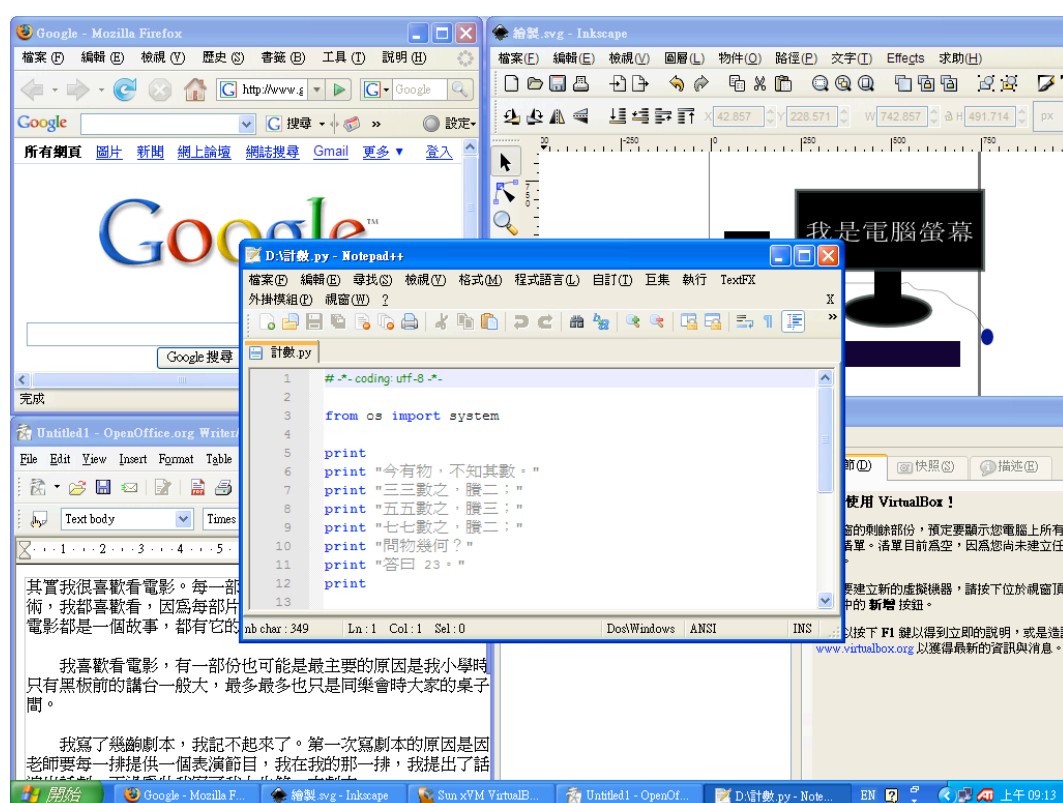


第壹篇 Python基礎

第一章 與直譯器互動

玩遊戲時速度曾經突然變慢嗎？為什麼呢？啊！MSN忘記登出，偶然的訊息打斷電腦目前的運作，原先一件件電腦已經排定的工作，現在又得重排一次。

電腦是如何安排工作的呢？對了，就是作業系統。作業系統是我們與電腦溝通最重要的介面，當我們利用電腦完成我們日常的工作時，開啟多個視窗好像已經是再普通不過的事情了。

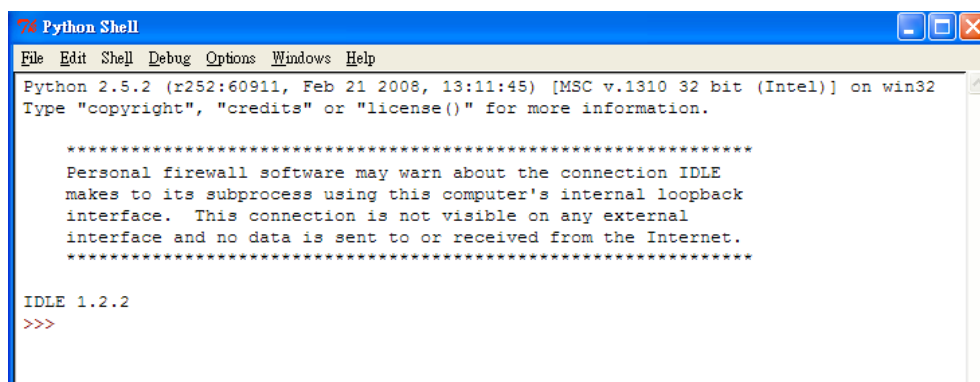


許許小小的應用軟體透過作業系統的協調，重疊的視窗讓我們得以同時進行處理。那麼，如果我們要寫程式跟電腦溝通又該如何進行呢？仍是一樣，透過作業系統。

利用編譯式語言寫程式，我們需要另外用文字編輯器，如記事本之類的軟體先寫好程式碼，然後進行連結、編譯等流程。當發展的程式越趨龐大時，每一次寫完程式之後的測試動作變成一件很恐怖的事情，畢竟人很容易犯錯，於是常常要在密密麻麻的文字中尋找錯誤，然後修正，然後進行編譯，然後測試，然後除錯，再一次修正，再一次這樣那樣。如果不滿意現階段的程式執行效果，所有的步驟又得重來一次。

很麻煩，不是嗎？但是**除錯**是很重要的，正因為犯錯是難免的。直譯式語言避開編譯的繁瑣，使程式的原始碼可以直接透過直譯器執行，執行的同時直譯器便會檢查是否存在錯誤，如果有，修改過後就可以連續下一階段的發展、測試、修改等等。

