



寫在最前面



先講結論，儘管本書的內容有些章節有點難，但這是一本針對新手寫的程式語言學習書！

話說，某個週末我在家裡趕專案進度，家裡讀國中的小朋友突然來跟我說...

//

兒子：「爸爸，我們學校好像要教寫程式了」

我：「是喔，你知道要教什麼程式語言嗎？」

兒子：「好像是教 Python...」

我：「蠻好的，我來寫一本 Python 的書給你好了」

//

我看小朋友的表情好像沒有很開心的樣子，但這本書的起源的確就是這樣來的。

在我大四那年開始接觸網路，當時為了想要做自己的網站而開始自學 HTML 跟 CSS，又想要網站可以看起來跟別人的不一樣，所以也開始學一些簡單的程式。我第一個學的程式語言是 ASP，才剛學完 `if...else...` 跟迴圈等簡單語法，連函數都還不太熟就開始在學校的電算中心打工寫選課系統。有一次看到同事打開了一個黑黑的畫面在敲打一些指令（後來才知道是終端機視窗搭配 Vim 在寫 Perl），感覺就像電影裡的電腦高手一樣，看起來好帥，而且程式語

看起來也比我寫的 ASP 有趣，便去學校圖書館找書來看，然後就試著用 Perl 幫家人的公司寫了一個簡易的排班系統。

這應該就是我第一次接觸開放原始碼的世界，接觸之後才發現開源的世界好寬廣而且好多厲害的人。後來因為工作的關係，寫了幾年的 Flash 跟 PHP（還因此辦了台灣的第一屆的 PHP 研討會）以及 Python。

在 2009 年到 2013 年期間我用 Python 跟 Django 幫客戶開發官方網站或活動網站，也有在台灣的開源社群開了好幾堂公開課程。不得不說 Python 寫起來真的比之前寫過的 ASP、Perl 以及 PHP 要好寫很多。直到後來接觸了 Ruby/Rails 之後，發現在 Rails 裡的模組自動載入機制，可以不用寫一堆的 `from ... import ...` 就能直接使用，當時覺得這樣好方便（但這其實有別的問題），所以在 2013 年就跳轉到 Ruby 直到 2024 年，一跳就超過十年。

目前全世界的程式語言種類非常多，每種程式語言被設計出來都有它想要解決的問題或專門的用途，例如大家現在如果聽到 PHP 這個程式語言，應該不會認為它是用來開發手機應用程式。講到 Python，現在大家可能會聯想到機器學習或人工智慧。Python 是一款泛用型的程式語言，早年的關注度還不錯但並沒有像現在這麼高，就以 [TIOBE](#) 這個程式語言排行榜的數字來看，原本差不多就跟 PHP、Ruby 等程式語言差不多的排名。誰也沒料到，自從 2016 年 AlphaGo 橫空出世打敗世界棋王之後，深度學習、機器學習、人工智慧等相關技術開始被大家關注到，雖然當時的熱度普遍都還只有在開發者之間，但這時候坊間 Python 的相關書籍已經明顯的增加了不少。

直到 ChatGPT 的出現，原本的舊時王謝堂前燕，講到人工智慧大家會想到的可能是得要研究演算法、要看好多論文，突然飛入尋常百姓家，連我坐在路邊攤吃個宵夜都能聽到隔壁桌的大叔或阿桑在談論 AI，書店的 Python 書突然變成好幾櫃，在 TIOBE 上的排行更是瘋狂的往前衝，直接衝到第一名，幹掉 C、C++、Java 這些長期霸榜的程式語言。



TIOBE Index: <https://www.tiobe.com/tiobe-index/>

認識我比較久的朋友可能知道我平常除了寫程式、開發專案外，同時也有經營一家電腦補習班（是台北市合法登記的那種補習班），所以我有一半以上的時間在我們的培訓班協助學員們轉職軟體工程師。這些年來我們都是使用程式語言 Ruby 以及網站開發框架 Ruby on Rails 做為我們培訓課程的後端教材。不過隨著市場趨勢的變化以及同學們的敲碗聲，外加我們家小朋友的課業需求，所以我也試著做一份 Python 版本的教材。

老實說，我並不認為我所學的程式知識有什麼特別之處，很多知識也是從網路上的開源資源學來的。秉持著吃果子拜樹頭的精神，本書內容將全部在公開於 「為你自己學 Python」 網站上，網站上內容除另有標示外將以 CC BY-NC-SA 4.0 方式授權。不論是已經工作一陣子的社會人士，或是在校學生，或甚至在學校準備教學生寫程式的教師，希望可以藉由這樣免費開源的方式釋出教材，能讓大家可以用更低的成本學習寫程式的技能。



