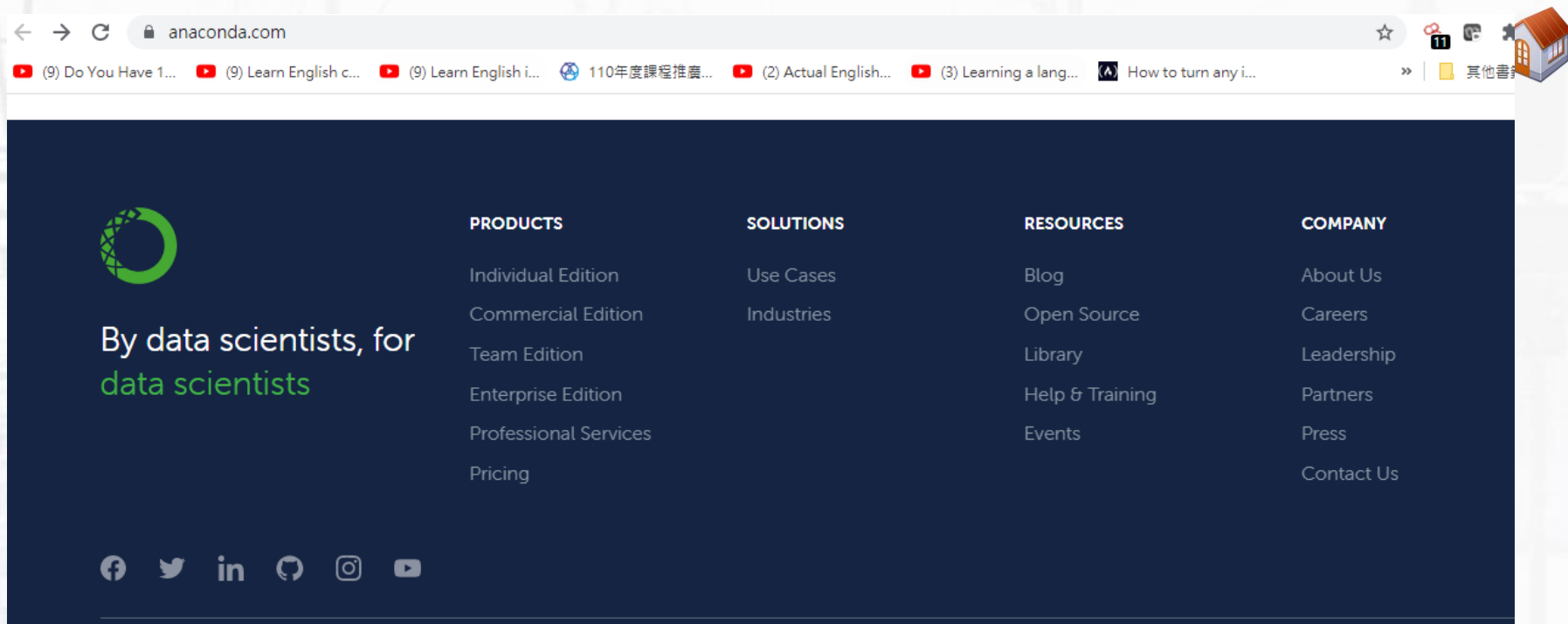




# 第1章 Python程式設計入門 (Introduce basic python language)

- 1-1 Python變數、資料型別與運算子
- 1-2 流程控制
- 1-3 函式、模組與套件
- 1-4 容器型別
- 1-5 類別與物件
- 1-6 檔案與例外處理






anaconda.com/products/individual-d

Individual Edition

# Your data science toolkit

With over 25 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.




## Anaconda Individual Edition

Download 

For Windows

Python 3.8 • 64-Bit Graphical Installer • 477 MB

Get Additional Installers

 |  | 



# 1-1 Python變數、資料型別與運算子 (Python variable, data type and operator)

- 1-1-1 使用Python變數
- 1-1-2 Python運算子
- 1-1-3 基本資料型別





## 1-1-1 使用Python變數 (use variables)

- Python是Guido Van Rossum開發的程式語言，這是一種優雅語法和高可讀性程式碼的程式語言，可以開發**GUI**視窗程式、**Web**應用程式、系統管理工作、財務分析和大數據資料分析等各種不同應用程式。Python語言分成2和3版，在本書是使用Python 3語言。
- 「變數」(Variables)是用來儲存程式執行期間的暫存資料，變數值就是指定資料型別(Data Types)的資料，例如：整數、浮點數、布林和字串值等。





## 1-1-1 使用Python變數(use variables)

- Python的變數不需宣告，只需指定變數值，就可以馬上建立變數，請注意！Python變數在使用前一定需要指定初值（Python程式：ch1-1-1.py），如下所示：

```
grade = 76
```

```
height = 175.5
```

```
weight = 75.5
```

```
print("成績 = " + str(grade))
```

```
print("身高 = " + str(height))
```

```
print("體重 = " + str(weight))
```

```
print("成績 =", grade)
```

```
print("身高 =", height)
```

```
print("體重 =", weight)
```

Run the above program





## 1-1-2 Python運算子

運算子	說明
()	brackets 括號運算子
**	exponent 指數運算子
~	位元運算子NOT
+ 、 -	正號、負號
* 、 / 、 // 、 %	算術運算子的乘法、除法、整數除法和餘數
+ 、 -	算術運算子加法和減法
&	位元運算子AND
^	位元運算子XOR
	位元運算子OR
in 、 not in 、 is 、 is not 、 < 、 <= 、 > 、 >= 、 <> 、 != 、 ==	成員、識別和關係運算子小於、小於等於、大於、大於等於、不等於和等於
not	邏輯運算子NOT
and	邏輯運算子AND
or	邏輯運算子OR
<< >>	向左、向右位移

