

# 函數 Function

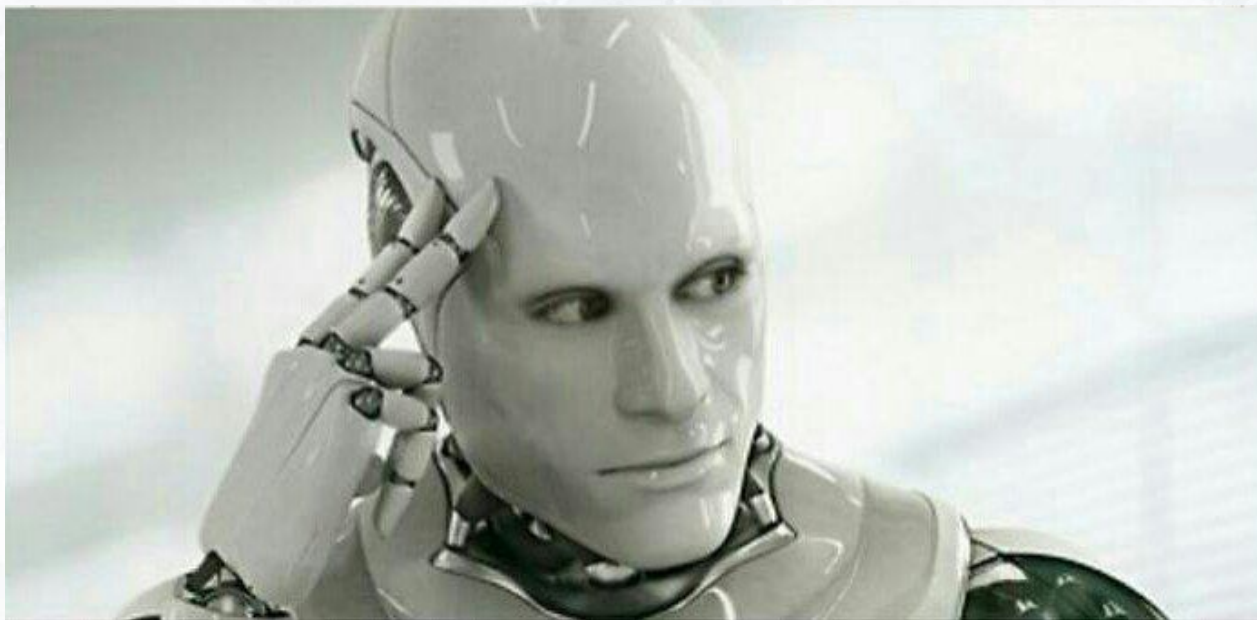
Python 編程課程 第 3 課

- 定義函數
- 內建函數

# 熱身 - 講早晨

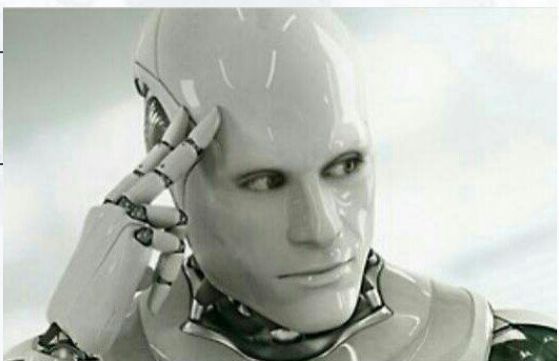
## 情境：

你是個機械人，每天都要和別人講「早晨」(Good morning)，  
講完再匯報當天天氣、交通和股市狀況。



# 熱身 - 講早晨

你的程式碼	輸出
<pre>target = input("Who are you: ")  print("Good morning," , target)  print("The weather is good.")  print("The traffic is good, no congestion among all cross-harbour tunnels.")  print("X指股市：萬六叢中一點紅")</pre>	<p>Who are you: 樓下陳伯</p> <p>Good morning, 樓下陳伯</p> <p>The weather is good.</p> <p>The traffic is good, no congestion among all cross-harbour tunnels.</p> <p>X指股市：萬六叢中一點紅</p>

