Python
技術者們 實踐!

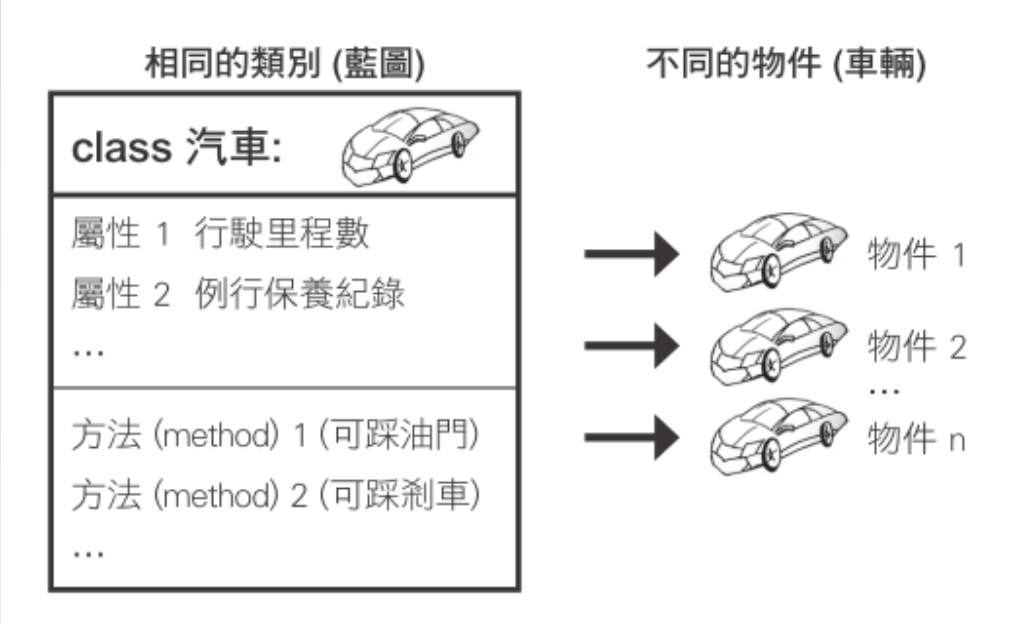
第 5 章
Python 最強功能:
內建函式庫與 第三方套件

5-0 物件 (object) 與類別 (class)

- 在 Python 中，所有的東西都是物件! 不只資料是物件，就連函式也是物件。而寫 Python 程式就是在操作這些物件來得到想要的結果。

- 物件 (Object) 是由類別 (Class) 產生的。
- 類別規劃了物件的資料儲存方式，這些儲存的資料就稱為物件的屬性 (attribute)。
- 類別規劃了物件的操作方式，這些操作方式就稱為物件的方法 (method)。

- 所謂『物件 (object) 是由 類別 (class) 產生的』是甚麼意思呢? 基本上類別就像是物件的設計藍圖。

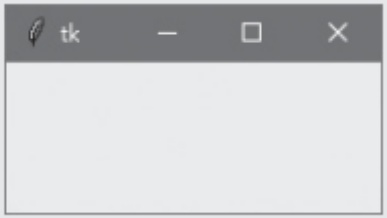


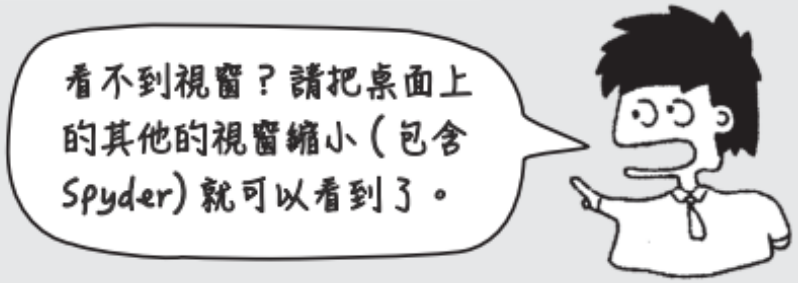
5-1 用現成類別與物件讓程式起飛！

ch5-1

```
from tkinter import Tk      ← 從標準函式庫的 tkinter 套件匯入 Tk 類別
win = Tk()                  ← 用 Tk() 類別來建立視窗物件 win
win.mainloop()              ← 用 win 的 mainloop() 方法來顯示視窗，
                              並進入等待使用者操作的迴圈
```

輸出





模組、套件、函式庫、與第三方套件

模組 (module) 就是儲存程式的檔案, 也就是我們一般所說的程式檔。若將幾個模組集合起來放在資料夾中, 則可以組成**套件** (package)。所以就程式儲存的角度來看, 模組與套件也就是儲存程式的檔案與資料夾。

而前面提到的 Python **標準函式庫**及各種**第三方套件**, 其實就是官方以及第三方 (非官方) 將已經寫好的各種類別、函式等儲存到模組或套件中, 以供我們在需要時將之匯入 (import) 到程式中使用。

如果想知道更多 Python 標準函式庫及第三方套件的功能, 可參考旗標出版的**「Python 函式庫語法範例字典」**一書。書中收錄了所有常用的函式及模組功能, 並依用途分門別類, 提供詳細的說明及豐富的語法範例, 因此無論是用來學習功能或查閱用法都非常方便實用。

用類別建立物件

- 所以, 用類別建立物件的語法就是：
  物件名 = 類別名()

使用物件

- 建立物件之後, 我們可以用： 物件.xxx
- 來使用物件, 你可以把句點 . 想成是「的」, 因此「物件.方法」就是「物件的方法)」, 「物件.屬性」就是「物件的屬性」。例如, 建立 win 物件後, 我們就可以用：

```
win.mainloop()
```

再看另外一個例子：

匯入 requests 套件(可用來讀取網頁資料)

```
import requests
from bs4 import BeautifulSoup
page = requests.get('http://www.flag.com.tw')
soup = BeautifulSoup(page.text, "html.parser")
print(soup.title)
```

從 bs4 套件匯入 BeautifulSoup 類別(可用來解析網頁資料)

