



巨匠直播教學

APCS Python語法基礎班

函式操作

www.pcschoolonline.com.tw

本堂教學重點

1. 函式宣告與使用
2. 傳入參數對應
3. 變數範圍
4. 參數傳遞
5. Lambda運算式

課程內容

1. 函式宣告與呼叫

1-1. 函式宣告

1-2. 函式呼叫

2. 傳入參數

2-1. 參數設定與對應

2-2. 不定個數參數

3. 函式進階

3-1. 區域變數及範圍

3-2. 參數傳遞方式

3-2. Lambda運算式

課程內容

1. 函式宣告與呼叫

1-1. 函式宣告

1-2. 函式呼叫

2. 傳入參數

2-1. 參數設定與對應

2-2. 不定個數參數

3. 函式進階

3-1. 區域變數及範圍

3-2. 參數傳遞方式

3-2. Lambda運算式

函式 function

◆ 函式 function

- ◆ 重複出現的程式流程
- ◆ 執行時呼叫函式，減少重複的程式碼
- ◆ 程式容易維護，維護程式只要修改一個地方

◆ python 中提供多種內建函式：

- ◆ `print()` / `int()` / `float()` / `str()` / `sum()` / `min()` / `max()` ...

◆ 自訂函式

- ◆ 將程式中重複出現的流程定義為自訂函式
- ◆ 函式名稱、傳入參數、傳回值

常用內建函式

- ◆ `ord()`：取得字元的編碼

`ord('A')` ➡ 65 `ord('匠')` ➡ 21280

- ◆ `chr(i)`：取得編碼數值對應的字元

`chr(65)` ➡ 'A' `chr(21280)` ➡ '匠'

- ◆ `eval(source)`：轉換字串內容為數值後運算

`eval('3+2*4')` ➡ 11

Python函式宣告與呼叫

◆ 函式宣告語法

def 函式名稱(參數, 參數, ...) :

敘述句

敘述句

return 傳回值

◆ 函式呼叫語法

變數 = 函式名稱(參數1, 參數2, ...)

函式宣告注意事項

◆ 注意事項

◆ 函式區塊以 **def** 開始，後接函式名稱和傳入參數小括號()

- 冒號下一行起相同縮排就是函式的內容範圍
- 結束縮排即離開函式範圍

◆ 在Python裡，函式也是物件，存放至變數中

◆ 函式名稱命名規則

- ◆ 大小寫字母、數字或 `_`，開頭字元不能是數字
- ◆ 不可與內建關鍵字同名，大小寫不同為不同的變數

傳入參數

◆ 傳入參數注意事項

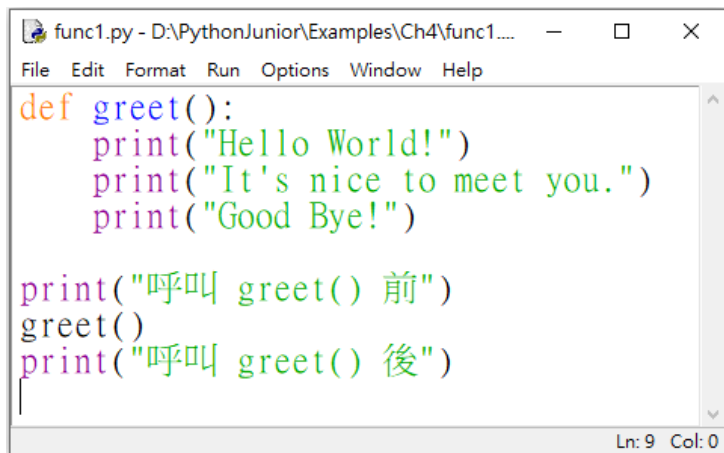
- ◆ 函式的傳入參數放在 () 內
- ◆ 可以接收零個、一個或多個傳入參數
 - 多個參數以逗點隔開
- ◆ Python 傳入參數是動態型別，可以是任意資料型態
 - 可以是單一參數，也可以是集合
- ◆ 呼叫時以參數的**位置**對應參數名稱

傳回值

◆ 傳回值注意事項

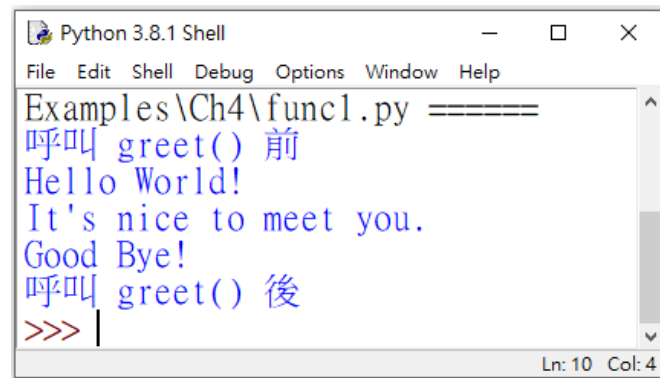
- ◆ 函式結尾處以 `return` 語法傳回物件
- ◆ `return` 後可接一個運算式或者變數資料
- ◆ `return` 只能回傳一個物件，多筆資料可放在集合容器內後回傳
- ◆ 未指定時回傳`None`物件
 - `return`
 - 省略 `return`

程式範例



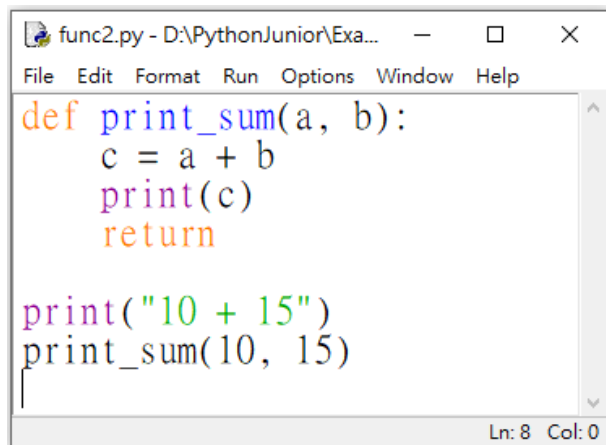
```
func1.py - D:\PythonJunior\Examples\Ch4\func1....
File Edit Format Run Options Window Help
def greet():
    print("Hello World!")
    print("It's nice to meet you.")
    print("Good Bye!")

print("呼叫 greet() 前")
greet()
print("呼叫 greet() 後")
Ln: 9 Col: 0
```



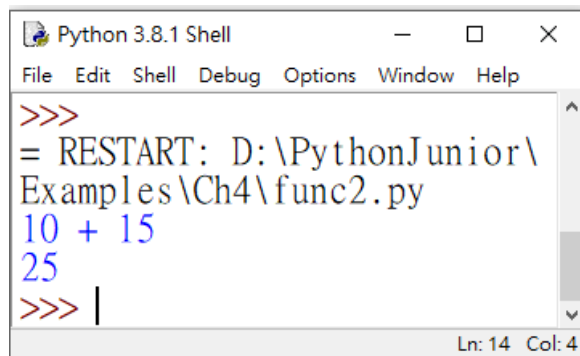
```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Examples\Ch4\func1.py =====
呼叫 greet() 前
Hello World!
It's nice to meet you.
Good Bye!
呼叫 greet() 後
>>> |
Ln: 10 Col: 4
```

