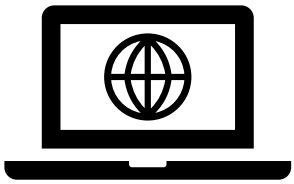
Chapter 6：迴圈 X CAPSTONE 3

本章重點

迴圈可重複執行相同的工作。

準備材料



可上網的電腦

學習目標

1. 了解 while 迴圈的使用時機。
2. 了解 for 迴圈的使用時機。
3. for 迴圈走訪容器中的元素。
4. CAPSTONE 3：拼字遊戲。(請參考電子檔)

6-1. 程式的重複工作

情境想一想

生活中常常需要重複動作的情境，請說明下列重複的動作為何，什麼條件下會停止工作，條件可能不只 1 項。

1. 登入網站時輸入密碼。
2. 聽音樂時開啟循環播放。
3. 參加球隊訓練，跑操場鍛鍊體能。
4. 上網搶購演唱會門票。
5. 玩數字賓果遊戲。

參考答案

1. 一直重複輸入帳號和密碼進行登入，直到輸入正確密碼完成登入為止；若密碼輸入次數太多，帳號會被鎖住。
2. 一直聆聽音樂，直到按下停止鈕為止。
3. 一直繞著操場跑道跑步，直到達成教練要求的圈數，或教練說停止。
4. 一直按購買選擇場次，直到買到門票或賣完了才停止。
5. 一直消去填寫的數字，直到有人消去的數字連成 5 條水平、垂直或對角線，遊戲才結束。

6-1-1. 重複執行一樣的動作

我們想執行 print('abc') 3 次，就要重複寫 3 行。

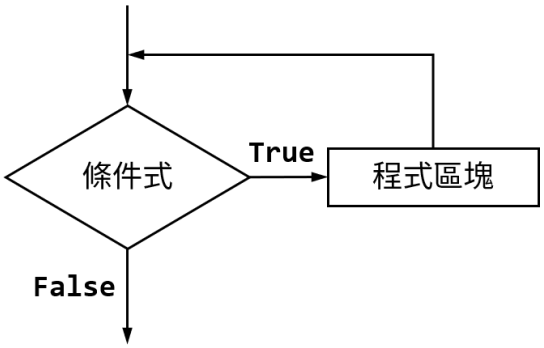
```
print('abc')
print('abc')
print('abc')
```

3 次還行，30 次呢？使用迴圈可以簡化重複執行的工作。

while 的語法

以下為 while 迴圈的基本語法。

```
while 條件式：
    程式區塊
```



固定次數的 while 迴圈

改寫執行 3 次 print('abc') 的程式。

6-1 指定訊息顯示的內容與次數。

```
s = input('請輸入顯示內容: ')
n = int(input('請輸入顯示次數: '))

while n > 0:
    print(s)
    n = n - 1
```

