

# Python 程式設計

林奇賦 [daky1983@gmail.com](mailto:daky1983@gmail.com)



# Outline

- ▶ 課程簡介
- ▶ Python介紹
- ▶ 環境安裝說明
- ▶ 變數與運算

# 課程簡介

- ▶ 台大系統訓練班263期
- ▶ 課程網站：<http://140.112.31.82/wordpress>
- ▶ 上課時間：(一)，(四) 19:00~22:00
- ▶ 給分方式：出席(30%)、作業(70%)



# Python 簡介

- ▶ Script Program Language 
- ▶ Object-Oriented Program Language 
- ▶ General-Purpose Program Language
- ▶ Easy to learn
- ▶ 誰在使用Python呢?
  - ▶ Google
  - ▶ 美國太空總署(NASA)
  - ▶ ...

# Python 簡介

- Google網站的搜尋系統
- Youtube視訊共享服務
- BitTorrent點對點檔案共享系統
- NSA的加密和智能分析
- iRobot開發商業機器人吸塵器
- NASA、Los Alamos、Fermilab、JPL的科學程式設計任務
- Industrial Light & Magic、Pixar製作電影動畫

# Python 介紹

- ▶ 軟體品質
  - ▶ 可讀性
  - ▶ 強制縮排
  - ▶ 物件導向
- ▶ 動態語言
  - ▶ 直譯式的語言 
  - ▶ 增加了使用上的彈性
  - ▶ 節省重新編譯的時間
- ▶ 強類型定義語言 
- ▶ 豐富的標準函式庫

# Python 介紹

- ▶ 豐富的標準函式庫
- ▶ 可移植
- ▶ 容易擴充和嵌入
  - ▶ Python本身非常容易被擴充
  - ▶ 負載量大的部份，用C語言來寫，然後用Python來引用，就可以加快速度
  - ▶ Python可以嵌在其它程式裡面，這樣的特性讓Python非常有彈性

# Python 介紹

- 創始人為吉多·范羅蘇姆 (Guido van Rossum)
- 打發聖誕節的無趣，決心開發一個新的指令碼解釋程式，作為ABC語言的一種繼承
- 以BBC喜劇Monty Python's Flying Circus命名
- Python 2.0於2000年10月16日發布，主要是實作了完整的垃圾回收 (Garbage Collection)，並且支援 Unicode
- 2008年12月3日發布Python 3.0。它不完全相容之前的 Python代碼。不過，很多新特性後來也被移植到舊的 Python 2.6/2.7版本



# Python的作者

- [http://en.wikipedia.org/wiki/Guido\\_van\\_Rossum](http://en.wikipedia.org/wiki/Guido_van_Rossum)



Guido van Rossum 吉多·范羅蘇姆

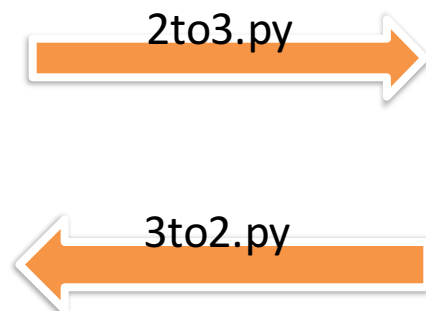
# Python2 或 Python3

- Python2.X

- ① 穩定版本
- ② **version2.7** 為最終版本
- ③ 較多的第三方函式庫
- ④ 仍有為數眾多的使用者

- Python3.X

- ① 現在進行式同時也是未來趨勢
- ② 活絡發展中
- ③ 對初學者較友善
- ④ 較少第三方函式庫

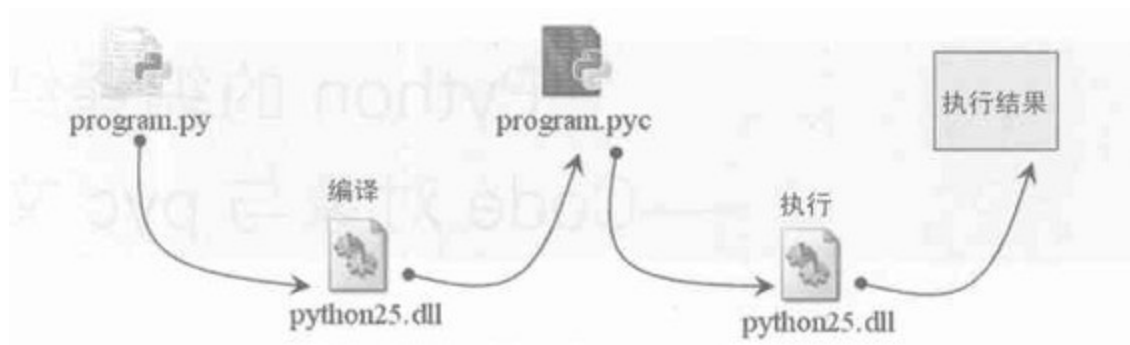
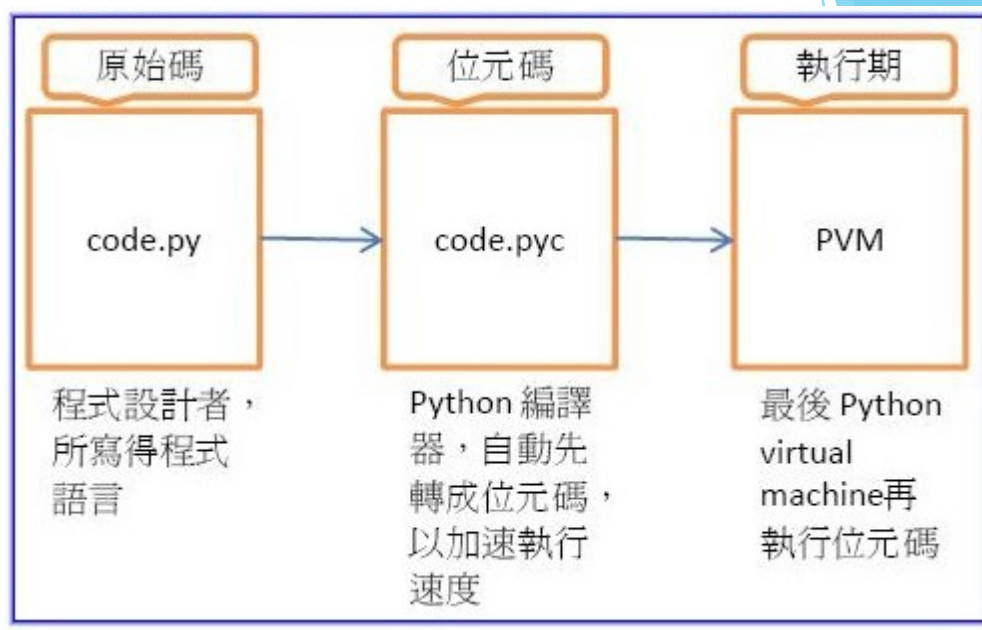


Brett Cannon, “Python 3.3: Trust Me, It's Better than 2.7”, PyCon 2013

[https://www.youtube.com/watch?v=f\\_6vDi7ywuA](https://www.youtube.com/watch?v=f_6vDi7ywuA)