用 request 的 get() 讀取網頁資料

建立 BeautifulSoup 類別的物件，叫做 soup，建立時有加入參數

看看 soup 的 title 屬性內容

輸出

```
<title>旗標科技</title>
```

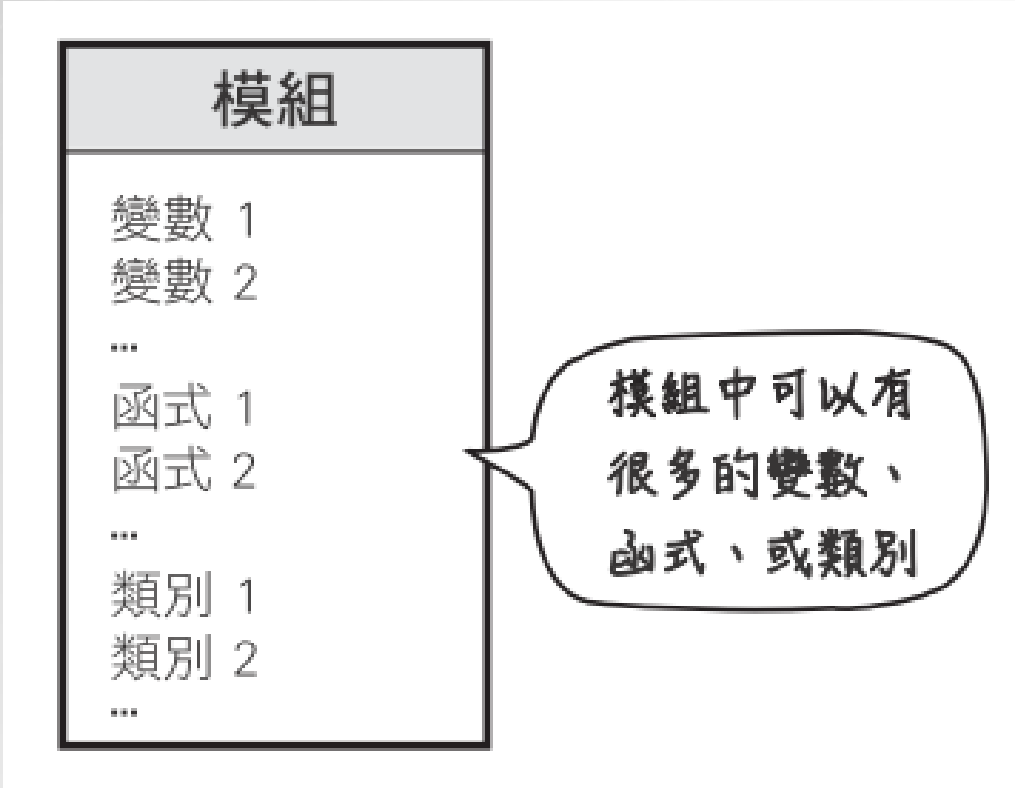
soup 的 title 屬性內容

功能	程式寫法	備註
建立物件	物件名 = 類別名()	類別名都以大寫開頭, () 內可能會加參數
呼叫物件的 method	物件名 .method()	物件名的方法, ( ) 內可能會加參數
取用物件的屬性	物件名.屬性	物件名的屬性
更改物件的屬性	物件名.屬性 = 屬性值	注意! 若無此屬性則會建立一個新的

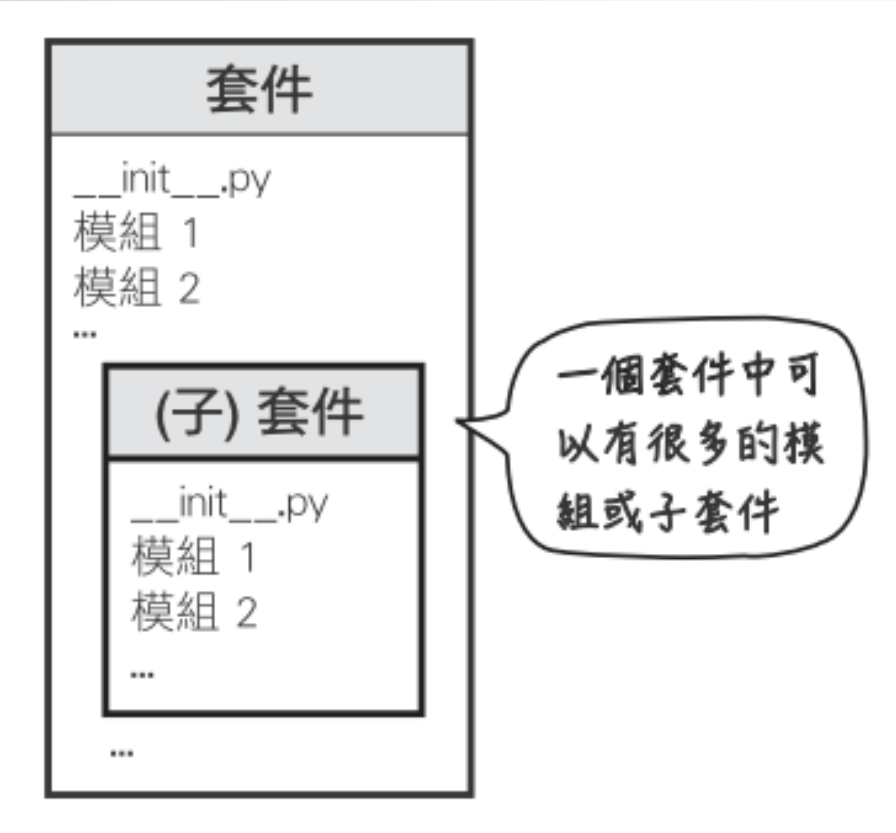
5-2 用 import 來匯入外部資源

- 底下先來看看 Python 函式庫或第三方套件的程式儲存架構, 可分為 2 種：
  - 模組
  - 套件

模組 (module)：



套件 (package)

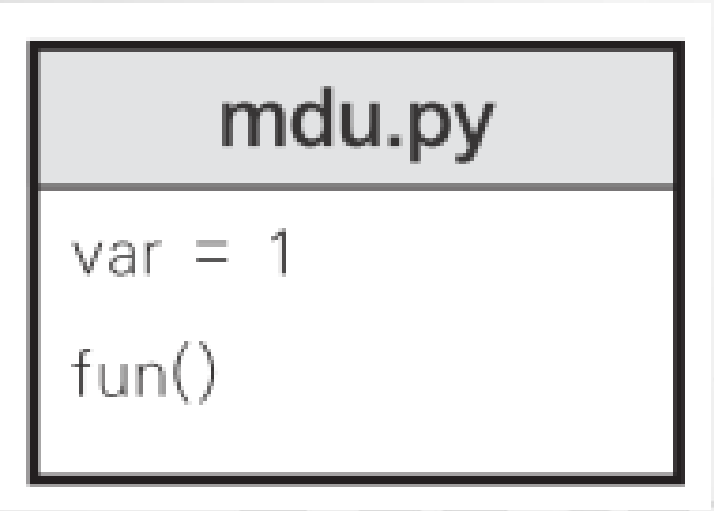


- 在使用 import xxx 時, 其實就是將 xxx 匯入到程式中使用, 因此程式中會多出一個 xxx 物件可以使用。xxx 可以是變數、函式、類別、模組、或套件
- 如果 xxx 是變數、函式、或類別, 那麼就可以直接使用, 例如 xxx 為函式時, 就可以呼叫 xxx()。

- 如果 xxx 是模組, 則會將該模組中所有定義的變數、函式、類別等, 都加到 xxx 物件的名稱空間中, 因此就可以用 xxx.yyy 的寫法來存取模組中的 yyy (變函、函式、或類別) 了
- 如果 xxx 是套件, 則和匯入模組類似, 但匯入的是套件中的 __init__.py 模組, 細節後述

