

函數

Function

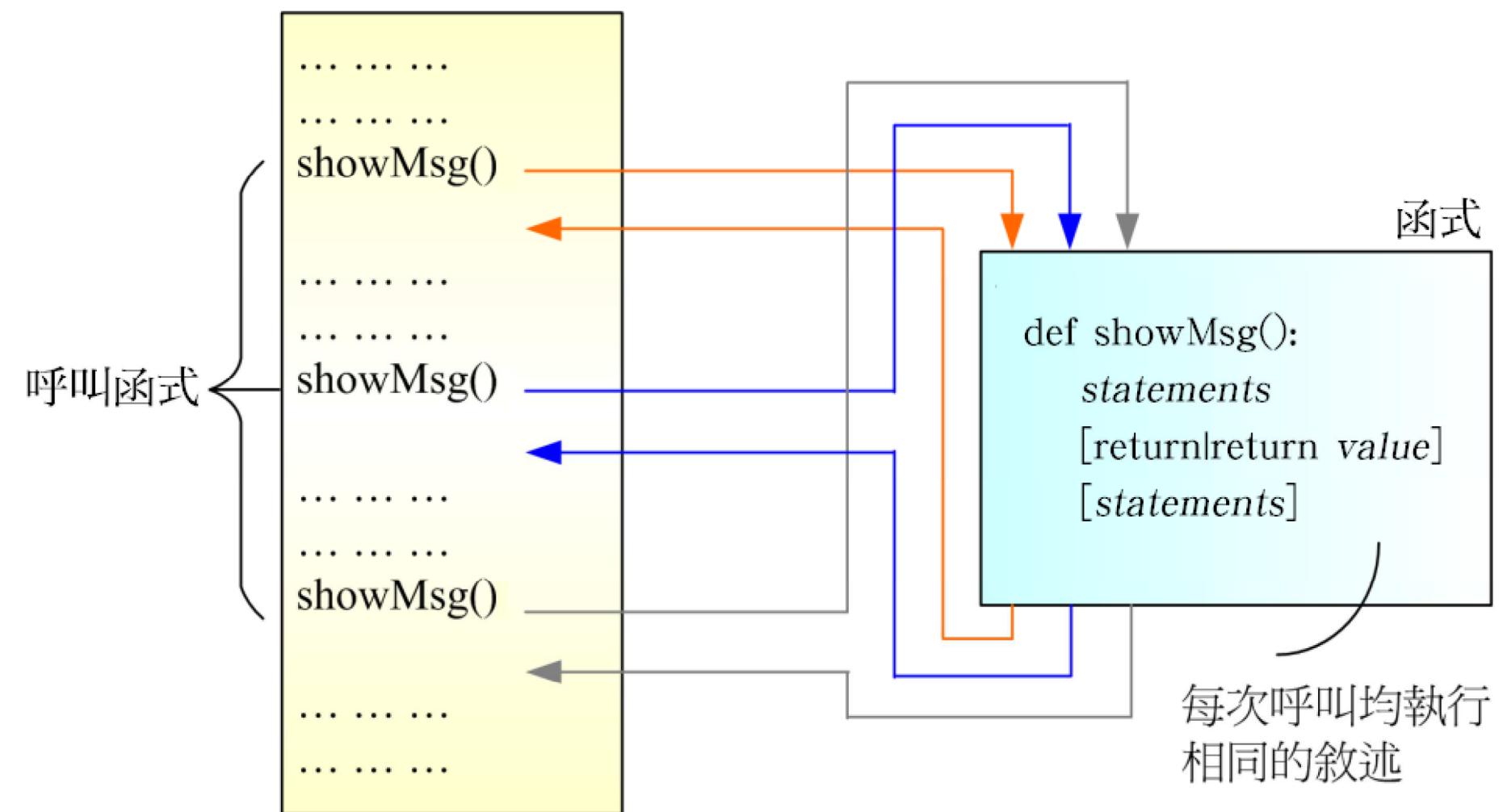


Python Fundamental



# 認識函數

- ◆ 函數 (function) 是將一段具有某種功能或重複使用的敘述寫成獨立的程式區塊，然後給予名稱，以供後續呼叫使用，可以簡化程式、提高可讀性。
- ◆ 有些程式語言將函數稱為函式、方法 (method)、程序 (procedure) 或副程式 (subroutine)，其概念是類似的。





# 定義函數

- ◆ 我們可以自行設計與定義函數。
- ◆ 函數必先定義後才可使用。

使用關鍵字「**def**」進行函數定義

```
def Function_name(Para1, Para2,...):  
    函數內容
```

任何函數內容必定縮排



# 定義函數

- ◆ 我們可以使用 def 關鍵字定義函數，語法如下：

```
def functionName([parameters]):  
    statements  
    [return|return value]  
    [statements]
```

```
def CtoF1(degreeC):  
    degreeF = degreeC * 1.8 + 32  
    print('攝氏', degreeC, '度可以轉換成華氏', degreeF, '度')
```

```
def CtoF2(degreeC):  
    degreeF = degreeC * 1.8 + 32  
    return degreeF
```



# 呼叫函數

- ◆ 函數定義後，必須呼叫函數才會執行。
- ◆ 函數名稱一樣，括號內的個數一樣，即可呼叫函數。





# 呼叫函數

```
def CtoF1(degreeC):  
  
    degreeF = degreeC * 1.8 + 32  
  
    print('攝氏', degreeC, '度可以轉換成華氏', degreeF, '度')  
  
temperatureC = eval(input('請輸入攝氏溫度 : '))  
  
CtoF1(temperatureC)
```

```
請輸入攝氏溫度 : 50  
攝氏 50 度可以轉換成華氏 122.0 度
```



