

第 2 章

開始撰寫

Python

程式

## 2-1 撰寫 Python 程式的第 1 步

- 程式語言的用途

- 當我們需要電腦做某些事情的時候，因為電腦不懂中文的意義。此時必須改用電腦可以理解的語句，而電腦可以理解的語句就稱為「程式語言」。
- 跟電腦溝通所用的程式語言也有 **Java**、**PHP**、**Python** 等各式各樣的種類。

# 撰寫程式有一定的規則

- 符號的使用方式

- ▼ 不同括號包圍起來的文字

`['蘋果', '橘子', '檸檬']`

`{'蘋果', '橘子', '檸檬'}`

## ● 空格的使用方式

—程式的嚴格之處不只在於符號的使用方式，連程式碼中如何安排空格也是重點

### ▶ 程式碼實例 1

```
>>> def happy():  
...     print('life')
```

### ▶ 程式碼實例 2

```
>>> def happy():  
... print('life')
```

## 2-2起手式：用 Python 做數值計算

- 加法和減法運算（注意空格與英數輸入）



```
>>> 1129 + 2344 ↵  
3473
```



```
>>> 1129 + 2344 ↵  
3473  
>>> 3473 + 376 ↵  
3849  
>>> 400 - 330 ↵  
70
```

- 乘法和除法運算



```
>>> 2800 * 1.08 ↵
```

```
3024.0
```

```
>>> 1920 / 12 ↵
```

```
160.0
```

- 運算的優先順序



```
>>> (40 + 50) * 3 - 50 ↵
```

```
220
```

```
>>> 40 + 50 * 3 - 50 ↵
```

```
140
```

## ● 餘數



```
>>> 255 % 3 ↵
```

```
0
```

```
>>> 255 % 7 ↵
```

```
3
```



- 餘數可以用在哪裡？e.g., 以座號分組...

▼ 利用餘數就能知道是奇數或偶數！

除以 2 的餘數...

為"0"的時候是偶數  
→ 2, 4, 6, 8, 10 等

為"1"的時候是奇數  
→ 1, 3, 5, 7, 9 等

- 次方
- Python標準函式庫：三角函數、指數函數、對數函數



```
>>> 2 ** 3 ↵
```

```
8
```

```
>>> 5 ** 4 ↵
```

```
625
```

# Python 2.x 和 3.x的差異

- `>>> 1931 / 12`
- `160 (2.x)` vs. `160.91666666666666 (3.x)`
- `>>> 1931//12`
- `160 (3.x)`

● 小結

▼ 算術運算子

算術運算子	使用例	意義
+	1 + 1	加法運算
-	2 - 2	減法運算
*	3 * 3	乘法運算
/	4 / 4	除法運算
%	5 % 3	餘數
**	6 ** 2	次方

## 2-3 瞭解存取資料的方式

### ● 變數

- 儲存文字、數字等資料的用途。
- 類似手機通訊錄的記錄方式。
- 寫程式前, 先構思程式執行過程中所需處理的數值或字串之的需要。
- 為資料的暫存或結果取名稱儲存, 之後只要透過這個名稱就能呼叫出資料, 供程式執行使用。
- Python 3 可以使用中文做變數名稱, 但建議以英文取名。
- E.g, tex, price, age...

# 練習指派值給變數....

## 語法

變數名稱 = 內容值



```
>>> tax = 0.08 ↵  
>>> price = 120 ↵  
>>> suzuki_telephone = '0988-123-456' ↵
```

# 叫出變數值練習...



```
>>> tax = 0.08 ↵
```

```
>>> price = 120 ↵
```

```
>>> suzuki_telephone = '0988-123-456' ↵
```

到此是執行過的部分

```
>>> tax ↵
```

```
0.08
```

```
>>> price ↵
```

```
120
```

```
>>> suzuki_telephone ↵
```

```
'0988-123-456'
```

```
>>>
```

# 練習使用變數做運算...



```
>>> price * tax ↵
```

```
9.6
```

```
>>> 120 * 0.08 ↵
```

```
9.6
```



# 變數的命名規則（非常重要）

- 第1個文字不能是數字
- 不能使用「保留字」(ref. pp.2-17)



```
>>> value = 100 ↵
```

```
>>> _value = 300 ↵
```

```
>>> 2value = 500 ↵
```

```
File "<stdin>", line 1
```

```
    2value = 500
```

```
        ^
```

```
SyntaxError: invalid syntax
```

# 練習用保留字會出現什麼...

## finally & global



```
>>> finally = 888 ↵  
File "<stdin>", line 1  
    finally = 888  
          ^  
SyntaxError: invalid syntax
```



```
>>> global = 127 ↵  
File "<stdin>", line 1  
    global = 127  
          ^  
SyntaxError: invalid syntax
```