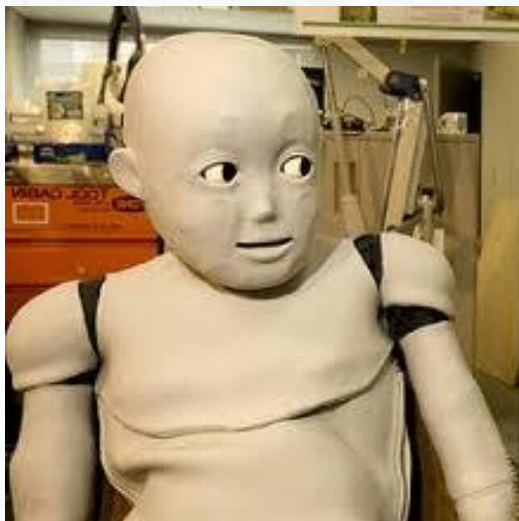




# 熱身 - 講早晨

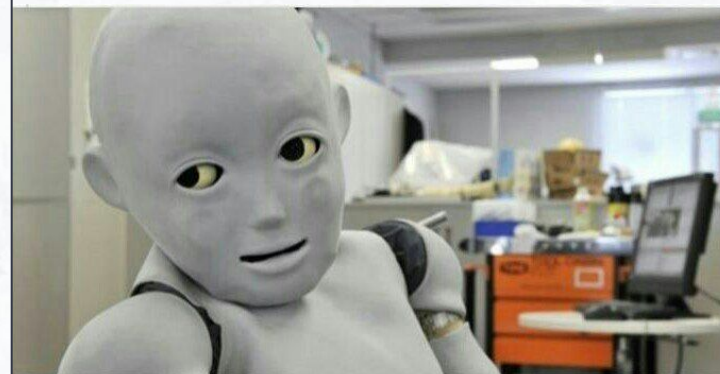
## 問題來了:

你的對象不止樓下陳伯，還有一連串市民，  
每個都要講「早晨」(Good morning)，  
講完再匯報天氣、交通.....



# 熱身 - 講早晨

你的程式碼？	輸出
<pre>target = input("Who are you: ")  print("Good morning," , target) print("The weather is good.") print("The traffic is good, no congestion among all cross-harbour tunnels.") print("X指股市:")  target = input("Who are you: ")  print("Good morning," , target) print("The weather is good.") print("The traffic is good, no congestion among all cross-harbour tunnels.") print("X指股市:")  target = input("Who are you: ")  print("Good morning," , target) print("The weather is good.") print("The traffic is good, no congestion among all</pre>	<p>Who are you: 樓下陳伯</p> <p>Good morning, 樓下陳伯 The weather is good. The traffic is good, no congestion among all cross-harbour tunnels. X指股市:</p> <p>Who are you: 樓上李太</p> <p>Good morning, 樓下陳伯 The weather is good. The traffic is good, no congestion among all cross-harbour tunnels. X指股市:</p> <p>Who are you: 隔離何太</p> <p>Good morning, 樓下陳伯 The weather is good. The traffic is good, no congestion among all cross-harbour</p>





# 熱身 - 講早晨

函數	sayGoodMorning()	
功能	講早晨、匯報天氣、交通、股市	
輸入	遇到的對象名稱	



重覆調用函數 sayGoodMorning()  
就可以實現講早晨的功能

# 函數 Function

函數 Function 是一個建構程式時的小區塊，像是一台機器／小程序。

你可以指定它的**功能**，以及所需要的**原料(輸入)**、**產出(輸出)**。

舉例來說，自動販賣機就像是一個 function。

**input** 是硬幣和商品的選擇，

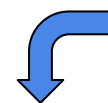
**output** 則是你所選的商品。



# 函數結構

調用函數名稱	定義 Definition	<pre>def sayGoodMorning(name):     print("Good morning,", name)     print("The weather is good.")     print("The stock market went crazy.") return &lt;data&gt;  sayGoodMorning(hisName)</pre> <p>縮進的區塊是函數主體， 這些程式碼將被執行。</p>
輸入 (外部)	參數 Parameter (引數 Argument)	
處理	程式主體 Function body	
輸出*	回傳 Return	

使用關鍵字 **def** 告訴  
Python 你要定義一個函數



呼叫函數，告訴 Python 你要  
使用這個函數。



# 傳遞參數

函數定義中的括號 ( ) 用來獲取外部信息——參數 (parameter)

```
def say_morning(username):  
    """display good morning"""  
    print(f"good morning~ {username}")  
  
say_morning('Keith')
```

