給自學者的Python教學(1) : 該如何理解「程式語言」？

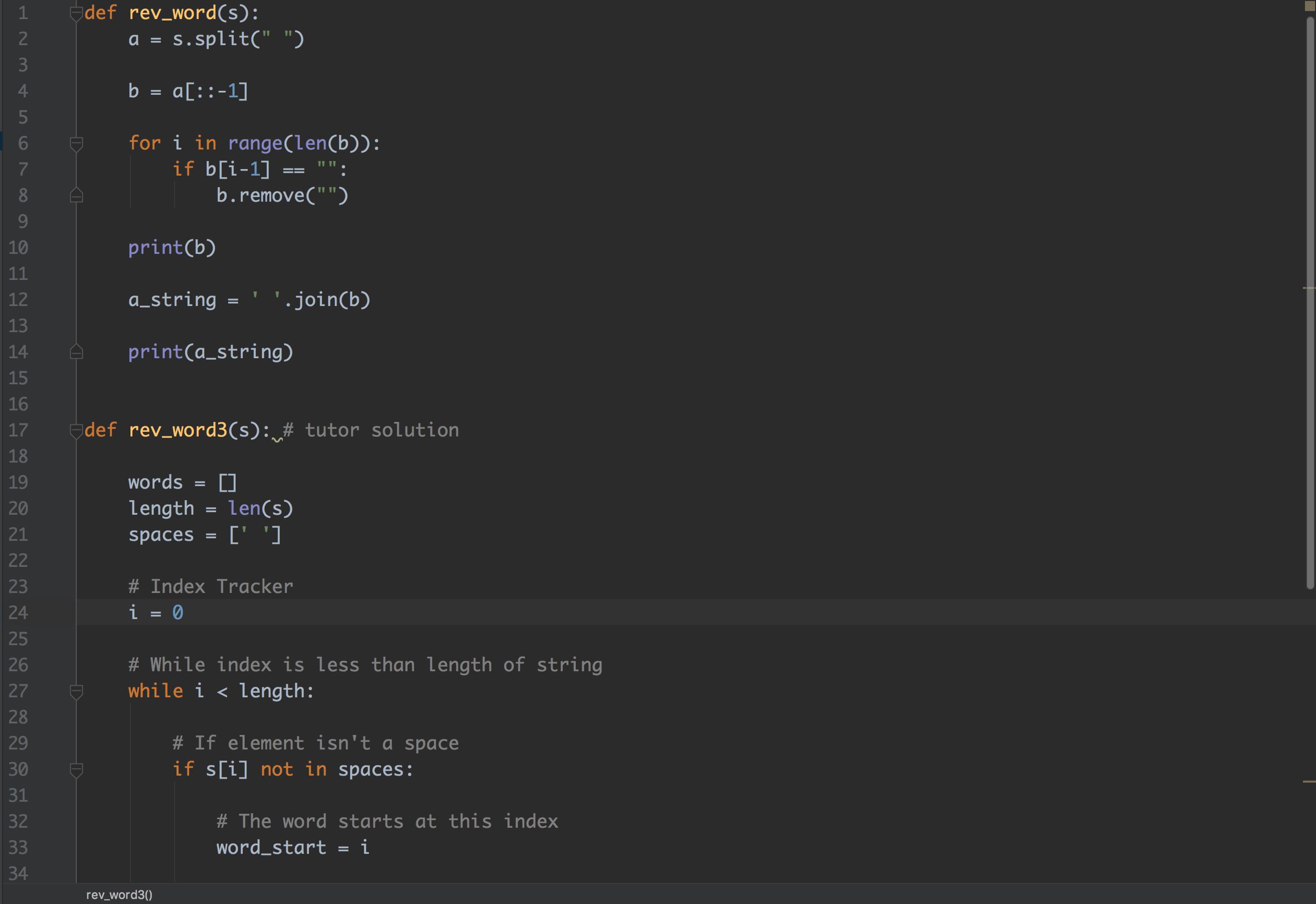
*程式語言或是構成其的程式碼，一直給人高深莫測的感覺，都是一堆電腦技術狂人在三四個螢幕或筆電前，手指飛快地輸入某些看起來很似密碼一樣的code。  
所以，程式語言或是程式在真的學習過程中又是怎麼一回事呢？*

**程式就像是菜譜**



***對，程式不是很高深的東西，就跟在書店看到的菜譜一樣。***

簡單來說，程式語言就是給電腦理解的語言。人們想要通過電腦去處理事情，自然就要「告知」電腦，你想要電腦做的是什麼。



所以，出現了一條條像上圖的指令。

於是，人們在電腦上用程式碼寫出一條條指令，來告訴電腦要做什麼。  
可是，對於不懂得煮菜的人來說，對於菜譜中的一堆材料、步驟也就跟程式語言一樣，兩眼一黑。但在懂得煮菜的人看到時，就一句「哦…就這樣呀」。就如：



當你要做梅菜扣肉時，你就會跟著上面的步驟來做。先煮肉到5分、再加上老抽、準備薑蔥…等等的步驟。你會知道經過這樣的一個個順序，你就可以煮出一個可能好吃的梅菜扣肉。當然，「可能好吃」是因為可能由於你經驗不足、配料調味的份量錯了等等原因引起。

那寫程式呢？  
同樣地，我們要寫出一個程式，就如煮一道菜一樣。我們給**電腦放入一條條指令（程式碼）**，電腦就會按照程式語言的規則來執行，達到程式員所想建立的效果。所以，如果我們了解程式語言中的一條條指令的意思與其語法，就可以看懂甚至開始寫下一點程式，就如你看懂菜譜後，就可以開始準備材料與試著煮菜囉。

***那學一種語言是不是要花很多時間？***

其實不會，可以老實說，因為程式語言是給電腦看的，所以語言的規則（語法）是真的不會太難，至少比自然語言要簡單多了。因為電腦不可能接受太過抽像化的資訊，如「我覺得這個紅色…唔…可以再紅一點」這種語句，電腦是很不能解理什麼叫「再紅一點」。可是，這一句我們卻可以大概明白，相信這也是人腦利害的地方。  
可是，同樣地要怎麼寫出高效、優雅的程式，就如做出色香味俱全的菜色一樣，需要很多的學習與練習。寫出來的程式，也會有bug（錯誤），就如你煮菜加錯了調味的份量一樣，只能通過不斷調整、修改，來消除bug了。*畢竟，會煮菜的很多，可是卻很少五星級大廚。*

***那為什麼有這麼多不同的程式語言？***

簡單來說，就是為了不同的需求與科技進步。  
在最早期，電腦的程式語言，是***機器語言***，也就是二進制。通過0和1，也就是Yes和No來控制電腦的電路開閉合，來達至給電腦輸入指令，其亦被標籤為低階語言。可是，如果要每次都需要輸入一大堆0001101011010，那寫一個程式也實在是太痛苦了，於是，人們創出***組合語言***。組合語言使用較簡單的字串和語法來取代機器語言。但是在漸漸的出現的不同裝置中，不同的組合語言只對應著不同的機器語言指令集，也就是一種組合語言是只可用於某電腦系統結構，那在不同平台之間要移植就變得非常麻煩。

***所以就有高階語言的出現。***

Python，也是高階語言的一種。那什麼是高階語言呢？根據維基百科：

***高階語言（High-level programming language）是高度封裝了的程式語言，與低階語言相對。它是以人類的日常語言為基礎的一種編程語言，使用一般人易於接受的文字來表示，使程式編寫員編寫更容易，亦有較高的可讀性，以方便對電腦認知較淺的人亦可以大概明白其內容。***

對的，現在我們學習的程式語言多是高階語言，如C、C++、Python、Java等等，都是更易被大眾所理解，也可以容易地移植到不同的平台中使用，就如Python可以在Windows、MacOS、Linux等不同平台運行，使開發成本跟學習難道亦大大降低。

因為這些不同的高階語言，使我們可以得到很多不同的程式撰寫方法，不同語言之間亦有很多不同的功能與特性，甚至是特地為某些功能而創造程式語言。如Swift的語言應用都是在寫Ios App上，Python可以很好的應用在大數據與機械學習上等等。

不過，不同的程式語言的複雜度與功能性差異，可能會使得撰寫程式的難度亦不一樣，所以要學習的程式語言，最好要選是你想進入或嘗試的領域中會用到的。如寫網頁要會Html、Css、Javascript等等。

另外，可以參考這篇文章：[世界上的程式語言已這麼豐富，為什麼我們還需要更多？](https://buzzorange.com/techorange/2014/06/23/why-does-the-world-need-more-programming-languages/)

***說了這麼多，那Python又是怎麼樣的程式語言？***

根據維基百科：

*Python，是一種物件導向、直譯式的電腦程式語言。它包含了一組功能完備的標準庫，能夠輕鬆完成很多常見的任務。它的語法簡單，與其它大多數程式設計語言使用大括弧不一樣，它使用縮排來定義語句塊。  
Python是完全物件導向的語言。函式、模組、數字、字串都是物件。並且完全支援繼承、重載、衍生、多重繼承，有益於增強原始碼的複用性。*

唔…是不是都看不明白？對，因為暫時大家都不需要看的明白。  
*更讓你有理由學習Python的原因*，可以是：  
1.它是MIT的必修語言。  
2.它優雅、簡潔，不需要太多的符號作為語法規則，如使用「；」、「｛｝」等。（代表你可以少打很多東西！）  
3.它有大量的資源可以學習，在Udacity、Udemy、Coursera都在Python的課。亦可以在容易地在書店買到用Python寫成的資源結構、演算法等電腦科學相關基礎課程的書。相信我，***資源永遠是自學不可或缺的條件。***  
4.Python可以使用在開發桌面型程式、網頁後端、大數據方面等等。

*沒有什麼是特別利害，可以應付一切的程式語言。很多時候，在不同場景與需求，就需要學習不同的程式語言。不過，程式語言的規則多是不會相差甚遠，當好好的學習了一門語言之後，相信再學習其他程式語言也不會是困難的事。更重要的是，****運用語言作為解決問題的工具，才可以發揮語言的功用。***

# 給自學者的Python教學(0) : 如何安裝Python(Mac/Windows)

[](https://medium.com/@ChunYeung?source=post_page-----126f8ce2f967----------------------)

[YC](https://medium.com/@ChunYeung?source=post_page-----126f8ce2f967----------------------)

Follow

[Oct 21, 2017](https://medium.com/@ChunYeung/%E7%B5%A6%E8%87%AA%E5%AD%B8%E8%80%85%E7%9A%84python%E6%95%99%E5%AD%B8-1-%E5%A6%82%E4%BD%95%E5%AE%89%E8%A3%9Dpython-126f8ce2f967?source=post_page-----126f8ce2f967----------------------) · 4 min read

*因為寫這篇文章的目標是希望讓初學者快速進入撰寫程式的世界，就不會特意說明一些要用指令列才能完裝的方法。*

***Mac OS 安裝***

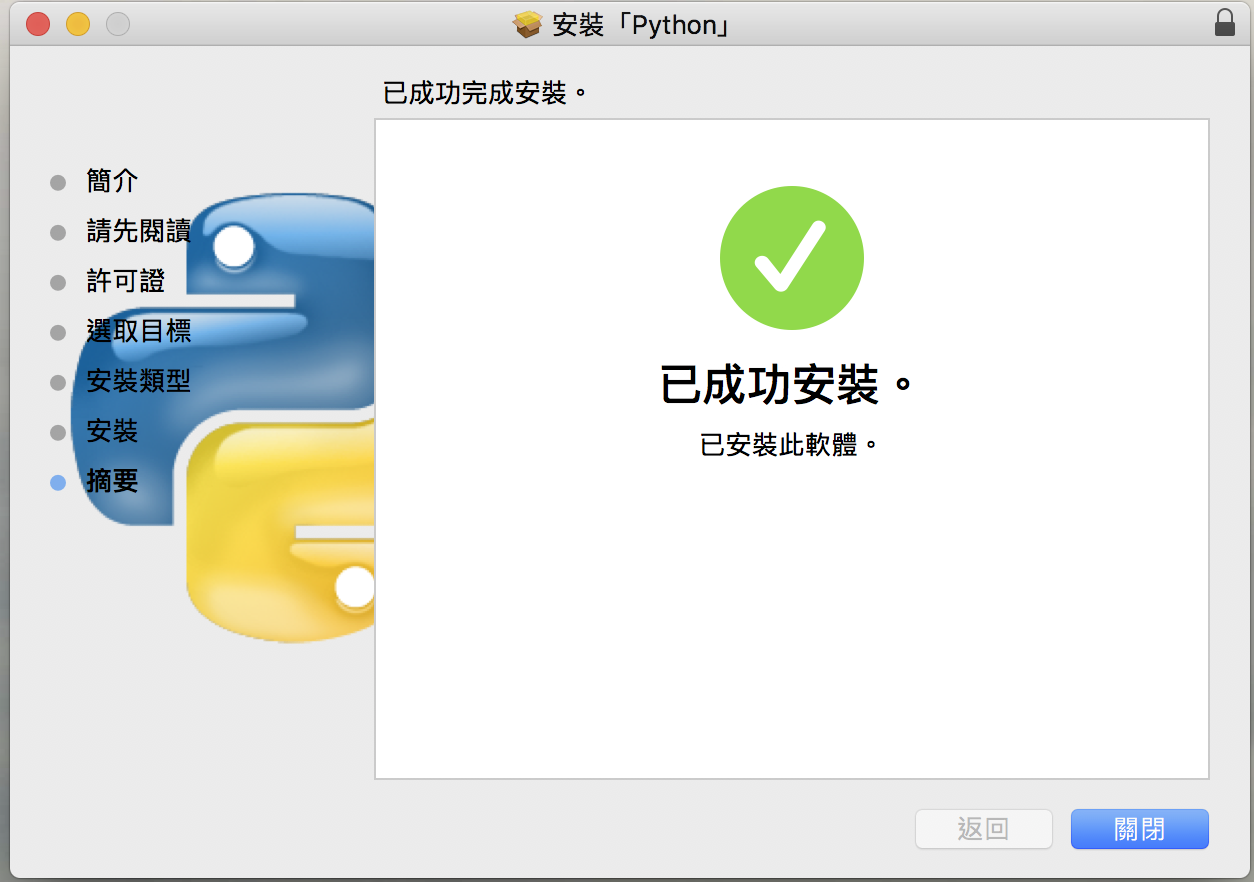
在Mac OS電腦上，若你的系統版本為OS X 10.8- 10.12，是預載了Python 2.7的。所以，簡單的來說，是不需要另外安裝就可以使用Python。