# 簡介

- Python如何執行



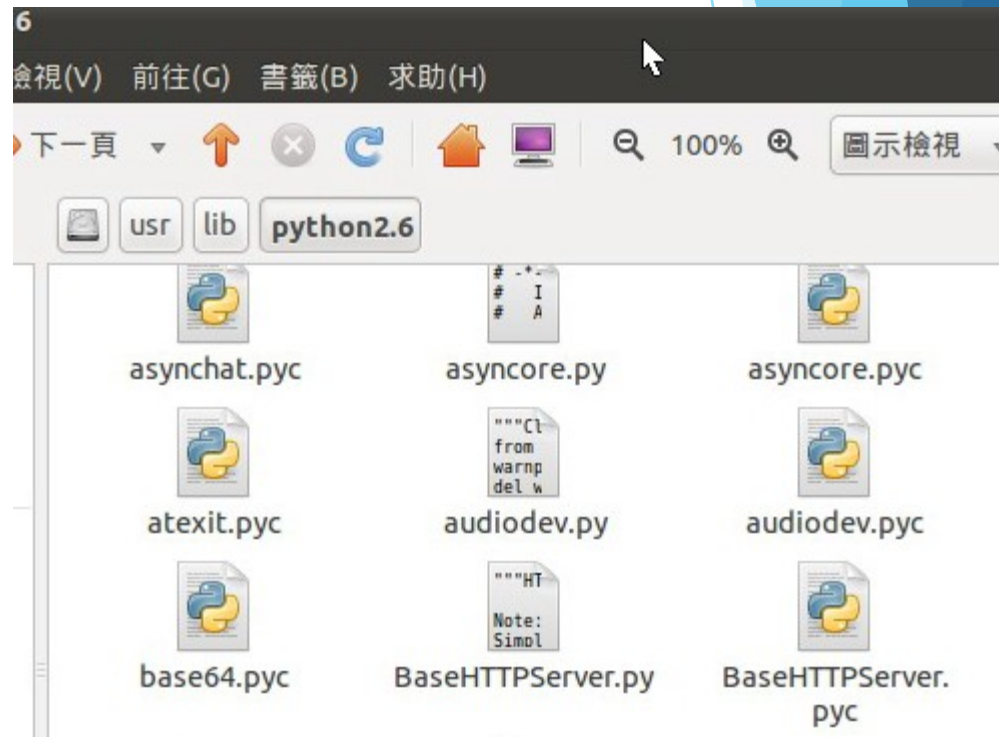
- 模組的存放位置

- Window

- /安裝Python的資料夾/Lib

- Linux

- /usr/lib/python2.x



# 簡介

- 安裝
  - Window
    - Python-3.x.msi
  - Linux
    - 系統本身
- 進入Python互動直譯器(for Linux)
  - Python 2.7以下版本
    - #python
  - Python 3.x
    - #python3

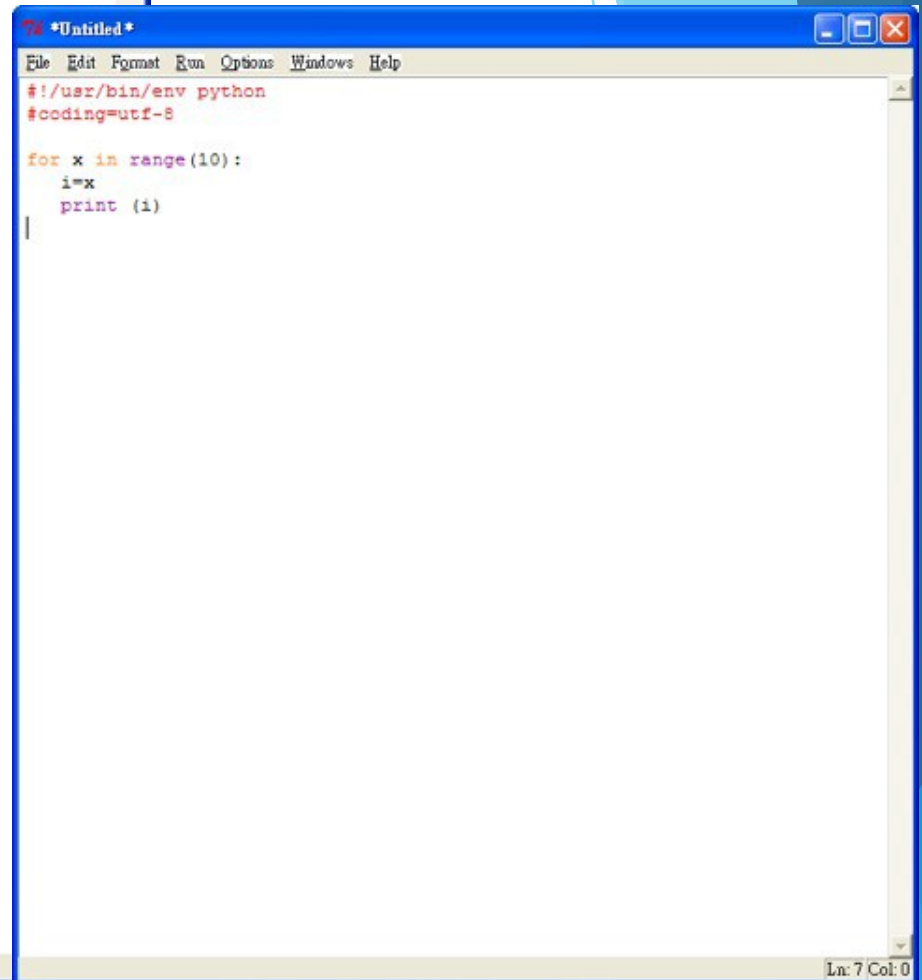
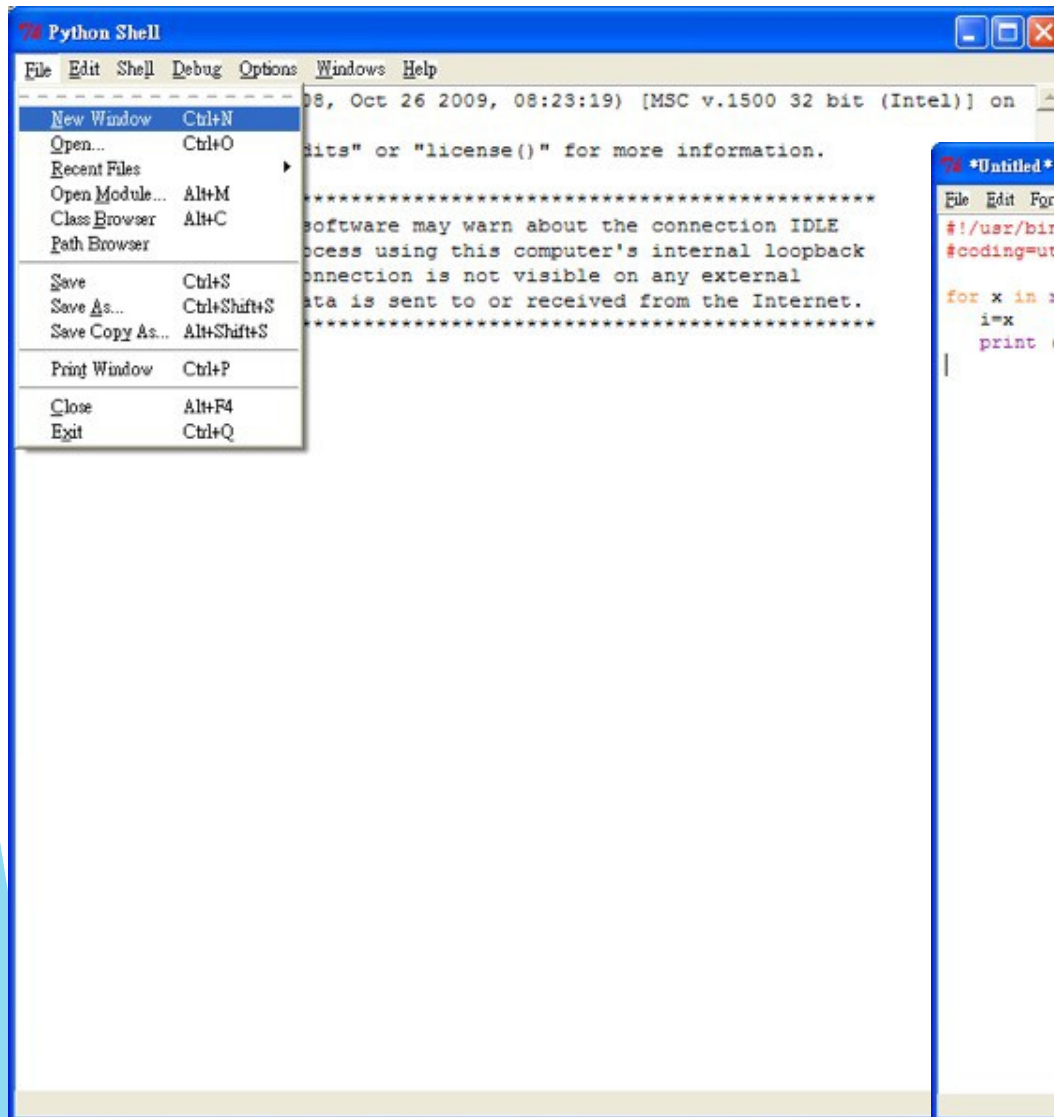
[http://140.112.31.82/python/01\\_install.pdf](http://140.112.31.82/python/01_install.pdf)

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.6.4 (r264:75708, Oct 26 2009, 08:23:19) [MSC v.1500 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.