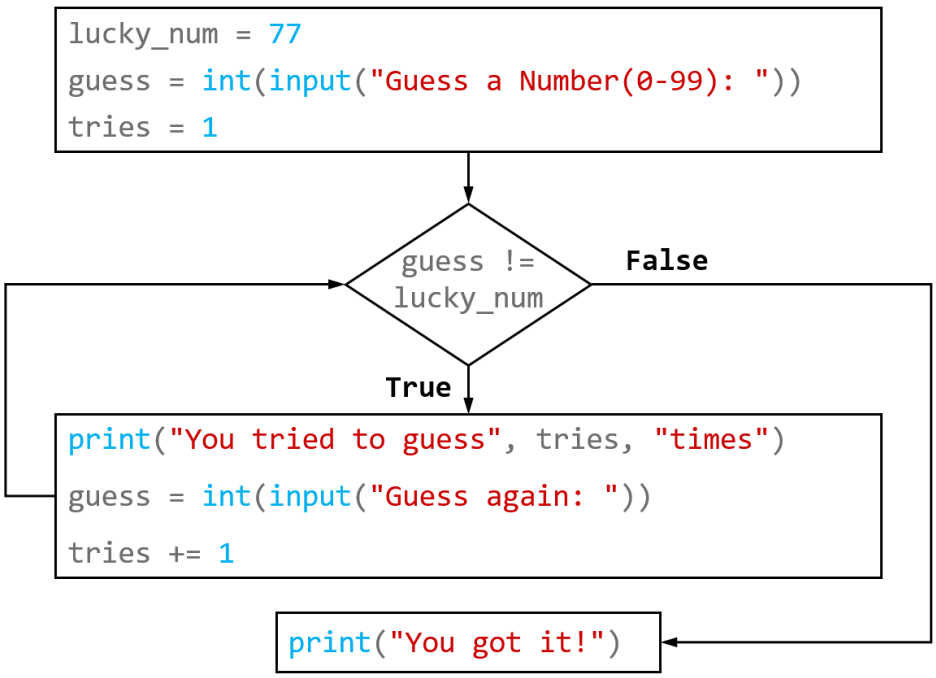
觀念驗證 6.1

請修改 **6-1**，讓程式除了顯示字串外，也顯示 n 值的變化。

answer.

不定次數的 while 迴圈

讓使用者可以一直輸入數字，直到猜中預設的幸運數字 (77)：



6-2 使用 while 迴圈實作猜數字遊戲。

```
lucky_num = 77
guess = int(input('Guess a Number(0-99): '))
tries = 1

while guess != lucky_num:
    print('You tried to guess', tries, 'times')
    guess = int(input('Guess again: '))
    tries += 1

print('You got it!')
```

觀念驗證 6.2

若將 **6-2** while 迴圈中的 3 行程式改寫如下，請問是否會影響執行結果？

```
    tries += 1
    print('You tried to guess', tries, 'times')
    guess = int(input('Guess again: ')
```

answer.

6-1-2. 用 break、continue 控制迴圈執行

while 迴圈只有當條件式為 False 時才會跳出。在 Python 中，可使用 break 與 continue 這兩個保留字進一步控制迴圈。

- break：可用來跳出整個迴圈。
- continue：可略過其後未執行的敘述，跳到『 下一圈 』的開頭。

提早跳離迴圈 (break)

6-3 猜謎語，若已猜了 100 次，使用 break 跳離迴圈。

```
secret = 'code'
max_tries = 100
guess = input('Guess a word: ')
tries = 1

while guess != secret:
    print('You tried to guess', tries, 'times')
    if tries == max_tries:
        print('You ran out of tries.')
        break
    guess = input('Guess again: ')
    tries += 1

if tries <= max_tries and guess == secret:
    print('You got it!')
```

跳離迴圈的情況有兩種：猜中單字或猜超過 100 次。

- 所以需要迴圈外的 if 判斷式。

在巢狀迴圈，若 break 的階層是屬於內層的迴圈，僅會跳出內層的迴圈，外層的迴圈仍會繼續執行。

程式設計師的思考

常常重複使用到某個特定的值，建議使用變數來宣告，以便後續維護與修改。在 **7-3** 中要修改猜測的次數 max_tries = 100，例如：max_tries = 200，就不必再檢查其他地方是否需要修正。

返回迴圈的開頭 (continue)

例如：使用 continue 來跳過 5 不印，最後再顯示 'END'。

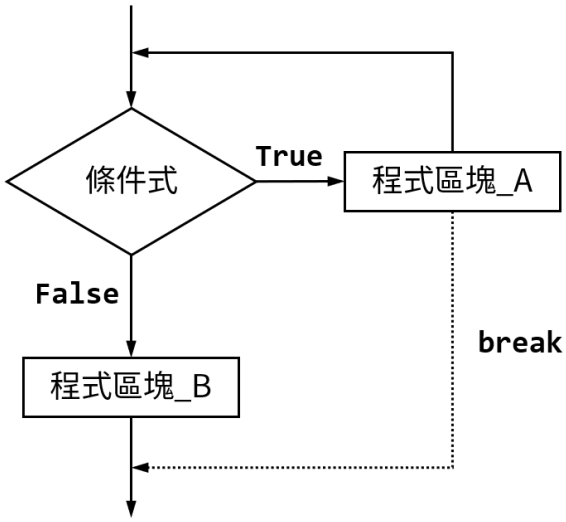
6-4 指定訊息顯示的內容與次數。

```
i = 1

while(i < 10):
    if i == 5:
        i += 1
        continue
    print(i, end = ' ')
    i += 1

print('End')
```

