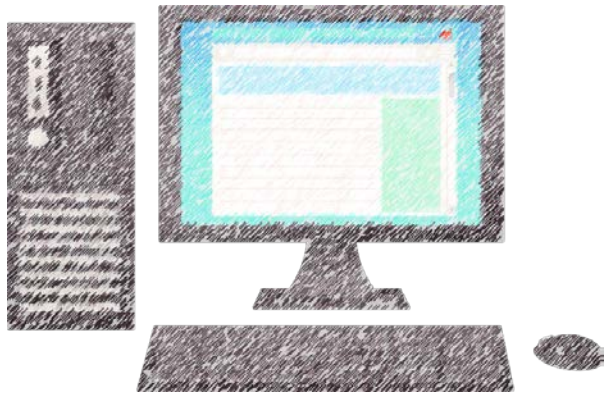
桃園市政府勞動局 112 年勞工學苑產業應用班
Python 程式設計：從零基礎入門到進階

第 7 單元

流程控制



林柏江老師
元智大學 電機工程學系 助理教授
pclin@saturn.yzu.edu.tw

預計課程進度

週次	日期	上午課程內容 (09:00 ~ 12:00)	下午課程內容 (13:00 ~ 16:00)
1	2023/07/23	01. 運算思維簡介	02. Python 快速上手
2	2023/07/30	03. Python 基礎	04. Python 基本資料結構
3	2023/08/06	05. 字串	06. 字典
4	2023/08/13	07. 流程控制	08. 函式
5	2023/08/20	09. 模組與作用域	10. Python 程式檔
6	2023/08/27	11. 檔案系統的使用與檔案讀寫	12. 例外處理
7	2023/09/03	13. 類別與物件導向程式設計	14.初探資料結構與演算法
8	2023/09/10	15. 陣列	16. 鏈結串列
9	2023/09/17	17. 堆疊與佇列	18. 圖形結構
	2023/09/24, 2023/10/01, 2023/10/08 (共三周) 放假		
10	2023/10/15	19. 樹狀結構	20. 分治法
11	2023/10/22	21. 動態規劃	22. 貪婪演算法
12	2023/10/29	23. 回溯	24. 分支定界法

課程大綱

1. `while` 迴圈
2. `if-elif-else` 判斷式
3. `for` 迴圈
4. list 以及 dictionary 的生成式
5. 敘述、區塊和縮排
6. 布林值與運算式
7. 寫一個簡單的程式來分析文字檔

課程大綱

1. `while` 迴圈
2. `if-elif-else` 判斷式
3. `for` 迴圈
4. list 以及 dictionary 的生成式
5. 敘述、區塊和縮排
6. 布林值與運算式
7. 寫一個簡單的程式來分析文字檔

while 迴圈範例

```
num = int(input("Enter a positive integer to calculate its factorial: "))
result = 1
i = 1

while i <= num:
    result = result * i
    i += 1

print(f"{num}! = {result}")
```

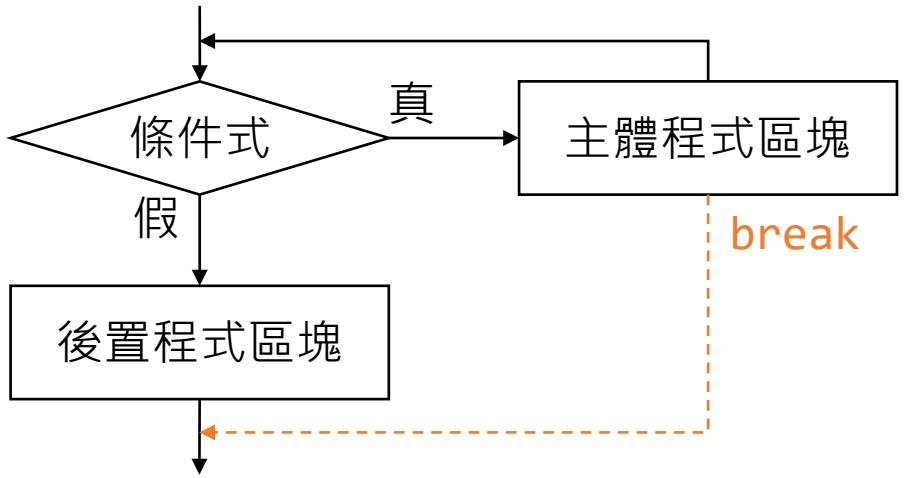
主體程式區塊的敘述必須縮排，且處於相同的縮排級別。

```
Enter a positive integer to calculate its factorial: 5
5! = 120
```

