

# UNIT 8



python<sup>TM</sup>

File, Module, Exception

張傑帆 Chang, Jie-Fan



# OUTLINE

- 檔案
- 例外
- 模組
- 第三方模組



# 檔案處理

- 從檔案讀進資料或將結果存入檔案之中
- 從檔案讀進資料
  - `file.read()`
  - `file.readline()`
  - `file.readlines()`
- 將結果存入檔案
  - `file.write()`
  - `file.writelines()`





# 檔案

- 使用 **open()** 函數來開啟電腦中已經存在的檔案，以便能進行檔案內容的讀取、寫入或修改等動作



- 而使用方式如下所示：

```
file object = open(file_name [, access_mode][, buffering] )
```

# 檔案

```
file object = open(file_name [, access_mode][, buffering] )
```

- `[]` 裡面的參數都是可省略的
- **file\_name** 就是想開啟的檔案，其中也包含了檔案的路徑，如果沒有包含路徑的話，則要開啟的檔案與目前在編輯的程式碼檔都位於同一個資料夾
- **access\_mode** 想要存取的模式，種類繁多，下面分別列表說明
- **buffering** 資料讀入的暫存空間
  - 0 代表沒有暫存空間，大於或等於1 則是代表有暫存空間，及一次可讀入多少資料量，例如，3 就是表示一次讀入3 行的資料量。至於-1，則表示會用預設的暫存區大小來暫存所讀入的資料內容。



# 檔案

模式	描述
r	以唯讀方式打開文件。文件游標將會放在文件的開頭。預設模式。
r+	用於讀寫。文件游標將會放在文件的開頭。
w	功能只限定於寫入。如果該文件已存在則將其覆蓋。如果該文件不存在，創建新文件。
w+	能夠讀寫。如果該文件已存在則將其覆蓋。如果該文件不存在，創建新文件。
a	功能限定為追加資料。文件游標將會放在文件的結尾，新的內容將會被寫入到已有內容之後。如果該文件不存在，創建新文件進行寫入。
a+	能夠讀寫。文件游標將會放在文件的結尾。文件打開時會是追加模式。如果該文件不存在，創建新文件用於讀寫。



模式	描述
rb	以 <b>唯讀</b> 方式打開文件( <b>二進制</b> 格式)。文件游標將會放在文件的 <b>開頭</b> 。 <b>預設模式</b> 。
rb+	以 <b>二進制</b> 格式打開一個文件用於 <b>讀寫</b> 。文件游標將會放在文件的 <b>開頭</b> 。
wb	( <b>二進位</b> 格式)功能只限定於 <b>寫入</b> 。如果該文件已存在則將其 <b>覆蓋</b> 。如果該文件不存在， <b>創建</b> 新文件。
wb+	(以 <b>二進位</b> 格式)能夠 <b>讀寫</b> 。如果該文件已存在則將其 <b>覆蓋</b> 。如果該文件不存在， <b>創建</b> 新文件。
ab	(以 <b>二進位</b> 格式且功能限定為 <b>追加資料</b> 。如果該文件已存在，文件游標將會放在文件的 <b>結尾</b> 。也就是說，新的內容將會被寫入到已有內容 <b>之後</b> 。如果該文件不存在， <b>創建</b> 新文件進行寫入。
ab+	(以 <b>二進位</b> 格式)功能限定為 <b>追加資料</b> 。如果該文件已存在，文件游標將會放在文件的 <b>結尾</b> 。如果該文件不存在， <b>創建</b> 新文件並且能夠 <b>讀寫</b> 。





# 檔案

- 用 **open** 開檔之後要執行的動作至少有以下三種：
- **讀檔**，**寫檔**及**關檔**，以**read()**、**write()**、**close()** 為代表
  - 讀檔有**read()**，**readline()**，**readlines()** 三種
  - 寫檔有**write()**、**writeline()**、**writelines()** 三種
  - 關檔就有**close()** 一種
- **read()** 一次可以**讀取整份文件**，通常會將整份文件的文字放在一個**string** 裡面，對於文件中「行」的概念處理起來相當麻煩，因此在讀檔時並不常使用
- **readlines()** 一次讀取整份文件後將文件依行為單位**存成 list**，接下來就可以使用**for 迴圈**進行處理
- **readline()** **一次只讀取一行**，會比較慢。如果記憶體的空間足夠，建議使用**readlines()**





01	f = open('text.txt', 'r')	# 以讀方式打開檔
02	for line in f.readlines():	# 依次讀取每行
03	line = line.strip()	# 去掉每行頭尾空白
04	print(line)	
05	f.close()	

>>>

I got a chance to Taipei Medical University Hospital for a part time job at this winter vacation. This opportunity is from my thematic advisor who was in charge of updating the information system of NTU Hospital. And now he is the advisor of TMUH, so he knew that the hospital is going to updating their information system during Chinese New Year.