6-1-3. while-else 迴圈

while 也可以有 else，跟 if-else 很像。當條件不成立時，就會執行 else 的程式區塊，然後結束迴圈。

```
while 條件式：
    程式區塊_A
else：
    程式區塊_B
```



6-5 要求輸入通關密語，迴圈中設定的密語是喵喵，使用者若不知密語，也可以輸入 'out' 放棄通關，只有正常結束時才會顯示成功訊息：

6-5 指定訊息顯示的內容與次數。

```
s = ''

while s != '喵喵':
    if s != '':
        print('不對喔!')
    s = input('請輸入通關密語: ')
    if s == 'out':
        break
else:
    print('恭喜你過關了')

print('再見! ')
```

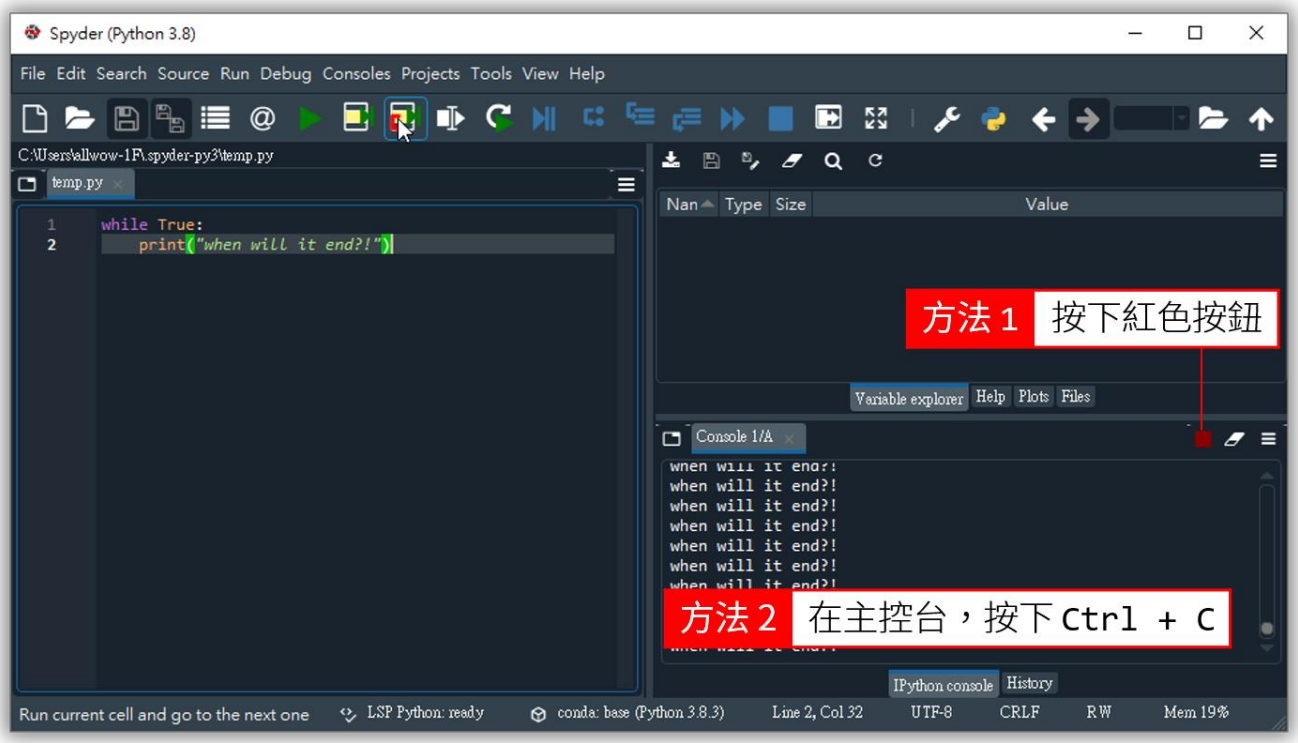
6-1-4. 無窮迴圈

永遠無法結束的迴圈稱為無窮迴圈。

6-6 無窮迴圈的例子。

```
while True:
    print('when will it end?!')
```

若要離開無窮迴圈，強制停止程式的兩種方法：



結論

1. while 迴圈會先確認條件式是否成立，才決定是否繼續執行所屬程式碼。
2. break 用來強制跳離整個迴圈，continue 會跳過當次迴圈後面的程式碼，直接返回下一次迴圈的開頭。
3. while … else … 的用法跟 if … else … 類似。
4. 設計 while 的條件式時，要注意迴圈是否有結束或跳離的可能 (條件式有不成立的時候)，否則會導致無窮迴圈，造成程式持續執行。

練習 6-1

1. 試著執行下列程式，你會發現它是一個無窮迴圈，請猜測該程式碼想要達成的目標是什麼?並修改該程式為可正確執行的程式。

```
num = 8
guess = int(input('Guess my number: '))
while guess != num:
    guess = input('Guess again: ')
print('Right!')
```

2. 試著編寫一個程式，在執行時會先詢問是否要玩猜數字的遊戲？若輸入 y 或 yes，則開始遊戲。程式中預先設定好一個 1 ~ 10 的數字作為謎底，然後每次執行讓使用者任意挑選一個數字，並持續詢問，直到猜中數字為止。如果猜中數字，則程式會顯示恭喜的訊息，然後再次詢問是否要繼續此遊戲，若輸入 y 或 yes，則程式會重複上述動作，若輸入其他字元，則結束程式。

