

# 資料類別與基本運算

當我們要在 Python 中建立資料時，可以將資料指定給一個物件 (或稱為變數)。Python 所提供的資料類別中，常見且重要的有布林值 (boolean)、數字 (number)、字串 (string)、列表 (list)、元組 (tuple) 和字典 (dictionary)。我們將在接下來的內容中做進一步的介紹。

底下我們列出有關在 Python 中建立物件名稱之一些命名規則：

- (1) 物件名稱可由英文字母、數字和底線符號 ‘\_’ 組成。
- (2) 英文字母大小寫有所區別 (Beauty 和 beauty 是代表完全不同的兩個物件)。
- (3) 物件名稱須起始於英文字母或底線符號 ‘\_’。
- (4) 關鍵字和內建名稱不能當作物件名稱。
- (5) 要避免同時使用底線符號 ‘\_’ 作為物件名稱的開頭與結尾字元。

有時為了更容易記住物件的內涵，在取物件名稱時我們常會將好幾個英文字合起來組成一個物件名稱。下面這些物件名稱皆是合乎規則的：

```
guy, handsome_guy, you_handsome_guy
```

若使用太簡略的縮寫來當物件名稱，有時常常會忘記此物件的內涵；比方說 LKK 是代表什麼內涵的物件呢？難道是指像我們作者這種年紀的人嗎？

要如何知道 Python 的關鍵字？執行下列指令就可看到，

```
>>> import keyword

>>> keyword.kwlist
['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']
```

在建立物件名稱之前，要如何知道 Python 內建名稱以及在工作區 (workspace) 中已經有那些物件了呢？為了底下說明之方便，首先我們重新啟動程式工作環境。這可以在 Python IDLE 選單所提供的 Shell 項目點選 Restart 即可重新啟動工作環境。

首先設定工作目錄，例如 D:\Practical-Python-Programming\Python-Files：

```
>>> import os

>>> mywd = "D:\\Practical-Python-Programming\\Python-Files"
>>> os.chdir(mywd)
```

我們可以查看設定好的工作目錄：

```
>>> os.getcwd()
'D:\\Practical-Python-Programming\\Python-Files'
```

接著我們查看工作區有那些物件：

```
>>> dir()
['__builtins__', '__doc__', '__loader__', '__name__',
'__package__', '__spec__', 'mywd', 'os']
```

請注意前六個物件是在開啟工作環境後伴隨而來的內建物件。如果要知道這些內建物件內的名稱也可以使用 `dir()` 這一個指令，例如：

```
>>> dir(__builtins__)
['ArithmeticError', 'AssertionError', 'AttributeError',
'BaseException',
', , ,
'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super',
'tuple', 'type', 'vars', 'zip']
```

接著產生一些物件：

```
>>> x = 1
>>> y = [1, 2]
>>> z = [1, 2, 3]
```

我們可以使用 `type()` 指令來檢查物件的資料類別，例如：

```
>>> type(x)
<class 'int'>

>>> type(y)
<class 'list'>
```

這說明了 `x` 是整數物件，`y` 是列表 (`list`)。

再來查看工作區有那些物件：

```
>>> dir()
['__builtins__', '__doc__', '__loader__', '__name__',
'__package__', '__spec__', 'mywd', 'os', 'x', 'y', 'z']
```

假如我們並不想列出那些在開啟工作環境所伴隨而來物件的名稱，而只想列出那些被定義變數 (`defined variables`) 的名稱，該如何做呢？我們可以定義下列的兩個函數：

```
>>> def ls():
    return [var for var in globals().keys() if var[0] != "_"]
```

```
>>> def objects():
    return [var for var in globals().keys()
            if not var.startswith('_')]
```

有關自訂函數在第 5.1 節有詳細的介紹。接著來驗證這兩個函數的正確性：

```
>>> ls()
['x', 'y', 'os', 'objects', 'ls', 'z', 'mywd']

>>> objects()
['x', 'y', 'os', 'objects', 'ls', 'z', 'mywd']
```

結果是正確的。

若要知道在工作目錄中有那些檔案，可以使用 `os.listdir()` 函數：

```
>>> os.listdir()
```

要如何刪除工作區中的物件呢？要注意的是刪除那些在開啟工作環境伴隨而來的物件是不智的。因此我們只要刪除那些自行定義的變數。要如何進行呢？我們可以定義下列的函數：

```
>>> def clearall():
    all = [var for var in globals() if var[0] != "_"]
    for var in all:
        del globals()[var]
```

接著來驗證這個函數的正確性：

```
>>> clearall()

>>> dir()
['__builtins__', '__doc__', '__loader__', '__name__',
 '__package__', '__spec__']
```

結果是正確的。

在控制台輸入資料時，若要使用先前鍵入或出現的指令，我們可以使用鍵盤上面的↑或↓兩個方向鍵，也可以直接在螢幕上剪貼所需之指令。若為節省空間或是將相關的一些指令打在同一列上，則只需使用符號 ";" 把各個指令分開即可。

