台科大圖書
since 1997
www.tiked.com.tw

敬愛的老師您好：

PDF檔僅限學校教師搭配紙本教材用於
課堂教學，並未授權其他用途！！

優質好書 · 盡收眼底

台科大圖書 SINCE1997 TIKED.com.tw

http://www.tiked.com.tw/

互動教學 · 活潑有趣

hoodaa 呼答網

http://hoodaa.net/

製卷考試 · 一次 OK

OTAS 題測網 .tiked.com.tw

http://otas.tiked.com.tw/

線上學習 · 沒有距離

優勢學習網 www.use7.net

http://use7.net/

書籍配套 · 軟硬兼備

勁園 · 紅動

http://ipoemaker.com/

研習競賽 · 沒有問題

活動 研習 競賽 use360 全方位競賽網

http://use360.net/

前進專業 · 淬煉身價

勁園國際認證 JinYuan Global Certificate

http://certificate.jyic.net/

科技教育 · 先覺先知

iPOE 科技誌 intelligent · Public · Open · Easy-learning

http://ipoe.cc/

台科大圖書
SINCE 1997

(·) 達人必學

Python 3.X 程式設計

勁樺科技 ——— 編著

序言

網際網路的時代來臨後，只要有創意及結合資訊技能，都有機會在全世界的平台展現個人才華及發掘潛力，現在是行動 App 相當流行的世代，讓人人擁有程式設計的能力，已是國家教育政策的重點方向。由於 Python 語言的特性包括簡潔、易懂易學，因此不用擔心沒有任何程式語言相關的背景知識。再加上它的用途廣泛，應用範圍包括網頁設計、App 設計、遊戲設計、自動控制、生物科技、大數據等領域，因此 Python 語言是第一次學習程式語言的最佳首選。

本書是一本介紹 Python 各種語法與結合大量範例實作的學習教材，非常適合想對 Python 語言有完整認識的初學者閱讀。寫作風格以淺顯易懂的文字，循序漸進介紹 Python 語言必須認識的主題，從程式語言中最基礎的演算法與流程圖開始談起，接著帶領各位認識 Python 的環境安裝，之後陸續介紹基礎語法的各種主題。有了對 Python 基礎語法的認識之後，我們再為讀者討論更實用的進階主題，為了幫助各位強化 Python 程式語言的實作能力，在每一章節均規劃多個精彩範例及上機實作演練。

另外，本書以 Anaconda 套件組來做 Python 環境安裝，快速就能安裝 Python 及常用程式套件，而且不用擔心安裝失敗的問題。除了實用的內容之外，章節之後規劃了習題提供練習，相信本書是您學習 Python 的最佳教材。雖然本書力求資訊正確無誤，但仍恐有疏失之處，還望各位先進不吝指正。

編者　謹識

目錄 »

Chapter 1 Python 簡介與建置開發環境

1-1 Python 簡介 ……4
1-1-1 Python 語言的迷人特點 ……4
1-2 程式語言與程式設計 ……6
1-2-1 程式語言簡介 ……7
1-2-2 演算法與程式設計 ……9
1-2-3 流程圖 ……12
1-2-4 程式設計流程簡介 ……13
1-3 Python 的相關應用 ……15
1-3-1 Web 開發框架 ……15
1-3-2 數位科技整合開發 ……16
1-4 建置 Python 開發環境 ……21
1-4-1 下載 Anaconda 套件組 ……21
1-4-2 安裝 Anaconda ……23
1-5 Python 撰寫工具 ……26
1-5-1 透過命令提示字元視窗 ……26
1-5-2 Spyder 編輯器 ……28
1-5-3 IPython 命令視窗 ……31
1-6 print 輸出指令 ……33
〈基本輸出〉
檔名 print.py ……34
1-7 Python 程式撰寫風格 ……35
1-7-1 編碼宣告 ……37
1-7-2 程式註解 ……39
1-7-3 指令的分行和合併 ……41
1-8 上機實作演練－ Hello World ……42
〈我的第一支 Python 程式—Hello World〉
檔名 HelloWorld.py ……42
重點回顧 ……45
學習評量 ……47

Chapter 2 資料型態、變數與常數

2-1 變數命名與指派 ……52
2-1-1 變數宣告與指派 ……53
2-1-2 變數命名規則 ……53
2-1-3 靜態型別與動態型別 ……56
2-2 Python 的數值型態 ……58
2-2-1 整數 ……58
2-2-2 浮點數 ……58
2-2-3 布林值 ……60
〈轉換布林型態〉
檔名 bool.py ……60
2-3 常數 ……61
2-4 格式化輸出入功能 ……62
2-4-1 格式化輸出 ……62
〈整數輸出不同進制〉
檔名 print_%.py ……63
2-4-2 輸入函數：input() ……65
〈format 格式化輸出〉
檔名 format.py ……66
2-5 資料型態轉換 ……67
〈資料型態轉換〉
檔名 conversion.py ……68
2-6 上機實作演練
－零用金記帳小管家 ……69
2-6-1 程式範例描述 ……69
2-6-2 程式碼說明 ……70
〈零用金記帳小幫手〉
檔名 money.py ……70
重點回顧 ……71
學習評量 ……72

Chapter 3 運算式與運算子

3-1 算術運算子 76
〈攝氏溫度轉換華氏溫度〉
檔名 temperature.py 78
3-2 指派運算子 79
〈指派運算子綜合應用〉
檔名 assign_operator.py 82
3-3 比較運算子 84
〈比較運算子綜合應用〉
檔名 compare_operator.py 85
3-4 邏輯運算子 86
〈判斷成績及格 / 不及格〉
檔名 coursePassOrFail.py 89
3-5 位元運算子 90
〈位元運算子綜合應用〉
檔名 bit_operator.py 93
3-6 位移運算子 94
〈位元運算子與位移運算子綜合運用〉
檔名 bit_shift.py 95
3-7 運算子優先順序 96
〈運算子優先順序綜合應用〉
檔名 precedence.py 97
3-8 上機實作演練
一成績單統計小幫手 98
3-8-1 程式範例描述 98
3-8-2 讀取 CSV 檔 99
3-8-3 程式碼說明 102
〈成績單統計小幫手〉
檔名 Review_scores.py 105
重點回顧 106
學習評量 107

Chapter 4 流程控制

4-1 認識流程控制 112
4-1-1 循序結構 112
4-1-2 決策結構 112
4-1-3 迴圈結構 113
4-2 決策流程控制 114
4-2-1 if...else 條件式 114
〈閏年判斷〉
檔名 leapYear.py 116
4-2-2 多重選擇 117
〈偵測目前時間決定問候語〉
檔名 currentTime.py 118
4-2-3 巢狀 if 120
〈巢狀 if 的綜合使用範例〉
檔名 nested_if.py 121
4-3 迴圈 122
4-3-1 while 迴圈 122
4-3-2 for 迴圈 124
〈某一個數字範圍內 5 的倍數進行加總〉
檔名 sum5.py 126
4-3-3 巢狀迴圈 127
〈九九乘法表〉
檔名 99Table.py 128
4-4 continue 和 break 指令 129
4-4-1 break 指令 129
4-4-2 continue 指令 130
4-5 上機實作演練
一密碼驗證程式不求人 131
4-5-1 程式範例描述 131
4-5-2 程式碼說明 132
〈簡單的密碼驗證程式〉
檔名 password.py 132
重點回顧 133
學習評量 134

Chapter 5 字元與字串

5-1 建立字串 138

5-2 認識跳脫字元 140

〈顯示特殊字元〉

檔名 escape01.py 141

5-3 參數格式化輸出 142

5-4 字串索引與切片 145

5-4-1 透過索引值（index）取某個字元 145

5-4-2 透過切片取某段字串 146

5-4-3 透過 split () 方法分割字串 147

〈字串分割〉

檔名 split.py 147

〈將 26 個小寫英文字母反轉輸出〉

檔名 reverseString.py 148

〈將 26 個大寫英文字母反轉輸出〉

檔名 reverseStringBIG.py 149

〈將數字反轉輸出〉

檔名 reverseStringDigit.py 149

5-5 字串常用運算子 150

5-5-1 比較運算子 150

5-5-2 in 與 not in 運算子 151

5-6 字串常用函數與方法 151

5-6-1 計算字串長度—len() 函數 151

5-6-2 與大小寫有關的方法 152

〈大小寫轉換與首字大寫〉

檔名 upper.py 152

5-6-3 搜尋特定字串出現次數—count() 153

〈搜尋特定字串出現次數〉

檔名 count.py 153

5-6-4 移除字串左右特定字元—strip()、lstrip()、rstrip() 154

5-6-5 字串替換—replace() 155

5-6-6 搜尋字串—find() 與 index() 155

5-6-7 startswith() 方法與 endswith() 方法 157

〈查文字檔內特定字串組合出現次數〉

檔名 redcap.py 158

5-7 上機實作演練 — Open Data 資料擷取與應用 159

5-7-1 什麼是 Open Data 159

5-7-2 如何取得 Open Data 資料 160

5-7-3 程式範例描述 163

5-7-4 程式碼說明 164

〈Open Data 資料擷取與應用 — XML 格式〉

檔名 xml_parse.py 167

〈Open Data 資料擷取與應用 — JSON 格式〉

檔名 json_parse.py 168

重點回顧 169

學習評量 171

Chapter 6 函　數

6-1 認識函數 174

6-1-1 定義函數 174

6-1-2 呼叫函數 176

〈呼叫函數—傳入不定個數的引數〉

檔名 CallFunc_01.py 178

6-1-3 函數的回傳值 179

〈分帳程式〉

檔名 SplitBill.py 180

6-2 Python 引數傳遞的機制 181

〈Python 的引數傳遞〉

檔名 Argument.py 182

6-3 變數的有效範圍 183

6-4 遞迴函數 184

6-4-1 遞迴的定義 185

〈利用 for 迴圈計算 0!~n!〉

檔名 factorial_for.py 186

6-4-2 費伯那序列……187
〈n 項費伯那序列的遞迴程式〉
檔名 fib.py……188
6-4-3 河內塔問題……189
〈以遞迴式來實作河內塔〉
檔名 hanoi.py……193
6-5 認識排序……194
6-5-1 選擇排序法……195
〈選擇排序法〉
檔名 selection_sort.py……197
6-5-2 氣泡排序法……198
〈氣泡排序法〉
檔名 bubble_sort.py……200
6-5-3 排序函數—sorted()……202
〈分數排序〉
檔名 sorted_key1.py……203
6-6 lambda 運算式……204
6-7 上機實作演練
一美輪美奐的金字塔圖形……205
6-7-1 程式範例描述……205
6-7-2 程式碼說明……209
〈輸出金字塔圖形〉
檔名 Review_pyramid.py……209
重點回顧……211
學習評量……212

Chapter 7 進階資料型態

7-1 容器資料型態的比較……216
7-2 list 串列……217
7-2-1 建立串列……217
7-2-2 常用串列內建函數……219
〈用內建函數 sum() 計算總分〉
檔名 sum_score.py……220
7-2-3 常用串列運算子……221
〈將串列中奇偶數分離〉
檔名 evenAndOdd.py……222
7-2-4 串列常用方法……223
7-2-5 用 del 刪除變數與元素……226
7-2-6 二維及多維串列……226
〈二階行列式〉
檔名 twoArray.py……228
〈三維陣列〉
檔名 threeArray.py……229
7-3 tuple 序對（或稱元組）……230
7-3-1 建立序對……230
〈新建 tuple〉
檔名 tuple_create.py……231
7-3-2 序對的內建函數……231
7-3-3 序對運算子……232
7-3-4 Unpacking（拆解）與
Swap（交換）……233
〈序對交換 I〉
檔名 tuple.py……234
〈序對交換 II〉
檔名 tuple_price.py……235
7-4 dict 字典……236
7-4-1 建立字典……236
7-4-2 字典的基礎操作……236
7-4-3 適用字典的函數……238
7-4-4 適用字典的運算子……238
7-4-5 適用字典的處理方法……238
〈字典新增、移除與存取〉
檔名 dict_example.py……242
7-5 set 集合……243
7-5-1 建立集合……243
7-5-2 適用集合的運算子……243
〈集合的聯集、交集與差集〉
檔名 set_operations.py……244
7-5-3 適用集合的方法……244
7-6 上機實作演練 (一) 一簡易單字翻譯機
（GUI 圖形介面）……247

7-6-1 程式範例描述……247
7-6-2 程式碼說明……247
〈簡易單字翻譯機〉
檔名 review_dictionary.py……249
7-7 上機實作演練 (二) 一使用 Facebook API 取得粉絲專頁貼文……251
7-7-1 程式範例描述……251
7-7-2 程式碼說明……254
〈取得 FB 粉絲專頁貼文〉
檔名 review_FB_parse.py……254
重點回顧……256
學習評量……258

Chapter 8 模組與套件

8-1 匯入模組……263
8-1-1 匯入整個模組……263
〈import 關鍵字匯入模組〉
檔名 import.py……264
8-1-2 替模組取個別名……265
〈import...as 匯入模組指定別名〉
檔名 importAs.py……266
8-1-3 只匯入特定的函數……267
〈from...import 匯入特定函數〉
檔名 fromImport.py……267
〈Namespace 機制〉
檔名 namespace.py……269
8-2 自製模組……270
〈自製模組〉
檔名 moduleDiy.py……270
〈自製模組主程式〉
檔名 use_module.py……271
8-2-1 認識 Python 的 __name__ 屬性……272
〈__name__ 屬性的用法〉
檔名 moduleDiy_name.py……272
8-3 實用內建模組與線上輔助……273
8-3-1 os 模組……274
〈os 模組常用函數練習〉
檔名 os.py……274
8-3-2 sys 模組……275
〈sys 模組常用函數練習—取得命令列引數〉
檔名 test_argv.py……276
8-3-3 random 模組……278
〈random 模組常用函數練習〉
檔名 random.py……279
8-3-4 datetime 模組……280
〈求某月最後一天的日期〉
檔名 lastDayOfMonth.py……282
8-3-5 查看模組的路徑與檔名……283
8-3-6 套件線上說明文件……284
8-4 第三方套件……286
8-4-1 利用 pip 管理程式安裝套件……286
8-4-2 從 PyPI 網站安裝套件……287
8-5 上機實作演練
一樂透隨機投注暨對獎程式……289
8-5-1 程式範例描述……289
8-5-2 程式碼說明……290
〈樂透系統發財夢〉
檔名 review_Lottery.py……293
重點回顧……295
學習評量……297

Chapter 9 檔案與資料流

9-1 認識檔案 300
9-1-1 檔案的種類 300
9-1-2 認識檔案與目錄 301
9-1-3 絕對路徑與相對路徑 302
9-2 檔案的讀與寫 303
9-2-1 檔案的寫入 303
〈使用 open() 方法開啟檔案 I〉
檔名 openfile.py 305
〈使用 open() 方法開啟檔案 II〉
檔名 openfile1.py 306
9-2-2 檔案的讀取 307
〈先行判斷檔案是否存在〉
檔名 isfile.py 307
〈read() 方法實作〉
檔名 readfile.py 308
〈read(n) 方法實作〉
檔名 readnfile.py 309
〈seek() 方法實作〉
檔名 seek.py 310
〈先行判斷檔案是否存在〉
檔名 copyfile.py 311
〈readlines() 方法實作〉
檔名 readlines.py 312
9-3 二進位檔案 313
〈二進位資料實作〉
檔名 binary.py 313
9-4 綜合範例
—考古題高頻率單字統計 314
〈計算檔案中每個單字出現的次數〉
檔名 countword.py 316
重點回顧 317
學習評量 319

Chapter 10 錯誤與例外處理

10-1 程式錯誤類型 322
10-1-1 語法錯誤 322
10-1-2 執行期間錯誤 324
10-1-3 邏輯錯誤 326
10-2 認識例外 327
10-2-1 例外的類型 327
10-2-2 例外處理的時機 329
10-3 例外處理方式介紹 330
10-3-1 例外處理的語法 330
〈計算平均值〉
檔名 except01.py 331
〈在計算平均值的程式中加入例外處理〉
檔名 except02.py 332
10-3-2 以 raise 丟出指定的例外 334
〈利用 try/except 來捕捉 raise 丟出的例外〉
檔名 except03.py 334
10-4 綜合範例—以例外處理程式控制使用者輸入的數值 336
〈try/exception/finally 上機實作〉
檔名 except04.py 336
重點回顧 337
學習評量 338

Chapter 11 物件導向程式設計

11-1 認識物件導向 342
11-1-1 類別與物件 342
11-1-2 物件導向的特點 344
11-2 類別與物件 347
11-2-1 定義類別 347
11-2-2 類別實體化 348
〈定義類別〉
檔名 define_class.py 349
〈在類別中依據需求傳入不同型別的資料〉
檔名 define_class1.py 350
〈在類別中的方法中傳入參數計算值〉
檔名 return.py 351
11-2-3 將物件初始化 __init__() 方法 352
〈透過 __init__() 方法為物件設定初值〉
檔名 init.py 352
11-2-4 匿名物件 353
〈匿名物件實作〉
檔名 anonymous.py 353
11-2-5 私有屬性與方法 354
〈私有屬性的應用例子 I〉
檔名 private.py 355
〈私有屬性的應用例子 II〉
檔名 private1.py 356
11-3 繼承 357
11-3-1 單一繼承與多重繼承 359
〈單一繼承〉
檔名 single.py 360
〈在子類別中擴展父類別的方法 I〉
檔名 single1.py 361
〈在子類別中擴展父類別的方法 II〉
檔名 chain.py 362
〈多重繼承〉
檔名 multiple.py 363
11-3-2 覆寫 364
〈覆寫的實作〉
檔名 override.py 364
11-3-3 繼承相關函數 366
〈繼承函數 issubclass()〉
檔名 issubclass.py 367
11-3-4 多型 367
〈多型實作〉
檔名 polymorphism.py 368
11-3-5 組合 369
〈組合程式實作〉
檔名 composition.py 369
11-4 上機實作演練
一加退選程式設計實務 371
〈加退選課程實作〉
檔名 add_drop.py 372
重點回顧 373
學習評量 375

Chapter 12 開發 GUI 視窗程式

12-1 GUI tkinter 套件 378
12-1-1 匯入 tkinter 套件 378
12-1-2 建立主視窗 379
〈建立視窗的第一支程式〉
檔名 tk_main.py 380
12-1-3 佈局（Layout）方式 380
〈利用 pack 方法加入視窗〉
檔名 pack.py 381
〈利用 grid 方法加入視窗〉
檔名 grid.py 382
〈利用 place 方法加入視窗〉
檔名 place.py 384

12-2 GUI 元件 386
12-2-1 標籤（Label）元件 386
〈將 Label 元件指定以 pack 方法排列〉
檔名 label.py 388
12-2-2 按鈕（Button）元件 388
〈GUI 介面—按鈕〉
檔名 button.py 389
12-2-3 文字編輯（Entry）元件 392
〈GUI 介面—entry〉
檔名 entry.py 393
12-2-4 多行文字（Text）元件 394
〈GUI 介面—text〉
檔名 tktext.py 395
12-2-5 單選按鈕（Radiobutton）元件 396
〈GUI 介面—Radiobutton〉
檔名 tkRadiobutton.py 397
12-2-6 核取按鈕（Checkbutton）元件 398
〈GUI 介面—Checkbutton〉
檔名 tkCheckbutton.py 399
12-2-7 捲軸（Scrollbar）元件 401
〈GUI 介面—Scrollbar〉
檔名 tkscrollbar.py 401

12-3 對話方塊 403
12-3-1 訊息方塊 403
〈GUI 介面—訊息方塊〉
檔名 tkmessagebox.py 404
12-3-2 簡單型對話方塊 405
〈GUI 介面—簡單型對話方塊〉
檔名 tksimpledialog.py 406
12-3-3 檔案對話方塊 408
〈GUI 介面—檔案對話方塊〉
檔名 tk filedialog.py 408
12-3-4 調色盤方塊 410
〈GUI 介面—調色盤方塊〉
檔名 colorchooser.py 411

12-4 功能表 413
〈GUI 介面—Menu〉
檔名 tkMenu.py 416

12-5 繪製圖形（Canvas） 419
12-5-1 加入點陣圖 419
12-5-2 以 Canvas 元件繪製幾何圖形 420
〈以 Canvas 元件繪製幾何圖形〉
檔名 tkCanvas.py 420

12-6 上機實作演練
—以 GUI 實作簡易計算機 422
12-6-1 建立主視窗 423
12-6-2 版面配置（建立 Frame） 423
12-6-3 建置 Label、Entry 與 Button 元件 424
12-6-4 加入事件處理函數 427
12-6-5 捕捉錯誤訊息 428
〈簡易計算機（GUI）〉
檔名 Review_calculator.py 430

重點回顧 432
學習評量 434

附　錄　學習評量解答

本書範例說明

為方便讀者學習 Pythan 程式設計，本書範例提供相關檔案，請至本公司網站（http://www.tiked.com.tw）的圖書專區下載，或可直接於首頁的關鍵字欄輸入本書相關字（例：書號、書名、作者）進行書籍搜尋，尋得該書後即可下載範例檔案內容。各範例檔案與所屬章節如下表所示：

Chapter 1 Python簡介與建置開發環境

print.py	基本輸出
ch01.py	程式碼縮排
HelloWorld.py	我的第一支 Python 程式—Hello World

Chapter 2 資料型態、變數與常數

bool.py	轉換布林型態
print_%.py	整數輸出不同進制
format.py	format 格式化輸出
conversion.py	資料型態轉換
money.py	零用金記帳小幫手

Chapter 3 運算式與運算子

temperature.py	攝氏溫度轉換華氏溫度
assign_operator.py	指派運算子綜合應用
compare_operator.py	比較運算子綜合應用
coursePassOrFail.py	判斷成績及格 / 不及格
bit_operator.py	位元運算子綜合應用
bit_shift.py	位元運算子與移位運算子綜合運用
precedence.py	運算子優先順序綜合應用
scores.csv	成績單資料
Review_scores.py	成績單統計小幫手

Chapter 4 流程控制

leapYear.py	閏年判斷
currentTime.py	偵測目前時間決定問候語
nested_if.py	巢狀 if 的綜合使用範例
sum5.py	某一個數字範圍內 5 的倍數進行加總
enumerate.py	enumerate 函數
99Table.py	九九乘法表
password.py	簡單的密碼驗證程式

Chapter 5 字元與字串

escape.py	跳脫字元
escape01.py	顯示特殊字元
str_format.py	format 格式化輸出
slice.py	使用切片方法取某段字串
split.py	字串分割
reverseString.py	將 26 個小寫英文字母反轉輸出
reverseStringBIG.py	將 26 個大寫英文字母反轉輸出
reverseStringDigit.py	將數字反轉輸出
len.py	計算字串長度
left.py	取得字串長度
upper.py	大小寫轉換與首字大寫
count.py	搜尋特定字串出現次數
strip.py	去除字串首尾的字元
replace.py	字串替換
strfind.py	find() 搜尋字串
strindex.py	index() 搜尋字串
startswith.py	設定欲查詢字元的索引值
redcap.txt	查特定字串組合出現次數文字檔
redcap.py	查文字檔內特定字串組合出現次數
pm_xml.csv	資料擷取－XML 格式儲存資料
xml_parse.py	Open Data 資料擷取與應用－XML 格式
json_parse.py	Open Data 資料擷取與應用－JSON 格式
twisters.txt	學習評量簡答題 3－英文繞口令文字檔

Chapter 6 函數

func.py	定義函數
CallFunc.py	呼叫函數
CallFunc_01.py	呼叫函數－傳入不定個數的引數
return.py	回傳值
return01.py	多個回傳值
SplitBill.py	分帳程式
Argument.py	Python 的引數傳遞
factorial_for.py	利用 for 迴圈計算 0!~n!
fib.py	n 項費伯那序列的遞迴程式
hanoi.py	以遞迴式來實作河內塔
selection_sort.py	選擇排序法
bubble_sort.py	氣泡排序法
sorted_key.py	排序函數
sorted_key1.py	分數排序
sorted_key2.py	分數排序－傳入匿名函數
Review_pyramid.py	輸出金字塔圖形

Chapter 7 進階資料型態

sum_score.py	用內建函數 sum() 計算總分
evenAndOdd.py	將串列中奇偶數分離
twoArray.py	二階行列式
threeArray.py	三維陣列

tuple_create.py	新建 tuple
tuple.py	序對交換 I
tuple_price.py	序對交換 II
dict_example.py	字典新增、移除與讀取
set_operations.py	集合的聯集、交集與差集
dictionary_undone.py	GUI 圖形介面的製作練習
review _dictionary.py	簡易單字翻譯機
review _FB _parse.py	取得 FB 粉絲專頁貼文
twisters.txt	學習評量簡答題 1 －英文繞口令文字檔

Chapter 8 模組與套件

import.py	import 關鍵字匯入模組
importAs.py	import...as 匯入模組指定別名
fromImport.py	from...import 匯入特定函數
namespace.py	Namespace 機制
moduleDiy.py	自製模組
use_module.py	自製模組主程式
__pycache__	自製模組主程式資料夾
moduleDiy_name.py	__name__ 屬性的用法
os.py	os 模組常用函數練習
newFolder	os 模組常用函數練習資料夾
test_argv.py	sys 模組常用函數練習—取得命令列引數
random.py	random 模組常用函數練習
lastDayOfMonth.py	求某月最後一天的日期
review_Lottery.py	樂透系統發財夢

Chapter 9 檔案與資料流

phrase.txt	使用 open() 方法開啟檔案 I 文字檔
openfile.py	使用 open() 方法開啟檔案 I
openfile1.py	使用 open() 方法開啟檔案 II
isfile.py	先行判斷檔案是否存在 I
readfile.py	read() 方法實作
readnfile.py	read(n) 方法實作
seek.py	seek() 方法實作
introduct.txt	先行判斷檔案是否存在 II 文字檔
introduct1.txt	先行判斷檔案是否存在 II 文字檔
copyfile.py	先行判斷檔案是否存在 II
readlines.py	readlines() 方法實作
binary.py	二進位資料實作
bytedata	二進位資料實作檔案
exam.txt	計算檔案中每個單字出現的次數文字檔
countword.py	計算檔案中每個單字出現的次數

Chapter 10 錯誤與例外處理

except01.py	計算平均值
except02.py	在計算平均值的程式中加入例外處理
except03.py	利用 try/except 來捕捉 raise 丟出的例外
except04.py	try/exception/finally 上機實作

Chapter 11 物件導向程式設計

define_class.py	定義類別
define_class1.py	在類別中依據需求傳入不同型別的資料
return.py	在類別中的方法中傳入參數計算值
init.py	透過 __init__() 方法為物件設定初值
anonymous.py	匿名物件實作
private.py	私有屬性的應用例子 I
private1.py	私有屬性的應用例子 II
single.py	單一繼承
single1.py	在子類別中擴展父類別的方法 I
chain.py	在子類別中擴展父類別的方法 II
multiple.py	多重繼承
override.py	覆寫的實作
issubclass.py	繼承函數 issubclass()
polymorphism.py	多型實作
composition.py	組合程式實作
add_drop.py	加退選課程實作

Chapter 12 開發GUI視窗程式

tk_main.py	建立視窗的第一支程式
pack.py	利用 pack 方法加入視窗
grid.py	利用 grid 方法加入視窗
place.py	利用 place 方法加入視窗
label.py	將 Label 元件指定以 pack 方法排列
button.py	GUI 介面—按鈕
entry.py	GUI 介面—entry
tktext.py	GUI 介面—text
tkRadiobutton.py	GUI 介面—Radiobutton
tkCheckbutton.py	GUI 介面—Checkbutton
tkscrollbar.py	GUI 介面—Scrollbar
tkmessagebox.py	GUI 介面—訊息方塊
tksimpledialog.py	GUI 介面—簡單型對話方塊
tkfiledialog.py	GUI 介面—檔案對話方塊
myfile.txt	檔案對話方塊文字檔
colorchooser.py	GUI 介面—調色盤方塊
tkMenu.py	GUI 介面—Menu
tkCanvas.py	以 Canvas 元件繪製幾何圖形
frame.py	建立 Frame
Review_calculator.py	簡易計算機（GUI）

版權聲明

本書提及之各註冊商標，分屬各註冊公司所有，不再一一說明。書中提及之共享軟體或公用軟體，其著作權皆屬原開發廠商或著作人，請於安裝後詳細閱讀各工具的授權和使用說明。範例檔案僅供讀者練習之用，與各軟體的著作權和其他利益無涉，如果在使用過程中，因軟體所造成的任何損失，與本書作者和出版商無關。

Chapter 1

Python 簡介與建置開發環境

本章的目的是幫助幾乎沒有 Python 程式設計的新手，成長為一位 Python 設計者的入門課，就算是一般的電腦使用者，也可以快速透過簡短的 Python 程式達成設計目的。本章將會跟各位討論 Python 的簡介與特殊應用，及如何建置完成 Python 開發環境的完整過程。

學習大綱

- Python 的特色
- 程式語言簡介
- 演算法概念
- 流程圖
- Python 的應用
- 建置 Python 開發環境
- 基本輸入與輸出
- Spyder 編輯器
- IPython 命令視窗
- Python 程式撰寫風格

由於目前程式設計能力已經被看成是國力的象徵，連教育部都將撰寫程式將列入國高中學生必修課程，寫程式不再是資訊相關科系的專業，而是全民的基本能力，擁有程式設計實作能力，已是各學校資訊教育的首要重點。相較於許多其它的傳統程式語言。隨著物聯網與大數據分析的火紅，讓在數據分析與資料探勘有著舉足輕重地位的 Python，越來越受到科技界的歡迎。

Tips

物聯網（Internet of Things, IOT）是近年資訊產業中一個非常熱門的議題，是將各種具有感測設備的物品，例如 RFID、環境感測器、全球定位系統（GPS）等裝置與網際網路結合起來，並透過網路技術讓各種實體物件、自動化裝置彼此溝通和交換資訊。

大數據（big data），由 IBM 於 2010 年提出，是指在一定時效內進行大量且多元性資料的取得、分析、處理、保存等動作，主要特性包含三種層面：大量性（Volume）、速度性（Velocity）及多樣性（Variety）。由於數據的來源有非常多的途徑，大數據的格式也將會越來越複雜。

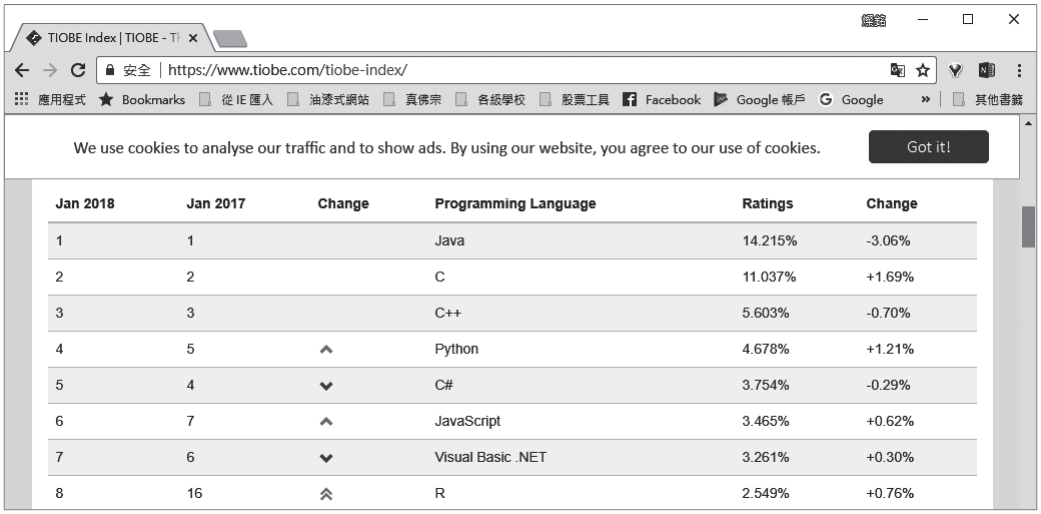
在目前這個雲端運算（Cloud Computing）的時代，沒有最好的程式語言，只有是否適合的程式語言，程式語言本來就只是工具，從來都不是重點，不過在統計分析與資料探勘有著舉足輕重地位的 Python，近年來人氣不斷飆升，並成為熱門高階程式語言排行榜的常勝軍，也可以說是現在最流行的機器學習（Machine Learning，ML）語言，可用於執行基本的機器學習任務，而且使用者也能在網路上找到大量的相關資源。

Tips

雲端其實就是泛指「網路」，因為通常工程師對於網路架構圖中的網路習慣用雲朵來代表不同的網路。雲端運算（Cloud Computing）就是將運算能力提供出來作為一種服務，只要使用者能透過網路登入遠端伺服器進行操作，就能使用運算資源。

機器學習（Machine Learning，ML）是通過演算法來分析數據、在大數據中找到規則，給予電腦大量的「訓練資料（Training Data）」，可以發掘多資料元變動因素之間的關聯性，進而自動學習並且做出預測，充分利用大數據和演算法來訓練機器，機器再從中找出規律，學習如何將資料分類。

Python 的優點包括物件導向程式設計（Object-Oriented Programming, OOP）、直譯、跨平台等特性，加上豐富強大的套件模組與免費開放原始碼，各種領域的使用者都可以找到符合需求的套件模組，涵蓋了網頁設計、App 設計、遊戲設計、自動控制、生物科技、大數據等領域，因此非常適合作為入門學習程式語言，目前在網路上都擁有非常活躍的社群以及擁護者。底下是 TIOBE Software（https://www.tiobe.com/tiobe-index/）於 2018 年 1 月份所公布的世界程式語言排名可以看出，Python 的人氣指標已升到第 4 名。



Jan 2018	Jan 2017	Change	Programming Language	Ratings	Change
1	1		Java	14.215%	-3.06%
2	2		C	11.037%	+1.69%
3	3		C++	5.603%	-0.70%
4	5	^	Python	4.678%	+1.21%
5	4	v	C#	3.754%	-0.29%
6	7	^	JavaScript	3.465%	+0.62%
7	6	v	Visual Basic .NET	3.261%	+0.30%
8	16	^^	R	2.549%	+0.76%

Tips

物件導向程式設計的主要精神就是將存在於日常生活中舉目所見的物件（object）概念，應用在軟體設計的發展模式（software development model）。也就是說，OOP 讓各位從事程式設計時，能以一種更生活化、可讀性更高的設計觀念來進行，並且所開發出來的程式也較容易擴充、修改及維護。

1-1 Python 簡介

Python 的英文原意是蟒蛇（發音 /'pa θ n/ 接近「派森」），是在 1990 年初，由荷蘭工程師吉多 • 范羅蘇姆（Guido van Rossum）創始開發，但 Guido 並不是因為喜歡蟒蛇而取這個名字，據他的說法是取自個人很喜愛的 BBC 著名的喜劇影集「Monty Python's Flying Circus（蒙提·派森的飛行馬戲團）」。雖然 Python 的名稱來源不是大蟒蛇，但 Python 軟體基金會還是採用了兩條蛇作為 logo，如右圖。

1-1-1 Python 語言的迷人特點

自從程式語言的發展進入到了高階語言後，出現了許多不同類型的程式語言，例如 C、C++、Java、PHP、JavaScript、C#、Delphi 等等，都具有不同的特色，用途也有很大的差異。就以 C 語言為例，C 語言雖然是一種高階語言，但兼具低階語言的特性，亦可算是一種中階語言，Unix/Linux 作業系統就是由 C 語言開發而成，主要優點有：程式輕薄短小、效能佳、可針對記憶體直接處理。

另外像 Java 語言則是參考 C/C++ 特性所開發的程式語言，標榜跨平台、穩定及安全等特性，主要應用領域為網際網路、無線通訊及電子商務，Java 也是一種物件導向程式設計的高階語言。

Guido 開發 Python 的動機，緣起於想設計出一種任何人都能輕鬆學習，屬於一般用途的高階語言，就分類上來說它是一個解譯式的動態程式語言，它不僅優雅簡潔，更具備有開發快速、容易閱讀、功能強大等優點。同時 Python 融合了多種程式語言的風格，加上 Python 是用 C 語言所撰寫，採用開放原始碼的概念，由於 C 的可移植性，使得 Python 能夠在任何 ANSI C 編譯器的平台執行。

以下我們為各位列出 Python 的迷人特點：

程式碼簡潔易讀

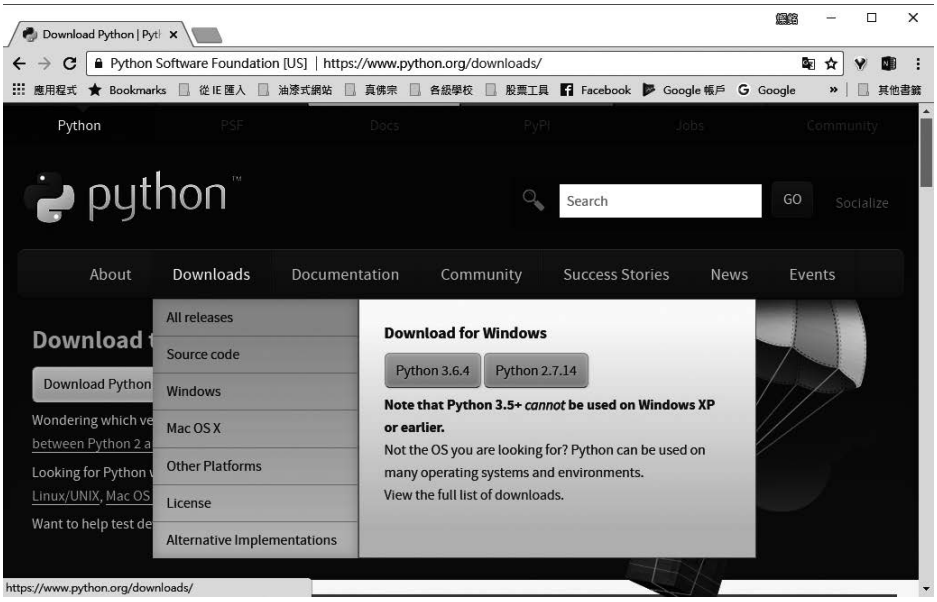
Python 開發的目標之一是讓程式碼像讀本書那樣容易理解，也因為簡單易記、程式碼容易閱讀的優點，在寫程式的過程中能專注在程式本身，而不是如何去寫，程式開發更有效率，團隊協同合作也更容易整合。

```
# -*- coding: utf-8 -*-
"""
程式名稱：輸出金字塔圖形
"""

def drawpyramid():        #定義drawpyramid函數
    h = int( input("請輸入您要顯示的金字塔層數(1~10):") )
    s = input("請輸入要顯示的符號:")

    for n in range(1,h+1):
        str="{0}{1:^20}"
        print(str.format(n,s*(n*2-1)))

    a=input("按x鍵離開, 按任意鍵繼續.")
    if a != "x":
        drawpyramid()    #呼叫drawpyramid函數
    else:
        print("Goodbye!!")


drawpyramid()    #呼叫drawpyramid函數
```

Python 語言風格相當簡潔易讀

跨平台

Python 程式可以在大多數的主流平台執行，具備在各個作業系統平台之間的高相容性及可移植性，不管是 Windows、Mac OS、Linux 以及手機，都有對應的 Python 工具，在 https://www.python.org/downloads/ 下載頁面顯示出支援各種平台的 Python 開發工具。例如你的個人電腦作業系統使用的是 MacOS 或是 Linux，只要直接在命令列（終端機 Terminal）下鍵入 python，就可以立即使用 Python 程式語言來設計程式。



自由／開放原始碼

所有 Python 的版本都是自由 / 開放原始碼（Free and Open Source），簡單來說，各位可以自由地閱讀、複製及修改 Python 的原始碼，或是在其他自由軟體中使用 Python 程式。

多重思維的程式語言

Python 具有物件導向的特性，像是類別、封裝、繼承、多型等設計，不過它卻不像 Java 這類的物件導向語言強迫使用者必須用物件導向思維寫程式，Python 是多重思維（Multi-paradigm）的程式語言，允許我們使用多種風格來寫程式，程式撰寫更具彈性，就算不懂物件導向觀念，也不會成為學習 Python 的絆腳石。

擴充能力強的膠水語言

由於 Python 十分容易上手，不但具有當做描述語言的能力，並且提供了豐富的 API 和套件，程式設計師能夠輕鬆地編寫擴充模組，也可以整合到其它語言的程式內使用，因此也有人將 Python 視為膠水語言（Glue Language）的一種，能把相關功能的程式（可能由不同的程式語言所撰寫）如同膠水般「黏」在一起。

1-2 程式語言與程式設計

程式語言是一種人類用來和電腦溝通的語言，也就是用來指揮電腦運算或工作的指令集合。就如同中文、英文、日文等語言一樣，無論哪一種語言都有字彙與文法。程式語言是一行行的指令（statement）與程式碼（code）組合而成，可以將人類的思考邏輯和溝通語言轉換成電腦能夠了解的語言，而「程式設計」就是透過程式語言的撰寫與執行，來達到使用者的工作需求，寫程式可以讓工作更有效率，像是每天必須重複做的工作，就可以找出規則寫一個程式來自動執行。

1-2-1 程式語言簡介

每一代的語言都有其特色，不論任何一種語言都有其專用語法、特性、優點及相關應用的領域。以發展過程來區分大致可分為「低階語言」與「高階語言」兩大類；低階語言又可分為「機器語言」及「組合語言」。

機器語言

機械語言（Machine Language）是最早期的程式語言，由 1 和 0 兩種符號構成，也是電腦能夠直接閱讀與執行的基礎語言，也就是任何程式在執行前都必須被轉換為機械語言。例如「10111001」代表「設定變數 A」，而「00000010」代表「數值 2」。當指示電腦將變數 A 設定為數值 2 時，機械語言寫法為：

```
10111001 （設定變數 A）
00000010 （將 A 設定為數值 2）
```

任何程式在執行前都必須被轉換為機器語言，因為電腦能直接執行，所以執行速度快，在處理資料上十分有效率，但可讀性低，學習不太容易。不過每一家電腦製造商，往往因為電腦硬體設計的不同而開發不同的機器語言。這樣不但使用不方便，可讀性低也不容易維護，並且不同的機器平台，編碼方式都不盡相同。

組合語言

組合語言（Assembly Language）指令比機器語言指令有意義多了，但與機器語言仍然成一對一的對應，所以一樣被歸類為低階語言（Low-level Language），和機器語言相比，更方便人類記憶與使用。由於組合語言與硬體有著密切關係，根據 CPU（中央處理器）或單晶片的指令集不同，語法也不相同，程式人員除了對指令要相當瞭解之外，還必須熟悉硬體，每一種系統的組合語言都不一樣，以 PC 而言，用的是 80×86 的組合語言。

由組合語言所撰寫的程式，電腦無法直接辨識，必須透過組譯程式（Assembler），將指令轉換成電腦可以識別的機器語言。例如 MOV 指令代表設定變數內容、ADD 指令代表加法運算、SUB 指令代表減法運算，如下所示：

```
MOV  A , 2   （變數 A 的數值內容為 2）
ADD  A , 2   （將變數 A 加上 2 後，將結果再存回變數 A 中，如 A=A+2）
SUB  A , 2   （將變數 A 減掉 2 後，將結果再存回變數 A 中，如 A=A-2）
```

高階語言

由於低階語言不易閱讀，為了能更方便快速地使用程式語言，因此又產生更接近生活用語的程式語言，稱為高階語言（High-level Language），高階程式語言就是比低階語言更容易懂的程式語言。舉凡是 Python、Fortran、COBOL、Java、Basic、C 或是 C++，都是高階語言的一員。一個用高階語言撰寫而成的程式碼，必須經過編譯器（Compiler）或直譯器（Interpreter）翻譯為電腦能解讀、執行的低階機器語言的程式，也就是執行檔，才能被 CPU 所執行。相對於組合語言，高階語言雖然執行速度較慢，但語言本身易學易用，因此被廣泛應用在商業、科學、教學及軍事等相關的軟體開發上。

Tips

編譯與直譯的不同

編譯式語言會先使用編譯器檢查整個程式，完全沒有語法錯誤之後，再連結相關資源輸出可執行檔（executable file）。一旦經過編譯後所產生的執行檔，在執行過程中不需要再次翻譯，因此執行效率較高，例如 C、C++、PASCAL、FORTRAN、COBOL 等都是屬於編譯式語言。

直譯式語言又稱為解譯式語言，是利用直譯器對程式碼一邊解讀原始碼，一邊執行，每解譯完一行程式碼，才會繼續解譯下一行程式碼。在解譯過程中，如果發生語法錯誤，那麼解譯動作將會立刻停止，例如 HTML、JavaScript、Python 等等。

非程序性語言

「非程序性語言」（Non-procedural Language）也稱為第四代語言，英文簡稱為 4GLS，特點是它的指令和程式真正的執行步驟沒有關聯。程式設計者只須將自己打算做什麼表示出來即可，而不須去理解電腦的執行過程。通常應用於各類型的資料庫系統，如醫院的門診系統、學生成績查詢系統等等。例如資料庫的結構化查詢語言（Structural Query Language，簡稱 SQL）就是一種第四代語言，清除資料命令相當簡單，如下所示：

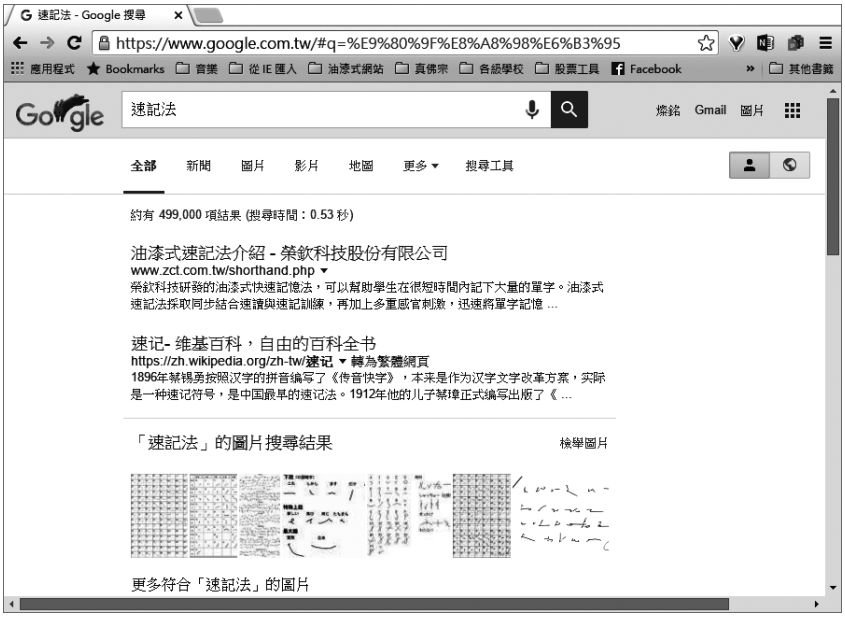
```
DELETE FROM employees
WHERE employee_id = 'C800312' AND dept_id = 'R01';
```

人工智慧語言

稱為第五代語言，或稱為自然語言（Natural Language），它是程式語言發展的終極目標，當然依目前的電腦技術尚無法完全辦到，因為自然語言使用者口音、使用環境、語言本身的特性（如一詞多義）都會造成電腦在解讀時產生不同的結果。所以自然語言必須搭配人工智慧（Artificial Intelligence, AI）技術來發展。

1-2-2 演算法與程式設計

演算法（algorithm）是程式設計領域中最重要的關鍵，每個人每天都會用到一些演算法，演算法是人類利用電腦解決問題的技巧之一，演算法並不是僅僅用於電腦領域上，包括在數學、物理或者是每天的生活上也有極大的用處。日常生活中也有許多工作都可以利用演算法來描述，例如員工的工作報告、寵物的飼養過程、廚師準備美食的食譜、學生的功課表等，甚至於連我們平時經常使用的搜尋引擎都必須藉由不斷更新演算法來運作。



搜尋引擎藉由不斷更新演算法來運作

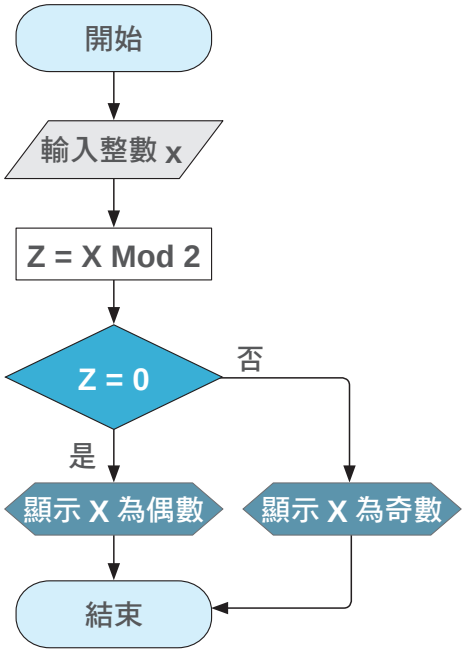
從程式語言實作的角度，不論你採用哪一種程式語言，程式能否有效率的完成任務，演算法絕對佔了十分重要的關鍵。同樣一個問題，每個人的解法可能不同，執行效率也會不同，好的演算法能夠以最精簡的程式碼，達到最佳的執行效率。

在韋氏辭典中將演算法定義為：「在有限步驟內解決數學問題的程式。」如果運用在計算機領域中，我們也可以把演算法定義成：「為了解決某一個工作或問題，所需要有限數目的機械性或重複性指令與計算步驟。」認識了演算法的定義後，我們還要說明描述演算法所必須符合的五個條件：

演算法特性	內容與說明
輸入（Input）	0 個或多個輸入資料，這些輸入必須有清楚的描述或定義。
輸出（Output）	至少會有一個輸出結果，不可以沒有輸出結果。
明確性（Definiteness）	每一個指令或步驟必須是簡潔明確而不含糊的。
有限性（Finiteness）	在有限步驟後一定會結束，不會產生無窮迴路。
有效性（Effectiveness）	步驟清楚且可行，可以正確地解決問題。

接下來要來思考到該用什麼方法來表達演算法最為適當呢？其實演算法的主要目的是在提供給人們閱讀了解所執行的工作流程與步驟，只要能夠清楚表現演算法的五項特性即可。常用的演算法表示工具如下：

- **一般文字指令**：中文、英文、數字等。文字指令法的特色是使用文字或語言指令來說明演算步驟。
- **虛擬語言（Pseudo-Language）**：接近高階程式語言的寫法，也是一種不能直接放進電腦中執行的語言。一般都需要一種特定的前置處理器（preprocessor），或者用手寫轉換成真正的電腦語言，經常使用的有 SPARKS、PASCAL-LIKE 等語言。
- **流程圖（Flow Diagram）**：算是一種通用的表示法，必須使用某些圖形符號。例如輸入一個數值，並判別是奇數或偶數。
- **程式語言**：目前演算法也能夠直接以可讀性高的高階語言來表示，例如 Visual C#、Java 語言、Python 語言、Visual Basic 語言、C 語言、C++ 語言，在本書中將以 Python 語言來實作。



Tips

演算法和程式有何不同？與流程圖又有什麼關係？

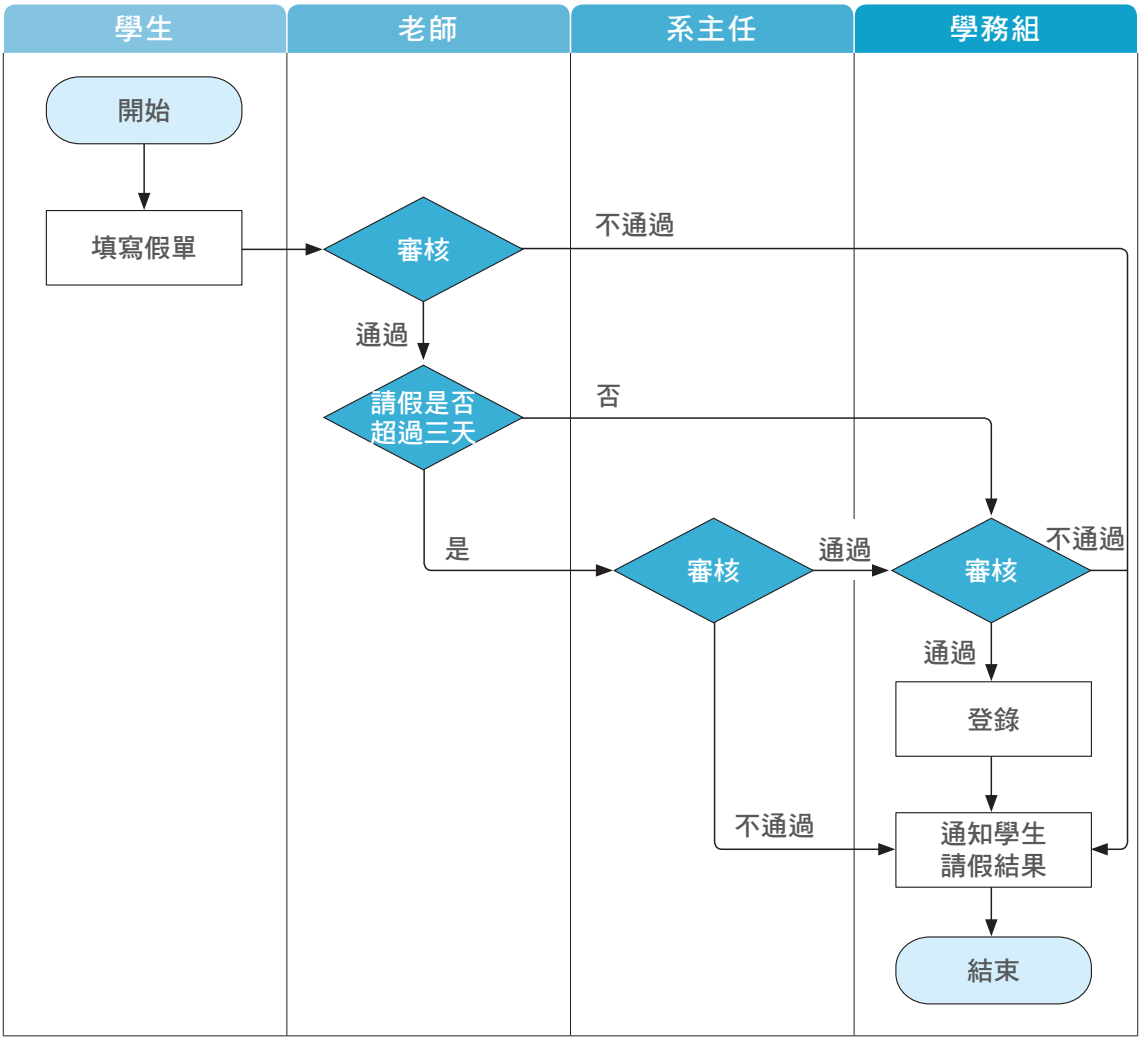
演算法和程序是有所區別，因為程式不一定要滿足有限性的要求，如作業系統或機器上的運作程序。除非當機，否則永遠在等待迴路（waiting loop），這也違反了演算法五大原則之一的「有限性」。另外，只要是演算法都能夠利用程式流程圖表現，但因為程序流程圖可包含無窮迴路，所以無法利用演算法來表達。

為了方便撰寫程式，將演算法透過流程圖（flow chart）來描述，是目前最普遍的方式，接下來我們就特別來看看流程圖的用法。

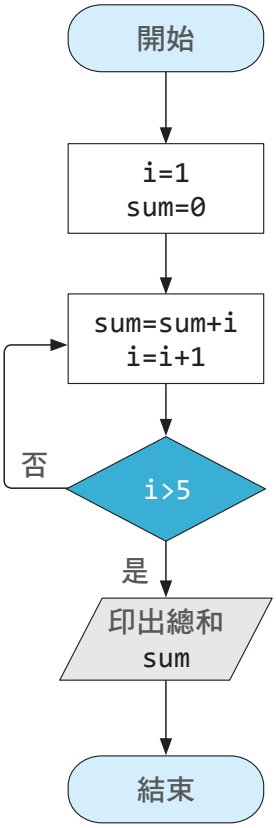
1-2-3 流程圖

流程圖是使用圖形符號來表示解決問題的步驟，繪製流程圖有助於程式的修改與維護。特別是當不同的程式開發人員撰寫程式時，能快速了解程式流程，有助於協同合作開發程式與有助程式移交工作。流程圖有很多種類型，程式開發最常用的是「系統流程圖」以及「程式流程圖」。

「系統流程圖」（system flowchart）是用來描述系統的完整流程，包含資訊流及作業流程，即人員、設備、各部門間的作業關係，例如學生請假會經過一些簽核流程，透過系統流程圖就能清楚了解完整的簽核流程，如下圖所示。



而「程式流程圖」（program flowchart）是用來描述程式的邏輯架構，從程式流程圖可以看出程式內的各種運算及執行順序。例如求 1+2+3+4+5 的演算法，可以繪製成如右的程式流程圖。



為了流程圖之可讀性及一致性，目前通用美國國家標準協會 ANSI 制定的統一圖形符號，以下簡單說明一些常見的符號：

名稱	說明	符號
起止符號	表示程式的開始或結束	
輸入 / 輸出符號	表示資料的輸入或輸出的結果	
程序符號	程序中的一般步驟，程序中最常用的圖形	
決策判斷符號	條件判斷的圖形	
文件符號	導向某份文件	
流向符號	符號之間的連接線，箭頭方向表示工作流向	
連結符號	上下流程圖的連接點	

1-2-4 程式設計流程簡介

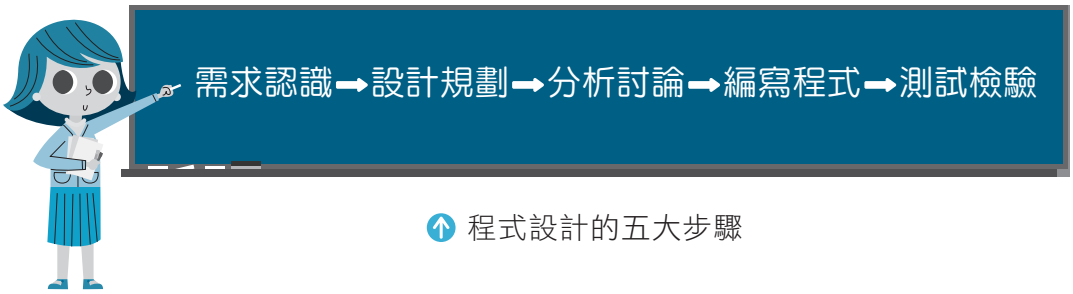
所謂程式，是由合乎程式語言語法規則的指令所組成，而程式設計的目的就是透過程式的撰寫與執行來達到使用者的需求。或許各位讀者認為程式設計的主要目的只是要「跑」出正確結果，而忽略了包括執行績效或者日後的維護成本，其實這是不清楚程式設計的真正意義。

程式設計時必須利用何種程式語言表達，通常可根據主客觀環境的需要，並無特別規定。一般評斷程式語言好壞的四項如下：

- **可讀性（Readability）高**：閱讀與理解都相當容易。
- **平均成本低**：成本考量不侷限於編碼的成本，還包括了執行、編譯、維護、學習、除錯與日後更新等成本。
- **可靠度高**：所撰寫出來的程式碼穩定性高，不容易產生邊際錯誤（Side Effect）。
- **可撰寫性高**：對於針對需求所撰寫的程式相對容易。

對於程式設計領域的學習方向而言，無疑就是朝向有效率、可讀性高的程式設計成果為目標。一個程式的產生過程，可區分為以下五個設計步驟：

1. **需求認識（requirements）**：了解程式所要解決的問題是什麼，有哪些輸入及輸出等。
2. **設計規劃（design and plan）**：根據需求，選擇適合的資料結構，並以任何的表示方式寫一個演算法以解決問題。
3. **分析討論（analysis and discussion）**：思考其他可能適合的演算法及資料結構，最後再選出最適當的標的。
4. **編寫程式（coding）**：把分析的結論，寫成初步的程式碼。
5. **測試檢驗（verification）**：最後必須確認程式的輸出是否符合需求，這個步驟得細部地執行程式並進行許多的相關測試。



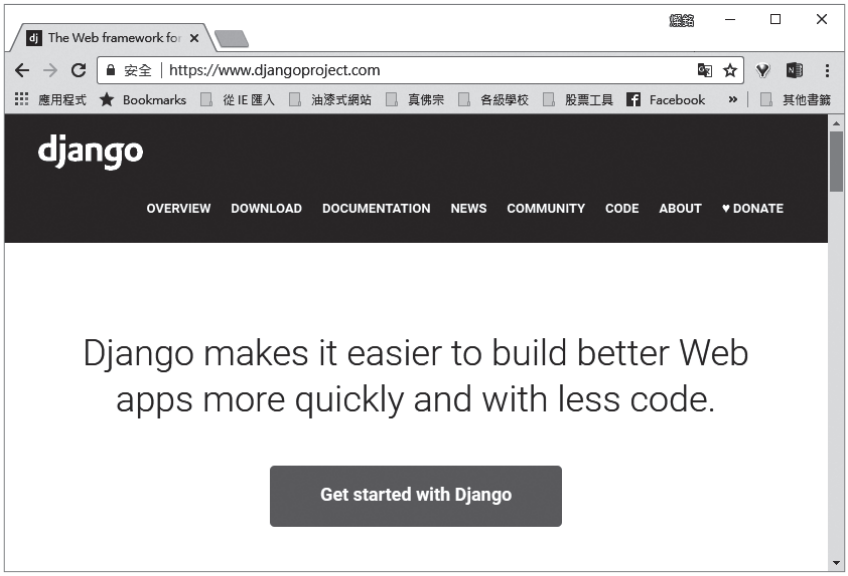
程式設計的五大步驟

1-3 Python 的相關應用

Python 應用的領域十分廣泛，除了本身擁有功能完備的標準函數庫外，也能加入第三方函數庫。再加上 Python 擁有龐大的開放式資源社群，各地的社群會定期舉辦聚會，彼此交流，Python 的應用可說是無遠弗屆，看完它應用的範圍，相信更能提高學習的動力。

1-3-1 Web 開發框架

Web 程式開發包括前端與後端技術，光是前端就有 html、JavaScript 及 CSS 等技術，更別提後端林林總總的技術，Web 框架簡單來說就是為建立 Web 應用程式制定了一套規範，它簡化了技術上的細節，只要運用 Web Framework 模組，就能輕易地建構出動態網站。在 Python 領域知名的 Web 框架常見的有 Django、CherryPy、Flask、Pyramid、TurboGear 等等，例如下圖的 Django Web 開發框架的官網（https://www.djangoproject.com/）。



Tips

JavaScript：是一種直譯式的描述語言，是在客戶端（瀏覽器）解譯程式碼，內嵌在 HTML 語法中，當瀏覽器解析 HTML 文件時就會直譯 JavaScript 語法並執行，JavaScript 不只能讓我們隨心所欲控制網頁的介面，也能夠與其他技術搭配做更多的應用。

Cascading Style Sheets, CSS：一般稱之為串聯式樣式表，其作用主要是為了加強網頁上的排版效果（圖層也是 CSS 的應用之一），可以用來定義 HTML 網頁上物件的大小、顏色、位置與間距，甚至是為文字、圖片加上陰影等功能。

1-3-2 數位科技整合開發

資訊科技不斷進步，數位化應用從日常生活到工作處處可見，各種裝置與網際網路、行動科技整合密切，甚至有統計預測未來十年內，許多工作將會被機器人取代。在各種數位化應用技術中，「大數據分析」、「物聯網」及「人工智慧」是最受關注的領域。Python 有許多容易擴增的資料分析與機器學習模組庫（library），像是 NumPy、Matplotlib、Pandas、Scikit-learn、SciPy、PySpark 等等，這也讓 Python 成為資料解析與機器學習主要語言之一。底下就來認識「物聯網」、「大數據分析」及「人工智慧」這些應用領域。

物聯網

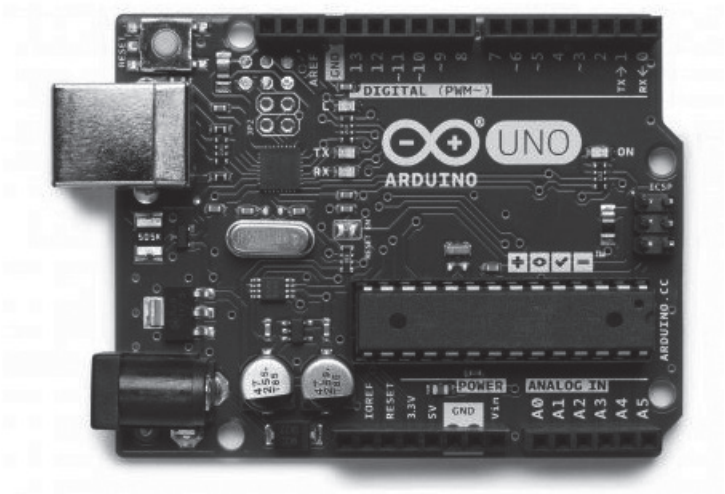
物聯網（Internet of Things, IoT）是近年資訊產業中一個非常熱門的議題，物聯網最早是在 1999 年時由學者 Kevin Ashton 所提出，是讓生活中的物品能透過互聯互通的傳輸技術進行感知與控制。例如智慧家電可讓使用者遠端透過 APP 操控電冰箱、空調等電器，不僅可以遠端操控還能自動調節。又例如 RFID、環境感測器、全球定位系統（GPS）雷射掃描器等種種裝置與網際網路結合起來而形成的一個巨大網路系統，全球所有的物品都可以透過網路主動交換訊息，透過網際網路技術讓各種實體物件、自動化裝置彼此溝通和交換資訊，漸漸讓現代人的生活正進入一個始終連接（Always Connect）的網路世代，其最終的目標則是要打造一個先進的智慧城市。

近來百貨公司、便利商店使用的 iBeacon 技術，也是物聯網的應用之一，商家只要在店內部署多個 Beacon 裝置，利用機器學習技術來對消費者進行觀察，賣場不只是提供產品，更應該領先與消費者互動，一旦顧客進入訊號區域時，就能夠透過手機上 App，對不同顧客進行精準的「個人化習慣」分眾行銷，提供「最適性」服務的體驗。由於 iBeacon 的覆蓋範圍較小，因此也有人把這樣的技術應用稱為**微定位**（micro-location）。

Tips

Beacon 是種低功耗藍牙技術（Bluetooth Low Energy, BLE），藉由室內定位技術應用，可做為物聯網和大數據平台的小型串接裝置，具有主動推播行銷應用特性，比 GPS 有更精準的微定位功能，可包括在室內導航、行動支付、百貨導覽、人流分析及物品追蹤等近接感知應用。

5G 時代即將來臨，它將帶動數位匯流，快速翻轉整個產業面貌，物聯網概念將為全球消費市場帶來新衝擊。在我們生活當中，已經有許多整合物聯網的技術與應用，像是醫療照護、公共安全、環境保護、政府工作、平安家居、空氣汙染監測、土石流監測等領域。例如物聯網提供了遠距醫療系統發展的基礎技術，當有患者生病時，透過智慧型手機或特定終端測量設備，對於各種發病症狀，醫院的系統中會自動進行比對與分析，提出初步解決方案以避免病症加重。另外，Python 在 Arduino 與 Raspberry Pi 的支援之下，也能控制硬體，打造各種物聯網應用。



大數據分析

物聯網的另一種應用是蒐集數據，加以分析，進而對用戶的行為或環境指標進行感知與預測，這些蒐集的資料通常相當巨量，也被稱為「巨量資料」或是「大數據」（Big Data），但是這些資料必須經過整理分析才能變成有用的資訊，因此造就了目前相當夯的「大數據分析」技術。

NETFLIX 借助大數據技術成功推薦影片給消費者

阿里巴巴創辦人馬雲在德國 CeBIT 開幕式上如此宣告：「未來的世界，將不再由石油驅動，而是由數據來驅動！」近年來由於社群網站和行動裝置風行，加上萬物互聯的時代無時無刻產生大量的數據，使用者瘋狂透過手機、平板電腦、電腦等，在社交網站上分享大量資訊，資料成長的速度愈來愈快、種類愈來愈多。面對不斷擴張的驚人資料量，大數據資料的儲存、管理、處理、搜尋、分析等處理資料的能力也將面臨新的挑戰，也為各產業的營運模式帶來新契機。國內外許多擁有大量顧客資料的科技龍頭企業，像是 Facebook、Google、Twitter、Yahoo 等，紛紛從中嗅到了商機。

例如透過即時蒐集用戶的位置和速度，經過大數據分析 Google Map 就能快速又準確地提供用戶即時交通資訊，其它大數據資訊技術，也能幫助衛星導航系統建構完備即時的交通資料庫，甚至大數據相關的資料探堪技術，也在美國大選中為歐巴馬陣營的競選活動提供大量的參考資訊，並成功打贏選戰。

人工智慧

近幾年人工智慧的應用領域愈來愈廣泛，人工智慧的概念最早是由美國科學家 John McCarthy 於 1955 年提出，目標為使電腦具有類似人類學習解決複雜問題與展現思考等能力，舉凡模擬人類的聽、說、讀、寫、看、動作等的電腦技術，都被歸類為人工智慧的可能範圍，例如推理、規畫、問題解決及學習等能力。微軟亞洲研究院曾經指出：「未來的電腦必須能夠看、聽、學，並能使用自然語言與人類進行交流。」

尤其在這個大數據時代，人工智慧（Artificial Intelligence, AI）儼然是未來科技的主流趨勢，主要原因之一就是 GPUs 加速運算日漸普及，使得平行運算的速度更快且成本更低廉，我們也因人工智慧而能享用許多個人化的服務，生活也變得更為便利。

Tips

圖形處理器（Graphics Processing Unit, GPU）

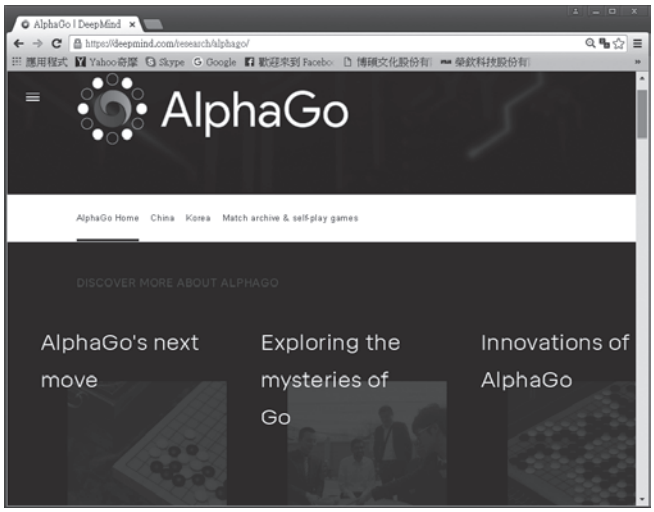
GPU 可說是近年來科學計算領域的最大變革，是指以圖形處理單元（GPU） 搭配 CPU 的微處理器，GPU 則含有數千個小型且更高效率的 CPU，不但能有效處理平行運算（Parallel Computing），還可以大幅增加運算效能，藉以加速科學、分析、遊戲、消費和人工智慧應用。

機器學習（Machine Learning, ML）是人工智慧發展相當重要的一環，機器通過演算法來分析數據、在巨量資料中找到規則，進而自動學習並且做出預測。2010 年後機器學習技術之一的深度學習算法（Deep Learning, DL），更將人工智慧推向類似人類學習模式的優異發展。深度學習算是 AI 的一個分支，也可以看成是具有層次性的機器學習法，源自於類神經網路（Artificial Neural Network）模型，並且結合了神經網路架構與大量的運算資源，目的在於讓機器建立與模擬人腦進行學習的神經網路，以解釋大數據中圖像、聲音和文字等多元資料，例如可以代替人們進行一些日常的選擇和採買，或者在茫茫網路海中，獨立找出分眾消費的行銷數據。

Tips

類神經網路是模仿生物神經網路的數學模式，取材於人類大腦結構，使用大量簡單而相連的人工神經元（Neuron）來模擬生物神經細胞受特定程度刺激來反應刺激架構為基礎的研究，由於類神經網路具有高速運算、記憶、學習與容錯等能力，可以利用一組範例，透過神經網路模型建立出系統模型，便可用於推估、預測、決策、診斷的相關應用。

透過深度學習的應用，機器正在變得越來越聰明，不但會學習也會進行獨立思考，人工智慧的運用也更加廣泛，深度學習包括建立和訓練一個大型的人工神經網路，可協助電腦理解影像、聲音和文字等資料，最為人津津樂道的深度學習應用，當屬 Google Deepmind 開發的 AI 圍棋程式 AlphaGo 接連大敗歐洲和南韓圍棋棋王，AlphaGo 的設計是大量的棋譜資料輸入，還有精巧的深度神經網路設計，透過深度學習掌握更抽象的概念，讓 AlphaGo 學習下圍棋的方法，接著就能判斷棋盤上的各種狀況，後來創下連勝 60 局的佳績，並且不斷反覆跟自己比賽來調整神經網路。



1-4 建置 Python 開發環境

不同於一般的程式語言一開始要準備非常複雜的執行環境，執行 Python 真的非常容易，在程式設計的過程中為了加速開發速度跟減少重複開發的成本，會使用現成的程式套件或模組，但是各個程式套件與 Python 版本相容性是個問題，經常給入門者帶來許多學習的困擾。本書將以 Anaconda 套件組來做安裝，這是一個可用於 Linux、OS X 和 Windows 上的專業級 Python 套件組，甚至於完整包含了日後各位對於機器學習所需的套件，能夠一併幫助各位安裝 Python 及常用程式套件。

1-4-1 下載 Anaconda 套件組

Anaconda 套件組具有以下特點，是初學者安裝 Python 環境的首選：

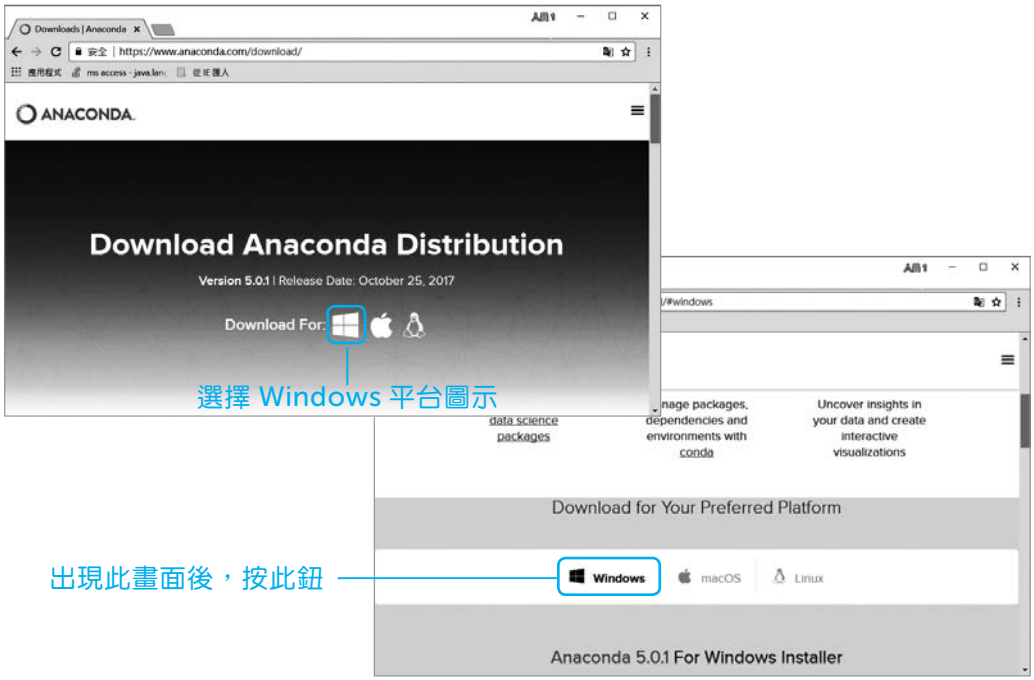
1. 包含了許多常用的數學科學、工程、數據分析的 Python 套件
2. 免費而且開放原始碼
3. 支援 Windows、Linux、Mac 平台
4. 支持 Python 2.x、3.x，而且可以自由切換
5. 內建 Spyder 編譯器
6. 包含 Conda 以及 Jupyter Notebook 環境

其中 Conda 是環境管理的工具，除了可以管理套件新增與移除，也能快速建立獨立的虛擬 Python 環境，也可以在虛擬的 Python 環境裡安裝套件及測試程式，而不用擔心會影響原本的工作環境；Jupyter Notebook 編輯器是 Web 擴充套件，讓使用者可以透過瀏覽器進行程式的開發與維護。以下我們就來下載並安裝 Anaconda。

