推薦序

在創新工場內，學鏞是一個很特殊的人，他的職位是首席佈道 / 架構師，在華人圈能夠擔任這樣職位的人有如鳳毛麟角，因為架構師（Architect）必須有很多年的軟體開發實務經驗，而佈道師（Evangelist）必須熟知新技術且熱愛宣傳技術，透過寫作、演講等方式推廣技術。兩者結合，且做得相當好，確實不容易。

在 IT 技術方面，學鏞是我認識最執著且對技術的深度與廣度都能兼顧的人。他對軟體技術的熱愛是發自內心的，且似乎總能從學習新技術的過程中得到樂趣。他寫過許多技術文章與書，參與過許多軟體的開發，講過許多技術課。現在他在創新工場，對我們的創業團隊進行技術上的指導以及擔任投資專案的技術評審。

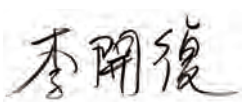
儘管學鏞是個專業的人，但他另一個很強的特點是：擅長把複雜的技術用簡單清楚的方式描述出來，這本《編程 ING：人人都能學會程式設計》正是這樣的一本書。要讓「人人」都能學會程式設計，這是一個很難又相當有價值的目標。透過這本書，學鏞確實做到了。

人人都能學會程式設計，並不是說人人都應該以軟體工程師為職業。現在社會高度電腦化，我們每天與手機、平板、電腦等設備上的各種軟體或網站為伍，如果我們能多一點軟體相關的知識，甚至能寫簡單的程式解決一些生活上的小問題，這是多麼棒的事！

這本書的風格非常像微博，一張圖搭配一則短文，讀這本書就像是讀了三百多則圖文並茂的微博。這本書也展現出學鏞的 PPT 設計功力，每張圖都是他自己精心繪製的。將概念圖像化，對於學習的幫助很大。

我喜歡這本多采多姿、深入淺出、走入群眾的書。我相信你也會喜歡。

創新工場 董事長兼首席執行官



序

我做過很多不同類型的工作，包括大學老師、培訓師、軟體工程師、架構師、出版社編輯、譯者、專欄作家，但其實這些工作都圍繞著程式設計這個專業領域，因為我熱愛程式設計。

程式設計既有趣、又有創造力、還能幫助提升日常工作效率。你只需要帶著你的人腦與一台電腦，就能開始進行程式設計，把腦海中的想法在電腦中實作出來。我有幸很早就體會到程式設計的迷人之處，從小學開始學習寫程式，至今 29 年，依然喜歡。可惜的是，像我這樣的人畢竟是幸運的少數，有許多人對於程式設計感興趣卻又不得其門而入。

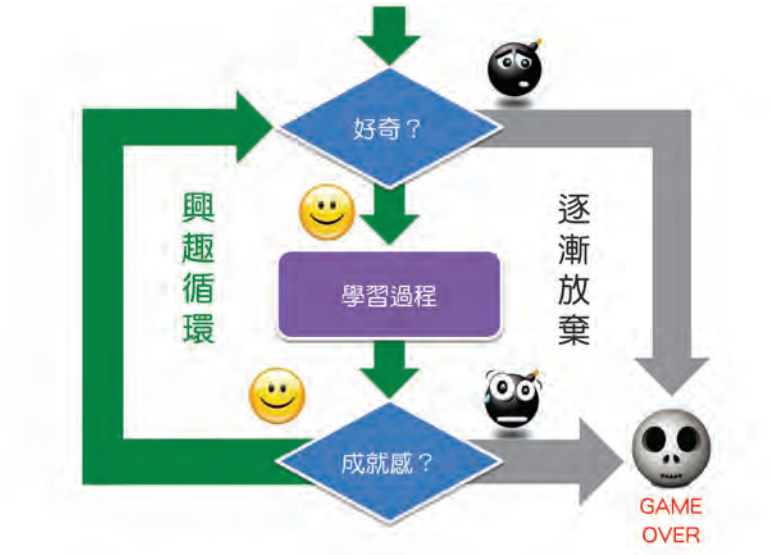
多年以來，我一直想寫一本程式設計入門書，以幫助程式設計初學者。為此我傾注了相當大的心力。現在，我的目標終於達成，成果就是你手上的這本書。對於初學者來說，死板的理論與生硬的說教都是禁忌，只會讓初學者打退堂鼓。面對初學者，我必須發揮創意，讓這本書的內容安排能深入淺出。除此之外，趣味性與實用性也是必要的，可讓初學者保持學習的動力。