*****
Personal firewall software may warn about the connection IDLE
makes to its subprocess using this computer's internal loopback
interface. This connection is not visible on any external
interface and no data is sent to or received from the Internet.
*****
```

```
IDLE 2.6.4
>>> from __future__ import print_function
>>> print
```

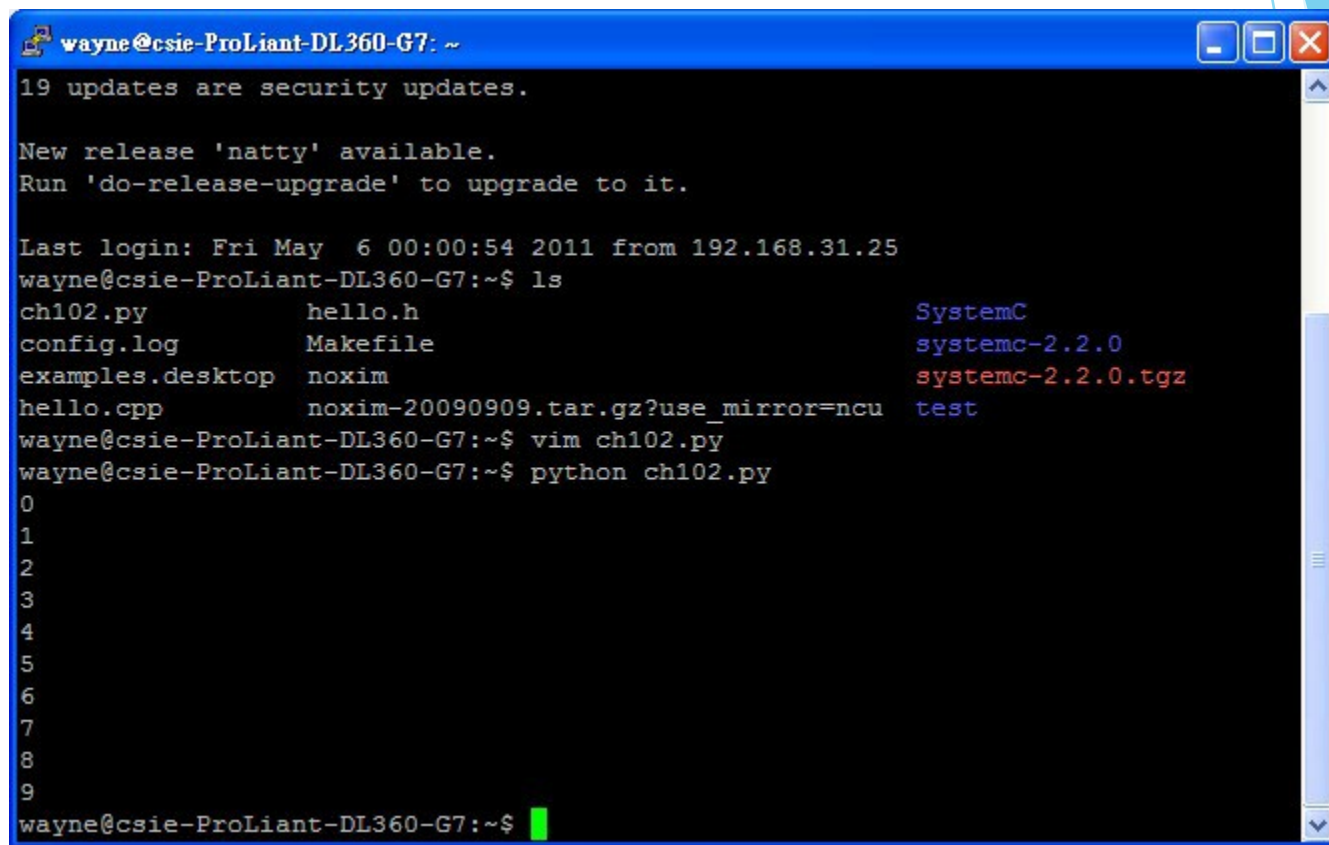
```
Python (command line)
Python 2.6.4 (r264:75708, Oct 26 2009, 08:23:19) [MSC v.1500 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print "ABC"
ABC
>>> ^Z_
```









# 簡介



```
wayne@csie-ProLiant-DL360-G7: ~  
19 updates are security updates.  
  
New release 'natty' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Fri May  6 00:00:54 2011 from 192.168.31.25  
wayne@csie-ProLiant-DL360-G7:~$ ls  
ch102.py          hello.h           SystemC  
config.log        Makefile          systemc-2.2.0  
examples.desktop  noxim             systemc-2.2.0.tgz  
hello.cpp         noxim-20090909.tar.gz?use_mirror=ncu  test  
wayne@csie-ProLiant-DL360-G7:~$ vim ch102.py  
wayne@csie-ProLiant-DL360-G7:~$ python ch102.py  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
wayne@csie-ProLiant-DL360-G7:~$
```

# 基本概念

- 語法特色

- 以冒號(:)做為敘述的開始
- 不必使用分號(;)做為結尾
- 井字號(#)做為註解符號，同行#字號後的任何字將被忽略 
- 使用tab鍵做為縮排區塊的依據 
- 不必指定變數型態 (runtime時才會進行binding)

# 變數與運算

- 變數的命名
  - 以英文字母a-z或A-Z或是\_為開頭
  - Ex: first-name                      false
  - Ex: first\_name                      right
  - Ex: fruit, Fruit
- Python關鍵字
- 刪除變數
  - del 變數

# 變數與運算

- 變數多重設定
  - Ex: orange, apple = 2, 5
- 告別許功蓋
  - Big5碼的衝碼問題
  - 採用unicode編碼

```
#include <iostream>
#include <time.h>
using namespace std;

