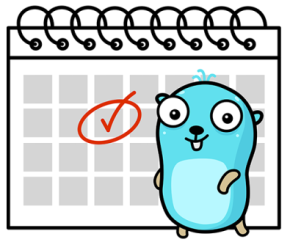
**F1741 完全自學！Go 語言 (Golang) 實戰聖經 初版勘誤／補充**



|  |  |
| --- | --- |
| **頁數** | **修正說明** |
| **1-8** | 範例程式 Example01.02 的 helloList 切片只有 4 個項目, 缺少 “Hello, world”, 這會使執行結果略有出入。至於可下載的範例程式則是正確的。 |
| **1-42** | 範例程式 Exercise01.13 的這一行  t := &&time.Time{}  在書上印出的結果為  time : &time.Time{wall:0x0, ext:0, loc:(\*time.Location)(nil)}  這在 Go 1.16 版原本是成立的, 但在本書出版後問世的 Go 1.17 給 time.Time 結構增加 GoString 方法, 這改變了它用 fmt 套件搭配 # 符號印出時的行為。在 1.17 版下會得到如下結果 (和印出非指標 time.Time 變數是一樣的)：  time.Date(1, time.January, 1, 0, 0, 0, 0, time.UTC) |
| **2-11** | 在 switch 敘述的 case 區塊中使用 fallthrough 時, 其實**不會**檢查下一個 case 的條件, 而是會直接執行該區塊內容。例如：  func main() {  day := time.Monday  switch day {  case time.Monday:  fmt.Println("星期一，猴子穿新衣")  fallthrough // 順便執行下一個 case 的內容  case time.Tuesday:  fmt.Println("星期二，猴子肚子餓")  default:  fmt.Println("日期不正確")  }  }  會輸出  星期一，猴子穿新衣  星期二，猴子肚子餓 ← 沒有檢查條件, 直接執行  這其實和在 C++ 的 case 區塊不使用 break; 時會連帶執行下一個 case 內容的行為是一樣的。  若你希望程式能繼續檢查下一個條件, 那麼可使用 if...else if。 |
| **3-4** | 範例程式 Exercise03.01 迴圈部分修正如下：  for \_, v := range pwR {  if unicode.IsUpper(v) {  hasUpper = true  }  if unicode.IsLower(v) {  hasLower = true  }  if unicode.IsNumber(v) {  hasNumber = true  }  if unicode.IsPunct(v) || unicode.IsSymbol(v) {  hasSymbol = true  }  } |
| **12-10** | 範例程式 Exercise12.01 的 import 區少列了 "syscall" 套件 (實際下載的範例程式中有)。若你使用 VS Code 並有安裝 Go 延伸套件, 它應該會在存檔時被自動加入。 |
| **17-3** | 從 Go 1.17 起, 編譯標籤 (build tags) 也可以寫成如下形式：  //go:build <標籤>  這個新語法也改用了 Go 語言的 and/or 算符, 使其語法上更好懂。例如, 17-4 的範例  // +build amd64,Darwin !386,windows  可以改成  //go:build (amd64 && Darwin) || (!386 && windows)  Go 1.17 支援這兩種語法, 但會優先使用 //go:build。詳情可以參考以下連結：<https://go.googlesource.com/proposal/+/master/design/draft-gobuild.md>。 |
|  |  |
|  |  |

此外 Go 1.18 起支援泛型 (generics)，簡易教學請見官方文件：https://go.dev/doc/tutorial/generics