當然，若你希望安裝最新的Python，可按步驟如下：  
1.進入Python官網的下載頁面: [https://www.python.org/downloads/mac-osx/](https://www.python.org/downloads/%20windows/)

2.查找一個穩定版本，如下圖中的Python 3.6.3–2017–10–03。  
在版本號後沒有附加英數的可以視為穩定版本。  
（如Python 3.6.3rc1–2017–09–19 就是一個測試版本。）



3.運行 .pkg 執行檔，一直按繼續來完成安裝。  
出現以下畫面表示安裝成功。



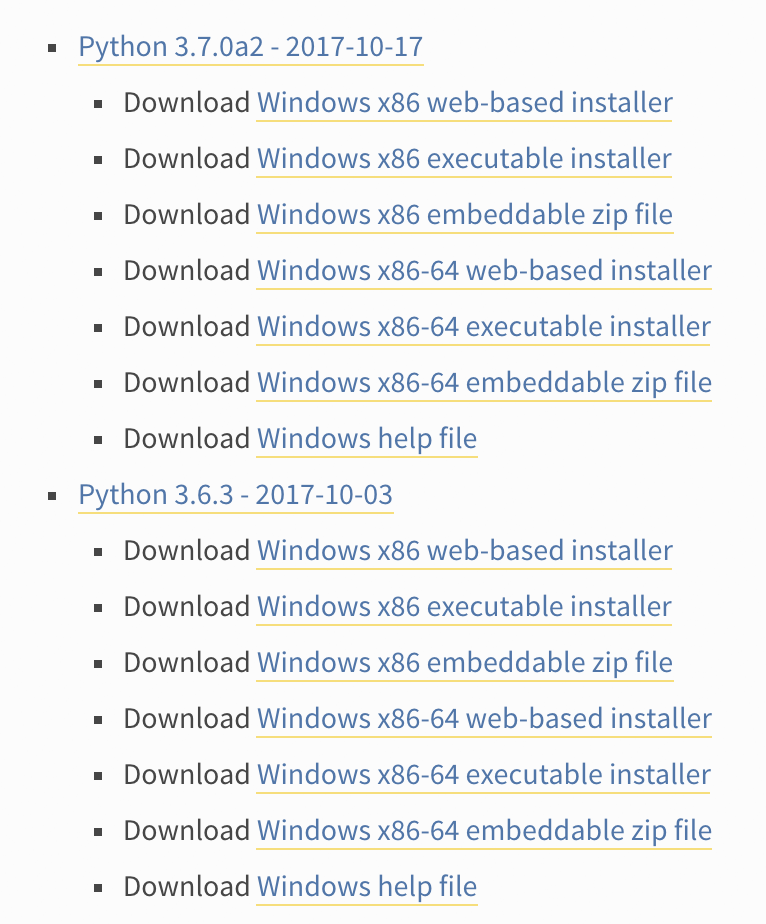
4 .現在我們可以檢查Python是否安裝成功了。  
先按command+space，然後打terminal.app來打開終端機（對，你進入很cool的文字介面了）。然後輸入python3 --version，即可得到你系統的已安裝的Python 版本。

$ python3 --version  
Python 3.6.3

***Windows 安裝***

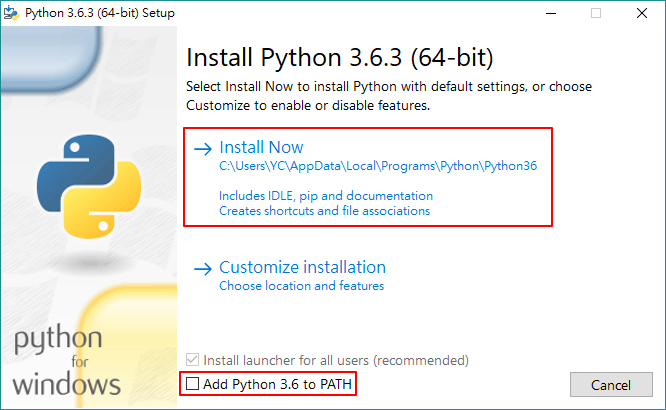
而在Windows上，我們就需要較多的步驟來安裝Python了。  
1.進入Python官網的下載頁面: [https://www.python.org/downloads/ windows/](https://www.python.org/downloads/%20windows/)

2.查找一個穩定版本，如下圖中的Python 3.6.3–2017–10–03。  
在版本號後沒有附加英數的可以視為穩定版本。  
（如Python 3.6.3rc1–2017–09–19 就是一個測試版本。）

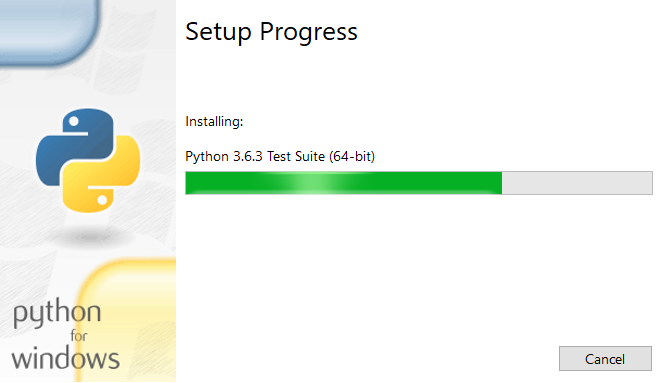


3.看到以上的畫面，點選你需要的版本。如你的Windows是32位元，請下載Windows x86的版本，如果是64位元，請下載Windows x86–64的版本。不管是web-based,executable,embeddable都是可以正常使用的安裝程式。（建議初學者可以使用Python 3.6版本，因為3.6版已經漸漸成為主流，直接學習3.6是不錯的選擇。）

4.直接運行已下載的.exe 執行檔，並按以下程序執行。（以下程序以64位元為準）。

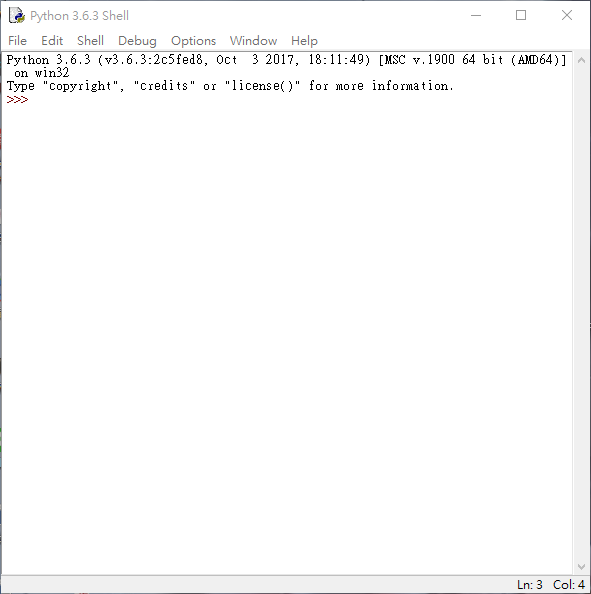


要先勾選Add Python 3.6 to PATH，再點Install Now。



等候安裝完成即可使用。

5.現在就可以運行Python的IDLE來寫一點小程式囉。



給自學者的Python教學(2) :兩種比較簡單的方式來執行Python程式碼

[](https://medium.com/@ChunYeung?source=post_page-----96310c8e585c----------------------)

[YC](https://medium.com/@ChunYeung?source=post_page-----96310c8e585c----------------------)

Follow

[Oct 22, 2017](https://medium.com/@ChunYeung/%E7%B5%A6%E8%87%AA%E5%AD%B8%E8%80%85%E7%9A%84python%E6%95%99%E5%AD%B8-2-%E4%BD%BF%E7%94%A8%E5%85%A9%E7%A8%AE%E6%AF%94%E8%BC%83%E7%B0%A1%E5%96%AE%E7%9A%84%E6%96%B9%E5%BC%8F%E4%BE%86%E5%9F%B7%E8%A1%8Cpython-96310c8e585c?source=post_page-----96310c8e585c----------------------) · 6 min read

***有想過寫了一串程式碼之後，應該要怎樣做才可以讓你寫的心血可以執行和測試嗎？是在隨便任何地方都可以寫程式碼嗎？****(在之後的文章，還是會看到一些平常很少會用到的「字詞」，如xx器，印出什麼，宣告什麼🤔，等等。別擔心，就算暫時你覺得怪異或不明所以，但在之後的文章，你會慢慢了解到這些詞彙在程式設計中的「位置」是什麼。)*

*在****給大家介紹執行Python的方法之前，我們要先認識一下IDE是什麼。***

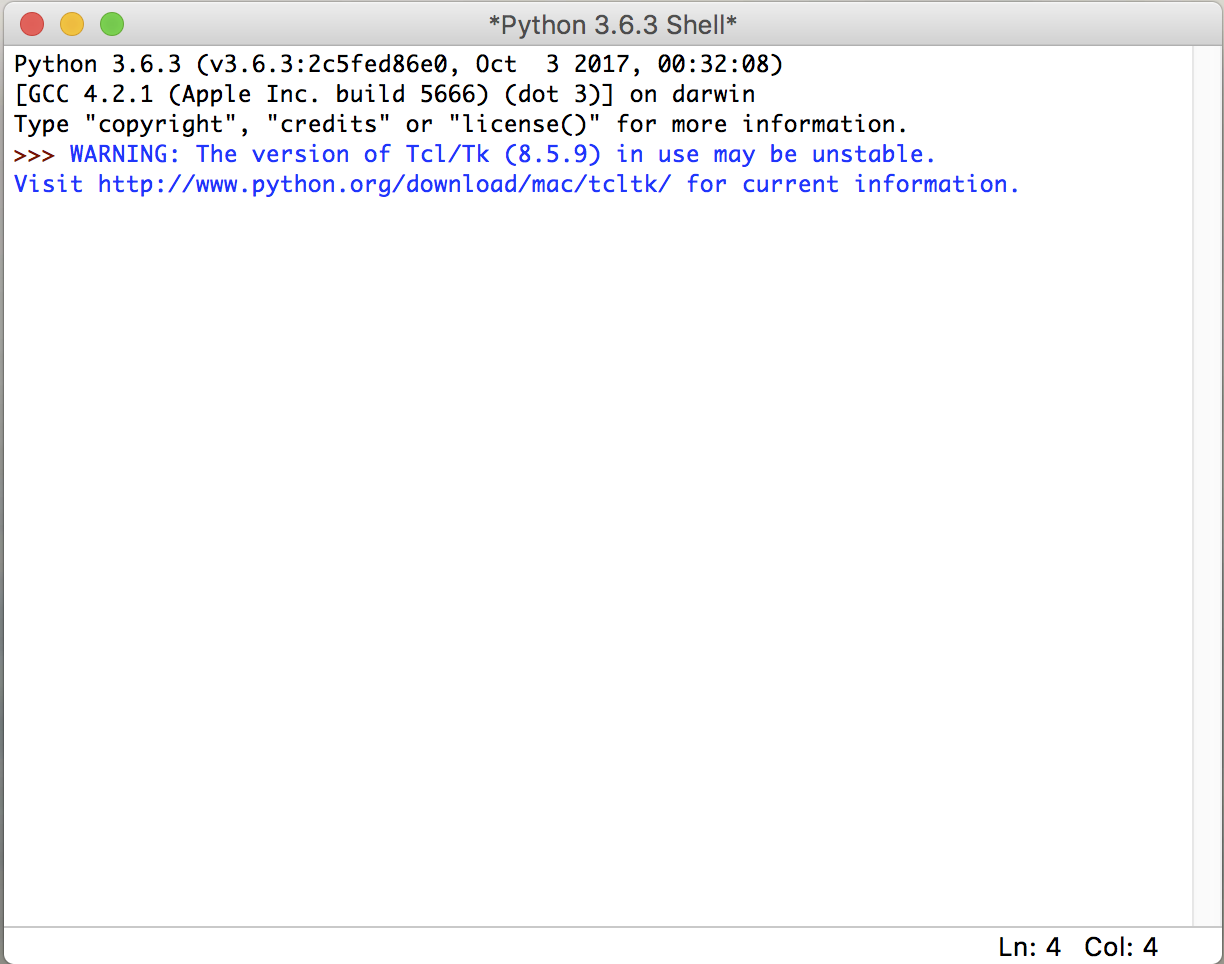
IDE全稱為**整合開發環境**（Integrated Development Environment），是一種可同時作為：