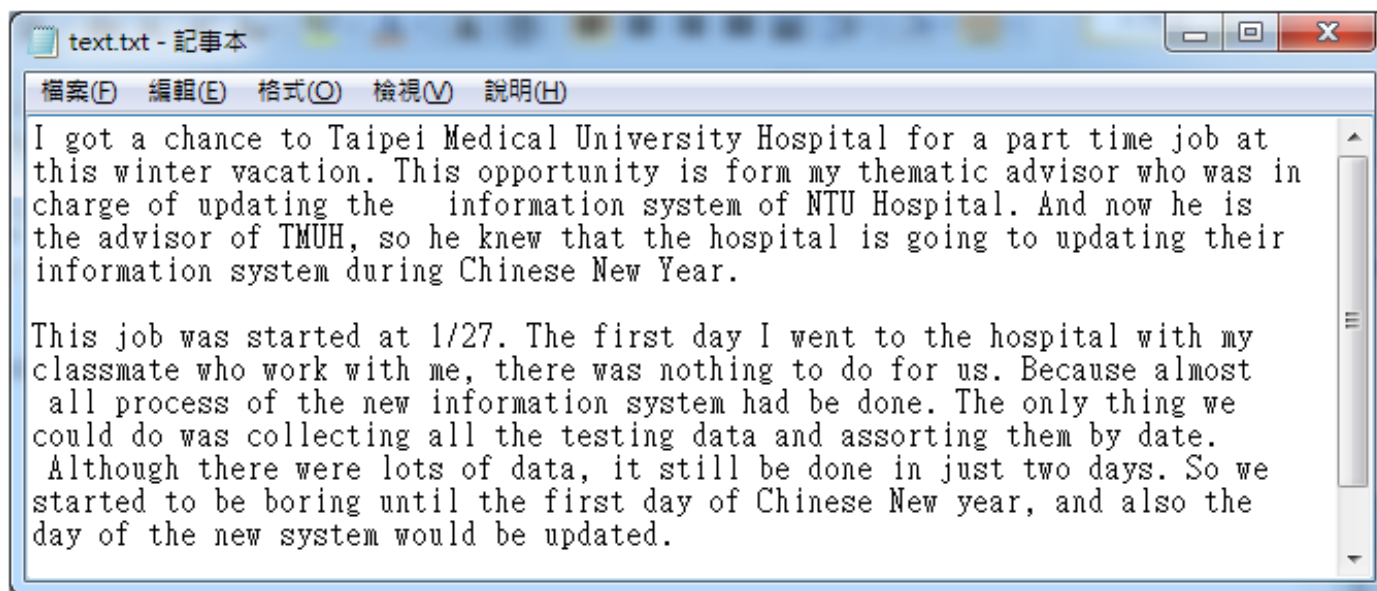
This job was started at 1/27. The first day I went to the hospital with my classmate who work with me, there was nothing to do for us. Because almost all process of the new information system had be done. The only thing we could do was collecting all the testing data and assorting them by date. Although there were lots of data, it still be done in just two days. So we started to be boring until the first day of Chinese New year, and also the day of the new system would be updated.

>>>



# 檔案

## ■ 文字檔案內容



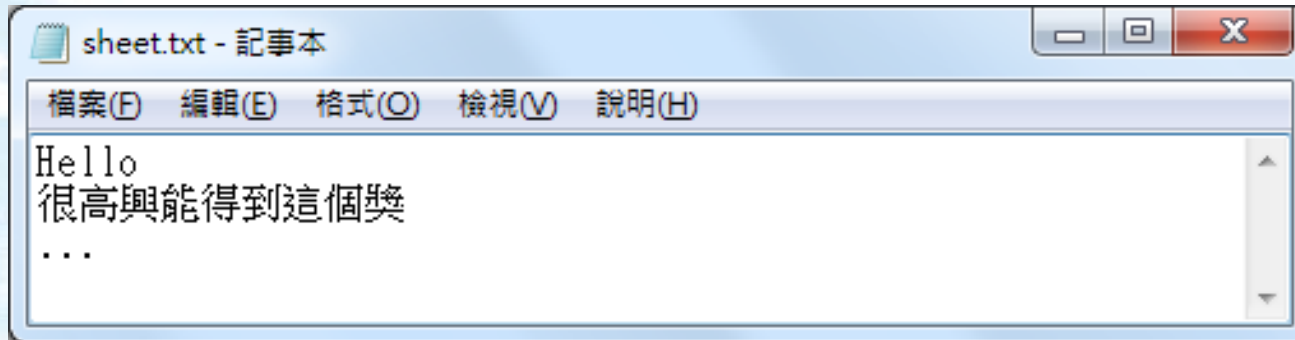
# 讀取檔案

- `name = input('請輸入檔名 :')`  
`file = open(name, 'r', encoding='UTF-8')`  
`content = file.read()`  
`print(content)`  
`file.close()`
- **read()** 方法會一次讀取所有的檔案內容，在不使用檔案時，可以使用 **close()** 將檔案關閉以節省資源。





# 檔案



```
01 f = open("sheet.txt", "r")           # 以唯獨模式開啟檔案
02
03 print(f.readline())                  # 單行讀入
04 print(f.readline())
05 print(f.readline())
06
07 f.close()                           # 關閉檔案
```

```
>>>
```

```
Hello
```

```
很高興能得到這個獎
```

```
...
```

```
>>>
```



# 相對路徑與絕對路徑

- 相對路徑 相對於現在目錄的路徑表示法，因此「相對路徑」所指到的檔案或目錄，會隨著現在目錄的不同而改變
- 假設當前目錄於E:\test\file
  - ./ 表示當前路徑，相當於E:\test\file
  - ../ 表示當前路徑的上一級路徑，相當於E:\test
  - ./data 表示當前路徑下一級路徑，相當於E:\test\file\data



# 相對路徑與絕對路徑

- 絕對路徑
- 指的是一個**絕對的位置**，並不會隨著現在目錄的改變而改變
- EX:
  - E:\test
  - E:\test\file
  - E:\test\file\data





# 小練習

- 試著使用 `readlines` 讀取一 `*.csv` 檔後將其印出



# CSV檔

- Comma-Separated Values (逗號分隔值)
- 檔案以純文字形式儲存表格資料（數字和文字）
- 是一種被分隔的資料格式，它有被逗號字元分隔的欄位/列和以換行結束的記錄/行。
- 由任意數目的記錄組成，每條記錄由欄位組成，欄位間的分隔符號，最常見的是逗號或制表符號
- CSV檔案格式的通用標準並不存在，但是7-bit ASCII是最基本的通用編碼。



# EXCEL開CSV檔亂碼解決

## 匯入字串精靈 - 步驟 3 之 1

資料剖析精靈判定資料類型為 分隔符號。

若一切設定無誤，請選取 [下一步]，或選取適當的資料類別。

原始資料類型

請選擇最適合剖析您的資料的檔案類型：

- ☒ 分隔符號(D) 一 用分欄字元，如逗號或 TAB 鍵，區分每一個欄位。
- ☐ 固定寬度(W) 一 每個欄位固定，欄位間以空格區分。

起始列號(R): 1 檔案原始格式(O): 65001 : Unicode (UTF-8)