模組 (module)

- 底下來看例子，假設目前程式所在的資料夾中有一個 mdu.py 模組，其內容如右：(可參考範例 ch05\mdu.py)

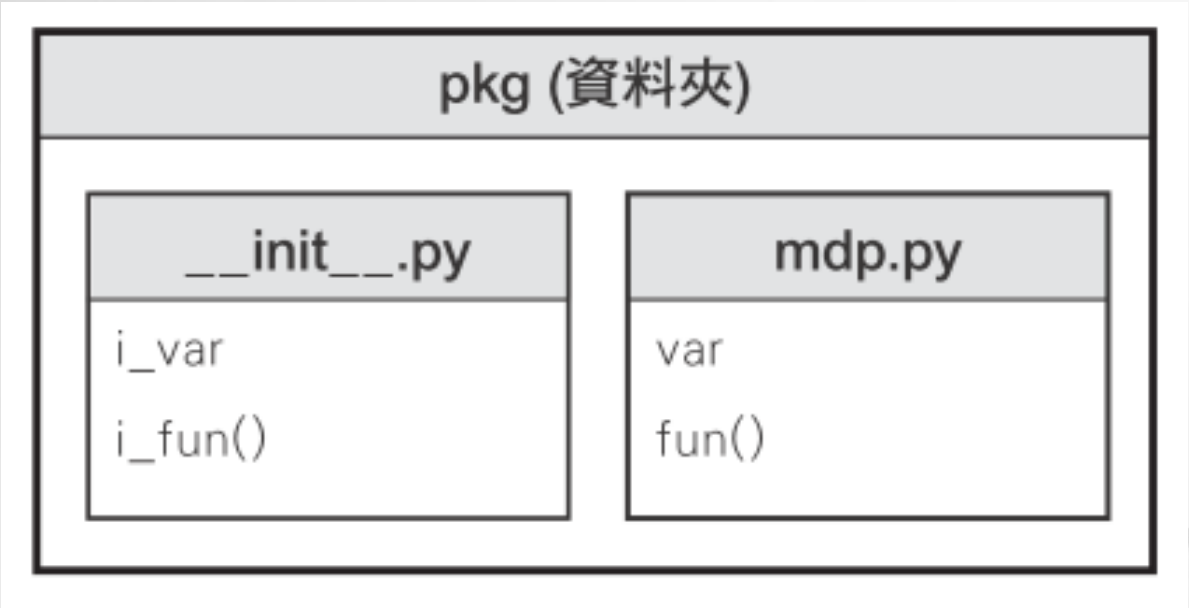


- 那麼底下用法都是正確的：

import 敘述	匯入的資源	匯入的物件名稱	匯入後的使用方式
import mdu	mdu.py 模組	mdu	mdu.var、mdu.fun()
from mdu import fun	mdu.py 的 fun()	fun	fun()
from mdu import *	mdu.py 的所有東西	var、fun	var、fun()

套件 (package)

- 再假設目前的資料夾中還有一個 pkg 套件(資料夾), 其內容如右：

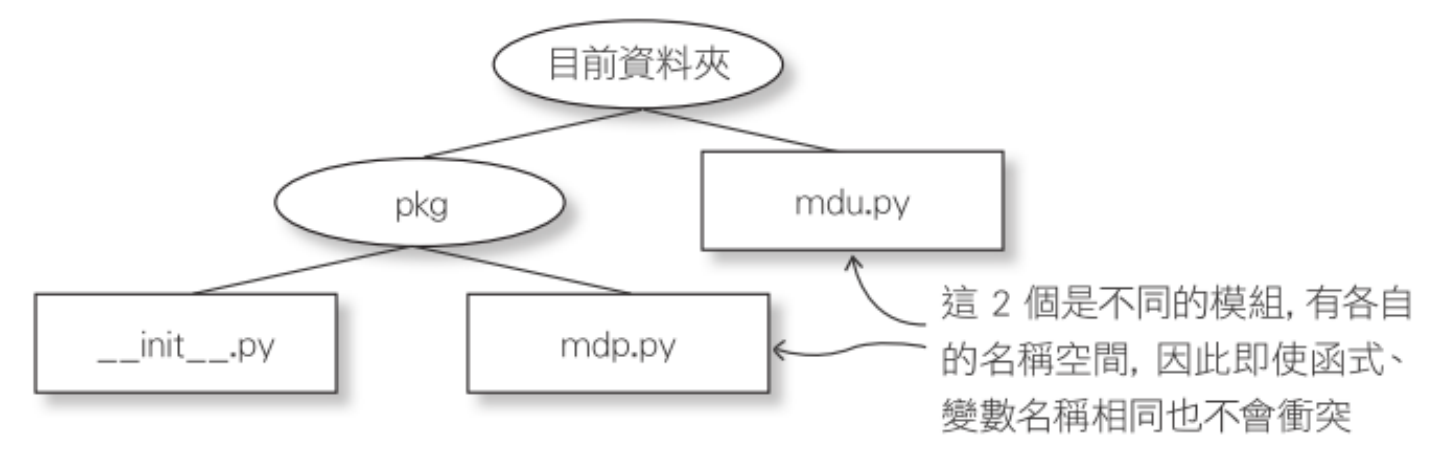


- 那麼底下用法也都是正確的：

import 敘述	匯入的資源	匯入的物件名稱	匯入後的使用方式
import pkg	pkg_init_.py	pkg	pkg.i_var、pkg.i_fun()
from pkg import mdp	pkg\mdp.py	mdp	mdp.var、mdp.fun()
from pkg.mdp import fun	pkg\mdp.py 的 fun()	fun	fun()

- 有二點值得注意:
  - 第一是在使用 import pkg 匯入套件時, 其實只匯入套件中的 __init__ 模組, 而不會匯入套件中的其他模組。若要使用 pkg 的 mdp 模組, 還必須另外用 from pkg import mdp 來匯入。

- 第二是 pkg 套件中 mdp 模組的 var、fun 名稱，和目前資料夾中 mdu 模組內的變數、函式重複了！不過因為它們所在的模組不同，因此並不會發生衝突：



- 但如果我們同時將它們以原始名稱匯入, 就可能發生衝突了, 例如：

```
from mdu import fun
from pkg.mdp import fun ← fun 名稱衝突了！後者會蓋掉前者
```


- 這時我們可以用 as 來自訂匯入物件的名稱, 例如：

```
from mdu import fun
from pkg.mdp import fun as fun2 ← 換個名稱即可避免衝突
```