1. 文字編輯器，以編寫程式碼。
2. 編譯器，編譯並將程式碼打包程式。（想詳細了解編譯器是什麼，又在開發程式中扮演什麼角色，建議參考：[維基百科的編譯器條目](https://zh.wikipedia.org/wiki/%E7%B7%A8%E8%AD%AF%E5%99%A8" \t "_blank)）
3. 甚至是設計程式使用者介面的工具，如Xcode（一個撰寫OSX,IOS程式的IDE）就是一個可以設計UI的IDE。

的程式。IDE可以說是一種強力的開發工具，但是**IDE在開發程式過程中卻不是必須的**。我們可以只透過使用文字介面（由terminal進入），甚至在記事本當中就可以進行程式開發，但過程可能就會多了一點步驟。當然，有好用又方便的東西可以幫助開發，我認為是值得去學習使用的。

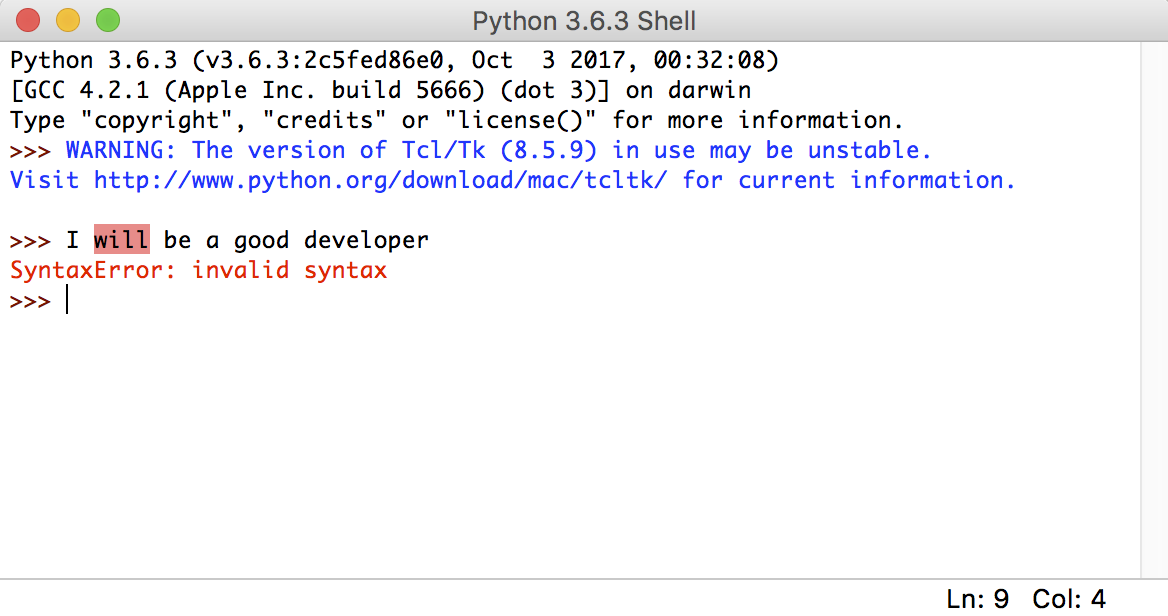
*現在來介紹常用的Python IDE*

**1.Python 內建的 IDLE 交互式介面**



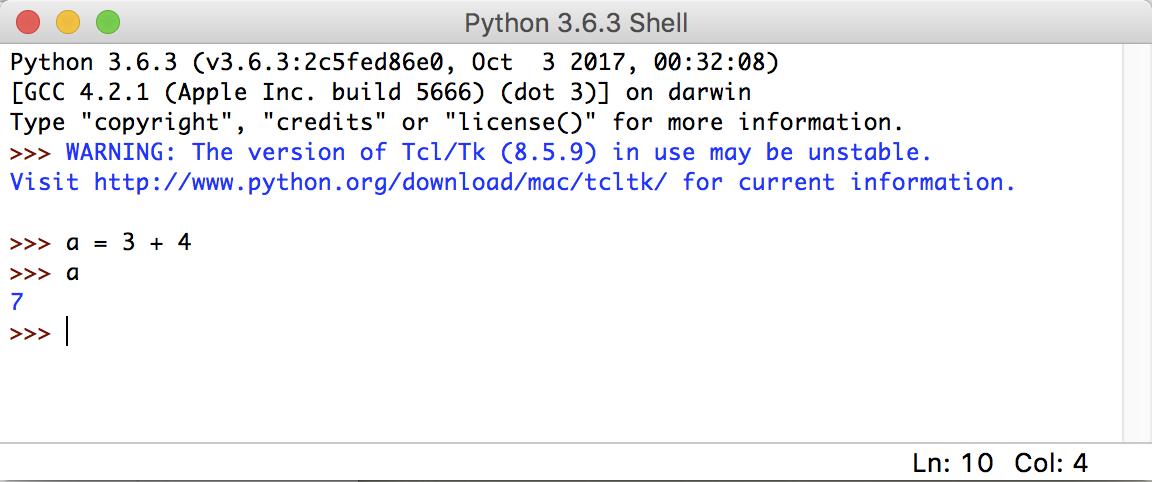
Python的交互式介面──也就是IDLE，是在前文安裝完Python後自帶的程式，他可以簡單地輸入指令後得到結果。當然，它是比較偏向簡單新手向的，不適合用來寫成完整的程式，但是用來練習的話，會是一個非常方便的工具。你可以把它當作是一個Playground，在上面試著任何你想實驗的東西，因為他會馬上顯示出結果或報錯。

例如在IDLE 中輸入 「I will be a good developer」：



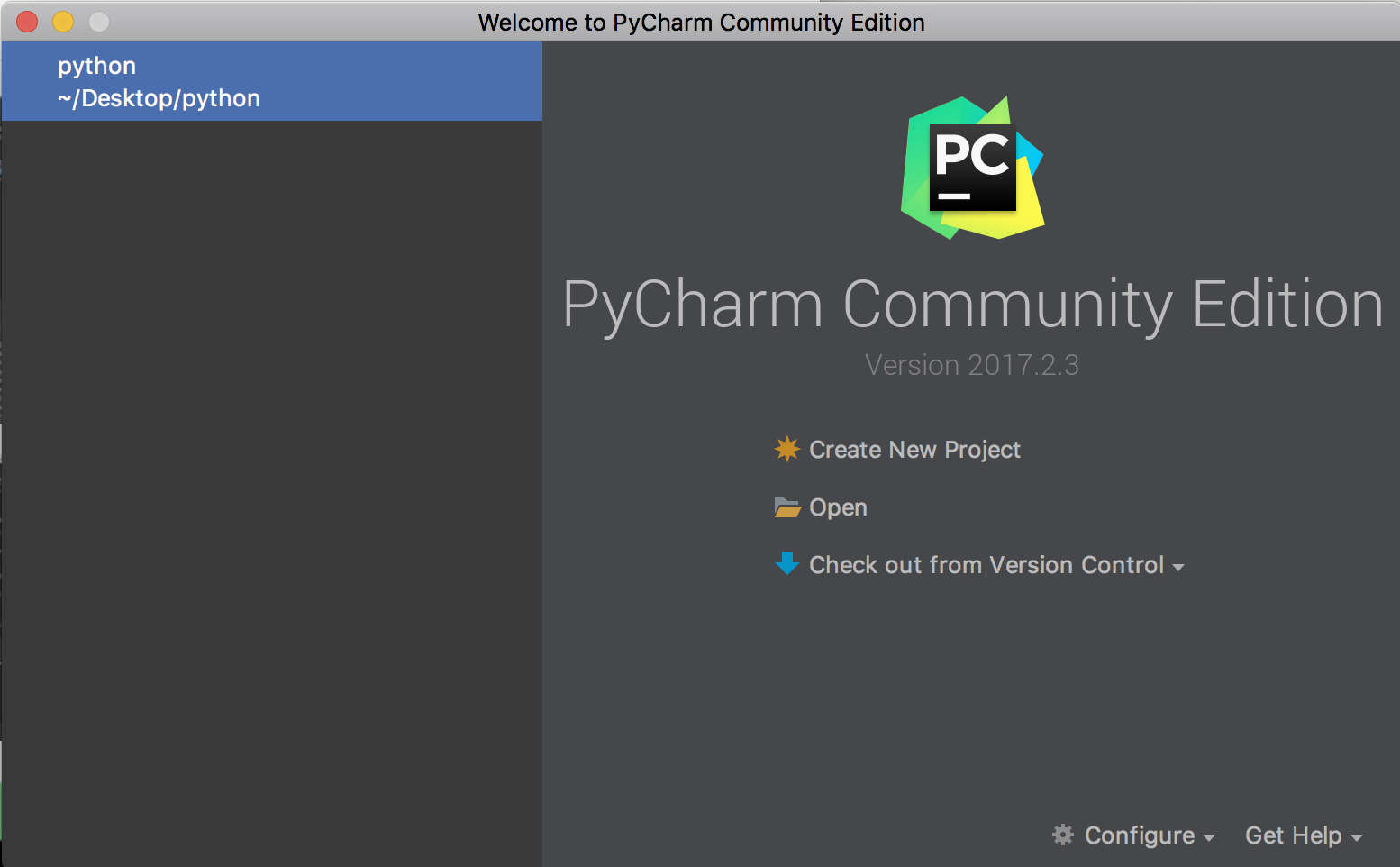
Python馬上就會給你跳出一個紅色的警告，「SyntaxError: invalid syntax」是說「語法錯誤：無效的語法」。對！如**[前文](https://medium.com/@ChunYeung/%E7%B5%A6%E8%87%AA%E5%AD%B8%E8%80%85%E7%9A%84python%E6%95%99%E5%AD%B8-1-%E8%A9%B2%E5%A6%82%E4%BD%95%E7%90%86%E8%A7%A3-%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80-56965d6db498" \t "_blank)**所說，程式語言是有特定的語法，如果你不跟規則來下指令，Python就會好像聽到外星語一樣，不知道你想要他幹嘛。

試著再在IDLE 中輸入 「a = 3 + 4」後return，再輸入「a」後return：



現在就出現a的數值：7 這個答案。（同時，恭喜你完成人生第一行程式囉）可以看到，在IDLE上，我們不需要做額外的動作，就可以做一點簡單的練習了。

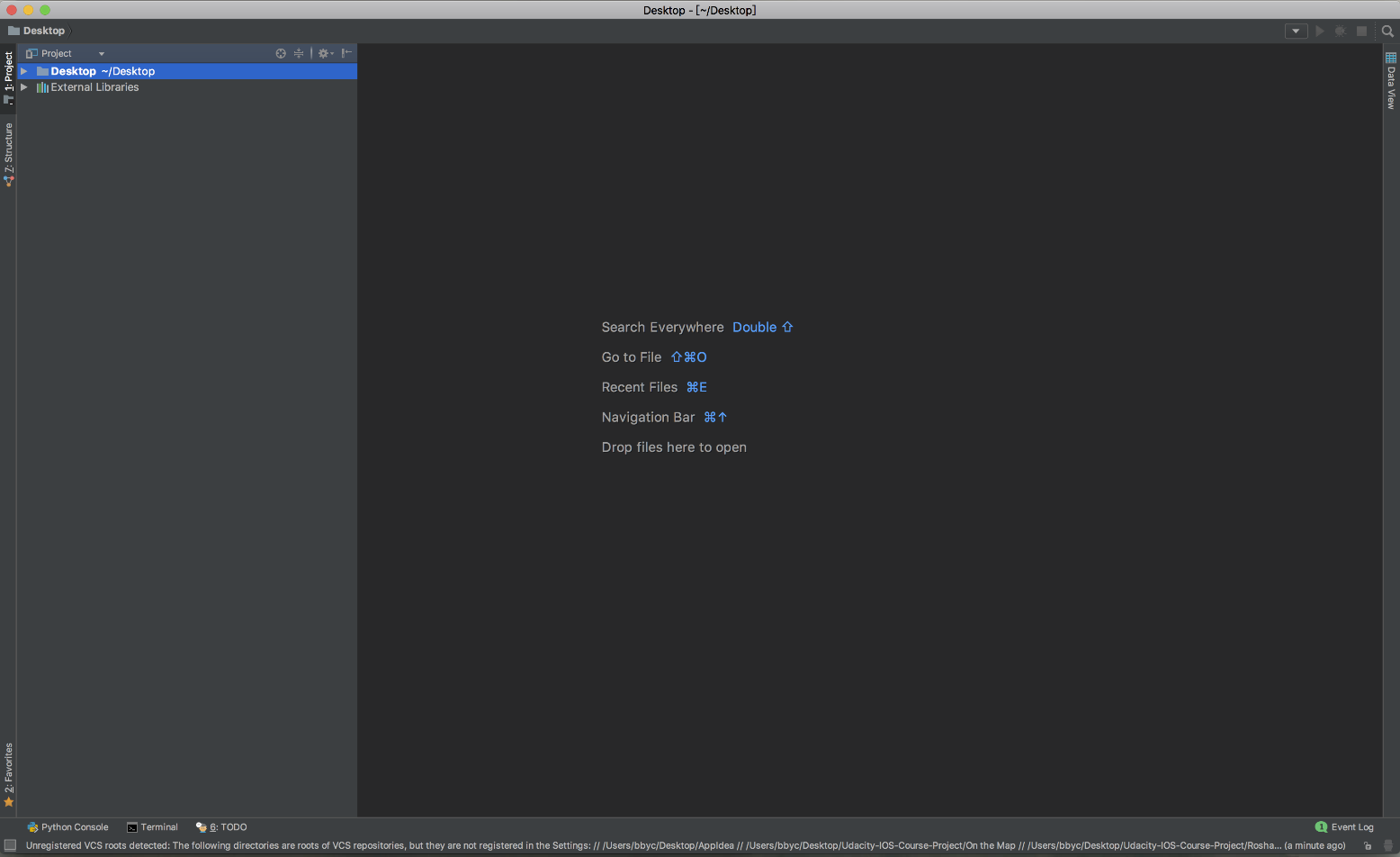
**2.PyCharm CE**



PyCharm CE是一款功能相當強大的IDE，你可以直接在PyCharm CE中創建Python文件、編寫與編譯，並馬上在console中顯示結果。

這是[**PyCharm CE的網頁**](https://www.jetbrains.com/pycharm/?fromMenu)，只要選擇你的電腦系統，再選Community版本下載就可以了。