Step 01

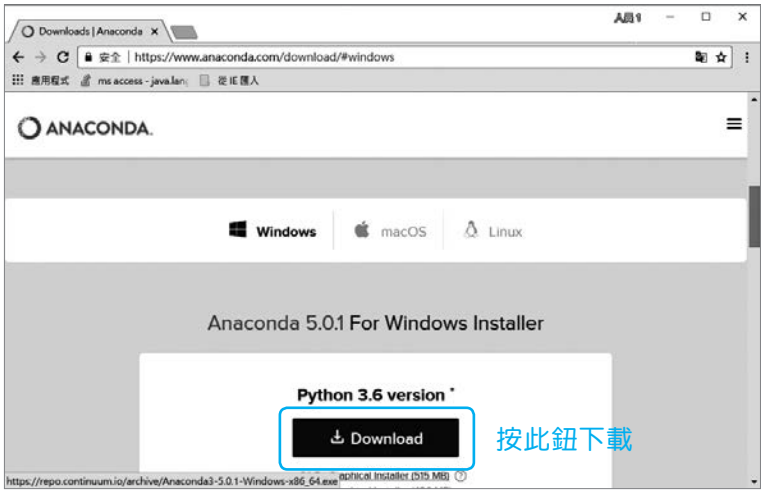
下載網址：https://www.anaconda.com/download

進入網頁之後請依據作業系統選擇適當的下載入口，這裡有 Windows、mac OS 以及 Linux 可供選擇。



Step 02

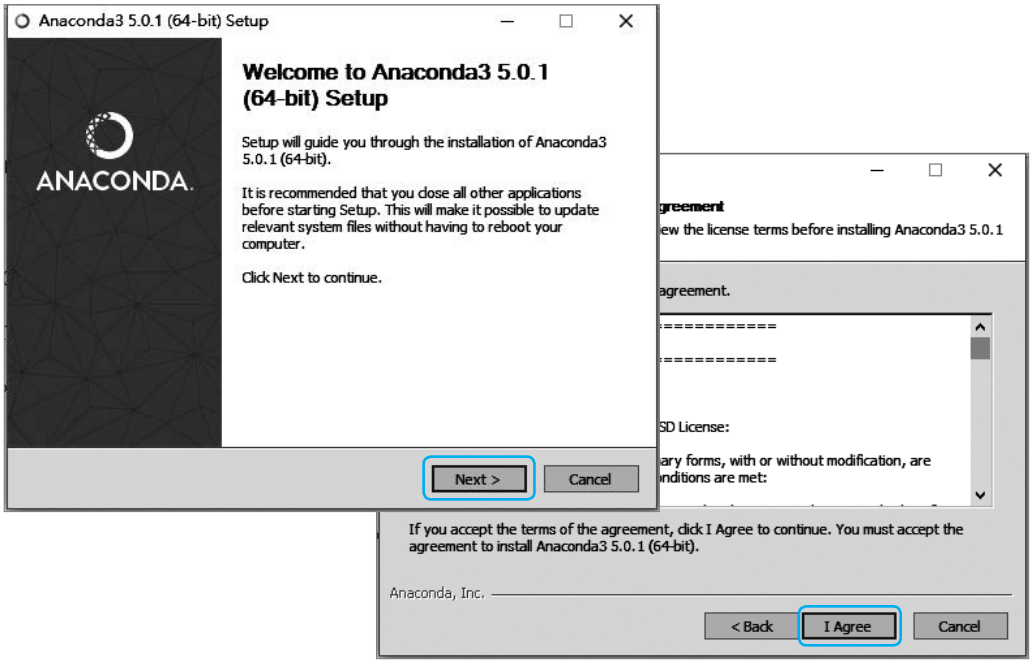
選擇下載的 Python 版本。我們下載的是 Python 3.6、64 位元 Windows 版本，下載完成會看到檔案名稱 Anaconda3-5.0.1-Windows-x86_64.exe 的執行檔。



1-4-2 安裝 Anaconda

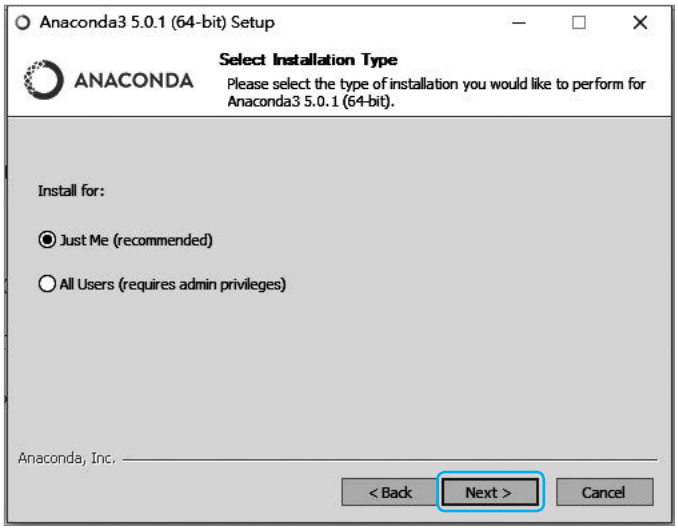
Step 01

快按兩下執行檔就會啟動安裝程式，依序點選「Next」按鈕安裝。當出現如下圖的版權宣告畫面，請閱讀版權說明事項之後點選同意「I Agree」按鈕，進行下一步。



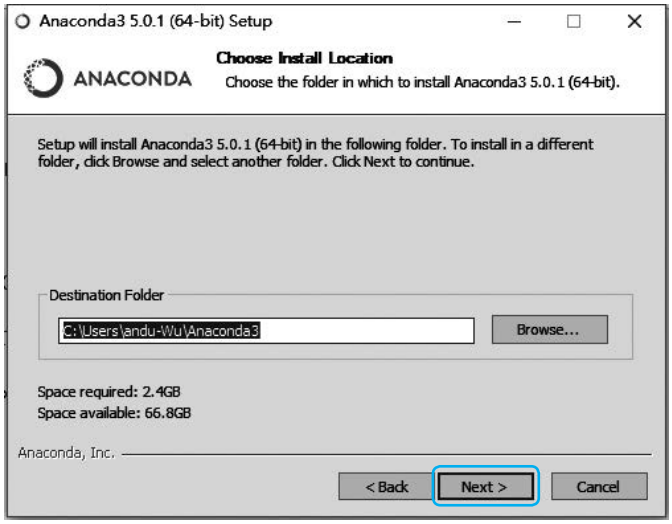
Step 02

出現選擇安裝類型的畫面，建議採用預設值，只安裝 Anaconda 給自己使用，再按下「Next」按鈕。



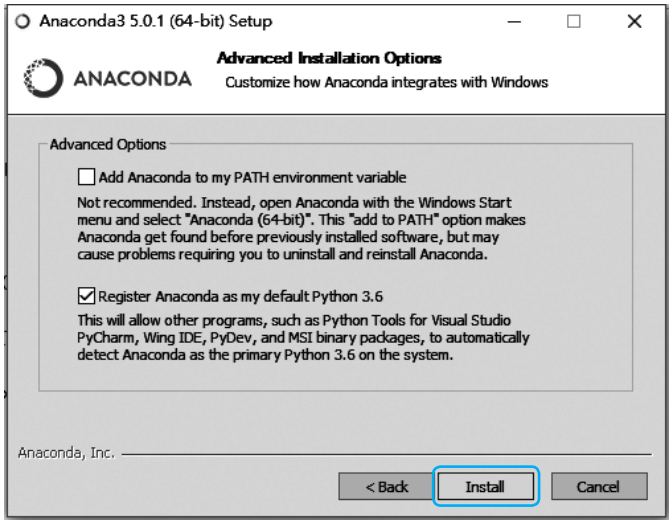
Step 03

設定安裝目錄，不更改目錄的話直接點選「Next」按鈕。



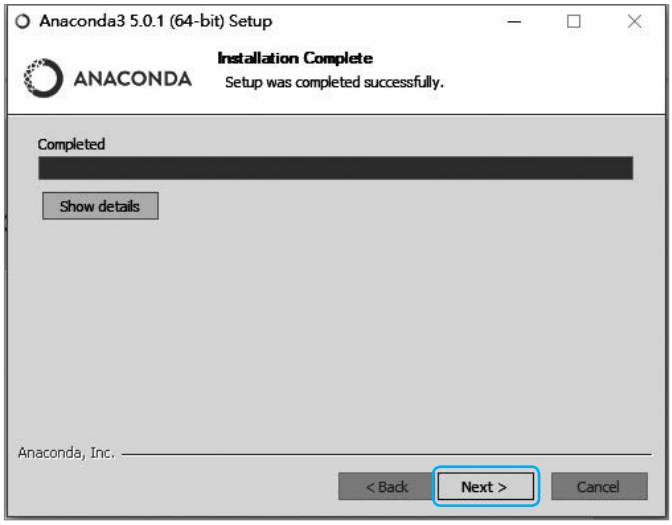
Step 04

接下來的步驟請勾選第二個核取方塊，再點選「Install」按鈕。

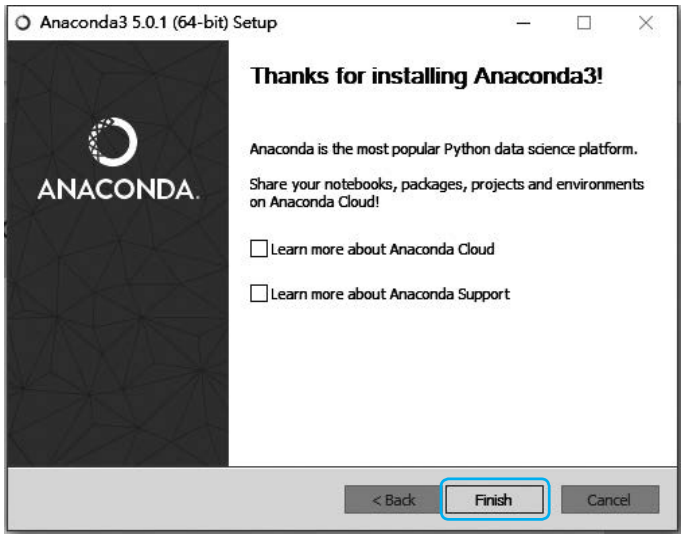


Step 05

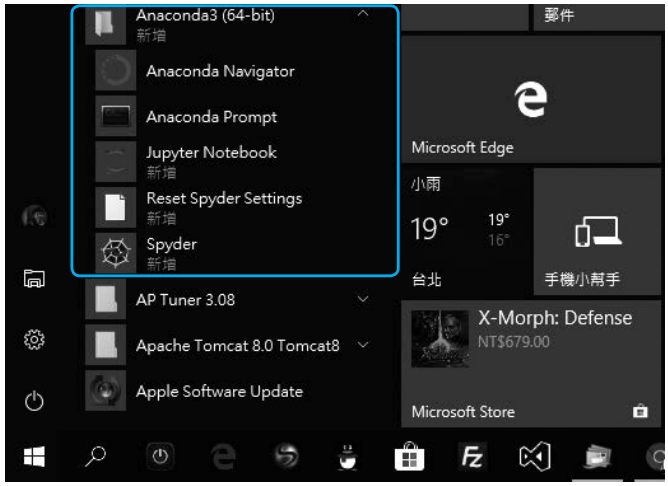
出現如右圖畫面表示安裝完成了，請直接點選「Next」按鈕。



Step 06 按下「Finish」鈕結束安裝。



Step 07 安裝完成之後，在「開始」功能表就有如下圖的 Anaconda3 選單（下圖是以 Windows 10 作業系統為例）。

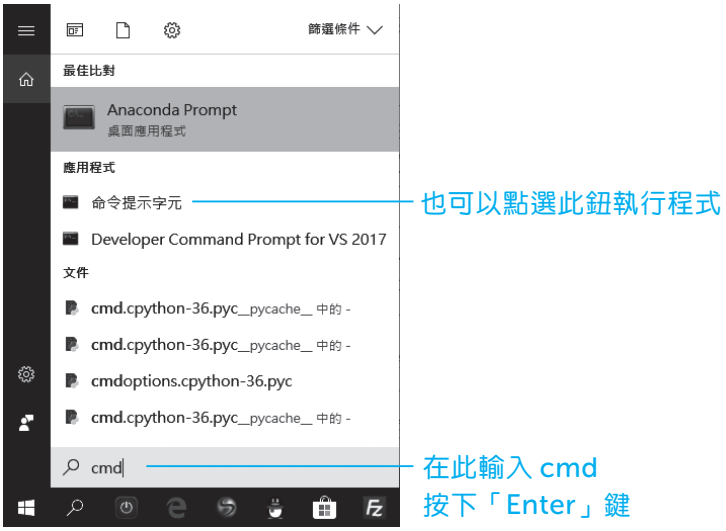


1-5 Python 撰寫工具

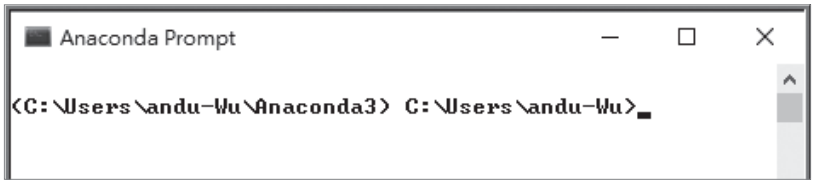
安裝好之後就可以準備來寫 Python 程式了，Python 相當簡單易學，往往簡單的幾行程式碼就可以滿足應用程式的多樣化功能，我們可以直接透過 Windows 命令提示字元視窗或是開啟 Spyder 編輯器來撰寫程式。

1-5-1 透過命令提示字元視窗

各位也可以從「開始」功能表的「搜尋 Windows」方塊輸入「cmd」，按下「Enter」鍵來開啟；或是從程式列表按一下「cmd.exe」鈕也可以執行「命令提示字元」視窗。



「命令提示字元」視窗是透過輸入文字指令的方式來操作電腦。「命令提示字元」視窗，有人稱為 Command line（命令列）、Console（主控台）、Terminal（終端機）或是直接稱為 DOS 視窗。在 Windows 下，提示字元是 > 符號，閃爍的游標就是輸入指令的地方，如下圖：



在提示字元之後輸入「Python」，按 Enter 鍵後就會進入 Python 主控台。當提示字元會變成「>>>」，表示各位已經成功進入 Python 主控台，在這裡只能使用 Python 的指令。如果想要離開 Python，只要輸入「exit()」按下 Enter 鍵就可以了。

```
python
(C:\Users\andu-Wu\Anaconda3) C:\Users\andu-Wu>python
Python 3.6.3 |Anaconda, Inc.| (default, Oct 15 2017, 03:27:45)
[MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more infor
mation.
>>>
```

現在我們先來熟悉一下 Python 主控台的操作。請輸入「5+3」，按下 Enter 鍵。

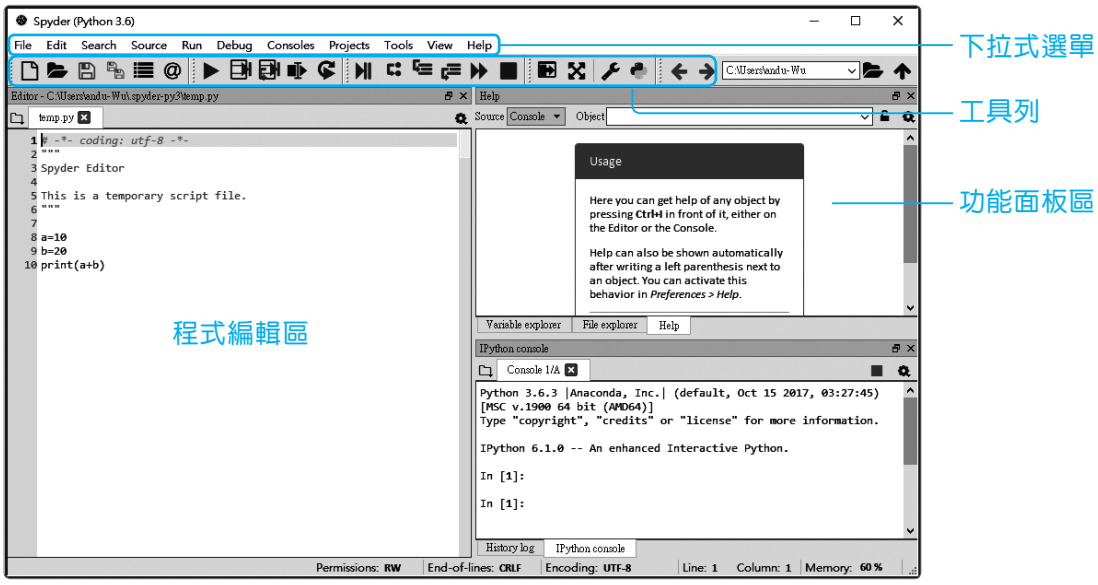
```
python
(C:\Users\andu-Wu\Anaconda3) C:\Users\andu-Wu>python
Python 3.6.3 |Anaconda, Inc.| (default, Oct 15 2017, 03:2
7:45) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more
 information.
>>> 5+3
8
>>>
```

執行之後會直接顯示執行結果，並且再次出現「>>>」提示字元，等著接收下一個指令。Python 就是這麼直覺易用，在還沒有開始學任何 Python 語法之前，如果需要進入互動的線上輔助說明，可以直接鍵入「help()」指令，要離開 help 的公用程式，只要鍵入「quit」即可。

```
Anaconda Prompt - python
>>> help()

Welcome to Python 3.6's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/3.6/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help>
```

1-5-2 Spyder 編輯器

Anaconda 內建的 Spyder 軟體是編輯及執行 Python 程式的整合開發環境（Integrated Development Environment, IDE），具有語法提示、程式除錯與自動縮排功能。請各位在開始功能表單中開啟 Spyder 編輯器，Spyder 編輯器預設的工作區配置上方是下拉式選單及工具列，左方為程式編輯區，右方是功能面板區，如下圖：

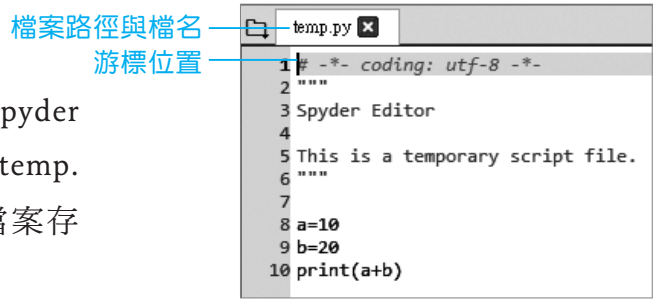


工具列

工具列包含常用的工具按鈕，例如檔案開啟、存檔、執行等功能，各位可以從下拉式選單「View / Toolbars」來開啟與關閉工具列。

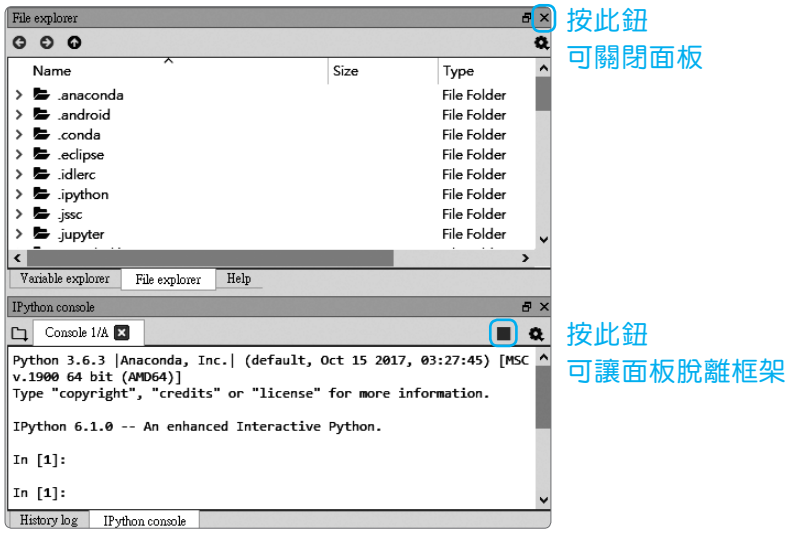
程式編輯區

是用來撰寫程式，啟動 Spyder 之後預設編輯的檔案名稱是「temp.py」，各位可以從標題列看到檔案存放的路徑與檔名。

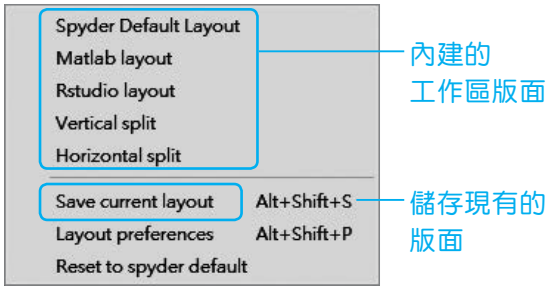


功能面板區

功能面板上方預設為檔案瀏覽面板（File explorer）、變數瀏覽面板（Variable explorer）及幫助面板（Help），下方是 IPython 主控台以及歷史紀錄面板（History log）。Spyder 提供許多功能面板可供使用，各位可以從下拉式選單「View / Panes」來開啟與關閉功能面板。



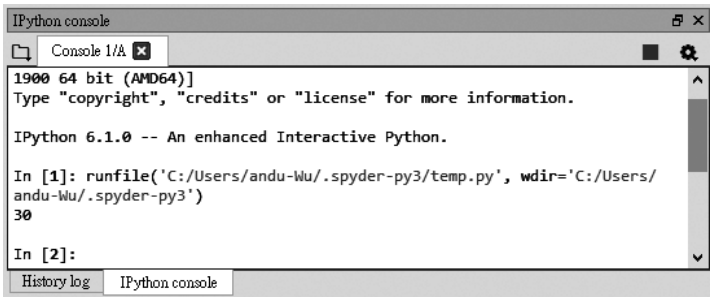
各位也可以從下拉式選單「View /Window layouts」指令選擇工作區或建立自己的工作區。



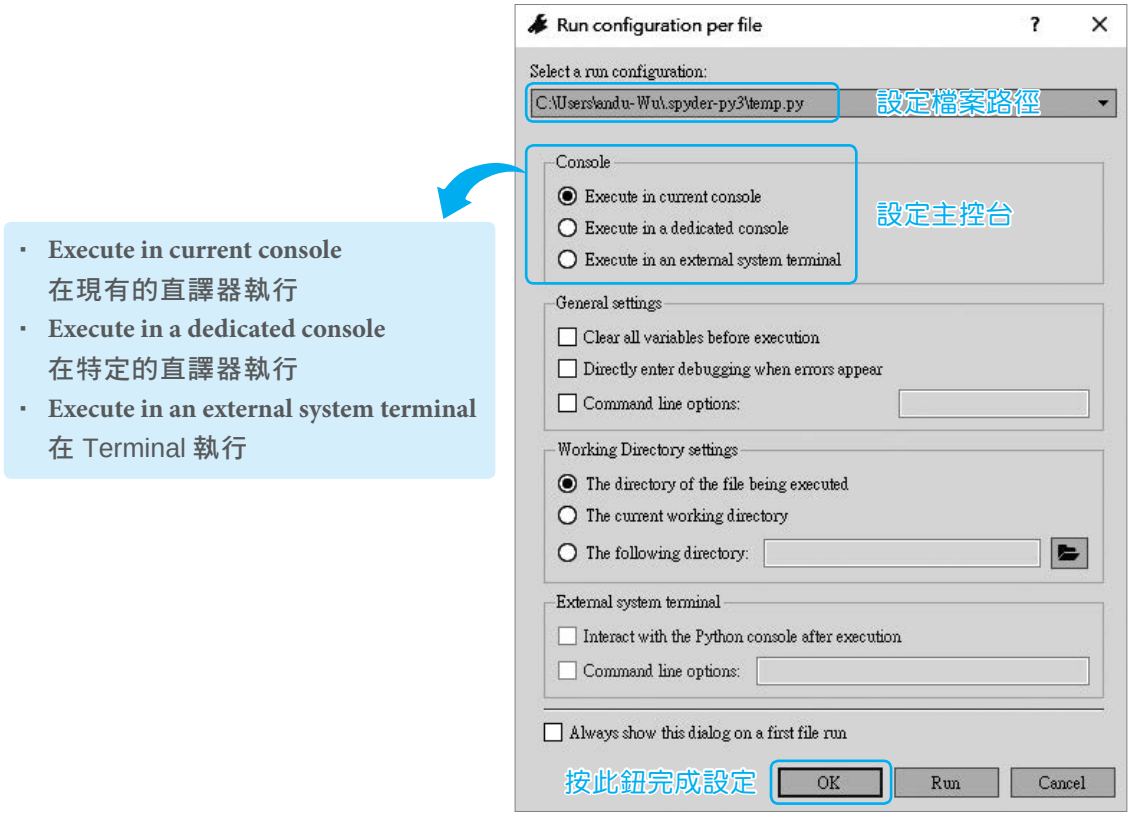
熟悉了 Spyder 的操作介面之後，實際來撰寫 Python 程式並執行看看。請在程式編輯區輸入底下文字。

```
a = 5
b = 6
print(a + b)
```

執行「Run」下拉式功能表的「Run」指令或按下鍵盤的「F5」鍵，也可以點選工具列的 ▶ 鈕來執行程式。執行結果如下圖：

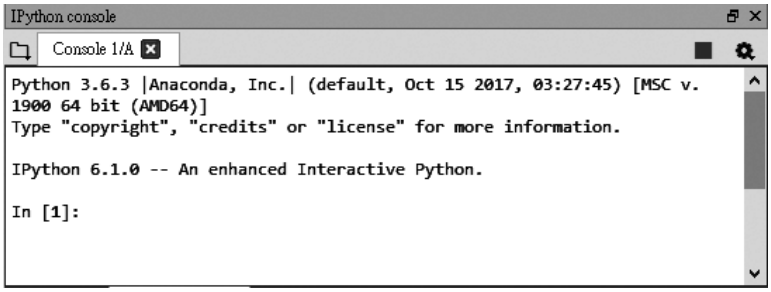


如果執行「Run」下拉式功能表的「Configure」指令就可以開啟「Run Settings」對話視窗，設定執行的主控台有三個選項，如下圖所示：

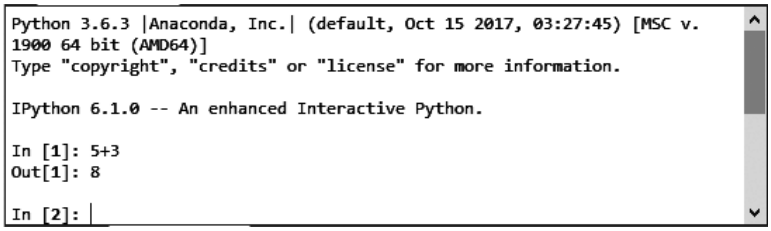


1-5-3 IPython 命令視窗

IPython（Interactive Python）除了可以執行 Python 指令，還提供許多進階的功能。在 IPython 命令視窗，閃爍的游標就是輸入指令的地方，每一列程式碼輸入與輸出都會自動編號。



請輸入「5+3」，按下 Enter 鍵後會立刻顯示執行結果。



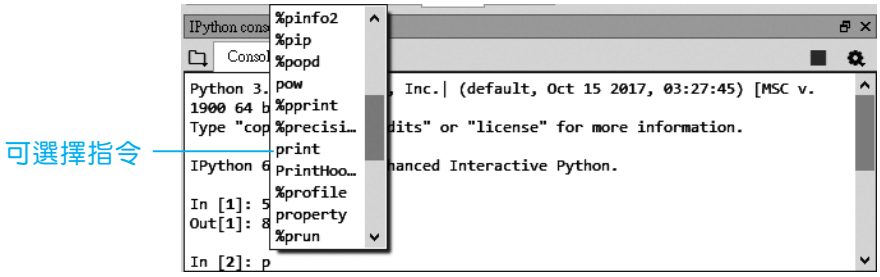
各位可以看到 IPython 的命令視窗多了顏色的輔助，很清楚地區分出運算元與運算子，輸入（In）與輸出（Out）也很容易從顏色來區分。IPython 命令視窗還有一些輔助功能可以幫助我們快速輸入指令，說明如下：

程式碼的自動完成功能

程式碼的自動完成功能是根據輸入的內容自動完成想輸入的程式碼，不但可提高程式輸入的速度，也可減少輸入錯誤的發生。使用方式非常簡單，只要在輸入部分文字之後按下「Tab」鍵，就會自動完成輸入，如果可套用的程式指令超過一個，就會列出所有命令或函數讓使用者參考。例如要輸入底下指令：

```
print("hello")
```

各位可以先輸入「p」後按「Tab」鍵，由於 p 開頭的指令不只一個，所以會列出所有 p 開頭的指令選單，各位可以繼續輸入，或從選單中選取想要的命令或函數，如下圖。



print() 函數是用來輸入文字，請在 print 之後輸入「（"hello"）」文字，按下 Enter 鍵，就會在視窗中輸出「hello」了！

```
In [2]: print("hello")
hello
```

如果指令只有一個，按「Tab」鍵就會自動完成，例如輸入「inp」後再按下「Tab」鍵，就會自動顯示「input」。

調用使用過的程式碼

如果要輸入的程式碼與前面輸入過的程式碼相同，可以利用向上或向下的方向鍵進行選擇，按向上鍵「↑」可顯示前次輸入的程式碼，按向下鍵「↓」可顯示下一列程式碼。找到程式碼之後再按下 Enter 鍵執行，也可以加以修改之後再按下 Enter 鍵執行。

1-6 print 輸出指令

通常在程式設計的過程中，會希望將所運算或程式設計的結果輸出到螢幕，這個時候就可以藉助內建的 print() 函數在螢幕上依自己所指定的輸出方式，將數字或字串或圖形輸出到螢幕上。IPython 提供非常強大的使用說明功能，不管是命令、函數或變數都可以在名稱後面加上「?」，就會顯示該項目的使用說明。例如想要知道 print() 函數的用法，只要輸入「print?」就會顯示使用說明。如下所示：

```
In [3]: print?
Docstring:
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file:  a file-like object (stream); defaults to the current sys.stdout.
sep:   string inserted between values, default a space.
end:   string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
Type:      builtin_function_or_method

In [4]:
```

語法說明如下：

```
print（value, ..., sep= ' ', end= '\n ', file=sys.stuout, flush=False）
```

- **value**：這個參數是用來設定要輸出的值，如果有多個要輸出的值，則請以逗號隔開。例如 print（10, 20, 30），會輸出 10 20 30，如果輸出字串需要在字串前後兩端加上雙引號或單引號，print（"10,20,30"）。
- **sep**：預設值是一個空白字串，這個參數可以用來設定隔開兩個值的字串，它是一個選擇性參數，可以不用設定。例如 print（10, 20, 30, sep="@"），會輸出 10@20@30，如果省略 sep 不寫，就會以預設的空格來分隔輸出的資料。
- **end**：結尾符號，也是一個選擇性參數，可以不用設定。預設值為「\n」，「\n」是換行的意思，如果省略 end 不寫，執行 print() 函數之後就會換行，如果不想換行只要將 end 設定為空字串就可以了。例如：print（10, 20, 30, end=''）
- **file**：是選擇性參數，是用來指定輸出裝置，預設值為輸出到螢幕。

接下來請看看實際範例，各位就會更清楚 print() 函數的用法了。

〈基本輸出〉

檔名 print.py

```
01  print（" 開始輸出 "）
02  print（1, 2, 3）
03  print（4, 5, 6, sep="@"）
04  print（7, 8, 9, sep="|", end=" "）
05  print（10, 11, 12, sep="*"）
06  print（" 結束輸出 "）
```

執行結果 »

```
開始輸出
1 2 3
4@5@6
7|8|9 10*11*12
結束輸出
```

程式第 1 行及第 6 行是輸出字串，分別顯示「開始輸出」及「結束輸出」文字；第 2 行輸出數字；第 3 行輸出以「@」分隔的數字；第 4 行輸出以「|」分隔的數字，而且不換行；第 5 行輸出以「*」分隔的數字。

1-7 Python 程式撰寫風格

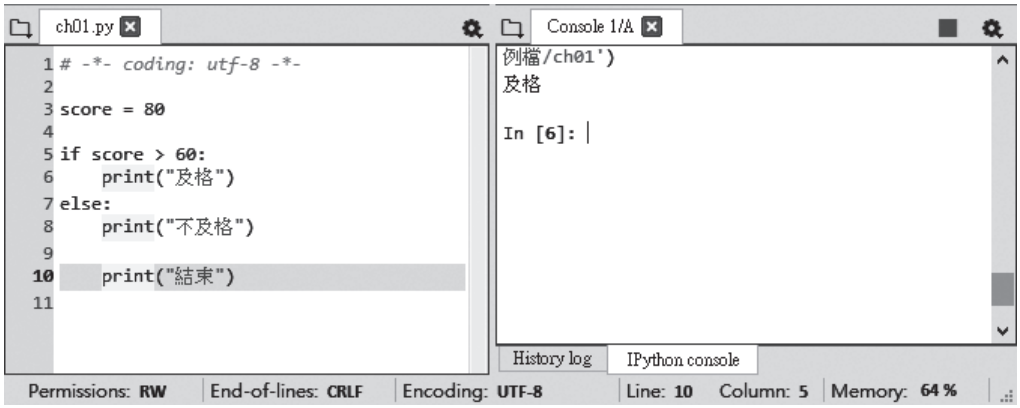
Python 的設計哲學是優雅、明確與簡單，跟其他程式語言比較起來較為寬鬆，所以程式設計人員不需要耗費太多時間在語法的細節上。不過為了讓程式可讀性高，Python 還是有一些撰寫風格慣例，例如運算子前後加上空白或是設定每行最大長度，本節就來看看有哪些需要注意的地方。好的程式碼撰寫慣例，讓大家有通用的原則可以遵循，以下列出幾個原則：

程式碼縮排

Python 程式裡的區塊，主要是透過「縮排」來表示，例如 if/else 冒號 (:) 的下一行程式必須縮排，例如：

```
score = 80

if score > 60:
    print(" 及格 ")            if 區塊
else:
    print(" 不及格 ")
                               else 區塊
    print(" 結束 ")
```

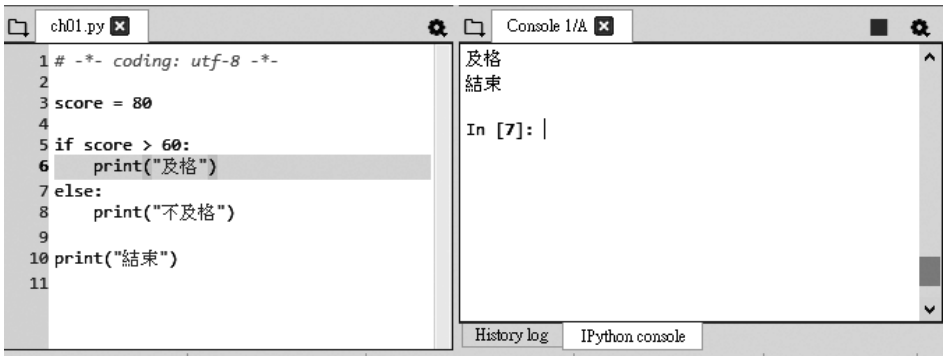
執行之後會得到如下圖的結果：



各位可以開啟範例「ch01/ch01.py」檔案執行看看。上述程式碼裡的 else 區塊雖然第 9 行空了一行，不過第 8 行與第 10 行是同樣的縮排距離，所以還是會被認為是同一區塊來執行。各位可以試著將程式修改如右，再執行看看。

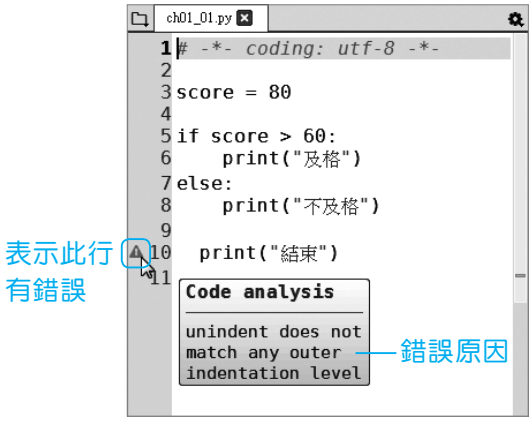
```
1  # -*- coding: utf-8 -*-
2
3  score = 80
4
5  if score > 60:
6      print("及格")
7  else:
8      print("不及格")
9
10 print("結束")
11
```

執行結果如下：



各位應該注意到，第 8 行與第 10 行程式縮排距離不同，所以第 10 行程式不屬於 else 區塊，而是單獨的一行程式了。也就是說，Python 程式語言在某些情況下（例如 if/else 指令）會以縮排來表達程式碼區塊（Code Block），Python 稱「Suite」。如果未遵照使用縮排的規則，解譯過程就會發生錯誤，這種作法其實是希望撰寫程式的人養成縮排的習慣。

從上述說明各位可以知道 Python 程式碼裡的縮排對執行結果有巨大的影響，也因此 Python 對於縮排是非常嚴謹的，同一個區塊的程式碼必須使用相同的空白數進行縮排，否則就會出現錯誤。一旦執行有錯誤在該行程式左方就會出現 ⚠ 圖示，將滑鼠移到 ⚠ 圖示就會提示錯誤原因。



縮排可以使用空白鍵或 Tab 鍵產生空格，PEP8 建議以 4 個空格進行縮排，在 Python 編輯工具中按一下「Tab」鍵預設就是 4 個空格。不過當改用記事本之類的一般文字編輯器來撰寫 Python 程式時，「Tab」鍵的間距並不會是 4 個空白，就有可能會造成程式無法執行，為了避免這樣的情形發生，建議最好以 4 個空格進行縮排，避免空白鍵與 Tab 鍵混用。

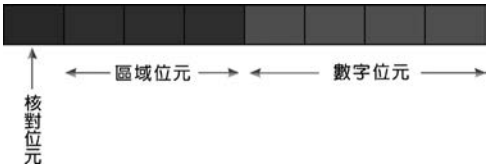
1-7-1 編碼宣告

當我們在 Spyder 開啟新的空白檔案時，並不是完全空白的檔案，預設會帶出編碼與註解文字，如右圖所示：

程式第一行是編碼宣告（Encoding declaration），為什麼需要編碼宣告呢？這是因為電腦電路只有開跟關兩種狀態，也因為電腦只能記錄 0 跟 1，當要存取文字資料時就必須透過編碼系統來做轉換，首先我們先來認識什麼是編碼。

ASCII 編碼

為了整合電腦資訊交換的共同標準，美國標準協會（ASA）提出了一組以 7 個位元（Bit）為基礎的「美國標準資訊交換碼」（American Standard Code for Information Interchange, ASCII）碼，來做為電腦中處理文字的統一編碼方式，是目前最普便的編碼系統。ASCII 採用 8 位元表示不同的字元，不過最左邊為核對位元，故實際上僅用到 7 個位元表示。也就是說 ASCII 碼最多可以表示 $2^7 = 128$ 個不同的字元，可以表示大小英文字母、數字、符號及各種控制字元。例如 ASCII 碼的字母 "A" 編碼為 1000001，字母 "a" 編碼為 1100001：



128	Ç	144	É	160	á	176	░	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	│	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	168	¿	184	╕	201	╔	217	┘	233	Θ	249	∙
137	ë	153	Ö	169	_	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	█	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	_
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	φ	253	²
141	ì	158	_	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

ASCII 碼擴充字元集的十進位代碼與圖形字元

Big5 編碼

通常我們在電腦上所看到的繁體中文字，幾乎都是由「Big-5」編碼格式所編定。Big-5 碼又稱為「大五碼」，是資策會在 1985 年所公布的一種中文字編碼系統。它主要是採用兩個字元組成一個中文字的方式來編碼，也就是說一個 Big-5 碼中文字，佔用 2 個位元組（16Bits）的資料長度。因為 Big-5 碼的組成位元數較多，相對地字集中也包含了較多的字元，在 Big-5 碼的字集中包含了 5401 個常用字、7652 個次常用字，以及 408 個符號字元，可以編出約一萬多個中文碼。由於中文字博大精深，數量遠大於編碼數量，造成許多中文字必須另外造字，不過在中國大陸所使用的簡體中文，卻是 GB 的編碼格式。

Unicode 與 UTF-8 編碼

由於全世界有許多不同的語言，甚至於同一種語言（如中文）都可能會有不同的內碼。在此，我們還要特別介紹一種萬國碼技術委員會（Unicode Technology Consortium：UTC）所制定做為支援各種國際性文字的 16 位元編碼系統—Unicode 碼。為了解決各種編碼不相容問題，國際標準 ISO/IEC 制訂一套全球通用的編碼標準「Unicode」又稱為「萬國碼」，Unicode 是以 16 個位元來表示一個字元，共可表示 65536 個字元。Unicode 編碼字元集包含各個國家標準字集，解決了在不同語言作業系統裡的亂碼問題。

常見的 Unicode 標準裡面，最常使用的是 UTF-8（8-bit Unicode Transformation Format），它是以 8 位元為一個單位，不同的文字採用不固定的字元長度，因為是可變長度字元編碼，佔用的空間比較小，是現在許多 E-Mail、網頁及程式語言使用的編碼方式。

由於 Python 2.x 是以 ASCII 編碼，如果在 Python 程式碼有包含中文字的話，執行就會出錯，所以必須宣告編碼方式。

宣告編碼方式只能放在檔案第 1 行或第 2 行，格式如下：

```
# -*- coding: 編碼名稱 -*-
```

例如指定 UTF-8 編碼，就可以如下表示：

```
# -*- coding: utf-8 -*-
```

其中「-*-」只是醒目並沒有作用，可以省略如下：

```
# coding:utf-8
```

Python 3.x 預設就是使用 UTF-8 編碼，所以編碼宣告也可以省略，但是習慣上還是會加入編碼宣告。

1-7-2 程式註解

有許多讀者總認為寫程式只要跑出結果就好，還要拖泥帶水的寫什麼註解，真是自找麻煩。其實隨著程式碼的規模日益龐大，現在軟體工程的重點就在可讀性與維護性，而適時使用「註解」就是達到這兩項重點的最主要方法。註解不僅可以幫助其他的程式設計師閱讀程式內容，在日後程式維護時，清楚的註解也能省下不少維護成本。

註解是用來說明程式碼或是提供其他資訊的描述文字，Python 直譯器會忽略註解，因此並不會影響執行結果。註解的目的在增加程式的可讀性，尤其是大型程式開發，更是需要簡單而清楚的註解，許多有經驗的程式設計人員習慣在註解裡記錄程式目的、變數及回傳值的描述、演算法的主要步驟、作者以及變更日期等資訊。Python 的註解有兩種，單行註解跟多行註解。

單行註解

單行註解符號是「#」，在「#」以後的文字都會被當成註解，例如：

```
# 這是單行註解
```

多行註解

多行註解是以三個引號包住註解文字，引號可以是成對的三個雙引號，例如：

```
"""
這是多行註解
用來說明程式的描述都可以寫在這裡
"""
```

也可以用三個單引號：

```
'''
這也是多行註解
用來說明程式的描述都可以寫在這裡
'''
```

1-7-3 指令的分行和合併

當程式碼同一行的指令太長時，可以使用倒斜線「\」將一行的指令折成兩行。

```
isLeapYear = (year % 4 == 0 and year % 100 != 0) or \
             (year % 400 == 0)
```

不過也有例外的情形；當指令的句子中有括號 ()、中括號 [] 或大括號 {} 時也可以折成多行。為了閱讀的方便性，配合這些不同的括號來折行指令，是個不錯的方法。

```
isLeapYear = (year % 4 == 0 and
              year % 100 != 0) or (year % 400 == 0)
```

另外，當兩行的指令很短時，可使用「;」（半形分號）把分行的指令合併成一行。不過多行的指令合併成一行時，有可能造成閱讀上的不方便，使用時得多方考量。

```
a = 10; b = 20; c = 30
```

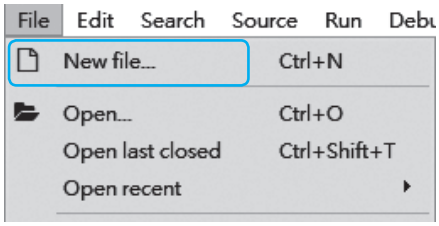
1-8 上機實作演練－ Hello World

這一小節我們將練習 Spyder 編輯器如何開新檔案、撰寫程式並存檔。啟動 Spyder 之後，預設會載入前一次編輯的 .py 檔，這裡我們將開一個新的 Python 檔案，請跟著範例實作練習。

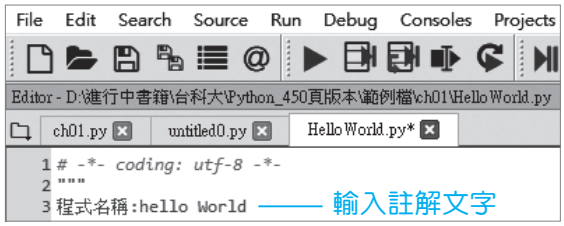
〈我的第一支 Python 程式—Hello World〉

檔名 HelloWorld.py

Step 01 請執行「File/New file」指令或點選工具列 鈕。



Step 02 修改註解文字。Spyder 編輯器會自動帶出註解格式，直接將文字修改為適當註解就可以了。

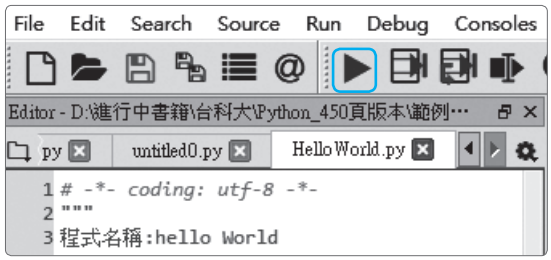


Step 03 輸入底下的程式碼：

```
print ('Hello World')

str="Hello World"
print (str)
```

Step 04

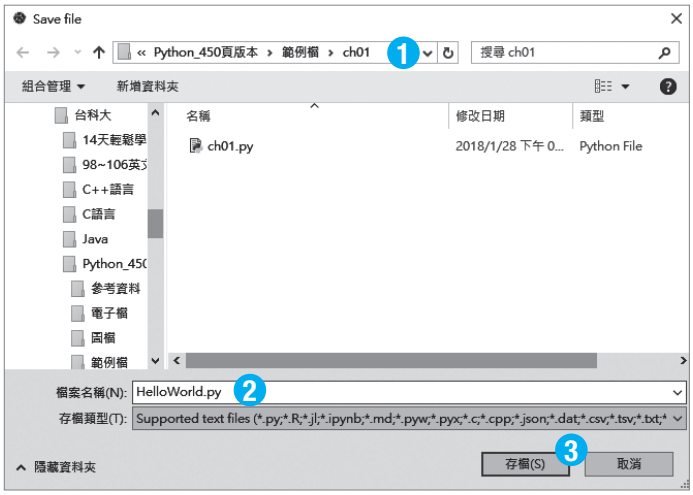
輸入完成就可以執行程式了，請按下 F5 鍵或點選工具列的 ▶ 鈕。



Step 05

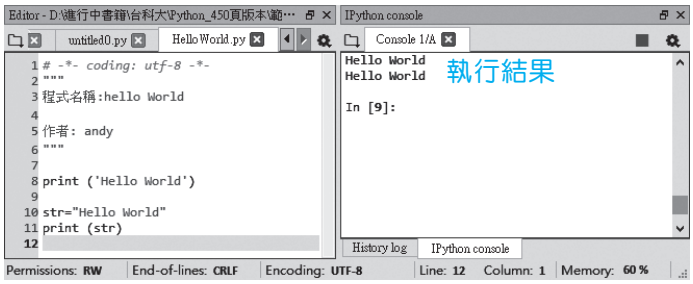
由於尚未存檔，執行之前會先跳出存檔視窗，要求存檔。

❶ 請選擇適當的資料夾後，❷ 輸入檔案名稱「helloworld.py」，❸ 再按存檔鈕。



Step 06

存檔後就會在 IPython 主控台顯示執行結果了。



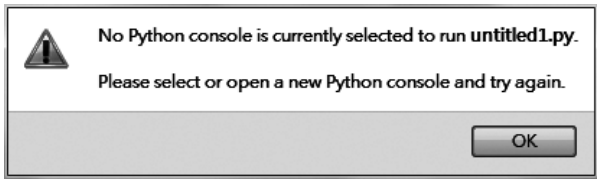
Tips

Python 程式的副檔名是「.py」，檔案名稱可以只輸入「HelloWorld」，存檔後會自動加上副檔名「.py」。

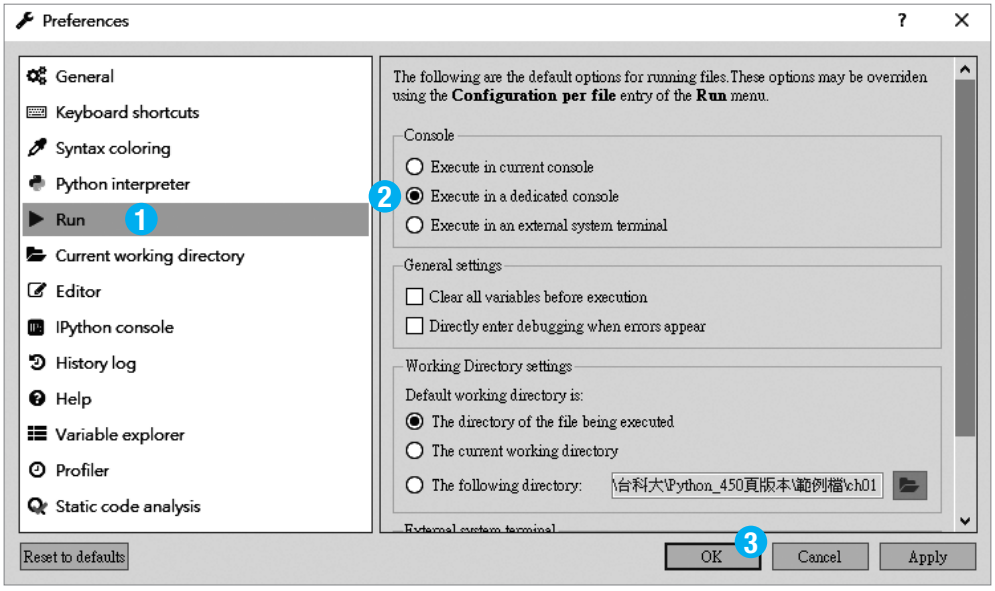
補充說明一點，Python 的字串可以用雙引號，也可以用單引號，並不會影響執行結果。

學習小教室

如果執行 Python 程式時沒有選定用哪一種直譯器，執行時就會出現如下錯誤。



各位可以執行「Tools/ Preferences」指令，將「Execute in a new dedicated Python console」設為預設值，如此一來就會一律在新的 Python 直譯器執行程式了。



重點回顧

1. Python 是一種物件導向、直譯的程式語言，語法直覺易學，具有跨平台的特性，加上強大的套件模組，讓 Python 的用途更為廣泛。
2. Python 特色：程式碼簡潔易讀、跨平台、物件導向、自由 / 開放原始碼。
3. 機器語言是一種最低階的程式語言，它是以 0 與 1 二進位組合的方式，將指令和機器碼輸入電腦。
4. 組合語言是以有意義的英文或數字來代替機器語言的程式語言，和機器語言相比，更方便人類記憶與使用。
5. 高階語言所設計的程式，電腦無法直接執行，必須經過編譯器（Compiler）或直譯器（Interpreter）轉換成機器語言才能執行。
6. 演算法的的五項特性：
   (1) 輸入資料（Input）：0 個或多個輸入
   (2) 輸出結果（Output）：至少 1 個以上的輸出結果
   (3) 明確性：描述的處理程序必須是明確，不能模稜兩可
   (4) 有限性：必須在有限的步驟內完成工作，不可以有無窮迴圈
   (5) 正確性：可以正確地解決問題
7. 演算法可以用圖形或文字表達，最簡單的方式就是透過流程圖來描述。
8. 流程圖是使用圖形符號來表示解決問題的步驟，繪製流程圖有助於程式的修改與維護。
9. 程式開發最常用的流程圖是「系統流程圖」及「程式流程圖」。
10. Web 框架就是為建立 Web 應用程式制定了一套規範，簡化了技術上的細節，只要運用 Web Framework 模組，就能輕易地建構出動態網站。
11. Python 有許多容易擴增的資料分析與機器學習模組庫（library），像是 NumPy、Matplotlib、Pandas、Scikit-learn、SciPy、PySpark 等，這也讓 Python 成為資料解析與機器學習主要語言之一。

12. 物聯網（Internet of Things, IoT）是讓生活中的物品能透過互聯互通的傳輸技術進行感知與控制。

13. Conda 是環境管理的工具，除了可以管理安裝新的套件，也能快速建立獨立的虛擬 Python 環境。

14. 我們可以透過 Windows 命令提示字元視窗或是開啟 Spyder 編輯器來撰寫程式。

15. 當提示字元變成「>>>」，表示已經成功進入 Python 主控台，如果想要離開 Python，只要輸入「exit()」按下 Enter 鍵即可。

16. 程式碼的自動完成能根據輸入的內容自動完成想輸入的程式碼。

17. IPython 提供非常強大的使用說明功能，不管是命令、函數或變數都可以在名稱後面加上「?」，就會顯示該項目的使用說明。

18. Anaconda 內建的 Spyder 編輯器是方便編輯及執行 Python 程式的整合開發環境（Integrated Development Environment, IDE）。

19. Python 程式裡的區塊，主要是透過「縮排」來表示，例如 if/else 冒號（:）的下一行程式必須縮排。

20. 縮排可以使用空白鍵或 Tab 鍵產生空格，建議以 4 個空格進行縮排。

21. print 指令是讓使用者輸出資料，而 input 指令則是讓使用者由「標準輸入裝置」（通常指鍵盤）輸入資料。

22. UTF-8（8-bit Unicode Transformation Format），它是以 8 位元為一個單位，不同的文字採用不固定的字元長度，因為是可變長度字元編碼，佔用的空間較小。

23. Python 的註解有兩種，單行註解跟多行註解。
    (1) 單行註解符號是「#」，在「#」以後的文字都會被當成註解。
    (2) 多行註解是以三個引號包住註解文字，引號可以是成對的三個雙引號。

24. 當程式碼同一行的指令太長時，可以使用倒斜線「\」將一行的指令折成兩行。

學習評量

選擇題

(　　) 1. 關於 Python 的應用領域不包括下列哪一種？
(A) 自動控制　(B) 大數據
(C) 遊戲設計　(D) 以上皆是 Python 應用領域。

(　　) 2. 關於 Python 語言的特性，下列何者有誤？
(A) 擴充能力強　(B) 可以大量使用指標
(C) 跨平台　(D) 簡潔易讀。

(　　) 3. 關於各代程式語言的描述，下列何者有誤？
(A) 高階語言更符合人類語言的形式
(B) 組合語言與硬體有著密切關係
(C) 機器語言是一種最低階的程式語言
(D) 組合語言是一種高階語言。

(　　) 4. 下列哪一種程式語言是屬於直譯式語言？
(A) Java　(B) Python　(C) C++　(D) C。

(　　) 5. 演算法的特性不包括？
(A) 至少會有一個輸出結果　(B) 0 個或多個輸入資料
(C) 允許無窮迴路　(D) 步驟清楚且可行。

填充題

1. 程式語言以發展過程來區分大致可分為_____語言與_____語言兩大類。

2. _____是一種最低階的程式語言，它是以 0 與 1 二進位組合的方式，將指令和機器碼輸入電腦。

3. 高階語言所設計的程式，必須經過_____或_____轉換成機器語言才能執行。

4. _____是使用圖形符號來表示解決問題的步驟。

5. _____是環境管理的工具，除了可以管理安裝新的套件，也能快速建立獨立的虛擬 Python 環境。

6. Anaconda 內建的_____編輯器是方便編輯及執行 Python 程式的整合開發環境。

7. Python 程式裡的區塊，主要是透過_____來表示。

簡答題

1. Python 的註解有哪兩種，請簡單說明之。

2. 請比較說明編譯與直譯的差別。

3. 請試著描述計算 1+2+3+4+5 的演算法。

4. 請試著描述計算 1+2+3+4+5 的流程圖。

5. 演算法的時候必須滿足哪些特性？

6. 請簡述 Python 語言的重要特性。

7. 請簡單說明註解的功用。

Chapter 2

資料型態、變數與常數

程式語言中最基本的資料處理對象就是常數與變數，主要的用途是儲存資料，以供程式中各種計算與處理之用。本章將會跟各位討論 Python 的資料型態、變數與常數等基本資料處理功能，針對初學者在編寫小型應用程式時所需要的基礎語法進行介紹。

學習大綱

- 變數宣告與指派
- 變數命名規則
- Python 的數值型態
- 常數
- 數值處理函數
- 數值格式化輸出
- 輸入函數：input()
- 資料型態轉換

對於任何一種程式語言，最基礎的部分就是把資料儲存在記憶體中加以處理的能力，無論你準備進行哪種運算，巧婦難為無米之炊，總不能跟空氣做運算吧！這些加以運算的對象，在 Python 中就是以常數與變數為主。其實兩者都是程式設計師用來存取記憶體中資料內容的一個識別代碼，兩者最大的差異在於變數的內容會隨著程式執行而改變，但常數的內容則是永遠固定不變。因為程式在執行過程時，經常會需要記憶或取用一些資料，例如想要寫一個期中考成績計算的程式，必須要先輸入學生成績，再經過計算之後，輸出總分、平均與排名。本章我們就來看看如何儲存與取用這些資料！

2-1 變數命名與指派

程式簡單來說就是告訴電腦要拿哪些資料（Data）依照指令敘述一步步來執行，這些資料可能是文字，也可能是數字。我們提到的變數（variable），是程式語言中最基本的角色，也就是在程式設計中由編譯器所配置的一塊具有名稱的記憶體，用來儲存可變動的資料內容。電腦會將它儲存在「記憶體」中，需要時再取出使用，為了方便識別，必須給它一個名字，就稱為「變數」例如：

變數就是程式中用來存放資料的地方

```
>>>a = 3
>>>b = 5
>>>c = a + b
```

上面敘述中的 a、b、c 就是變數，數字 3 就是 a 的變數值，由於記憶體的容量是有限的，為了避免浪費記憶體空間，每個變數會依照需求給定不同的記憶體大小，因此有了「資料型態」（Data type）來加以規範。

2-1-1 變數宣告與指派

Python 是物件導向的語言，所有的資料都看成是物件，在變數的處理上也是用物件參照（Object reference）的方法，變數的型態是在給定初始值時決定，所以不需要事先宣告資料型態。對於變數的值是使用等號（=）來指派，初學者很容易將指派運算子（=）的作用和數學上的「等於」的功能互相混淆，在程式語言中的「=」號主要是當作指定的功能。

變數宣告的語法如下：

```
變數名稱 = 變數值
```

例如：

```
number = 10
```

上式表示指派數值 10 給變數 number。

簡單來說，在 Python 語言中使用變數時，不需要事先指定資料型態，這點和在 C 中使用變數，一定都要事先宣告後才能使用不同，系統會依據所設定的變數值來自動決定該變數的資料型態。例如上述的變數 number 的資料型態為整數，如果變數內容為字串時，該變數的資料型態就是字串。

2-1-2 變數命名規則

程式碼的可讀性對於想要成為一個優秀程式設計師而言，可是非常重要的好習慣，雖然變數名稱只要符合 Python 的規定都可以自行定義，但是當變數越多時，你只是簡單取個 abc 等字母名稱的變數，就會讓人暈頭轉向，大幅降低可讀性，考慮到程式的可讀性，最好儘量以符合變數所賦予的功能與意義來命名，例如身高取名為「height」，體重取名為「weight」等。尤其是當程式規模越大時，有意義的變數名稱就會顯得更為重要。例如在宣告變數時，為了程式可讀性，一般習慣是以小寫字母開頭表示，如 score、salary 等。

在 Python 中的變數名稱命名也需要符合一定的規則，如果使用不適當的名稱，可能會造成程式執行時發生錯誤，Python 是屬於區分大小寫的語言，也就是說 number 與 Number 是兩個不相同的變數，變數名稱的長度不限，變數名稱有以下幾點限制：

- 變數名稱第一個字元必須是英文字母或是底線或是中文，不能是數字。
- 其餘字元可以搭配其他的大小寫英文字母、數字、_ 或中文，不能使用空白字元。
- 不能使用 Python 內建的保留字。

雖然 Python 3.x 版本的變數名稱支援中文，不過建議各位儘量不要使用中文來命名變數，一方面程式碼輸入時要切換輸入法較為麻煩，另一方面在程式碼的閱讀上也會顯得不太順暢。所謂保留字通常具有特殊意義與功能，所以它會被預先保留，而無法作為變數名稱或任何其他識別字名稱。

以下是有效變數名稱的範例：

```
_pagecount
fileName01
length
number_item
```

以下是無效變數名稱的範例：

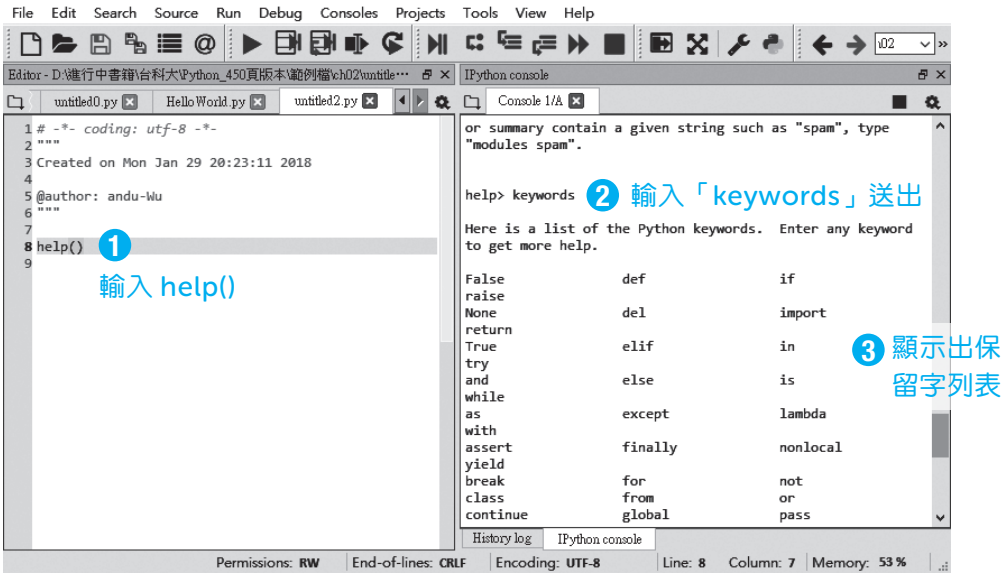
```
2_result
for
$result
user name
```

學習小教室

利用 help() 函數查詢 Python 保留字

help() 函數是 Python 的內建函數，對於特定物件的方法、屬性不清楚時，都可以利用 help() 函數來查詢。

像是前面提到的 Python 保留字，就可以利用 help() 函數來查看，只要執行「help()」就會進入 help 互動模式，在此模式下輸入要查詢的指令就會顯示相關說明，操作步驟如下。



各位可以在 help> 模式繼續輸入想要查詢的指令，想要離開 help 互動模式時，只要輸入 q 或 quit 就可以了。各位也可以在輸入 help() 指令時帶入引數，例如 help("keywords")，Python 就會直接顯示說明，而不會進入 help 互動模式。

雖然 Python 採用動態型別，但是對於資料的處理卻很嚴謹，屬於「強型別」的資料型態。舉例來說：

```
>>> a = 5
>>> b = "45"
>>> print( a+b )      # 顯示 TypeError
```

變數 a 是數值型態，變數 b 是字串型態，有些程式語言會在不知不覺中轉換型態，自動將數值 a 轉換為字串型態，因此 a+b 會得到 545，Python 禁止不同資料型態進行操作，所以執行上面敘述就會顯示型態錯誤的失敗訊息。

學習小教室

強型別 V.S 弱型別

程式語言的資料型態有「強型別」（strongly typed）和「弱型別」（weakly typed、loosely typed）的分別，權衡條件之一是對於資料型態轉換的安全性。強型別對於資料型態轉換有較嚴格的檢查，不同型態進行運算時必須明確轉換型態，程式不會自動轉換，像是 Python、Ruby 就是偏向強型別；而弱型別的程式語言大部分採取隱式轉換（Implicit conversion），如果不注意就會發生非預期的型態轉換而導致錯誤的執行結果，JavaScript 就是偏向弱型別的程式語言。

2-1-3 靜態型別與動態型別

Python 執行時才決定資料型態的方式屬於「動態型別」，什麼是動態型別呢？程式語言的資料型態依照型態檢查方式可區分為「靜態型別」（Statically-typed）與「動態型別」（Dynamically-typed）：

靜態型別

編譯時會先檢查型態，因此變數使用前必須先做明確的型態宣告，執行時不能任意更動變數的型態，像是 Java、C 語言就是屬於此類程式語言。例如下式 C 語言宣告變數 number 是 int 整數型態，預設值是 10，當 number 值指派 "apple" 時，就會出錯，因為 "apple" 是字串，在編譯階段就會因型態不符而編譯失敗。

```
int number = 10
number = "apple"      # Error: 型態不符
```

動態型別

編譯時不會事先進行型態檢查，而是在執行時才會依照變數值來決定資料型態，因此變數使用前不需要宣告型態，同一個變數還可以指定不同型態的值，Python 就是屬於動態型別。例如下式宣告變數 number 預設值是整數 10，當 number 值改為字串 "apple" 時，會自動轉換型別。

```
number = 10
number = "apple"
print( number )      # 輸出字串 apple
```

Python 也有垃圾回收機制（garbage collection），當物件不再使用時，直譯器會自動回收，釋放記憶體空間。例如上例中，當整數物件 number 重新指派給另一個字串物件時，原本的整數物件會被直譯器刪除掉。如果物件確定不需要使用了，各位也可以使用「del」來刪除，語法如下：

```
del 物件名稱
```

例如：

```
>>>number = "apple"
>>>print( number )  # 輸出 apple
>>>del number       # 刪除字串物件 number
>>>print(number)    # Error:number 未定義
```

執行結果如下。由於變數 number 已經刪除，如果再使用 number 這個變數，就會出現變數未定義的錯誤訊息：

出現變數未定義的錯誤訊息

2-2 Python 的數值型態

Python 的數值型態有整數（int）、浮點數（float）與布林值（bool），以下一一說明這些數值型態的用法。

2-2-1 整數

整數資料型態是用來儲存不含小數點的資料，跟數學上的意義相同，如 -1、-2、-100、0、1、2、100 等。Python 2.x 整數有 int（整數）跟 long（長整數）兩種類型，但 Python 3.x 之後就只有 int 整數類型，Python 的數值處理能力相當強大，基本上沒有位數的限制，只要硬體 CPU 支援的情況下再大的整數都可以處理。

有時為了某些可讀性的需要，我們可以使用不同的數字系統來表示整數值，例如儲存資料的記憶體位址就經常是以十六進位的方法來表示。整數是指正或負整數，除了以十進位（decimal）表示，也能以二進位（binary）、十六進位（hexadecimal）、八進位（octal）表示，只要分別在數字之前加上 0b、0x、0o 指定進位系統就可以了，右表是整數的一些例子。

整數	說明
100	十進位
0b1100100	二進位
0x64	十六進位
0o144	八進位
–745	負數

2-2-2 浮點數

浮點數資料型態指的就是帶有小數點的數字，也就是數學上所指的實數（real number）。除了一般小數點表示，也能使用科學記號格式以指數表示，例如 6e-2，其中 6 稱為假數，-2 稱為指數。右表都是合法的浮點數表示方式。

浮點數	說明
25.3	帶有小數點的正數
–25.3	帶有小數點的負數
1.	1.0
5e6	5000000.0

電腦中的數字是採用 IEEE 754 標準規範來儲存，IEEE 754 標準的浮點數並不能精確的表示小數，舉例來說：

```
num = 0.1 + 0.2
```

得到的 num 並不等於 0.3，而是 0.30000000000000004。這並不是 Python 獨有的問題，所有的程式語言對浮點數運算都有精確度的問題，因此做浮點數運算時必須特別小心，以下提供兩個小數運算的方法供讀者參考：

▶ 使用 decimal 模組做小數運算

decimal 模組是 Python 標準模組庫模組，使用它的時候需要先利用 import 指令將模組載入才能使用，載入之後利用 decimal.Decimal 類別來儲存精確的數字，如果引數為非整數時必須要以字串形式傳入，例如：

```
import decimal
num = decimal.Decimal("0.1") + decimal.Decimal("0.2")
```

得到的結果就會是 0.3。

▶ 利用 round() 函數強制小數點的指定位數

round(x[, n]) 是內建函數，會回傳參數 x 最接近的數值，n 是指定回傳的小數點位數，例如：

```
num =  0.1 + 0.2
print( round(num, 1) )
```

上面敘述是將變數 num 取到小數點 1 位，因此會得到 0.3。

2-2-3 布林值

布林資料型態是一種表示邏輯的資料型態，是 int 的子類別，只有真假值 True 與 False。布林資料型態通常使用於流程控制做邏輯判斷。你也可以採用數值「1」或「0」來代表 True 或 False。

False	說明
0	數字 0
""	空字串
None	None
[]	空的 List
()	空的 Tuple
{}	空的 Dict

Python 必須相同資料型態才能進行運算，例如字串與整數不能相加，必須將字串轉換為整數，如果運算子都是數值型態的話 Python 會自動轉換，不需要強制轉換型態，例如：

```
num = 5 + 0.3    # 結果 num=5.3 ( 浮點數 )
```

Python 會自動將整數轉換為浮點數再進行運算。另外，布林值也可以當成數值來運算，True 代表 1，False 代表 0，例如：

```
num = 5 + True   # 結果 num=6 ( 整數 )
```

如果是想把字串轉換為布林，可以透過 bool 函數來轉換，以下範例利用 print() 函數顯示布林值。

〈轉換布林型態〉

檔名 bool.py

```
01  print( bool(0) )
02  print( bool("") )
03  print( bool(" ") )
04  print( bool(1) )
05  print( bool("ABC") )
```

執行結果 >>

```
False
False
True
True
True
```

</> 程式解說 >>

- 第 02 行：傳入是一個空字串，所以會回傳 False。
- 第 03 行：傳入是含有一個空格的字串，所以會回傳 True。

Tips

使用布林值 False 與 True 時要特別注意第一個字母必須大寫。

2-3 常數

常數是指程式在執行的整個過程中，不能被改變的數值。例如整數常數 45、-36、10005、0 等，或者浮點數常數：0.56、-0.003、1.234E2 等等，常數擁有固定的資料型態和數值。變數（variable）與常數（constant）兩者最大的差異在於變數的內容會隨著程式執行而改變，但常數則固定不變。

Python 的常數是指字面常數（literal），也就是指該常數字面上的意義，例如 12 就代表整數 12 的意義。所謂字面常數就是直接寫進 Python 程式的數值。字面常數如果以資料型態來區分，會有不同的分類，例如：1234、65、963、0，都是一種整數字面常數（integer literal），而帶小數點的數字則為浮點數型態（floating-point type）的字面常數，例如 3.14、0.8467、744.084。至於以單引號（'）或雙引號（"）圍起來的字元都是字串字面常數，例如 "Hello World"、"0932545212" 都是一種字串字面常數（string literal）。

2-4 格式化輸出入功能

學習 Python 通常是從主控台輸出程式執行結果，或是從主控台取得使用者的輸入資料，前面我們經常使用 print() 函數輸出執行結果，本章就來看一下如何利用 print() 函數進行格式化輸出及如何輸入資料。

2-4-1 格式化輸出

print() 函數支援格式化輸出，有兩種格式化方法可以使用，一種是以「%」格式化輸出，另一種是透過 format 函數格式化輸出，以下先介紹「%」格式化輸出。

「%」格式化輸出

格式化文字可以用「%s」代表字串、「%d」代表整數、「%f」代表浮點數，語法如下：

```
print( 格式化文字 % ( 引數 1, 引數 2…引數 n))
```

例如：

```
score = 66
print(" 大明的數學成績：%d" % score)
```

執行結果

```
大明的數學成績：66
```

其中 %d 就是格式化格式，代表輸出整數格式，各種輸出格式請參考右表。

格式化符號	說明
%s	字串
%d	整數
%f	浮點數
%e	浮點數，指數 e 型式
%o	八進位整數
%x	十六進位整數

格式化輸出可以用來控制列印位置，讓輸出的資料能整齊排列，例如：

```
print("%5s 的數學成績：%5.2f" % ("Jenny",95))
print("%5s 的數學成績：%5.2f" % ("andy",80.2))
```

執行結果 >>

```
Jenny的數學成績：95.00
 andy的數學成績：80.20
```

上述範例中格式化文字有兩個引數，所以引數必須用括號包住，其中 %5s 表示固定列印 5 個字元，當少於 5 個字元，會在字串左方補上空白字元；%5.2f 表示固定列印 5 位數的浮點數，小數點佔 2 位數。

以下程式範例將數字 100，分別用 print 函數輸出浮點數、八進位、十六進位以及二進位，請各位跟著範例練習看看。

〈整數輸出不同進制〉

檔名 print_%.py

```
01  num = 100
02  print (" 數字 %s 的浮點數：%5.1f" % (num,num))
03  print (" 數字 %s 的八進位：%o" % (num,num))
04  print (" 數字 %s 的十六進位：%x" % (num,num))
05  print (" 數字 %s 的二進位：%s" % (num,bin(num)))
```

執行結果 >>

```
數字 100 的浮點數：100.0
數字 100 的八進位：144
數字 100 的十六進位：64
數字 100 的二進位：0b1100100
```

程式解說 >>

- 第 03~05 行：輸出八進位、十六進位及二進位。
- 第 05 行：由於二進位並沒有格式化符號，因此可以透過內建函數 bin() 將十進位數字轉換成二進位字元再輸出。

format() 函數輸出

格式化輸出也可以搭配 format() 函數來使用，相對於 % 格式化的方式，format() 函數更加靈活，用法如下：

```
print("{} 是個用功的學生 . ".format(" 王小明 "))
```

一般簡單的 format 用法會用大括號 {} 表示，{} 內則用 format() 裡的引數替換，format() 函數相當具有彈性，它有兩大優點：

- 不需要理會引數資料型態，一律用 {} 表示
- 可使用多個引數，同一個引數可以多次輸出，位置可以不同

舉例來說：

```
print("{0} 今年 {1} 歲 . ".format(" 王小明 ", 18))
```

其中 {0} 表示使用第一個引數、{1} 表示使用第二個引數，以此類推，如果 {} 內省略數字編號，就會依照順序填入。

各位也可以使用引數名稱來取代對應引數，例如：

```
print("{name} 今年 {age} 歲 . ".format(name=" 王小明 ", age=18))
```

直接在數字編號後面加上冒號「:」可以指定參數格式，例如：

```
print('{0:.2}'.format(5.5625))
```

表示第一個引數取小數點後 2 位。

另外也可以搭配「^」、「<」、「>」符號加上寬度來讓字串居中、靠左對齊或靠右對齊，例如：

```
print("{0:10} 成績：{1:_^10}".format("Jennifer", 95))
print("{0:10} 成績：{1:>10}".format("Brian", 87))
print("{0:10} 成績：{1:*<10}".format("Jolin", 100))
```

執行結果 >>

```
Jennifer   成績：____95____
Brian      成績：        87
Jolin      成績：100*******
```

其中 {1:_^10} 表示寬度 10，以底線「_」填充並置中；{1:>10} 表示寬度 10 靠右對齊，未指定填充字元就會以空格填充；{1:*<10} 表示寬度 10，以星號「*」填充並靠左對齊。

2-4-2 輸入函數：input()

input 是常用的輸入指令，可以讓使用者由「標準輸入裝置」（通常指鍵盤）輸出資料，把使用者所輸入的數值、字元或字串傳送給指定的變數。例如想設計一個計算每位學生的國文及數學的總分，就可以透過 input 指令來讓使用者輸入國文及數學成績，再去計算總分。語法如下：

```
變數 = input( 提示字串 )
```

當輸入資料並按下 Enter 鍵後，就會將輸入的資料指定給變數。上述語法中的「提示字串」是一段告知使用者輸入的提示訊息，例如希望使用者先輸入身高，再輸出身高的值，程式碼如下：

```
height =input(" 請輸入你的身高：")
print (height)
```

又例如：

```
score = input(" 請輸入數學成績：")
print("%s 的數學成績：%5.2f" % ("Jenny",float(score)))
```

執行結果 >>

```
請輸入數學成績：86
Jenny的數學成績：86.00
```

當程式執行時，遇到 input 指令會先等待使用者輸入資料，當使用者輸入完成按下 Enter 鍵，就會將使用者輸入的資料存入變數 score 中。

使用者輸入的資料是字串格式，我們可以透過內建的 int()、float()、bool() 等函數將輸入的字串轉換為整數、浮點數、布林值型態，範例裡指定的格式是浮點數 (%5.2f)，所以利用 float() 函數將輸入的 score 值轉換為浮點數。下一節將介紹更完整的資料型態轉換。

以下透過範例再次練習輸出與輸入的用法。

Tips

如果各位使用 Spyder 這類的編輯器，程式執行到輸入提示訊息時，別忘了將輸入游標切換到 Python 主控台再進行輸入。

〈format 格式化輸出〉

檔名 format.py

```
01  name = input(" 請輸入姓名：")
02  che_grade = input(" 請輸入國文成績：")
03  math_grade = input(" 請輸入數學成績：")
04
05  print("{0:8}{1:>5}{2:>5}".format(" 姓名 "," 國文 "," 數學 "))
06  print("{0:<10}{1:>6}{2:>6}".format(name,che_grade,math_
    grade))
```

執行結果 >>

```
請輸入姓名：陳金峰

請輸入國文成績：98

請輸入數學成績：96
姓名          國文    數學
陳金峰          98      96
```

