

# Python Programming

2023 Spring; week14

Instructor: Cheng-Chun Chang (張正春)  
Department of Electrical Engineering

Textbook: Python程式設計:從入門到進階應用(第三版) 2020

# 課程助教

**協助debug, 禁止同學看code照抄**

---

第一排:

第二排:

第三排:

第四排:

第五排:

第六排:

打游擊:

## Ch5 迴圈與生成式


- 
- 迴圈結構利用指定迴圈變數的**初始條件**、迴圈變數的**終止條件**與**迴圈變數**的增減值來控制迴圈執行次數。
  - 許多問題的解決都涉及迴圈結構的使用
  - 例如：
    - 加總、排序、找最大值…等，善用迴圈結構才能有效利用電腦的運算能力與簡化程式碼。

## Ch5 迴圈與生成式(ch5\5-for.py)

- 假設要撰寫程式產生1000 個「Hello」，若不使用迴圈結構需寫1000 個「`print('Hello')`」，如下表。



- 
- 使用迴圈結構可以簡化程式碼達成相同功能，如下表。

產生 1000 個「Hello」的程式碼 (  : ch5\5-for.py)

```
for i in range(1000):  
    print('Hello')
```

## Ch5 迴圈與生成式(ch5\5-for.py)

| 使用方法  | 範例   | 執行結果                  |
|---|--|-----------------------|
| <p>range( 終止值 )</p> <p>range 函式指定「終止值」，數字串列會到「終止值」的前一個數字為止，沒有指定起始值，預設起始值為 0，沒有指定遞增值，預設為遞增 1。</p>  | <pre>for i in range(5):<br/>    print(i)</pre>   | 0<br>1<br>2<br>3<br>4 |
| <p>range( 起始值, 終止值 )</p> <p>range 函式指定「起始值」與「終止值」，數字串列由「起始值」開始到「終止值」的前一個數字為止，沒有指定遞增值，預設為遞增 1。</p> | <pre>for i in range(2,6):<br/>    print(i)</pre> | 2<br>3<br>4<br>5      |

## Ch5 迴圈與生成式(ch5\5-for.py)

| 使用方法   | 範例   | 執行結果                  |
|--|--|-----------------------|
| <p>range( 起始值 , 終止值 , 遞增 ( 減 ) 值 )</p> <p>range 函式指定「起始值」、「終止值」與「遞增 ( 減 ) 值」，數字串列由「起始值」開始到「終止值」的前一個數字為止，每次遞增或遞減「遞增 ( 減 ) 值」。</p> | <pre>for i in range(2,10,2):<br/>    print(i)</pre>        | 2<br>4<br>6<br>8      |
|  | <pre>for i in<br/>range(100,90,-3):<br/>    print(i)</pre> | 100<br>97<br>94<br>91 |



## Ch5 迴圈與生成式

---

- Python 語言中迴圈結構有**for** 與**while** 兩種，迴圈當中可以包含迴圈稱做**巢狀迴圈**，另外迴圈當中可以跳出迴圈( 使用**break**)，跳過正在執行的迴圈執行迴圈的下一輪( 使用**continue**)。
- 有些迴圈的最後加上**else**，若迴圈正常結束，不是遇到**break** 跳出迴圈，就會執行**else** 程式區塊，以下我們就詳細介紹這些結構。

## 5-1 迴圈結構 — 使用for

---

- **for** 迴圈結構通常用於已知重複次數的程式，迴圈結構中指定迴圈變數的初始值、終止值與遞增(減)值，迴圈變數將由初始值變化到終止值的前一個數字，每次依照遞增(減)的值進行數值遞增或遞減。