完整的 while 迴圈語法

```
while 條件式：
    主體程式區塊
else:
    後置程式區塊
```

非必要，且不常使用



其中，條件式 (condition) 是一個布林運算式，運算結果為 True 或 False。
程式區塊的敘述必須縮排，且處於相同的縮排級別。

- 當條件式結果為 True 時，主體程式區塊就會執行，然後回到條件式。
- 當條件式結果為 False 時，while 迴圈會執行一次後置程式區塊，然後終止迴圈。
- 如果條件式一開始就是 False，主體程式區塊一次也不會執行，只會執行一次後置程式區塊，然後終止迴圈。

while 迴圈裡的 break 敘述

```
num = 10
i = 0

while i <= num:
    if i == 5:
        break
    print(i)
    i += 1
else:
    print('else')

print('end')
```

break 敘述可用在 while 迴圈的主體程式區塊中。執行 break 敘述會立即中止 while 迴圈，並且 (若有 else 子句的話) 不執行後置程式區塊。

執行結果

```
0
1
2
3
4
end
```

while 迴圈裡的 continue 敘述

```
num = 10
i = 0

while i <= num:
    i += 1
    if 3 <= i <= 9:
        continue
    print(i)
else:
    print('else')

print('end')
```

continue 敘述可用在 while 迴圈的主體程式區塊中。
執行 continue 敘述會跳過主體程式區塊中剩餘的敘述，回到條件式。

執行結果

```
1
2
10
11
else
end
```

while 迴圈的 else 子句

如果主體程式區塊裡沒有 break 敘述，
以下兩個迴圈的效果是一樣的。
右邊的寫法更容易理解。

```
while 條件式：
    主體程式區塊
else:
    後置程式區塊
```

```
while 條件式：
    主體程式區塊
後置程式區塊
```

課程大綱

1. `while` 迴圈
2. `if-elif-else` 判斷式
3. `for` 迴圈
4. list 以及 dictionary 的生成式
5. 敘述、區塊和縮排
6. 布林值與運算式
7. 寫一個簡單的程式來分析文字檔

if-elif-else 判斷式的語法

```
if 條件式1：
    程式區塊1
elif 條件式2：
    程式區塊2
elif 條件式3：
    程式區塊3
...
...
elif 條件式(n-1)：
    程式區塊(n-1)
else:
    程式區塊n
```

非必要，可省略

其中，每個條件式 (condition) 都是一個布林運算式，運算結果為 True 或是 False。

程式區塊的敘述必須縮排，且處於相同的縮排級別。

- 若條件式1為 True，執行程式區塊1。
- 否則，若條件式2為 True，執行程式區塊2。
- 依此類推。
- 若沒有條件式為 True，則執行 else 子句的程式區塊n。

簡潔易懂的單行 if 條件運算式

常見的 Python 程式碼片段：

```
if n > 100:
    result = "success"
else:
    result = "failed"
```

上面程式碼可以改成以下的單行 if 條件運算式：

```
result = "success" if n > 100 else "failed"
```

單行 if 條件運算式除了上述用來設定變數之外，
也可以用在其他程式敘述中：

```
print("success" if n > 100 else "failed")
```

if 子句的程式區塊

if 子句的程式區塊是必要的。
但是可以使用 pass 敘述來跳過，
不做任何動作。

```
if x < 5:
    pass
else:
    x = 5
```

