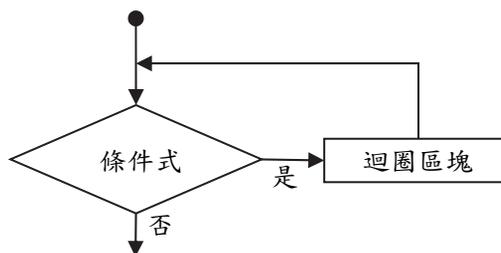


## 5.2 while 迴圈

### 5.2.1 while 敘述

while 迴圈在迴圈開始前，會先檢查條件式是否成立，如果不成立，程式流程會略過迴圈區塊，向下繼續執行。反之，程式流程會在迴圈內循環，直到條件式的結果不成立，才會脫離迴圈。語法如下：

```
while (條件式):
    while 迴圈區塊
```



[例] 使用 while 迴圈，計算  $1+2+ \dots +9+10 = ?$  (檔名：while\_1.py)

```
01 i = 1
02 sum = 0
03 while(i <= 10):
04     sum += i
05     i += 1
06 print(sum)                # 輸出 55
```



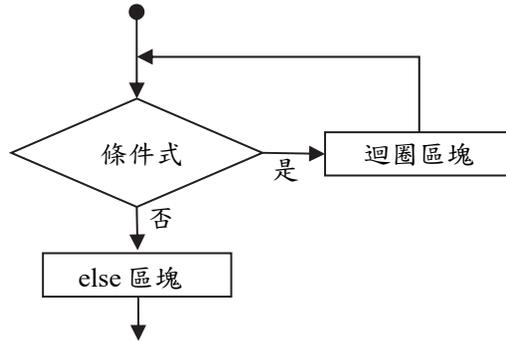
#### 說明

1. 第 3~5 行：為 while 迴圈，i 初值為 1，符合  $i \leq 10$  的條件式，所以會執行第 4、5 行的迴圈區塊。直到 i 的變數值為 11，不符合  $i \leq 10$  的條件式，才會脫離迴圈，執行第 6 行敘述。
2. 第 5 行：i 的變數值加 1，如果省略此行，i 的變數值將不會有不符合  $i \leq 10$  的情形，永遠不會脫離迴圈。所以，撰寫 while 迴圈程式時，必須要有不符合條件式的情形，程式才能正確執行。

### 5.2.2 while ... else 敘述

while 迴圈也可以加上 else 區塊，此時如果 while 迴圈正常結束，或因條件式不成立而略過迴圈區塊時，皆會執行 else 區塊。語法如下：

```
while (條件式):
    迴圈區塊
else:
    else 區塊
```



**【例】** 使用 `while...else` 迴圈，撰寫一個倒數 10~1 的程式，倒數完畢後顯示「時間到!」字串。(檔名：`while_2.py`)

```
01 i = 10
02 while(i > 0):
03     print(i)
04     i -= 1
05 else:
06     print('時間到!')
```

#### 結果

```
10
9
8
7
6
5
4
3
2
1
時間到!
```

#### 說明

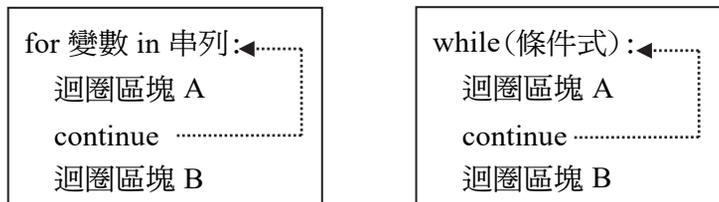
- 第 2~6 行：為 `while` 迴圈，`i` 初值為 10，符合 `i>0` 的條件式，所以會執行第 3、4 行迴圈區塊。直到 `i` 的變數值為 0，不符合 `i>0` 的條件式，就會執行 `else` 區塊的第 6 行敘述。

## 5.3 continue 與 break

在某些情況下，我們可能有中斷迴圈或放棄其後敘述的需求，這時候就要使用 `continue` 或 `break` 敘述。因為這兩個敘述有強制性，換句話說，程式流程遇到這兩個敘述，程式的流程會無條件轉移。所以在使用 `continue` 與 `break` 敘述時，必需配合選擇結構，以條件式來控制程式流向。

