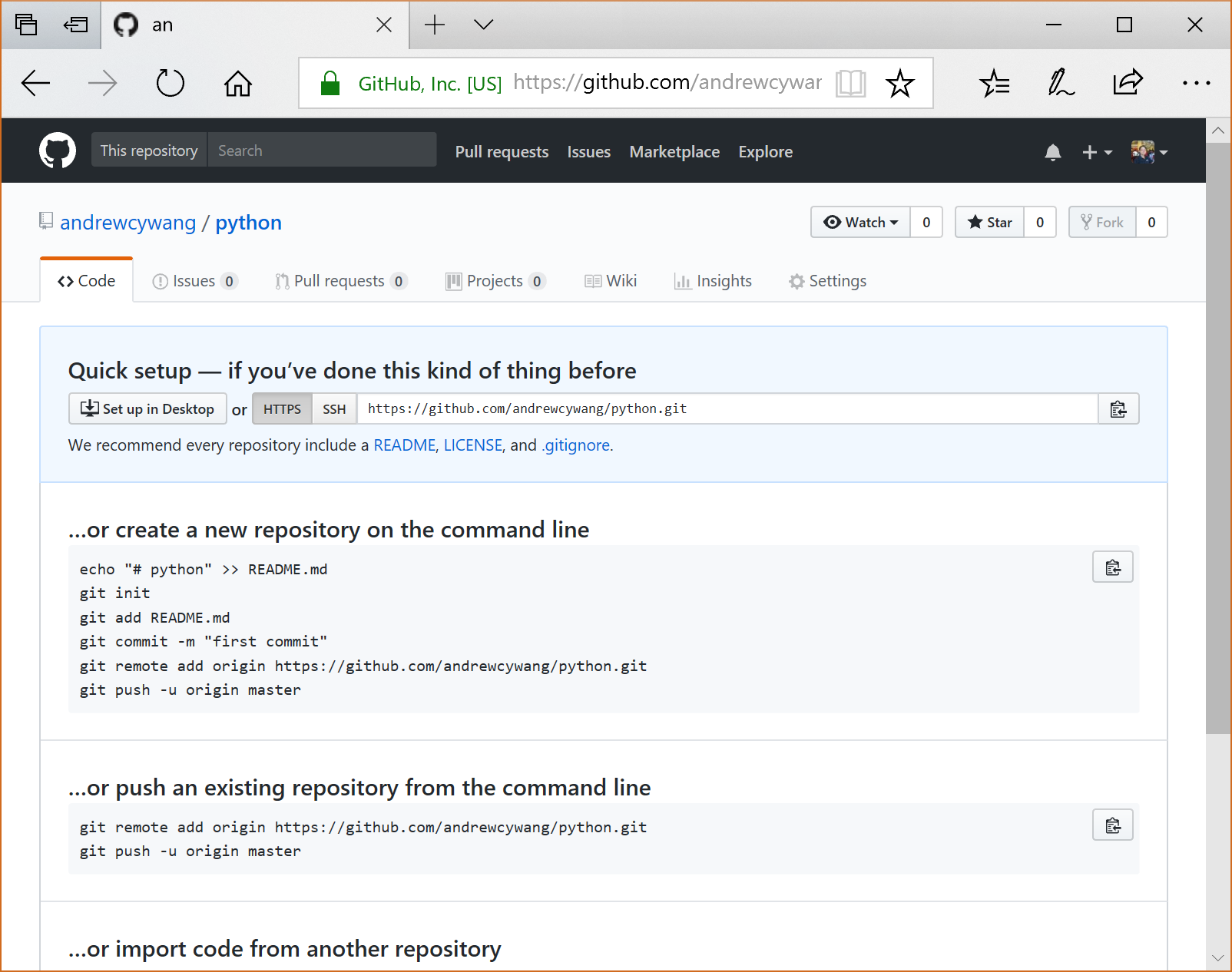
# . Python也可以這樣玩 .

作者：王昌元

## 課前準備

### GitHub Repository

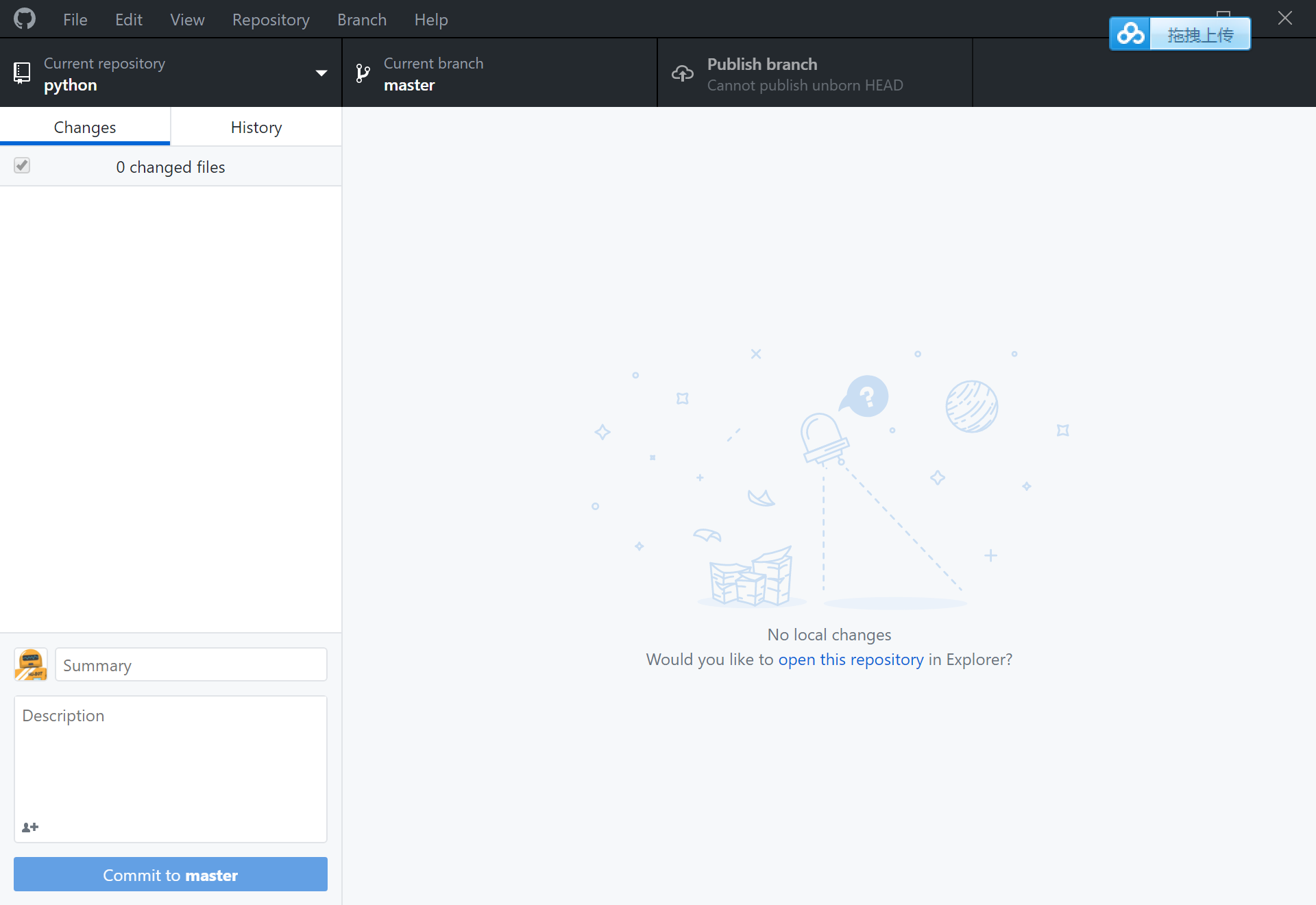
GitHub 是一個很好的工具，可以讓我們共同維護程式碼，因此我使用 GitHub作為教案以及程式碼分享的平台，如下圖：



在我的 andrewcywang 帳號中，我開了一個 python 的 Repository，所有的分享都會放在其中。

### GitHub Desktop

有了存放空間之後，也在桌面安裝了 GitHub Desktop，可以讓所有資料同步。如下圖：



## Python 簡介

Python（英國發音：/ˈpaɪθən/ 美國發音：/ˈpaɪθɑːn/），是一種廣泛使用的高階程式語言，屬於通用型程式語言，由吉多·范羅蘇姆 創造，第一版釋出於 1991 年。可以視之為一種改良的 LISP。作為一種直譯語言，Python 的設計哲學強調代碼的可讀性和簡潔的語法。相比於 C++ 或 Java，Python 讓開發者能夠用更少的代碼表達想法。不管是小型還是大型程式，該語言都試圖讓程式的結構清晰明了。

