自訂函式

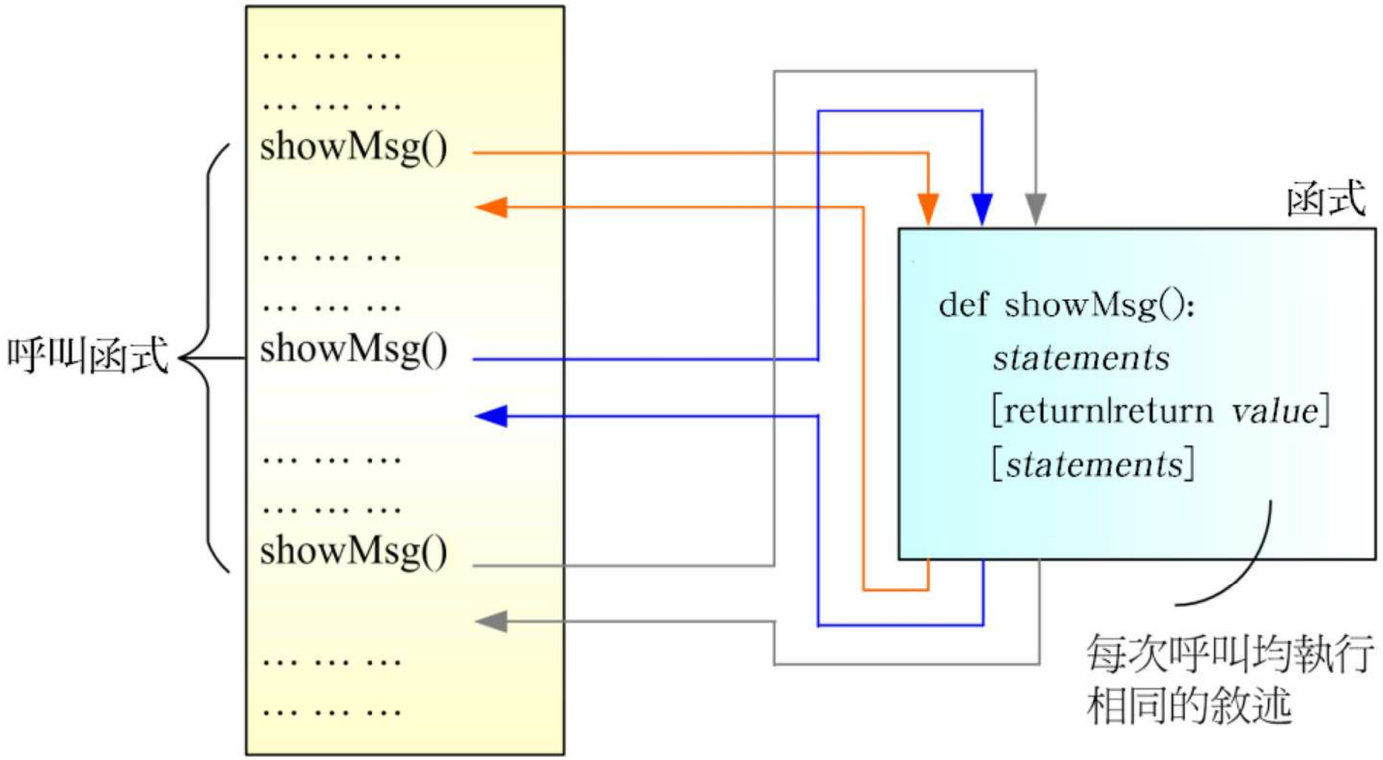
User-Defined Function

Python Fundamental

認識函式

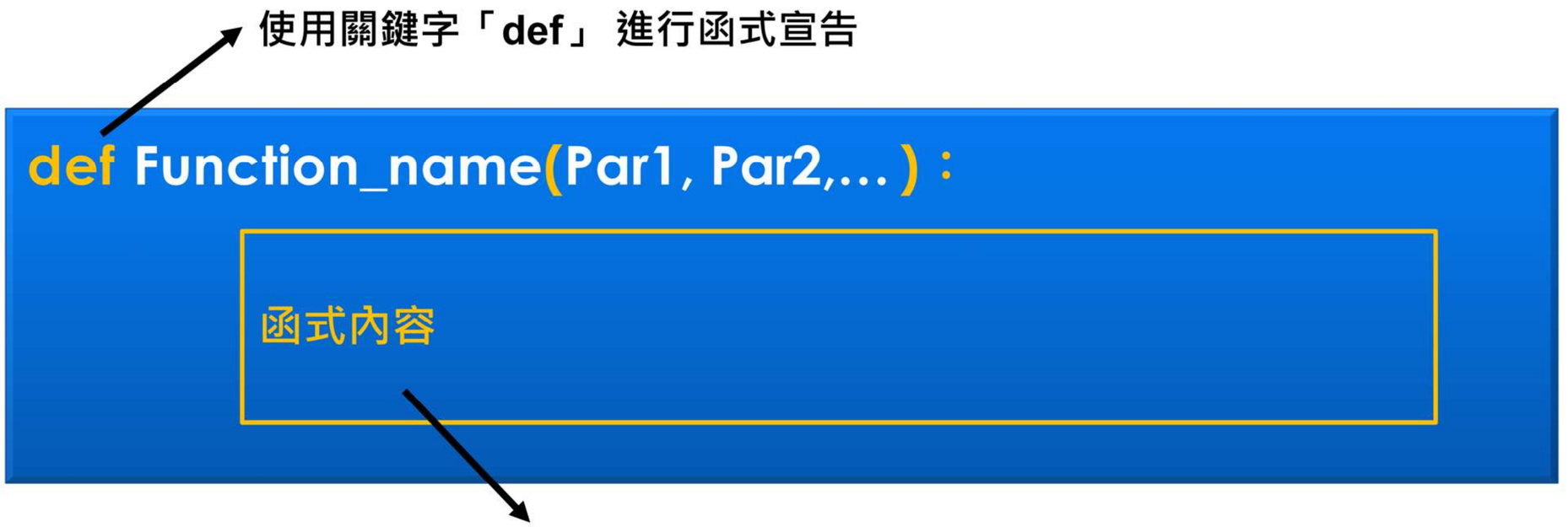
- 函式，function，是將一段具有某種功能或重複使用的敘述寫成獨立的程式區塊，然後給予適當的命名，讓後續呼叫使用，目的是簡化程式、提高可讀性。

- 有些程式語言會將函式稱為函數、方法 (method)、程式 (procedure) 或副程式 (subroutine)，但基本概念是類似的。



宣告與定義自訂函式

- 若不是內建函式時，我們可以宣告與定義自訂函式。
- 自訂函式一定要先進行宣告之後，後續才可以呼叫使用。



宣告與定義自訂函式

- 我們可以使用 def 關鍵字來宣告函式。

```
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
```

呼叫自訂函式

- 函式進行宣告/定義之後，接下來必須呼叫函式才會加以執行。
- **呼叫函式方式**：符合以下二個要件，就可以呼叫自定函式。