int main()
{
    cout << "以偏蓋全" << endl;
    |
    system("PAUSE");
    return 0;
}
```

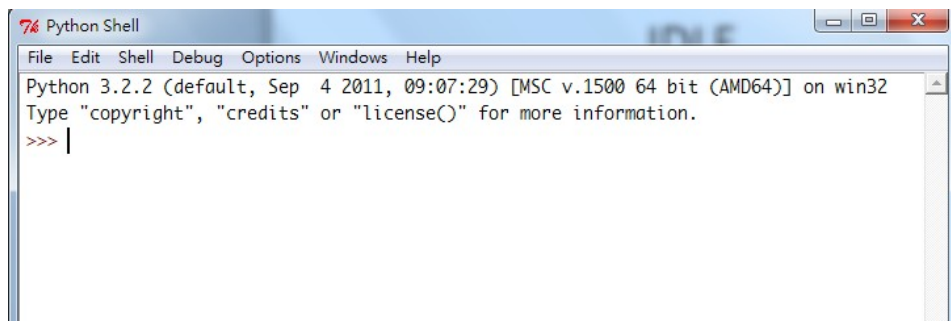


# 簡介

- `print()` 函數功能
  - `print("A","B","C","D")`
    - A B C D
  - `print("A","B","C","D",sep="")`
    - ABCD
  - `print("A","B","C","D",sep="|")`
    - A|B|C|D

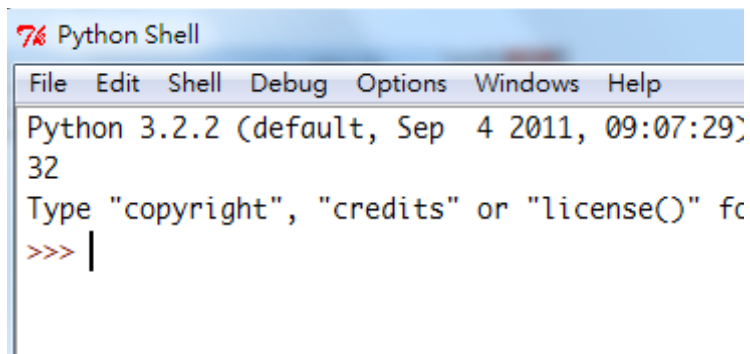
# 認識IDLE

- 小而實用的Python開發環境!
- 包含了實用的編輯器與Python直譯器(互動介面,Python Shell)
  - ① 自動縮排
  - ② 語法高亮
  - ③ 提供互動介面
  - ④ 容易學習與測試



# 認識IDLE

- 接下來介紹IDLE的各項功能



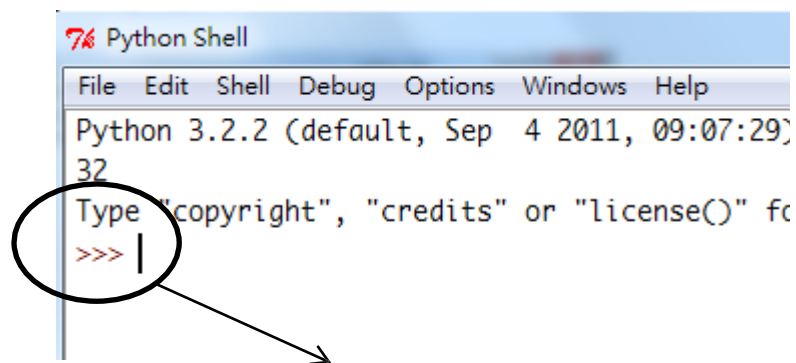
- ✓ File : 開檔, 關檔, 存檔, 開新視窗
- ✓ Edit : 複製, 貼上, 取代, 尋找, 復原
- ✓ Shell : 重啟shell
- ✓ Debug : 除錯的功能與工具
- ✓ Option : 可以設定IDLE
- ✓ Help : IDLE幫助與Python文件

# 認識IDLE

- 接著我們一步一步來看看IDLE的基本功能!
- 請跟著輸入並觀察結果(不太了解沒關係, 這邊只是體驗!)
- Part I. Shell

Step 1 : 輸入1+1

Step 2 : 輸入print("hello world!")



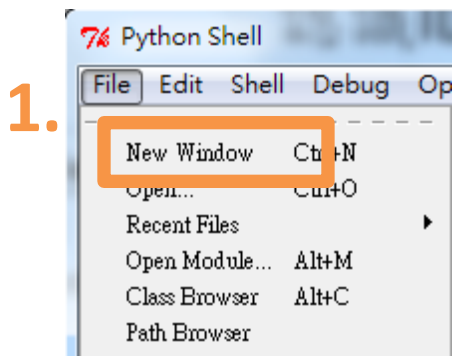
在Python Shell中, >>> 是一個輸入的提示字串  
我們所有要下給直譯器的命令都要打在>>>之後喔



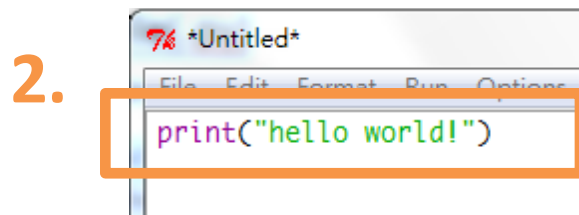
# 認識IDLE

- Part II. A New File(.py)

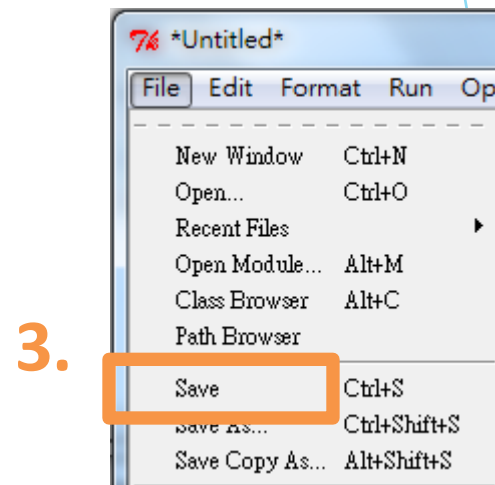
Step 1 : File > New Window



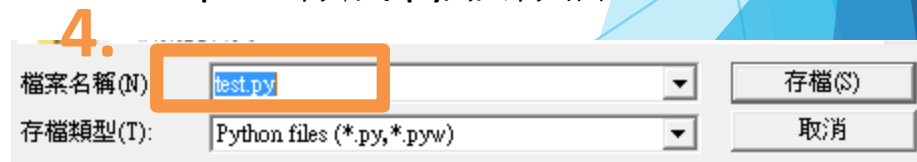
Step 2 : 輸入print("hello world!")



Step 3 : File > Save

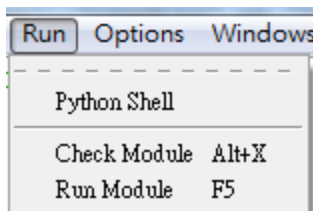


Step 4 : 存成.py, 按存檔



# 認識IDLE

- Step 5 : 按F5 或 Run > Run Module, 看看會出現什麼!



- 出現下面訊息代表成功了! 恭喜你, 踏出學習Python的第一步

```
Python 3.2.2 (default, Sep  4 2011, 09:07:29) [MSC v.1500 64 bit (AMD64)] on wi
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
hello world!
>>> |
```

# Python重要操作指令與快捷鍵

- 輸入終止符
  - Unix-Like : Ctrl+D
  - Windows : Ctrl+Z <Enter>
- IDLE熱鍵
  - Alt+p 上一個指令
  - Alt+n 下一個指令
  - Tab 指令補完
- Python in OS shell
  - Python -c "command" 執行一個python命令
  - Python -m SimpleHTTPServer run Python server

# 標記

- 直譯器利用標記 (token) 解析程式的功能，Python 中的標記有關鍵字 (keyword)、識別字 (identifier)、字面常數 (literal)、運算子 (operator) 等四類
- 關鍵字
- 識別字
- 字面常數
- 運算子

# 關鍵字

- 關鍵字為具有語法功能的保留字 (reserved word)，Python 的關鍵字，如以下列表

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

# 識別字

- 識別字為寫程式時依需求自行定義的名稱，包括變數 (variable)、函數 (function)、類別 (class) 等，皆為使用自行定義的識別字。除了關鍵字之外，Python 可用任何 Unicode 編碼的字元當作識別字。
- 習慣上識別字的命名仍是以英文字母大寫 A-Z (\u0041-\u005a)，小寫 a-z (\u0061-\u007a)，底線符號 (\_, \u005f) 與數字 0-9 (\u0030-\u0039) 為主。

# 字面常數

- 字面常數的意思就是字面上的意義，也就是說，1234就代表整數數值一千兩百三十四的意義，因此，所謂的字面常數就是直接寫進 Python 程式原始碼的數值，依資料型態分類有
  - 字串字面常數 (string literal)
  - 字節字面常數 (bytes literal)
  - 整數字面常數 (integer literal)
  - 浮點數字面常數 (floating-point literal)
  - 複數字面常數 (imaginary literal)

# 運算子

- Python 提供多樣、功能完整的運算子，如下列表

+	-	*	**	/	//	%
<<	>>	&		^	~	
<	>	<=	>=	==	!=	

- 分隔符號 (delimiter)

(	)	[	]	{	}
,	:	.	;	@	=
+=	-=	*=	/=	//=	%=
&=	=	^=	<<=	>>=	**=



# 變數(Variables)和 表示式(Expressions)

- 表示式

`3 + 5`

`3 + (5 * 4)`

`3 ** 2`

`'Hello' + 'World'`

- 變數指定

`a = 4 + 3`

`b = a * 4.5`

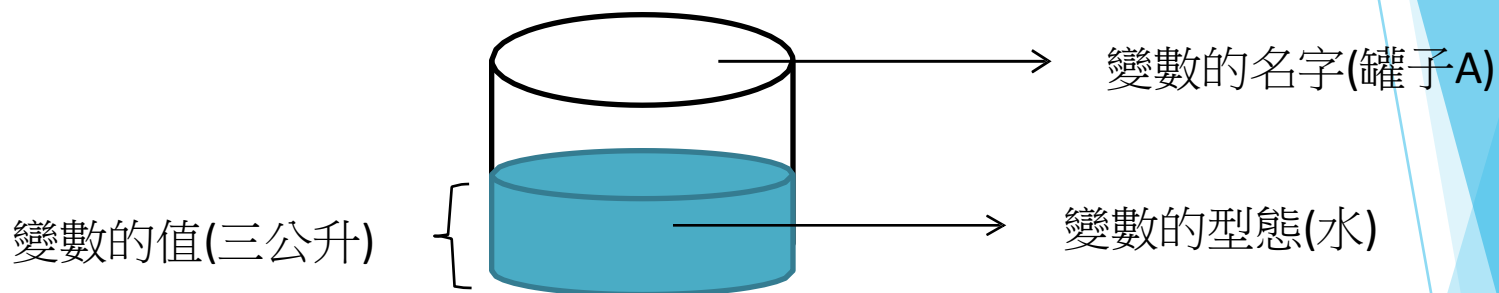
`c = (a+b)/2.5`

`a = "Hello World"`

- 型別是動態的，會根據指定時的物件來決定型別
- 變數單純只是物件的名稱，並不會和記憶體綁在一起。  
e.q. 和記憶體綁在一起的是物件，而不是物件名稱。

# 關於變數 1/3

- 為何需要變數？彈性與保存
- 變數好比容器



- 變數要如何命名？
  - 以底線或英文字母開頭字符
  - 以底線,英文字母和數字為後續字符
  - 不可與關鍵字 (保留字) 相同

## 關於變數 2/3

- 定義變數與初始化變數

Var = 70 (整數)

Var = 6.78 (浮點數)

Var = "alcom lab" (字串)

變數值同時可以看出變數型態



variable name = value

變數名稱必須符合規範

- Programmer撰寫Python時不必在意變數的size(容器的大小), 因為細節早被隱藏

# 關於變數 3/3

- Python的變數是可以更換型態的