Multiplication  
Division  
remainder

less than or equal to  
greater than  
greater than or equal to



## 1-1-2 Python運算子

- Python運算式的多個運算子如果擁有相同的優先順序時，如下所示：

**3 + 4 - 2 Please print the result of 3 + 4 - 2**

- 上述運算式的「+」和「-」運算子擁有相同優先順序，此時的運算順序是從左至右依序的進行運算，即先運算  $3+4=7$ ，然後再運算  $7-2=5$ 。請注意！Python多重指定運算式是一個例外，如下所示：

**a = b = c = 25 Please print the result of a**

- 上述多重指定運算式是從右至左，先執行  $c = 25$ ，然後才是  $b = c$  和  $a = b$ （所以變數  $a$ 、 $b$  和  $c$  的值都是 25）。
- 印出 3 + 4 - 2
- 請印出上面的 a



## 1-1-3 基本資料型別 整數（Integers）

- 整數資料型別是指變數儲存資料是整數值，沒有小數點，其資料長度可以是任何長度，視記憶體空間而定。例如：一些整數值範例，如下所示：

$a = 1$

$b = 100$

$c = 122$

$d = 56789$







## 1-1-3 基本資料型別 整數 (Integers)

- Python變數可以指定成整數值後，使用變數來執行相關運算（Python程式：ch1-1-3.py），如下所示：

```
x = 5
```

```
print(type(x)) # 顯示 "<class 'int'>"
```

```
print(x)      # 顯示 "5"
```

```
print(x + 1)  # 加法: 顯示 "6"
```

```
print(x - 1)  # 減法: 顯示 "4"
```

```
print(x * 2)  # 乘法: 顯示 "10"
```

```
print(x / 2)  # 除法: 顯示 "2.5"
```

Please show the class of x  
顯示x的類型





## 1-1-3 基本資料型別 整數 (Integers)

```
print(x // 2) # 整數除法: 顯示 "2"
```

```
print(x % 2) # 餘數: 顯示 "1"
```

```
print(x ** 2) # 指數: 顯示 "25"
```

```
x += 1
```

```
print(x) # 顯示 "6"
```

```
x *= 2
```

```
print(x) # 顯示 "12"
```



Run the above program



## 1-1-3 基本資料型別

### 浮點數（Floats）

- 浮點數資料型別是指變數儲存的是整數加上小數，其精確度可以達小數點下15位，基本上，**整數和浮點數的差異在是否有小數點**，**5是整數；5.0是浮點數**，例如：一些浮點數值的範例，如下所示：

```
e = 1.0
```

```
f = 55.22
```

- Python浮點數的精確度只有到小數點下15位。同樣的，Python變數可以指定成浮點數值後，使用變數來執行相關運算（Python程式：ch1-1-3a.py），如下所示：

```
y = 2.5
```

```
print(type(y)) # 顯示 "<class 'float'>"
```

```
print(y, y + 1, y * 2, y ** 2) # 顯示 "2.5 3.5 5.0 6.25"
```



## 1-1-3 基本資料型別

### 布林（Booleans）

- Python語言的布林（Boolean）資料型別是使用True和False關鍵字來表示，如下所示：  
`x = True`  
`y = False`
- 除了使用True和False關鍵字外，下列變數值也視為False，如下所示：
  - 0、0.0：整數值0或浮點數值0.0。
  - []、()、{}：容器型別的空串列、空元組和空字典。
  - None：關鍵字None。





## 1-1-3 基本資料型別 布林 (Booleans)

- 在實作上，當運算式使用關係運算子（`==`、`!=`、`<`、`>`、`<=`、`>=`）或邏輯運算子（`not`、`and`、`or`）時，其運算結果是布林值。例如：邏輯運算子（Python程式：`ch1-1-3b.py`），如下所示：

```
a = True
```

```
b = False
```

```
print(type(a)) # 顯示 "<class 'bool'>"
```

```
print(a and b) # 邏輯AND: 顯示 "False"
```

```
print(a or b) # 邏輯OR: 顯示 "True"
```

```
print(not a) # 邏輯NOT: 顯示 "False"
```







## 1-1-3 基本資料型別 布林 (Booleans)

- 然後是2個變數比較的關係運算子（Python程式：ch1-1-3c.py），如下所示：

```
a = 3
```

```
b = 4
```

```
print(a == b) # 相等: 顯示 "False"
```

```
print(a != b) # 不等: 顯示 "True"
```

```
print(a > b) # 大於: 顯示 "False"
```

```
print(a >= b) # 大於等於: 顯示 "False"
```

```
print(a < b) # 小於: 顯示 "True"
```

```
print(a <= b) # 小於等於: 顯示 "True"
```





## 1-1-3 基本資料型別 字串（Strings）

- Python「字串」（Strings）不允許更改字串內容，所有字串變更都是建立全新字串。字串是使用「'」單引號或「"」雙引號括起的一序列Unicode字元，如下所示：

```
s1 = "學習Python語言程式設計"
```

```
s2 = 'Hello World!'
```

- 上述s1和s2變數是字串資料型別，Python語言沒有字元型別，當引號括起的字串只有1個時，就可視為是字元，如下所示：

```
ch1 = "A"
```

```
ch2 = 'b'
```





## 1-1-3 基本資料型別 字串（Strings）

- 當在Python程式建立字串後，我們可以顯示字串、計算字串長度、連接2個字串和格式化顯示字串內容（Python程式：ch1-1-3d.py），如下所示：

```
str1 = 'hello'    # 使用單引號建立字串
```

```
str2 = "python"   # 使用雙引號建立字串
```

```
print(str1)       # 顯示 "hello"
```

```
print(len(str1))  # 字串長度: 顯示 "5"
```

```
str3 = str1 + ' ' + str2 # 字串連接
```

```
print(str3)       # 顯示 "hello python"
```

```
str4 = '%s %s %d' % (str1, str2, 12) # 格式化字串
```

```
print(str4)       # 顯示 "hello python 12"
```