</> 程式解說 >>

- 第 01~03 行：依序要求使用者輸入姓名、國文成績及數學成績。
- 第 05~06 行：依序輸出姓名、國文及數學的表頭，再於下一行輸出這兩科的成績。

2-5 資料型態轉換

對於針對運算式執行上的要求，還可以「暫時性」轉換資料的型態。當不同資料型態要進行運算時，就必須強制轉換資料型態，Python 強制轉換資料型態的內建函數有下列三種：

int()：強制轉換為整數資料型態

例如：

```
x = "5"
num = 5 + int(x)
print(num)          # 結果：10
```

變數 x 的值為 5 是字串型態，所以先用 int(x) 轉換為整數型態。

float()：強制轉換為浮點數資料型態

例如：

```
x = "5.3"
num = 5 + float(x)
print(num)          # 結果：10.3
```

變數 x 的值為 5.3 是字串型態，所以先用 float(x) 轉換為浮點數型態。

str()：強制轉換為字串資料型態

例如：

```
x = "5.3"
num = 5 + float(x)
print(" 輸出的數值是 " + str(num))    # 結果：輸出的數值是 10.3
```

上述程式碼中 print() 函數裡面「現在輸出的數值是 」這一串字是字串型態，「+」號可以將兩個字串相加，變數 num 是浮點數型態，所以必須先轉換為字串。

〈資料型態轉換〉

檔名 conversion.py

```
01  str = "{1} + {0} = {2}"
02  a = 150
03  b = "60"
04  print(str.format(a, b, a + int(b)))
```

執行結果 »

```
{1} + {0} = {2}
60 + 150 = 210
```

</> 程式解說 »

- 第 01 行：由於 b 是字串格式，先指定了顯示的格式，請注意！大括號 {} 的數字編號順序是 {1}、{0}、{2}，所以變數 a 與 b 顯示的順序會與 format 裡的引數順序不同。
- 第 04 行：先用 int() 轉換為整數型態，再進行計算。

2-6 上機實作演練－零用金記帳小管家

請設計一 Python 程式，可以輸入一週 7 天所花費的零用金，並將每一天所花費的零用金輸出。

2-6-1 程式範例描述

這支程式可以允許輸入使用者名稱，接著可以連續輸入一週內每一天的花費總和，並將每一天所花費的零用金輸出。執行結果如下圖所示：

```
請輸入姓名：陳德來

請輸入第一天的零用金總花費：500

請輸入第二天的零用金總花費：450

請輸入第三天的零用金總花費：380

請輸入第四天的零用金總花費：400

請輸入第五天的零用金總花費：520

請輸入第六天的零用金總花費：480

請輸入第七天的零用金總花費：500
name     day1 day2 day3 day4 day5 day6 day7
陳德來      500  450  380  400  520  480  500
```

2-6-2 程式碼說明

以下是本範例完整的程式碼。

〈零用金記帳小幫手〉

檔名 money.py

```
01 # -*- coding: utf-8 -*-
02 """
03 可以輸入一週 7 天所花費的零用金，
04 並將每一天所花費的零用金輸出。
05 """
06
07 name = input(" 請輸入姓名：")
08 day1 = input(" 請輸入第一天的零用金總花費：")
09 day2 = input(" 請輸入第二天的零用金總花費：")
10 day3 = input(" 請輸入第三天的零用金總花費：")
11 day4 = input(" 請輸入第四天的零用金總花費：")
12 day5 = input(" 請輸入第五天的零用金總花費：")
13 day6 = input(" 請輸入第六天的零用金總花費：")
14 day7 = input(" 請輸入第七天的零用金總花費：")
15
16 print("{0:<8}{1:^5}{2:^5}{3:^5}{4:^5}{5:^5}{6:^5}{7:^5}". \
17       format("name","day1","day2","day3", \
18              "day4","day5","day6", \
19              "day7"))
20 print("{0:<8}{1:^5}{2:^5}{3:^5}{4:^5}{5:^5}{6:^5}{7:^5}". \
21       format(name,day1,day2,day3,day4,day5,day6,day7))
```

重點回顧

1. 由於記憶體的容量是有限的，為了避免浪費記憶體空間，每個變數會依照需求給定不同的記憶體大小，因此有了「資料型態」來加以規範。
2. 在 Python 語言中使用變數時，不需要事先指定資料型態，系統會依據所設定的變數值來自動決定該變數的資料型態。
3. Python 是區分大小寫的語言，如 number 與 Number 是兩個不同的變數。
4. 保留字通常具有特殊意義與功能，會被預先保留，無法作為變數名稱或任何其他識別字名稱。
5. help() 函數是 Python 的內建函數，對於特定物件的方法、屬性不清楚時，可利用 help() 函數查詢。
6. Python 採用動態型別，但對資料的處理很嚴謹，屬於強型別的資料型態。
7. 程式語言的資料型態依照型態檢查方式區分為靜態型別與動態型別。
8. Python 有垃圾回收機制，當物件不再使用時，直譯器會自動回收，釋放記憶體空間。如果物件確定不需要使用，也可以使用「del」刪除。
9. Python 的數值型態有整數、浮點數與布林值。
10. 整數是指正或負整數，不帶小數點，除了以十進位表示，也能以二進位、十六進位、八進位表示，須分別在數字前加上 0b、0x、0o 指定進位系統。
11. decimal 模組是 Python 標準模組庫模組，使用時需要先利用 import 指令將模組載入才能使用，載入後利用 decimal.Decimal 類別儲存精確的數字。
12. round(x[, n]) 是內建函數，會回傳參數 x 最接近的數值，n 是指定回傳的小數點位數。
13. 布林值（bool）是 int 的子類別，只有真假值 True 與 False（要特別注意第一個字母必須大寫）。布林資料型態通常使用於流程控制做邏輯判斷。
14. Python 必須相同資料型態才能進行運算，例如字串與整數不能相加。
15. 字面常數就是直接寫進 Python 程式的數值。
16. print() 函數支援格式化輸出，有兩種格式化方法可以使用，一種是以「%」格式化輸出，另一種是透過 format 函數格式化輸出。
17. input() 函數可以指定提示文字，使用者輸入的文字則儲存在指定的變數中。

Chapter 2

學習評量

選擇題

(　　) 1. 有關 Python 變數命名與指派，下列何者有誤？
(A) 使用變數時要事先宣告其資料型態
(B) 每個變數有其資料型態
(C) 變數的值是使用等號 (=) 來指派
(D) Python 是區分大小寫的語言。

(　　) 2. 有關 Python 變數命名，下列何者有誤？
(A) 不能使用空白字元
(B) 變數名稱支援中文
(C) 不能使用 Python 內建的保留字
(D) 變數名稱第一個字元可以是數字。

(　　) 3. 有關 Python 資料型態的說明，下列何者正確？
(A) 靜態型別、強型別
(B) 動態型別、強型別
(C) 靜態型別、弱型別
(D) 動態型別、弱型別。

填充題

1. _____通常具有特殊意義與功能，所以它會被預先保留，而無法作為變數名稱或任何其他識別字名稱。

2. _____函數是 Python 的內建函數，對於特定物件的方法、屬性不清楚時，都可以利用這個函數來查詢。

3. 程式語言的資料型態依照型態檢查方式可區分為「_____」與「_____」。

4. 布林值（bool）是 int 的子類別，只有真假值_____與_____。

5. print() 函數有兩種格式化方法可以使用，一種是以_____格式化輸出，另一種是透過_____函數格式化輸出。

簡答題

1. 請說明下列的變數名稱為有效或無效？並說明無效原因。

   fileName01

   $result

   2_result

   number_item

2. 請說明三種較為常見的 Python 數值型態，並舉例說明。

3. 請設計一個程式，輸入姓名與數學成績，並輸出。

   例如姓名輸入：Jenny，數學成績輸入：80，輸出結果。

4. format() 函數相當具有彈性，它有哪兩大優點？

5. Python 強制轉換資料型態的內建函數有哪三種？

Chapter

3 運算式與運算子

電腦主要的功能就是強大的運算能力，當外界所得到的資料輸入電腦，並透過程式來進行運算，最後輸出所要的結果。本章將會跟各位討論 Python 中的運算子與運算式各種類型與功能，與瞭解如何運用 Python 設計運算式來計算、判斷這些想要處理的資料。

學習大綱

- 算術運算子
- 指派運算子
- 比較運算子
- 邏輯運算子
- 位元運算子
- 運算子優先順序

不論如何複雜的程式，目的上都是用來幫助我們從事各種運算的工作，而這些過程都必須依賴一道道的運算式來加以完成。運算式就像平常所用的數學公式一樣，例如：

```
A=(B+C)*(A+10)/3;
```

上面整個數學式子就是運算式，=、+、* 及 / 符號稱為運算子，而變數 A、B、C 及常數 10、3 都屬於運算元。運算式是由運算子（operator）與運算元（operand）所組成。什麼是運算元、運算子？從下面一個簡單的運算式可以了解：

```
a = b + 5
```

上面指令包含 3 個運算元 a、b 與 5，一個指派運算子「=」以及一個加法運算子「+」。Python 語言除了算術運算子外，還有應用在條件判斷式的比較及邏輯運算子，另外還有將運算結果指定給某一變數的指派運算子。

運算子如果只有一個運算元，稱為「一元運算子」，例如：表達負值的「-23」。當有兩個運算元，則是「二元運算子」，像是算術運算子加、減、乘、除就是一種「二元運算子」，例如：3 + 7。這些多樣、功能完整的運算子，有其執行的優先順序，本節將介紹這些運算子的用法。

3-1 算術運算子

算術運算子（arithmetic operator）是程式語言中使用率最高的運算子，常用於一些四則運算，像是加法運算子、減法運算子、乘法運算子、除法運算子、餘數運算子、取得整除數運算子、指數運算子等運算子。+-*/ 運算子與我們常用的數學運算方法相同，而正負號運算子主要表示運算元的正 / 負值，通常設定常數為正數時可以省略 + 號，例如「a=5」與「a=+5」意義是相同的。特別要提醒各位，負數也可以使用減法 (-) 運算子的符號來表示。當負數進行減法運算時，為了避免與減法運算子的分辨混淆，最好應以括號 ( ) 隔開負數。

下表中列出了 Python 的各種算術運算子功能說明、範例及運算後的結果值。

算術運算子	範例	說明
+	a+b	加法
-	a-b	減法
*	a*b	乘法
**	a**b	乘冪（次方）
/	a/b	除法
//	a//b	整數除法
%	a%b	取餘數

「/」與「//」都是除法運算子，「/」會有浮點數；「//」會將除法結果的小數部分去掉，只取整數，「%」是取得除法後的餘數。例如：

```
a = 5
b = 2
print(a / b)          # 浮點數 2.5
print(a // b)         # 整數 2
print(a % b)          # 餘數 1
```

如果運算結果並不指派給其它變數，則運算結果的資料型態將以運算元中資料型態最大的變數為主。另外當運算元兩者皆為整數，而運算結果產生小數，Pyhton 自動以小數方式輸出結果，各位無需擔心資料型態的轉換問題。

但是如果運算結果會指派給某個變數，則該變數長度必須足夠，以避免資料過長的部分遭到捨去，例如運算的結果為浮點數，而被指定至整數變數，則運算結果的小數部分將被捨去。

算術運算子中的除法「/」運算子是一般的除法，經運算後所求的商數是浮點數，如果要將將該商數以整數表示可以利用 int() 函數。

```
int(15/7)             # 輸出 2
```

另外還有「**」是乘冪運算，例如要計算 2 的 4 次方：

```
print(2 ** 4)        #16
```

請注意，算術運算子 (+-*/) 的優先順序為「先乘除後加減」，舉個例子來加以說明：

```
5+2*3
```

上式的運算結果會是 11。而括號的優先順序又高於乘除，如果上式改為 (5+2)*3 的話，運算結果就會是 21，如果同樣的優先順序，通常會以由左至右的順序來運算。

接著就以範例來看看簡單的四則運算應用。以下範例是讓使用者輸入攝氏（Celsius）溫度，並轉換為華氏（Fahrenheit）溫度，提示：F=(9/5)*C+32。

〈攝氏溫度轉換華氏溫度〉

檔名 temperature.py

```
01  # -*- coding: utf-8 -*-
02  """
03  輸入攝氏溫度轉換華氏溫度
04  提示：F=(9/5)*C+32
05  """
06  C= float( input(" 請輸入攝氏溫度："))
07  F = (9 / 5) * C + 32
08  print(" 攝氏溫度 {0} 轉換為華氏溫度為 {1}".format(C,F))
```

執行結果 »

```
請輸入攝氏溫度：38
攝氏溫度 38.0 轉換為華氏溫度為 100.4
```

</> 程式解說 >>

- 第 06 行：讓使用者輸入攝氏溫度，並利用 float() 函數將所輸入的內容轉換為浮點數的資料型態。
- 第 07 行：將所輸入的攝氏溫度轉換為華氏溫度。
- 第 08 行：依所指定的格式化字串，將兩者溫度間的轉換對應進行格式化輸出。

附帶說明的一點，「+」號可以用來連接兩個字串。

```
a ="abc" + "def"        #a="abcdef"
```

3-2 指派運算子

指派運算子 (=) 由至少兩個運算元組成，功能是將等號右方的值指派給等號左方的變數，由至少兩個運算元組成。許多程式語言的入門學習者，最不能理解的就是等號「=」在程式語言中的意義，很容易和數學上的等於功能互相混淆，在程式語言中的「=」號主要是當作指定的功能。但是以數學的理解角度，「=」這個符號以往都認為那是一種等於的觀念。例如下面這樣指令：

```
sum=0;
sum=sum+1;
```

上述指令中的 sum=0 還容易理解其所代表的意義，至於 sum=sum+1 這道指令，許多初學者往往無法想通這道指令所代表的意義！

其實 Python 程式語言中的「=」主要是當做「指派」（assignment）的功能，各位可以想像成當宣告變數時會先在記憶體上安排位址，等到利用指定運算子 (=) 設定數值給該變數時，才會將數值指定給該位址來儲存。也就是說 sum=sum+1 可以看成是將 sum 位址中的原資料值加 1 後的結果，再重新指定給 sum 的位址。

在指派運算子 (=) 右側可以是常數、變數或運算式，最終都會將值指定給左側的變數；而運算子左側也僅能是變數，不能是數值、函數或運算式等。例如運算式 X-Y=Z 就是不合法的。

Python 指派運算子有兩種指派方式：

單一指派

將指令等號 (=) 右邊的值指定給左邊的變數，例如：

```
a = 10
```

指派運算子除了一次指定一個數值給變數外，還能夠同時指定同一個數值給多個變數。如果要讓多個變數同時具有相同的變數值，各位也可以一併指定變數值，例如變數 x、y、z 的值皆為 100，指令如下：

```
x = y = z = 100
```

當各位想要在同一列中指定多個變數，則可以利用「,」（分隔變數）。例如變數 x 的值為 10，變數 y 的值為 20，變數 z 的值為 30，指令如下：

```
x, y, z =10, 20, 30
```

Python 也允許使用者以「;」（分隔運算式）來連續宣告不同的程式指令。例如以下兩行程式碼：

```
sum= 10
index = 12
```

我們可以利用「;」（分隔運算式）將上述兩行指令寫在同一行。請看以下的示範：

```
sum= 10 ; index = 12          # 以分號串接兩行指令
```

複合指派

複合指派運算子，是由指派運算子 (=) 與其它運算子結合而成。先決條件是「=」號右方的來源運算元必須有一個是和左方接收指定數值的運算元相同，如果一個運算式含有多個複合指定運算子，運算過程必須是由右方開始，逐步進行到左方。例如：

```
a += 1                    # 相當於 a = a + 1
a -= 1                    # 相當於 a = a - 1
```

例如以「A += B;」指令來說，它就是指令「A=A+B;」的精簡寫法，也就是先執行 A+B 的計算，接著將計算結果指定給變數 A。下表中除了「=」運算子以外，其他指派運算子都是複合指派運算子。

指派運算子	範例	說明
=	a = b	將 b 指派給 a
+=	a += b	相加同時指派，相當於 a=a+b
-=	a -= b	相減同時指派，相當於 a=a-b
*=	a *= b	相乘同時指派，相當於 a=a*b
**=	a **= b	乘冪同時指派，相當於 a=a**b
/=	a /= b	相除同時指派，相當於 a=a/b
//=	a //= b	整數相除同時指派，相當於 a=a//b
%=	a %= b	取餘數同時指派，相當於 a=a%b

Tips

在 Python 中單一個等號「=」是指派，兩個等號「==」用來做關係比較，不可混用。

請注意！使用指派運算子時，變數的值必須先設定，否則會出現錯誤！例如 num=num*10，因為還沒為 num 變數初值，如果直接使用指派運算子，就會出現錯誤，因為 num 變數這個變數沒有被定義過初始值。接下來的例子是指派運算子綜合應用。

〈指派運算子綜合應用〉

檔名 assign_operator.py

```
01 # -*- coding: utf-8 -*-
02 """
03 指派運算子練習
04 """
05
06 a = 1
07 b = 2
08 c = 3
09
10 x = a + b * c
11 print("{}".format(x))
12 a += c
13 print("a={0}".format(a,b))  #a=1+3=4
14 a -= b
15 print("a={0}".format(a,b))  #a=4-2=2
16 a *= b
17 print("a={0}".format(a,b))  #a=2*2=4
18 a **= b
19 print("a={0}".format(a,b))  #a=4**2=16
20 a /= b
21 print("a={0}".format(a,b))  #a=16/2=8
22 a //= b
23 print("a={0}".format(a,b))  #a=8//2=4
24 a %= c
25 print("a={0}".format(a,b))  #a=4%3=1
26 s = "Python" + " 好好玩 "
27 print(s)
```

執行結果 »

```
7
a=4
a=2
a=4
a=16
a=8.0
a=4.0
a=1.0
Python好好玩
```

</> 程式解說 »

- 第 12~13 行：將 a 與 c 相加後的值同時指派變數 a，再將 a 的結果值輸出。
- 第 14~15 行：將 a 與 b 相減後的值同時指派變數 a，再將 a 的結果值輸出。
- 第 16~17 行：將 a 與 b 相乘後的值同時指派變數 a，再將 a 的結果值輸出。
- 第 18~19 行：將 a 與 b 進行乘冪後的值同時指派變數 a，再將 a 的結果值輸出。
- 第 20~21 行：將 a 與 b 相除後的值同時指派變數 a，再將 a 的結果值輸出。
- 第 22~23 行：將 a 與 b 整數相除同時指派給變數 a，再將 a 的結果值輸出。
- 第 24~25 行：將 a 與 b 取餘數同時指派給變數 a，再將 a 的結果值輸出。

3-3 比較運算子

比較運算子也被稱為關係運算子，是用來判斷條件式左右兩邊的運算元是否相等、大於或小於，當使用關係運算子時，所運算的結果就是成立或者不成立兩種。下表為常用的比較運算子。

比較運算子	範例	說明
>	a > b	左邊值大於右邊值則成立
<	a < b	左邊值小於右邊值則成立
==	a == b	兩者相等則成立
!=	a != b	兩者不相等則成立
>=	a >= b	左邊值大於或等於右邊值則成立
<=	a <= b	左邊值小於或等於右邊值則成立

當運算式成立，就會得到「真」（True），不成立會得到「假」（False）。

比較運算子也可以串連使用，例如 a < b <= c 相當於 a < b，而且 b <= c。請注意！等號關係是 "==" 運算子，至於 "=" 則是指派運算子，這種差距很容易造成程式碼撰寫時的疏忽，日後程式除錯時，這可是非常熱門的小 bug 喔！

〈比較運算子綜合應用〉

檔名 compare_operator.py

```
01  # -*- coding: utf-8 -*-
02  """
03  比較運算子練習
04  """
05  a = 56
06  b = 24
07  c = 38
08  num1 = (a == b)     # 判斷 a 是否等於 b
09  num2 = (b != c)     # 判斷 b 是否不等於 c
10  num3 = (a >= c)     # 判斷 a 是否大於等於 c
11  print('a 是否等於 b: ',num1)        # 將 num1 顯示出來
12  print('b 是否不等於 c: ',num2)      # 將 num2 顯示出來
13  print('a 是否大於等於 : ',num3)     # 將 num3 顯示出來
```

執行結果 >>

```
a是否等於b:  False
b是否不等於c:  True
a是否大於等於:  True
```

</> 程式解說 >>

- 第 11 行：a=56 不等於 b=24，所以輸出 False。
- 第 12 行：b=24 不等於 c=38，所以輸出 True。
- 第 13 行：a=56 大於 c=38，所以輸出 True。

3-4 邏輯運算子

邏輯運算子（logical operator）是用來判斷基本的邏輯運算，可控制程式流程。經常與關係運算子合用，運算結果僅有「真（True）」與「假（False）」兩種值，邏輯運算子包括 and、or、not 等運算子。各運算子的功能分別說明如下：

邏輯運算子	說明	範例
and（且）	AND 運算（左、右兩邊都成立時才傳回真）	a and b
or（或）	OR 運算（只要左、右兩邊有一邊成立就傳回真）	a or b
not（非）	真變成假，假變成真	not a

程式初學者可以利用真值表（truth table）來觀察邏輯運算會更清楚。真值表是條列運算元真（T）及假（F）的全部組合以及邏輯運算結果，只要了解 and、or 及 not 的原理，再加上真值表輔助，很快就能熟悉邏輯運算，不需要去硬背它。

邏輯 and（且）

邏輯 and 必須左右兩個運算元都成立，運算結果才為真，任何一邊為假（False）時，執行結果都為假。例如下面指令的邏輯運算結果為真：

```
a = 10
b = 20
a < b and a != b        #True
```

邏輯 and 真值表如下：

a	b	a and b
T	T	True
T	F	False
F	T	False
F	F	False

邏輯 or（或）

邏輯 or 只要左右兩個運算元任何一邊成立，運算結果就為真，例如下面指令的邏輯運算為真：

```
a = 10
b = 20
a < b or a == b          #True
```

左邊的式子 a<b 成立，運算結果就為真，不需要再判斷右邊運算式了。邏輯 or 真值表如下：

a	b	a or b
T	T	True
T	F	True
F	T	True
F	F	False

邏輯 not（非）

邏輯 not 是邏輯否定，用法稍微不一樣，只要有 1 個運算元就可以運算，它是加在運算元左邊，當運算元為真，not 運算結果為假，下面的指令運算結果為真：

```
a = 10
b = 20
not a<5                  #True
```

原本 a<5 不成立，前面加一個 not 就否定了，變成只要 a 不小於 5 都成立，所以運算結果為真，邏輯 not 真值表如下：

a	not a
T	False
F	True

接著我們將以簡單兩道指令說明邏輯運算子的用法：

```
num = 24
result = (num % 6 == 0) and (num % 4 == 0)
```

使用 and 運算子，由於 24 能同時被 6 及 4 整除，所以 result 回傳 True。

我們再來看另外一個例子：

```
total = 31
value = total % 3 == 0 or total % 7 == 0
```

使用 or 運算子，由於 31 無法被 3 及 7 整除的要求，所以 value 回傳「False」。

這裡還有一點要補充說明的重點是，在 Python 程式語言中，當使用 and、or 運算子做邏輯運算時，會採用所謂的「快捷運算」（Short-circuit），我們先以 and 運算子為例來說明，快捷運算的判斷原則是如果第一個運算元回傳 True，才會繼續第二個運算的判斷；也就是說；如果第一個運算元回傳 False 就不需要再往下判斷，以加快程式的執行速度。例如：

```
print (15>8) and (58>35)      # 第一個運算結果傳回 True，會繼續往下判斷
```

另外如果快捷運算是應用在 or 運算子，當第一個運算元回傳 False，才會接下去進行第二個運算元的判斷。但如果第一個運算元回傳 True，就不需要再往下判斷，以加速程式的執行速度。

以下範例輸入兩次段考成績及期末考成績，段考只要其中一次及格 (大於 60 分)，期末考必須及格，學期成績才會及格，及格輸出 PASS，否則輸出 FAIL。

〈判斷成績及格 / 不及格〉

檔名 coursePassOrFail.py

```
01 # -*- coding: utf-8 -*-
02 """
03 輸入兩次段考成績及期末考成績
04 段考只要其中一次及格並且期末考及格
05 學期成績才會及格，及格輸出 PASS，否則輸出 FAIL
06 """
07 grade1 = int(input(" 請輸入第一次段考成績："))
08 grade2 = int(input(" 請輸入第二次段考成績："))
09 lastGrade = int(input(" 請輸入期末考成績："))
10
11 if (grade1>=60 or grade2>=60) and lastGrade>=60:
12     print("PASS")
13 else:
14     print("FAIL")
```

執行結果 >>

```
請輸入第一次段考成績：85

請輸入第二次段考成績：64

請輸入期末考成績：58
FAIL
```

</> 程式解說 >>

- 題目要求的及格條件有兩個：
  1.「段考只要其中一次及格」所以用邏輯 or 來判斷。
  2.「期末考必須及格」所以用邏輯 and 來判斷。
- 當運算式使用一個以上的邏輯運算子時，就必須考慮邏輯運算子優先順序的問題，邏輯 not 會第一個計算，接下來是邏輯 and，最後才是邏輯 or。

- 範例中使用兩個邏輯運算子 and 跟 or，如果直接寫成下式，邏輯 and 會先執行，語意變成第二次段考與期末考必須大於 60 分，得到的執行結果就不正確了。

```
grade1>=60 or grade2>=60 and lastGrade>=60
```

- 所以必須先加上括號，強迫條件式先執行邏輯 or 判斷。例如範例輸入第 1 次段考 90 分，第 2 次段考 59 分，期末考成績 80 分，經過如下圖的邏輯判斷之後會得到 True，所以就會顯示 PASS。

```
(grade1>=60 or grade2>=60) and lastGrade>=60
```

True

True

3-5 位元運算子

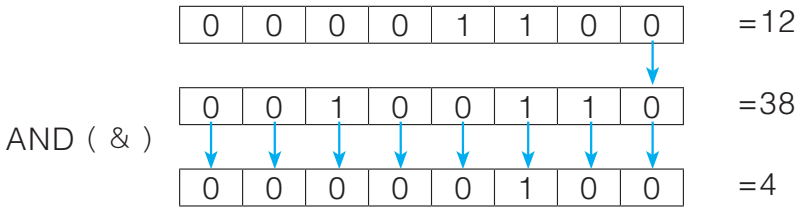
電腦實際處理的資料，其實只有 0 與 1 這兩種資料，也就是採取二進位形式。因此各位可以使用位元運算子（bitwise operator）來進行位元與位元間的邏輯運算。

位元邏輯運算子則是特別針對整數中的位元值做計算。Python 中提供有四種位元邏輯運算子，分別是 &、| 、^ 與 ~：

位元邏輯運算子	說明	使用語法
&	A 與 B 進行 AND 運算	A & B
|	A 與 B 進行 OR 運算	A | B
~	A 進行 NOT 運算	~A
^	A 與 B 進行 XOR 運算	A^B

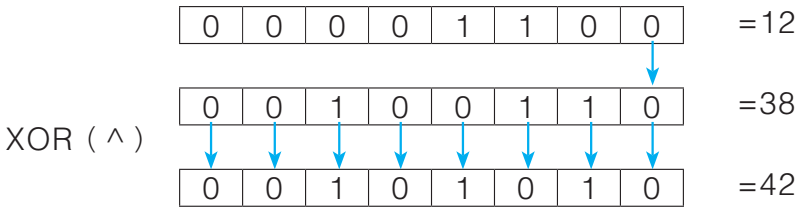
&（AND）

執行 AND 運算時，對應的兩位元都為 1 時，運算結果才為 1，否則為 0。例如 a=12，b=7，則「a&b」得到的結果為 4。因為 12 的二進位表示法為 1100，7 的二進位表示法為 0111，兩者執行 AND 運算後，結果為 $(100)_2$ 也就是 $(4)_{10}$。我們再來以圖解看一個例子，當 a=12，則 a&38 得到的結果為 4，因為 12 的二進位表示法為 00001100，38 的二進位表示法為 00100110，兩者執行 AND 運算後，結果為十進位的 4，如下圖所示：



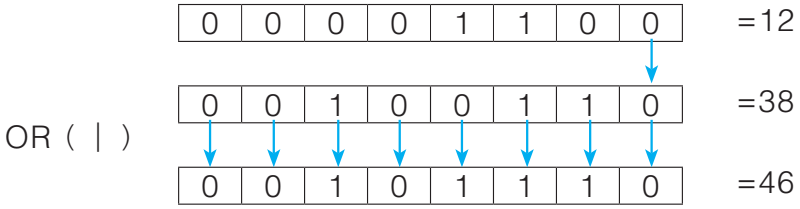
^（XOR）

執行 XOR 運算時，如果對應的兩位元只要任一位元為 1(True)，則運算結果即為 1(True) ，不過當兩者同時為 1(True) 或 0(False) 時，則結果為 0(False)。例如 a=12，則 a^7 得到的結果為 11。我們再來以圖解看一個例子，當 a=12，則 a^38 得到的結果為 42，如下圖所示：



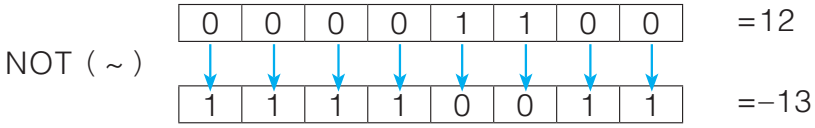
|（OR）

執行 OR 運算時，對應的兩位元只要任一位元為 1 時，運算結果為 1，也就是只有兩位元都為 0 時，才為 0。例如 a=12，則 a ｜ 7 得到的結果為 15。如果 a=12，則 a ｜ 38 得到的結果為 46，如下圖所示：



~（NOT）

NOT 作用是取 1 的補數（complement），也就是 0 與 1 互換。例如 a=12，二進位表示法為 00001100，取 1 的補數後，由於所有位元都會進行 0 與 1 互換，因此運算後的結果得到 -13，如下圖所示：



Tips

所謂「補數」，是指兩個數字加起來等於某特定數（如十進位制即為 10）時，則稱該二數互為該特定數的補數。例如 3 的 10 補數為 7，同理 7 的 10 補數為 3。對二進位系統而言，則有「1 補數系統」和「2 補數系統」兩種，「1 補數系統」是指如果兩數之和為 1，則此兩數互為 1 的補數，亦即 0 和 1 互為 1 的補數。也就是說，打算求得二進位數的補數，只需將 0 變成 1，1 變成 0 即可；例如 01101010_2 的 1 補數為 10010101_2。「2 補數系統」的作法則是必須事先計算出該數的 1 補數，再加 1 即可。

以下程式範例就是位元運算子說明與應用的各種實例。

〈位元運算子綜合應用〉

檔名 bit_operator.py

```
01  # -*- coding: utf-8 -*-
02  """
03  位元運算子綜合應用
04  """
05  x = 12; y = 38
06  bin(x); bin(y)   # 利用 bin() 函數將 x, y 轉為二進位
07  print(x & y)     # & 運算結果是 00000100 再轉成十進位數值
08  print(x ^ y)     # ^ 運算結果是 00101010 再轉成十進位數值
09  print(x | y)     # | 運算結果是 00101110 再轉成十進位數值
10  print(~x)        # ~ 運算結果是採 2 的補數
```

執行結果 >>

```
4
42
46
-13
```

</> 程式解說 >>

- 第 07 行：x=12 其二進位表示法 x=00001100，y=38 其二進位表示法 x=00100110，& 運算結果是 00000100 再轉成十進位數值 4。
- 第 08 行：x=12 其二進位表示法 x=00001100，y=38 其二進位表示法 x=00100110，^ 運算結果是 00101010 再轉成十進位數值 42。
- 第 09 行：x=12 其二進位表示法 x=00001100，y=38 其二進位表示法 x=00100110，| 運算結果是 00101110 再轉成十進位數值 46。
- 第 10 行：x=12 其二進位表示法 x=00001100，其補運算結果是 11110011 再轉成十進位數值 -13。

3-6 位移運算子

位元位移運算子可提供將整數值的位元向左或向右移動所指定的位元數，Python 提供有兩種位元位移運算子：

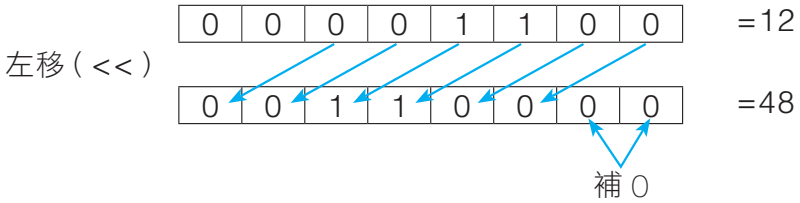
位元位移運算子	說明	使用語法
<<	A 進行左移 n 個位元運算	A<<n
>>	A 進行右移 n 個位元運算	A>>n

<<（左移）

左移運算子（<<）可將運算元內容向左移動 n 個位元，左移後超出儲存範圍即捨去，右邊空出的位元則補 0。語法格式如下：

```
a<<n
```

例如運算式「12<<2」。數值 12 的二進位值為 1100，向左移動 2 個位元後成為 110000，也就是十進位的 48。如下圖所示。

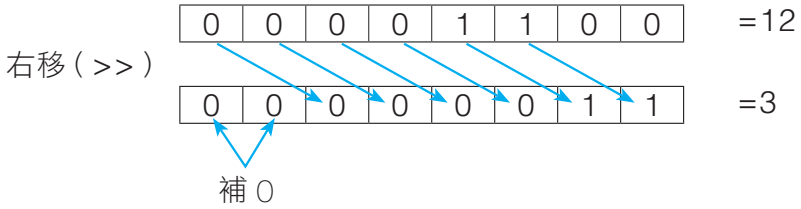


>>（右移）

右移運算子（>>）與左移相反，可將運算元內容右移 n 個位元，右移後超出儲存範圍即捨去。請留意，這時左邊空出的位元，如果這個數值是正數則補 0，負數則補 1。語法格式如下：

```
a>>n
```

例如運算式「12>>2」。數值 12 的二進位值為 1100，向右移動 2 個位元後成為 0011，也就是十進位的 3。如下圖所示：



以下程式範例將實作本節上述圖說的部分，在程式中宣告 a=12，對 38 進行 4 種位元邏輯運算子的運算結果，並對 a 分別左移與右移兩個位元的輸出結果。

〈位元運算子與位移運算子綜合運用〉

檔名 bit_shift.py

```
01  # -*- coding: utf-8 -*-
02  """
03  位元運算子綜合應用
04  """
05
06  a=12
07  print("%d&38=%d" %(a,a&38))      #AND 運算
08  print("%d|38=%d" %(a,a|38))      #OR 運算
09  print("%d^38=%d" %(a,a^38))      #XOR 運算
10  print("~%d=%d"%(a,~a))           #NOT 運算
11  print("%d<<2=%d" %(a,a<<2))      # 左移運算
12  print("%d>>2=%d" %(a,a>>2))
```

執行結果 »

```
12&38=4
12|38=46
12^38=42
~12=-13
12<<2=48
12>>2=3
```

3-7 運算子優先順序

一個運算式中往往包含了許多運算子，運算子優先順序會決定程式執行的順序，這對執行結果有重大影響，不可不慎。如何來安排彼此間執行的先後順序，就需要依據優先權來建立運算規則了。當運算式使用超過一個運算子時，例如 z=x+3*y，就必須考慮運算子優先順序。這個運算式會先執行 3*y 的運算，再把運算結果與 x 相加，最後才將相加的結果指定給 z，得到算式的答案。記得小時候數學課時，最先背誦的口訣就是「先乘除，後加減」，這就是優先順序的基本概念。

當遇到一個 Python 的運算式時，首先區分出運算子與運算元，接下來就依照運算子的優先順序作整理動作。例如：當運算式中有超過一種運算子時，會先執行算術運算子，其次是比較運算子，最後才是邏輯運算子。比較運算子的優先順序都是相同的，會由左到右依序執行，而算術和邏輯運算子則有優先順序。

以下是 Python 中各種運算子計算時的優先順序：

算術運算子的優先順序（由高到低）

算術運算子	說明
**	乘冪
*,/	乘法和除法
//	整數除法
%	取餘數
+,-	加法和減法

邏輯運算子的優先順序（由高到低）

邏輯運算子	說明
not	邏輯非
and	邏輯且
or	邏輯或

當然也可利用「()」括號來改變優先順序。最後由左至右考慮到運算子的結合性（associativity），也就是遇到相同優先等級的運算子會由最左邊的運算元開始處理。括號運算子擁有最高的優先權，需要先被執行的運算就加上括號 ()，括號 () 內的運算式就會先執行。例如：

```
x = 100 * (90 - 30 + 45)
```

上面運算式中有四個運算子：=、*、- 和 +，根據運算子優先順序的規則，括號內的運算會先執行，優先順序如下：-、+、*、=。

〈運算子優先順序綜合應用〉

檔名 precedence.py

```
01  # -*- coding: utf-8 -*-
02  """
03  運算子優先順序綜合應用
04  """
05  x = 2; y = 3
06  z = 9*(4/x + (9+x)/y)
07
08  print("x=", x)
09  print("y=", y)
10  print("9*(21/x + (9+x)/y)=", z)
```

執行結果 »

```
x= 2
y= 3
9*(21/x + (9+x)/y)= 127.5
```

3-8 上機實作演練－成績單統計小幫手

又到了實作演練的時候了，本章的主題是製作成績單統計程式，輸入 10 位學生的姓名、數學、英文及國文三科的成績，計算總分、平均並由平均判斷屬於甲、乙、丙、丁哪一個等級。

3-8-1 程式範例描述

這次學生的成績不用 input() 函數一個個輸入，太費時了，筆者事先建立了 scores.csv 檔案，檔案裡包含 10 位學生的姓名、數學、英文及國文三科的成績，所以本章節將會介紹如何讀取 CSV 檔案。

此次演練的題目要求如下：

1. 讀入 CSV 檔 - 檔名：scores.csv
2. 計算總分、平均以及等級（甲、乙、丙、丁）
   甲：平均 80~100 分　　乙：平均 60~79 分
   丙：平均 50~59 分　　丁：平均 50 分以下
3. 輸出學生姓名、總分、平均 (取到小數點 1 位) 及等級

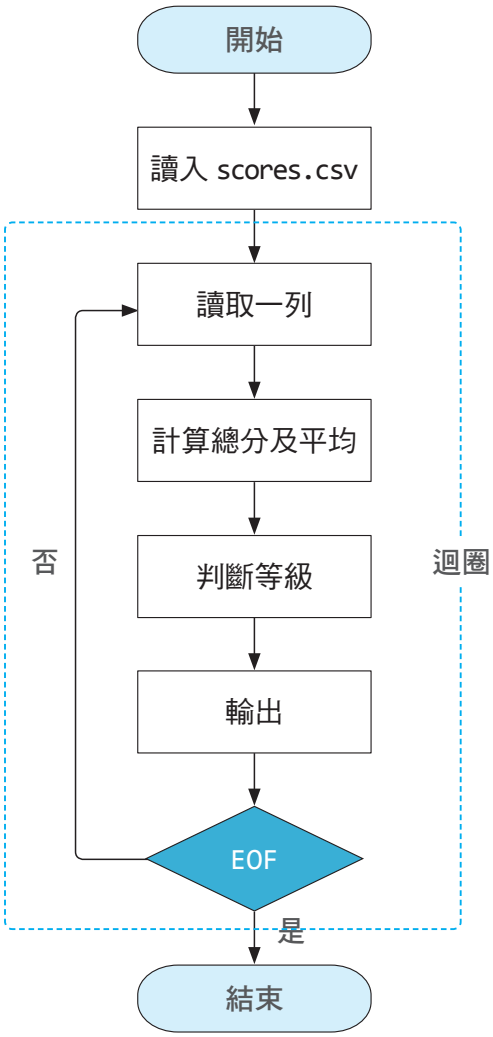
▶ 輸入說明

讀入 scores.csv 檔

▶ 輸出範例

```
姓名    總分  平均    等級
王小華   242  81      甲
陳小淩   179  60      乙
周小杰   136  45      丁
胡小宇   265  88      甲
蔡小琳   229  76      乙
方小花   285  95      甲
林小傑   232  77      乙
黃小偉   160  53      丙
陳小西   181  60      乙
胡小淩   291  97      甲
```

▶ 流程圖



3-8-2 讀取 CSV 檔

CSV 檔案是常見的開放資料（open data）格式，所謂開放資料是指可以被自由使用和散佈的資料，雖然有些開放資料會要求使用者標示資料來源與所有人，但大部分政府資料的開放平台，可以免費取得，這些開放資料會以常見的開放格式於網路上公開。不同的應用程式如果想要交換資料，必須透過通用的資料格式，CSV 格式就是其中一種，全名為 Comma-Separated Values，欄位之間以逗號 (,) 分隔，與 txt 檔一樣都是純文字檔案，可以用記事本等文字編輯器編輯。

CSV 格式常用在試算表以及資料庫，像是 Excel 檔可以將資料匯出成 CSV 格式，也可以匯入 CSV 檔案編輯。網路上許多的開放資料通常也會提供使用者直接下載 CSV 格式資料，當各位學會了 CSV 檔的處理之後，就可以將這些資料做更多的分析應用了。

本範例使用的 scores.csv 檔內容如下：

```
姓名,數學,英文,國文
王小華,90,80,72
陳小凌,60,80,39
周小杰,40,49,47
胡小宇,80,90,95
蔡小琳,65,88,76
方小花,87,100,98
林小傑,95,50,87
黃小偉,56,45,59
陳小西,62,54,65
胡小凌,99,97,95
```

Python 內建 csv 模組（module），非常輕鬆就能夠處理 CSV 檔案。csv 模組是標準模組庫模組，使用前必須先用 import 指令匯入。現在就來看看 csv 模組的用法。

csv 模組用法

csv 模組可以讀取 CSV 檔案也可以寫入 CSV 檔案，存取之前必須先開啟 CSV 檔案，再使用 csv.reader 方法讀取 CSV 檔案裡的內容，如下所示：

```
import csv                          # 載入 csv.py

with open("scores.csv", encoding="utf-8") as csvfile:
                                    # 開啟檔案指定為 csvfile
    reader = csv.reader(csvfile)    # 回傳 reader 物件
    for row in reader:              # for 迴圈逐行讀取資料
        print(row)
```

上面程式執行結果如下：

```
['姓名', '數學', '英文', '國文']
['王小華', '90', '80', '72']
['陳小凌', '60', '80', '39']
['周小杰', '40', '49', '47']
['胡小宇', '80', '90', '95']
['蔡小琳', '65', '88', '76']
['方小花', '87', '100', '98']
['林小傑', '95', '50', '87']
['黃小偉', '56', '45', '59']
['陳小西', '62', '54', '65']
['胡小凌', '99', '97', '95']
```

Tips

如果 CSV 檔案與 .py 檔案放在不同資料夾，必須加上檔案路徑。

open() 指令會將 CSV 文件開啟並回傳檔案物件，範例中將檔案物件指定給 csvfile 變數，預設文件使用 unicode 編碼，如果文件使用不同的編碼，必須使用 encoding 參數設定編碼。本範例所使用的 CSV 檔是 utf-8 格式，所以 encoding="utf-8"。

csv.reader() 函數會讀取 CSV 檔案轉成 reader 物件回傳，reader 物件是可以疊代（iterator）處理的字串（String）list 列表物件，上面程式中使用 reader 變數來接收 reader 物件，再透過 for 迴圈逐行讀取資料：

```
reader = csv.reader(csvfile)        # 回傳 reader 物件
for row in reader:                  #for 迴圈逐行讀取資料放入 row 變數
```

列表物件 list 是 Python 的容器資料型態（Container type），它是一串由逗號分隔的值，用中括號 [] 包起來，如下所示：

```
['方小花', '87', '100', '98']
```

上面 list 物件共有 4 個元素，利用中括號 [] 搭配元素的索引（index）就能存取每一個元素，索引從 0 開始，由左至右分別是 row[0]、row[1]……以此類推，例如要取得第 4 個元素的值，可以如下表示：

```
name = row[3]
```

Tips

使用 with 指令開啟檔案

讀取或寫入檔案前，必須先使用 open() 函數將檔案開啟，當讀取或寫入完成時，必須使用 close() 函數將檔案關閉，確保資料已正確被讀出或寫入檔案。如果在調用 close() 方法之前發生異常，那麼 close() 方法將不會被呼叫，舉例來說：

```
f = open("scores.csv")      # 開啟檔案
csvfile = f.read()          # 讀取檔案內容
1 / 0                       #error
f.close()                   # 關閉檔案
```

第 3 行程式犯了分母為 0 的錯誤，執行到此程式就會停止執行了，所以 close() 不會被調用，這樣就可能會有檔案損壞或資料遺失的風險。

有兩個方式可以避免這樣的問題，一是加上 try…except 指令捕捉錯誤；另外一個方法是使用 with 指令。Python 的 with 指令配有特殊的方法，檔案開啟之後如果程式發生異常會自動調用 close() 方法，如此一來，就能確保檔案會被正確安全地打開和關閉。

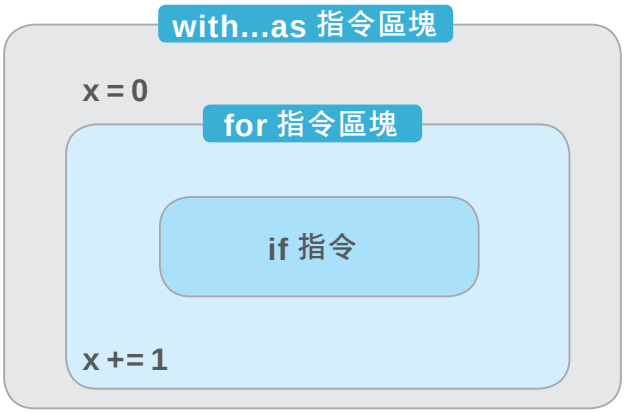
3-8-3 程式碼說明

此次範例使用的 scores.csv 檔案，包含 10 位學生的姓名、數學、英文及國文三科的成績，我們需要將三科成績加總、平均，再以平均分數來評比等級。

scores.csv 檔案第一列是標題，必須略過不處理，所以我們利用一個變數 x 來記錄目前讀取的列數，x 初始值為 0，x 必須大於 0，if 條件判斷式才會為真，如下所示：

```
with open("scores.csv",encoding="utf-8") as csvfile:
    x = 0            # 設定 x 初始值為 0
    for row in csv.reader(csvfile):
        if x > 0:    # 當 x>0, if 判斷式為真
        …
        x += 1       # 相當於 x=x+1
```

撰寫 Python 程式的時候不同區塊記得縮排，上面指令共有三個區塊，with...as 區塊、for 迴圈區塊、if 區塊，x=0 的宣告必須放在 for 迴圈外面，x+=1 指令放在 for 迴圈內，這樣每一次迴圈 x 才會累加，如下圖所示。



進入 if 區塊之後就要將三科成績加總，由於 csv.reader 函數讀入都是字串（String）格式，所以計算前必須先轉換成 int 格式，再將加總結果指定給變數 scoreTotal：

```
scoreTotal = int(row[1]) + int(row[2]) + int(row[3])
```

接著再計算平均值，題目要求平均值取到小數點 1 位：

```
average = round(scoreTotal / 3, 1)
```

利用平均來評定等級，4 個等級的分數區間如下：

甲：平均 80~100 分

乙：平均 60~79 分

丙：平均 50~59 分

丁：平均 50 分以下

平均 80 分 ~100 分就給「甲」，80 分也在這一區間，因此必須用「>=」大於等於運算子，如果只用 average > 80 來判斷，80 分就不會落在這一區間。

平均 60 分 ~79 分就給「乙」，這個判斷需要兩個條件，average > = 60 以及 average < 80，而且兩個條件必須都符合，所以必須用 and(且) 來判斷：

```
average > = 60 and average < 80
```

由於這兩個條件是一個數值區間，所以可以寫成下式，表示 average 的值必須在 60~79 以內。

```
60 <= average < 80
```

完整 if...else 指令如下：

```
if average >= 80 :
    grade = " 甲 "
elif 60 <= average < 80:
    grade = " 乙 "
elif 50 <= average < 60:
    grade = " 丙 "
else:
    grade = " 丁 "
```

最後只要將總分（scoreTotal）、平均（average）以及等級（grade），用 print 指令輸出就完成了。

執行結果 >>

```
    姓名   總分  平均   等級
1   王小華  242  80.7  甲
2   陳小凌  179  59.7  丙
3   周小杰  136  45.3  丁
4   胡小宇  265  88.3  甲
5   蔡小琳  229  76.3  乙
6   方小花  285  95.0  甲
7   林小傑  232  77.3  乙
8   黃小偉  160  53.3  丙
9   陳小西  181  60.3  乙
10  胡小凌  291  97.0  甲
```

以下是完整的程式碼：

〈成績單統計小幫手〉

檔名 Review_scores.py

```
01 # -*- coding: utf-8 -*-
02 """
03 程式名稱：成績單統計小幫手
04 題目要求：
05 讀入 CSV 檔
06 列出總和、平均以及等級 ( 甲、乙、丙、丁 )
07 甲：平均 80~100 分
08 乙：平均 60~79 分
09 丙：平均 50~59 分
10 丁：平均 50 分以下
11 """
12 import csv
13
14 print("{0:<3}{1:<5}{2:<4}{3:<5}{4:<5}".format("", " 姓名 ",
   " 總分 ", " 平均 ", " 等級 "))
15 with open("scores.csv",encoding="utf-8") as csvfile:
16     x = 0
17     for row in csv.reader(csvfile):
18
19         if x > 0:
20             scoreTotal = int(row[1]) + int(row[2]) + int(row[3])
21             average = round(scoreTotal / 3, 1)
22
23             if average >= 80 :
24                 grade = " 甲 "
25             elif 60 <= average < 80:
26                 grade = " 乙 "
27             elif 50 <= average < 60:
28                 grade = " 丙 "
29             else:
30                 grade = " 丁 "
31
32             print("{0:<3}{1:<5}{2:<5}{3:<6}{4:<5}".
               format(x, row[0], scoreTotal, average, grade))
33
34         x += 1
```

Chapter 3

重點回顧

1. 運算式是由運算子（operator）與運算元（operand）所組成。
2. 運算子如果只有一個運算元，稱為一元運算子，例如：表達負值的「-23」。當有兩個運算元，則是二元運算子，像是算術運算子加、減、乘、除。
3. 負數也可以使用減法 (-) 運算子的符號來表示。當負數進行減法運算時，為了避免與減法運算子的分辨混淆，最好應以括號 ( ) 隔開負數。
4. 「/」與「//」都是除法運算子，「/」會有浮點數；「//」會將除法結果的小數部分去掉，只取整數，「%」是取得除法後的餘數。
5. 如果運算結果並不指派給其它變數，則運算結果的資料型態將以運算元中資料型態最大的變數為主。
6. Python 指派運算子有兩種指派方式：單一指派及複合指派。
7. 在 Python 中單一個等號「=」是指派，兩個等號「==」用來做關係比較，不可混用。
8. 使用關係運算子時，所運算的結果就是成立或者不成立兩種。
9. 邏輯運算子（logical operator）是用來判斷基本的邏輯運算，可控制程式流程。邏輯運算子包括 and、or、not 運算子。
10. 在 Python 程式語言中，當使用 and、or 運算子做邏輯運算時，會採用所謂的快捷運算來加快執行速度。
11. 可以使用位元運算子（bitwise operator）進行位元與位元間的邏輯運算。
12. 位元位移運算子可提供將整數值的位元向左或向右移動所指定的位元數。
13. 當運算式使用超過一個運算子時，必須考慮運算子的優先順序。會先執行算術運算子，其次是比較運算子，最後才是邏輯運算子。
14. 比較運算子的優先順序都是相同的，會由左到右依序執行，而算術和邏輯運算子則有優先順序。

學習評量

選擇題

(　　) 1. 要轉數值轉為整數，使用哪一個函數？
(A) count()　(B) int()　(C) float()　(D) decimal()。

(　　) 2. 運算式「a = 15 % 4」，變數 a 會儲存？
(A) 3　(B) 5　(C) 4　(D) 0。

(　　) 3. 下列有關運算子與運算式的描述，何者有誤？
(A) 運算式是由運算子與運算元所組成
(B) 運算子如果只有一個運算元則稱為「一元運算子」
(C)「/」與「%」都是除法運算子，「/」會有浮點數；「%」會將除法結果的小數部分去掉
(D) 運算式運算結果的資料型態將以運算元中資料型態最大的變數為主。

(　　) 4. 有關指派運算子的描述，何者有誤？
(A) 主要作用是將等號右方的資料值指派給等號左方的變數
(B) 有單一指派和複合指派兩種指派方式
(C) 在 Python 中單一個等號「=」是指派，兩個等號「==」用來做關係比較
(D) 使用指派運算子時，變數的值不必事先設定。

(　　) 5. 有關邏輯運算子的描述，何者有誤？
(A) 運算結果僅有「真（True）」與「假（False）」兩種值
(B) 包括 and、or、not 等運算子
(C) result = (48 % 6 == 0) and (24 % 4 == 0) 運算後的 result 回傳 False
(D) 使用 and、or 運算子做邏輯運算時，會採用快捷運算。

填充題

1. 運算式是由_____與_____所組成。

2. Python 指派運算子有兩種指派方式：_____及_____。

3. 邏輯運算子包括_____、_____、_____運算子。

4. 在 Python 程式語言中，當使用 and、or 運算子做邏輯運算時，會採用所謂的_____來加快執行速度。

5. 比較運算子的優先順序都是相同的，會_____依序執行。

簡答題

1. 請問執行下列程式碼得到的 result 值是多少？

```
n1 = 80
n2 = 9
result = n1 % n2
```

2. 請問執行下列程式碼得到的 result 值是多少？

```
n1 = 4
n2 = 2
result = n1 ** n2
```

3. a=15，則「a&10」的結果值為何？

4. 試說明 ~NOT 運算子的作用。

5. 請問 "==" 運算子與 "=" 運算子有何不同？

6. 已知 a=20、b=30，請計算下列各式的結果：

```
a-b%6+12*b/2
(a*5)%8/5-2*b
(a%8)/12*6+12-b/2
```

7. 開心蛋糕店有販售蛋糕一個 60 元、餅乾一盒 80 元與咖啡 55 元，請試著撰寫一個程式讓使用者可以輸入訂購數量，並計算出訂購的總金額。例如：

```
請輸入購買的蛋糕數量：2
請輸入購買的餅乾數量：5
請輸入購買的咖啡數量：3
購買總金額：685
```

【提示】將蛋糕、餅乾以及咖啡金額放在 list 串列，使用者輸入的數量分別放於 3 個變數，商品乘以對應的價格，再加總起來。

Chapter

4 流程控制

程式的進行順序可不是像我們中山高速公路，由北到南一路通到底，有時複雜到像北宜公路上的九彎十八轉，幾乎讓人暈頭轉向。程式設計最重要的部分其中之一就是流程控制，想要寫出好的程式，程式執行的流程相當重要，如果沒有它們，絕對不能做什麼複雜的工作，本章將會跟各位討論 Python 的各種流程控制結構。

學習大綱

- 三種流程控制結構
- if...else 條件式
- 多重選擇
- 巢狀 if
- while 迴圈
- for 迴圈
- 巢狀迴圈
- break 指令
- continue 指令

經過近數十年來程式語言的不斷發展，結構化程式設計（Structured Programming）的趨勢慢慢成為程式開發的一種重要概念，Python 主要是依照原始碼的順序由上而下執行，不過有時也會視需要來改變順序，此時就可由流程控制指令來告訴電腦，應該優先以何種順序來執行指令。

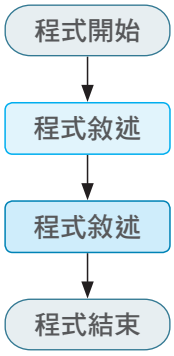
程式流程控制像是公路四通八達的設計方向

4-1 認識流程控制

通常大部分程式碼是由上往下一行接著一行循序執行，但是對於重複性高的動作，並不適合以循序的方式來執行。任何 Python 程式不管其結構如何複雜，皆可利用以下基本控制流程來加以表達，基本上可以利用這三種基本流程控制結構來加以表達與陳述。

4-1-1 循序結構

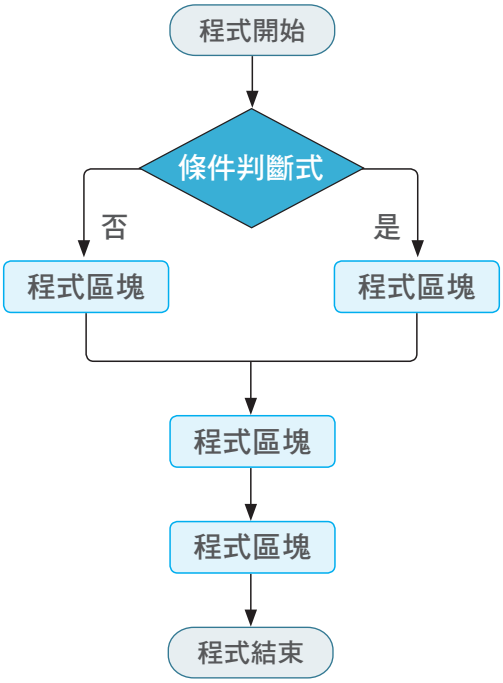
以程式第一行指令為進入點，由上往下執行到程式的最後一行指令。右圖是循序結構流程示意圖。



4-1-2 決策結構

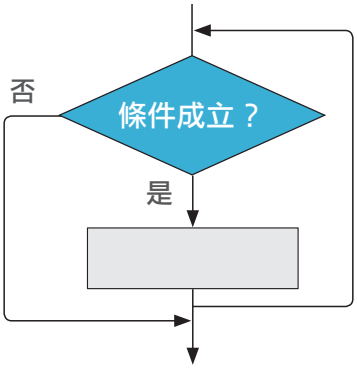
決策結構是讓程式根據測試條件的成立與否，來選擇應該執行的程式碼區塊。如果條件為真，則執行某些指令，一旦條件為假，則執行另一些指令。在口語上，我們經常會這麼想：如果遇到情況 A，就去執行動作 A，否則如果是情況 B 就執行動作 B，就好比各位開車到十字路口看到的燈號，紅燈要停車，綠燈則通行，不同的目的地有不同的方向，可以根據不同的狀況來選擇需要的路徑。也就是說，決策結構代表程式會依指定條件來決定程式走向。以下是決策結構流程示意圖：

決策結構就像馬路的十字路口需要選擇方向



4-1-3 迴圈結構

會重複執行一個程式區塊的程式碼，直到符合特定的結束條件為止。迴圈流程控制則是符合條件時重複執行一段程式指令，Python 有 for 迴圈與 while 迴圈。



4-2 決策流程控制

決策流程控制是一種條件控制指令，包含有一個條件判斷式，如果條件為真，則執行某段程式指令，一旦條件為假，則執行另一段程式指令。接著就來介紹 Python 語言中和決策流程控制相關的指令功能。

4-2-1 if...else 條件式

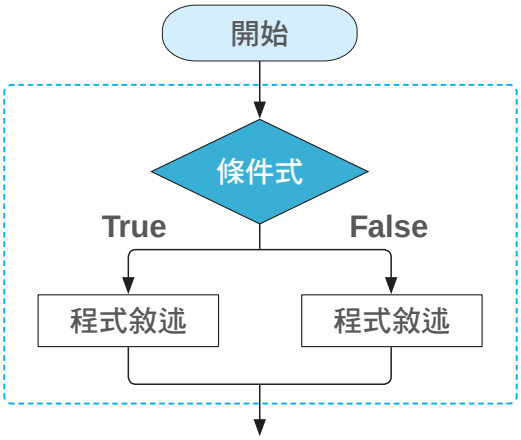
if...else 條件式的作用是判斷條件式是否成立，是個相當普遍且實用的指令，當條件成立（True）就執行 if 的指令，條件不成立（False，或用 0 表示）則執行 else 的指令。如果有多重判斷，可以加上 elif 指令。if 條件式語法如下：

```
if 條件判斷式：
    # 如果條件成立，就執行這裡面的指令
else：
    # 如果條件不成立，就執行這裡面的指令
```

例如各位要判斷 a 變數的內容是否大於等於 b 變數，條件式就可以這樣寫：

```
if a >= b：
    # 如果 a 大於等於 b，就執行這裡面的指令
else：
    # 如果 a「不」大於或等於 b，就執行這裡面的指令
```

if...else 條件式流程圖如下：



if...else 條件式的語法使用上，如果條件不成立時不執行任何指令，可以省略 else 語法，如下所示。

```
if 條件判斷式：
    # 如果條件成立，就執行這裡面的指令
```

如果 if...else 條件式使用 and 或 or 等邏輯運算子，建議加上括號區分執行順序，來提高程式可讀性。例如：

```
if (a==c) and (a>b)：
    # 如果 a 等於 c 而且 a 大於 b，就執行這裡面的指令
else：
    # 如果上述條件不成立，就執行這裡面的指令
```

另外，Python 提供一種更簡潔的 if...else 條件表達式（Conditional Expressions），格式如下：

```
X if C else Y
```

根據條件式傳回兩個運算式的其中一個，上式當 C 為真時傳回 X，否則傳回 Y。例如判斷整數 X 是奇數或偶數，原本程式會這樣表示：

```
if (x % 2)==0:
    y="偶數"
else:
    y="奇數"
print('{0}'.format(y))
```

改成表達運算式只要簡單一行程式就能達到同樣的目的，如下行所示：

```
print('{0}'.format("偶數" if (X % 2)==0 else "奇數"))
```

當 if 判斷式為真就傳回「偶數」，否則就傳回「奇數」。

以下程式範例我們將透過實際範例來練習 if...else 指令的用法。範例題目是製作一個簡易的閏年判斷程式。讓使用者輸入西元年（4 位數的整數 year），判斷是否為閏年。

滿足下列兩個條件之一即是閏年：

1. 逢 4 年閏（除 4 可整除）但逢 100 年不閏（除 100 不可整除）
2. 逢 400 年閏（除 400 可整除）

〈閏年判斷〉

檔名 leapYear.py

```
01  # -*- coding: utf-8 -*-
02  """
03  程式名稱：閏年判斷程式
04  題目要求：
05  輸入西元年 (4 位數的整數 year) 判斷是否為閏年
06  條件 1. 逢 4 閏 ( 除 4 可整除 ) 而且逢 100 不閏 ( 除 100 不可整除 )
07  條件 2. 逢 400 閏 ( 除 400 可整除 )
08  滿足兩個條件之一即是閏年
09  """
10  year = int(input(" 請輸入西元年份："))
11
12  if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
13      print("{0} 是閏年 ".format(year))
14  else :
15      print("{0} 是平年 ".format(year))
```

執行結果 »

```
請輸入西元年份：2000
2000是閏年
```

</> 程式解說 >>

- 第 10 行：輸入一個西元年份，但記得要利用 int() 函數將其轉換成整數型別。
- 第 12~15 行：判斷是否為閏年，條件 1. 逢 4 閏 (除 4 可整除) 而且逢 100 不閏 (除 100 不可整除)，條件 2. 逢 400 閏 (除 400 可整除)，滿足兩個條件之一即是閏年。

請讀者查詢下列西元年是否為閏年：
1900 (平年)、1996 (閏年)、2004 (閏年)、2017 (平年)、2400 (閏年)

4-2-2 多重選擇

如果條件判斷式不只一個，就可以再加上 elif 條件式，elif 就像是「else if」的縮寫，雖然使用多重 if 條件指令可以解決各種條件下的不同執行問題，但始終還是不夠精簡，這時 elif 條件指令就能派上用場了，還可以讓程式碼可讀性更高。請留意！ if 敘述視我們程式中邏輯上的需求，後面並不一定要有 elif 和 else，可以只有 if，或是 if/else，或是 if/elif/else 三種情形。格式如下：

```
if 條件判斷式 1：
    # 如果條件判斷式 1 成立，就執行這裡面的指令
elif 條件判斷式 2：
    # 如果條件判斷式 2 成立，就執行這裡面的指令
else：
    # 如果上面條件都不成立，就執行這裡面的指令
```

例如：

```
if a==b：
    # 如果 a 等於 b，就執行這裡面的指令
elif a>b：
    # 如果 a 大於 b，就執行這裡面的指令
else：
    # 如果 a 不等於 b 而且 a 小於 b，就執行這裡面的指令
```

以下是透過實際範例來練習 if 多重判斷的用法。範例題目是偵測目前時間決定問候語。

〈偵測目前時間決定問候語〉

檔名 currentTime.py

```
01  # -*- coding: utf-8 -*-
02  """
03  程式名稱：偵測目前時間決定問候語
04  題目要求：
05  依目前時間判斷 (24 小時制 )
06  5 點 ~10:59，輸出「早安」
07  11 點 ~17:59，輸出「午安」
08  18~ 凌晨 4:59，輸出「晚安」
09  """
10
11  import time
12
13  print (" 現在時間 :{}".format( time.strftime("%H:%M:%S")))
14  h = int( time.strftime("%H") )
15
16  if h>5 and h < 11:
17      print (" 早安 !")
18  elif h >= 11 and h<18:
19      print (" 午安 !")
20  else:
21      print (" 晚安 !")
```

執行結果 »

```
現在時間:Monday, Jan 29 22:40:51
晚安!
```

範例中取得目前時間來判斷早上、下午或晚上顯示適當的問候語。Python 的 time 模組提供各種與時間有關的功能，time 模組是 Python 標準模組庫的模組，使用前要先執行 import，再利用 strftime 函數將時間格式化成為我們想要的格式，例如下式是獲取目前的時間。

```
import time
time.strftime("%H:%M:%S")     # 16:29:26 (24 小時制 下午 4:29:26)
time.strftime("%I:%M:%S")     # 04:29:26 (12 小時制 下午 4:29:26)
```

括號內是要設定的格式參數，常用的參數如下表：

格式參數	說明
%a	星期縮寫，例如 Mon
%A	完整星期名稱，例如 Monday
%b	月份縮寫，例如 Apr
%B	完整月份名稱，例如 April
%c	日期與時間，例如 Mon Apr 01 16:43:52 2017
%d	月的第幾天，值為 01~31
%U	年的第幾周，值為 00~53
%w	周的第幾天，值為 0~6 (星期天為 0)
%Y	西元年，例如：2017
%y	西元年末兩位數，例如：17
%m	月份，值為 01~12
%H	小時，24 小時制，值為 00~23
%I	小時，12 小時制，值為 01~12
%M	分鐘，值為 00~59
%S	秒數，值為 00~61(秒的範圍允許閏秒)
%p	AM. 或 PM.

請注意格式符號的大小寫。下式會顯示星期，月日以及時分秒。

```
import time
print(time.strftime("%A, %b %d %H:%M:%S"))
```

執行結果 >>

```
Monday, Jan 29 22:43:22
```

4-2-3 巢狀 if

有時在出現 if 條件指令中，又有另外一層的 if 條件指令，這樣多層的選擇結構，就稱作巢狀（nested）if 條件指令。通常在示範巢狀 if 判斷式的使用方式，比較常見的作法是以數字範圍或成績來討論多重選擇。也就是說，當不同的成績會有不同等級的合格證書。「如果是 60 以上就給第一張合格證書，如果是 70 分以上就再給第二張合格證書，如果是 80 分以上就再給第三張合格證書，如果是 90 分以上就再給第四張合格證書，如果 100 分以上就再給全能專業的合格證書。依據巢狀 if 指令，我們可以把程式碼撰寫如下：

```
getScore= int(input(" 請輸入分數 :"))
if getScore >= 60:
   print(' 第一張合格證書 ')
   if getScore >= 70 :
      print(' 第二張合格證書 ')
      if getScore >= 80 :
         print(' 第三張合格證書 ')
         if getScore >= 90 :
            print(' 第四張合格證書 ')
            if getScore == 100 :
               print(' 全能專業的合格證書 ')
```

其實這種一層又一層往下探索的 if 指令，我們可以利用 if/elif 指令將這樣的多重選擇以條件運算逐一過濾，選擇最適合的條件（True）來執行某個區段的指令，語法如下：

```
if 條件運算式 1:
    符合條件運算式 1 要執行的程式區塊
elif 條件運算式 2:
    符合條件運算式 2 要執行的程式區塊
elif 條件運算式 N:
    符合條件運算式 N 要執行的程式區塊
else:
    如果所有條件運算式都不符合，則執行此程式區塊
```

當條件運算式 1 不符合時會向下尋找到適合的條件運算式為止。其中 elif 指令是 else if 之縮寫。elif 指令可以依據條件運算來產生多個指令；其條件運算式之後也要有冒號，表示下方符合此條件運算式的程式區塊要進行縮排。

以下範例就是一種典型的巢狀 if 及 if/elif 指令的綜合使用範例，這個程式利用 if 判斷所查詢的成績屬於哪一種等級。除此之外，以下程式還加入了另外一個判斷，當所輸入的成績整數值沒有介於 0 到 100 之間，則會輸出「輸入錯誤，所輸入的數字必須介於 0-100 間」的提示訊息。

〈巢狀 if 的綜合使用範例〉

檔名 nested_if.py

```
01 # -*- coding: utf-8 -*-
02 """
03 巢狀 if 的綜合使用範例
04 """
05 score = int(input(' 請輸入期末總成績：'))
06
07 # 第一層 if/else 指令 判斷所輸入成績是否介於 0 到 100 之間
08 if score >= 0 and score <= 100:
09     # 第二層 if/elif/else 指令
10     if score <60:
11         print('{0} 分以下無法取得合格證書 '.format(score))
12     elif score >= 60 and  score <70:
13         print('{0} 分的成績等級是 D 級 '.format(score))
14     elif score >= 70 and  score <80:
15         print('{0} 分的成績等級是 C 級 '.format(score))
16     elif score >= 80 and  score <90:
17         print('{0} 分的成績等級是 B 級 '.format(score))
18     else:
19         print('{0} 分的成績等級是 A 級 '.format(score))
20 else:
21     print(' 輸入錯誤， 所輸入的數字必須介於 0-100 間 ')
```

執行結果 >>

```
請輸入期末總成績：85
85  分的成績等級是B級
```

程式解說 >>

- 第 07~21 行：第一層 if/else 指令 判斷所輸入成績是否介於 0 到 100 之間。
- 第 10~19 行：第二層 if/elif/else 指令，判斷所查詢的成績屬於哪一種等級。

4-3 迴圈

重複結構主要談到的是迴圈控制的功能，根據所設立的條件，重複執行某一段程式指令，直到條件判斷不成立，才會跳出迴圈。簡單來說，重複結構可以執行相同的程式區塊，也可讓程式碼更符合結構化設計的精神。例如想要讓電腦計算出 1+2+3+4+…+10 的值，在程式碼中並不需要各位大費周章地從 1 累加到 10，這時只需要利用迴圈結構，原本是很繁瑣又重複的運算，用迴圈很輕鬆就能達成目標。Python 有 while 迴圈跟 for 迴圈，以下來看相關的用法。

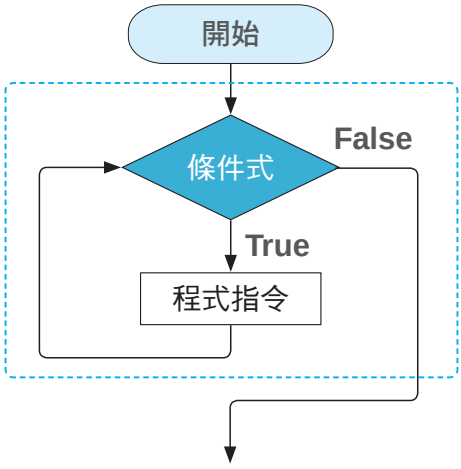
4-3-1 while 迴圈

如果所要執行的迴圈次數確定，那麼使用 for 迴圈指令就是最佳選擇。但對於某些不確定次數的迴圈，while 迴圈就可以派上用場了。while 迴圈指令與 for 迴圈指令類似，都是屬於前測試型迴圈。前測試型迴圈的運作方式就是在程式指令區開頭時必須先檢查條件判斷式，當判斷式結果為真時，才會執行區塊內的指令。

while 迴圈也是利用條件式判斷真假，當條件式為真時，才會執行迴圈裡面的指令，當條件式為假時，程式就會跳出迴圈，格式如下：

```
while 條件判斷式：
    # 如果條件判斷式成立，就執行這裡面的指令
```

流程圖如下：



while 迴圈必須自行加入控制變數起始值以及遞增或遞減運算式，撰寫迴圈程式時必須檢查離開迴圈的條件是否存在，如果條件不存在會讓迴圈一直循環執行而無法停止，導致「無窮迴圈」。

迴圈結構通常需要具備三個要件：

1. 變數初始值
2. 迴圈條件式
3. 調整變數增減值

例如下面的程式：

```
i=1
while i < 10:          # 迴圈條件式
    print( i)
    i += 1             # 調整變數增減值
```

當 i 小於 10 時會執行 while 迴圈內的指令，所以 i 會加 1，直到 i 等於 10，條件式為 False，就會跳離迴圈了。

4-3-2 for 迴圈

for 迴圈又稱為計數迴圈，是程式設計中較常使用的一種迴圈型式，可以重複執行固定次數的迴圈。如果程式設計上所需要的迴圈執行次數固定，那麼 for 迴圈指令就是最佳選擇。Python 的 for 迴圈可以走訪任何序列項目，序列項目可以是數字串列、列表（list）或字串（string），按序列順序執行，語法架構如下：

```
for 元素變數 in 序列項目：
    # 執行的指令
else:
    #else 的程式區塊，可加入或者不加入
```

也就是說，使用 for 迴圈，可加入或者不加入 else 指令。上述 Python 語法所代表的意義是 for 迴圈會將一序列（sequence），例如字串 string 或串列 list 內所有的元素走訪一遍，走訪的順序是依目前序列內元素項目（item）的順序來處理。例如下列的 x 變數值都可以作為 for 迴圈的走訪序列項目：

```
x = "abcdefghijklmnopqrstuvwxyz"
x = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday',
'Friday', 'Saturday']
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

此外，如果要計算迴圈的執行次數，在 for 迴圈控制指令中各位必須設定「迴圈起始值」、「結束條件」以及每執行完一次迴圈的增減值。for 迴圈每次執行一次時，如果增減值沒有特別指定，會自動累加 1，加到條件符合為止。例如以下是一數字串列 1 ～ 5，利用 for 迴圈將數字 print 出來：

```
x = [1, 2, 3, 4, 5]
for i in x:
    print (i)
```

執行結果 >>

```
1
2
3
4
5
```

數字串列比較有效率的寫法，可以直接使用 range() 函數，range() 函數格式如下：

```
range([ 起始值 ], 終止值 [, 增減值 ])
```

數字串列由「起始值」開始到「終止值」的前一個數字為止，沒有指定起始值，預設為 0；沒有指定增減值，預設為遞增 1。有關 range() 函數的使用範例如下：

- range(3) 代表由索引值 0 開始，輸出 3 個元素，即 0,1,2 共 3 個元素。
- range(1,6) 代表由索引值 1 開始，到索引編號 6-1 前結束，也就是說索引編號 6 不包括在內，即 1,2,3,4,5 共 5 個元素。
- range(4,10,2) 代表由索引值 4 開始，到索引編號 10 前結束，也就是說索引編號 10 不包括在內，遞增值為 2，即 4,6,8 共 3 個元素。

下段的程式碼示範了在 for 迴圈中搭配使用 range() 函數輸出 2 到 11 之間的偶數。

```
for i in range(2, 11, 2):
    print(i)
```

執行結果 >>

```
2
4
6
8
10
```

在使用 for 迴圈時還有一個地方要特別留意，就是 print() 函數！如果該 print() 有縮排的話，就表示在 for 迴圈內要執行的工作，就會依照執行次數來輸出。如果沒有縮排，就表示不在 for 迴圈內，只會輸出最後的結果。

我們知道利用 range() 函數及 for 迴圈除了可以進行累加的工作外，也可以配合 range() 函數的參數，進行更多變化的加總工作。例如將某一範圍內所有 5 的倍數進行加總。以下程式範例說明如何利用 for 迴圈進行某一個數字範圍內 5 的倍數進行加總。

〈某一個數字範圍內 5 的倍數進行加總〉

檔名 sum5.py

```
01  # -*- coding: utf-8 -*-
02  """
03  某一個數字範圍內 5 的倍數進行加總
04  """
05  sum = 0 # 儲存加總結果
06
07  # 進入 for/in 迴圈
08  for count in range(0, 21, 5):
09      sum += count # 將數值累加
10
11  print('5 的倍數累加結果 =',sum) # 輸出累加結果
```

執行結果 >>

```
5的倍數累加結果= 50
```

</> 程式解說 >>

- 第 08~09 行：將 5、10、15、20 數字進行累加。

另外，執行 for 迴圈時，如果想要知道元素的索引值，可以使用 Python 內建的 enumerate 函數。語法如下：

```
for 索引值 , 元素變數 in enumerate( 序列項目 ):
```

例如：(enumerate.py)

```
names = ["Eileen", "Jennifer", "Brian"]
for index, x in enumerate(names):
    print ("{0}--{1}".format(index, x))
```

執行結果 >>

```
0--Eileen
1--Jennifer
2--Brian
```

4-3-3 巢狀迴圈

接下來還要為各位介紹一種 for 的巢狀迴圈（Nested loop），也就是多層次的 for 迴圈結構。在巢狀 for 迴圈結構中，執行流程必須先等內層迴圈執行完畢，才會逐層繼續執行外層迴圈。兩層式的巢狀 for 迴圈結構格式如下：

```
for 外層迴圈

    [                    ]
    for 內層迴圈
        [                    ]
```

例如九九乘法表，就可以利用兩個 for 迴圈輕鬆完成。以下範例就來看看如何利用兩個 for 迴圈製作九九乘法表。

〈九九乘法表〉

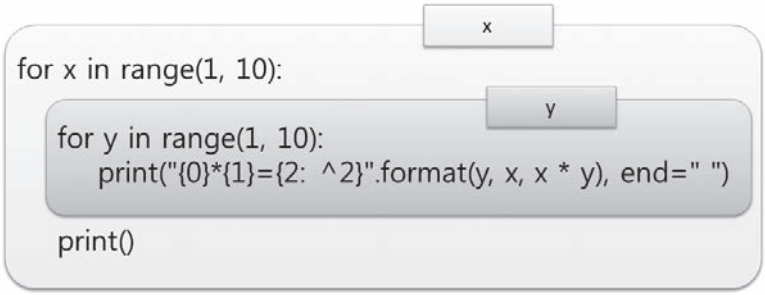
檔名 99Table.py

```
01 # -*- coding: utf-8 -*-
02 """
03 程式名稱：九九乘法表
04 """
05
06 for x in range(1, 10):
07     for y in range(1, 10):
08         print("{0}*{1}={2: ^2}".format(y, x, x * y), end=" ")
09     print()
```

執行結果 »

```
1*1=1  2*1=2  3*1=3  4*1=4  5*1=5  6*1=6  7*1=7  8*1=8  9*1=9
1*2=2  2*2=4  3*2=6  4*2=8  5*2=10 6*2=12 7*2=14 8*2=16 9*2=18
1*3=3  2*3=6  3*3=9  4*3=12 5*3=15 6*3=18 7*3=21 8*3=24 9*3=27
1*4=4  2*4=8  3*4=12 4*4=16 5*4=20 6*4=24 7*4=28 8*4=32 9*4=36
1*5=5  2*5=10 3*5=15 4*5=20 5*5=25 6*5=30 7*5=35 8*5=40 9*5=45
1*6=6  2*6=12 3*6=18 4*6=24 5*6=30 6*6=36 7*6=42 8*6=48 9*6=54
1*7=7  2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49 8*7=56 9*7=63
1*8=8  2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64 9*8=72
1*9=9  2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81
```

九九乘法表是巢狀迴圈非常經典的範例，如果各位學過其他程式語言，相信會對於 Python 語法的簡潔感到驚嘆。從這範例各位可以清楚了解巢狀迴圈的運作，以下簡稱外層 for 迴圈為 x 迴圈，內層 for 迴圈為 y 迴圈，如下圖。



當進入 x 迴圈時 x=1，等到 y 迴圈從 1 到 9 執行完成之後，才會再回到 x 迴圈繼續執行，y 迴圈 print 指令不換行，y 迴圈執行完成離開 y 迴圈之後，才執行 print() 換行，執行完成就會得到第一行的九九乘法表，如下所示。

```
1*1=1   2*1=2   3*1=3   4*1=4   5*1=5   6*1=6   7*1=7   8*1=8   9*1=9
```

當 x 迴圈都執行完畢，九九乘法表就完成了。請特別留意，一般初學者容易犯錯的地方就是內外迴圈的指令交錯，這一點在多層次的巢狀迴圈架構，迴圈間不可交錯，否則會造成錯誤。

4-4 continue 和 break 指令

我們之前介紹的兩種迴圈指令，在正常的執行情形之下，while 迴圈是在進入之前先判斷，條件不成立的話就會離開迴圈，而 for 迴圈則是在所有指定的元素都被取出之後，就結束迴圈執行。不過迴圈也可以使用 continue 及 break 來中斷迴圈執行，break 指令的主要用途是用來跳躍出迴圈，break 指令就像它的英文意義一般，代表中斷的意思，如果是在迴圈中遇到指定的情況就要離開整個迴圈的話，那麼就要使用 break，它是用來跳離最近的 for、while 的程式迴圈，並將控制權交給所在區塊之外的下一行程式。也就是說，break 指令是用來中斷迴圈的執行，並離開目前所在的迴圈。

4-4-1 break 指令

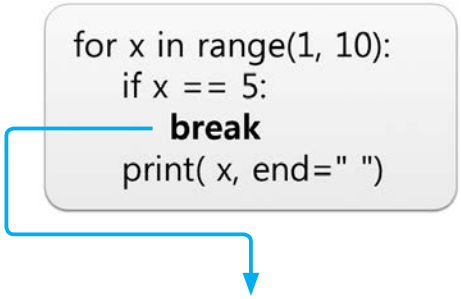
當遇到巢狀迴圈時，break 指令只會跳離最近的一層迴圈，而且多半會配合 if 指令來一起使用。例如：

```
for x in range(1, 10):
    if x == 5:
        break
    print( x, end=" ")
```

執行結果 >>

```
1 2 3 4
```

當 x 等於 5 的時候執行 break 離開 for 迴圈，for 迴圈就不會繼續往下執行了，請參考右圖示意圖。



如果各位希望某一段迴圈程式可以不斷執行，直到某一個條件成立時，才離開迴圈程式的執行工作，這個時候就可以利用 break 指令。

例如我們可以要求使用者輸入一個介於 1 到 100 之間的數字，如果輸入錯誤則會告知所輸入的數字是太大或太小，並要求使用者重複輸入，一直到所輸入的數字和原先預設的答案完全相同，這時就可以利用 break 指令來跳離迴圈，並輸出答案正確或遊戲結束等訊息。請看以下程式碼：

```
number=9
while True:
    guess = int(input('輸入 1~100 之間的數字 --> '))
    if guess == number:
        print('你猜對了，數字是：', number)
        break
```

4-4-2 continue 指令

continue 指令的功用是強迫 for、while 等迴圈指令，結束目前正在迴圈本體區塊內進行的程序，並將控制權轉移到下一次迴圈開始處。也就是說，在迴圈的進行過程中，如果遇到 continue 指令，會馬上放棄同一層之後所有要執行的指令，程式流程直接回到 while 或是 for 那一行接下去執行。不同之處是 break 會結束迴圈執行，而 continue 指令只會結束接下來迴圈區塊裡的指令，並不會離開迴圈，例如：

```
for x in range(1, 10):
    if x == 5:
        continue
    print( x, end=" ")
```

執行結果 >>

```
1 2 3 4 6 7 8 9
```

當 x 等於 5 的時候執行 continue 指令，程式不會繼續往下執行，所以 5 沒有被 print 出來，for 迴圈仍繼續運作，請參考右圖示意圖。

```
for x in range(1, 10):
    if x == 5:
        continue
    print( x, end=" ")
```

4-5 上機實作演練－密碼驗證程式不求人

本節將利用一個範例來複習前面所介紹的相關內容，製作一個簡單的密碼驗證程式。

4-5-1 程式範例描述

撰寫一個 Python 程式能夠讓使用者輸入密碼，並且進行簡單密碼驗證工作，不過輸入次數以三次為限，超過三次則不准登入，假如目前密碼為 1234。

▶ 輸入說明

第一次輸入時各位可以刻意輸入錯誤的密碼，則程式會輸出「密碼發生錯誤 !!」的文字輸出訊息，並要求使用者再輸入一次密碼。各位可以再試著輸入錯誤的密碼，同樣會輸出「密碼發生錯誤 !!」的文字輸出訊息，並要求使用者再輸入一次密碼，當連續輸入錯誤密碼超過三次，則不再允許使用者繼續輸入密碼進行登入工作，並會輸出「密碼錯誤三次，取消登入 !!」的提示訊息。但是如果輸入過程中所輸入的密碼正確，則會輸出「密碼正確 !!」，並正確結束程式的執行。

▶ 輸出範例

```
請輸入密碼:1235
密碼發生錯誤!!