程式範例



```
def print_sum(a, b):  
    c = a + b  
    print(c)  
    return  
  
print("10 + 15")  
print_sum(10, 15)
```

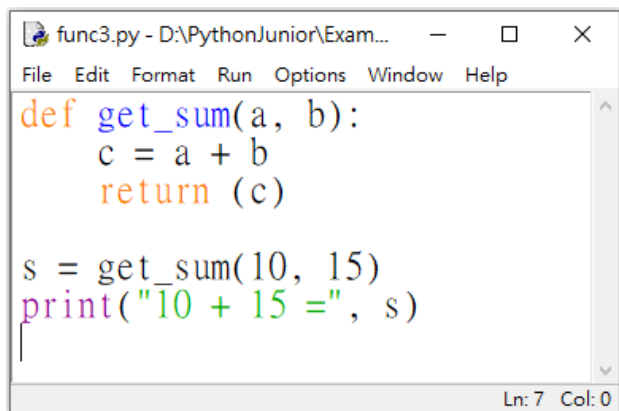
Ln: 8 Col: 0



```
>>>  
= RESTART: D:\PythonJunior\  
Examples\Ch4\func2.py  
10 + 15  
25  
>>> |
```

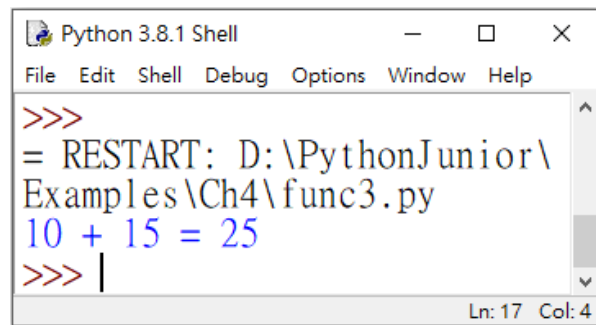
Ln: 14 Col: 4

程式範例



```
def get_sum(a, b):  
    c = a + b  
    return (c)  
  
s = get_sum(10, 15)  
print("10 + 15 =", s)
```

Ln: 7 Col: 0



```
>>>  
= RESTART: D:\PythonJunior\  
Examples\Ch4\func3.py  
10 + 15 = 25  
>>> |
```

Ln: 17 Col: 4

同名函式

◆ Python 程式內出現同名稱的函式定義

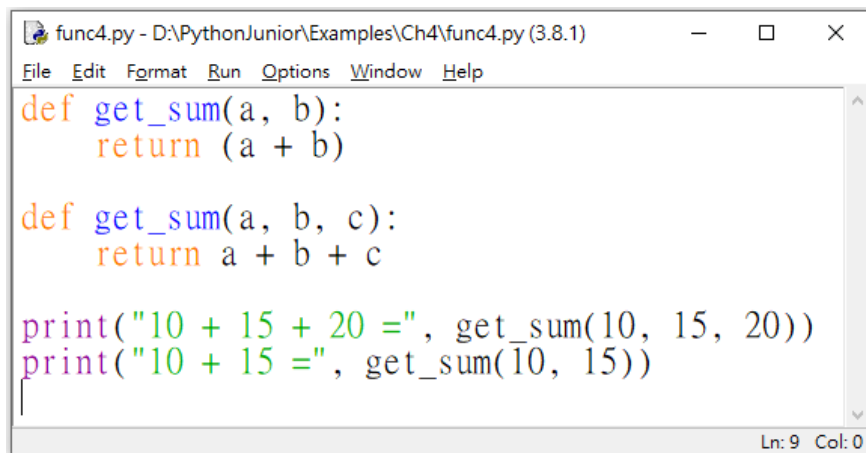
◆ 最後定義的函式覆蓋前面的定義

- 程式中出現同名函式時，執行內容為最後一個函式

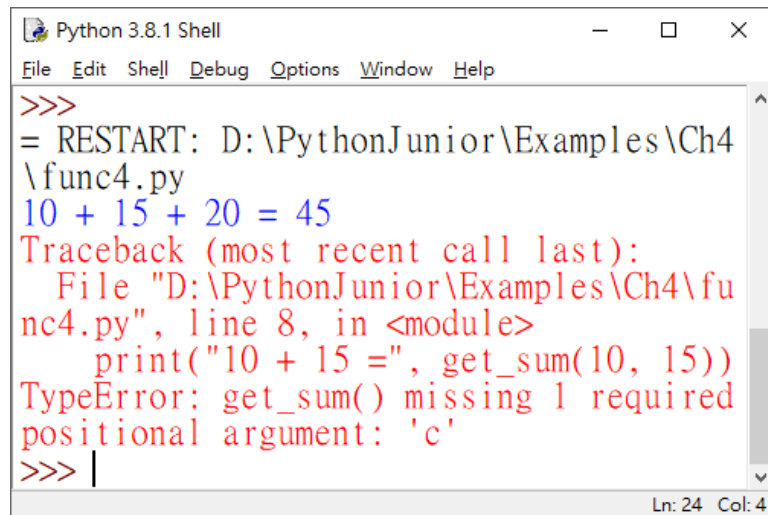
◆ Python 函式不支援參數多載

- 執行函式時所有參數均需要有數值
- 函式中參數沒有型別宣告，無法以參數型別辨別資料意義。

程式範例

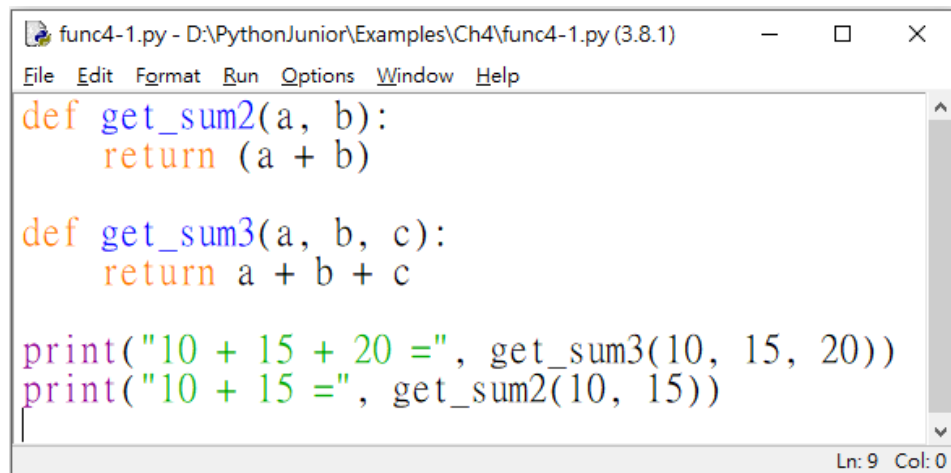


```
def get_sum(a, b):  
    return (a + b)  
  
def get_sum(a, b, c):  
    return a + b + c  
  
print("10 + 15 + 20 =", get_sum(10, 15, 20))  
print("10 + 15 =", get_sum(10, 15))
```