## 5-1 迴圈結構 — 使用for

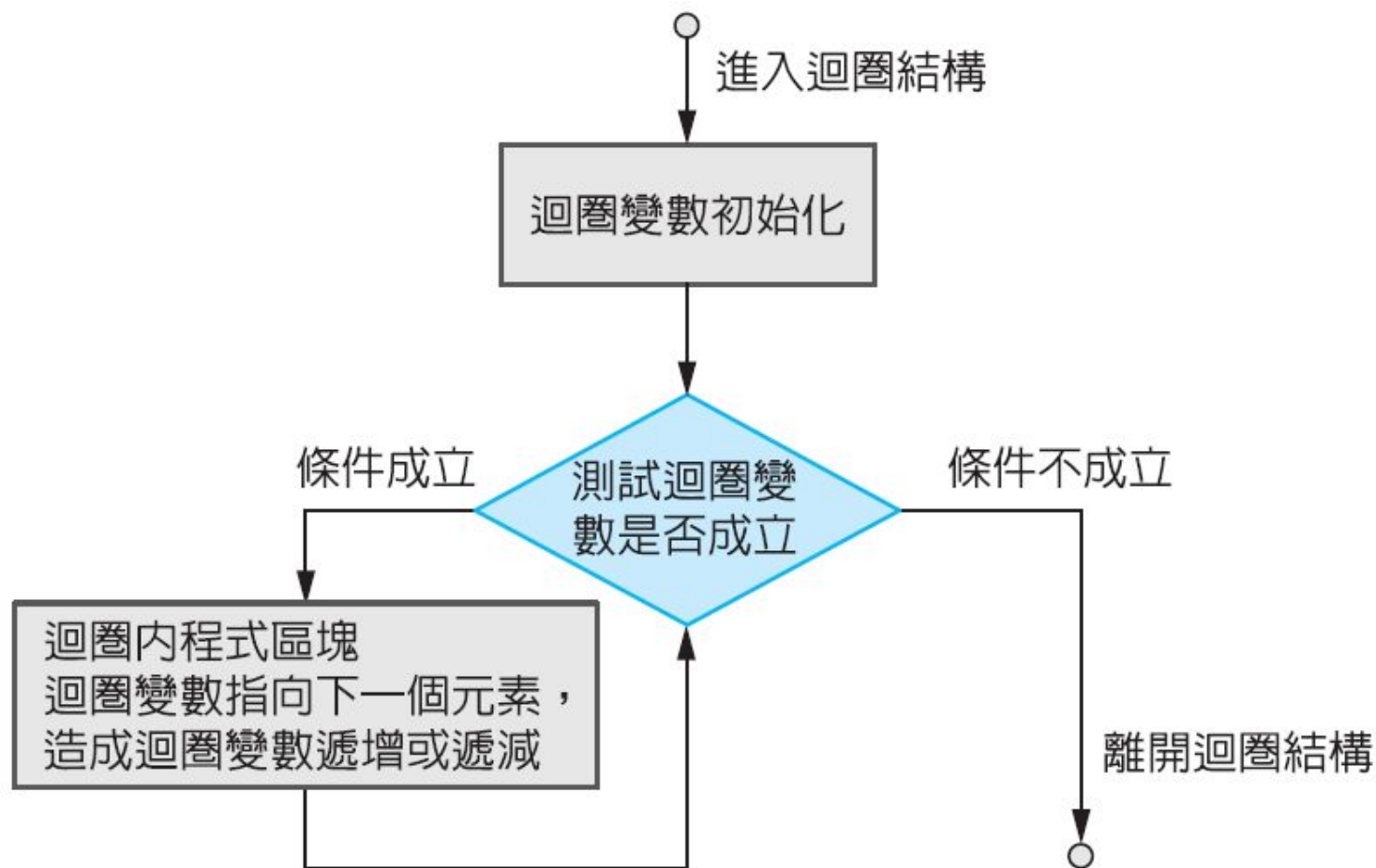


圖 5-1 for 迴圈結構流程圖

表 5-1 for 迴圈結構

| for 程式語法   | 程式範例 ( 印出 1000 個 Hello )                      |
|--|---|
| for 迴圈變數 in range( 起始值 , 終止值 , 遞增 ( 減 ) 值 ):<br>重覆的程式            | for i in range(0, 1000, 1):<br>print('Hello') |
| 說明   |   |
| for 迴圈內迴圈變數由起始值變化到終止值的前一個數字，每重複執行一次程式迴圈變數就會遞增 ( 減 ) 值，重複執行迴圈內程式。 |   |

---

# Programming codes I



## 範例5-1-2 加總 (ch5\5-1-2-sum.py)

---

- 寫一個程式允許使用者輸入加總的開始值、結束值與遞增值，計算數值加總的結果，例如要計算 $3+6+9+12$ 的結果，就輸入3 為開始值，13 為結束值，3 為遞增值。

---

## □ 解題想法

- 可以使用迴圈結構撰寫程式, 迴圈變數起始值為輸入的加總起始值, 迴圈變數終止值為輸入的加總終止值, 迴圈每執行一次迴圈變數就會依照輸入的遞增(減)值進行遞增(減), 迴圈內使用「**sum=sum+ 迴圈變數**」進行數值的加總, 顯示加總的過程。

## 範例5-1-2 加總

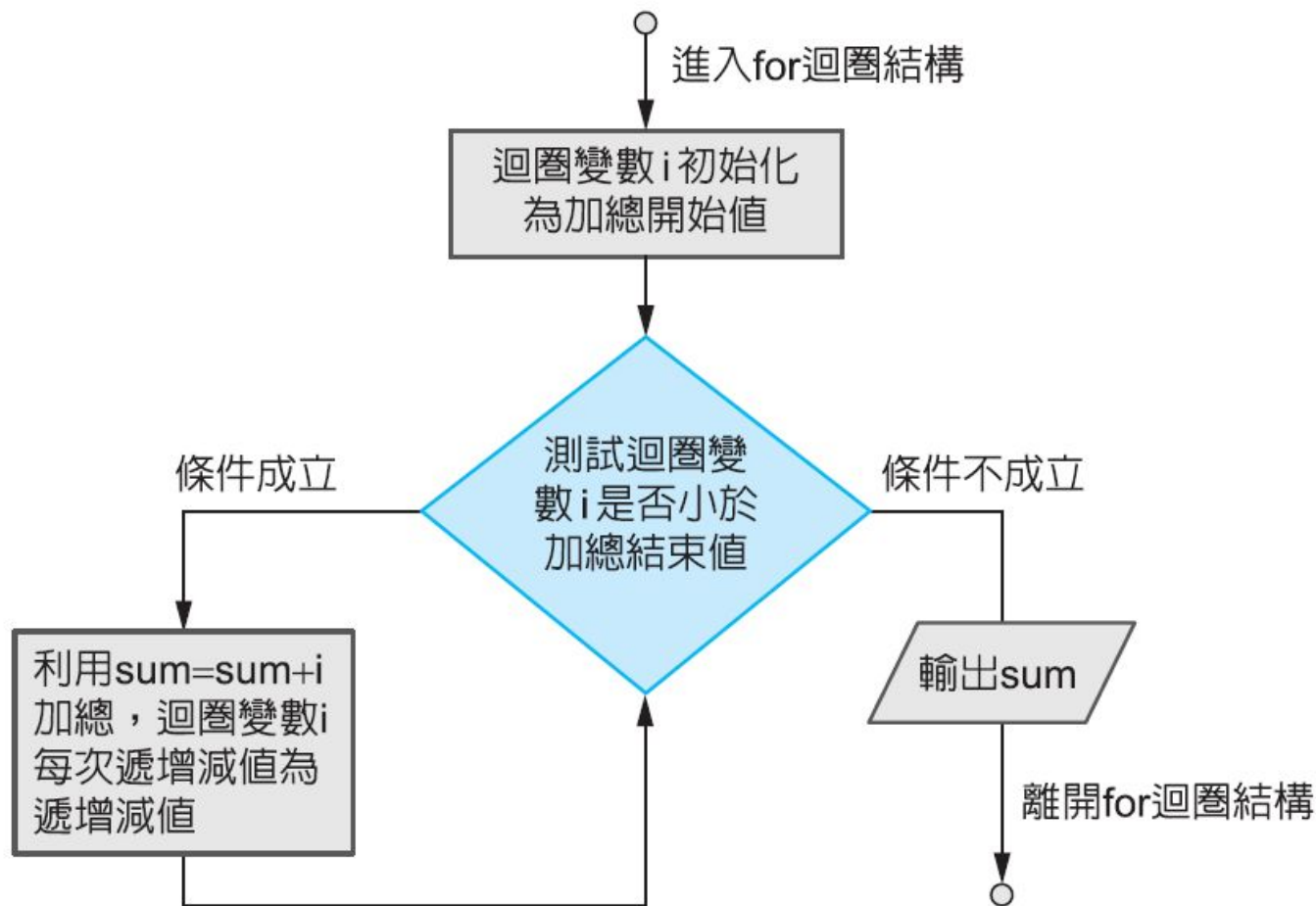


圖 5-3 加總流程图



## 範例5-1-2 加總 (ch5\5-1-2-sum.py)

| 行數 | 程式碼                              | 執行結果   |
|----|----------------------------------|--|
| 1  | s = int(input(' 請輸入加總開始值？ '))    | <p>加總開始值輸入「3」，加總結束值輸入「13」，加總遞增值輸入「3」，結果如下。</p> <p>請輸入加總開始值？ 3<br/>請輸入加總終止值？ 13<br/>請輸入遞增減值？ 3<br/>i 為 3 加總結果為 3<br/>i 為 6 加總結果為 9<br/>i 為 9 加總結果為 18<br/>i 為 12 加總結果為 30</p> |
| 2  | e = int(input(' 請輸入加總終止值？ '))    |  |
| 3  | inc = int(input(' 請輸入遞增減值？ '))   |  |
| 4  | sum = 0                          |  |
| 5  | for i in range(s, e, inc):       |  |
| 6  | sum = sum + i                    |  |
| 7  | print('i 為 ', i, ' 加總結果為 ', sum) |  |