拿著這本書，用一個週末假期的時間仔細閱讀並動手操作，你很可能會發現，原來程式設計這件事可以這麼有趣，這麼吸引人。接下來你或許要擔心上癮了！

蔡學鏞

於北京中關村

目 錄



待在興趣循環內，別讓你的學習 Game Over ！

想要有良好的學習成果，必須進入興趣循環。這個循環是由好奇心、學習過程、以及成就感所組成。一開始是由好奇心觸發學習動機，接下來展開學習，學習後產生成就感，而對更深入的內容感到好奇，於是繼續學習。一旦沒了好奇心或成就感，很可能就會放棄。

成就感是一種心理狀態，與挫折感相反。想要獲取成就感，就需要有好的學習成果。請務必堅持學習，直到下一次成就感產生。

除了成就感，好奇心也可以刺激學習。不妨帶著一絲疑惑進行學習探索，直到真相大白的那一刻，那是一種豁然開朗的喜悅。

如何待在興趣循環內，你需要成就感、好奇心、目標、與獎賞

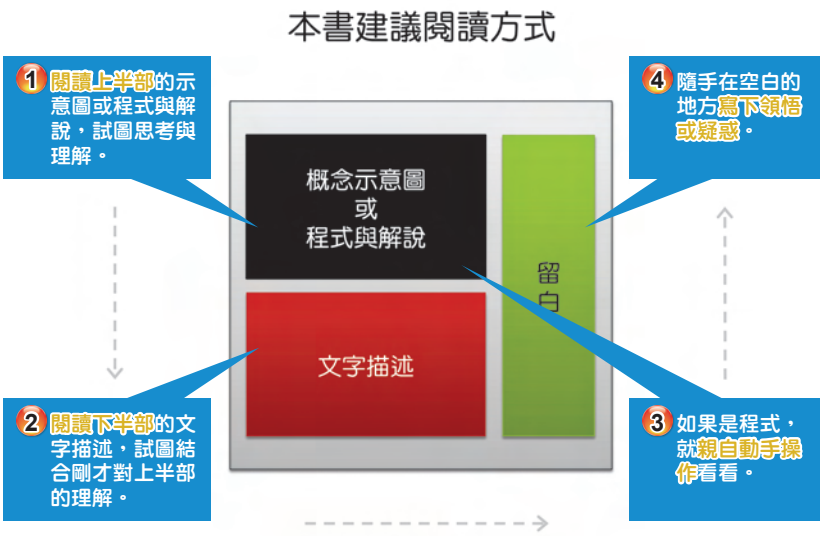
前面提到成就感與好奇心是學習的兩大關鍵，成就感與好奇心可不是說來就來的。但你可以透過一些手段激發你的成就感與好奇心：

1. 給自己定好許多可行的短期目標。如果你不知道該定怎樣的目標，可以參考本書每篇一開始列出來的學習目標，每次達成目標，就勾選該目標前的方框。當你很肯定地勾選，表示目標達成時，你的內心會出現一絲成就感。

2. 你可以在達到一定的學習目標之後，就犒賞自己。例如完成四個學習目標，就獎賞自己奢侈地大吃一頓（如果大吃一頓是你所熱愛的）。對於獎賞的渴望，會讓你的學習可以堅持得更久一點，學習過程也會更順利一點。獎賞自己的時候，成就感會更明確。