課程大綱

1. `while` 迴圈
2. `if-elif-else` 判斷式
3. `for` 迴圈
4. list 以及 dictionary 的生成式
5. 敘述、區塊和縮排
6. 布林值與運算式
7. 寫一個簡單的程式來分析文字檔

for 迴圈

- Python 的 for 迴圈不需要索引，而是直接走訪物件中的元素，逐一取出來做處理。
- 可走訪物件包括 list、tuple、set、字串、產生器、enumerate() 等等。
- for 迴圈的語法如下：

```
for item in 可走訪物件：
    主體程式區塊
else:
    後置程式區塊
```

} 非必要，且不常使用

- 每讀取一次可走訪物件的元素，指派給 item，就會執行一次主體程式區塊。
- 直到可走訪物件的所有元素都走訪過了，就會結束迴圈。

for 迴圈的範例

以下範例會印出 x 中每個數字的倒數。

```
x = [1.0, 2.0, 3.0]
for n in x:
    print(1 / n)
```

執行結果

```
1.0
0.5
0.3333333333333333
```

range() 函式

- 若想要使用索引來進行 for 迴圈的走訪，可使用 range() 函式搭配 len() 函式來產生索引。
- range(n) 會依序回傳 0, 1, 2, …, n-1，總共 n 個數字。

```
x = [1, 3, -7, 4, 9, -5, 4]
for i in range(len(x)):
    if x[i] < 0:
        print("Found a negative number at index ", i)
```

執行結果

```
Found a negative number at index  2
Found a negative number at index  5
```

以 range(m, n) 控制數列範圍

range(m, n) 會產生 m 到 n-1 的數列，但無法產生反向數列。

```
>>> list(range(3, 7))
[3, 4, 5, 6]
>>> list(range(2, 10))
[2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(5, 3))
[]
```

可使用 range() 的第三個參數來指定遞增量。
若指定遞增量為負值，數列就會反向由大到小排列。

```
>>> list(range(0, 10, 2))
[0, 2, 4, 6, 8]
>>> list(range(5, 0, -1))
[5, 4, 3, 2, 1]
```

在 for 迴圈中使用 break 和 continue

- break 和 continue 敘述可用在 for 迴圈的主體程式區塊中。
- 執行 break 敘述會立即中止 for 迴圈，並且 (若有 else 子句的話) 不執行後置程式區塊。
- 執行 continue 敘述會跳過主體程式區塊中剩餘的敘述，繼續執行迴圈。

for 迴圈與 tuple 解包多重設定變數

以下範例程式會計算每個 tuple 中兩個數字的乘積，並計算出所有乘積的總和。

```
somelist = [(1, 2), (3, 7), (9, 5)]
result = 0
for t in somelist:
    result = result + (t[0] * t[1])
```

這個 list 裡的元素是 tuple。

逐一取出各個 tuple。

以下是建議寫法，更清楚易懂。

```
somelist = [(1, 2), (3, 7), (9, 5)]
result = 0
for num1, num2 in somelist:
    result = result + (num1 * num2)
```

num1、num2 是一個 tuple，也就是 (num1, num2)。

從 somelist 取出一個 tuple 後會解包，然後分別設定給 num1、num2 這兩個變數。

enumerate() 函式

enumerate() 函式用來把 list、tuple 內的元素取出，
並加上該元素的索引編號。

```
>>> x = ['a', 'b', 'c']
>>> enumerate(x)
<enumerate object at 0x0000022F50AD7A40>
>>> list(enumerate(x))
[(0, 'a'), (1, 'b'), (2, 'c')]
```

enumerate() 函式會回傳一個 enumerate 物件。

使用 list() 函式轉成 list 物件，才能看到內容。

enumerate() 函式運用於 for 迴圈

若 for 迴圈同時需要索引與元素值，可用 enumerate()，
不需要自行產生索引編號。

```
x = [1, 3, -7, 4, 9, -5, 4]
for i, n in enumerate(x):
    if n < 0:
        print("Found a negative number at index ", i)
```