## 範例5-1-2 加總 (ch5\5-1-2-sum.py)

### □ 舉例說明

- 加總使用`sum=sum+i` 原理, 如下表, 在Python 語言中等號右邊「`sum+i`」的算式會先計算, 結果回存到等號左邊(`sum`)。

| <pre>sum = 0 for i in range(3, 13, 3):     sum = sum + i</pre> | i 值  | sum 加總過程    | sum 加總後 |
|--|------|-------------|---------|
|  | i=3  | sum=0 + 3   | sum=3   |
|  | i=6  | sum=3 + 6   | sum=9   |
|  | i=9  | sum=9 + 9   | sum=18  |
|  | i=12 | sum=18 + 12 | sum=30  |

## 範例5-1-2 加總 (ch5\5-1-2-sum2.py)

---

### □ 補充說明:

- Python 提供內建的函式`sum`, 會自動將所輸入的所有串列元素加總起來, 就不需要寫`for` 迴圈與「`sum = sum + i`」, 修改後的程式如下。

| 行數               | 程式碼  | 執行結果   |
|------------------|--|--|
| 1<br>2<br>3<br>4 | <pre>s = int(input(' 請輸入加總開始值？ ')) e = int(input(' 請輸入加總終止值？ ')) inc = int(input(' 請輸入遞增減值？ ')) print(sum(range(s, e, inc)))</pre> | <p>加總開始值輸入「3」，加總結束值輸入「13」，加總遞增值輸入「3」，結果如下。</p> <pre>請輸入加總開始值？ 3 請輸入加總終止值？ 13 請輸入遞增減值？ 3 30</pre> |

## 5-2 迴圈結構——使用while

---

- while 迴圈結構與for 迴圈結構十分類似，while 迴圈結構常用於不固定次數的迴圈，由迴圈中測試條件成立與否，決定是否跳出迴圈，測試條件為真時繼續迴圈，當測試條件為假時結束迴圈。

---

## □ 例如：猜數字遊戲

- 兩人(A 與B)玩猜數字遊戲，一人(A)心中想一個數，另一人(B) 去猜
- A 就B 所猜數字回答「猜大一點」或「猜小一點」，直到B 猜到A 所想數字，這樣的猜測就屬於不固定次數的迴圈，適合使用while 而不適合使用for。

- while 指令後面所接測試條件，若為真時會不斷做迴圈內動作，直到測試條件的結果為假時跳出while 迴圈。

## 5-2 迴圈結構——使用while

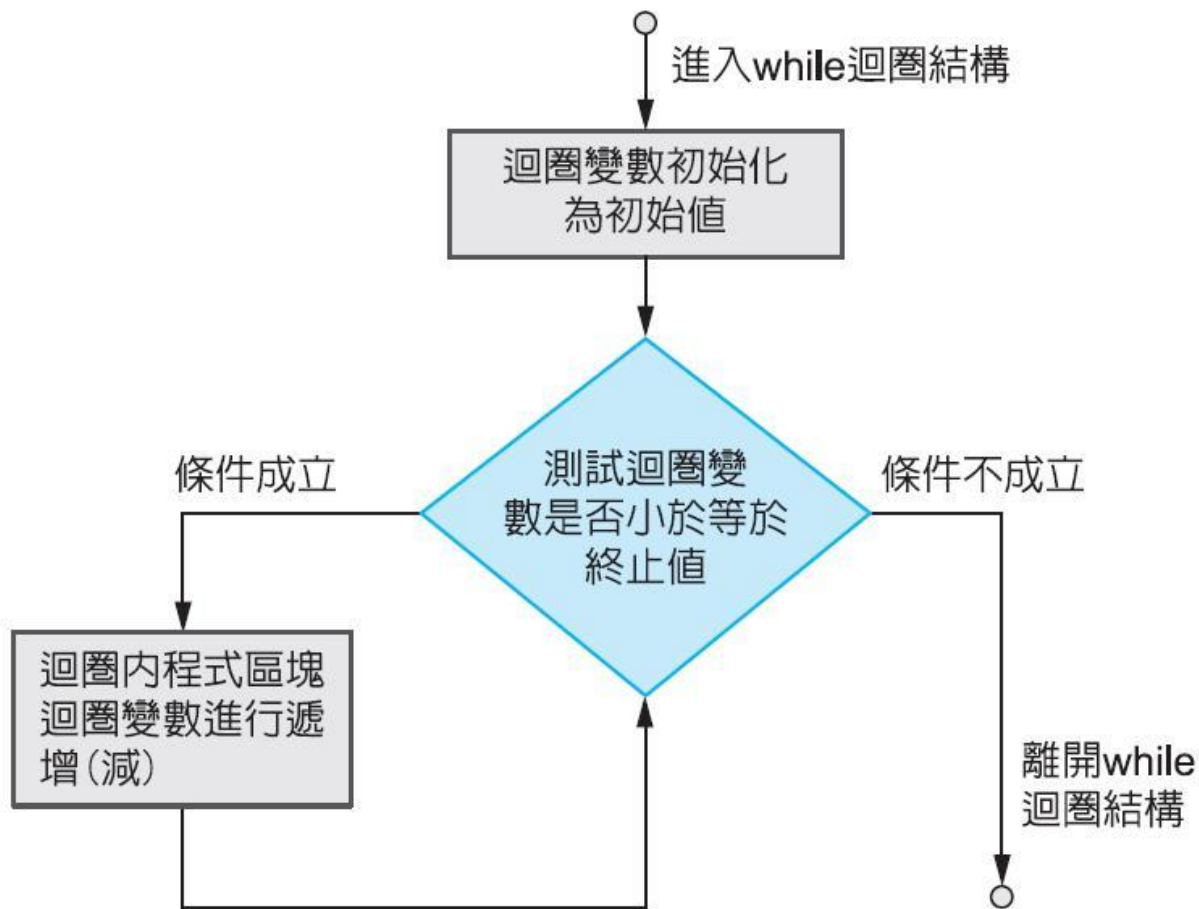


圖 5-4 while 迴圈結構流程圖

表 5-2 while 迴圈結構

| while 迴圈語法  | 程式範例   |
|---|--|
| 迴圈變數 = 初始值<br>while 迴圈變數 <= 終止值 :<br>重覆的程式<br>迴圈變數 = 迴圈變數 + 遞增 ( 減 ) 值  | <pre>i = 3 while i &lt;= 13:     sum = sum + i     i = i + 3</pre> |
| 說明  |  |
| while 迴圈內迴圈變數由起始值變化到終止值，每重複執行一次迴圈變數就會遞增 ( 減 ) 值，重複執行迴圈內程式，直到超過終止值後停止執行。 |  |



---

# Programming codes 2



## 範例5-2-2 猜數字 (ch5\5-2-2-guess.py)

---

- 大家是否玩過一個遊戲，兩人(A 與B) 一起玩，A 心中想一個數字，B 猜A 心中所想的數字
- B 每猜一次，A 就回答「猜大一點」、「猜小一點」與「猜中了」，當B 猜到A 所想的數字遊戲就結束，我們可以將此遊戲寫成程式，**所猜數字介於1 到100。**