### 5.3.1 continue 敘述

如果在迴圈區塊中，在某些條件下，要忽略其後的敘述，跳回迴圈開頭繼續執行，這時候就要使用 `continue` 敘述。`continue` 敘述的語法和流程如下：



#### 簡例 (檔名: continue.py)

試以 `continue` 敘述撰寫一程式，程式要求如下：

1. 計算出 1~100 之間所有奇數值的加總。
2. 輸出計算結果。

#### 程式碼

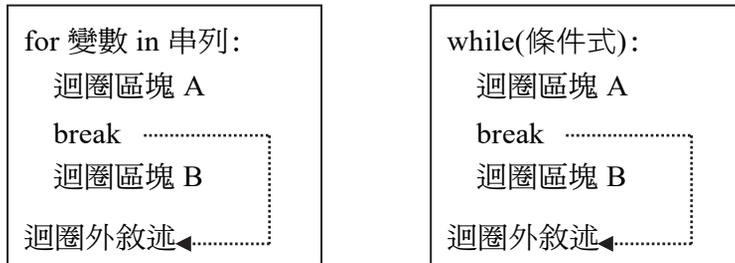
```
檔名: \ex05\continue.py
01 result = 0
02 for i in range(1, 100):
03     if(i % 2 == 0):
04         continue
05     result += i
06 print(result)           # 輸出 2500
```

#### 說明

1. 第 2 行：for 迴圈會逐一取出 `range` 串列元素值指定給變數 `i`。
2. 第 3~5 行：變數 `i` 如果除以 2 時餘數為 0，表示變數 `i` 為偶數，此時程式流程會執行第 4 行的 `continue` 敘述，結束此次迴圈巡訪，跳到第 2 行執行下一個迴圈巡訪流程。反之，會執行第 5 行敘述進行加總後，再繼續迴圈流程。
3. 第 6 行：輸出加總結果。

### 5.3.2 break 敘述

在迴圈區塊中，當程式流程遇到 `break` 敘述時，會忽略其後的敘述，跳至迴圈結尾處直接脫離迴圈，向下繼續執行。假如有 `else` 區塊，因為 `else` 區塊被視為迴圈的一部分，所以同樣會被跳過。`break` 敘述的語法和流程如下：



#### 簡例 (檔名：break.py)

試撰寫一程式，程式會一直等待使用者輸入帳號，直到使用者輸入的帳號是「gotop」才會跳出迴圈，結束程式。

#### 程式碼

檔名：\ex05\break.py

```
01 while True:
02     s = input('請輸入帳號：')
03     if(s == 'gotop'):
04         break
05     print('帳號錯誤!')
06 print('帳號正確!')
```

#### 結果

```
請輸入帳號：168 Enter↵
帳號錯誤!
請輸入帳號：gotop Enter↵
帳號正確!
```

#### 說明

1. 第 1~5 行：為 `while` 迴圈，迴圈的條件式是 `True`，表示每次檢視迴圈的條件式時，所得結果皆會是成立，所以迴圈會不斷執行，這就是稍後章節會介紹的「無窮迴圈」。

- 假如使用者輸入的帳號是「gotop」時，會執行第 4 行的 `break` 敘述，然後脫離迴圈執行第 6 行敘述。反之，會執行第 5 行敘述輸出錯誤訊息，然後繼續迴圈流程。
- 第 4 章曾介紹的海象運算子「`:=`」，也可以運用於 `while` 迴圈，會使程式碼較為簡潔，上面的程式可以改寫為如下：

```
01 while (s := input('請輸入帳號:')) != 'gotop':
02     print('帳號錯誤!')
03     continue
04 print('帳號正確!')
```

## 5.4 巢狀迴圈與無窮迴圈

### 5.4.1 巢狀迴圈

若迴圈內還有迴圈，一層一層由外而內即構成「巢狀迴圈」(Nested Loop)，也可以稱為「多重迴圈」。無論是 `for` 或 `while` 迴圈，或是同時使用都可以構成巢狀迴圈。撰寫巢狀迴圈程式時，要特別注意縮排長度，程式才能正確執行。

**[例]** 試以巢狀迴圈顯示如右圖之直角三角形。(檔名：`n_loop.py`)

```
01 for x in range(1, 6):
02     for y in range(1, 6 - x): print(' ', end = '')
03     for y in range(1, x + 1): print('*', end = '')
04     print()
05 print()
```

 結果

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



## 說明

1. 第 1 行：將 `x` 當成外層迴圈的控制變數(1~5)，表示外迴圈會執行 5 次。
2. 第 2~4 行：外迴圈區塊，迴圈區塊中包含 2 個同階層的內迴圈區塊，執行第 4 行敘述時，會進行換行列印。
3. 第 2 行：第一個內迴圈區塊，負責輸出半形空白。
4. 第 3 行：第二個內迴圈區塊，負責輸出半形星號。

## 5.4.2 無窮迴圈

假使迴圈的條件式永遠為 `True`(真)，會形成無窮迴圈，程式流程會周而復始的執行迴圈區塊。因此撰寫無窮迴圈內的迴圈區塊時，必需有改變條件的敘述，或者使用 `break` 作為脫離迴圈的出口。假如前面的範例 `while_1.py`，不小心寫成以下的程式碼，就會造成無窮迴圈。