請輸入密碼:3388
密碼發生錯誤!!

請輸入密碼:1234
密碼正確!!
```

4-5-2 程式碼說明

以下列出完整的程式碼，其中 password 預設的密碼為數字 1234，而變數 i 則是用來記錄輸入的總次數，當輸入次數超過三次，則會跳離迴圈。

〈簡單的密碼驗證程式〉

檔名 password.py

```
01 # -*- coding: utf-8 -*-
02 """
03 能夠讓使用者輸入密碼，
04 並且進行簡單密碼驗證工作
05 不過輸入次數以三次為限，超過三次則不准登入，
06 假如目前密碼為 1234。
07 """
08
09 password=1234 # 利用 password 變數來儲存密碼以供驗證
10 i=1
11
12 while i<=3: # 輸入次數以三次為限
13     new_pw=int(input(" 請輸入密碼 :"))
14     if new_pw != password:  # 如果輸入的密碼與 password 不同
15         print(" 密碼發生錯誤 !!")
16         i=i+1
17         continue # 跳回 while 開始處
18     else:
19         print(" 密碼正確 !!")
20         break
21 if i>3:
22         print(" 密碼錯誤三次，取消登入 !!\n"); # 密碼錯誤處理
```

重點回顧

1. 三種基本流程控制結構：循序結構、決策結構、迴圈結構。
2. if...else 條件式的作用是判斷條件式是否成立，當條件成立（True）就執行 if 裡的指令，條件不成立（False，或用 0 表示）則執行 else 的指令。如果條件不成立時不執行任何指令，可以省略 else 語法。
3. Python 提供一種更簡潔的 if...else 條件表達式，格式如下：X if C else Y。
4. 如果條件判斷式不只一個，可以加上 elif 條件式，elif 是「else if」縮寫。
5. Python 的 time 模組提供各種與時間有關的功能，使用前要先執行 import。
6. 有時在出現 if 條件指令中，又有另外一層的 if 條件指令，這樣多層的選擇結構，就稱作巢狀 if 條件指令。
7. 重複結構主要談到的是迴圈控制的功能，根據所設立的條件，重複執行某一段程式指令，直到條件判斷不成立，才會跳出迴圈。
8. while 迴圈與 for 迴圈指令類似，都屬於前測試型迴圈，在程式指令區開頭時必須先檢查條件判斷式，當結果為真時，才會執行區塊內的指令。
9. while 迴圈必須自行加入控制變數起始值以及遞增或遞減運算式，撰寫迴圈程式時必須檢查離開迴圈的條件是否存在，如果條件不存在會讓迴圈一直循環執行而無法停止，導致「無窮迴圈」。
10. 迴圈結構需具備三個要件：變數初始值、迴圈條件式、調整變數增減值。
11. Python 的 for 迴圈可以走訪任何序列項目，序列項目可以是數字串列、列表（list）或字串（string）。
12. 數字串列比較有效率的寫法，可以直接使用 range() 函數。
13. 如果 print() 函數有縮排，就表示在 for 迴圈內有工作要執行，會依照執行次數輸出。如果沒有縮排，就表示不在 for 迴圈內，只會輸出最後的結果。
14. 在巢狀 for 迴圈結構中，執行流程必須先等內層迴圈執行完畢，才會逐層繼續執行外層迴圈。
15. 在多層次的巢狀迴圈架構，迴圈間不可交錯，否則會造成錯誤。
16. break 指令是用來中斷迴圈的執行，並離開目前所在的迴圈。
17. continue 指令只會結束接下來迴圈區塊裡的指令，並不會離開迴圈。

Chapter 4

學習評量

選擇題

(　　) 1. 對於 for/in 迴圈的描述，何者不正確？
(A) 巢狀迴圈架構，迴圈間不可交錯
(B) 計數器要有起始值和終止值
(C) for 迴圈可以走訪任何序列項目
(D) 遞增值預設加 2。

(　　) 2. 對於 while 迴圈的描述，何者不正確？
(A) 是一種可以重複執行固定次數的迴圈
(B) else 指令不可以省略
(C) 進入迴圈並不會做條件檢查
(D) 必須檢查離開迴圈的條件是否存在。

(　　) 3. for/in 迴圈每次執行一次時，如果增減值沒有特別指定，預設值為何？　(A) 0　(B) 1　(C) -1　(D) 3。

(　　) 4. 對於迴圈的描述，何者不正確？
(A) 在 for/in 迴圈中，還可以包含其它的 for/in 迴圈
(B) 巢狀迴圈內外迴圈可以交錯使用
(C) 如果跳離迴圈的條件設定不當，有可能陷入無窮迴圈
(D) while 迴圈會先檢查條件運算式。

(　　) 5. 試問下列程式碼中，最後 k 值會為多少？

```
k=10
while k<=13:
    k +=1
print(k)
```

(A) 14　(B) 12　(C) 13　(D) 10。

填充題

1. 迴圈指令包含：可計次的_____迴圈和不可計次的_____迴圈。

2. 數字串列比較有效率的寫法，可以直接使用_____函數。

3. _____指令是用來中斷迴圈的執行，並離開目前所在的迴圈。

4. _____指令的功用是強迫 for、while 等迴圈指令，結束目前正在迴圈本體區塊內進行的程序，並將控制權轉移到迴圈開始處。

5. 迴圈結構通常需要具備三個要件：_____、_____、_____。

簡答題

1. 請試著撰寫一個程式，讓使用者傳入一數值 N，判斷 N 是否為 3 的倍數，是請印出 True，不是請印出 False。

2. 請使用 while 迴圈計算 1 到 100 所有整數的和。

3. 請使用 for 迴圈計算 1 到 100 所有整數的和。

4. 請利用 for 迴圈指令來讓使用者輸入 n 值，並計算出 1!+2!+…+n! 的總和。如下所示：

```
1!+2!+3!+4!+…+n-1!+n!
```

5. 請寫出下列指令中 while 迴圈輸出的 count 值。

```
count = 1
while count <= 14:
    print(count)
    count += 3
```

6. 以 while 迴圈撰寫 1~50 的偶數和。

Chapter

5 字元與字串

一個英文字母、數字或符號，我們稱它為字元，「字串」這兩個字從字面上的意思來看，可以解釋成「把字元一個一個給串起來」。字串的用途相當的廣泛，它可以比數值性的資料表達出更多的訊息。例如一個人的名字、一首歌的句子甚至一整個段落的文字。在程式設計的世界中，操作字串只是基本功夫，Python 中字串有多種的表示方式，本章將會跟各位討論 Python 的字元與字串簡介與各種應用。

學習大綱

- 建立字串
- 常用的跳脫字元
- 參數格式化輸出
- 透過索引值取某個字元
- 透過切片取某段字串
- split () 函數分割字串
- 字串常用運算子
- 字串常用函數與方法

字串是型別為 String 的物件，可以將一連串字元放在單引號或雙引號來表示，內建函數 str() 可以利用它將資料轉為字串，而 String 型別提供的字串方法來處理字串物件，本章將說明這些功能實用的字串處理方法。另外，使用字串免不了要配合一些特殊的字元或符號，輸出字串時為了讓資料更具體化，格式化字串也是討論範圍。

5-1 建立字串

字串（String）是由一連串的字元所組成，將一連串字元放在單引號或雙引號括起來，就是一個字串（String）。例如：

```
01  "13579"
02  "1+2"
03  "Hello, how are you?"
04  "I'm all right, but it's raining."
05  'I\'m all right, but it\'s raining.'
```

用來包含字串的雙引號與單引號可以交替使用，上例中第 4 行字串由雙引號括住，第 5 行字串則用單引號括住，然而第 5 行字串裡已經有單引號，就要避免使用單引號括住字串，如果遇到只能使用單引號的狀況下，可以在字串裡的單引號之前加上跳脫字元「\」。

如果輸出字串時想要分行顯示，可以在要斷行的地方加入「\n」，例如：

```
str1 = "Hello!\nHow are you?"
print(str1)
```

執行結果 >>

```
Hello!
How are you?
```

如果要將字串指定給特定變數時，可以使用「=」指派運算子。以下就是 Python 字串建立方式：

```
wordA = ''                # 當單引號之內沒有任何字元時，它是一個空字串
wordB = 'P'               # 單一字元
wordC ="Python"           # 建立字串時，也可以使用雙引號。
```

當各位想直接將數值資料轉為字串，可以使用內建函數 str() 來達成，例如：

```
str()                     # 輸出空字串 ''
str(123)                  # 將數字轉為字串 '123'
```

當字串較長時，也可以利用「\」字元將過長的字串拆成兩行。例如：

```
wordD ="What's wrong with you? \
Nothing!"
```

對於 Python 來說，也能利用三重單或雙引號把多行字串，固定其輸出的模式，例如：

```
>>> title="""
末祝
    2018
        新年快樂 """
>>> print(title)

末祝
    2018
        新年快樂
>>>
```

5-2 認識跳脫字元

字串裡有一些特殊的字元無法從鍵盤輸入或是該字元已經被定義為其他用途，如果字串裡要使用這些字元就必須加上跳脫字元（escape character）。例如以單引號包起來的字串，如果在字串資料中遇到單引號時，則必須以 \ 來跳脫（escape）處理，避免被誤認為字串結束。跳脫字元通常由反斜線「\」組成，請看以下實例：

```
str1 = 'it\'s raining.'
```

當直譯器遇到反斜線，就知道下一個字元必須另外處理，就不會將它視為字串結尾的單引號。另外還有一些換行字元、定位字元等無法由鍵盤輸入，也可以用跳脫字元來處理。下表是常用的跳脫字元：

跳脫字元	說明
\	倒斜線
\'	單引號
\"	雙引號
\b	退後 (backspace 鍵)
\n	換行
\t	定位 (tab 鍵)
\uXXXX	\u 加上 4 個 16 進位數表示一個 Unicode 字符

跳脫字元「\」本身還有另外一個用途，就是當程式碼太長時只要在該行末端加一個反斜線可以換行繼續撰寫。例如以下指令 format 指令加上跳脫字元「\」就可以換下一行繼續寫了。（escape.py）

```
a = "Beautiful"
b = len(a)
print("{} 有 {} 個字元 ".\
      format(a, b))
```

以下我們就來實作跳脫字元的範例。

〈顯示特殊字元〉

檔名 escape01.py

```
01  str1 = "Never say \tNever!\nNever say \"Impossible!\"\u2665"
02  print(str1)
03  str2 = "Never say Never\b\b\b\b\b"
04  print(str2)
05  str3 = "c:\\temp"
06  print(str3)
07  str4 = r"c:\temp"
08  print(str4)
```

執行結果 »

```
Never say       Never!
Never say "Impossible!"♥
Never say
c:\temp
c:\temp
```

</> 程式解說 »

- 第 01 行：使用了「\t」跳格、「\n」換行以及「\u2665」顯示愛心符號。
- 第 03 行：使用了「\b」跳脫字元，所以最後的「Never」被刪除了。
- 第 05 行：要印出「c:\temp」，然而其中的字元「\t」是跳脫字元，所以必須再加上「\」跳脫字元才能正確印出。
- 第 07 行：同樣是要印出「c:\temp」，但不使用跳脫字元，而是在字串前加上「r 」前置字元，如此一來，就可以用字串原貌顯示。

5-3 參數格式化輸出

參數格式化輸出就是將資料依所指定的格式輸出，讓閱讀更容易，在 Python 中可以使用 format() 函數就可以格式化資料。例如：

```
num=1.41421
print("num= {:.5f}".format(num))        # num= 1.41421
```

{:.5f} 表示要將數值格式化成小數點後 5 位

```
num=1.41421
print("num= {:7.3f}".format(num))       # num=   1.414
```

其中 {:7.3f } 表示數值總長度為 7 的浮點數，且小數點後 3 位，此處的小數點也在總長度內。從執行結果來看，總長度為 7，但在數值前會補空白。

如果希望數值前補 0，則必須修正格式化字串如下：

旗標字元　　精確度

```
num=1.41421
print("num= {:07.3f}".format(num))      # num=001.414
```

欄寬　　意指浮點數

上圖中的「欄寬」代表字元所佔的寬度；「精確度」則是使用浮點數時可設定輸出的小數位數。下表為 format() 方法中的旗標字元參數功能說明如下：

旗標字元	說明
'#'	配合十六、八進位做轉換時，可在前方補上 0
'0'	數值前補 0
'-'	靠左對齊，若與 0 同時使用，會優於 0
' '	會保留一個空格
>	向右對齊
<	向左對齊
\uXXXX	\u 加上 4 個 16 進位數表示一個 Unicode 字符

此外，str.format() 方法還可以置換欄名，不過必須使用大括號 {} 來包裹要置換的欄名，請注意！大括號 {} 的索引編號由零開始，例如：

```
print('{0}{1}'.format('num = ', 1.41421))
# 輸出 num= 1.41421
```

上述指令表示字串「num=」會帶入欄名 1(即大括號 {0})，而數值 1.41421 則會帶入欄名 2(即大括號 {1})，才會輸出「num= 1.41421」。

我們還可以在大括號中直接指定要輸出的參數名稱，語法如下：

```
{ 欄名 }
```

下例說明大括號 {} 中的欄名如何配合 format() 方法來使用。

```
# 輸出四月有 30 天
print('{month} 月有 {day} 天 '\
      format(month = ' 四 ', day = 30))
```

上述大括號 {} 中使用兩個關鍵字引數，它會以「變數 = 設定值」的用法，所以「month」會被變數值「四」取代；同樣地「day」會被「30」取代。

如果要位置參數與關鍵字引數混合使用，也可以參考以下作法：

```
# 輸出  七月有 31 天
# 四月有 30 天
print('{month} 月有 {0} 天 '\
    .format(30, month = ' 四 '))
```

最後我們還要介紹的是將 format() 方法轉換成指定型式，其語法為：

```
{ 欄名 ： 轉換指定形式 }
```

轉換指定形式請看下表說明：

轉換指定形式	說明
fill	可填補任何字元，但不包含大括號
align	以 4 種字元做對齊 ＜靠左； ＞靠右； ＝填補； ^ 置中
sign	使用「+」、「-」或空格，同 % 格式字串
#	用法與 % 格式運算子同
0	用法與 % 格式運算子同
width	欄寬
,	千位符號，就是每 3 位數就加上逗點
.precision	精確度，用法與 % 格式運算子同
typecode	用法與 % 格式運算子幾乎相同

以下程式範例以 format() 方法的欄名，配合 format-spec(格式規格) 做格式化輸出，同時也使用了 % 格式運算子。(str_format.py)

```
print('PI = %10.5f'%(3.14159))              # 輸出 5 位小數
print('PI = {0:010f}'.format(3.14159))      # 前方補 0 欄寬 10

radius = (3.14159) * 20 *20 # 計算圓面積
area = int(radius) # 將半徑轉換成整數
print(' 靠右 = {0:=>12d}'.format(area))     #* 字元填滿
print(' 置中 = {0:=^12d}'.format(area))
print('PI = {0:.5f}\n'
      ' 圓面積 = {1:,.4f}'.format(3.14159, radius))
      # 圓面積加千位逗點
# 圓面積以十進位、十六進位、二進位輸出
print(' 圓面積 = {0:d}, {0:#b}, {0:#x}'.format(area))
```

執行結果 >>

```
PI =    3.14159
PI = 003.141590
靠右 = ========1256
置中 = ====1256====
PI = 3.14159
圓面積 = 1,256.6360
圓面積 = 1256, 0b10011101000, 0x4e8
```

5-4 字串索引與切片

字串是由字元所組成的串列物件，如果想要取得字串中的字元，有三種方式：

1. 透過索引值（index）取某個字元
2. 使用切片（slice）方法取某段字串
3. 使用 split() 方法分割字串

5-4-1 透過索引值（index）取某個字元

字串物件可以使用索引值來取得字元，索引值從 0 開始，假設有一字串 str1="Hello"，那麼 str1 的長度就是 5，字串裡的元素分別為 str1[0]、str1[1]、str1[2]、str1[3]、str1[4]。

索引值為正值表示從字串開始處往右起算；負值則從字串結尾往左起算，例如 str1[0] 表示取字串第 1 個字元，str1[-1] 表示取字串最後一個字元，譬如以下指令分別是取得第 1 個字元與倒數第 2 個字元：

```
str1 = "Hello!How are you?"
print(str1[0])      # 執行結果：H
print(str1[-2])     # 執行結果：u
```

學習小教室

字串物件一旦被配置，它的內容是不可變的（immutable），重新賦值是建立一個新的字串而不是去修改原字串，原來的字串物件會在適當的時機被 Python 的垃圾回收機制 (Garbage collection) 回收。因此字串物件可以使用索引取出字元，但是不能對某個索引賦值，例如下式就會發生錯誤：

```
str1[0] = "A"
```

5-4-2 透過切片取某段字串

slice 切片顧名思義就是從字串中取出某一段字串，字串的字元具有順序性，利用 [] 運算子來擷取字串單一字元或子字串，稱為「切片」。格式如下：

字串 [起始索引：結束索引：間隔值]

字串如果不間隔取值，間隔值可省略不寫。我們以下表整理出如何使用 [] 運算子存取序列元素。

運算	說明（s 表示序列）
s[n]	依指定索引值取得序列的某個元素
s[n : m]	由索引值 n 至 m-1 來取得若干元素
s[n:]	依索引值 n 開始至最後一個元素
s[:m]	由索引值 0 開始，到索引值 m-1 結束
s[:]	表示會複製一份序列元素
s[::-1]	將整個序列元素反轉

例如以下的切片運算：

```
str1 = "ABCDEFGHIJK"
print(str1[3:6])    # 執行結果 DEF
```

上面指令表示從索引值 3 開始，取到結束索引 6-1 為止，所以會取出 DEF 字串。也就是說，slice 方法取出的字串長度正好就是結束索引與起始索引相減值，舉例來說 str1[3:6] 取出的字串長度就是 3。

如果是從頭開始取字串，那麼起始索引可以省略不寫；如果字串要取到字串結尾，那麼結束索引可以省略不寫，舉例來說：（slice.py）

```
01  str1 = "ABCDEFGHIJK"
02  print(str1[:7:2])     # ACEG
03  print(str1[2::2])     # CEGIK
04  print(str1[::2])      # ACEGIK
```

上述程式第 2 行到第 4 行都是每 2 間隔取字串，第 2 行指令沒有寫起始索引，表示從起始位置也就是索引值 0 開始；第 3 行指令沒有寫結束索引，表示取到最後一個字元；第 4 行指令沒有起始索引也沒有結束索引，表示目標是完整的 str1 字串。

5-4-3 透過 split () 方法分割字串

split() 方法可以根據指定分隔符號將字串分割為子字串，並回傳子字串的串列。格式如下：

```
字串 .split( 分隔符號 , 分割次數 )
```

預設的分隔符號為空字串，包括空格、換行符號 (\n)、定位符號 (\t)。使用 split() 方法分割字串時，會將分割後的字串以串列 (list) 回傳。例如以下範例：

〈字串分割〉

檔名 split.py

```
01  str1 = "Do \none \nthing \nat a time!"
02  print( str1.split() )
03  print( str1.split(' ', 1 ) )
```

執行結果 >>

```
['Do', 'one', 'thing', 'at', 'a', 'time!']
['Do', '\none \nthing \nat a time!']
```

</> 程式解說 >>

- 第 02 行：指令沒有指定分割字元，所以會以空格與換行符號 (\n) 來分割。
- 第 03 行：指令指定以空格分割，分割 2 個子字串之後的字串就不再分割。

以下範例利用前面介紹的切片運算將 26 個英文字母反轉印出，各位不妨練習看看。

〈將 26 個小寫英文字母反轉輸出〉

檔名 reverseString.py

```
01  # -*- coding: utf-8 -*-
02  letters = ""
03  for x in range(97, 123):
04      letters += str(chr(x))
05  print(letters)
06
07  revletters = letters[::-1]
08  print(revletters)
```

執行結果 »

```
abcdefghijklmnopqrstuvwxyz
zyxwvutsrqponmlkjihgfedcba
```

範例中利用 chr() 函數回傳 ASCII 碼對應的字元，並利用 for 迴圈將字元相加後指定給變數 letters，再利用 slice 方式將字串反轉。這裡所用的 ASCII 碼是十進制表示法，97~122 分別對應的是小寫的英文字母 a~z；65~90 對應大寫的英文字母 A~Z，供各位參考。

Tips

chr() 函數可以回傳 ASCII 碼對應的字元，使用 ord() 函數可以傳回字元對應的 ASCII 碼。

〈將 26 個大寫英文字母反轉輸出〉

檔名 reverseStringBIG.py

```
01  # -*- coding: utf-8 -*-
02  letters = ""
03  for x in range(65, 91):
04      letters += str(chr(x))
05  print(letters)
06
07  revletters = letters[::-1]
08  print(revletters)
```

執行結果 »

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ZYXWVUTSRQPONMLKJIHGFEDCBA
```

同理，如果要反轉數字，則可以參考下面範例。

〈將數字反轉輸出〉

檔名 reverseStringDigit.py

```
01  # -*- coding: utf-8 -*-
02  letters = ""
03  for x in range(48, 58):
04      letters += str(chr(x))
05  print(letters)
06
07  revletters = letters[::-1]
08  print(revletters)
```

執行結果 »

```
0123456789
9876543210
```

這裡要補充一點，split() 功用是會把字串分割，但是 join() 方法正好相反，它會把字串結合。

5-5 字串常用運算子

Python 裡的字串可以透過串接運算子「+」將兩個字串進行串接的工作。不過，字串相加兩邊都必須是字串型態，如果是字串與非字串型態相加，非字串型態必須要先利用 str( ) 函數轉換為字串型態再進行運算。例如字串相加，例如：

```
str1 = "Hello!" + "How are you?"
print(str1)    # 執行結果：Hello!How are you?
```

除了以 + 運算子進行字串的串接工作外，也可以使用乘號 (*) 重複字串，例如：

```
str1 = "Hello!" * 3
print(str1)    # 執行結果：Hello!Hello!Hello!
```

5-5-1 比較運算子

前面提過比較運算子可以用來比較兩個數值之間的大小關係，但事實上，Python 的任何物件都可用來做比較運算。Python 字串的大小比較是根據字元的 Unicode 值的大小。例如數字 ' 0 ' ~ ' 9 ' 的 Unicode 值小於大寫字母 ' A ' ~ ' Z '，例如大寫字母 ' A ' ~ ' Z ' 的 Unicode 值小於小寫 ' a ' ~' z ' 字母。而中文字元 Unicode 值又大於剛才所舉的數字字元及英文字母字元。另外，比較運算式可以任意串連，例如，x < y <= z 就相當於 x < y and y <= z。

```
>>> ' 快樂 ' > 'Happy'
True
>>> 'Happy Birthday' < 'happy birthday'
True
>>> 'abc' > 'ABC' > '123'
True
>>> 'HAPPY' == 'happy'
False
```

5-5-2 in 與 not in 運算子

in 和 not in 只適用於序列物件，諸如字串、串列等。應用在字串中 in 運算子可以用來檢測指定的字串是否在另一個字串之中。同理，not in 運算子可以用來檢測指定的字串是否不存在另一個字串之中，例如：

```
>>> str1 = "happy"
>>> "y" in str1
True
>>> "0" in str1
False
>>> "0" not in str1
True
```

5-6 字串常用函數與方法

在撰寫程式的過程當中，字串函數是非常重要而且很實用的功能，由於跟字串有關的方法眾多，有些方法來自於物件（object）；有些則由類別提供屬性和方法。宣告了字串變數之後，表示實作了 str() 類別，而它的方法皆能由宣告的字串物件使用，透過「.」（dot）運算子來取得物件的方法。

常用的函數包括計算字串長度、替換字串、查找字串甚至是比較兩個字串等等。以下就來介紹 Python 提供哪些好用的函數與方法可供字串使用。

5-6-1 計算字串長度—len() 函數

len() 內建函數會傳回字串的字串長度，空白也會計算在內，但是跳脫字元產生的字符算 1 個長度，例如：（len.py）

```
s= "The first wealth is health\u266C"
print("{} 長度是 {}".format(s, len(s)))
```

執行結果 >>

```
The first wealth is health♬ 長度是27
```

我們再來看下一個例子：（left.py）

```
str1 = "Do one thing at a time!"
str2 = str1[13:]
str_w = len(str2)     # 取得字串長度
print(" 取出字串 =「{}」, 長度：{}".format(str2,str_w))
```

執行結果 >>

```
取出字串=「 at a time!」,長度：11
```

5-6-2 與大小寫有關的方法

下表介紹一些跟字母大小寫有關的方法，用法請參考範例。

方法	說明
capitalize()	只有第一個單字的首字元大寫，其餘字元皆小寫
lower()	將字母轉換為小寫
upper()	將字母轉換為大寫
title()	採標題式大小寫，每個單字的首字大寫，其餘皆小寫
islower()	判斷字串是否所有字元皆為小寫
isupper()	判斷字串是否所有字元皆為大寫
istitle()	判斷字串首字元是否為為大寫，其餘皆小寫

〈大小寫轉換與首字大寫〉

檔名 upper.py

```
01  str1="The first wealth is health."
02  print(str1.upper())
03  print(str1.lower())
04  print("health.".capitalize())
```

執行結果 »

```
THE FIRST WEALTH IS HEALTH.
the first wealth is health.
Health.
```

如果想要知道字串是否全部是大寫或全部是小寫，可以用 isupper() 或 islower() 方法查詢，例如：

```
str= "girl"
s=str.islower()    # 執行結果：True
```

isupper() 或 islower() 方法只要字串裡有大小寫摻雜，得到的結果都會是 False。

5-6-3 搜尋特定字串出現次數—count()

資料分析的時候常常需要計算特定字串出現的次數，Python 提供好用的 count() 方法可以使用，格式如下：

```
目標字串 .count( 特定字串 [, 開始索引 [, 結束索引 ]])
```

開始索引與結束索引可省略，表示搜尋整個目標字串，例如：

〈搜尋特定字串出現次數〉

檔名 count.py

```
01  str1="Never say Never! Never say Impossible!"
02  str2=" 浪花有意千重雪，桃李無言一隊春。\n 一壺酒，一竿綸，世上如儂有幾
    人 ?"
03  s1=str1.count("Never",15)
04  s2=str1.count("e",0,3)
05  s3=str2.count(" 一 ")
06  print("{}\n「Never」出現 {} 次 ,「e」出現 {} 次 ".format(str1,s1,s2))
07  print("\n{}\n「一」出現 {} 次 ".format(str2,s3))
```

執行結果 >>

```
Never say Never! Never say Impossible!
「Never」出現1次,「e」出現1次

浪花有意千重雪，桃李無言一隊春。
一壺酒，一竿綸，世上如儂有幾人?
「一」出現3次
```

程式解說 >>

- 第 03 行：從 str1 字串索引 15 的位置開始搜尋。
- 第 04 行：搜尋 str1 從索引值 0 到索引值 3-1 的位置。
- 第 05 行：搜尋整個 str2 字串。

5-6-4 移除字串左右特定字元—strip()、lstrip()、rstrip()

函數 strip() 用於去除字串首尾的字元，lstrip() 用於去除左邊的字元，rstrip() 用於去除右邊的字元，三種方法的格式相同，以下以 strip() 做說明：

```
字串 .strip([ 特定字元 ])
```

特定字元預設為空白字元，特定字元可以輸入多個，例如：（strip.py）

```
str1="Never say Never!"
s1=str1.strip("N!")
print(s1)
```

執行結果 >>

```
ever say Never
```

由於傳入的是 ("N!") 相當於要去除「N」與「!」，執行時會依序去除兩端符合的字元，直到沒有匹配的字元為止，所以上面範例分別去除了左邊的「N」與右邊的「!」字元。

Tips

strip()、lstrip() 與 rstrip() 方法是用來移除字串「左右」兩邊的字元，並不是移除整個字串內匹配的字元喔！

5-6-5 字串替換—replace()

函數 replace() 可以將字串裡的特定字串替換成新的字串，格式如下：

```
字串 .replace( 原字串 , 新字串 [, 替換次數 ])
```

例如：（replace.py）

```
01  str= "Jennifer is a beautiful girl."
02  s=str.replace("Jennifer", "Joan")
03  str= " 蘋果可以做成蘋果汁、蘋果乾、蘋果沙拉 ."
04  s=str.replace(" 蘋果 ", " 葡萄 ")
```

執行結果 »

```
Joan is a beautiful girl.
葡萄可以做成葡萄汁、葡萄乾、葡萄沙拉.
```

5-6-6 搜尋字串—find() 與 index()

find() 方法用來尋找指定字元，回傳第一個出現該字元或字串的索引編號，同樣以索引編號來設定開始和結束的尋找範圍，但是開始索引及結束索引可省略。語法簡介如下：

```
str.find( 字元或字串 [, 開始索引 [, 結束索引 ]])
```

例如：（strfind.py）

```
word = '''We all look forward to the annual ball
              because it's great time to dress up.'''
print(word)
print(word.find('all'))     # 尋找 all 子字串，從索引編號 0 開始
print(word.find('all', 7)) # 尋找子字串 all，從索引編號 7 開始
```

執行結果 >>

```
We all look forward to the annual ball
          because it's great time to dress up.
3
35
```

index() 方法用來回傳指定字元的索引值，所以它的用法和 find() 函數非常接近，同樣以索引編號來設定開始和結束的範圍，語法簡介如下。

```
str.index( 字元或字串 [, 開始索引 [, 結束索引 ]])
```

其中字元或字串：欲尋找的字元或字串，若未找到會回傳錯誤值 ValueError，不可省略。但是開始索引及結束索引可省略。例如：（strindex.py）

```
wd = ''' A very low one.
         If you take away tipping,
         you run risk of losing good service. '''
print(' 字串 :', wd)
print(' 字串 -you 索引值 : ', wd.find('you'))
print(' 找不到字串 : ', wd.find('yov'))
print(' 字串 -one 索引值 : ', wd.index('one'))
print(' 找不到字串 ', wd.index('services'))
```

find() 方法未找到指定的子字串會回傳 -1 值。index() 方法找不到指定子字串則是「ValueError」的錯誤訊息。

```
    print('找不到字串', wd.index('services'))

ValueError: substring not found
```

5-6-7 startswith() 方法與 endswith() 方法

依據設定範圍判斷設定的子字串是否存在於原有字串，若結果相符會以 True 回傳。startswith() 方法用來比對前端字元，endswith() 方法則以尾端字元為主，其語法如下。

```
startswith( 開頭的字元 [, 開始索引 [, 結束索引 ]])
endswith( 結尾的字元 [, 開始索引 [, 結束索引 ]])
```

- 開頭的字元：表示字串中開頭的字元。
- 結尾的字元：字串中結尾的字元。
- 開始索引、結束索引為選項，可利用切片的計算來設定欲查詢字元的索引值。

例如：（startswith.py）

```
wd = 'Programming design'
print(' 字串 :', wd)
print('Prog?', wd.startswith('Prog'))        # 回傳 True
print('gram?', wd.startswith('gram', 0))     # 回傳 False
print('de?', wd.startswith('de', 12))        # 回傳 True
print('ign?', wd.endswith('ign'))            # 回傳 True
print('ing?', wd.endswith('ing', 0, 11))     # 回傳 True
```

執行結果 >>

```
字串: Programming design
Prog? True
gram? False
de? True
ign? True
ing? True
```

startswith() 方法未設參數開始索引，結束索引時，只會搜尋整句的開頭文字是否符合。若要搜尋第二個子句的開頭字元是否符合，startswith() 方法就得加入 start 或 end 參數。endswith() 方法要搜尋非句尾的末端字元，同樣要設開始索引或結束索引參數才會依索引值做搜尋。

以下程式範例是開啟一篇較長的文章 (redcap.txt) 取自格林童話小紅帽，我們試著從這篇文章中找出指定的關鍵字出現的次數。

Python 內建有文字檔案的函數，不需要 import 其它模組就能使用。使用 open() 函數開啟檔案，第一個參數是檔名，第二個參數是使用文字檔的方法，參數說明如下：

參數	說明
r	讀取模式
w	寫入模式開啟檔案
a	寫入模式，寫入的資料會附加在現有的檔案內容之後。
r+	讀取與寫入模式

第 5 行指定了要搜尋的文字串列，裡面有三個字串：grandmother、wolf 以及 Little Red-Cap，透過 for 迴圈搭配 count 函數就能找出這些字串在文章裡出現多少次了。

〈查文字檔內特定字串組合出現次數〉

檔名 redcap.py

```
01  # -*- coding: utf-8 -*-
02  with open("redcap.txt", "r") as f:
03      story=f.read()     # 讀出檔案內容
04
05  words=["grandmother", "wolf", "Little Red-Cap"]
06
07  for w in words:
08      sc=story.count(w)
09      print("{} 出現了 {} 次 ".format(w,sc))
```

執行結果 »

```
grandmother 出現了 8 次
wolf 出現了 3 次
Little Red-Cap 出現了 1 次
```

5-7 上機實作演練 － Open Data 資料擷取與應用

近來政府推行開放資料（Open Data）不遺餘力，不僅設立了「政府資料開放平臺」，各個縣市政府及單位也分別設立了「Open Data」網站供民眾使用，例如交通部中央氣象局開放資料平臺、台北市政府資訊開放平台等等，程式設計人員可以很方便地取得所需的開放資料，透過程式的開發，將這些資料做更有效的應用，本章範例即將介紹如何從政府公開資料平台取得資料並加以運用。

5-7-1 什麼是 Open Data

開放資料是開放、免費、透明的資料，不受著作權、專利權所限制，任何人都可以自由使用和散佈。這些開放資料通常會以開放檔案格式如 CSV、XML 及 JSON 等格式，提供使用者下載應用，經過彙整之後這些開放資料就能提供更有效的資訊甚至成為有價值的商品。

例如有人將政府開放的預算資料轉化成易讀的視覺圖表，讓民眾更了解公共支出狀況；也有人整理政府開放的空汙與降雨數據，彙整成圖表，並且在超標的時候提出警示。

「政府資料開放平臺」，網址為 http://data.gov.tw/，該網站集合了中央及各個地方政府機關的 Open Data，各位不妨前往尋寶、挖寶！

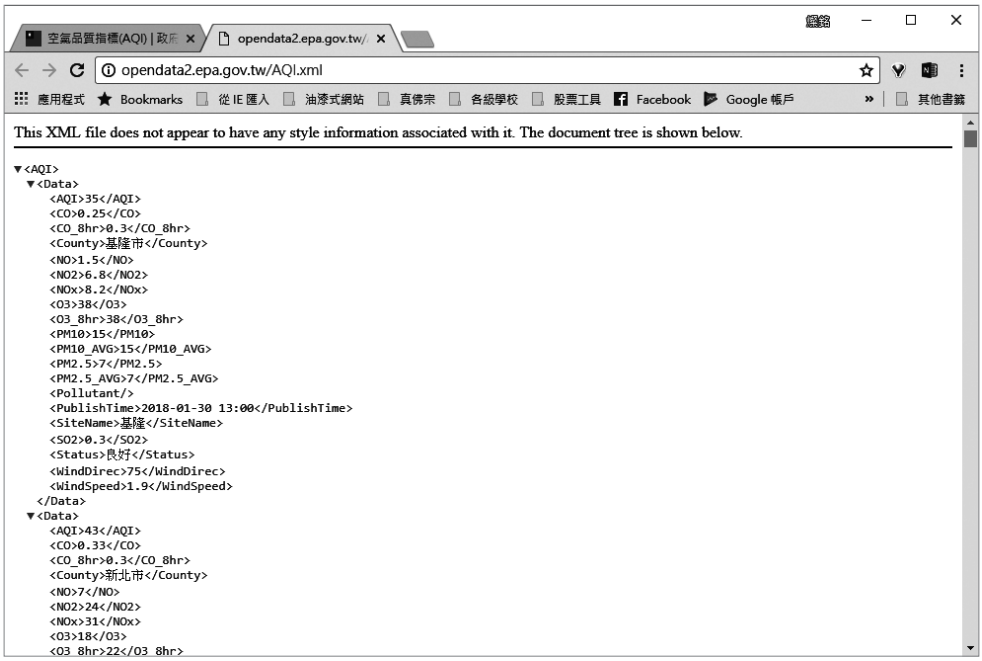
5-7-2 如何取得 Open Data 資料

以下介紹如何從「政府資料開放平臺」取得 Open Data 資料，我們以「空氣品質指標」為例說明。首先在搜尋欄輸入關鍵字查詢，按下搜尋鈕之後就會開始找尋相關的開放資料。

下載頁面如下，這裡提供了 XML、JSON 及 CSV 三種檔案格式，直接點擊連結就可以取得資料。

XML 格式

XML 是可延伸標記語言（Extensible Markup Language），允許使用者自行定義標籤（tags），可以定義每種商業文件的格式，並且能在不同的應用程式中都能使用，XML 格式類似 HTML，與 HTML 最大的不同在於 XML 是以結構與資訊內容為導向，由標籤定義出文件的架構，像是標題、作者、書名等等，補足了 HTML 只能定義文件格式的缺點，XML 具有容易設計的優點，並且可以跨平台使用。如下圖。



XML 格式的網址 http://opendata2.epa.gov.tw/AQI.xml

標籤是以「<」與「>」符號包起來，各個標籤稱為「元素（element）」，標籤必須成對，包括了「開啟標籤」與「結束標籤」，標籤之間的文字稱為內容（content），如下所示。

開始標籤　　結束標籤

```
<SiteName> 基隆 </SiteName>
```

內容（content）

XML 的文件結構就像樹狀結構一樣，以此份 XML 文件為例，<SiteName> 的父元素是 <Data>，每一組 <Data> 元素裡面都包含測站名稱、縣市以及各種指標等子元素。

JSON 格式

JSON（JavaScript Object Notation）格式是 JavaScript 的物件表示法，是輕量的資料交換格式，由於檔案小，適合使用於網路資料傳輸。

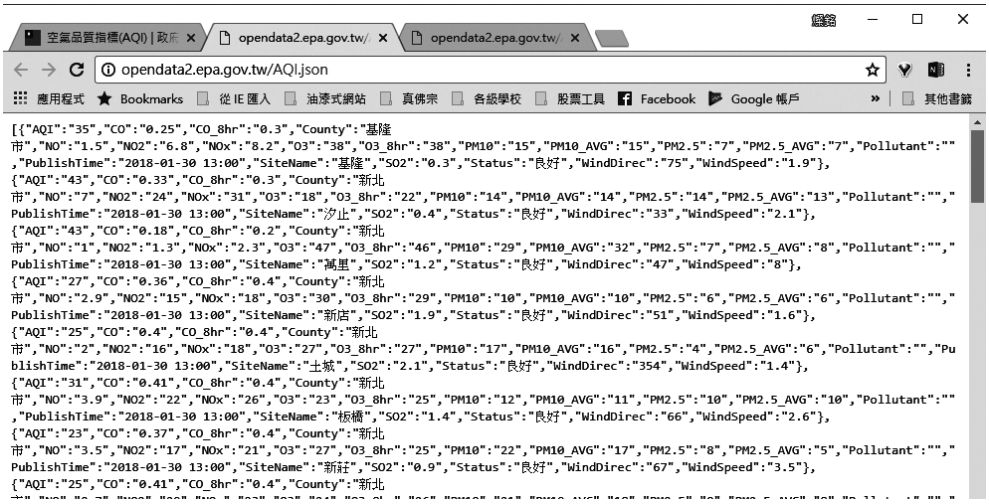
JSON 有兩種形式：串列（陣列）與物件。串列是以「[」符號開始，以「]」符號結束，裡面通常會包含物件集合（collection），每一組集合用逗號分隔，例如：

```
[collection, collection]
```

物件（Object）以「{」符號開始，以「}」符號結束，裡面包含名稱與值，形式如下：

```
{name:value}
```

以此份文件為例，JSON 格式如下：



JSON 格式的網址 http://opendata2.epa.gov.tw/AQI.json

JSON 不需要換行，所以看起來密密麻麻，如果將它寫成以下的格式，就可以很清楚看出每個集合（collection）裡的內容：

```
[
{
    "AQI":"65",
    "CO":"0.32",
    "CO_8hr":"0.5",
    "County":" 基隆市 ",
    ...
},
{
    "AQI":"97",
    "CO":"0.36",
    "CO_8hr":"0.6",
    "County":" 新北市 "
    ...
}
    ...
]
```

XML 是一種標記語言，程式需要解析標記會花費比較多的功夫；而 JSON 格式檔案小、非常容易解析。以下我們將實際操作，各位可以比較兩種格式的差別。

5-7-3 程式範例描述

本範例將從「政府資料開放平臺」取得「空氣品質即時污染指標」的 JSON 與 XML 格式資料，取出「測站名稱」、「縣市」、「PM2.5 濃度」、「狀態」與「發布時間」，分別儲存為 CSV 檔案與 print 輸出。

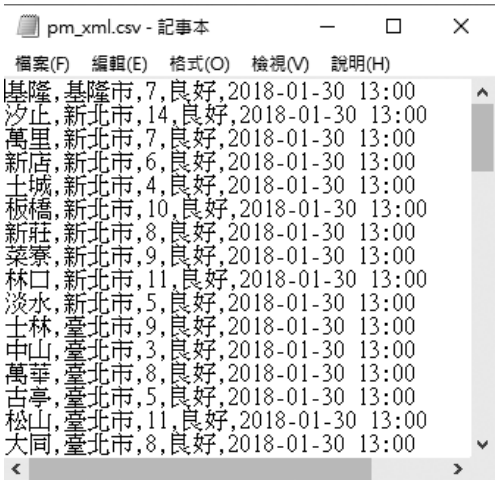
➤ 輸入說明

從「政府資料開放平臺」取得「空氣品質即時污染指標」的 JSON 與 XML 連結 URL。

▶ 輸出範例

XML 格式儲存為 pm_xml.csv 檔案，內容如下圖。

JSON 格式不儲存，只將結果 print 輸出。



▶ 製作步驟

XML 格式擷取步驟如下：

取出網頁（**url**）內容 → 利用 **BeautifulSoup** 模組解析 **XML** 標記 → 儲存為 **CSV** 檔

JSON 格式擷取步驟如下：

取出 **url** 內容 → 利用 **json** 模組解析 **JSON** 資料 → **print** 輸出

5-7-4 程式碼說明

Python 抓取網頁資料有多種模組可供使用，這裡我們使用 urllib.request 模組，以下程式是最基本的使用方式：

```
import urllib.request as ur
with ur.urlopen(od_url) as response:
    get_xml=response.read()
```

urllib.request 模組使用非常簡單，只要將網址傳入 urlopen 函數就會傳回 HttpResponse 物件，接著就可以使用 read() 方法將網頁內容讀取出來。

取出來的網頁內容一大串，我們要寫程式從裡面找到需要的內容，也就是所謂的「爬蟲（Crawler）」。

Python 提供許多的爬蟲模組，由於 HTML 與 XML 網頁都是屬於標記結構，適合使用 BeautifulSoup 模組來解析；而 JSON 格式直接使用 JSON 模組最方便。

以下我們就先來看 XML 格式的解析方式，各位可以開啟範例檔 xml_parse.py 查看完整程式碼。

Beautiful Soup 4 模組

BeautifulSoup 模組使用方式如下：

```
from bs4 import BeautifulSoup
data = BeautifulSoup(get_xml,'xml')
SiteName = data.find_all('SiteName')
```

BeautifulSoup 模組是用來從 HTML 或 XML 格式透過標記找出想要的資料，版本是 Beautiful Soup 4.x，模組庫已更名為 bs4，所以使用前要先載入 bs4 模組。我們可以直接用 from bs4 import BeautifulSoup 來載入 BeautifulSoup 類別。

BeautifulSoup 常用的方法與屬性如下表：

屬性與方法	說明	範例
title 屬性	傳回網頁標題	data.title
text 屬性	除去所有標記，只傳回內容（content）	data.text
find 方法	傳回第一個符合條件的字串物件	data.find('SiteName')
find_all 方法	傳回所有符合條件的字串物件	data.find_all('SiteName')
select 方法	傳回 CSS 選擇器篩選的所有內容	data.select('#id')
get_text 方法	傳回字串物件的標記內容	data.find('SiteName').get_text()

範例中使用 find_all 方法來搜尋特定的標記，如下：

```
SiteName = data.find_all('SiteName')
County = data.find_all('County')
Status = data.find_all('Status')
pm25 = data.find_all('PM2.5')
PublishTime = data.find_all('PublishTime')
```

利用 for 迴圈就可以取出所有的標記內容了，程式指令如下：

```
for i in range(0, len(SiteName)):
    csv_str += "{},{},{},{},{}\n".\
format(SiteName[i].get_text(),County[i].get_text(),pm25[i].get_
text(),Status[i].get_text(),PublishTime[i].get_text())
```

取出的資料如果還有其他用途，可以將它儲存為 CSV 檔，語法如下：

```
with open("pm_xml.csv", "w") as f:
    story=f.write(csv_str)    # 寫入檔案
```

接著我們繼續來介紹 JSON 格式的解析方式，各位可以開啟範例檔 json_parse.py 查看完整程式碼。

JSON 模組

JSON 模組使用前同樣需要 import json，然後使用 loads() 方法將 JSON 格式的位元組（bytes）資料解碼成 Python 的 list 串列結構，使用方式如下：

```
import json
data = json.loads(s)
```

如此一來，就可以直接用 Python 的 list 串列操作方式取出資料了，例如取出第一個元素的 SiteName，只要如下表示：

```
data[0]["SiteName"]
```

因此利用 for 迴圈就能輕易取出需要的標記內容了。這個範例就不再儲存 CSV 檔，而是直接 print 輸出格式化資料。

Tips

如果想要解析的不是 JSON 字串而是 JSON 文件檔案，可以使用 load() 方法，例如：

```
with open('data.json', 'r') as f:
    data = json.load(f)
```

以下為完整的 xml_parse.py 及 json_parse.py 程式碼供各位參考。

〈Open Data 資料擷取與應用—XML 格式〉

檔名 xml_parse.py

```
01 # -*- coding: utf-8 -*-
02 """
03 OpenData 資料擷取與應用
04 XML 格式
05 """
06 od_url="http://opendata2.epa.gov.tw/AQI.xml"
07
08 import urllib.request as ur
09
10 with ur.urlopen(od_url) as response:
11     get_xml=response.read()
12
13 from bs4 import BeautifulSoup
14
15 data = BeautifulSoup(get_xml,'xml')
16 SiteName = data.find_all('SiteName')
17 County = data.find_all('County')
18 Status = data.find_all('Status')
19 pm25 = data.find_all('PM2.5')
20 PublishTime = data.find_all('PublishTime')
21
22 csv_str = ""
23 for i in range(0, len(SiteName)):
```

```
24      csv_str += "{},{},{},{},{}\n".\
25                  format(SiteName[i].get_text(),\
26                         County[i].get_text(),\
27                               pm25[i].get_text(),\
28                               Status[i].get_text(),\
29                                     PublishTime[i].get_text())
30
31  with open("pm_xml.csv", "w") as f:
32      story=f.write(csv_str)     # 寫入檔案
33
34  print(" 完成 ")
```

〈Open Data 資料擷取與應用—JSON 格式〉

檔名 json_parse.py

```
01  # -*- coding: utf-8 -*-
02  """
03  OpenData 資料擷取與應用
04  JSON 格式
05  """
06  od_json="http://opendata2.epa.gov.tw/AQI.json"
07
08  import urllib.request as ur
09  with ur.urlopen(od_json) as response:
10      s=response.read()
11
12  import json
13  data = json.loads(s)
14  csv_str=""
15  for i in range(0, len(data)):
16      csv_str += "{},{},{},{},{}\n".\
17                  format(data[i]["SiteName"],\
18                         data[i]["County"],data[i]["PM2.5"],\
19                             data[i]["Status"],
                               data[i]["PublishTime"])
20
21  print(csv_str)
```

重點回顧

1. 將一連串字元放在單引號或雙引號括起來，就是一個字串（string）。
2. 輸出字串時想要分行顯示，可以在要斷行的地方加入「\n」。
3. 要將字串指定給特定變數時，可以使用「=」指派運算子。
4. 想直接將數值資料轉為字串，可以使用內建函數 str() 來達成。
5. 字串較長時，可以利用「\」字元將過長的字串拆成兩行。
6. 利用三重單或雙引號把多行字串，固定其輸出的模式。
7. 當直譯器遇到反斜線，就知道下一個字元必須另外處理，就不會將它視為字串結尾的單引號。另外還有一些換行字元、定位字元等無法由鍵盤輸入，也可以用跳脫字元來處理。
8. 跳脫字元「\」本身還有另外一個用途，就是當程式碼太長時只要在該行末端加一個反斜線可以換行繼續撰寫。
9. 在 Python 語言中可以使用 format() 函數就可以格式化資料。
10. 想要取得字串中的字元，有三種方式：
    (1) 透過索引值（index）取某個字元
    (2) 使用切片（slice）方法取某段字串
    (3) 使用 split() 方法分割字串
11. 字串物件一旦被配置，它的內容是不可變的 (immutable)。
12. 利用 [] 運算子來擷取字串單一的字元或某個範圍的子字串，稱為「切片」。
13. split() 函數可以指定分隔符號將字串分割為子字串，並回傳子字串的串列。
14. chr() 函數可以回傳 ASCII 碼對應的字元，使用 ord() 函數可以傳回字元對應的 ASCII 碼。
15. 字串可以透過串接運算子「+」將兩個字串進行串接的工作。

16. 比較運算子也可以用來比較兩個字串間的大小。
17. Python 字串的大小比較是根據字元的 Unicode 值的大小。
18. in 和 not in 只適用於序列物件，諸如字串、串列等。
19. 函數 len() 會傳回字串的字元個數也就是字串長度，空白也會計算在內。
20. 字串是否全部是大寫或全部是小寫，可以用 isupper() 或 islower() 方法查詢。
21. 資料分析的時候常常需要計算特定字串出現的次數，Python 提供好用的 count() 方法可以使用。
22. 函數 strip() 用於去除字串首尾的字元，lstrip() 用於去除左邊的字元，rstrip() 用於去除右邊的字元。
23. 函數 replace() 可以將字串裡的特定字串替換成新的字串。
24. find() 方法用來尋找指定字元，回傳第一個出現的索引編號，同樣以索引編號來設定開始和結束的尋找範圍。
25. index() 方法用來回傳指定字元的索引值。
26. startswith() 方法用來比對前端字元，endswith() 方法則以尾端字元為主。

Chapter 5 學習評量

選擇題

(　　) 1. 關於字串的描述，下列何者有誤？
(A) 輸出字串時想要分行顯示，可以在要斷行的地方加入「\b」
(B) 利用三重單或雙引號把多行字串，固定其輸出的模式
(C) 當程式碼太長時只要在該行末端加一個反斜線可以換行繼續撰寫
(D) 字串物件一旦被配置，它的內容是不可變的。

(　　) 2. 下列哪一個函數（或方法）可以指定分隔符號將字串分割為子字串？
(A) len() 函數　(B) strip() 函數
(C) split() 函數　(D) ord() 函數。

(　　) 3. 下列哪一個函數可以回傳 ASCII 碼對應的字？
(A) ord() 函數　(B) chr() 函數
(C) strip() 函數　(D) split() 函數。

填充題

1. 將一連串字元放在單引號或雙引號括起來，就是一個_____。

2. 要將字串指定給特定變數時，可以使用_____運算子。

3. 當字串較長時，也可以利用_____字元將過長的字串拆成兩行。

4. 在 Python 語言中可以使用_____函數就可以格式化資料。

5. 利用 [] 運算子來擷取字串單一的字元或某個範圍的子字串，稱為_____。

6. Python 字串的大小比較是根據字元的_____值的大小。

7. 函數_____可以將字串裡的特定字串替換成新的字串。

簡答題

1. 讓使用者輸入一個字串，計算字串裡英文字的個數？例如：

```
輸入：cute2017#*/-
輸出：共有 4 個英文字母，字母是 cute
```

【提示】ord() 函數傳回字元對應的 ASCII 碼

小寫字母 ASCII 碼 97 ～ 122

2. 請將「ATTITUDE 反轉輸出。例如：

```
請輸入字串：ATTITUDE
原字串：ATTITUDE
反轉後：EDUTITTA
```

3. 檔案「twisters.txt」內容是英文繞口令文字檔，請撰寫一程式統計檔案內容裡的「Peter」出現了幾次？

4. 想要取得字串中的字元，有哪三種方式？

Chapter

6 函　數

模組化的概念就是採用結構化分析方式，把程式由上而下逐一分析，並將大問題逐步分解成各個較小的問題，從程式設計實作的角度來看，就是函數（function）。函數可視為一種獨立的模組。當需要某項功能程式時，只須呼叫撰寫完成的函數來執行即可。本章將會跟各位討論 Python 函數的各種應用。

學習大綱

- 定義函數
- 呼叫函數
- 函數的參數
- 引數傳遞
- 關鍵字引數
- 任意引數串列
- 函數的回傳值
- 變數的有效範圍
- 遞迴函數
- 河內塔問題
- 選擇排序法
- 氣泡排序法
- 排序函數 -sorted()
- lambda 運算式

在中大型程式的開發中，為了程式碼的可讀性及利於程式專案的規劃，通常會將程式切割成一個個功能明確的函數，而這就是一種模組化概念的充份表現。函數簡單來說就是將特定功能或經常重複使用的程式獨立出來，並且給予一個名稱來代表此段程式碼，讓主程式可以呼叫。通常撰寫程式之前會先經過分析的過程，如果有現成的函數或模組可以使用，可以省去不少程式開發的時間；如果沒有，也盡可能將程式拆成獨立功能的模組或函數，日後就可以重複叫用。前面我們已經使用過許多的函數，本章再深入介紹函數的用法。

6-1 認識函數

使用函數不僅可以省去重複撰寫相同程式碼，大幅縮短開發的時間，更有助於日後程式的除錯和維護。自訂函數則是使用者依照需求來設計的函數，這也是本章即將說明的重點所在，包括函數宣告、引數的使用、函數的主體與傳回值。以下來看看定義函數與呼叫函數的方法。

6-1-1 定義函數

函數可分為**內建函數**（built-in）與**自訂函數**（user-defined）。Python 本身就內建許多的函數，像是之前使用過的 help()、round()、len() 都是內建的函數，直接可以呼叫使用。另外還有更多用途廣泛的函數，放在標準模組庫（Standard library），或是第三方開發模組庫，使用這類函數之前都必須在程式裡先載入模組庫，就可以呼叫使用。所謂**模組**（Module）就是指特定功能的函數組合。

至於自訂函數（user-defined），需要先定義函數，然後才能呼叫函數。Python 定義函數是使用關鍵字「def」，其後空一格接函數名稱，串接一對小括號，小括號可以填入傳入函數的參數，小括號之後再加上「:」，格式如下所示：

```
def 函數名稱 ( 參數 1, 參數 2, …):
    程式指令區塊
    return 回傳值        # 有回傳值時才需要
```

函數的程式指令區塊必須縮排，參數可有可無，如果有定義參數，呼叫函數時必須連帶傳入所需的引數（arguments）。也就是說，定義函數時要有「形式參數」（Formal parameter）來準備接收資料，而呼叫函數要有「實際引數」（Actual arguments）做進行資料的傳遞工作。

- **形式參數**：定義函數時，用來接收實際引數所傳遞的資料，進入函數主體執行指令或運算。
- **實際引數**：程式中呼叫函數時，將資料傳遞給自訂函數。

當函數執行結束後，會傳回結果（return value），也就是函數的回傳值。沒有回傳值時，函數會自動回傳 None 物件，例如以下函數有回傳值：（func.py）

```
def func(a,b):
    x = a + b
    return x

print(func(1,2))
```

執行結果 >>

```
3
```

如果沒有回傳值，則會返回 None，例如下式：

```
def func(a,b):
    x = a + b
    print(x)

print(func(1,2))
```

執行結果 >>

```
3
None
```

學習小教室

各位可以用 type() 函數來查詢物件的資料型態，例如上面所傳回的 None 物件，可以使用 type() 來查詢，如下所示：

```
print( type( func(1,2) ) )
```

傳回的結果會是 <class 'NoneType'>

6-1-2 呼叫函數

宣告函數之後，程式編譯時就會產生與函數同名的物件，呼叫函數時只要使用括號「()」運算子就可以了，如下所示。

```
函數名稱 ( 引數 1, 引數 2, …)
```

位置引數

Python 函數的引數分為位置引數（Positional Argument）與關鍵字引數（Keyword Argument），位置引數就是依照參數位置傳入引數，如果函數定義了 3 個參數，就要帶入 3 個引數，缺一不可，它具有順序性，不可錯亂。

或者採用預設引數的方式，當實際引數未傳遞時，以「預設參數 = 值」做接收。如下所示：（CallFunc.py）

```
def func(a,b,c=0):
    x = a + b + c
    return x

print(func(1,2,3))     # 輸出 6
print(func(1,2))       # 輸出 3
```

上面指令 func 函數裡的參數 c 預設值為 0，因此呼叫函數時就可以只帶入 2 個引數。

呼叫函數時不想依序做一對一的引數傳遞時，關鍵字引數（Keyword Argument）就能派上用場。

關鍵字引數

關鍵字引數就是藉由關鍵字來傳入引數，只要必要的參數都有指定，關鍵字引數的位置並不一定要依照參數順序，例如：

```
def func(a,b,c):
    x = a + b + c
    return x

print(func(c=2,b=3,a=1))    # 輸出 6
```

以下的呼叫都具有相同效果：

```
func(1, 2, 3)
func(a=1, b=2 , c=3)
func(1, c=3 , b=2)
```

如果位置引數與關鍵字引數混用要特別注意下列兩點：

1. 位置引數必須在關鍵字引數之前，例如下式會顯示 SyntaxError: positional argument follows keyword argument 的錯誤訊息：

```
func(a=1, 2 , c=3)
```

2. 每個參數只能對應一個引數，例如：

```
func(1, a=2 , c=3)
```

上式第一個位置引數是傳入給參數 a，第 2 個引數又指定參數 a，就會顯示「TypeError: func() got multiple values for argument 'a'」的錯誤訊息。

任意引數串列

如果事先不知道要傳入的引數個數，可以在定義函數時在參數前面加上一個星號 (*)，表示該參數接受不定個數的引數，傳入的引數會視為一組元組（tuple）；參數前面加上 2 個星號 (**)，傳入的引數會視為一組字典（dict），有關字典的資料型態的詳細說明，可以參閱第 7 章進階資料型態的介紹。

〈呼叫函數 — 傳入不定個數的引數〉

檔名 CallFunc_01.py

```
01  def func(*num):
02      total=0
03      for n in num:
04          total += n
05      return total
06
07  print(func(1, 2))
08  print(func(1, 2, 3))
09  print(func(1, 2, 3, 4))
10
11
12  def func(**num):
13      return num
14
15  print(func(a=1, b=2, c=3))
```

執行結果 »

```
3
6
10
{'a': 1, 'b': 2, 'c': 3}
```

</> 程式解說 >>

- 第 01~05 行：如果事先不知道要傳入的引數個數，可以在定義函數時在參數前面加上一個星號 (*)，表示該參數接受不定個數的引數，傳入的引數會視為一組元組（tuple）。
- 第 12 行：參數前面加上 2 個星號 (**)，傳入的引數會視為一組字典（dict）。

6-1-3 函數的回傳值

具有回傳值的函數，在函數程式指令內可以包含一個以上的 return 指令，當程式執行到 return 指令就終止，然後將值傳回，請參考以下指令：（return.py）

```
def func(x):
    if x < 10:
        return x
    else:
        return "Over"

a = func(15)
print(a)          # 輸出 Over
print(type(a))    # 輸出 <class 'str'>
```

Python 的函數也可以一次回傳多個值，只要以逗號 (,) 分隔回傳值，例如：（return01.py）

```
def func(a,b):
    n = a + b
    x = a * b
    return n, x

num1 ,num2 = func(10, 20)
print(num1)       # 輸出 30
print(num2)       # 輸出 200
```

以下程式範例建立分帳函數（SplitBill），讓使用者輸入帳單金額及分帳人數，帳單金額須加上服務費 10% 計算出應付金額及取整數的金額。

〈分帳程式〉

檔名 SplitBill.py

```
01  # -*- coding: utf-8 -*-
02  '''
03  分帳程式
04  '''
05
06  def SplitBill():
07      bill = float(input(" 帳單金額："))
08      split = float(input(" 分帳人數："))
09      tip = 0.1  #10% 服務費
10      total = bill + (bill * tip)
11      each_total = total / split
12      each_pay = round(each_total, 0)
13      return each_total, each_pay
14
15
16  e1 ,e2 = SplitBill()
17  print(" 每人應付 {}, 應付：{}".format(e1, e2))
```

執行結果 »

```
帳單金額：5000

分帳人數：3
每人應付1833.3333333333333,應付：1833.0
```

</> 程式解說 »

- 第 06~13 行：定義自訂函數 SplitBill()，該函數有兩個回傳值為 each_total, each_pay。
- 第 16 行：變數 e1 ,e2 分別用來接收 SplitBill() 的兩個回傳值。

6-2 Python 引數傳遞的機制

以下是大多數程式語言常見的兩種參數傳遞方式：

- **傳值（Call by value）呼叫**：表示在呼叫函數時，會將引數的值一一地複製給函數的參數，在函數中對參數值作任何修改，都不會影響到原來的引數值。

- **傳址（Pass-by-reference）呼叫**：傳址呼叫表示在呼叫函數時所傳遞給函數的參數值是變數的記憶體位址，如此一來函數的引數將與函數中的接收參數共享同一塊記憶體位址，因此對參數值的變動連帶著也會影響到原來的引數值。

但是 Python 的引數傳遞是利用不可變和可變物件來運作：

- **不可變物件（Immutable Object）**（如數值、字串）傳遞引數時，接近於「傳值」。

- **可變物件（Mutable Object）**（如串列），傳遞引數時以「傳址」處理。簡單來說，如果可變物件被修改內容值，因為佔用同一位址，會連動影響函數外部的值。

以下程式範例是說明在函數內部變動字串的內容值不會影響函數外部的值，不過在函數內部修改串列的內容值，則會連動影響函數外部的值。

〈Python 的引數傳遞〉

檔名 Argument.py

```
01  def passFun(name, score):
02      name = 'Macheal'
03      print(' 函數內部修改過的名字及分數 ')
04      print('======================')
05      print(' 名字 :', name)
06      # 新增一個成績分數，會同步更動函數的的串列值
07      score.append(85)
08      print(' 分數 :', score)
09
10  name1 = 'Andy'            # 未呼叫函數前的名字設定值
11  score1 = [56, 84, 63]     # 未呼叫函數前的成績分數串列
12  print(' 函數呼叫前預設的名字及分數 ')
13  print(' 名字 :', name1)
14  print(' 分數 :', score1)
15  passFun(name1, score1)
16
17  print(' 函數內部被修改過回傳的名字及分數 ')
18  print(' 各位可以注意到名字沒變，但分數被修改了 ')
19  print(' 名字 :', name1)
20  print(' 分數 :', score1)
```

執行結果 »

```
函數呼叫前預設的名字及分數
名字: Andy
分數: [56, 84, 63]
函數內部修改過的名字及分數
======================
名字: Michael
分數: [56, 84, 63, 85]
函數內部被修改過回傳的名字及分數
各位可以注意到名字沒變,但分數被修改了
名字: Andy
分數: [56, 84, 63, 85]
```

6-3 變數的有效範圍

變數依其有效範圍分為全域變數與區域變數：

- **全域（Global）變數**：全域變數是宣告在程式區塊與函數之外，且在宣告指令以下的所有函數及程式區塊都可以使用到該變數。事實上，全域變數的使用應該相當謹慎，以免某個函數不小心給予了錯誤的值，進而影響到整個程式的邏輯，其有效範圍適用於整個檔案 (*.py)。
- **區域（Local）變數**：適用於所宣告的函數或流程控制的程式區塊，離開此範圍就會結束其生命週期。

如何判別變數的適用範圍？以第一次宣告時所在地來表示其適用範圍。接著以下面的例子來說明全域變數和區域變數的不同。

```
score = [78, 65, 84, 91]     # score 為全域變數
for item in score:
    total = 0                # 區域變數，儲存加總結果
    total += item            # 每次 total 的值都從 0 開始，無法累計
print(total)
```

score 儲存串列元素是全域變數，任何位置呼叫它皆可以。total 宣告於 for/in 迴圈，離開迴圈區塊就結束其生命週期。print(total) 時，此時 total 變數已離開迴圈，所以無法輸出加總結果。

所以上述的程式必須修正如下，如此變數 total 為全域變數時，才能累計儲存。

```
score = [78, 65, 84, 91]     # score 為全域變數
total = 0                    # 全域變數，儲存加總結果
for item in score:
    total += item            # 儲存累計值
print(total)
```

如果程式中有相同名稱的全域變數與區域變數，則會以區域變數優先，例如在函數內必須以區域變數優先，當離開函數外時，則會採用全域變數。

```
def global_local():
    num=100
    print('num=',num)

num=500
global_local()          # 輸出區域變數 100
print('num=',num)       # 輸出全域變數 500
```

但是如果要在函數內使用全域變數，則必須在函數中將該變數以 global 宣告。

```
def global_local():
    global num
    print('num=',num)   # 輸出全域變數 500
    num=100             # 全域變數值改變成 100

num=500
global_local()          # 在函數中輸出全域變數 500
print('num=',num)       # 輸出全域變數的新設定值 100
```

6-4 遞迴函數

遞迴是種很特殊的演算法，簡單來說，對程式設計師而言，「函數」（或稱副程式）不單純只是能夠被其它函數呼叫（或引用）的程式單元，在某些語言還提供了自身引用的功能，這種功用就是所謂的「遞迴」。遞迴是早期人工智慧所用的語言，如 Lisp、Prolog 幾乎都是整個語言運作的核心，當然在 Java 中也有提供這項功能，因為它們的繫結時間可以延遲至執行時才動態決定。

「何時才是使用遞迴的最好時機？」，是不是遞迴只能解決少數問題？事實上，任何可以用選擇結構和重複結構來編寫的程式碼，都可以用遞迴來表示和編寫。

6-4-1 遞迴的定義

談到遞迴的定義，我們可以正式這樣形容，假如一個函數或副程式，是由自身所定義或呼叫的，就稱為遞迴（Recursion），它至少要定義 2 種條件：

- 一個可以反覆執行的遞迴過程
- 一個跳出執行過程的出口

例如我們知道階乘函數是數學上很有名的函數，對遞迴式而言，也可以看成是很典型的範例，我們一般以符號"！"來代表階乘。如 4 階乘可寫為 4!，n! 可以寫成：

```
n!=n×(n-1)*(n-2)…*1
```

各位可以一步步分解它的運算過程，觀察出一定的規律性：

```
5! = (5 * 4!)
   = 5 * (4 * 3!)
   = 5 * 4 * (3 * 2!)
   = 5 * 4 * 3 * (2 * 1)
   = 5 * 4 * (3 * 2)
   = 5 * (4 * 6)
   = (5 * 24)
   = 120
```

至於 Python 的遞迴函數演算法可以寫成如下：

```
def factorial(i):
    if i==0:
        return 1
    else:
        ans=i * factorial(i-1)     # 反覆執行的遞迴過程
    return ans
```

其實遞迴可以被 while 或 for 取代，例如下例就是利用 for 迴圈設計一個計算 0!~n!。

〈利用 for 迴圈計算 0!~n!〉

檔名 factorial_for.py

```
01  # 以 for 迴圈計算 n!
02  sum = 1
03  n=int(input(' 請輸入 n='))
04  for i in range(0,n+1):
05      for j in range(i,0,-1):
06          sum *= j     # sum=sum*j
07      print('%d!=%3d' %(i,sum))
08      sum=1
```

執行結果 »

```
請輸入n=6
0!=  1
1!=  1
2!=  2
3!=  6
4!= 24
5!=120
6!=720
```

此外，遞迴因為呼叫對象的不同，可以區分為以下兩種：

- **直接遞迴（Direct Recursion）**：指遞迴函數中，允許直接呼叫該函數本身。如下例：

```
def Fun(...):
  .
  .
    if ... :
        Fun(...)
    .
    .
}
```

- **間接遞迴（Indirect Recursion）**：指遞迴函數中，如果呼叫其他遞迴函數，再從其他遞迴函數呼叫回原來的遞迴函數。

```
def Fun1(...):          def Fun2(...):
          .                       .
          .                       .
    if ... :                if ... :
        Fun2(...)               Fun1(...)
              .                       .
              .                       .
```

Tips

「尾歸遞迴」（Tail Recursion）就是程式的最後一個指令為遞迴呼叫，因為每次呼叫後，再回到前一次呼叫的第一行指令就是 return，所以不需要再進行任何計算工作。

6-4-2 費伯那序列

以上遞迴應用的介紹是利用階乘函數的範例來說明遞迴式的運作。相信各位應該不會再對遞迴有陌生的感覺了吧！我們再來看一個很有名氣的費伯那序列（Fibonacci Polynomial），首先看看費伯那序列的基本定義：

$$F_n = \begin{cases} 0 & n=0 \\ 1 & n=1 \\ F_{n-1}+F_{n-2} & n=2,3,4,5,6\cdots（n\ 為正數） \end{cases}$$

簡單來說，就是一序列的第零項是 0、第一項是 1，其它每一個序列中項目的值是由其本身前面兩項的值相加所得。從費伯那序列的定義，也可以嘗試把它轉成遞迴的形式：

```
def fib(n):                    # 定義函數 fib()
    if n==0 :
        return 0               # 如果 n=0 則傳回 0
    elif n==1 or n==2:
        return 1
    else:                      # 否則傳回 fib(n-1)+fib(n-2)
        return (fib(n-1)+fib(n-2))
```