---

## □ 解題想法

- 利用Python 語言內建模組random 中的隨機函式randint, 產生介於1 到99 的目標值
- 使用while 迴圈結構不斷允許使用者輸入數字進行猜測, 測試猜測值與目標值是否相等
- 若相等則終止迴圈, 否則根據猜測值與目標值的大小關係, 顯示「猜大一點」、「猜小一點」與「猜中了」等提示。

## 範例5-2-2 猜數字 (ch5\5-2-2-guess.py)

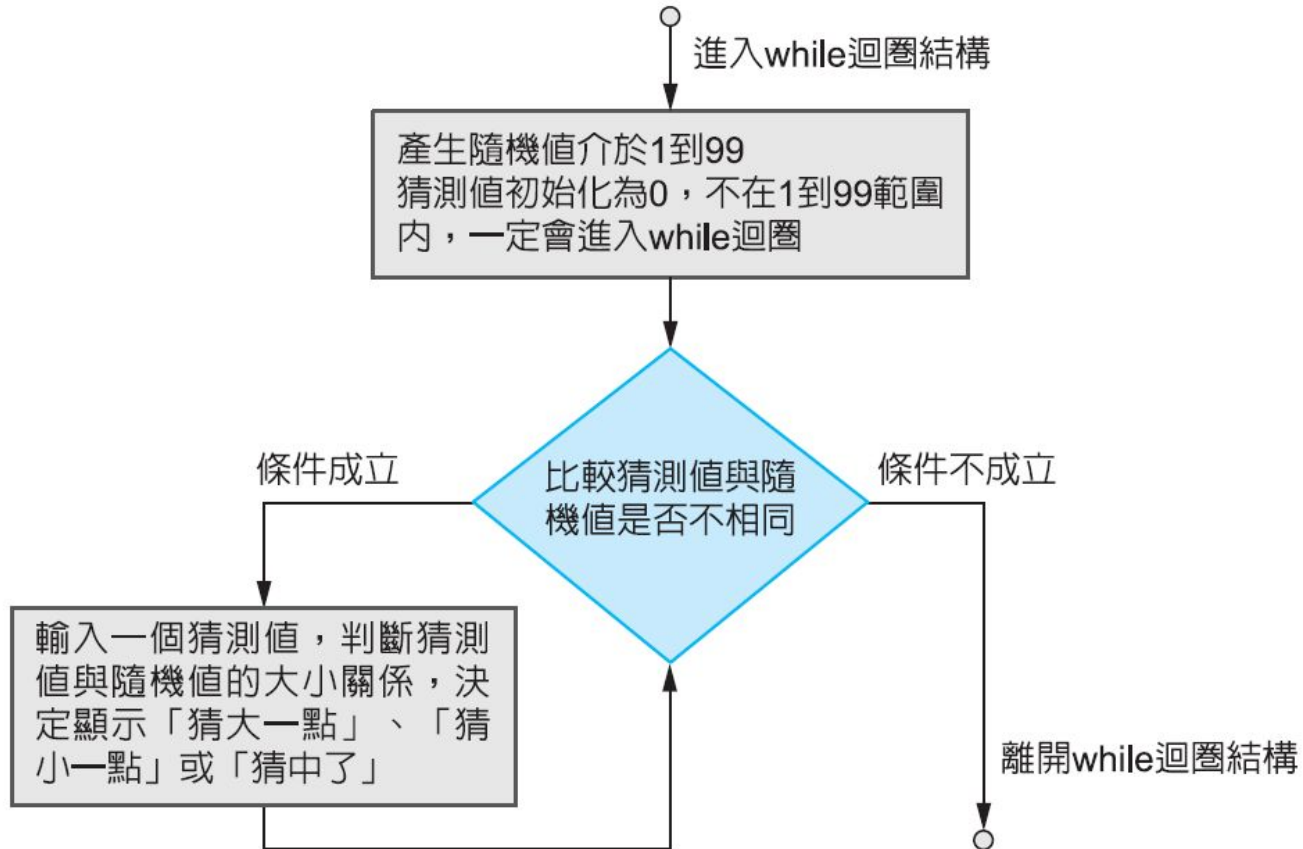


圖 5-6 猜數字流程圖

## 範例5-2-2 猜數字 (ch5\5-2-2-guess.py)

| 行數 | 程式碼                                     | 執行結果  |
|----|---|---|
| 1  | import random                           | <p>不斷輸入數字進行猜測，結果如下。</p> <p>請輸入 1 到 99 的數字？ 50<br/>猜大一點<br/>請輸入 1 到 99 的數字？ 75<br/>猜小一點<br/>請輸入 1 到 99 的數字？ 62<br/>猜小一點<br/>請輸入 1 到 99 的數字？ 57<br/>猜中了</p> |
| 2  | target = random.randint(1,99)           |   |
| 3  | guess = 0                               |   |
| 4  | while target != guess:                  |   |
| 5  | guess = int(input(' 請輸入 1 到 99 的數字？ ')) |   |
| 6  | if target < guess:                      |   |
| 7  | print(' 猜小一點 ')                         |   |
| 8  | elif target > guess:                    |   |
| 9  | print(' 猜大一點 ')                         |   |
| 10 | else:                                   |   |
| 11 | print(' 猜中了 ')                          |   |

---

# Programming codes 3



## 範例5-2-4 質數判斷 (ch5\5-2-4-prime.py)

---

- 某數的因數只有1 與自己, 沒有其他因數, 稱為質數。
- 程式中要判斷一個數字是否是質數, 就要判斷他的因數是否只有1 與自己。

## 範例5-2-4 質數判斷 (ch5\5-2-4-prime.py)

---

### □ 解題想法

#### □ 舉例說明

- 1. 判斷2 是否整除 $n$ , 若是則輸出「 $n$  不是質數」程式結束。
- 2. 判斷3 是否整除 $n$ , 若是則輸出「 $n$  不是質數」程式結束。
- 3. 判斷4 是否整除 $n$ , 若是則輸出「 $n$  不是質數」程式結束。
- 4. 依此方式, 直到判斷 $n-1$  是否整除 $n$ , 若是則輸出「 $n$  不是質數」程式結束。



---

## □ 解題想法

### □ 舉例說明

- 5. 到此, 若2、3、4、...、 $n-1$  皆無法整除 $n$ , 則輸出「 $n$  是質數」程式結束。利用一個旗標變數`prime` 先設定為`True`, 表示先認定該數為質數
- 使用迴圈依序找出從2到 $n-1$  的每個數, 都去判斷2 到 $n-1$  的每個數是否整除 $n$ , 若可以整除, 則旗標變數設定為`False`, 跳出迴圈, 最後根據旗標變數`prime` 決定 $n$  是否為質數。

