1.1 認識 Python 語言

1.1.1 程式語言簡介

使用電腦就一定會用到程式，程式是使用程式語言撰寫的。平常使用的軟體，如作業系統、瀏覽器、文書處理、試算表、簡報、防毒、手機應用程式等，都是使用程式語言設計的。

程式語言是人和電腦溝通的工具，就像人與人溝通，需要用英語、華語等，人如果要和電腦溝通，就要使用程式語言。程式設計是使用程式語言，將解決問題的方法設計成電腦能執行的程式（圖 1-1）。

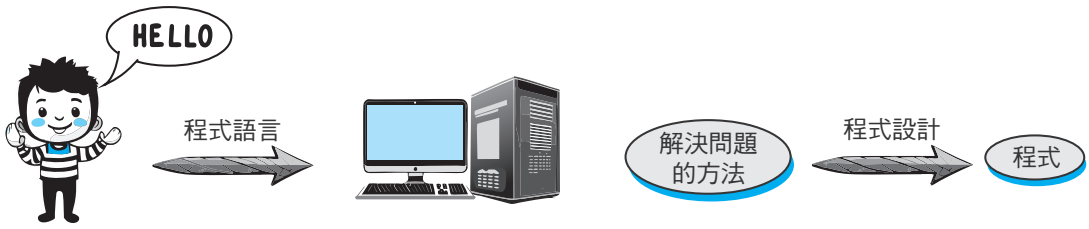


圖 1-1 程式語言與程式設計

電腦硬體只認識機器語言（machine code），機器語言是一長串 0 與 1 的組合，例如：圖 1-2 左是某一電腦的機器語言程式，用來計算兩數之和。

由於機器語言程式不易撰寫，也不易閱讀。為了使程式設計更簡單，程式更容易了解與維護，於是有高階語言（high level language）的發展（圖 1-2 右）。

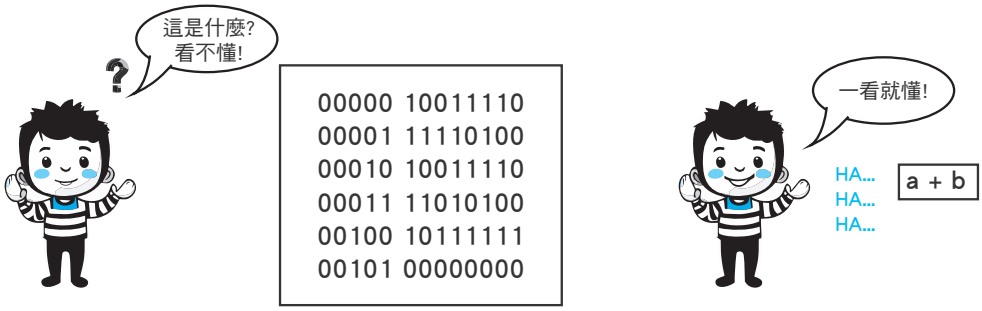


圖 1-2 機器語言與高階語言

高階語言比較接近人類的語言，但它不能直接執行，需轉譯成機器語言後，電腦才能執行。機器語言程式可以直接執行，不需轉譯，所以執行速度較高階語言快。

高階語言有非常多種，如 C、C++、C#、Python、Java、JavaScript、PHP、R、Ruby 等，每種程式語言各有不同的適用場域。雖然各種語言的語法不同，但程式設計的邏輯相近，學會其中一種，很容易轉換到其他種語言。

1.1.2 Python 簡介

Python 是由荷蘭資訊科學家 Guido van Rossum（吉多・范羅蘇姆）創始的，他認為程式設計應該是容易學習的，程式碼應該容易了解的，因此創建了一種新的程式語言，讓設計者更容易設計程式，也容易擴展既有的程式，後來他將此程式語言以他喜愛的喜劇團 Python 為名。

Python 1.0 在 1991 年推出，2008 年更新到 3.0，目前最新版是 3.xx 版。Python 強調程式碼的可讀性和簡潔的語法，使用空格縮排來劃分程式碼塊，支援結構化、程序式、物件導向等程式設計，目前已成為熱門且重要的程式語言。

Python 廣泛應用於大數據、資料科學、人工智慧、網站開發、遊戲開發、電腦輔助設計等領域（圖 1-3）。學會 Python 程式設計，可提升許多專業領域解決問題的能力，如金融、商管、機械、設計、建築、醫藥、生物科技等，對未來生涯有很大幫助。