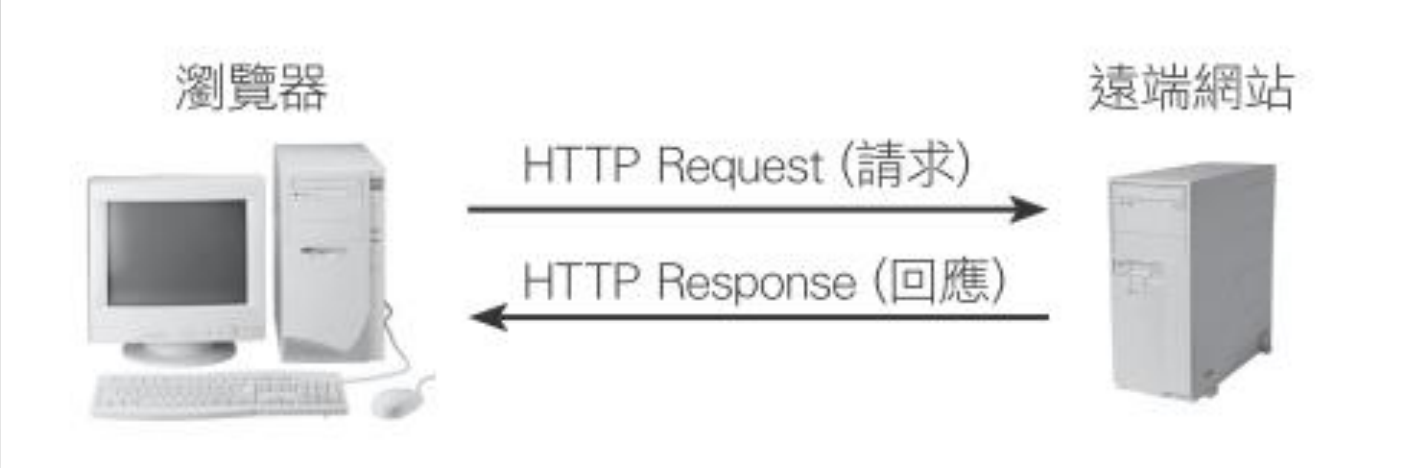
Spyder 好用的「開啟原始檔」功能

在 Spyder 中按住 Ctrl 鍵然後用滑鼠在 from ppp import xxx 的 ppp 或 xxx 上按一下, 可以立即開啟相關檔案供您檢視, 此時也可在工具列的下方看到該檔案的所在路徑。

不過如果點選的是套件名稱, 則實際上會開啟套件中的 __init__.py 檔, 您可依據該檔的所在路徑, 再用檔案總管或 Spyder 右上方的**檔案瀏覽** (File Explorer) 窗格來檢視整個套件的內容。

5-3 用 requests 套件存取網路 Web 資源

- 當我們使用瀏覽器來瀏覽網站時，其實瀏覽器都是以 HTTP 協定來與網站伺服器做溝通，其溝通方式很簡單：



以 GET 讀取網頁資料

Anaconda 已經預先幫我們安裝好 requests 套件了，不需額外安裝就可以 import（本書安裝版本為 2.18.4）

5-0.py

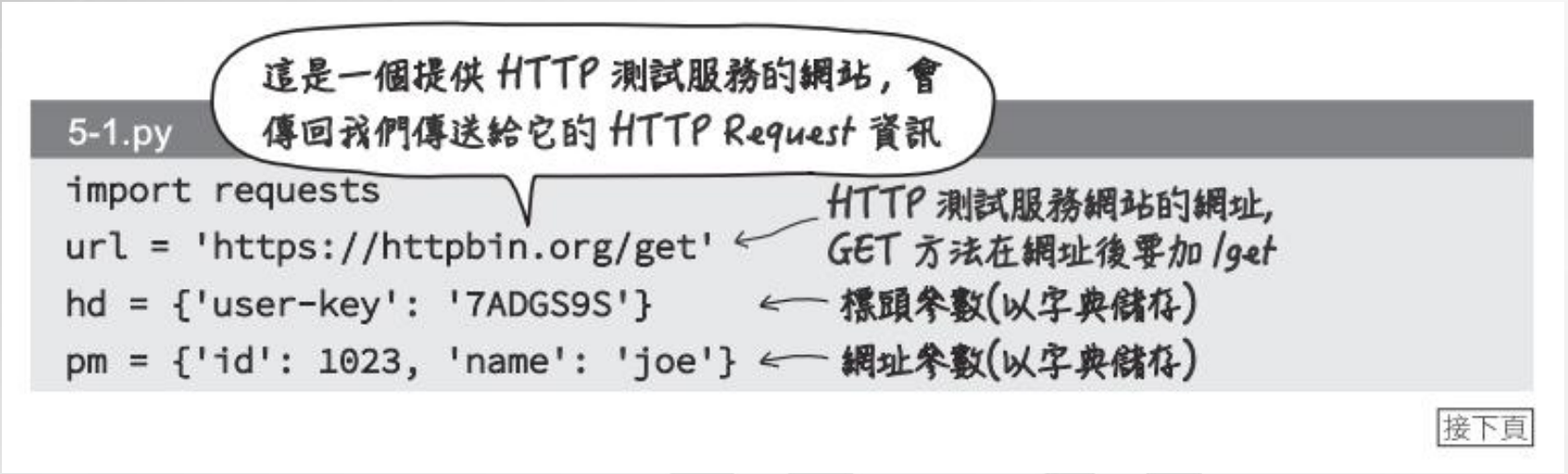
```
import requests  ← 匯入 requests 套件
r = requests.get('http://www.flag.com.tw') ← 向旗標網站發出 GET 請求，並將回應物件儲存到 r
if r.status_code == 200:            # 回應的狀態碼若為 200 表示 OK
    print(r.text)                   # 將回應的文字(網頁原始碼)印出來
else:
    print(r.status_code, r.reason) # 若發生錯誤(狀態碼不是 200)，
                                    則印出狀態碼及錯誤原因
```

輸出

```
<!doctype html>
<html>
    <head>
    <meta charset="utf-8">
    ...
    <title>旗標科技</title>
    <link rel="stylesheet" ...>
    ...
    </head>
    <body class="homepage">
        ...
    </body>
</html>
```

由於內容很多，只列出主要的網頁標籤結構

- 另外我們也可以在 HTTP 請求中以 headers 參數來加入標頭參數, 或是以 params 參數在網址之後加上 ? 開頭的網址參數



```
r = requests.get(url, headers = hd, params = pm)
```
← 加入 headers 及 params 參數

```
print(r.text)
```
← 將回應的文字 (text 屬性) 印出來

輸出

```
{
  "args": {
    "id": "1023",
    "name": "joe"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Connection": "close",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.18.4",
    "User-Key": "7ADGS9S"
  },
  "origin": "220.135.49.167",
  "url": "https://httpbin.org/get?id=1023&name=joe"
}
```

"id": "1023", "name": "joe" ← 這是我們加的網址參數

"Accept": "*/*", "Accept-Encoding": "gzip, deflate", "Connection": "close", "Host": "httpbin.org", "User-Agent": "python-requests/2.18.4" ← 這些是 Requests 預設會加入的標頭參數

"User-Key": "7ADGS9S" ← 這是我們加的自訂標頭參數

Requests 會將網址參數加在 HTTP Request 網址的後面，其格式為 url?name1=val1&name2=val2&...