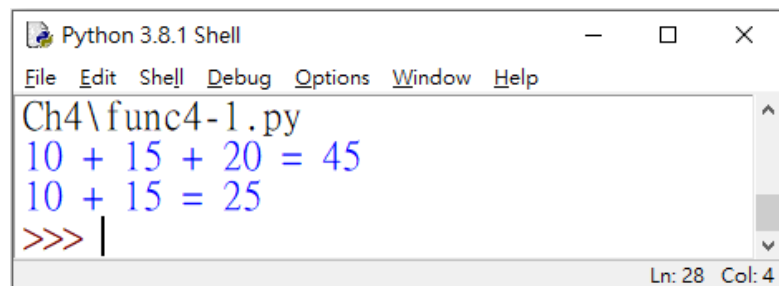
```
>>>  
= RESTART: D:\PythonJunior\Examples\Ch4\func4.py  
10 + 15 + 20 = 45  
Traceback (most recent call last):  
  File "D:\PythonJunior\Examples\Ch4\func4.py", line 8, in <module>  
    print("10 + 15 =", get_sum(10, 15))  
TypeError: get_sum() missing 1 required positional argument: 'c'  
>>> |
```

程式範例



```
def get_sum2(a, b):  
    return (a + b)  
  
def get_sum3(a, b, c):  
    return a + b + c  
  
print("10 + 15 + 20 =", get_sum3(10, 15, 20))  
print("10 + 15 =", get_sum2(10, 15))
```

Ln: 9 Col: 0



```
Python 3.8.1 Shell  
Ch4\func4-1.py  
10 + 15 + 20 = 45  
10 + 15 = 25  
>>> |
```

Ln: 28 Col: 4

Q：下列關於函式的敘述哪些正確？

- a) 將程式重複出現的指令寫成函式，需要時呼叫使用，可縮短程式碼
- b) 函式可以讓程式運作得更快
- c) 函式需要宣告傳回型態，且需與實際傳回物件型態一致
- d) 函式的傳回值使用return傳回，只能傳回一個物件

Q：函式沒有傳回值時程式應如何撰寫？

- a) 直接取消縮排離開區段即可
- b) return
- c) return void
- d) return None

函式練習

◆ 攝氏溫度轉換華氏溫度程式

◇ 將攝氏溫度 (C) 轉換為華氏溫度 (F) 的運算寫成一個函式

- 傳入參數為攝氏溫度
- 傳回值為華氏溫度
- 轉換公式為： $F=9/5*C+32$ 。

◇ 取得使用者輸入的攝氏溫度後，呼叫函式取得華氏溫度

- 輸入q時離開

課程內容

1. 函式宣告與呼叫

1-1. 函式宣告

1-2. 函式呼叫

2. 傳入參數

2-1. 參數設定與對應

2-2. 不定個數參數

3. 函式進階

3-1. 區域變數及範圍

3-2. 參數傳遞方式

3-2. Lambda運算式

函式參數設定與對應

◆ Python 不支援函式多載

- ◆ 相同功能，不同參數數量

- ◆ 相同功能，不同參數型態(意義)

◆ 參數宣告預設值

- ◆ 參數未傳入(不足)時，表示該參數值為預設值

- ◆ 函式宣告中，一但某個參數宣告預設值，之後的參數都要宣告預設值

程式範例

```
func5.py - D:\PythonJunior\Example...
File Edit Format Run Options Window Help

def get_sum(a, b=3, c=5):
    return a + b + c

x = get_sum(1, 2, 3)
print("get_sum(1, 2, 3) ->", x)

y = get_sum(1, 2)
print("get_sum(1, 2) ->", y)

z = get_sum(1)
print("get_sum(1) ->", z)

w = get_sum()
print("get_sum() ->", w)
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

s\Ch4\func5.py
get_sum(1, 2, 3) -> 6
get_sum(1, 2) -> 8
get_sum(1) -> 9
Traceback (most recent call last):
  File "D:\PythonJunior\Examples\C
h4\func5.py", line 13, in <module>
    w = get_sum()
TypeError: get_sum() missing 1 req
uired positional argument: 'a'
>>> |
```

有預設值參數
未傳入時使用預設值
 $x = 1 + 2 + 3 = 6$
 $y = 1 + 2 + 5 = 8$
 $z = 1 + 3 + 5 = 9$

無預設值參數
未傳入發生錯誤

函式參數設定與對應

◆ Python 不支援函式多載

- ◆ 相同功能，不同參數數量

- ◆ 相同功能，不同參數型態(意義)

- Python 是動態型別，無法用型態來辨識參數的意義

◆ 呼叫時以關鍵字指定參數 Keyword Argument

- 關鍵字指定之前的參數以位置對應
- 關鍵字指定之後的參數均需以關鍵字指定

程式範例

```
func6.py - D:\PythonJunior\Examples\Ch4\...  
File Edit Format Run Options Window Help  
def weighted_sum(c, e=80, m=60):  
    return (c+e*2+m*3)  
  
x = weighted_sum(100, m=90)  
print("x =", x)  
  
y = weighted_sum(e=90, m=80, c=70)  
print("y =", y)  
  
z = weighted_sum(e=60, c= 80)  
print("z =", z)  
  
#w = weighted_sum(m=60, e= 80, 75)  
#print("w =", w)  
Ln: 7 Col: 33
```

```
Python 3.8.1 Shell  
File Edit Shell Debug Options Window Help  
>>>  
===== RESTART: D:\PythonJunior  
\Examples\Ch4\func6.py =====  
x = 530  
y = 490  
z = 380  
>>> |  
Ln: 6 Col: 4
```

使用關鍵字參數

$x = 100 + 80 * 2 + 90 * 3 = 530$
 $y = 70 + 90 * 2 + 80 * 3 = 490$
 $z = 80 + 60 * 2 + 60 * 3 = 380$

關鍵字參數之後的參數
均需以關鍵字指定

函式呼叫參數傳遞規則

◆ 函式呼叫參數傳遞規則

- ◆ 位置參數(Positional Argument)：根據位置先後對應

f(1, 2, 3)

- ◆ 關鍵字參數(Keyword Argument)：關鍵字指定

f(y=2, x=1, z=3)

- ◆ 位置參數在前，關鍵字參數在後

f(1, 2, z=3)

- ◆ 參數不足時使用預設值

f(1, 2)

```
def f(x, y, z=3):  
    print(x, y, z)
```

函式呼叫參數傳遞規則

◆ 函式呼叫注意事項

◆ 個數不正確時產生錯誤

f() *f(1,2,3,4)*

◆ 關鍵字參數後不可再用位置參數

f(x=1, 2, 3)

◆ 參數不可重複

f(1, x=2)

```
def f(x, y=2, z=3):  
    print(x, y, z)
```

程式範例

```
func7.py - D:\PythonJunior\Examples\Ch4\fu...
File Edit Format Run Options Window Help

def weighted_sum(c, e=80, m=60):
    return (c+e*2+m*3)

x = weighted_sum(70, 80, 90, 100)
print("x =", x)

y = weighted_sum(70, m=90, 80)
print("y =", y)

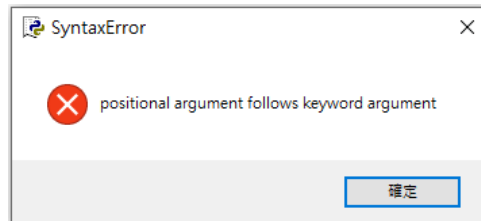
z = weighted_sum(70, 80, c=90)
print("z =", z)

Ln: 12 Col: 0
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

func7.py =====
Traceback (most recent call last):
  File "D:\PythonJunior\Examples\Ch4\func7.py", line 4, in <module>
    x = weighted_sum(70, 80, 90, 100)
TypeError: weighted_sum() takes from 1 to 3 positional arguments but 4 were given
>>>

Ln: 15 Col: 4
```