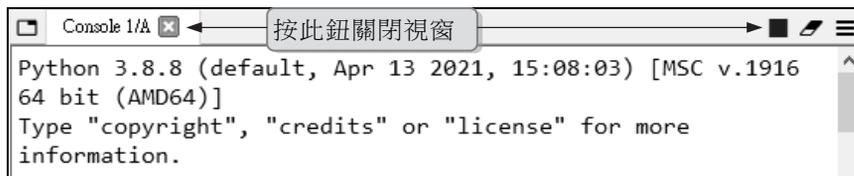
**[例]** 如下簡例為無窮迴圈的程式。(檔名：`while_3.py`)

```
01 i = 1
02 sum = 0
03 while(i <= 10):
04     sum += i
05 i += 1
06 print(sum)
```



## 說明

1. 第 5 行：因為沒有和第 4 行對齊，所以迴圈區塊只有第 4 行不包含第 5 行。執行期間 `i` 值不會變化，迴圈條件式判斷結果永遠皆為 `True`(真)，迴圈會持續執行下去，形成無窮迴圈。
2. 程式執行時若發生執行時間異常，且無反應，就有可能程式陷入無窮迴圈。這時在 `Spyder` 整合環境下，可按  鈕關閉目前的 `Console` 視窗，或是右側的  按鈕中斷程式執行。



 **簡例** (檔名：loop.py)

綜合本章所學習的重復結構來列印九九乘法表。

 **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
```

 **程式碼**

```
檔名：\ex05\loop.py
01 for i in range(10):
02     if(i <= 0):
03         continue
04     j = 1
05     while 1:
06         print(i, '*', j, '=', i*j, end='\t')
07         j = j + 1
08         if(j > 9):
09             break
10     print()
```

 **說明**

1. 第 1 行：將  $i$  當成外層迴圈的控制變數，變數範圍由 0~9。
2. 第 2、3 行：假如迴圈變數小於等於 0，則離開本次迴圈，回到外層迴圈開頭， $i$  值加 1。
3. 第 5 行：內層迴圈條件式為 1，表示內層迴圈是無窮迴圈。
4. 第 6~9 行：迴圈流程如下：
  - ① 列印出  $i*j$  的值。
  - ②  $j$  累加 1。
  - ③ 假如  $j$  大於 9 就結束內層迴圈，否則繼續迴圈。
5. 第 10 行：換行列印。

## 5.5 檢測模擬試題解析

### 題目 (一)

小明撰寫了以下的程式碼。請您檢視這段程式碼(行號僅供參考)之後，回答小明的提問。

```

01 product = 2
02 n = 5
03 while(n != 0):
04     product *= n
05     print(product)
06     n -= 1
07     if n == 3:
08         break

```

請問這一個程式碼片段會列印幾行？

- (A) 0                      (B) 1                      (C) 2                      (D) 3

#### 說明

1. 迴圈執行過程：

敘述 迴圈	n != 0	n 值	product	輸出	n -= 1	n == 3
初始值		5	2			
第 1 圈	成立	5	5 * 2 = 10	10	4	不成立
第 2 圈	成立	4	10 * 4 = 40	40	3	成立

2. 由上表可知迴圈共執行 2 次 print(product)敘述，因此答案：(C)。

程式碼請參考 test05\_1.py。

### 題目 (二)

金山銀行要求您修改符合下列需求的程式碼：

- 允許使用者不斷輸入一串數字。
- 輸出每一串數字的總位數。

這程式有所缺漏，你要如何完成這個程式？

```

01 n = "12345"
02 _____ ① n != "-1":
03     x = 0
04     _____ ② c _____ ③ n:
05         x = x + 1
06     print(x)
07     n = input("請輸入數字或輸入-1 結束程式：")

```

請選取符合需求的程式片段來修訂程式(行號僅供參考)。

- ① (A) for      (B) if      (C) while  
 ② (A) for      (B) if      (C) while  
 ③ (A) and      (B) or      (C) in      (D) not

#### 說明

1. 因為迴圈必需一直執行到使用者輸入「-1」為止，所以① 的答案是 (C)。
2. 第 04 行使用 for 迴圈取出字串內所有的字元，所以② 的答案是(A)，③ 的答案是(C)。程式碼請參考 test05\_2.py。

### 題目 (三)

您正在為小學孩童建立一套互動式乘法表電腦輔助教學程式。其中有一個名為 tables 的函式，該函式計算並顯示從 2 到 12 的所有乘法表組合。