如我們在MS-Windows底下利用Python，我們可以直接開啟IDLE（Integrated DeveLopment Environment），也就是整合的發展環境，如下圖。



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.5.2 (r252:60911, Feb 21 2008, 13:11:45) [MSC v.1310 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.

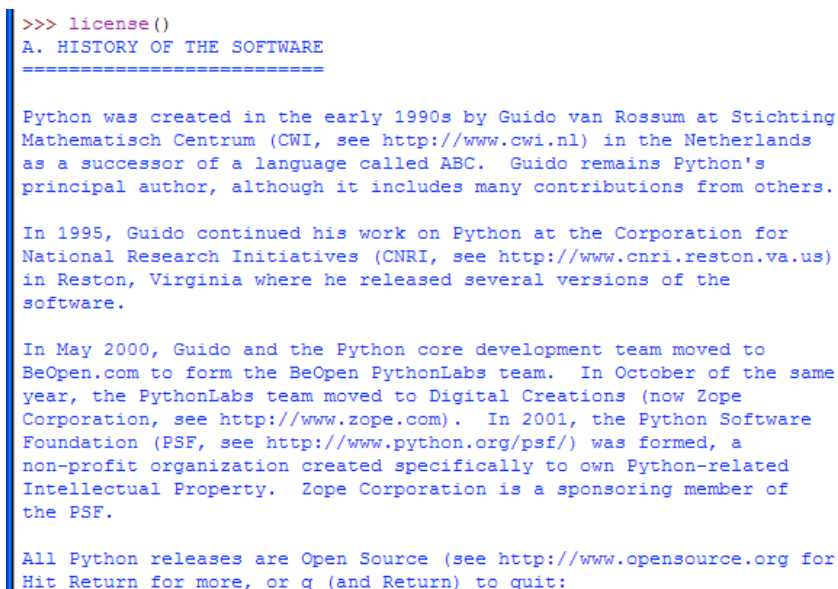
*****
Personal firewall software may warn about the connection IDLE
makes to its subprocess using this computer's internal loopback
interface. This connection is not visible on any external
interface and no data is sent to or received from the Internet.
*****