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

func7.py =====
Traceback (most recent call last):
  File "D:\PythonJunior\Examples\Ch4\func7.py", line 10, in <module>
    z = weighted_sum(70, 80, c=90)
TypeError: weighted_sum() got multiple values for argument 'c'
>>> |

Ln: 15 Col: 4
```

程式範例

```
func8.py - D:\PythonJunior\Examples\Ch4\func8.p...
File Edit Format Run Options Window Help
def printinfo( name="Amy", age=20 ):
    print ("Name: ", name)
    print ("Age ", age)
    print ("-----")
    return( )
```

```
printinfo( )
printinfo("Bill")
printinfo(30)
printinfo('Sean',45)
printinfo(50,"Ivy")
```

```
Python 3.8.1 Sh...
File Edit Shell Debug Options
Window Help
\Ch4\func8.py =====
Name: Amy
Age 20
Name: Bill
Age 20
Name: 30
Age 20
Name: Sean
Age 45
Name: 50
Age Ivy
>>> |
Ln: 18 Col: 4
```

程式範例

```
func8-1.py - D:\Python\Junior\Examples\Ch4\func...
File Edit Format Run Options Window Help
def printinfo( name="Amy", age=20 ):
    print ("Name: ", name)
    print ("Age ", age)
    print ("-----")
    return( )
```

```
printinfo( )
printinfo("Bill")
printinfo(age=30)
printinfo('Sean', 45)
printinfo(age=50, name="Ivy")
```

```
Python 3.8.1 Sh...
File Edit Shell Debug Options
Window Help
func8-1.py
Name: Amy
Age 20
-----
Name: Bill
Age 20
-----
Name: Amy
Age 30
-----
Name: Sean
Age 45
-----
Name: Ivy
Age 50
-----
>>> |
Ln: 35 Col: 4
```

不定個數的參數

◆ `def function_name(..., *listvar, **dictvar):`

◆ 函式參數數量有很多可能性時使用

◆ 一般參數之後宣告集合型態參數，讓函式可接受變動的參數數量

- `*listvar` 將多餘的位置參數視為序列 (tuple)。

- `**dictvar` 將多餘的關鍵字參數視為字典 (dictionary)。

◆ `*listvar` 與 `**dictvar` 可以同時使用。

- 函數中 `*listvar` 與 `**dictvar` 都只能有一個。

- `*listvar` 宣告在 `**dictvar` 之前

不定個數的參數

◆ 不定個數的參數

◆ *listvar :

- 呼叫函式時多餘參數使用位置對應(逗號隔開)

◆ **dictvar :

- 呼叫函式時多餘參數需使用關鍵字指定

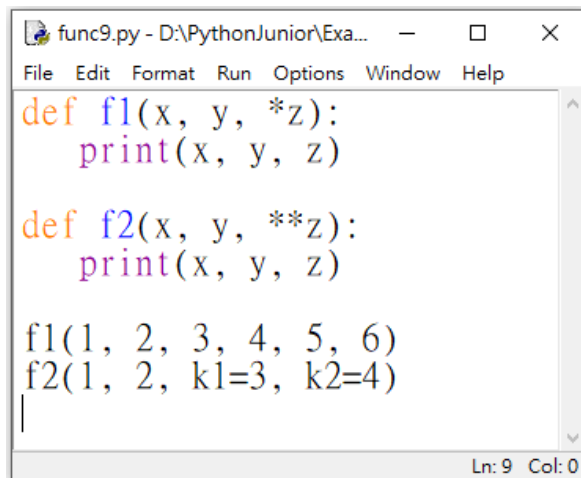
```
def f1(x, y, *z):  
    print(x, y, z)
```

```
>>> f1(1, 2, 3, 4, 5, 6)  
1 2 (3, 4, 5, 6)
```

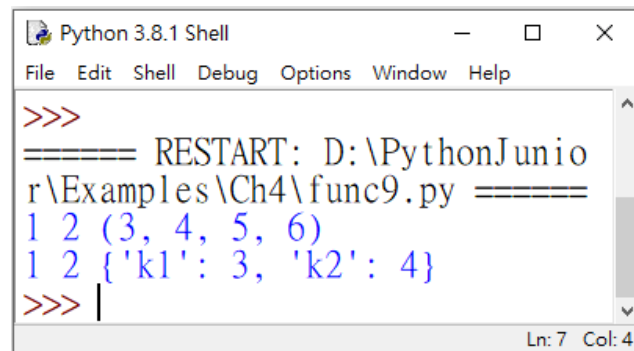
```
def f2(x, y, **z):  
    print(x, y, z)
```

```
>>> f2(1, k1=3, k2=4, y=2)  
1 2 {'k1': 3, 'k2': 4}
```

程式範例

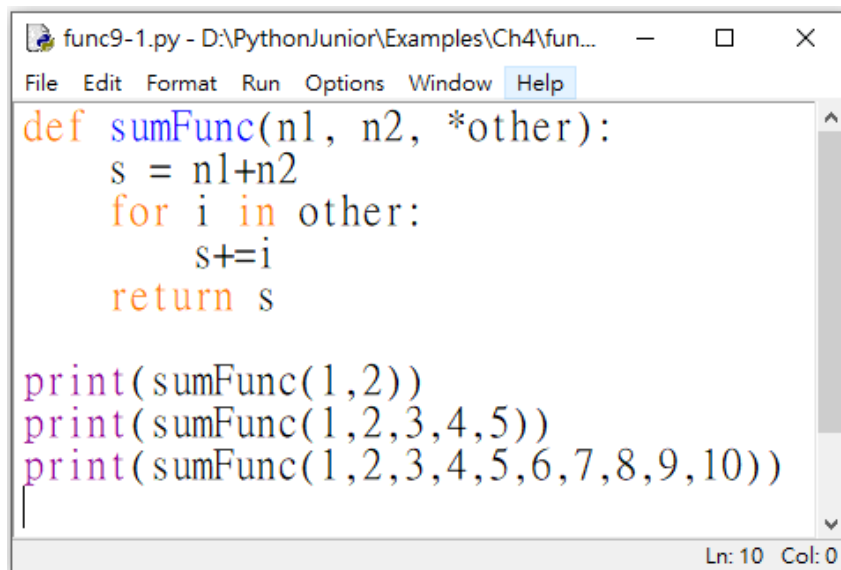


```
def f1(x, y, *z):  
    print(x, y, z)  
  
def f2(x, y, **z):  
    print(x, y, z)  
  
f1(1, 2, 3, 4, 5, 6)  
f2(1, 2, k1=3, k2=4)  
|  
Ln: 9 Col: 0
```



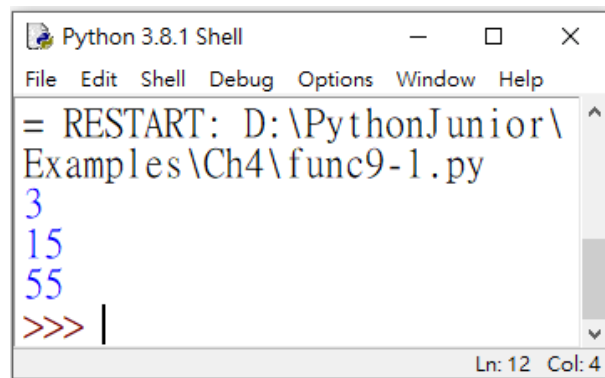
```
>>>  
===== RESTART: D:\PythonJunio  
r\Examples\Ch4\func9.py =====  
1 2 (3, 4, 5, 6)  
1 2 {'k1': 3, 'k2': 4}  
>>> |  
Ln: 7 Col: 4
```


程式範例