## 範例5-2-4 質數判斷 (ch5\5-2-4-prime.py)

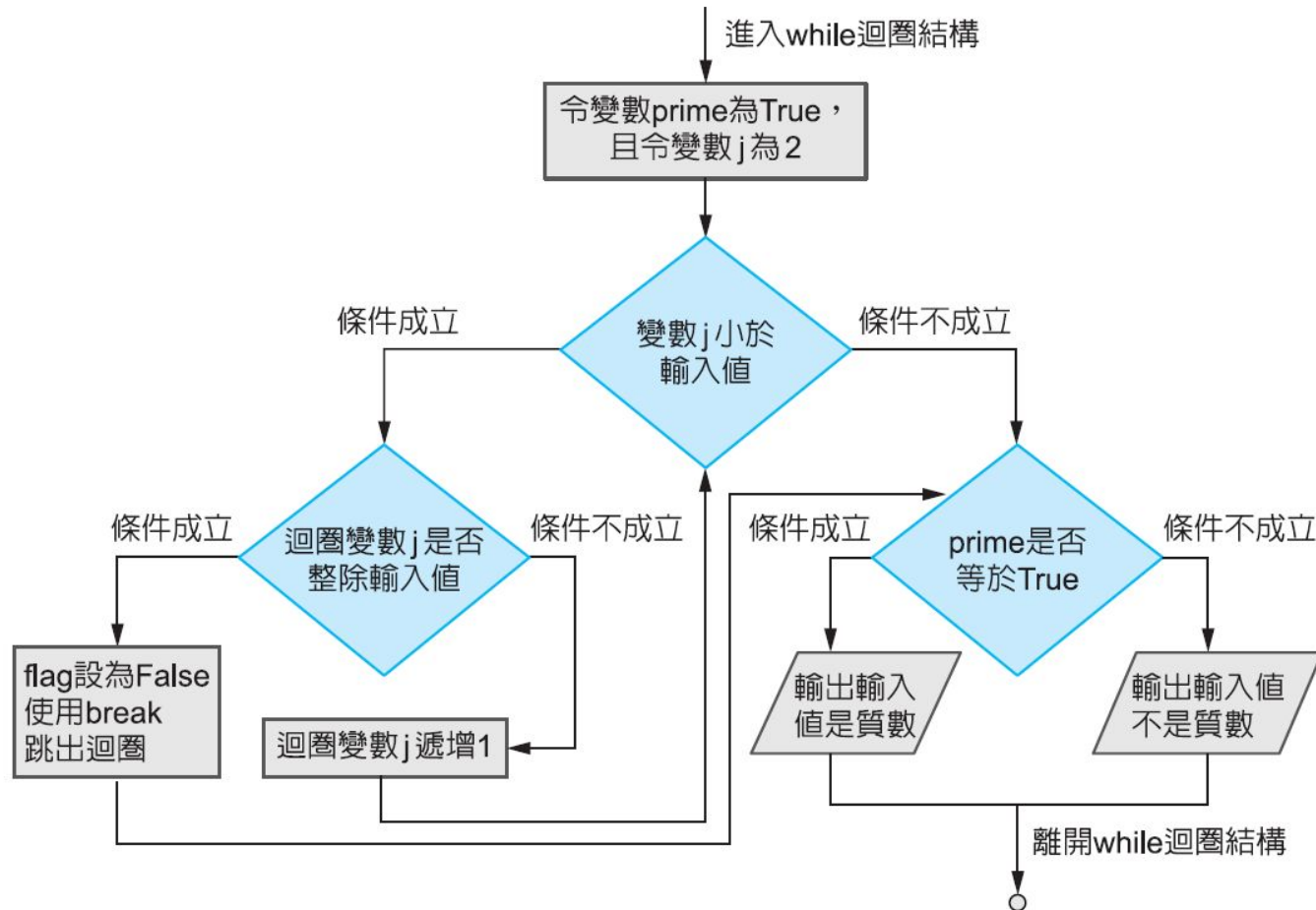


圖 5-8 質數判斷流程圖

## 範例5-2-4 質數判斷 (ch5\5-2-4-prime.py)

| 行數 | 程式碼                            | 執行結果   |
|----|--------------------------------|--|
| 1  | num = int(input(' 請輸入一個整數？ ')) | <p>輸入「37」，結果如下。</p> <p>請輸入一個整數？ 37</p> <p>37 是質數</p> |
| 2  | j = 2                          |  |
| 3  | prime = True                   |  |
| 4  | while j < num:                 |  |
| 5  | if (num%j == 0):               |  |
| 6  | prime = False                  |  |
| 7  | break                          |  |
| 8  | j += 1                         |  |
| 9  | if prime:                      |  |
| 10 | print(num, ' 是質數 ')            |  |
| 11 | else:                          |  |
| 12 | print(num, ' 不是質數 ')           |  |

## 5-3 巢狀迴圈

---

- 巢狀迴圈並不是新的程式結構，只是迴圈範圍內又有迴圈，巢狀迴圈可以有好幾層，巢狀迴圈與單層迴圈運作原理相同，從外層迴圈來看，內層迴圈只是外層迴圈內的動作，因此外層迴圈作用一次，內層迴圈需要執行完畢。
- 以輸出九九乘法表為例，當外層迴圈作用一次，內層迴圈要執行九次，當外層迴圈作用九次，內層迴圈總共執行八十一一次。

## 範例5-3-1 九九乘法表 (ch5\5-3-1-99.py)

---

- 寫一個程式印出九九乘法表。
- 解題想法
  - 巢狀迴圈的外層迴圈使用迴圈變數*i*，內層迴圈使用迴圈變數*j*，外層迴圈*i* 等於1，內層迴圈*j* 由1 變化到9，印出「 $1*1=1$ ,  $1*2=2$ ,  $1*3=3$ , ...,  $1*9=9$ 」，
  - *i* 遞增1，外層迴圈*i* 等於2，內層迴圈*j* 由1 變化到9，印出「 $2*1=2$ ,  $2*2=4$ ,  $2*3=6$ , ...,  $2*9=18$ 」，
  - 依此類推，直到外層迴圈*i* 等於9，內層迴圈*j* 由1 變化到9，印出「 $9*1=9$ ,  $9*2=18$ ,  $9*3=27$ , ...,  $9*9=81$ 」。