6-2. for 迴圈－計次執行

情境想一想

根據以下描述情境，試回答重複執行的次數是可以預期還是無法預期的？

1. 幫全班同學每人拍攝一張畢業紀念照。
2. 一片接一片，把整包洋芋片吃完。
3. 連續擲骰子，擲出的總點數相加超過 20 點。
4. 在 20 公尺跑道折返跑，總共跑 400 公尺。
5. 自行車繞圈競逐賽，限時 30 分鐘。

參考答案

1. 可以，全班有幾位同學就拍攝幾次。
2. 可以，一包洋芋片有幾片就吃幾片。
3. 不可以，每次擲出的點數不同，無法得知要擲幾次。
4. 可以，不過去程和回程算一趟，總共 400 公尺相當於折返跑 10 次。
5. 不可以，多輛自行車同時競逐、無法預期車速，所以不知道會繞多少圈。

6-2-1. 指定迴圈執行次數

如果已知要重複的次數，使用 for 迴圈搭配 range() 函數。

```
for 變數 in range(N):
    程式區塊
```

其中，range() 是一個函式，N 是一個整數，range(N) 可以讓迴圈執行 N 次。(從 0 開始，到 N – 1，第 N 次不執行)

- 例如：使用 for 迴圈連續印 10 次。

6-7 for 迴圈 VS 不使用迴圈。

```
# 使用 for 迴圈 (6-7a.py)
for i in range(10):
    print('Hello Python')
```

```
# 不使用迴圈 (6-7b.py)
print('Hello Python')
print('Hello Python')
print('Hello Python')
print('Hello Python')
print('Hello Python')
print('Hello Python')
print('Hello Python')
print('Hello Python')
print('Hello Python')
print('Hello Python')
```

range() 函式

range() 函式產生的這串整數稱為數列 (sequence of numbers)。

6-8 使用 for 迴圈依序顯示數列元素。

```
for v in range(2):
    print('var v is', v)
```

觀念驗證 6.3

下列程式碼會產生哪些整數？(range() 需搭配 for 迴圈，才能正確執行)

1. range(1)
2. range(5)
3. range(100)