# 參數與引數

## ◆ 參數與引數

- 引數 (Argument) 是用於呼叫函數，參數 (Parameter) 是函數簽章 (函數的宣告)。
- Parameter / 參數
  - 函數運行時之參考。
  - 放在函數的標記式，用來說明這個函數，當它被呼叫時必須接收到什種型態、數量的資料。
- Argument / 引數
  - 用以引發函數。
  - 當你呼叫函數的時候，你可以放在括號內的東西。

```
def Function_name(Pra1, Pra2,...):
```

```
Function_name ( Arg1, Arg2,... )
```

命名的函數如同變數一般

給予引數



# 參數與引數 - 傳遞引數給參數

- ◆ 以引數呼叫函數而將其值賦予參數以供函數運行。

**Function\_name(Arg1, Arg2, ... )**

**Function\_name(1, 2.5, 'ABC')**

**def Function\_name(Pra1, Pra2, ... )**

**def Function\_name(X, Y, Z )**

- ◆ Pra1 = Arg1
- ◆ Pra2 = Arg2
- ◆ PraN = ArgN

**Function\_name ( Arg1, Arg2, ... )**

**def Function\_name ( Pra1, Pra2, ... )**



# 函數的參數 - 預設引數值

- ◆ 預設引數值可減少呼叫函數(function call)撰寫上面的麻煩。

```
def Function ( Pra1=value1, Pra2=value2, ... ) :
```

- ◆ 即便呼叫函數時沒有給定引數，函數依然有參考值(預設引數值)：

1. 當自行定義函數時：

- 無預設值之參數在前。
- 有預設值之參數在後。

2. 當有預設值之參數仍有引數之分配時，引數值將會取而代之。

```
def Function ( Pra with no default | Pra with default ) :  
    預設值
```



# 函數的參數 - 預設引數值

- ◆ 我們可以在定義函數時設定預設引數值 (default argument value) 。

```
def teaTime(dessert, drink = '紅茶'):  
    print('我的甜點是', dessert, '，飲料是', drink)  
  
teaTime('馬卡龍', '咖啡')  
  
teaTime('帕尼尼')  
  
teaTime(drink = '奶茶', dessert = '三明治')  
  
teaTime('紅豆餅', drink = '綠茶')
```

我的甜點是 馬卡龍 ，飲料是 咖啡  
我的甜點是 帕尼尼 ，飲料是 紅茶  
我的甜點是 三明治 ，飲料是 奶茶  
我的甜點是 紅豆餅 ，飲料是 綠茶



# 函數的傳回值

---

- ◆ 有時我們可能需要從函數傳回某個值或某些值，此時可以使用 **return** 敘述，後面再加上傳回值。
- ◆ 一個函數於運行完畢之後可以回傳結果 (也可以忽略不回傳)。
- ◆ 當程式運行到 **return** 時會無條件直接離開(結束)函數並回傳結果。
- ◆ 當未使用 **return** 時，預設將會回傳 **None** 。
- ◆ **return** 語法：

**return something**

- ◆ 回傳值可以為：
  - 單一的值或物件。
  - 多個值或物件所構成的容器型態。



# 函數的傳回值

```
def div(x, y):  
    div = x // y  
  
    mod = x % y  
  
    return div, mod  
  
a, b = div(100, 7)  
  
print('100除以7的商數為', a, '，餘數為', b)  
  
c, d = div(200, 13)  
  
print('200除以13的商數為', c, '，餘數為', d)
```

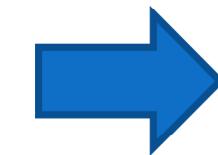
100除以7的商數為 14 ，餘數為 2  
200除以13的商數為 15 ，餘數為 5



# 變數的有效範圍

- ◆ 一個名稱在指定值時，就可以成為變數，並建立起自己的作用範圍（Scope）。
- ◆ 在讀取變數時，會就目前範圍中是否有變數名稱，若有則使用，若無則向外尋找，但不能向內尋找。

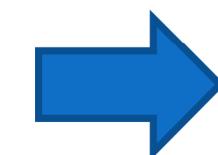
```
X=100  
def fun():  
    print(X)  
  
fun()
```



100

- ◆ 如果在 func() 中，對名稱 x 作了指定值的動作呢？

```
X=100  
def fun():  
    X=200  
    print(X)  
  
fun()  
print(X)
```



200



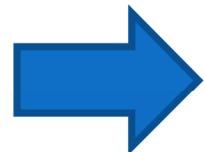
100



# 全域變數 & 區域變數

- ◆ 每個變數有其有效範圍，也就是被認定(認識)的範圍。
  - 定義在函數外的變數稱為全域變數，其範圍是一整個模組。
  - 定義在函數中的變數稱為區域變數，其範圍是一整個函數。
  - 當全域變數與區域變數名稱相同產生衝突時，區域內以區域變數為主，區域外以全域變數為主。

```
X=100  
  
def fun():  
    Y=200  
    print('Y=',Y)  
  
    print(X)  
  
fun()  
  
print(Y)
```



```
100  
Y = 200  
NameError: name 'Y' is not defined
```

Q & A