# 動手查察有哪些保留字...

- 保留字會因版本不同有異



```
>>> import keyword ↵  
>>> keyword.kwlist ↵
```

## ▼ Python 的保留字

False	None	True	and	as	assert	break
class	continue	Def	del	elif	else	except
finally	for	From	global	if	import	in
is	lambda	nonlocal	not	or	pass	raise
return	try	while	with	yield		

# 2-4哪邊比較多？比較大？

- 什麼是比較運算子？

- 程式中的比較是以資料為對象。
- 執行「哪邊數字比較大？」之類的判斷。

# 試著使用比較運算子



```
>>> 34 > 22 ↵
```

```
True
```



```
>>> 34 < 22 ↵
```

```
False
```

# 這些都是比較運算子

## ▼ 比較運算子一覽表

比較運算子	實例	意義
>	$x > y$	x 比 y 大
>=	$x \geq y$	x 大於或等於 y
<	$x < y$	x 比 y 小
<=	$x \leq y$	x 小於或等於 y
==	$x == y$	x 與 y 相等
!=	$x != y$	x 不等於 y

'==' 與 '=' 不同

● 等於 vs. 指派



```
>>> apple = 15 ↵ ● ————— 指派數值給變數
>>> apple == 15 ↵ ● ————— 將變數 apple 和 15 做比較，看是否相等
True
```

## 2-5 Python 的各類型資料

- 資料：存在於世上的各種資訊（如何用Python處理）。
- 為了正確或是更加便利存取資料，程式語言會提供不同資料型別（資料的類型）和使用方式。
- Python有....
  - 「數值型別」、「字串型別」
  - 「邏輯型別」、「串列型別」、「字典型別」、「集合型別」（這 4 種也被合稱為「容器型別」）
- 寫程式時未告知程式語言所處理的資料是何種型別，程式語言就不會處理或處理不正確（結果也就待質疑）。



# 3種數值型別

- 為了執行數學運算或計數
- Python有三種：整數、浮點數（小數）、複數
- 英文：int, float, complex

# 整數(int)

- 整數, e.g, 1,2,10...
- 有時數值也會被以文字處理，例如電話號碼、學號

```
>>> 34 + 56 ↵
```

```
90
```

```
>>> number = 55 ↵
```

# 浮點數(float)

- 浮點數（有小數點的數值）

```
>>> 5 + 3.4 ↵
```

```
8.4
```

```
>>> 5 / 2 ↵
```

```
2.5
```


# 複數 ( complex)

- 很少用，需數以 **j or J** 表達
- 虛數為**1**時，不能審略
- 請練習...

```
>>> complex = 5 + 5j ↵  
>>> complex + (3 + 1j) ↵  
(8+6j)
```