預覽檔案 C:\Users\user\Downloads\政府資料開放平臺資料集清單.csv。

1 "資料集名稱", "檔案格式", "下載連結", "資料集類型", "資料集描述", "主要欄位說明", "資料集提供機  
2 "蒙藏委員會補助團體私人情形季報表", "csv", "http://www.mtac.gov.tw/pages/283/MTAC005.csv", "  
3 "蒙藏用語對照表", "csv", "http://www.mtac.gov.tw/pages/283/MTAC004.zip", "rawData", "蒙藏用語  
4 "蒙藏用語對照表", "zip", "http://www.mtac.gov.tw/pages/283/MTAC004.zip", "other", "蒙藏用語對

取消

< 上一步(B)

下一步(N) >

完成(F)

## 匯入字串精靈 - 步驟 3 之 2

您可在畫面中選擇輸入資料中所包含的分隔符號，您可在預覽視窗內看到分欄的結果。

分隔符號

☒ Tab 鍵(T)

☐ 分號(M)

☐ 逗點(C)

☐ 空格(S)

☒ 其他(O): ,

☐ 連續分隔符號視為單一處理(R)

文字辨識符號(Q): "

預覽分欄結果(P)

資料集名稱	檔案格式	下載連結	資料
蒙藏委員會補助團體私人情形季報表	csv	http://www.mtac.gov.tw/pages/283/MTAC005.csv	rawD
蒙藏用語對照表	csv	http://www.mtac.gov.tw/pages/283/MTAC004.zip	rawD
蒙藏用語對照表	zip	http://www.mtac.gov.tw/pages/283/MTAC004.zip	othe

取消

< 上一步(B)

下一步(N) >

完成(F)

名稱

修改日期

政府資料開放平臺資料集清單

2015/6/12 上午 0...



NTU CSIE

檔案名稱(N): 政府資料開放平臺資料集清單

文字檔案

工具(L)

匯入(M)

取消

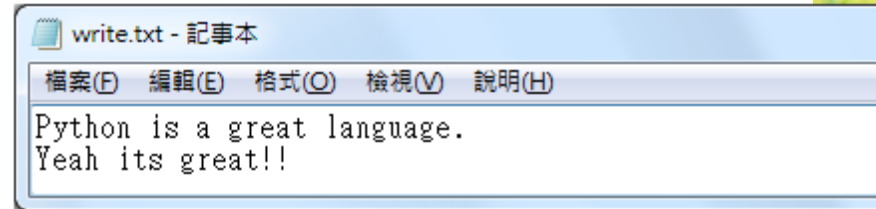


- `write()` 方法可將任何字串寫入目前已經被打開的檔案
- `write()` 方法不會自動在字串的結尾中添加分行符號（`'\n'`）
- 因此如果要換行則必須自行加上分行的符號

```
01 # 打開一個檔
02 fo = open("write.txt", "w+")
03 fo.write("Python is a great language.\nYeah its great!!\n")
05 fo.close()      # 關閉打開的文件
07 # 執行完上面的程式碼，會在這個Python 檔案的路徑中新建一個
   write.txt
```

```
09 # 下面要再開啟這個檔，並且輸出這個檔的內容
```

```
10 fo = open("write.txt", "r")
11 for line in fo.readlines():
12     line = line.strip()
13     print(line)
14 fo.close()
```



```
>>>
```

```
Python is a great language.
```

```
Yeah its great!!
```

```
>>>
```



# 檔案

- **writeline()**
  - 輸入完之後會換行
- **writelines()**
  - 則是將**list** 當作輸入參數，以**list** 的資料型態將資料寫入檔案
  - 當**list** 中每個元素被寫入後，並不會自動換行



# 小練習 I-PLUS

- 1. 讀取\*.csv檔
- 2. 寫入另一\*.csv文字檔中  
(僅換行，不用分隔「,」)
- 3. 將其填入list中
  - str.strip()
  - str.split()
- 4. 將上述list寫到另一新的csv檔中





# 進階練習 II

- 讀取 stores\_old.csv 檔案後，將 sid,name,tel,wifi 篩選出來並寫入檔案至 stores\_new.csv 中 (要分隔「,」)

	A	B	C	D	E	F	G	H
1	sid	longitude	latitude	name	address	tel	wifi	notes
2	1	121.740089	25.132571	海景門市	200基隆市仁愛區港西街6-2號	02-2428-4223	Y	
3	2	121.51767	25.047029	新世界門市	100台北市中正區忠孝西路一段49號B1	02-2371-7092		
4	3	121.509098	25.041583	寶平門市	100台北市中正區寶慶路32號	02-2331-0388	Y	
5	4	121.531439	25.017797	羅斯福門市	106台北市大安區羅斯福路三段301號	02-2364-6845	Y	
6	5	121.514167	25.046001	懷寧門市	100台北市中正區懷寧街10號	02-2389-3776	Y	
7	6	121.57694	25.062673	舊宗門市	114台北市內湖區舊宗路一段188號一樓	02-8792-8408	Y	
8	7	121.563588	25.041256	聯合門市	110台北市信義區忠孝東路四段561號1樓	02-2749-5657	Y	
9	8	121.548296	25.051753	環亞門市	105台北市松山區南京東路三段337號1F	02-8712-0625	Y	
10	9	121.566807	25.082169	環山門市	114台北市內湖區內湖路一段289號	02-2799-5897	N	
11	10	121.540795	25.06212	龍樓門市	104台北市中山區民權東路三段22號	02-2517-9259	Y	
12	11	121.548336	25.041369	龍門門市	106台北市大安區忠孝東路四段134號	02-2740-6782	Y	
13	12	121.543721	25.03332	與南門市	106台北市大安區復興南路一段323號1樓	02-2325-9473	Y	
14	13	121.544467	25.053455	與北門市	105台北市松山區復興北路147號1、2F	02-2712-8189	Y	
15	14	121.557886	25.048184	德復門市	105台北市松山區八德路三段251號1~2F	2578-8732	Y	
16	15	121.507392	25.043915	漢中門市	108台北市萬華區漢中街51號1~3樓	02-2370-5893	Y	
17	16	121.546205	25.083801	富隆門市	104台北市中山區大直街70號1樓	02-2532-5223	Y	
18	17	121.553019	25.041263	鼎豐門市	106台北市大安區忠孝東路四段218號一樓	2773-7207	Y	
19	18	121.557309	24.999556	萬芳門市	116台北市文山區興隆路三段111號	02-2933-6950	Y	
20	19	121.569521	25.078441	瑞備門市	114台北市內湖區瑞光路476號	02-8798-3266	Y	
21	20	121.572749	25.077284	瑞光門市	114台北市內湖區瑞光路356號	02-2659-3912	Y	
22	21	121.567366	25.035954	新光門市	110台北市信義區松壽路11號1樓	02-8789-4099	N	
23	22	121.527067	25.118405	新天母門市	111台北市士林區天母西路33號1樓	02-2876-9351	N	
24	23	121.547395	25.152201	陽明山門市	112台北市北投區湖山路一段14號之11樓	02-2861-0399	Y	
25	24	121.556934	25.042442	華南門市	106台北市大安區光復南路180號14號	02-8771-9692	Y	
26	25	121.575919	25.075292	港南門市	114台北市內湖區瑞光路216號	02-8733-5105	Y	

