

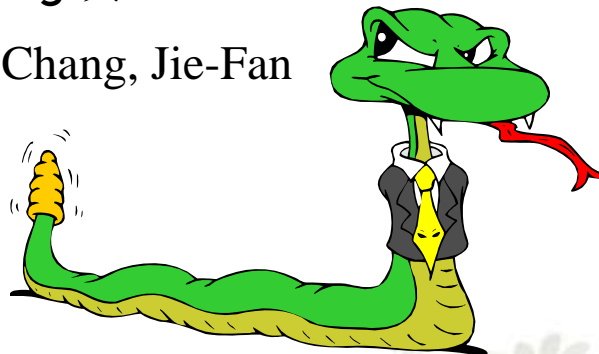
UNIT 8



python™

Importing 與 Modules

張傑帆 Chang, Jie-Fan



OUTLINE

- 內建模組
- 自訂模組
- 第三方模組



模組

- 其實模組就是.py 檔，之前做過的練習儲存起來的.py 檔都是模組
- 而當各位需要使用到的時候，都可以藉由import 指令來匯入現在所編輯的程式中使用
- 至於之前所import 的模組，多是Python內建的模組，以方便大部分程式設計人員所使用，而不用再重新撰寫
- 模組就是保存了程式碼的.py 檔，裡面可能定義了函數、類別、變數等內容



模組

- 要匯入模組，則有兩個指令可以加以使用：**import** 跟 **from...import**，其語法分別是

```
import module1[,module2[,...module]
```

```
from modname import name1[,name2[,...nameN]]
```

- **import** 可以一次匯入多個模組，中間以「，」分隔
- 而**from...import** 則是更精確的描述想要使用的是哪個模組裡面的哪個函數
- 如果匯入模組檔案的路徑底下找不到你所指定的檔案，那**Python** 便會到**sys.path** 裡面尋找



ALT+F+P: 打開路徑瀏覽器，方便選擇導入包進行查看、瀏覽



模組

■ 示範程式碼

```
01 import sys           # 匯入sys 模組
02 print(sys.path)      # 印出sys.path
```

■ 執行結果

```
>>>
```

```
['C:\\Users\\John\\Desktop\\Google 雲端硬碟\\Python  
Wizard\\examples\\Ch9', 'C:\\Python34\\Lib\\idlelib',  
C:\\Windows\\system32\\python34.zip', 'C:\\Python34\\DLLs',  
'C:\\Python34\\lib', 'C:\\Python34', 'C:\\Python34\\lib\\site-packages']
```

```
>>>
```



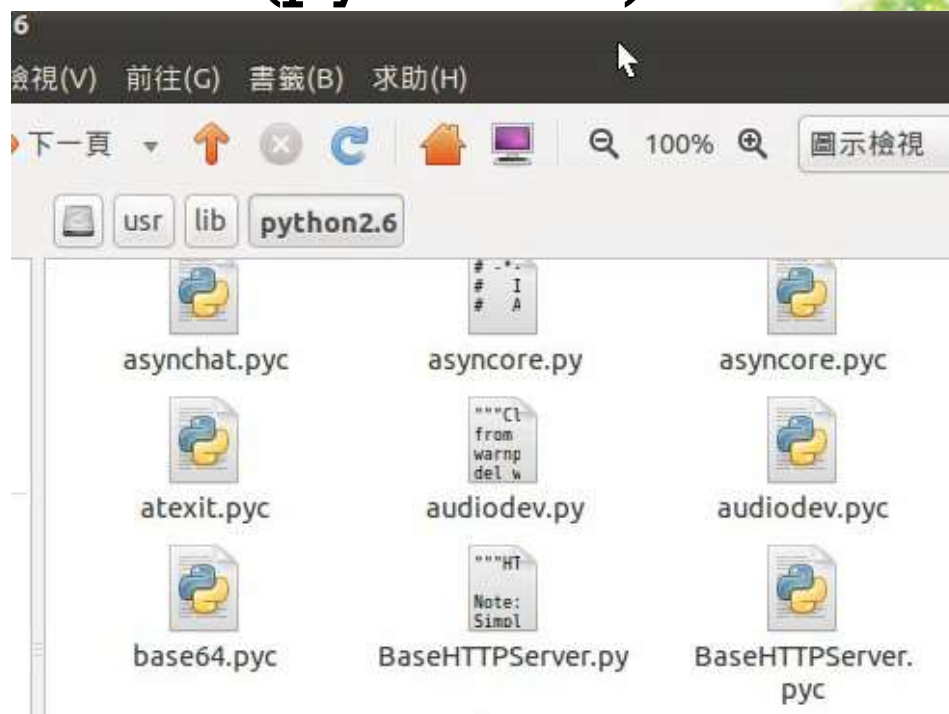
模組的存放位置

- Window

- /安裝Python的資料夾/Lib

- .Linux

- /usr/lib/python2.x (python3.x)



模組

- 透過上面的範例便可知道Python 會在哪裡系統的路徑下找尋要import 的模組檔案
- 如果import 的模組名稱太長，或是想要換名稱，則可使用import...as 這個指令

```
01 import math as ma      # import math 模組，並將其重新命名為ma
02 print(ma.pi)           # 輸出ma 模組裡的變數pi
```

```
>>>
```

```
3.141592653589793
```

```
>>>
```