與 Scheme、Ruby、Perl、Tcl 等動態型別程式語言一樣，Python 擁有動態型別系統和垃圾回收功能，能夠自動管理記憶體使用，並且支援多種編程範式，包括物件導向、命令式、函數式和程序式編程。其本身擁有一個巨大而廣泛的標準庫。

### Python應用範圍

#### Web程式

Python經常被用於Web開發。比如，通過mod\_wsgi模組，Apache可以運行用Python編寫的Web程式。使用Python語言編寫的Gunicorn作為Web伺服器，也能夠執行Python語言編寫的Web程式。Python定義了WSGI標準應用介面來協調Http伺服器與基於Python的Web程式之間的溝通。一些Web框架，如Django、Pyramid、TurboGears、Tornado、web2py、Zope、Flask等，可以讓程式設計師輕鬆地開發和管理複雜的Web程式。

Python對於各種網路協定的支援很完善，因此經常被用於編寫伺服器軟體、網路爬蟲。第三方函式庫Twisted支援非同步線上編寫程式和多數標準的網路協定（包含用戶端和伺服器），並且提供了多種工具，被廣泛用於編寫高效能的伺服器軟體。另有gevent這個流行的第三方庫，同樣能夠支援高效能高並行的網路開發。

#### GUI開發

Python本身包含的Tkinter庫能夠支援簡單的GUI開發。但是越來越多的Python程式設計師選擇wxPython或者PyQt等GUI套件來開發跨平台的桌面軟體。使用它們開發的桌面軟體執行速度快，與用戶的桌面環境相契合。通過PyInstaller還能將程式釋出為獨立的安裝程式包。

#### 作業系統

在很多作業系統裡，Python是標準的系統元件。大多數Linux發行版和Mac OS X都整合了Python，可以在終端機下直接執行Python。有一些Linux發行版的安裝器使用Python語言編寫，比如Ubuntu的Ubiquity安裝器、Red Hat Linux和Fedora的Anaconda安裝器。在RPM系列Linux發行版中，有一些系統元件就是用Python編寫的。Gentoo Linux使用Python來編寫它的Portage軟體包管理系統。Python標準庫包含了多個調用作業系統功能的函式庫。通過pywin32這個第三方軟體包，Python能夠存取Windows的COM服務及其它Windows API。使用IronPython，Python程式能夠直接調用.Net Framework。

#### 科學、遊戲、與其他應用

NumPy、SciPy、Matplotlib可以讓Python程式設計師編寫科學計算程式。有些公司會使用Scons代替make構建C++程式。

很多遊戲使用C++編寫圖形顯示等高效能模組，而使用Python或者Lua編寫遊戲的邏輯、伺服器。相較於Python，Lua的功能更簡單、體積更小；而Python則支援更多的特性和資料類型。很多遊戲，如EVE Online使用Python來處理遊戲中繁多的邏輯。

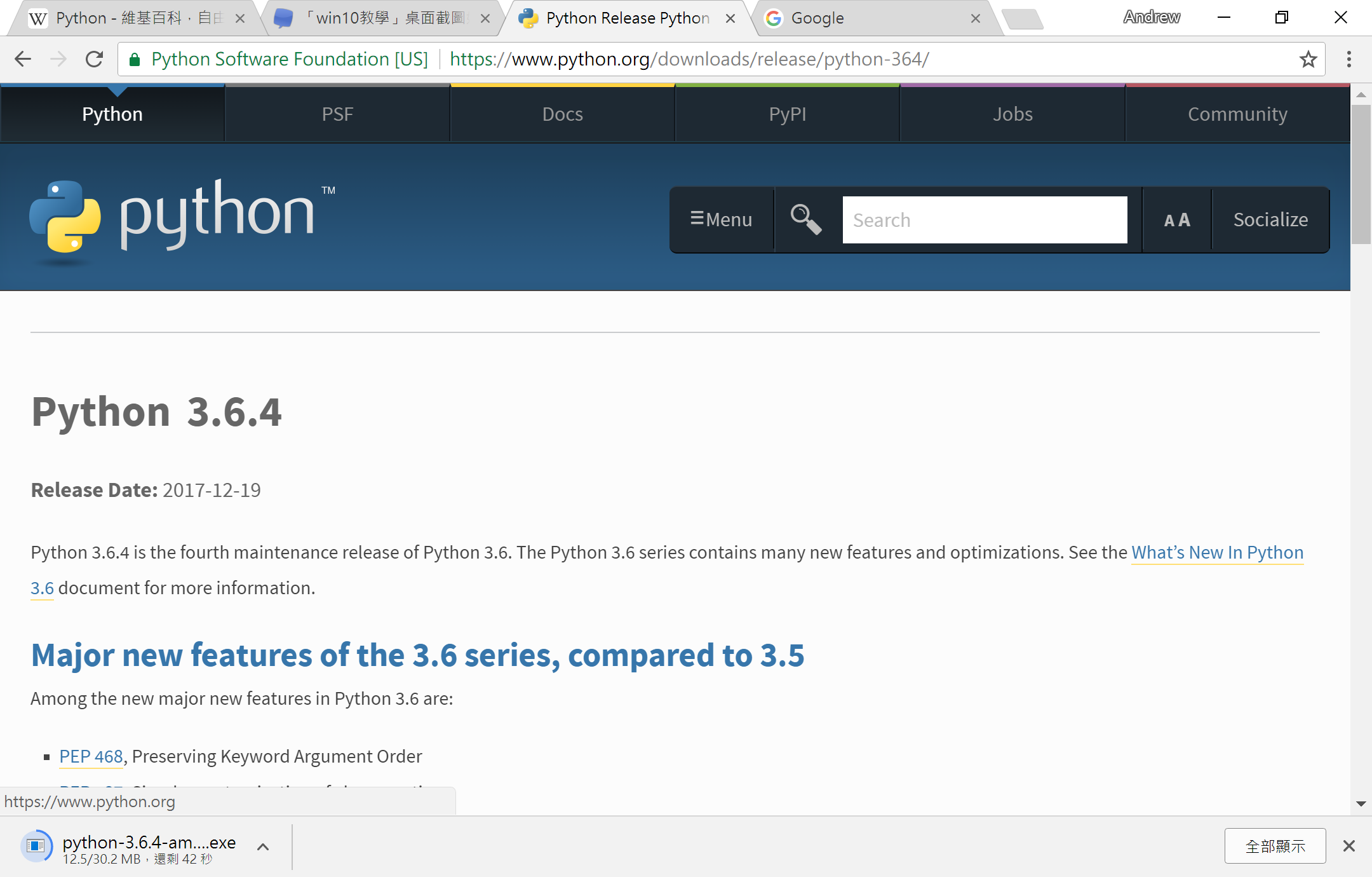
YouTube、Google、Yahoo!、NASA都在內部大量地使用Python。OLPC的作業系統Sugar項目的大多數軟體都是使用Python編寫。

## Python 開發環境

工欲善其事，必先利其器，程式設計也是這樣，熟悉開發環境是應該學習的第一步。

### IDLE

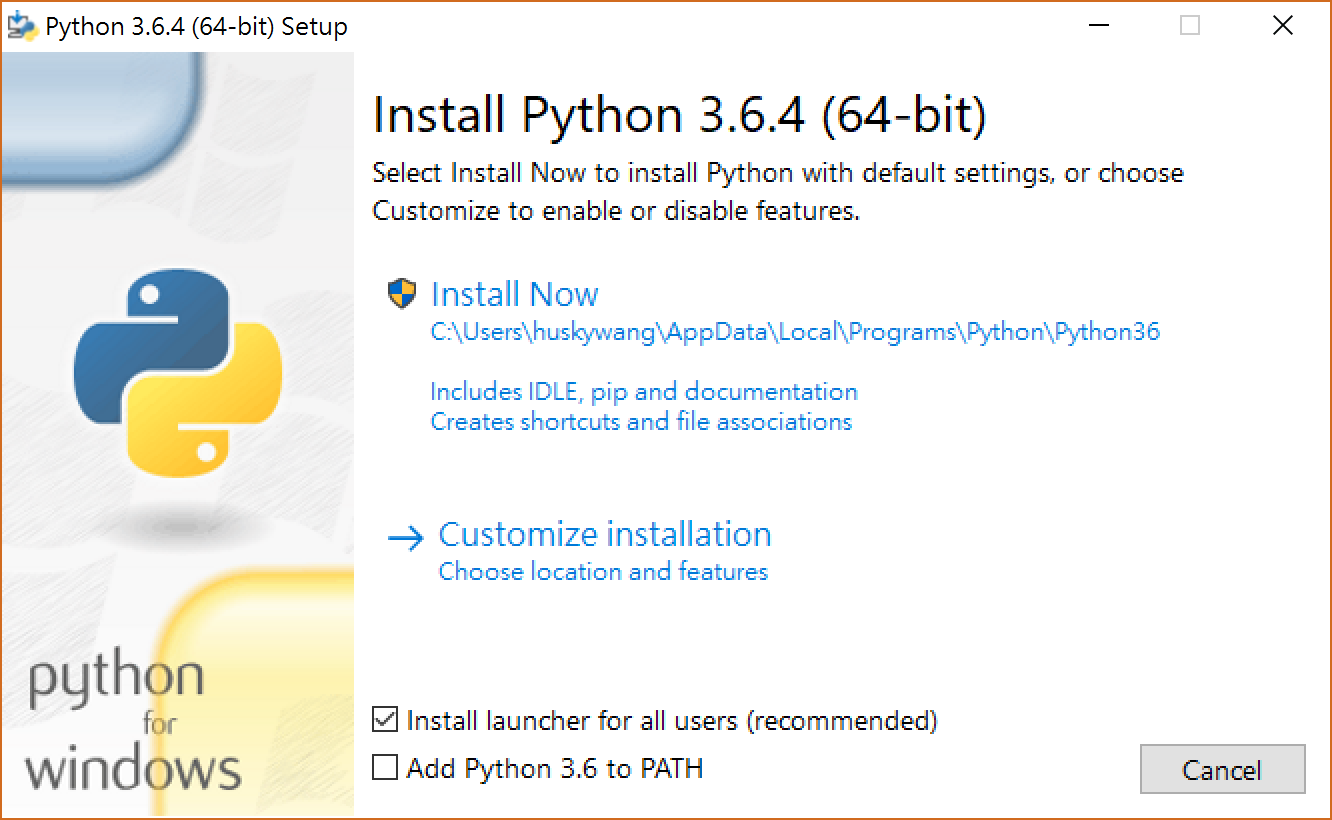
IDLE是Python的官方標準開發環境，可以從官方網站www.python.org下載。