```
def sumFunc(n1, n2, *other):  
    s = n1+n2  
    for i in other:  
        s+=i  
    return s  
  
print(sumFunc(1,2))  
print(sumFunc(1,2,3,4,5))  
print(sumFunc(1,2,3,4,5,6,7,8,9,10))
```

Ln: 10 Col: 0



```
= RESTART: D:\PythonJunior\  
Examples\Ch4\func9-1.py  
3  
15  
55  
>>> |
```

Ln: 12 Col: 4

程式範例

```
func9-2.py - D:\PythonJunior\Examples\Ch4\func9-2.py (3.8.1)
File Edit Format Run Options Window Help
def personInfo(name, age, **other):
    print('===info===')
    print("name :", name)
    print("age :", age)
    for key in other:
        print(key, ":", other[key])

personInfo("Sean", 40)
personInfo('David', 35, phone='0987654321', company='IBM')
personInfo('Amy', 28, email='amy@gmail.com', company='Google', gender='Female')
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Examples\Ch4\func9-2.py
===info===
name : Sean
age : 40
===info===
name : David
age : 35
phone : 0987654321
company : IBM
===info===
name : Amy
age : 28
email : amy@gmail.com
company : Google
gender : Female
>>>
Ln: 28 Col: 4
```

Q : 下列程式碼輸出為何 ?

```
def greetPerson(*name):  
    print('Hello', name)  
  
greetPerson('David', 'Sean')
```

- a) Hello David
- b) Hello David Sean
- c) Hello ('David', 'Sean')
- d) 發生錯誤

Q：依下列函式定義，哪個選項執行結果為4？

```
def calc(amount=5, rate=2) :  
    if amount > 5 :  
        return amount * rate  
    else :  
        return amount * rate * 2
```

- a) calc(10, 3)
- b) calc(7)
- c) calc()
- d) calc(1)

Q：依下列函式定義，哪個選項不能正確呼叫函式？

```
def add(a=0, b=0) :  
    return a + b
```

- a) add a=3, b=5
- b) add(10)
- c) add()
- d) add('3','2')

課程內容

1. 函式宣告與呼叫

1-1. 函式宣告

1-2. 函式呼叫

2. 傳入參數

2-1. 參數設定與對應

2-2. 不定個數參數

3. 函式進階

3-1. 區域變數及範圍

3-2. 參數傳遞方式

3-2. Lambda運算式

變數範圍 Scope

- ◆ Python 是動態語言，變數在指定值時配置記憶體，建立物件
 - ◇ 靜態語言(ex: C, Java)，變數在宣告時建立
- ◆ 變數指定的位置不同，有不同的變數範圍
 - ◇ 在變數範圍內，可以存取變數內容
 - ◇ 離開變數範圍或使用del 刪除變數，變數回收，不可存取
 - 變數指向的物件，如果沒有被其他變數指向，會被系統自動垃圾收集

Python 變數範圍

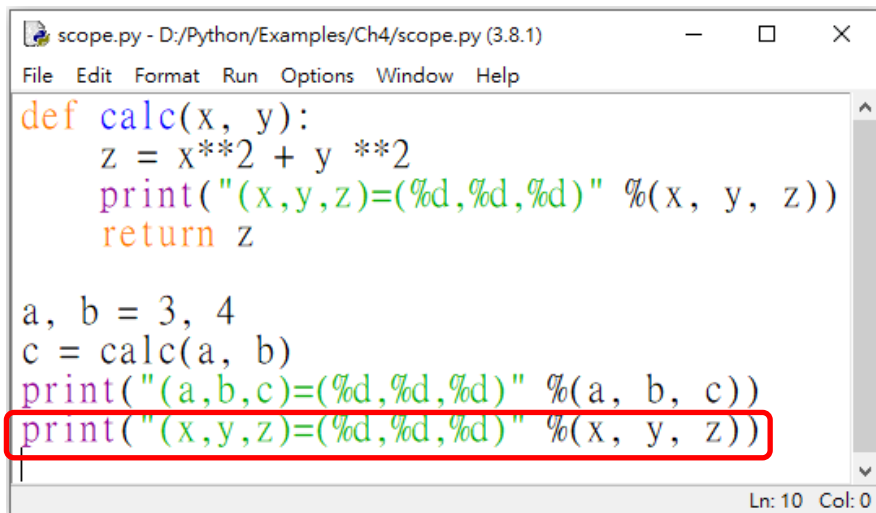
- ◆ Python 變數範圍(scope)：LEGB
 - ◆ Local 區域變數：函式或方法內指派的變數
 - ◆ Enclosing 閉包變數
 - ◆ Global 全域變數：程式(模組)內指派的變數
 - ◆ Builtin 內建變數：Python執行環境內建的變數

```
x = 10    # 全域
```

```
def func1():  
    y = 20    # 在 func1() 函式範圍
```

```
def func2():  
    z = 30    # 在 func2() 函式範圍
```


程式範例

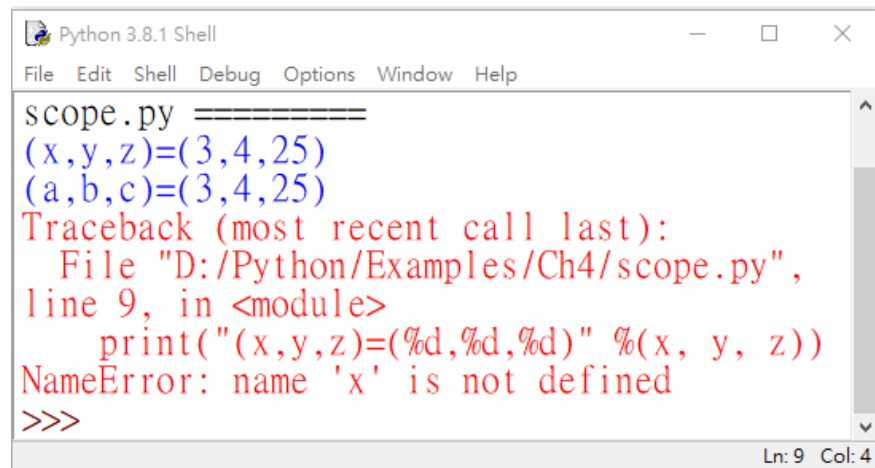


```
scope.py - D:/Python/Examples/Ch4/scope.py (3.8.1)
File Edit Format Run Options Window Help

def calc(x, y):
    z = x**2 + y **2
    print("(x,y,z)=(%d,%d,%d)" %(x, y, z))
    return z

a, b = 3, 4
c = calc(a, b)
print("(a,b,c)=(%d,%d,%d)" %(a, b, c))
print("(x,y,z)=(%d,%d,%d)" %(x, y, z))

Ln: 10 Col: 0
```



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

scope.py =====
(x,y,z)=(3,4,25)
(a,b,c)=(3,4,25)
Traceback (most recent call last):
  File "D:/Python/Examples/Ch4/scope.py",
    line 9, in <module>
      print("(x,y,z)=(%d,%d,%d)" %(x, y, z))
NameError: name 'x' is not defined
>>>

Ln: 9 Col: 4
```