## 範例5-3-1 九九乘法表 (ch5\5-3-1-99.py)

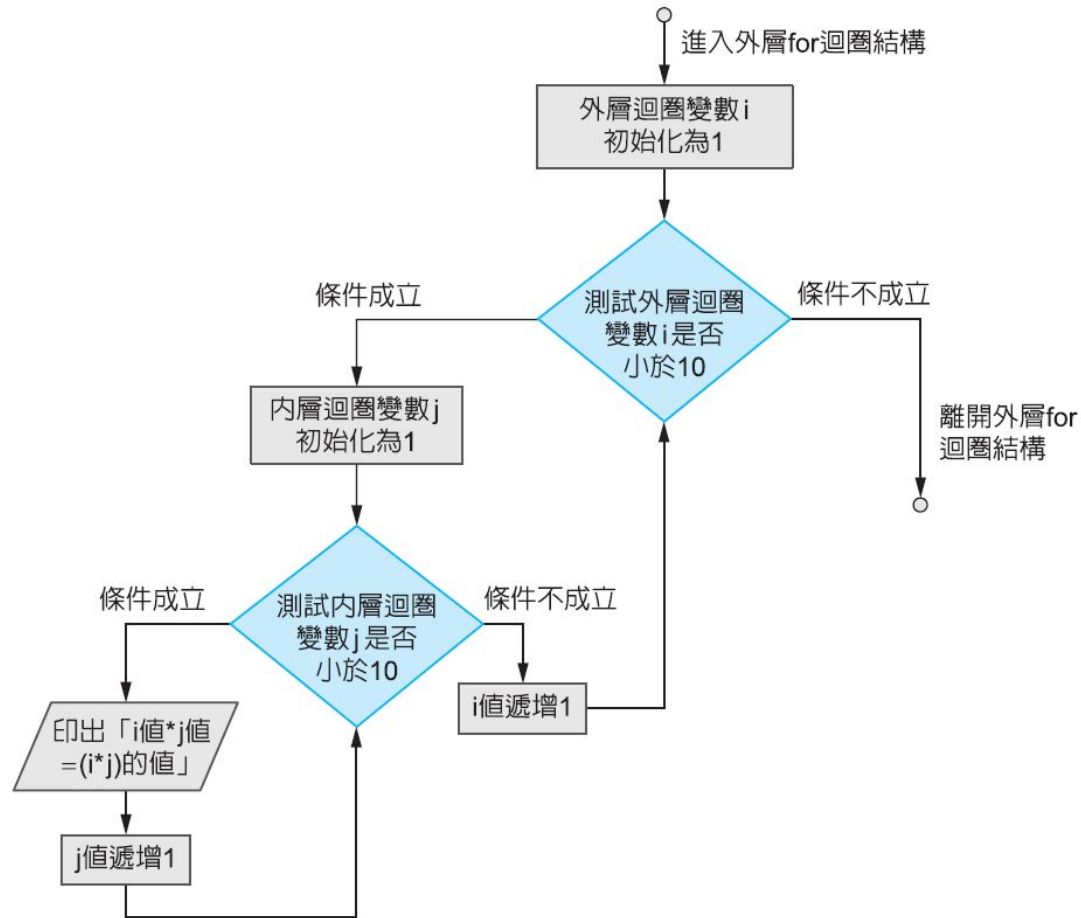


圖 5-9 九九乘法表流程圖

---

# Programming codes 4



## 範例5-3-1 九九乘法表 (ch5\5-3-1-99.py)

```
1 for i in range(1,10):  
2     for j in range(1,10):  
3         print(i, '*', j, '=', i*j, '\t', sep="", end="")  
4     print()
```

### 執行結果

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



---

# Readings



## 範例5-1-1 產生ASCII 碼 (ch5\5-1-1-ascii.py)

---

- 電腦中所有資料皆以二進位方式儲存，大小寫英文字母、數字都有國際標準的二進位編碼，這樣的編碼稱為ASCII 碼

---

□ 例如：

- A 的ASCII 碼以二進位表示為01000001，十進位表示為65
- B 的ASCII 碼以二進位表示為01000010，十進位表示為66
- C 的ASCII 碼以二進位表示為01000011，十進位表示為67，依此類推。
- 請寫一個程式利用迴圈與「chr」函式，「chr」函式將整數轉換成對應的ASCII 字元。

