【南區Fintech研習營】Python 程式設計基礎: Google finance股價爬蟲應用

講者：林萍珍

第六章自訂函數

6-1　自訂函數的定義、特點與語法

6-2　呼叫函數與回傳值用法

6-3　參數

6-4　變數命名空間的搜尋路徑

6-5　除錯

自訂函數的定義、特點與語法

- 函數是依特定規則撰寫且能夠重複使用的程式碼，可以設計成一個單一或一組相 互關連指令集的程式碼。善用自訂函數的寫作技巧，提升程式的模組化，來提高程式 碼重複使用的效率。
- 自訂函數可以依照使用者自己的 需求設計特定的功能，自訂函數又稱使用者定義 的函數（user-defined functions）。
- 使用者若要客製化某個內建函數不提供的功能，就 必須自己動手撰寫自訂函數。

自訂函數的特點

- 使用者可以依需要的功能設計自訂函數。
- 自訂函數程式碼區域內的第一個保留字是def，後面接函數名稱、小括弧「( )」、括弧內的參數群「(parameters)」以及最後以冒號「:」結尾。
- 所有的傳入參數必須放在此括弧內。
- 自訂函數內的第1列可以放註解，指令區要內縮對齊。
- 回傳指令(return)負責將指定變數的結果或運算式回傳給呼叫函數的程式。
- 自訂函數的名稱盡量避免與Python保留字、內建函數。

自訂函數語法

```
def 函數名稱 ( 傳入參數 ):
# 註解說明函數用法、功能
函數主體，撰寫指令集
return ( 變數名稱或指令的運算式 )
```

範例 6-1 輸入一個正整數 n，從 1 累加到 n，使用函數的寫法。

示範程式碼

```
1   #E_6_1.py 功能 : 主程式呼叫 1 層自訂函數
2   def sumnfunc(n):
3   #A_func 功能 : 計算累加
4       sumn=0
5       for i in range(n+1):
6           sumn=sumn+i
7       return(sumn)
8   num=int(input(' 輸入一個正整數 n = '))
9   result=sumnfunc(num)
10   print('1 累加到 n 的結果 = ', result)
```

程式執行步驟

步驟 1 程式執行進入點

本範例「程式執行進入點」是第 8 列請對照圖 6-2。使用者由鍵盤輸入的正整數 n 會儲存在 num 變數中。

步驟 2 主程式呼叫自訂函數

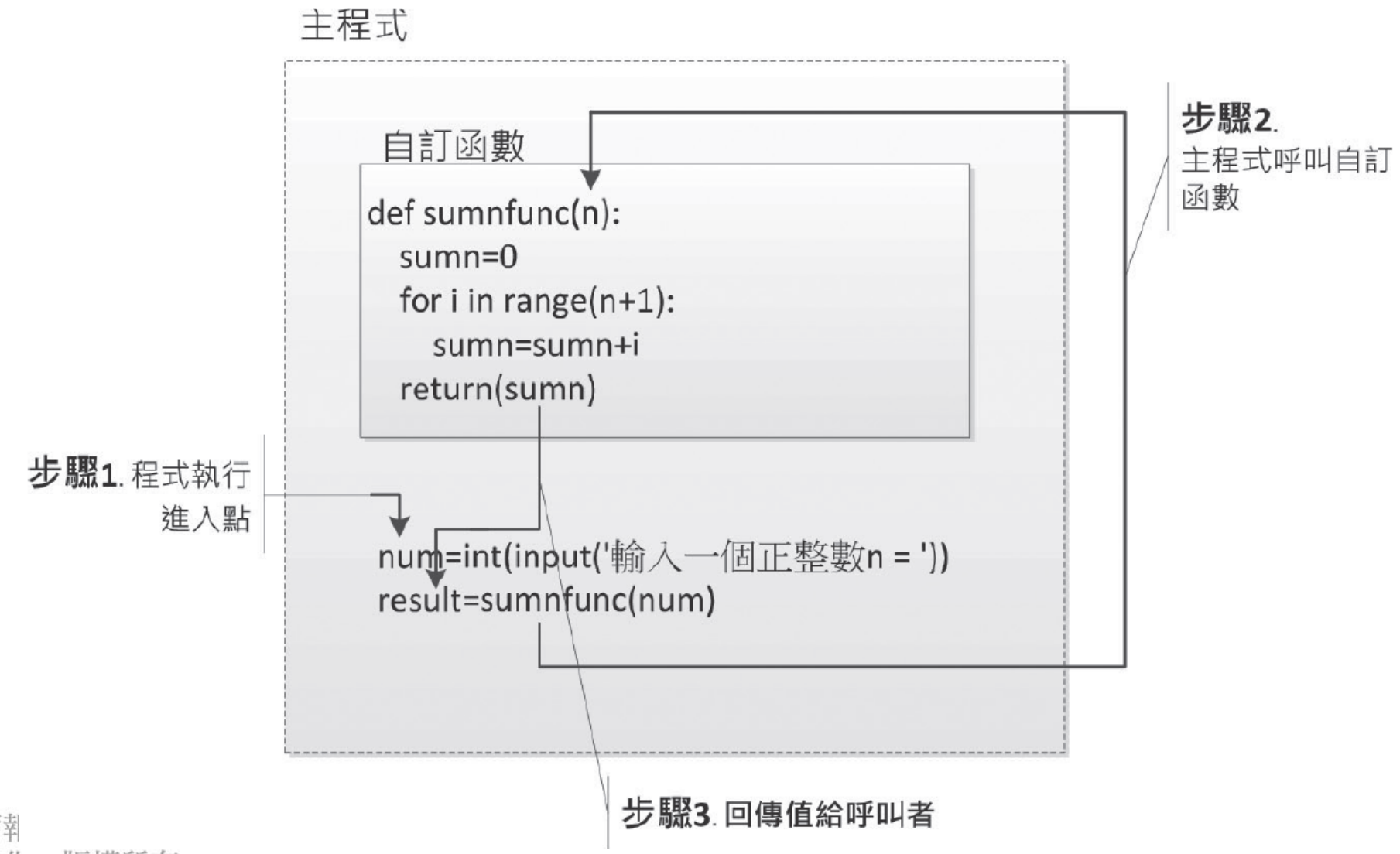
主程式呼叫自訂函數，將主程式的變數 num 的值指派給被呼叫函數中的 變數 n。由主程式呼叫 sumnfunc 自訂函數（見第 9 列），小括弧內代入 num，傳入給自訂函數，sumnfunc 有定義一個傳入變數 n 來接受主程式傳進來的值，假設 num 的值是 5，則 n 也會被指派 5 的值見第 1 列請對照圖 6-2。執行自訂函數內容到 return, 會將函數內變數 sumn 的值傳回給主程式取用。