## 1-1-3 基本資料型別 字串（Strings）

- Python可以使用類似C語言printf()函式的格式字串來建立字串內容，格式字元「%s」是字串；「%d」是整數；「%f」是浮點數（Python程式：ch1-1-3e.py），如下所示：

```
s = "world"
```

```
print("hello %s" % s)    # 輸出字串: "hello world"
```

```
num = 10
```

```
print("分數: %d" % num)  # 輸出整數: 分數: 10
```

```
print('hello {}, {}'.format(s, num)) # 輸出: hello world, 10
```

```
1.輸出: hello world, 10.000000
```

```
2.輸出: hello world,          10.000000
```

```
3.輸出" hello world,  
10.000000
```





## 1-1-3 基本資料型別 字串（Strings）

- Python字串物件提供一些好用方法來處理字串（Python程式：ch1-1-3f.py），如下所示：

```
s = "hello"
```

```
print(s.capitalize()) # 第1個字元大寫: 顯示 "Hello"
```

```
print(s.upper())      # 轉成大寫: 顯示 "HELLO"
```

```
print(s.rjust(7))     # 左邊填充空白字元: 顯示 " hello"
```

```
print(s.center(7))    # 置中顯示: 顯示 " hello "
```

```
print(s.replace('l', 'L')) # 取代字串: 顯示 "heLLo"
```

```
print(' python '.strip()) # 刪除空白字元: 顯示 "python"
```

1. Show the result “hELLo”. 如何顯示結果為 “hELLo”
2. How to transfer “hELLo” into “hello”? 將“hELLo”轉成小寫





## 1-2 流程控制

- 1-2-1 條件控制
- 1-2-2 迴圈控制





# 1-2-1 條件控制

## if單選條件敘述

- if條件敘述是一種是否執行的單選題，只是決定是否執行程式區塊內的程式碼，如果條件運算式的結果為**True**，就執行程式區塊的程式碼。例如：判斷氣溫決定是否加件外套的if條件敘述（Python程式：ch1-2-1.py），如下所示：

```
t = int(input("請輸入氣溫 => "))
```

```
if t < 20:
```

```
    print("加件外套!")
```

```
print("今天氣溫 = " + str(t))
```

1.If temperature=15.2, what is the result?

(溫度15.2會顯示什麼?如何修正)

2.Change print("今天氣溫 = " + str(t)) “

(改變一下程式，刪除“+”)



## 1-2-1 條件控制

### if單選條件敘述

- 更進一步，可以活用邏輯運算式，當氣溫在20~22度之間時，顯示「加一件薄外套!」訊息文字，如下所示：

```
if t >= 20 and t <= 22:  
    print("加一件薄外套!")
```

How to show the results?

請顯示結果





# 1-2-1 條件控制

## if/else 二選一條件敘述

- 單純if條件只能選擇執行或不執行程式區塊的單選題，更進一步，如果是排它情況的兩個執行區塊，只能二選一，我們可以加上**else**關鍵字，依條件決定執行哪一個程式區塊。例如：學生成績以60分區分是否及格的if/else條件敘述（Python程式：ch1-2-1a.py），如下所示：

```
s = int(input("請輸入成績 => "))
```

```
if s >= 60:
```

```
    print("成績及格!")
```

```
else:
```

```
    print("成績不及格!")
```





# 1-2-1 條件控制

## if/elif/else多選一條件敘述

- Python多選一條件敘述是if/else條件的擴充，在之中新增elif關鍵字來新增一個條件判斷，就可以建立多選一條件敘述，在輸入時，別忘了輸入在條件運算式和else之後的「:」冒號。







## 1-2-1 條件控制

### if/elif/else多選一條件敘述

- 例如：輸入年齡值來判斷不同範圍的年齡，小於**13**歲是兒童；小於**20**歲是青少年；大於等於**20**歲是成年人，因為條件不只一個，所以需要使用多選一條件敘述（**Python**程式：**ch1-2-1b.py**），如下所示：

```
a = int(input("請輸入年齡 => "))
```

```
if a < 13:
```

```
    print("兒童")
```

```
elif a < 20:
```

```
    print("青少年")
```

```
else:
```

```
    print("成年人")
```



## 1-2-1 條件控制 單行條件敘述

- Python語言並沒有「條件運算式」（Conditional Expressions），我們可以使用單行if/else條件敘述來代替，其語法如下所示：

**變數 = 變數1 if 條件運算式 else 變數2**

- 上述指定敘述的「=」號右邊是單行if/else條件敘述，如果條件成立，就將變數指定成變數1的值；否則指定成變數2的值。例如：12/24制的時間轉換運算式（Python程式：ch1-2-1c.py），如下所示：

**h = h-12 if h >= 12 else h**

**1.print h=15**

**(請印出h=15的值)**



## 1-2-2 迴圈控制

### for計數迴圈

- 在for迴圈的程式敘述中擁有計數器變數，計數器可以每次增加或減少一個值，直到迴圈結束條件成立為止。基本上，如果已經知道需要重複執行幾次，就可以使用for計數迴圈來重複執行程式區塊。
- 例如：在輸入最大值後，計算從1加至最大值的總和（Python程式：ch1-2-2.py），如下所示：

```
m = int(input("請輸入最大值 =>"))
```

```
s = 0
```

```
for i in range(1, m + 1):
```

```
    s = s + i
```

```
print("總和 = " + str(s))
```