3. 有了成就感，你就會想要繼續挑戰下一個目標。整體進入一個良性循環。

4. 好奇心會在你良性循環的學習過程中隨時出現，比較難捉摸。請務必把握機會，在好奇心出現時，加強學習。

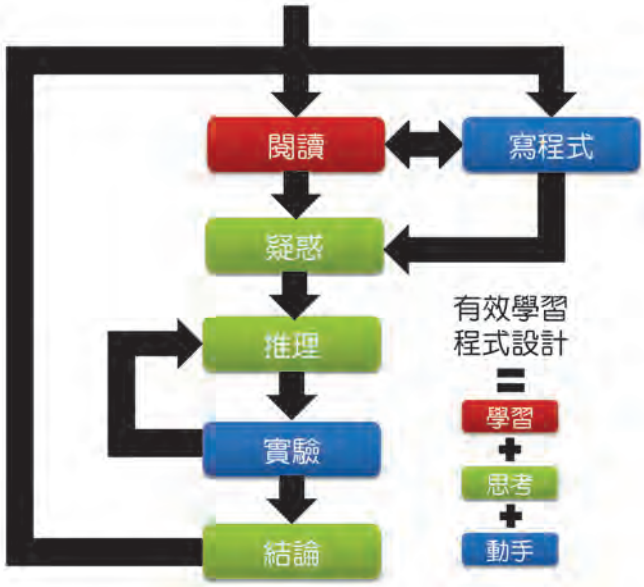


採用正確的方法，可以提高學習效率

「書都看了，也都看懂了，但還是不會寫程式」很多人有這樣的問題。學習效果不佳，通常是因為沒有思考與動手所致。如果你保持思考與動手的習慣，並堅持一段時間，我保證學習成效會不錯。

你必須一邊閱讀，一邊思考，甚至質疑書中的內容。動手跟著書本實際操作，以加深印象。對於不清楚的部份，透過動手實驗得到解答。把無法證實的疑惑，立刻記錄下來，等待以後某天知識累積足夠了而頓悟。

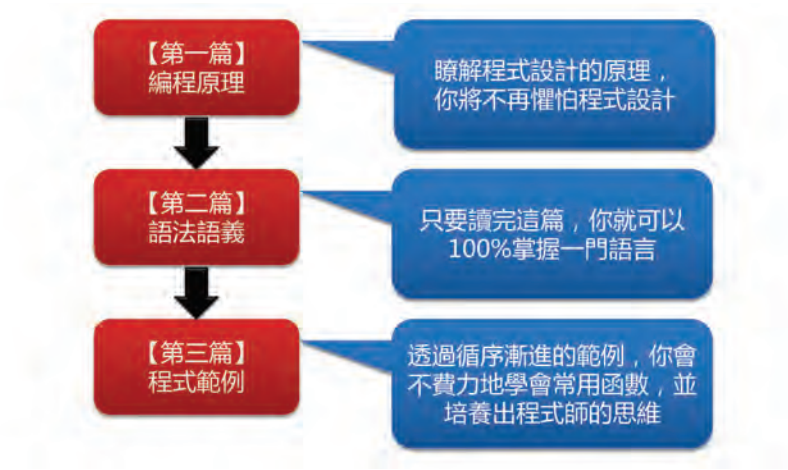
閱讀本書每一頁都可以採用圖中描述的這四個步驟。首先閱讀並思考上半頁的精華，再看下半頁文字描述的細節，接著實際動手操作加強領悟，過程有任何問題與想法都要馬上記錄下來。



有效地學習 = 學習 + 思考 + 動手

學習程式設計與學習其他技術一樣，不能光看書，必須從做中學習，才有實際效果。所以請先準備好一台電腦，桌上型電腦或筆記型電腦都可以。電腦上會附帶一個作業系統（OS），最有可能是微軟的 Windows，或蘋果的 Mac OS X，也可能是其他作業系統。當然光有電腦與作業系統依然不夠，想做程式設計還必須有編譯器或解譯器，關於這方面，第一章說明。一開始不知道該寫什麼程式時，可以跟著本書的範例一起動手。

想學好程式設計，懷疑的態度與設計實驗的能力也是相當重要的。多多懷疑某些事，然後推理出一番原理，最後做實驗證實或推翻自己的想法。有這種好奇心與實驗精神的人，才可能學好程式設計。