程式執行步驟

步驟 3 回傳值給呼叫者

自訂函數執行完畢時，若有回傳值則將指定的變數回傳給呼叫者，本範例呼叫者是主程式，則回傳給主程式指定的變數 result 接受回傳值（見第 9 列），最後印出累加結果（見第 10 列）。即主程式用變數 result 來承接回傳的值。

函數呼叫



執行結果

```
1   輸入一個正整數 n = 10
2   1 累加到 n 的結果 = 55
```

結果說明

使用者若從鍵盤輸入 10，即會將 10 的內容值傳給自訂函數算完 1 到 10 的累加後回傳給呼叫者印出見執行結果 1 到 2 列。

自訂函數呼叫自訂函數

自訂函數內部也可以呼叫另一個自訂函數，可以依程式架構的需求無限伸展。若考慮程式的可讀性以及大程式的維護修改需要，不宜太多層以免過於複雜。以下示範多層的自訂函數呼叫自訂函數的架構見圖 6-3。主程式（Main program）呼叫第 callfunc 自訂函數，callfunc 再呼叫 sumnfunc 與 prodnfunc 自訂函數。

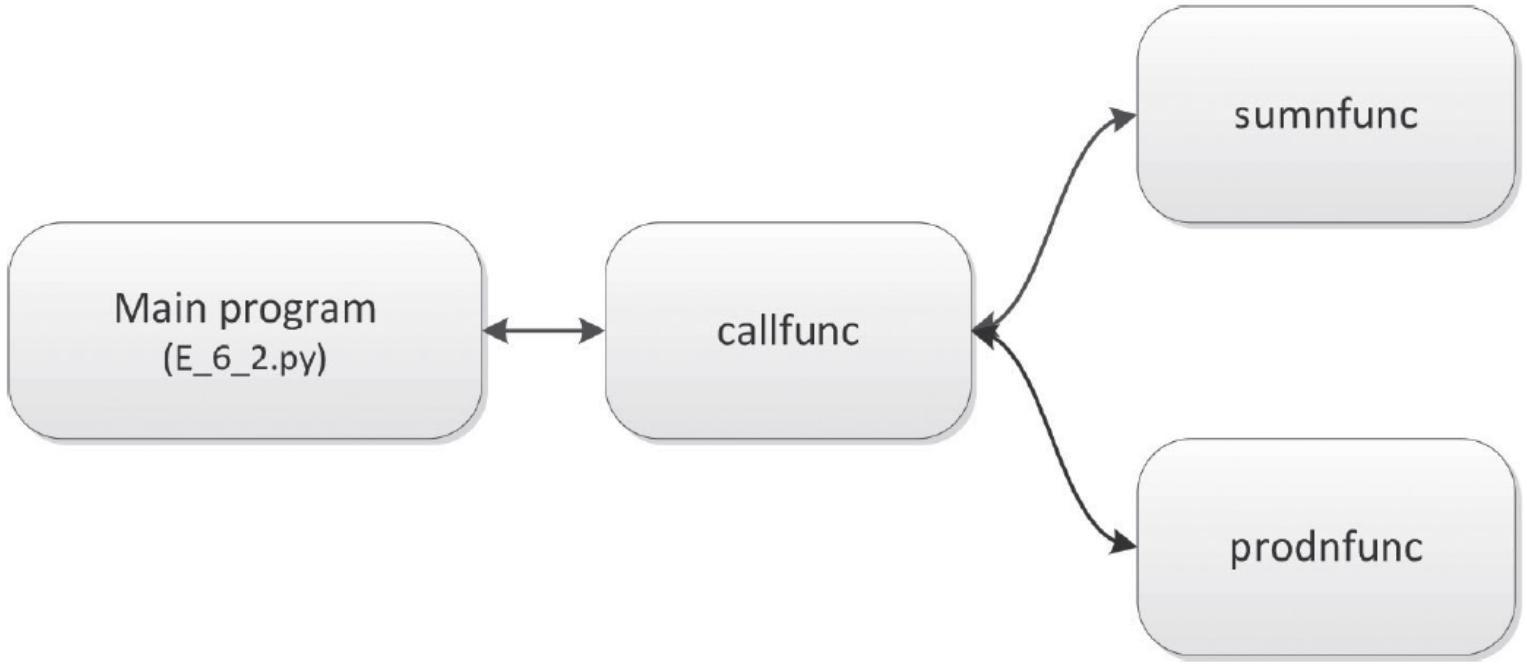


圖 6-3　多層的自訂函數呼叫自訂函數示意圖

範例 6-2 輸入一個正整數 n，計算從 1 到 n 的累加及累乘，使用自訂函數呼叫自訂函數的寫法。

示範程式碼

```
1   #E_6_2.py 功能 : 主程式呼叫 2 層的自訂函數
2   def callfunc(n):
3   #callfunc 功能 : 檢查數值型別與呼叫函數
4       if n.isdigit():
5           number=int(n)
6           print('1 累加到 n 的結果 =', sumnfunc(number))
```

傳址與傳值

呼叫程式與被呼叫的函數之間，其參數的傳遞方式分為傳址（call by reference）與傳值（call by value）。

- 傳址：傳入參數在參考的記憶體位址，函數內的對此參數進行修改，會影響原參數的內容值。因為，傳入者與接收者同一個記憶體位址。
- 傳值：複製一份傳入參數的內容值，給函數內負責接收的參數，兩者是相互獨立的參數，不同記憶體位址，若函數內的對此參數進行修改，不會影響原參數的內容值。