IDLE 1.2.2
>>>
```

視窗名稱為Python Shell。Shell的英文原意是貝殼，或是類似貝殼的東西，在電腦的領域裡，shell被借來當成「環境」的意思，這就像貝殼是軟體動物所依賴的居住「環境」。

這樣的 Python Shell 就是屬於我們電腦裡的一個應用軟體，仍是透過作業系統來運作。我們可以看到視窗最上方顯示Python的版本為2.5.2，版本建立時間為2008年的2月21日，所運作的電腦平台在Intel CPU上的win32系統，也就是MS-Windows。

底下建議我們可以鍵入copyright、credits或是license()以獲得更多資訊，像是這樣的英文字，在Python Shell中如同「命令提示字元」的命令一般，下方的「>>>」是等待我們輸入的提示字元，如我們鍵入license()，可以得到如下。



```
>>> license()
A. HISTORY OF THE SOFTWARE
=====

Python was created in the early 1990s by Guido van Rossum at Stichting
Mathematisch Centrum (CWI, see http://www.cwi.nl) in the Netherlands
as a successor of a language called ABC. Guido remains Python's
principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for
National Research Initiatives (CNRI, see http://www.cnri.reston.va.us)
in Reston, Virginia where he released several versions of the
software.

In May 2000, Guido and the Python core development team moved to
BeOpen.com to form the BeOpen PythonLabs team. In October of the same
year, the PythonLabs team moved to Digital Creations (now Zope
Corporation, see http://www.zope.com). In 2001, the Python Software
Foundation (PSF, see http://www.python.org/psf/) was formed, a
non-profit organization created specifically to own Python-related
Intellectual Property. Zope Corporation is a sponsoring member of
the PSF.

All Python releases are Open Source (see http://www.opensource.org for
Hit Return for more, or q (and Return) to quit:
```

我們可以看到打完license後文字顏色變成紫色，其下所顯示的內容文字顏色則為藍色，這是整合發展環境的特色之一，具有不同意義的文字，分別用不同顏色來表示。

Note

兩行星號之間的字提醒使用者有關防火牆軟體的問題。

License的中文是許可或是執照、牌照的意思，在電腦的領域我們多拿來當作「授權」之意。鍵入license()之後我們首先看見Python發展的歷史資訊，然後依照最下方的提示，按Return鍵（也就是Enter鍵）到下一頁，或是可以鍵入q離開。

計算機功能

在Python Shell之中，我們可以直接與Python直譯器互動，如簡單的加減乘除計算。

```
>>> 7+2
9
>>> 7-2
5
>>> 7*2
14
>>> 7/2
3
>>>
```

鍵盤裡表示加號和減號的按鍵跟我們數學上所用的相同，乘號用星號表示，除號則是用斜線表示。咦？為什麼7除以2的結果不是3.5，而是3呢？

這是因為電腦裡的數字都是整數，小數則是利用模擬的方式計算出來的，因而這樣的除法被稱為**整數除法**，所以我們得到的結果是7除以2的商，也就是3。小數的模擬情況是怎樣的呢？如果我們鍵入0.1到Python Shell中，我們會得到如下。

```
>>> 0.1
0.10000000000000001
```

雖然是非常非常小的誤差，但是當我們要求一定的精確度時，這樣的誤差大概就會顯著影響到計算結果。那又如果我們希望得到的是我們所習慣的小數表示方法呢？我們可以將除數或被除數其中之一改成帶有小數點，如下。

```
>>> 7/2.
3.5
```

我們也能夠用百分比符號取餘數，而用兩個連續的星號計算次方。

```
>>> 7%2
1
>>> 7**2
49
```

Python利用這些符號進行計算，這些符號被稱為**運算子**，進行計算的數值則被稱為**運算元**，運算元與運算子構成**運算式**。

運算種類	運算子
加法	+
減法	-
乘法	*
除法	/
餘數	%
次方	**

基本資料型態

程式設計中**資料型態**是很重要的觀念，因為每一種資料型態都有其特定處理的方式。當然，我們可以直接在Python Shell詢問直譯器這筆資料是屬於哪一種資料型態。