## 2.1 布林值

程式中常會用到判斷真偽的指令，例如 `if` 或 `while`，其判斷結果不是 `True` 就是 `False`，這就是布林值。除了判斷指令之外，還有一些布林運算操作，其結果也是布林值。若將布林值轉換成數值，則 `True` 為 1，`False` 為 0。布林與比較運算指令如表 2.1.1 所示。

表 2.1.1：布林與比較運算指令

運算子	說明
<code>not x</code>	NOT 運算
<code>x &amp; y</code> <code>x and y</code>	AND 運算
<code>x   y</code> <code>x or y</code>	OR 運算
<code>&lt;</code>	小於
<code>&lt;=</code>	小於等於
<code>&gt;</code>	大於
<code>&gt;=</code>	大於等於
<code>==</code>	等於
<code>!=</code>	不等於
<code>is</code>	比較物件是否相同
<code>is not</code>	比較物件是否相異
<code>x in s</code>	檢查 <code>x</code> 是否存在 <code>s</code> 中
<code>x not in s</code>	檢查 <code>x</code> 是否不存在 <code>s</code> 中

簡單的布林運算如下所示：

```
>>> x = True; y = False

>>> x and y
False

>>> x or y
True

>>> not x
False

>>> a = 9; b = 5

>>> a > b
True

>>> a < b
False

>>> a >= b
True

>>> a <= b
False

>>> a == b
False

>>> a != b
True

>>> y = a

>>> y is a
True
```

```
>>> y is not a
False
```

## 2.2 數字

Python 提供三種不同的數字類別：整數 (int)、浮點數 (float) 及複數 (complex)。基本數學運算子如表 2.2.1 所示。

表 2.2.1：基本數學運算子

運算子	說明
+, -, *, /	加、減、乘、除
//	整數除法取商
%	整數除法取餘數
+x, -x	單運算元，取正和取負
abs(x)	取絕對值
int(x)	將x變為整數 (整數位無條件捨去)
float(x)	將x變為浮點數
complex(re, im)	複數的實部 re, 虛部 im
c.conjugate()	c 的共軛複數
divmod(x, y)	(x//y, x%y)
pow(x, y) x ** y	x 的 y 次方

簡單的基本數學運算如下：

```
>>> 3.6 + 1.25
4.85

>>> x = 3.6; y = 1.25

>>> x + y
```

```
4.85
```

```
>>> x - y
```

```
2.35
```

```
>>> x * y
```

```
4.5
```

```
>>> x / y
```

```
2.88
```

```
>>> z = int(x)
```

```
>>> z
```

```
3
```

```
>>> float(z)
```

```
3.0
```

```
>>> w = -x
```

```
>>> w
```

```
-3.6
```

```
>>> abs(w)
```

```
3.6
```

```
>>> a = 5; b = 2
```

```
>>> a // b
```

```
2
```

```
>>> a % b
```

```
1
```

```
>>> divmod(a, b)
```

```
(2, 1)
```

```
>>> x = 2; y = 3
```



```
>>> x**y
8

>>> pow(y, x)
9

>>> c = 1 + 1j; d = complex(2, 2)

>>> d
(2+2j)

>>> c + d
(3+3j)

>>> d.conjugate()
(2-2j)
```

## 2.3 字串

Python 接受使用者以單引號或雙引號內的字元來建立字串，並可將引號內的文字指定給一字串變數。例如：

```
>>> color = 'red'
>>> look = "beautiful flower"
```

我們可以使用切片運算符 (`[]` 和 `[:]`) 與指標來取得字串變數內的字元。變數內字元的順序若從左到右的話，指標則分別為 `0, 1, 2, ...`，若從右到左的話，指標則分別為 `-1, -2, -3, ...`。使用加號 `+` 可以進行字串連接運算，乘號 `*` 進行字串重複操作。舉例如下：

```
>>> print(look)
beautiful flower

>>> print(look[4])
t
```

```
>>> print(look[2:6])
auti

>>> print(look[-6:-3])
flo

>>> look1 = look[0:9]
>>> print (look1)
beautiful

>>> look2 = look[-6:]
>>> print (look2)
flower

>>> look3 = look1 + ' ' + color + ' ' + look2
>>> print('It is a ' + look3 + '.')
It is a beautiful red flower.

>>> print((look + ' ') * 3)
beautiful flower beautiful flower beautiful flower
```