## 範例5-1-1 產生ASCII 碼

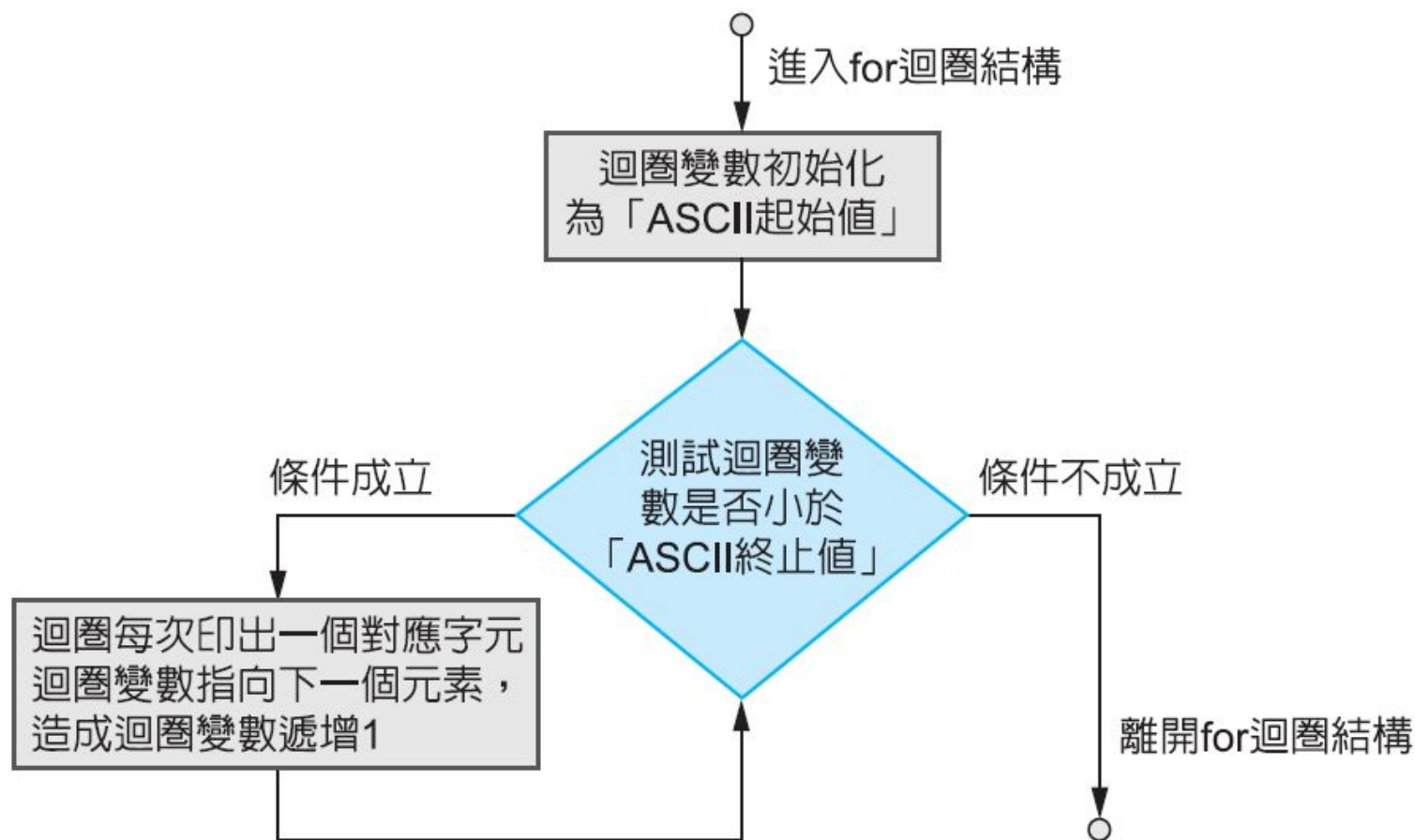


圖 5-2 產生 ASCII 碼流程圖

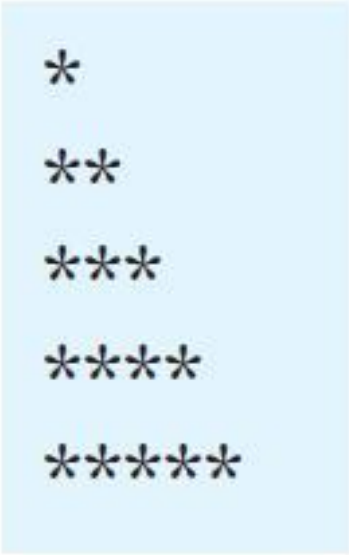
## 範例5-1-1 產生ASCII 碼 (ch5\5-1-1-ascii.py)

| 行數 | 程式碼                                 | 執行結果   |
|----|-------------------------------------|--|
| 1  | s = int(input(' 請輸入 ASCII 的起始值？ ')) | ASCII 起始值輸入「65」，ASCII 終止值輸入「70」，本程式就會顯示 ASCII 介於 65 到 69 的字元。<br><br>請輸入 ASCII 的起始值？ 65<br>請輸入 ASCII 的終止值？ 70<br>A<br>B<br>C<br>D<br>E |
| 2  | e = int(input(' 請輸入 ASCII 的終止值？ ')) |  |
| 3  | for i in range(s, e):               |  |
| 4  | print(chr(i))                       |  |

## 範例5-3-2 印星號 (ch5\5-3-2-star.py)

---

- 請使用巢狀迴圈印出如右圖所示的星號，第一行一個星號，第二行兩個星號，…，第五行五個星號，全部靠左排列。



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

## 範例5-3-2 印星號(ch5\5-3-2-star.py)

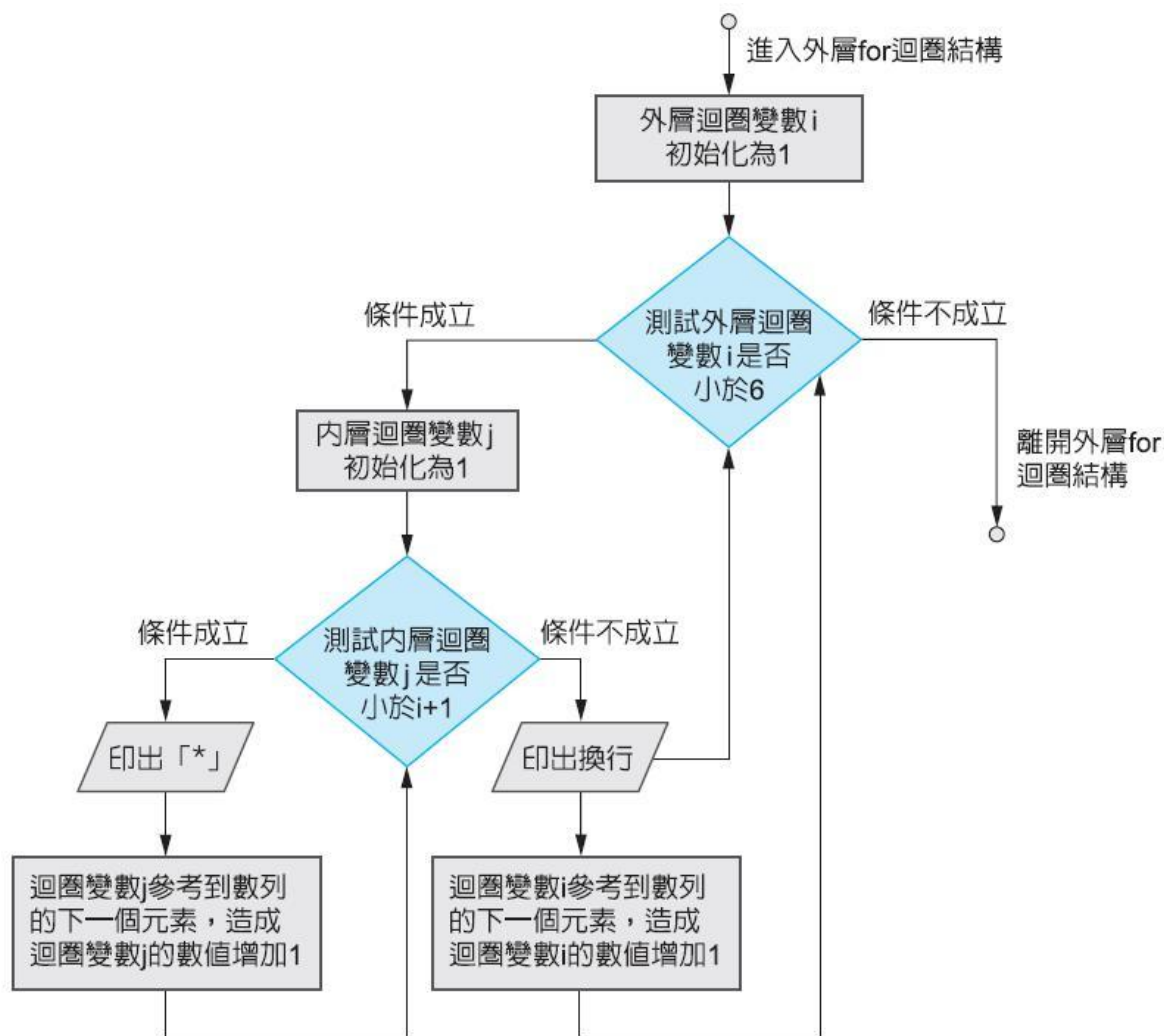


圖 5-10 印星號流程圖

# 範例5-3-2 印星號

## (ch5\5-3-2-star.py)

### 程式碼



WordPad  
Document

| 行數 | 程式碼                    | 執行結果   |
|----|------------------------|--|
| 1  | for i in range(1,6):   | <pre>*<br/>**<br/>***<br/>****<br/>*****</pre> |
| 2  | for j in range(1,i+1): |  |
| 3  | print("*", end="")     |  |
| 4  | print()                |  |



