函式

- 函式的介紹
- 函式的可用位置
- 函式外與函式內的變數

摘要：函式

- Who(對象)：函式Function
- Why(目的緣由)：為什麼需要函式Function
  - ➢ 節省重複撰寫相同程式碼的時間，程式精簡易讀，方便維護程式
- When(應用時機)/Where(應用場域)：將大型程式切分為許多小模組
  - ➢ 用函式去包裝各個功能(用來完成某一項功能的程式碼)
- What(內容定義)：函式的定義結構
  1.符號：def 2.用來完成某一項功能的程式碼
- How(操作機制)：1.函式定義 2. 呼叫函式 3. 存取變數 4. 參數傳遞
  1. 函式定義：函式名稱、傳入參數、傳出參數
  2. 呼叫函式：函式的可用位置
  3. 存取變數：全域變數 V.S. 區域變數
  4. 參數傳遞：函式間變數值的傳遞

函式是什麼

函式(Function)是專門用來完成某一項功能的程式碼。

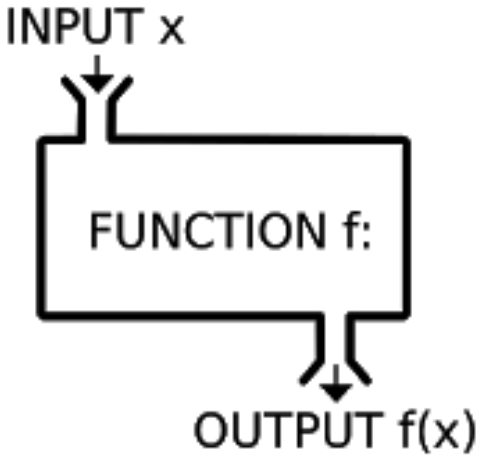
我們可以先定義一個函式的功能，

將來只要有需要用到這個功能時，

就可以使用預先寫好的這個函式，

節省重複撰寫相同程式碼的時間，

也讓程式更加精簡易讀。



為什麼要用函式

函式在開發大型程式中是非常重要的，
它可以讓工程師把每一個功能獨立拉出來寫，
將來某個功能有問題時，直接到相對應的函式去除錯(debug)，
方便去維護程式。

用函式去包裝各個功能，
能讓程式碼看起來較為精簡，
除此之外，亦可以增加程式的可讀性，
對於大型專案來說是非常重要的。

為什麼要用函式

- 模組化 (Modularity)：

將大型程式切分為許多小模組：因為模組為一小部分的程式碼，所以在撰寫、除錯、後續維護上都會比較方便

- 抽象化 (Abstraction)：

程式語言裡，抽象化指的是隱藏作業的細節，只提供必要的使用說明

1. 要輸入什麼：參數與型別
2. 該模組會做什麼事：模組的功能
3. 輸出是什麼：模組可能會回傳一個物件或是顯示處理後的資訊

為什麼要用函式

經過剛剛的介紹，可以了解函式就像個黑盒子。
黑盒子指的是只需要知道該作業的主要內容，不需知道其詳細步驟。

只要事先定義好函式的功能，
之後就可以直接將參數丟給函式，
而函式就會回傳我們想要的結果。

想想看，在生活中或是撰寫程式的過程中，有沒有類似函式的例子呢？

函式在生活中的例子

在生活上很多機器幫我們完成很多的事情，

這些機器幫我們所完成的事情，就像是函式完成的功能，

因此，這些生活上的機器可以看做是函式。

例如：自動販賣機。

我們不知道自動販賣機的內部結構，因此自動販賣機就像黑盒子。

這個黑盒子就像是一個函式，它已經定義好要做物品販賣的功能。

生活中的函式範例：「自動販賣機」

舉例：需要自動販賣機的服務(買飲料)

輸入 (input)：奶茶的編號與投入相應的金錢

函式：自動販賣機(黑盒子函式)

