

# Python程式設計

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

# Outline

- 課程簡介
- python介紹
- 變數與運算

# 課程簡介

- 台大系統訓練班246期
- 課程網站：<http://140.112.31.82/wordpress>
- 上課時間：(一)(四) 19:00~21:30
- 級分方式：出席(30%)、作業(70%)

# 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)
  - ...
- [How to Become a Hacker] 一文中推薦使用

# 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**
  - 強制數據類型定義的語言
- **豐富的標準函式庫**

# 簡介

- 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和其他語言比較
  - Tcl
    - Tcl沒有複雜的資料結構
    - Python更支援大型程式設計與應用較大系統的開發
  - Perl
    - Perl容易寫
    - Python容易讀
  - Java
    - 比Java更簡單以及更易於使用
    - Java承繼了系統語言的複雜度和語法
  - Visual Basic
    - 比Visual Basic更為跨平台

# 簡介

- Python的使用
  - 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

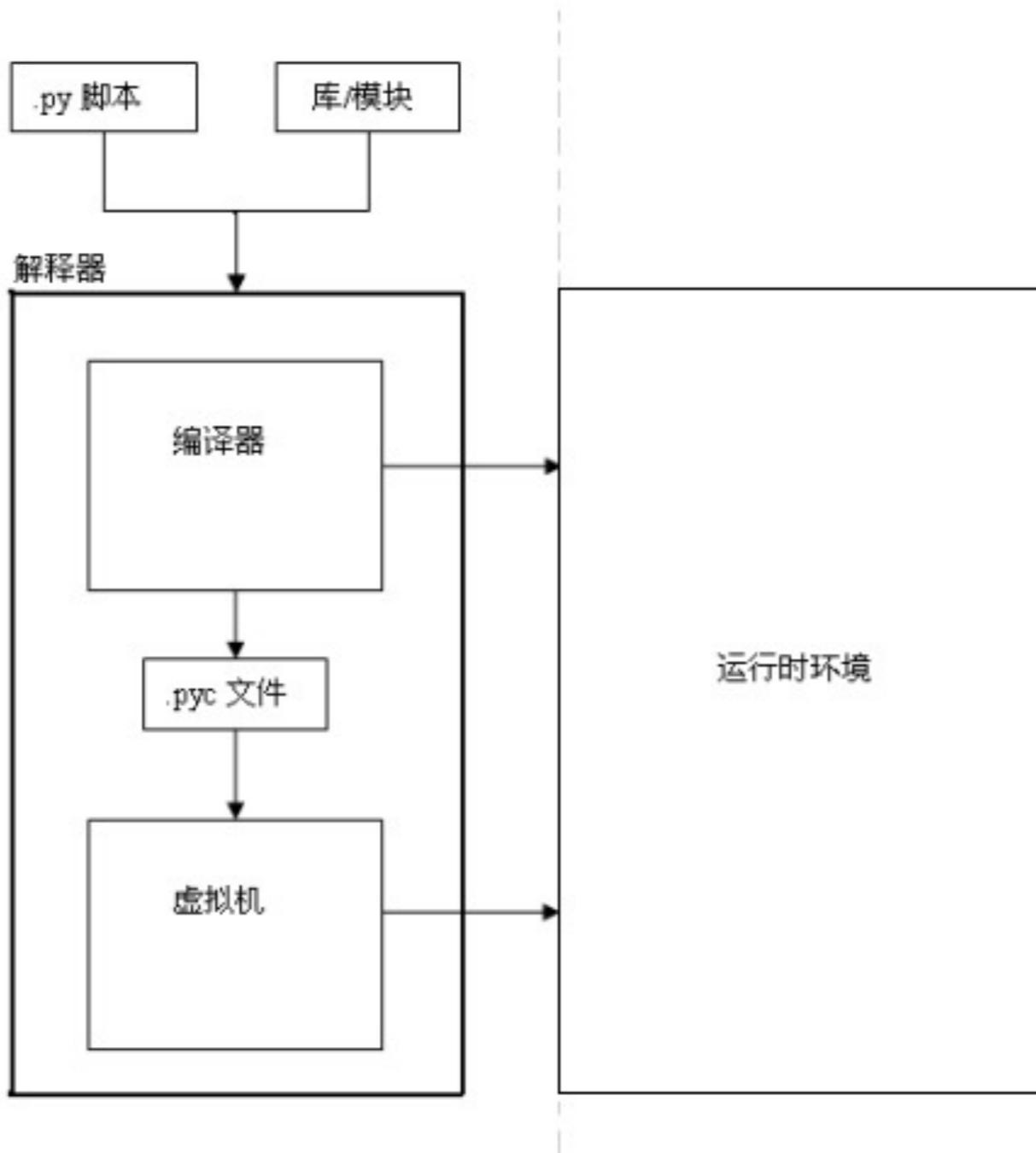
# 簡介

- Python的功能
  - 系統程式設計
  - GUIs
    - 搭配Tk GUI API標準物件導向介面
  - 網際網路描述機制
    - 以客戶和伺服器模式進行各種網路任務
  - 元件整合
    - 透過C和C++系統予以擴充和內嵌以描述其他系統和元件的行為
  - 資料庫程式設計
  - 迅速建構原型
  - 數值和科學程式設計
  - 遊戲開發、影像、AI、XML、機器人