- 使用import...as 並不會改變這個模組真正的名字，僅是在被匯入的這支程式裡面被更改名稱而已



- <https://stackoverflow.com/questions/2792650/import-error-no-module-name-urllib2>

```
#import urllib2.request  
import urllib.request as urllib2
```

```
response  
= .urlopen("http://www.google.com")  
html = response.read()  
print(html)
```



好用標準函式庫: TIME

- **time.time()**
 - 取得系統時間
- **time.sleep(num)**
 - 設定暫停時間
- **time.localtime()**
 - 回傳格式：
 - **time.struct_time(tm_year, tm_mon, tm_mday, tm_hour, tm_min, tm_sec, tm_wday, tm_yday_, tm_isdst)**
- **time.strftime(TimeFormat)**
 - 格式化輸出時間
- **time.gmtime()**
 - 取得 UTC 世界標準時間

```
>>>
```

```
現在時間: 2014-06-21 17:43:54
```

```
>>>
```



```
>>> help(time.strftime)
```

Help on built-in function strftime in module time:

```
strftime(...)
```

```
    strftime(format[, tuple]) -> string
```

Convert a time tuple to a string according to a format specification. See the library reference manual for formatting codes. When the time tuple is not present, current time as returned by localtime() is used.

Commonly used format codes:

%Y Year with century as a decimal number.

%m Month as a decimal number [01,12].

%d Day of the month as a decimal number [01,31].

%H Hour (24-hour clock) as a decimal number [00,23].

%M Minute as a decimal number [00,59].

%S Second as a decimal number [00,61].

%z Time zone offset from UTC.

%a Locale's abbreviated weekday name.

%A Locale's full weekday name.

%b Locale's abbreviated month name.

%B Locale's full month name.

%c Locale's appropriate date and time representation.

%I Hour (12-hour clock) as a decimal number [01,12].

%p Locale's equivalent of either AM or PM.



小練習

- 請使用系統內建**time**模組 存檔成(MyTime.py)
- 定義一函式**GetTime**可回傳呼叫/現在時間
- 令呼叫函式可自定輸出時間格式
- **Ex:**
 - **GetTime('%Y-%m-%d %H:%M:%S')**
 - **Hint:使用time.strftime() if __name__ == '__main__':**



模組

- 引入上面的小練習
- 呼叫 **GetTime()** 函式

- 程式碼

```
01 import MyTime  
02  
03 print(" 現在時間: " + MyTime.GetTime('%Y-%m-%d %H:%M:%S'))
```

- 執行結果

```
>>>
```

```
現在時間: 2014-06-21 17:43:54
```

```
>>>
```



小練習

- 請試著將以前練習的「排列、組合」函式的.py檔重新命名成「**permutations.py**」後拿來做模組引入練習
- 呼叫以下函式並列印出結果
 - **permutations.fact(n)**
 - **permutations.C(n,m)**
 - **permutations.P(n,m)**



IMPORTING 與 MODULES

- 使用定義在別的檔案的類別(**classes**)與函式(**functions**)
- Python 的 **module** 的名稱會與其檔名相同(加上 **.py** 副檔名)
- 就像 Java 的 **import**, C++ 的 **include**
- 有三種使用方式:

import somefile

from somefile **import** *

from somefile **import** className

- 有什麼不同呢?

in the file and what name
ing



IMPORT ...

```
import somefile
```

- 所有存在於 **somefile.py** 裡的東西都會被加入
- 要引用檔案中的東西，只要加入“**somefile**”文字於其名稱之前

```
somefile.className.method("abc")  
somefile.myFunction(34)  
somefile.cut_off_threshold
```



*FROM ... IMPORT **

`from somefile import *`

- 所有存在於 **somefile.py** 裡的東西都會被加入
- 要引用檔案中的東西只要直接使用其名稱即可
- 所有在其模組中的東西都存在於現有名稱空間中
- 注意：使用這種import方法有可能會覆寫現有同樣名稱的的函式或變數！

`className.method("abc")`

`myFunction(34)`

`cut_off_threshold`



FROM ... IMPORT ...

```
from somefile import className
```

- 只有 在 `somefile.py` 中叫 `className` 的物件會被引入
- 在引入 `className` 後，便可以使用該模組，而且不需在前面加入 `somefile`，因該物件已被引入現有名稱空間中
- 注意：使用這種 `import` 方法有可能會覆寫現有同樣名稱的的函式或變數！

```
from somefile import method
```

```
className.method("abc") ← imported
```

```
myFunction(34) ← Not imported
```

```
cut_off_threshold ← Not imported
```



RANDOM 模組

- 隨機整數：

```
>>> random.randint(0,100)
```
- 隨機選取0到100間的偶數：

```
>>> random.randrange(0, 100, 2)
```
- 隨機浮點數：

```
>>> random.random()  
>>> random.uniform(1, 10)
```
- 隨機字元：

```
>>> random.choice('abcdefg&#x21%*f')
```
- 多個字元中選取特定數量(且不重覆)的內容：

```
>>> random.sample('abcdefghij',3)  
>>> random.sample(range(10),10)
```
- 隨機選取字串：

```
>>> random.choice ( ['apple', 'pear', 'peach', 'orange', 'lemon'] )
```
- 洗牌：

```
>>> items = [1, 2, 3, 4, 5, 6]  
>>> random.shuffle(items) 7
```