輸出 (output)：欲買的飲料

函式在程式中的例子

學會函式之前，
每當需要比大小的時候，都需要用到一個if去比較兩數的大小，
這會造成程式碼有很多的重複並且冗長；

在學會函式後，
我們是否可以定義一個專門用來比大小的函式，
每當我們需要比大小時就利用這個函式，
透過呼叫函式的做法，一行程式碼就可以做完比大小的動作，
這樣是不是方便許多呢？

函式與方法

- 我們在先前已經用過的print()、range()、len()、…，它們都是Python語言內建的函式。
- 物件自己有函式。例如：List列表的insert()、append() …等。不過物件的函式，我們會用方法(method)來稱呼它們。
- 本章我們將要學習如何自己建立屬於自己專用的函式。

函式語法

```
def 函式名稱(參數):
        函式程式碼區塊(某特定功能)
    return 傳回值
```

- 函式的命名規則和變數一樣

- 參數：函式的輸入資料，

代表執行該函式程式碼區段所需要的輸入資料。
參數的個數視功能需求而定，可以不用參數，也可有多個參數。

- 傳回值：函式的輸出資料，

代表執行完該函式程式碼區段所得的結果。
傳回值可透過return傳回多個傳回值，
不用傳回值的話便不用寫return。

函式：無傳入參數、無傳回值

```
def 函式名稱():
        函式程式碼區塊
```

範例:

```
def hello():
    print('Hello Python')
hello()
```

```
Hello Python
```

呼叫hello()函式，
自動幫我們執行print('Hello Python')

hello() → 印出Hello Python

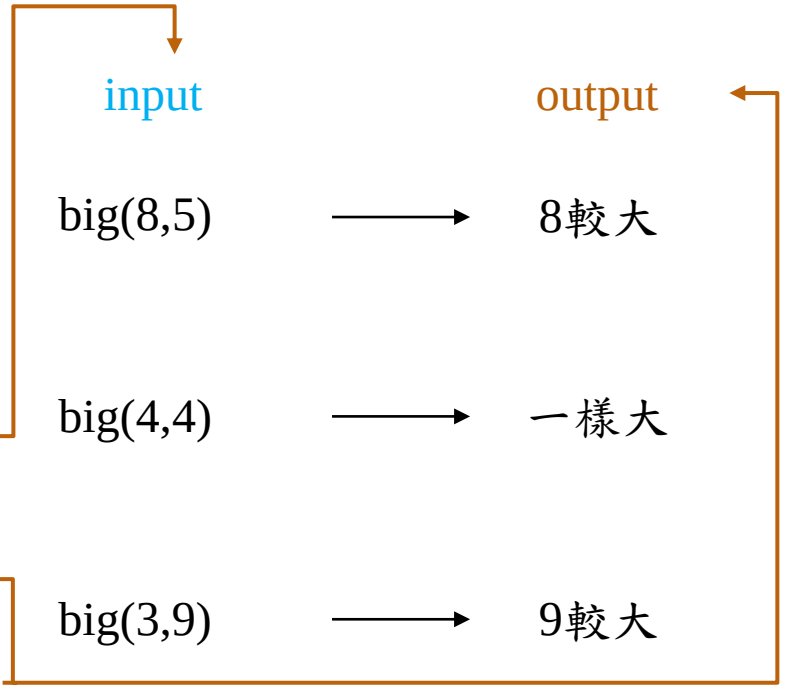
函式：有傳入參數、無傳回值

參數在我們需要給函式input的時候來使用，
像是剛剛所提及的比大小函式，
我們必須給函式2個數字才能讓函式進行比大小，
這時候就要利用兩個參數來提供函式所需要的值。

```
def 函式名稱(參數):
        函式程式碼區塊
```

```
def big(a, b):
    if(a>b):
        print(a, "較大")
    elif(a==b):
        print("一樣大")
    else:
        print(b, "較大")
big(8,5)
big(4,4)
big(3,9)
```

```
8 較大
一樣大
9 較大
```



函式：有傳入參數、有傳回值(1/2)