stores\_old.csv

	A	B	C	D	E	F	G	H	I
1	sid	name	address	wifi					
2	1	海景門市	200基隆市仁愛區港西街6-2號	Y					
3	2	新世界門市	100台北市中正區忠孝西路一段49號B1						
4	3	寶平門市	100台北市中正區寶慶路32號	Y					
5	4	羅斯福門市	106台北市大安區羅斯福路三段301號	Y					
6	5	懷寧門市	100台北市中正區懷寧街10號	Y					
7	6	舊宗門市	114台北市內湖區舊宗路一段188號一樓	Y					
8	7	聯合門市	110台北市信義區忠孝東路四段561號1樓	Y					
9	8	環亞門市	105台北市松山區南京東路三段337號1F	Y					
10	9	環山門市	114台北市內湖區內湖路一段289號	N					
11	10	龍樓門市	104台北市中山區民權東路三段22號	Y					
12	11	龍門門市	106台北市大安區忠孝東路四段134號	Y					
13	12	與南門市	106台北市大安區復興南路一段323號1樓	Y					
14	13	與北門市	105台北市松山區復興北路147號1、2F	Y					
15	14	德復門市	105台北市松山區八德路三段251號1~2F	Y					
16	15	漢中門市	108台北市萬華區漢中街51號1~3樓	Y					
17	16	富隆門市	104台北市中山區大直街70號1樓	Y					
18	17	鼎豐門市	106台北市大安區忠孝東路四段218號一樓	Y					
19	18	萬芳門市	116台北市文山區興隆路三段111號	Y					
20	19	瑞備門市	114台北市內湖區瑞光路476號	Y					
21	20	瑞光門市	114台北市內湖區瑞光路356號	Y					
22	21	新光門市	110台北市信義區松壽路11號1樓	N					
23	22	新天母門市	111台北市士林區天母西路33號1樓	N					
24	23	陽明山門市	112台北市北投區湖山路一段14號之11樓	Y					
25	24	華南門市	106台北市大安區光復南路180號14號	Y					
26	25	港南門市	114台北市內湖區瑞光路216號	Y					

stores\_new.csv



# 進階練習 III

- 讀取 `stores_old.csv` 檔案後
- 小練習 1:  
請列出 **公館門市** 的所有資訊
- 小練習 2:  
令使用者輸入一 **門市名稱**，並列出該門市的所有資訊



# (回家) 小練習 IV

- 讀取 `stores_old.csv` 檔案後
- 列出所有有wifi的門市的所有資訊





# 例外

- 程式碼在執行時，也可能會發生錯誤
- 這代表著在設計這支程式的時候，有些地方不甚周詳
- 或是在執行這支程式時，發生了**意想不到的錯誤**
- 如果**不希望程式就此終止**
- 就需要將可預期的錯誤另外撰寫程式來預防
- 使用**try...except**。
- 至少要存在一個**except** 敘述
- **else:** 跟**finally:**則是可有可無

**try:**

欲執行的程式碼T

**except** 例外情形1 [參數]:

欲執行的程式碼1

**except** 例外情形2 [參數]:

欲執行的程式碼2

...

**except** 例外情形N [參數]:

欲執行的程式碼N

**[else:**

欲執行的程式碼E ]

**[finally:**

欲執行的程式碼F ]



# 為什麼需要例外處理

- 除以0

```
a = 10
```

```
b = 0
```

```
print(a/b)
```

```
print("hello python")
```



# 加入例外處理後

```
a = 10
```

```
b = 0
```

```
try:
```

```
    print(a/b)
```

```
except:
```

```
    print("例外發生，不可除以0")
```

```
print("hello python")
```





# 例外

- 如果當try 裡面的程式碼執行時發生異常，Python 就跳回到try 的起始位置，並執行第一個符合該異常的except 敘述。等異常處理完畢後，程式流程就算執行完整個try 區塊中的程式碼，除非在處理異常時又引發新的異常
- 如果在try 裡面的程式碼裡發生了異常，卻沒有匹配的except 敘述，異常將被轉送到上一層的try 區塊處理，或者到程式的最上層。若是遇到後者情況，將結束程式，並印出錯誤資訊。
- 如果在try 的程式碼執行並無引發異常，如果有else: 的區塊內容的話，Python 便會去執行else: 區塊內部的程式碼
- 最後執行finally: 區塊內部的程式碼，如此便算順利執行完整個try...except 指令區(不管有沒有異常，都會執行)



# 例外處理

- **try-except** 也可以和 **else** 連用，**else** 後的程式區塊放的是沒有發生例外，程式所執行的工作，例如

```
a = 10
```

```
b = 0
```

```
try:
```

```
    print(a/b)
```

```
except:
```

```
    print("例外發生，不可除以0")
```

```
else:
```

```
    print("沒有例外發生")
```

```
print("hello python")
```