小練習 EXERCISE

■ 樂透模擬器

讓程式倒數3秒後 Hint: `time.sleep(x)`

亂數產生6+1個不重覆的數字

6樂透號碼

1個特別號



範例程式碼

```
import time
import random
import time, random

print("抽獎開始 到數3秒")
for i in range(3, 0, -1):
    print(i)
    time.sleep(1)

res = random.sample(range(1, 50), 7)
print("樂透號碼為：")
print(res[:6])
print("特別號為：")
print(res[6])
```

抽獎開始 到數3秒

3

2

1

樂透號碼為：

[38, 34, 23, 20, 49, 40]

特別號為：

31

>>>

1)) ;



使用 SYS 模組

- **sys.argv[0]**
 - 會回傳此程式檔案的位置與名稱
 - 數字代入1以上則是檔案傳入參數
- **len(sys.argv)>1**
 - 是判斷是否有帶入參數
- **sys.builtin_module_names**
 - 回傳Python程式語言內所有內置模組名稱
- **sys.modules.keys()**
 - 得知目前已經載入的模組
- **sys.platform**
 - 取得目前作業系統的版本
- **sys.exit()**
 - 宣告sys.exit(0)終止程式



使用標準函數庫

- **sys.version**

- 回傳目前安裝在系統上的Python版本
- 格式：'(#build_number, build_date, build_time)[compiler]'

- **sys.api_version**

- 回傳Python直譯器的C API版本

- **sys.version_info**

- 回傳一個tuple型態的值
- ('主要版本', '次要版本', '小版本')

- **sys.winver**

- 回傳的版本數字是註冊在Windows裡的Python版本

- **sys.path**

- 定義Python搜尋模組的路徑



範例

sys_demo.py

```
1 import sys
2
3 print('sys.argv =', sys.argv)
4 print('len(sys.argv) =', len(sys.argv))
5 for i in range(len(sys.argv)):
6     print(sys.argv[i])
7
8 print('sys.platform =', sys.platform)
9 print('sys.version =', sys.version)
10 prir >>> n)
11 prir nfo)
12 prir sys.argv = ['D:\\sys_demo.py']
13 prir len(sys.argv) = 1
14 prir 'D:\\sys_demo.py'
```

sys.platform = win32

sys.version = 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24
2015, 22:43:06) [MSC v.1600 32 bit (Intel)]



範例

sys_demo.py

```
1  import sys
2
3  print('sys.api_version =', sys.api_version)
4  print('sys.version_info =', sys.version_info(major=3, minor=4,
5  for i in sys.version_info(major=3, minor=4, micro=3, releaselevel='final', serial=0)
6      print('sys.winver =', sys.winver, 'beta', 'candidate' or 'final')
7
8  print('sys.path =', sys.path)
9  print('sys.exit(0)')
10
11
12
13
14
```

>>>

sys.argv = ['D:\\sys_demo.py']

len(sys.argv) = 1

'D:\\sys_demo.py'

sys.platform = win32

sys.version = 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24

2015, 22:43:06) [MSC v.1600 32 bit (Intel)]

sys.api_version = 1013

sys.version_info = sys.version_info(major=3, minor=4,

micro=3, releaselevel='final', serial=0)

sys.winver = 3.4 'beta'、'candidate'或'final'

sys.path = ...太長請自己試著執行看看

>>>



示範

■ `sys.argv` 於檔案執行時傳入參數

```
import sys, time
```

```
print(sys.argv)
```

```
if '-t' in sys.argv:
```

```
    print('電腦要爆炸啦~~')
```

```
    for i in range(3,0,-1):
```

```
        print(i)
```

```
        time.sleep(1)
```

```
    print("Boom~")
```

```
if '/?' in sys.argv:
```

```
    print('作者很懶，沒寫說明')
```

```
import sys, time
```

```
print(sys.argv)
```

```
if '-t' in sys.argv:
```

```
    print('電腦要爆炸啦')
```

```
    for i in range(3,0,-1):
```

```
        print(i)
```

```
        time.sleep(1)
```

```
    print("Boom~")
```

```
if '/?' in sys.argv:
```

```
    print("參數說明:")
```

```
    print("沒有說明")
```



將特定路徑加入PATH

- 由於python引入模組時，將會尋找sys.path的路徑裡的資料夾中的.py檔案來引入，因此可以自行添加資料夾路徑來新增引入模組的位置

```
import sys  
sys.path.append( '自訂模組所放的資料夾' )  
import MyModule  
  
MyModule.MyFunction()
```



使用標準函數庫

os模組：顯示系統環境參數與指令功能函數

- **os.rename(src, dst)**

- 對檔案或目錄更換名稱
- **src**引數是原本的資料夾
- **dst**引數是修改後的資料夾名稱

- **os.remove(path)**

- 移除檔案
- **path**引數傳入檔案位置
- **不會移除資料夾**，若要移除資料夾可用下一個指令

- **os.removedirs(path)** 或 **os.rmdir(path)**

- 移除**空的資料夾**

- **os.listdir(path)**

- 輸出**path**引數位置的目錄和檔案名稱



使用標準函數庫

使用os模組

- **os.chdir(path)與os.getcwd()**
 - os.chdir(path)函數是切換目錄到path引數位置
 - os.getcwd()是顯示目前所在的目錄位置
- **os.mkdir(path[, mode])與os.rmdir(path)**
 - os.mkdir()是建立資料夾
 - path引數是建立/刪除目錄的位置
 - mode引數是Unix平台使用的
 - os.rmdir()函數是刪除目錄
- **os.path.getsize(path)**
 - 取得檔案大小
- **os.path.getctime(path)**
 - 取得檔案的建立日期