## 1-2-2 迴圈控制

### for迴圈與range()函式

- Python的for計數迴圈需要使用range()函式來產生指定範圍的計數值，這是Python內建函式，可以有1、2和3個參數，如下所示：
  - 擁有1個參數的range()函式：此參數是終止值（不包含終止值），預設的起始值是0，如下表所示：

range()函式	整數值範圍
range(5)	0~4
range(10)	0~9
range(11)	0~10





## 1-2-2 迴圈控制

### for迴圈與range()函式

- 擁有2個參數的range()函式：第1參數是起始值，第2個參數是終止值（不包含終止值），如下表所示：

range()函式	整數值範圍
range(1, 5)	1~4
range(1, 10)	1~9
range(1, 11)	1~10







- 1. How to show six number?
- 顯示6個數值=> 0 1 2 3 4 5
- 2. Input a number and then show the front number.
- 輸入任意1個數，顯示之前的所有數字(例如輸入8)

第1個數值是 1  
第2個數值是 2  
第3個數值是 3  
第4個數值是 4  
第5個數值是 5  
第6個數值是 6  
第7個數值是 7



## 1-2-2 迴圈控制

### for迴圈與range()函式

### for loop and function range( )

- 擁有3個參數的range()函式：第1參數是起始值，第2個參數是終止值（不包含終止值），第3個參數是間隔值，如下表所示：

(start, end(not include itself), interval)

range()函式	整數值範圍
range(1, 11, 2)	1、3、5、7、9
range(1, 11, 3)	1、4、7、10
range(1, 11, 4)	1、5、9
range(0, -10, -1)	0、-1、-2、-3、-4...-7、-8、-9
range(0, -10, -2)	0、-2、-4、-6、-8

請輸入一個數 => 8

第 0 個數值是 1

第 1 個數值是 3

第 2 個數值是 5

第 3 個數值是 7

輸入任意1個數，顯示間隔的數字(例如輸入8)



## 1-2-2 迴圈控制

### while條件迴圈 **while loop**

- **while**迴圈敘述需要在程式區塊自行處理計數器變數的增減，迴圈是在程式區塊開頭檢查條件，條件成立才允許進入迴圈執行。例如：使用**while**迴圈計算階層函式值（Python程式：ch1-2-2a.py），如下所示：

```
m = int(input("請輸入階層數 =>"))
```

```
r = 1
```

```
n = 1
```

```
while n <= m:
```

```
    r = r * n
```

```
    n = n + 1
```

```
print("階層值! = " + str(r))
```



## 1-3 函式、模組與套件

- 1-3-1 函式
- 1-3-2 使用Python模組與套件





## 1-3-1 函式 Functions

- **Python**「函式」（**Functions**）是一個獨立程式單元，可以將大工作分割成一個個小型工作，我們可以重複使用之前已經建立的函式，或是直接呼叫**Python**內建函式。
- 函式名稱如同變數是一種識別字，其命名方式和變數相同，程式設計者需要自行命名，在函式的程式區塊中，可以使用**return**關鍵字回傳函式值，和結束函式執行，函式參數（**Parameters**）列是使用介面，在呼叫時，我們需要傳入對應的引數（**Arguments**）。







## 1-3-1 函式 定義函式

- 在Python程式建立沒有參數列和傳回值的print\_msg()函式（Python程式：ch1-3-1.py），如下所示：[函式寫在前面](#)

```
def print_msg():  
    print("歡迎學習Python程式設計!")
```

1.請顯示: 歡迎學習Python程式設計 [參考p.37](#)





## 1-3-1 函式 定義函式

- Python函式如果有傳回值，我們需要使用`return`關鍵字來回傳值。例如：判斷參數值是否在指定範圍的`is_valid_num()`函式，如下所示：

```
def is_valid_num(no):  
    if no >= 0 and no <= 200.0:  
        return True  
    else:  
        return False
```

```
請輸入 =>30  
合法!
```

---

```
In [14]: runfil  
式')
```

```
請輸入 =>250  
不合法
```

1.輸入任意一個數，並顯示合法或不合法 參考p.37

Input a number and show that it is legal or not




## 1-3-1 函式

### 定義函式

- 再來是一個執行運算的`convert_to_f()`函式，如下所示：

```
def convert_to_f(c):  
    f = (9.0 * c) / 5.0 + 32.0  
    return f
```



1.輸入任意一個數，並顯示華式溫度  
Input a number and show f

參考p.37



# 1-3-1 函式

## 函式呼叫

- **Python**程式碼呼叫函式如果沒有傳回值和參數列，呼叫函式是使用函式名稱加上空括號，如下所示：

```
print_msg()
```

- 函式有傳回值，在呼叫時是使用指定敘述取得回傳值，如下所示：

```
f = convert_to_f(c)
```

- 如果函式回傳值為**True**或**False**，可以在**if**條件敘述呼叫函式作為判斷條件，如下所示：

```
if is_valid_num(c):
```

```
    print("合法!")
```

```
else:
```

```
    print("不合法")
```



## 1-3-2 使用Python模組與套件 匯入模組或套件

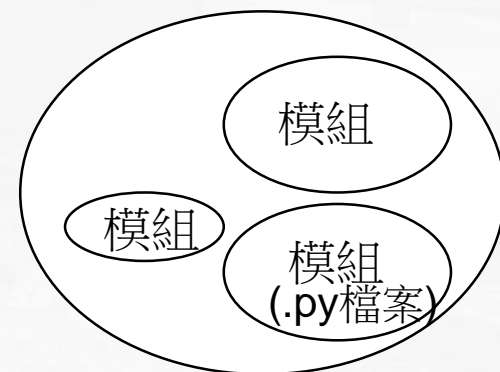
- Python程式是使用**import**關鍵字匯入模組或套件，例如：匯入名為**random**的模組，然後呼叫此模組的函式或物件的方法（即物件的函式）來產生亂數值（Python程式：**ch1-3-2.py**），如下所示：