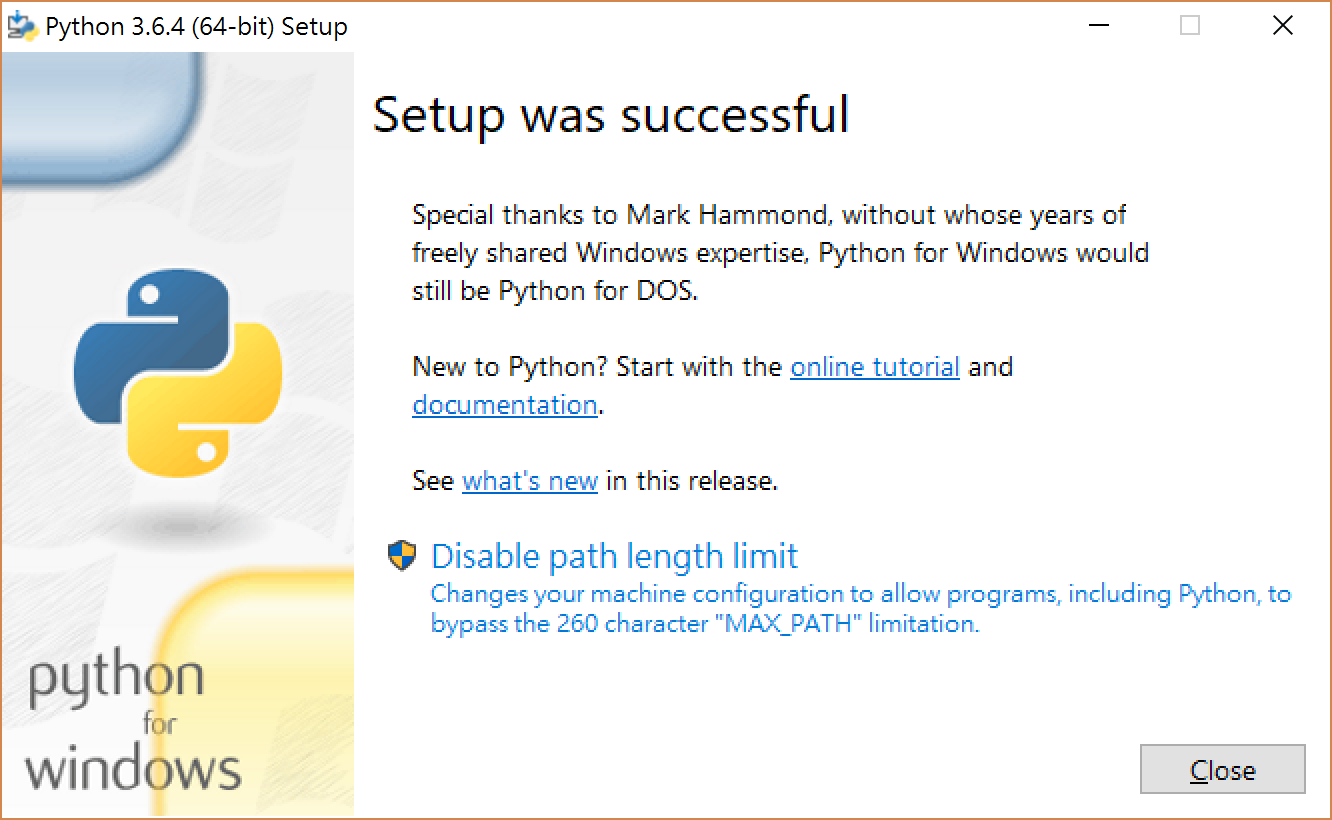
目前最新的版本會顯示在官方網頁，要下載 Windows 的安裝版本。有了最基本的工具，就可以開始寫程式了。為了打好基礎，建議從最基本的開始。

### 安裝 Python 3.6.4

直接執行 python-3.6.4-amd64.exe，進入安裝畫面：



安裝的第一步，建議將 Add Python 3.6 to PATH打勾，這樣在DOS視窗執行的時候，就不必一直切換目錄，按下 Install Now安裝。安裝完畢之後會出現下圖：