執行結果

```
Found a negative number at index  2
Found a negative number at index  5
```

zip() 函式

zip() 函式用來把兩個可走訪的物件結合起來。

```
>>> x = [1, 2, 3, 4]
>>> y = ['a', 'b', 'c']
>>> z = zip(x, y)
>>> list(z)
[(1, 'a'), (2, 'b'), (3, 'c')]
```

zip() 函式結合物件時，以長度比較短的物件為準，超過的會被捨棄。

課程大綱

1. `while` 迴圈
2. `if-elif-else` 判斷式
3. `for` 迴圈
4. list 以及 dictionary 的生成式
5. 敘述、區塊和縮排
6. 布林值與運算式
7. 寫一個簡單的程式來分析文字檔

為什麼需要生成式

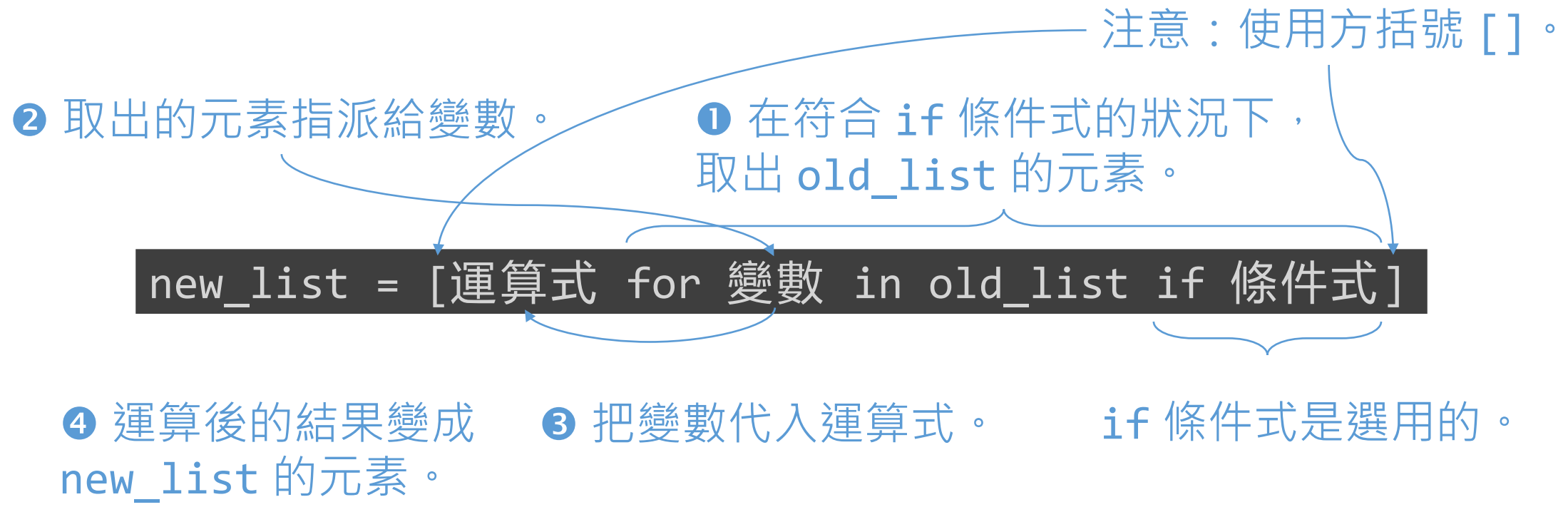
- for 迴圈的一種常見應用：把 list 中的元素取出來進行運算，產生新的 list。
- 以下範例把 list x 中的元素平方後建立新的 list x_squared。

```
>>> x = [1, 2, 3, 4]
>>> x_squared = []
>>> for item in x:
...     x_squared.append(item * item)
...
>>> x_squared
[1, 4, 9, 16]
```

