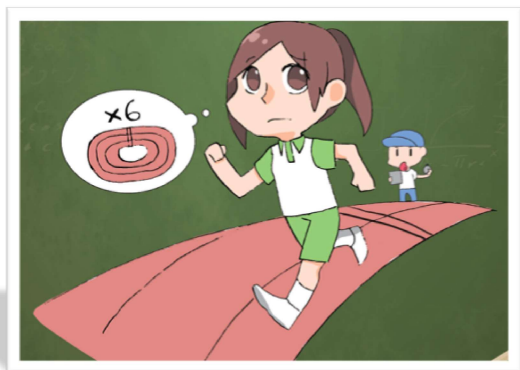


10-1. 重複結構

我們在聽音樂時，如果聽到喜歡的歌曲，可以設定讓這首歌持續播放，而不用每次動手重新播放音樂。就好比上體育課時，老師叫我們跑操場，而且一跑就是十圈，重複跑十次操場。



Question 1: 什麼是迴圈？

迴圈就是讓一段程式碼，重複執行很多次的結構。

10.0.0.1 Question 2: 迴圈有幾種？

Python 有兩種常用的迴圈，分別是條件迴圈（`while`）與計次迴圈（`for`）。

(1) `while` 條件迴圈：

在 `while` 迴圈中，依據測試條件是否成立，決定要不要繼續或跳出迴圈。

`while` 迴圈通常用於不固定次數的迴圈，只要條件符合就繼續做。

例如：猜數字遊戲

兩人（A 與 B）玩猜數字遊戲，一人(A)寫下心中所想的一個數，由另一人(B)去猜。然後提示B所猜數字的回答要「猜大一點」或「猜小一點」，直到B猜對為止。

(2) `for` 計次迴圈

在 `for` 迴圈中，依據計數變數值的變化，從初值到終值反覆執行區塊程式碼，當變數等於終值時離開迴圈。


`for` 迴圈結構通常用於已知重複次數的程式。

例如：假設要撰寫程式產生1000個「Hello」。

方法一：寫1000個「`print('Hello')`」。

產生 1000 個「Hello」的程式碼

```
print('Hello')
print('Hello')
print('Hello')
...
print('Hello')
```



1000 個
print('Hello')

方法二：使用迴圈結構，簡化程式碼，達成相同功能。

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

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

10-2. while 條件迴圈

10-2-1. 基本觀念

1. 條件迴圈(conditional loop): 程式碼要執行的確切次數未知。

只要布林條件式的判斷結果為True，電腦就會繼續執行迴圈，也就是重複執行這個程式區塊中的程式碼。通常，不知道要重複多少次時，就使用條件迴圈。比如玩猜數字遊戲，只要使用者沒有猜對就繼續玩，直到猜對為止。

2. 條件迴圈用while敘述式來建立。

範例1：從if 條件式到 while迴圈

```
1  #|echo: true
2  #|eval: false
3  #|includes: false
4
5  ## if 條件式到 while迴圈
6  ## if 條件式
7  n = 5
8  if n < 10:
9      print("n小於10!")
10
11  ## while迴圈 (一): 將if改成while，電腦列不停止印
12  n = 5
13  while n < 10:
14      print("n小於10!")  # ctrl+c 停掉
15
16  ## while迴圈 (二): 要加上「變更布林條件的值」
```

```

17  n = 5                                #初始值
18  while n < 10:                        #當n小於10印出
19      print("n小於10!")
20      n = 100                          #變更條件
21      print("我要出迴圈！！")
22  print("我出來了")
23
24  ## while迴圈 (三)：靈活地「變更布林條件的值」
25  n = 5                                # 初始值start
26  while n < 10: # 判斷條件：當x小於10印出 stop
27      print("n小於10!")
28      n = n + 1 # 變更條件step
29  print("我出來了")