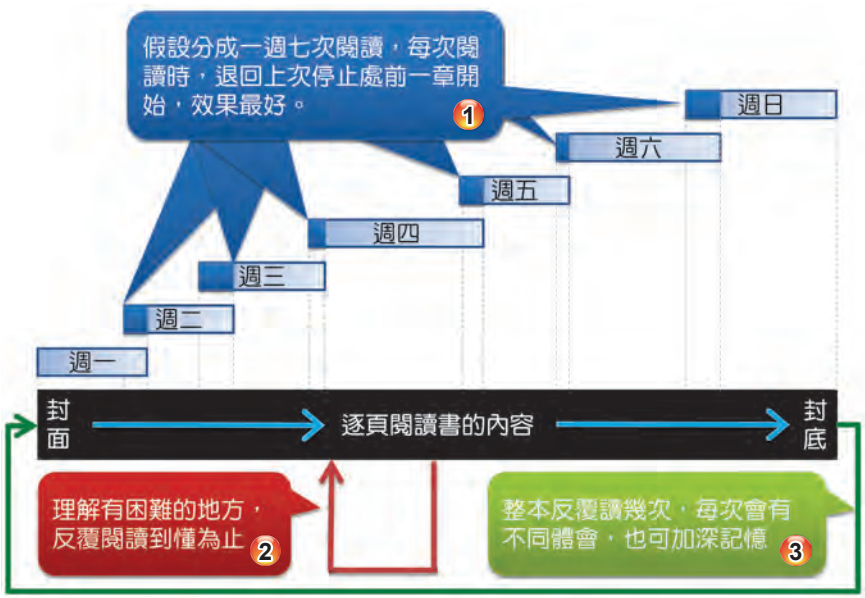


本書共有三篇，這三篇各具目的：

第一篇是編程原理，這會是你見過最詳細的程式設計概念解說。讀完本篇，你就能瞭解程式設計的原理，有了大局觀。你將不再懼怕程式設計。

第二篇是語法語義，完整地解說一個語言，沒有遺漏。不可思議地，只要學習完這麼簡短的一篇，你就能 100% 瞭解一個語言。接下來你就可以迎接真正程式設計的挑戰了。

第三篇是程式範例。前兩篇有一些簡單的操作，目的是要讓你熟悉語言的個別元素，但第三篇會用更具體的範例，有具體的需求，具體寫程式。這些程式是逐漸遞增功能的，所以學習坡度相當和緩。透過這些程式，你將會學習到許多常用函數，並培養出專業程式師一般的思維習慣。



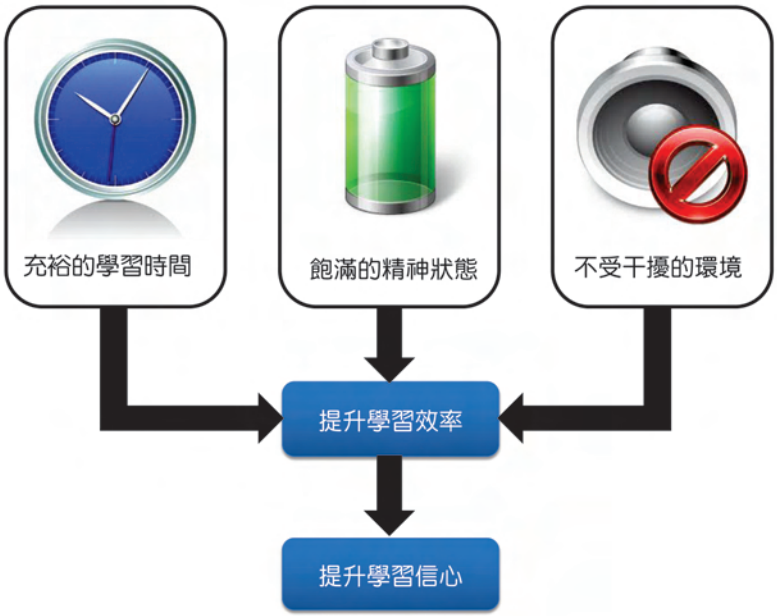
學習的次序：重疊、局部反覆、整體反覆

我們往往無法一口氣讀完一本書（尤其是要一邊閱讀、一邊思考、一邊動手操作），要分很多次才能閱讀完畢。每次閱讀時，我建議不要從上次停下來的部分開始，而應該倒退回一兩章開始。重疊的部份一方面可當做複習，也可以讓自己進入上次的心理狀態。另外，上次閱讀停頓可能是因為學習效果開始大幅下降了，所以重複閱讀可以彌補上次學習時理解上的缺失。