```

01 # 顯示 2 - 12 乘法表
02 def tables():
03     _____ ①
04     _____ ②
05         print(x, '*', y, '=', x * y, end = '\t')
06     print()
07 # main
08 tables()

```

請選擇適當的程式碼片段，完成這段程式碼(行號僅供參考)。

- ① (A) for x in range(13):                      (B) for x in range(2, 13):  
     (C) for x in range(2, 12, 1):            (D) for x in range(12):
- ② (A) for y in range(13):                      (B) for y in range(2, 12, 1):  
     (C) for y in range(2, 13):                (D) for y in range(12):

# 函式

## 7.1 何謂函式

在撰寫程式時，常會碰到一些具有特定功能而又重複出現的程式片段，將這樣的程式片段獨立出來，必要時給予一些參數就能重複呼叫執行使用。像這樣的獨立出來的程式片段就成為一個單獨的程式單元，這種單元稱為「函式」(Function) 或稱為「方法」(Method)。

函式可以分為系統內建函式、使用者自定函式兩類。系統內建函式(簡稱「內建函式」)是編譯系統設計好可直接呼叫使用的函式庫，例如：輸出入函式 `print()`、串列函式 `len()`、字串函式 `find()` …等。內建函式只要會使用，知道給予什麼參數就可傳回所要的結果，不必瞭解函式的內部設計情形，也無法做更改。

而使用者自定函式(簡稱「自定函式」)，是程式設計者在撰寫程式時，應程式需求自己定義出來的函式。自定函式是程式設計者無中生有產生的，隨時可以調整更改。使用自定函式有下列好處：

1. **函式可以重複使用**：大程式只需要著重在系統架構的規劃，功能性或主題性的工作交給函式處理，程式碼可較精簡。
2. **分工合作**：若是較大程式軟體，可依功能切割成多個程式單元，再交由多人共同設計。如此不但可縮短程式開發的時間，也可以達到程式模組化的目的。
3. **容易除錯和維護**：將相同功能的程式敘述片段寫成函式，只需做一次，有助於提高程式的可讀性，也讓程式的除錯及維護更加容易。

## 7.2 內建函式

Python 語言的編譯器提供一個已定義好的函式集合稱為「標準函式庫」，在標準函式庫裡有輸出函式、數值函式、字串函式、檔案輸出入函式、時間函式、亂數函式…等。本章只介紹數值函式、亂數函式、時間函式，其餘函式會在其它章節陸續介紹。

### 7.2.1 數值函式

下表為常用的數值函式：（簡例請參考 funNum.py）

函式	使用說明
abs(n)	取得 n 的絕對值。 例：num = abs(-8) # num ← 8
int(n)	將 n 轉換成整數（小數部分直接捨去）。 例：num = int(12.99) # num ← 12
round(n)	取得 n 四捨六入後的整數值。當小數第一位數字是 5 時，若前一位數字是偶數則捨去；若前一位數字是奇數則進位。 例：num = round(12.6) # num ← 13 例：num = round(12.4) # num ← 12 例：num = round(12.5) # num ← 12, 因 2 是偶數 例：num = round(11.5) # num ← 12, 因 1 是奇數
float(n)	將 n 轉換成浮點數。 例：num = float(12) # num ← 12.0
hex(n)	將 n 轉換成十六進位數值。 例：num = hex(254) # num ← 0xfe
oct(n)	將 n 轉換成八進位數值。 例：num = oct(12) # num ← 0o14
divmod(n, m)	取得 n 除以 m 的商和餘數。 例：(x, y) = divmod(23, 5) # x ← 4, y ← 3 例：ret = divmod(23, 5) # ret[0] ← 4, ret[1] ← 3
pow(n, m)	取得 n 的 m 次方值。 例：num = pow(5, 3) # num ← 125
chr(n)	取得 Unicode 編碼 n 的字元。 例：s = chr(20013) # s ← '中'
ord(s)	取得 s 字元的 Unicode 編碼值。 例：num = ord('中') # num ← 20013

上列所列舉的數值函式，是一般常用的方式。有些內建函式另有比較複雜的用法，下面以 `round()` 和 `pow()` 函式為例說明進階的用法。

## 一. `round()` 函式

`round()` 函式是以四捨六入的方式將浮點數轉換成整數值，但也可以指定要轉換的小數位數，使轉換成小數位數較小的浮點數。語法如下：

```
round(n, m)
```



### 說明

- 語法中參數 `n` 是指定被轉換的浮點數，參數 `m` 是指定轉換結果的小數位數。
  - 例：`num = round(12.365, 1)`      # `num` ← 12.4
  - 例：`num = round(12.367, 2)`      # `num` ← 12.37
  - 例：`num = round(12.364, 2)`      # `num` ← 12.36
- 若參數 `m` 省略，則預設為 0，轉換結果為整數。

## 二. `pow()` 函式

`pow()` 函式用來做指數運算，但也可以計算餘數。語法如下：