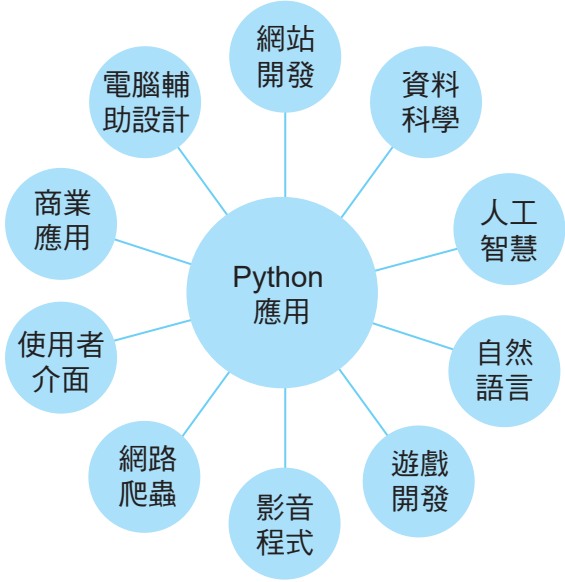


圖 1-3 Python 的應用領域

重複結構

本章學習重點

- for 迴圈
- for 雙重迴圈
- while 迴圈
- 改變迴圈的執行
- APCS 實作題

本章學習範例

- 範例 4.1-1 數列和 1 加到 n
- 範例 4.1-2 個位數是 7 或 7 的倍數之和
- 範例 4.1-3 偶數和
- 範例 4.1-4 交錯調和級數
- 範例 4.1-5 計算複利
- 範例 4.2-1 九九乘法表
- 範例 4.2-2 星號三角形
- 範例 4.2-3 畢氏三元數
- 範例 4.3-1 數字倒轉（a038）
- 範例 4.3-2 位數和
- 範例 4.3-3 找出所有因數
- 範例 4.3-4 最大公因數（a024）
- 範例 4.4-1 猜數字遊戲
- 範例 4.4-2 質數判斷（a007）
- 範例 4.5-1 人力分配（201710 APCS 第 1 題）

4.1 for 迴圈

4.1.1 認識迴圈

程式往往會需要重複執行某些步驟，例如：要輸出 100 行字串 'Hello Python!'。若使用下面的程式碼，會非常不方便。

```
print('Hello Python!')
.............
.............
print('Hello Python!')
```

100 行

程式內重複性的步驟，可使用重複結構來完成。重複結構是指重複執行某些敘述，直到滿足特定條件為止，也稱為迴圈（loop）。

Python 的重複結構有 for 和 while 兩種（圖 4-1），這兩種迴圈可以互相轉換。for 迴圈會在一定的範圍內重複執行迴圈，所以又稱計數迴圈，常用於有固定次數的迴圈。

while 迴圈則會根據條件判斷的結果，決定是否要繼續執行迴圈，所以又稱條件迴圈，常用於不固定次數的迴圈。

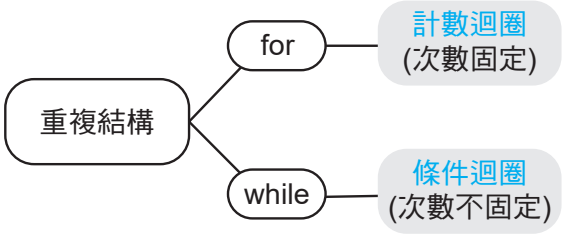


圖 4-1 重複結構的種類

4.1.2 認識 range() 函式

使用 for 迴圈前，先來認識 range() 函式。range 的意思是範圍，range() 函式可以建立一個整數序列，讓程式依照序列裡的整數，來執行迴圈裡的敘述，其語法如下：

range([開始值 ,] 結束值 [, 間隔值]）

0 可省略　不含結束值　1 可省略

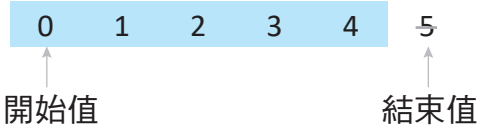
range(開始值 , 結束值 , 間隔值 ）是指建立一個整數序列，範圍是從「開始值」到「結束值」，每次隔一個「間隔值」。注意，範圍包含開始值，但並不包含結束值，也就是「顧頭不顧尾」。

range() 的開始值 0 或間隔值 1 可省略。以下以例子說明其使用方法：

1. range(5)

建立的序列是 0, 1, 2, 3, 4，共 5 個，不包含 5。

等同 range(0, 5, 1)，開始值 0 和間隔值 1 可省略。序列的範圍從 0 開始到 5 結束，每次增加 1。



2. range(n)

建立的序列有 n 個整數，範圍是 0 ~ n - 1，不包含 n。

3. range(2, 5)

建立的序列是 2, 3, 4，共 3 (5 - 2) 個，不包含 5。

序列的範圍從 2 開始到 5 結束，每次間隔 1，不包含 5。

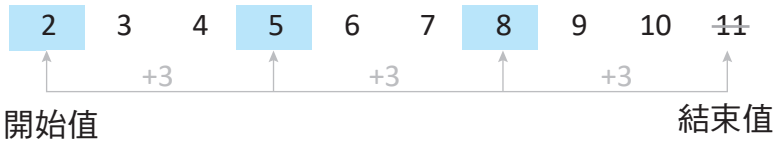
4. range(m, n)