〈n 項費伯那序列的遞迴程式〉

檔名 fib.py

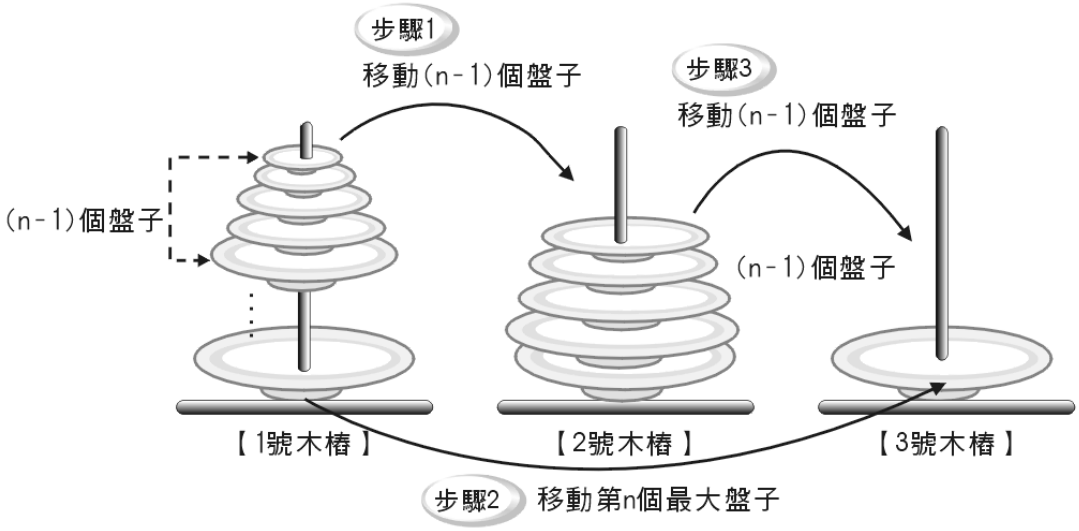
```
01  def fib(n):                # 定義函數 fib()
02      if n==0 :
03          return 0           # 如果 n=0 則傳回 0
04      elif n==1 or n==2:
05          return 1
06      else:                  # 否則傳回 fib(n-1)+fib(n-2)
07          return (fib(n-1)+fib(n-2))
08
09  n=int(input(' 請輸入所要計算第幾個費式數列 :'))
10  for i in range(n+1):   # 計算前 n 個費氏數列
11      print('fib(%d)=%d' %(i,fib(i)))
```

執行結果 »

```
請輸入所要計算第幾個費式數列:10
fib(0)=0
fib(1)=1
fib(2)=1
fib(3)=2
fib(4)=3
fib(5)=5
fib(6)=8
fib(7)=13
fib(8)=21
fib(9)=34
fib(10)=55
```

6-4-3 河內塔問題

法國數學家 Lucas 在 1883 年介紹了一個十分經典的河內塔 (Tower of Hanoil) 智力遊戲，是典型使用遞迴式與堆疊觀念來解決問題的範例，內容是說在古印度神廟，廟中有三根木樁，天神希望和尚們把某些數量大小不同的圓盤，由第一個木樁全部移動到第三個木樁。

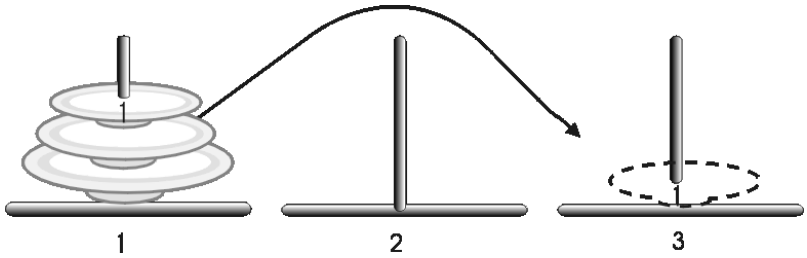


更精確來說，河內塔問題可以這樣形容：假設有 A、B、C 三個木樁和 n 個大小均不相同的套環（Disc），由小到大編號為 1,2,3…n，編號越大直徑越大。開始的時候，n 個套環境套在 A 木樁上，現在希望能找到將 A 木樁上的套環藉著 B 木樁當中間橋樑，全部移到 C 木樁上最少次數的方法。不過在搬動時還必須遵守下列規則：

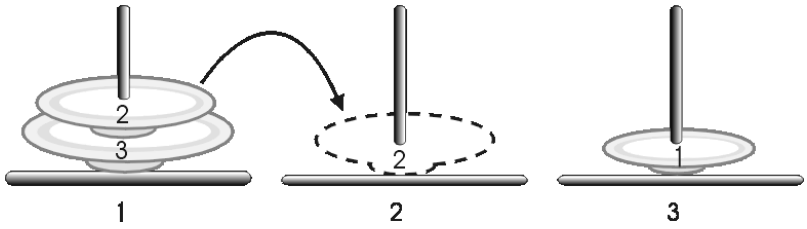
1. 直徑較小的套環永遠置於直徑較大的套環上。
2. 套環可任意地由任何一個木樁移到其他的木樁上。
3. 每一次僅能移動一個套環，而且只能從最上面的套環開始移動。

現在我們考慮 n=3 的狀況，以圖示方式為各位示範處理河內塔問題的步驟：

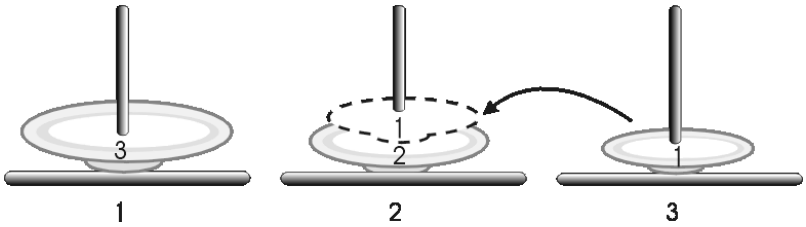
Step 01 將套環從 1 號木樁移動到 3 號木樁



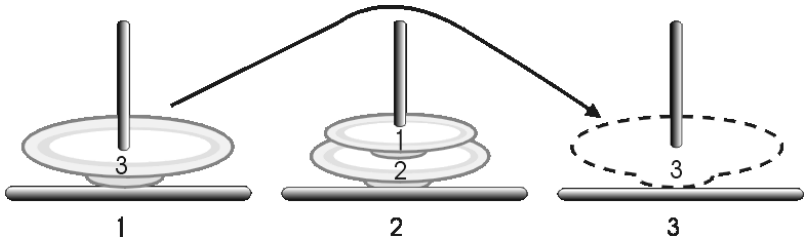
Step 02 將套環從 1 號木樁移動到 2 號木樁



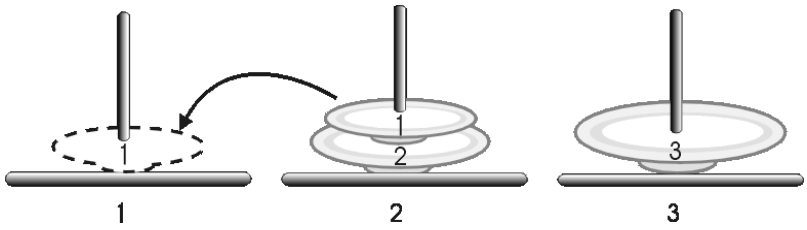
Step 03 將套環從 3 號木樁移動到 2 號木樁



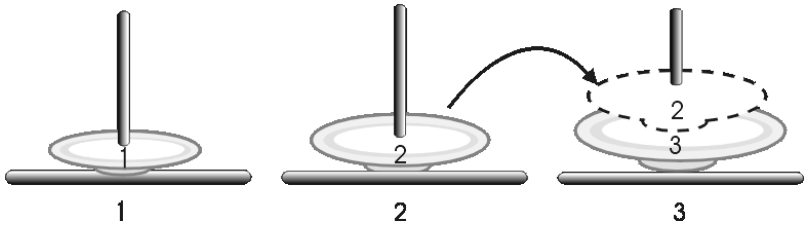
Step 04 將套環從 1 號木樁移動到 3 號木樁



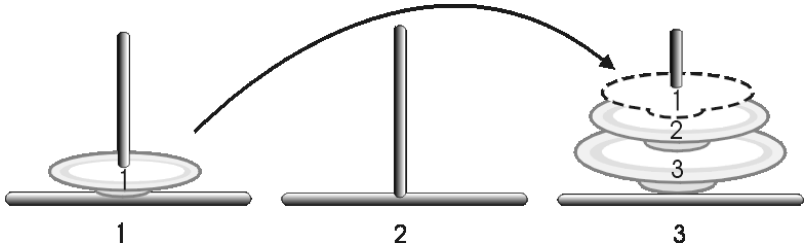
Step 05 將套環從 2 號木樁移動到 1 號木樁



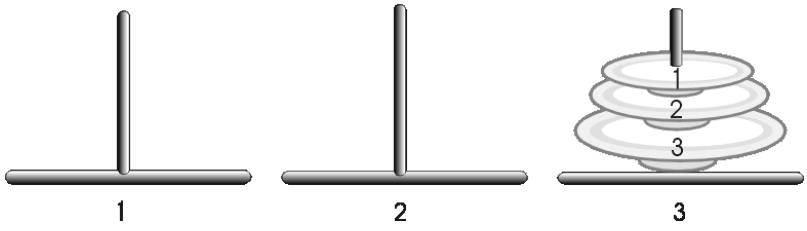
Step 06 將套環從 2 號木樁移動到 3 號木樁



Step 07 將套環從 1 號木樁移動到 3 號木樁，就完成了



完成



結論 移動了 2^3-1=7 次，盤子移動的次序為 1,2,1,3,1,2,1（盤子次序）

步驟為 1 → 3，1 → 2，3 → 2，1 → 3，2 → 1，2 → 3，1 → 3（木樁次序）

當有 4 個盤子時，我們實際操作後（在此不作圖說明），盤子移動的次序為 121312141213121，而移動木樁的順序為 1 → 2，1 → 3，2 → 3，1 → 2，3 → 1，3 → 2，1 → 2，1 → 3，2 → 3，2 → 1，3 → 1，2 → 3，1 → 2，1 → 3，2 → 3，而移動次數為 2^4-1=15。

當 n 不大時，各位可以逐步用圖示解決，但 n 的值較大時，那可就十分傷腦筋了。事實上，我們可以得到一個結論，例如當有 n 個盤子時，可將河內塔問題歸納成三個步驟：

步驟 1：將 n-1 個盤子，從木樁 1 移動到木樁 2。

步驟 2：將第 n 個最大盤子，從木樁 1 移動到木樁 3。

步驟 3：將 n-1 個盤子，從木樁 2 移動到木樁 3。

由上圖中，各位應該發現河內塔問題是非常適合以遞迴式與堆疊來解決。因為它滿足了遞迴的兩大特性：①有反覆執行的過程，②有停止的出口。以下則以遞迴式來表示河內塔遞迴函數演算法：

```
def hanoi(n, p1, p2, p3):
    if n==1:   # 遞迴出口
        print('套環從 %d 移到 %d' %(p1, p3))
    else:
        hanoi(n-1, p1, p3, p2)
        print('套環從 %d 移到 %d' %(p1, p3))
        hanoi(n-1, p2, p1, p3)
```

〈以遞迴式來實作河內塔〉

檔名 hanoi.py

```
01  def hanoi(n, p1, p2, p3):
02      if n==1:     # 遞迴出口
03          print('套環從 %d 移到 %d' %(p1, p3))
04      else:
05          hanoi(n-1, p1, p3, p2)
06          print('套環從 %d 移到 %d' %(p1, p3))
07          hanoi(n-1, p2, p1, p3)
08
09  j=int(input('請輸入所移動套環數量：'))
10  hanoi(j,1, 2, 3)
```

執行結果 »

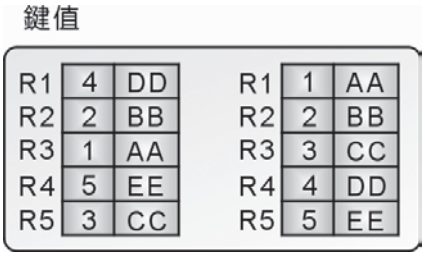
```
請輸入所移動套環數量：4
套環從 1 移到 2
套環從 1 移到 3
套環從 2 移到 3
套環從 1 移到 2
套環從 3 移到 1
套環從 3 移到 2
套環從 1 移到 2
套環從 1 移到 3
套環從 2 移到 3
套環從 2 移到 1
套環從 3 移到 1
套環從 2 移到 3
套環從 1 移到 2
套環從 1 移到 3
套環從 2 移到 3
```

6-5 認識排序

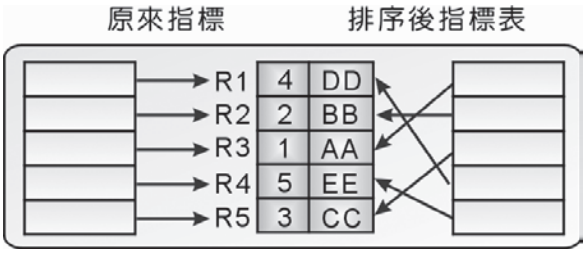
排序（Sorting）可以形容是最常使用到的一種演算法，目的是將一串不規則的數值資料依照遞增或是遞減的方式重新編排。隨大數據和人工智慧技術（Artificial Intelligence, AI）的普及和應用，排序成為非常重要的工具之一，甚至在年輕人愛玩的遊戲程式設計中，就經常會利用到排序的技巧。

所謂「排序」，就是將一群資料按照某一個特定規則重新排列，使其具有遞增或遞減的次序關係。按照特定規則，用以排序的依據，我們稱為鍵（Key），它所含的值就稱為「鍵值」，通常鍵值資料型態有數值型態、中文字串型態及非中文字串型態三種。

當手邊有一群數據資料，如何透過程式將它整理成的遞增或遞減的線性關係。舉一個簡單的例子，輸入 10 個整數，由小排到大，就是最基本的排序。在排序的過程中，電腦中資料的移動方式可分為「直接移動」及「邏輯移動」兩種。「直接移動」是直接交換儲存資料的位置，而「邏輯移動」並不會移動資料儲存位置，僅改變指向這些資料的輔助指標的值。



直接移動排序



邏輯移動排序

兩者間優劣在於直接移動會浪費許多時間進行資料的更動，而邏輯移動只要改變輔助指標指向的位置就能輕易達到排序的目的，例如在資料庫中可在報表中可顯示多筆記錄，也可以針對這些欄位的特性來分組並進行排序與彙總，這就是屬於邏輯移動，而不是真正移動實際改變檔案中的位置。基本上，資料在經過排序後，會有下列三點好處：

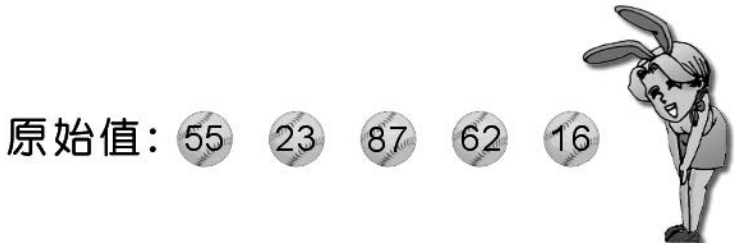
1. 資料較容易閱讀。
2. 資料較利於統計及整理。
3. 可大幅減少資料搜尋的時間。

處理排序問題有很多種演算法，其中**選擇排序**（Selection sort）與**氣泡排序**（Bubble sort）是最適合入門的演算法，在這章節將介紹這兩種演算法，Python 本身也有排序的函數，本章也會一併介紹。

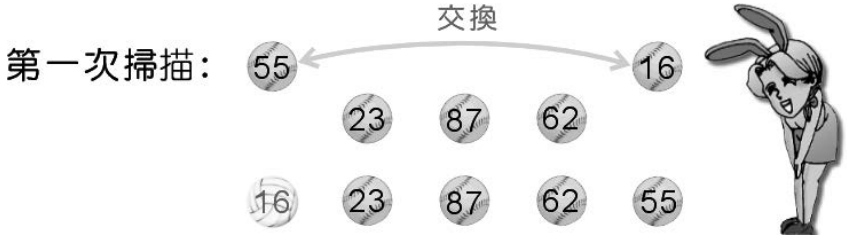
6-5-1 選擇排序法

選擇排序法（Selection Sort）的概念就是反覆從未排序的數列中取出最小的元素，加入到另一個的數列，結果即為已排序的數列。選擇排序法可使用兩種方式排序，一為在所有的資料中，當由大至小排序，則將最大值放入第一位置；若由小至大排序時，則將最大值放入位置末端。例如一開始在所有的資料中挑選一個最小項放在第一個位置（假設是由小到大），再從第二筆開始挑選一個最小項放在第 2 個位置，依樣重複，直到完成排序為止。

以下我們仍然利用 55、23、87、62、16 數列的由小到大排序過程，來說明選擇排序法的演算流程：

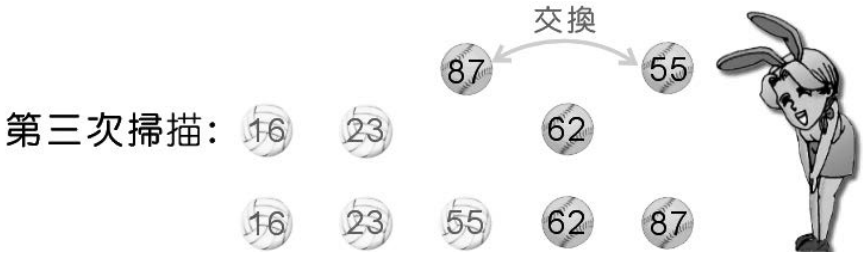


Step 01 首先找到此數列中最小值後與第一個元素交換：



Step 02 從第二個值找起，找到此數列中（不包含第一個）的最小值，再和第二個值交換：

Step 03 從第三個值找起，找到此數列中 (不包含第一、二個) 的最小值，再和第三個值交換：



Step 04 從第四個值找起，找到此數列中 (不包含第一、二、三個) 的最小值，再和第四個值交換，則此排序完成：

選擇排序法比較次數固定為 n*(n-1)/2 次，以下範例利用 Python 實作選擇排序，程式碼如下：

〈選擇排序法〉

檔名 selection_sort.py

```
# -*- coding: utf-8 -*-

def selectionSort(L):
    N = len(L)
    cc = 0
    x=0
    for i in range(N-1):
        minL = i
        for j in range(i+1, N):    # 找出最小值
            x+=1
            if L[minL] > L[j]:
                minL = j

        # 把最小值跟第 i 個做交換
        L[minL], L[i] = L[i], L[minL]
        cc += 1
        print(" 第 {} 次排序結果：{}".format(cc,L))
    return L,x

a = [10, 3, 12, 20, 16]            # 排序的資料
print(" 排序前： {}".format(a))
L,x = selectionSort(a)
print(" 排序後： {}".format(L))
print(" 比較次數：{}".format(x))
```

執行結果 »

```
排序前: [10, 3, 12, 20, 16]
第1次排序結果：[3, 10, 12, 20, 16]
第2次排序結果：[3, 10, 12, 20, 16]
第3次排序結果：[3, 10, 12, 20, 16]
第4次排序結果：[3, 10, 12, 16, 20]
排序後: [3, 10, 12, 16, 20]
比較次數：10
```

</> 程式解說 >>

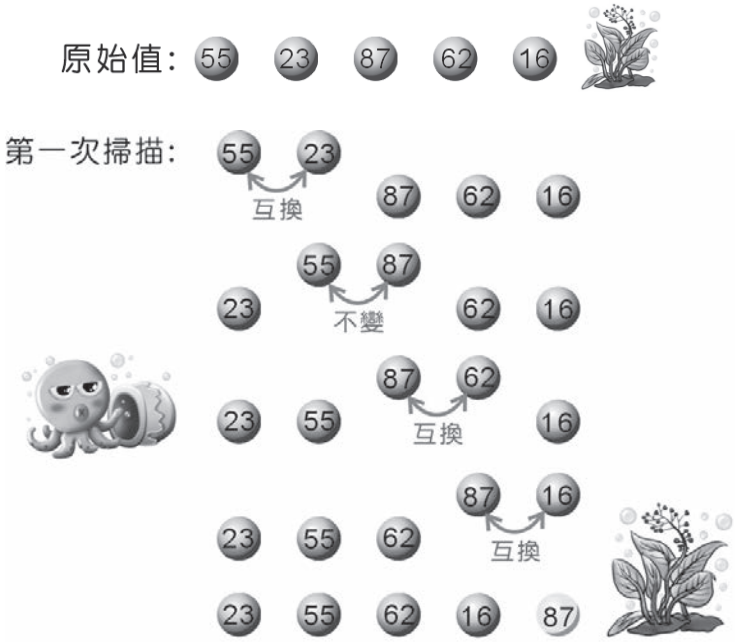
- 第 21 行：輸出尚未排序前的數列順序。
- 第 03~18 行：定義選擇排序法函數，其中變數 x 是用來累計比較次數，變數 cc 是用來記錄第幾回合的排序結果。主要原理是反覆從未排序的數列中取出最小的元素，加入到另一個的數列，結果即為已排序的數列。
- 第 23 行：輸出排序後的數列順序。

6-5-2 氣泡排序法

氣泡排序法又稱為交換排序法，是由觀察水中氣泡變化構思而成，原理是由第一個元素開始，比較相鄰元素大小，若大小順序有誤，則對調後再進行下一個元素的比較，就彷彿氣泡逐漸由水底逐漸冒升到水面上一樣。如此掃描過一次之後就可確保最後一個元素是位於正確的順序。接著再逐步進行第二次掃描，直到完成所有元素的排序關係為止。

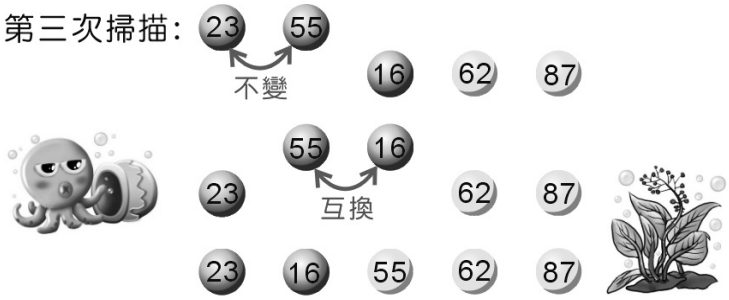
以下排序我們利用 55、23、87、62、16 的排序過程，各位可以清楚知道氣泡排序法的演算流程：

由小到大排序：

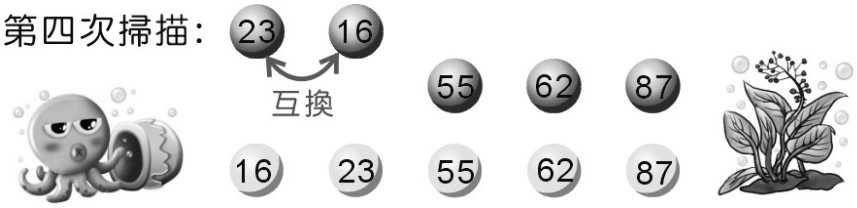


第一次掃描會先拿第一個元素 55 和第二個元素 23 作比較，如果第二個元素小於第一個元素，則作交換的動作。接著拿 55 和 87 作比較，就這樣一直比較並交換，到第 4 次比較完後即可確定最大值在陣列的最後面。

第二次掃描亦從頭比較起，但因為最後一個元素在第一次掃描就已確定是陣列最大值，故只需比較 3 次即可把剩餘陣列元素的最大值排到剩餘陣列的最後面。



第三次掃描完，完成三個值的排序



第四次掃描完，即可完成所有排序

由此可知 5 個元素的氣泡排序法必須執行 5-1 次掃描，第一次掃描需比較 5-1 次，共比較 4+3+2+1=10 次。

〈氣泡排序法〉

檔名 bubble_sort.py

```
01 # -*- coding: utf-8 -*-
02 '''
03 氣泡排序法
04 '''
05
06 def bubble_sort(L):
07     N = len(L)
08     cc=0
09     x=0
10     for i in range(N-1):
11         for j in range(1, N - i):    # 從 1 比較到倒數 n-i
12             x+=1
13             print("{},{}".format(L[j - 1],L[j]))
14             if L[j - 1] > L[j]:
15                 L[j - 1], L[j] = L[j], L[j - 1]
16         cc+=1
17         print(" 第 {} 次排序結果：{}".format(cc,L))
18     return L,x
19
20
21 a = [10, 3, 12, 20, 16]              # 排序的資料
22 print(" 排序前： {}".format(a))
23 L,x = bubble_sort(a)
24 print(" 排序後： {}".format(L))
25 print(" 比較次數：{}".format(x))
```

執行結果 >>

```
排序前: [10, 3, 12, 20, 16]
10,3
10,12
12,20
20,16
第1次排序結果：[3, 10, 12, 16, 20]
3,10
10,12
12,16
第2次排序結果：[3, 10, 12, 16, 20]
3,10
10,12
第3次排序結果：[3, 10, 12, 16, 20]
3,10
第4次排序結果：[3, 10, 12, 16, 20]
排序後: [3, 10, 12, 16, 20]
比較次數：10
```

</> 程式解說 >>

- 第 06~18 行：定義氣泡排序法函數。
- 第 11 行：從 1 比較到倒數 N-i。
- 第 18 行：回傳排序後的串列內容及比較次數。

上面介紹的選擇排序與氣泡排序都是基礎的排序演算法，主要是用來建立排序的觀念，而不是拿來實際應用，程式語言通常都會內建排序函數，輕鬆就能解決排序問題。Python 內建 sorted 函數可供排序使用，以下就來介紹它的用法。

6-5-3 排序函數—sorted()

sorted 函數用法很直覺，格式如下：

```
sorted(iterable, key=None, reverse=False)
```

第一個參數帶入的是要排序的物件，只要是可疊代的物件都可以排序，預設是由小排到大，如果將 reverse 參數設為 True，就會反轉排列的順序，變成由大排到小，例如：

```
a = [5, 2, 3, 1, 4]
print( sorted(a) )
print( sorted(a,reverse=True) )
```

執行結果>>

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

sorted 函數與 sort () 方法都是排序，兩者功能大同小異，都有 reverse 與 key 參數，差別在於 sort() 方法只支援 list 串列資料進行排序，要注意的是 sort() 方法沒有回傳值，會直接排序序列的內容，舉例來看：

```
a = [5, 2, 3, 1, 4]
a.sort()
print(a)  #[1, 2, 3, 4, 5]
```

上面指令是用 sort() 方法排序，排序之後原本的 a 物件順序也變了，以下使用 sorted 函數，排序過後產生新的物件，原本的 a 物件並不會改變：

```
a = [5, 2, 3, 1, 4]
print( sorted(a) )    # [1, 2, 3, 4, 5]
print(a)  # [5, 2, 3, 1, 4]
```

參數 key 接受函數或方法當作參數，在排序之前會自動對每個元素執行一次 key 所指定的函數，例如我們想將以下字串切割之後按照英文字母排序，就可以用 str.lower 方法先將字串轉成小寫，例如：（sorted_key.py）

```
x = "Every thing is gonna be alright"
print( sorted(x.split(), key=str.lower) )
```

執行結果 >>

```
['alright', 'be', 'Every', 'gonna', 'is', 'thing']
```

如果上例沒有轉成小寫的話，排序時大寫會在小寫前面，執行結果就會變成這樣：

執行結果 >>

```
['Every', 'alright', 'be', 'gonna', 'is', 'thing']
```

透過 key 的特性就可以做一些進階排序，例如有一個 dict 字典物件的資料是學生的姓名與分數，利用 key 就可以很輕易選擇用姓名或分數排序，請看以下範例：

〈分數排序〉

檔名 sorted_key1.py

```
01  student = {'Judy': 90, 'Candy':46, 'Andy':69}    # 姓名：分數
02
03  def x(d):
04      print("**"+str(d[1]))
05      return d[1]
06
07  print(sorted(student.items(), key = x))
```

執行結果 >>

```
**90
**46
**69
[('Candy', 46), ('Andy', 69), ('Judy', 90)]
```

這個範例可以很清楚看到在排序之前每個元素都執行一次 x 函數，並傳回分數 (d[1])，然後 sored 函數再將分數進行排序。上面範例可以改寫為 lambda 傳入匿名函數，程式就更簡潔了，程式如下所示：（sorted_key2.py）

```
print(sorted(student.items(), key = lambda x:x[1]))
```

6-6 lambda 運算式

lambda 運算式，又稱 lambda 函數，它沒有函數名稱，lambda() 函數只會有一行指令。語法如下：

```
lambda 參數串列 , … : 運算式
```

其中運算式之前的冒號「:」不能省略，運算式不能使用 return 指令。自訂函數與 lambda() 有何不同？先以一個簡例做解說。

```
def result(x, y):                # 自訂函數
    return x+y
result = lambda x, y : x + y     #lambda() 函數
```

請注意，自訂函數時，函數名稱 result，可作為呼叫 lambda() 函數的變數名稱。定義函數時，函數主體有多行指令；但是 lambda() 函數只能有一行運算式。另外自訂函數有名稱，lambda() 函數無名稱，必須指定一個變數來儲存運算結果，再以變數名 result 來呼叫 lambda() 函數，依其定義傳入參數。在 result () 函數中以 return 指令回傳；lambda() 的運算結果由變數 result 儲存。

下面就是以 lambda() 函數先定義再呼叫指定的變數 result。

```
result = lambda x, y: x+y        # 表示 lambda 有二個參數
result(4, 7)                     # 傳入兩個數值讓 lambda() 函數做運算
```

6-7 上機實作演練－美輪美奐的金字塔圖形

本節將利用一個範例來複習前面所介紹的內容，範例主題是輸出金字塔圖形。

6-7-1 程式範例描述

這個演練題目是要輸出金字塔圖形，題目要求如下：

1. 讓使用者輸入金字塔層數 h 以及要構成金字塔的符號 s，輸出金字塔圖案。
2. 金字塔每一層左側必須列出層數編號。
3. 執行完成之後詢問使用者要離開或繼續，待使用者輸入「x」鍵時顯示「Goodbye!! 」文字，或按任意鍵重複輸出金字塔圖形。

▶ 輸入說明

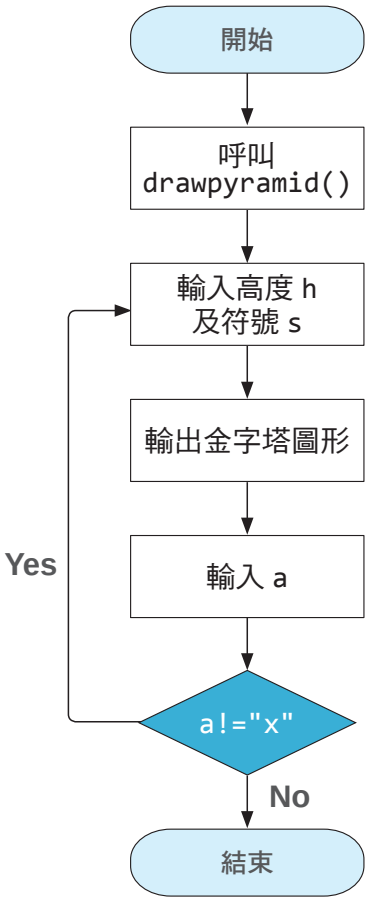
金字塔層數 h，格式為數字 1~10，以 h=8 為例。

金字塔符號 s，格式為任意一個字元，以 s="*" 為例。

▶ 輸出範例

```
1          *
2         ***
3        *****
4       *******
5      *********
6     ***********
7    *************
8   ***************
```

▶ 流程圖



6-7-2 程式碼說明

這個練習可以利用 format() 輸出，只要計算出每一層放置的符號數量，將符號置中，一層層輸出就會呈現金字塔的圖形。我們先試著找出演算法，當高度 h 輸入 1 時，列印 1 個符號；輸入 2 時，列印 3 個符號；輸入 3 時，列印 5 個符號……以此類推，所以只要將每一層高度 *2 再減 1，就是需要的符號數量了，請參考下表：

集合運算	範例	說明
1 層	(1*2)-1	1 個
2 層	(2*2)-1	3 個
3 層	(3*2)-1	5 個
4 層	(4*2)-1	7 個
n 層	(n*2)-1	……

各位可以發現每一層的輸出是具有規則的，而且重複執行 n 次就可以完成輸出，這種具有規則而且重複執行的流程非常適合使用流程控制的「迴圈」來完成。

在這個範例，使用了「for」迴圈指令，使用方式如下：

```
for  變數 in 序列：
    程式指令
```

序列可以是 list、str、tuple 以及 range 等型態，這個範例中使用的是 range 序列，它是整數序列，用法如下：

```
range([start], stop[, step])
```

參數 start 是起始值，可省略，表示從 0 開始；stop 是終止值，產生的序列不包含終止值本身；step 是增減值，可省略，表示遞增 1，例如 range(5)，表示「0, 1,2,3,4」序列；range(5, 10)，表示「5,6,7,8,9」序列。例如範例中使用的 for 迴圈語法如下：

```
for n in range(1,h+1):
    str="{0}{1:^20}"
    print(str.format(n,s*(n*2-1)))
```

變數 n 是 for 迴圈需要用到的變數，range 序列從 1 開始，直到 h+1 終止，沒有指定增減值，表示每次執行 n 加 1。這一段的完整程式如下：

```
h = int( input(" 請輸入您要顯示的金字塔層數 (1~10):") )
s = input(" 請輸入要顯示的符號 :")

for n in range(1,h+1):
    str="{0}{1:^20}"
    print(str.format(n,s*(n*2-1)))
```

這裡利用 format() 指定輸出格式，括號內帶入兩個引數，分別是 n 以及 s*(n*2-1)，上述程式執行之後就會輸出金字塔圖形並於左側列出層數了。

題目還沒完成！還需要讓使用者決定是否結束程式，因此需要讓程式可以重複上述的程式碼，我們可以將上述程式碼寫成函數（function）。函數內的程式指令必須縮排，而且同一區塊的縮排距離必須相同。例如範例中所定義的函數名稱為 def drawpyramid，函數內包含了 for 迴圈與 if 判斷式，都有各自的程式區塊範圍，撰寫程式時，必須要利用縮排清楚定義出各自的程式區塊，請參考以下示意圖，雙向箭頭表示縮排距離：

```
def drawpyramid():
    h = int( input("請輸入您要顯示的金字塔層數(1~10):") )
    s = input("請輸入要顯示的符號:")

    for n in range(1,h+1):
        str="{0}{1:^20}"
        print(str.format(n,s*(n*2-1)))

    a=input("按x鍵離開，按任意鍵繼續.")
    if a != "x":
        drawpyramid()
    else:
        print("Goodbye!!")
```

drawpyramid 函數區塊

for loop 區塊

if 區塊

else 區塊

drawpyramid 函數執行完繪製金字塔的程序之後，就會提示使用者「按 x 鍵離開，按任意鍵繼續 .」，只要判斷使用者輸入的值是不是「x」，如果是就顯示「Goodbye!」，否則就再次呼叫 drawpyramid() 函數。

我們可以利用 if...else 條件式來判斷使用者輸入的值，if...else 條件式的語法如下：

```
if 條件式 1：
    如果條件式 1 成立，就執行這一個區塊的指令
elif 條件式 2：
    如果條件式 2 成立，就執行這一個區塊的指令
else：
    如果條件不成立，就執行這一個區塊的指令
```

條件式經常使用等於 (==) 或不等於 (!=) 來判斷，例如範例中判斷使用者輸入的值 a 是不是等於「x」，可以這樣表示：

```
if a != "x":
    drawpyramid()
else:
    print("Goodbye!!")
```

說明到此，相信各位應該清楚了解範例執行的流程以及程式碼了。以下列出完整程式碼供各位參考。

〈輸出金字塔圖形〉

檔名 Review_pyramid.py

```
01  # -*- coding: utf-8 -*-
02  """
03  程式名稱：輸出金字塔圖形
04  """
05
06  def drawpyramid():        # 定義 drawpyramid 函數
07      h = int( input(" 請輸入您要顯示的金字塔層數 (1~10):") )
08      s = input(" 請輸入要顯示的符號 :")
09
10      for n in range(1,h+1):
11          str="{0}{1:^20}"
12          print(str.format(n,s*(n*2-1)))
13
14      a=input(" 按 x 鍵離開，按任意鍵繼續 .")
15      if a != "x":
16          drawpyramid()     # 呼叫 drawpyramid 函數
17      else:
18          print("Goodbye!!")
19
20
21  drawpyramid()             # 呼叫 drawpyramid 函數
```

執行結果 »

```
請輸入您要顯示的金字塔層數(1~10):8

請輸入要顯示的符號:$
1          $
2         $$$
3        $$$$$
4       $$$$$$$
5      $$$$$$$$$
6     $$$$$$$$$$$
7    $$$$$$$$$$$$$
8   $$$$$$$$$$$$$$$

按x鍵離開，按任意鍵繼續.x
Goodbye!!
```

重點回顧

1. 將特定功能或經常重複使用的程式獨立出來撰寫成一個副程式，讓主程式可以呼叫，即所謂的「函數」（function）。
2. 函數可分為內建函數（built-in）與自訂函數（user-defined）。
3. 函數的程式指令區塊必須縮排，函數也可以無參數，如果有定義參數，呼叫函數時必須連帶傳入所需的引數。
4. 定義函數時要有「形式參數」（Formal parameter）來準備接收資料，而呼叫函數要有「實際引數」（Actual arguments）做進行資料的傳遞工作。
5. Python 函數的引數分為位置引數（Positional Argument）與關鍵字引數（Keyword Argument）。
6. 關鍵字引數就是藉由關鍵字來傳入引數，只要必要的參數都有指定，關鍵字引數的位置並不一定要依照參數順序。
7. 如果事先不知道要傳入的引數個數，可以在定義函數時在參數前面加上一個星號 (*)，表示該參數接受不定個數的引數。
8. Python 的參數以不可變和可變物件來運作：
   (1) 不可變物件（Immutable Object）傳遞引數時，接近於「傳值」。
   (2) 可變物件（Mutable Object），傳遞引數時以「傳址」處理。
9. Python 的函數也可以有多個回傳值，只要以逗號 (,) 分隔回傳值。
10. 變數依其有效範圍分為全域變數與區域變數。
11. 如果要在函數內使用全域變數，則必須在函數中將該變數以 global 宣告。
12. 一個函數或副程式，是由自身所定義或呼叫的，稱為遞迴（Recursion）。
13. 遞迴至少要定義 2 種條件，包括一個可以反覆執行的遞迴過程，與一個跳出執行過程的出口。
14. 資料在經過排序後，有下列優點：資料更容易閱讀、資料方便統計及整理、大幅減少資料搜尋的時間。
15. Python 內建 sorted 函數可供排序使用，第一個參數帶入的是要排序的物件，只要是可疊代的物件都可以排序，預設是由小排到大。
16. lambda 運算式，又稱 lambda 函數，它沒有函數名稱，只會有一行指令。
17. 自訂函數有名稱，lambda() 函數無名稱，必須指定一個變數來儲存運算結果。

Chapter 6

學習評量

選擇題

(　　) 1. 關於函數的說明下列何者有誤？
(A) 可以省去重複撰寫相同程式碼
(B) 可以大幅縮短開發的時間
(C) 有助於日後程式的除錯和維護
(D) 要使用函數的唯一管道就是自訂函數。

(　　) 2. 關於定義函數的說明下列何者有誤？
(A) Python 定義函數是使用關鍵字「def」
(B) 函數的程式指令區塊必須縮排
(C) 定義函數時一定要傳入參數
(D) 沒有回傳值時，函數會自動回傳 None 物件。

(　　) 3. 關於定義函數的說明下列何者有誤？
(A) 呼叫函數時使用括號「()」運算子
(B) 引數分為位置引數與關鍵字引數
(C) 無法採用預設引數的方式
(D) 關鍵字引數的位置並不一定要依照參數順序。

(　　) 4. 關於函數回傳值的說明下列何者有誤？
(A) 當程式執行到 return 指令就終止，然後將值傳回
(B) 可以包含一個以上的 return 指令
(C) 多個回傳值，只要以逗號 (,) 分隔回傳值
(D) 定義函數時一定要有回傳值。

(　　) 5. 關於函數與參數傳遞的說明下列何者有誤？
(A) 可變物件（Mutable Object）（如串列），傳遞引數時以「傳址」處理
(B) 區域（Local）變數適用於所宣告的函數或流程控制的程式區塊
(C) 一個函數或副程式，是由自身所定義或呼叫的，就稱為遞迴
(D) 河內塔問題使用遞迴式與佇列觀念來解決問題。

填充題

1. 函數可分為_____與_____。

2. 定義函數時要有_____來準備接收資料，而呼叫函數要有_____做進行資料的傳遞工作。

3. Python 函數的引數分為_____引數與_____引數。

4. Python 的不可變物件（如數值、字串）傳遞引數時，接近於_____呼叫。

5. 如果要在函數內使用全域變數，則必須在函數中將該變數以_____宣告。

簡答題

1. 試簡述 sorted 函數與 sort () 方法兩者間的異同？

2. 請問 lambda 函數與一般自訂函數的不同。

3. 請問使用選擇排序法，將數列資料 (10、5、25、30、15) 由大到小依序排列，共需進行幾次比較？

4. 請撰寫一儲蓄存款試算程式，年利率預設為 2%，以複利計，讓使用者可以輸入本金與存款期間(年)，計算到期後的本利和。試算結果可參考下表：

本金	存款期間(年)	本金和
15000	2	15612
15000	3	15927
30000	2	31223
30000	3	31854

【提示】本利和 = 本金 * (1 + 月利率) ^ 存款期數 (^ 表示次方)

5. 請說明在函數中提到的「形式參數」（Formal parameter）與「實際引數」（Actual arguments）兩者間的功能差異。

6. Python 函數的引數可分為哪兩種，試比較說明之。

7. 請簡單說明 Python 的引數傳遞的機制。

8. 變數依其有效範圍分為全域變數與區域變數，兩者間的差異為何？

9. 什麼是遞迴？它的定義條件為何？

Chapter

7 進階資料型態

在本書前面章節中，相信各位已經學會了 Python 的基本資料型態及基礎語法。本章將會跟各位繼續討論 Python 的特殊資料型態的內容與應用，包括 tuple 元組、list 串列、dict 字典、集合 set 等複合式資料型態的觀念。

學習大綱

- 容器資料型態的比較
- 常用串列內建函數
- 串列常用方法
- 二維及多維串列
- tuple 序對 (或稱元組)
- 序對運算子
- Unpacking（拆解）與 Swap（交換）
- 字典的基礎操作
- 適用字典的運算子
- 適用字典的處理方法
- 適用集合的運算子
- 適用集合的方法

列表（list）、序對（或稱元）（tuple）、集合（set）和字典（dict）是容器型態（Container type），顧名思義它們就像容器一樣，可以裝進各種不同型態的資料，這些容器資料型態還能互相搭配使用，可說是學習 Python 非常重要的關鍵。

7-1 容器資料型態的比較

Python 的容器資料型態分為序對（tuple）、串列（list）、字典（dict）與集合（set），有各自的使用方法與限制，物件可區分可變（mutable）與不可變（immutable），不可變物件一旦創建後內容就不能再改變，容器物件只有 tuple 是不可變物件，其他三種都是可變物件，底下將四種容器資料型態先做個簡單介紹。

- **tuple（序對）**：資料放置於括號 () 內，資料有順序性，是不可變物件。
- **list（串列）**：資料放置於中括號 [] 內，資料有順序性，是可變物件。
- **dict（字典）**：是 dictionary 的縮寫，資料放置於大括號 {} 內，是「鍵（key）」與「值（value）」對應的物件，是可變物件。
- **set（集合）**：類似數學裡的集合概念，資料放置於大括號 {} 內，是可變物件，資料具有無序與互異的特性。

下表是四種容器型態的比較：

資料型態	tuple	list	dict	set
中文名稱	序對	串列	字典	集合
使用符號	()	[]	{}	{}
具順序性	有序	有序	無序	無序
可變 / 不可變	不可	可	可	可
舉例	(1, 2, 3)	[1,2,3]	{'word1':'apple'}	{1, 2, 3}

7-2 list 串列

使用單一變數來儲存資料時，當程式變數需求不多時，這種作法看似不會有太大的問題，為了方便儲存多筆相關的資料，大部分的程式語言會以**陣列**（Array）方式處理，不像其它的程式語言都有的「陣列」資料結構，在 Python 中是以串列 List 來扮演儲存大量有序資料的角色，它是一串由逗號分隔的值，用中括號 [] 包起來，如下所示：

```
fruitlist =  ["Apple", "Orange", "Lemon", "Mango"]
```

上面 list 物件共有 4 個元素，長度是 4，利用中括號 [] 搭配元素的索引（index）就能存取每一個元素，索引從 0 開始，由左至右分別是 fruitlist[0]、fruitlist [1]……以此類推。

7-2-1 建立串列

串列可以是空串列，串列中元素可以包含不同的資料型態或是其它的子串列，底下都是正確的串列表示方式：

```
data = []                          # 空的串列
data1 = [28, 16, 55]               # 儲存數值的 list 物件
data2 = ['1966', 50, 'Judy']       # 含有不同型別的串列
data3 = ['Math', [88, 92], 'English', [65, 91]]
```

Python 程式語言提供**生成式**（Comprehension）的作法，是一種建立串列更快速彈性的作法，串列中括號裡面可以結合 for 敘述及其它 if 或 for 敘述的運算式，此運算式所產生的結果就串列的元素。例如：

```
>>> list1 =[i for i in range(1,6)]
>>> list1
[1, 2, 3, 4, 5]
>>>
```

上述例子串列元素是 for 敘述的 i。

又例如：

```
>>> list2=[i+10 for i in range(50,60)]
>>> list2
[60, 61, 62, 63, 64, 65, 66, 67, 68, 69]
```

有關 range() 函數的使用方式有以下三個語法，分別是在不同個數參數的宣告方式：

▶ 1 個參數

range（整數值）會產生的串列是 0 到「整數值 -1」的串列，例如 range(4) 表示會產生 [0,1,2,3] 的串列。

▶ 2 個參數

range（起始值，終止值）會產生的串列是「起始值」到「整數值 -1」的串列，例如 range(2,5) 表示會產生 [2,3,4] 的串列。

▶ 3 個參數

range（起始值，終止值，間隔值）會產生的串列是「起始值」到「整數值 -1」的串列，但每次會遞增間隔值，例如：range(2,5,2) 表示會產生 [2,4] 的串列，這是因為每次遞增 2 的原因。

為什麼要使用「串列生成式」？除了提高效能之外，讓 for 迴圈讀取元素更加自動化。例如要找出數值 10~50 之間可以被 7 整除的數值，for 迴圈可以配合 range() 函數，再以 if 敘述做條件運算的判斷，能被 7 整除者以 append() 方法加入 List 中，以下述簡例來說明。

```
numA = []                    # 空的 List
for item in range(10, 50):
    if(item % 7 == 0):
        numA.append(item)    # 整除的數放入 List 中
print('10~50 被 7 整除之數：', numA)
```

利用串列生成式可以將上述的簡例以更簡潔的敘述來表現：

```
numB = [] # 空的 List
numB = [item for item in range(10, 50)if(item % 7 == 0)]
print('10~50 被 7 整除之數：', numB)
```

另外，也可以利用串列生成式將兩個串列以條件符合者做串接，例如：

```
num = ['AB01', 'AB425', 'CH004', 'CK4131',
       'DD0048', 'Dy00231']
room = ['A', 'B', 'C']
rooms = [r + '-' + n for r in room for n in num
    if r[0] == n[0]]
```

其中 num 和 room 皆是 list 物件。串列生成式中以 if 敘述做條件運算，找出 num 和 room 的元素中有字元相等者就加入串列中。最後輸出如下：

```
['A-AB01', 'A-AB425', 'C-CH004', 'C-CK4131']
```

7-2-2 常用串列內建函數

以下將介紹串列中常見的內建函數：

- **len(L)**：傳回串列物件 L 的長度，亦即該串列包含幾個元素。例如：

```
fruitlist = ["Apple", "Orange", "Lemon"]
print( len(fruitlist) )      # 長度 =3
```

- **內建函數 sum()**：內建函數 sum() 來計算總分，以下述範例解說 sum() 函數的用法。

〈用內建函數 sum() 計算總分〉

檔名 sum_score.py

```
score = []                          # 建立 list 來存放成績
# for 迴圈建立輸入成績的 list
for item in range(5):
    data = int(input(' 分數 %2d ' %(item + 1)))
    score += [data]
print('%5s %5s ' % ('index', 'score'))

# for 迴圈讀取成績並輸出
for item in range(len(score)):
    print('%3d %4d'% (item, score[item]))

print('-'*12)
# 利用內建函數 sum() 來計算總分
print(' 總分 ', sum(score), ', 平均 = ', sum(score)/5)
score.sort(reverse = True)          # 使用 sort() 方法由大而小排序
print(' 遞減排序：', score)
print(' 遞增排序：', sorted(score))  # 使用 BIF
```

執行結果 »

```
分數 1 89

分數 2 65

分數 3 91

分數 4 92

分數 5 84
index score
  0   89
  1   65
  2   91
  3   92
  4   84
------------
總分 421 , 平均 =  84.2
遞減排序： [92, 91, 89, 84, 65]
遞增排序： [65, 84, 89, 91, 92]
```

</> 程式解說 >>

- 第 03~05 行：for 迴圈建立輸入成績的 list。
- 第 09~10 行：for 迴圈讀取成績並輸出。
- 第 14 行：利用內建函數 sum() 來計算總分。
- 第 15 行：使用 sort() 方法由大而小排序。
- 第 17 行：使用 BIF sorted() 函數遞增排序。

▶ **內建函數 max(L)**：傳回串列物件 L 中最大的元素。例如：

```
print(max([1,3,5,7,9]))   #9
```

▶ **內建函數 min(L)**：傳回串列物件 L 中最小的元素。例如：

```
print(min([1,3,5,7,9]))   #1
```

7-2-3 常用串列運算子

之前介紹字串時提到 + 運算子可以串接字串，* 運算子可用來重複字串，比較運算子可以用來比較字串的大小。同樣的情形，這三種運算子也適用於串列。例如：

```
>>> [3,5,7]+[9,22,3,56]
[3, 5, 7, 9, 22, 3, 56]
>>> [3,5,7]*3
[3, 5, 7, 3, 5, 7, 3, 5, 7]
[3, 5, 7, 3, 5, 7, 3, 5, 7]
>>> [3,5,7]<[3,5,8]
True
>>> [3,5,7]==[3,5,7]
True
>>> [3,5,7]>[4,5,7]
False
```

另外，如果要檢查某一個元素是否存在或不存在於串列中，則可以使用 in 與 not in 運算子，例如：

```
>>> "Mon" in ["Mon","Tue","Fri"]
True
>>> "Sun" not in ["Mon","Tue","Fri"]
True
```

索引與切片運算

list 物件也可以搭配切片 (slice) 方法來取出元素，例如：

```
str1 = ['A','B','C','D','E','F']
print(str1[:3])    # 取出索引 0~2 的元素
print(str1[2:4])   # 取出索引 2~3 的元素
print(str1[4:])    # 取出索引 4 之後的元素
```

執行結果 >>

```
['A', 'B', 'C']
['C', 'D']
['E', 'F']
```

以下範例可以將串列中的奇偶數分離。

〈將串列中奇偶數分離〉

檔名 evenAndOdd.py

```
01  # -*- coding: utf-8 -*-
02  '''
03  將串列中奇偶數分離
04  '''
05
06  Number = [1,2,3,4,5,6,7,8,9,10]
07  even_num = Number[1::2]
08  odd_num = Number[0::2]
09
10  print(" 偶數：{}\n 奇數：{}".format(even_num,odd_num))
```

執行結果 >>

```
偶數：[2, 4, 6, 8, 10]
奇數：[1, 3, 5, 7, 9]
```

</> 程式解說 >>

- 第 07 行：利用切片運算取出偶數。
- 第 08 行：利用切片運算取出奇數。

各位可以用中括號搭配索引值來指定要修改哪一個元素的值，例如：

```
fruitlist = ["Apple", "Orange", "Lemon"]
fruitlist[1]="Kiwi"
```

執行結果 >>

```
['Apple', 'Kiwi', 'Lemon']
```

7-2-4 串列常用方法

串列 list 資料是可變物件，可以修改元素資料，也可以新增或移除元素，底下整理 list 常用的方法：

方法	說明
append()	附加新元素
count(x)	計算 list 串列中 x 出現的次數
insert()	插入新元素
pop()	移除元素，預設是最後一個
remove()	移除元素
reverse()	倒轉元素的順序
sort()	排序

底下詳細說明 list 串列的方法：

附加 append()

append() 方法會將新的元素加到末端，例如：

```
fruitlist = ["Apple", "Orange", "Lemon"]
fruitlist.append("Mango")
```

執行結果 >>

```
['Apple', 'Orange', 'Lemon', 'Mango']
```

插入 insert ()

insert () 方法可以指定新的元素要放置在哪個索引處，格式如下：

```
list.insert( 索引值 , 新元素 )
```

索引值是指 list 的索引位置，索引值為 0 表示放置於最前端，舉例來說，要將新元素插入在索引 1 的位置，可以這樣表示：

```
fruitlist = ["Apple", "Orange", "Lemon"]
fruitlist.insert(1,"banana")
```

執行結果 >>

```
['Apple', 'banana', 'Orange', 'Lemon']
```

移除元素 remove ()

remove() 方法可以在括號內直接指定要移除的元素，例如：

```
fruitlist = ["Apple", "Orange", "Lemon"]
fruitlist.remove("Orange")
```

執行結果 >>

```
['Apple', 'Lemon']
```

移除元素 pop ()

pop () 方法可以在括號內指定要移除的元素索引，就可以將該元素移除，例如：

```
fruitlist = ["Apple", "Orange", "Lemon"]
fruitlist.pop(1)
```

執行結果 >>

```
['Apple', 'Lemon']
```

如果 pop() 括號內沒有指定索引值，則預設移除最後一個。

```
fruitlist = ["Apple", "Orange", "Lemon"]
fruitlist.pop()
```

執行結果 >>

```
['Apple', 'Orange']
```

排序 sort ()

sort () 方法可以將 list 串列資料進行排序，例如：

```
fruitlist = ["Apple", "Orange", "Lemon"]
fruitlist.sort()
```

執行結果 >>

```
['Apple', 'Lemon', 'Orange']
```

7-2-5 用 del 刪除變數與元素

del 敘述除了可以刪除串列中指定索引的元素外，也可以一次刪除切片運算子所指定的索引範圍。當你不再使用串列這個變數時，也可以透過 del 敘述刪除串列變數。例如底下的敘述表示從串列 L 刪除索引位置為 3 的元素，即數字 4：

```
>>> L=[1,2,3,4,5,6,7,8,9]
>>> del L[3]
>>> L
[1, 2, 3, 5, 6, 7, 8, 9]
>>>
```

又如下面的敘述會刪除 L 串列索引位置 0 到 2-1 的元素，即刪除串列中前兩個元素。

```
>>> L=[1,2,3,4,5,6,7,8,9]
>>> del L[0:2]
>>> L
[3, 4, 5, 6, 7, 8, 9]
>>>
```

7-2-6 二維及多維串列

在 Python 中，串列中可以有串列，這種就稱為二維串列，要讀取二維串列的資料可以透過 for 迴圈。二維串列簡單來講就是串列中的元素是串列，下述簡例說分明：

```
number = [[11, 12, 13], [22, 24, 26], [33, 35, 37]]
```

上述中的 number 是一個串列。number[0] 或稱第一列索引，存放另一個串列；number[1] 或稱第二列索引，也是存放另一個串列，依此類推。第一列索引有 3 欄，個別存放元素，其位置 number[0][0] 是指向數值「11」，number[0][1] 是指向數值「12」，依此類推。所以 number 是 3*3 的二維串列（two-dimensional list），其列和欄的索引示意如下：

	欄索引 [0]	欄索引 [1]	欄索引 [2]
列索引 [0]	11	12	13
列索引 [1]	22	24	26
列索引 [2]	33	35	37

number 二維 List 同樣是以 [] 運算子來表達其索引並存取元素，語法如下：

```
串列名稱 [ 列索引 ][ 欄索引 ]
```

例如：

```
number[0]            # 輸出第一列的三個元素
[11, 12, 13]
number[1][2]         # 輸出第二列的第三欄元素
26
```

如果要宣告一個 N*N 維的二維串列，其語法範例如下：

```
arr=[[None] * N for row in range(N)]
```

這裏假設 arr 為一個 3 列 5 行的二維陣列，也可以視為 3*5 的矩陣。在存取二維陣列中的資料時，使用的索引值仍然是由 0 開始計算。

在二維陣列設定初始值時，為了方便區隔行與列。所以除了最外層的 [] 外，必須以 [] 括住每一列的元素初始值，並以「,」區隔每個陣列元素，語法如下：

```
陣列名稱 =[ [ 第 0 列初值 ],[ 第 1 列初值 ],…,[ 第 n-1 列初值 ] ]
```

例如：

```
arr=[[1,2,3],[2,3,4]]
```

接下來將利用 Python 語言的二維串列的方式來撰寫一個求二階行列式的範例。二階行列式的計算公式為：a1*b2-a2*b1。

〈二階行列式〉

檔名 twoArray.py

```
N=2
# 宣告 2x2 陣列 arr 並將所有元素指定為 None
arr=[[None] * N for row in range(N)]
print('|a1 b1|')
print('|a2 b2|')
arr[0][0]=input(' 請輸入 a1:')
arr[0][1]=input(' 請輸入 b1:')
arr[1][0]=input(' 請輸入 a2:')
arr[1][1]=input(' 請輸入 b2:')
# 求二階行列式的值
result = int(arr[0][0])*int(arr[1][1])-int(arr[0][1])*int(arr[1][0])
print('|%d %d|' %(int(arr[0][0]),int(arr[0][1])))
print('|%d %d|' %(int(arr[1][0]),int(arr[1][1])))
print(' 行列式值 =%d' %result)
```

執行結果 »

```
|a1 b1|
|a2 b2|

請輸入a1:5

請輸入b1:9

請輸入a2:3

請輸入b2:4
|5 9|
|3 4|
行列式值=-7
```

在 Python 語言中三維陣列宣告方式如下：

```
num=[[[33,45,67],[23,71,66],[55,38,66]],[[21,9,15],[38,69,18],
[90,101,89]]]
```

假設一個三維陣列元素內容如下：

```
num=[[[33,45,67],[23,71,66],[55,38,66]],[[21,9,15],[38,69,18],
[90,101,89]]]
```

請設計一支 Python 程式，利用三層巢狀迴圈來找出此 2x3x3 三維陣列中所儲存數值中的最小值。

〈三維陣列〉

檔名 threeArray.py

```
01  # 宣告三維陣列
02  num=[[[33,45,67],[23,71,66],[55,38,66]], \
03       [[21,9,15],[38,69,18],[90,101,89]]]
04  value=num[0][0][0]     # 設定 main 為 num 陣列的第一個元素
05  for i in range(2):
06      for j in range(3):
07          for k in range(3):
08              if(value>=num[i][j][k]):
09                  value=num[i][j][k]     # 利用三層迴圈找出最小值
10  print(" 最小值 = %d" %value)
```

執行結果 »

```
最小值= 9
```

7-3 tuple 序對（或稱元組）

tuple 是有序物件，類似 list 串列，差別在於 tuple 是不可變物件，一旦建立之後，序對中的元素不能任意更改其位置與更改內容值。

7-3-1 建立序對

序對是一串由逗號分隔的值，可以用括號 () 建立 tuple 物件，也可以用逗號建立 tuple 物件，如下所示：

```
fruitlist = ("Apple", "Orange", "Lemon")
fruitlist = "Apple", "Orange", "Lemon"
```

上面兩個式子都是建立 tuple 物件：("Apple", "Orange", "Lemon")，如果 tuple 物件裡只有一個元素，仍必須在元素之後加上逗號，例如：

```
fruitlist = ("Apple",)
```

序對可以存放不同資料型態的元素。序對每個元素的索引編號左邊是由 [0] 開始，右邊則是由 [-1] 開始。串列是以中括號 [] 來存放元素，但是序對卻是以小括號 () 來存放元素。

因為序對內的元素有對應的索引編號，因此可以使用 for 迴圈或 while 迴圈來讀取序對內的元素。例如以下敘述以 for 迴圈將元組中的元素輸出，其中 len() 函數可以求取元組的長度：

〈新建 tuple〉

檔名 tuple_create.py

```
01  tuple1 = (24, 38, 32, 48, 4)
02  print('tuple 所有的元素如下：')
03  for item in range(len(tuple1)):
04      print ('tuple1[%2d] %3d' %(item, tuple1[item]))
```

執行結果 >>

```
tuple 所有的元素如下：
tuple1[ 0]  24
tuple1[ 1]  38
tuple1[ 2]  32
tuple1[ 3]  48
tuple1[ 4]   4
```

雖然在序對內的元素不可以用 [] 運算子來改變元素的值，不過序對內元素仍然可以配合「+、*」運算子做一些其他應用。利用「+」運算子可以將兩個序對資料內容串接成一個新的序對，而「*」運算子可以複製序對的元素成多個。

7-3-2 序對的內建函數

前面串列提到的內建函數，大部分適用於序對，說明如下：

- **內建函數 sum()**：內建函數 sum() 來計算總分。

```
score = (90,100,98,86,86) # 建立 tuple 來存放成績
print(' 總分 ', sum(score), ', 平均 = ', sum(score)/5)
```

執行結果 >>

```
總分 460 , 平均 =  92.0
```

- **max(T)**：傳回串列物件 T 中最大的元素。例如：

```
>>>max((1,3,5,7,9))
9
```

- **min(T)**：傳回串列物件 T 中最小的元素。例如：

```
>>>min((1,3,5,7,9))
1
```

7-3-3 序對運算子

tuple 物件同樣可以利用中括號 [] 搭配元素的索引 (index) 讀取個別元素，例如：

```
fruitlist = ("Apple", "Orange", "Lemon")
print( fruitlist[1] )   #Orange
```

原則上那些在串列中不會變動到元素內容的運算子都適用序對，例如連接運算子 (+)、重複運算子 (*)、比較運算子、索引運算子、切片運算子、in 與 not in 運算子等。有關序對運算子的使用方式，舉例如下：

```
>>> (3,5,7)+(9,22,3,56)
(3, 5, 7, 9, 22, 3, 56)
>>> (3,5,7)*3
(3, 5, 7, 3, 5, 7, 3, 5, 7)
>>> (3,5,7)<(3,5,8)
True
>>> (3,5,7)==(3,5,7)
True
>>> (3,5,7)>(4,5,7)
False
>>> "Mon" in ("Mon","Tue","Fri")
True
>>> "Sun" not in ("Mon","Tue","Fri")
True
```

7-3-4 Unpacking（拆解）與 Swap（交換）

tuple 是不可變物件，不可以直接修改 tuple 元素的值，因此串列中像 append()、insert() 等會改變元素個數或值的方法，都不可以使用。但是像 count() 用來統計項目出現次數，或是 index() 用來取得某項目第一次出現的索引值等，就可以應用在序對資料型態。

Python 針對序對有個很特別的用法 Unpacking（拆解）。舉例來說，下列第 1 行敘述將「Andy」、「25」以及「高雄」這三個值定義為 tuple 序對，第 2 行則使用變數取出 tuple 中的元素值，稱為 Unpacking（拆解）：

```
datalist = ("Andy", "25", " 高雄 ")    # Packing
name, age, addr= datalist              # Unpacking
print(name)                            # 輸出 Andy
```

Unpacking 不只限於 tuple，包括 list 跟 set 等序列物件，都可以用同樣的方式來指定變數，序列拆解的等號左邊的變數數量必須與等號右邊的序列元素數量相同。

在其他程式語言，如果想要交換 (Swap) 兩個變數的值，通常需要第三個變數來輔助，例如 x=10、y=20，如果要讓 x 與 y 的值對調，程式會如下表示：

```
temp = x
x = y
y = temp
```

利用 Unpacking 的特性，變數值交換變得非常簡單，只要底下一行程式，就可以達到交換的目的：

```
y,x = x,y
```

〈序對交換 I〉

檔名 tuple.py

```
01  # -*- coding: utf-8 -*-
02
03  x = 10
04  y = 20
05  print('x={},y={}'.format(x,y))
06
07  y,x = x,y
08  print('Swap x={},y={}'.format(x,y))
```

執行結果 »

```
x=10,y=20
Swap x=20,y=10
```

</> 程式解說 »

- 第 07 行：利用 Unpacking 的特性，變數值交換變得非常簡單，只要底下一行程式，就可以達到交換的目的。

以下範例是將 for 迴圈結合 Unpacking 的概念，來輔助購買物品單價加總或平均的計算工作。

〈序對交換 II〉

檔名 tuple_price.py

```
01  money = [[' 書籍 ', 250, 480, 365],
02                          [' 音樂 CD', 450, 380, 600],
03                          ['POLO 上衣 ', 680, 390, 480]]
04
05  for(product, price1, price2, price3) in money:
06      print('%6s'%product,' 三次購買價格總和 :',
07              (price1 + price2 + price3))
```

執行結果 »

```
    書籍  三次購買價格總和: 1095
  音樂CD  三次購買價格總和: 1430
POLO上衣  三次購買價格總和: 1550
```

</> 程式解說 »

- 第 01~03 行：建立二維串列，即串列中有串列，存放名稱和三次購買價格。
- 第 05~07 行：利用 Unpacking 的功能，for 迴圈讀取時才給予識別字名稱，並輸出其值。

7-4 dict 字典

字典（dict）是 dictionary 的縮寫，資料放置於大括號 {} 內，每一筆資料是一對 key:value，格式如下：

```
{key:value}
```

7-4-1 建立字典

dict 字典中的 key 必須是不可變的（immutable）的資料型態，例如數字、字串，而 value 就沒有限制，可以是數字、字串、list 串列、tuple 序對等等，資料之間必須以逗號 (,) 隔開，例如：

```
d={'name':'Andy', 'age':18, 'city':'台北'}
```

上面敘述共有三筆資料，利用每一筆資料的 key 就可以讀出代表的值，例如：

```
print(d['name'])  #輸出 Andy
print(d['age'])   #輸出 18
print(d['city'])  #輸出台北
```

建立字典的方式除了利用大括號 {} 產生字典，也可以使用 dict() 函數，或是先建立空的字典，再利用 [] 運算子以鍵設值。

字典和串列（list）、序對（tuple）很大的不同點，正因為字典儲存資料是沒有順序性的，它是使用「鍵」查詢「值」，所以適用於序列型別的「切片」運算，在字典中就無法使用。

7-4-2 字典的基礎操作

字典中的「鍵」必須是唯一，而「值」可以是相同值，字典中如果有相同的「鍵」卻被設定不同的「值」，則只有最後面的「鍵」所對應的「值」有效。

例如以下的範例中，字典中的 'city' 鍵被設定兩個不同的值，前面那一個是設定為 ' 台北 '，後面那一個是設定為 ' 高雄 '，所以前面會被後面那一個設定值 ' 高雄 ' 所覆蓋。請參考以下的程式碼說明：

```
dic={'name':'Andy', 'age':18, 'city':' 台北 ','city':' 高雄 '} # 設定字典
print(dic['city']) # 會印出高雄
```

要修改字典的元素值必須針對「鍵」設定新值，才能取代原先的舊值。例如：

```
dic['name']= 'Tom'     # 將字典中的「' name '」鍵的值修改為 'Tom'
print(dic) # 會輸出 {'name': 'Tom', 'age': 18, 'city': ' 高雄 '}
```

如果要新增字典的鍵值對，只要加入新的鍵值即可。語法如下：

```
dic['hobby']= ' 籃球 '
# 在字典中新增「' hobby '」，該鍵所設定的值為 ' 籃球 '
print(dic)
# 新增元素後的字典 {'name': 'Tom', 'age': 18, 'city': ' 高雄 ', 'hobby': ' 籃球 '}
```

如果要刪除字典中的特定元素，語法如下：

```
del 字典名稱 [ 鍵 ]
```

例如：

```
del dic['hobby ']
```

當字典不再使用時，如果想刪除整個字典，則可以使用 del 指令，

例如：

```
del dic
```

7-4-3 適用字典的函數

前面介紹的內建函數中的 len() 函數適用於字典，它會傳回字典中包含幾組「key:value」，例如：

```
dic={'name':'Andy', 'age':18, 'city':' 台北 ','city':' 高雄 '}
# 設定字典
print(len(dic)) # 會印出 3 表示字典 dic 包含 3 個 key:value
```

7-4-4 適用字典的運算子

由於字典是一種無序的資料型態，所以不支援串接運算子 (+) 或重複運算子 (*) 等，在比較運算子中可以使用「==」和「!=」運算子將字典逐項目做比對，其他的比較運算子則不能使用於字典。序列型別使用 [] 運算子指明索引之後，可以取得元素值。而字典的項目也能使用 []、in 及 not in 運算子，例如：

```
del d[key]              # 刪除字典項目，依 key 執行
key in d                # 判斷鍵「key」是否在字典中
key not in d            # 判斷鍵「key」是否不在字典中
```

7-4-5 適用字典的處理方法

dict 是可變物件，下表是常用的一些方法：

方法	說明
clear()	清空 dict 物件
copy()	複製 dict 物件
get()	以 key 來搜尋資料
pop()	移除元素
update()	合併或更新 dict 物件
keys()	取出 key 以 dict_items 物件型態回傳
values()	取出 value 以 dict_items 物件型態回傳

底下詳細說明 dict 字典元素的修改方法：

清除 -clear()

clear() 方法會清空整個字典，例如：

```
d1={"name":"Andy", "age":18, "city":" 台北 "}
d1.clear()
print(d1)
```

執行結果 >>

```
{}
```

複製 dict 物件 - copy()

使用 copy() 方法可以複製 dict 物件，例如：

```
d1={"name":"Andy", "age":18, "city":" 台北 "}
d2={"name":"Brian", "age":25, "city":" 高雄 "}
d1=d2.copy()
print(d1)
```

執行結果 >>

```
{"name":"Brian", "age":25, "city":" 高雄 "}
```

各位可能會提出疑問，可以直接寫「d1=d2」嗎？我們來看一下兩者的差別，使用 copy() 方法複製的 dict 字典只是將資料複製過去，d1 與 d2 兩者沒有關聯，仍然是兩個不同的物件；如果使用「d1=d2」，表示 d2 物件指定給 d1 物件，此時修改 d2 的資料，d1 也會跟著更改。我們舉同樣的例子來看，d1 物件與 d2 物件如下：

```
d1={"name":"Andy", "age":18, "city":" 台北 "}
d2={"name":"Brian", "age":25, "city":" 高雄 "}
```

使用 copy 方法，程式敘述與執行結果如下：

```
d1=d2.copy()
d2["name"]="Jennifer"
print(d1["name"])  # 輸出 Brian
```

使用「d1=d2」，程式敘述與執行結果如下：

```
d1=d2
d2["name"]="Jennifer"
print(d1["name"])  # 輸出 Jennifer
```

搜尋元素值 - get()

get() 方法會以 key 搜尋對應的 value，格式如下：

```
v1=dict.get(key[, default=None] )
```

例如：

```
d1={"name":"Andy", "age":18, "city":" 台北 "}
city=d1.get("age")   # 輸出 18
```

如果指定的 key 不存在，會傳回 default 值也就是 None，各位也可以改變 default 值，那麼當 key 不存在，就會顯示出來，例如：

```
d1={"name":"Andy", "age":18, "city":" 台北 "}
city=d1.get("home"," 找不到 ")  # 輸出「找不到」
```

移除元素 -pop()

pop() 方法可以移除指定的元素，例如：

```
d1={"name":"Andy", "age":18, "city":" 台北 "}
d1.pop("city")
print(d1)
```

執行結果 >>

```
{'name': 'Andy', 'age': 18}
```

更新或合併元素 -update()

update() 方法可以將兩個 dict 字典合併，格式如下：

```
dict1.update(dict2)
```

dict1 會與 dict2 字典合併，如果有重複的值，括號內的 dict2 字典元素會取代 dict1 的元素，例如：

```
d3={"name":"Joan","height":'180cm'}
d4={"hobby":"dancing", "height":'168cm'}
d3.update(d4)
print(d3)
```

執行結果 >>

```
{'name': 'Joan', 'hobby': 'dancing', 'height': '168cm'}
```

items()、keys() 與 values()

items() 方法是用來取 dict 物件的 key 與 value，keys() 與 values() 這兩個方法是分別取 dict 物件的 key 或 value，回傳的型態是 dict_items 物件，例如：

```
d1={"name":"Andy", "age":18, "city":" 台北 "}
print(d1. items())
print(d1. keys())
print(d1.values())
```

執行結果 >>

```
dict_items([('name', 'Andy'), ('age', 18), ('city', ' 台北 ')])
dict_keys(['name', 'age', 'city'])
dict_values(['Andy', 18, ' 台北 '])
```

通常 items()、keys() 與 values() 方法會搭配 for 迴圈來取值，譬如底下敘述：

```
d1={"name":"Andy", "age":18, "city":"台北"}
for v in d1.values():
    print(v)
```

底下再透過範例練習字典的新增、移除與讀取元素。

〈字典新增、移除與存取〉

檔名 dict_example.py

```
01  # -*- coding: utf-8 -*-
02
03  dictStr = {'bird':'鳥', 'cat':'貓', 'dog':'狗', 'pig':'豬'}
04  #新增 wolf
05  dictStr['wolf']="狼"
06
07  #刪除 pig
08  dictStr.pop("pig")
09
10  #列出 dictStr 所有的 value
11  print("dictStr 目前的元素：")
12  for v in dictStr.values():
13      print(v)
14
15  #搜尋
16  print("搜尋 dog==>"+dictStr.get("dog","不在 dictStr"))
```

執行結果 »

```
dictStr目前的元素：
鳥
貓
狗
狼
搜尋dog==>狗
```

</> 程式解說 >>

- 第 05 行：新增字典元素。
- 第 08 行：pop() 方法可以移除指定的元素。
- 第 12~13 行：列出 dictStr 字典所有的 value。
- 第 16 行：利用 get() 方法會以 "dog" key 搜尋對應的 value。

7-5 set 集合

set 集合與 dict 字典一樣都是把元素放在大括號 {} 內，不過 set 集合只有鍵 (key) 沒有值 (value)，類似數學裡的集合，可以進行聯集 (|)、交集 (&)、差集 (-) 與互斥或 (^) 等運算。另外，集合裡的元素沒有順序之分及不可重複出現。

7-5-1 建立集合

set 集合可以使用大括號 {} 或 set() 方法建立，使用大括號 {} 建立的方式如下：

```
fruitlist = {"Apple", "Orange", "Lemon"}
```

7-5-2 適用集合的運算子

兩個集合可以做聯集 (|)、交集 (&)、差集 (-) 與互斥或 (^) 等運算，如下表所示。

集合運算	範例	說明
聯集 (|)	A|B	存在集合 A 或存在集合 B
交集 (&)	A&B	存在集合 A 也存在集合 B
差集 (-)	A-B	存在集合 A 但不存在集合 B
互斥或 (^)	A^B	排除相同元素

底下範例說明集合的運算操作方式。

〈集合的聯集、交集與差集〉

檔名 set_operations.py

```
01 # -*- coding: utf-8 -*-
02 '''
03 set 集合
04 聯集 (|)、交集 (&)、差集 (-) 與互斥或 (^) 運算
05 '''
06
07 zooA= {"bird", "cat", "dog","pig"}
08 zooB = {"wolf", "cat", "dog","turtle"}
09 print(zooA & zooB)
10 print(zooA | zooB)
11 print(zooA - zooB)
12 print(zooA ^ zooB)
```

執行結果 »

```
{'cat', 'dog'}
{'bird', 'turtle', 'dog', 'pig', 'cat', 'wolf'}
{'pig', 'bird'}
{'bird', 'turtle', 'pig', 'wolf'}
```

7-5-3 適用集合的方法

底下是使用 set() 方法建立集合，請注意，括號 () 裡只能有一個 iterable（迭代）物件，也就是 str，list，tuple，dict 等等，例如：

```
strObject = set("ABCD")
listObject = set(["Apple", "Orange", "Lemon"])
tupleObject = set(("Apple", "Orange", "Lemon"))
dictObject = set({"name":"Andy", "age":18, "city":" 台北 "})
```

set() 使用 dict 當引數時只會保留 key，所以上面敘述產生的 set 物件如下：

```
{'A', 'B', 'C', 'D'}
{'Apple', 'Orange', 'Lemon'}
{'Apple', 'Orange', 'Lemon'}
{'age', 'city', 'name'}
```

集合常用的方法如下表：

方法	說明
add()	新增元素
remove()	移除元素
update()	合併或更新 set 物件
clear()	清空 set 集合

以下說明這些方法的使用方式。

新增與刪除元素 -add() / remove()

add 方法一次只能新增一個元素，如果要新增多個元素，可以使用 update() 方法，底下是 add 與 remove 方法的使用方式：

```
animal = {"bird", "cat", "dog"}
animal.add("fish")
print(animal)
```

執行結果 >>

```
{'dog', 'cat', 'bird', 'fish'}
```

```
animal = {"bird", "cat", "dog"}
animal.remove("cat")
print(animal)
```

執行結果 >>

```
{'dog', 'bird'}
```

更新或合併元素 -update()

update() 方法可以將兩個 set 集合合併，格式如下：

```
set1. update(set2)
```

set1 會與 set2 合併，由於 set 集合不允許重複的元素，如果有重複的元素會被忽略，例如：

```
animal = {"bird", "cat", "dog"}
animal.update({"bird","monkey"})
print(animal)
```

執行結果 >>

```
{'dog', 'bird', 'cat', 'monkey'}
```

建立集合後，可以使用 in 敘述來測試元素是否在集合中，例如：

```
animal = {"bird", "cat", "dog"}
print("fish" in animal)  #輸出 False
```

"fish" 並不在 animal 集合內，所以就會傳會 False。

7-6 上機實作演練（一）一簡易單字翻譯機（GUI 圖形介面）

這一個範例將資料儲存於字典（dict）結構，利用 dict 的特性製作一個簡易的單字翻譯機。範例使用 Tkinter 製作 GUI 介面，各位可以開啟 dictionary_undone.py 檔案，檔案已經將 GUI 圖形介面完成，各位可以練習加入程式，當然最好能從無到有完成這一個範例，也能練習 GUI 圖形介面的製作。