網站連結

為你自己學 Python：<https://pythonbook.cc>

本書內容

目前主流的 Python 版本是 3.x 版，事實上 Python 3 在 2008 年就上市了，當年由於 Python 2.x 跟 3.x 之間的語法的不相容，導致 Python 的開發者社群從 2.x 版升級到 3.x 版之間拉扯（或分裂）了滿久的時間。隨著時間推進，雖然 Python 2.x 版還是可以使用，但官方已宣佈在 2020 年之後不會再繼續幫 2.x 版的 Python 增加新功能。本書在撰寫的當下最新版本是 **3.12.7** 版，沒有特別註明的話，本書將使用這個版本做展示。

這本書主要目的是跟大家一起學習 Python 這個程式語言，除了邏輯判斷以及流程控制外，也會介紹常用的資料型態以及物件導向程式設計，並且動手做一些小工具來完成日常的瑣碎工作。

也就是說，本書主要將以 Python 程式語言以及網站開發為主，不會有人工智慧或機器學習等相關內容（我自己目前也還夠不熟悉，而且我相信坊間可以找到更多優秀的參考書籍），或是也許在一本書再來跟大家介紹這方面的主題。

關於 Python

Python 怎麼唸？

雖然 Python 的英文是「蟒蛇」的意思，但這個程式語言跟蛇沒什麼關係。

Python 應該怎麼發音？查了一下線上的劍橋詞典（Cambridge Dictionary），Python 的英式發音跟美式發音有一點不同：

- 英式： /'paɪ.θən/
- 美式： /'paɪ.θa:n/

這兩種唸法我都有聽過，就算發音的不太一樣大家應該還是知道對方在唸什麼。如果硬是要選邊站的話，可以看看 Python 的發明者 Guido van Rossum 是怎麼唸的。Guido 是荷蘭人，我看了好幾部他在研討會演講或是專訪影片裡，他本人講到 Python 的時候聽起來比較像是美式發音的 /'paɪ.θa:n/，以中文音譯比較接近的大概是「拍桑」。

哪些單位在使用 Python？

Python 目前在 TIOBE 是排行第一名的程式語言，所以大概也不需要我再吹捧它有多受歡迎或是哪些大型企業採用 Python 進行開發，老實說，到底是誰在使用 Python 這件事我也不是很在意。

如同本書的標題「為你自己學 Python」，我一直相信，學習不需要為公司、為長官、同事、老師或是爸媽，而是為你自己。只有當自己真心認為學習對自己有幫助或是有趣，這條路才會走的長遠。

Python 可以用來做哪些事

Python 是一款泛用型的程式語言，一開始並沒有特別設計用來處理特定的業務，目前普遍常看到的用途有：

- 撰寫自動化程式替我們完成原本需要人工操作的繁瑣工作。
- 撰寫爬蟲程式抓取資料，並對資料進行進一步的整理及分析。
- 網站應用程式開發。

- 機器學習、人工智慧相關領域的應用。

Python 容易學嗎？

大家應該很常聽到 Python 是一款很容易上手的程式語言，這是真的。

就是因為 Python 的語法簡單容易上手，這對有學過其他程式語言的老司機來說可以學的更快，但也更容易發生「拿明朝的劍斬清朝的官」的情況，例如拿著其他程式語言的物件導向的經驗來學 Python 還滿容易踩到坑的。

所以，Python 容易學嗎？這個問題我先保留，等你看完這本書再告訴我答案。

關於本書

需要先學點什麼嗎？

如果之前曾經寫過 Python 以外的程式語言是不錯，沒有也沒關係，反正這本書本來就是設定給沒寫過程式的讀者看的。

閱讀順序

不一定要從第一章開始照著章節順序閱讀，你想要跳著看、躺著看或倒立著看也都可以。不過因為我在撰寫本書的時候所設定的讀者背景是沒有程式背景的新手，所以本書的章節安排算是針對新手所設計的學習路線，建議不妨可以試著按照著章節的順序閱讀，可能會比較容易上手。

在本書中，偶爾會遇到比較難理解的章節，我也會特別提醒各位如果看不懂可以先跳過，等比較熟悉之後再回來看也沒問題。學習本身不是單行道也不是線性的，遇到障礙的時候，偶爾繞個路也沒什麼問題。逃避雖然可能有一點點可恥，但在學習的路上倒是挺有用的。

程式碼慣例

在本書中使用的程式碼慣例，如果你在範例裡看到 \$ 符號開頭的，像這樣：

```
$ cd /tmp/hello-python  
$ python -V
```

這個不是程式碼，這個 `$` 表示後面輸入的是一個系統指令，你應該是在終端機視窗（在 Windows 作業系統則是命令提示字元或 PowerShell）環境下透過敲打鍵盤輸入這些指令，而且不需要跟著輸入 `$` 符號，輸入了反而會出錯。