建立的序列有 n - m 個整數（m < n），範圍是 m ~ n - 1，不包含 n。

5. range(2, 11, 3)

建立的序列是 2, 5, 8，共 3 個，不包含 11。

序列範圍從 2 開始到 11 結束，每次增加 3，不包含 11。



6. range(5, 2)

序列的範圍從 5 開始到 2 結束，每次增加 1，很明顯此序列不包含任何整數。

7. range(5, 2, -1)

序列的範圍從 5 開始到 2 結束，不包含 2，每次 -1，因此是 5, 4, 3，共 3 個。

4.1.3 設計 for 迴圈

for 迴圈的語法如下，其中「可迭代物件（iterable object）」是指可使用重複的方式，一次傳回物件內的一個元素。

所以 for 的第 1 次迴圈會取出物件的第 1 個元素，將它指定給變數，第 2 次迴圈取出第 2 個，再將它指定給變數，依此類推，依序取出全部元素。

```
for 變數 in 可迭代物件：        # range()、字串、串列都是可迭代物件
    敘述
```

可使用 for 迴圈遍歷的物件，都是可迭代物件，如 range()、字串、串列等，都可使用迴圈取出每個元素。

for 迴圈常用 range() 來控制迴圈的執行，語法與流程圖如下（圖 4-2），注意，for 的行末要加冒號 :，表示迴圈內的敘述要縮排。

```
for 變數 in range(開始值, 結束值, 間隔值):
    敘述
```

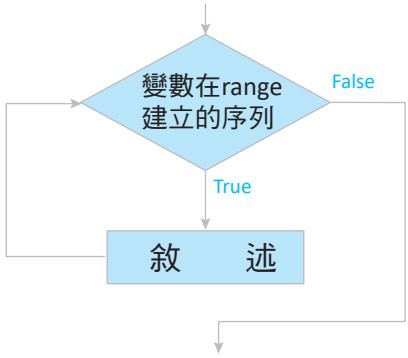


圖 4-2 for 迴圈的語法與流程圖

在 for 迴圈中，range() 函式會建立一個整數序列，程式會依序將序列裡的整數，指定給變數後，再執行迴圈裡的敘述。

此語法可以看成是「變數在範圍內，從開始值到結束值，每隔一個間隔值，執行迴圈。」以下列舉幾個例子說明：

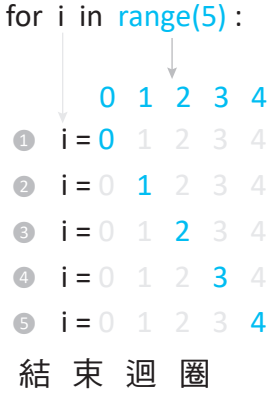
例題 1

如以下程式，range(5) 的序列是 0, 1, 2, 3, 4，所以 i 會在（in）0, 1, 2, 3, 4 重複執行 print(i)。因此會輸出 0, 1, 2, 3, 4 共 5 列 i 值，輸出結果如下：

```
for i in range(5):                  # i 依序是 0,1,2,3,4
    print(i)
```

```
0
1
2
3
4
```

for 迴圈執行的過程如下：



❶ 取序列第 1 個數 0，所以 i = 0，執行迴圈內的 print(i)，輸出 0 後換行。

❷ 取序列第 2 個數 1，所以 i = 1，執行迴圈內的 print(i)，輸出 1 後換行。

……

❺ 取序列第 5 個數 4，所以 i = 4，執行迴圈內的 print(i)，輸出 4 後換行。

依此類推，「for i in range(n):」表示迴圈會重複 n 次，i = 0 ~ n - 1，但不包含 n。

例題 2

如以下程式，range(2, 5) 的序列是 2, 3, 4，所以會在 i = 2, 3, 4 重複執行 print(i)，因此會輸出 2, 3, 4 共 3 列 i 值，輸出結果如下：

```
for i in range(2, 5):                #i 的範圍是 2 ~ 4，共 3 個，不包含 5
    print(i)
```

```
2
3
4
```

例題 3

如以下程式，range(2, 11, 3) 的序列是 2, 5, 8，所以會在 i = 2, 5, 8 重複重複執行 print(i)，因此會輸出 2, 5, 8 共 3 列 i 值，輸出結果如下：

```
for i in range(2, 11, 3):            #i 的範圍是 2,5,8，不包含 11
    print(i)
```

```
2
5
8
```

例題 4

如以下程式，range(5, 2) 的序列裡沒有任何一個整數，因此不會輸出任何 i 值。

```
for i in range(5, 2):                # 此範圍不存在任何一個整數
    print(i)
```

範例 4.1-4 交錯調和級數

寫一程式，輸入整數 n，計算並輸出$1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}\cdots\cdots\cdots\frac{1}{n}$之值，取至小數第 6 位。