在函式這個黑盒子，除了有傳入參數之外，
有個非常重要的概念是傳回值(return)，
在比大小的範例是在函式中直接印出數值，
但有時候我們需要獲取函式的結果來進行後續的操作，
這時候就必須使用回傳值(return)來回傳函式運算結果。

```
def 函式名稱(參數):
        函式程式碼區塊(某特定功能)
        return 傳回值
```

函式：有傳入參數、有傳回值(2/2)

範例：用函式比大小，回傳較大的值。之後再乘以2倍進行輸出。

```
def big(a, b):
    if(a>b):
        return a
    else:
        return b

Double_big = big(3,7) * 2
print(Double_big)
print("big(3,7)的型態==>",type(big(3,7)))
print("big的型態==>",type(big))
```

big(3,7)會回傳較大的值，
也就是 7

```
14
big(3,7)的型態==> <class 'int'>
big的型態==> <class 'function'>
```

用type()來觀察資料型態的不同

C8_P1：計算兩個數字的和

定義一個函式 sum_of_numbers

- function名稱要一模一樣
- 括號裡的參數名稱可以不一樣
- 但參數個數一定要一致

傳回num1和num2的總和

C8_P2：計算矩形面積

定義一個函式名為calculate_area(width, height)

使用寬度乘以高度計算面積 area
傳回面積

函式的可用位置

- Python是直譯器，所以函式在被呼叫之前需要先定義(見過)函式。
- 我們大多會將定義函式放在程式的最上方，
  避免函式可能會因為還沒定義而產生錯誤訊息。

```
f()
def f():
        print("function f")
```

在程式尚未看過f函式的定義時就進行函式呼叫，
會導致程式不知道這個f函式的功能是甚麼，
並回傳錯誤訊息：
NameError: name 'f' is not defined

```
def f():
        print("function f")
f()
```

```
function f
```

函式的可用位置

```
def f():
        print('f()')
        g()

def g():
        print('g()')
f()
```

想想看，左邊的這段程式碼，

先呼叫函式g()，之後才定義函式g()，

會不會出現

NameError: name 'f' is not defined

或

NameError: name 'g' is not defined

Python 語言中，函式的「見過」與否是以執行流程來定，
不是以程式碼的位置來定。

函式的可用位置-常見用法

定義一個主函式 main()，讓主函式做所有的事，
最後在程式的最下面呼叫它。

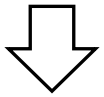
```
def main():
    f()
    print('.')
    g()

def f():
    print('f()')
    g()

def g():
    print('g()')

main()
```

函式內與函式外的變數

函式裡面與外面的同一個變數名稱i，表示兩個具有相同名稱的變數。
函式內稱為區域變數；函式外稱為全域變數。

```
i = 9
def f():
    i=10
    print("in  f() :",i,"id(i) :",id(i))
f()
print("out f() :",i,"id(i) :",id(i))
```

取得內存地址



```
in  f() : 10 id(i) : 94633203160384
out f() : 9 id(i) : 94633203160352
```