使用標準函數庫

使用os模組

- **os.path.getmtime(path)**
 - 取得檔案的修改日期
- **os.path.getatime(pah)**
 - 取得檔案的存取日期
- **os.path.isfile(path)**
 - 判斷傳入的path引數是否為檔案
- **os.path.isdir(path)**
 - 判斷傳入的path引數是否為目錄
- **os.path.exists(path)**
 - 判斷傳入的path引數目錄或檔案是否存在
- **os.walk(...)**
 - 遞迴列出所有子目錄與檔案
- **os.system(...)** **#os.system("pause")**
 - 執行 Command line 命令



OS.WALK

- 遞迴列出所有子目錄與檔案

os_walk_demo.py

```
1  from os import walk
2
3  #指定要列出所有檔案的目錄
4  mypath = "./"
5
6  #遞迴列出所有子目錄與檔案
7  for root, dirs, files in walk(mypath):
8      print("路徑:", root)
9      print("    目錄:", dirs)
10     print("    檔案:", files)
```



使用標準函數庫

使用 **shutil** 模組 - **High-level file operations**

比較高階的應用層，提供數個針對檔案操作的功能

- **shutil.copytree(src, dst)**
 - 複製整個目錄，包含目錄內的所有檔案
- **shutil.copy(src, dst)**
 - 複製檔案
- **shutil.rmtree(path)**
 - 移除整個目錄，包含目錄內的所有檔案
- **shutil.move(src, dst)**
 - 移動檔案，移動時也可以進行更換檔案名稱
- **shutil.copystat(src, dst)**
 - 複製檔案，會連同檔案屬性一同複製



小練習

- 請使用 **os** 及 **shutil** 模組
- 在 **目前所在的目錄** 下建立一 **files** 資料夾
- 令使用者輸入一數字 **N**
- 並在 **files** 資料夾中建立 **f1, f2... fN** 等 **N** 個資料夾
後列出 **files** 的資料夾內容
- 將 **files** 資料夾裡的 **f1** 資料夾重新命名成 **folder1**
後再列出 **files** 的資料夾內容
- 移除 **files** 資料夾中的 **folder1** 後再列出 **files** 的資料夾內容
- 最後移除 **files** 資料夾
※須先退出 **files** 資料夾 (**os.chdir("../")**)



第三方函式庫

- Python社群提供了大量的**第三方模組**，使用方式與標準庫類似。它們的功能無所不包，覆蓋
 - 科學計算(Scipy, numpy, matplotlib, pandas)
 - 影像處理(opencv, PIL)
 - Web開發(django)
 - 資料庫介面(pymssql, pysqlite)
 - 圖形系統(wxPython, PyQt)
 - 其它學科
- 等多個領域，並且大多成熟而穩定
- 第三方模組可以使用Python或者C語言編寫
- 您也許聽過「不要重造輪子」這句話，或是 DRY (Don't Repeat Yourself)，講得就是「別人已經寫好的東西，就拿去用吧，不用自己再重新寫一套」



安裝第三方函式庫

■ 方法1

- 下載原始碼，手動執行 `python setup.py install` 安裝

■ 方法2：利用第三方安裝工具自動化安裝，如：

- `pip`
- `easy_install`
- `distribute`

■ 方法3:

- 安裝打包好套件程式 `ex: anaconda`



如果有系統中裝有多版本的PYTHON

- 請務必確認你安裝的套件版本
- 確認適合的python版本與位元
- 安裝時是裝到哪個版本的python



PYTHON3 PACKAGES

- PyPI (Python Package Index) ， Python的第三方套件集散地，擁有數萬個套件，包羅萬象的各種需求幾乎都可以找到合適的套件。而使用pip 套件管理程式，能夠方便我們管理、安裝這些套件
- 列出所有支援python3以上的函式庫
- <https://pypi.python.org/pypi?:action=browse&c=533&show=all>



PYQRCODE 0.09

- QR Code生成器，使用python3寫成的可以輸出SVG與PNG的格式
- <https://pypi.python.org/pypi/PyQRCode>
- 手動執行 `python setup.py install` 安裝



```
系統管理員: C:\Windows\system32\cmd.exe

C:\>D:

D:\>cd PyQRCode-1.2.1

D:\PyQRCode-1.2.1>python setup.py install

Installed c:\python34\lib\site-packages\pyqrcode-1.2.1-py3.4.egg
Processing dependencies for PyQRCode==1.2.1
Finished processing dependencies for PyQRCode==1.2.1
```

- 安裝完成後可以 `import pyqrcode` 試試



PYQRCODE 範例

- 將網址輸出成svg格式，比例設定為8

Pyqrcode_ex.py

```
1 from pyqrcode import QRCode
2 url = QRCode('http://www.ntu.edu.tw')
3 url.svg('url.svg', scale=8)
```

- 若出現

`from pyqrcode import QRCode`
`ImportError: No module named 'pyqrcode'`

- 就是沒安裝好



PYQRCODE 範例

- 將網址改成輸出成png圖片格式，比例設定為10

Pyqrcode_ex2.py

```
1 from pyqrcode import QRCode
2 url = QRCode('http://www.ntu.edu.tw')
3 url.png('url.png', scale=10)
```

```
import png
ImportError: No module named 'png'
```