在 Python 程式語言，傳址或傳值與資料型別的可變與不可變結合，即若是純量（scalar）如整數與浮點數、布林或字串等（見第 3 章圖 3-1），其值是不可變的，其值與函數內的參數值是相互獨立的，會將原始參數值複製一份給呼叫的函數，所以即使函數內的參數值被改變，原始值仍不受影響，屬於「傳值」的參數傳遞方式。

提示

純量指某一個變數的值是單一個值，在 Python 中整數、浮點數、字串與布林都是純量的型別。

提示

不可變的資料型別屬傳值的參數傳遞方式。

可變的資料型別包含 list, set, dict 三種，則其參數的傳遞方式是將參數的記憶體位址傳給呼叫的函數，屬於「傳址」的參數傳遞方式。函數內參數的的記憶體位址是與呼叫者程式共用，因此函數的參數其內容值改變，連帶改變原始傳入的參數值。

提示

可變的資料型別屬於傳址的參數傳遞方式。

傳址的操作範例

範例 6-4 傳址的操作範例

以 list 資料型別為傳遞標的，試範可變型別使用傳址方式傳遞參數。

示範程式碼

```
1   #E_6_4.py 功能 : 傳址的操作範例
2   def callbyreference(number):
3   #callbyreference 函數功能 : 計算平均新增到傳入參數之後
4       n=len(number)
5       meanv=sum(number)/n
6       number.append(meanv)
7       return number
8   sample=[5,8,9,6,4,1,5,3,6,2]
9   print(' 呼叫函數前的原始值 ', sample)
10   print(' 呼叫函數後的內容值 ', callbyreference(sample))
11   print(' 呼叫函數後的原始值也會被改變 ', sample)
```

程式說明

步驟 1 初始值與呼叫函數

程式進入點在第 8 列設定 sample 串列的初始值，先印出呼叫函數前的原始值（見第 9 列）。呼叫 callbyreference 自訂函數並代入 sample 串列為參數，其回 傳值以 print 函數印出（見第 10 列）。