範例一：輸入	範例二：輸入
2	5
範例一：正確輸出	範例二：正確輸出
0.5	0.783333

解題方法

1. 本題的解題方法和計算 1 加到 n 類似，不同的是整數項變成分數項，且數列每一項的 + - 號會交錯。
2. 使用 for 迴圈累加，如果每一項是 1 / (i + 1)，i 的範圍會是 range(0, n)，等同 range(n)。
3. 要讓數列每一項的 + - 號交錯出現，可將每一項的值 1 / (i + 1) 乘上 (-1) ** i。因為

   i = 0，(-1) ** 0 = 1，會乘 1。

   i = 1，(-1) ** 1 = -1，會乘 -1。

   依此類推，i = 2 乘 1，i = 3 乘 -1，……
4. 使用 total 存放累加值，初始值為 0，每次迴圈 total 累加

   1 / (i + 1) * (-1) ** i。
5. 要輸出 total 至小數第 6 位，可使用 f 字串，格式為 .6f。
6. 解題演算法可設計如下：

   輸入整數 n

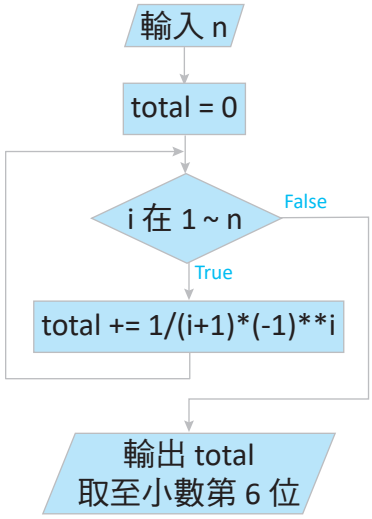
   累加的值 total 初始值為 0

for i in range(n):

將 1 / (i + 1) * (-1) ** i 累加到 total

輸出 total，取至小數第 6 位

7. 解題流程圖如下：



程式設計

```
1 n = int(input())                          # 讀取輸入，轉成整數 n
2 total = 0                                 # 用 total 存放級數的和，初始值為 0
3 for i in range(n):                        # 執行迴圈，i 從 0~n-1
4     total = total + 1/(i+1)*(-1)**i       # 將 1,-1/2,1/3,…累加到 total
5 print(f'{total:.6f}')                     #:.6f 表示小數點 6 位的浮點數
```

執行結果

4	20
0.583333	0.668771

說明

若要將 + - 號放在每一項前面，因為運算的優先順序，1 / (i + 1) 需要加上括號，寫成敘述 (-1) ** i * (1 / (i + 1))

範例 4.1-5 計算複利

複利是指計算利息時，除了根據本金計算外，新得的利息也會納入下一期的本金中。例如：本金 1000 元，年利率 1.5%，第 1 年結束本金加利息（本利和）共 1000 * (1 + 0.015) = 1015 元，第 2 年變成 1015 * (1 + 0.015) 元，依此類推。寫一程式，能以複利計算期滿後的本利和。

輸入：本金（整數）、年利率（浮點數）%、年數（整數），三者用空白隔開。

輸出：本利和，取至整數。

範例一：輸入	範例一：正確輸出
500000 1.6 5	541301

解題方法

1. 思考計算本利和的方法，第 1 年的本利和就是第 2 年的本金；第 2 年的本利和，就是第 3 年的本金，以此類推。若本金 p、年利率 r、年數 y

   第 1 年本利和 = p * (1 + r)

   第 2 年本利和 = 第 1 年本利和 * (1 + r) = p * $(1 + r)^2$

   第 3 年本利和 = 第 2 年本利和 * (1 + r) = p * $(1 + r)^3$

   ……

   第 y 年本利和 = 第 y - 1 年本利和 * (1 + r) = p * $(1 + r)^y$

2. 本題可使用 for 迴圈解題。年數是 y，所以範圍是 range(y + 1)，迴圈內的敘述如下：

   這一年的本利和 = 前一年的本利和 * (1 + r)

   也就是 p = p * (1 + r)

3. 因為年利率是使用 % 計算，所以讀入的年利率 r 在計算本利和時，需先除以 100。

4. 解題演算法可設計如下：

   輸入本金 p、年利率 r、年數 y

   for _ in range(y + 1):

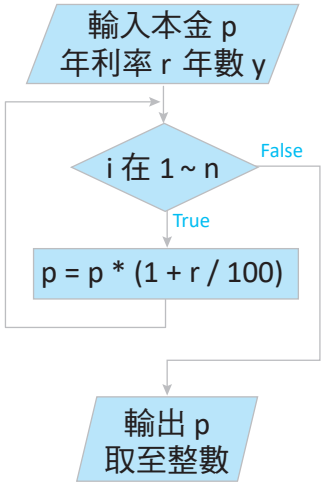
   　　本利和 p = 前一年的本利和 p * (1 + r)