## 2.4 列表

列表 (list) 是 Python 最常用的一種資料類型。一個列表的內容是以中括弧 "[]" 包圍起來的資料項目，這些項目是以逗號 "," 分開的。一個空列表可以使用空的中括弧 [] 設定，也可以使用 list() 指令設定。依實際需要，列表中的資料類別可以是完全相同的，也可以是不相同的，而且這些資料項目的位置順序是重要的。也就是說，列表是一些有順序資料項的一個集合。存放在列表中的資料項目可以使用切片運算符 ([ 和 [:]) 與指標來取得，列表內資料項目的順序若從前到後的話，指標則分別為 0, 1, 2, ...，若從後面算起的話，指標則分別為 -1, -2, -3, ...。使用加號 "+" 可以進行資料項目連接運算，乘號 "\*" 進行資料項目重複操作。例如：

```
>>> a = []; b = list()
>>> a; b
```

```
[ ]

>>> type(a); type(b)
<class 'list'>

>>> list_1 = [5, 3, 2, 9, 7]
>>> list_2 = ['red', 123, 4.56, 'flower']

>>> list_1; list_1[:]
[5, 3, 2, 9, 7]

>>> list_1[3]
9

>>> list_1[1:4]
[3, 2, 9]

>>> list_1[3:]
[9, 7]

>>> list_2[-1]
'flower'

>>> list_2[:2]; list_2[0:2]
['red', 123]

>>> list_2[-3:]; list_2[1:]
[123, 4.56, 'flower']

>>> list_2[-3:-1]; list_2[1:-1];
[123, 4.56]

>>> list_3 = list_1 + [10, 100, 1000]
>>> list_3
[5, 3, 2, 9, 7, 10, 100, 1000]

>>> list_4 = list_2 + [50.3, 'beautiful']
>>> list_4
```

```
['red', 123, 4.56, 'flower', 50.3, 'beautiful']

>>> list_2*3
['red', 123, 4.56, 'flower', 'red', 123, 4.56, 'flower',
'red', 123, 4.56, 'flower']
```

除了使用加號 "+" 來增加列表的內容之外，也可使用 `append()`、`extend()` 與 `insert()` 這三個指令增加列表內容。舉例如下：

```
>>> list_2.append(99) # 在列表變數最後面增加一個資料項目
>>> list_2
['red', 123, 4.56, 'flower', 99]

>>> list_2.extend(['banana', 1.99]) # 將一列表之資料項目依序增加到列表的最後面
>>> list_2
['red', 123, 4.56, 'flower', 99, 'banana', 1.99]

>>> list_2.insert(2, 'better') # 將資料項目better插入到原列表指標2的位置
>>> list_2
['red', 123, 'better', 4.56, 'flower', 99, 'banana', 1.99]
```

尋找列表內資料內容的操作有下列幾種，如下所示：

```
>>> list_5 = ['red', 'blue', 'green', 'yellow', 'red' ]
>>> list_5.count('red') # 計算出資料項目在列表變數中的個數
2

>>> 'green' in list_5 # 檢查資料項目是否在列表變數之中
True

>>> 'black' in list_5
False
```

```
>>> list_5.index('yellow') # 找出資料項目在列表變數中第一次出現的
                               位置指標
3

>>> list_5.index('white')
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    list_5.index('white')
ValueError: 'white' is not in list
```

從列表中刪除資料項目的操作有下列幾種，如下所示：

```
>>> list_6 = ['banana', 10, 'grape', 11, 'apple']

>>> del list_6[1]           # 利用指標刪除資料項目
>>> list_6
['banana', 'grape', 11, 'apple']

>>> list_6.remove('grape') # 利用資料項目名稱或數值移除資料項目
>>> list_6
['banana', 11, 'apple']

>>> list_6.remove(11)
>>> list_6
['banana', 'apple']

>>> list_6.remove('banana', 'apple')
# 一次只能移除一個資料項目，否則會出現錯誤訊息
Traceback (most recent call last):
  File "<pyshell#58>", line 1, in <module>
    list_6.remove('banana', 'apple')
TypeError: remove() takes exactly one argument (2 given)

>>> list_6.remove('grape') # 要刪除的資料不在列表變數中會出現錯誤訊息
Traceback (most recent call last):
  File "<pyshell#59>", line 1, in <module>
    list_6.remove('grape')
ValueError: list.remove(x): x not in list
```

除了 `del` 與 `remove()` 之外，`pop()` 指令也可以刪除列表之資料項目，如下所示：

```
>>> list_7 = ['red', 'blue', 'green']
>>> list_7.pop() # 刪除列表中最後一個資料項目並將它顯示出來
'green'

>>> list_7
['red', 'blue']

>>> list_7.pop(1) # 移除列表中被指定之指標位置之資料項目並將它顯示出來
'blue'

>>> list_7
['red']

>>> list_7.pop()
'red'

>>> list_7
[]

>>> list_7.pop() # 移除空集合列表之資料項目會出現錯誤訊息
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    list_7.pop()
IndexError: pop from empty list
```

## 2.5 元組

元組 (tuple) 就相當於是一個產生之後就不可改變的列表。一個元組的內容是以逗號 "," 分開的資料項目，習慣上會以小括弧 () 將這些資料項目包圍起來。單一個資料項目的元組在資料項目後面一定要有一個逗號。一個空元組可以使用空的小括弧 () 設定，也可以使用 `tuple()` 指令設定。例如：