**import random**

- 上述程式碼匯入名為**random**的模組後，可以呼叫物件的**randint()**方法，產生指定範圍之間的整數亂數值，如下所示：

**target = random.randint(1, 100)**

套件







## 1-3-2 使用Python模組與套件 模組或套件的別名

- 在Python程式檔匯入模組或套件，除了使用模組或套件名稱來呼叫函式，我們也可以使用**as**關鍵字替模組取一個別名，然後使用別名來呼叫方法（Python程式：ch1-3-2a.py），如下所示：

```
import random as R
```

```
target = R.randint(1, 100)
```

1.請顯示一個亂數值

**Homework:**

**How to show a random number?**



## 1-3-2 使用Python模組與套件 匯入模組或套件的部分名稱

- 當Python程式使用import關鍵字匯入模組後，匯入的模組預設是全部內容，在實務上，我們可能只需模組的1或2個函式或物件，此時請使用from/import程式敘述匯入模組的部分名稱，例如：在Python程式匯入BeautifulSoup模組（Python程式：ch1-3-2b.py），如下所示：

```
from bs4 import BeautifulSoup
```

- 上述程式碼匯入BeautifulSoup模組後，就可以建立BeautifulSoup物件，如下所示：

```
html_str = "<p>Hello World!</p>"  
soup = BeautifulSoup(html_str, "lxml")  
print(soup)
```

html5lib  
html.parser

lxml 套件是用來作為 BeautifulSoup 的解析器自動補全標籤



## 1-4 容器型別

- 1-4-1 串列
- 1-4-2 字典
- 1-4-3 元組





## 1-4-1 串列 **Lists**

- **Python**支援的容器型別有：串列、字典和元組等，容器型別如同一個放東西的盒子，我們可以將項目或元素的東西丟到盒子中來儲存，而不用考量記憶體空間的問題。
- **Python**「串列」（**Lists**）類似其他程式語言的「陣列」（**Arrays**），中文譯名有清單、串列和列表等。不同於字串不能更改，串列允許更改（**Mutable**）內容，可以新增、刪除、插入和更改串列項目（**Items**）。





## 1-4-1 串列

### 串列的基本使用

- Python串列是使用「[]」方括號括起的多個項目，每一個項目使用「,」逗號分隔（Python程式：ch1-4-1.py），如下所示：

```
ls = [6, 4, 5]    # 建立串列
print(ls, ls[2])  # 顯示 "[6, 4, 5] 5"
print(ls[-1])     # 負索引從最後開始: 顯示 "5"
ls[2] = "py"      # 指定字串型別的項目
print(ls)         # 顯示 "[6, 4, 'py']"
ls.append("bar")   # 新增項目
print(ls)         # 顯示 "[6, 4, 'py', 'bar']"
ele = ls.pop()     # 取出最後項目
print(ele, ls)    # 顯示 "bar [6, 4, 'py']"
```





# 增加list 中的元素

- 如果想要指定所增加物件的位置呢？可以使用 **list** 的另一個方法**insert** 來指定位置。括號後面第一個值代表的是想要插入的位置，後面則是想要加入的元素。

`my_list.insert(position, object)`

1.加入一個元素到第三個位置

- 如果一次想加入很多個值、或是想將某個 **list** 中的元素加到另一個 **list** 的時候，可以用 **extend** 這個方法。(不要用insert)

`list_1 = [object1, object2, object3]`

`list_2 = [object4, object5]`

`list_1.extend(list_2)`

2.加入多個元素到另一個串列



# 移除 list 中的元素

- 想要移除一個 list 中的元素，可以使用 **remove method**，在 **remove** 後面的括號中輸入想要移除的元素，即可移除

```
my_list.remove(object)
```

```
hello = ['hi','you','I','a Man','Diana', 88]
```

```
hello.remove('you')
```

```
print(hello)
```

- 如果只想移除 list 最後一個元素，可以用 **pop method**。在括號後面不放任何東西，預設 list 中的最後一個 **element** 會被移除。

```
list.pop()
```

1. 分別利用兩種方法移除 I、a Man 和 88

(或混合)



## 1-4-1 串列 切割串列

- **Python**串列可以在「[]」方括號中使用「:」符號的語法，即指定開始和結束來分割子串列（**Python**程式：**ch1-4-1a.py**），如下所示：


```
nums = list(range(5)) # 建立一序列的整數串列
print(nums)           # 顯示 "[0, 1, 2, 3, 4]"
print(nums[2:4])      # 切割索引2~4(不含4): 顯示 "[2, 3]"
print(nums[2:])        # 切割索引從2至最後: 顯示 "[2, 3, 4]"
print(nums[:2])        # 切割從開始至索引2(不含2): 顯示 "[0, 1]"
print(nums[:])         # 切割整個串列: 顯示 "[0, 1, 2, 3, 4]"
print(nums[:-1])       # 使用負索引切割: 顯示 "[0, 1, 2, 3]"
nums[2:4] = [7, 8]     # 使用切割來指定子串列
print(nums)           # 顯示 "[0, 1, 7, 8, 4]"
```



## 1-4-1 串列 走訪串列

- Python程式是使用for迴圈走訪顯示串列的每一個項目（Python程式：ch1-4-1b.py），如下所示：  

```
animals = ['cat', 'dog', 'bat']  
for animal in animals:  
    print(animal)
```
- 上述for迴圈一一取出串列每一個項目和顯示出來，其執行結果如下所示：



```
cat  
dog  
bat
```





## 1-4-1 串列 走訪串列

- 如果需要顯示串列各項目的索引值，我們需要使用 `enumerate()` 函式（Python 程式：ch1-4-1c.py），如下所示：

```
animals = ['cat', 'dog', 'bat']  
for index, animal in enumerate(animals):  
    print(index, animal)
```

- 上述 `enumerate()` 函式有 2 個回傳值，第 1 個 `index` 就是索引值，其執行結果如下所示：

```
0 cat  
1 dog  
2 bat
```