   輸出本利和 p，取至整數

5. 讀取輸入的本金 p、年利率 r、年數 y 時，因為資料可能包含整數或浮點數，所以不能使用 int()，需改用 eval()，也就是

   p, r, y = map(eval, input().split())

6. 解題流程圖如下：



程式設計

```
1 p,r,y = map(eval,input().split())# 讀取輸入，轉成數值，指定給 p,r,y
2 for _ in range(y):                # 迴圈共執行 y 次
3     p = p * (1 + r / 100)           # 新的本利和 = 舊的本利和 *(1+ 利率 %)
4 print(f'{p:.0f}')
```

執行結果

```
10000 1.5 10

11605
```

4.2 for 雙重迴圈

4.2.1 認識雙重迴圈

for 雙重迴圈是 for 外層迴圈內還有一層 for 內迴圈（圖 4-3）；而 for 三重迴圈則是 for 外迴圈內有 for 中迴圈，中迴圈內還有 for 內迴圈。

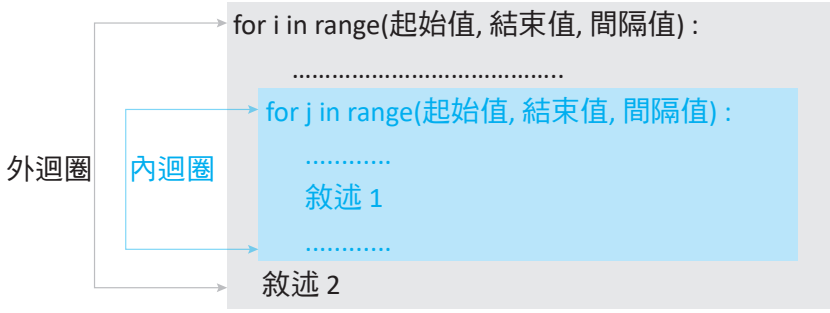
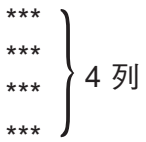


圖 4-3 for 雙重迴圈的語法與結構

雙重迴圈內的敘述要留意縮排，因為縮排會決定它是外迴圈或內迴圈的敘述，上圖中，敘述 2 縮排在外迴圈 for i 的區塊內，與內迴圈 for j 敘述開頭對齊，所以是外迴圈內的敘述。敘述 1 則是縮排在內迴圈 for j 的區塊內，所以是內迴圈內的敘述。

4.2.2 設計雙重迴圈

設計雙重迴圈的程式時，可先設計內迴圈，再設計外迴圈，以下以輸出下圖的 * 號矩形為例，說明雙重迴圈的設計方法。



先設計輸出 1 列 *** 的程式，再設計重複輸出 4 列 *** 的程式。

1. 輸出 1 列 ***

- 1 列有 3 個 *，所以迴圈範圍為 range(3)。
- 輸出 * 號可用 print('*')。print() 預設會在輸出的結果後加上一個隱藏的換行字串 '\n'，下一個 print() 從下一行開始輸出。

  若要 print() 不換行輸出，可在 () 內加入 end = ''。end 是結束，表示以空字串 '' 結束，不是以換行結束，也就是 print('*', end = '')。兩者輸出的差異如下：

```
print('*')
print('*')
```

```
*
*
```

```
print('*',end='')
print('*')
```

```
**
```

- 因此「輸出 1 列 ***」的程式可設計為：

```
for j in range(3):                  # 有 3 個 *，所以迴圈範圍為 range(3)
    print('*', end = '')            # 將輸出以 '' 結束，不是以換行結束
```

2. 重複輸出 4 列 ***

- 已設計出輸出 1 列 *** 的程式，要輸出 4 列，只要再用另 1 個 for 迴圈，讓輸出 1 列 *** 的程式重複執行 4 次即可，因此另一個迴圈的範圍為 range(4)。
- 每輸出完 1 列 *** 後，應換行輸出，讓下 1 列 ***，從下 1 行的開頭開始輸出，所以在外迴圈內，內迴圈結束後，應有 1 行輸出換行的 print() 敘述。
- 最後程式設計如下：

```
for i in range(4):                  # 外迴圈，共執行 4 次
    for j in range(3):                  # 內迴圈，可輸出 1 列 3 個 *
        print('*', end = '')
    print()                             # 換行輸出
```