ERROR! 缺少pypng套件

- 須再安裝 pypng 套件

<https://pypi.python.org/pypi/pypng>

安裝成功後才能QR code輸出成png檔案



- 所以要再安裝其它套件
- 這樣安裝套件的方式有點不便
- 有沒有更方便的安裝方法？
- 套件管理套件



PIP安裝步驟

1. 下載原始檔

<https://pypi.python.org/packages/source/p/pip/pip-1.3.1.tar.gz>

2. 解壓縮後照正常套件的方式安裝

3. 執行 `python setup.py install`

4. 設定環境變數



快速安裝法

1. 下載快速安裝檔

<https://bootstrap.pypa.io/get-pip.py>

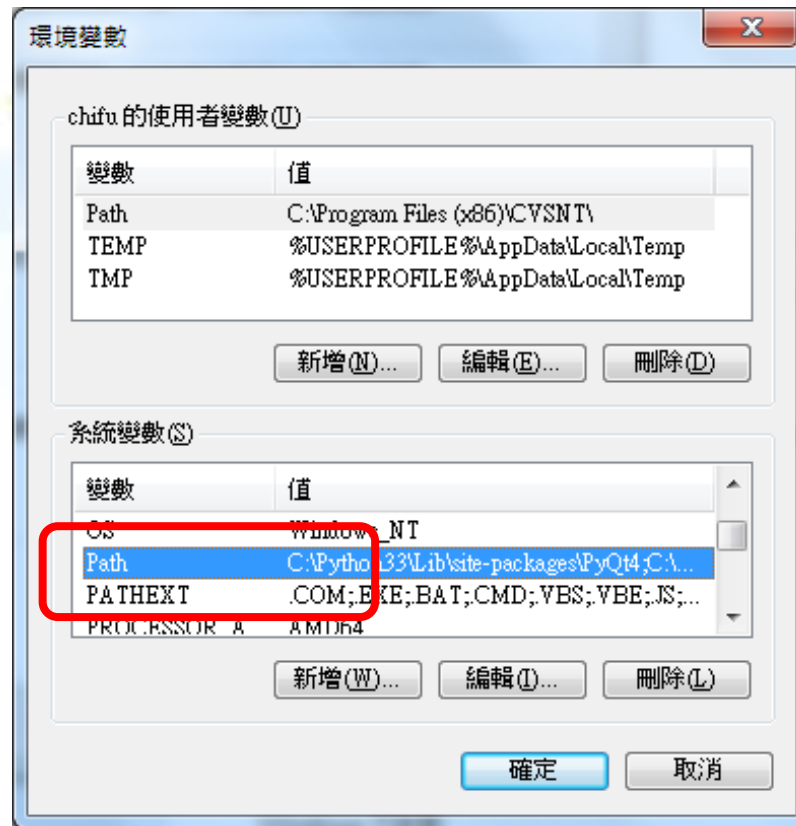
2. 執行 `python get-pip.py`

3. 設定環境變數



設定環境變數

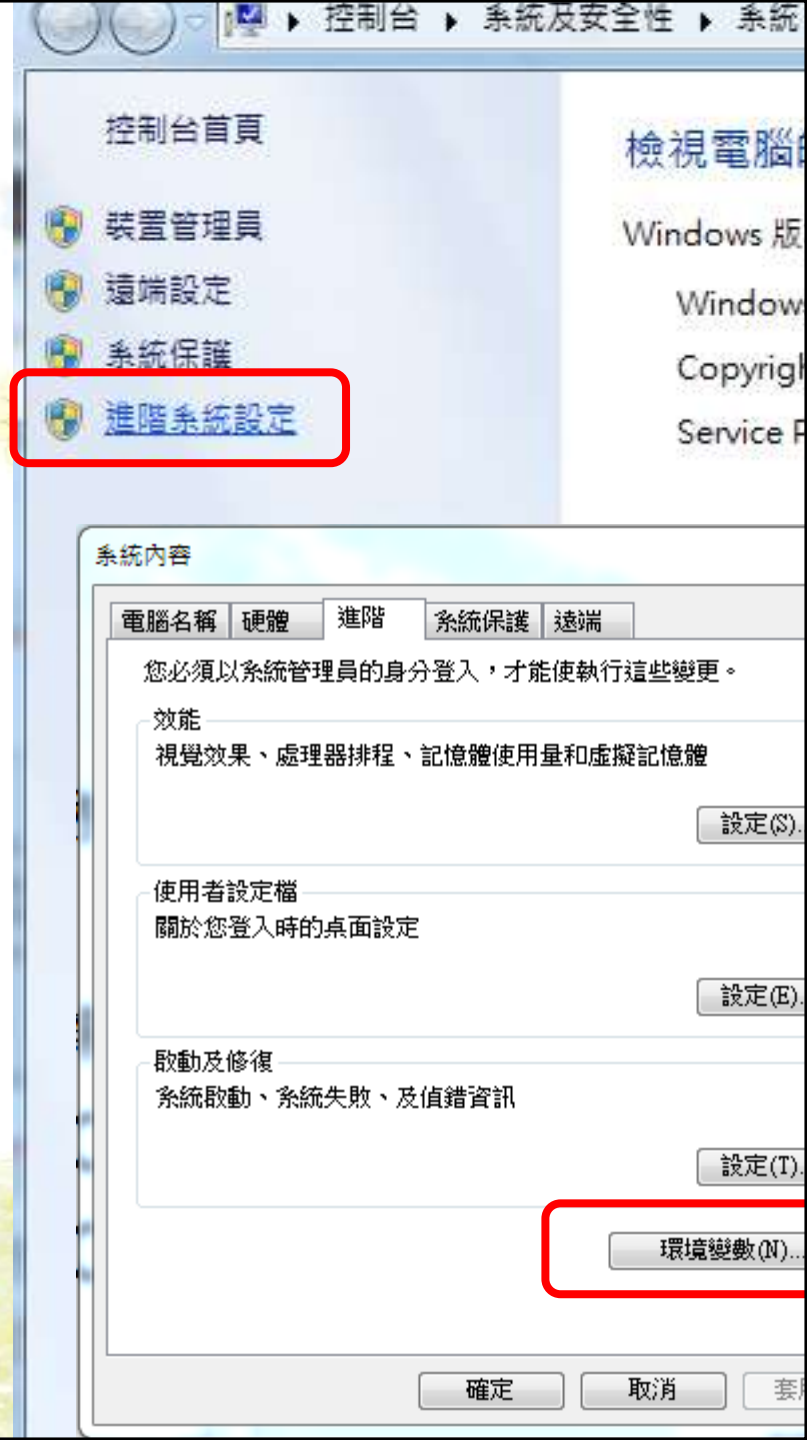
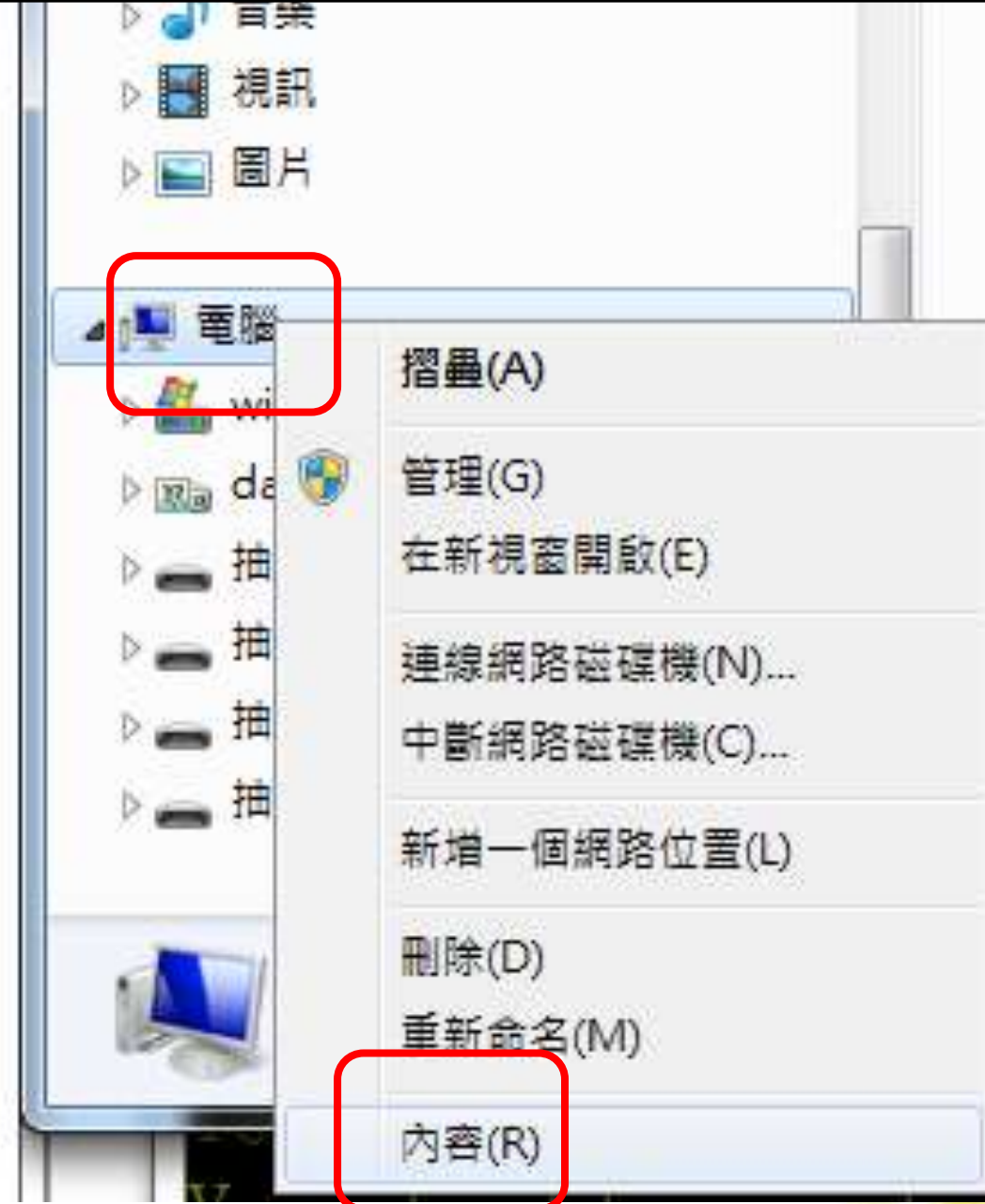
- 於系統變數中的Path
- 新增(記得用「;」號分隔)
- C:\Python34\Scripts;
- C:\Python34;
- C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\Scripts;
- C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64;