輸出

```
good morning~ Keith
```

通過括號 ( ) 獲取的信息可以在函式**內部**使用。

在這種情況下，參數 username 被賦值為參數 'Keith'。

'Keith'  
username

# 位置引數Positional Argument 與 關鍵字引數Keyword Argument

調用函數時，有兩種傳遞參數的方式：

## 位置引數

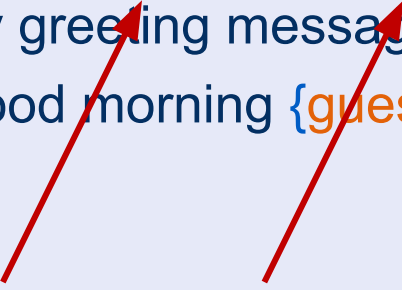
```
def say_morning(username):  
    print(f"good morning~ {username}")  
  
say_morning('LP')
```

## 關鍵字引數

```
def say_morning(username):  
    print(f"good morning~ {username}")  
  
say_morning(username='LP')
```

# 位置引數：次序重要

```
def greeting(guestname, myname):  
    """display greeting message """  
    print(f"Good morning {guestname}. My name is {myname}")  
  
greeting('Anthony', 'Leo')
```



輸出

Good morning Anthony. My name is Leo

傳入多個引數時，  
Python會根據引數次序



# 關鍵字引數：次序不重要

```
def greeting(guestname, myname):  
    """display greeting message """  
    print(f"Good morning {guestname}. My name is {myname}")  
  
greeting(myname='Leo', guestname='Anthony')
```

如果你告訴Python哪個引數**應該**被傳入哪個參數, Python就不會根據次序

輸出

Good morning Anthony. My name is Leo

# 函式的預設值

```
def greeting(guestname, myname='Keith'):
```

```
    print(f"Good morning {guestname}. My name is {myname}")
```

```
greeting(guestname='Anthony')
```

由於 myname 沒有被傳遞, Python 將使用預設值。

在函式定義中, 你可以為每個參數定義一個預設值:

<parameter> = <default value>

這樣該參數就變成了可選的(可傳可不傳)

輸出

Good morning Anthony. My name is Keith

**i** 具有預設值的參數應該放在沒有預設值的參數之後。

# 錯誤 – 缺少參數

典型錯誤：

- 參數不符
- 欠缺傳入參數

```
def greeting(guestname, myname='LP'):  
    """display greeting message """  
    print(f"Good morning {guestname}. My name is {myname}")  
  
greeting()
```

輸出

```
TypeError: greeting() missing 1 required positional argument:  
'guestname'
```

錯誤訊息：

我們缺少必要的參數：**guestname**



# 堂課 - 請打開Thonny一起做

# 調用函數 - 計算長方形面積

粉紅色斜體: 題目

黑色正常字體: 提供的程式碼

紅色底線: 請填入程式碼

## 程式碼

*# 定義函數 rectangleArea, 傳入參數 length 和 width*

def rectangleArea(length, width):

*# 建立變量計算面積*

area = \_\_\_\_\_ \* \_\_\_\_\_

*# 列印面積*

print(f"The area is: {\_\_\_\_\_} m^2.")

*# 用 input() 讀入 length 和 width*

length = input()

width = \_\_\_\_\_

*# 調用函數*

rectangleArea( \_\_\_\_\_ , \_\_\_\_\_ )

## 輸出

4

2

The area is: 8 m^2.

# 調用函數 - 計算長方體體積

粉紅色斜體: 題目

黑色正常字體: 提供的程式碼

紅色底線: 請填入程式碼

## 程式碼

```
# 定義函數 volume, 傳入參數 height、length、width
def volume(height, _____, _____):
    # 建立變量計算面積
    volume = _____ * _____ * _____
    # 列印面積
    print(f"The volume is: {_____} m^3.")

# 用 input() 讀入 length 和 width
height = input()
length = _____
width = _____

# 調用函數
volume(_____, _____, _____)
```