你也可以不使用 get() 的 params 參數來指定網址參數，而改為直接將網址參數以 ? 加在網址的後面。這 2 種方法都可以，哪種方便就用哪種。

以 POST 送出資料

- POST 也是常用的 HTTP Request 方法，主要用來送出資料

5-2.py

```
import requests
url = 'http://httpbin.org/post' # 使用測試服務網站，POST 方法網址要加 /post
r = requests.post(url, data = 'Hello') ← 送出字串資料
print(r.text)
r = requests.post(url, data = {'id':'123', 'name':'Joe'}) ← 送出字典資料
```

接下頁

```
print(r.text)
```

輸出

```
{
  "args": {},
  "data": "Hello",
  "files": {},
  "form": {},
  ...
}
```

← 字串會以 data 格式送出

```
{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "id": "123",
    "name": "Joe"
  },
  ...
}
```

字典則會以 form (表單) 格式送出

以其他方法送出 HTTP 請求

方法	以存取網站中 a.txt 的內容為例
GET	讀取網站中 a.txt 的內容
POST	以上傳的資料來新增一個名為 a.txt 的檔案
PUT	以上傳的資料來複蓋 a.txt, 若 a.txt 不存在則新增一個
PATCH	以上傳的資料來更改 a.txt 中的部份內容
DEL	刪除 a.txt

5-3.py

```
import requests
r = requests.put('https://httpbin.org/put', data = {'key':'abc'})
print(r.text)
r = requests.patch('https://httpbin.org/patch', data = {'key':'xyz'})
print(r.text)
r = requests.delete('https://httpbin.org/delete')
print(r.text)
```


```
{ ...
  "form": {
    "key": "abc" ←— put 的傳回結果
  }, ...
}
{ ...
  "form": {
    "key": "xyz" ←— patch 的傳回結果
  }, ...
}
{ ...
  "form": {},    ←— delete 的傳回結果
  ...
}
```

5-4 用 BeautifulSoup 套件解析網頁內容

- 使用上一節的 Requests 套件可以輕鬆取得網頁原始碼, 而 BeautifulSoup 套件則可解析網頁原始碼中的 HTML 標籤 (Tag), 幫我們篩選出需要的內容。

建立 BeautifulSoup 物件來解析網頁

```
from bs4 import BeautifulSoup ← 由 bs4 套件中匯入 BeautifulSoup 類別
```

```
物件 = BeautifulSoup('網頁原始碼資料', '解析器名稱')
```

5-4.py

```
01 page = """ ← 將簡化的網頁原始碼儲存在名為 page 的字串中
02 <html>
03   <head><title>旗標科技</title></head>
04   <body>
05     <div class="section" id="main">
06       <img alt="旗標圖示" src="http://flag.tw/logo.png">
07       <p>產品類別</p>
08       <button id="books"><h4 class="bk">圖書</h4></button>
09       <button id="maker"><h4 class="pk">創客</h4></button>
10       <button id="teach"><h4 class="pk">教具</h4></button>
11     </div>
12     <div class="section" id="footer">
13       <p>杭州南路一段15-1號19樓</p>
14       <a href="http://flag.tw/contact">聯絡我們</a>
15     </div>
16   </body>
17 </html>
18 """
19
20 from bs4 import BeautifulSoup ← 由 bs4 套件中匯入 BeautifulSoup 類別
21 bs = BeautifulSoup(page, 'lxml') ← 以 lxml 解析網頁然後建立 BeautifulSoup 物件
```

這裡的 class 和 id 是網頁設計語言 HTML 的 keyword, 而不是 Python 的 class 和 id() 函式

以標籤做為 bs 物件的屬性來查詢資料

ch5-10

(接續前例)

```
print(bs.title)  輸出  <title>旗標科技</title>  ← 傳回第 3 行的 title 標籤
print(bs.a)      輸出  <a href="http://flag.tw/contact">聯絡我們</a>
```

a 標籤　屬性 = 屬性值　夾在標籤中的文字　傳回第 14 行的 a 標籤

(接續前例)

```
print(bs.a)              輸出  <a href="http://flag.tw/contact">聯絡我們</a>
print(bs.a.text)         輸出  聯絡我們  ← 夾在 a 標籤中的文字
print(bs.a.get('href'))  輸出  http://flag.tw/contact  ← a 標籤中 href 屬性的值
print(bs.a['href'])      輸出  http://flag.tw/contact  ← 功能同上一行
```

使用 find() 方法

(接續前例)

```
print(bs.find('h4'))
<h4 class="bk">圖書</h4>
print(bs.find('h4', {'class': 'pk'}))
<h4 class="pk">創客</h4>
print(bs.find('h4').text)
圖書
```

傳回第 8 行的 h4 標籤

限制 class 屬性為 'pk' 的才要

傳回第 9 行的 h4 標籤

取得夾在 h4 標籤中的文字

注意，這裡的 class 是網頁語法的 class，不是物件類別的 class。

使用 find_all() 方法

(接續前例)

```
print(bs.find_all('h4')) ← 以 list 傳回所有 h4 標籤
print(bs.find_all('h4', {'class': 'pk'})) ← 使用 class 屬性篩選
```

輸出

```
[<h4 class="bk">圖書</h4>, <h4 class="pk">創客</h4>, <h4 class="pk">
教具</h4>]  ← 有 3 個 <h4> 標籤
[<h4 class="pk">創客</h4>, <h4 class="pk">教具</h4>]←
                                    只有 2 個 class 為 'pk' 的 <h4> 標籤
```

(接續前例)

```
print(bs.find_all(['title', 'p']))          ← 傳回所有的 title 及 p 標籤
print(bs.find_all(['title', 'p'])[1].text)  ← 輸出傳回串列的第 1 個
                                              (由 0 算起) 標籤中的文字
```

輸出

```
[<title>旗標科技</title>, <p>產品類別</p>, <p>杭州南路一段15-1號19樓</p>]
產品類別
```

使用 select() 方法

- select() 除了可查詢標籤名稱外，還可用 CSS 選擇器來查詢
  - 查詢所有標籤中的 id 屬性，則前面要加 #
  - 查詢所有標籤中的 class 屬性，則前面要加「.」

使用 select() 方法

- select() 除了可查詢標籤名稱外，還可用 CSS 選擇器來查詢，這主要是針對 HTML 標籤中的 id 及 class 屬性做查詢：
  - 如果是要查詢所有標籤中的 id 屬性，則前面要加
  - 如果是要查詢所有標籤中的 class 屬性，則前面要加「.」

(接續前例)

```
print('h4:', bs.select('h4'))          ← 查詢所有 h4 標籤
print('#book:', bs.select('#books'))   ← 查詢所有 id 為 'books' 的標籤
print('.pk:', bs.select('.pk'))        ← 查詢所有 class 為 'pk' 的標籤
print('h4.bk:', bs.select('h4.bk'))    ← 查詢所有 class 為 'bk' 的 h4 標籤
```