要使用PyCharm CE可以先按「Create New Project」，選擇你要建立新專業資料夾的位置，再按Create就可以創建新的Python專案了。



在創建專案後，我們只需要按下command+N(MacOS)／alt+Insert(Windows)就會出現選單，再點選python file就可以創建新的py檔。

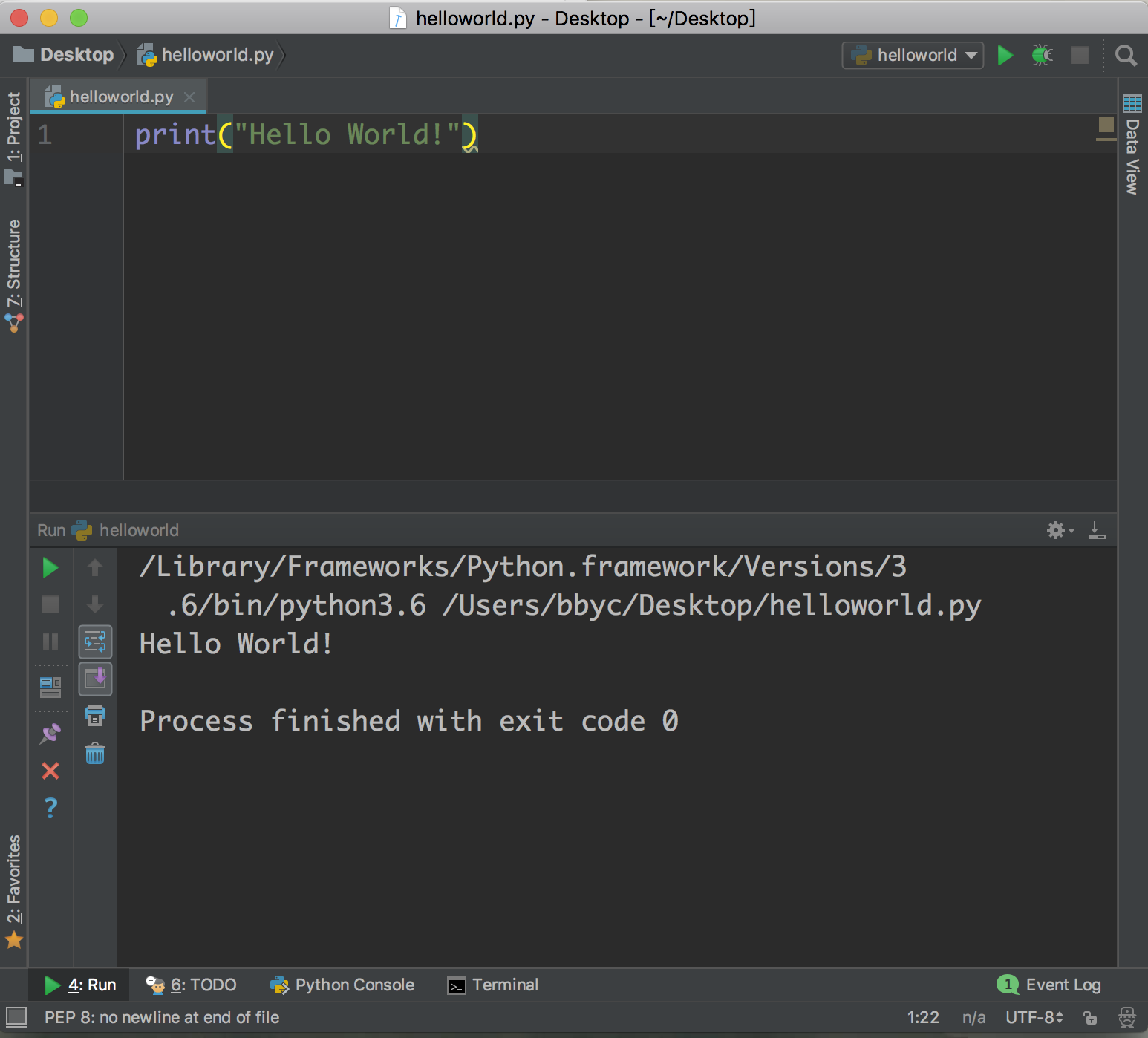
之後，我們就可以開始動手寫下我們第一個程式了！

*來寫個Hello World吧！*

我們先新創一個叫做HelloWorld的檔案。（記得要選python file來創建喔）  
然後，我們在文字輸入區中，輸入：

print(“Hello World!”)

然後按右鍵選單中的*run*來運行程式。應該就會出現如下圖中的情況。



在下面我們就可以看到你事入電腦的文字Hello World!出現在螢幕了！

*現在，你們已經會如何執行你的Python 程式碼了。  
馬上，我們就會進入到學習程式語言語法與規則中。*

等等！先別走！這裡還有專屬於Python 的彩蛋！

在交互式解釋器中，我們可以輸入 import this。然後Python就會印出 「Python 之詩」！這首詩的內容蘊含著Python語言的編程哲學。

The Zen of Python, by Tim Peters

Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren’t special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.  
Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one — and preferably only one — obvious way to do it.  
Although that way may not be obvious at first unless you’re Dutch.  
Now is better than never.  
Although never is often better than \*right\* now.  
If the implementation is hard to explain, it’s a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea — let’s do more of those!

# 一小時Python入門-part 1

Python 是一個簡潔易讀的語言，學習者幾乎可以立刻上手，也適用於大量的商業應用上。目前已超越 C/C++、Java，成為各大學課程中的主流入門程式語言，美國 Top 10 Computer Science (電腦科學) 系所中便有 8 所採用 Python 作為入門語言。

今天就讓我們來試著寫寫看 Python 這個好玩的語言，並利用學習結果寫個簡單的爬蟲程式，能夠在 FB 自動發文、自動洗讚，統計出朋友的留言按讚數!

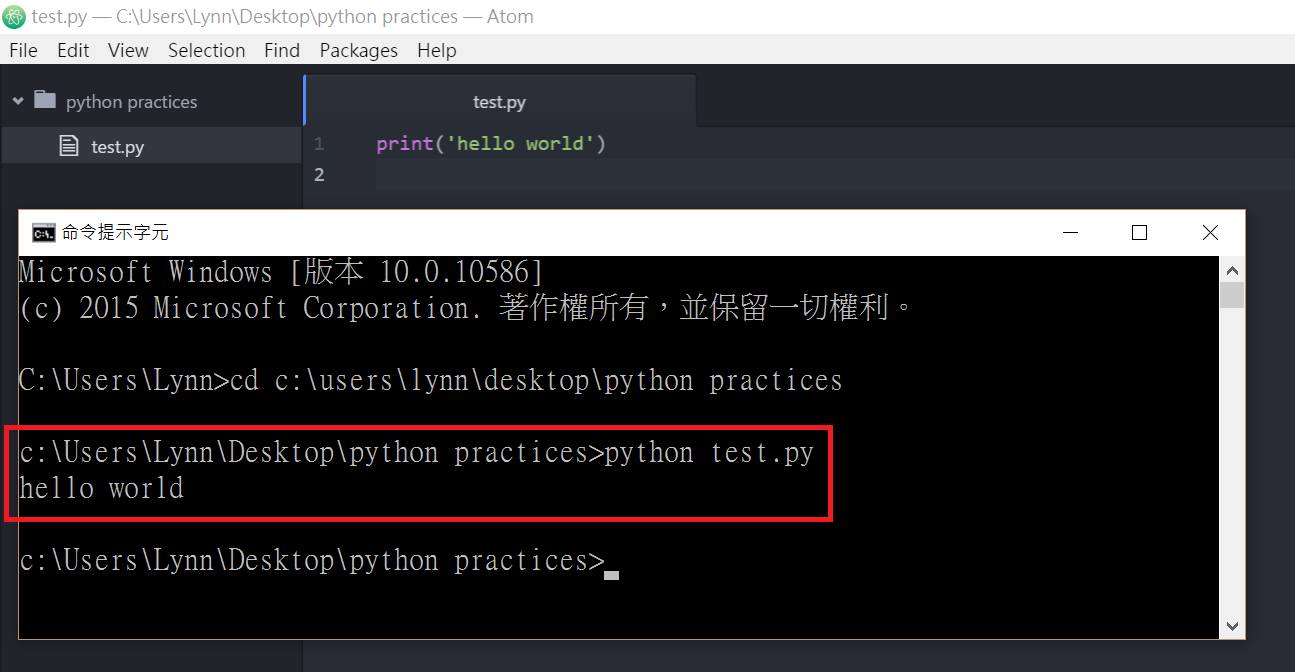
# 環境設置

首先可以至官網下載下載Python: [**https://www.python.org/**](https://www.python.org/) 。下載完成後，你將獲得: Python直譯器與IDLE (Python的IDE)。身為一個IDE、該有的都有了，只是由於有點醜，所以大家都不愛用。

接下來如何執行程式有兩個選項：

## 1. 使用主機Terminal

讓我們找一個寫程式用的文字編輯器: Notepad++、Sublime Text、Vim、Atom… 記事本也行，寫完程式碼後可以打開 Terminal 、打上 python 檔名.py 執行看看，不過此舉較不推薦新手用。



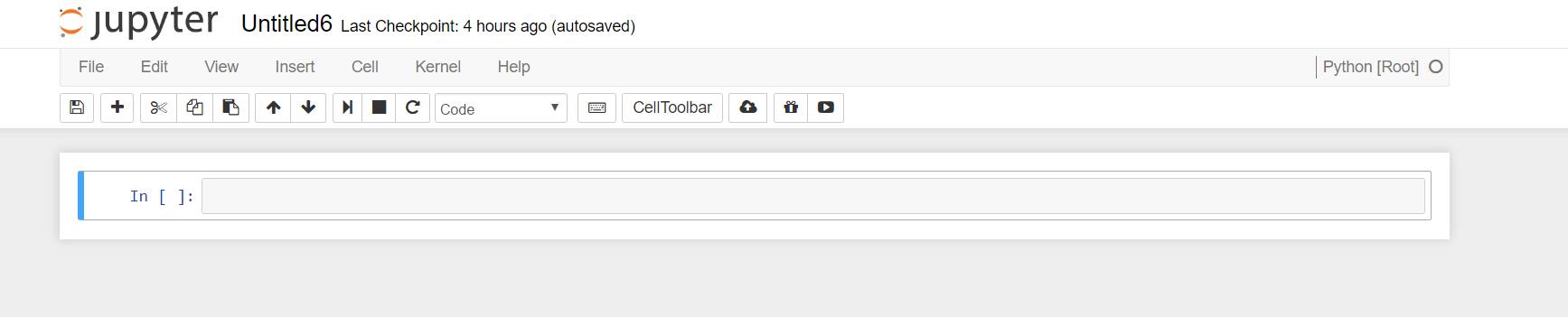
 可以看到Terminal印出hello world字樣了

## 2. 下載使用Jupyter Notebook

如果要更方便一點的話可以下載 Jupyter，也是 Python 的 IDE，包含了編輯器和編譯功能，非常適合 Python 的學習噢！安裝方式請參考：[**http://www.largitdata.com/course/29⋯⋯**](http://l.facebook.com/l.php?u=http%3A%2F%2Fwww.largitdata.com%2Fcourse%2F29%2F&h=ATPQvdvNRgK5bh5nJwYY4SrMzu0lYXIt_zPOaYu_dn7mf59do_7qCWT2_JBPghe6XMo62rRQ30lnUiJAOMZBsPznBZbnxiIJlWS7HTH4bXdXXwWKMhFRTqWXLBnTi64ef0_W4XfhVGDm&s=1)

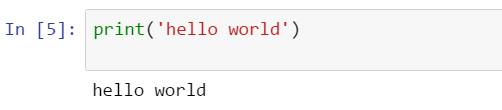
# Python基本語法與練習

好了，來實作看看吧！打開 Jupyter Notebook 應用程式、跳出  Jupyter Notebook 的編輯頁面就可以來撰寫我們的程式碼啦。



## 1. Print：永遠是跟世界打聲招呼的第一個程式

print語法是在print函式中放進你要印的東西，比如 print(我要印這個)。說聲 Hello World，跟程式的世界打聲招呼吧！

  
第一個跟世界打招呼的程式!