1. 函式名稱一樣。
2. 括號裡面的個數(數量)一樣，跟名稱，型態無關。

呼叫函式

```
def ctof(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：'))
ctof(temperatureC)
print('程式結束！')
```

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

參數與引數

- 參數 (Parameter) 是函式簽章 (函式的宣告)，引數 (Argument) 是用於呼叫函式。

- **Parameter / 參數**
  - 函式運行時之參考。
  - 放在函式的標記式，用來說明這個函式，當它被呼叫時需要接收到多少數量的引數。

```
def Function_name(Par1, Par2,... ) :
```

- **Argument / 引數**
  - 用以引發函式。
  - 呼叫函式的時候，放在括號內的資料。

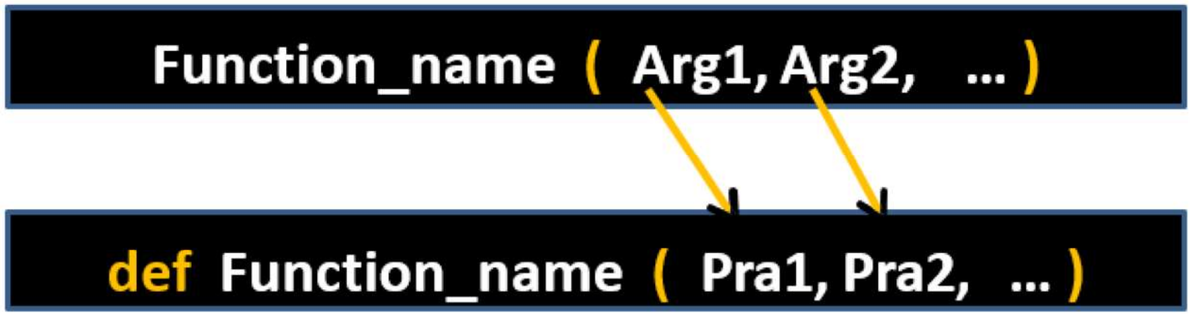
```
Function_name    ( Arg1, Arg2,... )
```