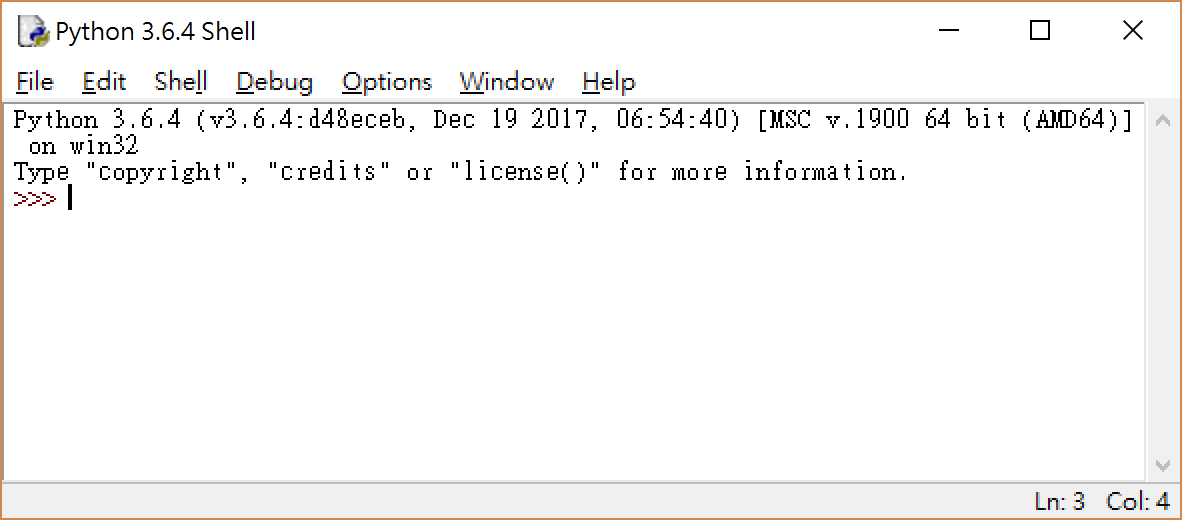


我也建議按下 Disable path length limit，突破過去DOS的限制。

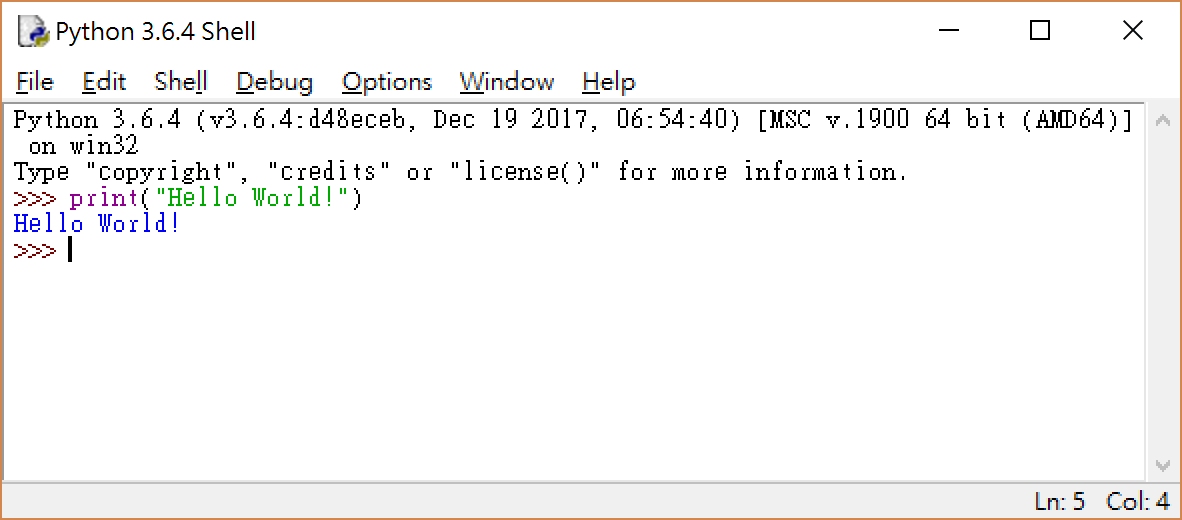
安裝完成之後，我們就可以從[開始]功能表執行[IDLE Python 3.6]，正式進入 Python 的世界。

### 撰寫第一個Python程式

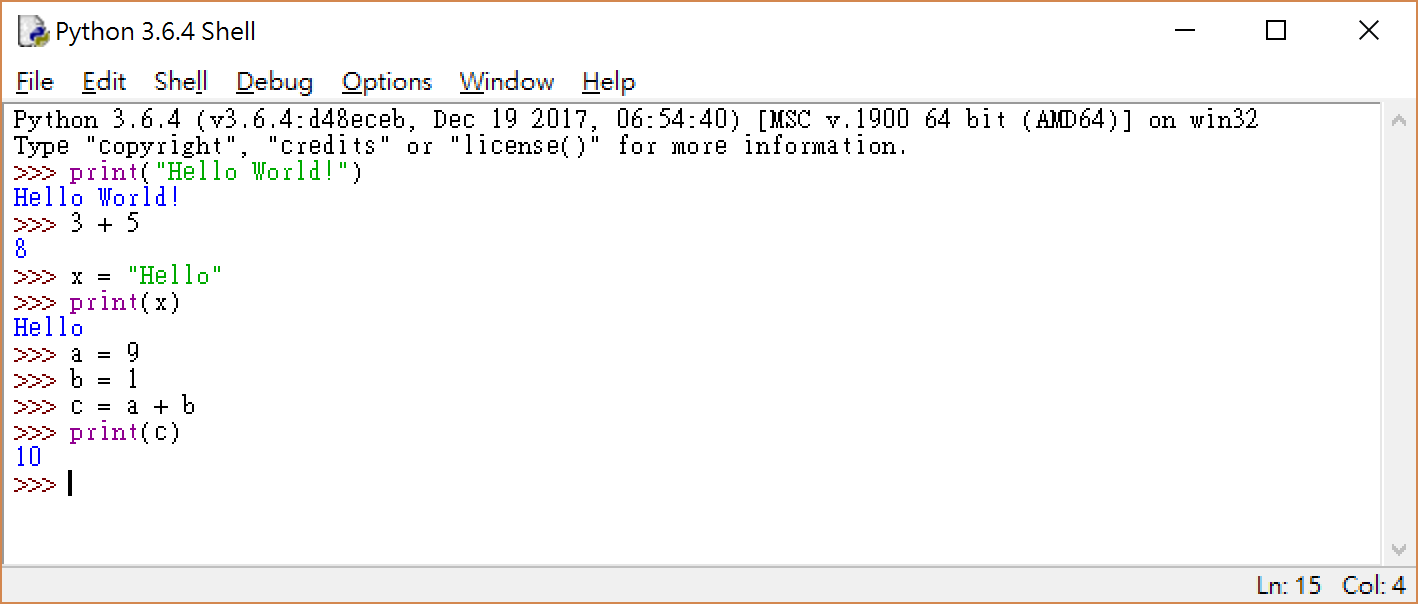
讓我們撰寫一個 Hello World! 程式，這是學習任何一種程式語言的慣例。開啟 Python IDLE之後，會看到下面的畫面：



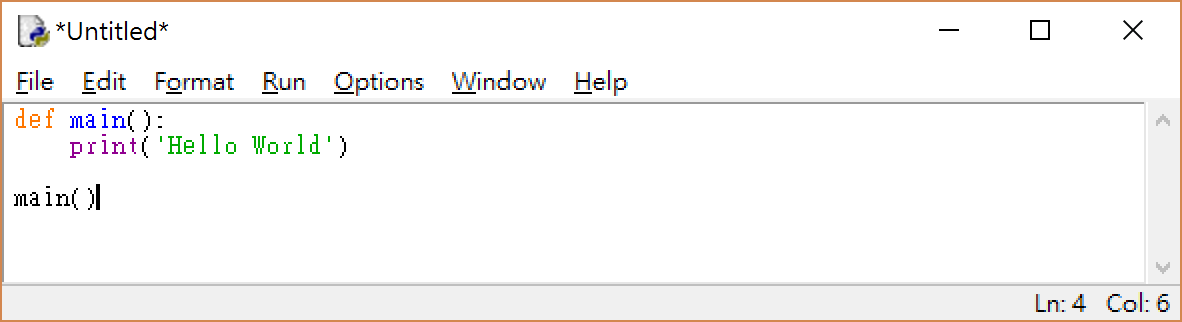
感覺上，我們又好像回到了DOS時代，這畫面就是一個簡單的執行殼，我們直接在>>>後面輸入程式指令，按下ENTER之後執行。Hello World! 程式就是讓 Python 輸出一個 Hello World!字串，我們使用 print 函數來達成：