本書內容前後有相當高的依賴性，如果某些觀念沒弄懂，對於後續的學習可能會形成障礙。所以我建議，對於理解有困難的內容，要多讀幾次，直到懂為止，不要輕易跳過。

整本書讀完之後，你還可以從頭讀第二次、第三次，每次都會有不同的收穫、注意到不同的細節，你對程式設計的理解會越來越清晰。

本書內容交互參考的地方相當多，你可以根據自己的學習狀況，選擇翻到參考章節快速瀏覽複習或者不理會。當你第一次閱讀本書，內容提到對未來章節的參考時，你可以直接忽略。



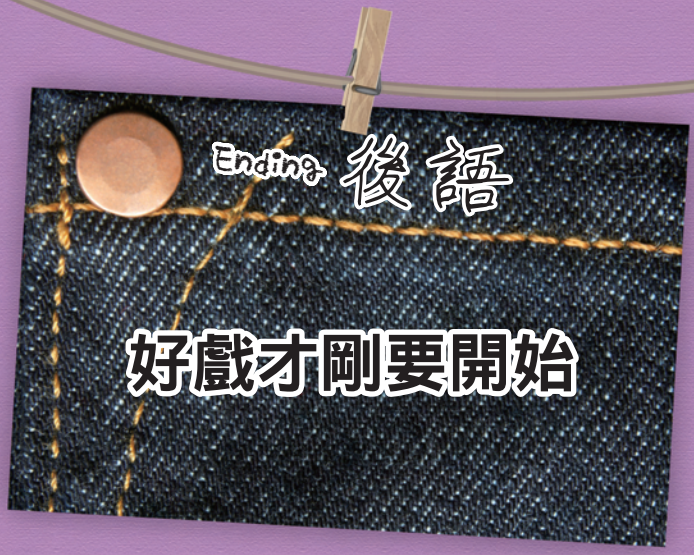
時間 、環境、精神都必須配合

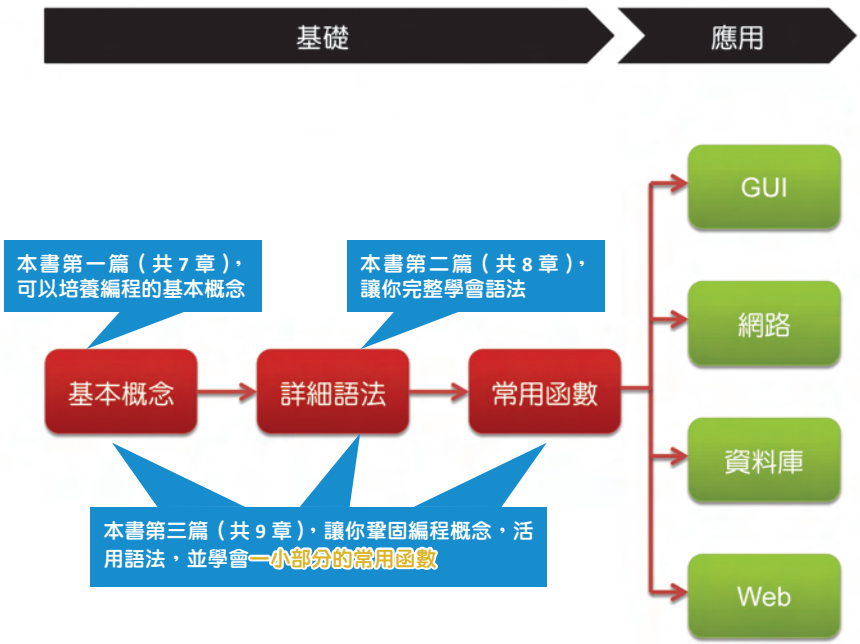
學習時的時間、精神狀態、與環境都很關鍵。如果沒有在充裕的時間、飽滿的精神狀態、與不受干擾的環境下，學習效果是不可能會很好的。更糟糕的是，這會給你一個錯誤的假像：我不是學習程式設計的料。這種心理暗示的殺傷力很大。