# 例外

- 以下舉例說明：

```
01 i = 10
02 for j in range(3, -4, -1):
03     print('%d/%d=%0.3f'%(i, j, i / j))
```

```
>>>
```

```
10/3=3.333
```

```
10/2=5.000
```

```
10/1=10.000
```

Traceback (most recent call last):

File "C:\Users\John\Desktop\Google 雲端硬碟\Python Wizard\examples\Ch9\ex09\_05.py", line 3, in <module>

```
    print('%d/%d=%0.3f'%(i, j, i / j))
```

ZeroDivisionError: division by zero

```
>>>
```





# 例外

## ■ 示範程式碼

```
01 i = 10
02 for j in range(3, -4, -1):
03     try:
04         print('%d/%d=%0.3f'%(i, j, i / j))
05     except ZeroDivisionError:
06         print('除數為0，無法進行除法')
```

```
>>>
10/3=3.333
10/2=5.000
10/1=10.000
除數為0，無法進行除法
10/-1=-10.000
10/-2=-5.000
10/-3=-3.333
>>>
```



# 例外

- 如果沒有加上錯誤的類型，則所產生任何錯誤都會被歸類到某個特定的**except** 語句來處理。
- 好處是沒有錯誤會被遺漏，程式一定可以正常執行完畢
- 壞處是如此一來，便無法得知發生什麼錯誤，也就沒有辦法輕易地修改程式碼來避免錯誤的發生
- 建議最好不要把**except** 後面的錯誤類型留白
- Python 有內建相當多的錯誤類型，每一種對應的情況都不同，像上面範例所示範的**ZeroDivisionError** 便是專門對應 0 在除數的除法錯誤
- 另外，還有對應**type** 錯誤的**TypeError**、對應導入模組錯誤的**ImportError** 等等，大部分都可以從錯誤名稱看出錯誤的原因



# 例外處理

- 例外的名稱也可以寫在 **except** 之後，如

```
a = 10
```

```
b = 0
```

```
try:
```

```
    print(a/b)
```

```
except NameError:
```

```
    print("例外發生，NameError")
```

```
except ZeroDivisionError:
```

```
    print("例外發生，不可除以0")
```

```
print("hello python")
```





# 例外處理

- **try-except** 也可以和 **else** 連用，**else** 後的程式區塊放的是沒有發生例外，程式所執行的工作，例如

```
a = 10
```

```
b = 0
```

```
try:
```

```
    print(a/b)
```

```
except NameError:
```

```
    print("例外發生，NameError")
```

```
except ZeroDivisionError:
```

```
    print("例外發生，不可除以0")
```

```
else:
```

```
    print("沒有例外發生")
```

```
print("hello python")
```



## ■ 確保程式能完整執行完畢

```
01 try:
02     print(" 進入try 區塊")
03     for i in range(5, -5, -1):
04         print("10 / " + str(i) + " = ", end = "")
05         print(str(10/i))
06 except:
07     print(" 發生錯誤，進入except 區塊")
08     print(" (except) i = " + str(i))
09 else:
10     print(" 沒有錯誤，進入else 區塊")
11     print(" (else) i = " + str(i))
12 finally:
13     print(" 進入finally 區塊")
14     print(" (finally) i = " + str(i))
```