7-6-1 程式範例描述

設計一個程式可讓使用者輸入單字之後，點選「中翻英」顯示英文，點選「英翻中」顯示中文。

➤ 輸入說明

範例中已預設好字典（dict）內的資料，使用者只需在輸入框輸入查詢的中文字或英文字，點選「中翻英」或「英翻中」即可查詢。

➤ 輸出範例

7-6-2 程式碼說明

首先，定義了一個 dict 字典物件命名為 dictionary，元素資料如下：

```
dictionary = {'bird':'鳥', 'cat':'貓', 'dog':'狗', 'pig':'豬'}
```

其中 key 是儲存英文字，value 是儲存對應的中文字。

範例使用了三個按鈕，分別是「中翻英」鈕、「英翻中」鈕與「清除」鈕。按下「中翻英」鈕時呼叫 ctoe() 函數；按下「英翻中」鈕時呼叫 etoc () 函數；按下「清除」鈕時呼叫 clear() 函數。

我們先來看看 ctoe() 與 etoc () 函數能夠做哪些事情：

1. 取得使用者輸入的內容（entry 元件）。
2. 搜尋 dictionary 的 key 或 value 是否有符合的字。
3. 在 label 元件顯示搜尋結果。

其中 etoc() 函數執行的動作為英翻中，所以用 key 來找對應的 value 值，dict 字典物件本身就有 get() 方法可以使用，程式如下所示：

```
def etoc():
    i = entry.get()                 # 取得 entry 元件輸入的內容
    ans = dictionary.get(i," 找不到 ["+i+"]")
    label.config(text = ans)        # 在 label 元件顯示文字
```

get() 方法直接用 key 來搜尋 value，找到符合的字就傳給變數 ans，找不到就將 get() 方法的第 2 個引數值傳給變數 ans，因此顯示找不到的訊息。

ctoe() 函數執行的動作為是中翻英，是用 value 來找對應的 key 值，由於 dict 字典並沒有適用的方法，所以這裡使用 for 迴圈一一比對 value 值，找到適合的就跳離迴圈，程式如下：

```
i = entry.get()
for k,v in dictionary.items():
    if v == i:
        ans = k
        break

if ans:
   label.config(text = ans)                 # 在 label 元件顯示文字
else:
   label.config(text = " 找不到 ["+i+"]")    # 在 label 元件顯示文字
```

dict 字典的 items() 方法是用來取出 dict 物件的 key 與 value，將 value 與輸入的 i 作比較，找到正確的字之後就將 key 傳給變數 ans。變數 ans 有值就表示有找到符合的字，否則就顯示找不到的訊息。

當按下「清除」鈕時呼叫 clear() 函數，清除 entry 元件與 label 元件的內容，程式如下：

```
def clear():
    entry.delete(0, "end")
    label.config(text = "")
```

底下為範例完整的程式碼：

〈簡易單字翻譯機〉

檔名 review_dictionary.py

```
01  # -*- coding: utf-8 -*-
02  """
03  程式名稱：簡易單字翻譯機
04  題目要求：
05  讓使用者輸入單字之後，點選「中翻英」顯示英文，點選「英翻中」顯示中文
06  """
07  def ctoe():
08      i = entry.get()                          # 取得 entry 元件輸入的內容
09      ans=""
10      for k,v in dictionary.items():
11          if v == i:
12              ans = k
13              break
14
15      if ans:
16          label.config(text = ans)             # 在 label 元件顯示文字
17      else:
18          label.config(text = " 找不到 ["+i+"]") # 在 label 元件顯示文字
```

```
19
20 def etoc():
21     i = entry.get()             # 取得 entry 元件輸入的內容
22     ans = dictionary.get(i," 找不到 ["+i+"]")
23     label.config(text = ans)    # 在 label 元件顯示文字
24
25 def clear():
26     entry.delete(0, "end")
27     label.config(text = "")
28
29 dictionary = {'bird':' 鳥 ', 'cat':' 貓 ', 'dog':' 狗 ', 'pig':' 豬 '}
30
31 #GUI 介面
32 import tkinter as tk
33 win = tk.Tk()
34 win.title(" 簡易單字翻譯機 ")
35
36 frame = tk.Frame(win)
37 frame.pack(padx=5, pady=5)
38 frame1 = tk.Frame(win)
39 frame1.pack(padx=5, pady=5)
40
41 entry = tk.Entry(frame, bg="#99ffcc", font = "JhengHei 15" ,
   borderwidth = 3)
42 entry.config(width=10)
43 entry.grid(column=0,row=0)
44
45 label = tk.Label(frame, bg="#ffffcc", font = "JhengHei 15", text = "")
46 label.config(width=10)
47 label.grid(column=1,row=0)
48
49 btnCtoe = tk.Button(frame1, text=" 中翻英 ", command=ctoe)
50 btnCtoe.grid(column=0,row=0)
51 btnEtoc = tk.Button(frame1, text=" 英翻中 ", command=etoc)
52 btnEtoc.grid(column=1,row=0)
53 btnClear = tk.Button(frame1, text="  清  除  ", command=clear)
54 btnClear.grid(column=2,row=0)
55 win.mainloop()
```

7-7 上機實作演練（二）—使用 Facebook API 取得粉絲專頁貼文

臉書—Facebook 是目前最熱門且擁有最多會員人數的社群網站，許多人幾乎每天一睜開眼就先上臉書，關注朋友們的最新動態，一般人除了由臉書來了解朋友的最新動態和訊息外，透過朋友的分享也能從中獲得更多更廣泛的知識，之前章節我們練習了 Open Data 的 XML 與 JSON 資料格式的擷取方法，本章特別增加此一範例，練習擷取 FB 粉絲頁的 PO 文，在各位學過了列表（list）、序對（或稱元組）（tuple）、集合（set）和字典（dict）等型態之後，相信對於資料擷取會更駕輕就熟。

7-7-1 程式範例描述

▶ 輸入說明

從「facebook for developers」的圖形 API 測試工具取得「存取權杖」與粉絲專頁 id。

▶ 輸出範例

將粉絲專頁的建立日期與貼文內容 print 輸出。

```
2015-12-04T01:25:18+0000
祖穆拉的愛恨情愁

2015-12-03T00:44:48+0000
忘情酒的由來~敢愛敢恨的百合花精祖穆拉
```

▶ 製作步驟

步驟 ① 從「facebook for developers」的圖形 API 測試工具取得「存取權杖」與粉絲專頁 id

步驟 ② 利用 json 模組解析 JSON 資料

步驟 ③ print 輸出

Facebook 提供了圖形 API 測試工具，只要有粉絲團的網址，就可以擷取粉絲團的資訊，本例以取得粉絲團最近兩筆的貼文為例說明。

首先請先開啟各位想擷取貼文的粉絲團，並複製粉絲團的網址。

接著到「facebook for developers」圖形 API 測試工具網頁，網址如下：https://developers.facebook.com/tools/explorer/

進入頁面之後 ❶ 先點選「取得權杖」，從下拉式選單中選擇「取得粉絲專頁存取權杖」，❷ 在 GET 處貼上剛剛複製的網址，❸ 點選「提交」，就會顯示粉絲專頁的 name 與 id，如下圖。請將 token 與 id 複製到記事本，待會程式中會需要用到這兩個資訊。

Tips

各位可以進入 https://developers.facebook.com/，點選上方的「工具及支援」，再點選「圖形 API 測試工具」也可進入網頁。

取得的 Token 是暫時的，大約只有 1 個小時的時效，過期再重新點選「取得權杖」就可以了，有了 Token 就可以擷取與使用 FB 資料，因此要小心保存，不要將 Token 外流喔！

7-7-2 程式碼說明

底下為取得粉絲專頁貼文的程式碼。

〈取得 FB 粉絲專頁貼文〉

檔名 review_FB_parse.py

```
01  import urllib.request as ur
02
03  access_token="EAACEdE…"
04  access_id="368113116692686"
05
06  urls = "https://graph.facebook.com/v2.11/{}/posts?limit=
    2&access_token={}".format(access_id,access_token)
07
08  with ur.urlopen(urls) as response:
09      s=response.read()
10
11  import json
12  r = json.loads(s)
13  r_data=r['data']
14
15  for i in range(len(r_data)):
16      print("{0}\n{1}\n".format(r_data[i]["created_time"],
        r_data[i]["message"]))
```

程式第 3 行及第 4 行分別是放置 token 及 id，範例的程式中並沒有放置 token 所以無法直接執行，請將各位複製的 token 及 id 貼上即可。

取得貼文的網址如下：

```
https://graph.facebook.com/v2.11/{id}/posts?limit=2&access_token={token}
```

其中 limit 參數表示一次要擷取幾筆資料，facebook API 限制一次上限為 100 筆資料，本範例擷取 2 筆，所以 limit 改為 2。如果想要擷取超過 100 筆的資料，可以利用迴圈將資料取出。

想要擷取資料，必須要知道資料的結構是如何，我們可以使用 print(r) 指令輸出看看裡面包含哪些元素，輸出後格式大概是這樣：

```
{"data":[{
    'created_time':'...',
    'message':'.....',
    'story': '...',
    'id': '...'
}]}
```

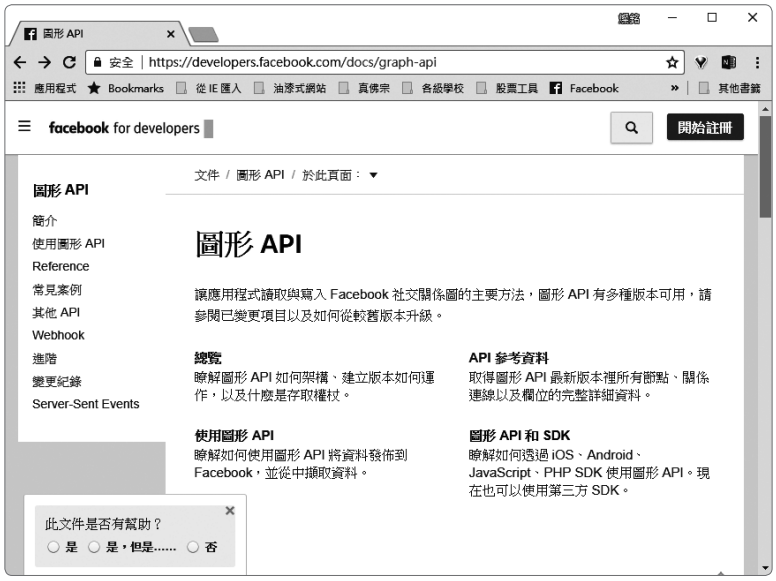
程式第 13 行取出 data 元素，程式碼如下：

```
r_data=r['data']
```

各位可以使用 print(type(r_data)) 以及 print(type(r_data[0])) 輸出看看它的資料型態，顯示出來的資料型態是 list 串列，串列裡的元素是 dict 型態。接著，只要用迴圈將資料讀出就完成了。

Facabook 的 graph-api 提供了相當多的功能，除了從中擷取資料之外，也能將資料發佈到 Facebook，如果各位對這方面有興趣，可以前往 graph-api 的說明文件網站，網址如下：https://developers.facebook.com/docs/graph-api

進入網頁之後點選「使用圖形 API」，就可以看到詳細的說明文件。



Chapter 7

重點回顧

1. 列表（list）、序對（或稱元組）（tuple）、集合（set）和字典（dict）是容器型態（Container type）。
2. 物件可區分可變（mutable）與不可變（immutable），不可變物件一旦創建後內容就不能再改變，容器物件只有 tuple 是不可變物件，其他三種都是可變物件。
3. list 串列資料型態，類似其他程式語言的陣列（Array）結構，它是一串由逗號分隔的值，用中括號 [] 包起來。
4. 串列可以是空串列，串列中元素可以包含不同的資料型態或是其它的子串列。
5. 串列 list 資料是可變物件，可以修改元素資料，也可以新增或移除元素。
6. del 敘述除了可以刪除串列中指定索引的元素外，也可以一次刪除切片運算子所指定的索引範圍。當不再使用串列這個變數時，也可以透過 del 敘述刪除串列變數。
7. 在 Python 中，串列中可以有串列，這種就稱為二維串列，要讀取二維串列的資料可以透過 for 迴圈。二維串列簡單來講就是串列中的元素是串列。
8. tuple 是有序物件，類似 list 串列，差別在於 tuple 是不可變物件，一旦建立之後，序對中的元素不能任意更改其位置與更改內容值。
9. 序對可以存放不同資料型態的元素。序對每個元素的索引編號左邊是由 [0] 開始，右邊則是由 [-1] 開始。
10. 序對內的元素不可以用 [] 運算子來改變元素的值。
11. 利用「+」運算子可以將兩個序對資料內容串接成一個新的序對，而「*」運算子可以複製序對的元素成多個。
12. tuple 是不可變物件，不可以直接修改 tuple 元素的值，因此 append()、insert() 等方法會改變元素個數或值的方法，都不可以使用。

13. Unpacking 不只限於 tuple，包括 list 跟 set 等序列物件，都可以用同樣的方式來指定變數，序列拆解的等號左邊的變數數量必須與等號右邊的序列元素數量相同。

14. dict 字典中的 key 必須是不可變的 (immutable) 的資料型態，例如數字、字串，而 value 就沒有限制，可以是數字、字串、list 串列、tuple 序對等等，資料之間必須以逗號 (,) 隔開。

15. 建立字典的方式除了利用大括號 {} 產生字典，也可以使用 dict() 函數，或是先建立空的字典，再利用 [] 運算子以鍵設值。

16. 正因為字典儲存資料是沒有順序性的，它是使用「鍵」查詢「值」，所以適用於序列型別的「切片」運算，在字典中就無法使用。

17. 字典中如果有相同的「鍵」卻被設定不同的「值」，則只有最後面的「鍵」所對應的「值」有效。

18. 如果要新增字典的鍵值對，只要加入新的鍵值即可。

19. 如果想刪除整個字典，則可以使用 del 指令。

20. 字典是一種無序的資料型態，所以不支援串接運算子 (+) 或重複運算子 (*)。

21. set 集合只有鍵 (key) 沒有值 (value)，類似數學裡的集合，可以進行聯集 (|)、交集 (&)、差集 (-) 與互斥或 (^) 等運算。

22. 使用 set() 方法建立集合，括號 () 裡只能有一個 iterable(迭代) 物件，也就是 str、list、tuple、dict 等等。

Chapter 7

學習評量

選擇題

(　　) 1. 關於容器資料型態的陳述，下列何者不正確？
(A) set(集合) 資料具有無序與互異的特性
(B) set 資料放置於中括號 [] 內
(C) tuple 是不可變物件
(D) dict（字典）是「鍵（key）」與「值（value）」對應的物件，是可變物件。

(　　) 2. 關於 list 串列資料型態的陳述，下列何者不正確？
(A) 類似其他程式語言的陣列（Array）結構
(B) 一串由逗號分隔的值
(C) 不可以包含不同的資料型態
(D) Python 提供生成式，是一種建立串列更快速彈性的作法。

(　　) 3. range(2,8) 會產生下列哪一個串列？
(A) [2,3,4,5,6,7]　　(B) [2,8]
(C) [2,3,4,5,6,7,8]　　(D) [1,2,3,4,5,6,7]。

(　　) 4. 關於 tuple 序對資料型態的陳述，下列何者不正確？
(A) 序對可以存放不同資料型態的元素
(B) tuple 是有序物件
(C) 序對內的元素可以用 [] 運算子來改變元素的值
(D)「*」運算子可以複製序對的元素成多個。

(　　) 5. 關於 dict 字典資料型態的陳述，下列何者不正確？
(A) 資料放置於大括號 {} 內
(B) 每一筆資料是一對 key:value
(C) 適用於序列型別的「切片」運算，在字典中就無法使用
(D) dict 字典中的 value 必須是不可變的（immutable）的資料型態。

填充題

1. 容器物件只有_____是不可變物件，其他三種都是可變物件。

2. 當不再使用串列這個變數時，也可以透過_____敘述刪除串列變數。

3. _____運算子可以將兩個序對資料內容串接成一個新的序對，而_____運算子可以複製序對的元素成多個。

4. dict 字典資料型態中的_____必須是不可變的（immutable）的資料型態，例如數字、字串，而_____就沒有限制。

5. 適用字典的處理方法_____方法會以 key 搜尋對應的 value。

簡答題

1. 請撰寫一程式統計檔案「twisters.txt」內容裡有哪些英文單字，各出現了幾次？請以 { 英文字：出現次數 } 顯示，輸出結果如下：

```
{'Peter': 4, 'Piper': 4, 'picked': 2, 'a': 3, 'peck': 4,
'of': 4, 'pickled': 4, 'peppers.': 1, 'Did': 1, 'pick': 1,
'peppers': 2, 'If': 1, 'Picked': 1, 'peppers,': 1, "Where's":
1, 'the': 1}
```

【提示】replace()：將不必要的字元（? 及 \n）移除

split()：切割字串

2. 請寫出下列程式執行後輸出結果。

```
A0 = {'a': 1, 'b': 3, 'c': 2, 'd': 5, 'e': 4}
A1 = {i:A0.get(i)*A0.get(i) for i in A0.keys()}
print(A1)
```

3. 請簡單比較 tuple、list、dict、set 四種容器型態的比較。

4. 下列的串列生成式的執行結果為何？

```
list1 =[i for i in range(4,11)]
print(list1)
```

5. 請寫出下段程式碼的輸出結果。

```
dic={'name':'Andy', 'age':18, 'city':'台北','city':'高雄'}
dic['name']= 'Tom'
dic['hobby']= '籃球'
print(dic)
```

Chapter

8 模組與套件

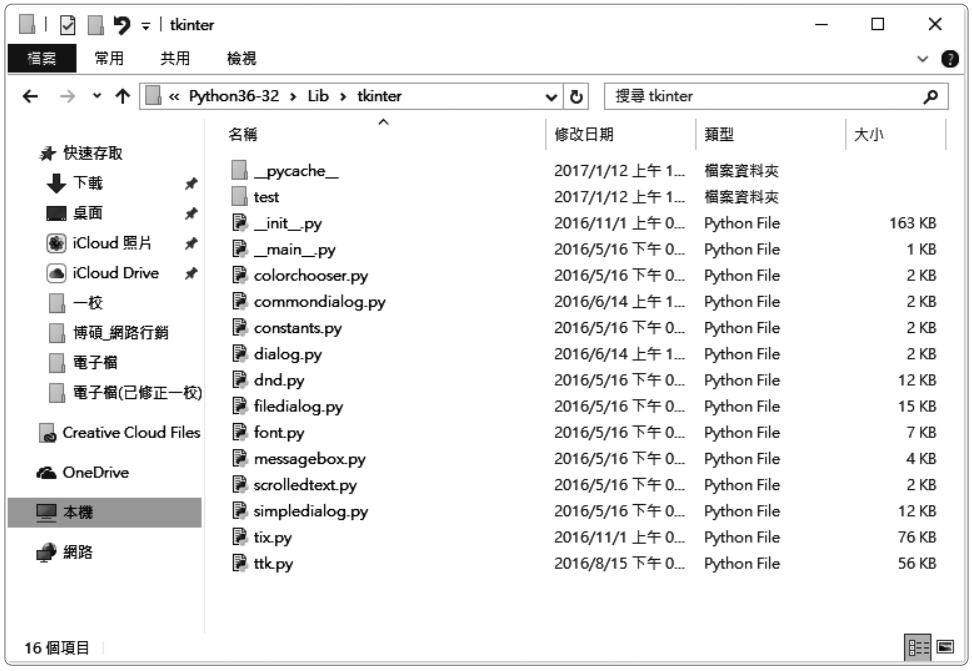
在 Python 的程式中可以輕易地加入許許多多由其他高手熱心設計的模組，使得許多複雜功能的程式，只要短短幾行程式就可以運作，需要的時候在加入自己的程式中就可以簡單地引入使用，節省許多自行開發的時間。本章將會跟各位討論 Python 的模組與套件簡介與特殊應用。

學習大綱

- 模組與套件的區別
- 將指定模組匯入
- 替模組取個別名
- 匯入套件中指定的類別或函數
- 常見內建模組
- 查看模組的路徑與檔名
- 認識第三方套件集中地
- 常見的第三方套件
- 套件線上說明文件

Python 自發展以來累積了相當完整的標準函數庫，這些標準函數庫裡包含相當多模組，所謂模組指的是已經寫好的 Python 檔案，也就是一個「*.py」檔案，模組中包含可執行的敘述和定義好的資料、函數或類別。一般來說，將多個模組組合在一起還能產生套件（Package）。如果說模組就是一個檔案，而套件就是一個目錄！目錄中除了包含檔案外，也可能包含其它的子目錄。

在 Python 語言中擁有「__init__.py」檔案的目錄就會被視為一個套件，套件包含了許多相關的模組或是子套件，因此可以這麼形容套件就是一種模組庫、函數庫。例如下圖為 tkinter 套件的檔案所在目錄，此目錄上可以找到一個名稱為「__init__.py」檔案，因此 tkinter 目錄會被 Python 視為一個套件。這裏所舉例的 tkinter 是一個跨平台的 GUI（圖形使用者介面）套件，能夠在 Windows、Mac、Linux、Unix 等作業平台開發 GUI 程式，在安裝 Python 時，預設值會一併將這個套件安裝進來。



Python 除了本身的內建套件外，也支援第三方公司所開發的套件，這使得 Python 功能更為強大，因此受到許多使用者的喜愛，本章將完整說明模組的用法以及如何自訂模組。

8-1 匯入模組

使用 Python 現成的模組，可以節省許多重複開發的工作，如果需要使用模組的時候，只要使用 import 指令就可以載入。import 敘述並沒有強制必須放在甚麼位置，只要放在呼叫函數或方法之前，習慣上會把 import 敘述放在程式最上方，匯入模組基本上有三種用法。

8-1-1 匯入整個模組

匯入整個模組，使用模組中的函數時要加上模組名稱，格式如下：

```
import 模組名稱
```

以 Python 內建的 math 模組為例，裡面定義了一些和數學有關的常數和函數，只要透過 import 指令將該模組匯入，就可以使用該模組所定義的數學函數。例如：

```
>>> import math
>>> math.floor(9.5)
9
>>> math.pow(2,3)
8.0
>>> math.gcd(12,18)
6
>>>
```

說明如下：

- 第 01 行：匯入數學模組。
- 第 02 行：floor 函數可以取小於參數的最大整數，例如此處小於 9.5 的最大整數為 9，所以傳回數值 9。
- 第 03 行：pow 函數需要兩個參數，第二個參數作為第一個參數的指數，回傳指數運算結果。例如本例中第一個參數為 2，第二個參數為 3，所以回傳 2 的 3 次方，其結果值為 8。
- 第 04 行：可以用來求取兩數間的最大公因數，12 及 18 的最大公因數為 6。

如果要一次匯入多個套件，則必須以逗點 (,) 隔開不同的套件名稱，語法如下：

```
import 套件名稱 1, 套件名稱 2, ...., 套件名稱 n
```

例如：同時匯入 Python 標準模組的數學和亂數模組。

```
import math, random
```

import 指令除了用來載入標準套件之外，也可以載入使用者自行定義的套件。但是如果系統沒有找到想要匯入的路徑，就會出現 ImportError 的錯誤。另外，如果要使用第三方公司所開發的套件，必須事先安裝，才可以使用「import」命令匯入。

以下範例使用 random 模組裡的 randint 函數來取得隨機整數以及 shuffle 函數將數列洗牌。

〈import 關鍵字匯入模組〉

檔名 import.py

```
01  import random
02
03  a = random.randint(0,99)   # 使用 randint 函數隨機取得整數
04  print(a)
05
06  items = [1, 2, 3, 4, 5]
07  random.shuffle(items)      # 使用 shuffle 函數將數列洗牌
08  print(items)
```

執行結果 »

```
47
[3, 1, 5, 2, 4]
```

程式解說

- 第 03 行：使用 randint 函數隨機取得整數。
- 第 06 行：給定 items 數列的初始值。
- 第 07 行：使用 shuffle 函數將數列洗牌。

8-1-2 替模組取個別名

如果模組的名稱過長，每次呼叫模組內的函數還要寫上這模組的名稱，確實會給程式設計人員帶來一些不必要的麻煩，而且也有可能在輸入模組名稱的過程中增加輸入錯誤的可能，為了改善這個問題，當遇到較長的套件名稱時，也可以另外取一個簡短、有意義又好輸入的別名，要將套件名稱指定別名的語法如下：

```
import 套件名稱 as 別名
```

當套件名稱有了別名之後，就可以利用「別名 . 函數名稱」的方式進行呼叫。例如：

```
import random as r    # 將 random 取別名為 r
r.randint(1,100)      # 以別名來進行呼叫
```

例如以下匯入 random 模組並指定別名為 r，使用 randint 函數時就必須用 r.randint()：

〈import...as 匯入模組指定別名〉

檔名 importAs.py

```
01  # -*- coding: utf-8 -*-
02
03  import random as r
04
05  a = r.randint(0,99)   # 使用 randint 函數隨機取得整數
06  print(a)
07
08  items = [1, 2, 3, 4, 5]
09  r.shuffle(items)       # 使用 shuffle 函數將數列洗牌
10  print(items)
```

執行結果 >>

```
18
[5, 2, 4, 1, 3]
```

</> 程式解說 >>

- 第 03 行：匯入 random 模組並指定別名為 r。
- 第 05 行：使用 randint 函數隨機取得整數。
- 第 08 行：給定 items 數列的初始值。
- 第 09 行：使用 shuffle 函數將數列洗牌。

8-1-3 只匯入特定的函數

如果只用到特定的函數，也可以將函數複製到目前模組，這時如果使用模組中的函數時不需要加上模組名稱，直接輸入函數名稱就可以呼叫該函數。格式如下：

```
from 模組名稱  import  函數名稱
```

另外也可以下達「from 套件名稱 import *」指令，表示匯入該套件的所有函數，例如以下語法就是指將 random 套件的所有函數匯入：

```
from random import *
```

因此上面談到的例子，就可以改寫成：

```
from random import *
randint(1, 10)
```

如果使用多個函數可以使用逗號 (,) 分隔，例如以下範例使用 random 模組裡的 randint 函數來取得隨機整數以及 shuffle 函數將數列洗牌。

〈from...import 匯入特定函數〉

檔名 fromImport.py

```
01  # -*- coding: utf-8 -*-
02
03  from random import randint,shuffle
04
05  a = randint(0,99)   # 使用 randint 函數隨機取得整數
06  print(a)
07
08  items = [1, 2, 3, 4, 5]
09  shuffle(items)       # 使用 shuffle 函數將數列洗牌
10  print(items)
```

執行結果 >>

```
78
[1, 3, 2, 5, 4]
```

</> 程式解說 >>

- 第 03 行：匯入 random 模組裡的 randint 函數來取得隨機整數以及 shuffle 函數將數列洗牌。
- 第 05 行：使用 randint 函數隨機取得整數。
- 第 08 行：給定 items 數列的初始值。
- 第 09 行：使用 shuffle 函數將數列洗牌。

Python 的標準函數庫裡面有非常多好用的模組，可以讓我們省下不少程式開發的時間，當程式裡會同時匯入多個模組，這時函數名稱就有可能會重複，好在 Python 提供命名空間（Namespace）機制，它就像是一個容器，將模組資源限定在模組的命名空間內，避免不同模組之間同名衝突的問題。

藉由實際的例子就更清楚 Namespace 機制的好處。以下範例使用了 random 模組的 randint 函數，當我們自訂了一個同名的函數，執行時會各自呼叫各自的函數執行，不用擔心執行會有衝突。

〈Namespace 機制〉

檔名 namespace.py

```
01  import random
02
03  def randint():
04      print(" 執行了自訂的 randint 函數 ")
05
06  a = random.randint(0,99)   # 使用 random 模組的 randint 函數
07  print(" 執行了 random 模組的 randint 函數：{}".format(a))
08
09  randint()                  # 呼叫自訂的 randint 函數
```

執行結果 »

```
執行了random模組的randint函數：71
執行了自訂的randint函數
```

</> 程式解說 »

- 第 03~04 行：自訂的 randint 函數。
- 第 06 行：使用 random 模組的 randint 函數
- 第 07 行：執行了 random 模組的 randint 函數。
- 第 09 行：呼叫自訂的 randint 函數。

因此匯入模組會比較建議使用第一種方法，除了不會有同名衝突之外，在函數前面加上模組名稱容易辨識函數來自哪個模組，讓程式更易讀。

第三種方法則是比較不建議的做法，除非階層較複雜的模組，可以適度使用，否則使用 from...import 方法當遇到同名函數時，Python 仍然能執行，後匯入的函數會先被呼叫，一不小心就容易導致 bug（臭蟲）發生。

8-2 自製模組

累積了大量寫程式的經驗之後，必定會有許多自己寫的函數，這些函數也可以自行整理成模組，等到下一個專案時直接匯入就可以重複使用這些函數。只要將函數放在 .py 文件，儲存之後就可以當作模組被匯入使用。

以下就實際來操作看看。請先建立一個 Python 文件，本例檔案命名為 moduleDiy.py，裡面寫好 SplitBill() 函數，程式碼如下：

〈自製模組〉

檔名 moduleDiy.py

```
01  def SplitBill(bill,split):
02      '''
03      函數功能：分帳
04      bill: 帳單金額
05      split: 人數
06      '''
07      tip = 0.1    #10% 服務費
08      total = bill + (bill * tip)
09      each_total = total / split
10      each_pay = round(each_total, 0)
11      return each_pay
```

寫好的 .py 文件儲存在與主文件相同資料夾就可以當成模組來使用了。我們建立一個主程式，把剛剛寫好的 moduleDiy 模組載入，就可以呼叫模組裡的函數，程式如下：

〈自製模組主程式〉

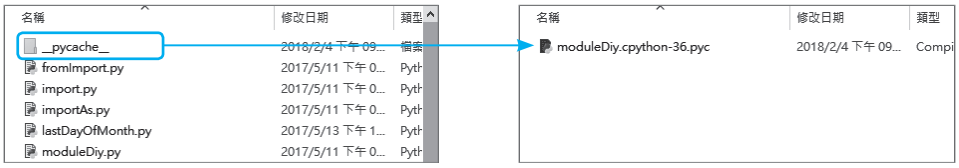
檔名 use_module.py

```
01  # -*- coding: utf-8 -*-
02  import moduleDiy
03
04  pay = moduleDiy.SplitBill(5000,3)     # 呼叫 SplitBill 函數
05  print(pay)
```

執行結果 »

```
moduleDiy
1833.0
```

執行完成之後，各位會發現在資料夾以下，多了一個「__pycache__」資料夾，這是因為第一次載入 moduleDiy.py 文件，Python 會將 .py 檔案編譯並儲存在「__pycache__」資料夾的 .pyc 檔案，下次執行主程式時，如果 moduleDiy.py 程式碼沒有變更，Python 就會跳過編譯，直接執行「__pycache__」資料夾的 .pyc 文件，加快程式執行的速度。



匯入所使用的 moduleDiy.py 程式碼，有可能需要修改或測試，如果每次都要在別的文件測試好再複製到 moduleDiy.py 文件未免也太麻煩了。我們還是可以直接在 moduleDiy.py 直接撰寫程式並測試，只要利用 Python 提供的 __name__ 屬性判斷程式是直接執行還是被 import 當成模組。下一小節就一起來看看好用的 __name__ 屬性。

Tips

Python 尋找模組時，會依照 sys.path 所定義的路徑來尋找，預設會先從目前工作資料夾尋找，再由環境變數 PythonPATH 指定的目錄或是 Python 安裝路徑尋找。

8-2-1 認識 Python 的 __name__ 屬性

Python 的文件都有 __name__ 屬性，當 Python 的 .py 裡的程式碼直接執行的時候，__name__ 屬性會被設定為「__main__」；如果文件被當成模組 import 時，屬性就會被定義為 .py 的文件名稱。

我們同樣使用上一小節的 moduleDiy.py 檔案當作範例來示範 __name__ 屬性的用法，請看以下程式碼。

〈__name__ 屬性的用法〉

檔名 moduleDiy_name.py

```
01  def SplitBill(bill,split):
02      '''
03      函數功能：分帳
04      bill: 帳單金額
05      split: 人數
06      '''
07      print(__name__)          # 輸出 __name__ 設定值
08
09      tip = 0.1  #10% 服務費
10      total = bill + (bill * tip)
11      each_total = total / split
12      each_pay = round(each_total, 0)
13      return each_pay
14
15
16  if __name__ == '__main__':   # 判斷 __name__
17      pay = SplitBill(5000,3)
18      print(pay)
```

執行結果 »

```
__main__
1833.0
```

當 moduleDiy_name.py 被當成模組使用時，執行結果如下：

```
moduleDiy_name
1833.0
```

當 moduleDiy_name.py 被當成模組使用時由於 __name__ 屬性並不等於 __main__，所以第 16 行的 if 判斷式並不會被執行。如此一來，自己寫的程式就可以被 import 使用，也可以直接拿來執行囉！

8-3 實用內建模組與線上輔助

Python 的標準函數庫提供了非常多不同用途的模組供程式設計人員使用，在此我們僅列出一些常用模組：

- math 模組提供了 C 函數庫中底層的浮點數運算的函數。
- random 模組提供了隨機選擇的工具。
- datetime 模組有許多操作日期以及時間的函數，並支援時區換算。
- 更具閱讀性的 pprint 套件。
- time 和 calendar 模組可以用於格式化日期和時間，也定義了一些與時間和日期相關的函數。
- os 模組是作業系統相關模組。
- sys 模組包含與 Python 直譯器相關的屬性與函數

這一小節將介紹四個實用的模組，包括 os 模組、sys 模組、random 模組以及 datetime 模組。接下來，我們就來認識 os 模組。

8-3-1 os 模組

功能包括查詢目前工作的資料夾路徑、建立資料夾、刪除資料夾等等。常用的函數如下表：

函數	說明	範例
os.getcwd()	取得當前工作路徑	os.getcwd()
os.listdir()	取得指定資料夾裡的檔名（包含資料夾）	os.listdir("D:/Python")
os.mkdir()	建立資料夾	os.mkdir("D:/Python/test")
os.rmdir()	刪除資料夾	os.rmdir("D:/Python/test")
os.rename()	更改資料夾名稱	os.rename("old", "new")
os.path.getsize()	取得文件大小	os.path.getsize("D:\Python")

〈os 模組常用函數練習〉

檔名 os.py

```
01  # -*- coding: utf-8 -*-
02  import os
03  CF=os.getcwd()
04
05  os.mkdir(CF+"/newFolder")    # 建立資料夾
06  os.mkdir(CF+"/newFolder1")   # 建立資料夾
07
08  os.rename(CF+"/newFolder1",CF+"/renewFolder") # 更名
09
10  CF_listdir=os.listdir( CF )
11
12  print(" 目前資料夾 :"+CF)
13  print(" 資料夾裡的文件與資料夾 :{}".format(CF_listdir))
```

執行結果 »

```
目前資料夾:D:\進行中書籍\台科大\Python_450頁版本\範例檔\ch08
資料夾裡的文件與資料夾:['fromImport.py', 'import.py', 'importAs.py',
'lastDayOfMonth.py', 'moduleDiy.py', 'moduleDiy_name.py', 'namespace.py',
'os.py', 'random.py', 'test_argv.py', 'use_module.py', '__pycache__']
```

</> 程式解說 »

- 第 03 行：取得目前工作路徑。
- 第 05~08 行：建立了兩個資料夾 newFolder 與 newFolder1，再利用 os.rename() 函數將 newFolder1 資料夾更名為 renewFolder。
- 第 10 行：利用 os.listdir() 函數將目前工作路徑的檔案與資料夾列出。

8-3-2 sys 模組

常用的功能如下：

函數	說明	範例
sys.argv	取得命令列參數	sys.argv[0]
sys.path	定義 Python 搜尋模組路徑	print(sys.path)
sys.version	取得目前 Python 的版本	print(sys.version)
sys.platform	取得作業系統平台	print(sys.platform)
sys.modules	取得所有載入的模組	print(sys.modules)
sys.exit(0)	終止程式	sys.exit(0)

sys.modules 是一組字典，包含所有被匯入過的套件模組，各位可以使用下列敘述清楚列出匯入的模組。

```
print('\n'.join(sys.modules))
```

如果各位是使用像 Anaconda 的 Python IDE 環境，sys.modules 列出的是所有已導入的模組。如果想知道某個模組的檔案路徑，也可以使用 sys.modules 來查詢，例如想查詢 random 模組的檔案路徑可以這樣表示：

```
print(sys.modules["random"])
```

執行之後就會顯示檔案路徑了。

```
<module 'random' from 'C:\\Users\\andu-Wu\\Anaconda3\\lib\\random.py'>
```

sys.argv 可以用來取得命令列引數，如果希望程式在命令列執行的時候可以接收使用者輸入的引數，就可以透過 sys.argv 方法來取得，它本身是 list 物件，索引 0 是執行的 .py 檔名，索引 1 之後的引數就是程式所需要的引數。

sys.exit(0) 函數則是用來終止程式，括號裡的數字是明確定義程式結束時的回傳值，通常會用回傳值為 0 表示是正常結束，非 0 代表程式異常結束。當執行 sys.exit(0) 時並不會立刻退出程式，而是會先觸發 SystemExit 的異常，我們可以捕捉這個異常，以便在離開程式之前做一些處置。

以下透過範例說明 sys.argv 以及 sys.exit(0) 的用法。

〈sys 模組常用函數練習 — 取得命令列引數〉

檔名 test_argv.py

```
01  # -*- coding: utf-8 -*-
02
03  import sys
04
05  print("sys.argv:{}".format(sys.argv))
06  print(" 文件名稱 {}".format(sys.argv[0]))
07  length = len(sys.argv)
08
09  if len(sys.argv) < 2:
```

```
10      try:
11          sys.exit(0)
12      except:
13          tp, val, tb=sys.exc_info()
14          print("exit!..{}:{}".format(tp,val))
15
16
17  for i in range(1,length):
18      n1 = sys.argv[i]
19      print( " 第 {} 個引數是 {}".format(i,n1))
```

執行結果 >>

```
(C:\Users\andu-Wu\Anaconda3) C:\Users\andu-Wu>python test_argv.py a b c
sys.argv:['test_argv.py', 'a', 'b', 'c']
文件名稱test_argv.py
第1個引數是a
第2個引數是b
第3個引數是c
```

</> 程式解說 >>

- 第 05~07 行：先列出完整的 sys.argv 的 list 元素，索引 0 是執行的 .py 檔名，索引 1 之後的引數就是程式所需要的引數。先利用 len() 函數取得 sys.argv 的元素個數，當元素小於 2 表示沒有輸入引數，如果有輸入引數就透過 for 迴圈顯示出來，由於第 1 個元素是檔名，所以 range() 是從 1 開始。
- 第 10~14 行：利用 try...except 接收 sys.exit() 發出的 SystemExit 異常，except(例外) 區塊是放置異常的處理敘述，except 如果不接任何例外型態（例如：IOError、ValueError），表示捕捉所有例外。我們利用 sys.exc_info() 來取得異常訊息，sys.exc_info() 會傳回三個值的元組（Tuple），分別是 type（異常類型）、value（異常參數）以及 traceback（回溯物件），其中 value 就是在 sys.exit() 括號內所設定的回傳值。以下是不加引數的執行結果：

```
(C:\Users\andu-Wu\Anaconda3) C:\Users\andu-Wu>python test_argv.py
sys.argv:['test_argv.py']
文件名稱test_argv.py
exit!..<class 'SystemExit'>:0
```

8-3-3 random 模組

隨機亂數是在程式設計中很常使用到的功能，特別是在製作遊戲的時候，像是撲克牌的發牌、猜數字遊戲等等，Python 提供了一個 random 模組，可以用來產生亂數，用法如下：

函數	說明	範例
random()	產生隨機浮點數 n，0 <= n < 1.0	random.random()
uniform()	產生指定範圍的隨機浮點數	random.uniform(5, 10)
randint()	產生指定範圍內的整數	random.randint(12, 20)
randrange()	從指定範圍內，按照遞增基數取一個隨機數	random.randrange(0, 10, 2)
choice()	從序列中取一個隨機數	random.choice(["A", "B", "C"])
shuffle(x)	將序列打亂	random.shuffle(["A", "B", "C"])
sample(population, k)	從序列或集合擷取 k 個不重複的元素	random.sample('ABCDEFG', 2)

random 模組裡的函數都很容易使用，比較特別的是 randrange() 與 shuffle() 函數。randrange() 函數是在指定的範圍內，依照遞增基數隨機取一個數，所以取出的數一定是遞增基數的倍數，相當於 range(start, stop[, step])，例如下式表示從 1-100 取一個奇數：

```
print ( random.randrange(1, 100, 2) )
```

下式則表示從 0-100 取一個隨機數。

```
print ( random.randrange(100) )
```

shuffle(x) 函數是直接將序列 x 打亂，並傳回 None，所以不能直接用 print() 函數來將它輸出。以下來看 random 模組的操作範例。

〈random 模組常用函數練習〉

檔名 random.py

```
01  # -*- coding: utf-8 -*-
02
03  import random
04
05  print( random.random() )
06  print( random.uniform(1, 10) )
07  print( random.randint(1, 10) )
08  print( random.randrange(0, 50, 5) )
09  print( random.choice(["真真", "小宇", "大凌"]) )
10
11  items = [1, 2, 3, 4, 5, 6, 7]
12  random.shuffle(items)
13  print( items )
14
15  print( random.sample('ABCDEFG', 2) )
```

執行結果 >>

```
0.6260053594739912
6.423779792416352
1
30
小宇
[3, 7, 5, 2, 1, 6, 4]
['A', 'G']
```

</> 程式解說 >>

- 第 05 行：產生隨機浮點數 n，0 <= n < 1.0。
- 第 06 行：產生 1-10 之間的隨機浮點數。
- 第 07 行：產生 1-10 之間的整數。
- 第 08 行：從 0-50 之間，按照遞增基數 5 取一個隨機數。
- 第 09 行：將 items 序列打亂。
- 第 12 行：從序列或集合擷取 2 個不重複的元素。

8-3-4 datetime 模組

日期與時間也是程式開發經常用到的功能，Python 提供了 time 模組以及 datetime 模組。datetime 模組除了顯示日期時間之外，還可以進行日期時間的運算以及進行格式化，常用的函數如下：

函數	說明	範例
datetime.date(年 , 月 , 日)	取得日期	datetime.date(2017,4,10)
datetime.time(時 , 分 , 秒)	取得時間	datetime.time(18, 30, 45)
datetime.datetime(年 , 月 , 日 [, 時 , 分 , 秒 , 微秒 , 時區])	取得日期時間	datetime.datetime(2017, 2, 4, 20, 44, 40)
datetime.timedelta()	取得時間間隔	datetime.timedelta(days=1)

datatime 模組可以單獨取得日期物件（datetime.date），也可以單獨取得時間物件（datetime.time）或者兩者一起使用（datetime. datetime）。

日期物件：datetime.date（year, month, day）

日期物件包含年、月、日。常用的方法如下：

date 方法	說明
datetime.date.today()	取得今天日期
datetime.datetime.now()	取得現在的日期時間
datetime.date.weekday()	取得星期數，星期一返回 0，星期天返回 6，例如： datetime.date(2017,5,10).weekday() 回傳 2
datetime.date. isoweekday()	取得星期數，星期一返回 1，星期天返回 7，例如： datetime.date(2017,5,10). isoweekday() 回傳 3
datetime.date. isocalendar()	返回 3 個元素的元組 tuple，(年 , 週數 , 星期數)，例如： datetime.date(2017,5,10).isocalendar() 回傳 (2017, 19, 3)

日期物件常用的屬性：

date 屬性	說明
datetime.date.min	取得支援的最小日期 (0001-01-01)
datetime.date.max	取得支援的最大日期 (9999-12-31)
datetime.date().year	取得年，例如 datetime.date(2017,5,10).year
datetime.date().month	取得月，例如 datetime.date(2017,5,10).month
datetime.date().day	取得日，例如 datetime.date(2017,5,10).day

時間物件：datetime.time(hour=0,minute=0,second=0,microsecond=0,tzinfo=None)

時間物件允許的值範圍如下：

0 <= hour < 24

0 <= minute < 60

0 <= second < 60

0 <= microsecond < 1000000

時間常用的屬性如下：

date 屬性	說明
datetime.time.min	取得支援的最小時間 (00:00:00)
datetime.time.max	取得支援的最大時間 (23:59:59.999999)
datetime.time().hour	取得時，例如： datetime.time(15,30,59).hour
datetime.time().minute	取得分，例如： datetime.time(15,30,59). minute
datetime.time().second	取得秒，例如： datetime.time(15,30,59). second
datetime.time().microsecond	取得微秒，例如： datetime.time(15,30,59, 26164).microsecond

另外，datatime 模組提供了 timedelta 物件可以計算兩個日期或時間的差距，例如想要取得明天的日期可以如下表示：

```
datetime.date.today() + datetime.timedelta(days=1)
```

timedelta 物件括號裡的參數可以是 days、seconds、microseconds、milliseconds、minutes、hours 以及 weeks，參數值可以是整數或浮點數，也可以是負數。

以下利用 datetime 模組讓使用者輸入年、月，判斷當月的最後一天的日期。

〈求某月最後一天的日期〉

檔名 lastDayOfMonth.py

```
01  # -*- coding: utf-8 -*-
02
03  import datetime
04
05  def lastDayOfMonth(y,m):
06      d=datetime.date(y,m,1)
07      yy = d.year
08      mm = d.month
09
10      if mm == 12 :
11          mm = 1
12          yy += 1
13      else:
14          mm += 1
15
16      return datetime.date(yy,mm,1)+ datetime.timedelta(days=-1)
17
18
19  if __name__ == '__main__':
20      isYear=int(input(" 請輸入年："))
21      isMonth=int(input(" 請輸入月："))
22      lastDay=lastDayOfMonth(isYear,isMonth)
23      print(lastDay)
```

執行結果 »

```
請輸入年：2018

請輸入月：12
2018-12-31
```

程式設計常常會需要知道某個月份的最後一天日期。每個月的最後一天並不是固定不變的，小月是 30 號、大月 31 號，二月份一般是 28 號，遇到閏年變 29 號。程式除了判斷大月小月，還得去判斷是否為閏年。

其實思考方式轉個彎，就會發現其實不難，既然算最後一天不容易，那麼我們就利用下個月的第一天減一天，同樣可以得到答案。

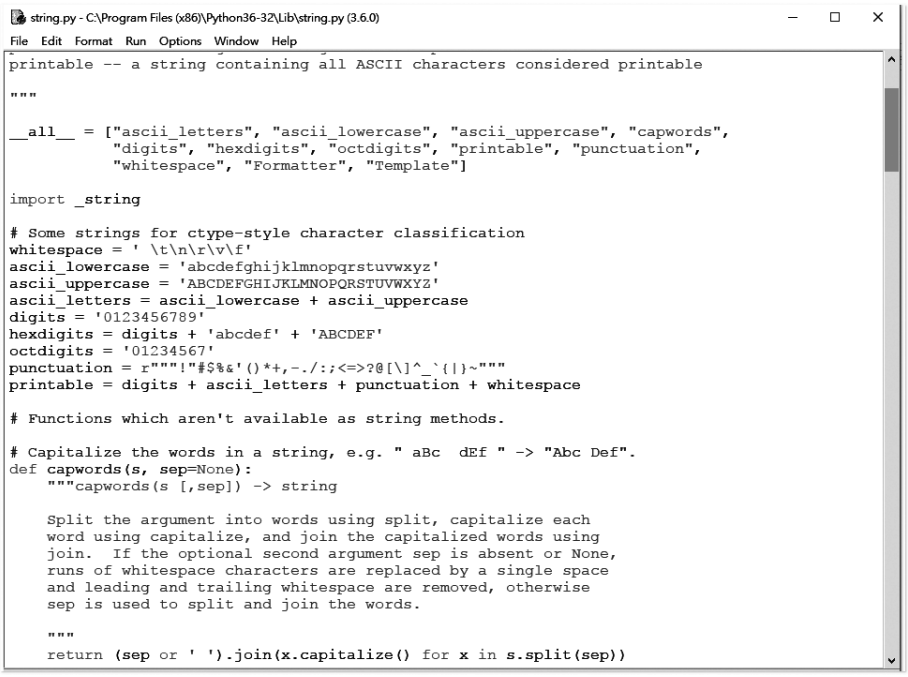
這樣的寫法要注意的是當使用者輸入的月份是 12 月，就必須將年份 +1，月份改為 1 月，再利用 datetime.timedelta(days=-1) 就能將日期減一天。

8-3-5 查看模組的路徑與檔名

如果要查詢模組的路徑與檔名可以利用 __file__ 屬性，例如以下語法就可以清楚知道 math 模組所在位置的路徑與檔名：

```
>>> import string
>>> print(string.__file__)
C:\Program Files (x86)\Python36-32\lib\string.py
```

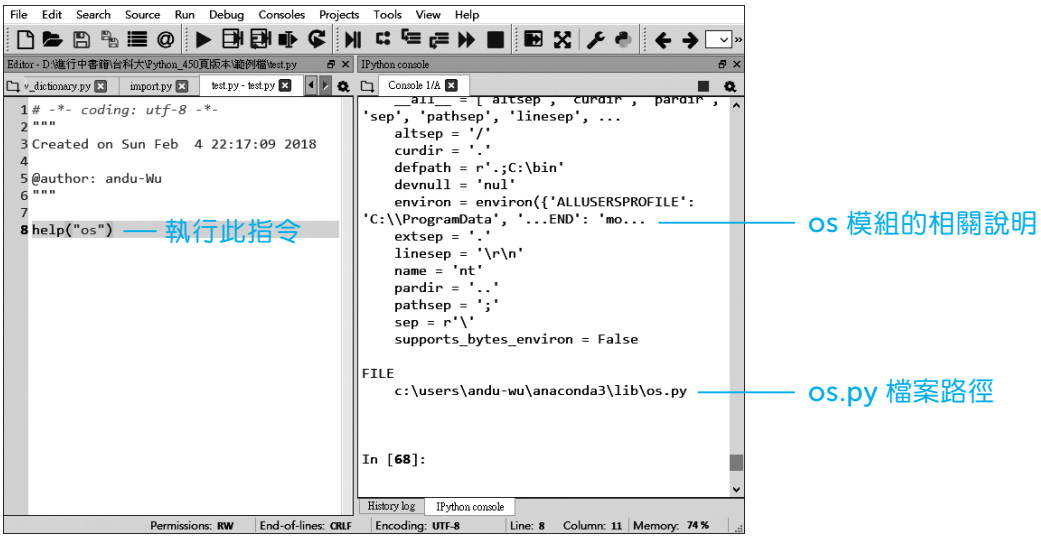
各位可以根據檔案所顯示的路徑與檔名，開始該檔案，就可以看到有關 string.py 檔案的相關內容，包括這個模組內所定義及函數，如下圖所示：



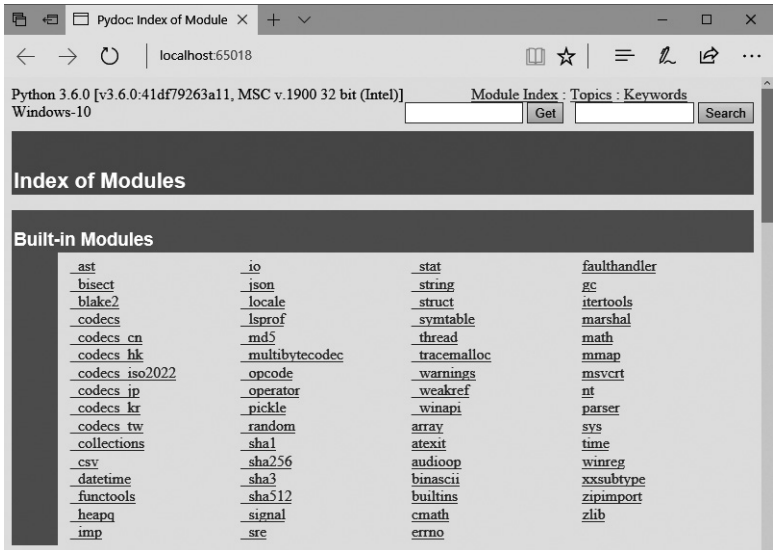

```
printable -- a string containing all ASCII characters considered printable

"""

__all__ = ["ascii_letters", "ascii_lowercase", "ascii_uppercase", "capwords",
           "digits", "hexdigits", "octdigits", "printable", "punctuation",
           "whitespace", "Formatter", "Template"]

import _string

# Some strings for ctype-style character classification
whitespace = ' \t\n\r\v\f'
ascii_lowercase = 'abcdefghijklmnopqrstuvwxyz'
ascii_uppercase = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
ascii_letters = ascii_lowercase + ascii_uppercase
digits = '0123456789'
hexdigits = digits + 'abcdef' + 'ABCDEF'
octdigits = '01234567'
punctuation = r"""!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""
printable = digits + ascii_letters + punctuation + whitespace

# Functions which aren't available as string methods.

# Capitalize the words in a string, e.g. " aBc  dEf " -> "Abc Def".
def capwords(s, sep=None):
    """capwords(s [,sep]) -> string

    Split the argument into words using split, capitalize each
    word using capitalize, and join the capitalized words using
    join.  If the optional second argument sep is absent or None,
    runs of whitespace characters are replaced by a single space
    and leading and trailing whitespace are removed, otherwise
    sep is used to split and join the words.

    """
    return (sep or ' ').join(x.capitalize() for x in s.split(sep))
```

8-3-6 套件線上說明文件

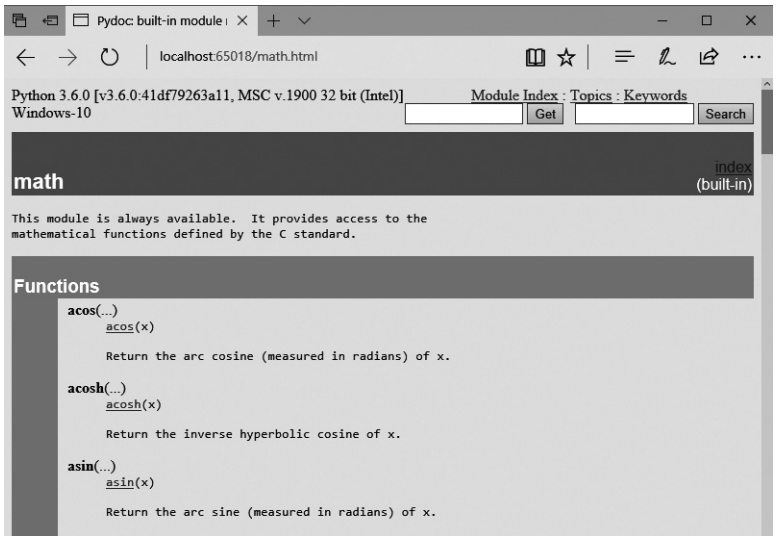
各個模組提供的函數很多，僅能針對常用的函數介紹，無法一一說明每個函數的用法。各位可以執行 help(模組名稱) 來查詢說明文件。例如查詢 os 模組，各位可以輸入 help("os")，執行之後就會看到 os 模組的說明，如下圖所示。



另外，也可以開啟模組的線上說明文件，可以執行「開始 /Python 3.6/ Python 3.6 Module Docs」指令，查看 Python 所提供模組的線上說明文件，每一個模組都是一個 Python 檔案，在該檔案內定義了該模組相關功能的資料、屬性、函數或類別，例如 time 模組提供了一些和時間有關的常數和數學函數，下圖為 Python 內建模組的線上索引系統，線上文件中可以有效幫助使用者了解各種模組的使用說明。



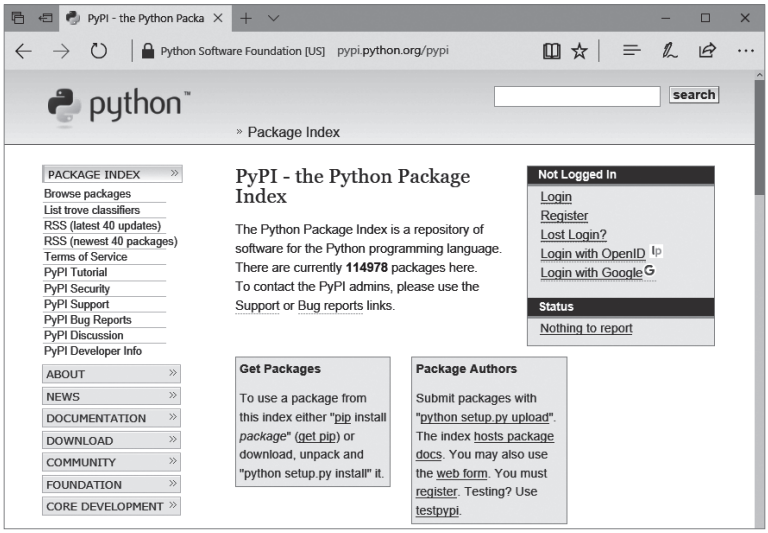
如果想進一步查看 math 模組的使用方法，可以直接在上圖中按一下「math」文件超連結，就可以開啟如下的線上說明文件，在這線上文件中可以清楚知道該模組內各種函數的使用方式及所定義的資料。



8-4 第三方套件

有別於內建的模組與套件，在安裝 Python 時會一併安裝，所謂第三方套件就是指在安裝 Python 時不會被安裝進來，必須另外安裝的套件。

PyPI 是 Python Package Index 的縮寫，這是 Python 的第三方套件集中地，目前蒐集了超過數萬個的第三方套件，下圖為 the Python Package Index 網站，在這個網站可以說是第三方套件的最大本營，幾乎所有能想像到的功能，都可以在這找到合適的套件。網址：https://pypi.Python.org/pypi



8-4-1 利用 pip 管理程式安裝套件

Python 中提供一種方便強大的套件管理程式 -pip，它集合下載、安裝、升級、管理、移除套件等功能，只要透過簡單的指令和步驟，可以幫助我們從 PyPI 下載想要的套件，並作妥善的安裝與管理。藉由統一的管理，除了事半功倍，也避免了手動執行上述任務會發生的種種錯誤。

如果是在 Windows 平台安裝 Python 3.6 的版本，可以在安裝資料夾內的 Scripts 資料夾內找到 pip 程式。但是如果各位的電腦沒有安裝 pip 程式，也可以參考 pip 的官方文件網站：
http://pip.readthedocs.org/en/latest/installing.html

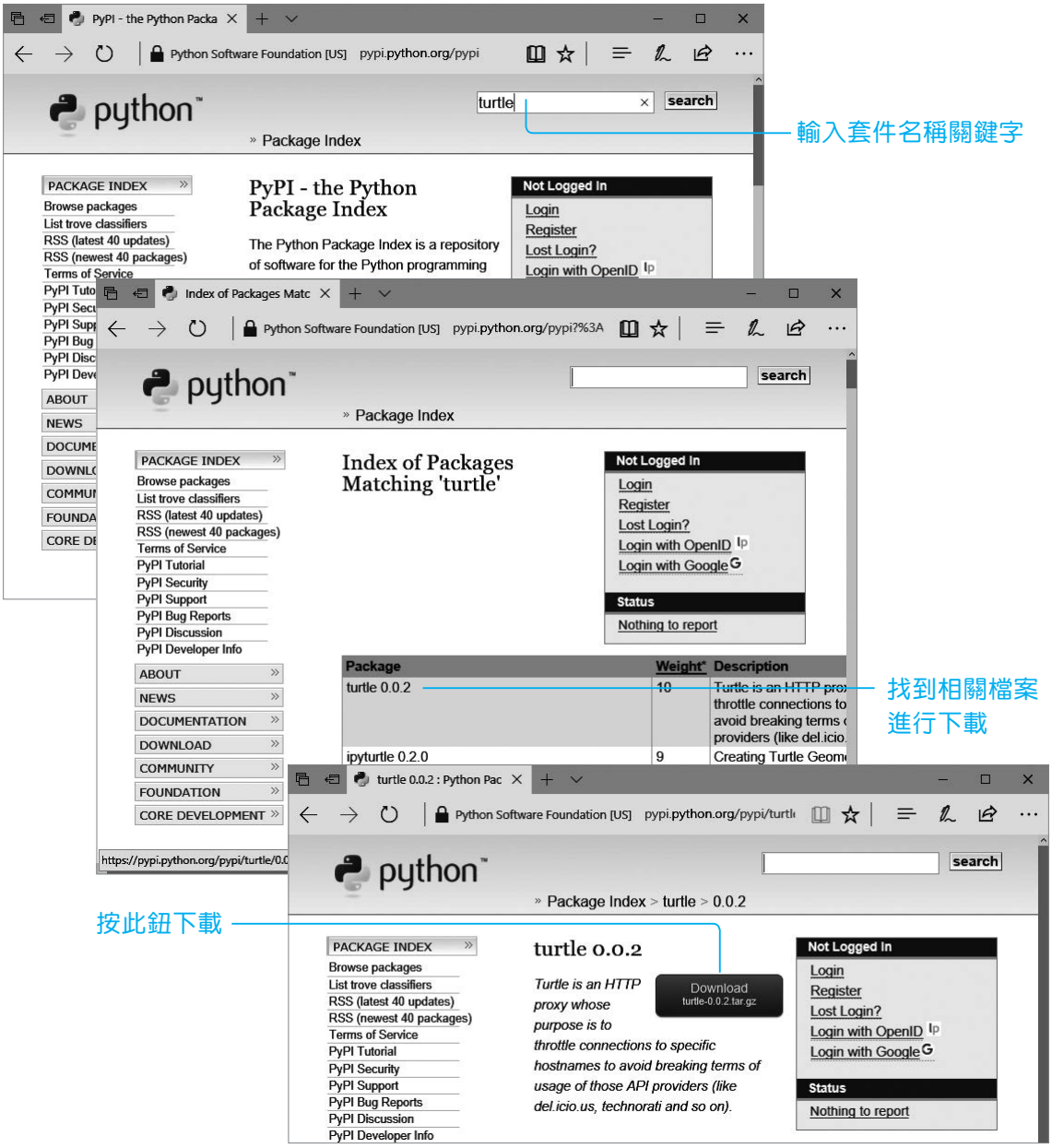
或直接參照以下的步驟說明：
http://pip.readthedocs.org/en/latest/installing.html

在 install pip 章節可以找到一個名為 get-pip.py 的檔案，並下載該檔案，取得 get-pip.py 檔案後，請於「命令提示字元」視窗，切換到 get-pip.py 的目錄並輸入如下的指令，就可以輕鬆完成安裝 pip 程式的工作，執行：

```
>Python get-pip.py
```

8-4-2 從 PyPI 網站安裝套件

除了上述的方式外，各位也可以直接連上 https://pypi.python.org/pypi 第三方套件集中地的網站，並於該網站中輸入要搜尋的套件名稱，例如：XXX.py，在找到要安裝的套件後，就可以將該檔案下載並安裝到自己的電腦中。完整的操作過程，請參閱以下的示範步驟：



找到相關檔案後，就可以將該檔案下載，再利用 pip 工具將其安裝在電腦即可。

8-5 上機實作演練－樂透隨機投注暨對獎程式

本章學會了 Python 的 Random 模組，我們就利用它來實作一個樂透投注遊戲。

8-5-1 程式範例描述

模擬今彩 539 的樂透投注遊戲，投注者必須從 1~39 的號碼任選 5 個不同的號碼進行投注。利用程式產生開獎號碼，並與投注者 5 個號碼比對，看看對中幾個號碼。

輸入說明

投注者輸入 5 個不重複的號碼，每個號碼以逗號 (,) 隔開。由程式隨機產生 5 個開獎號碼。

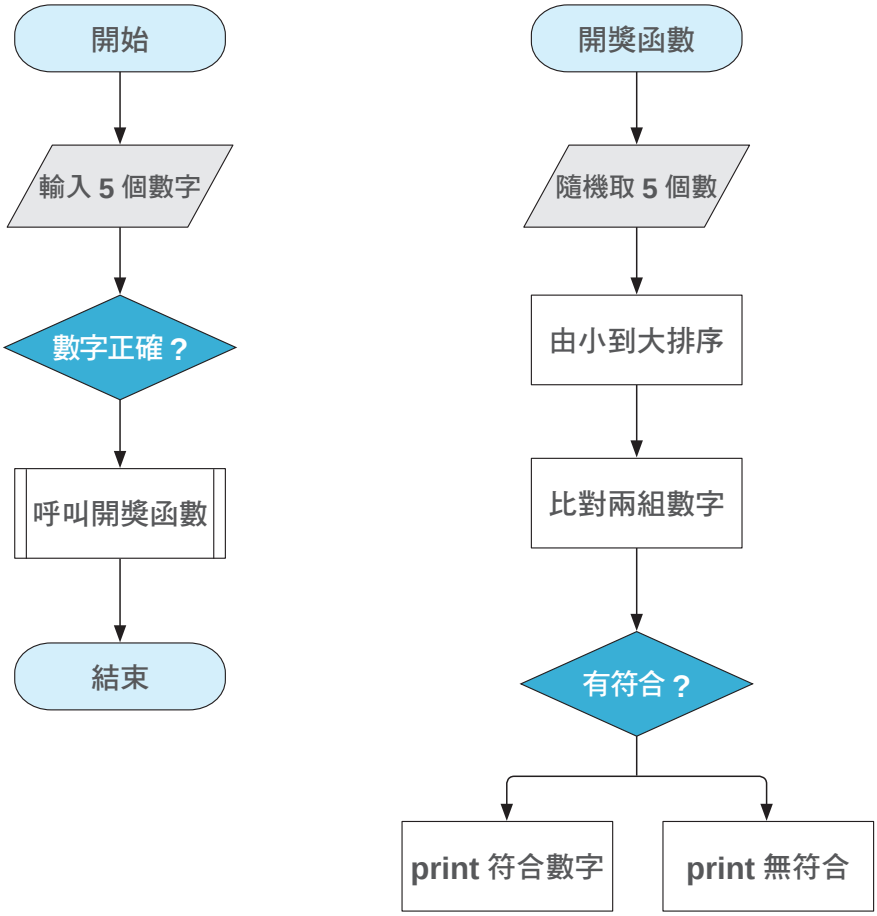
輸入範例

輸出需包含開獎號碼的開出順序、大小順序，投注者選的號碼，符合的號碼；如果沒有符合則顯示「沒有符合！」。

畫面如下：

```
請從1~39個號碼任選5個不同號碼，每個號碼請以逗號(,)隔開：3,6,7,8,9
開出順序：[6, 3, 36, 18, 31]
大小順序：[3, 6, 18, 31, 36]
您選的號碼：[3, 6, 7, 8, 9]
符合：{3, 6}
```

▶ 流程圖



8-5-2 程式碼說明

這個範例主要可以區分為幾個部分：

1. 投注者輸入 5 個數字，並檢查是否符合規則。
2. 隨機產生 5 個不重複開獎數字。
3. 比對兩組數字是否符合。

首先來看投注者輸入 5 個數字的程式格式：

```
while True:
    try:
        user_number = input("請輸入5個 同號碼，每個號碼請以逗號(,)隔開:")

            # 檢查規則
            1. 數字是否是 5 個
            2. 數字是否重複
            3. 數字是否在 1~39 範圍

        break           # 離開迴圈
    except:
        例外處理
```

使用者輸入數字時有可能不符合規則，所以這裡使用 while 迴圈直到數字完全符合規則才跳離迴圈，while 迴圈的離開條件式設為 True，表示迴圈會不斷執行，直到遇到 break 才會跳離迴圈。

使用者輸入數字時出錯的狀況難以預料，例如輸入英文字母、特殊符號或是空白等等，這裡先利用 try...except 來捕捉異常。

由於使用者輸入的數字是以逗號 (,) 分隔，所以要先將字串用 split() 方法分割，再利用 for 迴圈比對數字是否符合規則，程式如下所示：

```
n1=[]                               # 宣告空串列
for n in user_number.split(","):
    n = int(n)
    if n in n1:
        print(" 重複輸入，",end="")
        raise ValueError            # 觸發異常
    elif n not in range(1,40):
        print(" 超出範圍！數字必須是 1~39，",end="")
        raise ValueError            # 觸發異常
    else:
        n1.append(n)                # 將數字加入串列
    lottoCheck(n1)                  # 呼叫開獎程式
    break                           # 跳離迴圈
```

上面程式可以看到當數字不符合題目的規則時，就利用 raise 敘述來觸發異常。當數字完全正確時就可以呼叫開獎函數 (lottoCheck) 來對獎了。這裡我們只讓程式執行一次所以加上 break 敘述來跳離迴圈；如果各位想要讓程式不斷執行只要將 break 拿掉就可以了，不過別忘了要另外再加上跳離迴圈的程式，否則程式就會不斷執行。

lottoCheck 函數一開始先呼叫 generate_num 函數隨機產生 5 個開獎號碼，我們先來看產生 generate_num 函數的程式碼：

```
def generate_num():
    auto_num = []                    # 宣告空串列
    while len(auto_num)<5:
        x = random.randint(1, 40)    # 隨機取一個數字
        if x not in auto_num:        # 檢查數字是否重複
            auto_num.append(x)
    return auto_num
```

上面敘述利用 while 迴圈來產生 5 個數字，利用 len() 函數來檢查串列的元素個數，當小於 6 時迴圈就會不斷循環，直到滿 5 個數字就跳離迴圈。

當投注者與開獎數字都成功產生之後，就必須比對兩組數字是否有符合的數字：

```
def lottoCheck(a):
    b=generate_num()                 # 呼叫 generate_num 函數
    b_sort=sorted(b)                 # 將數字排序
    print(" 開出順序：{}".format(b))
    print(" 大小順序：{}".format(b_sort))
    print(" 您選的號碼：{}".format(sorted(a)))
    ans = set(a) & set(b_sort)       # 比對兩組數字
    if len(ans):
        print(" 符合：{}".format(ans))
    else:
        print(" 沒有符合 !")
```

比對兩組數字是否符合的程式重點在以下這一行：

```
ans = set(a) & set(b_sort)
```

set() 函數是將串列轉成集合 (set)，集合可以做聯集、交集、差集的運算。想想看，比對元素同時存在集合 A 也存在集合 B，應該執行哪一種運算？沒錯！只要做交集 (&) 運算就可以輕鬆找出符合的數字了。

以下列出完整程式碼供各位參考。

〈樂透系統發財夢〉

檔名 review_Lottery.py

```
01                  # -*- coding: utf-8 -*-
02 """
03 樂透開獎與對獎程式
04 題目要求：
05 1. 讓投注者輸入 5 個不重複的號碼，每個號碼以逗號 (,) 隔開
06 2. 隨機產生開獎號碼。
07 3. 計算投注者的 5 個號碼有幾個號碼開出。
08 4. 輸出需包含開獎號碼的開出順序、大小順序，投注者選的號碼以及符合的號
   碼；
09 如果沒有符合則顯示「沒有符合！」。
10 """
11
12 import random
13
14 def generate_num():
15     auto_num = []
16     while len(auto_num)<5:
17         x = random.randint(1, 40)
18         if x not in auto_num:
19             auto_num.append(x)
20     return auto_num
21
22 def lottoCheck(a):
23     b=generate_num()
```

```
24      b_sort=sorted(b)
25      print(" 開出順序：{}".format(b))
24      b_sort=sorted(b)
25      print(" 開出順序：{}".format(b))
26      print(" 大小順序：{}".format(b_sort))
27      print(" 您選的號碼：{}".format(sorted(a)))
28      ans = set(a) & set(b_sort)
29      if len(ans):
30          print(" 符合：{}".format(ans))
31      else:
32          print(" 沒有符合 !")
33
34
35
36  if __name__ == "__main__":
37      while True:
38          try:
39              user_number = input(" 請從 1~39 個號碼任選 5 個 同號碼，
                每個號碼請以逗號 (,) 隔開：")
40
41              if user_number.count(",")<4:
42                  print(" 號碼不足，",end="")
43                  raise ValueError
44              else:
45                  n1=[]
46                  for n in user_number.split(","):
47                      n = int(n)
48                      if n in n1:
49                          print(" 重複輸入，",end="")
50                          raise ValueError
51                      elif n not in range(1,40):
52                          print("超出範圍！數字必須是1~39，",end="")
53                          raise ValueError
54                      else:
55                          n1.append(n)
56                  lottoCheck(n1)
57                  break
58          except ValueError:
59              print(" 請再輸入一次 !")
```

重點回顧

1. 模組指的是已經寫好的 Python 檔案，模組中包含可執行的敘述和定義好的資料、函數或類別。
2. 在 Python 語言中擁有「__init__.py」檔案的目錄就會被視為一個套件。
3. 若是需要使用模組的時候，只要使用 import 指令就可以載入。
4. 當遇到較長的套件名稱時，也可以另外取一個簡短、有意義又好輸入的別名。
5. 「from 套件名稱 import *」指令，表示匯入該套件的所有函數。
6. 如果使用多個函數可以使用逗號 (,) 分隔。
7. Python 提供命名空間 (Namespace) 機制，將模組資源限定在模組的命名空間內，避免不同模組之間同名衝突的問題。
8. Python 尋找模組時，會依照 sys.path 所定義的路徑來尋找，預設會先從目前工作資料夾尋找，再由環境變數 PythonPATH 指定的目錄或是 Python 安裝路徑尋找。
9. 當 Python 的 .py 裡的程式碼直接執行的時候，__name__ 屬性會被設定為「__main__」；如果文件被當成模組 import 時，屬性就會被定義為 .py 的文件名稱。
10. datetime 模組有許多操作日期以及時間的函數，並支援時區換算。
11. os 模組功能包括查詢目前工作的資料夾路徑、建立資料夾、刪除資料夾等等。
12. sys.modules 是一組字典，包含所有被匯入過的套件模組。
13. 如果希望程式在命令列執行的時候可以接收使用者輸入的引數，就可以透過 sys.argv 方法來取得。
14. except（例外）除了由直譯器自動觸發，我們也可以利用 raise 敘述來觸發例外。

15. randrange() 函數是在指定的範圍內，依照遞增基數隨機取一個數。
16. datatime 模組可以單獨取得日期物件（datetime.date），也可以單獨取得時間物件（datetime.time）或者兩者一起使用（datetime.datetime）。
17. 如果要查詢模組的路徑與檔名可以利用 __file__ 屬性。
18. 可以執行 help(模組名稱) 來查詢說明文件。
19. PyPI 是 Python Package Index 的縮寫，這是 Python 的第三方套件集中地。
20. 套件管理程式 -pip，它集合下載、安裝、升級、管理、移除套件等功能。

學習評量

選擇題

(　　) 1. 關於模組與套件的說明，下列何者有誤？
(A) 模組是一個「*.py」檔案
(B) 多個模組組合在一起還能產生套件
(C) 擁有「__init__.py」檔案的目錄就會被視為一個套件
(D) 如果要一次匯入多個套件，則必須以分號 (;) 隔開不同的套件名稱。

(　　) 2. 如何避免不同模組之間同名衝突的問題？
(A) 將模組資源限定在模組的命名空間內
(B) 絕對不能有同名的模組
(C) 將模組以大小寫區分
(D) 下達不同的匯入指令。

填充題

1. 將多個模組組合在一起還能產生_____。

2. 在 Python 語言中擁有「_____」檔案的目錄就會被視為一個套件。

3. 若是需要使用模組的時候，只要使用_____指令就可以載入。

4. 當套件名稱有了別名之後，就可以利用_____的方式進行呼叫。

5. 當 Python 的 .py 裡的程式碼直接執行的時候，__name__ 屬性會被設定為「_____」。

簡答題

1. 請撰寫一函數具有年、月、日三個參數，例如 isVaildDate(yy,mm,dd)，檢查代入的年月日是否為合法日期，如果是就輸出此日期，否則輸出「日期錯誤」。

   例如：

```
isVaildDate(2017,3,30) ，輸出「2017-03-30」
isVaildDate(2017,2,30) ，輸出「日期錯誤」
```

2. 如何才能一次匯入多個套件？

3. 要將套件名稱指定別名的語法為何？

4. Python 的標準函數庫裡面非常多好用的模組，這些模組內的函數難免會有重複，試用 Python 如何避免不同模組之間同名衝突的問題。

5. 試舉出至少五種 Python 的常用模組。

Chapter

9 檔案與資料流

當程式執行完畢之後，所有儲存在記憶體的資料都會消失，這時如果需要將執行結果儲存在不會揮發的儲存媒體上（如磁碟等），必須透過檔案模式來加以保存。本章將會跟各位討論 Python 的檔案與資料流的應用與透過檔案的操作存取資料方式。

學習大綱

- 認識檔案
- 檔案的種類
- 檔案與目錄
- 絕對路徑與相對路徑
- 檔案的寫入
- 檔案的讀取
- 檔案的複製
- 二進位資料

檔案（File）是電腦中數位資料的集合，是一種儲存資料的單位，也是在磁碟機上處理資料的重要單位，這些資料以位元組的方式儲存，可以是一份報告、一張圖片或一個執行程式，並且包括了資料檔、程式檔或可執行檔等格式，利用檔案存取資料，也是處理資料中非常重要的一環。

9-1 認識檔案

在程式的運作的過程中，所有的資料都是儲存在記憶體中，一旦結束程式再重新執行，之前輸入的資料就會全部消失。程式執行的過程中，如果要將資料計算而得的資料永久保存下來，就必須將資料寫入檔案並存放在非揮發性（Non-volatile）的儲存媒介中，例如硬碟。每個檔案都必須有檔案名稱 (File Name)，檔名可區分成「主檔名」與「副檔名」，中間以句點 (.) 做分隔，「副檔名」的功能在於記錄檔案的類型，例如 .cpp 表示 C++ 的原始程式檔。.py 表示 Python 的原始程式檔。透過這樣的命名方式可以清楚分辨其檔案名稱與檔案類型。如下所示：