```
pow(n, m, k)
```



### 說明

- `pow()` 函式的運算意義是，將 `n` 的 `m` 次方結果再除以 `k`，然後傳回餘數。
  - 例：`mod = pow(4, 3, 10)`      # `mod` ← 4
  - 例：`mod = pow(10, 2, 9)`      # `mod` ← 1



簡例 (檔名：`divmod.py`)

假設時間是 135 分鐘，請換算為小時和分鐘數。



### 結果

```
135 分鐘換算為 2 小時 15 分鐘
```



### 程式碼

```
檔名：\ex07\divmod.py
```

```
01 total_minutes = 135
```

```
02 hours, minutes = divmod(total_minutes, 60)
```

```
03 print(f"{total_minutes} 分鐘換算為 {hours} 小時 {minutes} 分鐘")
```



## 說明

- 第 2 行：使用 `divmod()` 內建函式，將 `total_minutes` 除以 60 分，其商 `hours` 為小時數而餘數 `minutes` 為分鐘數。
- 如果不用 `divmod()` 函式，程式寫法也可以如下：

---

```
hours = int(total_minutes / 60)      # 使用 int() 函式取整數值
minutes = total_minutes % 60
```

---

## 7.2.2 random 套件函式

亂數函式主要用來產生不同的數值，通常用於統計學、模擬、離散數學、抽樣、作業研究、數值分析、決策 ... 等各領域。亂數函式的原型定義在 `random` 套件，套件中提供許多函式給使用者使用，例如 `choice()`、`randint()`、`random()` ... 等。程式若有使用亂數函式時，必須在使用前匯入 `random` 套件名稱，匯入敘述如下：

---

```
import random
```

---

套件名稱在匯入時，可以另外取較簡短或有特殊意義的別名。若要將 `random` 套件在程式匯入時取別名為 `R`，其匯入敘述如下：

---

```
import random as R
```

---

下表為常用的亂數套件函式：（簡例請參考 `funRnd.py`）

函式	使用說明
<code>randint(n1, n2)</code>	從 <code>n1</code> 到 <code>n2</code> 隨機產生一個整數，包含 <code>n1</code> 和 <code>n2</code> 。 例： <code>random.randint(1, 10)</code> # 由 1~10 產生一個整數，如：6
<code>randrange(n1, n2, n3)</code>	從 <code>n1</code> 到 <code>(n2-1)</code> 每隔 <code>n3</code> 的數，隨機產生一個整數。 例： <code>random.randrange(0, 6, 2)</code> # 由 0,2,4 產生一個整數，如：4
<code>random()</code>	從 0 到 1 之間隨機產生一個浮點數。 例： <code>random.random()</code> # 由 0.000000000000000001 ~ 0.9999999999999999 之間產生一個浮點數，如：0.834657283069456
<code>uniform(f1, f2)</code>	從 <code>f1</code> 到 <code>f2</code> 之間隨機產生一個浮點數。 例： <code>random.uniform(1, 10)</code> # 由 1.000000000000000001 ~ 9.999999999999999 之間產生一個浮點數，如：5.2934203283061023
<code>choice(s)</code>	1. 從 <code>s</code> 字串中隨機取得一個字元。 例： <code>random.choice('abc12')</code> # 由 'abc12' 中產生一個字元，如：'c' 2. 從 <code>s</code> 串列中隨機取得一個元素。

 **簡例** (檔名：randint02.py)

隨機產生 6 個 18 ~ 35 不會重複的整數。

 **結果**