### 1. 請印出

```
第 1 隻動物是 cat  
第 2 隻動物是 dog  
第 3 隻動物是 bat
```





## 1-4-1 串列 串列推導

- 串列推導（**List Comprehension**）是一種簡潔語法來建立串列，我們可以在「**[]**」方括號中使用**for**迴圈產生串列項目，如果需要，還可以加上**if**條件子句來篩選出所需的項目（**Python**程式：**ch1-4-1d.py**），如下所示：

```
list1 = [x for x in range(10)]
```

- 上述程式碼的第1個變數**x**是串列項目，這是使用之後**for**迴圈來產生項目，以此例是0~9，可以建立串列：**[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]**。在方括號第1個**x**是變數，也可以是運算式，例如：使用**x+1**產生項目，如下所示：

```
list2 = [x+1 for x in range(10)]
```

- 上述程式碼可以建立串列：**[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]**。

1. 請印出



## 1-4-1 串列 串列推導

- 在for迴圈後還可以加上if條件子句，例如：只顯示偶數項目，如下所示：

```
list3 = [x for x in range(10) if x % 2 == 0]
```

- 上述程式碼在for迴圈後是if條件子句，可以判斷 $x \% 2$ 的餘數是否是0，也就是只顯示值是0的項目，即偶數項目，可以建立串列：`[0, 2, 4, 6, 8]`。同樣可以使用運算式來產生項目，如下所示：

```
list4 = [x*2 for x in range(10) if x % 2 == 0]
```

- 上述程式碼建立串列：`[0, 4, 8, 12, 16]`。

1. 請印出上面
2. 請印出整除2後，值為3

```
list3= [6, 7]  
第 1 數是 6  
第 2 數是 7
```



## 1-4-2 字典（Introduce Dictionaries）

- Python「字典」（**Dictionaries**）是一種儲存鍵值資料的容器型別，可以使用鍵（**Key**）來取出和更改值（**Value**），或使用鍵來新增和刪除項目，對比其他程式語言，就是結合陣列（**Associative Array**）。





## 1-4-2 字典

### 字典的基本使用(How to use dictionaries)

- Python字典使用大括號「{}」(**bracket**)定義成對的鍵和值 (Key-value Pairs)，每一對使用「,」(**comma**)逗號分隔，鍵和值是使用「:」(**colon**)冒號分隔 ( Python程式：ch1-4-2.py )，如下所示：

```
d = {"cat": "white", "dog": "black"} # 建立字典 Double/single quotation marks
print(d["cat"]) # 使用Key取得項目: 顯示 "white"
print("cat" in d) # 是否有Key: 顯示 "True"
d["pig"] = "pink" # 新增項目
print(d["pig"]) # 顯示 "pink"
print(d.get("monkey", "N/A")) # 取出項目+預設值: 顯示 "N/A"
print(d.get("pig", "N/A")) # 取出項目+預設值: 顯示 "pink"
del d["pig"] # 使用Key刪除項目
print(d.get("pig", "N/A")) # "pig"不存在: 顯示 "N/A"
```



請印出前頁並修改一下，如下

How to show this? white , 類型是 <class 'str'>  
True  
pink  
N/A  
pink  
N/A







## 1-4-2 字典

### 走訪字典 (**visit dictionaries**)

- Python程式一樣是使用for迴圈以鍵來走訪字典（Python程式：ch1-4-2a.py），如下所示：


```
d = {"chicken": 2, "dog": 4, "cat": 4, "spider": 8}
```

```
for animal in d:
```

```
    legs = d[animal]
```

```
    print(animal, legs)
```

- 上述程式碼建立字典變數d後，使用for迴圈走訪字典的所有鍵，可以顯示各種動物有幾隻腳，其執行結果如下所示：



```
chicken 2  
dog 4  
cat 4  
spider 8
```

1. 請印出  
print

```
chicken 有 2 隻腳  
dog 有 4 隻腳  
cat 有 4 隻腳  
spider 有 8 隻腳
```



## 1-4-2 字典 走訪字典

- 如果需要同時走訪字典的鍵和值，請使用`items()`方法（Python程式：`ch1-4-2b.py`），如下所示：  

```
d = {"chicken": 2, "dog": 4, "cat": 4, "spider": 8}  
for animal, legs in d.items():  
    print("動物: %s 有 %d 隻腳" % (animal, legs))
```
- 上述for迴圈走訪`d.items()`，可以回傳鍵`animal`和值`legs`，其執行結果如下所示：

```
動物: chicken 有 2 隻腳  
動物: dog 有 4 隻腳  
動物: cat 有 4 隻腳  
動物: spider 有 8 隻腳
```

1. 請印出`print`
2. 最後一行改成`format`





## 1-4-2 字典 字典推導

- 字典推導（**Dictionary Comprehension**）是一種簡潔語法來建立字典，可以在「**{}**」大括號中使用**for**迴圈產生字典項目，和加上**if**條件子句來篩選出所需的項目（**Python**程式：**ch1-4-2c.py**），如下所示：  
**d1 = {x:x\*x for x in range(10)}**
- 上述程式碼的第1個**x:x\*x**是字典項目，位在「**:**」前是鍵；之後是值，這是使用之後**for**迴圈產生項目，以此例是0~9，可以建立字典：**{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}**。


1. Print -> {0: 1, 1: 2, 2: 3, 3: 4, 4: 5, 5: 6}





## 1-4-2 字典 字典推導

- 我們還可以在**for**迴圈後加上**if**條件子句，例如：只顯示奇數的項目，如下所示：  
$$d2 = \{x:x*x \text{ for } x \text{ in range}(10) \text{ if } x \% 2 == 1\}$$
- 上述程式碼在**for**迴圈後是**if**條件子句，可以判斷 $x \% 2$ 的餘數是否是1，也就是只顯示值是1的項目，即奇數項目，可以建立字典： $\{1: 1, 3: 9, 5: 25, 7: 49, 9: 81\}$ 。