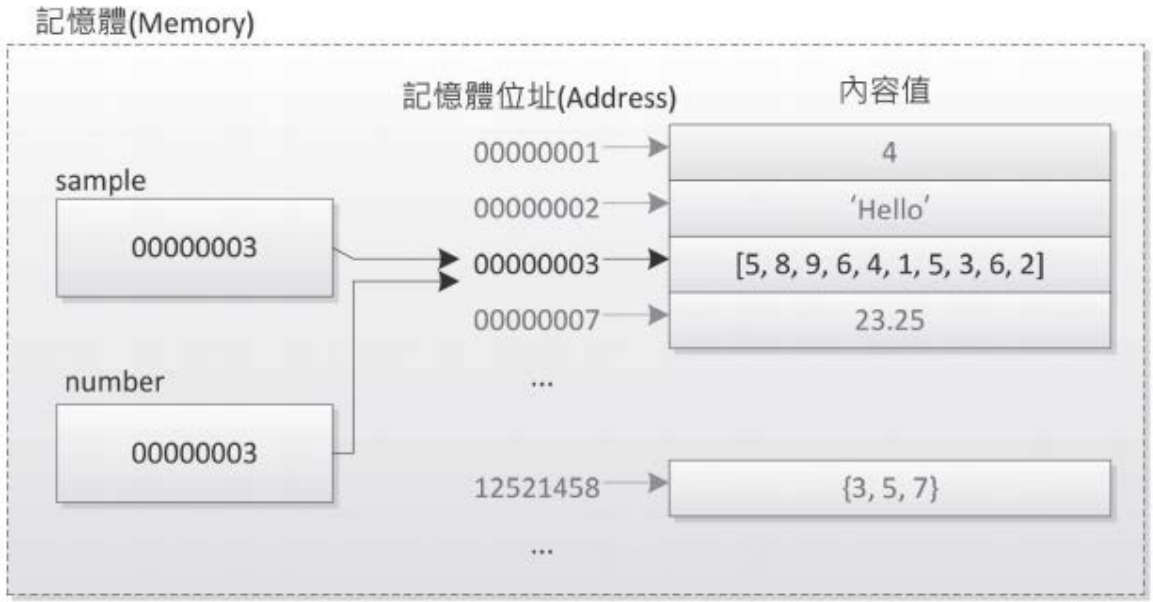


圖 6-8　傳址方式的記憶體位址參考示意圖

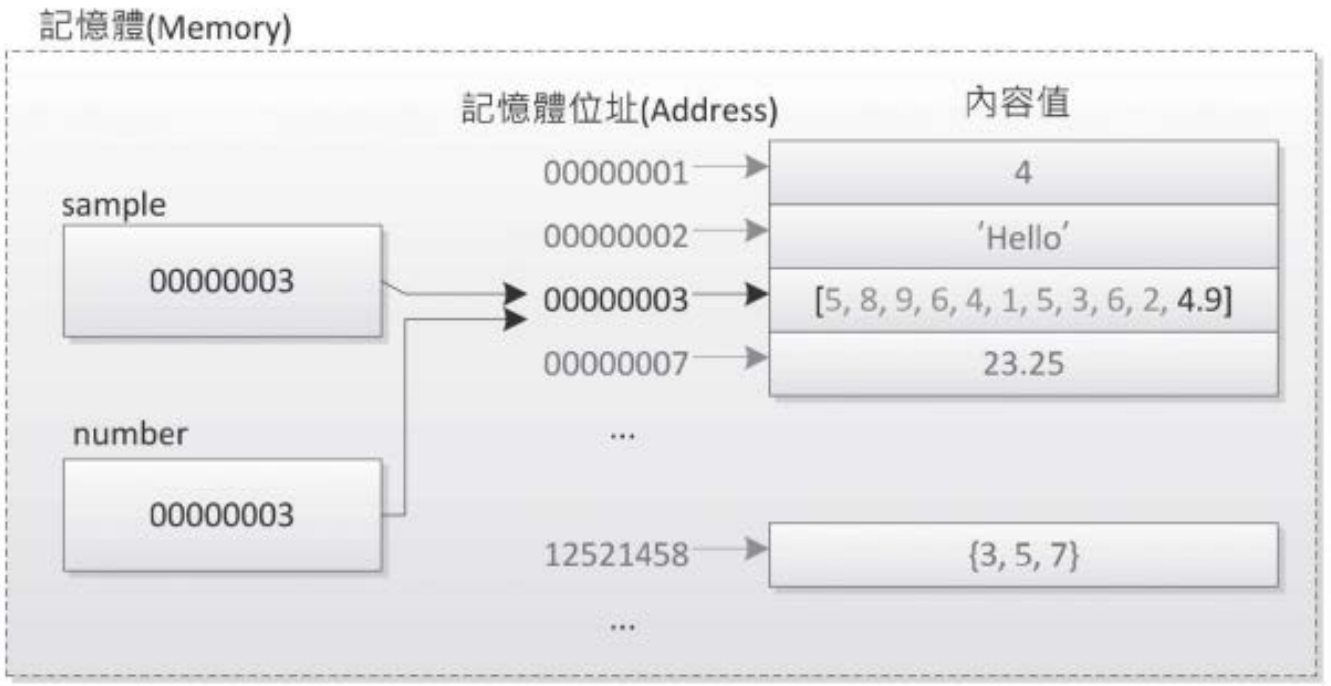


圖 6-9　內容改變後的傳址參數示意圖

傳值的操作範例

範例 6-5 傳值的操作範例

以 int 資料型別為傳遞標的，試範不可變型別使用傳值方式傳遞參數。函數功能是傳入參數乘上自己後回傳。

示範程式碼

```
1  #E_6_5.py 功能 : 傳值的操作範例
2  def callbyvalue(number):
3  #callbyvalue 函數功能 : 計算參數乘上自己後回傳
4      number*=number
5      return number
6  sample=4
7  print(' 呼叫函數前的原始值 ', sample)
8  print(' 呼叫函數後的內容值 ', callbyvalue(sample))
9  print(' 呼叫函數前的原始值不會被改變 ', sample)
```