```
第 1 個亂數 : 27
第 2 個亂數 : 33
第 3 個亂數 : 18
第 4 個亂數 : 28
第 5 個亂數 : 24
第 6 個亂數 : 30
```

 **程式碼**

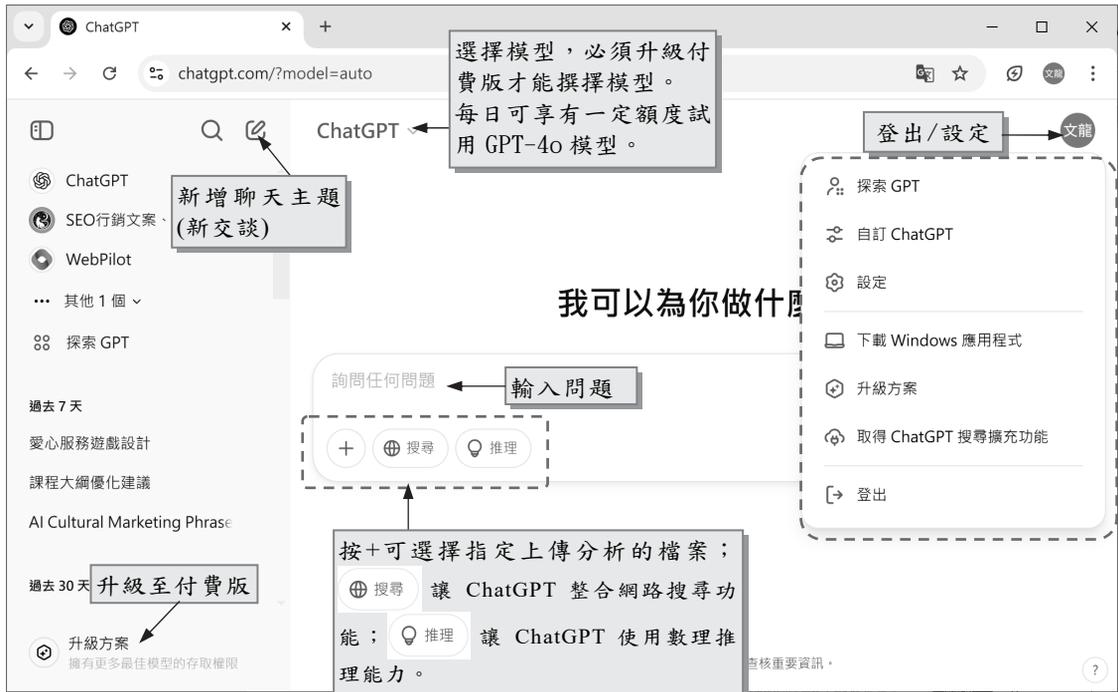
檔名：\ex07\randint02.py

```
01 import random as R
02
03 max, min, num = 35, 18, 6      # 宣告整數最大值、最小值、亂數數量
04 arr = [0 for x in range(num)] # 存放所產生的亂數
05 n = 0                          # 串列註標
06 while (n < num):
07     isRepeat = False          # 亂數沒重複
08     rnd = R.randint(min, max) # 產生一個亂數
09     if arr.count(rnd) == 0:   # rnd 在 arr 串列中元素數目為 0, 表示沒有重複
10         arr[n] = rnd          # 存放所產生的亂數到串列中
11         n += 1
12
13 for i in range(num):
14     print(f'第 {i+1} 個亂數 : {arr[i]}')
```

 **說明**

1. 第 4 行：使用 `arr` 串列來存放將產生的亂數。
2. 第 8 行：從 `min` ~ `max` 整數中取一個亂數，存放在 `rnd` 變數中。
3. 第 9~11 行：使用 `arr.count(rnd)` 函式查詢 `rnd` 在 `arr` 串列中的元素數目，如果為 0，表示該 `rnd` 數值在 `arr` 串列中還沒有出現過，所以可將 `rnd` 亂數值存放入指定註標 `n` 的串列元素中。
4. 如果使用 `sample()` 函式能產生不重複的亂數，可取代第 4 ~ 11 行敘述區段。