除了 `$` 符號外，有時候你可能會在本書看到這樣的範例：

```
$ python  
Python 3.12.7 (main, Oct 17 2024, 20:07:59) [Clang 15.0.0  
(clang-1500.3.9.4)] on darwin  
Type "help", "copyright", "credits" or "license" for more  
information.  
>>> print("hello")  
hello  
>>> 1 + 2  
3
```

上面出現的 `>>>` 是進入 Python 的 Shell 環境，這個環境通常會稱它為 REPL，是 Read-Eval-Print Loop 幾個字的縮寫，我會常在 REPL 環境下試一些比較簡單的程式碼，所以同樣的也不要跟著輸入 `>>>` 符號。

有時候會為了聚焦重點或是不要讓整頁的篇幅被程式碼範例給佔滿了，我會視情況省略部份程式碼，簡化之後看起來大概會像這樣：

```
def hello  
    print("hi")  
    # ... 略 ...  
    print("hey")
```

別擔心，如果有需要，完整的程式碼我會另外再提供讓各位下載。

另外，在本書中如果看到我用開頭大寫的「Python」的時候，通常指的是 Python 這個程式語言；如果是全小寫的 `python` 的時候，通常指的是終端機指令或是或是用來執行 Python 程式碼的執行檔。

有問題可以問誰

在閱讀本書的過程中如果遇到些問題或發現有寫錯的地方，你可以在這本書的網站上留言，同時我們還有經營線上的 Discord 社群，裡面有個專門討論 Python 的頻道，應該也可以在裡面找到人。

不過如果你還是比較喜歡實體交流的話，不管只是想要找人聊天或討論技術上的問題，我們每週二晚上 19:00 ~ 22:00 在我們的教室（在台北市 228 公園旁）都有定期舉辦名為「默默會」的實體社群活動，歡迎大家有事沒事都來泡茶聊天。

網站連結

- 線上討論：
 - 線上網站：<https://pythonbook.cc/>
 - Discord：<https://5xcamp.us/discord>
- 實體活動：
 - 默默會：<https://www.facebook.com/rubymokumokukai>

調整學習方式

我自己早年的學習方法，會照著書本上的程式碼範例或是跟著線上教學影片的程式碼跟著敲打一次，我在教室講課的時候也很常看到同學們一邊盯著講台上的投影布幕一邊在自己的電腦上跟著打字。這樣的學習方式很常見也很不能說有什麼問題，因為這樣的確會有一種跟上進度的安心感甚至是進步的假象，認為自己學會了。但真的要自己動手寫的時候，卻發現不知道從何下手...

不要只是模仿

大部份的學習者可能沒意識到，像這樣的練習可能只有練到打字速度。只照著書本或課堂上的範例跟著打字練習，但卻沒有想為什麼這樣寫，就像鸚鵡模仿人類講話一樣，遇到沒學過的句子就講不出來。



試試看換個方式，在閱讀這本書程式碼範例或是線上課程的時候，先不要急著跟著敲打鍵盤，可以先把書本或課程的內容看一點進度，然後把書本或螢幕蓋起來，試著對自己講解剛才這段的重點在講什麼。如果覺得自言自語有點怪，也可以找一隻黃色小鴨跟它對話（這看起來可能更怪？），都是很好的理解練習。只要能夠講的出來，通常就是代表你對這段知識的掌握度還不錯。

接著，在不看著書或影片的情況下，試著自己寫看看，如果寫不出來，可以偷看一下再回來繼續寫。一開始會發現這樣的學習很沒效率，但相信我，這樣學習程式的效果會越來越明顯。

建立 SSOT

不要隨便相信網路上的文章，你可以參考，但不要太快就相信，因為說不定這些文章的作者跟你一樣都是剛學 Python 沒多久的新手而已。

「單一真相來源（Single Source of True，SSOT）」是指在眾多的資料中，只有一個是真正正確的來源，所以建立 SSOT 的過程差不多就相當於建立信仰。SSOT 就像憲法與法律的關係一樣，任何法律跟憲法牴觸無效，特別是在現在這種資訊爆炸謠言滿天飛的時代，建立單一真相來源才能夠避免被假消息給帶偏了風向。當然我這裡說的 SSOT 指的不是我這本書，請各位在閱讀本書的時候也同樣抱持著懷疑的態度。我不可能知道 Python 所有的東西，我也可能會寫錯，所以覺得有哪裡怪怪的或是說不通的地方都歡迎線上或實體一起討論。

建立 SSOT 的過程對新手來說是很困難的，因為新手就是不知道什麼才是對的所以才是新手。在學習程式建立基礎觀念的過程中，最容易獲得的 SSOT 就是官方文件。不過因為官方文件通常比較無聊或比較好睡，所以可以再搭配其他像是維基百科、Stack Overflow 之類的資料來源。