## 輸出

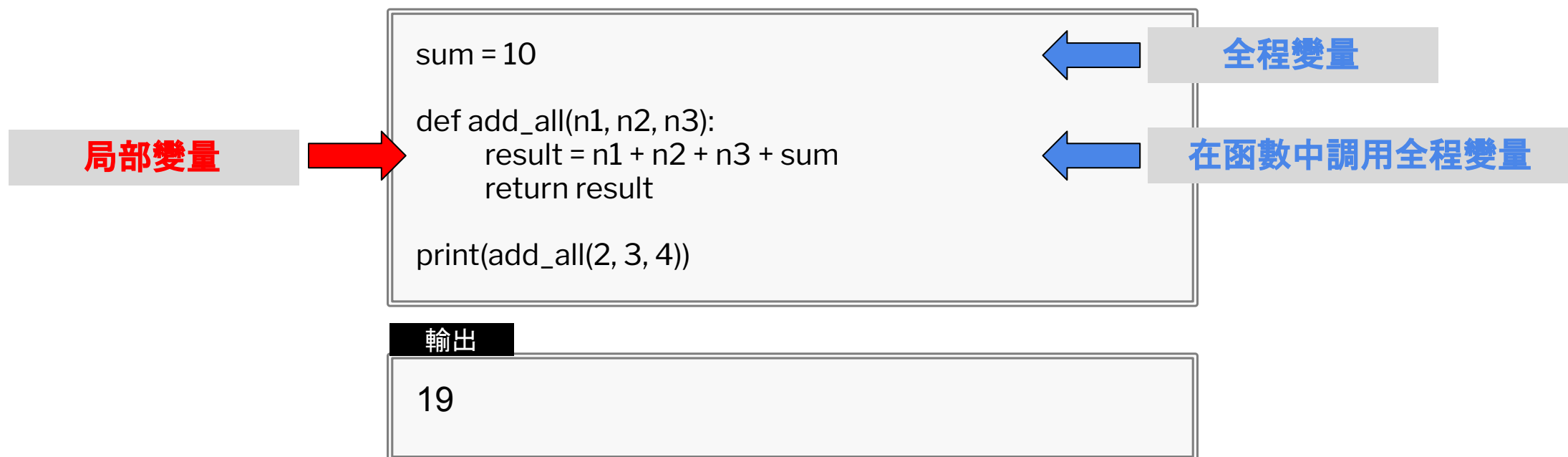
```
5
10
2
The volume is: 100 m^3.
```



# 局部變量 vs 全程變量

變量有其有效的使用範圍：**局部**、**全程**

局部變量	在函數(子程式)內宣告，只在 <b>該函數中有效</b>
全程變量	在函數(子程式)以外的範圍宣告，在 <b>整個程式都可以調用</b> ，包括任何子程式



# 回傳值 - return

```
def 函數名稱 (參數名稱):  
    函數內部的程式碼  
    return
```

# 在這裡，`return`的用法是：結束函數，因為沒有定義資料，所以回傳`None`，這裡的`None`我們稱為回傳值，這個`return`寫不寫也沒有分別。

```
def 函數名稱 (參數名稱):  
    函數內部的程式碼  
    return 資料
```


# 在這裡，`return`的用法是：結束函數，回傳「資料」，這裡的資料我們稱為回傳值。

# 回傳值 - return

有時候，你希望函數輸出結果

```
def get_greeting_msg(guestname, myname='Keith'):
    """display greeting message """
    return f"Good morning {guestname}. My name is {myname}"

msg = get_greeting_msg('Anthony')
print(msg)
```



return 定義 get\_greeting\_msg() 這個函數輸出的值。

呼叫函數，函數輸出的值，可直接賦值給 msg。

輸出

Good morning Anthony. My name is Keith

# return 與 print 的分別

用 **return** 回傳的值可以賦值給一個變量

```
def get_greeting_msg(myname='LP'):  
    """return greeting message """  
    return f"My name is {myname}"
```

```
msg = get_greeting_msg()  
print(msg)
```

問候訊息被回傳並指派給 msg

輸出

My name is Keith

**print()** 僅在控制台顯示值，而不將其存儲

```
def greeting(myname='LP'):  
    """display greeting message """  
    print(f"My name is {myname}")
```

```
msg = greeting()  
print(msg)
```

在函式內執行 **print()**

輸出

My name is Keith  
None

由於函式沒有回傳任何內容，因此沒有任何值被指派給 msg



# 堂課 - 請打開Thonny一起做

# 使用回傳長方形面積

粉紅色斜體: 題目

黑色正常字體: 提供的程式碼

紅色底線: 請填入程式碼

## 程式碼

```
def rectangleArea(length, width):  
    # 建立變量計算面積  
    area = _____ * _____  
    # 回傳面積  
    return _____  
  
# 使用 input()  
length = _____  
width = _____  
  
# 直接使用 rectangleArea() 回傳結果  
print(f"Area is {_____} m^2.")
```

## 輸出

4

2

Area is: 8 m^2.