主檔名 . 副檔名

9-1-1 檔案的種類

檔案如果依儲存方式來分類，可以分文字檔（text file）與二進位檔（binary file）兩種。分別說明如下：

文字檔

文字檔是以字元編碼的方式進行儲存，在 Windows 作業系統的記事本程式中則預設以 ASCII 編碼來儲存文字檔，每個字元佔有 1 位元組。例如在文字檔中存入 10 位數整數 1234567890，由於是以字元循序存入，所以總共需要 10 位元組來儲存。

二進位檔

所謂二進位檔，就是以二進位格式儲存，也就是說將記憶體中資料原封不動儲存至檔案之中，這種儲存方式適用於非字元為主的資料。如果以記事本程式開啟，各位只會看到一堆亂碼喔！

其實除了以字元為主的文字檔外，所有的資料都可以說是二進位檔，例如編譯過後的程式檔案、圖片或影片檔案等。二進位檔的最大優點在於存取速度快、佔用空間小以及可隨機存取，在資料庫應用上較文字檔案來得適合。

9-1-2 認識檔案與目錄

在電腦或相關行動裝置中，一個「目錄」或「資料夾」裡面儲存著一組檔案和其它一些目錄（資料夾）。一個檔案系統可能會包含成千上萬個檔案及目錄（資料夾），並以一種有組織的方式進行管理，以達到有效儲存檔案的目的。在一個目錄（資料夾）中的另一個目錄（資料夾）被稱作它的子目錄（子資料夾）。這些目錄（資料夾）就構成了層次（hierarchy）或樹形結構。如下圖的檔案總管：

也就是說，檔案系統是一種儲存和組織電腦資料的方法，透過檔案系統的管理工具，例如 Windows 作業系統中的「檔案總管」，可以幫助使用者在存取檔案時，變得更為直覺容易。事實上，在檔案系統使用檔案和目錄（或稱資料夾）的抽象邏輯概念，代替了硬碟和光碟等物理裝置使用資料塊的概念，使用者儲存資料時不必關心資料儲存在硬碟（或者光碟）的實際位址及佔用多少位址空間，只需要記住這個檔案的所屬目錄和檔案名稱。

不同的作業系統所採用的檔案系統會有所不同，例如早期 MS-DOS 採用的檔案系統為 FAT（File Allocation Table），而現今較常見的作業系統 Windows 7/8/10 的檔案系統則有 FAT32、NTFS（New Technology File System）或 exFAT（Extended File Allocation Table）。

9-1-3 絕對路徑與相對路徑

首先說明「絕對路徑」和「相對路徑」的差異。簡單的說，「絕對路徑」指的是一個絕對的位置，並不會隨著現在目錄的改變而改變。例如：

```
C:\Windows\system\
```

「相對路徑」就是相對於現在目錄的路徑表示法，因此「相對路徑」所指到的檔案或目錄，會隨著現在目錄的不同而改變。通常我們用「.」代表現在目錄，而用「..」代表上一層目錄。

9-2 檔案的讀與寫

當程式執行完畢之後，所有儲存在記憶體的資料都會消失，這個時候如果需要將執行結果儲存在不會揮發的儲存媒體上（如硬碟等），就必須透過檔案存取模式來將檔案儲存在硬碟中。

9-2-1 檔案的寫入

Python 在處理檔案的讀取與寫入都是透過檔案物件，因此，無論是進行檔案的寫入或讀取工作，第一項工作就是利用 Python 的 open() 內建函數建立檔案物件，所謂檔案物件（File Object）就是一個提供存取的介面，它並非實際的檔案。當開啟檔案之後，就必須透過「檔案物件」做讀（Read）或寫（Write）的動作。

open() 函數語法如下：

```
open(file, mode)
```

- **file**：以字串來指定想要開啟檔案的路徑和檔案名稱。
- **mode**：以字串指定開啟檔案的存取模式，下表為可以設定的存取模式：

mode	說明
"r"	讀取模式（預設值）
"w"	寫入模式，建立新檔或覆蓋舊檔（覆蓋舊有資料）
"a"	附加（寫入）模式，建立新檔或附加於舊檔尾端
"x"	寫入模式，檔案不存在建立新檔，檔案存在則有錯誤
"t"	文字模式（預設）
"b"	二進位模式
"r+"	更新模式，可讀可寫，檔案必須存在，從檔案開頭做讀寫
"w+"	更新模式，可讀可寫，建新檔或覆蓋舊檔內容，從檔案開頭做讀寫
a+	更新模式，可讀可寫，建立新檔或從舊檔尾端做讀寫

如果利用 open() 建立檔案物件的工作成功，就會傳回檔案物件；但是如果建立失敗，就會發生錯誤。當以讀取模式 "r" 開啟檔案，如果該檔案不存在，就會發生錯誤。

例如底下的程式敘述是以讀取模式開啟一個名稱為 C:\test.txt 檔案，如果檔案的路徑找不到這個檔案，就會發生 FileNotFoundError 的錯誤訊息，這個訊息告知使用者所開啟的檔案或目錄不存在：

```
file1=open("C:\\test.txt","r")
FileNotFoundError: [Errno 2] No such file or directory: 'C:\\test.
txt'
```

剛才談到的情形是指以讀取模式開啟檔案，當檔案不存在時會發生錯誤，但是如果以寫入模式開啟檔案，第一次開啟該檔案，該檔案不會存在，這個時候系統就會自動以該名稱建立新檔，不會發生類似讀取模式發生找不到檔案的錯誤。

例如以下的程式敘述是以寫入模式開啟一個名稱為 C:\test.txt 檔案，如果在該檔案路徑找不到這個檔案，就會以該名稱建立檔案，並會建立一個檔案物件，再指派給變數 file1：

```
file1=open("C:\\test.txt","w")
```

請各位要特別留意，使用 open() 函數開啟檔案時，所指定的檔案路徑必須以跳脫字元 \\ 來表示 \，例如：

```
>>> file1=open("C:\\lab\\test.txt"."r")
```

如果各位覺得上述的路徑表示方式不實用，也可以在絕對路徑前面加 r，來告知編譯器系統接著所使用路徑的字串是原始字串，如此一來，原先用 \\ 來表示 \ 的表示方式，就可以簡化如下的表示方式：

```
>>> file1=open(r"C:\lab\test.txt"."r")
```

也就是說，對於文字檔案而言，要將資料寫入檔案中必須事先以 open() 方法建立新檔，再使用檔案物件所提供的 write() 方法將文字寫入檔案，最後再以 close() 方法關閉檔案。

請注意！寫入檔案時會從目前檔案指標的所在位置開始，所以寫入檔案時，必須指定存取模式，所謂檔案指標是用來記錄目前檔案寫入或讀取到哪一個位置。

〈使用 open() 方法開啟檔案 I〉

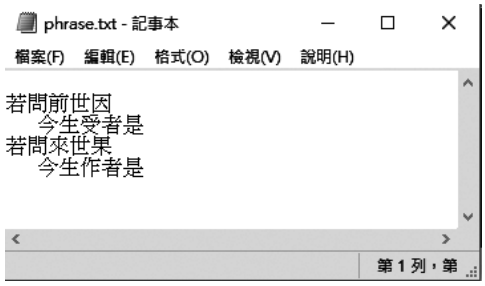
檔名 openfile.py

```
01 yeats = '''
02 若問前世因
03     今生受者是
04 若問來世果
05     今生作者是
06 '''
07 # 建立新檔，以文字模式寫入
08 fn = open('phrase.txt', 'wt')
09 fn.write(yeats)     # 將字串寫入檔案
10 fn.close()          # 關閉檔案
```

程式解說

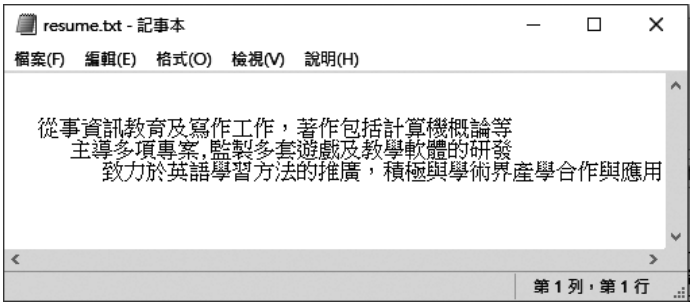
- 第 01~06 行：程式碼先以長文字（使用 3 個單或雙引號）建立字串內容。接著使用 open() 函數時必須指定給檔案物件（File Object）變數 fn 做儲存。第一個參數是欲建立的檔案名稱，此處使用文字檔，第二個參數 mode 為 wt (以字串方式) 表示「以文字寫入」。
- 第 09 行：以 fn (檔案物件) 呼叫 write() 方法並傳入參數。
- 第 10 行：以 fn 呼叫 close() 方法來關閉檔案，如此才能將位於緩衝區的內容全部寫入檔案；未使用此方法會讓建立的檔案是空的。

我們利用 Windows 檔案總管找到這個新建的檔案，再以記事本開啟該檔案，可以看到所寫入的文字，會依所指定的字串呈現方式，如右圖所示。



隨堂練習

請撰寫一個 Python 程式，該程式功能可以在所在位置的上一層目錄下建立一個名稱為 resume.txt 的文字檔案，其內容如下圖所示：



〈使用 open() 方法開啟檔案 II〉

檔名 openfile1.py

```
content = '''

    從事資訊教育及寫作工作，著作包括計算機概論等
        主導多項專案，監製多套遊戲及教學軟體的研發
            致力於英語學習方法的推廣，積極與學術界產學合作與應用

'''
# 建立新檔，以文字模式寫入
fn = open('..\\resume.txt', 'wt')
fn.write(content)     # 將字串寫入檔案
fn.close()            # 關閉檔案
```

9-2-2 檔案的讀取

相對於寫入檔案的步驟，要讀取檔案的資料，首先必須以 open() 方法開啟指定的檔案，接著使用檔案物件所提供的 read()、readline() 或 readlines() 方法從檔案讀取資料，最後再以 close() 方法關閉檔案。

前面曾提醒各位，讀取檔案和寫入檔案有點不同，如果以讀取模式開啟檔案，當檔案不存在時，會發生找不到檔案的錯誤。為了避免這類錯誤，可以在開啟檔案之前，事先以 os.path 模組所提供的 isfile(file) 函數來檢查指定檔名的檔案是否存在。如果檔案存在則傳回 True，否則傳回 False。

以下程式就會先行判斷檔案是否存在，如果存在，則列印出所有的內容。

〈先行判斷檔案是否存在〉

檔名 isfile.py

```
01  import os.path # 匯入 os.path
02
03  # 利用 isfile() 方法先行判斷檔案是否存在
04  # 如果檔案存在則印出檔案內容
05  if os.path.isfile("resume.txt"):
06      fb=open("resume.txt","r")
07      for word in fb:
08          print(word)
09  # 如果檔案不存在則輸出檔案不存在的文字
10  else:
11      print(" 所指定開啟的檔案不存在 ")
```

執行結果 >>

```
所指定開啟的檔案不存在
```

程式解說 >>

- 第 05~08 行：利用 isfile() 方法先行判斷檔案是否存在，如果檔案存在則印出檔案內容。
- 第 10~11 行：如果檔案不存在則輸出檔案不存在的文字。

當檔案建立之後，就可以使用 read()、readline() 或 readlines() 方法來讀取檔案，先認識這三個方法。

read() 方法

使用 read() 方法讀取資料，會從檔案目前指標的所在位置，一個字元一個字元去做讀取，讀取指定個數的字元，並回傳該字串，其語法如下：

```
read(n)
```

但如果將參數 n 省略，則會以字串型態回傳檔案的所有資料。請特別留意，read() 方法如果沒有指定讀取的字元數，檔案很大則會耗掉資源。

下面的例子，會以 "r" 的檔案存取模式開啟檔案，檔案指標會指向檔案的開頭處，並從檔案的開頭讀取檔案所有資料內容，再以字串的資料型態指派給變數 text，這個時候系統會將檔案指標從原先的檔案開頭，移動到檔案的結尾處。接下來，程式設計人員就可以利用 print 指令將所取得檔案內容的字串印出，最後再透過 close() 方法關閉檔案。

〈read() 方法實作〉

檔名 readfile.py

```
01  # 以 open() 方法開啟指定檔案的文字檔
02  fb=open("introduct.txt","r")
03  # 以 read() 方法讀取檔案內容
04  text=fb.read()
05  # 輸入字串變數 text 的內容
06  print(text)
07  # 以 close() 方法關閉檔案
08  fb.close()
```

執行結果 »

```
Word全方位排版實務：紙本書與電子書製作一次搞定

Word排版技巧大進擊
輕鬆活用Word做到專業文件排版
做出具視覺效果、易於閱讀的專業質感文件
```

上面的 read() 方法沒有指定要讀取的字元數，其實我們也可以在 read() 方法中傳入一個參數來告知要讀取幾個字元。

接下來的例子，我們仍以上述的文字檔案為例，一開始檔案指標指向檔案的開頭，在讀取所指定的 12 個字元後，檔案指標也會移動到該 12 個字元之後，接著再讀取下一道指令所指定的字元數，一旦不再需要存取檔案時，就可以利用 close() 方法關閉檔案。

〈read(n) 方法實作〉

檔名 readnfile.py

```
01  # 以 open() 方法開啟指定檔案的文字檔
02  fb=open("introduct.txt","r")
03  # 以 read(n) 方法讀取檔案內容
04  text=fb.read(12)
05  # 輸入字串變數 text 的內容
06  print(text)
07  # 以 read(n) 方法讀取檔案內容
08  text=fb.read(13)
09  # 輸入字串變數 text 的內容
10  print(text)
11  # 以 close() 方法關閉檔案
12  fb.close()
```

執行結果 »

```
Word全方位排版實務：
紙本書與電子書製作一次搞定
```

我們知道檔案指標是指向目前要寫入或讀取檔案的位置，但是前面兩個例子的檔案指標的移動都是系統自行移動，如果程式設計人員想要自行透用指令移動到指定的位置，這個時候就可以使用 seek(offset) 方法，這個指令所代表的功能是將檔案指標移動到第 offset+1 個位元組，例如 seek(0) 表示將檔案指標移動到檔案中的第一個位元組位置，也就是檔案的開頭，接下來的例子，各位可以清楚知道 seek() 的主要功用。

〈seek() 方法實作〉

檔名 seek.py

```
01  # 以 open() 方法開啟指定檔案的文字檔
02  fb=open("introduct.txt","r")
03  # 將檔案指標移動到檔案的開頭處
04  fb.seek(0)
05  # 以 read(n) 方法讀取檔案內容
06  text=fb.read(4)
07  # 輸入字串變數 text 的內容
08  print(text)
09  # 將檔案指標往前移動 20 個位元組
10  fb.seek(20)
11  # 以 read(n) 方法讀取檔案內容
12  text=fb.read(13)
13  # 輸入字串變數 text 的內容
14  print(text)
15  # 以 close() 方法關閉檔案
16  fb.close()
```

執行結果 »

```
Word
紙本書與電子書製作一次搞定
```

綜合上述所介紹的各種方法，接下來將示範如何進行檔案的複製。

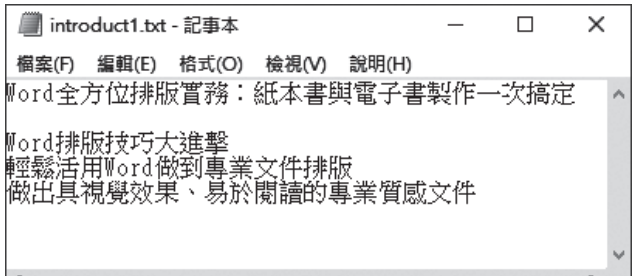
〈先行判斷檔案是否存在〉

檔名 copyfile.py

```
01  import os.path            # 匯入 os.path
02  import sys
03
04  # 利用 isfile() 方法先行判斷檔案是否存在
05  # 如果檔案存在則取消複製工作
06  if os.path.isfile("introduct1.txt"):
07      print(" 所指定開啟的檔案存在，不要進行複製 ")
08      sys.exit()
09  else:
10      # 以 open() 方法開啟指定檔案，檔案開啟模式為 "r"
11      fb1=open("introduct.txt","r")
12      # 以 open() 方法開啟指定檔案，檔案開啟模式為 "w"
13      fb2=open("introduct1.txt","w")
14      text=fb1.read()       # 以 read() 方法讀取檔案內容
15      text=fb2.write(text)  # 以 write() 方法寫入檔案
16      print(" 檔案複製工作完成，請開啟 introduct1.txt 查看 ")
17      fb1.close()           # 以 close() 方法關閉檔案
18      fb2.close()           # 以 close() 方法關閉檔案
```

執行結果 »

```
檔案複製工作完成,請開啟introduct1.txt查看
```



但如果檔案已存在，則會出現如下的執行結果。

```
所指定開啟的檔案存在,不要進行複製
```

readline() 方法

read() 方法是一次讀取一個字元，但是 readline() 方法可以整行讀取，並將整行的資料內容以字串的方式回傳，如果說所傳回的是空字串，就表示已讀取到檔案的結尾。以下程式碼則是以 realline() 方法，以一次一次讀取的方式，將檔案內容逐筆輸出。

```
01  obj=open("chapter.txt",r)
02  line=obj.readline()
03  while True:
04      print(line)
05      line=obj.readline()
06  obj.close()
```

上例也可以使用 for 迴圈改寫，也可以得到相同的輸出結果。

```
01  obj=open("chapter.txt",r)        # 開啟檔案
02  for line in obj:
03      print(line)
04  obj.close()
```

readlines() 方法

還有一個 readlines() 方法可以幫助使用者讀取檔案，這個方法會一次讀取檔案所有行，再以串列（List）的型式傳回所有行，以下述簡例做了解。

〈readlines() 方法實作〉

檔名 readlines.py

```
01  with open('introduct.txt', 'rt') as foin:
02      total = foin.readlines()     # 一次讀取所有行
03
04  # 取得行數，再以 for 迴圈讀取
05  print(' 行數：', len(total))
06  for line in total:
07      print(line, end = '')
```

執行結果 »

```
行數： 5
Word全方位排版實務：紙本書與電子書製作一次搞定

Word排版技巧大進擊
輕鬆活用Word做到專業文件排版
做出具視覺效果、易於閱讀的專業質感文件
```

其中 readlines() 方法取得文件的總行數，再以 for 迴圈一行一行讀出。

9-3 二進位檔案

電腦上的資料並非只有文字類型，常見的資料格式還有圖片、音樂，或者經過編譯的 EXE 檔案等，這些資料無法以文字類型的方式來處理，就必須以其他的資料格式來處理。如果要新建二進位檔案，就是把 open() 方法的 mode 參數加入「b」，而且是二進位，否則會引發錯誤，而內建函數 bytearray() 取得二進位資料。

〈二進位資料實作〉

檔名 binary.py

```
01  tmp = bytearray(range(8))
02  # 二進位資料的寫入
03  with open('bytedata', 'wb') as fob:
04      fob.write(tmp)
05  # 二進位資料的讀取
06  with open('bytedata', 'rb') as fob:
07      fob.read(3)
08      print(type(tmp))
09      print(' 二進位：', tmp)
```

執行結果 >>

```
<class 'bytearray'>
二進位： bytearray(b'\x00\x01\x02\x03\x04\x05\x06\x07')
```

程式解說 >>

- 第 03~04 行：open() 方法建立二進位新檔，mode 設「wb」，以 write() 方法寫入二進位資料。
- 第 06~09 行：進行二進位資料的讀取，並將所讀取的資料進行輸出。

9-4 綜合範例—考古題高頻率單字統計

坊間有許多英文考試書籍標榜收錄歷年考試出現頻率較高的單字，而收錄成考前衝刺 1000 單字，其作法就是將所收錄的考題，作適當的文字檔案的整理，再以編寫程式的方式，將該檔案中每個單字出現的次數加以統計。

接著我們就來利用 Python 語言及本章所教導的檔案開啟技巧，實際來統計一個檔案中每個單字出現的次數。

程式一開始會先以檔案開啟的方式將所收集的文字檔案以唯讀的方式開啟，並將讀取的檔案內容存入 text 變數，最後再關閉檔案。

接著程式會呼叫這個範例程式中事先定義的一個函數，這個函數的功能可以用來計算所傳入字串中相同單字出現的字數，接著字串物件中的 split() 方法可以根據指定分隔符號將字串分割為子字串，並回傳子字串的串列。格式如下：

```
字串 .split( 分隔符號 , 分割次數 )
```

預設的分隔符號為空字串，分隔符號包括空格、換行符號 (\n)、定位符號 (\t)。使用 split() 方法分割字串時，會將分割後的字串以串列 (list) 回傳。

為了避免因為大小寫的不同，而造成誤判為不同的單字，在進行各單字出現次數的統計時，還必須先行將該單字以 lower() 函數轉換成小寫，如此的作法，就可以在這份檔案中不分大小寫相同的單字進行出現次數的累計工作。

其執行結果如下：

```
英文單字 abrupt 出現 1 次
英文單字 advertiser 出現 6 次
英文單字 advertising 出現 6 次
英文單字 agent 出現 5 次
英文單字 assembly 出現 5 次
英文單字 assurance 出現 5 次
英文單字 auction 出現 5 次
英文單字 automaker 出現 4 次
英文單字 automate 出現 4 次
英文單字 automatic 出現 3 次
英文單字 automaton 出現 3 次
英文單字 backlog 出現 3 次
英文單字 barren 出現 3 次
英文單字 batch 出現 3 次
英文單字 board 出現 3 次
英文單字 bonus 出現 3 次
英文單字 brand 出現 3 次
```

而完整的程式碼條列如下：

〈計算檔案中每個單字出現的次數〉

檔名 countword.py

```
01  # 定義函數用來計算所傳入字串中相同單字出現的字數
02  def check(string):
03      wordlist = string.split()
04      for word in wordlist:
05          if word.lower() in mydict:
06              mydict[word.lower()] = mydict[word.lower()] + 1
07          else:
08              mydict[word.lower()] = 1
09
10  fb= open("exam.txt", "r")
11  text = fb.read()
12  fb.close()
13
14  mydict = {}

15  check(text)
16
17  for key in mydict:
18      print(" 英文單字 ",key," 出現 ",mydict[key]," 次 ")
```

程式解說

- 第 02~08 行：定義一個函數，這個函數的功能可以用來計算所傳入字串中相同單字出現的字數。
- 第 05~08 行：利用 for 迴圈計算串列中出現相同單字的次數，其中 mydict 是一個全域變數，是用來存放每個單字出現次數的字典型別。不過要計算各單字的出現次數之前，必須行將該單字以 lower() 函數轉換成小寫，如此的作法，就可以在這份檔案中不分大小寫相同的單字進行出現次數的累計工作。

重點回顧

1 檔案（File）是電腦中數位資料的集合，可以是一份報告、一張圖片或一個執行程式，並且包括了資料檔、程式檔與可執行檔等格式。

2. 檔名可區分成「主檔名」與「副檔名」，中間以句點 (.) 做分隔，透過這樣的命名方式可以清楚分辨其名稱與類型。

3. 檔案如果依儲存方式來分類，可以分文字檔（text file）與二進位檔（binary file）兩種。

4. 文字檔是以字元編碼的方式進行儲存，在 Windows 作業系統的記事本程式中則預設以 ASCII 編碼來儲存文字檔，每個字元佔有 1 位元組。

5. 所謂二進位檔，就是以二進位格式儲存，這種儲存方式適用於非字元為主的資料。

6. 在一個目錄（資料夾）中的另一個目錄（資料夾）被稱作它的子目錄（子資料夾）。這些目錄（資料夾）就構成了層次（hierarchy）或樹形結構。

7. 檔案系統是一種儲存和組織電腦資料的方法，例如 Windows 作業系統中的檔案總管，可以幫助使用者在存取檔案時，變得更為直覺容易。

8. 常見的作業系統 Windows 7/8/10 的檔案系統則有 FAT32、NTFS(New Technology File System) 或 exFAT(Extended File Allocation Table)。

9. 「絕對路徑」指的是一個絕對的位置，並不會隨著現在目錄的改變而改變。

10. 「相對路徑」就是相對於現在目錄的路徑表示法，因此「相對路徑」所指到的檔案或目錄，會隨著現在目錄的不同而改變。通常我們用「.」代表現在目錄，而用「..」代表上一層目錄。

11. 檔案的寫入或讀取工作，第一項工作就是利用 Python 內建的 open() 函數建立檔案物件，檔案物件（File Object）是一個提供存取的介面，開啟檔案之後，得藉由「檔案物件」做讀（Read）或寫（Write）的動作。

12. 利用 open() 建立檔案物件的工作成功，就會傳回檔案物件，但是如果建立失敗，就會發生錯誤。

13. 如果是指寫入模式開啟檔案，第一次開啟該檔案，該檔案不會存在，此時系統就會自動建立該名稱的新檔。

14. 使用 open() 函數開啟檔案時，所指定的檔案路徑必須以跳脫序列 \\ 來表示 \。

15. 可以在絕對路徑前面加 r，用以告知編譯器系統底下所用路徑的字串是原始字串。

16. 對於文字檔案而言，要將資料寫入檔案必須事先以 open() 方法建立新檔，再使用檔案物件所提供的 write() 方法將文字寫入檔案，最後以檔案物件所提供的 close() 方法關閉檔案。

17. 相對於寫入檔案的步驟，要讀取檔案的資料，首先必須以 open() 方法開啟指定的檔案，接著使用檔案物件所提供的 read()、readline() 或 readlilnes() 方法從檔案讀取資料，最後再以 close() 方法關閉檔案。

18. 為了避免找不到檔案的錯誤，可以在開啟檔案之前，事先以 os.path 模組所提供的 isfile(file) 函數來檢查指定檔名的檔案是否存在。

19. 當檔案建立之後，就可以使用 read()、readline() 或 readlines() 方法來讀取檔案。

20. read() 方法讀取資料，會從檔案目前指標的所在位置，一個字元一個字元去做讀取，讀取指定個數的字元，並回傳該字串。

21. 如果程式設計人員想要自行透用指令移動到指定的位置，這個時候就可以使用 seek(offset) 方法，這個指令所代表的功能是將檔案指標移動到第 offset+1 個位元組。

22. readline() 方法可以整行讀取，並將整行的資料內容以字串的方式回傳，如果所傳回的是空字串，就表示已讀取到檔案的結尾。

23. readlines() 方法可以幫助使用者讀取檔案，這個方法會一次讀取檔案所有行，再以串列 (List) 的型式傳回所有行。

24. 要新建二進位檔案，就是把 open() 方法的 mode 參數加入「b」，而且是二進位，否則會引發錯誤。

Chapter 9 學習評量

選擇題

(　　) 1. 有關檔案的描述，下列何者有誤？
(A) 檔名可區分成「主檔名」與「副檔名」
(B) 使用 open() 函數開啟檔案時，檔案路徑必須以跳脫序列 \\ 來表示 \
(C)「相對路徑」就是相對於現在目錄的路徑表示法
(D) 通常我們用「..」代表現在目錄，而用「.」代表上一層目錄。

(　　) 2. 使用檔案物件所提供的方法從檔案讀取資料不包括？
(A) readlilnes() 方法　(B) readline() 方法
(C) read() 方法　(D) seek() 方法。

(　　) 3. 要新建二進位檔案，就是把 open() 方法的 mode 參數加入哪一個參數？
(A) t　(B) m　(C) b　(D) h。

填充題

1. 所謂_____是以二進位格式儲存，這種儲存方式適用於非字元為主的資料。

2. _____是一種儲存和組織電腦資料的方法。

3. 利用 open() 函數開啟檔案之後，得藉由_____做讀（Read）或寫（Write）的動作。

4. 可以在絕對路徑前面加_____，用以告知編譯器系統底下所用路徑的字串是原始字串。

5. _____方法可以整行讀取，並將整行的資料內容以字串的方式回傳。

6. 要新建二進位檔案，就是把 open() 方法的 mode 參數加入_____，而且是二進位，否則會引發錯誤。

簡答題

1. 檔案如果依儲存方式來分類可以分為哪幾種類型？

2. 絕對路徑與相對路徑的差別，試簡述之。

3. 請簡述檔案讀取的步驟。

Chapter

10 錯誤與例外處理

當我們在撰寫應用程式的過程中，通常都會遇到一些錯誤或者例外（Exception），也就是指程式中需要特別處理的狀況。一個好的程式應該要能夠事先考量到所有可能發生的情形，把這些例外的情況都攔截下來，並加以適當的處理。本章將會跟各位討論 Python 程式設計時的錯誤種類與例外處理的功能。

學習大綱

- 語法錯誤
- 執行期間錯誤
- 邏輯錯誤
- 例外的類型
- 例外處理的時機
- 例外處理的語法
- 以 raise 丟出指定的例外

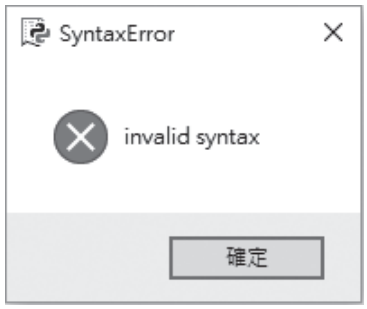
在程式執行的過程中，可能會碰到許多錯誤，如果可以清楚知道各種錯誤的類型，將有利程式設計人員的除錯方向。另外，當錯誤發生時的例外處理機制，可以確保程式可以正常的執行與結束，而不會因為錯誤的發生，導致程式異常中斷。

10-1 程式錯誤類型

程式寫作的過程中，有可能因為語法不熟悉、指令的誤用或是設計邏輯有誤，導致程式發生錯誤或產生不是原先預期的結果。因此，一支好的程式，小至一些簡易計算工作的程式，甚至大型應用程式、資料庫程式或是遊戲的開發，都必須經過很嚴謹及細心的除錯工作，反覆測試各種可能出現錯誤的檢查點，有了這些嚴謹的作業流程，才可以提高程式的正確性，降低程式發生錯誤的機率。通常程式的錯誤類型可以分為三種類型：語法錯誤、執行期間錯誤、邏輯錯誤。

10-1-1 語法錯誤

語法錯誤是最常見的錯誤，這種錯誤有可能是撰寫程式時，不小心寫錯語法所造成。例如定義函數或使用流程結構指令時忘記加了「:」來形成 suite。以下述範例來說，它會顯示如右的「SyntaxError: invalid syntax」的語法錯誤的訊息視窗。



例如底下的程式碼中就會出現如同上圖的語法錯誤（SyntaxError）訊息視窗：

```
01  month=int(input(' 請輸入月份： '))
02  if 2<=month and month<=4
03      print(' 充滿生機的春天 ')
04  elif 5<=month and month<=7:
05      print(' 熱力四射的夏季 ')
06  elif month>=8 and month <=10:
07      print(' 落葉繽紛的秋季 ')
08  elif month==1 or (month>=11 and month<=12):
09      print(' 寒風刺骨的冬季 ')
10  else:
11      print(' 很抱歉沒有這個月份 !!!')
```

各位可以發現在上述程式碼中的第 2 行少了冒號，只要進行語法修改就可以正常執行程式碼。

```
01  month=int(input(' 請輸入月份： '))
02  if 2<=month and month<=4:
03      print(' 充滿生機的春天 ')
04  elif 5<=month and month<=7:
05      print(' 熱力四射的夏季 ')
06  elif month>=8 and month <=10:
07      print(' 落葉繽紛的秋季 ')
08  elif month==1 or (month>=11 and month<=12):
09      print(' 寒風刺骨的冬季 ')
10  else:
11      print(' 很抱歉沒有這個月份 !!!')
```

執行結果 »

```
請輸入月份： 6
熱力四射的夏季
```

10-1-2 執行期間錯誤

執行期間錯誤是指程式在執行期間遇到錯誤，這類錯誤可能是邏輯上的錯誤，也可能是資源不足所造成的錯誤。例如底下的程式碼片段打算要讀取串列的資料，但卻發生「IndexError: list index out of range」超出索引範圍的錯誤。

```
01  data=[[1,2],[2,1],[1,5],[5,1], \
02        [2,3],[3,2],[2,4],[4,2], \
03        [3,4],[4,3]]
04  for i in range(14):  # 讀取圖形資料
05      for j in range(6):  # 填入 arr 矩陣
06          for k in range(6):
07              tmpi=data[i][0]    #tmpi 為起始頂點
08              tmpj=data[i][1]    #tmpj 為終止頂點
09              arr[tmpi][tmpj]=1  # 有邊的點填入 1
```

這個錯誤告訴使用者的訊息是在取用串列內的元素時發生超出索引的錯誤，遇到這種錯誤的正確解決方法就是詳細檢查所宣告的串列元素個數，再去比對在迴圈存取串列的元素時，是否超出索引值所設定的範圍。

以本範例的情況來看，各位只要仔細檢查就可以發現 for 迴圈中控制數字範圍的上限出了問題，因此只要修改成如下的程式碼，就可以解決這項錯誤。

```
01  data=[[1,2],[2,1],[1,5],[5,1], \
02        [2,3],[3,2],[2,4],[4,2], \
03        [3,4],[4,3]]
04  for i in range(10):  # 讀取圖形資料
05      for j in range(2):  # 填入 arr 矩陣
06          for k in range(6):
07              tmpi=data[i][0]    #tmpi 為起始頂點
08              tmpj=data[i][1]    #tmpj 為終止頂點
09              arr[tmpi][tmpj]=1  # 有邊的點填入 1
```

又例如底下的程式碼並沒有將輸入的字串轉換成整數型態的處理動作，表面上在編譯過程中語法並沒有任何錯誤，但實際執行過程中就發生並非原先預期的錯誤。

以下程式原先設計的邏輯是當輸入「-1」時，執行 break 指令而跳離迴圈，但在下面的程式碼中即使用輸入 -1，卻始終無法跳離迴圈程式。

【尚未除錯的程式片段】

```
while(True):
    print('請輸入要插入其後的員工編號，如輸入的編號不在此串列中，')
    position=input('新輸入的員工節點將視為此串列的串列首，要結束插入過程，請輸入 -1：')
    if position ==-1:
        break
    else:

        ptr=findnode(head,position)
        new_num=int(input('請輸入新插入的員工編號：'))
        new_salary=int(input('請輸入新插入的員工薪水：'))
        new_name=input('請輸入新插入的員工姓名： ')
        head=insertnode(head,ptr,new_num,new_salary,new_name)
    print()
```

這個除錯的關鍵點在於 input() 函數所輸入的 -1，事實上是字串型態，所以「position==-1」這樣的指令，永遠不會是 True 的結果，因此會造成無法跳離迴圈的錯誤，所以只要針對程式作如下的修正，就可以正確執行程式了。

```
while(True):
    print('請輸入要插入其後的員工編號，如輸入的編號不在此串列中，')
    position=int(input('新輸入的員工節點將視為此串列的串列首，要結束插入過程，請輸入 -1：'))
    if position ==-1:
        break
    else:

        ptr=findnode(head,position)
        new_num=int(input('請輸入新插入的員工編號：'))
        new_salary=int(input('請輸入新插入的員工薪水：'))
        new_name=input('請輸入新插入的員工姓名： ')
        head=insertnode(head,ptr,new_num,new_salary,new_name)
    print()
```

10-1-3 邏輯錯誤

邏輯錯誤是三種錯誤中最不容易被發現的錯誤，邏輯錯誤常會產生令人出乎意料之外的輸出結果。與語法錯誤不同的是，邏輯錯誤從程式語法上來說是正確的一段程式碼，但其執行結果卻與預期不符。

例如以下的程式碼，原先預期兩數的平均是 7，但執行結果卻是 10。這是因為第 2 行中要計算兩數的平均值必須先以括號的方式來先行計算兩數的總和，取得兩數總和後才除以 2，才可以得到最後正確的結果。

但是底下錯誤的程式碼卻沒有考慮到了運算子的執行優先順序，就造成了所得到的執行結果不是原先預期的。

```
01  def average(a,b):
02      return a+b/2 # 此程式的正確寫法應該為 (a+b)/2
03
04  print(" 兩數平均： ",average(6,8))
```

上面所示範的程式還算小程式，或許透過細心的除錯可以找到邏輯錯誤的原因，萬一程式越來越大時，邏輯錯誤的真正原因很不容易被查覺，程式除錯過程中，如果運氣差的時候，還可能會花上程式設計人員不少的時間進行除錯的工作。

10-2 認識例外

何謂例外（Exception）情況？當程式執行時，產生了不是程式設計人員原先預期的結果，也就是於程式執行中所可能發生的「不可預期」的特殊情形。這種情況下，Python 直譯器會接手管理，並終止程式的運作。也就是說，當例外發生時，如果所撰寫的程式沒有進行任何處置時，程式會發出錯誤訊息，而造成程式的終止。

例如要求使用者連續輸入兩個數字，而且第二個數字不可以為 0，如果程式碼中沒有編寫例外處理的程式碼，當使用者不小心第二個數字輸入 0，就會發生除零的錯誤訊息，並造成程式的中斷，這當然不是一種好的處理方式。比較正確的作法是，一旦使用者不小心在第二個數字輸入到 0，就會捕捉到這個錯誤，然後要求使用者重新輸入一個非零的數值，如此一來，程式就不會因為這個例外的發生而結束執行的工作。

在還沒有示範如何利用 Python 提供「例外處理機制」（Exception handling）來捕捉程式的錯誤前，我們先來看看例外的類型。

10-2-1 例外的類型

系統會根據不同的錯誤情況丟出不同的例外，有關各種例外類型的詳細說明，各位可以自行參考 Python 的線上文件說明。以下只針對幾種常見的例外類型說明如下：

- **LookupError**：當映射或序列型別的鍵或索引無效時所引發，所以它有兩個衍生類別：IndexError 和 KeyError。其中 IndexError 就是指索引運算子錯誤。

- **NameError**：是指名稱沒有定義的錯誤，例如以下的程式碼中呼叫函數時，函數的名稱並未定義，就會引發 NameError: name 'inpu' is not defined 的錯誤，關鍵原因是漏了一個字母 t，因此必須將函數名稱修改成 input 就可以修正這項錯誤。

```
01  print('1.80 以上 ,2.60~79,3.59 以下 ')
02  ch=inpu(' 請輸入分數群組 : ')
03  # 條件指令開始
04  if ch=='1':
05      print(' 繼續保持 !')
06  elif ch=='2':
07      print(' 還有進步空間 !!')
08  elif ch=='3':
09      print(' 請多多努力 !!!')
10  else:
11      print('error')
```

或是例如 total 變數是在 for 迴圈內使用的區域變數，卻在 for 迴圈之輸出 total 累加的結果，也會產生例外。

- **OSError**：操作系統函數發生錯誤時所引發。
- **RuntimeError**：執行期間錯誤。
- **SyntaxError**：從字面上來解釋就是「語法錯誤」。Python 直譯器無法理解要解譯的程式碼語法就會引發此例外。
- **ValueError**：使用內建函數時，參數中的型別正確，但值不正確，就引發此例外。例如使用 input() 函數來接收資料，並以 int() 函數轉為數值，但是所輸入卻是字串，就會引發這類的例外。
- **ZeroDivisionError**：除零錯誤，當兩數相除時，如果除數不小心設定為數值 0 或不小心輸入到數值為 0 的除數，就會引發除零錯誤。
- **FileNotFoundError**：這是指程式中所要求開啟或寫入的檔案不存在，有可能因為儲存目錄位置改變或還沒建立該檔案，也有可能使用者輸入的檔案錯誤，造成程式找不到該指定名稱的檔案，而丟出這種找不到檔案的例外錯誤。
- **FileExistsError**：這種錯誤通常發生在程式設計人員想在已存在的檔案重複要新建該檔案。

- **TypeError**：不論是運算式或是函數中的參數，當所輸入或指定資料的型別與該運算式或函數指定的型別不符時，就會引發這種類別的錯誤。
- **OverflowError**：進行算術運算所得到數值超出該物件型別所能表示的範圍，也就是通常我們所稱的溢位錯誤。
- **MemoryError**：記憶體錯誤表示程序已經用完了記憶體，這意味著程序建立太多的物件而造成記憶體不足的錯誤。

10-2-2 例外處理的時機

有了這些常見例外類型的基礎認識後，各位可能會想知道在哪一種情況下必須加入例外處理程式？其實當程式與外部的人員或裝置進行互動的 I/O 行為時，是常見的例外處理的時機。

例如要求使用者輸入數字或輸入字串或是與外部資料庫連結時，就算是所撰寫的程式碼完全正確，但可能因為網路線路不穩定或網路中斷，或是資料庫連結過程發生異常，造成程式發生錯誤，這種情況下都會引發系統丟出例外，並強迫中斷程式的執行。為了避免這種錯誤情況的發生，程式設計人員可以適當加入了例外處理的程式碼，輸出使用者的解決步驟的提示或是錯誤訊息的提醒，並要求使用者修正錯誤後再度嘗試，當程式加入了完善的例外處理機制，就不會因為例外的發生，而提前強制中斷程式的執行。

10-3 例外處理方式介紹

清楚了各種常見的例外類型及例外處理的時機後，接下來將說明如何在 Python 中捕捉例外。

10-3-1 例外處理的語法

在 Python 處理例外可以使用 try/except 指令，先了解它的完整語法。

```
try:
    可能發生例外的指令
except 例外型別名 :  # 只處理所列示的例外
    處理狀況一
except ( 例外型別名 1, 例外型別名 2, ...):
    處理狀況二
except 例外型別名 as 名稱 :
    處理狀況三
except :   # 處理所有例外情形
    處理狀況四
else :
    # 未發生例外的處理
finally :
    # 無論如何，最後一定執行 finally 指令
```

- try 指令之後要有冒號「:」來形成 suite，並在此加入可能引發例外的指令。
- except 指令配合「例外型別」，用來截取或捕捉 try 指令區段內引發例外的處理。同樣地，except 指令之後要給予冒號「:」形成 suite。
- else 指令則是未發生例外時所對應的區段。else 指令為選擇性指令，可以加入也可以省略。
- 無論有無例外引發，finally 指令所形成的區段一定會被執行。finally 指令為選擇性指令，可以加入也可以省略。

下面例子要求使用者輸入總分 score 和總人數 total，然後計算所有人員的分數總平均，接著令變數 ave=score / total，再印出所有人員的分數平均值。如果輸入的數值型態符合程式的需求，就不會引發例外錯誤的發生，並且正確求得所有數字總和的平均值。

〈計算平均值〉

檔名 except01.py

```
score=eval(input(" 請輸入數字總和： "))
total=eval(input(" 請輸入要求取平均的數字個數： "))
ave=score/total
print(" 所有數字的平均數 =", ave)
```

但是針對上述程式碼，如果在輸入過程中不小心輸入了不正確型態的資料，就會產生各種情況的錯誤，例如以下各圖為各種不正確的輸入的執行結果：

```
請輸入要求取平均的數字個數: A
Traceback (most recent call last):

  File "<ipython-input-81-157da277fb12>", line 1, in <module>
    runfile('D:/進行中書籍/台科大/Python_450頁版本/範例檔/ch10ok/except01.py',
wdir='D:/進行中書籍/台科大/Python_450頁版本/範例檔/ch10ok')
```

輸入數字個數時不小心輸入成英文字母

```
請輸入數字總和: 500

請輸入要求取平均的數字個數: 0
Traceback (most recent call last):

  File "<ipython-input-82-157da277fb12>", line 1, in <module>
    runfile('D:/進行中書籍/台科大/Python_450頁版本/範例檔/ch10ok/except01.py',
wdir='D:/進行中書籍/台科大/Python_450頁版本/範例檔/ch10ok')
```

計算平均時分母不可以為 0 的錯誤

```
請輸入數字總和: 500\
Traceback (most recent call last):

  File "C:\Users\andu-Wu\Anaconda3\lib\site-packages\IPython\core
\interactiveshell.py", line 2862, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)
```

輸入數字總和時不小心按到鍵盤上其它的按鍵

- 上圖中第 1 個錯誤：當使用者輸入數字總和為 500 且求取平均的數字個數輸入英文字母 A，系統會丟出 NameError 的例外訊息，並馬上終止程式的執行工作。
- 上圖中第 2 個錯誤：當使用者輸入數字總和為 500 且求取平均的數字個數輸入數值 0，系統會丟出 ZeroDivisionError: division by zero 的例外訊息，並馬上終止程式的執行工作。
- 上圖中第 3 個錯誤：當使用者輸入數字總和為「500:」，系統會丟出 SyntaxError: unexpected EOF while parsing 的例外訊息，並馬上終止程式的執行工作。

為了避免類似上述三種情況的錯誤輸入行為，而造成程式結束執行，就可以利用例外處理機制，只要將上面的程式碼改寫如底下的程式。這個程式會捕捉輸入數字為 0 的錯誤和其它的例外，並於程式中輸出如何處理各種例外的作法，當在程式中加入了完善的例外處理機制後，就可以正確執行，而不會出現如同上一個例子中的各種紅色文字的錯誤訊息。

〈在計算平均值的程式中加入例外處理〉

檔名 except02.py

```
01 try:
02     score=eval(input(" 請輸入數字總和 : "))
03     total=eval(input(" 請輸入要求取平均的數字個數 : "))
04     ave=score/total
05 except ZeroDivisionError:
06     print(" 所輸入的數字總數不可以為 0")
07 except Exception as e1:
08     print(" 錯誤訊息 ",e1.args)
09 else:
10     print(" 沒有捕捉到例外 , 所有數字的平均值 = ", ave)
11 finally:
12     print(" 成功離開此例外處理的區塊 ")
```

執行結果 >>

```
請輸入數字總和: 500

請輸入要求取平均的數字個數: 20
沒有捕捉到例外, 所有數字的平均值=  25.0
成功離開此例外處理的區塊
```

</> 程式解說 >>

- 第 05~06 行：程式中使用了 except 指令來捕捉 ZeroDivisionError 例外，如果程式捕捉到這種除零錯誤的例外，就會印出「所輸入的數字總數不可以為 0」。
- 第 07~08 行：程式中使用了 except 指令來捕捉其它例外，如果程式捕捉到其它例外，就會執行第 8 行的輸出指令，印出變數 e1 的 args 屬性，該屬性值可取得此例外的相關訊息。
- 第 09~10 行：如果 try 的區塊沒有發生例外時，則會直接執行 else 指令內的程式區塊，並執行第 10 行要求印出的字串及 ave 的計算結果。
- 第 11~12 行：無論程式執行過程中有沒有發生例外，當要離開 try... except 時，都會執行 finally 區塊內程式碼，此例會印出「成功離開此例外處理的區塊」。

10-3-2 以 raise 丟出指定的例外

除了由 Python 系統丟出的例外，在程式中也可以使用 raise 指令自行丟出指定的例外。例如底下的指令會丟出一個除零的例外：

```
>>> raise Exception(12/0)
Traceback (most recent call last):
    File"<pyshell#0>", line 1, in <module>
            raise Exception(12/0)
ZeroDivisionError: division by zero
```

在程式捕捉到例外，並做處理，而程式設計人員又不希望中斷程式執行，這個時候就可以利用 try/except 來捕捉 raise 述丟出的例外，以下述簡例說明。

〈利用 try/except 來捕捉 raise 丟出的例外〉

檔名 except03.py

```
01  def show(data, index):
02      try:
03          data[index]
04      except IndexError as err:
05          print(err)
06          raise IndexError('索引超出界值')
07      else:
08          print(data[index])
09  number = [15, 20, 60, 100] #List
10  show(number, 0)
11  show(number, 1)
12  show(number, 2)
13  show(number, 3)
14  show(number, 4)
```

執行結果 »

```
15
20
60
100
list index out of range
Traceback (most recent call last):

  File "<ipython-input-85-868193aef62b>", line 1, in <module>
    runfile('D:/進行中書籍/台科大/Python_450頁版本/範例檔/ch10ok/except03.py',
wdir='D:/進行中書籍/台科大/Python_450頁版本/範例檔/ch10ok')

  File "C:\Users\andu-Wu\Anaconda3\lib\site-packages\spyder\utils\site
\sitecustomize.py", line 710, in runfile
    execfile(filename, namespace)

  File "C:\Users\andu-Wu\Anaconda3\lib\site-packages\spyder\utils\site
\sitecustomize.py", line 101, in execfile
    exec(compile(f.read(), filename, 'exec'), namespace)

  File "D:/進行中書籍/台科大/Python_450頁版本/範例檔/ch10ok/except03.py", line
14, in <module>
    show(number, 4)

  File "D:/進行中書籍/台科大/Python_450頁版本/範例檔/ch10ok/except03.py", line
6, in show
    raise IndexError('索引超出界值')
```

又例如要觸發輸入錯誤（ValueError），可以如下表示：

```
try:
    n = int(input(' 請輸入 0~10 的整數：'))
    if n not in range(0,11):
        raise ValueError  # 自己觸發例外
    else:
        print(n)
except ValueError:
    print(' 必須是 0~10 的整數 .')
```

如果使用者輸入的不是數字，在 int() 轉換時就會引發錯誤，這是直譯器觸發的錯誤，如果使用者輸入的數字不是介於 0~10 之間，我們就可以自己利用 raise 指令來觸發錯誤。

10-4 綜合範例—以例外處理程式控制使用者輸入的數值

撰寫一個 Python 程式，當然也可以將 try/exception/finally 指令搭配在一起使用，要求使用者輸入兩個數值，並以逗點隔開，如果所輸入的名稱不是一個合法的數值格式，則要求重新輸入，直到輸入正確的字串名稱為止，請參考下圖的執行結果。

```
請輸入兩個數值，用逗點隔開：19,0
錯誤 integer division or modulo by zero
完成計算

請輸入兩個數值，用逗點隔開：19, 3
計算結果 (6, 1)
完成計算
```

【參考解答】

〈try/exception/finally 上機實作〉

檔名 except04.py

```
while True:
    one, two = eval(
            input('請輸入兩個數值，用逗點隔開：'))
    try:
        result = divmod(one, two)
    except ZeroDivisionError as err:
            print('錯誤', err)
    else:
            print('計算結果', result)
            break
    finally:
            print('完成計算')
```

重點回顧

1. 程式錯誤的三種類型：語法錯誤、執行期間錯誤、邏輯錯誤。
2. 語法錯誤是最常見的錯誤，這種錯誤有可能是撰寫程式時，不小心寫錯語法所造成。
3. 執行期間錯誤是指程式在執行期間遇到錯誤，可能是邏輯上的錯誤，也可能是資源不足所造成的錯誤。
4. 邏輯錯誤是最不容易被發現的錯誤，邏輯錯誤會產生意外的輸出或結果。與語法錯誤不同的是，邏輯錯誤從程式語法上來說是正確的，但其執行結果卻與預期不符。
5. 當例外發生時，如果所撰寫的程式沒有進行任何處置時，程式會發出錯誤訊息，而造成程式的終止。
6. 常見的例外類型：
   (1) ArithmeticError：未將數值運算做妥善處理所引發。
   (2) NameError：名稱沒有定義的錯誤。
   (3) RuntimeError：執行期間錯誤。
   (4) ZeroDivisionError：除零錯誤。
   (5) FileNotFoundError：開啟或寫入的檔案不存在。
   (6) TypeError：輸入或指定資料的型別與該運算式或函數指定的型別不符。
   (7) OverflowError：數值超出該物件型別所能表示的範圍。
7. 較常見的例外處理之時機，為程式與外部的人員或裝置進行互動的 I/O 行為時。
8. 在 Python 處理例外可以使用 try/except 指令，try 指令之後要有冒號「:」來形成 suite，並在此加入可能引發例外的指令。except 指令配合「例外型別」，用來截取或捕捉 try 指令區段內引發例外的處理。
9. 除了由 Python 系統丟出的例外，在程式中也可以使用 raise 指令自行丟出指定的例外。

Chapter 10

學習評量

選擇題

(　　) 1. 當資料的型態輸入錯誤時，系統會丟出底下何種例外？
(A) IndexError　(B) TypeError
(C) OverflowError　(D) SystemError。

(　　) 2. 當運算所得到數值超出該型別所能表示的範圍，系統會丟出何種例外？
(A) IndexError　(B) TypeError
(C) OverflowError　(D) SystemError

(　　) 3. 當所要求開啟或寫入的檔案不存在時，系統會丟出底下何種例外？
(A) FileNotFoundError　(B) TypeError
(C) OverflowError　(D) SystemError

(　　) 4. 使用內建函數時，參數的型別正確，但值不正確，系統會丟出何種例外？
(A) IndexError　(B) ValueError
(C) OverflowError　(D) SystemError

(　　) 5. 當索引界值發生例外時，可使用那個內建型別來捕捉例外？
(A) NameError　(B) ValueError
(C) IndentationError　(D) IndexError。

(　　) 6. 對於 try/except 的描述，何者正確？
(A) 無論有無例外，皆會將程式執行完畢
(B) try 敘述用來捕捉例外
(C) except 指令只能指定一個例外型別
(D) 無論有無例外引發，else 指令所形成的區段一定會被執行。

(　　) 7. 對於 finally 指令描述，何者正確？
(A) 捕捉例外
(B) 引發例外
(C) 處理例外
(D) 無論有無例外皆會把程式執行完畢。

填充題

1. _____錯誤是最常見的錯誤，可能是撰寫程式時，不小心所造成的語法或指令的輸入錯誤。

2. _____是指程式在執行期間遇到錯誤，可能是邏輯上的錯誤，也可能是其他資源不足所造成的錯誤。

3. _____是最不容易被發現的錯誤，這種錯誤常會產生意外的輸出或結果。

4. 當 total 變數是在 for 迴圈內使用的區域變數，卻在 for 迴圈之外輸出 total 累加的結果，會產生_____例外。

5. 在 Python 處理例外無論有無例外引發，_____指令所形成的區段一定會被執行。

簡答題

1. 程式的錯誤類型可以分為哪三種類型？

2. 何謂例外？試簡述之。

3. 請填入底下錯誤所引發的例外類型。

_____：當映射或序列型別的鍵或索引無效時所引發。

_____：操作系統函數發生錯誤時所引發。

_____：是指名稱沒有定義的錯誤。

_____：執行期間錯誤。

4. 請問哪一種情況下是較常見的例外處理的時機？

Chapter

11 物件導向程式設計

在 Python 世界，所有東西都是物件，物件導向的觀念已經倡導多年，最早的雛形是 1960 年的 Simula 語言，它導入和「物件」有關的概念。一直到 20 世紀 70 年代的 SmallTalk 語言的出現，它除了匯集 Simula 的特性之外，也引入「訊息」，第一個物件導向的程式語言才算真正的誕生。物件導向程式設計的重點是強調軟體的可讀性（Readability）、重複使用性（Reusability）與延伸性（Extension），本章將會跟各位討論 Python 的物件導向程式設計的主題。

學習大綱

- 類別與物件
- 物件導向的特點
- 定義類別
- 類別實體化
- 物件初始化
- 匿名物件
- 私有屬性與方法
- 單一繼承與多重繼承
- 覆寫（override）
- 繼承相關函數
- 多型
- 組合

物件導向程式設計（Object-Oriented Programming, OOP）的主要精神就是將日常生活中的物件概念，應用在軟體設計的發展模式（software development model），讓我們在設計程式時，能以一種更生活化、可讀性更高的設計思維來進行，並且所開發出來的程式也較容易擴充、修改及維護，以彌補「結構化程式設計」的不足，如 Python、C++、Java 等語言。

11-1 認識物件導向

現代許多程式語言中最讓人津津樂道的創新功能，無疑就是物件導向程式設計，這也是程式設計領域的一大創新。在傳統程式設計的方法中，主要是以「結構化程式設計」為主，就是「由上而下設計」 與「模組化設計」，也就是將整個程式需求從上而下、由大到小逐步分解成較小的單元，或稱為「模組」（module）。這樣使得程式設計師可針對各模組分別開發，不但減輕設計者負擔、可讀性較高，對於日後維護也容易許多。每一個模組都有其特定的功能，主程式則組合每個模組後，完成最後要求的功能。不過一旦主程式要求功能變動時，則可能許多模組內的資料與程式碼都需要同步變動，而這也是結構化程式設計無法有效使用程式碼的主要原因。

11-1-1 類別與物件

何謂物件（object）？在現實生活中充滿了各種形形色色的物體，每個物體都可視為一種物件，例如正在閱讀的書是一個物件，手上的筆也是一個物件。任何物件導向程式設計語言中最主要的單元就是物件，物件導向程式設計模式就是將問題實體分解成一個或多個物件，再依據需求加以組合。這些物件都各自擁有狀態（state）及行為（behavior）（或稱使用方法）。狀態代表了物件所屬特徵，行為則代表物件所具有的功能，使用者可依據物件的使用方法來操作物件，進而取得或改變物件的狀態資料。

舉例來說，如果使用者想要購買一台電腦，品牌、螢幕尺寸、CPU 等級、記憶體大小和硬體容量可能是購買時的考量因素。品牌、螢幕尺寸、CPU 等級、記憶體大小和硬體容量皆可用來描述電腦的特徵；以物件觀點來看，它

具有「屬性」（Attribute）（或稱成員變數）。物件除了屬性之外，還包含了「行為」，「行為」是一種動態的表現。以電腦來說，就是它具有的功能，隨著科技的普及，開機、即時通訊、人臉辨識等相關功能；以物件觀點來看，就是方法（或稱成員函數）。

在物件導向設計的理念就是認定每一個物件是一個獨立的個體，而每個獨立個體有其特定之功能，對我們而言，根本不需去理解這些特定功能如何達成這個目標過程，僅須將需求告訴這個獨立個體，如果此個體能獨立完成，便可直接將此任務，交付給發號命令者。

物件除了具有屬性和方法外，還要有溝通方式。人與人之間藉由語言的溝通來傳遞訊息。那麼物件之間如何進行訊息的傳遞？以提款機 ATM 來說，放入提款卡，輸入密碼才能跟 ATM 做進一步的溝通。如果將 ATM 視為物件，輸入密碼就是與 ATM 溝通的方法。

由於物件並不會無端憑空產生，每一個物件在程式語言中的實作都必須透過類別（class）的宣告。簡單來說，物件必須有一個可以依據的原型（Prototype），也就是建立一個具有相同特性和行為的物件集合。

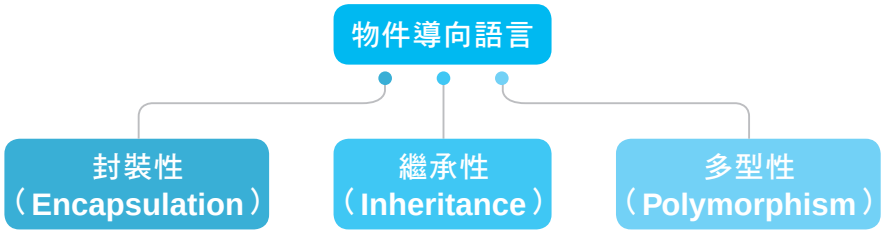
通常初學者剛接觸「類別」與「物件」時，往往會搞不清楚這兩個名詞間的差別。其實如果我們將「類別」當成是「自訂的資料型態」，而「物件」當成「變數」，那就簡單的多了！

基本上，類別其實就是由類似 C/C++ 語言的結構型態衍生而來，二者的差別在於結構型態只能包含資料變數，而類別型態則可擴充到包含處理資料的函數。

在以往的結構化程式設計中，是著重於程式可分解成許多函數來加以執行，其中資料變數與處理資料變數的函數是互相獨立的，因此一旦主程式要求功能變動時，則可能許多函數內的資料與程式碼都需要同步變動，當程式發展到很大時，程式的開發及維護就變得更為困難。但是物件導向程式設計中的類別型態是將函數與資料結合在一起，形成獨立的模組，除了可以加速程式的開發，也使得程式的維護變得容易。

11-1-2 物件導向的特點

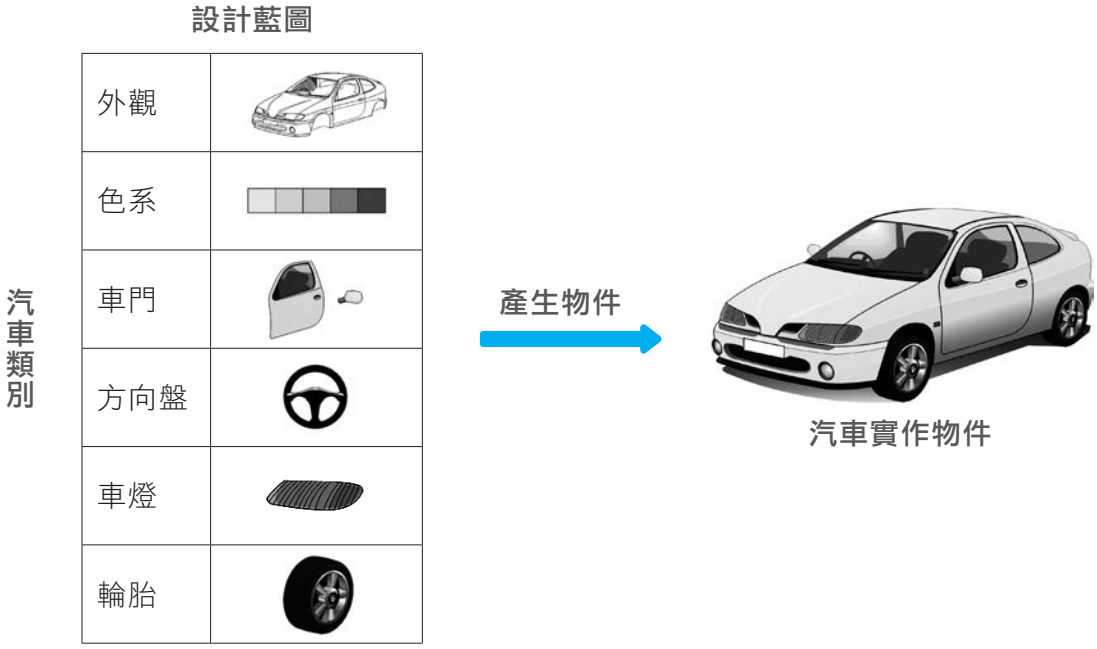
若要模擬真實世界，必須把真實世界的東西抽象化為電腦系統的資料。在物件導向世界裡是以各個物件自行分擔的功能來產生模組化，基本上包含三個基本元素：資料抽象化（封裝）、繼承和多型（動態繫結）。



物件導向程式設計的三種特性

封　裝

封裝（Encapsulation）是利用「類別」（class）來實作「抽象化資料型態」（ADT）。類別是一種用來具體描述物件狀態與行為的資料型態，也可以看成是一個模型或藍圖，按照這個模型或藍圖所生產出來的實體（Instance），就被稱為物件。



類別與物件的關係

所謂「抽象化」，就是將代表事物特徵的資料隱藏起來，並定義「方法」（Method）做為操作這些資料的介面，讓使用者只能接觸到這些方法，而無法直接使用資料，符合了「資訊隱藏」（Information Hiding）的意義，這種自訂的資料型態就稱為『抽象化資料型態』。

繼　承

繼承性稱得上是物件導向語言中最強大的功能，因為它允許程式碼的重複使用（Code Reusability），及表達了樹狀結構中父代與子代的遺傳現象。「繼承」（inheritance）類似現實生活中的遺傳，允許我們去定義一個新的類別來繼承既存的類別（class），進而使用或修改繼承而來的方法（method），並可在子類別中加入新的資料成員與函數成員。

多　型

多型（Polymorphism）也是物件導向設計的重要特性，可讓軟體在發展和維護時，達到充份的延伸性。多型按照英文字面解釋，就是一樣東西同時具有多種不同的型態。在物件導向程式語言中，多型的定義簡單來說是利用類別的繼承架構，先建立一個基礎類別物件。使用者可透過物件的轉型宣告，將此物件向下轉型為衍生類別物件，進而控制所有衍生類別的「同名異式」成員方法。簡單的說，多型最直接的定義就是讓具有繼承關係的不同類別物件，可以呼叫相同名稱的成員函數，並產生不同的反應結果。

物　件

物件可以是抽象的概念或是一個具體的東西，它包括了「資料」（Data）以及其所相應的「運算」（Operations 或稱 Methods），它具有狀態（State）、行為（Behavior）與識別（Identity）。

每一個物件均有其相應的屬性（Attributes）及屬性值（Attribute values）。例如有一個物件稱為學生，「開學」是一個訊息，可傳送給這個物件。而學生有學號、姓名、出生年月日、住址、電話等屬性，目前的屬性值便是其狀態。學生物件的運算行為則有註冊、選修、轉系、畢業等，學號則是學生物件的唯一識別編號（Object Identity, OID）。

類　別

類別（Class）是具有相同結構及行為的物件集合，是許多物件共同特徵的描述或物件的抽象化。例如小明與小華都屬於人這個類別，他們都有出生年月日、血型、身高、體重等類別屬性。類別中的一個物件有時就稱為該類別的一個實例（Instance）。

屬　性

屬性 （Attribute）則是用來描述物件的基本特徵與其所屬的性質，例如：一個人的屬性可能會包括姓名、住址、年齡、出生年月日等。

方　法

方法 （Method）則是物件導向資料庫系統裡物件的動作與行為，我們在此以人為例，不同的職業，其工作內容也就會有所不同，例如：學生的主要工作為讀書，而老師的主要工作則為教書。

有了這些物件導向的基本觀念後，接著本章將以 Python 語言配合物件導向程式設計（OOP）的概念，深入探討類別和物件的實作。

Python 語言的類別機制的主要特性包括：

1. 預設所有的類別與其包含的成員都是 public，使用時不用宣告該類別的型別。
2. 採用多重繼承，衍生類別（Derived class）和基礎類別（base class）的方法可以相同名稱，也能覆寫（override）其所有基礎類別（base class）的任何方法（method）。

11-2 類別與物件

簡單來說，類別提供實作物件的模型，類別就如同蓋房屋之前的規劃藍圖，也可以形容是一種資料型態，但沒有實體。撰寫程式時，必須先定義類別，並設定成員的屬性和方法。

產生類別之後，還要具體化物件，稱為「實體化」（Instantiation），經由實體化的物件，稱為「執行個體」（Instance）或實體。也就是說要產生物件之前，必須先行定義類別，如此一來，才可以透過該類別實作出物件的實體。

11-2-1 定義類別

類別由類別成員（Class Member）組成，類別使用之前要進行宣告，語法如下：

```
class 類別名稱 ():
    # 定義初始化內容
    # 定義方法
```

- class：建立類別的關鍵字，但必須配合冒號「:」產生 suite。
- 類別名稱：用來指定所要建立類別的名稱，類別名稱同樣必須遵守識別字的命名規範。
- 定義方法時，跟先前介紹過的自訂函數一樣，須使用 def 敘述。

以下的語法可以說是最簡單的類別定義方式，它是用來建立一個空類別：

```
class nothing:
    pass
```

上述語法是用來建立 nothing 類別，使用 pass 敘述是表示什麼事都不做。在定義類別的過程中可以加入屬性和方法（Method），再以物件來存取其屬性和方法。

11-2-2 類別實體化

要將類別實體化就是產生物件，有了物件才可以進一步存取類別裡所定義的屬性和方法。語法如下：

```
物件 = 類別名稱 ( 引數串列 )
物件 . 屬性
物件 . 方法 ()
```

其中的物件名稱同樣得遵守識別字的規範。引數串列可依據物件初始化做選擇。所以以下的程式範例中，可產生兩個物件 boy1 和 boy2。

```
boy1 = Person()
boy2 = Person()
```

上述例句可視為「建立 Person 類別的實體，並將該物件指派給變數 boy1 和 boy2」。

Python 定義方法的第一個參數必須是自己，習慣上使用 self 做表達，它代表建立類別後實體化的物件。這是一個相當重要的特性，這和其它程式語言是有所不同的。self 有點類似其他語言中的 this，指向物件自己本身，在定義類別的所有方法都必須宣告它，以下以一個簡單的範例來說明在 Python 程式語言中定義類別的用法。

〈定義類別〉

檔名 define_class.py

```
01  class Person:
02      # 定義方法一：取得姓名和年齡
03      def setData(self, name, age):
04          self.name = name
05          self.age = age
06      # 定義方法二：輸出姓名和年齡
07      def showData(self):
08          print('姓名:{0:6s}, 年齡:{1:4s}'.format(
09              self.name, self.age))
10  # 產生物件
11  boy1=Person()      # 物件 1
12  boy1.setData('John', '16')
13  boy1.showData()    # 呼叫方法
14  boy2=Person()      # 物件 2
15  boy2.setData('Andy', '14')
16  boy2.showData()
```

執行結果 >>

```
姓名:John  , 年齡:16
姓名:Andy  , 年齡:14
```

</> 程式解說 >>

- 第 01~09 行：建立 Person 類別，定義了二個方法。
- 第 03~05 行：定義第一個方法，用來取得物件的屬性。跟定義函數相同，要使用 def 敘述為開頭；方法中的第一個參數必須 self 敘述，它類似其他程式語言的 this。如果未加 self 敘述，則以物件呼叫此方法時會發生 TypeError。
- 第 04~05 行：將傳入的參數透過 self 敘述來作為物件的屬性。
- 第 07~09 行：定義第二個方法，用它來輸出物件的相關屬性。
- 第 11~13 行：產生 boy1 物件並呼叫其方法。
- 第 14~16 行：產生 boy2 物件並呼叫其方法。

此外，前述範例還使用一個特別的字 self（此處以 self 敘述來稱呼它）。通常 name 和 age 只是變數，它們定義於方法內，屬於區域變數，離開此適用範圍（Scope）就結束了生命週期。由於 self 不做任何引數的傳遞，但藉由 self 敘述的加入，它們成了物件變數，能讓方法之外的物件來存取。

```
def setData(self, name, age):
    self.name = name
    self.age = age
```

所以將參數 name 的值傳給「self.name」，會讓一個普通的變數轉變成物件變數（也就是屬性），並由物件來存取。定義類別之後，還能依據需求傳入不同型別的資料，以下述簡述做說明。

〈在類別中依據需求傳入不同型別的資料〉

檔名 define_class1.py

```
01  class Money:
02      def setValue(self,amount):   # 第一種方法
03          self.amount =amount
04      def showValue(self):         # 第二種方法
05          print(" 錢的總數為 :",self.amount)
06  s1 = Money()                     # 第一個物件
07  s1.setValue(" 參萬捌仟元整 ")     # 呼叫方法時傳入字串
08  s1.showValue()
09  s2 = Money()                     # 第二個物件
10  s2.setValue(34500)               # 呼叫方法時傳入浮點數值
11  s2.showValue()
```

執行結果 »

```
錢的總數為: 參萬捌仟元整
錢的總數為: 34500
```

程式解說 >>

- 第 02~03 行：定義 setValue() 方法，藉由 self 將傳入的參數 amount 設為物件的屬性。
- 第 04~05 行：定義 showValue () 方法，輸出此物件的屬性。
- 第 06~11 行：第一個物件 s1 是以字串做傳遞；第二個物件 s2 是以浮點數為參數值。這兩個不同的物件傳入不同型別的資料，這是因為 Python 採動態型別。

還有一種技巧要加以說明，定義類別時，也能夠透過該類別內的方法來傳入參數，完成計算後將所計算的值回傳，請看以下的範例：

〈在類別中的方法中傳入參數計算值〉

檔名 return.py

```
01  class MySum():
02      def add(self,n1,n2,n3):      # 第一種方法
03          return n1+n2+n3
04
05  b1=MySum()
06  print(" 三數總和 =",b1.add(104,55,73))
```

執行結果 >>

```
三數總和= 232
```

程式解說 >>

- 第 01~03 行：：建立 MySum 類別，這個類別只定義一個方法， 該類別可以傳入 3 個參數值，並加 3 個參加以加總後回傳。
- 第 05~06 行：產生新物件，呼叫 add() 方法傳入 3 個參數。

11-2-3 將物件初始化 __init__() 方法

剛才介紹是物件的屬性與方法，接著要談可以為物件進行初始化工作的 __init__() 方法，當建立物件時，這個特殊方法會為物件進行初始化的工作，例如設定初始值或進行資料庫連線等工作。這個方法的第一個參數是 self，是被用來指向剛建立的物件本身。但是如果沒有利用 __init__() 方法設定初值，此處以計算 Rentangle 類別為例，通常會先建立 Rentangle 物件，並設定其長與寬的初始值，再於程式中將長與寬設定成要計算的值。雖然說這種方式也可以將長與寬設定成要計算的值，但比較好的作法是在建立物件時，就透過 __init__() 方法為該物件設定長與寬的初值。以下例子將為各位說明如何透過 __init__() 方法為物件設定初值：

〈透過 __init__() 方法為物件設定初值〉

檔名 init.py

```
01  class Rectangle:
02      def __init__(self, length=10, width=5):
03          self.length=length
04          self.width=width
05
06      def getArea(self):
07          return self.length* self.width
08
09  R1=Rectangle()
10  print(" 透過 init() 方法預設值的面積： ",R1.getArea())
11
12  R2= Rectangle(125,6)
13  print(" 透過 init() 方法傳入參數的面積： ",R2.getArea())
```

執行結果 »

```
透過init()方法預設值的面積:  50
透過init()方法傳入參數的面積:  750
```

程式解說

- 第 02~04 行：建立定義一個 init() 方法，並設定預設值。但是如果有傳入參數，透過 __init__() 方法將長與寬以所傳入的參數設定初值，就可以達到為物件設定初值的目的。

11-2-4 匿名物件

通常宣告類別後，會將類別實體化為物件，並將物件指派給變數，再透過這個變數來存取物件。在 Python 中有一項重要特性，那就是每個東西都是物件，所以寫程式時，也可以在不需要將物件指派給變數的情況下去使用物件，這就是一種稱為匿名物件（anonymous object）的程式設計技巧。

以下程式將改寫上一個範例，直接以匿名物件的方式去存取物件，就不用像上一個例子，還要先將物件指派給一個變數，再透過變數來存取該物件。

〈匿名物件實作〉

檔名 anonymous.py

```
01 class Rectangle:
02     def __init__(self, length=10, width=5):
03         self.length=length
04         self.width=width
05
06     def getArea(self):
07         return self.length* self.width
08
09 print(" 透過 init() 方法預設值的面積： ",Rectangle().getArea())
10
11 print(" 透過 init() 方法傳入參數的面積： ",Rectangle(125,6).
   getArea())
```

執行結果 >>

```
透過init()方法預設值的面積:  50
透過init()方法傳入參數的面積:  750
```

程式解說 >>

- 第 09 行：直接以匿名物件的方式去存取物件，此處是求取 init() 方法預設值的面積。
- 第 11 行：直接以匿名物件的方式去存取物件，透過 init() 方法傳入參數的面積。

11-2-5 私有屬性與方法

前面所示範的例子，類別外部的指令可以直接存取類別內部的資料，這種作法的風險，容易造成內部資料被不當修改。萬一某些類別內部的重要不希望被任何人擅自修改，這種程式寫作方式，就不算是一種好的方式。

為了達到保護類別內部資料的目的，我們可以將這些重要資料設定為私有屬性，如此一來，就可以限定該資料只能在類別內部的指令敘述可以存取。當類別外部想要存取這些私有的屬性資料時，並不能夠直接由類別外部進行存取，必須透過該類別所提供的公用方法。這種作法較為安全，可以有效保護一些不想被更動的重要資料。

想要指定屬性為私有的，只要在該屬性名稱前面加上兩個下底線 __，就代表該屬性名稱為私有屬性。但請特別注意，在名稱後方不能有下底線，例如 __age 是私有屬性，但是 __age__ 就不是私有屬性。以下為私有屬性的應用例子，其中 __rate 是私有屬性，當類別外部想要存取 __rate 私有屬性的資料時，並不能夠直接由類別外部進行存取，必須透過該類別所提供的 getRate() 公用方法。

〈私有屬性的應用例子 I〉

檔名 private.py

```
01  class Discount:
02      def __init__(self, r=0.9):
03          self.__rate=r
04
05      def getRate(self):
06          return self.__rate
07
08      def howMuch(self):
09          return money*self.__rate
10  money=10000
11  obj1=Discount(0.7)
12  print(" 此件商品的原有價格為 ",money," 元 ")
13  print(" 這件商品的折扣為 ", obj1.getRate())
14  print(" 打折扣後商品價格為 ", obj1.howMuch(), " 元 ")
```

執行結果 »

```
此件商品的原有價格為 10000 元
這件商品的折扣為 0.7
打折扣後商品價格為 7000.0 元
```

從上面的執行結果中，各位可以看出由於 __rate 是一個私有屬性，所以類別外部的指令，無法直接存取該私有屬性，必須透過類別碼中的 getRate() 方法，才能取得實際的年齡。如果各位試圖將上述程式碼中的第 13 行，改寫成直接存取私有屬性 __rate，如以下的程式碼：

〈私有屬性的應用例子 II〉

檔名 private1.py

```
01  class Discount:
02      def __init__(self, r=0.9):
03          self.__rate=r
04
05      def getRate(self):
06          return self.__rate
07
08      def howMuch(self):
09          return money*self.__rate
10  money=10000
11  obj1=Discount(0.7)
12  print(" 此件商品的原有價格為 ",money," 元 ")
13  print(" 這件商品的折扣為 ", obj1.__rate)
14  print(" 打折扣後商品價格為 ", obj1.howMuch(), " 元 ")
```

執行結果 »

上述程式的執行結果，就會出現 AttributeError: ‘Discount’ object has no attribute ‘__rate’ 的錯誤訊息。如下圖所示：

```
    print("這件商品的折扣為", obj1.__rate)

AttributeError: 'Discount' object has no attribute '__rate'
```