- 為了這種情況，Python 建立了快捷語法，稱為生成式 (comprehension)。

list 生成式

list 生成式的語法如下：



list 生成式的範例

原本的寫法：

```
>>> x = [1, 2, 3, 4]
>>> x_squared = []
>>> for item in x:
...     x_squared.append(item * item)
...
>>> x_squared
[1, 4, 9, 16]
```

使用 list 生成式的寫法：

```
>>> x = [1, 2, 3, 4]
>>> x_squared = [item * item for item in x]
>>> x_squared
[1, 4, 9, 16]
```

list 生成式加上 `if` 條件式的範例

```
>>> x = [1, 2, 3, 4]
>>> x_squared = [item * item for item in x if item > 2]
>>> x_squared
[9, 16]
```

字典生成式

字典生成式的語法如下：

與 list 生成式的差別 1：
使用大括號 {}。

```
new_dict = {運算式1:運算式2 for 變數 in old_list if 條件式}
```

與 list 生成式的差別 2：
使用兩個運算式來產生 key:value。

字典生成式的範例

```
>>> x = [1, 2, 3, 4]
>>> x_squared_dict = {item: item * item for item in x}
>>> x_squared_dict
{1: 1, 2: 4, 3: 9, 4: 16}
```

產生器 (Generator)

- 產生器 (Generator) 的語法跟 list 生成式很像。
- 差別在於使用了小括號 ( ) 來代替方括號 [ ]。
- list 生成式會建立一個新的 list。
- 產生器會建立一個產生器物件，可以用 for 迴圈來走訪這個產生器物件。

```
>>> x = [1, 2, 3, 4]
>>> x_squared = (item * item for item in x)
>>> x_squared
<generator object <genexpr> at 0x102176708>
>>> for square in x_squared:
...     print(square,)
...
1 4 9 16
```

課程大綱

1. `while` 迴圈
2. `if-elif-else` 判斷式
3. `for` 迴圈
4. list 以及 dictionary 的生成式
5. 敘述、區塊和縮排
6. 布林值與運算式
7. 寫一個簡單的程式來分析文字檔

區塊與縮排

- Python 使用縮排來區分區塊結構，每個區塊由一行或多行敘述所組成。
- 多個敘述可以放在同一行，使用分號隔開。
- 若流程控制的程式區塊內的程式很短，也可以放在冒號後的同一行上。

```
>>> x = 1; y = 0; z = 0
>>> if x > 0: y = 1; z = 10
... else: y = -1
...
>>> print(x, y, z)
1 1 10
```

縮排錯誤：不該縮排卻縮排

```
>>>      x = 1
  File "<stdin>", line 1
    x = 1
IndentationError: unexpected indent
```

縮排錯誤：縮排不一致

```
>>> x = 1
>>> if x == 1:
...         y = 2
...     z = 2
  File "<stdin>", line 3
    z = 2
        ^
IndentationError: unindent does not match any outer indentation level
```

把一行敘述拆成多行

- 可用反斜線 \ 放在行尾，Python 會把此行與下一行連成同一行。
- Python 會把 ()、[]、{} 等括號內所有敘述視為同一個敘述，所以在括號內可以任意換行。

```
>>> print('string1', 'string2', 'string3' \
...     , 'string4', 'string5')
string1 string2 string3 string4 string5
>>> x = 100 + 200 + 300 \
...     + 400 + 500
>>> x
1500
>>> v = [100, 300, 500, 700, 900,
...     1100, 1300]
>>> v
[100, 300, 500, 700, 900, 1100, 1300]
>>> max(1000, 300, 500,
...         800, 1200)
1200
>>> x = (100 + 200 + 300
...             + 400 + 500)
>>> x
1500
```

課程大綱

1. `while` 迴圈
2. `if-elif-else` 判斷式
3. `for` 迴圈
4. list 以及 dictionary 的生成式
5. 敘述、區塊和縮排
6. 布林值與運算式
7. 寫一個簡單的程式來分析文字檔