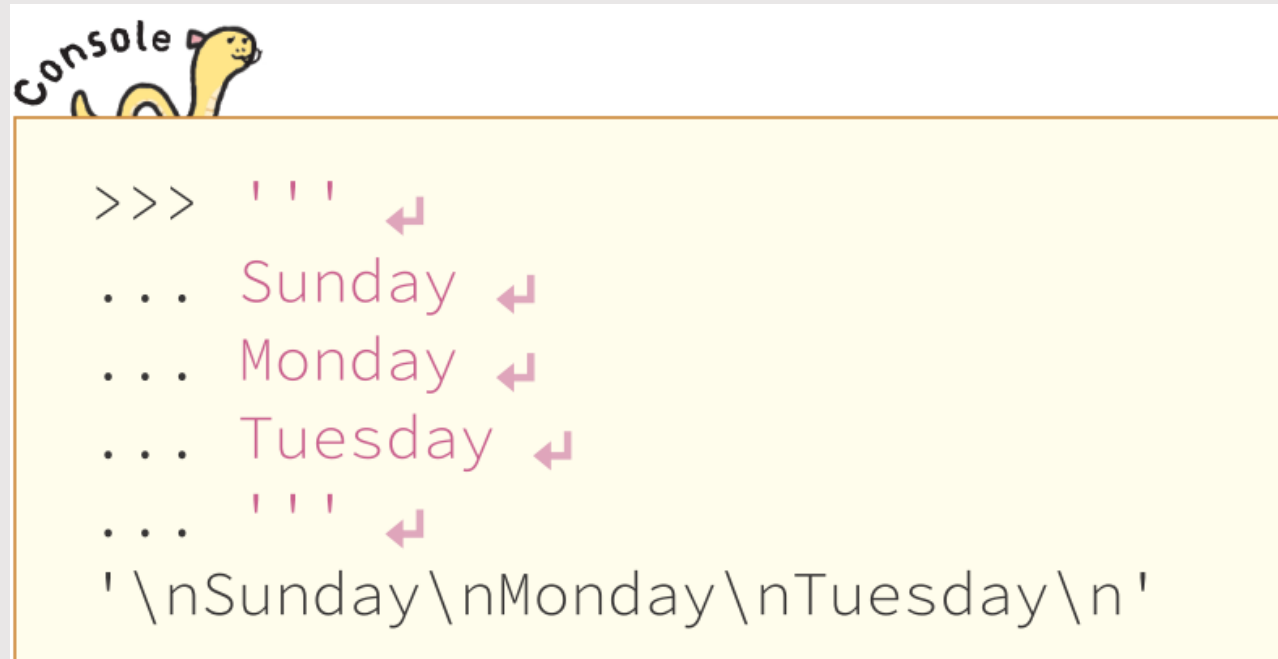
# 字串型別

- 字串型別: 用單引號或雙引號為著的資料 (英數)

```
>>> 'happy birthday!!'   
'happy birthday!!'  
>>> message = "生日快樂!"
```

# 3個連續的單引號或雙引號:做多行的字串輸入(請練習)

- 「\n」叫電腦換行的符號



```
>>> '''  
... Sunday  
... Monday  
... Tuesday  
... '''  
'\nSunday\nMonday\nTuesday\n'
```

做字串術運算：「+」、「\*」

- 使用 + 號合併字串 (same type only)



```
>>> 'thunder' + 'bolt' ↵  
'thunderbolt'
```

```
>>> 'thunder' + 100 ↵  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: Can't convert 'int' object to str implicitly
```

## \*號用在重複字串



```
>>> 'hunter' * 2 ↵  
'hunterhunter'
```

```
>>> 'dragon' * 'head' ↵
```

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

```
TypeError: can't multiply sequence by non-int of type 'str'
```



何時會把數值宣告為  
字串？

# 字串型別相關的函數：upper(), lower(), count()



```
>>> text = 'hello' ↵  
>>> text.upper() ↵  
'HELLO'
```



```
>>> word = 'maintenance' ↵  
>>> word.count('n') ↵  
3
```

# 邏輯型別（布林值）：True, False

- 第一個字母要大寫，否則不是。



```
>>> 46 < 49 ↵
```

```
True
```

```
>>> 46 > 49 ↵
```

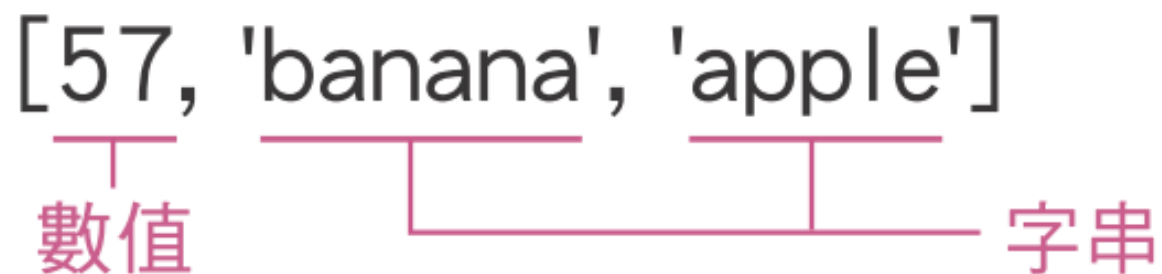
```
False
```

## 串列型別 ( List ) : []; 逗號區隔

- 最大特別在可以匯集不同的資料型別

### ▼ 將數值和字串資料彙集在一起

[57, 'banana', 'apple']



數值                      字串

```
>>> Agroup = ['kazu', 'gorou'] ← 這段代表的意義為 A 團體內的 2 名成員  
>>> Bgroup = ['syun', 'haruka'] ← 這段代表的意義為 B 團體內的 2 名成員
```

E,g, 如何以List儲存五月天團員

# 在List中加入新元素: Append()

- Append(): 在串列中加入新元素



## 加入或退出串列



```
>>> Agroup = ['kazu', 'gorou'] ↵  
>>> Agroup.append('dai') ↵  
>>> Agroup ↵  
['kazu', 'gorou', 'dai']
```

# 從List中刪除元素: remove()

- 從串列中刪除元素: remove()
- 自舉例練習



```
>>> Agroup = ['kazu', 'gorou', 'dai']  
>>> Agroup.remove('kazu')  
>>> Agroup  
['gorou', 'dai']
```