---

# □ Exercise



# 學號重複字元個數計算

---

- 請計算您的學號中各字元重複出現次數, 若學號非 9 位數, 顯示「學號錯誤」訊息
- 用 for 迴圈掃描學號的每個字元, 並將各字元出現次數結果儲存於 dictionary 資料型態變數中(資料如下), 最終使用 for 迴圈顯示 dictionary 的儲存結果

key value

```
dictionary = {'學號包含的某個字元': 該字元出現次數,  
              ...}
```

# 學號重複字元個數計算

---

```
1 student_ID = '109310001'  
2 dic = {}  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16
```

Exercise I  
(2 pt)

# 學號重複字元個數計算

□ 執行成果：

1. student\_ID 長度不為 9 碼：

```
您的學號是： 1093100012  
學號錯誤！
```

```
您的學號是： 10931001  
學號錯誤！
```

2. student\_ID 為您的真實學號：

```
您的學號是： 109310001  
輸出結果：  
1 共出現 3 次  
0 共出現 4 次  
9 共出現 1 次  
3 共出現 1 次
```

3. student\_ID 前 3 碼為您的入學學年 +1：

```
您的學號是： 110310001  
輸出結果：  
1 共出現 4 次  
0 共出現 4 次  
3 共出現 1 次
```

## 範例5-2-3 複利計算 (ch5\5-2-3-ins.py)

---

- 請寫一個程式，計算存一筆錢(100000) 在銀行，以複利計算，年利率為(X)，計算最少需要幾年才能超過目標金額(你的學號)。

$$\text{複利計算公式： 本金} \times (1 + \text{利率})^{(\text{總期數})} = \text{本利和 (終值)}$$

## 範例5-2-3 複利計算 (ch5\5-2-3-ins.py)

---

### □ 解題想法

- 使用while 迴圈進行複利計算，當還未超過目標金額(你的學號) 時，繼續執行複利計算，當超過目標金額(你的學號) 時，中止while 迴圈。

## 範例5-2-3 複利計算 (ch5\5-2-3-ins.py)

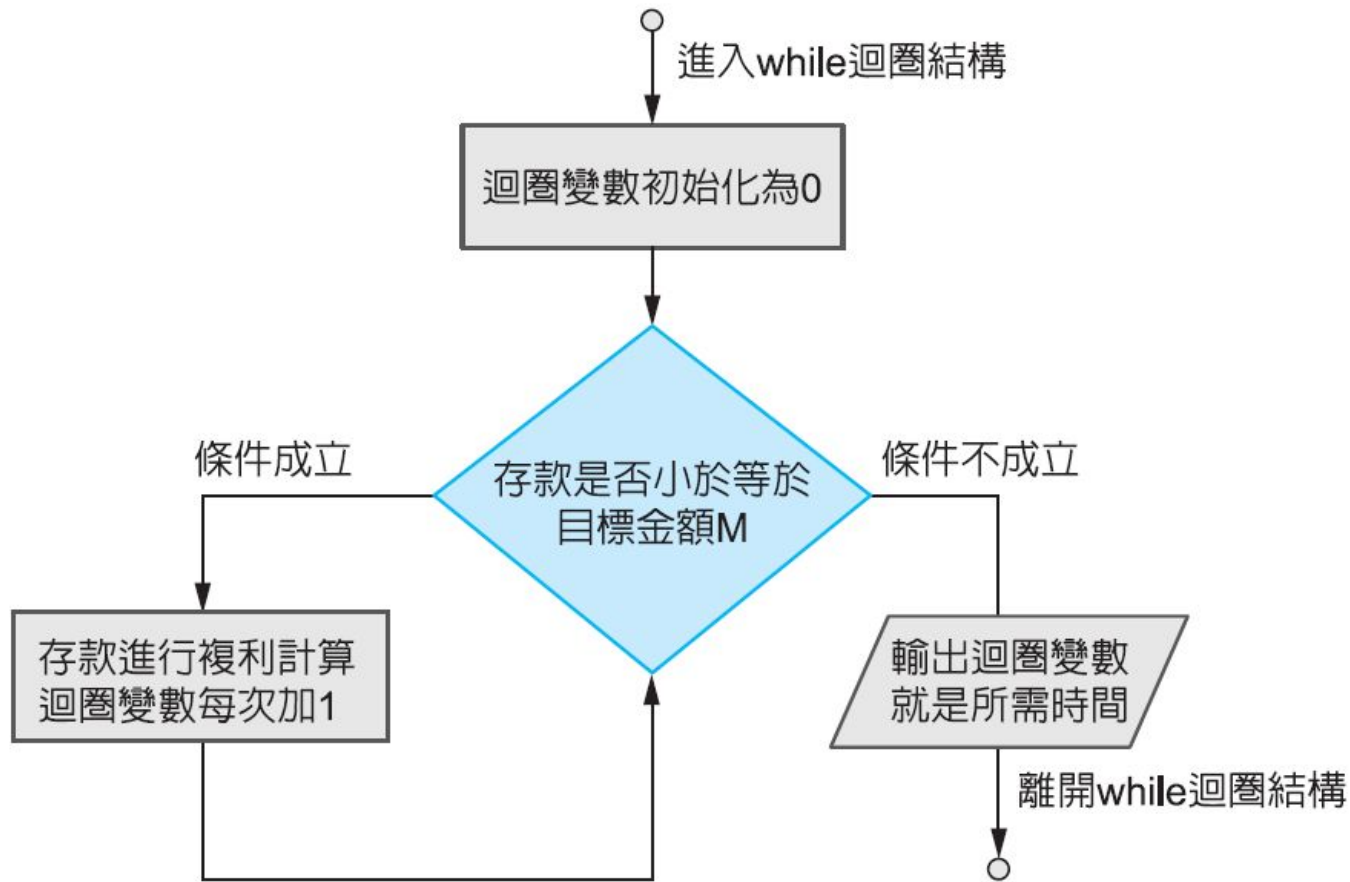


圖 5-7 複利計算流程圖

## 範例5-2-3 複利計算 (ch5\5-2-3-ins.py)

---

```
存款金額 = 100000
```

### Exercise I (2 pt)



# 執行結果:

---

## 1. 年利率20

20  
請輸入年利率 (按 'Enter' 鍵確認或按 'Esc' 鍵取消)

✓ 存款金額 = 100000 ...  
第 39 年後本利和為 122480963.99742368 超過 107650014

## 2. 年利率50

50  
請輸入年利率 (按 'Enter' 鍵確認或按 'Esc' 鍵取消)

✓ 存款金額 = 100000 ...  
第 18 年後本利和為 147789188.00354004 超過 107650014

## 3. 年利率80

80  
請輸入年利率 (按 'Enter' 鍵確認或按 'Esc' 鍵取消)

✓ 存款金額 = 100000 ...  
第 12 年後本利和為 115683138.14261763 超過 107650014

# To be continued.....

Instructor: Cheng-Chun Chang (張正春)  
Department of Electrical Engineering