# 簡介

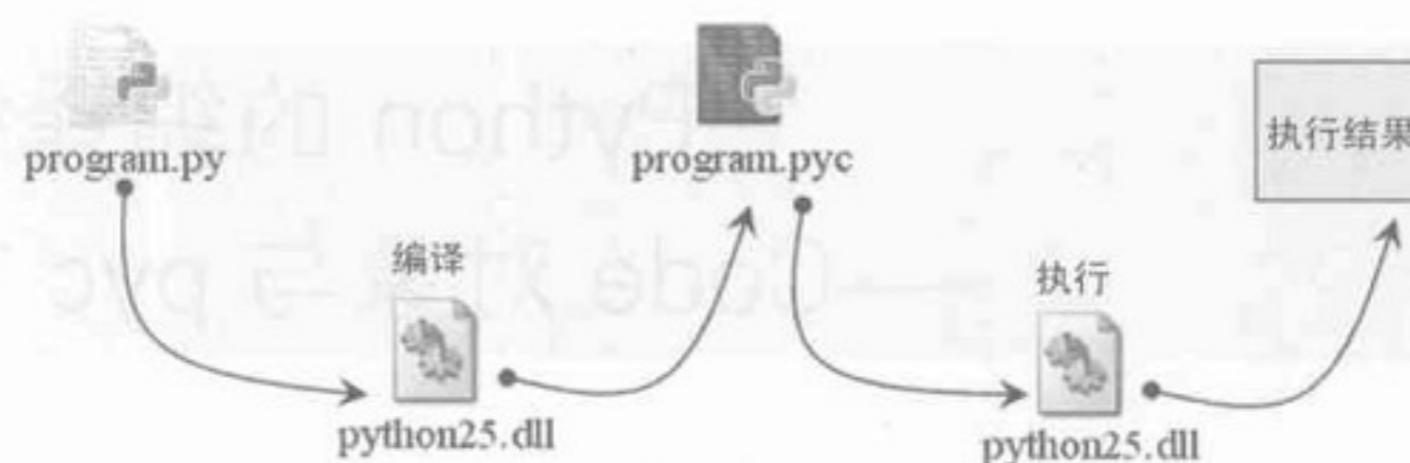
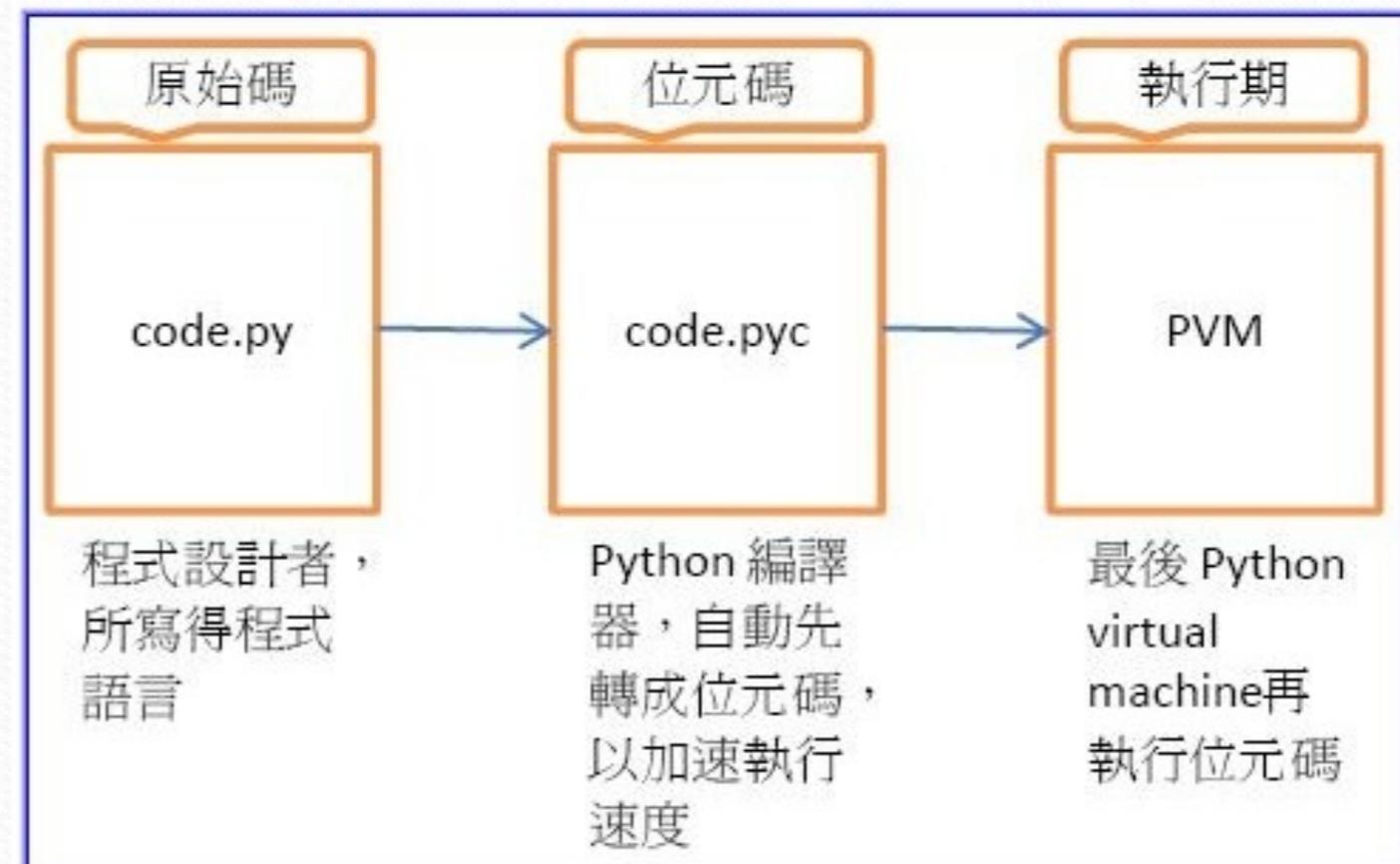
- Python的歷史
  - 創始人為吉多·范羅蘇姆（Guido van Rossum）
  - 打發聖誕節的無趣，決心開發一個新的指令碼解釋程式，作為ABC語言的一種繼承
  - 以BBC喜劇Monty Python's Flying Circus命名
  - Python 2.0於2000年10月16日發布，主要是實作了完整的垃圾回收（Garbage Collection），並且支援Unicode
  - 2008年12月3日發布Python 3.0。它不完全相容之前的Python代碼。不過，很多新特性後來也被移植到舊的Python 2.6/2.7版本