```
>>> a=5  
>>> a="kkk"  
>>> a=7.890
```

- 這並非指Python沒有型態,而是同一個變數名稱可以任意指稱裝載任何型態的容器(物件參照)
- 在Python Shell中直接輸入變數名稱可看到變數的再現型態

# 賦值運算

- 賦值運算是最常被使用的運算
- 將運算的結果賦予(存到)一個變數, 將右邊運算的結果(右值)賦予左邊(左值)

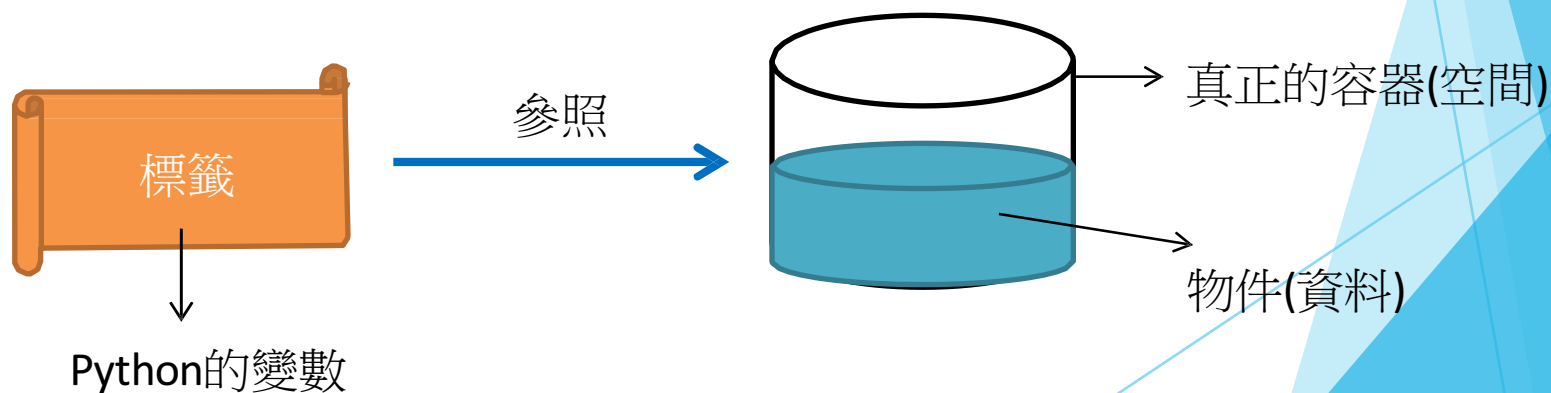
left value = right value



- 等號是賦值運算的運算子, 代表把右邊給予左邊, 跟一般在數學上的意義“相等”是完全不同的

# 物件參照 1/3

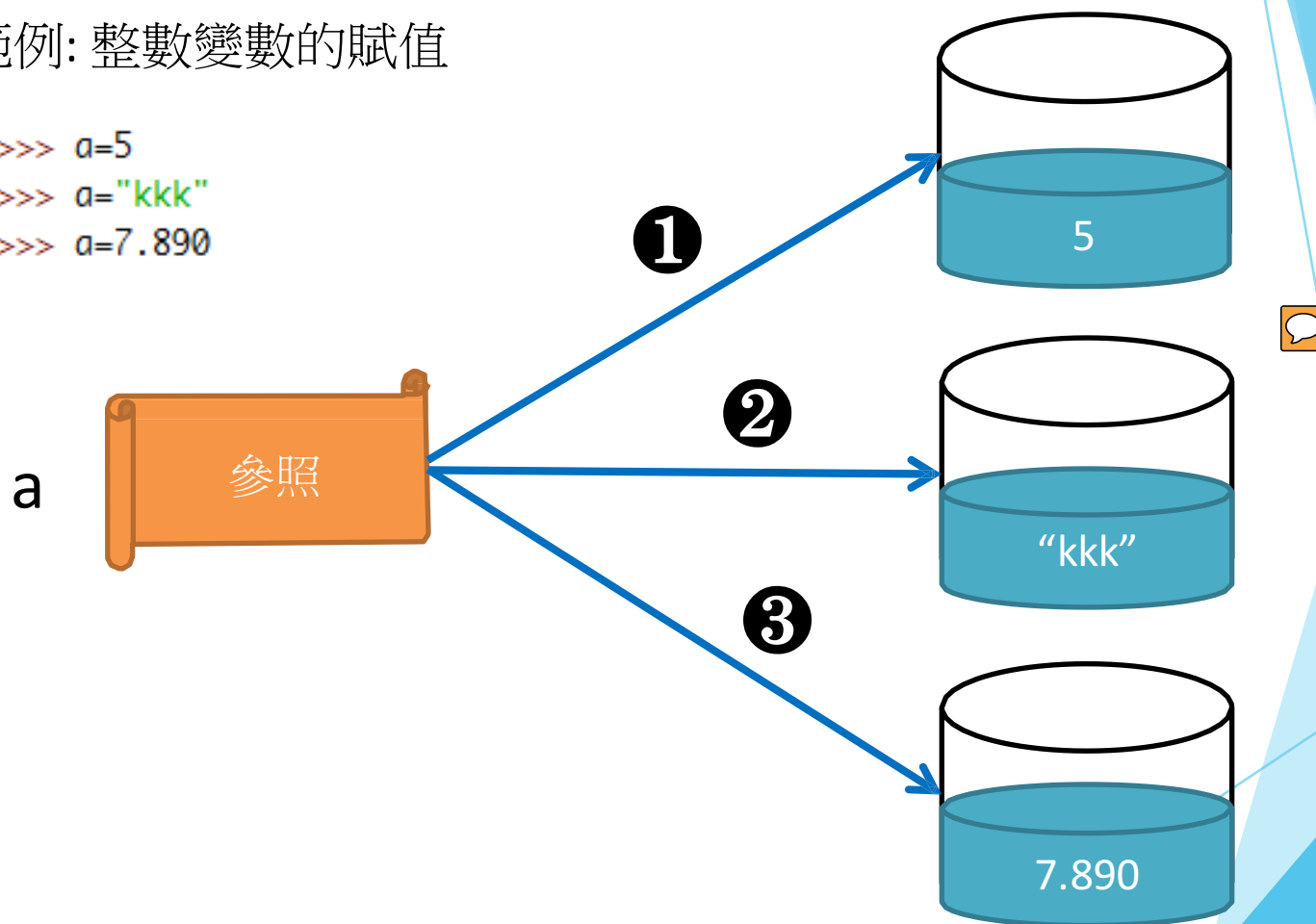
- Python在實作上採取了參照物件的作法
- 一個資料(物件)必須得有一個空間(容器)來保存它,而在許多程式語言中,變數就是容器的名字,但在Python中,變數只是一張標籤,此標籤指出了真正容器之所在,因而參照了該容器裡面保存的物件



## 物件參照 2/3

- 範例: 整數變數的賦值

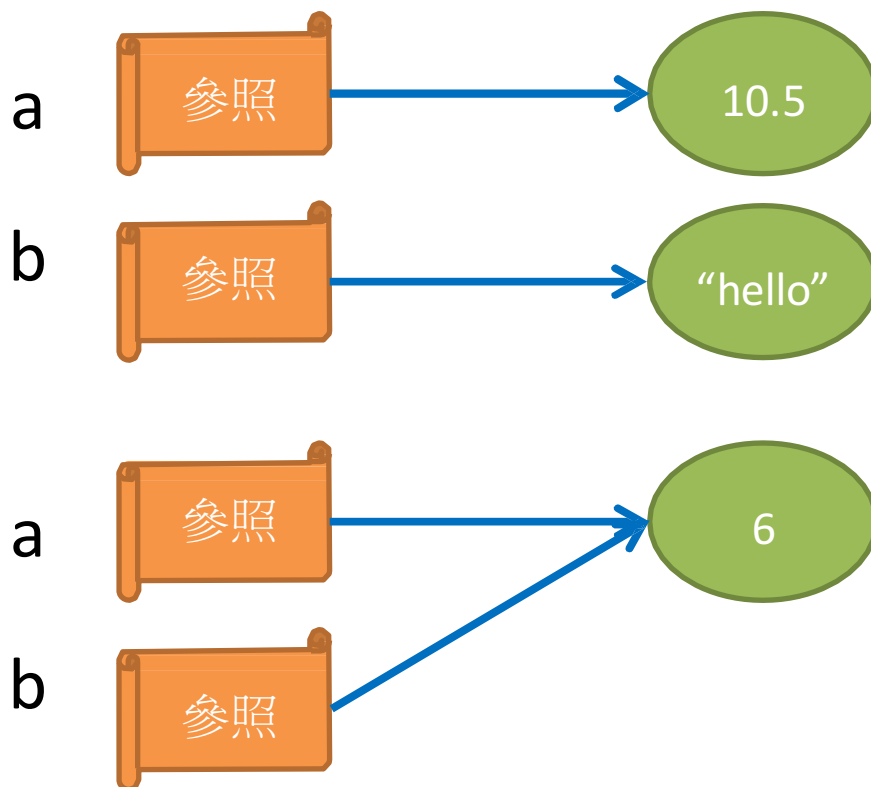
```
>>> a=5  
>>> a="kkk"  
>>> a=7.890
```



# 物件參照 3/3

- 範例:

```
>>> a="hello"  
>>> b=a  
>>> a=10.5  
>>> a  
10.5  
>>> b  
'hello'
```



**is** : 判斷兩個物件的**id**是否相同

**==** : 判斷兩個物件的**value**是否相同



# 變數與物件

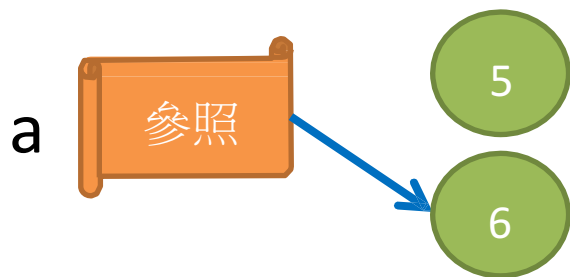
- Python 中所有東西都是物件 (object) ，這是說 Python 裡的資料 (data) 都是物件。凡是物件都有：
  - (id) 號碼
  - (type) 型態
  - (value) 數值
- 物件的值可以是可變的 (mutable) ，或是不可變的 (immutable) ，通常這是說複合資料型態 (compound data type) 的元素 (element) 是否可以替換，例如序對 (tuple) 及字串是不可變的，而串列 (list) 或字典 (dictionary) 是可變的。
- 當物件不再使用時，直譯器會自動垃圾收集 (garbage collection) ，釋放記憶體空間。

# 可變與不可變 1/3

- 資料型態有分成可變的(mutable)與不可變的(immutable)
- 指的是變數所參照的物件本身可否改變
- 不可變資料型態: 整數, 浮點數, 字串

```
>>> a=5
```

```
>>> a=6
```

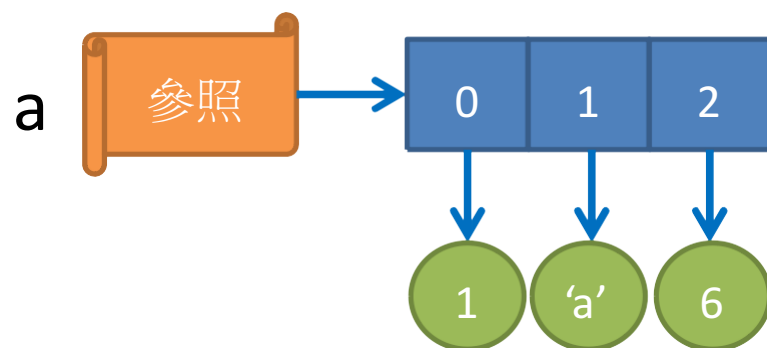


被自動回收

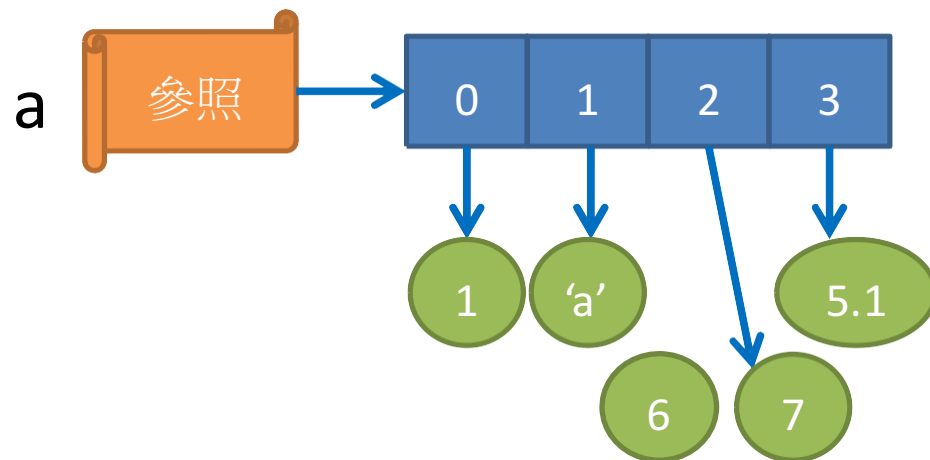
變數a原先參照的物件:整數5並沒有被改變  
而是產生了新的整數物件6 並且將變數a重新繫結到物件6

# 可變與不可變 2/3

- 可變資料型態: 串列, 字典




```
>>> a=[1, 'a', 6]
>>> a+= [5.1]
>>> a[2]=7
>>> a
[1, 'a', 7, 5.1]
```

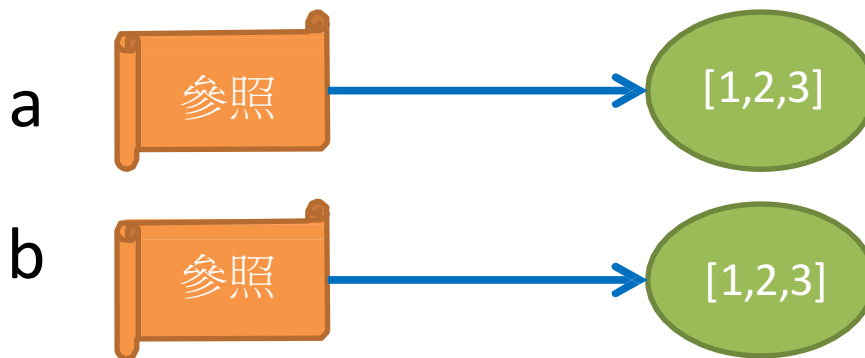


變數a原先參照的串列被改變了, 增加了一個元素5.1也改變了第三個元素7

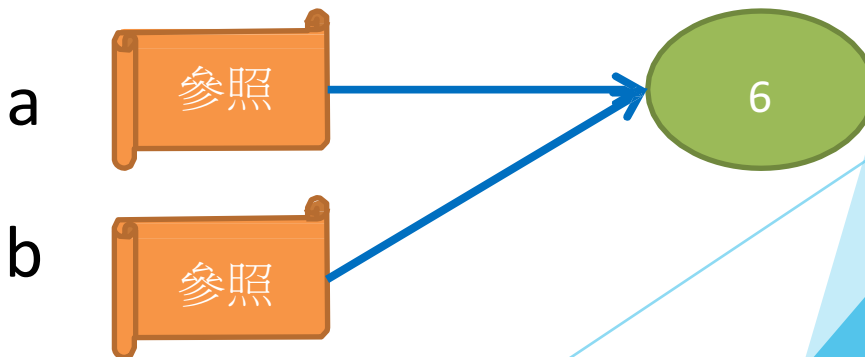
# 可變與不可變 3/3

- 可變的資料型態，變數指定時會產生新的 id
- 不可變的資料型態，變數指定時會對應相同的 id 

