，轉換 else 為串接的 elif。

```
num_a = int(input('pick a number: '))
num_b = int(input('pick another number: '))
lucky_num = 7

if num_a == num_b:
    print('You enter the same number')
elif num_a > 0 and num_b > 0:
    print('both numbers are positive')
elif num_a < 0 and num_b < 0:
    print('both numbers are negative')
else:
    print('numbers have opposite sign')

if num_a == lucky_num or num_b == lucky_num:
    print('you also guessed my lucky number!')
else:
    print('I have a secret number in mind…')
```

5-10 英文或西班牙文的判斷。

```
greeting = input('Say hi in English or Spanish! ')
greet_en = ('hi', 'Hi', 'hello', 'Hello')
greet_sp = ('hola', 'Hola')

if greeting not in greet_en and greeting not in greet_sp:
    print('I don't understand your greeting. ')
elif greeting in greet_en:
    num = int(input('Enter 1 or 2: '))
    print('You speak English!')

    if num == 1:
        print('one')
    elif num == 2:
        print('two')

elif greeting in greet_sp:
    num = int(input('Enter 1 or 2: '))
    print('You speak Spanish! ')

    if num == 1:
        print('uno')
    elif num == 2:
        print('dos')
```

結論

if-elif-else 可完成多重條件判斷。

1. if-elif-else 會依序確認各條件式的結果，並執行第一個結果為 True 的程式區塊。
2. 可以把複雜的程式畫成流程圖，這樣可以了解其中的邏輯判斷。

練習 5-2

1. 編寫一個程式，輸入 2 個數字，程式會依據輸入資訊顯示出這 2 個數字間的關係，例如：
   - 這 2 個數字相等；第 1 個數字小於第 2 個數字；第 1 個數字大於第 2 個數字。

2. 試著編寫一個程式，輸入一個字串，如果該字串包含所有的母音字母 (a, e, i, o, u)，則顯示 'You have all the vowels'，如果字串開頭為 a，且結尾為 z，則顯示 'And it's sort of alphabetical'。