# Python整體架構



# 簡介

- Python如何執行



# 簡介

- 儲存Python程式的宣告
  - Linux使用者調用直譯程式方式
    - `#!/usr/bin/python`
  - Unix-like的路徑會在底下位置
    - `#!/usr/local/bin/python`
- Python 3.0要在宣告的尾端加上版本
- 完整的宣告方式
  - 宣告路徑與程式編碼
  - `#!/usr/bin/env python` 第一行
  - `# -*- coding: utf-8 -*-` 第二行
  - `(#coding=utf-8 )`

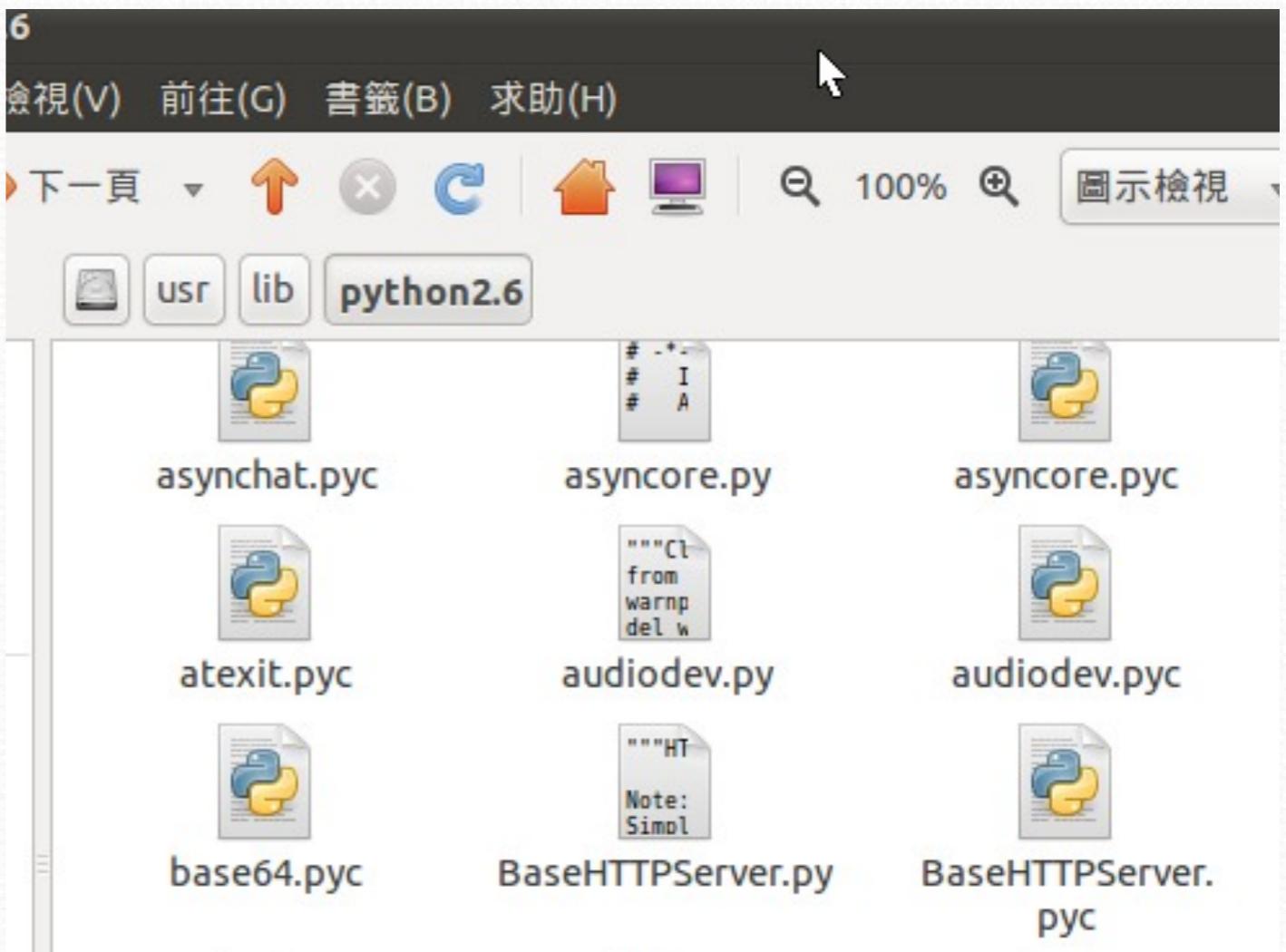
- 模組的存放位置

- Window

- /安裝Python的資料夾/Lib

- Linux

- /usr/lib/python2.6



# 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 2.6以下版本
    - #python
  - Python 3.0
    - #python3.0

# 簡介

- 結束Python(in command line)
  - Linux
    - Ctrl+D
  - Windows
    - Ctrl+Z
  - 通用
    - Import sys; sys.exit()

\*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_
```

The image shows two windows from the Python IDE (IDLE). The left window is titled "Python Shell" and displays the Python startup message:

```
08, Oct 26 2009, 08:23:19) [MSC v.1500 32 bit (Intel)] on win32  
Copyright (c) 2009, Python Software Foundation, Inc.  
All Rights Reserved.  
This software may or may not be free; see the license for details.  
For more information, see "http://www.python.org/".  

```

The right window is titled "\*Untitled\*" and contains the following Python code:

```
#!/usr/bin/env python  
#coding=utf-8  
  
for x in range(10):  
    i=x  
    print (i)
```

# 簡介

- 執行指令
  - # python 檔名.py
- 將Python程式改為可執行檔

- # chmod 755 檔名.py
  - 或# chmod +x 檔名.py
  - # /your/path/ 檔名.py(直接執行)

The image shows a Linux desktop environment with two terminal windows open. The top-left terminal window displays a standard Ubuntu 10.10 login screen with system information and upgrade notifications. The bottom-right terminal window shows a vim editor session for a Python script named 'ch102.py'.

**Terminal 1 (Top Left):**

```
wayne@csie-ProLiant-DL360-G7: ~
login as: wayne
wayne@163.25.101.188's password:
Linux csie-ProLiant-DL360-G7 2.6.35-22-generic #35-Ubuntu SMP Sat Oct 16 20:36:4
8 UTC 2010 i686 GNU/Linux
Ubuntu 10.10