我們現在看到的這個模式，稱之為交互模式，直接在Python提示字元 “>>>” 輸入命令並按下[ENTER]按鈕即可，如果命令正確的話，馬上就可以看到結果：

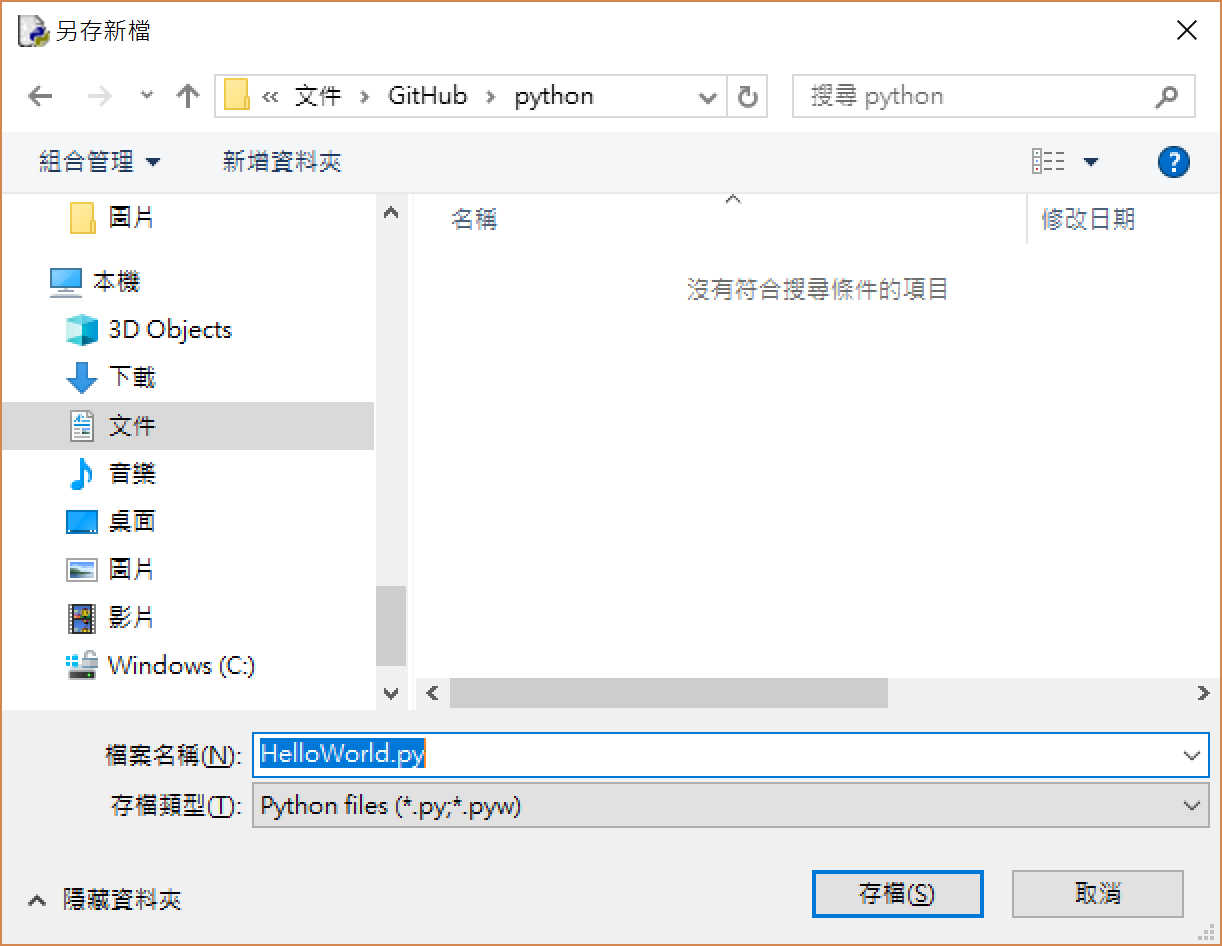


在這裡執行命令就好像跟Python對話一樣，但是都是一行一行執行的，並不像執行程式，程式都是由多個命令所組成的，然後儲存在一個檔案之中，如果我們要把 Hello World 寫成一個程式該如何做呢？第一步當然是建立一個新的程式檔案。在IDLE介面使用功能表[FILE] [NEW FILE]新建一個檔案，這時候會出現一個新的文字編輯器，按照 Python 的語法，程式碼是寫在一個叫做 main() 的函數裡面，這和C語言很像，定義好之後再執行 main() 即可，寫法如下：

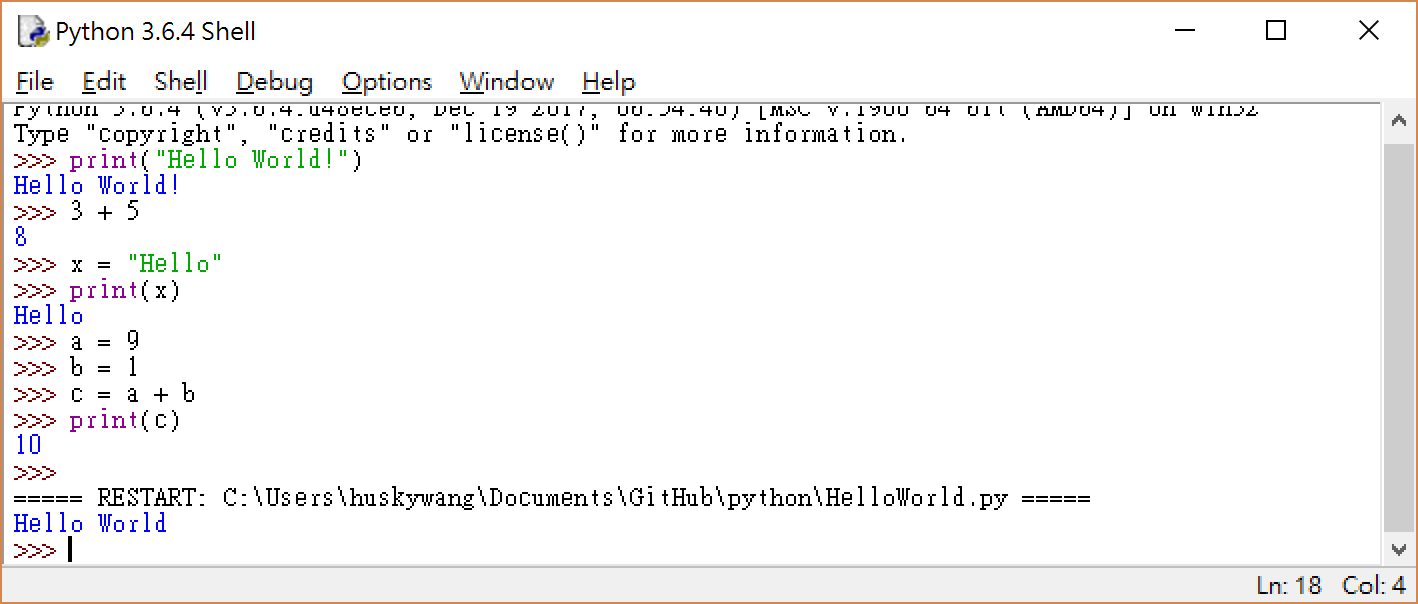


前兩行我們使用關鍵字 def 定義了 main()，要做的事情則利用縮排寫在裡面，就是 print('Hello World') ，然後第三行就是讓 Python 執行 main()。

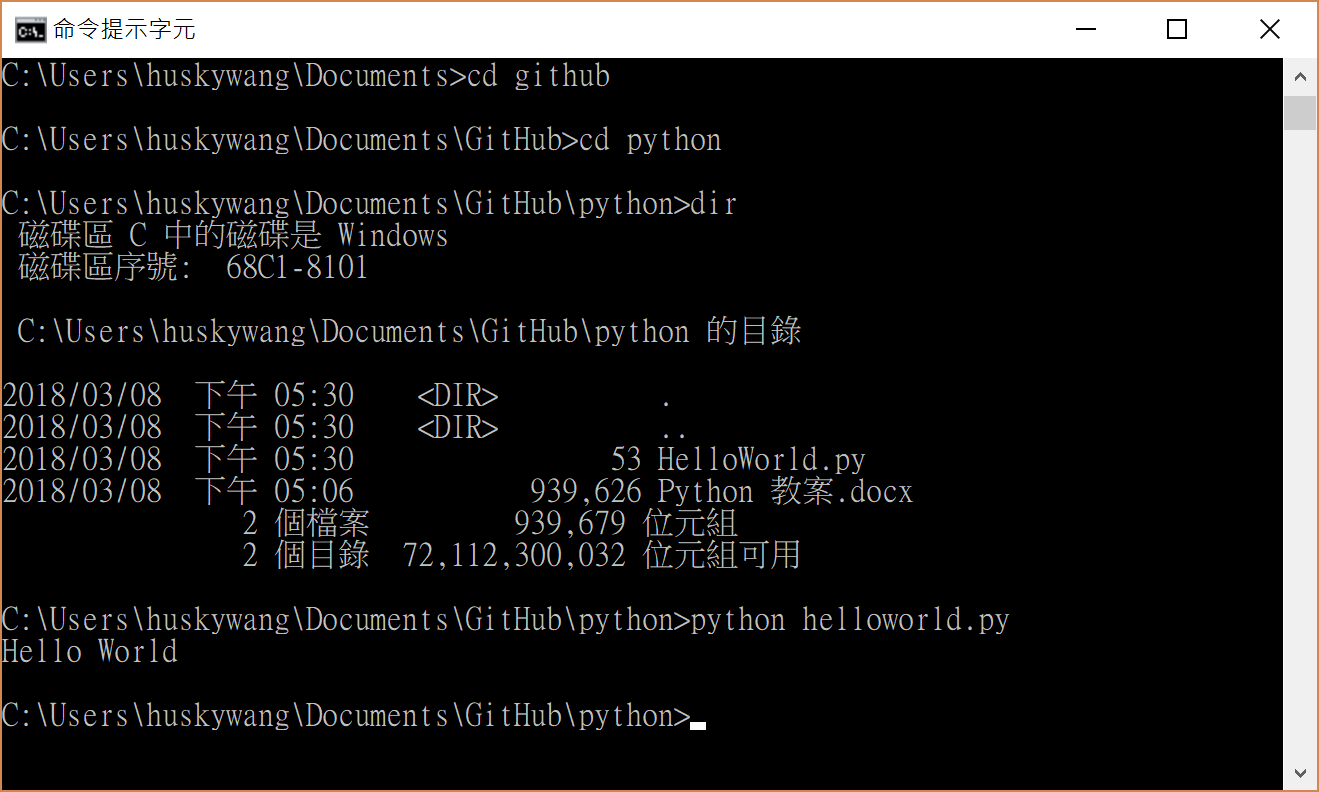
程式寫好之後一定要存檔，Python程式檔是以 .py 結尾的，請點選[FILE] [SAVE AS...]，然後選擇自己放程式的目錄，輸入檔案名稱之後即可：



如何執行這個程式呢？直接開啟剛才的 HelloWorld.py，在編輯器裡面選取[Run] [Run Module]，結果就會出現在IDLE介面之中，如下圖的最後兩行：



這種方式，是直接在 Python 的環境之下執行，如何我們要在電腦作業系統的環境下執行該如何做呢？就必須在DOS視窗中下指令 “python helloworld.py” 來執行，結果如下：