參數傳遞方式

◆ Python 參數傳遞方式

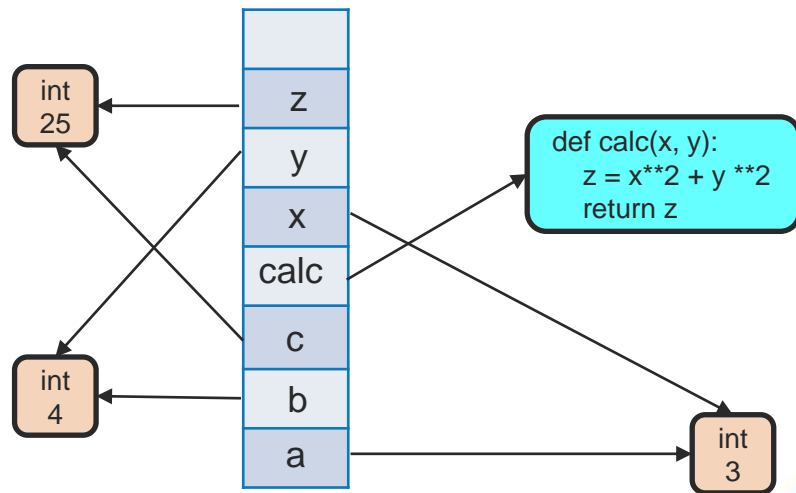
◆ 傳遞物件記憶體位址

◆ 參數型態不同有不同行為

◆ 不可變物件

◆ 可變物件

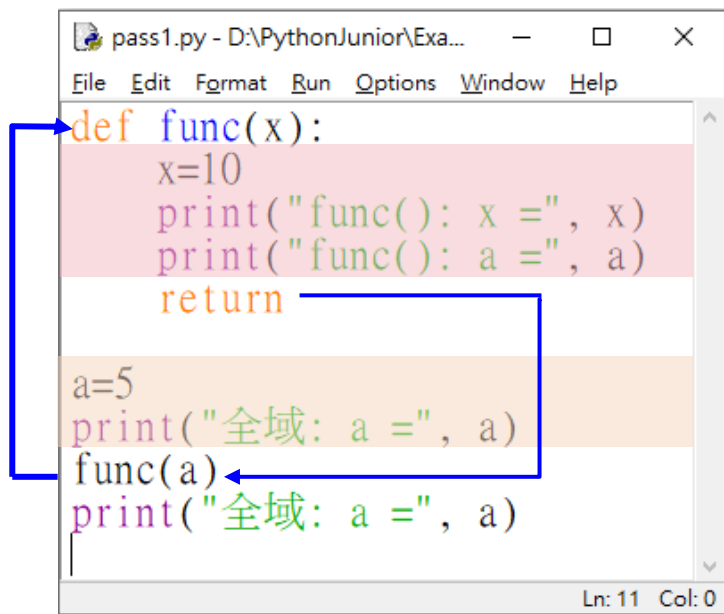
```
def calc(x, y):  
    z = x**2 + y **2  
    return z  
a, b = 3, 4  
c = calc(a, b)
```



不可變物件參數傳遞

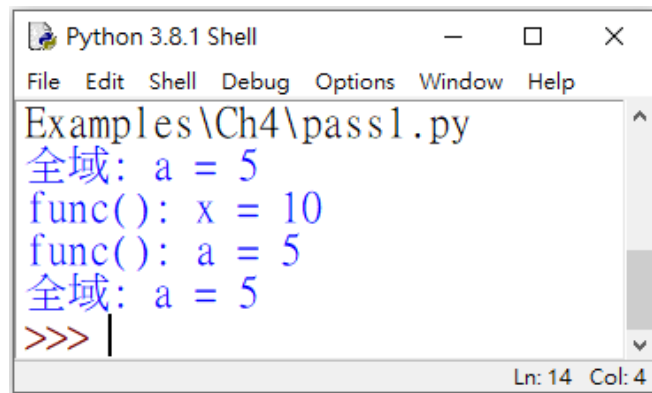
- ◆ 不可變(immutable)物件參數傳遞
 - ◆ 常見不可變物件：數字、字串或元組(tuple)
 - ◆ 函式收到不可變物件記憶體位址
 - ◆ 對傳入參數重新賦值，不影響函式外的全域變數
 - ◆ 不能直接修改原始物件，資料修改會建立一個新的物件
 - ◆ 類似其他語言的傳值呼叫

操作範例：請動手操作，並留意輸出結果



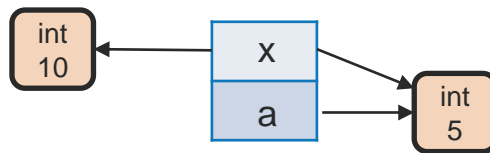
```
def func(x):  
    x=10  
    print("func(): x =", x)  
    print("func(): a =", a)  
    return  
  
a=5  
print("全域: a =", a)  
func(a)  
print("全域: a =", a)
```

Ln: 11 Col: 0



```
Python 3.8.1 Shell  
File Edit Shell Debug Options Window Help  
Examples\Ch4\pass1.py  
全域: a = 5  
func(): x = 10  
func(): a = 5  
全域: a = 5  
>>> |
```

Ln: 14 Col: 4



可變物件參數傳遞

- ◆ 可變(`mutable`)物件參數傳遞
 - ◆ 常見可變物件：序列(`list`)或字典 (`dict`)
 - ◆ 函式收到可變物件記憶體位址
 - ◆ 對傳入參數修改內容，會影響函式外的全域變數
 - ◆ 對傳入參數重新賦值，不影響函式外的全域變數
 - ◆ 類似其他語言的傳址呼叫

程式範例

```
pass2.py - D:\PythonJunior\Examples\Ch4\pass2.py (3.8.1)
File Edit Format Run Options Window Help

def func(thelist):
    print("func(): ", id(thelist), thelist)
    thelist[2]='Hi'
    print("func(): ", id(thelist), thelist)
    return

mylist = [10,20,30]
print("全域:", id(mylist), mylist)
func(mylist)
print("全域:", id(mylist), mylist)
```

Ln: 11 Col: 0

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

\Ch4\pass2.py =====
全域: 1542012717824 [10, 20, 30]
func(): 1542012717824 [10, 20, 30]
func(): 1542012717824 [10, 20, 'Hi']
全域: 1542012717824 [10, 20, 'Hi']
>>> |
```

Ln: 9 Col: 4



程式範例

```
pass3.py - D:\PythonJunior\Examples\Ch4\pass3.py (3.8.1)
File Edit Format Run Options Window Help

def func(thelist):
    print("func():", id(thelist), thelist)
    thelist = [1,2,3,4]
    print("func():", id(thelist), thelist)
    return