輸出

```
h4: [<h4 class="bk">圖書</h4>, <h4 class="pk">創客</h4>, <h4
 class="pk">教具</h4>] ← 有 3 筆
#book: [<button id="books"><h4 class="bk">圖書</h4></button>] ← 有 1 筆
.pk: [<h4 class="pk">創客</h4>, <h4 class="pk">教具</h4>] ← 有 2 筆
h4.bk: [<h4 class="bk">圖書</h4>] ← 只有 1 筆
```

使用 select() 方法

- 針對多層套疊的標籤，還可以用 select ('外層 內層 內層 ...') 來逐層尋找，例如：

(接續前例)

```
print(bs.select('#main button .pk')[1].text)
print(bs.select('#footer a')[0]['href'])
```

輸出

```
教具
http://flag.tw/contact
```

輸出傳回 list 的第 1 個標籤中的文字

輸出傳回 list 的第 0 個標籤的 href 屬性值

(接續前例)

```
print(bs.select('#main button .pk'))
```

輸出

```
[<h4 class="pk">創客</h4>, <h4 class="pk">教具</h4>]
```

查詢在 #main 內的 button 內的 .pk 標籤

符合的有 2 筆

5-5 用 re 模組以「常規表達式」來搜尋字串

- 常規表達式 (Regular Expression, 又稱 regex) 可用來在字串中搜尋「符合特定規則」的子字串

常規表達式的語法

- 「單一字元」的表達

語法單元	說明	範例語法	搜尋結果 (符合的文字以灰底顯示)
一般字元	直接比對	pre	expressed, pre 完全相同
.	代表任意字元, 但不包含換行字元 (\n)	p..s	expressed, p 和 s 間任意 2 字元
[]	在 [] 中可列舉符合的字元	e[dsx]	expressed, ed、es、ex 都可以
	在 [] 中最前面加 ^ 表示不包含	e[^dx]	expressed, 不含 ed、ex
	在 [] 中也可用 - 表示區間範圍	e[s-x]	expressed, ed 不在 e[s-x] 中

- 「重複次數」的表達

語法單元	說明	範例語法	搜尋結果 (符合的文字以灰底顯示)
+	代表前一個字元可以出現 1 次以上 (無上限)	Ap+	AleApleApple
*	代表前一個字元可以出現 0 次以上 (無上限)	Ap*	AleApleApple

語法單元	說明	範例語法	搜尋結果 (符合的文字以灰底顯示)
?	代表前一個字元可以出現 0 或 1 次	Ap?	AleApleApple
{m}	代表前一個字元要出現 m 次	Ap{2}	AleApleApple
{m,n}	代表前一個字元出現 m-n 次都可以符合	Ap{1,3}	AleApleApple

- 「頭尾字元」的表達

語法單元	說明	範例語法	搜尋結果
^	必須以後面的字元為開頭	^App	AppApp (以 A 開頭)
$	必須以前面的字元為結尾	App$	AppApp (以 p 結尾)

- 「轉義字元」的表達

語法單元	說明	範例語法	搜尋結果
\	後面的符號以一般符號處理	\+2\=3\?	1+2=3?
\\	代表 \ 字元	b\\c	ab\cd
\n	換行字元		
\r	歸位 (回到本行開頭) 字元		
\t	tab 定位字元		
\f	換頁字元		
\d	數字字元, 即 [0-9]	a\d+	a1aa22aaa333
\D	非數字字元, 即 [^0-9]	a\D+	a1aa22aaa333
\s	空白字元, 即 [\r\t\n\f]	a\s	[a_a 123]
\S	非空白字元, 即 [^\r\t\n\f]	a\S	[a_a 123]
\w	數字、英文字、底線, 即 [0-9a-zA-Z_]	\w+	[a_a 123]
\W	非數字、英文字、底線, 即 [^0-9a-zA-Z_]	\W+	[a_a 123]

- 最後來看幾個經常會用到的常規表達式：

搜尋目標	常規表達式	符合條件的字串例
整數	[0-9]+ 或 \d+	32767
浮點數	[0-9]+\.[0-9]+	3.14159
英文字	[a-zA-Z]+	HelloWorld
手機號碼	09\d\d-?\d{6}	0912345678 或 0912-345678
E-mail	[\w-]+@[\w-]+(\.[\w-]+)+	ken@flag.com.tw 或 ken-L@n-e-w.flag.tw
網址	https?://[\w-]+(\.[\w-]+)+/?	http://flag.tw 或 https://n-e-w.flag.tw/

用 re 模組以常規表達式來搜尋字串

- match() 與 search()：找出第一個符合的子字串

5-5.py

```
import re    # 使用前要先匯入 re 模組

print(re.match (r'pyt', 'python'))    ← pyt 由開頭即符合, 因此成功
print(re.match (r'yth', 'python'))    ← yth 與開頭不符合, 因此失敗
print(re.search(r'yth', 'python'))    ← search() 不限開頭, 因此成功
```

輸出

一般會在 regex 字串前都加 r 來告訴 Python 不要轉義

```
<_sre.SRE_Match object; span=(0, 3), match='pyt'>
None
<_sre.SRE_Match object; span=(1, 4), match='yth'>
```

搜尋成功會傳回一個物件　找到的位置 1-3　找到的子字串

方法	傳回內容
group()	傳回搜尋到的子字串
start()	傳回子字串在字串中的開始位置 (由 0 算起)
end()	傳回子字串在字串中的結束位置
span()	傳回一個 tuple：(開始位置, 結束位置)

5-6.py

```
import re

m = re.search(r'p[a-z]+e', 'apples')
print(m)  ➡ <_sre.SRE_Match object; span=(1, 5), match='pple'>
print(m.group())  ➡ pple
print(m.start())  ➡ 1
print(m.end())  ➡ 5 ← 注意！pple 的位置是 1-4
print(m.span())  ➡ (1, 5)
```

- sub()：將找到的子字串置換為另一個字串

```
import re
print(re.sub(r'[0-9]+', '#', 'a1b23c456d'))  ➧  a#b#c#d
```