```
>>> a = [1,2,3]
>>> b = [1,2,3]
>>> a == b
True
>>> a is b
False
>>> id(a)
4314178696
>>> id(b)
4314180872
```



```
>>> a = 6
>>> b = 6
>>> a is b
True
>>> a == b
True
```



# 資料型態

- 整數 int
- 浮點數 float
- 複數 complex
- 字串 str
- 字節 bytes
- 字節陣列 bytearray
- 串列 list
- 序對 tuple
- 集合 set
- 字典 dict

# 數字型態

- ▶ 內建的數字型態 (numeric types) 共有三種
  - ▶ int整數
    - ▶ 100
  - ▶ float浮點數
    - ▶ 123.45
  - ▶ complex複數
    - ▶  $3+4j$

# 數學運算

- 基本四則運算:  $+$   $-$   $*$   $/$   $\%$
- 運算特性: 以複雜者為主且運算精確性有限

```
>>> 10/3
3.3333333333333335
>>> 50*0.5
25.0
>>> 10+3j-3j
(10+0j)
```

- Python所支援的特殊運算:  
次方:  $2^{**}10$   
整商除法:  $10//3$

# 複數運算

```
>>> a = 3 + 4j
>>> b = 2 - 5j
>>> c = a * b
>>> c
(26-7j)
>>> c.real
26.0
>>> c.imag
-7.0
```

c.real 實部  
c.imag 虛部



# 內建序列型態

- 內建的序列型態 (sequence types) 共有六種
  - `str`: 字串 (string) , 不可變 (immutable)
  - `bytes`: 字節 (byte) , 不可變 (immutable)
  - `bytearray`: 字節陣列 (byte array) , 可變 (mutable)
  - `list`: 串列 (list) , 可變 (mutable)
  - `tuple`: 序對 (tuple) , 不可變 (immutable)
  - `range`: 內建函數 `range()` 回傳的物件 (object) , 常用於 `for` 迴圈 (for loop)

# 字串 (String)

- 字串的特徵為引號, 包含單引號, 雙引號和三引號, 再現時預設以單引號表示
- 使用索引位置可以存取元素
- 字串的元素是**有序的**(誰前誰後有關係)
- 字串頭尾的引號必須相同對稱

- **Strings (字串)**

```
a = 'Hello'           # Single quotes
b = "World"           # Double quotes
c = "Bob said 'hey there.'" # A mix of both
d = """A triple quoted string
can span multiple lines like this"""
e = """Also works for double quotes"""
```

# 序列型態(sequence types)

## 可進行以下的計算

計算	描述
<code>x in s</code>	判斷 <code>x</code> 是否在 <code>s</code> 中
<code>x not in s</code>	判斷 <code>x</code> 是否不在 <code>s</code> 中
<code>s + t</code>	連接 <code>s</code> 及 <code>t</code>
<code>s * n, n * s</code>	將 <code>s</code> 重複 <code>n</code> 次連接 <code>s</code> 本身
<code>s[i]</code>	取得索引值 <code>i</code> 的元素
<code>s[i:j]</code>	取得索引值 <code>i</code> 到 <code>j</code> 的子序列 
<code>s[i:j:k]</code>	取得索引值 <code>i</code> 到 <code>j</code> ，間隔 <code>k</code> 的子序列
<code>len(s)</code>	回傳 <code>s</code> 的元素個數
<code>min(s)</code>	回傳 <code>s</code> 中的最小值
<code>max(s)</code>	回傳 <code>s</code> 中的最大值
<code>s.index(i)</code>	取得 <code>s</code> 中第一次出現 <code>i</code> 的索引值
<code>s.count(i)</code>	累計 <code>s</code> 中 <code>i</code> 出現的個數

# 串列 list

- 是最實用也最常用的群集
- 可以收集不同資料型態的元素
- list中的元素允許是list
- list的元素是**有序的**(誰前誰後**有關係**)
- list的特徵是**中括號[]**, 使用**索引位置**可以存取元素

```
>>> list1=[1,2+0j,"three",4.0] → 不同型態的資料可以被收集在同一個清單中
>>> list1[0]
1
>>> list1[2]
'three'
>>> list1[-1]
4.0
```

顯示list1的第一個元素  
(在電腦領域裡計數由零開始)

# 串列(list)

- 任意物件的串列

```
a = [2, 3, 4]
```

```
b = [2, 7, 3.5, "Hello"]
```

```
c = []
```

```
d = [2, [a, b]]
```

```
e = a + b
```

```
# A list of integer
```

```
# A mixed list
```

```
# An empty list
```

```
# A list containing a list
```

```
# Join two lists
```

- 串列的操作

```
x = a[1]
```

```
y = b[1:3]
```

```
z = d[1][0][2]
```

```
b[0] = 42
```

```
# Get 2nd element (0 is first)
```

```
# Return a sub-list
```

```
# Nested lists
```

```
# Change an element
```

# 串列(list)型態有以下的方法

方法	描述
<code>list.append(x)</code>	將 x 附加到 list 的最後
<code>list.extend(x)</code>	將 x 中的元素附加到 list 的最後
<code>list.count(x)</code>	計算 list 中 x 出現的次數
<code>list.index(x[, i[, j]])</code>	回傳 x 在 list 最小的索引值
<code>list.insert(i, x)</code>	將 x 插入 list 索引值 i 的地方
<code>list.pop([i])</code>	取出 list 中索引值為 i 的元素，預設是最後一個
<code>list.remove(x)</code>	移除 list 中第一個 x 元素
<code>list.reverse()</code>	倒轉 list 中元素的順序
<code>list.sort([key[, reverse]])</code>	排序 list 中的元素

# 序對(tuple)

- 可以收集不同資料型態的元素
- tuple中的元素允許是tuple
- tuple的元素是**有序的**(誰前誰後**有關係**)
- tuple的特徵是**小括號()**, 使用**索引位置**可以存取元素
- 與**list**類似，最大的不同tuple是一種**唯讀且不可變更**的資料結構，不可取代tuple中的任意一個元素，因為它是唯讀不可變更的

# 序對(tuple)

- tuple

```
f = (2,3,4,5)
```

```
# A tuple of integers
```

```
g = ()
```

```
# An empty tuple
```

```
h = (2, [3,4], (10,11,12))
```

```
# A tuple containing mixed  
objects
```

- tuple的操作

```
x = f[1]
```

```
# Element access. x = 3
```

```
y = f[1:3]
```

```
# Slices. y = (3,4)
```

```
z = h[1][1]
```

```
# Nesting. z = 4
```



# 內建集合型態 (set)

- 內建的集合型態 (set types) 共有兩種
  - `set`: 集合，可變 (mutable)
  - `frozenset`: 原封集合，建立後不可新增或刪除元素 (element)，因此為不可變 (immutable)
- 集合型態的字面常數使用**大括弧{}**，其物件屬於複合資料型態 (compound data type)，也就是說單一集合型態物件可以包含多個元素，但沒有重複的元素。

# 內建集合型態 (set)

- set 的元素是無序的(誰前誰後沒關係)