```
>>> type(7)
<type 'int'>
>>> type(7.0)
<type 'float'>
```

這裡type()的用法像是一種在Python Shell中的命令，如同前面提過的license()，小括弧內的數值便是我們提供給這個命令的參數，也就是我們所希望藉由這個命令得到7或是7.0是那一種資料型態的結果。

7是屬於int型態，int是integer的縮寫，也就是說7是屬於**整數**型態，而7.0屬於float型態，float則是 floating point 的縮寫，在電腦術語裡，我們把像這樣帶有小數點的數值稱為**浮點數**型態。整數與浮點數兩者是Python中的基本資料型態。

7與7.0不是相同的數值嗎？可是分屬兩個不同的型態，他們的數值的確是相等的，然而7.0帶有小數，因而兩者最主要的差別就在於精確度。

如果我們直接鍵入type()不給於任何的參數，又會發生什麼事情呢？

```
>>> type()

Traceback (most recent call last):
  File "<pyshell#54>", line 1, in <module>
    type()
TypeError: type() takes 1 or 3 arguments
```

結果Python Shell告訴我們這是一個錯誤情況，叫做TypeError，這是說type()需要1或3個參數，同時利用Traceback告訴我們錯誤發生在程式的那一行。

Note

雖然argument常被翻譯成引數，用來跟翻譯成參數的parameter相對，然而兩者實際的意義並無不同，所以在這裡我們將只會用「參數」，而不會用引數之詞。

變數的觀念

變數的用法其實和數學中一樣，我們都可以利用英文字母來代替某些數字。這要如何做呢？我們可以利用**指派陳述**來做到。

```
>>> x=7
>>> y=2
>>> x+y
9
>>> x-y
5
>>> x*y
14
>>> x/y
3
```

陳述是程式中的基本元件，所謂的陳述是我們可以要求Python替我們做事情的句子，如利用上述的指派陳述，我們便要求將7儲存到x變數中，另外也將2儲存到y的變數之中。運算式也是陳述的一種。

在觀念上，數學中的變數通常表示未知數，變數藉由某種運算過程而可求出未知數的值，程式裡的變數則是給數字的替代名稱，這樣有什麼好處呢？當變數作為名稱時，我們便可對這個名稱進行操作，如將1指派到變數i，再將i+1指派到變數i當中，於是現在變數i所儲存的數字變成了2。

```
>>> i=1
>>> i=i+1
>>> i
2
```

正因為變數允許我們用名稱代替數字，於是我們寫程式時便可以將目光集中在程式本身的目的，譬如我們想要計算圓的面積，我們先將圓周率3.14指派到變數pi之中，再將半徑2指派到變數radius之中，最後利用變數area儲存面積的計算結果。

```
>>> pi=3.14
>>> radius=2
>>> area=radius*radius*pi
>>> area
12.56
```

英文中area就是面積的意思，我們用area做變數名稱，讓我們輕易掌握到這個變數在程式中所代表的意義，然而，這要等到程式越來越長的時候，才比較容易感覺出這樣的效益。

Help! Help!

我們也可以直接在Python Shell向直譯器討教，如直接鍵入help()。

```
>>> help()

Welcome to Python 2.5! This is the online help utility.

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://www.python.org/doc/tut/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, or topics, type "modules",
"keywords", or "topics". Each module also comes with a one-line summary
of what it does; to list the modules whose summaries contain a given word
such as "spam", type "modules spam".

help>
```

我們可以看到這樣就進入了另一個互動模式，這裡，我們可以直接查詢Python的說明文件，在底下的help後面跟著「>」的提示符號後進行輸入，譬如我們輸入if來看看會有怎麼樣的結果。

```

help> if
-----

7.1 The if statement

The if statement is used for conditional execution:

    if_stmt ::=      "if" expression[1] ":" suite[2]
                    ( "elif" expression[3] ":" suite[4] ) *
                    ["else" ":" suite[5]]

Download entire grammar as text.[6]

It selects exactly one of the suites by evaluating the expressions one
by one until one is found to be true (see section 5.10[7] for the
definition of true and false); then that suite is executed (and no other
part of the if statement is executed or evaluated). If all expressions
are false, the suite of the else clause, if present, is executed.