用 compile() 來提升多次搜尋的效率

ch5-19

5-7.py

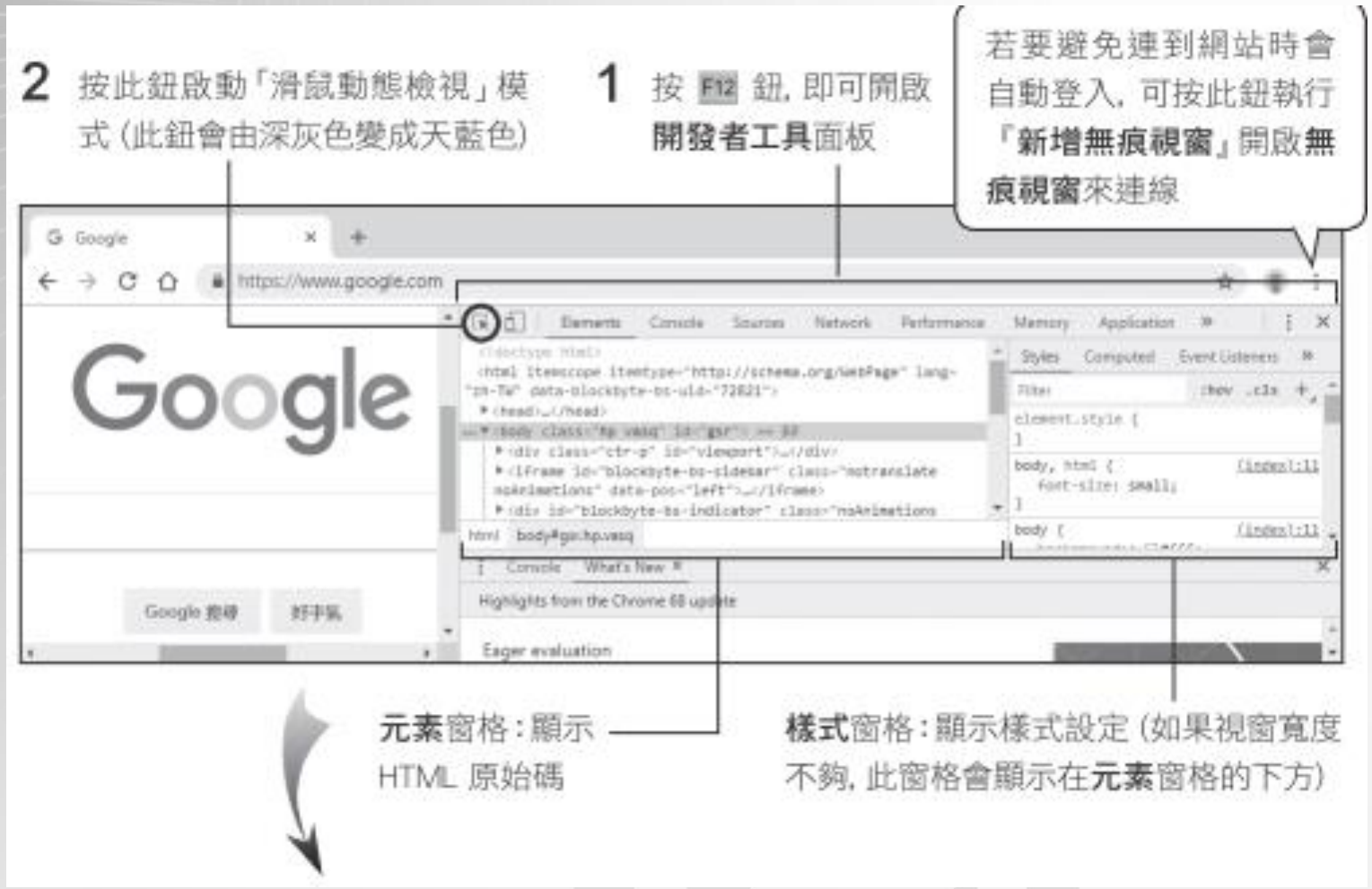
```
import re

ptn = re.compile(r'[0-9]+')
print(ptn.search('a1b23c456d').group())
print(ptn.findall('a1b23c456d'))
print(ptn.sub('#', 'a1b23c456d'))
```


```
1
['1', '23', '456']
a#b#c#d
```


5-6 用 Chrome 來檢視網頁各部份

- 網站的 HTML 原始碼通常都相當多而且複雜，我們可以利用 Chrome 瀏覽器的「開發者工具」來方便地檢視網頁各部份的 HTML 碼。
- 請先開啟 Chrome 瀏覽器，我們以 Google 網站來示範


2 按此鈕啟動「滑鼠動態檢視」模式 (此鈕會由深灰色變成天藍色)
1 按 F12 鈕, 即可開啟**開發者工具**面板
若要避免連到網站時會自動登入, 可按此鈕執行**『新增無痕視窗』**開啟**無痕視窗**來連線
Google
https://www.google.com
Google
Google 搜尋
好手氣
Elements
Console
Sources
Network
Performance
Memory
Application
Styles
Computed
Event Listeners
Console
What's New
Highlights from the Chrome 68 update
Eager evaluation
元素窗格：顯示 HTML 原始碼
樣式窗格：顯示樣式設定 (如果視窗寬度不夠, 此窗格會顯示在**元素**窗格的下方)


3 將滑鼠移到網頁中任意元素上
再按一下此鈕可結束「滑鼠動態檢視」模式
按此鈕選擇『Help』可查閱「開發者工具」的說明文件
4 會立即顯示元素的「標籤名稱#id名稱.類別名稱 (如果有的話)」及元素寬高
5 這裡也會自動標示出對應的 HTML 碼
按此鈕可關閉開發者工具面板
6 在想要的元素上按一下滑鼠即可結束「滑鼠動態檢視」模式，並自動在元素窗格中選取對應的 HTML 碼

5-7 用 Selenium 套件來操控瀏覽器

- 瀏覽器已是現代生活中不可或缺的工具，而借助第三方套件 Selenium，我們就可以用程式來操控瀏覽器，達到自動化操作的目的。

安裝 Selenium 套件與 WebDriver 程式

ch5-20

```
conda install selenium          # 本書安裝版本為 3.14.1
```

建立瀏覽器物件來操控 Chrome

```
from selenium import webdriver ← 匯入 selenium 套件的 webdriver 子套件
browser = webdriver.Chrome()   ← 用 webdriver 的 Chrome 類別建立瀏覽器物件
```

5-8.py

```
from selenium import webdriver # 匯入 selenium 的 webdriver 子套件
from time import sleep         # 匯入內建 time 模組的 sleep() 函式(計時用)

browser = webdriver.Chrome()  ← 建立 Chrome 瀏覽器物件
browser.get('http://www.flag.com.tw')  ← 開啟 Chrome 並連到旗標網站
sleep(5)                      # 暫停 5 秒
browser.close() ← 關閉網頁(目前分頁)
```

為了安全起見, selenium 會以空的 Google 帳號開啟 Chrome, 因此不會帶入我們慣用帳號的資料 (如書籤、歷史記錄、自動登入網站等)

這裡會顯示 Chrome 目前是受到程式的控制

- 瀏覽器物件常用的網頁操作方法及屬性如右表：

方法或屬性	說明
get(url)	連到 url 網址
forward()	到前一頁
back()	到前一頁
refresh()	重新讀取網頁
current_url	(屬性) 目前的網址
title	(屬性) 網頁的標題
page_source	(屬性) 網頁原始碼