```
>>> s1 = {1,1,1,2,2,3,3,4,5}
```

```
>>> s2 = {5,3,1,2,4}
```

```
>>> s1 == s2
```

```
True
```

```
>>> s1
```

```
{1, 2, 3, 4, 5}
```

```
>>> s2
```

```
{1, 2, 3, 4, 5}
```

```
>>> s1 is s2
```

```
False
```

```
>>> id(s1)
```

```
4367107912
```

```
>>> id(s2)
```

```
4367109928
```

沒有重複的元素，  
而且設定時的順序也不影響

set為可變型態，  
所以變數指定時會  
產生新的 id

# 集合型態的物件可進行之運算

計算	描述
$x \text{ in } s$	判斷 $x$ 是否在 $s$ 中
$x \text{ not in } s$	判斷 $x$ 是否不在 $s$ 中
$s1 \& s2$	且運算，取得 $s1$ 與 $s2$ 的交集
$s1   s2$	或運算，取得 $s1$ 與 $s2$ 的聯集
$s1 \wedge s2$	對稱差運算，取得 $s1$ 與 $s2$ 的對稱差集
$s1 - s2$	差運算，取得 $s1$ 與 $s2$ 的差集
$s1 < s2$	判斷 $s1$ 是否為 $s2$ 的真子集
$s1 \leq s2$	判斷 $s1$ 是否為 $s2$ 的子集
$s1 > s2$	判斷 $s2$ 是否為 $s1$ 的真子集
$s1 \geq s2$	判斷 $s2$ 是否為 $s1$ 的子集
$\text{len}(s)$	回傳 $s$ 的元素個數
$\text{min}(s)$	回傳 $s$ 中的最小值， $s$ 中的元素必須是相同型態
$\text{max}(s)$	回傳 $s$ 中的最大值， $s$ 中的元素必須是相同型態

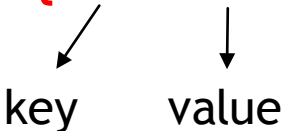
# 集合型態物件相對應的方法

方法	描述
<code>s1.intersection(s2)</code>	等於 $s1 \cap s2$
<code>s1.union(s2)</code>	等於 $s1 \cup s2$
<code>s1.symmetric_difference(s2)</code>	等於 $s1 \oplus s2$
<code>s1.difference(s2)</code>	等於 $s1 - s2$
<code>s1.issubset(s2)</code>	等於 $s1 \subseteq s2$
<code>s1.issuperset(s2)</code>	等於 $s1 \supseteq s2$
<code>s1.isdisjoint(s2)</code>	判斷 $s1$ 與 $s2$ 是否無交集，若無交集，回傳 True
<code>s.copy()</code>	回傳 $s$ 的拷貝

- 由於 `set` 型態是可變的，因此有額外兩個新增與刪除元素的方法

方法	描述
<code>s.add(e)</code>	增加 $e$ 為 $s$ 的元素
<code>s.remove(e)</code>	從 $s$ 中刪除元素 $e$

# 內建字典型態 (dict)

- 建立字典變數可利用**大括弧{ }**，裡頭以 **key:value** 為配對的資料項目，若有多筆資料再以逗號區隔開
- 例如 `d1 = { "a":100, "b":200 }` 共兩筆資料  

- 與list, tuple不同，不以**索引位置**存取元素，改以**key**當作索引存取元素

## 內建字典型態 (dict)

- 使用字典須注意，key 必須是不可變的 (immutable) 資料型態，如數字、字串 (string) 等，value 沒有限制
- dict 的元素是無序的(誰前誰後沒關係)

# 字典物件可進行的運算

計算	描述
d[key]	從 d 中取得 key 的 value
d[key] = value	將 d 的 key 指定為 value
del d[key]	刪除 d 中 key 所指定的 value
key in d	判斷 key 是否在 d 中
key not in d	判斷 key 是否不在 d 中
iter(d)	回傳由 d 的 key 建立的迭代器
len(d)	回傳 d 的配對資料個數

# 字典物件的方法 (method)

方法	描述
<code>dict.clear()</code>	清空 dict 的所有配對資料
<code>dict.copy()</code>	回傳 dict 的拷貝
<code>classmethod dict.fromkeys(seq[, value])</code>	由 seq 中的元素構成 key，每個 key 都給相同的 value 值
<code>dict.get(key[, default])</code>	從 dict 中取得 key 的 value，若無此 key 則回傳 default，default 預設為 None
<code>dict.items()</code>	回傳 dict_items 物件，使 key:value 儲存為序對，然後依序儲存在 dict_items 物件中
<code>dict.keys()</code>	回傳 dict_items 物件，使 key 依序儲存在 dict_items 物件中
<code>dict.pop(key[, default])</code>	將 key 的 value 從 dict 移除，若無此 key，回傳 default
<code>dict.popitem()</code>	從 dict 移除任意一組 key:value
<code>dict.setdefault(key[, default])</code>	如果 key 在 dict 中，回傳 value 值，反之，將 key:default 加入 dict 之中
<code>dict.update([other])</code>	將 dict 以 other 更新
<code>dict.values()</code>	回傳 dict_items 物件，使 value 依序儲存在 dict_items 物件中



# 型態轉換

運算適中的自動型態轉換是由儲存範圍小的轉換到儲存範圍大的，若是相反過來，浮點數轉換為整數，可以利用內建函數 (function) `int()` 進行轉換

```
a = 1
```

```
b = 2
```

```
c = 3.6
```




```
d = 3.6
```

```
print(int(a + c), int(a + d))
```

```
print(float(a + b))
```

```
print(complex(a + b))
```

# 動動腦

- ▶ `num = '1000'` #str字串型態
  - ▶ 若需要除以2運算後得到數字500的話怎麼辦？
- ▶ `nums = [1,1,1,2,2,3,4,5]` #list串列型態
  - ▶ 若需要將重複的元素去除的話怎麼辦？
- ▶ `nums = (10,5,7,1,6,2)` #tuple序對型態
  - ▶ tuple並不提供排序的方法，若要排序(由小到大)該怎辦？
- ▶ 適時的轉換型態可以解決非常多的問題

# 布林運算

- ▶ 有三種布林運算 and, or, not

運算	範例	結果
or	$2==3$ or $3 < 7$	True
and	$2==3$ or $3 < 7$	False
not	not $3 < 7$	False

# 比較運算子

運算符號	描述
<	小於
<=	小於或等於
>	大於
>=	大於或等於
==	比較值是否相等
!=	比較值是否不相等
is	比較是否同一個物件(id)
is not	比較是否不同個物件(id)

# Python的標準模組函式庫

Python本身就包含了大量的模組提供使用

- String processing
- Operating system interfaces
- Networking
- Threads
- GUI
- Database
- Language services
- Security



使用模組

```
>>> import math
>>> math.log(math.e)
1.0
```

# Python 2.x與Python 3.x的差異

- print從陳述式(statement)改為函數(function)
  - from \_\_future\_\_ import print\_function
  - print([object, ...][, sep=' '][, end='n'][, file=sys.stdout])

```
Python 2.7.6 (default, Sep  9 2014, 15:04:36)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.39)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> print "hello world"
hello world
>>> |
```

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 23 2015, 02:52:03)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> print("hello world")
hello world
>>> |
```

# Homework1

- ▶ 作業格式
  - ▶ 檔名：py2XX\_中文姓名\_hw1.py
- ▶ 繳交方式
  - ▶ 上傳到以下網址
  - ▶ <https://goo.gl/zQT47x>

# Homework1

## ▶ 作業要求

### ▶ Q1. 使用set型別完成下列問題: 本班期末考試

- ▶ 數學及格的有: Tom, John, Mary, Jimmy, Sunny, Amy
- ▶ 英文及格的有: John, Mary , Tony , Bob , Pony, Tom , Alice
- ▶ 分別印出數學及格但英文不及格的名單，數學不及格但英文及格的名單，兩科都及格的名單
- ▶ 最後印出全班總共有幾個同學

### ▶ Q2. 使用dict，list 型別完成下列問題:

- ▶ Tom 作業成績為 80, 100, 90, 95，John 作業成績為 100,93,75,80
- ▶ 請以dict 型別存放兩個同學的資料  
key:名字，value:分數列表(list)
- ▶ 請分別算出兩位同學的平均分數並且印出



# 參考

- ▶ 官方網站

- ▶ <https://www.python.org/>

- ▶ Python code 視覺化網站

- ▶ <http://www.pythontutor.com/>