```
arr = R.sample(range(min,max), num)
```

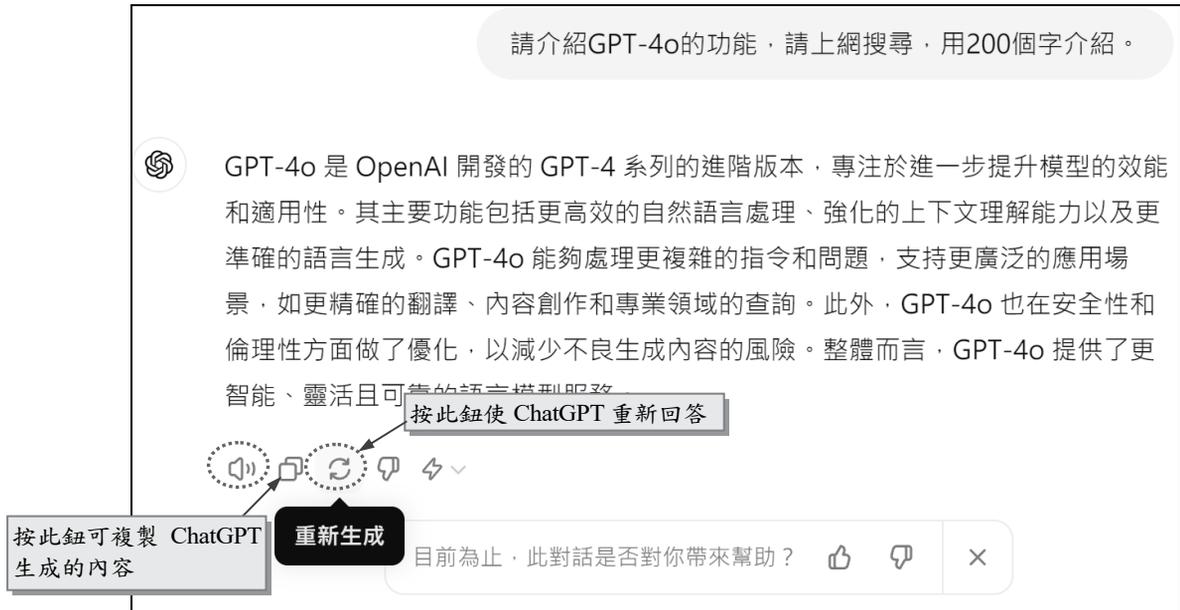


操作 ChatGPT 時應該注意下列事項，可以提高 ChatGPT 回答的正確性：

1. **新增聊天主題(新交談)**：因為 ChatGPT 會記住所有對話的內容，作為後續聊天的基礎，所以一個聊天主題中應盡量保持同一類型內容。點按左上角的  圖示鈕就可以新增一個聊天主題，聊天主題也可以重新命名或刪除。



2. **要求重新回答**：對 ChatGPT 的回答不滿意，可以點按  圖示鈕請 ChatGPT 重新回答。



3. **語音回覆**：點按  鈕，會將文字回覆結果以語音方式進行回覆。
4. **模擬角色**：若要解決問題是具有某種特殊性，請 ChatGPT 模擬角色會讓回答的效果更加精準。例如要設計 Python 課程的十八週課程大綱，可請 ChatGPT 模擬成資深教授 Python 的教師；又例如請 ChatGPT 模擬成 IG 與 Facebook 的小編，請它撰寫「人中之龍 8」遊戲的行銷標語與貼文。
5. **提供相關背景資料**：如果能夠提供人、事、時、地、物…等與問題相關的資料，可以幫助 ChatGPT 提供更具體的答案。
6. **詳細說明問題**：俗話說：「垃圾進垃圾出」，若希望 ChatGPT 的答案準確，那提問也必須要準確，提問前應該先想清楚問題的核心目標。另外，詞語的意義可能因為文化背景的不同，而有不同的含意，所以要使用明確的詞語來表達。例如「中華隊大勝韓國隊」、「中華隊大敗韓國隊」都是中華隊贏，就應該避免這類詞語。
7. **避免問題過於廣泛**：問題範圍過於廣泛會導致 ChatGPT 無法給出具體的回答，因此要縮小問題的範圍，例如將一個問題拆成幾個子問題。
8. **善用追問**：因為 ChatGPT 會記錄對答的內容，所以若要了解 ChatGPT 答案的觀點，可以繼續追問：「為什麼會這麼認為？」、「有什麼具體的證據或事實？」、「如果沒有會如何？」。另外，ChatGPT 的答案需要做部分修改時，也可以利用追問來要求修正。

9. **驗證正確性**：ChatGPT 只是一個人工智能模型，它會盡量給出有用的回答，但是不一定是正確。因此可以直接請 ChatGPT 附上數據、原理…等證明資料，或是自行運用搜尋引擎、書籍等其他管道查證。

## 10.2 ChatGPT 協作開發 Python

了解 ChatGPT 的功能和優缺點後，接著說明使用 ChatGPT 的基本步驟。

- Step 1 進入 ChatGPT 網站**：開啟瀏覽器進入 ChatGPT 網站，輸入網址：<https://chatgpt.com/>。若已登入會自行連結到 ChatGPT 使用畫面。
- Step 2 模擬角色**：輸入問題內容後按  鍵，或是按  向 ChatGPT 提問。請指定 ChatGPT 模擬「資深 Python 程式設計師」的角色，提問句為「你是一位資深的 Python 程式設計師，請您之後以此身份回答問題」。
- Step 3 提問**：使用 Python 將 [56,78,54,23,90] 陣列由小到大排序並顯示出來。本例輸入提問句為「有一整數串列 [56,78,54,23,90]，請提供 Python 程式碼將上述整數串列進行由小到大排序，最後將排序後的串列內容印出」。如下 ChatGPT 回答使用串列的 `sort()` 方法進行排序整數陣列。(ChatGPT 回答的內容可能每次都不同) (檔名：sample01.py)

有一整數串列 [56,78,54,23,90]，請提供Python程式碼將上述整數串列進行由小到大排序，最後將排序後的串列內容印出

當然可以，以下是使用 Python 將整數串列 [56, 78, 54, 23, 90] 進行由小到大排序並印出的程式碼：