要退出DOS畫面，輸入 exit 即可。

## Python 語言基礎

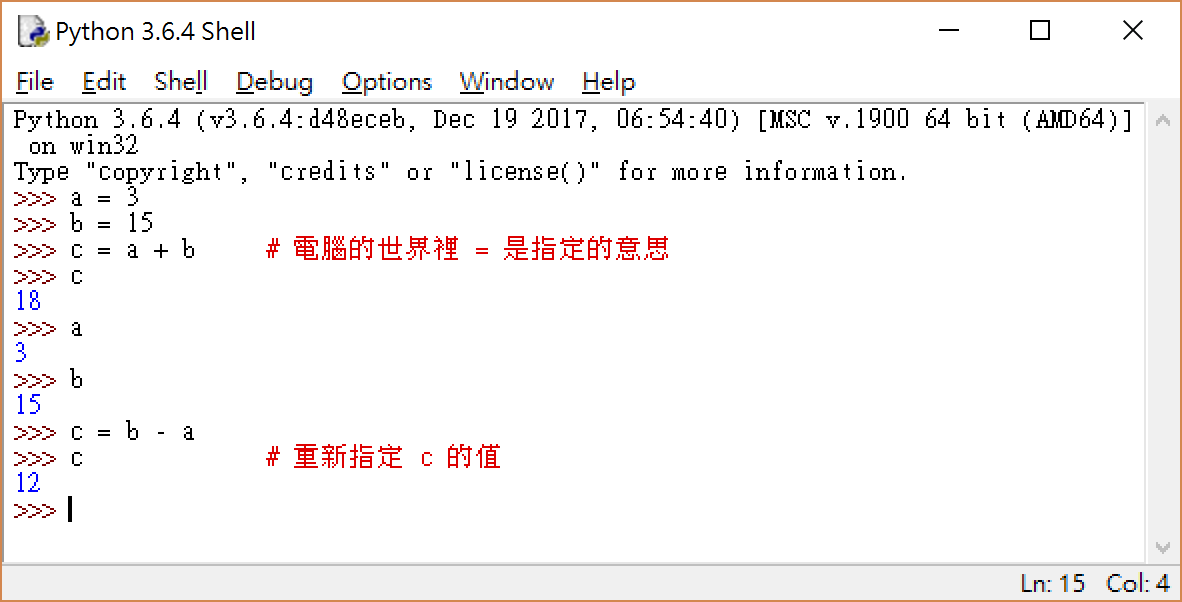
物件是 Python 語言最基本的概念之一，在 Python 中的一切都是物件，常用的內建物件如數字、字串、列表、字典、元組、檔案、檔案、集合等等，連函數都是物件的一種。除此之外，還有大量的標準庫物件和擴展庫物件。Python 預設會安裝標準庫，但需要匯入之後才能使用其中的物件；擴展庫物件則需要先安裝擴展庫，然後再匯入來使用其中的物件。

### 變數、運算子與運算式

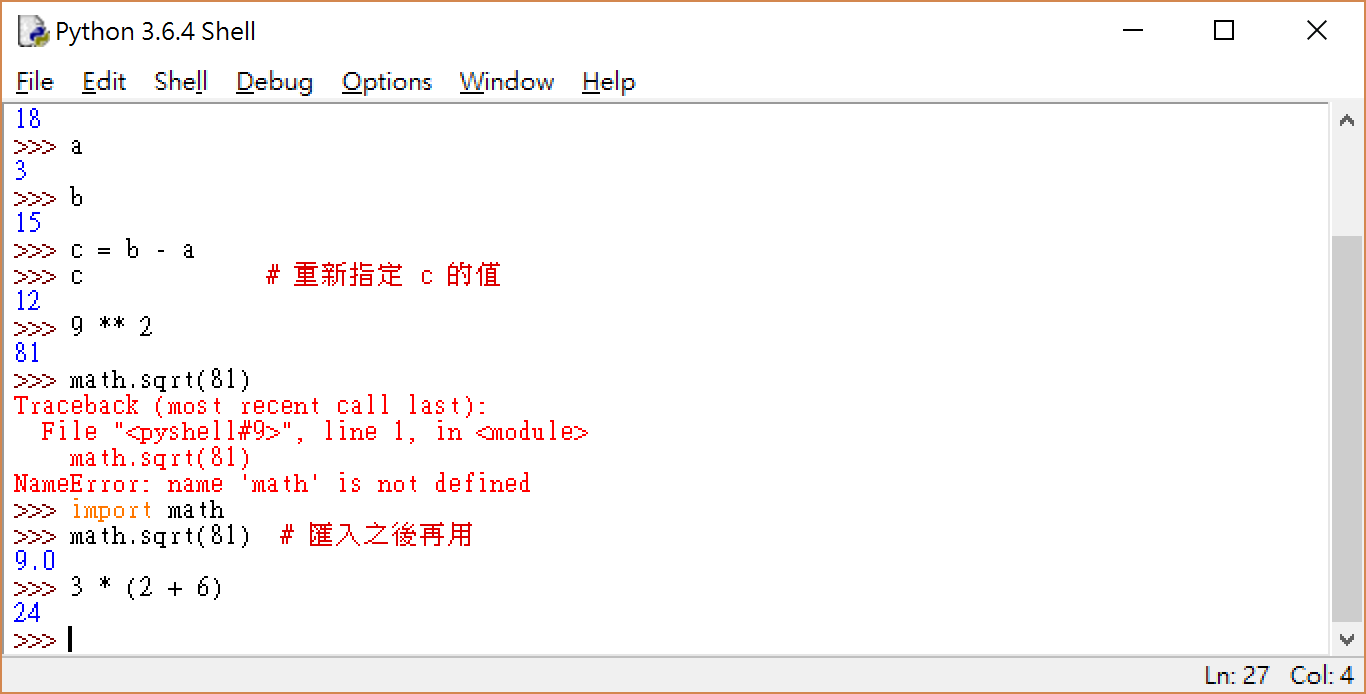
電腦語言是由數學算式演化過來的，所以一切都可以從數學來看待，如果有一個數學式子是這樣寫的：

A + B = C

這整個式子我們就稱為運算式，其中A、B、C是變數，用來存放Python的物件(例如數字)， “+” 與 “=” 就是運算;子，了解這個之後，我們就可以寫程式了：



這裡面出現的加減乘除運算，是最基本的，Python 已經將最基本的物件都匯入了，所以我們無須再做匯入的動作，但是如果是要做平方根的運算，它則是放在 math 這個標準庫裡面，我們必須使用 import 將它匯入才能使用：



Sqrt() 是 math 庫的函數之一，如果我們常常會使用到 sqrt() 函數，在匯入 math 之後，每次都要寫成 math.sqrt ，其實有點麻煩。

>>> import math

>>> math.sqrt(81) # 匯入之後再用

9.0

>>> 3 \* (2 + 6)

24

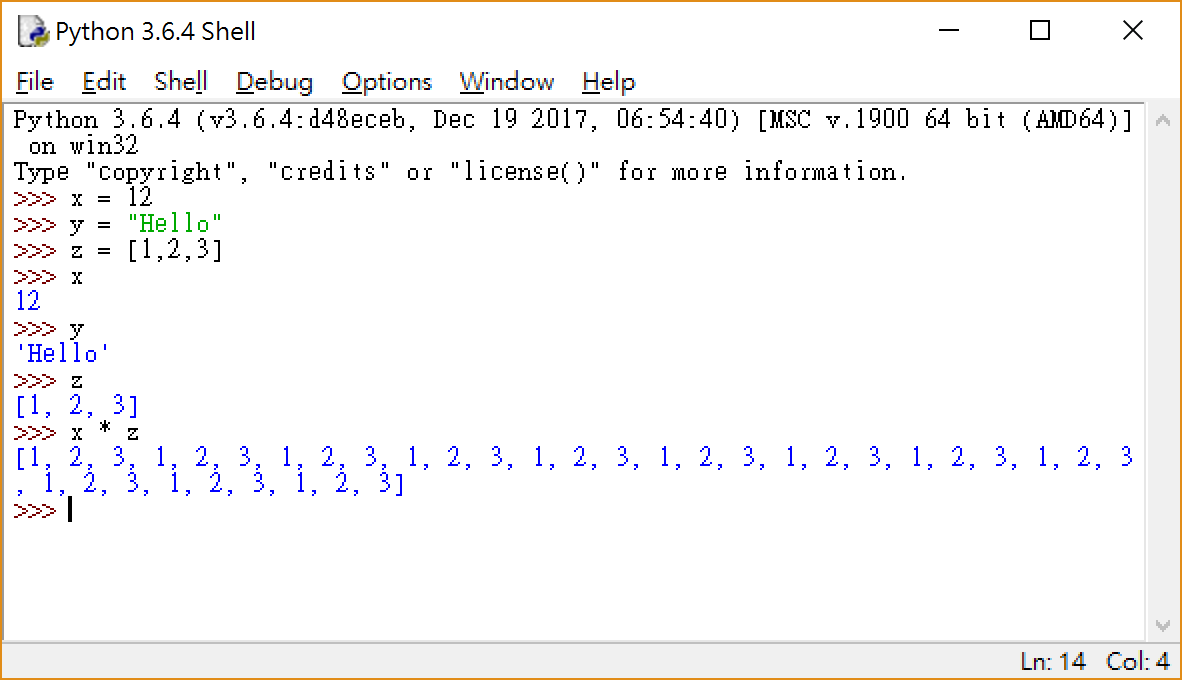
>>> from math import sqrt

>>> sqrt(81)