大多數 Python 物件都可以當作布林值

- 流程控制的條件式運算結果都是布林值。
- Python 的布林值只有 `True` 跟 `False` 兩種值。
- 具有布林運算的任何運算式都只會回傳 `True` 或 `False`。
- 在 Python 中，`0` 或空值為 `False`，任何其他值為 `True`。
  - 數字 `0`、`0.0` 和 `0+0j` 均為 `False`；任何其他數字都為 `True`。
  - 空字串 `""` 為 `False`；任何其他字串都為 `True`。
  - 空 list `[]` 為 `False`；任何其他 list 都為 `True`。
  - 空 tuple `()` 為 `False`；任何其他 tuple 都為 `True`。
  - 空字典 `{}` 為 `False`；任何其他字典都為 `True`。
  - None 為 `False`。

布林算符與比較算符

- 比較算符：
  - `<`
  - `<=`
  - `>`
  - `>=`
  - `==`
  - `!=`
- `in` 以及 `not in`：用來測試序列 (list、tuple、字串、字典) 中的成員資格。
- `is` 以及 `is not`：用來測試兩個物件是否為同一個。
- `and`、`or`、`not`

and 以及 or 算符的回傳

- Python 的 and 以及 or 算符並不會回傳運算後的布林值，而是會回傳物件。
- and 算符會回傳：
  - 運算式中第一個運算結果為 False 的物件，或是
  - 最後一個物件
- or 算符會回傳：
  - 運算式中第一個運算結果為 True 的物件，或是
  - 最後一個物件

```
>>> [2] and [3, 4]
[3, 4]
>>> [] and 5
[]
>>> [2] or [3, 4]
[2]
>>> [] or 5
5
```

== 、 != 以及 is 、 is not 算符

== 以及 != 算符用來測試兩邊是否為相同的值。

is 以及 is not 算符用來測試兩邊是否為同一個物件。

```
>>> x = [0]
>>> y = [x, 1]
>>> x is y[0]
True
>>> x = [0]
>>> x is y[0]
False
>>> x == y[0]
True
```

x 以及 y[0] 參照到同一個物件。

x 改參照到另一個新建立的物件 [0]。

x 以及 y[0] 已參照到不同物件。

x 以及 y[0] 參照到的值相同。

課程大綱

1. `while` 迴圈
2. `if-elif-else` 判斷式
3. `for` 迴圈
4. list 以及 dictionary 的生成式
5. 敘述、區塊和縮排
6. 布林值與運算式
7. 寫一個簡單的程式來分析文字檔

計算檔案中的行數、字數、字元數

```
infile = open('word_count.tst')
lines = infile.read().split("\n")

line_count = len(lines)
word_count = 0
char_count = 0

for line in lines:
    words = line.split()
    word_count += len(words)
    char_count += len(line)

print("File has {0} lines, {1} words, {2} characters".format(
      line_count, word_count, char_count))
```

```
python word_count.py
File has 4 lines, 30 words, 189 characters
```

重點整理

- `while` 迴圈可以在特定條件下重複執行程式碼。
- `if-elif-else` 敘述用來判斷在不同條件下執行不同的程式碼。
- 使用 `for` 迴圈可以把可走訪物件中的元素逐一取出來做處理。
- `break` 會立即中止迴圈，`continue` 則會跳過後面的程式，回到條件式繼續執行迴圈。
- list 和字典生成式是 Python 中很常用的語法。

重點整理 (續)

- 下表比較 range()、enumerate()、zip() 函式：

函式	說明	使用時機
range()	產生等差數列	想要在 for 迴圈用索引走訪物件
enumerator()	把 list、tuple 內的索引與元素同時取出	for 走訪物件時需要同時取出索引
zip()	把兩個可走訪物件結合起來	兩個 list 依序將其值做對應

- 下表比較了生成式與產生器的異同：

	相同	不同
生成式	把 list 或字典中的元素取出修改	回傳新的 list 或字典
產生器		回傳可走訪的產生器物件