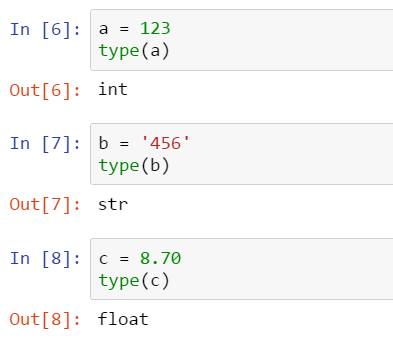
print的內容可以是變數(variable): print(a)，數字(init): print(1234)，或是字串(string): print(‘abc123’)。

## 2. Variable (變數)：一切TypeError的泉源

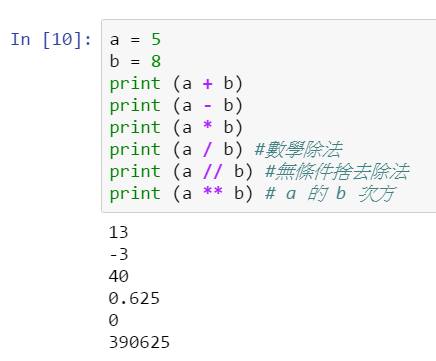
上面提到了變數，但這是什麼意思呢? 其實就是幫資料取個名字、把資料儲存起來。比如: 變數 = 資料，a = 123。當我們輸入a、就會找到123。

變數雖然能儲存資料，但資料有很多種類型、所以就會有資料型態(Type)。舉例來說：

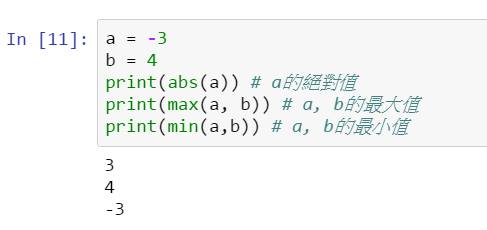
* a = 123 , type(a)會告訴我們 init (整數)。
* b = ‘456’ , type(b)會告訴我們 str (字串)。
* c = 8.70 , type(c)會告訴我們這是一個 float (浮點數)。



數值可以做運算：



也可以使用各種你知道的函數：



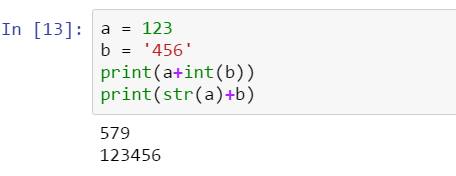
但不同的形態運算要注意，不要把字串和數值加在一起、否則會產生TypeError。



記得學會看錯誤訊息再回去看自己寫錯了什麼。

那該怎麼辦呢?

別怕別怕，若要運算則需要做型態轉換，可以用 int(), str(), float() 來轉換變數型態。可以看到下圖，int() 將字串’456’轉成整數、str() 將整數 123 轉成字串。



##### 謹記：「Error都是暫時的，沒有解不掉的Bug，莫忘心靈祥和。」

就算身邊沒有程序激勵員也要保持好心情喔(啾咪^\_<

## 3. Input：讓使用者輸入資料唄

這邊我們使用的是Python 3的環境，使用的是 input；如果環境是Python 2的朋友注意是用 raw\_input噢，Python 2的input有別的意義。

input的功能是讓使用者所有輸入的資料會被當成字串儲存起來，需要時再作轉型。舉例而言：

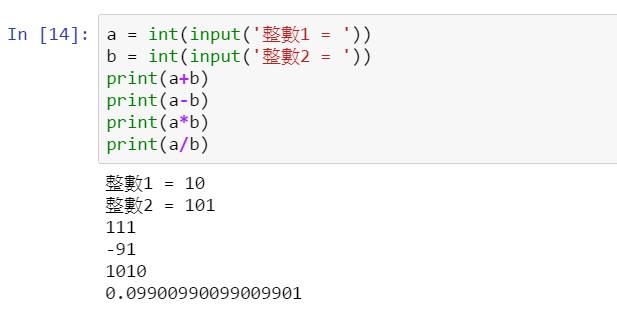
* a = input() : 它會等你輸入，並以字串(string)的形式存入 a 這個變數中。
* b = input(‘Your Name: ‘) : 在螢幕上會出現 ’Your Name: ’的提示，並把資料以字串(string)形式存入 b 這個變數中。



記得這個數字289是字串的形式喔~

**● 小練習**

**請寫一支Python程式，能讀取兩個整數，並把他們的四則運算印出來。(提示: 用input()印出來的東西是字串, 記得轉型噢!)**

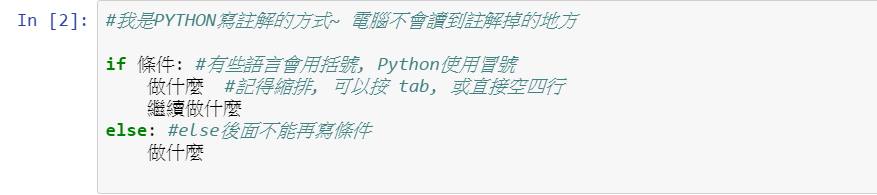


## 4. If-Else 日常的邏輯判斷

別說你沒用過 if-else 函數式，我們每天都在面臨 if else 的抉擇──如果(if) 百貨公司周年慶到了: 那就是大採購的時間啦! 否則(else)我們就晚點再買。

照樣造句一下還有──如果 (if)手機快沒電: 我就乖乖回家充電! 否則 (else) 我就繼續抓寶可夢。

來看一下 if-else 語法解說。if 寫完之後接冒號、下一行縮排後開始寫要做什麼事情。注意 else 是否則的意思，就不用再寫一次條件了。



但如果我們不只有兩層 (周年慶就去買東西、否則就等下次)，可能有好幾層條件呢?

比如: 如果我們平板電池的電量還很高、就去抓寶可夢；電池電量快沒了、就乖乖回家；如果電池電量普普通通、看看小說就好。

把這件事情用Python表達出來吧！



電池只有50，在30~80之間，因此看看小說就好！

if-else主要用在邏輯運算的判斷上：

* > 大於
* < 小於
* >= 大於等於
* <= 小於等於
* == 等於
* !== 不等於

或是布林值(Boolean)：

* true 真的
* false 假的
* and 且
* or 或
* not 非

不小心買到Samsung Note 7、聽說電池快沒電時容易有爆炸的風險，怎麼辦?! 利用邏輯運算來提醒我們這個危機吧！



**● 小練習**

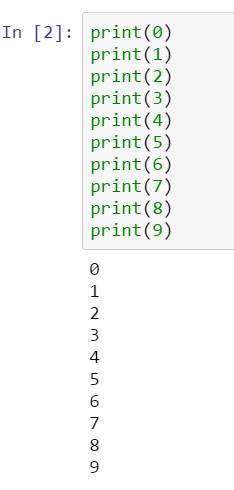
**寫一支Python整數機，第一步讓使用者輸入想要做的符號運算，比如「+, -, \*, /」，第二步讓使用者輸入’整數1’和 ‘整數2’，最後讓這兩個整數進行運算。如果輸入的運算符號不是「+, -, \*, /」，便輸出「錯誤」。**

## https://kopu.chat/wp-content/uploads/2017/01/py5-1.jpg

## 5. For Loop 轉圈圈吧哈姆太郎

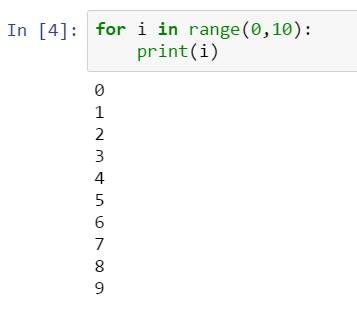
如果要把0到9的數字印出來，應該怎麼做呢?

嗯當然你可以這樣做…



**但如果今天是要從0印到100呢? (´・ω・`) …**

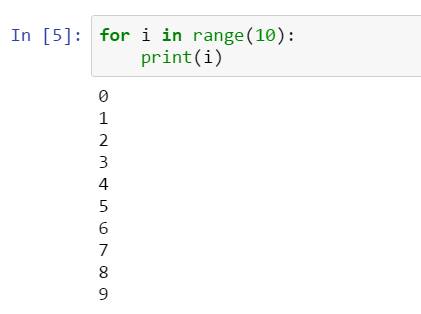
我們之所以寫程式，就是為了要讓程式能自動化地幫忙做一些事情，這樣顯得太沒有效率了。因此在這邊我們需要寫一個「迴圈」(Loop)。首先來介紹一個最基礎的Python迴圈「For Loop」。

比起痛苦地寫十行print，引進For迴圈後、區區兩行便能解決了：

不是很看得懂程式碼的意思嗎? 別緊張，這邊來解釋一下這個式子的涵義。若要枚舉出從0到n-1的數字，寫法是：

for [變數名稱] in range(n): (縮排) print([變數名稱])

來試試看：



在這裡的n=10，是不是把0~9 (n-1)的數字都印出來了呢?

**到底range又是什麼意思呢?**

事實上 range(n) = range(0, n)，簡單來說就是個產生器，你可以在這個range範圍中拿資料、拿完到n-1為止。以range(3)為例，第一次拿到資料0、第二次拿到資料1、第三次拿到資料2，然後就結束了。

讓我們重新看一下for函數結構：

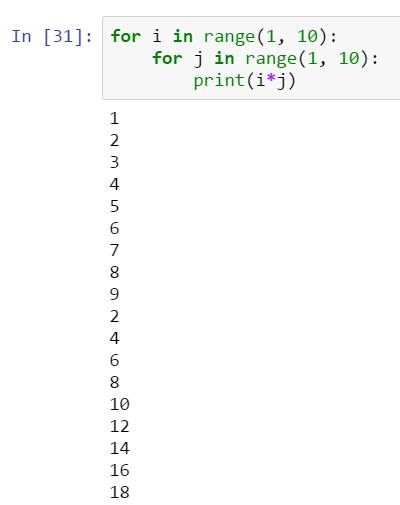


|  |  |
| --- | --- |
| 1  2  3 | for [變數名稱] in range(範圍, 比如從數字X到Y):  (縮排) 做一件事  (縮排) 做一件事 |

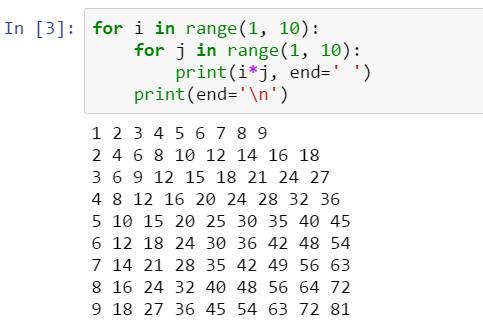
瞭解的話，就讓我們來做個小練習吧！

**● 小練習**

**請輸出一個九九乘法表。(提示1: for loop裡面也可以有第二個for loop噢!)**



**進階版: 如果不希望print出來的結果換行且要有空格，可以使用end=’ ’；若要再換行可使用end=’\n’。**



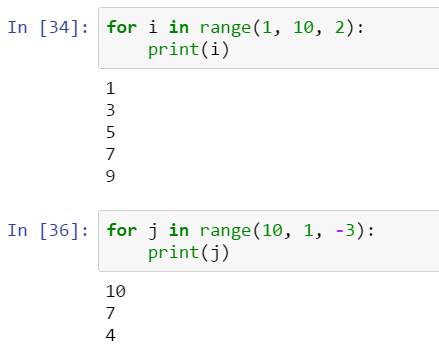
事實上range的數列是有間距的喔！

range 的結構是：range(起點, 終點, 間距)，其中的間距預設為1 、比如 range(0, 6) = range(0, 6, 1) 。代表從 0 到 5，每次加 1 的意思。

**分別印出:**

**(1) 從1到10、每次加2的數列;**

**(2) 從10到2、每次-3的數列 吧！**



**● 小練習**

**請做出一支能猜數字的程式：每次讓使用者猜一個整數，若猜對就輸出Bingo；使用者最多可以猜3次。(提示: Bingo後可以使用break來離開迴圈)**



## 6. List：能無限儲存資料的格子

想像一下，若今天有三筆資料要計算時，我們會這樣寫：



|  |  |
| --- | --- |
| 1  2  3 | a = 1  b = 2  c = 3 |

**… 那如果你有成千上萬筆資料呢?**

再想像一下，如果不知道使用者要輸入多少資料又要怎麼辦呢?



|  |  |
| --- | --- |
| 1  2  3 | in = input()  a = in[0]  b = in[1] |

這樣真的行嗎…

##### 「神說要有了List，所以有了List。」- 來自深海大鳳梨裡的神祕訊息。

繼續運用你的想像力！我們現在有個異次元裡的櫃子，它有無限個格子，第一格可以放東西、第二格也可以放東西…以此類推。

櫃子裡可以放任何東西，我們會用一個數字代表東西放在櫃子中的第幾層格子，這個數字就叫做 Index (索引)。

把櫃子的概念換成 Python 的 List 語法，就會變成：放在第 0 層的變數、放在第 1 層的變數… 這個數字就是 Index(索引)，而變數就是我們塞進去的資料。

一個Python的List長相如下，以中括號[] 包起來、逗號, 分隔：



|  |  |
| --- | --- |
| 1 | a = ['Lynn', 0.87, 1234, True] |

這裡的 a 就是一個 List，第 0 格放的是 Lynn 這個字串、第 1 格放的是 0.87 這個浮點數、第 2 格放的是 1234 這個整數、第 3 格放的是 True 這個布林值。

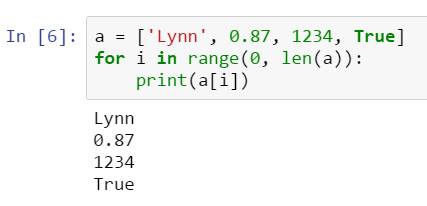
有了格子的概念後，我們可以使用**a[第幾格]**來取出變數，比如 a[0] 代表’Lynn’、a[3] 代表True。可以看到由於 a[4] 並不存在，因此會出現 IndexError的警示。