Welcome to Ubuntu!
 * Documentation: https://help.ubuntu.com/

23 packages can be updated.
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
examples.desktop  noxim          systemc-2
hello.cpp      noxim-20090909.tar.gz?use_mirror=ncu test
wayne@csie-ProLiant-DL360-G7:~$ vim ch102.py
wayne@csie-ProLiant-DL360-G7:~$
```

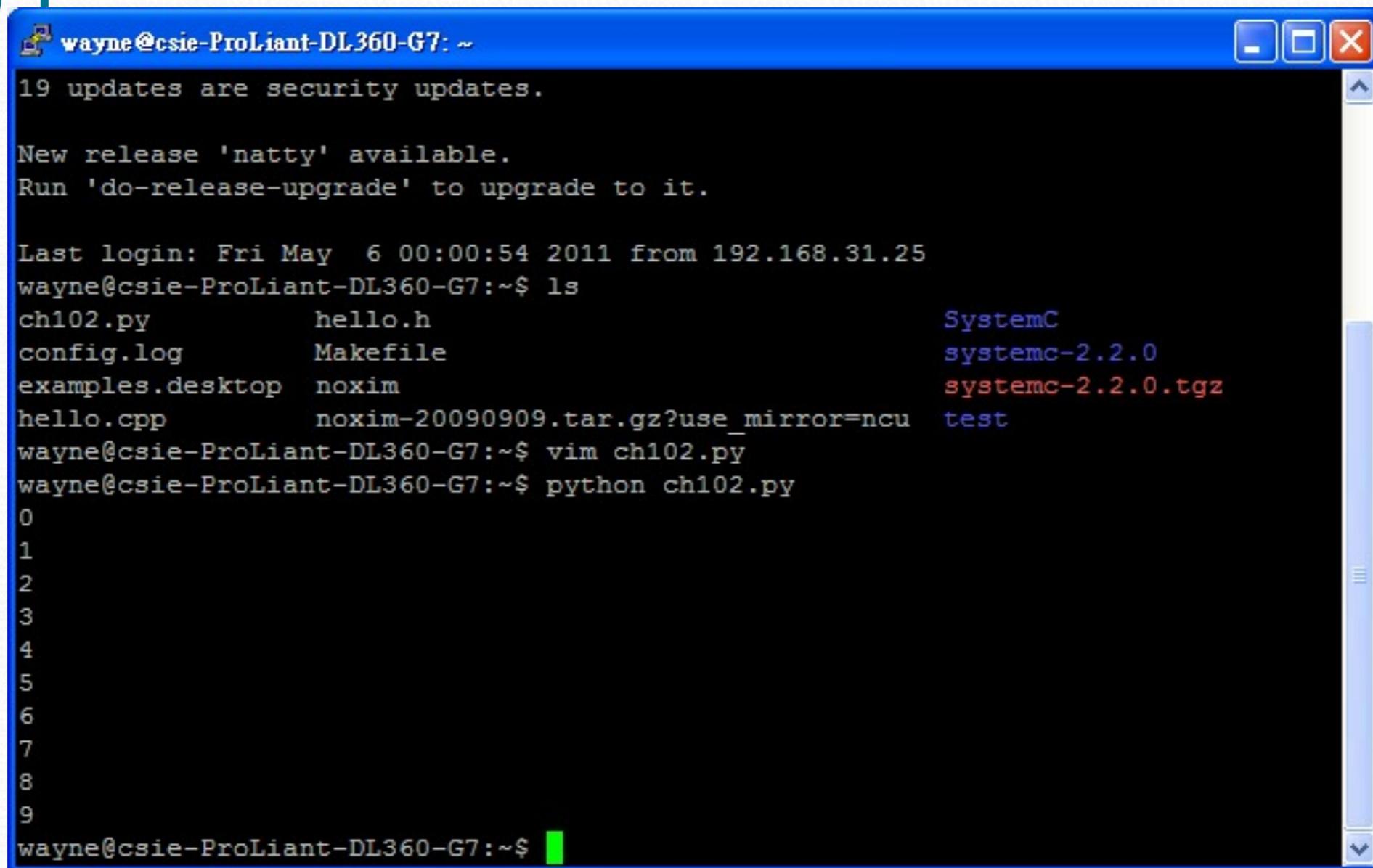
**Terminal 2 (Bottom Right):**

```
wayne@csie-ProLiant-DL360-G7: ~
#!/usr/bin/env python
#coding=utf-8

for x in range(10):
    i=x
    print (i)

"ch102.py" 7L, 78C
```

# 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:~$
```

# 簡介

- 程式中的註解
  - 單行
    - #
  - 多行
    - ""或"""
- 需要每行註解的情況，建議註解在每行的上面

# 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，同行井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

- Python 2.6版與Python 3.0版的差異
  - print從陳述式(statement)改為函數(function)
    - `from __future__ import print_function`
    - `print([object, ...][, sep=""][, end='\n'][, file=sys.stdout])`
  - ex:
    - `print "123?", 10**2`
      - 123? 100
    - `print ("123?", 10**2)`
      - 123? 100

# 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

# 您的第一個python程式 - Hello World

- 使用互動式命令列

```
>>> print "Hello World"  
Hello World  
>>>
```

- 放在檔案裡

```
#!/usr/bin/env python  
print "Hello World"
```

- 記得將檔案改成可執行 chmod a+x <檔名>

# 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。

# 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、型態 (type) 及數值 (value)
- 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。

# 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

# 內建數字型態

- 內建的數字型態 (numeric types) 共有三種，分別是
  - `int` 整數
  - `float` 浮點數
  - `complex` 複數

# 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
  - bytearray: 字節陣列 (byte array) , 可變 (mutable)
  - list: 串列 (list) , 可變
  - tuple: 序對 (tuple) , 不可變
  - range: 內建函數 range() 回傳的物件 (object) , 常用於 for 迴圈 (for loop)