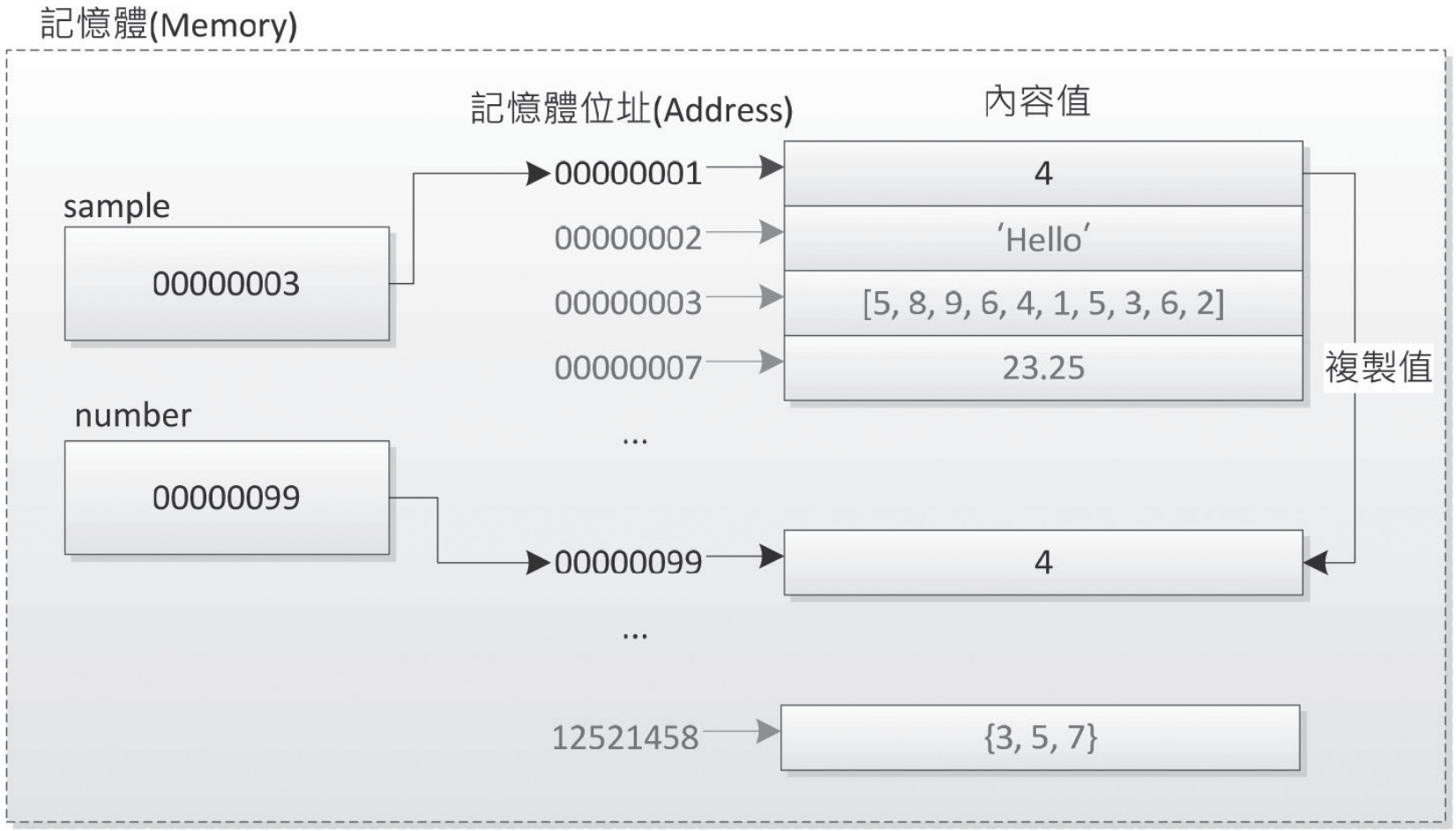


圖6-10　傳值方式之複製值

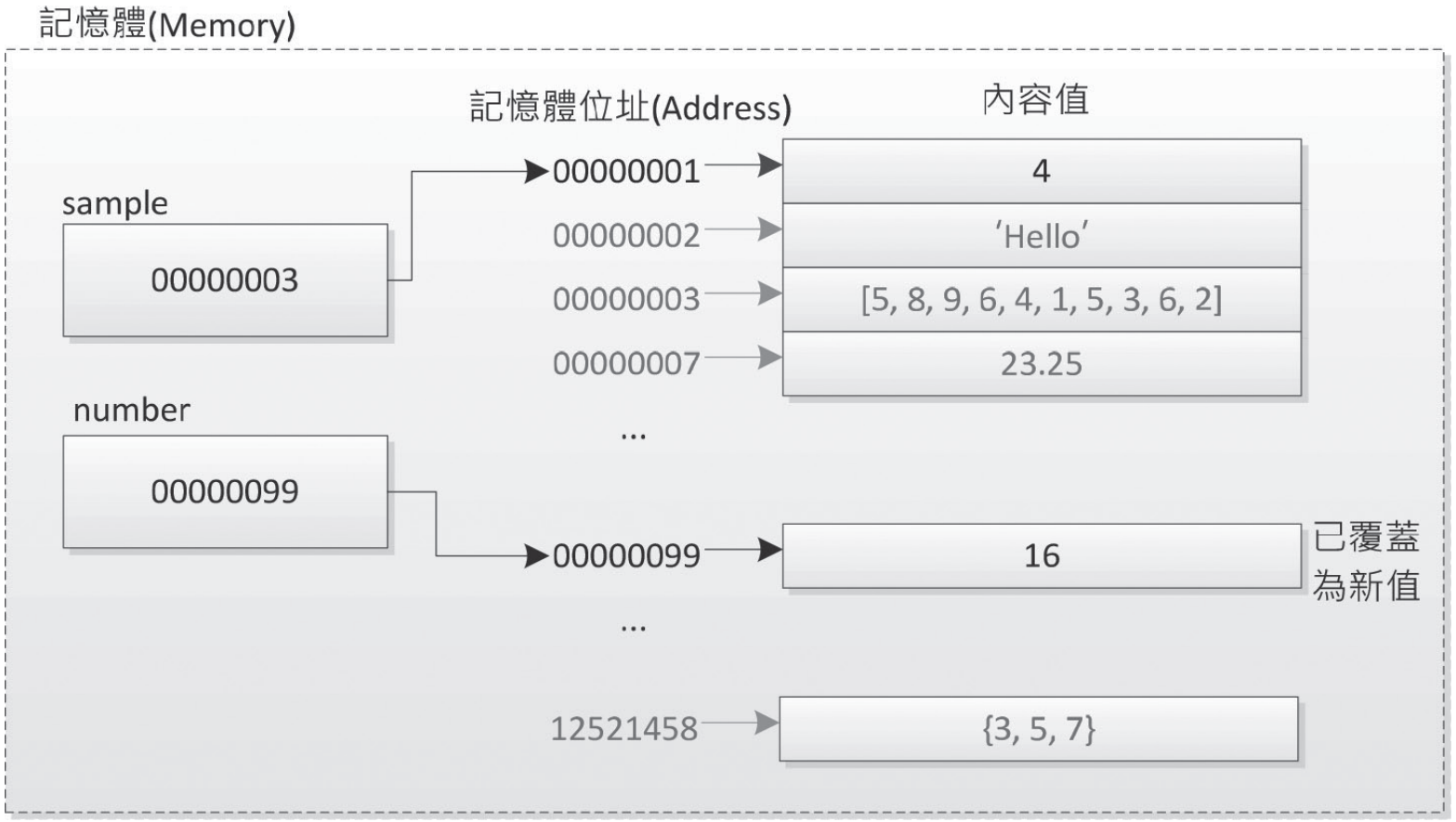


圖6-11　傳值方式之覆蓋新值

參數用法介紹

1. **實際參數**：函數呼叫者將實際資料傳給函數。即呼叫函數時從外界傳給函數所會使用的參數。見以下指令，詳見【範例 6-6】。

```
result=polyequ(sumple)
```

2. **形式參數**：定義函數時寫在函數名稱之後的就是形式參數，是定義函數時的一部分，功能是接收實際參數的主要窗口。見以下指令，詳見【範例 6-6】。

```
def polyequ (x):
```