週末假期睡眠充足，然後把手機等干擾物都關了，一整個下午和晚上關在房內讀這本書，效果是最好的。

☑ 我已經做好學習程式設計的心理準備

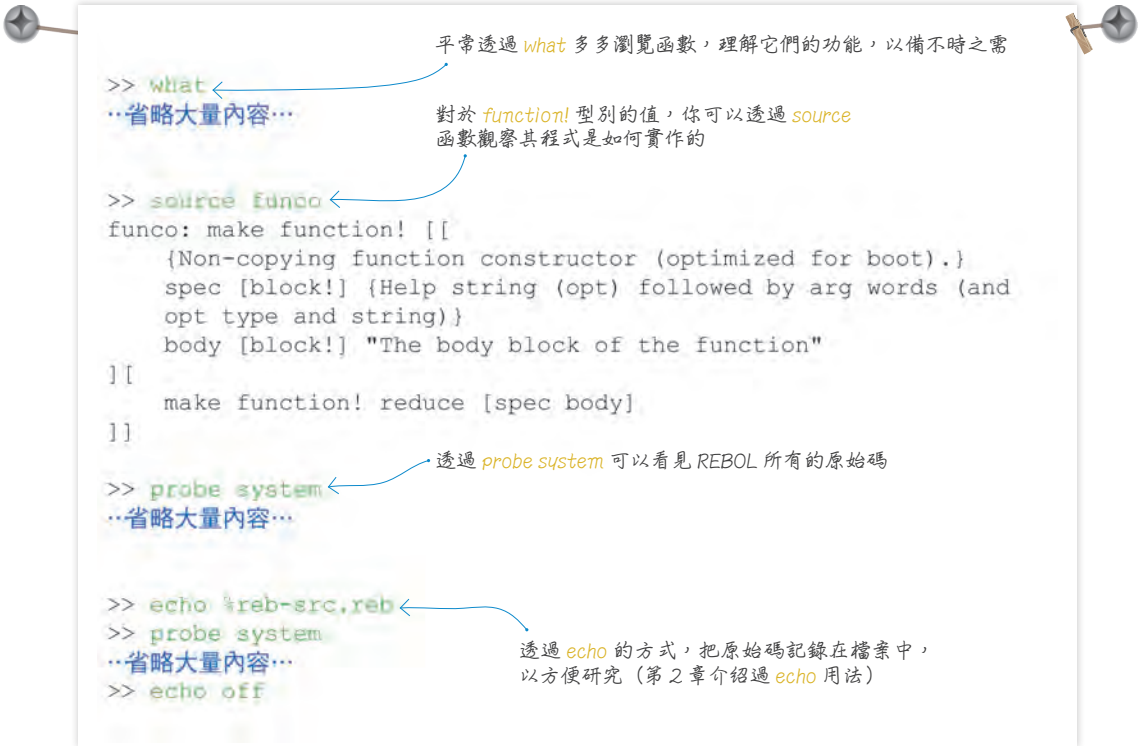
如果你確定做到了，開始閱讀這本書吧！





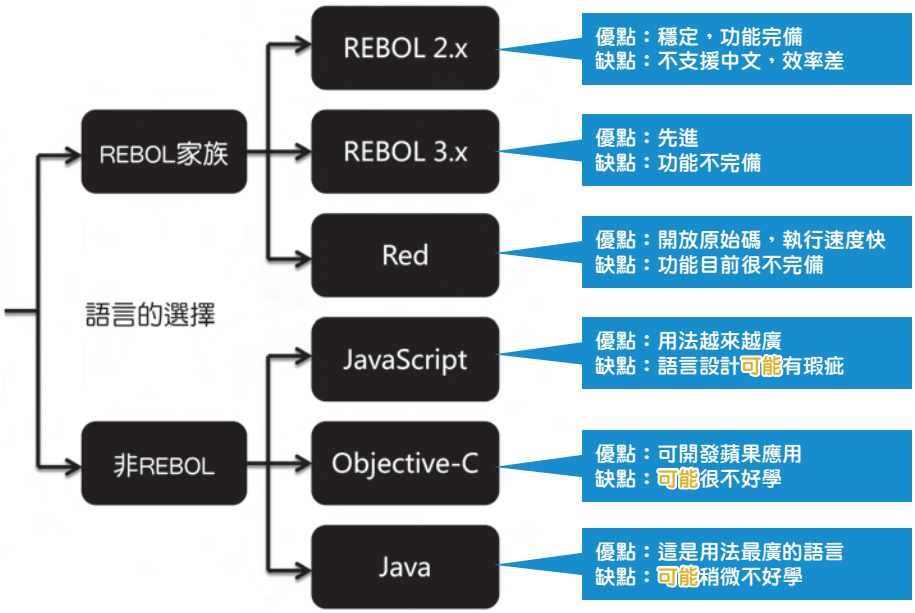
本書分成三篇，第一篇共有七章，講述程式設計的基本概念；第二篇共八章，詳細介紹 REBOL 的所有語法；第三篇共有九章，主要是透過程式， 加深對前兩篇的理解。第三篇還有另外兩個目的，分別是介紹一些常用的函數，以及透過一步一步的程式演進，讓讀者理解實際開發過程就是這樣對程式進行反覆修改、添加、刪除、優化。

讀完了這三篇，其實勉強算是完成了基礎。這只是個開端，接下來應該學習更多常用函數，並熟練這些函數，然後你就可以學習應用的開發了。我粗略地將應用分成 GUI、網路、資料庫、Web 四個領域。懂得這些應用領域的開發技術之後，你就可以開發大部分的程式了，例如：網站、遊戲 … 等。



雖然本書已經介紹過一些常用函數，但其實這還不太夠。我建議你透過 what 函數條列出所有的函數，並一一讀過這些說明，有一個大概的理解。然後找一些你覺得感興趣的函數，個別研究，漸漸累積自己熟識的函數。當你懂得函數越多，寫起程式就越輕鬆容易。

型別為 function! 的函數都是開放原始碼的，你可以透過 source 函數來看這些函數的定義方式。你也可以透過 probe system，看到 REBOL 所有的原始碼。