# 改變List元素的順序：sort()

- 改變串列中元素的順序



```
>>> Agroup = ['kazu', 'gorou', 'dai'] ↵  
>>> Agroup.sort() ↵  
>>> Agroup ↵  
['dai', 'gorou', 'kazu']
```

# List中為數值也可以用sort()排序



```
>>> test_result = [87, 55, 99, 50, 66, 78] ↵  
>>> test_result.sort() ↵  
>>> test_result ↵  
[50, 55, 66, 78, 87, 99]
```



# List中為數值和字串無法用sort()排序

```
>>> mix_list = [85, 'kazu', 'dai', 100] ↵  
>>> mix_list.sort() ↵  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: unorderable types: str() < int()
```

# 字典型別 ( Dictionary Type )

- 利用索引(**key**) 找到資料(數值或字串)

## ▼ 中文辭典的格式

蘋果 . . . . . 蘋果指的是薔薇科蘋果屬的落葉喬木或其果實， . . . . .

▪

▪

▪

路由器 . . . . . 電腦網路之間負責轉接的通訊機器 . . . . .

▪

▪

▪

# 索引與標題成對; 接受不同資料型態

- '{ }'; ', '

## 語法

{索引標題 1:內容資料 1, 索引標題 2:內容資料 2, ...}

```
{'1stClass': 65, '2ndClass':55, '3rdClass':45}
```

- 用途....

# How to apply? ...try ...

```
>>> activities = {'Monday': '籃球', 'Tuesday': '自行車',  
                  'Wednesday': '輕音樂', 'Friday': '游泳'} ↵
```



```
>>> activities['Tuesday'] ↵  
'自行車'  
>>> activities['Friday'] ↵  
'游泳'
```

# Methods of Dictionary Type...

- 字典典型別可用的方法：
- `keys()`: show index/key
- `values()`: show content or value



```
>>> activities.keys() ↵  
dict_keys(['Wednesday', 'Friday', 'Tuesday', 'Monday'])  
>>> activities.values() ↵  
dict_values(['輕音樂', '游泳', '自行車', '籃球'])
```

# 序對型別 ( Tuple Type ) (組)

- Tuple: 意思為「多個元素組成一個群組」
- '()'; ',' #like List
- 也接受不同型別的元素，如字串與數值 #與List相同

## 語法

(元素 A, 元素 B, 元素 C, . . . . .)



```
>>> tuple_sample = ('apple', 3, 90.4) ↵  
>>> print(tuple_sample) ↵  
( 'apple', 3, 90.4)
```

# 意思？用在哪？



```
>>> tuple_sample = ('apple', 3, 90.4) ↵  
>>> print(tuple_sample) ↵  
( 'apple', 3, 90.4)
```

# Tuple Type與List Type有何不同

- 1. 宣告後不能再變更存放的資料
- Try....



```
>>> flavor_list = ['薄荷', '巧克力', '草莓', '香草']
>>> flavor_list[0] = '蘭姆葡萄乾'
>>> print(flavor_list)
```

宣告串列型別

['蘭姆葡萄乾', '巧克力', '草莓', '香草']



```
>>> flavor_tuple = ('薄荷', '巧克力', '草莓', '香草')
>>> flavor_tuple[0] = '蘭姆葡萄乾'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    TypeError: 'flavor_tuple' object does not support item
assignment
>>> print(flavor_tuple)
```

宣告序對型別

錯誤訊息

('薄荷', '巧克力', '草莓', '香草')



# Tuple Type與List Type有何不同

- 2. Tuple可用來當Dictionary type 的 key，List  
不能用來當Dictionary type 的 key
  - 因為Tuple值不能再被異動
- Try pp.2-37~2-38 upper....

# Try ...

- 1. diary by Dictionary
- 2. key by Tuple (non-change)
- 3. assign value by key as index of diary



```
>>> diary = {}  
>>> key = ('kamata', '08-01')  
>>> diary[key] = '70kg'  
>>> print(diary)  
{('kamata', '08-01'): '70kg'}
```

字典型別  
序對型別

# Error ....

- Tuple可以當Dictionary的Key, List 不行
- Key 為不能異動的值



```
>>> diary = {}
>>> key = ['nakata', '08-01']
>>> diary[key] = '50kg'
```

Traceback (most recent call last):  
File "<stdin>", line 1, in <module>  
TypeError: unhashable type: 'list'

字典型別  
串列型別

# t如何記錄某人在某天的體重...

- **Tuple**可以當**Dictionary**的**Key**使用時機...
- 多項資料成為一個**key**時...
- 用在哪裡...?

```
>>> diary['kamata', '08-03'] ←  
'72kg'  
>>> diary['nakata', '08-09'] ←  
'58kg'  
>>> diary['nakata', '08-04'] ←  
'53kg'
```