上例的雙重迴圈中，i, j 的執行順序如下（圖 4-4）：

1. 外迴圈以 i 為計數器，從 0 ~ 3，執行 4 次。
2. 外迴圈每執行 1 次，內迴圈 j 會從 0 ~ 2 執行 3 次。

   (1) i = 0 時，j 從 0 ~ 2 執行 3 次。

   (2) i = 1 時，j 從 0 ~ 2 執行 3 次。

   (3) i = 2 時，j 從 0 ~ 2 執行 3 次。

   (4) i = 3 時，j 從 0 ~ 2 執行 3 次後，結束迴圈。

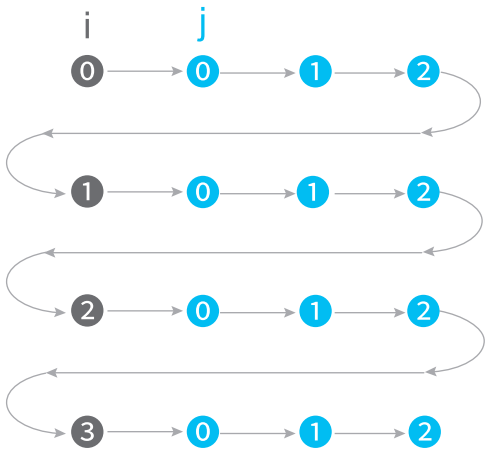


圖 4-4 雙重迴圈 i, j 的執行順序

範例 4.2-1 九九乘法表

使用 for 迴圈，印出九九乘法表，請使用定位符號 \t 對齊各項。

解題方法

1. 九九乘法表可使用雙重迴圈來設計，可先設計內迴圈，再設計外迴圈。
2. 設計內迴圈。分析下方的第 1 列，可發現每項的第 1 個數依序為 2, 3,… 9，所以可使用 for 迴圈，變數 j 在 range(2, 10) 執行迴圈。

2 × 1 = 2　3 × 1 = 3　4 × 1 = 4　……　8 × 1 = 8　9 × 1 = 9

3. 設計外迴圈。九九乘法表共 9 列，觀察第 1 行，可發現每項的第 2 個數依序為 1, 2, … 9，所以使用另一個 for 迴圈，變數 i 在 range(1, 10) 執行迴圈。

```
2 × 1 = 2
2 × 2 = 4
2 × 3 = 6
```

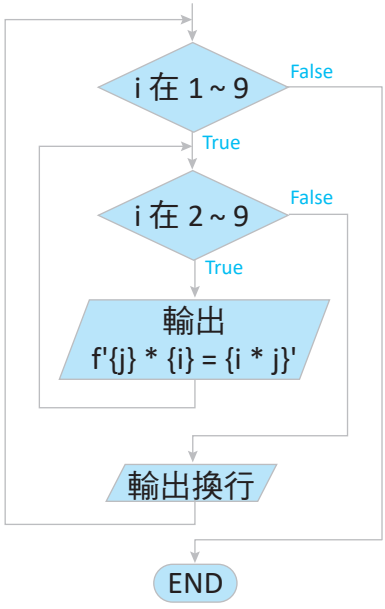
4. 使用 f 字串輸出每一項，以 3 × 5 = 15 為例，3、5、15 都是變數，所以將它們所代表的變數名稱，放在 {} 裡，print() 敘述可設計如下：

```
3 × 5 = 15
print( f'{j} × {i} = {i * j}', end = '\t' )
                              定位符號
```

5. print() 裡的參數 end = '\t'，是要用定位符號 \t 來取代預設的換行，這樣輸出的每一項後面會加上一個定位符號，就會輸出在固定的位置，可用來對齊輸出的各項。

6. 每輸出完 1 列後，應換行輸出，讓下 1 列從下 1 行的開頭開始輸出，所以在外迴圈內，內迴圈結束後，應有 1 行輸出換行的 print() 敘述。

7. 解題流程圖如下：



程式設計

```
1 for i in range(1, 10):                              # 外迴圈 i=1~9
2     for j in range(2, 10):                              # 內迴圈 j=2~9
3         print(f'{j}*{i}={i*j}', end='\t')                    # 輸出一項
4     print()                                           # 輸出 1 列後，換行輸出
```

執行結果

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

說明

程式中，外迴圈的控制變數 i 從 1 執行到 9，外迴圈 i 每執行一次，內迴圈的控制變數 j 會從 2 到 9 執行 8 次。for 雙重迴圈是執行一個 i，就執行 j = 2 ~ 9，再換下一個 i，再執行 j = 2 ~ 9。執行順序如下：

範例 4.2-2 星號三角形

寫一程式，輸入整數 n 後，使用雙重 for 迴圈，依序輸出下列幾種 n 列星號三角形。例如：n = 5 時，輸出的星號三角形如下：

①
```
*
* *
* * *
* * * *
* * * * *
```

②
```
* * * * *
* * * *
* * *
* *
*
```

③
```
    * * * * *
      * * * *
        * * *
          * *
            *
```

④
```
            *
          * * *
        * * * * *
      * * * * * * *
    * * * * * * * * *
```