實際參數與形式參數的對照圖（見圖 6-12）。sample 是實際參數，內容值是整數 4 型別；x 是形式參數，其值是複製自 sample，內容值也是整數 4。

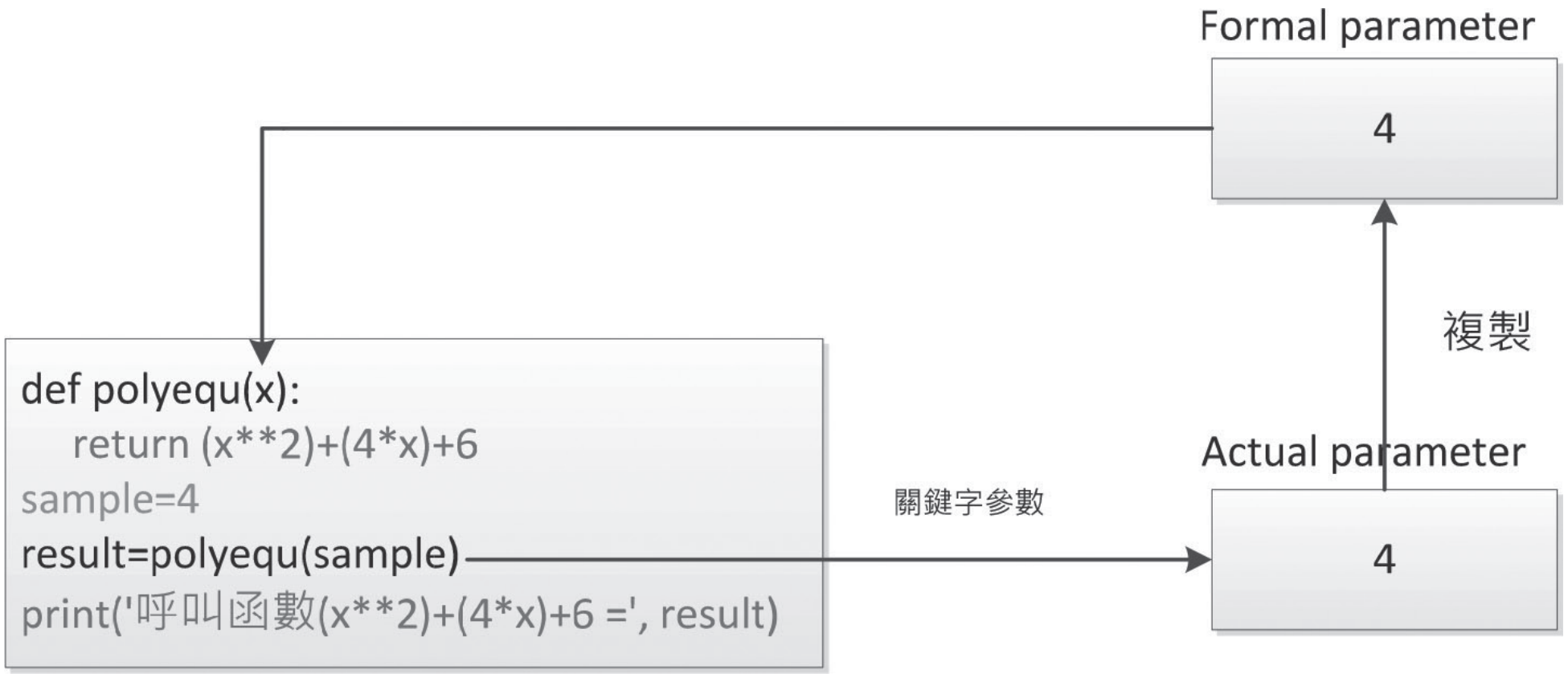


圖6-12　實際參數與形式參數的對映關係示意圖

3. **位置參數**：呼叫自訂參數時可以用位置參數也可以用關鍵字參數，但在 Python 系統內部的處理機制是以位置參數為主。形式參數必須依位置行為依參數的順序對應到實際參數，否則會出現錯誤。位置參數是實際參數的其中一個傳遞方式，不帶變數而是帶常數（見圖 6-13），圖中的 actual parameter 沒有設 sample 參數而是直接傳遞，此為位置參數的用法。

4. **關鍵字參數**：關鍵字參數的用法是實際參數中代入變數，如圖 6-12 的用法，即呼叫 polyequ 自訂函數時代入 sample，此 sample 即為關鍵字參數。

```
sample=4
result=polyequ(sample)
```