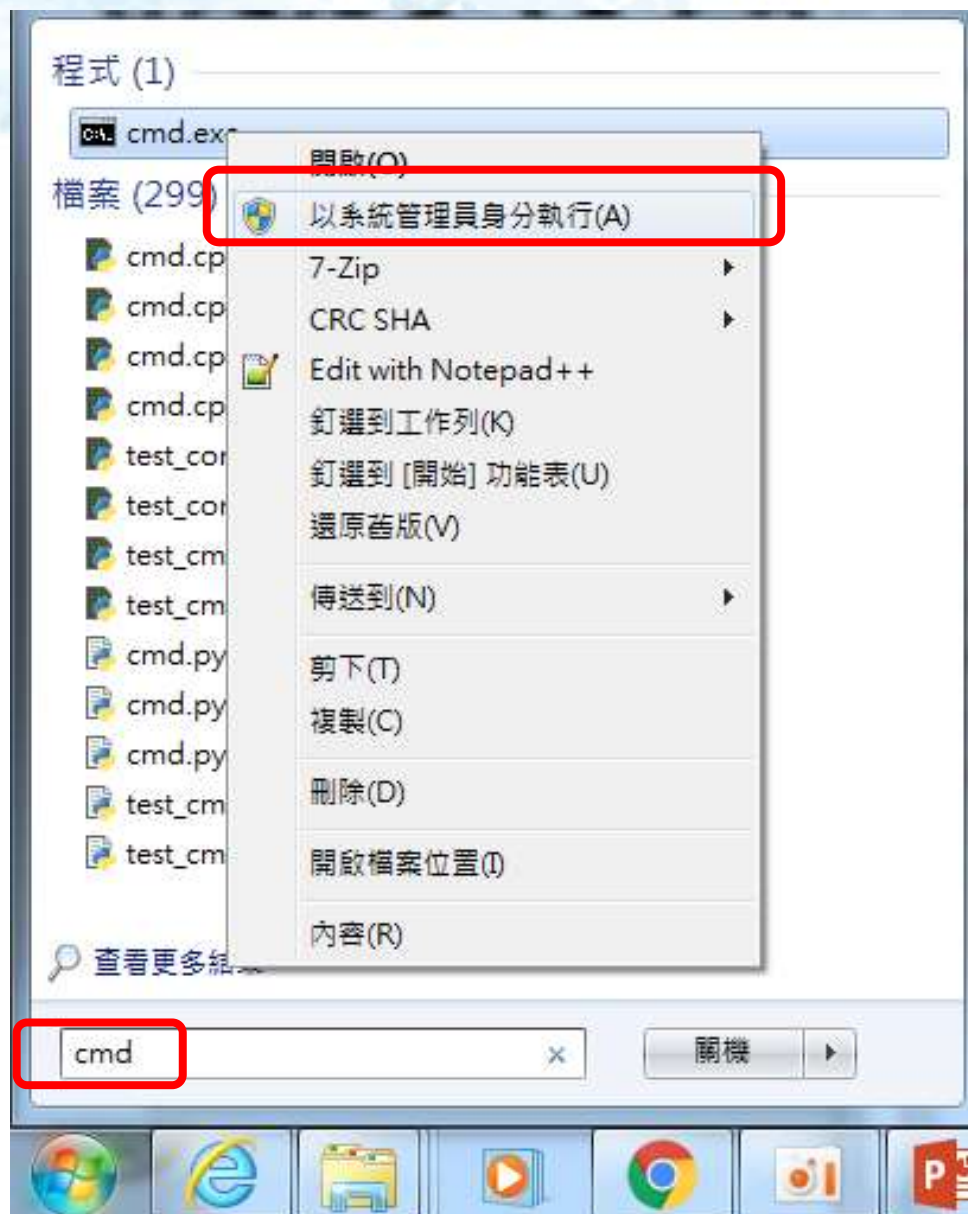
```
C:\>pip
```

'pip' 不是內部或外部命令、可執行的程式或批次檔。



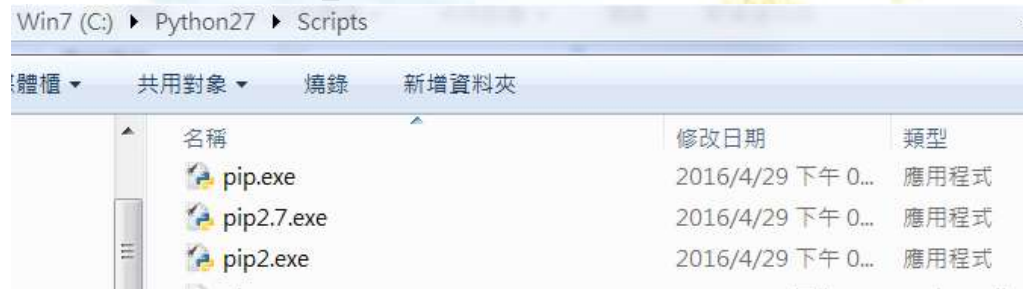


PYTHON 3.6 安裝套件權限

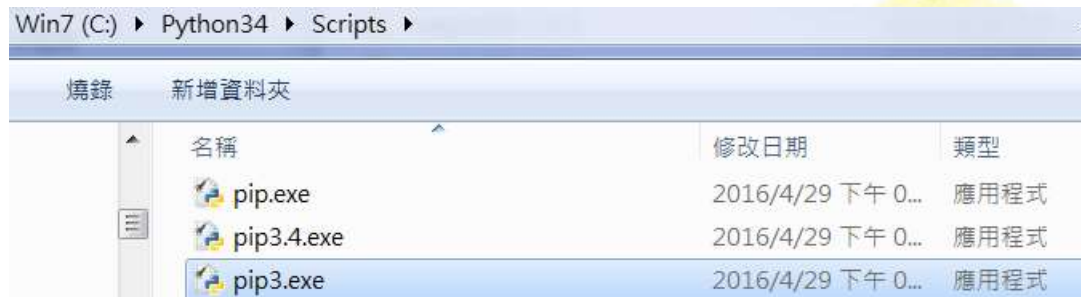


同時裝了PYTHON3和PYTHON2，怎麼用PIP？

- Windows下的解決方案：
- 用python2安裝pip，它的預設路徑是C:\Python27\Scripts



- 然後用python3安裝pip，它的預設路徑是C:\Python34\Scripts

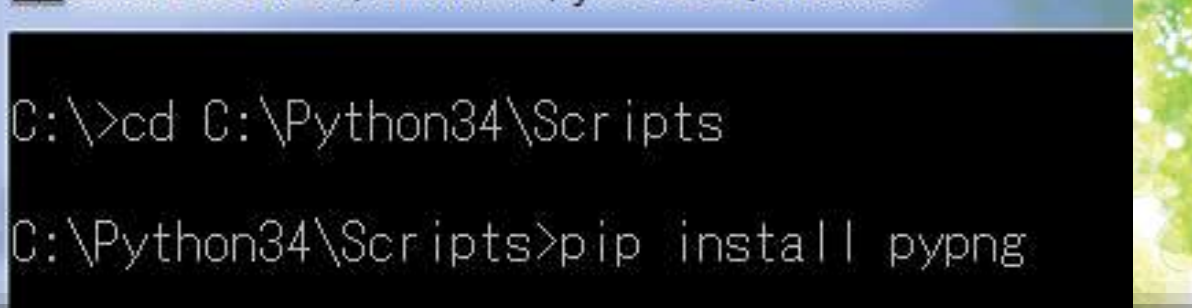


- 然後你把這兩個路徑填加到環境變數中，就可以分別用
pip2 install ... (python2)
pip3 install ... (python3)
分別安裝庫了。



其它的PIP執行方式

- 切換到已安裝pip的python資料夾
Ex: C:\Python34\Scripts
- 然後執行 `pip install pypng` (套件名稱) 來安裝



```
系統管理員: C:\Windows\system32\cmd.exe

C:\>cd C:\Python34\Scripts

C:\Python34\Scripts>pip install pypng

92% | 348kB
94% | 358kB
97% | 368kB
100% | 378kB

B 254kB/s
Building wheels for collected packages: pypng
Running setup.py bdist_wheel for pypng ... done
Stored in directory: C:\Users\shiuny\AppData\Local\pip\Cache\wheels\6d\f0\c9\6
31bc9ec080ed16c9b84b305c55716ea227026904af2d52f1
Successfully built pypng
```



常用的套件簡介

- 列出一些有名、常用到的或是有趣的第三方套件，這些套件可以單獨使用或是配合Django來完成強大的功能。
- 網站框架：
 - Django：完整強大的Web框架