# 例外

## ■ 執行結果

>>>

進入try 區塊

$10 / 5 = 2.0$

$10 / 4 = 2.5$

$10 / 3 = 3.3333333333333335$

$10 / 2 = 5.0$

$10 / 1 = 10.0$

$10 / 0$  = 發生錯誤，進入except 區塊

(except)  $i = 0$

進入finally 區塊

(finally)  $i = 0$

>>>



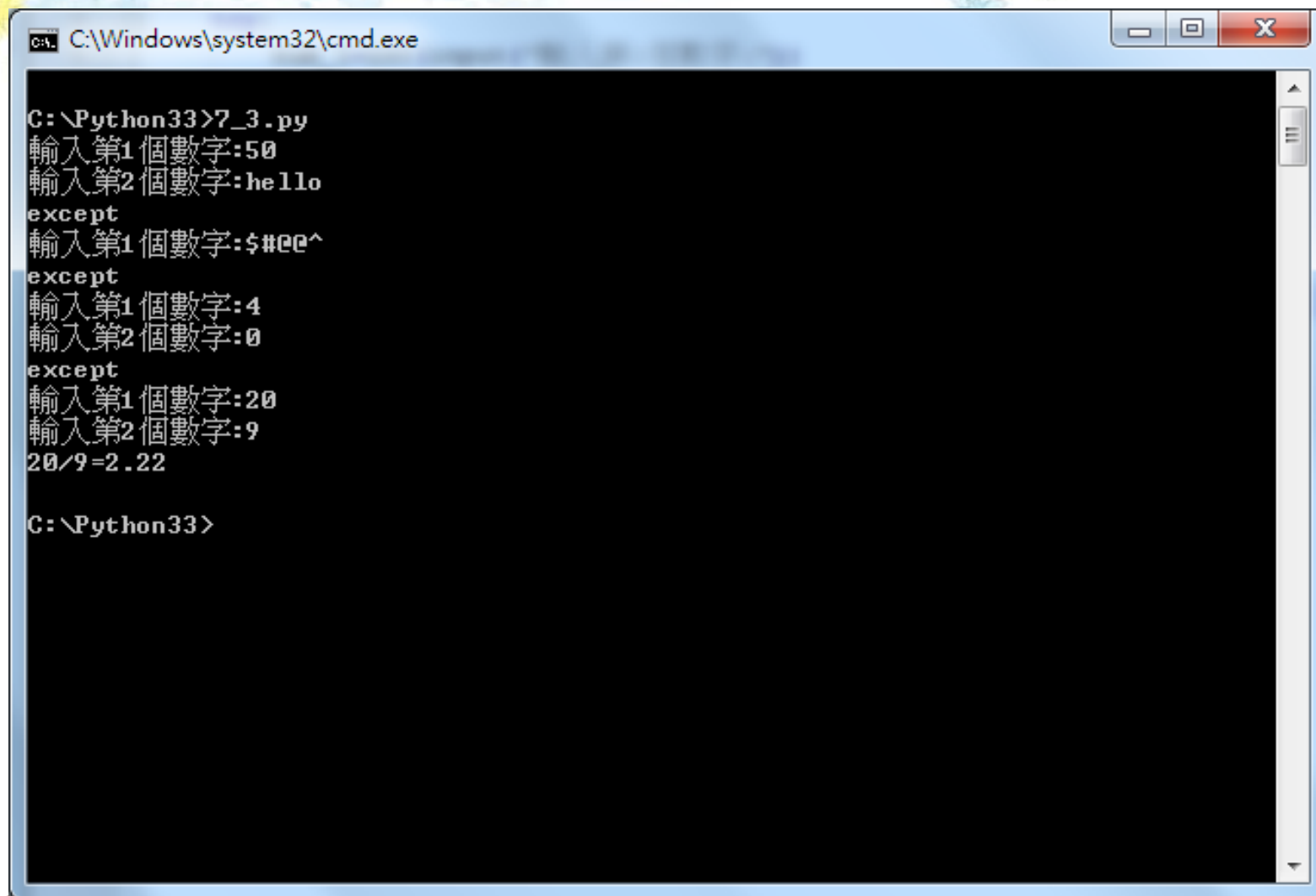


# 課堂練習

- 利用**input**的方法，輸入兩個數字，並將兩個數字相除的結果印出。
- 加入例外處理機制，若發生任何例外則讓使用者重新輸入直到正確將相除的結果印出才終止程式



# 執行範例



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with white text. The text shows the execution of a Python script named "7\_3.py" in the directory "C:\Python33". The script prompts for two inputs: "輸入第1個數字:" and "輸入第2個數字:". The first input is "50" and the second is "hello". The script then prompts for "輸入第1個數字:" again, and the input is "\$#ee^". The script then prompts for "輸入第1個數字:" and "輸入第2個數字:" again, with inputs "4" and "0" respectively. The script then prompts for "輸入第1個數字:" and "輸入第2個數字:" again, with inputs "20" and "9" respectively. The final output is "20/9=2.22". The command prompt then shows "C:\Python33>".

```
C:\Windows\system32\cmd.exe

C:\Python33>7_3.py
輸入第1個數字:50
輸入第2個數字:hello
except
輸入第1個數字:$#ee^
except
輸入第1個數字:4
輸入第2個數字:0
except
輸入第1個數字:20
輸入第2個數字:9
20/9=2.22

C:\Python33>
```



# 例外

- 以下列出Python 常見的錯誤類型：

異常名稱	描述
BaseException	所有異常的基礎類型
SystemExit	編譯器請求退出
KeyboardInterrupt	用戶中斷執行(通常是輸入^C)
Exception	常規錯誤的基礎類型
StopIteration	迭代器沒有更多的值
GeneratorExit	generator發生異常來通知退出
SystemExit	Python 編譯器請求退出
StandardError	所有的內建標準異常的基礎類型
ArithmeticError	所有數值計算錯誤的基礎類型
FloatingPointError	浮點計算錯誤
OverflowError	數值運算超出最大限制





# 例外

異常名稱	描述
OverflowError	數值運算超出最大限制
ZeroDivisionError	除(或取餘數)零(所有資料類型)
AssertionError	斷言語句失敗
AttributeError	物件沒有這個屬性
EOFError	沒有內建輸入,到達EOF(End Of File) 標記
EnvironmentError	作業系統錯誤的基礎類型
IOError	輸入/輸出操作失敗
OSError	作業系統錯誤
WindowsError	系統調用失敗
ImportError	導入模組/對象失敗
KeyboardInterrupt	用戶中斷執行(通常是輸入 ^C)



# 例外

異常名稱	描述
LookupError	無效數據查詢的基礎類型
IndexError	序列中沒有此索引(index)
KeyError	集合中沒有這個鍵
MemoryError	內存溢出錯誤(對於Python 編譯器不是致命的)
NameError	未宣告/初始化對象(沒有屬性)
UnboundLocalError	訪問未初始化的本地變數
ReferenceError	Weak reference試圖訪問已經垃圾回收了的對象
RuntimeError	一般的執行階段錯誤
NotImplementedError	尚未實作的方法
SyntaxError	Python 語法錯誤
IndentationError	縮排錯誤



# 例外

異常名稱	描述
TabError	Tab 和空格混用
SystemError	一般的編譯器系統錯誤
TypeError	對類型無效的操作
ValueError	傳入無效的參數
UnicodeError	Unicode 相關的錯誤
UnicodeDecodeError	Unicode 解碼時的錯誤
UnicodeEncodeError	Unicode 編碼時錯誤
UnicodeTranslateError	Unicode 轉換時錯誤
Warning	警告的基礎類型
DeprecationWarning	關於被棄用的特徵的警告
FutureWarning	關於構造將來語義會有改變的警告