要如何知道一個List有多長呢? 我們能使用**len([list變數名稱])**。

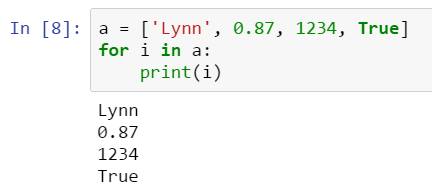


|  |  |
| --- | --- |
| 1  2 | a = [1, 3]  len(a) = 2 |

在拿到 List 的長度後，讓我們能來玩一點變化吧！比如 List 加上上面教學的 For Loop 看看會發生什麼事：

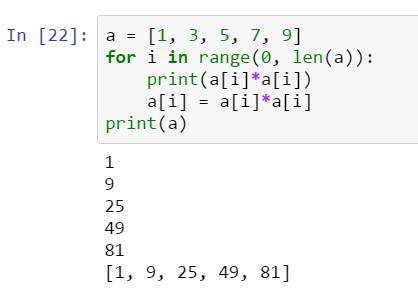


若要更簡潔一點的話，由於 in 這個關鍵字有「在… 之中」的意思，in range(0, 3)代表在0~2之中。因此能改成：



**● 小練習**

**現在有一個 list a = [1, 3, 5, 7, 9]，請對每一個元素都平方後印出來，且須將 a 也變成 [1, 9, 25, 49, 81]。**



來玩玩更多list的操作吧！

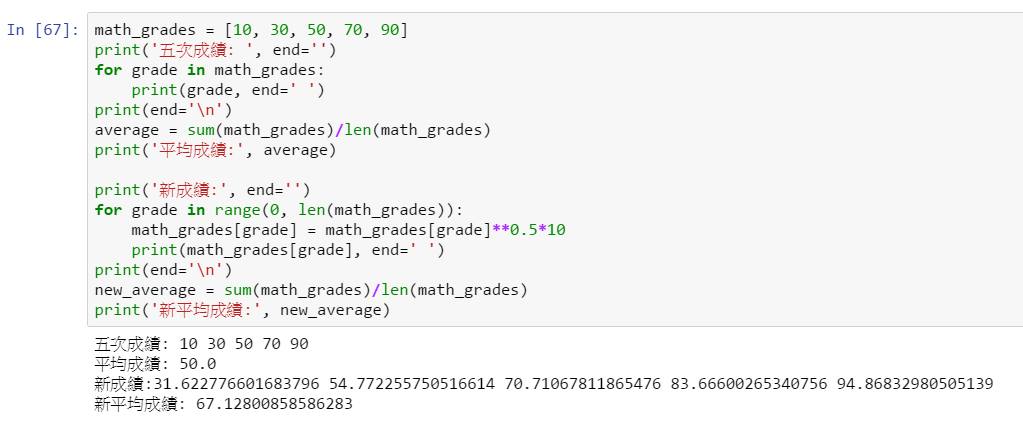
* 建立一個空的list: a = list() 或是 a = []都可以
* 動態增加元素:
  + list.append(x): 把變數x塞到list的最後面
  + list.insert(i, x): 把變數x塞到i這個位置上
  + list.pop(): 把list的最後一格丟掉
  + list.pop(i): 把list的第i格丟掉
  + list.remove(x): 會把第一個出現的變數x拿掉
  + list.clear(): 把list內的資料全部清光光
* 與常見函數的結合:
  + max(list): 找出list中最大值
  + min(list): 找出list中最小值
  + sum(list): 找出list數字總和



● **小練習**

**五次數學段考的成績分別為10、30、50、70、90分，算出平均後，老師發現大家考太爛、只好將成績開根號再乘上10 (提示: 成績\*\*0.5\*10)，再算出一個新平均。**

**請印出: 1. 五次成績; 2. 平均成績; 3. 五次新成績; 4. 新分數的平均。**

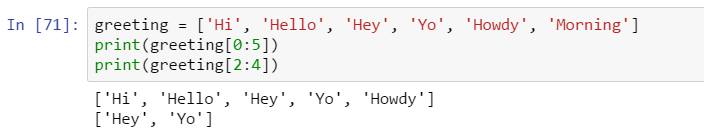


以為就這樣結束了嗎? List 還有切片 (Slice) 的功能哦大大！

Python 的 slice 功能讓我們能拿出 List 櫃子的一部份。

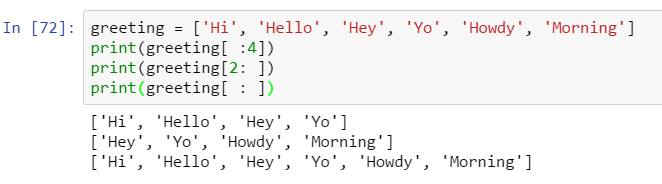


|  |  |
| --- | --- |
| 1  2 | list[start: end]  #拿到 list 的start ,start+1, start+2, ..., end-2, end-1 |



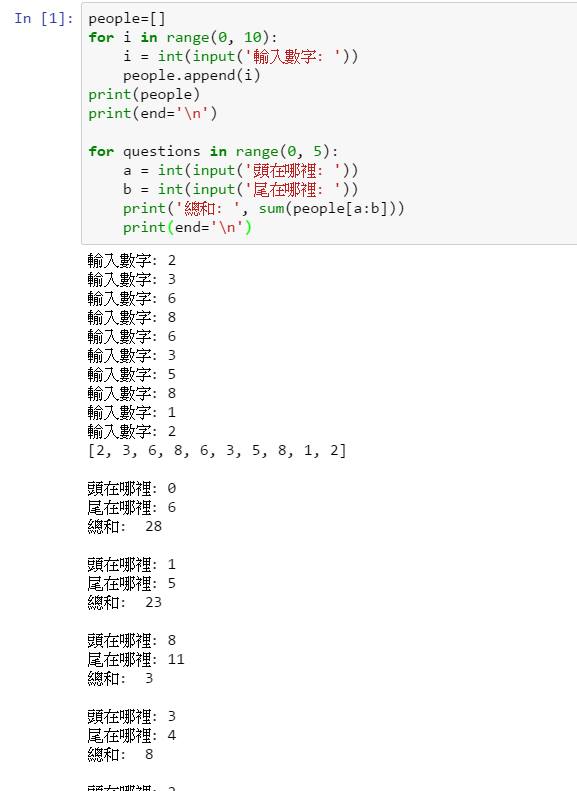
針對slice語法，讓我們深入介紹一些細節部分：

* list[start: end]，start和end都可以省略不寫
* start的預設為0
* end的預設為len(list)
* liest[ :end]代表 0~end-1
* list[start: ]代表start~len(list)-1
* list[ : ]代表0~len(list)-1



● **小練習**

**來辦場Party吧! 輸入十個整數、存入一個名為people清單中 (表示我們的宴客人數)；之後會有五次詢問，每次會輸入清單開始和結束的位置，再輸出從開始到結束位置的總和。**



## 7. Dict 一個靠標籤能查到東西的字典

在上一部分中我們提到，List 能動態增加長度、塞任何型態的資料 (字串、整數、浮點數)，並能透過 Index 來取得一層層櫃子中的資料。

但如果標籤不是按照櫃子順序行嗎? 比如給各個資料一個專屬的標籤名稱、再根據標籤名稱取出資料?

再一次運用想像力，我們有另外一種櫃子，上面會貼有標籤名稱，比如放在「冰箱」標籤中的東西，或是放在「抽屜」標籤中。

這個標籤代表著東西放在哪邊，稱為 key (鍵值)。標籤可以是任何東西、裡面什麼都能放。

把這個概念移到 Python 中的 Dict 語法吧！

Dict 其實就是 Dictionary (字典) 的意思，我們有可以放在key是 ‘box’裡的變數、也可以有放在key是123裡的變數，key只能是不能修改的型態，包括字串(str)、整數(int)與浮點數(float)。

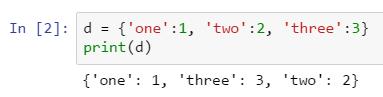
變數則和 List 一樣什麼都能放，比如變數可以是一個 List。

來看看一個 dict 語法吧，dict由大括號{}包住、元素以key:value的key-value pair組成 (key後面接冒號和變數value)，並以由逗點, 隔開。



|  |  |
| --- | --- |
| 1 | d = {123:'Snoopy', 'cat':['Pusheen', 'Kitty']} |

注意：dict在輸出時並不保證key的順序。



我們一樣能用 len(dict) 來找出 dict 的大小：



但如果 dict 中有 list 呢? 長度會變成多少?



|  |  |
| --- | --- |
| 1 | customer\_profile = {1: ['Lynn', 1205, 'college'], 'id': 235} |

印出來就知道啦。事實上是一樣的，我們只看key的數量。



另外，類似list可以用index來取值、比如list[index]，dict同樣能用key來取值。當取到不存在的key時，相對於list中的IndexError，dict會顯示KeyError：

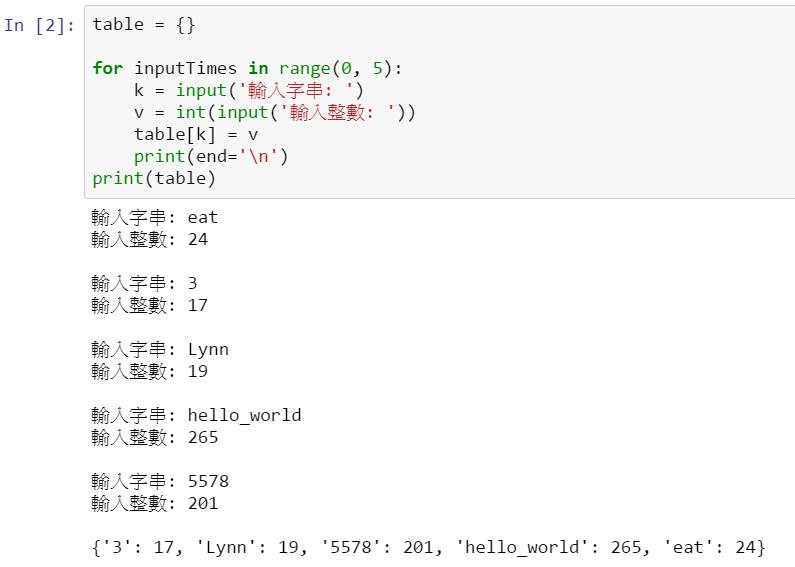


這邊來介紹一下 dict 的語法：

* 建立與刪除
  + dict={}: 建立空的dict
  + del dict[key]: 刪除特定的key-value pari
* 增加與更新
  + dict[key]=value: 如果key不存在，會增加這組K-V；如果key已存在，會更新這組K-V。
* for … in
  + for [變數名稱] in dict:每一回合會拿到一把在dict裡的key，就可以用dict[key]拿到對應的變數。

**● 小練習**

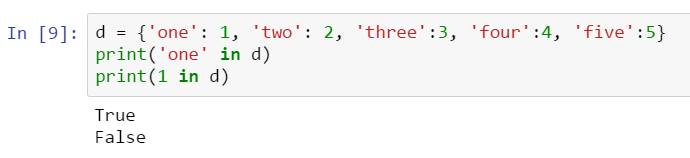
**讓使用者在名為table的dict中，輸入五組的key-value pair，key是字串、value是數字，最後把這個dict印出來。(提示: dict沒有append，直接用增加或更新(InsertOrUpdate)的方式)**



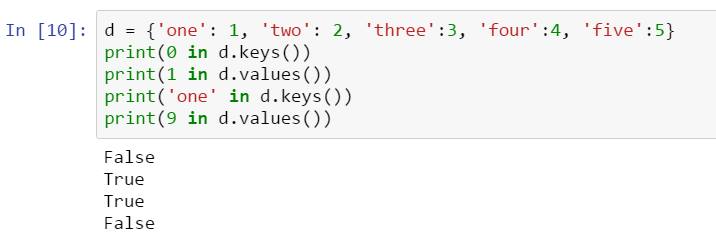
除了用key找value，我們也可以反查，用value找出相對應的key：



或是檢查dict中有沒有包含特定的key：



若想拿到一個由所有key組成的list、或一個由所有value組成的list，可以使用dict.keys()或是dict.values()。結合反查的方法，可以檢查一個特定的key或value在這個dict中存不存在。



那如果有了key，想要把dict中相對應的資料取出來應該要怎麼做呢? 作法有兩個：

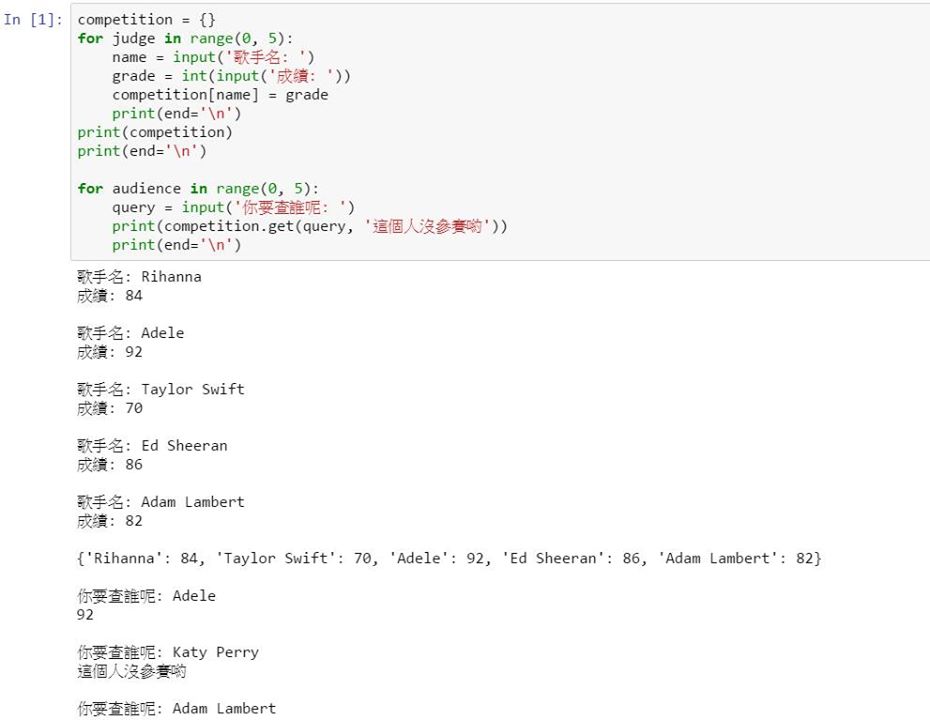
* d[key]: 這個做法相對不安全，key如果不存在的話就會出現KeyError (如同上面我們所示範的)。
* d.get(key, default\_value): 是比較安全的作法，如果key不存在的話就會回傳 default\_value (下面例子中的deafult\_value就是’找不到耶’)。