另一個也很容易獲得而且更精準的 SSOT 來源，就是原始碼。官方文件也是有可能不小心寫錯，可能是漏寫或是工程師懶得寫，但原始碼不會騙人，原始碼在程式設計領域裡面是最真實的資料來源，只是它會比官網更難理解。在這本書裡我可能會引部份官網文件、維基百科甚至是 Python 的原始碼，大家也可以順著這個思路去找更多的資料來源，建立自己的 SSOT。

ChatGPT 之類的 AI 工具很厲害，但在你建立 SSOT 的過程請小心服用，它會一本正經的回答問題，而且語氣看起來很有自信，有點難分辨它講的是對的還是不對的。

學習不是直線的，有時候繞一點彎路或是看看沿途的風景也不是壞事。就像我們在玩拼圖的時候，大部份的人不會從最左上角逐片拼到最右下角，通常會先找出邊邊角角這種特徵明顯的形狀，接著找顏色或圖案接近的，最後就能把整張圖拼起來。而且，就算最後整份拼圖最後少幾塊，你還是能看的出來拼圖的圖案是什麼，學習本身也是一樣的道理，你不用學會所有語法，也能學會如何寫程式。

不用擔心自己大腦的記憶體、理解力比別人差或是學習的速度比別人慢，持續往對的方向前進，維持步調，不久的將來一定會感受到進步。

關於

封面設計

不得不說，這本書的封面我真的超喜歡的，從顏色搭配到角色設計，處處都是細節，我知道這年頭 AI 繪圖越來越厲害，但我還是喜歡這種有溫度的手繪設計。很榮幸也很開心能有機會請大師出馬，讓我的書有這麼漂亮的封面。



由 [Croter Illustration & Design Studio](#) 設計

Croter，本名洪添賢

設計師與插畫工作者，2004 年開始投入獨立創作與設計，擅長使用多種插畫風格與設計結合，並且喜歡使用超現實變異的手法繪製插畫，融合神話故事與諷刺性的幽默，用天真爛漫的語氣緩緩傾訴人生與社會的現實。2021 年獨立出版的繪本《什麼將把你帶走》獲選為 2022 波隆納拉加茲獎年度百大繪本與 46 屆金鼎獎最佳圖書插畫獎。執業以來也累積豐富的商業插畫作品，表現跨越設計、插畫與藝術，屢獲各界肯定。

目前居住在高雄，每天仍持續不斷的在現實量尺與創作理想中持續用畫筆奮鬥著。

 網站連結

Croter Illustration & Design Studio

<https://www.facebook.com/croter.taiwan>

我

我是高見龍，是一個喜歡寫程式而且希望可以寫一輩子程式的電腦阿宅。高見龍這個看起來有點像武俠小說的名字並不是筆名，這是我父母給我的本名，我很喜歡這個名字。

我從 1998 年開始寫各式各樣的網站應用程式，2009 年的時候因為朋友的介紹開始接觸了開源相關的社群活動，發現這個圈子好多傻子，都無私的貢獻自己的時間跟精力在開源專案跟技術社群上，發現了新玩具就巴不得趕快跟大家分享。當年我實在搞不懂這樣做有什麼好處，所以我就加入大家，看看這些人到底在幹嘛。搞到最後，我光是參加社群活動還不過癮，甚至還自己辦活動，就是想認識更多跟我一樣的傻子。

我從社群分享到開班授課已經超過 15 年，近年有寫了幾本技術書籍，比較多人知道的是「[為你自己學 Git](#)」以及「[為你自己學 Ruby on Rails](#)」，平常也會拍影片或寫文章記錄自己的學習歷程。

以下是我的個人網站以及社群帳號，歡迎追蹤或加好友：

網站連結

- 個人網站 <https://kaochenlong.com>
- Threads <https://www.threads.net/@kaochenlong>
- Instagram <https://www.instagram.com/kaochenlong/>
- Facebook <https://www.facebook.com/eddiekao>
- X <https://x.com/eddiekao>
- GitHub <https://github.com/kaochenlong>
- YouTube <https://www.youtube.com/@kaochenlong>

最後，我盡力把這本書寫得盡可能的完整與正確，但我就只是個普通人，而且還是不務正業的非資訊本科系畢業，技術跟經驗都有限，也都還在學習的過程中，難免有不小心寫錯或是觀念理解錯誤的地方，還請先進們多多包涵並不吝指正。接下來就請大家準備跟著我一起上車吧 :)

工商服務

想學 Python 嗎？我教你啊 :)

想要成為軟體工程師嗎？這不是條輕鬆的路，除了興趣之外，還需要足夠的決心、設定目標並持續學習，我們的 **ASTROCamp 軟體工程師培訓營** 提供專業的前後端課程培訓，幫助你在最短時間內建立正確且扎實的軟體開發技能，有興趣而且不怕吃苦的話不妨來試試看！