- 除了操作網頁外, 也可以用以下的方法來操作瀏覽器視窗：

方法	說明
maximize_window()	將視窗最大化
minimize_window()	將視窗最小化
fullscreen_window()	將視窗設為全螢幕模式
get_window_position()	傳回視窗左上角的位置, 例如 :{'x': 10, 'y': 10}
get_window_size()	傳回視窗的寬度和高度, 例如 :{'width':625, 'height':830}
get_window_rect()	傳回視窗的位置及寬高, 例如 :上 2 行例子的聯集
set_window_position(x,y)	設定視窗左上角的位置
set_window_size(w, h)	設定視窗的寬度和高度
set_window_rect(x,y,w,h)	設定視窗的位置及寬高
close()	關閉網頁 (瀏覽器分頁)
quit()	關閉所有 Selenium 開啟的視窗並結束驅動程式
save_screenshot(path)	將網頁畫面儲存為 PNG 檔, path 為完整路徑並以 .png 結尾

5-9.py

```
from selenium import webdriver  # 匯入 selenium 的 webdriver
from time import sleep          # 匯入內建 time 模組的 sleep() 函式

browser = webdriver.Chrome()    # 建立 Chrome 瀏覽器物件
browser.get('http://www.google.com')  ← 開啟 Chrome 並連到 Google 網站
print('標題:' + browser.title)              # 輸出網頁標題
print('網址:' + browser.current_url)        # 輸出網頁網址
print('內容:' + browser.page_source[0:50]) # 輸出網頁原始碼的前 50 個字
print('視窗:', browser.get_window_rect())   # 輸出視窗的位置及寬高
browser.save_screenshot('d:/scrcap.png') ← 擷取網頁畫面
sleep(3)                                    # 暫停 3 秒
browser.set_window_rect(200, 100, 500, 250) ← 改變視窗位置及大小
sleep(3)
browser.fullscreen_window() ← 將視窗設為全螢幕
sleep(3)
browser.quit()  # 關閉視窗結束驅動
```

輸出

注意！在路徑字串中可使用 / 或 \ 來分隔資料夾，例如 'C:/s.png' 或 'C:\s.png' 都可以，但 \ 可能被視為轉義字元（例如 \n 會被轉為換行字元），所以本書一律會使用 /，以省去避免轉義（例如要寫成 'C:\\n.png' 或 r'\n.png'）的麻煩。

接下頁

```
標題：Google
網址：https://www.google.com/?gws_rd=ssl
內容：<!DOCTYPE html><html xmlns="http://www.w3.org/1999
視窗： {'height': 1030, 'width': 825, 'x': 10, 'y': 10}
```

注意字典中的元素是無順序的

方法或屬性	說明
find_element_by_tag_name(tag)	以標籤 (Tag) 尋找元素
find_element_by_class_name(class)	以類別名稱尋找元素
find_element_by_id(id)	以 id 尋找元素
find_element_by_name(name)	以名稱 (標籤中的 name 屬性) 尋找元素
find_element_by_link_text(text)	以連結文字尋找元素
find_element_by_partial_link_text(text)	以部份的連結文字尋找元素
find_element_by_css_selector(selector)	以 CSS 選擇器 (#id、.class) 尋找元素
find_element_by_xpath(xpath)	以 xpath 尋找元素

5-10.py

```
from selenium import webdriver          # 匯入 selenium 的 webdriver
browser = webdriver.Chrome()            # 建立 Chrome 瀏覽器物件
browser.get('http://www.google.com')    # 開啟 Chrome 並連到旗標網站
e1 = browser.find_element_by_tag_name('head') ← 尋找 head 標籤
print(e1.tag_name) 輸出 head ← 確認已找到 (tag_name 屬性為標籤名稱, 詳見下表)
e2 = e1.find_element_by_tag_name('title') ← 在 head 元素中尋找 title 標籤
print(e2.tag_name) 輸出 tite ← 確認已找到
browser.quit()                          # 關閉視窗結束驅動
```

方法或屬性	說明
click()	模擬滑鼠按一下
send_keys(str)	模擬按鍵輸入, 會輸入 str 字串中的文字
submit()	送出表單
get_attribute(attrname)	讀取元素的屬性值
is_displayed()	元素是否有顯示出來 (沒有被隱藏)
is_enabled()	元素是否可操作 (沒有被 disable)
is_selected()	元素是否被選取 (適用於表單的 checkbox 或 radio button)
screenshot(path)	將元素畫面儲存為 PNG 檔, path 為完整路徑並以 .png 結尾
tag_name	(屬性) 元素的標籤名稱
text	(屬性) 元素的文字內容
size	(屬性) 元素的大小

實例 1：自動登入 Facebook 網站

```
from selenium import webdriver  # 匯入 selenium 的 webdriver
browser = webdriver.Chrome()    # 建立 Chrome 瀏覽器物件

browser.get('http://www.facebook.com') ← 開啟 Chrome 並連到 FB 網站
browser.find_element_by_id('email').send_keys('您的帳號')
browser.find_element_by_id('pass').send_keys('您的密碼')
browser.find_element_by_id('loginbutton').click()
```
輸入帳密並按登入鈕

5-11.py

```
from selenium import webdriver  # 匯入 selenium 的 webdriver

opt =  webdriver.ChromeOptions()      ← 建立選項物件
opt.add_experimental_option('prefs', ← 在選項物件中加入「禁止顯示訊息框」的選項
    {'profile.default_content_setting_values' : {'notifications' : 2}})
browser = webdriver.Chrome(options = opt)← 以 options 參數來建立瀏覽器物件

browser.get('http://www.facebook.com') ← 開啟 Chrome 並連到 FB 網站
browser.find_element_by_id('email').send_keys('您的帳號')
browser.find_element_by_id('pass').send_keys('您的密碼')
browser.find_element_by_id('loginbutton').click()
```
輸入帳密並按登入鈕

這裡請換成實際的 FB 帳號及密碼

實例 2：自動登入 Google 網站

5-12.py

```
from selenium import webdriver    # 匯入 selenium 的 webdriver
from time import sleep            # 匯入內建的 time 模組的 sleep() 函式

opt =  webdriver.ChromeOptions()          # 建立選項物件
opt.add_experimental_option('prefs',      # 加入「禁止顯示訊息框」的選項
    {'profile.default_content_setting_values' : {'notifications' : 2}})
browser = webdriver.Chrome(options = opt) # 以 options 參數來建立瀏覽器物件
```

接下頁

```
browser.get('http://www.google.com') ← 開啟 Chrome 並連到 Google 網站
browser.maximize_window() ← 將視窗最大化以避免最右邊的登入鈕沒顯示出來
browser.find_element_by_id('gb_70').click() ← 按登入鈕
sleep(3) ← 暫停 3 秒等待進入下一頁

                                                          輸入帳號
browser.find_element_by_id('identifierId').send_keys('您的帳號')
browser.find_element_by_id('identifierNext').click()  ← 按繼續鈕
sleep(3) ← 暫停 3 秒等待進入下一頁
                                  請換成實際的
                                  Google 帳號及密碼
                                                          輸入密碼
browser.find_element_by_name('password').send_keys('您的密碼')
browser.find_element_by_id('passwordNext').click()

                                        按繼續鈕即可登入 google 了
```