1. Print ->  $\{1: 2, 3: 4, 5: 6\}$



## 1-4-3 元組

- 「元組」(**Tuple**)是一種類似串列的容器型別，事實上，元組就是一個唯讀串列，一旦指定元組的項目，就不再允許更改元組的項目內容。







## 1-4-3 元組

- Python元組是使用「()」括號來建立，每一個項目使用「,」逗號分隔（Python程式：ch1-4-3.py），如下所示：

```
t = (5, 6, 7, 8) # 建立元組
print(type(t))  # 顯示 "<class 'tuple'>"
print(t)        # 顯示 "(5, 6, 7, 8)"
print(t[0])      # 顯示 "5"
print(t[1])      # 顯示 "6"
print(t[-1])     # 顯示 "8"
print(t[-2])     # 顯示 "7"
for ele in t:    # 走訪項目
    print(ele, end=" ") # 顯示 "5, 6, 7, 8"
```



# 1-5 類別與物件 class and object

- 1-5-1 定義類別和建立物件





## 1-5-1 定義類別和建立物件

- Python是一種物件導向程式語言，事實上，Python所有內建資料型別都是物件，包含：模組和函式等也都是物件。
- 物件導向程式是使用物件來建立程式，每一個物件儲存資料（Data）和提供行為（Behaviors）。
- 類別（Class）是物件的模子，也是藍圖，我們需要先定義類別，才能依據類別的模子來建立物件。
- 生產一部汽車時，都會有設計圖(design diagram)，藉此可以知道此類汽車會有哪些特性及功能，類別(Class)就類似設計圖，會定義未來產生物件(Object)時所擁有的屬性(Attribute)及方法(Method)





# 1-5-1 定義類別和建立物件

## 定義類別

- Python是使用class關鍵字來定義Student類別（Python程式：ch1-5-1.py），如下所示：

```
class Student:
```

```
#Constructor
```

```
def __init__(self, name, grade):
```

```
    self.name = name           #Attribute
```

```
    self.grade = grade
```

```
#method
```

```
def displayStudent(self):
```

```
    print("姓名 = " + self.name)
```

```
    print("成績 = " + str(self.grade))
```

```
def whoami(self):
```

```
    return self.name
```

name	grade
steven	88
alice	66
tony	100



## 1-5-1 定義類別和建立物件 類別建構子

- 類別建構子是每一次使用類別建立新物件時，就會自動呼叫的方法，**Python**類別的建構子名為「**`__init__`**」，不能更名，在**init**前後是2個「**`_`**」底線，如下所示：

```
def __init__(self, name, grade):
```

```
    self.name = name
```

```
    self.grade = grade
```

1.**self**代表了實體物件的參考，也就是目前的物件(Object)，**self**就是告訴類別(Class)目前是在設定哪一個物件的屬性(Attribute)

2.此物件的**name**屬性等於傳入的**name**屬性值，此物件的**grade**屬性等於傳入的**grade**屬性值





## 1-5-1 定義類別和建立物件 建構子和方法的**self**變數

- 在Python類別建構子和方法的第1個參數是**self**變數，這是一個特殊變數，並不是Python語言的關鍵字，只是約定俗成的變數名稱，**self**變數的值是參考呼叫建構子或方法的物件，以建構子\_\_init\_\_()方法來說，參數**self**的值是參考新建立的物件，如下所示：

**self.name = name**

**self.grade = grade**





## 1-5-1 定義類別和建立物件 資料欄位

- 類別的資料欄位，或稱為成員變數（Member Variables），在Python類別定義資料欄位並不需要特別語法，只要是使用**self**開頭存取的變數，就是資料欄位，在Student類別的資料欄位有name和grade，如下所示：

```
self.name = name
```

```
self.grade = grade
```





## 1-5-1 定義類別和建立物件 方法(method)

- 類別的方法就是Python函式，只是第1個參數一定是self變數，而且在存取資料欄位時，不要忘了使用self變數來存取（因為有self才是存取資料欄位），如下所示：

```
def displayStudent(self):  
    print("姓名 = " + self.name)  
    print("成績 = " + str(self.grade))
```





## 1-5-1 定義類別和建立物件 使用類別建立物件

- 在定義類別後，可以使用類別建立物件，也稱為實例（**Instances**），同一類別可以如同工廠生產一般的建立多個物件，如下所示：

```
s1 = Student("陳會安", 85)
```

- 然後可以使用「**.**」運算子呼叫物件方法，如下所示：

```
s1.displayStudent()
```

```
print("s1.whoami() = " + s1.whoami())
```

- 同樣的語法，我們可以存取物件的資料欄位，如下所示：

```
print("s1.name = " + s1.name)
```

```
print("s1.grade = " + str(s1.grade))
```

```
姓名 = tony
```

```
成績 = 88
```

```
tony
```

1. print





# 定義Student類別

class Student:

# 建構子

def \_\_init\_\_(self, name, grade):

self.name = name

self.grade = grade

# 方法

def displayStudent(self):

print("姓名 = " + self.name)

print("成績 = " + str(self.grade))

def whoami(self):

return self.name

# 使用類別建立物件

s1 = Student("陳會安", 85)

s1.displayStudent() # 呼叫方法

print("s1.whoami() = " + s1.whoami())

# 存取資料欄位

print("s1.name = " + s1.name)

print("s1.grade = " + str(s1.grade))

```
class Student:
    班級 = 資工三丙
    學號 = s10888888
    電話 = 8825252
    s1.whoami() = s10888888
    s1.班級 = 資工三丙
    s1.學號 = s10888888
    s1.電話 = 8825252
```

```
class Student:
    班級 = 資工三丙
    學號 = s10888888
    電話 = 8825252
    班級 = 資工三丙
    學號 = s10666666
    電話 = 8825225
    s1.whoami() = s10888888
    s1.班級 = 資工三丙
    s1.學號 = s10888888
    s1.電話 = 8825252
```





# Homework : please print

請參考下方印出 `My car is blue and has 4 seats.`

# 汽車類別

class Cars:

# 建構子Constructor

def \_\_init\_\_(self, color, seat):

self.color = color # 顏色屬性

self.seat = seat # 座位屬性

# 方法(Method)

def drive(self):

print("My car is "+ self.color+" and has "+str(self.seat)+" seats.")