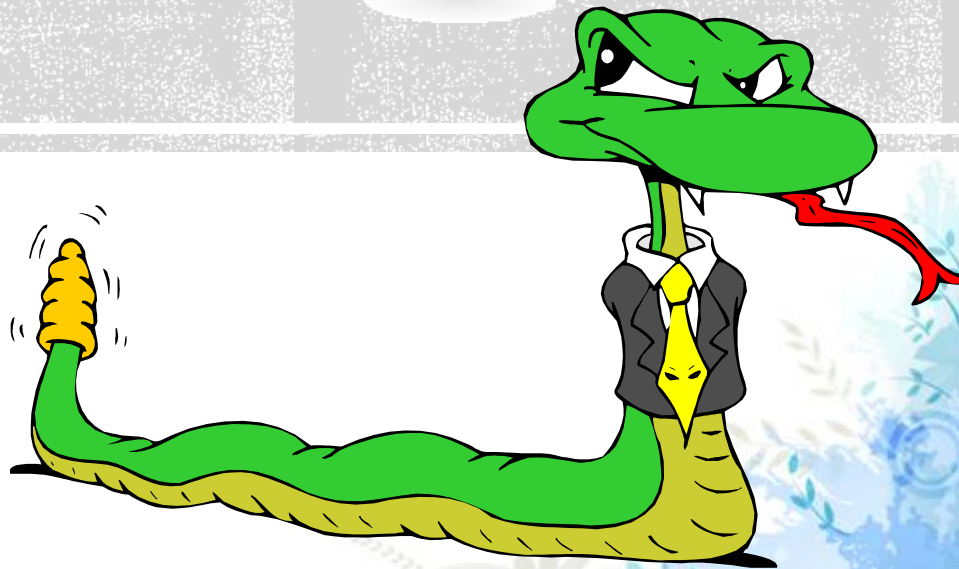


# 例外

異常名稱	描述
OverflowWarning	溢位警告
PendingDeprecationWarning	關於特性將會被廢棄的警告
RuntimeWarning	可疑的runtime behavior警告
SyntaxWarning	可疑的語法的警告
UserWarning	用戶代碼生成的警告



# IMPORTING 與 MODULES



# 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



# 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

- 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





# 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





# 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()`



# 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
```

```
>>>
```



# 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的字元：  

```
>>> random.sample('abcdefghij',3)
```
- 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
```

```
peopleList = range(1,51)
#seconds = list(i for i in range(1,4));
seconds = list(range(1,4))
seconds.reverse()
```

```
print('抽獎開始 倒數10秒')
for i in list(seconds):
    print(i)
    time.sleep(1)
```

```
result = random.sample(peopleList,7)
print('樂透號碼為:\n',result[0:6])
print('特別號為:\n',result[6])
```

>>>

抽獎開始 倒數10秒

3

2

1

樂透號碼為:

[23, 10, 3, 19, 38, 40]

特別號為:

9

>>> |



# 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



# 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.system("...") #os.system("pause")**
  - 執行 Command line 命令

■ <http://docs.python.org/3.3/library/os.path.html>



# 小練習

- 請使用 **os** 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** 資料夾當中建立 **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** 資料夾依序重新命名成 **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



# 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(matplotlib, opencv)、Web開發(django)、資料庫介面(pymssql, pysqlite)、圖形系統(wxPython, PyQt)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`