程式設計師的思考

為何不直接控制 for 迴圈的次數？常常無法明確知道迴圈要執行的次數，透過讀取數列或容器中的元素來控制迴圈的次數，讓程式自己算要執行幾次。(參見 Chpater 6-3)

例如：老師要學生繳交作業，只要跟班代說：幫我把全班的作業都收回，班代就會一一跟每位同學收作業，而不用先問：班上有多少人。

6-2-2. 指定 range() 函式產生的數列範圍

range() 完整的參數如下：

```
range(起始值, 終止值, 間隔值)
```

例如：

- range(2)：產生 (0, 1) 的數列，代表終止值為 2，忽略起始值 (0) 與間隔值 (1)。
- range(2, 5)：產生 (2, 3, 4) 的數列，代表起始值為 2 與終止值為 5，忽略間隔值 (1)。
- range(10, 1, -1)：產生 (10, 9, 8, 7, 6, 5, 4, 3, 2) 的數列，間隔值為 -1。

6-9 不同數列範圍的 for 迴圈。

```
for i in range(10, 1, -1):
    print(i, end = ' ')

print()

for i in range(0, 9, 2):
    print(i, end = ' ')
```

觀念驗證 6.4

使用 for 迴圈，印出下列 range() 函式所產生的數列。

1. range(0, 9)
2. range(3, 8)
3. range(-2, , 2)
4. range(5, -5, -3)
5. range(4, 1, 2)

6-2-3. 多層 for 迴圈

多層 for 迴圈可搭配 range() 做多層的數值運算。

6-10 簡易九九乘法表。

```
for i in range(1, 10):
    for j in range(1, 10):
        print(j, 'x', i, '=', j * i, end = ' ')
    print()
```

觀念驗證 6.5

請使用雙層 for 迴圈，讓使用者輸入計時的分鐘數，然後在畫面上模擬倒數計時的分鐘數與秒數 (只需顯示數字，不用考慮實際時間)。

answer.

6-2-4. 使用 break、continue 與 for-else

使用 break、continue

例如：結合 break 與 continue。

6-11 連續印出 1 ~ 10，但跳過 5 不印。

```
for i in range(1, 20):
    if i == 5:
        continue
    print(i, end = ' ')
    if i == 10:
        break

print('END')
```

使用 for-else。

例如：結合 break 與 continue。

```
for 變數 in range(N)：
    程式區塊_A
else：
    程式區塊_B
```

- 當 for 迴圈正常結束後，會接著執行 else 區塊。
- 若在 for 的區塊使用 break 跳出迴圈，則不執行 else 區塊。

6-12 找出指定範圍內的所有質數。

```
num = int(input('Please specify the range(0~N): '))

for i in range(2, num + 1):
    for j in range(2, i):
        if(i % j == 0):
            break
    else:
        print(i)
```

注意：質數只能被 1 與自己整除，內層迴圈測試外層的數字 i 是否會被 2 到 i-1 之間的整數整除，只要一整除就不是質數。

結論

for 搭配 range() 指定迴圈次數、巢狀 for 迴圈、break、continue、else 的使用。

1. for 用來可預期執行次數的迴圈。
2. range(N) 可產生 0 到 N-1 的整數數列。
3. for 可以使用 break 來跳出迴圈，或用 continue 來跳到下一個迴圈的開頭，或者搭配 else，在迴圈正常結束後執行另一個程式區塊。