解題方法

先以 n = 5 為例，說明設計的方法。每個三角形有 5 列，所以外迴圈 i 的範圍為 range(5)。

1. 第 ① 個星號三角形

(1) i = 0，輸出 1 個 *，j = 0。

i = 1，輸出 2 個 *，j = 0, 1。

i = 2，輸出 3 個 *，j = 0, 1, 2。依此類推。

(2) 外迴圈 i 時，內迴圈執行 i + 1 次，所以內迴圈 j 的範圍在 range(i + 1)。

2. 第 ② 個星號三角形

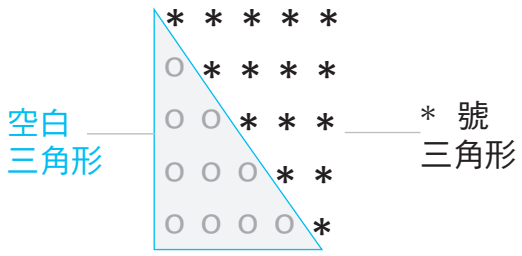
(1) i = 0，輸出 5 個 *，j = 0, 1, 2, 3, 4。

i = 1，輸出 4 個 *，j = 0, 1, 2, 3。

i = 2，輸出 3 個 *，j = 0, 1, 2。依此類推。

(2) 外迴圈 i 時，內迴圈執行 5 - i 次，所以內迴圈 j 的範圍在 range(5 - i)。

3. 第 ③ 個星號三角形



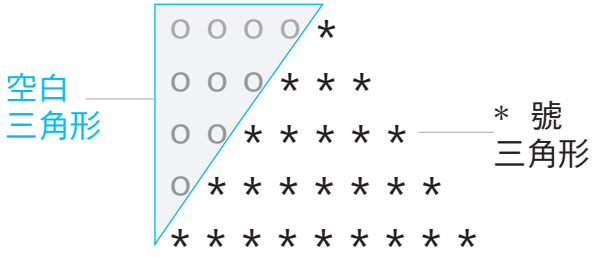
(1) 每一列都先輸出空格，再輸出 *，所以有 2 個內迴圈，第 1 個輸出空格，第 2 個輸出 *。

(2) i = 0，輸出 0 個空格 5 個 *。i = 1，輸出 1 個空格 4 個 *。依此類推。

(3) 第 1 個內迴圈輸出空格，外迴圈 i 時，內迴圈執行 i 次，所以 j 在 range(i + 1)。

第 2 個內迴圈輸出 *，外迴圈 i 時，內迴圈執行 5 - i 次，所以內迴圈 j 在 range(5 - i)。

4. 第 ④ 個星號三角形



(1) 每列都先輸出空格，再輸出 *。

(2) i = 0，輸出 4 個空格 1 個 *。

i = 1，輸出 3 個空格 3 個 *。

i = 2，輸出 2 個空格 5 個 *。

4.3 while 迴圈

已知迴圈數的問題可使用 for 解題，無法預知迴圈數的問題則可使用 while 解題。while 在執行迴圈前，會先檢查條件式是否成立，所以又稱為條件式迴圈。

while 迴圈的語法與流程圖如下（圖 4-5）。若條件式為 False，則不執行迴圈，直接跳到迴圈外的下一個敘述；若條件式為 True，則執行迴圈內的敘述，執行完後，再檢查條件式是否成立，如此不斷重複，直到條件式不成立。

while 迴圈先判斷條件式，可能條件式一開始就不成立，所以迴圈執行次數可以是 0 次。

```
while 條件式：
    敘述
    ……
```

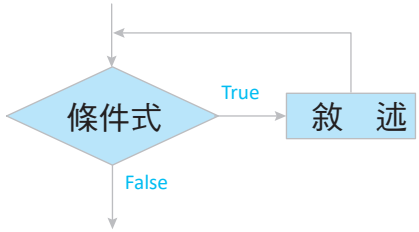


圖 4-5 while 迴圈的語法與流程圖

例如：使用 while 迴圈輸出 0 1 2 三行數字的程式可設計如下，此程式執行過程如下：

① i = 0 時，i < 3 (0 < 3) 成立，執行迴圈，輸出 0，i 加 1，i = 1。

② i = 1 時，i < 3 (1 < 3) 成立，執行迴圈，輸出 1，i 加 1，i = 2。

③ i = 2 時，i < 3 (2 < 3) 成立，執行迴圈，輸出 2，i 加 1，i = 3。

④ i = 3 時，i < 3 (3 < 3) 不成立，結束迴圈。

實際上，此例也可使用 for 迴圈設計，程式碼如右下：

```
i = 0
while i < 3:
    print(i)
    i = i + 1
```

```
for i in range(3):
    print(i)
```

程式設計

```
n = int(input())                          # 讀取輸入，轉成整數，指定給 n
prime = 1                                 # 使用 prime 標示 n 是質數
for i in range(2, int(n ** 0.5) + 1):     # 執行迴圈，i 在範圍 2~ √n
    if n % i == 0:
        prime = 0
        break
print('Yes') if prime == 1 else print('No')
```