9.0

如果我們用 from math import sqrt 指定把 math 中的 sqrt() 匯入，那麼在使用的時候就比較省事了。

前面所提的所有變數的值都是數字以及字串(例如 “Hello World”)，我們再介紹Python 第三種很有用的物件叫做列表，而列表是用 [ … ] 來包裝。



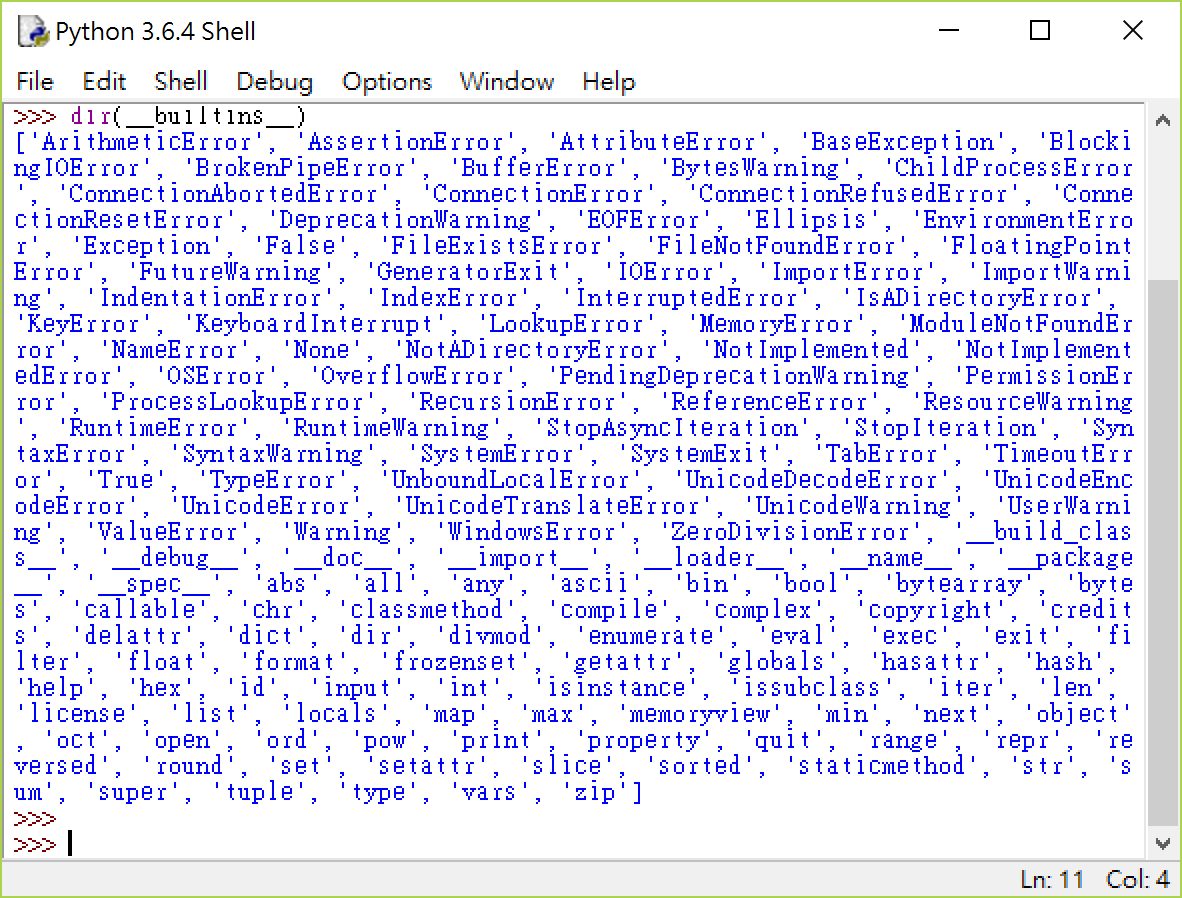
PS: 數字跟列表可以相乘，結果會跑出 12 個 [1,2,3]。但是不同型態不能相加。

### 常用的內建函數

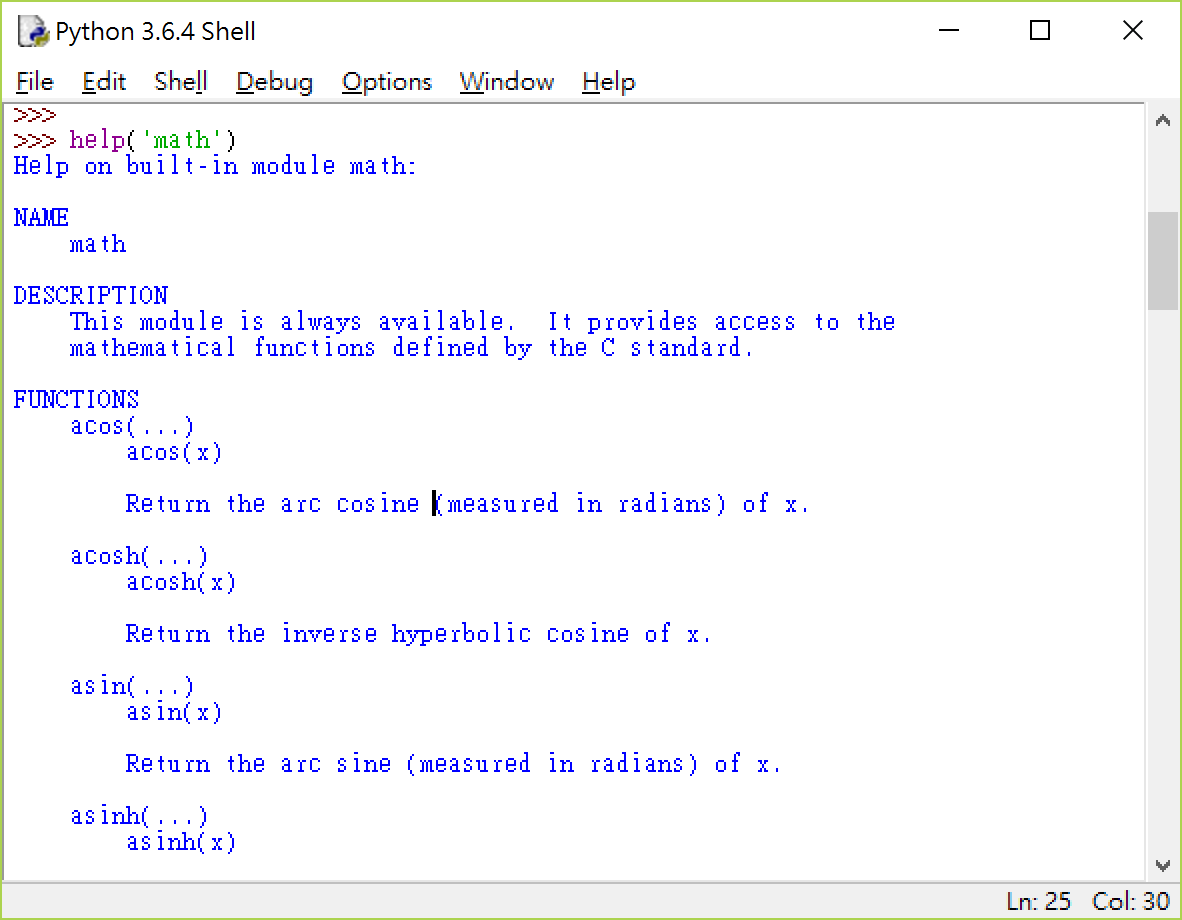
數學函數：y = f(x)

電腦語言函數：y = function\_name(x) 🡺 有傳回值，proc(x) 🡺 無傳回值

可以透過 dir() 函數，列出所有 Python 的內建函數：



而 help() 函數，可以查看某個模組或是函數的用法，不需要匯入函數就可以查看該模組的說明文件，例如：help(‘math’)，參考下圖：



我們就來使用 dir() 與 help() 函數來舉個例子，如果使用者想知道 math 模組裡面有哪些函數可以用，然後進一步想知道其中的 sin 函數的用法：

>>> import math

>>> dir(math)

['\_\_doc\_\_', '\_\_loader\_\_', '\_\_name\_\_', '\_\_package\_\_', '\_\_spec\_\_', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']

>>> help(math.sin)

Help on built-in function sin in module math:

sin(...)

sin(x)

Return the sine of x (measured in radians).