-----

Release 2.5.2, documentation updated on 21th February, 2008.

Related help topics: TRUTHVALUE

```

我們很快的會用到if陳述，相同的內容在Python Reference Manual可以找到，除了連同IDLE安裝的Python Documentation可以閱讀外，另外在<http://docs.python.org/ref/if.html>的網址也能在線上閱讀if的說明文件。

Note

Python的說明文件都放在[Python Documentation](http://docs.python.org)。

我們也可以提供參數給help()，來查詢有關參數的說明文件，如查詢help本身。

```

>>> help(help)
Help on _Helper in module site object:

class _Helper(__builtin__.object)
|   Define the built-in 'help'.
|   This is a wrapper around pydoc.help (with a twist).
|
|   Methods defined here:
|
|   __call__(self, *args, **kwargs)
|   __repr__(self)
|
|   -----
|   Data descriptors defined here:
|
|   __dict__
|       dictionary for instance variables (if defined)
|
|   __weakref__
|       list of weak references to the object (if defined)

```

嗯，內容有點難懂，不是嗎？放心，直接在 Python Shell 裡查詢說明文件是個很棒的功能！尤其往後發展程式的時候，如果有所疑慮，進入 Python Shell 中不但可以進行測試，同時可以查閱說明文件。雖然有些我們現在看不太懂，但是隨著我們繼續下去，慢慢的都能接受說明文件的表達方式。

如果是鍵入help(if)呢？

```

>>> help(if)
SyntaxError: invalid syntax

```


錯誤的語法，為什麼？這是有關help的設計問題了，我們可以在help互動模式下查詢if的資料，卻不能利用help(if)來查詢。

Note

有關help的詳情，請參考[PEP 233](#)。

第一個複合資料型態：字串

程式裡經常處理的除了整數與浮點數兩種基本資料型態外，有很多時候都要處理文字，文字在Python屬於複合的資料型態，這是用兩個英文的引號標點圍繞起來。

```
>>> "Hello, world!"  
'Hello, world!'
```

一樣的，我們可以利用type(“Hello, world!”)詢問這是屬於哪一種資料型態。

```
>>> type("Hello, world")  
<type 'str'>
```

str是string的縮寫，我們稱這種資料型態為**字串**。

英文的單引號或雙引號都可以用來表示字串。為什麼說字串是複合資料型態呢？因為我們可以將字串切開來，利用中括弧個別存取其中單一個字元。首先我們把“Hello, world!”指派到變數greeting中，然後存取greeting的第一個字元。