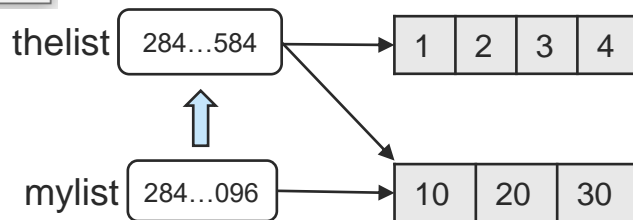
mylist = [10,20,30]
print("全域:", id(mylist), mylist)
func(mylist)
print("全域:", id(mylist), mylist)
```

Ln: 11 Col: 0

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

Ch4\pass3.py
全域: 2849254036096 [10, 20, 30]
func(): 2849254036096 [10, 20, 30]
func(): 2849260291584 [1, 2, 3, 4]
全域: 2849254036096 [10, 20, 30]
>>>
```

Ln: 27 Col: 4



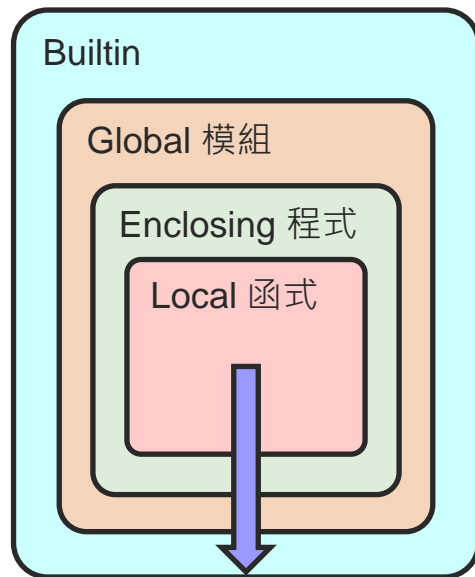
同名變數

◆ 同名變數指派

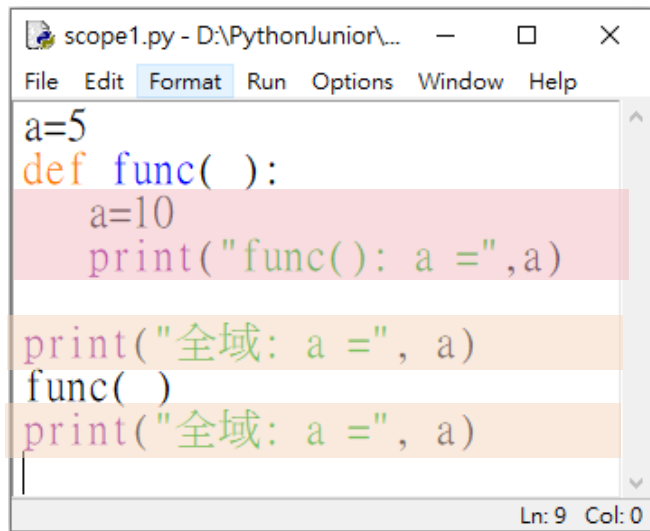
- ◆ 相同範圍內再次指派同名變數，變數指向新的物件
- ◆ 不同範圍指派同名變數，存在兩個名稱相同的不同變數

◆ 同名變數讀取

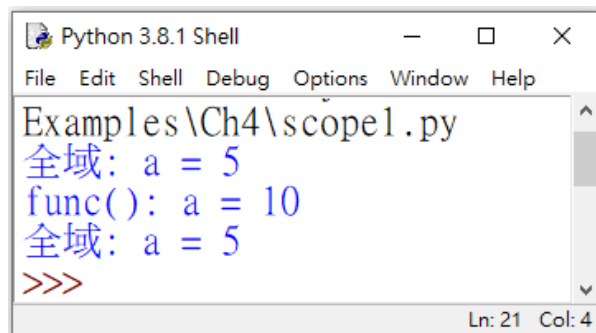
- ◆ 從所在範圍由內而外(LEGB)搜尋變數
- ◆ 不同範圍有同名變數時，會有變數遮蔽現象
 - 內層變數會遮蔽外層變數



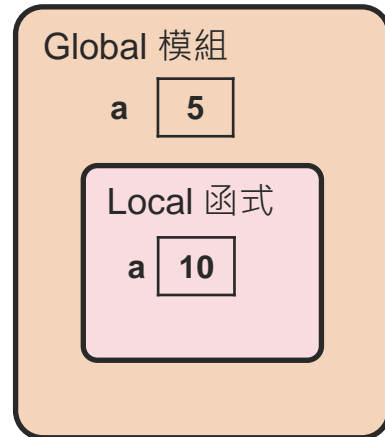
程式範例



```
scope1.py - D:\PythonJunior\...  
File Edit Format Run Options Window Help  
a=5  
def func( ):  
    a=10  
    print("func(): a =",a)  
  
print("全域: a =", a)  
func( )  
print("全域: a =", a)  
Ln: 9 Col: 0
```



```
Python 3.8.1 Shell  
File Edit Shell Debug Options Window Help  
Examples\Ch4\scope1.py  
全域: a = 5  
func(): a = 10  
全域: a = 5  
>>>  
Ln: 21 Col: 4
```



程式範例

```
scope1-1.py - D:\PythonJ...
File Edit Format Run Options Window Help

a=5
def func( ):
    #a=10
    print("func(): a =",a)

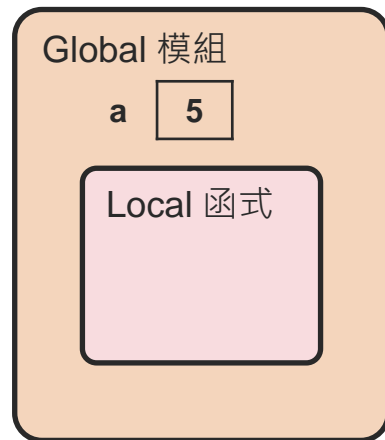
print("全域: a =", a)
func( )
print("全域: a =", a)

Ln: 9 Col: 0
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

Examples\Ch4\scope1-1.py
全域: a = 5
func(): a = 5
全域: a = 5
>>> |

Ln: 21 Col: 4
```

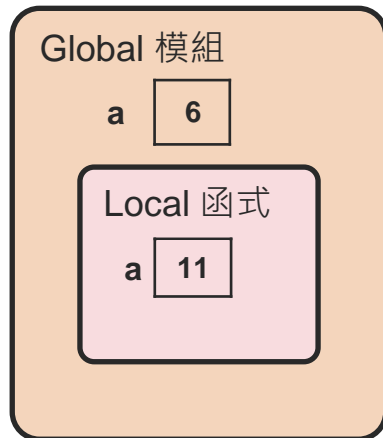


程式範例

```
scope2.py - D:\PythonJunior...
File Edit Format Run Options Window Help
a=5
def func( ):
    a=10
    a+=1
    print("func(): a =",a)

print("全域: a =", a)
func( )
a+=1
print("全域: a =", a)
Ln: 11 Col: 0
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Examples\Ch4\scope2.py
全域: a = 5
func(): a = 11
全域: a = 6
>>> |
Ln: 26 Col: 4
```



程式範例

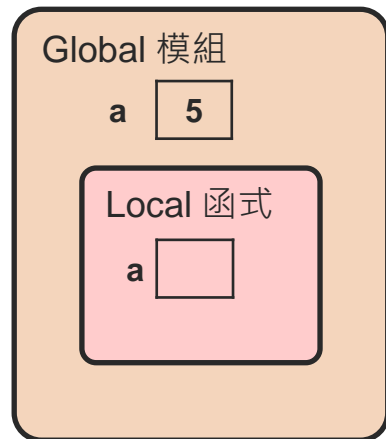
```
scope2-1.py - D:\PythonJuni...
File Edit Format Run Options Window Help
a=5
def func( ):
    #a=10
    a=a+1
    print("func(): a =", a)

print("全域: a =", a)
func( )
a+=1
print("全域: a =", a)
```