>>>

有三個常用的內建函數，很適合用來寫遊戲，就是 max()、min()、與 sum()，如果有 a b c 三顆骰子，分別擲出 2 4 6 三個點數，Python 要如何找出最大值與最小值呢？又要如何算出三顆骰子的總和？

**隨堂操作練習**

在互動環境中，查詢 math 標準庫中有哪些與三角函數相關的函數，並比較 cos() 與 acon() 的差異

答：

用Python 表示會是這樣：

>>> a = 2

>>> b = 4

>>> c = 6

>>> dice = [a,b,c]

>>> dice

[2, 4, 6]

>>> min(dice)

2

>>> max(dice)

6

>>> sum(dice)

12

>>>

上面的例子有一個缺點，就是骰子a永遠是2，骰子b永遠是4…，事實上骰子應該是隨機的，所以再介紹一個更好用的函數，就是產生隨機數字的函數：randint()，這個函數放在 random 標準庫裡面。如果要取 1 到 6 之間的亂數，可以用 randint(1,6) 來達成，重寫上面的例子：

>>> from random import randint

>>> a = randint(1,6)

>>> b = randint(1,6)

>>> c = randint(1,6)

>>> dice = [a,b,c]

>>> dice

[4, 2, 3]

>>> print(min(dice),max(dice),sum(dice))

2 4 9

>>>

如果同樣的例子，不是擲三顆，而是擲10顆骰子呢？有一個簡單的方法，就是直接在列表裡面放十個骰子的值，就可以省去很多變數：

>>> dice = [randint(1,6), randint(1,6), randint(1,6), randint(1,6), randint(1,6), randint(1,6), randint(1,6), randint(1,6), randint(1,6), randint(1,6)]

>>> dice

[2, 4, 6, 1, 5, 3, 2, 1, 6, 3]

>>> print(min(dice),max(dice),sum(dice))

1 6 33

>>>

這種寫法很難看，dice = [ … ] 太長，當然，沒有人會這樣寫，這違背了 Python 簡潔的特性。我們可以在列表裡面使用 for 迴圈來達成同樣的目的：

>>> dice = [randint(1,6) for i in range(10)]

>>> dice

[6, 4, 2, 2, 4, 2, 5, 4, 3, 1]

>>> print(min(dice),max(dice),sum(dice))

1 6 33

>>>

這種寫法可以先記起來，後面會解釋這種語法。有了這樣的語法，就算要擲100顆骰子也沒有問題。如下：

>>> dice = [randint(1,6) for i in range(100)]

>>> dice

[5, 5, 1, 1, 4, 2, 5, 1, 5, 4, 1, 1, 2, 1, 1, 3, 4, 2, 3, 3, 5, 6, 5, 5, 4, 3, 6, 4, 3, 2, 4, 4, 6, 2, 5, 4, 3, 1, 5, 1, 2, 5, 5, 5, 2, 5, 3, 1, 6, 2, 2, 5, 3, 2, 6, 4, 2, 6, 6, 2, 2, 4, 5, 3, 3, 6, 2, 4, 1, 6, 1, 2, 1, 1, 4, 4, 4, 1, 1, 2, 2, 4, 2, 1, 1, 1, 2, 5, 2, 6, 5, 2, 3, 1, 4, 5, 6, 3, 5, 2]

>>> print(min(dice),max(dice),sum(dice))

1 6 325

>>>

隨堂測驗

題目：一副撲克牌有52張牌，有四種花色，每種花色有13張牌，從每種花色中隨機抽出一張牌，找出這四張牌的最大數、最小數、與總和。

答：>>> from \_\_\_\_\_\_\_ import \_\_\_\_\_\_\_\_\_

>>>

>>>

>>>

### 常用運算子與運算式

Python 是純物件導向語言，熟悉物件導向的人應該知道，物件由資料和行為兩個部分組成。對於初學者而言，可以先跳過這個部分，只要知道運算子也屬於表現物件行為的一種形式。

一般我們常用的運算子包含算術運算子(大家都學過數學)、關係運算子(大家都很愛比較)、邏輯運算子(大家都學過中文)以及位元運算子(比較難，因為大家不見得都學過電腦)。當然，Python 還提供了其他類別的運算子，先不提，我們就從最基礎的前三個開始介紹。

#### 算數運算子

直接看例子：

>>> x = 12

>>> y = 4

>>> x + y

16

>>> x - y

8

>>> x \* y

48

>>> x / y

3.0

>>> z = 5

>>> x / z

2.4

>>> x // z #直接求整商

2

>>> x % z #直接求餘數

2

>>>2 \*\* 3 # n 次方

8

想想我們學過的三種物件的資料類別，數字、字串、以及列表，數字做上面的那些運算絕對沒有問題，但是文字和列表呢？有些就不行了，但是至少相加應該沒有問題：

>>> a = 'abcd'

>>> b = '1234'

>>> a + b

'abcd1234'

>>> c = ['x','y','z']

>>> d = [5,6,7]

>>> c + d

['x', 'y', 'z', 5, 6, 7]

>>>

除了數字、字串、列表之外，現在我再多介紹一種資料類別，這也是我們在國中數學裡面學過的，就是集合。同樣的，集合的表示方法與數學相同，用 { … } 表示，而集合的運算有聯集(|)、交集(&)、差集(-)、與對稱差集(^)：

你和朋友一起玩了三場槍戰遊戲，第一場出賽的有 a b c d e 五個人，第二場出賽的有 a c e f 四個人，第三場出賽的有 a b g 三個人，我們可以用Python 的集合來表示：

>>> game1 = {'a','b','c','d','e'}

>>> game2 = {'a','c','e','f'}

>>> game3 = {'a','b','g'}

請問這三場遊戲總共有哪些人出賽？有哪些人一直都有參賽？

>>> game1 | game2 | game3 #聯集,三場遊戲總共有哪些人出賽

{'b', 'c', 'd', 'g', 'f', 'a', 'e'}

>>> game1 & game2 & game3 #交集,有哪些人一直都有參賽

{'a'}

第一場有出賽，但是第三場沒有出賽的有那些人？第一場與第三場遊戲，只參加一場的有那些人？

>>> game1 - game3 #差集,第一場有出賽，但是第三場沒有出賽的有那些人

{'c', 'd', 'e'}

>>> game1 ^ game3 #對稱差集,第一場與第三場遊戲，只參加一場的有那些人

{'c', 'd', 'g', 'e'}

不用害怕數學跟電腦語言，其實只不過是把我們平常講的中文用另一種符號表示而已。

#### 關係運算子

妳比我大三歲，我薪水比你高，大家都會比較，用 Python 來表示如下：

>>> yourage = 19

>>> myage = 16

>>> yourage > myage

True

>>> yourage <= myage

False

>>> yourage == myage

False

>>> yourage != myage

True

>>>

電腦的回答只有兩種，真(True)或假(False)。

#### 邏輯運算子

And、or、not 這三種邏輯運算子通常會和關係運算子合併使用，你今年19歲，我今年16歲，他今年20歲。妳比我大而且我比他大嗎？這句中文裡面的 “比” 就是關係運算子，出現的 “而且” 就是邏輯運算子中的 “and”。妳比我大或者我比他大嗎？其中 “或者” 就是 “or”。我不(沒有)比他大嗎？其中的 “不(沒有)” 就是 “not”。

>>> yourage = 19

>>> myage = 16

>>> hisage = 20

>>> (yourage > myage) and (myage > hisage)

False

>>> (yourage > myage) or (myage > hisage)

True

>>> not (myage > hisage)

True

>>>

### 人機對話基本介面

到目前為止，在 Python 的互動環境中，都是我們對 Python 說話而 Python 回答，那是否可以反過來？ Python 問我們問題我們回答。當然可以，內建函數 input() 就可以做到。我們就讓 Python 問我們年紀好了。

>>> input('你今年幾歲:')

你今年幾歲:16

'16'

>>> x = 16

>>> x

16

>>>

Python 的 input() 很單純，它問的問題就當成一個字串寫在()裡面，我們回答的答案，也會被當成字串，對 input() 而言，無論我們輸入的是數字還是文字，都會被當成字串，上面的例子很清楚，我們回答 16 原本是個數字，但是後面的輸出則是 ‘16’，跟x 比較一下就知道不同。

因此，我們需要另一個函數幫我們轉換，就是eval()，順便可以用 type() 函數查看一下資料類別。請看下面的例子：

>>> age = input('How old are you? ')

How old are you? 20

>>> type(age)

<class 'str'>

>>> eval(age) # 把age 轉成數字之後輸出，並不影響age本身

20

>>> age

'20'

>>> age = eval(age) # 轉成數字之後再指派給 age

>>> age

20

>>>

跟 input() 相對應的函數，就是 print()