# 序列型態

- 可進行以下的計算

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

# 串列(lists)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

# 內建集合型態 (set)

- 內建的集合型態 (set types) 共有兩種
- set: 集合 (set) , 可變 (mutable)
- frozenset: 原封集合 (frozenset) , 建立後變不新增或刪除元素 (element) , 因此為不可變 (immutable)
- 集合型態的字面常數使用大括弧圍起來，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。例如
  - `s1 = {1, 1, 1, 2, 2, 3, 3, 4, 5}`
  - `s2 = {1, 2, 3, 4, 5}`
  - `s1` 實際等於 `s2`

# 集合型態的物件可進行之運算

計算	描述
$x \in s$	判斷 $x$ 是否在 $s$ 中
$x \notin s$	判斷 $x$ 是否不在 $s$ 中
$s_1 \& s_2$	且運算，取得 $s_1$ 與 $s_2$ 的交集
$s_2   s_2$	或運算，取得 $s_1$ 與 $s_2$ 的聯集
$s_1 ^ s_2$	對稱差運算，取得 $s_1$ 與 $s_2$ 的對稱差集
$s_1 - s_2$	差運算，取得 $s_1$ 與 $s_2$ 的差集
$s_1 < s_2$	判斷 $s_1$ 是否為 $s_2$ 的真子集
$s_1 \leq s_2$	判斷 $s_1$ 是否為 $s_2$ 的子集
$s_1 > s_2$	判斷 $s_2$ 是否為 $s_1$ 的真子集
$s_1 \geq s_2$	判斷 $s_2$ 是否為 $s_1$ 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 $s1 \Delta s2$
<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)

- 內建的字典型態 (dictionary type) 只有一種，由 key:value 配對的複合資料型態 (compound data type)
- 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，每一筆資料再以逗號區隔開，例如
- `d1 = {1:"a", 2:"b"}`
- 上述字典型態的變數 d1 有兩筆資料，第一筆資料的 key 為 1 ， value 為 "a" ，第二筆資料的 key 為 2 ， value 為 "b" 。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

# 字典物件可進行的運算

計算	描述
<code>d[key]</code>	從 d 中取得 key 的 value
<code>d[key] = value</code>	將 d 的 key 指定為 value
<code>del d[key]</code>	刪除 d 中 key 所指定的 value
<code>key in d</code>	判斷 key 是否在 d 中
<code>key not in d</code>	判斷 key 是否不在 d 中
<code>iter(d)</code>	回傳由 d 的 key 建立的迭代器
<code>len(d)</code>	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

# 基本型態 (Numbers and String)

- **Numbers (數)**

a = 3

# Integer (整數)

b = 4.5

# Float point (浮點數)

c = 4 + 3j

# Complex number (複數)

- **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"""

# 基本型態 - 串列(Lists)

- 任意物件的串列

```
a = [2, 3, 4]          # A list of integer  
b = [2, 7, 3.5, "Hello"] # A mixed list  
c = []                 # An empty list  
d = [2, [a, b]]        # A list containing a list  
e = a + b              # Join two lists
```

- 串列的操作

```
x = a[1]              # Get 2nd element (0 is first)  
y = b[1:3]      # Return a sub-list  
z = d[1][0][2]      # Nested lists  
b[0] = 42            # Change an element
```

# 基本型態 - 固定有序列 (Tuples)

- Tuples

```
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

- Tuples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
```

- 特色

- 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

# 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))`

# 複數運算

```
a = 3 + 5j
```

```
print("a =", a)
```

```
b = 3 - 5j
```

```
print("b =", b)
```

```
c = a * b
```

```
print("c =", c)
```

c.real 實部

c.imag 虛部

# 串列(list)、序對(tuple)

- 資料型態
  - Mutable Data Types

- List

```
>>> lists=[1,2,3,4,5]
>>> print(lists)
[1, 2, 3, 4, 5]
>>> lists[1]=lists[3]=0
>>> print(lists)
[1, 0, 3, 0, 5]
```