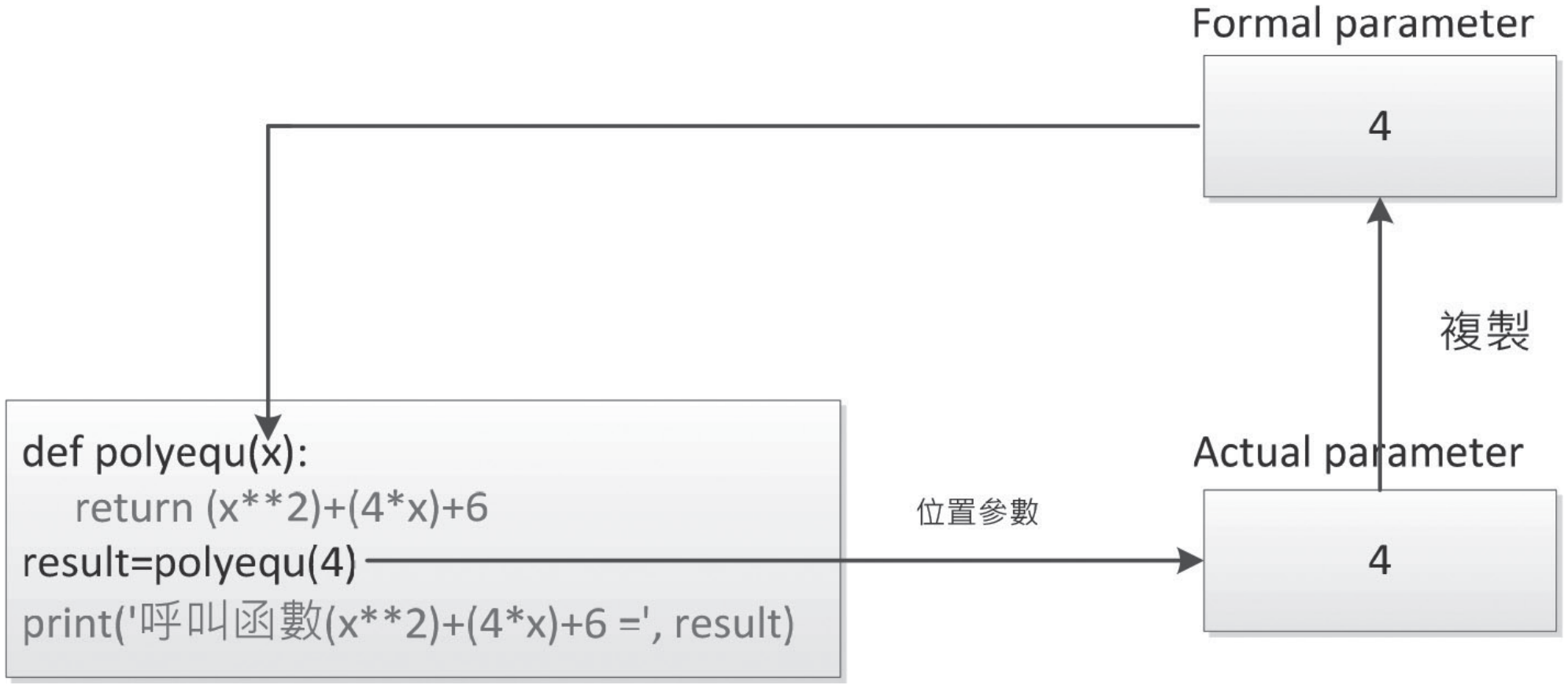


圖6-13　位置參數傳遞示意圖

範例 6-6 實際參數與形式參數的函數傳遞參數範例

代入 x，求 x^2+4x+6 的值。

示範程式碼

```
1  #E_6_6.py 功能：實際參數與形式參數的函數傳遞參數範例
2  def polyequ(x):
3  #polyequ: 計算 (x**2)+(4*x)+6 的結果
4      return (x**2)+(4*x)+6
5  sample=4
6  result=polyequ(sample)
7  print(' 呼叫函數 (x**2)+(4*x)+6 =', result)
```

程式說明

呼叫 polyequ 自訂函數代入實際參數 sample（關鍵字參數），回傳值指派給 result（見第 6 列），polyequ 自訂函數，以形式參數 x 接收 sample 的值（見第 2 列）。計算並回傳 (x**2)+(4*x)+6 的結果（見第 4 列），此寫法更簡化，把運算式直寫在 return 之後。

執行結果

```
1  呼叫函數 (x**2)+(4*x)+6 = 38
```

結果說明

假設 x 代入 4，x^2+4x+6 的結果等於 38。

範例 6-7 位置參數的使用範例

同【範例 6-6】改成呼叫自訂函數代入位置參數的用法。

示範程式碼

```
1  #E_6_7.py 功能 : 位置參數的使用範例
2  def polyequ(x):
3  #polyequ: 計算 (x**2)+(4*x)+6 的結果
4      return (x**2)+(4*x)+6
5  result=polyequ(4)
6  print(' 呼叫函數 (x**2)+(4*x)+6 =', result)
```

程式說明

呼叫 polyequ 自訂函數代入實際參數不是變數，而是純量 4，即指派給形式變數的是位置參數（見第 5 列）。其餘程式碼與執行結果同【範例 6-6】。

本章講解完畢

現場同學們如有不懂的地方，
請提出問題。