```

10-2-2. 基礎語法

1. while 指令後面的測試條件，若為真就執行該區塊程式碼，直到測試條件為假時跳出。

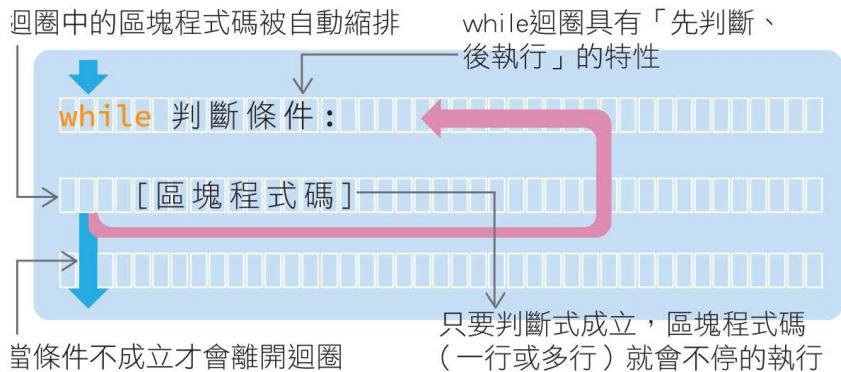


圖 4-3.5 while 迴圈語法說明

while 迴圈

while是指「當」判斷條件成立或為真值時，才會執行底下區塊的內容，執行完區塊後，再回while判斷是否條件繼續成立，若不成立，則離開迴圈，若成立就一直反覆執行區塊內容。

2. 虛擬程式碼: 所有條件迴圈的結構形式(3個s)

```

設定初始條件          # 初始條件 (Start)
條件為True時：        # 判斷條件 (Stop)
    程式碼每次在迴圈內執行
    變更條件            # 變更條件 (Step)

```

3. 流程圖

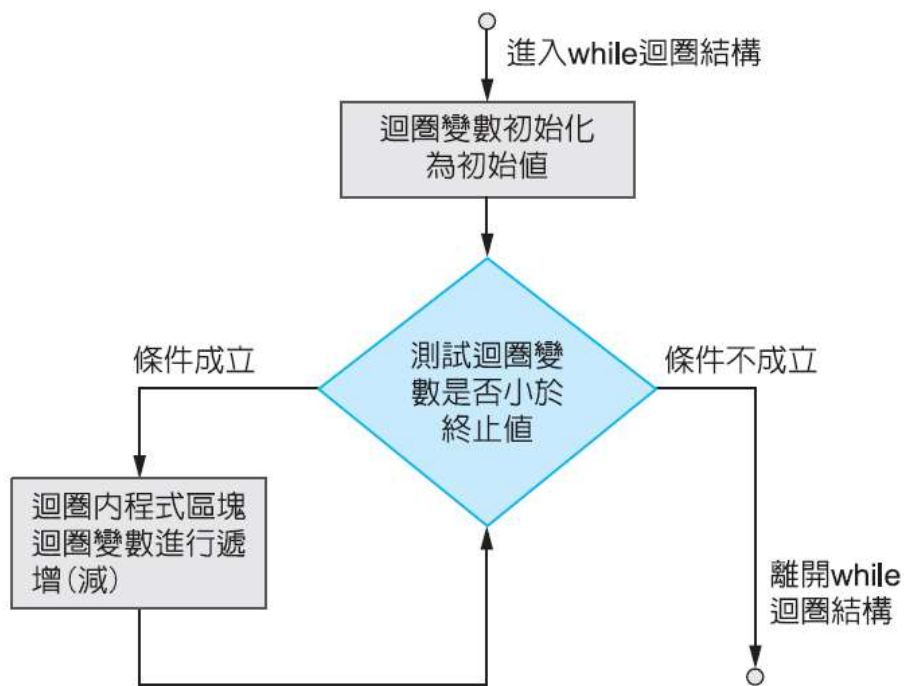


圖 A-14 流程圖

範例2：列印10次hello

```

1  ## 一般型迴圈
2  ## 方法（一）：
3  n = 0 # 初始條件 Start（Python習慣從0開始算）
4  while n < 10: # 判斷條件 Stop（n = 9，印第10次；n = 10，跳出迴圈。）
5      print("hello")
6      n = n + 1 # 更新條件 Step

```

```

hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello

```

```

1  #|echo: true
2  #|eval: true
3  #|includes: true
4
5  ## 方法（二）：（順序不同，更合邏輯）
6  n = 0 # 初始條件
7  while n < 10: # 判斷條件
8      n = n + 1 # 更新條件
9      print("這是第" + n + "次的hello")