如果你學習程式設計，短期沒有就業的目的，那麼我會建議學習 REBOL 語言。目前比較重要的 REBOL 語言有三種：REBOL 2.7.8 版（簡稱 R2），REBOL 3.0 alpha 版（2.100.xxx，簡稱 R3），還有一個開放原始碼的版本 Red。

R2 不支援 Unicode，所以無法處理中文，這是嚴重的問題，所以我不建議使用它。R3 功能不完善，稍微不穩定，但設計理念比較先進。如果你日常想寫一些小工具程式幫你做事，那麼我會建議使用 R3，但如果要設計商務軟體，我目前不建議使用 R3（目前版本是 2.100.111）。讓我們期待後續 R3 版本解決不穩定的問題。

Red 是開放原始碼的 REBOL，且支援編譯器（關於編譯器請見第 1 章的說明）。我相當期待 Red 未來的發展，但目前它太年輕了，欠缺許多功能。

如果基於找工作的需要，我會建議學習 JavaScript，因為它的用途越來越廣，且也不難學。如果你想學習蘋果 iPhone 的程式開發，你必須學習 Objective-C 語言，但這語言比較難學一點。Java 則是用途最廣的語言，從安卓手機上的應用，到伺服器上的程式都可以做到，且方案都很成熟。

對於初學者，我推薦 R3、Red、JavaScript 這三個語言。

☑ 我認為程式設計很有趣、也很有用

如果你也這麼覺得，把這本書推薦給別人吧！