```
python

numbers = [56, 78, 54, 23, 90]

# 使用 sort() 方法進行就地排序
numbers.sort()

# 印出排序後的結果
print("排序後的串列：", numbers)
```

 複製  編輯

按此鈕  
可複製程式碼

**Step 4 追問(程式改寫)**：如果不滿意可以按  鈕，請 ChatGPT 重新回答。也可以將題目再詳細說明，尤其是比較複雜的程式可分拆成幾個步驟單元分別提問。例如延續上例再輸入追問「請使用氣泡排序法排序」：(檔名：sample02.py)

請使用氣泡排序法排序

當然可以，以下是使用 **氣泡排序法 (Bubble Sort)** 將整數串列 [56, 78, 54, 23, 90] 由小到大排序的 Python 程式碼：

```
python 複製 編輯

numbers = [56, 78, 54, 23, 90]

# 氣泡排序法實作
n = len(numbers)
for i in range(n):
    for j in range(0, n - i - 1):
        if numbers[j] > numbers[j + 1]:
            # 交換位置
            numbers[j], numbers[j + 1] = numbers[j + 1], numbers[j]

# 印出排序後的結果
print("排序後的串列：", numbers)
```

**Step 5 追問(加上程式註解說明)**：若程式比較複雜看不太懂，可以請 ChatGPT 協助生成程式註解說明。例如延續上例再輸入追問「上述程式的每行後面加上程式註解說明」：(檔名：sample03.py)

**Step 6 程式除錯**：若誤打程式語言的符號，或變數名稱不一致、語法錯誤，再加上程式邏輯複雜使用肉眼很難分辨，程式便更難除錯，此時可以請 ChatGPT 協助除錯。做法是先提供程式的功能，接著在提示後面加上「請協助除錯」，同時附上錯誤程式即可。例如下圖請 ChatGPT 協助除錯的提示：「下面程式使用串列儲存使用者連續輸入的 5 個整數，再從串列中找出最大數。請協助除錯。」(錯誤程式檔名：sample04\_Err.py；正確程式檔名：sample04.py)

程式中虛線錯誤處，是筆者故意留下，主要練習請 ChatGPT 除錯：

下面程式使用串列儲存使用者連續輸入的5個整數，再從串列中找出最大數。請協助除錯。

```

lst = [0 for x in range(5)]
print('請依序輸入5個整數...')
for i in range(6):
    print('輸入第 %d 個元素內容：'%(i+1), end = ")
    lst[i] = eval(input())

max_value = lst[0]
for item in lst:
    if max < item:
        max = item

print()
print('最大值為', max)

```

圖中標註的錯誤如下：

- `range(6)`：迴圈次數正確為 5
- `eval(input())`：少一個)號
- `max_value = lst[0]`：max\_value 和 max 變數名稱不統一
- `if max < item:`：少一個：號
- `max = item`：正確為 =

取消 傳送

上面程式由 ChatGPT 除錯，結果如下圖程式除錯完成，並給予正確程式碼：

你這段程式碼邏輯大致正確，但有幾個語法與邏輯錯誤，我幫你一一指出並修正如下：

修正後的程式碼：

```

python
lst = [0 for x in range(5)]

print("請依序輸入 5 個整數...")
for i in range(5): # 錯誤在這裡：原本是 range(6)，會超出串列範圍
    print('輸入第 %d 個元素內容：' % (i + 1), end=" ")
    lst[i] = int(input()) # eval 不建議使用，改用 int()

max_value = lst[0] # 使用 max_value，而不是保留字 max
for item in lst:
    if item > max_value: # 錯誤在這裡：if max < item -> 變數名錯誤，且語法少了冒號
        max_value = item # 錯誤在這裡：== 是比較，= 才是指定

print()
print("最大值為", max_value) # 錯誤：max 是 Python 保留字

```

ChatGPT 同時提供程式修正說明：

錯誤位置	問題	修正建議
<code>range(6)</code>	超出串列長度	應為 <code>range(5)</code>
<code>eval(input())</code>	潛在安全風險	改用 <code>int(input())</code>
<code>if max &lt; item</code>	未定義變數、漏冒號	改為 <code>if item &gt; max_value:</code>
<code>max == item</code>	錯用比較符號	應為 <code>max_value = item</code>
<code>print("最大值為", max)</code>	<code>max</code> 為保留字	改用 <code>max_value</code>

Step 7 **管理聊天主題**：提問後 ChatGPT 會自動為聊天主題命名，可以在目前聊天主題項目列右側點按 `...` 鈕由選單點按 `重新命名` 鈕修改聊天名稱，也可以點按 `刪除` 鈕刪除聊天主題。若要新增聊天的主題時，可以點按左上角的 `+` 鈕。