```

```
print( "這是第", n, "次的hello" )
```

```
1 #|echo: true
2 #|eval: true
3 #|includes: true
4
5 ## 方法 ( 三 ) : (更完整)
6 n = 0 # 初始條件
7 while n < 10: # 判斷條件
8     n = n + 1 # 更新條件
9     left = 10 - n
10    print("這是第", n, "次的hello", "還有", left, "次機會")
```

【隨堂練習1】：從1到5逐一列印

```
1 #|echo: true
2 #|eval: false
3 #|includes: false
4
5 ## 從1到5
6 n = 0 # Start
7 while n < 5: # Stop
8     n = n + 1 # Step
9     print("n= ", n)
10    print("Finished")
```

範例3：求算1到10之和， $1+2+3+\dots+8+9+10=?$

```
1 ## 記憶型迴圈
2 n = 0 # 初始條件
3 sum = 0
4 ##  $1+2+3+\dots+8+9+10=?$ 
5 while n < 10: # 判斷條件
6     n = n + 1
7     sum = sum + n # 更新條件
8    print("答案是：", sum) # 最後結果
```

答案是： 55

```
1 #|echo: true
2 #|eval: false
3 #|includes: false
4
5 ## 類似寫法，但初始條件不同、(使得程式的順序也不同)
6 n = 1 # 初始條件(Start)
7 sum = 0
8 while n <= 10: # 判斷條件 (一直加到最後一項的數字) (Stop)
9     sum = sum + n # 更新條件 (Step)
10    n = n + 1
```

```
11 print("The answer is", sum)
```

【隨堂練習2】：求算1到9奇數之和， $1+3+5+7+9=?$

~ 請注意Start, Stop, and Step的設定

```
1 #|echo: true
2 #|eval: false
3 #|includes: false
4
5 n = 1
6 sum = 0
7 while n < 10:
8     sum = sum + n
9     n = n + 2
10 print("1+3 + 5 + 7 + 9", sum)
```

範例4：求算1到n之和大於50， $1+2+3+.....+n > 50$ 的n是多少？

```
1 ## 記憶型迴圈
2 n = 0          # 初始條件
3 sum = 0
4 while sum <= 50:    # 判斷條件 (只要總和小於等於50就繼續加)
5     n = n + 1
6     sum = sum + n    # 更新條件
7 print("最後加的n是", n) # 最後結果
```

最後加的n是 10

Summary: while迴圈三大功能 - 列印、計數、加總

10-3 進階迴圈

10-3-1 特殊的指令

在特殊需求下，迴圈會使用break與continue，以及else指令。

- 1. 要跳出迴圈，可以使用break指令讓迴圈終止不再執行

```
1 ## 針對對while迴圈執行
2 n = 0
3 while (n < 7):
4     n = n + 1
5     if n == 4:
6         break
7     print(n)
```

- 要跳過某一迴圈不執行，但還是繼續做後面的迴圈，則使用continue。

範例：請寫一個程式，輸入總樓層，列出大樓所有樓層名稱。

華人有時蠻避諱「4樓」的，會把「4樓」稱為「5樓」。

```
1  ##針對對while迴圈執行continue
2  n = 0
3  while (n < 7):
4      n += 1
5      if n == 4:
6          continue
7      print(n)
```

- 迴圈正常結束時，執行else 程式區塊，若迴圈經由break 中斷，就不會執行else程式區塊。

10.0.1 10-3-2 無窮迴圈

當迴圈使用「while True:」指令時，測試條件會永遠成立，形成無窮迴圈。

- 要跳出迴圈，請使用break中斷。

```
while True:
    條件數值運算式
    if 條件:
        要被重複執行的指令
```

範例: 猜大猜小遊戲

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，並假設所猜的數字介於1到100之間。

解題想法

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。

```
1  ## 直接變數賦值比較簡單
2  target = 88
3  while True:
4      guess = int(input("Enter a number: "))
5      if guess == target:
6          print("Well done!猜中了!")
7          break # 跳出迴圈
8      else:
9          print("Try again")
10         if guess > target:
11             print("Guess smaller, 猜小一點")
12         else:
13             print("Guess larger, 猜大一點")
```