命名的函式如同變數一般

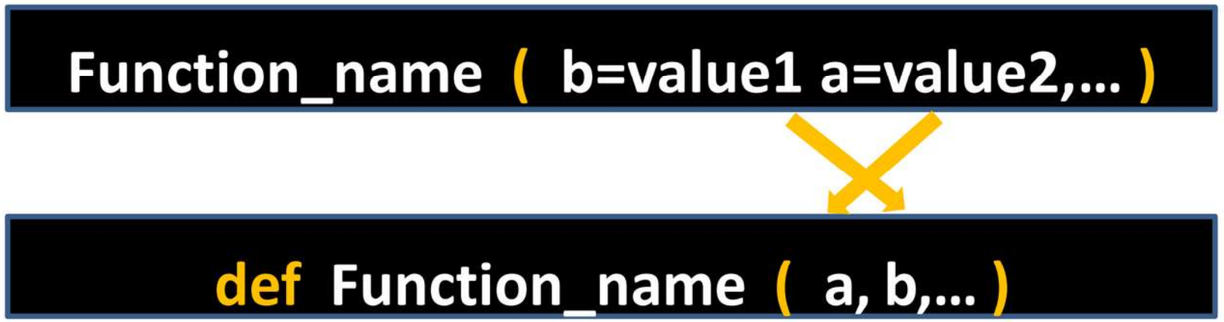
給予引數

參數與引數 - 傳遞引數給參數

- **Positional argument**：呼叫函式時依參數順序，一個對一個，這種給定的引數稱為位置引數 Positional argument。



- **Keyword argument**：呼叫函式時不依序一對一對應傳遞，換成以參數名稱來進行傳遞與函式之呼叫(名稱呼叫)，這種依名稱傳遞的引數稱為關鍵字引數 Keyword argument。



函式的參數 - 關鍵字引數

- Python預設採取位置引數 (Position argument)，但有時有些參數順序不好記，此時可以使用關鍵字引數 (Keyword argument) 來做區分，也就是**在呼叫函式時指定引數所對應的參數名稱**。

```
def Area(top, bottom, height):
    result = (top + bottom) * height / 2
    print('這個梯形面積為', result)
Area(10, 20, 5)
Area(10, height = 5, bottom = 20)
Area(height = 5, bottom = 20, top = 10)
```

```
這個梯形面積為 75.0
這個梯形面積為 75.0
這個梯形面積為 75.0
```

函式的參數 - 預設引數值

- 預設引數值可減少呼叫函式(function call)撰寫上面的麻煩。

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

- 即便呼叫函式時沒有給定引數，函式依然有參考值(預設引數值)：

1. 當自行定義函式時：
   - 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
```

函式的參數 - 預設引數值

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

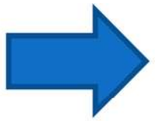
```
def teaTime(dessert,drink = '紅茶'):
    print('我的甜點：',dessert,'；飲料：', drink, sep="")
teaTime('馬卡龍', '咖啡')
teaTime('帕尼尼')
teaTime(drink = '奶茶', dessert = '三明治')
teaTime('紅豆餅', drink = '綠茶')
```

```
我的甜點：馬卡龍；飲料：咖啡
我的甜點：帕尼尼；飲料：紅茶
我的甜點：三明治；飲料：奶茶
我的甜點：紅豆餅；飲料：綠茶
```

變數的有效範圍

- 一個名稱在指定值時，就可以成為變數，並且建立起自己的作用範圍（Scope）。
- 讀取變數時，會就「目前範圍」檢查是否有變數的內容。如果有就使用；如果沒有會向「外」尋找，但無法向「內」尋找。

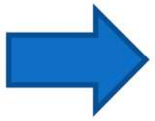
```
X=100
def  test():
    print(X)
test()
```



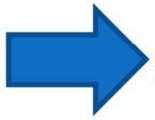
100

- 如果在 test() 中，對名稱 X 進行了指定值，其結果會如何呢？

```
X=100
def  test():
    X=200
    print(X)
test()
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。
  - 定義在函式外面的變數稱為全域變數，其範圍是一整個模組。
  - 定義在函式裡面的變數稱為區域變數，其範圍是一整個函式。
  - 當全域變數與區域變數的名稱相同時，函式裡面以區域變數為主，函式外面以全域變數為主。

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

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

函式的回傳

- 有時我們可能需要在函式運行完畢之後，回傳某個值或某些值，此時可以使用 return 敘述，後面再加上回傳的內容。
- 當未使用 return 時，預設將會回傳 None 。
- 當程式運行到 return 時，會無條件直接離開(結束) 函式並回傳結果。
- return 語法:　　將區域變數的內容變成全部變數

```
return something
```

- 回傳值可以為單一的值或物件，也可以是多個值或物件所構成的容器型態。

函式的回傳

```
def cal(x, y):
    div = x // y
    mod = x % y
    return div, mod
a, b = cal(120, 7)
print('120除以7的商數為', a, '，餘數為', b)
c, d = cal(250, 15)
print('250除以15的商數為', c, '，餘數為', d)
```

```
120除以7的商數為 17 ，餘數為 1
250除以15的商數為 16 ，餘數為 10
```

Q & A