## 1-6 檔案處理 (FILE PROCESSING)

- 1-6-1 開啟檔案來寫入和新增資料
- 1-6-2 讀取檔案內容
- 1-6-3 例外處理





## 1-6-1 開啟檔案來寫入和新增資料

### 開啟檔案來寫入資料 (Open file and write data)

- Python程式使用open()函式開啟檔案後，可以呼叫write()方法將參數字串寫入檔案（Python程式：ch1-6-1.py），如下所示：

```
fp = open("c:\\temp\\note.txt", "w")
```

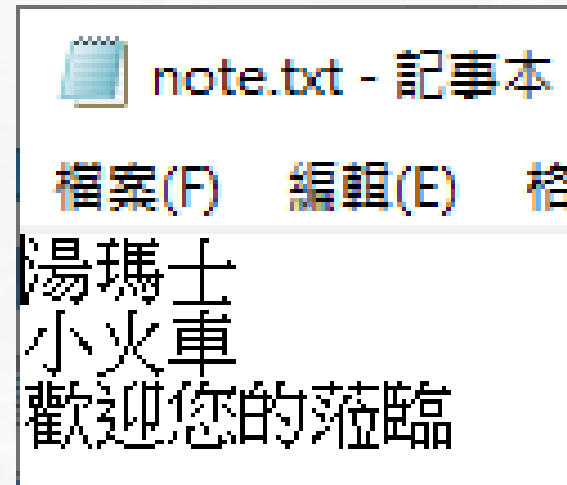
```
fp.write("陳會安\n")
```

```
fp.write("Python")
```

```
fp.write("程式設計\n")
```

```
fp.close()
```

1. print

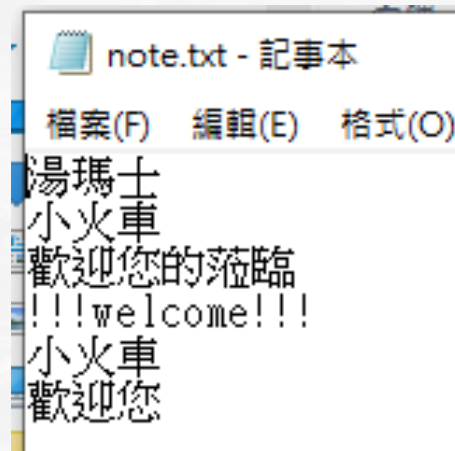




## 1-6-1 開啟檔案來寫入和新增資料 在檔案最後新增資料(add data)

- 如果想在檔案現有資料的最後新增資料，例如：在 **note.txt** 檔案最後再新增一名姓名資料，請使用 "a" 模式字串來開啟新增檔案（Python 程式：ch1-6-1a.py），如下所示：

```
fp = open("temp\\note.txt", "a")  
fp.write("陳允傑\n")  
fp.close()
```



1. print



## 1-6-2 讀取檔案內容

### 使用readline()方法讀取1行內容 (read data)

- 檔案物件的readline()方法可以一次只讀取1行內容（Python程式：ch1-6-2.py），如下所示：

```
fp = open("c:\\temp\\note.txt", "r")
```

```
str1 = fp.readline()
```

```
print(str1)
```

```
str2 = fp.readline()
```

```
print(str2)
```

```
fp.close()
```

1. print

湯瑪士

小火車

歡迎您的蒞臨

!!!welcome!!!







## 1-6-2 讀取檔案內容

### 使用readlines()方法和with/as程式區塊

(read data)

- 檔案物件的readlines()方法可以讀取檔案內容成為串列，每一行是一個項目，在實務上，Python檔案處理需要自行呼叫close()方法來關閉檔案，如果擔心忘了執行事後清理工作，可以使用with/as程式區塊讀取檔案內容（Python程式：ch1-6-2a.py），如下所示：

```
with open("c:\\temp\\note.txt", "r") as fp:
```

```
    lst1 = fp.readlines()
```

```
    print(lst1)
```

```
['陳會安\n', 'Python程式設計\n', '陳允傑\n']
```





# 顯示前頁並修改成顯示如下

## Show the results

```
['湯瑪士\n', '小火車\n', '歡迎您\n']
```

湯瑪士  
小火車  
歡迎您





## 1-6-3 例外處理 **Exception Handling**

- 當程式執行時偵測出的錯誤稱為「例外」（**Exception**），**Python**例外處理（**Exception Handling**）是建立**try/except**程式區塊，以便當**Python**程式執行時產生例外時，能夠撰寫程式碼來進行處理。
- **Python**例外處理程式敘述分為**try**和**except**二個程式區塊，其基本語法，如下所示：

**try:**

**# 產生例外的程式碼**

**except <Exception Type>:**

**# 例外處理**





## 1-6-3 例外處理

- 例如：如果開啟檔案不存在，就會產生**FileNotFoundError**例外，**Python**程式可以使用**try/except**處理檔案不存在的例外，如下所示：

try:

```
fp = open("myfile.txt", "r")
```

```
print(fp.read())
```

```
fp.close()
```

except FileNotFoundError:

```
print("錯誤: myfile.txt does not exist!")
```

1. print