**● 小練習**

**一年一度的世界歌王大賽來啦！身為評審的你，請輸入五個歌手的名字與成績。接下來會有五次查詢的機會，每次查詢都可以讓觀眾輸入一個名字、來查看心愛的歌手。**

**如果歌手不在名單中，請輸出’這個人沒參賽喲’；如果歌手在名單中，請輸出該歌手的名字與成績。**



## 最終專案：FB狂讚士

呼~好不容易走到這一步了，有沒有很興奮呢!!!!!! 接下來我們將開始撰寫人生中第一個爬蟲程式啦~

首先我們得到Facebook Developer(開發者中心) 網站上點選工具及支援，進入Graph API Explorer。

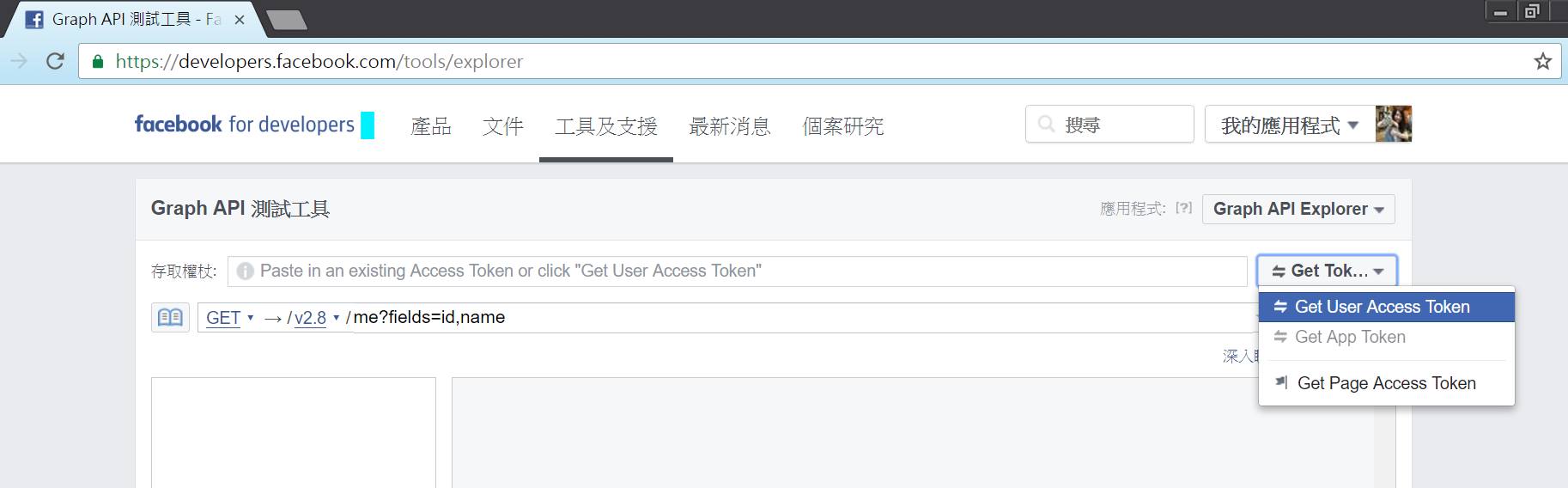
什麼是Graph API? 其實大家想要使用Facebook API的目的大多都是希望能夠擷取與使用者相關的一些物件 (人、網頁、相片…)與 連結關係(按讚、留言、標籤…)。

Facebook將這個概念稱之為Social Graph，而存取這些關係的API，就稱之為Graph API。透過Graph API，任何你需要Facebook提供的資訊都能從中拿到。

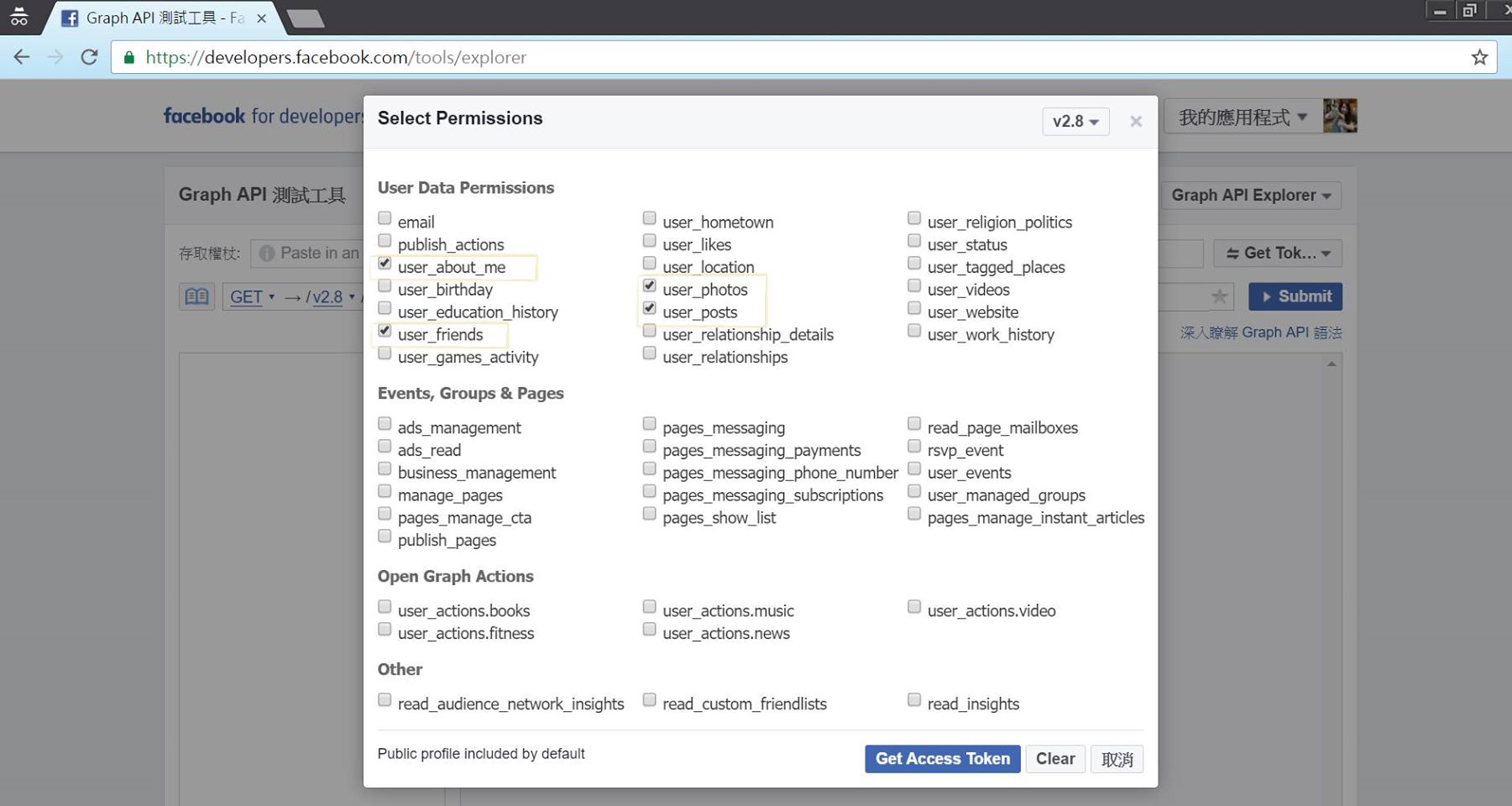


進入Graph API Explorer後拿取FB Access Token。這個Token代表你的身分，因此請謹記：絕對不要外流Token!!!

來設定一下權限吧，點擊Get User Access Token：

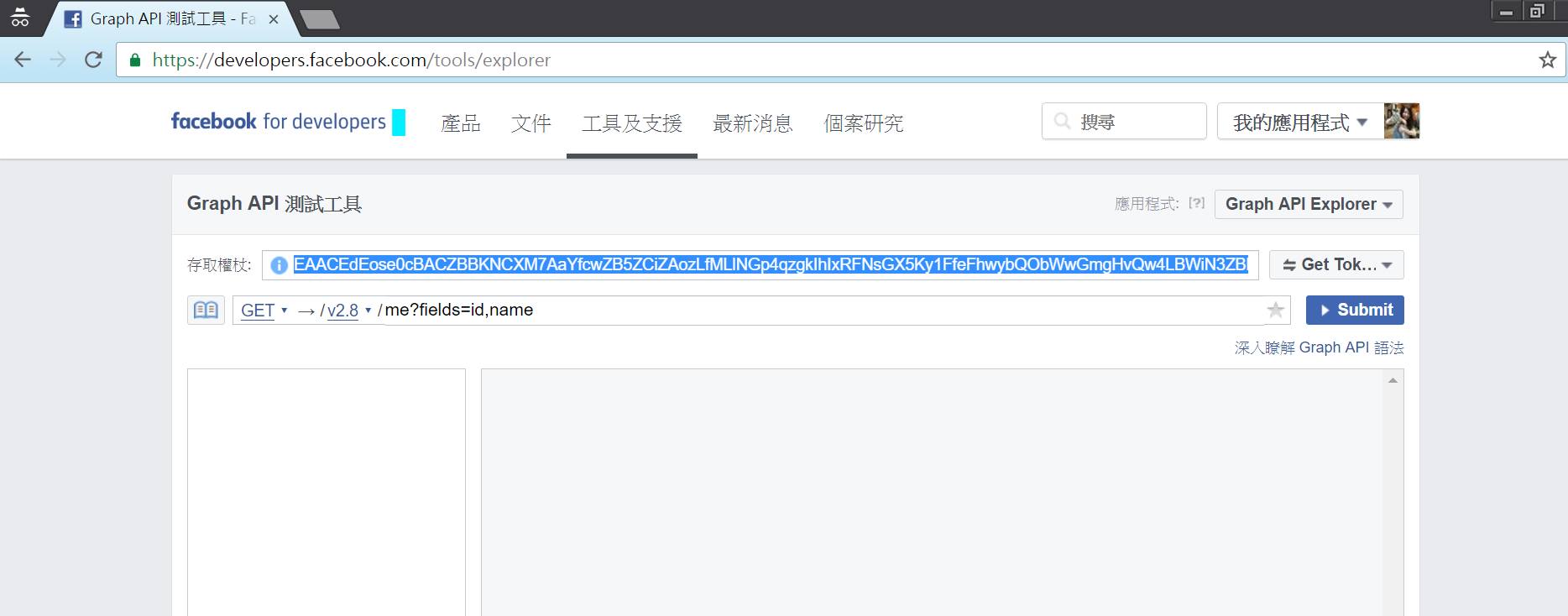


點選要開啟的權限，這邊我們設定 user\_about\_me、user\_friends、user\_photos和user-posts；也可以根據你想分析的內容勾選。最後按下Get Access Token。



如果想要自動發文和按讚的話，就要點選 publish\_actions噢!

最後將那行超長的「存取權杖」字串給複製下來：



由於拿到了權限，我們可以在 GET 這邊輸入一些 API 的字串，按下Submit 後就可以看到一些個人訊息，比如/me查看個人檔案、/me/friends查看 FB 好友、/me/posts 查看貼文紀錄。

如果沒有出現相對應資料的話，表示Get Access Token剛剛勾選的時候沒有打開權限，再回去確認一下即可。

接下來的 code 比較多，我們切換回來編輯器 (Editor) 這邊、等程式碼寫完後再使用終端機 (Terminal) 把檔案啟動~ 做法是 python 檔名.py。

在使用 Graph API 之前，先安裝 Request Package，做法是打開Terminal，輸入**$pip install requests**即可。

接下來就把facebook這個module載入進檔案中，故Editor的最上方輸入import facebook，並放進你剛剛在Graph API網站上複製的token：



|  |  |
| --- | --- |
| 1  2 | import facebook  graph = facebook.GraphAPI(access\_token='your\_token', version='2.7') |

嘗試玩幾個有趣的小專案！

這邊因為筆者的腦已經快爆炸了就純粹附上程式碼和結果供大家參考，有問題再來詢問嗚嗚嗚嗚QQQQ (也可以直接去看Graph API的Document, 相當清楚!)