# 集合型別 ( Set Type )

- List, Tuple, Set 都可以存多項資料在一起的資料型別
- 使用'{'}' (和Dictionary相同),
- 資料存放和List, Tuple相同



```
>>> candy = {'delicious', 'fantastic'}  
>>> print(candy)  
{'delicious', 'fantastic'}
```

# set() 函數

- 注意！拆解後的字母會變化...隨機...
- 不留相同的資料...
- 用在哪裡？
- 試試中文字串... `set('不留相同不留相同')`
  - {'不', '留', '同', '相'}



```
>>> candy = set('delicious')  
>>> print(candy)  
{ 'd', 'u', 's', 'l', 'e', 'o', 'c', 'i' }
```

# 使用 set 函式建立集合型別資料

- `set()` 可以將 `List` 轉為 `Set`...
- 每一個字元沒有被打散，因為使用了 `List`
- 在 `update()` 裡用 `[]` 可以增加 `Set` 的資料...
- 應用：使用 `List` 來新增資料，再以 `set()` 來轉成 `Set`，再以 `update([])` 來添加新元素...



```
01 >>> flavor = ['apple', 'peach', 'soda']  ◀ 宣告串列型別
02 >>> candy = set(flavor)  ◀
03 >>> candy  ◀
04 {'peach', 'apple', 'soda'}  ◀ 轉換成集合型別
05 >>> candy.update(['grape'])  ◀
06 >>> candy  ◀
07 {'peach', 'apple', 'grape', 'soda'}  ◀
```

## 集合型別的便利使用技巧 1 ~ 刪除重覆元素 ~

- 1. 建立List data (有重複)
- 2. 透過set()去掉重複(但變為Set type了)
- 4. 用list()將Set type轉回List type
- set() list()用在何處?



```
01 >>> flavor = ['apple', 'soda', 'chocolate', 'apple', 'grape',  
                'grape', 'soda']  
02 >>> flavor_set = set(flavor)  
03 >>> print(flavor_set)  
{'grape', 'soda', 'apple', 'chocolate'}  
04 >>> flavor = list(flavor_set)  
05 >>> print(flavor)  
['grape', 'soda', 'apple', 'chocolate']
```



# 集合型別的便利使用技巧 2 ~ 集合型別間的運算 ~

- 1. Set type, 2. Set type
- 集合型別可以用比較與運算的(List不行)



```
01 >>> flavor_set_A = {'apple', 'peach', 'soda'} ↵  
02 >>> flavor_set_B = {'apple', 'soda', 'chocolate'} ↵  
03 >>> flavor_set_A - flavor_set_B ↵  
{'peach'}  
04 >>> flavor_set_A & flavor_set_B ↵  
{'apple', 'soda'}
```

flavor\_set\_A - flavor\_set\_B

flavor\_set\_A & flavor\_set\_B

## ● Try below...



```
01 >>> flavor_set_A = {'apple', 'peach', 'soda'} ↵  
02 >>> flavor_set_B = {'apple', 'soda', 'chocolate'} ↵
```

### ▼ 集合型別可使用的符號與功能

符號	功能
$A \leq B$	判斷 B 集合是否包含 A 集合的所有元素
$A \geq B$	判斷 A 集合是否包含 B 集合的所有元素
$A \mid B$	將 A 和 B 集合的所有元素組合成新的集合資料（聯集）
$A \& B$	將 A 和 B 集合共同的元素組合成新的集合資料（交集）
$A - B$	將 A 有但 B 沒有的元素組合成新的集合資料（差集）
$A \wedge B$	將 A 和 B 共同元素之外的元素組合成新的集合資料（XOR）

# 小結

- 數值型別

- 整數

```
data_type_integer = 89
```

- 浮點數

```
data_type_float = 0.89
```

- 複數

```
data_type_complex = 8+9j
```

- 字串型別

```
data_type_string = 'lucky 7'  
data_type_string = "lucky 7"
```

- 串列 **List** 型別

```
data_type_list = ['歐蕾咖啡', '摩卡咖啡', 980]
```

- 字典型別

```
data_type_dictionary = {1: 'January', 2: 'February', 3: 'March'}
```

- 序對 **Tuple** 型別

```
data_type_tuple = ('雞', '牛', '豬')
```

- 集合 **Set** 型別

```
data_type_set = {'Python', 'Ruby', 'PHP'}
```

```
data_type_set = set(['Python', 'Ruby', 'PHP'])
```