# 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



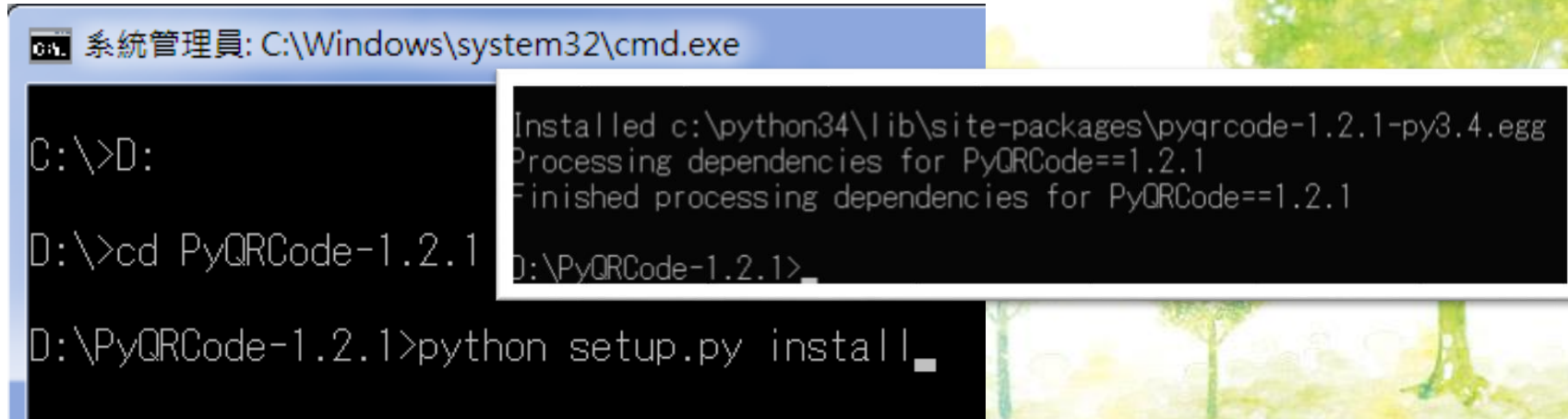
# PYTHON3 PACKAGES

- 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



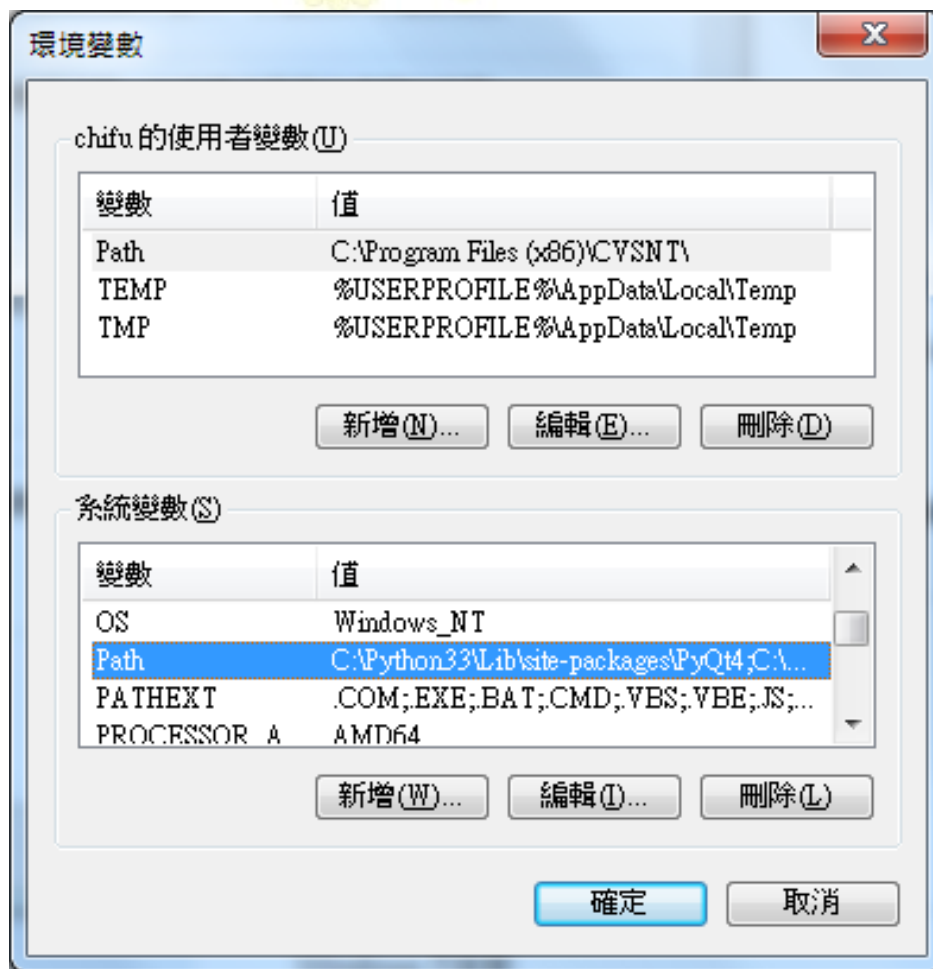


# 設定環境變數

```
C:\>pip
```

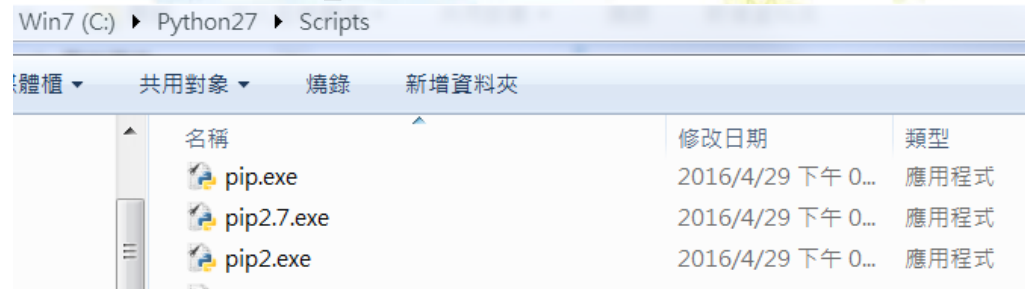
'pip' 不是內部或外部命令、可執行的程式或批次檔。

- 於系統變數中的Path
- 新增  
**C:\Python34\Scripts**
- 記得用「;」號分隔

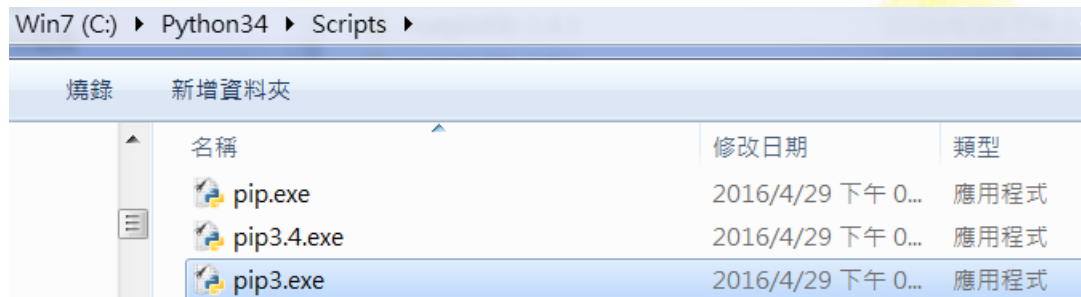


# 同時裝了PYTHON3和PYTHON2，怎麼用PIP？

- 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

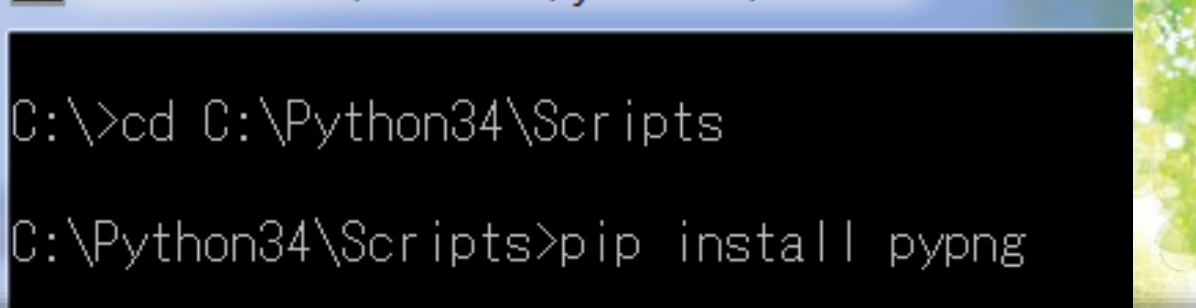


- 然後你把這兩個路徑填加到環境變數中，就可以分別用  
pip2 install ... (python2)  
pip3 install ... (python3)  
分別安裝庫了。



## 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 xxx`(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 4
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\63\1bc9ec080ed16c9b84b305c55716ea227026904af2d52f1
Successfully built pypng
```



# 小練習

## 練習一

- 試著下 `pip install pypng` 安裝套件
- 並完成輸出png 的 `qrcode`

## 練習二

- 試著讀取 `urls.txt` 檔案中的超連結
- 並將其輸出成Qrcode存成png檔
- 1.png, 2.png ...





# 延申閱讀

- 用 **pip** 安裝 \*.whl 檔
- 用 **distribute** 安裝套件
- 用 **easy install** 安裝套件