 - Pyramid：另一個完整強大的Web框架
 - web2py：Google app engine 預設使用的框架
 - flask：相較而言是輕量的網站框架



常用的套件簡介

■ 影像、圖片處理：

- **PIL**：Python Imaging Library，PIL 是Python 下最有名的影像處理套件，由許多不同的模組所組成，並且提供了許多的處理功能，允許我們在簡單的Python 程式裡進行影像的處理。
- **Pillow**：PIL的fork版本，另一套圖形處理的套件。
- **OpenCV**：知名影像處理函式庫的Python 版本。

■ 科學計算：

- **Numpy**：支援非常多的科學計算，包含矩陣運算、線性代數、傅立葉轉換等。大多數的科學計算領域上都派得上用場。
- **Scipy**：提供最佳化、線性代數、訊號處理和圖像處理、常微分方程求解...和其他科學與工程中常用的計算等功能
- **Matplotlib**：可以畫出各種圖形如長條圖、分布圖、立體圖等...
- **pandas**：提供處理特殊資料結構，具有數據處理與資料分析的功能
- **scikit-learn**：機器學習的套件，包含內建的分群分類計算、回歸、統計等功能



常用的套件簡介

- 命令列操作及遠端登入：
 - **fabric**：可以直接撰寫**shell**命令，透過**fabric**執行，也支援遠端登入和自定義**shell**命令。
 - **paramiko**：提供遠端登入和部分指令呼叫的功能。
- 網路爬蟲：
 - **Scrapy**：Python爬蟲框架，可以輕易的與**Django**合作。
 - **Requests**：發送網路請求，抓取網頁文件。
- 文件解析器：
 - **beautifulsoup**：可以處理**HTML**、**XML**等標記格式。
 - **lxml**：可以處理**HTML**、**XML**等標記格式，使用**xpath**選取內容。



延申閱讀

- 用 pip 安裝 *.whl 檔

<https://www.lfd.uci.edu/~gohlke/pythonlibs/>

- `pip install --upgrade pip`
- `python -m pip install --upgrade pip`
- 用 distribute 安裝套件
- 用 easy install 安裝套件
- 用 conda 安裝套件