- Immutable Data Types

- tuple

```
>>> fruit = "apple"
>>> fruit = fruit[:4] + "y"
>>> print(fruit)
apply
```

# 變數與運算

- 變數與運算元的使用

- Ex:

```
>>> 7+7
```

```
14
```

```
>>> _+7
```

```
21
```

# 變數與運算

- 加法

- Ex:

```
>>> x=[2,4,6]
>>> x[0]+x[2]
8
>>> type(x)
<class 'list'>
>>> y={'a1':12,'b2':24}
>>> y['a1']+y['b2']
36
>>> type(y)
<class 'dict'>
```

# 變數與運算

- 乘法
  - 對字串型態做乘法
  - Ex:

```
>>> s = "*"
>>> print(s)
>>> s = s * 2
>>> print(s)
```

\*

\*\*

# 隨堂練習

- 請寫一支程式輸出下列結果

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

# 變數與運算

- 對tuple做乘法
- Ex:

```
>>> x="dog"  
>>> x*6  
'dogdogdogdogdogdog'  
>>> y=("dog")  
>>> y*6  
'dogdogdogdogdogdog'  
>>> y=("dog",)  
>>> y*6  
('dog', 'dog', 'dog', 'dog', 'dog', 'dog')
```

# 變數與運算

- 對字典內的元素做乘法

- Ex:

```
>>> x={'dog':10}  
>>> x['dog']*6  
60
```

# 變數與運算

- 除法

- Ex:

```
>>> float(1/2)
```

```
0.5
```

```
>>> int(1/2)
```

```
0
```

# 變數與運算

- 常用數學函數

- **math**

- `math.ceil(x)` 傳回 $\geq x$ 的最小整數
    - `math.floor(x)` 傳回 $\leq x$ 的最小整數
    - `math.log(x[, base])`

- **Ex:**

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

# 變數與運算

- 內置的數學函數
  - `abs(x)`
  - `min(iterable[, args...][, key])`
  - `max(iterable[, args...][, key])`

# 流程控制及迴圈

- 資料比較

- 數字型態
- 字串型態
  - 單引號與雙引號
  - Ex:

```
>>> x="a"  
>>> y='a'  
>>> x==y
```

True

- ord(c) 傳回字元c的Unicode的編碼值
- Ex:

```
>>> ord('A')
```

# 使用標準函數庫

- 使用time模組
  - time.time()
    - 取得系統時間
  - time.sleep(num)
    - 設定暫停時間
  - time.localtime()
    - 回傳的格式如下：
    - time.struct\_time(tm\_year, tm\_mon, tm\_mday, tm\_hour, tm\_min, tm\_sec, tm\_wday, tm\_yday\_, tm\_isdst)
  - time.gmtime()
    - 取得UTC時間

# 基本型態 - 字典 (Dictionaries)

- Dictionaries (關聯陣列)

```
a = {}                                # An empty dictionary
b = { 'x': 3, 'y': 4 }
c = { 'uid': 105,
      'login': 'beazley',
      'name' : 'David Beazley'
    }
```

- Dictionaries的存取

```
u = c['uid']                          # Get an element
c['shell'] = "/bin/sh"                 # Set an element
if c.has_key("directory"):
    d = c['directory']
else:
    d = None

d = c.get("directory",None)            # Same thing, more compact
```

# 迴圈 (Loops)

- while敘述

```
while a < b:  
    # Do something  
    a = a + 1
```

- for敘述 (走訪序列的元素)

```
for i in [3, 4, 10, 25]:  
    print i
```

```
# Print characters one at a time  
for c in "Hello World":  
    print c
```

```
# Loop over a range of numbers  
for i in range(0,100):  
    print i
```

# 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 string
```

```
...
```

```
a = string.split(x)
```

# 參考

- 官方網頁(英)
  - <http://www.python.org>
- Python教學文件(中)
  - [http://www.freebsd.org.hk/html/python/tut\\_tw/tut.html](http://www.freebsd.org.hk/html/python/tut_tw/tut.html)