執行結果

89	29996224275833
Yes	Yes

4.5 APCS 實作題

範例 4.5-1 人力分配（201710 APCS 第 1 題）

若某公司有兩個工廠，分別配置 X1 和 X2 位員工時，獲利為 Y1 和 Y2。獲利與員工數 X1 和 X2 的關係式如下：

$$Y1 = a1\ X1^2 + b1\ X1 + c1$$

$$Y2 = a2\ X2^2 + b2\ X2 + c2$$

設計一個程式，將 n 個員工分配到兩個工廠，以取得最大獲利。

輸入：第 1 行和第 2 行各有三個整數，分別為 ai, bi, ci (i = 1, 2) 之值，第 3 行有一個正整數，表示員工人數。

輸出：一個整數，代表最大獲利。

範例一：輸入	範例二：輸入
2 -1 3 4 -5 2 2	-1 -2 -3 3 2 1 5
範例一：正確輸出	範例二：正確輸出
11	83

解題方法

1. 本題可使用 for 迴圈解題，計算出 n 位員工分配到兩間工廠，所有的獲利情形，再找出最大獲利。for 迴圈可設計為 for i in range(n + 1):。
2. 若分配 i 人到工廠一，工廠二便會分配到 n - i 人。讓 i 從 0 到 n，一一試算每種分配方法的總獲利，並找出最大獲利。
3. 工廠一分配 i 人，工廠二 n - i 人，所以 X1 = i, X2 = n - i。

	人數	獲利
工廠一	i	a1 * i * i + b1 * i + c1
工廠二	n - i	a2 * (n - i) * (n - i) + b2 * (n - i) + c2

總獲利 y = Y1 + Y2

= a1 * X1 * X1 + b1 * X1 + c1 + a2 * X2 * X2 + b2 * X2 + c2

= a1 * i * i + b1 * i + c1 + a2 * (n - i) * (n- i) + b2 * (n - i) + c2

4. 解題演算法可設計如下：

輸入 a1, b1, c1, a2, b2, c2 及員工人數 n

最大獲利 M 初始值設成很小，如 -99999

執行迴圈 i 從 0 ~ n

獲利 y = a1 * i * i + b1 * i + c1 + a2 * (n - i) * (n - i) + b2 * (n - i) + c2

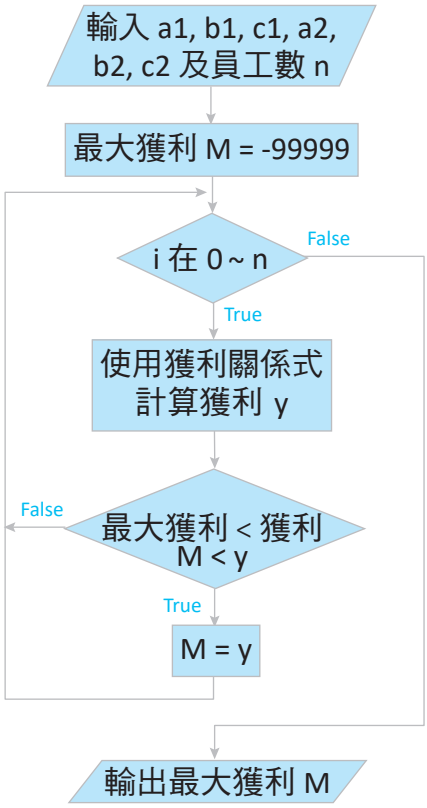
if 最大獲利 M < 獲利 y:

最大獲利 M = 獲利 y

輸出最大獲利 M

5. 最大獲利 M 初始值設成很小的數，目的是讓迴圈執行第一次時，if 條件式就能成立，使 M 能被 y 取代，等同迴圈一開始，就將第一種分配方法的獲利設為最大獲利 M。

6. 解題流程圖如下：



程式設計

```
1 a1,b1,c1 = map(int,input().split())
2 a2,b2,c2 = map(int,input().split())
3 n = int(input())
4 M = -99999                                    #M 是最大獲利，初始值設成很小
5 for i in range(n + 1):
```

學習挑戰

一、選擇題

1. range(10, 1, 2) 產生的數列有多少個數？

   （A）10　　（B）9

   （C）5　　（D）0

2. 若 a = 5，執行以下程式，輸出為何？

```
for i in range(20):
    i = i + a
print(i)
```

   （A）19　　（B）20

   （C）24　　（D）25

3. 執行以下程式，total = ？

```
total = 10
for i in range(1, 11):
    total = total + i
```

   （A）10　　（B）45

   （C）55　　（D）65

4. 執行以下程式，total = ？

```
total = 0
for i in range(1, 11, 3):
    total = total + i
```

   （A）12　　（B）22

   （C）35　　（D）55