練習 6-2

1. 將 1 到 100 間，所有 3 的倍數印出來，並在最後顯示總共有幾個數字。

2. 請寫一個可以產生任意層數三角形的程式。

6-3. for 迴圈走訪容器中的元素

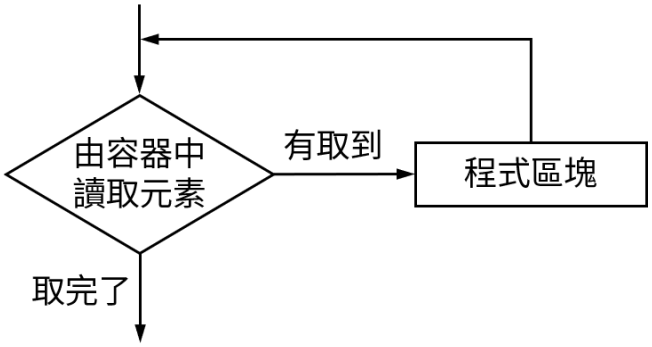
走訪容器

容器（又稱為資料結構）：『 內部還可以存放物件的物件 』，與日常生活的容器一樣，容器內也可擺放容器。

- 例如：字串、tuple、串列（list）、集合（set）、字典（dict）等。

走訪容器：用 for 迴圈將容器中的元素，一一取出來處理。

```
for 變數 in 容器：
    程式區塊
```



情境想一想

學校將舉辦兩天的迎新活動，新生可以自由選擇要出席哪一天的活動，或者兩天都出席。你負責製作邀請卡以提醒每位新生準時出席，想一想要怎麼進行這項工作，其中有哪些工作是屬於重複性的工作。

參考答案

1. 先整理好報名的名單，要包含參加者的姓名與出席日期。
2. 製作邀請卡的範本，將參加者的姓名和出席日期留空。
3. 依照報名的人數，印製相同份數的邀請卡。
4. 依照名單，在每份邀請卡中填上每個新生的姓名和出席日期。

上述步驟 3、4 都是屬於重複性的作業，其中步驟 3 是依照固定次數（報名人數）來重複相同的作業，步驟 4 則是依照名單來將每一個報名者的姓名和出席日期填到邀請卡中。

Python 的基本資料結構（容器）

基本的資料結構：程式經常需要處理資料，而資料結構是存放資料的容器。

- 字串容器：由字元組成。例如：string = '52python'。

```
>>> string = '52python'
>>> string[0]
5
```

- tuple 容器：由資料物件組成。例如：tuple = (1, '2', 3)。

```
>>> tuple = (1, '2', 3)
>>> tuple[0]
1
```

- 串列容器 (list)：由資料物件組成。例如：list = [1, '2', 3]。

```
>>> list = [1, '2', 3]
>>> list[2]
3
```

- 集合容器 (set)：由資料物件組成。例如：set = {1, '2', 3}。集合為無序，沒有索引。

```
>>> set = {1, '2', 3}
```

- 字典容器 (dict)：由資料物件組成，以鍵：值表示。例如：dick = {'A':1, 'B':'2', 'C':3}。

```
>>> dick = {'A':1, 'B':'2', 'C':3}
>>> dick['B']
'2'
```

Python 的第三方資料結構：numpy.array

- Numpy：Python 的擴充模組，也常用於機器學習的資料處理。

```
>>> import numpy as np    # 匯入 numpy
>>> a = np.array([10, 2, 45, 32, 24])    # 建立五個元素
>>> len(a)    # 計算元素個數
```

```
>>> a[2:4]     # 取出第 2、3 個元素
>>> a[:4]      # 取出第 1 ~ 3 個元素
```

6-13 用 for 迴圈走訪 'Python is fun!' 中的每一個字元：

6-13 使用 for 迴圈走訪字串中的字元。

```
for ch in 'Python is fun!':
    print('the character is', ch)
```

觀念驗證 6.6

寫一支程式讓使用者輸入任意字串，然後用 for 迴圈走訪字串中的字元，當字元為母音 (a, e, I, o, u) 時，就將字元顯示出來。

answer.

6-3-1. 走訪字串裡的字元

6-14 使用 for 迴圈走訪字串的索引值。

6-14 for 走訪字串的索引值。

```
my_string = 'Python is fun!'
len_s = len(my_string)

for i in range(len_s):
    print('the character is', my_string[i])
```