Ln: 11 Col: 0

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Ch4\scope2-1.py
全域: a = 5
Traceback (most recent call last):
  File "D:\PythonJunior\Examples\Ch4\scope2-1.py", line 8, in <module>
    func( )
  File "D:\PythonJunior\Examples\Ch4\scope2-1.py", line 4, in func
    a=a+1
UnboundLocalError: local variable 'a'
referenced before assignment
>>>
```

Ln: 35 Col: 4



程式範例

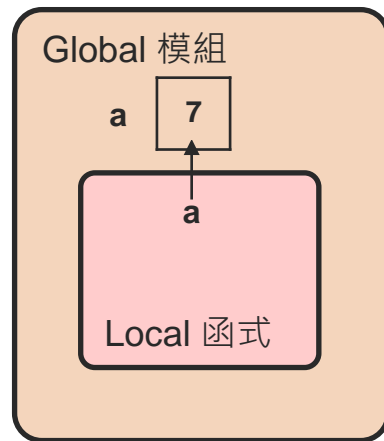
```
scope2-2.py - D:\PythonJunio...
File Edit Format Run Options Window Help
a=5
def func( ):
    global a
    a=a+1
    print("func(): a =",a)

print("全域: a =", a)
func( )
a=a+1
print("全域: a =", a)
```

Ln: 11 Col: 0

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
\\Examples\\Ch4\\scope2-2.py
全域: a = 5
func(): a = 6
全域: a = 7
>>> |
```

Ln: 40 Col: 4



Q：依下列函式定義，執行結果為何？

```
def calc(rate, item) :  
    item *= 1+rate  
  
rate = 0.25  
item = 12000  
calc(rate, item)  
print('Rate:', rate, ', Value:', item)
```

- a) Rate: 1.25, Value: 15000
- b) Rate: 0.25, Value: 15000
- c) Rate: 0.25, Value: 12000
- d) Rate: 1.25, Value: 12000

Q：依下列函式定義，執行後變數y為何？

```
def calc(balance):  
    balance += 2  
    return balance * 2
```

```
x = 4  
y = calc(x)
```

- a) 8
- b) 12
- c) 6
- d) 4

練習：BMI 計算函式

- 改寫BMI的練習，使用函式實作BMI計算程式。

Lambda運算式

◆ Lambda運算式

◆ 宣告匿名(沒有名稱)函式

- 只能包含一個運算式
- 可以有一個傳回值，即運算式的運算結果

◆ 語法：

lambda 參數, 參數, ... : 運算式

◆ 函式不會重複使用時，以**lambda**運算式取代函式宣告

◆ 函式中以**lambda**運算式作為函式的參數

```
def sq(x):  
    return x**2
```

```
sq = lambda x : x**2
```

```
print(sq(3))
```

Lambda搭配map()函式

◆ map() 函式

◇ map(function, iterable, ...)

- function : 轉換資料的Lambda運算式
- iterable : 要轉換的資料集合(List)。
- 傳回 iterator物件，元素內容為原集合元素以Lambda運算式轉換的結果
- 使用 list() 轉換為List

```
def sq(x):  
    return x**2
```

```
list1 = [1, 2, 3, 4, 5]  
#iter2 = map(sq, list1)  
iter2 = map(lambda x : x**2, list1)  
list2 = list(iter2) ➡ [1, 4, 9, 16, 25]
```

程式範例

```
lambda1.py - D:\PythonJunior\Examples\Ch4\la...
File Edit Format Run Options Window Help
list1 = [1,2,3,4,5]
list2 = map(lambda x : x**2, list1)
print(list1)
print(list2)
print(list(list2))
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Ch4\lambda1.py
[1, 2, 3, 4, 5]
<map object at 0x000001A89078FA00>
[1, 4, 9, 16, 25]
>>> |
Ln: 32 Col: 4
```

Lambda搭配filter()函式

◆ filter() 函式

◇ filter(function, iterable, ...)

- function：過濾用的Lambda運算式，運算結果為布林值
- iterable：要過濾的資料集合(List)。
- 傳回 iterator物件，內容為原集合中所有以Lambda運算得到true的元素
- 使用 list() 轉換為List

```
def odd(x):  
    return x%2!=0
```

```
list1 = [1, 2, 3, 5, 8,13,21,34,55,89]  
#iter2 = filter(odd, list1)  
iter2 = filter(lambda x : x%2!=0, list1)  
list2 = list(iter2) ➡ [1, 3, 5,13,21,55,89]
```

程式範例

```
lambda2.py - D:\PythonJunior\Examples\Ch4\lambda2.py ...
File Edit Format Run Options Window Help
list1 = [1,2,3,5,8,13,21,34,55,89]
print(list1)

list2 = filter(lambda x : x%2==0, list1)
print(list2)
print(list(list2))
|
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
\lambda2.py
[1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
<filter object at 0x00000237EA94F9D0>
[2, 8, 34]
>>> |
Ln: 37 Col: 4
```

Q：匿名函式使用哪個關鍵字宣告？

- a) anonymous
- b) def
- c) lambda
- d) secret

Q：依下列函式定義，執行結果為何？

```
result = lambda x : x * x
print(result(5))
```

- a) 10
- b) 25
- c) 5*5
- d) lambda x : x * x

Lambda函式練習

- ◆ 攝氏溫度轉換華氏溫度程式
 - ◆ 將攝氏溫度 (C) 轉換為華氏溫度 (F) 的運算寫成一個Lambda函式
 - ◆ 使用map()將攝氏溫度的集合轉換成華氏溫度的集合

練習：身份證字號驗證

◆ 台灣身份證字號驗證規則

◆ 首碼英文轉換為n0及n1，如下表(紅色代號已不再使用)

英文碼	數字碼	出生地	英文碼	數字碼	出生地	英文碼	數字碼	出生地	英文碼	數字碼	出生地
A	10	臺北市	G	16	宜蘭縣	M	21	南投縣	T	27	屏東縣
B	11	臺中市	H	17	桃園市	N	22	彰化縣	U	28	花蓮縣
C	12	基隆市	I	34	嘉義市	O	35	新竹市	V	29	臺東縣
D	13	臺南市	J	18	新竹縣	P	23	雲林縣	W	32	金門縣
E	14	高雄市	K	19	苗栗縣	Q	24	嘉義縣	X	30	澎湖縣
F	15	新北市	L	20	臺中縣	R	25	臺南縣	Y	31	陽明山
						S	26	高雄縣	Z	33	連江縣

練習：身份證字號驗證

◆ 英文之後的數值依序為n2~n10

◇ 將n1~n11乘以不同權重相加後為10的倍數者，為正確身分證字號

A	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

n0	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10
----	----	----	----	----	----	----	----	----	----	-----

(Note: In the original image, a green arrow points from the 'A' in the first row to the 'n1' in the second row.)



1	9	8	7	6	5	4	3	2	1	1
---	---	---	---	---	---	---	---	---	---	---



$$n0*1+n1*9+n2*8+n3*7+n4*6+n5*5+n6*4+n7*3+n8*2+n9*1+n10*1 = 10\text{的倍數}$$



巨匠直播教學

www.pcschoolonline.com.tw