```
>>> greeting="Hello, world!"  
>>> greeting[0]  
'H'
```

Note

這是由於編碼的關係，二十六個英文字母分別都是一個字元，包括可以直接從鍵盤輸入的半形標點符號，而中文字及全形標點符號都是兩個字元。

中括弧內的數字稱為**索引值**。咦？第一個字元的索引值怎麼是0呢？其實這是一種在程式語言中的習慣，大多數的程式語言都由0開始計數，所以0是第一個字元的索引值，1是第二個，餘下以此類推。

由0開始，我們剛開始可能會不太適應，不過沒關係，慢慢的習慣這種方式，也能慢慢的體會其實這樣的表達會更自然。除了一個一個的切開單獨字元，我們還可以一次切出好幾個字元。

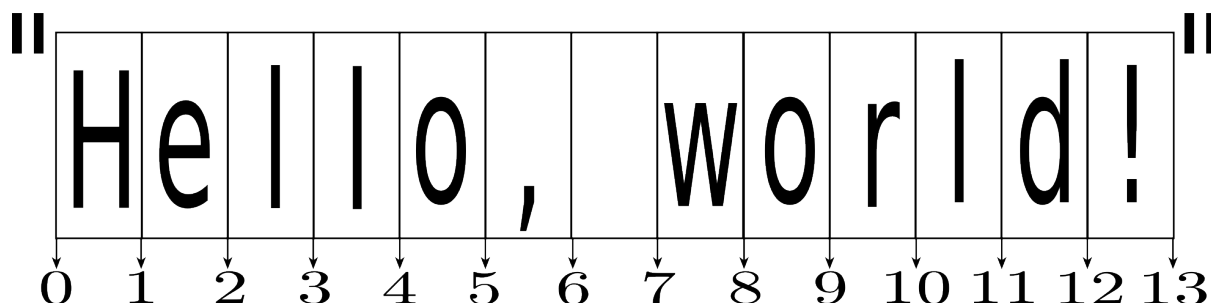
```
>>> greeting[7:12]  
'world'
```

索引值，加冒號，然後接著另一個索引值，第一個索引值表示切下去的起始點，第二個索引值則是結束的地方。我們所用的“Hello, world!”字串之中，逗點之後有一個空格叫做為空白字元，被稱為**跳脫序列**，因為這樣不可見的字元有許多種，所以用序列來稱呼。

Note

這仍然是編碼的關係，基本上，我們在螢幕上所見到的任何東西都經過編碼，不管可見還是不可見，或是那一種顏色。跳脫序列則是不可見字元，如Tab鍵、空格鍵、換行符號等的種類稱呼。

雖然存取單一個字元是以0開始計數，事實上，索引值的計算是採取間隔的方式，如下圖所示。



所以當我們要存取world這個單字的時候，索引值是從7到12，而不是到11就結束了呢！

Note

若是不輸入第一個索引值，預設從0開始，同樣不輸入第二個索引值，預設則到最末的索引值，而要是兩個索引值都沒有輸入，如greeting[::]的話則會存取整個字串。

印出“哈囉！你好。”

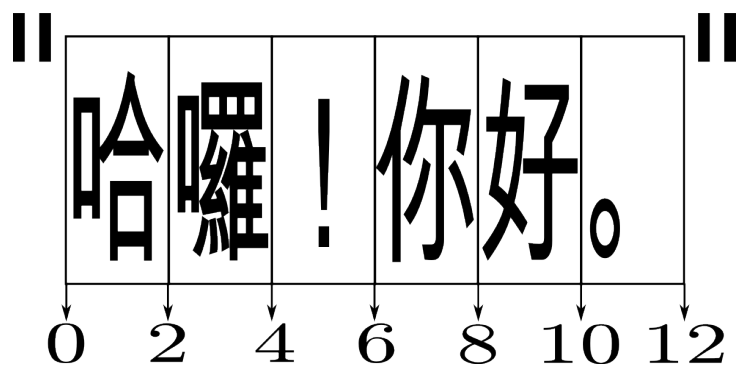
英文字串的處理大體上如此，中文卻有點不太一樣，如何的不一樣呢？

```
>>> hello="哈囉！你好。"
>>> hello
'\xab\xa2\xc5o\xa1I\xa7A\xa6n\xa1C'
>>> type(hello)
<type 'str'>
```

我們將「哈囉！你好。」的文字放進雙引號中，然後指派給hello變數，但是在Python shell裡卻對這個變數顯示奇怪的數字，我們用type(hello)來檢查，沒錯，變數hello之中所儲存確實是字串。

雖然在IDLE中可以直接在提示符號(>>>)後輸入中文字串，但是IDLE所顯示的會是系統預設的編碼，在Windows下便是Big5編碼，每一個文字符號都採用兩個字元，有別於一般英文所用一個字元的ASCII編碼。所以，當我們想要直接查看變數hello所儲存的內容時，我們所得到的便是ASCII的編碼結果。

如果我們要從中文字串中切出一部分，要特別小心一個中文字含括兩個字元。



所以要切出「你好」這個字串時，索引值從6到10。

```
>>> hello[6:10]
'\xa7A\xa6n'
```

確實沒錯，從6到10的索引值的確切出「你好」。同樣的，如果我們要查看hello的內容，就用print hello。

```
>>> print hello[6:10]
你好
```