程式設計師的思考

KISS (Keep Instruction Simple and Stupid)：

能夠寫出很多行或複雜的程式，並不代表你是很好的程式設計師。Python 最大的特色就是語法簡潔且可讀性高，別忘了保持 KISS 程式。

可用下列方法增加 KISS：

1. 程式的邏輯要直覺，不夠直覺或太複雜就要改進。
2. 用紙筆寫出你的想法，思考是不是還有其他更好的方法。
3. 上網搜尋或問問其他人。

6-3-2. 走訪 tuple 裡的元素

將得獎人姓名放在同一個 tuple 中，依序用 for 迴圈顯示出來：

6-15 使用 for 迴圈走訪 tuple。

```
winners = ('Peter', 'Mark', 'Joy')
print('恭喜以下得獎人: ')

for name in winners:
    print(name)
```

可以改用前面 **6-13**『 走訪索引值 』的方式來處理，例如：**6-16**。

6-16 使用 for 迴圈走訪 tuple。

```
prizes = ('頭獎', '二獎', '三獎')
winners = ('Peter', 'Mark', 'Joy')
print('恭喜以下得獎人: ')

for i in range(3):
    print(prizes[i]+ ':' + winners[i])
```

將得獎名稱和得獎人姓名都放在同一個 tuple 中，成為二層的 tuple：

6-17 使用 for 迴圈走訪 tuple。

```
winners = (('頭獎', 'Peter'), ('二獎', 'Mark'), ('三獎', 'Joy'))
print('恭喜以下得獎人: ')

for e in winners:
    print(e[0] + ':' + e[1])
```

觀念驗證 6.7

有時候我們將姓和名分開儲存，例如：("張", "天才")，以方便未來應用於只需要姓或名的場合。請用此儲存格式，依序將"張天才"、"王子帥"、"陳美美"，儲存到二層的 tuple 中，並用 for 迴圈依序將他們的全名顯示出來。

answer.

6-3-4. for 迴圈 VS while 迴圈

while 迴圈：以條件判斷來決定是否繼續執行。

for 迴圈：以走訪容器，例如：走訪數列、字串、tuple… 等。

6-18 是 for 與 while 迴圈的比較：

6-18 for 迴圈 VS while 迴圈。

```
# for 迴圈 (6-18a.py)
for x in range(3):
    print('var x is', x)
```

```
# while 迴圈 (6-18b.py)
x = 0

while x < 3:
    print('var x is', x)
    x += 1
```

- 所有的 for 迴圈都可以用 while 來改寫，但並非所有的 while 迴圈可以用 for 迴圈來

改寫，除非事先知道迴圈的執行次數，或是已確定要走訪容器的內容為何。

- 例如：猜數字，你事先不會知道要猜幾次？或是物聯網器實作，要讓感測器一直監測。就只能用 while 迴圈。

觀念驗證 6.8

請將下列 while 迴圈改寫成 for 迴圈：

```
password = 'robot fort flower graph'
space_count = 0
i = 0

while i < len(password):
    if password[i] == ' ':
        space_count += 1
    i += 1

print(space_count)
```

answer.

結論

for 迴圈比 while 較簡潔，能用 for 就不用 while。

1. for 用來走訪各種容器，包括 range() 產生的數列、字串、tuple…等。
2. 可以用 for 來依序走訪字串中的每一個字元，取代索引的方式。
3. while 可以通用於各種需要迴圈的場合，但必須自行處理所有細節。
4. for 可以方便我們走訪容器物件，或走訪 range() 所產生的特定數列，它會幫我們處理迴圈變數，並在走訪完時自動跳離迴圈。

練習 6-3

1. 請寫一個程式，可讓使用者輸入一串名字，每個名字使用空白分隔，然後程式會逐行針對每個名字說 Hi。例如：使用者輸入 Zoe Xander Young，你的程式就會逐行顯示：

```
Hi Zoe
Hi Xander
Hi Young
```

2. 請寫一個程式， 先用 while 迴圈讓使用者一一輸入想購買的物品，並將輸入的物品加到 tuple 中，當輸入 "沒了" 時就結束迴圈，然後再用 for 迴圈把完整的購物清單列出來。例如：

```
請輸入要買的商品：牙刷
請輸入要買的商品：牛奶
請輸入要買的商品：衛生紙
請輸入要買的商品：沒了
以下是您的購物清單：
牙刷
牛奶
衛生紙
```