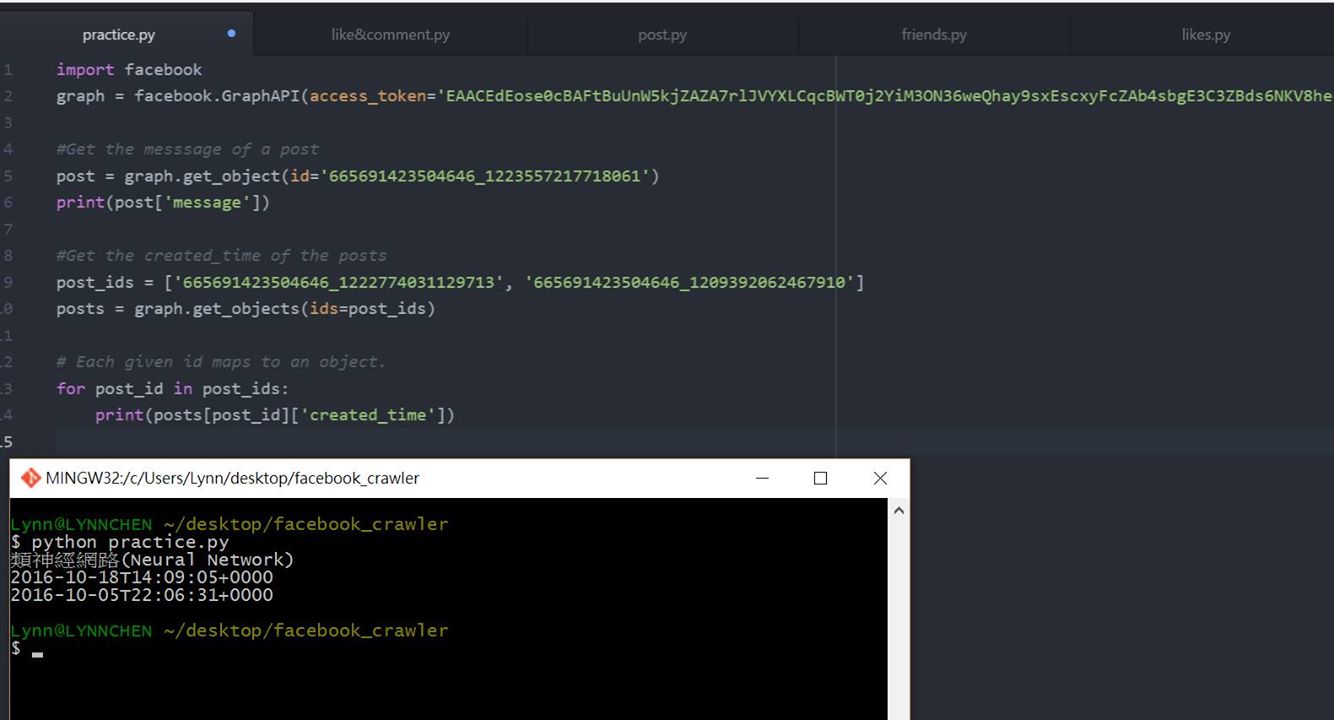
### 

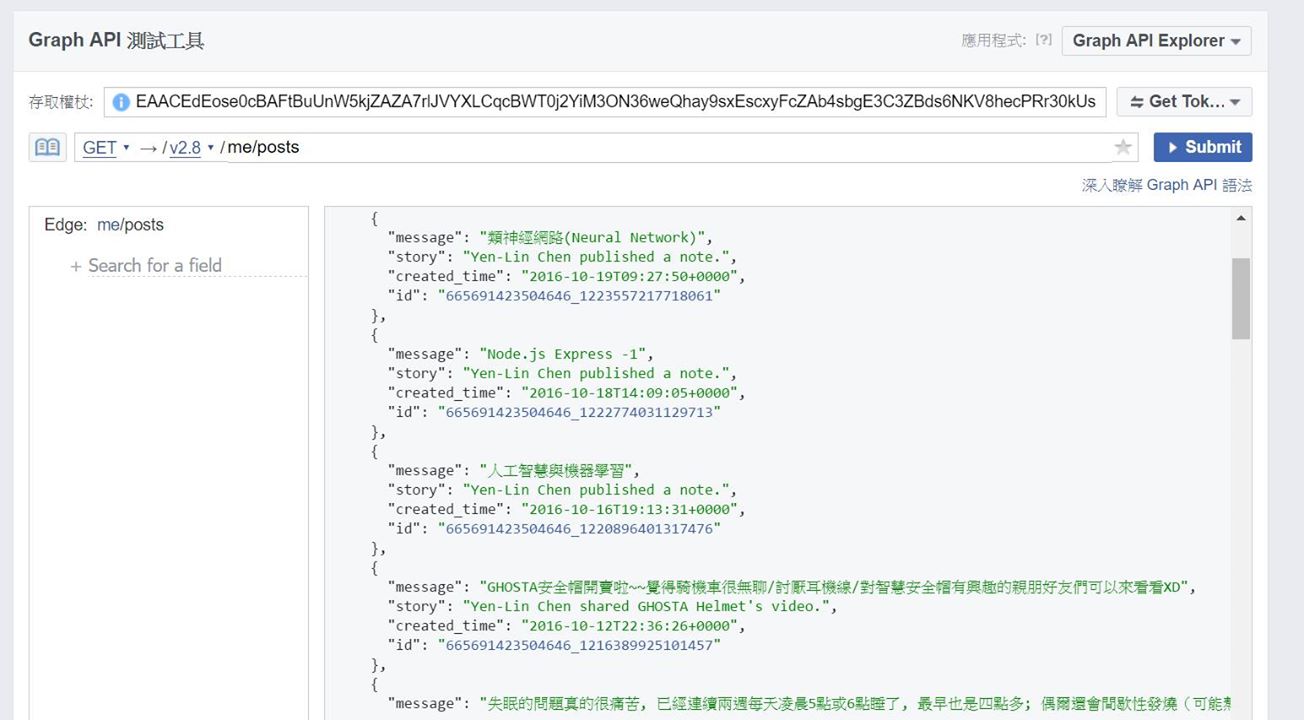
### 1. 查出PO文的message，與創建PO文的時間



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | import facebook  graph = facebook.GraphAPI(access\_token='your\_token', version='2.7')    # Get the messsage of a post  post = graph.get\_object(id='665691423504646\_1223557217718061')    print(post['message'])  # Get the created\_time of the posts    post\_ids = ['665691423504646\_1222774031129713', '665691423504646\_1209392062467910']  posts = graph.get\_objects(ids=post\_ids)    # Each given id maps to an object.  for post\_id in post\_ids:  print(posts[post\_id]['created\_time']) |

對照一下… 文章”id”是”665691423504646\_1222774031129713”的message是”類神經網路(Neural Network)”，而另外兩篇文章id的created\_time也成功印在Terminal上了! 噢耶!





### 2. 自己幫自己按讚和PO文



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | import facebook  graph = facebook.GraphAPI(access\_token='your\_token', version='2.7')    #Put a like! 我是Node.js那篇網誌!  graph.put\_like(object\_id='665691423504646\_1222774031129713')    # Put a comment! 我是GHOSTA Helmet影片分享  graph.put\_comment(object\_id='665691423504646\_1216389925101457', message='我是一個Graph API測試~~~') |

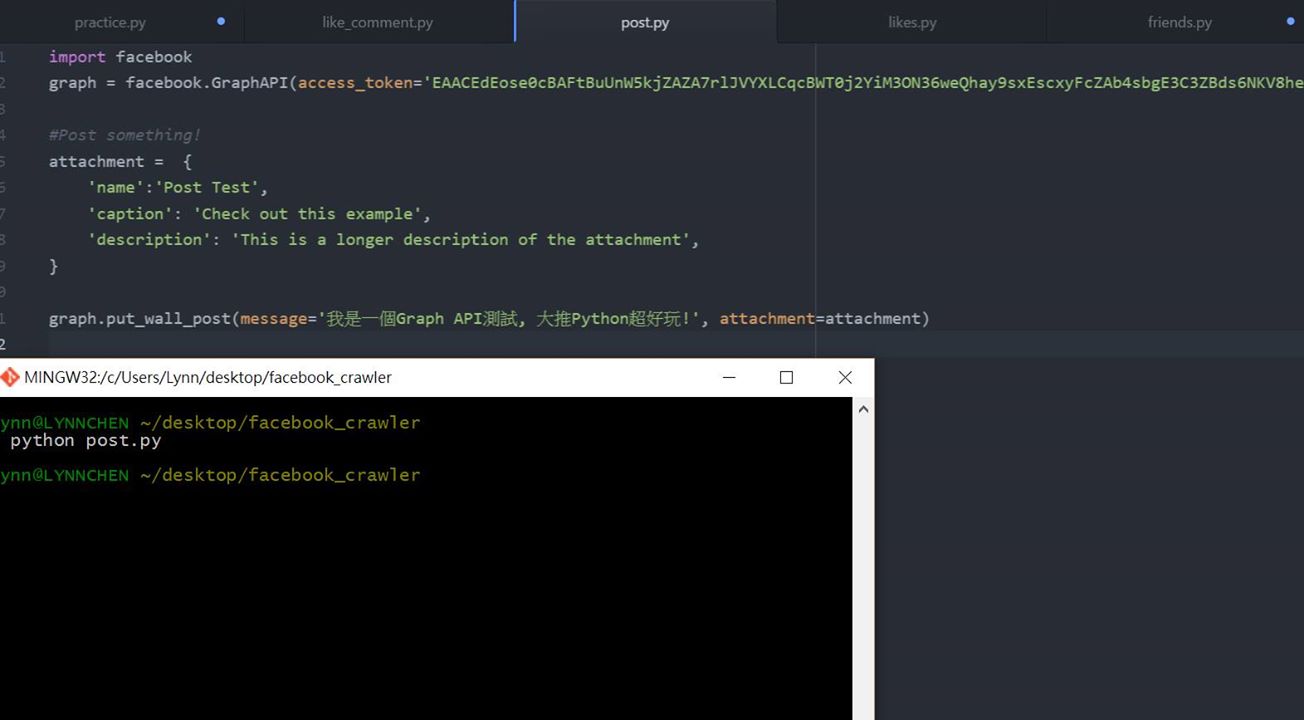
得澄清這真的不是我自己先按讚或自己去留言的XDD” 一樣成功了!



### 3. 自動PO文



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | import facebook  graph = facebook.GraphAPI(access\_token='your\_token', version='2.7')    # Post something!  attachment =  {      'name':'Post Test',      'caption': 'Check out this example',      'description': 'This is a longer description of the attachment',  }    graph.put\_wall\_post(message='我是一個Graph API測試, 大推Python超好玩!', attachment=attachment) |



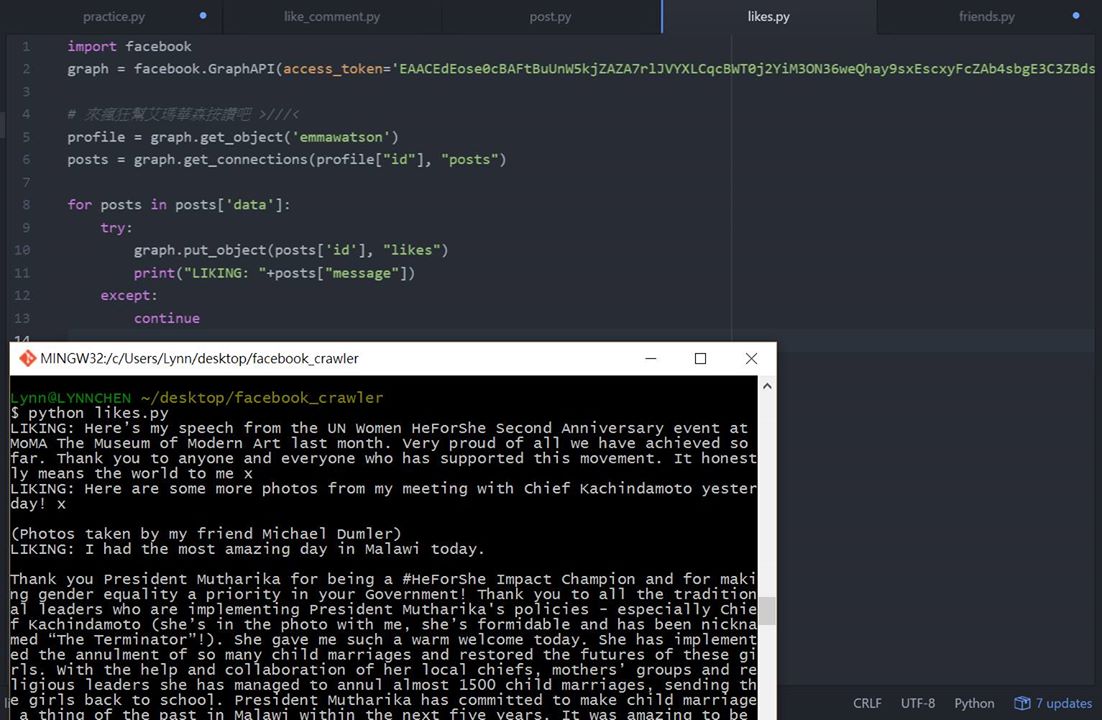
哇~~自動PO文出來啦~~~



### 4. 為心愛的人成為狂讚士吧!



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | import facebook  graph = facebook.GraphAPI(access\_token='your\_token', version='2.7')    # 來瘋狂幫艾瑪華森按讚吧 >///<  profile = graph.get\_object('emmawatson')  posts = graph.get\_connections(profile["id"], "posts")    for posts in posts['data']:      try:          graph.put\_object(posts['id'], "likes")          print("LIKING: "+posts["message"])      except:          continue |



成功洗了 N 則艾瑪華森 FB 粉專貼文的讚…



今天的Python與FB Graph API教學先到這邊