函式外使用區域變數

函式外面不可以使用函式裡面定義的區域變數，會產生錯誤訊息。

函式內定義的變數

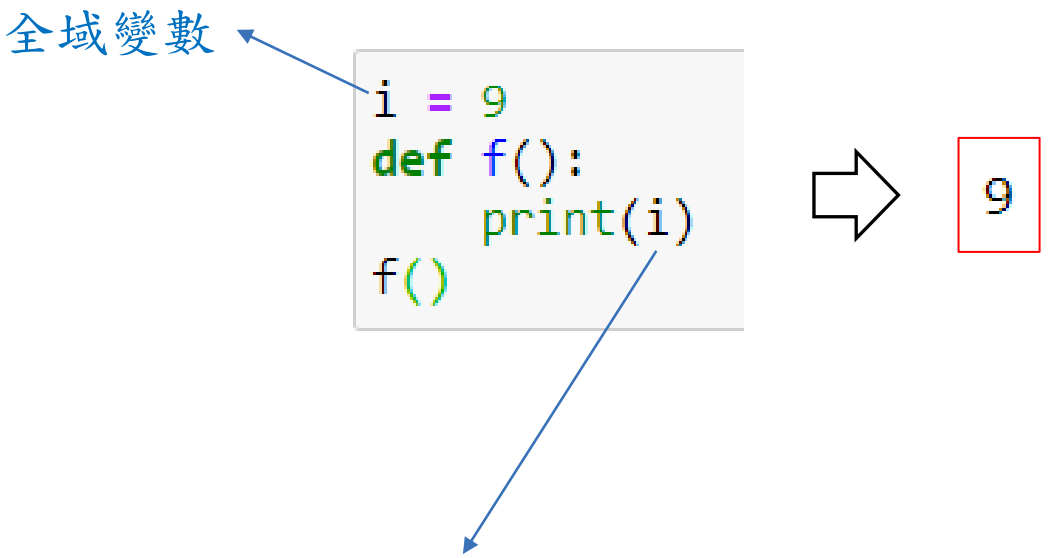
函式外未定義

```
def f():
    fi = 9

print(fi)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-16-c01e06f60f62> in <module>
      2     fi = 9
      3
----> 4 print(fi)

NameError: name 'fi' is not defined
```

函式內使用全域變數(1/2)

函式內可以使用函式外定義的全域變數。



函式內使用全域變數(2/2)

- 我們剛剛提到函式內可以使用外面的全域變數，

但切記不能在函式內修改全域變數！

- 只要在函式中出現指派賦值（包含 a=, for a in …）的敘述，

變數a便被認定為區域變數，在函式中的a都視為該區域變數a。

```
i = 9
def f():
    i=i+1
    print(i)
f()
```

錯誤:

```
UnboundLocalError: local variable 'i' referenced before assignment
```

函式內修改全域變數： global

在函式中使用global來定義變數，
用來明確指出我們要使用全域變數。

```
i = 9
def f():
    global i
    i=i+1
    print(i)
f()
```

```
10
```


程式碼中的 global 可以算是一個警示，
提醒工程師這裡可能會修改到全域變數。

傳遞參數給其他函式

函式跟函式之間的變數不能通用，
需要利用參數來互相傳遞！

```
def f():
    score = int(input('請輸入分數>'))
    check()

def check():
    if score>=60:
        print('及格')
    else:
        print('不及格')
f()
```

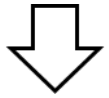
直接使用其他函式的變數將產生Error

```
NameError: name 'score' is not defined
```

傳遞參數給其他函式

```
def f():
    score = int(input('請輸入分數>'))
    check(score)

def check(sc):
    if sc>=60:
        print('及格')
    else:
        print('不及格')
f()
```

利用參數來傳遞資料



```
請輸入分數>60
及格
```

C8_P3：Maximum Triplet Sum

定義一個函式 functionmax_tri_sum

- Function名稱要一模一樣，括號裡的參數名稱可以不一樣
- 但參數個數一定要一致(這題參數為一個list，參數就是一個，可以命名為numbers)

利用sorted function將列表中的數字排序 sorted_nums

初始化總和 total為0

初始化list為空串列 top3_list

遍歷排序後的數字列表

如果 list長度小於3 且 這個數字沒有重複

將取出來的數字num加到總和total中

將取出來的數字num新增加到top3_list中

傳回 total

C8_P4：計算總和直到指定數字的範圍

定義一個函式名為sum_until_number(number)

初始化總和為0 total
使用for in range迴圈從1到number
　　依序將取出的數字加入總和
傳回總和

Contest

Problem

C8_P1：計算兩個數字的和
C8_P2：計算矩形面積
C8_P3：Maximum Triplet Sum
C8_P4：計算總和直到指定數字的範圍

Exercise

C8_E1：判斷是否為質數
C8_E2：判斷字串是否為迴文
C8_E3：Digital Root
C8_E4：Powers of 2

總結

Q

本章關鍵字
本章重點概念簡述

A

請將你的章節總結寫在Memo上，
Memo的title請設定為總結