# 使用回傳計算體積

粉紅色斜體: 題目

黑色正常字體: 提供的程式碼

紅色底線: 請填入程式碼

## 程式碼

```
# 繼續沿用 rectangleArea()
def rectangleArea(length, width):
    .....
# 定義函數 volume(), 傳入參數 height、length、width
def volume(height):
    # 使用 rectangleArea() 回傳的底面積計算體積
    volume = _____ * _____
    # 回傳體積
    return _____

height = input()
length = _____
width = _____
# 直接使用 volume() 回傳結果
print(f"Volume is: {_____} m^3.")
```

## 輸出

```
3
2
5
Volume is: 30 m^3.
```

# 撰寫函數的良好習慣

函數名稱應該具有**描述性**  
而且只包含**小寫字母和底線**

```
def greeting(guestname, myname='LP'):
```

```
    """display greeting message """
```

在函數內部加上**文字簡單說明**  
**函數功能**

```
    print(f"Good morning {guestname}. My name is {myname}")
```

# Python 內建函數

		內建函式		
<a href="#">abs()</a>	<a href="#">delattr()</a>	<a href="#">hash()</a>	<a href="#">memoryview()</a>	<a href="#">set()</a>
<a href="#">all()</a>	<a href="#">dict()</a>	<a href="#">help()</a>	<a href="#">min()</a>	<a href="#">setattr()</a>
<a href="#">any()</a>	<a href="#">dir()</a>	<a href="#">hex()</a>	<a href="#">next()</a>	<a href="#">slice()</a>
<a href="#">ascii()</a>	<a href="#">divmod()</a>	<a href="#">id()</a>	<a href="#">object()</a>	<a href="#">sorted()</a>
<a href="#">bin()</a>	<a href="#">enumerate()</a>	<a href="#">input()</a>	<a href="#">oct()</a>	<a href="#">staticmethod()</a>
<a href="#">bool()</a>	<a href="#">eval()</a>	<a href="#">int()</a>	<a href="#">open()</a>	<a href="#">str()</a>
<a href="#">breakpoint()</a>	<a href="#">exec()</a>	<a href="#">isinstance()</a>	<a href="#">ord()</a>	<a href="#">sum()</a>
<a href="#">bytearray()</a>	<a href="#">filter()</a>	<a href="#">issubclass()</a>	<a href="#">pow()</a>	<a href="#">super()</a>
<a href="#">bytes()</a>	<a href="#">float()</a>	<a href="#">iter()</a>	<a href="#">print()</a>	<a href="#">tuple()</a>
<a href="#">callable()</a>	<a href="#">format()</a>	<a href="#">len()</a>	<a href="#">property()</a>	<a href="#">type()</a>
<a href="#">chr()</a>	<a href="#">frozenset()</a>	<a href="#">list()</a>	<a href="#">range()</a>	<a href="#">vars()</a>
<a href="#">classmethod()</a>	<a href="#">getattr()</a>	<a href="#">locals()</a>	<a href="#">repr()</a>	<a href="#">zip()</a>
<a href="#">compile()</a>	<a href="#">globals()</a>	<a href="#">map()</a>	<a href="#">reversed()</a>	<a href="#">__import__()</a>
<a href="#">complex()</a>	<a href="#">hasattr()</a>	<a href="#">max()</a>	<a href="#">round()</a>	

你不需要記住所有內建函數

- ☐ 常用便會記得
- ☐ 繼而提升編程效率
- ☐ 繼而提升編程體驗
- ☐ 繼而感到生活愉快



# 回顧

# 定義函數

```
def greeting():  
    print("Good morning!")
```

# 參數

```
def get_greeting_msg(guestname, myname='LP'):  
    print(f"Hi {guestname}. My name is {myname}")
```

# 回傳

```
def add(num1, num2):  
    sum = num1 + num2  
    return sum
```

# 局部變量 vs 全程變量

```
sum = 10  
def add(n1, n2, n3):  
    return n1+n2+n3+sum
```

# 位置引數

```
say_morning('LP')
```

# 關鍵字引數

```
say_morning(username='LP')
```

# 位置引數應該在關鍵字引數之前

```
get_greeting_msg('Anthony', myname='LP')
```