除了屬性成員可以設定為私有外，類別的方法成員如果要設定為私有，作法和指定私有屬性類似，只要在方法名稱前加上兩個下底線 __，而且名稱之後不能有下底線。一旦被宣告為私有方法後，該方法只能被類別內部的敘述呼叫，在類別外部不能直接呼叫該私有方法。其實不論是私有屬性或私有方法，其目的就是幫助程式設計人員將需要保護的屬性與方法加上一道保護，以達到資訊隱藏的目的。

11-3 繼承

前面提過，基本上繼承就類似遺傳的觀念，例如父母親生下子女，如果沒有例外情況，則子女一定會遺傳到父母的某些特徵。當物件導向技術以這種生活實例去定義其功能時，就稱為繼承（Inheritance）。

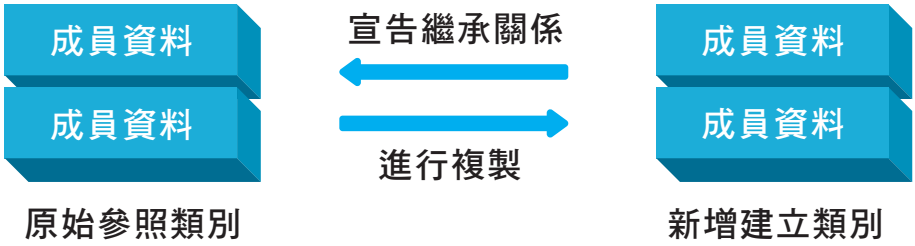
繼承是物件導向程式設計相當核心的理念，可以依照既有的類別，透過繼承的機制衍生出新類別，而這種程式碼再利用的觀念，可以省去重複撰寫相同程式碼的大量時間。

繼承如果以程式語言的觀點來解釋，就是一種「承接基礎類別的實體變數及方法」的概念，更嚴謹來定義可以說是：「類別之間具有階層關係，基礎類別（Super-class）建立通用類別，衍生類別（Sub-class）則接收基礎類別的類別成員，並在通用類別的基礎上發展出不同的類別成員」。

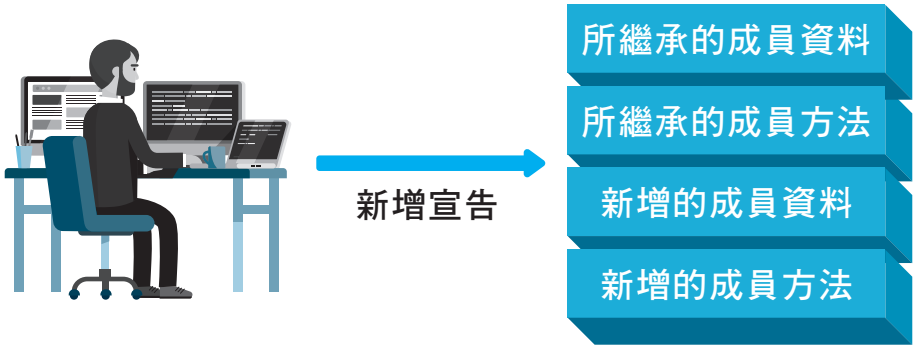
繼承除了可重複利用已開發的類別之外，另外一項好處在於維持物件封裝的特性。這是因為繼承時，不容易改變已經設計好的類別，這樣可以降低類別設計的錯誤發生。

在 Python 中，已建立好的類別稱為「基礎類別」（Base Class），而經由繼承所產生的新類別就稱之為「衍生類別」（Derived Class）。通常會將基礎類別稱為父類別（Parent Class），而衍生類別稱為子類別（Child Class）。類別之間如果要有繼承（Inheritance）的關係，必須先建立好基礎類別，也就是父類別（Super-class），然後衍生類別，也就是子類別（Sub-class）。

從另外一個思考角度來看，我們可以把繼承單純地視為一種複製（copy）的動作。換句話說，當開發人員以繼承機制宣告新類別時，會先將所參照的原始類別內所有成員，完整地寫入新增類別之中。就如同下面類別繼承關係圖所示：



在新增類別內完整地包含了原始參照類別的所有類別成員，使用者可直接於新增類別中，針對這些成員進行呼叫或存取動作。當然除了原始類別的各個成員外，也可以在新增類別中依需求來新增必要的資料與方法。如下圖所示：



簡單來說，繼承有兩項優點：

- 提高軟體的重複使用性
- 程式設計人員可以善用繼承標準函數庫及第三方函數庫

當程式人員花了心血及許多時間除錯，好不容易完成父類別的功能，但是在一些特殊的情況下，這個父類別無法處理，一種作法就是直接修改父類別的功能，但這種作法的風險就是有可能動到原先已設計好的功能，萬一不小心，又得花上不少時間進行除錯。

其實繼承並不是一種很難理解的觀念，但是許多程式設計師在實作繼承的程式碼時，通常比較不知道如何規劃類別之間的繼承關係。也就是說，大部分初學者，對如何區分哪些功能該納入父類別，又哪些功能該納入子類別，

往往找不到一種好的設計方式，造成在使用繼承時，無法很貼切表達一種完善的繼承架構。接著我們將列出一些參考原則，希望有助各位在設計繼承架構的程式，能有參考的依據。

原則上，父類別定義較廣義的功能，而子類別則定義較狹義的功能，例如：父類別 Biology 泛指生物，而其子類別 Animal、Plant 分別表示動物與植物。這是因為無論是動物或是植物，都屬於生物的一種，所以子類別 Animal、Plant 均繼承了父類別 Biology 的非私有屬性或方法。在子類別中除了繼承這些成員外，還可以加入新的成員或覆蓋繼承自父類別的方法。

11-3-1 單一繼承與多重繼承

單一繼承（Single Inheritance）是指衍生類別只繼承一個基礎類別。在 Python 中要使用繼承機制定義子類別的語法格式如下：

```
class ChildClass(ParentClass):
    指令敘述
```

或

```
class ChildClass(ParentClass1, ParentClass2,…):
    指令敘述
```

第一種語法子類別只繼承單一父類別，這是單一繼承的語法格式，而第二種語法子類別繼承超過一個以上的父類別，它則是一種多種繼承的語法格式。接下來我們先來看單一繼承的實例，這個例子會先定義 Vehicle 基礎類別，接著會以繼承的語法格式去定義 Bike 衍生類別，請看以下的程式碼說明：

〈單一繼承〉

檔名 single.py

```
01  # 類別的繼承
02
03  class Vehicle: # 基礎類別
04      def move(self):
05          print(' 我是可以移動的交動工具 ')
06
07  class Bike(Vehicle): # 衍生類別
08      pass
09
10  # 產生子類別實體
11  giant = Bike()
12  giant.move()
```

執行結果 >>

```
我是可以移動的交動工具
```

</> 程式解說 >>

- 第 03~05 行：定義 Vehicle 基礎類別。
- 第 07~08 行：定義 Bike 衍生類別。
- 第 11 行：產生 giant 子類別實體。
- 第 12 行：giant 子類別實體呼叫繼承自 Vehicle 基礎類別的 move() 方法。

其實子類別還可以擴展父類別的方法而非完全取代它，我們可以將上例修改如下，這個範例中除了呼叫父類別的方法，並依自己的需求擴展了子類別的方法：

〈在子類別中擴展父類別的方法 I〉

檔名 single1.py

```
01  # 類別的繼承
02
03  class Vehicle: # 基礎類別
04      def move(self):
05          print(' 我是可以移動的交動工具 ')
06
07  class Bike(Vehicle): # 衍生類別
08      def move(self):
09          Vehicle.move(self)
10          print(' 但我必須以腳踏的方式來移動 ')
11
12  # 產生子類別實體
13  giant = Bike()
14  giant.move()
```

執行結果 >>

```
我是可以移動的交動工具
但我必須以腳踏的方式來移動
```

</> 程式解說 >>

- 第 03~05 行：定義 Vehicle 基礎類別。
- 第 07~10 行：定義 Bike 衍生類別，但在這個衍生類別中擴展父類別的方法。
- 第 13 行：產生 giant 子類別實體。
- 第 14 行：giant 子類別實體呼叫繼承自 Vehicle 基礎類別的 move() 方法。

另外，Python 支援鏈狀繼承，例如下圖中，類別 Father 繼承類別 GrandFather，而類別 Son 繼承類別 Father。

〈在子類別中擴展父類別的方法 II〉

檔名 chain.py

```
01  # 類別的繼承
02
03  class Vehicle:          # 基礎類別
04      def move(self):
05          print(' 我是可以移動的交動工具 ')
06
07  class Bike(Vehicle):    # 衍生類別
08      def move(self):
09          Vehicle.move(self)
10          print(' 但我必須以腳踏的方式來移動 ')
11
12  class Child_Bike(Bike): # 孫類別
13      def move(self):
14          Bike.move(self)
15          print(' 為了小朋友的安全我特別加上輔助輪 ')
16
17  # 產生孫類別實體
18  small_giant = Child_Bike()
19  small_giant.move()
```

執行結果 »

```
我是可以移動的交動工具
但我必須以腳踏的方式來移動
為了小朋友的安全我特別加上輔助輪
```

</> 程式解說 »

- 第 03~05 行：定義 Vehicle 基礎類別。

- 第 07~10 行：定義 Bike 衍生類別，但在這個衍生類別中擴展父類別的方法。
- 第 12~15 行：定義 Child_Bike 孫類別，但在這個孫類別中擴展繼承類別的方法。
- 第 18 行：產生 small_giant 孫類別實體。
- 第 19 行：small_giant 孫類別實體呼叫 move() 方法。

不僅如此，一個子類別可以同時繼承多個父類別，我們以一個簡單的實例來示範 Python 的多重繼承的作法：

〈多重繼承〉

檔名 multiple.py

```
01  # 類別的繼承
02
03  class Animal:                # 基礎類別一
04      def feature1(self):
05          print(' 動物是多細胞真核生命體中的一大類群 ')
06
07  class Human:                 # 基礎類別二
08      def feature2(self):
09          print(' 人類是一種具有情感高等智商動物 ')
10
11  class Boy(Animal, Human):    # 衍生類別
12      pass
13
14  # 產生子類別實體
15  Andy = Boy()
16  Andy.feature1()
17  Andy.feature2()
```

執行結果 >>

```
動物是多細胞真核生命體中的一大類群
人類是一種具有情感高等智商動物
```

</> 程式解說 >>

- 第 03~05 行：定義 Animal 基礎類別一。
- 第 07~09 行：定義 Human 基礎類別二。
- 第 11~12 行：子類別 Boy 同時繼承 Animal 父類別及 Human 父類別。
- 第 15 行：產生 Andy 子類別實體。
- 第 16 行：呼叫繼承自 Animal 基礎類別一的 feature1() 方法。
- 第 17 行：呼叫繼承自 Human 基礎類別二的 feature2() 方法。

11-3-2 覆寫

覆寫（override）是一種重新定義父類別中繼承而來的方法，雖然在子類別中重新改寫所繼承的父類別的方法，但卻不會影響到父類別中被覆寫的方法。先以一個例子來說明如何在子類別覆寫父類別的方法。

〈覆寫的實作〉

檔名 override.py

```
# 子類別覆寫父類別的方法
class Discount():                # 父類別
    def rate(self, total):
        self.price = total
        if self.price >= 20000:
            print('平日假日的折扣為 9 折：', end = ' ')
```

```
07              return total * 0.9
08
09  class Festival(Discount):      # 子類別
10      def rate(self, total):     # 覆寫 rate 方法
11          self.price = total
12          if self.price >= 50000:
13              print('節慶特優惠折扣為 5 折：', end = ' ')
14              return total * 0.5
15
16  Jane = Discount()              # 建立父類別物件
17  print(Jane.rate(78000))
18
19  Mary = Festival()              # 建立子類別物件
20  print(Mary.rate(78000))
```

執行結果 »

```
平日假日的折扣為9折： 70200.0
節慶特優惠折扣為5折： 39000.0
```

</> 程式解說 »

- 第 02~07 行：定義 Discount 父類別。
- 第 09~14 行：定義 Festival 子類別，這個子類別會覆寫 rate() 方法。
- 第 16~17 行：建立 Jane 父類別物件，並呼叫父類別中的 rate() 方法，再將其結果輸出。
- 第 19~20 行：建立 Mary 子類別物件，並呼叫子類別中覆寫 rate() 方法，再將其結果輸出。

11-3-3 繼承相關函數

在 Python 中與繼承較相關的函數有 isinstance() 及 issubclass()。

isinstance()

isinstance() 是 Python 中的一個內建函數，其語法如下：

```
isinstance(object, classinfo)
```

這個內建函數的功能是如果參數 object 是參數 classinfo 所指定的類別，或者參數 object 是參數 classinfo 類的子類別的一個物件，傳回 True，否則傳回 False。

例如：

```
>>> num = 8  # 設定數值 8 給變數 num
>>> isinstance (num,int)  # 數值 8 是 int 類別的物件
True
>>> isinstance (num,str)  # 數值 8 不是 str 類別的物件
False
>>> num = "Happy"
>>> isinstance(num,str)
True
>>> isinstance(num,int)
False
```

issubclass()

issubclass() 是 Python 中的一個內建函數，其語法如下：

```
issubclass(object, classinfo)
```

這個內建函數的功能是如果參數 object 是參數 classinfo 所指定的子類別，則傳回 True，否則傳回 False。請看下例的說明：

〈繼承函數 issubclass()〉

檔名 issubclass.py

```
01  class Parents:
02      def hello(self):
03          print("I am in Parents")
04
05  class Child(Parents):
06      def hello(self):
07          print("I am in Child")
08
09  print(issubclass(Child, object))
10  print(issubclass(Child, Parents))
```

執行結果 »

```
True
True
```

</> 程式解說 »

- 第 09 行：判斷 Child 是否為 object 的子類別，並將其結果值輸出。
- 第 10 行：判斷 Child 是否為 Parents 的子類別，並將其結果值輸出。

11-3-4 多型

一般在程式裏，常常會在基礎類別或是衍生類別中宣告相同名稱但不同功能的 public 成員函數。這時可以把這些函數稱做同名異式或是多型（polymorphism）。例如各位已建立某個基礎類別的成員函數 open()，以及建立多個由基礎類別所衍生出來的成員 open() 函數。當程式開始執行時，可依據打算開啟物品的編號來指定要使用哪個衍生類別的 open() 成員函數來開啟該物品。多型是物件導向程式語言的一項主要特性，簡單來說，Python 多型可以讓子類別的物件以父類別來處理。如何實作多型，請看以下的範例說明：

〈多型實作〉

檔名 polymorphism.py

```
01 # 定義多型
02 class Product():                          # 父類別
03     def __init__(self, name, price):
04         self.name = name
05         self.price = price
06
07     def profit(self):
08         return self.price
09
10     def show(self):
11         return self.name
12
13 class Book(Product):                      # 子類別
14     def profit(self):
15         return self.price * 1.2
16
17 class Software(Product):                  # 子類別
18     def profit(self):
19         return self.price *1.5
20
21 obj1 = Product('一般商品', 20000)      # 父類別物件
22 print('{:8s} 利潤 {:,}'.format(obj1.show(), obj1.profit()))
23
24 obj2 = Book('百科全書', 48000)         # 子類別物件
25 print('{:8s} 利潤 {:,}'.format(obj2.show(), obj2.profit()))
26
27 obj3 = Software('電腦軟體', 120000)    # 子類別物件
28 print('{:8s} 利潤 {:,}'.format(obj3.show(), obj3.profit()))
```

執行結果 »

```
一般商品     利潤 20,000
百科全書     利潤 57,600.0
電腦軟體     利潤 180,000.0
```

程式解說

- 第 02~11 行：定義 Product 父類別，其中定義的 profit() 方法會回傳 price 的值。
- 第 13~15 行：定義 Book 子類別，其中定義的 profit() 方法會回傳 price * 1.2 的值。
- 第 17~19 行：定義 Software 子類別，其中定義的 profit() 方法會回傳 price *1.5 的值。

11-3-5 組合

組合（composition）在繼承機制中是 has_a 的關係，例如公司是由會議日期、會議成員和開會地點組合而成，就利用這個概念配合 Python 的程式碼，為各位示範如何撰寫一個組合的程式。

〈組合程式實作〉

檔名 composition.py

```
01 from datetime import date
02
03 # 組合的簡易作法
04 class Employee: # 公司員工
05     def __init__(self, *title):
06         self.title = title
07
08 class Meeting: # 會議
09     def __init__(self, topic, tday):
10         self.topic = topic
11         self.today = tday
12         print(' 開會日期：', self.today)
13         print(' 開會地點：', self.topic)
14
```

```
15  class Company: # 公司
16      def __init__(self, Employee, Meeting):
17          self.Employee = Employee
18          self.Meeting = Meeting
19
20      def show(self):
21          print(' 參與會議人員 :', self.Employee.title)
22
23  tday = date.today()# 取得今天日期
24  #Employee 物件
25  member = Employee(' 研發部主管 ',' 會計部主管 ',' 業務部主管 ',' 行銷
    部主管 ')
26  place = Meeting(' 公司總部 805 會議室 ', tday) # 開會地點
27  obj = Company(member, place)#Company 實體
28  obj.show()# 呼叫方法
```

執行結果 »

```
開會日期： 2018-02-06
開會地點： 公司總部805會議室
參與會議人員: ('研發部主管', '會計部主管', '業務部主管', '行銷部主管')
```

</> 程式解說 »

- 第 04~06 行：定義 Employee 公司員工類別。
- 第 08~13 行：定義 Meeting 會議類別。
- 第 15~21 行：定義 Company 公司類別，Company 公司類別是由 Employee 公司員工類別和 Meeting 會議類別組合而成。

11-4 上機實作演練－加退選程式設計實務

實作一個 Python 程式，該程式有加退選課程的程式概念。該程式可以允許學生以物件導向程式設計的作法加選課程及退選課程，最後再列出該學生的姓名及所有選修課程清單。這支程式必須先行定義 ElectiveCourses 類別，在這個類別中，除了以 __init__() 方法設定姓名 name 及課程 course 的初始值外，還定義了以下幾種方法：

- def get_name(self)：用來回傳學生的姓名。
- def addcourse(self, course)：用來將傳入的參數內容加入到選修課程 course 的串列內容。
- def dropcourse(self, course)：用來將傳入的參數內容從選修課程 course 的串列內容中移除。
- def getcourse(self)：用來回傳選修課程的串列內容。

其執行結果外觀如下：

```
本學期 陳元俊 同學的選修課程有：
['機器學習', '人工智慧', '大數據', '自動控制', 'Python程式設計']
```

參考解答：

〈加退選課程實作〉

檔名 add_drop.py

```
01 class ElectiveCourses():
02     def __init__(self, name):
03         self.__name = name
04         self.__course = []
05
06     def get_name(self):
07         return self.__name
08
09     def addcourse(self, course):
10         self.__course.append(course)
11
12     def dropcourse(self, course):
13         self.__course.remove(course)
14
15     def getcourse(self):
16         return self.__course
17
18 student = ElectiveCourses(" 陳元俊 ")
19 student.addcourse(" 機器學習 ")
20 student.addcourse(" 人工智慧 ")
21 student.addcourse(" 大數據 ")
22 student.addcourse(" 電子商務 ")
23 student.addcourse(" 物聯網 ")
24 student.addcourse(" 自動控制 ")
25 student.addcourse("Python 程式設計 ")
26 student.dropcourse(" 物聯網 ")
27 student.dropcourse(" 電子商務 ")
28 print(" 本學期 {0} 同學的選修課程有：".format(student.get_name()))
29 print(student.getcourse())
```

重點回顧

1. 物件導向程式設計能以一種更生活化、可讀性更高的設計觀念來進行，並且所開發出來的程式也較容易擴充、修改及維護。
2. 在傳統程式設計的方法中，主要是以「結構化程式設計」為主，就是「由上而下設計」與「模組化設計」。
3. 物件擁有狀態（state）、行為（behavior）（或稱使用方法）。狀態代表了物件所屬特徵，行為則代表物件所具有的功能。
4. 每一個物件在程式語言中的實作都必須透過類別（class）的宣告。
5. 結構型態只能包含資料變數，而類別型態則可擴充到包含處理資料的函數。
6. 物件導向的特點包含三個基本元素：資料抽象化（封裝）、繼承和多型（動態繫結）。
7. 封裝（Encapsulation）是利用「類別」來實作「抽象化資料型態」（ADT）。所謂「抽象化」，是讓使用者只能接觸到這些方法（函數），而無法直接使用資料。
8. 繼承（inheritance）允許我們去定義一個新的類別來繼承既存的類別（class），進而使用或修改繼承而來的方法（method），並可在子類別中加入新的資料成員與函數成員。
9. 多型最直接的定義就是讓具有繼承關係的不同類別物件，可以呼叫相同名稱的成員函數，並產生不同的反應結果。
10. Python 預設所有的類別與其包含的成員都是 public，使用時不用宣告該類別的型別。
11. Python 採多重繼承，衍生類別（Derived class）可以和基礎類別（base class）的方法同名稱，也能覆寫（override）其所有基礎類別（base class）的任何方法（method）。
12. 產生類別之後，還要具體化物件，稱為「實體化」（Instantiation），經由實體化的物件，稱為「執行個體」（Instance）或實體。

13. Python 程式語言定義方法的第一個參數必須是自己，習慣上使用 self 做表達，它代表建立類別後實體化的物件。

14. _init__() 方法會為物件進行初始化的工作，例如設定初始值，這個方法的第一個參數是 self，是被用來指向剛建立的物件本身。

15. 在 Python 程式語中的特性中，每個東西都是物件，所以我們也可以在不需要將物件指派給變數的情況下去使用物件，這種作法就稱為匿名物件（anonymous object）。

16. 要指定屬性為私有的，必須將屬性名稱前面加上兩個下底線 __，但請特別注意，在名稱後方不能有下底線，例如 __age 是私有屬性，但是 __age__ 就不是私有屬性。

17. 不論是私有屬性或私有方法，其目的就是幫助程式設計人員保護資料，以達到資訊隱藏的目的。

18. 繼承是物件導向程式設計相當核心的理念，我們可以在既有的類別，透過繼承衍生出新的類別，而這種程式碼再利用的觀念，省去我們重複撰寫相同的程式碼。

19. 已建立好的類別稱之為「基礎類別」（Base Class），而經由繼承所產生的新類別就稱之為「衍生類別」（Derived Class）。

20. 繼承的優點：提高軟體的重複使用性、善用繼承標準函數庫及第三方函數庫。

21. 單一繼承（Single Inheritance）是指衍生類別只繼承單獨一個基本類別。

22. 覆寫可以在子類別中重新改寫所繼承的父類別的方法，但卻不會影響到父類別中被覆寫的方法。

23. 在 Python 中與繼承較相關的函數有 isinstance() 及 issubclass()。

24. 組合（composition）在繼承機制中是 has_a 的關係。

Chapter 11 學習評量

選擇題

(　　) 1. 關於物件導向的描述，下列何者不正確？
(A) 最早的雛形早在 1960 年的 Simula 語言
(B) SmallTalk 語言引入「訊息」（message）
(C) 是一種「由上而下設計」 與「模組化設計」的設計模式
(D) 以一種生活化、可讀性更高的設計觀念來進行程式開發工作。

(　　) 2. 關於類別與物件的描述，下列何者不正確？
(A) 物件擁有狀態（state）、行為（behavior）（或稱使用方法）
(B) 物件除了具有屬性和方法外，還要有溝通方式
(C) 物件必須有一個可以依據的原型（Prototype）
(D) 類別外部的指令能存取類別內部私有的資料成員與方法。

(　　) 3. 關於 Python 物件導向特點的描述，下列何者不正確？
(A) 封裝也符合了資訊隱藏的意義
(B) Python 預設所有的類別與其包含的成員都是 public
(C) 採多重繼承
(D) 產生物件之後，還要具體化類別，這個過程稱為「實體化」（Instantiation）。

填充題

1. 結構化程式設計就是_____設計與_____設計。
2. 每一個物件在程式語言中的實作都必須透過_____的宣告。
3. 物件導向的三項主要特點：_____、_____和_____。
4. Python 預設所有的類別與其包含的成員都是_____。
5. 產生類別之後，還要具體化物件，稱為_____。

6. _init__() 方法的第一個參數是_____，是被用來指向剛建立的物件本身。

7. 要指定屬性為私有的，要將屬性名稱前面加上_____。

8. 經由繼承所產生的新類別就稱之為_____類別。

9. _____可以在子類別中重新改寫所繼承的父類別的方法，但卻不會影響到父類別中被覆寫的方法。

10. 組合（composition）在繼承機制中是_____的關係。

簡答題

1. 簡述傳統結構化程式設計與物件導向程式設計不同？

2. 簡述物件導向程式設計封裝的特點。

3. 簡述多型的定義。

4. 請簡單陳述 __init__() 方法在 Python 語言物件導向程式設計所扮演的角色。

5. 簡述匿名物件（anonymous object）的程式設計技巧。

Chapter 12

開發 GUI 視窗程式

現在所有程式設計都脫離不了視窗程式設計，身為一位 Python 設計者，各位遲早都會用到圖形介面來開發應用，開始使用 GUI 設計的步驟並不複雜，Python 提供了多種套件來支援 GUI 介面的撰寫，很方便建立完整與功能健全的 GUI 用戶界面，本章中將會開始跟各位討論如何利用 Python 來開發 GUI 視窗程式的完整過程。

學習大綱

- 建立主視窗
- 佈局（Layout）方式
- 標籤（Label）元件
- 按鈕（Button）元件
- 文字編輯（Entry）元件
- 多行文字（Text）元件
- Radiobutton（單選按鈕）
- 核取按鈕（Checkbutton）元件
- 捲軸（Scrollbar）元件
- 對話方塊
- 功能表
- 繪製圖形（Canvas）

Python 提供了多種套件來支援 GUI 介面的撰寫，不過本章內容會以 GUI tkinter 套件為主，除了主視窗物件外，尚有其他 GUI 視窗程式元件，例如：Label、Entry、Text、Button、Checkbutton、Radiobutton。另外還包含標準對話方塊、以 Menu 元件製作功能表和快捷功能表及以 Canvas 元件作圖形的繪製，最後本章將使用 GUI 介面來實作簡易計算機功能的程式。

12-1 GUI tkinter 套件

Graphical User Interface（圖形使用者介面，簡稱 GUI）是指使用圖形方式顯示使用者操作介面，圖形介面和命令列介面比較起來，不管是在操作上或是視覺上都更容易被使用者接受。tkinter（發音成 tk-inter）是 Tool Kit Interface 的縮寫，tkinter 套件是一種內建標準模組，在安裝 Python 3 時就會一併安裝此套件。另外，tkinter 允許使用者在不同的作業平台下建構 GUI 介面，以下先來看看它的用法。

12-1-1 匯入 tkinter 套件

與前面所介紹的模組一樣，使用前必須先匯入模組。

```
import tkinter
```

各位也可以替它取個別名稱，使用時就更為方便，例如想以 tk 作為 tkinter 的別名，就可以參考如下的匯入語法：

```
import tkinter as tk
```

如果所匯入的模組有設定別名，當使用 tkinter 提供的函數時，就可以直接使用別名呼叫，格式如下：

```
tk.函數名稱()
```

12-1-2 建立主視窗

GUI 介面的最外層是一個視窗物件，我們稱為主視窗，建立好主視窗元件後，就可以在主視窗加入像是標籤、按鈕等視窗內部的元件，要建立主視窗語法如下：

```
主視窗名稱 = tk.Tk()
```

例如視窗名稱為 win，建立主視窗的語法如下：

```
win = tk.Tk()
```

主視窗常用的方法有：

方法	說明	實例
geometry(" 寬 x 高 ")	設定主視窗尺寸 （「x」是小寫字母 x）	win.geometry("300x100") 表示設定視窗大小為寬度 300 像素，高度 100 像素。
title(text)	將參數 text 所指定的文字設定為主視窗標題列文字，例如右邊實例會於視窗的標題列秀出「視窗標題」的文字。	win.title(" 視窗標題 ")

視窗的尺寸大小並不是一定要設定，如果沒有提供主視窗尺寸的資訊，預設會以視窗內部的元件來決定視窗的寬與高。另外，如果沒有設定視窗標題，預設為「tk」。

當主視窗設定完成之後，還必須在程式最後使用 mainloop() 方法讓程式進入循環偵聽模式來偵測使用者觸發的「事件（Event）」，一直到關閉視窗為止。語法如下：

Tips

所謂「事件」是由使用者的動作或系統所觸發的訊號。舉例來說，當使用者按下滑鼠左鍵時會觸發 click 事件，這個時候系統會呼叫指定的事件處理函數來回應此一事件。

```
win.mainloop()
```

前面已完整說明了建立一個空的視窗的指令流程，以下列出完整的建立視窗程式：

〈建立視窗的第一支程式〉

檔名 tk_main.py

```
01  # -*- coding: utf-8 -*-
02
03  import tkinter as tk
04  win = tk.Tk()
05  win.geometry("300x100")
06  win.title(" 視窗標題 ")
07  win.mainloop()
```

執行結果 »

各位可以執行 tk_main.py 程式，就能看到如下視窗，視窗右上角有標準視窗的縮小、放大以及關閉按鈕，還能夠拖曳邊框調整視窗大小。

12-1-3 佈局（Layout）方式

前面建立的視窗是空的視窗，還必須要放入與使用者互動的元件，這些元件並不能隨意亂放，必須依照 tkinter 的佈局方式，共有 3 種佈局方法：pack、grid 以及 place。

pack 方法

pack 預設以由上而下方式擺放元件，pack 方法常用的參數如下：

參數	說明
padx	設定水平間距
pady	設定垂直間距
side	設定位置，設定值有 left、right、top、bottom
expand	左右分散對齊，參數值 0 跟 1，0 表示不要分散；1 表示平均分配
fill	是否填滿寬度 (x) 或高度 (y)，參數值有 x、y、both、none

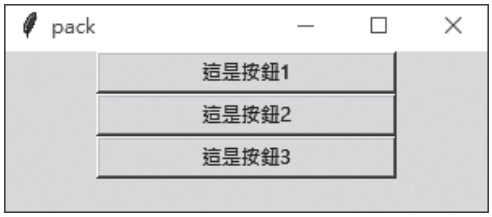
位置及長寬的單位都是像素（pixel），例如以下程式碼將 3 個按鈕利用 pack 方法加入視窗，其中的按鈕元件中的 width 屬性是指按鈕元件的寬度，而 text 屬性為按鈕上的文字。

〈利用 pack 方法加入視窗〉

檔名 pack.py

```
01  # -*- coding: utf-8 -*-
02
03  import tkinter as tk
04  win = tk.Tk()
05  win.geometry("300x100")
06  win.title("pack")
07
08  btn1=tk.Button(win, width=25, text=" 這是按鈕 1")
09  btn1.pack()
10  btn2=tk.Button(win, width=25, text=" 這是按鈕 2")
11  btn2.pack()
12  btn3=tk.Button(win, width=25, text=" 這是按鈕 3")
13  btn3.pack()
14
15  win.mainloop()
```

執行結果 »



grid 方法

grid 是以表格方式擺放元件，常用的參數如下：

參數	說明
column	設定放在哪一行
columnspan	左右欄合併的數量
row	設定放在哪一列
rowspan	上下欄合併的數量
padx	設定水平間距
pady	設定垂直間距
sticky	設定元件排列方式，參數值有 4 種：n、s、e、w：靠上、靠下、靠右、靠左

例如以下將 4 個按鈕利用 grid 方法加入視窗：

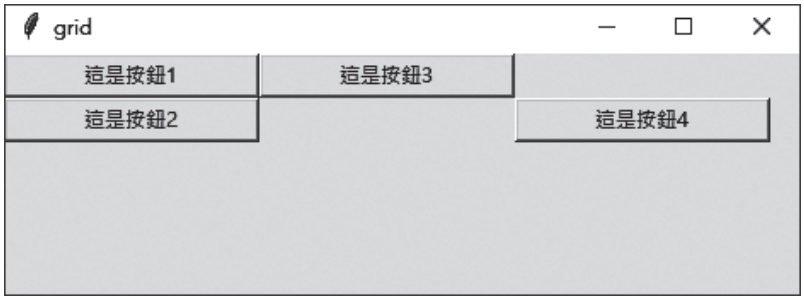
〈利用 grid 方法加入視窗〉

檔名 grid.py

```
01  # -*- coding: utf-8 -*-
02
03  import tkinter as tk
04  win = tk.Tk()
05  win.geometry("300x100")
06  win.title("grid")
07
```

```
08  btn1=tk.Button(win, width=20, text="這是按鈕 1")
09  btn1.grid(column=0,row=0)
10  btn2=tk.Button(win, width=20, text="這是按鈕 2")
11  btn2.grid(column=0,row=1)
12  btn3=tk.Button(win, width=20, text="這是按鈕 3")
13  btn3.grid(column=1,row=0)
14  btn4=tk.Button(win, width=20, text="這是按鈕 4")
15  btn4.grid(column=2,row=1)
16  win.mainloop()
```

執行結果 »



使用 grid 配置時很容易被行與列混淆，想像一下版面被切成如下的行列表格，儲存格內的數字分別代表（column, row），就像以下表格，就能輕易填入 column 與 row 的數值。

	column：0	column：1	column：2
row：0	0,0	1,0	2,0
row：1	0,1	1,1	2,1
row：2	0,2	1,2	2,2

place 方法

place 方法是透過元件在視窗中的絕對位置與相對位置來指定位置。相對位置的方法是將整個視窗寬度視為「1」，視窗中間位置 relx 就是 0.5，高度也是一樣，以此類推，常用參數如下表。

參數	說明
x	水平絕對位置
y	垂直絕對位置
relx	相對水平位置，值為 0~1
rely	相對垂直位置，值為 0~1
anchor	定位基準點，參數值有下列 9 種： center: 正中心 n、s、e、w：上方中間、下方中間、右方中間、左方中間 ne、nw、se、sw：右上角、左上角、右下角、左下角

例如以下將 3 個按鈕利用 place 方法加入視窗：

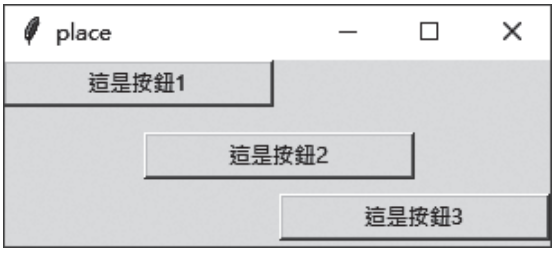
〈利用 place 方法加入視窗〉

檔名 place.py

```
01  # -*- coding: utf-8 -*-
02
03  import tkinter as tk
04  win = tk.Tk()
05  win.geometry("300x100")
06  win.title("place")
07
08  btn1=tk.Button(win, width=20, text=" 這是按鈕 1")
09  btn1.place(x=0, y=0)
10  btn2=tk.Button(win, width=20, text=" 這是按鈕 2")
11  btn2.place(relx=0.5, rely=0.5, anchor="center")
```

```
12  btn3=tk.Button(win, width=20, text=" 這是按鈕 3")
13  btn3.place(relx=0.5, rely=0.7)
14
15  win.mainloop()
```

執行結果 »



其中按鈕 2 與按鈕 3 利用相對位置定位，因此當視窗縮放時，元件位置仍會在相對比例的位置上。

12-2 GUI 元件

示範了如何建立一個視窗及 tkinter 的佈局方式之後，接下來就可以在視窗上面加入 GUI 元件，以下就從常用的元件（widget）開始談起。

12-2-1 標籤（Label）元件

要做好圖形化介面的視窗程式，有關文字的輸出入部分是不可或缺的一環。在 tkinter 常用的元件（widget）文字輸出的最基本元件就是「Label」，即所謂的「文字標籤」。文字標籤最主要的功能，是用來顯示使用者所要了解的文字敘述，例如文字輸入方塊（TextField）、按鈕（Button）說明，甚至可直接使用 Label 元件來顯示文字，它是一個非互動式的元件，只能顯示文字，點擊它並不會觸發任何事件，建立 Label 元件語法如下：

```
元件名稱 = tk.Label( 容器名稱 , 參數 )
```

容器名稱是指父類別的容器，也就是上一層容器名稱。當建立了一個元件，就可以指定前景顏色、字型以及寬跟高等參數，參數之間用逗號 (,) 分隔，常用的參數如下表：

參數	說明
height	設定高度
width	設定寬度
text	設定 Label 文字
font	設定字型及字大小
fg	設定文字顏色
bg	設定背景顏色
padx	與容器（Frame）的水平間距
pady	與容器（Frame）的垂直間距

參數	說明
anchor	文字位置，設定值有下列 9 種： center：正中心 n、s、e、w：上方中間、下方中間、右方中間、左方中間 ne、nw、se、sw：右上角、左上角、右下角、左下角
borderwidth	設框線寬度，可以「bd」取代
bitmap	標籤指定的點陣圖片
image	標籤指定的圖片
justify	標籤若有多行文字的對齊方式

一般來說要設定字型，它會以 tuple（序對）來表示 font 元素，以簡述說明。

```
font =('Verdana', 14, 'bold', 'italic')
```

tuple 包括字型名稱、字的大小以數值表示、字體中是否要加入粗體（bold）或斜體（italic）。除了字的大小之外，皆要以字串形式做設定。

而指定顏色可以使用顏色名稱（例如 red、yellow、green、blue、white、black）或使用十六進位值顏色代碼，例如紅色 #ff0000、黃色 #ffff00。

建立元件的同時必須要指定佈局方式，例如要將 Label 元件指定以 pack 方法排列，就可以如以下範例表示：

〈將 Label 元件指定以 pack 方法排列〉

檔名 label.py

```
01  # -*- coding: utf-8 -*-
02
03  import tkinter as tk
04  win = tk.Tk()
05  win.geometry("200x100")
06  win.title(" 常用 widget")
07
08  label = tk.Label(win, bg="#ffff00", fg="#ff0000", font =
    "Helvetica 15 bold", padx=20, pady=5, text = " 這是 Label")
09  label.pack()
10
11  win.mainloop()
```

執行結果 »

12-2-2 按鈕（Button）元件

按鈕元件是與使用者互動不可或缺的元件，按鈕元件主要應用於指令的下達或是功能的區分，由於 Button 按鈕元件可以帶給使用者在操作上直覺式的命令傳達，因此常在視窗應用程式中廣泛地被使用，當使用者按下按鈕時會觸發 click 事件，並接著呼叫對應的事件處理函數。建立按鈕元件的語法如下：

```
元件名稱 = tk.Button( 容器名稱 , 參數 )
```

常用的參數與 Label 元件一樣都具有顏色與字型等外觀設定，Button 元件多了一個 command 參數，這個參數是用來設定按下按鈕時要呼叫的事件處理函數。常用的參數如下表：

參數	說明
height	設定高度
width	設定寬度
text	設定按鈕文字
font	設定字型及字大小
textvariable	設定文字變數
fg	設定文字顏色
bg	設定背景顏色
padx	與容器（Frame）的水平間距
pady	與容器（Frame）的垂直間距
command	事件處理函數

每一個 Button 元件同樣要指定它的版面配置方式。以下範例放置了兩個按鈕，當按下按鈕 1 時會更換按鈕上的文字；點擊按鈕 2 時會變更按鈕上文字的顏色。

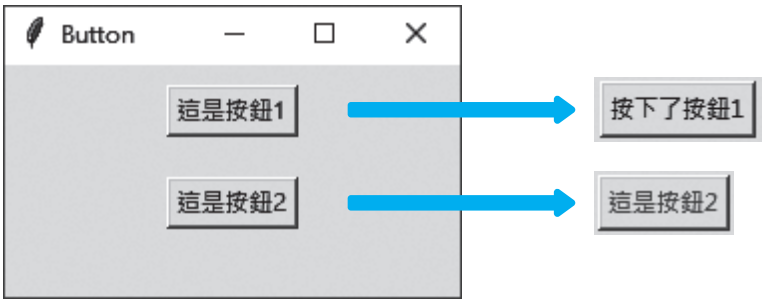
〈GUI 介面—按鈕〉

檔名 button.py

```
01  # -*- coding: utf-8 -*-
02
03  def btn_click():
04      btnvar.set(" 按下了按鈕 1")
05
06  def btn1_click():
07      btn1.config(fg = "red")
08
09  import tkinter as tk
10  win = tk.Tk()
```

```
11  win.title("Button")
12
13  btnvar = tk.StringVar()
14  btn = tk.Button(win, textvariable=btnvar, command=btn_click)
15  btnvar.set(" 這是按鈕 1")
16  btn.pack(padx=20, pady=10)
17
18  btn1 = tk.Button(win, text=" 這是按鈕 2", command=btn1_click)
19  btn1.pack(padx=20, pady=10)
20
21  win.mainloop()
```

執行結果 »



按鈕 1 的 command 參數指定的函數是「btn_click」；按鈕 2 的 command 參數指定的函數是「btn1_click」，當按下按鈕時就會去呼叫指定的函數。

當需要改變元件上的文字內容或變更屬性 (文字顏色、底色、寬、高、字型等等)，有兩個做法，以上面範例的按鈕為例，為各位說明如下：

透過 textvariable 參數指定文字變數

如果想變更元件的文字，各位可以指定文字變數給 textvariable 屬性，textvariable 屬性就會將文字變數與 text 屬性連結，當文字變數變更時元件上的文字也就跟著變更了。不過，文字變數值必須是 tkinter 的 IntVar（整數）、DoubleVar（浮點數）或 StringVar（字串）物件，再透過物件的 get() 與 set() 方法存取內容。

以下程式會產生 StringVar 物件，並用 set 方法設定按鈕上的文字，如下所示：

```
# 建立 StringVar 字串物件 btnvar
btnvar = tk.StringVar()
# 將 btnvar 指定給 textvariable
btn = tk.Button(win, textvariable=btnvar, command=btn1_click)
# 設定 btnvar 文字
btnvar.set(" 這是按鈕 1")
```

透過 config 方法更改文字內容或屬性值

config 方法可以修改元件的屬性值，格式如下：

```
元件名稱 .config( 屬性名稱 = 屬性值 )
```

例如：

```
btn1.config(fg = "red")
```

表示將 btn1 元件的文字顏色改為紅色。

想要改變按鈕文字也可以使用 config 方法改變 text 屬性，語法如下：

```
btn1.config(text = " 這是按鈕 1")
```

如果想要設定某些屬性可是又不知道該使用什麼參數，可以透過 config() 將元件目前的屬性值 print 出來查看，如下所示：

```
print(btn1.config())
```

Tips

當元件透過 textvariable 屬性設置了文字變數，text 屬性就與文字變數連結，這種情況下，就無法透過 config() 方法修改 text 屬性值。

12-2-3 文字編輯（Entry）元件

Entry 元件可以讓使用者輸入資料，它是單行模式，想要輸入多行就要使用 Text 元件，建立 Entry 元件的語法如下：

```
元件名稱 = tk.Entry( 容器名稱 , 參數 )
```

常用參數如下：

參數	說明
height	設定高度
width	設定寬度
font	設定字型及字大小
fg	設定文字顏色
bg	設定背景顏色
padx	與容器（Frame）的水平間距
pady	與容器（Frame）的垂直間距
borderwidth	設定邊框寬度
relief	設定邊框的浮雕效果，設定值有：flat、groove、raised、ridge、sunken、solid
justify	文字對齊方式，設定值：left、right、center，預設為 left
state	Entry 元件的狀態，normal（一般）、readonly（唯讀）、disabled（不可用）

如果要輸入 Entry 元件的預設值，可以使用 insert 方法：

```
entry.insert( 索引值 , 預設文字 )
```

索引值是指字串的索引位置，可以是數字或是字串「end」，索引從 0 開始，例如 Entry 元件裡面有文字「beauty」，字母 b 的索引值就是 0，字母 a 索引值為 2。當索引值小於或等於零，則插入點會在開始處；如果索引值大於或等於當前的字數，則插入點在字串末端。如果要取得字串最末端的位置，可以使用值「end」。透過以下的範例，就能更清楚 insert 方法索引值的妙用。

〈GUI 介面—entry〉

檔名 entry.py

```
01  # -*- coding: utf-8 -*-
02
03  import tkinter as tk
04  win = tk.Tk()
05  win.title("Entry")
06
07  entry = tk.Entry(win, bg="#ffccff", font = "Helvetica 15 bold",
    borderwidth = 3)
08  entry.insert(0," 這是 Entry")
09  entry.insert("2"," 實用的 ")
10  entry.insert("end",", 真好玩 ")
11  entry.pack(padx=20, pady=10)
12
13  win.mainloop()
```

執行結果 »

</> 程式解說 »

- 第 08 行：先放入字串「這是 Entry」。
- 第 09 行：將字串「實用的」放在索引 2 的位置，所以 Entry 元件裡的文字變成「這是實用的 Entry」。
- 第 10 行：程式將「, 真好玩」字串位置指定在「end」，表示放在字串最末端，所以 Entry 元件的文字變成「這是實用的 Entry, 真好玩」。

如果要刪除 Entry 元件裡的文字，可以使用 delete 方法，格式如下：

```
entry.delete( 起始索引值 , 結束索引值 )
```

例如：

```
entry.delete(0, 2)       # 刪除前面兩個字元
entry.delete(3, "end")   # 刪除第 3 個字元之後的字元
entry.delete(0, "end")   # 刪除全部
```

12-2-4 多行文字（Text）元件

Text 元件用來儲存或顯示多行文字，屬性和 Entry 元件大多雷同。包括純文字或格式文件，Text 元件也可以被用作文字編輯器。建立 Text 元件的語法如下：

```
元件名稱 =tk.Text( 容器名稱 , 參數 1, 參數 2,….)
```

Text 元件的屬性和 Entry 元件大多雷同，常用參數如下：

參數	說明
height	設定高度
width	設定寬度
font	設定字型及字大小
fg 或 foreground	設定文字顏色
bg 或 background	設定背景顏色
padx	與容器（Frame）的水平間距
pady	與容器（Frame）的垂直間距
borderwidth	設定邊框寬度
state	設定元件內容是否允許編輯，預設值為「tk.NORMAL」表示文字元件內容可以編輯；如果參數值為「tk.DISABLED」，表示文字元件內容不可以修改。

如果想在已建立的文字方塊設定文字內容，必須呼叫 insert() 方法，語法如下：

```
insert( 索引值 , 預設文字 )
```

- index：依索引值插入字串。有三個常數值：INSERT、CURRENT(目前位置) 和 END(將字串加入文字方塊，並結束文字方塊內容)。
- text：欲插入的字串。

當建立 Text 元件後，如果要變更元件的參數設定，可以使用 config 方法，語法如下：

```
元件名稱 .config( 參數 1, 參數 2, …)
```

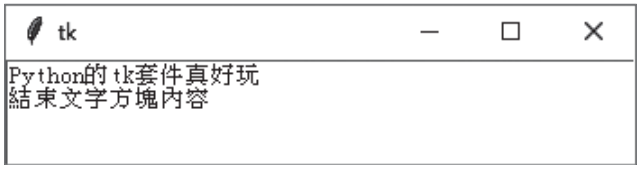
預設情況下，各位可以編輯文字元件的內容。如果要禁止使用者編輯，請將 state 參數值設為「tk.DISABLED」，一旦設定禁止編輯後，文字元件的內容就無法被修改或加入文字。

〈GUI 介面—text〉

檔名 tktext.py

```
01 import tkinter as tk
02 win = tk.Tk()
03 text=tk.Text(win)
04 text.insert(tk.INSERT, "Python 的 tk 套件真好玩 \n")
05 text.insert(tk.END, " 結束文字方塊內容 ")
06 text.pack()
07 text.config(state=tk.DISABLED)
08 win.mainloop()
```

執行結果 »



</> 程式解說 >>

- 第 03 行：建立 Text 元件。
- 第 04 行：在已建立的文字方塊設定「Python 的 tk 套件真好玩」文字。
- 第 05 行：插入「結束文字方塊內容」文字內容到文字方塊中，並結束文字方塊內容。
- 第 07 行：將 state 參數值設為「tk.DISABLED」，一旦設定禁止編輯後，文字元件的內容就無法被修改或加入文字。

12-2-5 單選按鈕（Radiobutton）

選項元件有兩種：Checkbutton（核取方塊）和 Radiobutton（單選按鈕）。Checkbutton 提供多選的功能，但是 Radiobutton 只能從多個項目中擇一，無法多選。建立 Radiobutton 的語法如下：

```
元件名稱 =tk.Radiobutton( 容器名稱 , 參數 1, 參數 2,….)
```

Radiobutton（單選按鈕）常用的參數如下：

參數	說明
background	設背景色，可以「bg」取代
foreground	設前景色，可以「fg」取代
borderwidth	定框線粗細，可以「bd」取代
font	設定字型
height	元件高度
width	元件寬度
justify	有多行文字的對齊方式
text	元件中的文字
variable	元件所連結的變數

參數	說明
padx	元件與容器間的水平距離
pady	元件與容器間的垂直距離
value	設定使用者點選後的元件值
command	設定點選後要執行的函數

〈GUI 介面—Radiobutton〉

檔名 tkRadiobutton.py

```
01  from tkinter import *
02  wnd = Tk()
03  wnd.title('Radiobutton')
04  def myOptions():
05      print(' 你的選項是 :', var.get())
06  ft = (' 新細明體 ', 14)
07  Label(wnd,
08        text = " 選擇喜愛的顏色 ", font = ft,
09        justify = LEFT, padx = 20).pack()
10  color = [(' 紅色 ', 1), (' 綠色 ', 2),
11            (' 藍色 ', 3)]
12  var = IntVar()
13  var.set(1)
14  for item, val in color:
15      Radiobutton(wnd, text = item, value = val,
16          font = ft, variable = var, padx = 15,
17          command = myOptions).pack(anchor = W)
```

執行結果 »

</> 程式解說 >>

- 第 04~05 行：定義方法 myOptions() 方法，用來回應單鈕按鈕的 command 屬性，呼叫 get() 方法來顯示哪一個按鈕被選取。
- 第 12~13 行：將單選按鈕被選的元件以 Intvar() 方法來轉為數值，再以 set() 方法將單選按鈕以第三個元件為預設值。
- 第 14~17 行：以 for 迴圈來產生單選按鈕並讀取 color 的元素，將屬性 variable 來取得變數值後，再透過屬性 commnad 來呼叫 myOptions() 函數來顯示哪一個單選按鈕被選取。

12-2-6 核取按鈕（Checkbutton）元件

Checkbutton 是從列出的項目當中做不同的選擇，通常應用在程式中提供使用者條件勾選時。Checkbutton 使用上非常簡單，只要點選 Checkbutton 元件，就會出現打勾的符號，再點選一次，打勾的符號就會消失。可以通通不選，也能同時選取，或者只挑選幾個。建立核取按鈕的語法如下：

```
元件名稱 =tk. Checkbutton ( 容器名稱 , 參數 1, 參數 2,….)
```

Checkbutton 常用的參數如下：

參數	說明
anchor	文字對齊方式
background	設背景色，可以「bg」取代
foreground	設前景色，可以「fg」取代
borderwidth	定框線粗細，可以「bd」取代
relief	配合 borderwidth，設定框線樣式
bitmap	按鈕上顯示的圖片
font	設定字型
height	元件高度

參數	說明
width	元件寬度
justify	有多行文字的對齊方式
text	元件中的文字
onvalue/offvalue	元件選取 / 未選後所連結的變數值
variable	元件所連結的變數

一般來說，核取方塊有核取（勾選）和不核取（不勾選）兩種狀態。

- **核取（勾選）**：以預設值「1」表示；使用屬性 onvalue 來改變其值。
- **不核取（未勾選）**：設定值「0」表示；使用屬性 offvalue 變更設定值。

核取方塊的變數，可呼叫 Intvar() 和 Stringvar() 方法來處理數值和字串的問題。

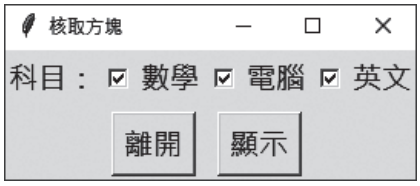
〈GUI 介面—Checkbutton〉

檔名 tkCheckbutton.py

```
01  from tkinter import *
02  wnd = Tk()
03  wnd.title('核取方塊')
04
05  def varStates():     #回應核取方塊變數狀態
06      print('選取的科目有:', var1.get(), var2.get()
07            ,var3.get())
08
09  ft1 = ('微軟正黑體', 14)
10  ft2 = ('微軟正黑體', 14)
11  lb1=Label(wnd, text = '科目:', font = ft1)
12  lb1.grid(row = 0, column = 0)
13  item1 = '數學'
14  var1 = StringVar()
15  chk = Checkbutton(wnd, text = item1, font = ft1,
16      variable = var1, onvalue = item1, offvalue = '')
```

```
17  chk.grid(row = 0, column = 1)
18  item2 = '電腦'
19  var2 = StringVar()
20  chk2 = Checkbutton(wnd, text = item2, font = ft1,
21      variable = var2, onvalue = item2, offvalue = '')
22  chk2.grid(row = 0, column = 2)
23  item3 = '英文'
24  var3 = StringVar()
25  chk3 = Checkbutton(wnd, text = item3, font = ft1,
26      variable = var3, onvalue = item3, offvalue = '')
27  chk3.grid(row = 0, column = 3)
28
29  btnQuit = Button(wnd, text = '離開', font = ft2,
30      command = wnd.destroy)
31  btnQuit.grid(row = 2, column = 1, pady = 4)
32  btnShow = Button(wnd, text = '顯示', font = ft2,
33      command = varStates)
34  btnShow.grid(row = 2, column = 2, pady = 4)
35  mainloop()
```

執行結果 »



</> 程式解說 »

- 第 05~07 行：定義方法 varStates() 回應核取方塊變數狀態，當核取方塊被核取時，透過變數呼叫 get() 方法。
- 第 13 行：設定變數 item1 來作為核取方塊的屬性 text、onvalue 的屬性值。
- 第 14 行：將變數 var1 以 Stringvar() 轉為字串，並指定給核取方塊的屬性 variable 使用，回傳核取方塊「已核取」或「未核取」的回傳值。
- 第 15~16 行：產生核取方塊並設屬性 onvalue、offvalue。
- 第 32~33 行：按鈕的屬性 command 會去呼叫 varStates() 方法做回應。

12-2-7 捲軸（Scrollbar）元件

捲軸常被使用在文字區域（Text）、清單方塊（Listbox）或是畫布（Canvas）等元件，要在這些元件中建立捲軸，其語法如下：

```
Scrollbar( 父物件 , 參數 1= 設定值 1, 參數 2= 設定值 2,…)
```

這些參數屬於選擇性參數，以下列出較常用的參數名稱：

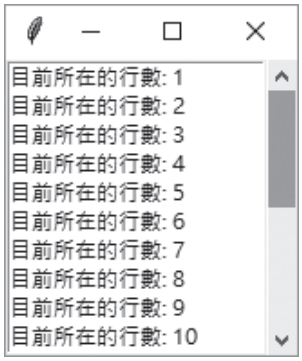
參數	說明
activebackground	移動捲軸時，捲軸與箭頭的色彩
background	設背景色，可以「bg」取代
borderwidth	定框線粗細，可以「bd」取代
command	移動捲軸時，會呼叫此參數所指定的函數
highlightbackground	反白背景色彩
highlightcolor	反白色彩
orient	預設值 =VERTICAL，代表垂直捲軸，如果值為 HORIZONTAL 則代表水平捲軸
width	元件寬度

〈GUI 介面—Scrollbar〉

檔名 tkscrollbar.py

```
01  from tkinter import *
02
03  Window = Tk()
04  scrollbar = Scrollbar(Window)
05  scrollbar.pack( side = RIGHT, fill = Y )
06
07  test_List = Listbox(Window, yscrollcommand = scrollbar.set )
08  for line in range(20):
09     test_List.insert(END, " 目前所在的行數 : " + str(line+1))
10
11  test_List.pack( side = LEFT, fill = BOTH )
12  scrollbar.config( command = test_List.yview )
13
14  mainloop()
```

執行結果 >>



</> 程式解說 >>

- 第 04 行：在 Window 父物件下建立捲軸。
- 第 05 行：呼叫 pack() 方法將捲軸由右向左排列，fill=Y 表示元件高度和父物件相同。但如果各位的元件寬度和父物件相同，則必須設定 fill=X。
- 第 07 行：將文字方塊元件的 yscrollcommand 選擇性參數設定為 scrollbar.set，表示將捲軸連結到文字方塊元件。
- 第 08~09 行：利用迴圈將目前所在的行數顯示在文字方塊中。

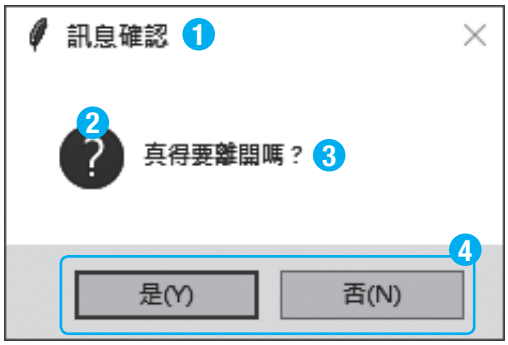
12-3 對話方塊

對話方塊（Dialog）是一種使用者與程式間互動的介面，tkinter 套件中的對話方塊包括：

- **messagebox 訊息方塊**：用來提供不同種類的訊息。
- **simpledialog 簡單型對話方塊**：用來輸入字串或整數、浮點數。
- **filedialog 檔案對話方塊**：利用它可開啟檔案。
- **colorchooser 調色盤方塊**：它會回傳所選取顏色的色（RGB）值。

12-3-1 訊息方塊

訊息方塊主要目的就是以簡便的訊息與使用者互動。基本結構如下：



1. messagebox 的標題列，呼叫相關方法時會以參數「title」表示。
2. 代表 messagebox 的小圖示，呼叫相關方法時會以參數「icon」表示。
3. 顯示 messagebox 的相關訊息，呼叫相關方法時會以參數「message」代表。
4. 顯示 messagebox 的對應按鈕，每個按鈕皆有回應的訊息，呼叫相關方法時會以參數「type」表示。

訊息方塊概分兩大類，詢問和顯示。詢問「ask」為開頭，伴隨 2~3 個按鈕來產生互動行為。顯示以「show」開頭，只會顯示一個「確定」鈕，如下表所示。

種類	messagebox 方法
詢問	askokcancel(title = None, message = None, **options)
	askquestion(title = None, message = None, **options)
	askretrycancel(title = None, message = None, **options)
	askyesno(title = None, message = None, **options)
	askyesnocancel(title = None, message = None, **options)
顯示	showerror(title = None, message = None, **options)
	showinfo(title = None, message = None, **options)
	showwarning(title=None, message=None, **options)

其中參數 title 是作為標題列欲顯示的文字。而參數 message 則是作為顯示訊息的文字內容。以下例子將示範如何產生訊息方塊：

〈GUI 介面—訊息方塊〉

檔名 tkmessagebox.py

```
01  from tkinter import *
02  from tkinter import messagebox
03  wnd = Tk()
04  wnd.title('Messagebox 訊息方塊 ')
05  wnd.geometry('180x120+20+50')
06  def answer():
07      messagebox.showerror(' 回答 ',
08              ' 抱歉！ , 你的問題無法回答 ')
09  def callback():
10      if messagebox.askyesno(' 訊息確認 ',
11              ' 真得要離開嗎？ '):
12          messagebox.showwarning(' 訊息 - Yes',
13              ' 抱歉！無法離開 ')
```

```
14      else:
15          messagebox.showinfo(
16              '訊息 - No', '取消「離開」指令')
17  Button(wnd, text='離開', command =
18         callback).pack(side = 'left', padx = 10)
19  Button(wnd, text='回答', command =
20         answer).pack(side = 'left')
21  mainloop()
```

執行結果 »

12-3-2 簡單型對話方塊

simpledialog 有三個方法：處理字串的 askstring() 方法；用於整數的 askinteger() 方法；用於浮點數的 askfloat() 方法。其語法格式如下：

```
simpledialog.askinteger(title, prompt, **kw)
simpledialog.askfloat(title, prompt, **kw)
simpledialog.askstring(title, prompt, **kw)
```

- title：用於對話方塊的標題，必要參數。
- prompt：提示字串，必要參數。
- kw：選項參數，請參考下表。

選項	型別	說明
initialvalue	integer、float、string	輸入時的初值
minvalue	integer / float	最小值
maxvalue	integer / float	最大值

下述範例的兩個按鈕，當被使用者按下時透過 command 屬性去呼叫所對應的方法，分別來輸入字串和整數。其中 processWord() 方法是用來處理輸入字串時該進行的工作。另一個 processInt() 方法是用來處理輸入整數值時該進行的工作。

〈GUI 介面—簡單型對話方塊〉

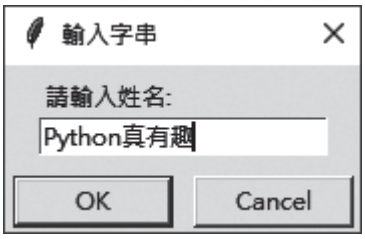
檔名 tksimpledialog.py

```
01 # simpledialog - askinteger() 方法
02 from tkinter import *
03 from tkinter import simpledialog
04
05 def processWord(): # 處理輸入字串
06     name = simpledialog.askstring(title = '輸入字串',
07             prompt = '請輸入姓名：')
08     print('名稱：', name)
09
10 def processInt(): # 處理輸入整數值
11     score = []
12     count = 0
13     while True:
14         number = simpledialog.askinteger(
15             title = '輸入整數值', prompt = '分數：',
16             maxvalue = 100, minvalue = 60)
17         score.append(number)
18         count += 1
19         if count == 5:
20             break
21     total = sum(score)
22     print('分數：', score)
23     print('合計：', sum(score))
24
25
26 # 建立視窗物件
27 wnd = Tk()
28 wnd.title('簡單型對話方塊')
```

```
wnd.geometry('140x60+10+10')
# 視窗物件加入按鈕
Button(text = '輸入字串', command = processWord).pack(
    side = 'left')
Button(text = '輸入整數', command = processInt).pack(
    side = 'right', padx = 5)

mainloop()
```

執行結果 »

❶ 當按下「輸入字串」鈕的輸出結果：



❷ 按下「OK」鈕後，就會產生如下的輸出結果：

```
名稱： Python真有趣
```

❸ 當按下「輸入整數」鈕的輸出結果：

❹ 當各位依上述輸入整數值的方式，連續輸入 5 筆分數，按下「OK」鈕後，就會產生如下的輸出結果：

```
分數： [100, 98, 87, 96, 95]
```

12-3-3 檔案對話方塊

filedialog（檔案對話方塊）有兩個方法：其中 askopenfile() 方法可以開啟檔案。asksaveasfile() 方法則是儲存檔案的作用。接著就以一個範例來說明 askopenfile() 方法和 asksavefile() 方法的使用。

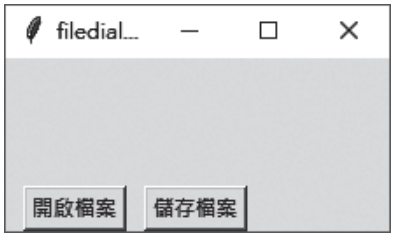
〈GUI 介面—檔案對話方塊〉

檔名 tkfiledialog.py

```
''' filedialog.askopenfile() 方法開啟檔案
    filedialog.asksaveasfile() 方法儲存檔案 '''

from tkinter import *
from tkinter import filedialog

# 開啟檔案對話方塊
def OpenFile():
    name = filedialog.askopenfilename(title = ' 開啟檔案 ',
            filetypes = [('Text File', '*.txt'),
            ('Python Files', '*.py *.pyw'),
            ('All Files', '*')])
    print(' 開啟的檔案 ', name)
    # 建立檔案 -- 使用 open() 函數
    with open('myfile.txt', 'rt') as foin:
        total = foin.read()
        print(' 字元數：', len(total))
        for line in total:
            print(line, end = '')

# 儲存檔案
def SaveFile():
    save = filedialog.asksaveasfilename(title = ' 儲存檔案 ',
            filetypes = [('Text Files', '*.txt *.csv')],
            initialfile = 'myfile.txt')
    # 以附加寫入檔案
```

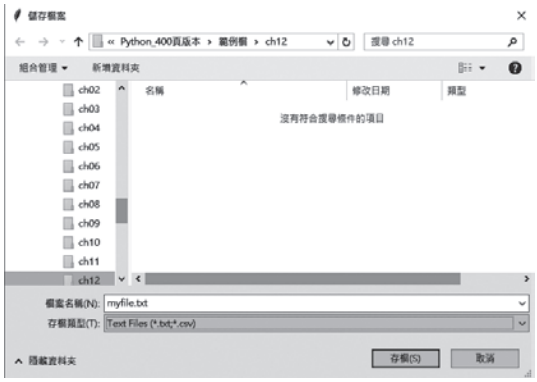
```
27      with open('myfile.txt', 'a+') as fout:
28          show = '天道酬勤 地道酬善\n'
29          print('字串長度：', len(show))
30          fout.write(show)
31      print('儲存檔案', save)
32
33  #建立視窗物件
34  wnd = Tk()
35  wnd.title('filedialog')
36  wnd.geometry('100x50+10+10')
37
38  #視窗物件加入兩個按鈕
39  Button(text='開啟檔案', command = OpenFile).pack(
40      anchor = 's', side = 'left', padx = 10)
41  Button(text='儲存檔案', command = SaveFile).pack(
42      anchor = 's', side = 'left')
43  mainloop()
```

執行結果 »



- 開啟檔案對話方塊外觀

- 儲存檔案對話方塊外觀



</> 程式解說 >>

- 第 08~19 行：定義 Open File() 方法，當「開啟檔案」鈕被按下時，commnad 屬性所要呼叫的方法。它會去呼叫 askopenfilename() 方法，而參數 filetypes 設其參數值文字檔案、Python 檔案和所有檔案為檔案類型。
- 第 22~31 行：定義 Save File() 方法，當「儲存檔案」鈕被按下時，commnad 屬性所要呼叫的方法。它會去呼叫 asksaveasfilename() 方法。

12-3-4 調色盤方塊

colorchooser 元件提供色彩的選擇，askcolor() 方法可產生標準對話方塊，以調色盤提供顏色的選擇，語法如下：

```
colorchooser.askcolor([color [,options]])
```

- color：設定色彩。
- options：選項參數。

選項	型別	說明
initialcolor	color	以 RGB 為主的顏色
title	string	檔案對話方塊標題

askcolor() 方法會以 tuple 物件回傳 RGB 的值，回傳值如下：

```
((0.0, 128.5, 255.99609375), '#0080ff ')
```

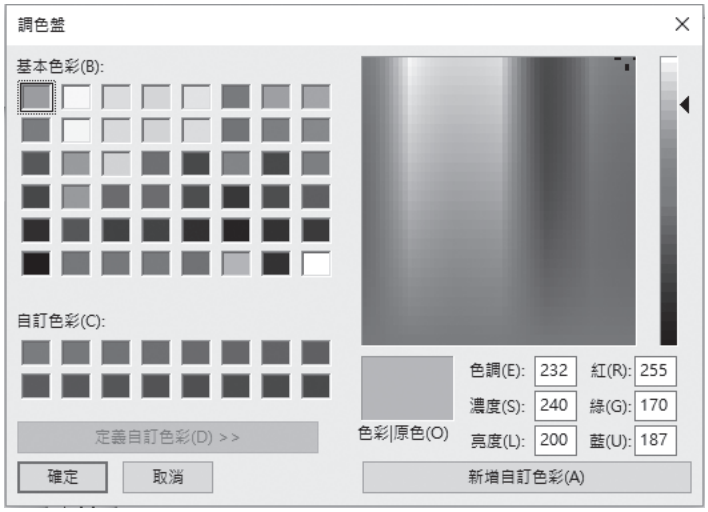
〈GUI 介面—調色盤方塊〉

檔名 colorchooser.py

```
01  from tkinter import *
02  from tkinter import colorchooser
03
04  # 調色盤方塊呼叫 askcolor() 方法讓使用者作顏色選取
05  def SelectColor():
06      tint = colorchooser.askcolor(title = ' 調色盤 ',
07              initialcolor = '#FF88CC')
08      rgbs = tint[0]
09      print('R: {:.3f}'.format(rgbs[0]))
10      print('G: {:.3f}'.format(rgbs[1]))
11      print('B: {:.3f}'.format(rgbs[2]))
12      print(' 色彩 16 進位值 :', tint[1])
13  # 建立視窗物件
14  wnd = Tk()
15  wnd.title(' 提供色彩的 colorchooser')
16  wnd.geometry('90x50+10+10')
17  # 視窗物件加入按鈕
18  Button(text=' 調色盤 ', command = SelectColor).pack(
19      side = 'bottom')
20  mainloop()
```

執行結果 »

調色盤視窗外觀如下：



選定好色彩後，按下「確定」鈕就會輸出如下的結果：

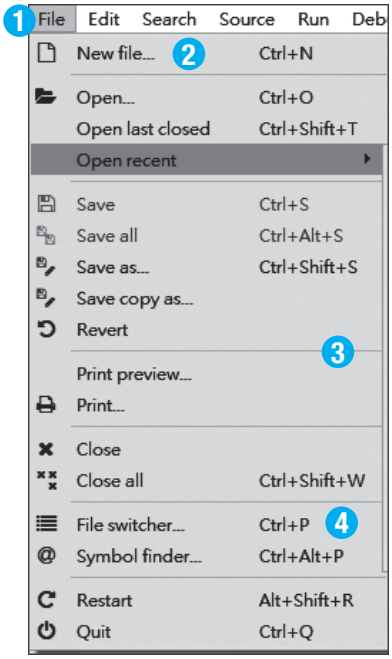
```
R: 255.996
G: 0.000
B: 0.000
色彩16進位值: #ff0000
```

</> 程式解說 >>

- 第 05~12 行：定義方法 SelectColor()，用來回應「調色盤」鈕被按時，屬性 command 所呼叫的方法。
- 第 08~12 行：取得的顏色值會以 tuple 物件儲存，分別讀取 tuple 元素，再做輸出。

12-4 功能表

功能表通常位於視窗標題列下方，將操作的相關指令集結，只要使用者去按某個指令，就能執行相關程序。例如「檔案」功能表提供跟檔案有關的「開啟檔案」或「儲存檔案」指令。按主功能表的「檔案」展開下拉選單，再去執行相關項目的指令。要產生功能表就得使用 Menu 元件。不過 Menu 元件只能產生功能表的骨架，還必須配合 Menu 元件的相關方法。未製作功能表之前，先認識功能表的組成。



1. 主功能表項目：上圖所看到的 File、Edit 皆是；Python 稱為「pulldown menu」。要產生主功能表項目，Menu 元件的 add_cascade() 方法能協助。

2. 下拉選單項目（或第二層功能表）：有了主功能表項目之後，才能進行下拉選單項目的設置。如位於 File 主功能表的下拉選單，有：New File、Open、Save 等。不過這些選單的項目有點像按鈕，呼叫 add_command() 方法來處理，有可能是一個群組，只能單選其中一個，add_Radiobutton() 方法能符合其需求。

3. 分隔線：下拉選單的項目之間若要做區隔，可呼叫 add_separator() 方法加入分隔線。

4. 快速鍵：設於下拉選單的項目，Python 以 Accelerator key 表示。依據其設定值，就能快速執行某個指令。

下表列出 Menu 元件有關的方法：

方法	說明
activate(index)	動態方法
add(type, **options)	增加功能表項目
add_cascade(**options)	新增主功能表項目
add_checkbutton(**options)	加入 checkbutton(核取方塊)
add_command(**options)	以按鈕形式新增子功能表項目
add_radiobutton(**options)	以單選按鈕形式新增子功能表項目
add_separator(**options)	加入分隔線，用於子功能表的項目之間

其中 add() 方法中的參數 type 可用來指定功能表的種類，它包括：command、cascade(submenu)、checkbutton、radiobutton 或 separator。

至於如何使用 menu 元件來產生功能表，請參考以下的步驟說明：

▶ **步驟 1**：先建立主視窗物件，再把 Menu 元件放入主視窗中，再以 Menu 元件的實體 menubar 來儲存。

```
root = Tk()
menubar = Menu(root)# 將 Menu 元件加入主視窗，產生功能表骨架
```

▶ **步驟 2**：將功能表物件 menubar 佈置到主視窗的頂部，顯示於畫面。

```
root.config(menu = menubar)# 顯示功能表
```

▶ **步驟 3：加入主功能表項目。**

```
menu_file = Menu(menuBar, tearoff = 0)
```

- 建立主功能表項目 menu_file 加到 menubar(功能表物件)，設 tearoff 之值為零，避免第一個子功能表項目的上方有虛線。

步驟 4：呼叫 add_cascade() 方法產生主功能項目實體。

```
menuBar.add_cascade(label = 'File', menu = menu_file)
```

- 通常要到此步驟才能看到功能表顯示於主視窗中。
- 以 add_cascade() 產生主功能表項目時，以 label 設定其名稱，將 menu_file 指派給 menu，如此新增的項目 File 才能真正加入功能表物件 menuBar。

步驟 5：加入下拉式選單的項目。

```
filemenu.add_command(label = 'Open', command = OpenFile)
```

- 有了主功能表項目 File 之後，呼叫 add_command() 方法以按鈕型式產生下拉選單的項目，因為是按鈕所以參數 command 要有回應方法做處理。

步驟 3 中使用 Menu 元件的建構函數時，將其中參數 tearoff 變更為「1」時，會在下拉選單的第一個項目上方加一條橫虛線。當然也可以將參數 tearoff 的值變為「0」就不會有此橫虛線。

以下範例就來示範如何以 Menu 元件建置功能表。

〈GUI 介面 — Menu〉

檔名 tkMenu.py

```
01  # 以 Menu 元件建置功能表
02  from tkinter import *
03
04  # 定義回應函數
05  def NewFile():
06      print('New File!')
07
08  def OpenFile():
09      file = filedialog.askopenfilename()
10      print(file)
11
12  def SaveFile():
13      save = filedialog.asksaveasfilename()
14
15  def About():
16      print(' 這是一個很簡單的功能表 ')
17
18  wnd = Tk()     # 主視窗物件
19  wnd.title(' 功能表 ')
20
21  # Step1. 產生功能表物件 menuBar，Menu 加到主視窗物件
22  menuBar = Menu(wnd)
23
24  # Step2. 將功能表物件 menubar 佈置到主視窗的頂部，顯示於畫面
25  wnd.config(menu = menuBar)
26
27  # Step3. 加入主功能表項目
28  menu_file = Menu(menuBar, tearoff = 0)
29  menu_font = Menu(menuBar, tearoff = 0)
30  menu_help = Menu(menuBar, tearoff = 0)
31
32  # Step4. add_cascade() 方法產生主功能項目實體
```

```
33 menuBar.add_cascade(label = '檔案', menu = menu_file)
34 menuBar.add_cascade(label = '字型', menu = menu_font)
35 menuBar.add_cascade(label = '線上說明', menu = menu_help)
36
37 # Step5. 加入下拉選單的項目 Step5-1. File 主能表表
38 menu_file.add_command(label = '新檔案',
39         command = NewFile)
40 menu_file.add_command(label = '開啟',
41         underline = 1, accelerator = 'Ctrl+O',
42         command = OpenFile)
43 menu_file.add_separator()# 加入分隔線
44 menu_file.add_command(label = '儲存',
45         command = SaveFile)
46 menu_file.add_separator()# 加入分隔線
47 menu_file.add_command(label = '離開',
48         command = lambda : wnd.destroy())
49
50 # Step5-2. Font 主功能表
51 labels = (12, 14, 16, 18)
52 for item in labels:
53     menu_font.add_radiobutton(label = item)
54
55 # Step5-3. Help 主功能表
56 menu_help.add_command(label = '關於', command = About)
57
58 mainloop()
```

執行結果 »

程式解說

- 第 05~13 行：定義方法，回應功能表中下拉式項目為 command 時所呼叫的方法。
- 第 22 行：將 menu 元件加入視窗物件中，以功能表物件 menuBar 儲存。
- 第 25 行：由主視窗物件 wnd 呼叫 config() 將功能表物件指派給 menu 之後，才能顯示於螢幕上。
- 第 28~30 行：先產生主功能表項目 menu_file，再以 Menu 元件的建構函數將它加入功能表物件 menuBar，並設 tearoff 之值為零。其他兩個主功能表物件 menu_font 和 menu_help 亦同。
- 第 33~35 行：有了主功能表物件後，再以 add_cascade() 方法，以參數 lable 設定顯示名稱，參數 command 呼叫回應的方法。
- 第 38~42 行：為 File 主功能表加入下拉式選單的項目。由 menu_file 物件呼叫 add_command() 方法以按鈕型式加入。
- 第 43 行：menu_file 物件呼叫 add_separator()，將下拉式選單的項目做分隔。
- 第 51~53 行：加入下拉式選單的項目，但以 add_radiobutton() 方法產生，由於是單選鈕形式，項目之間會彼此互斥。

12-5 繪製圖形（Canvas）

Canvas 元件（畫布控制元件）可以用來繪圖，包括線條、幾何圖形或文字等。由於 Canvas 元件具有畫布功能，能藉由滑鼠的移動做基本繪製。

利用 Canvas 元件進行圖形繪製有兩種座標系統：

- Windows 座標系統，以螢幕的左上角為原點（x = 0, y = 0））。
- Canvas 元件的座標系統，依指定位置進行繪製。

除非有特別指定，否則繪製的物件會以 Canvas 元件的座標系統為主。所有元件都要加入主視窗物件，再呼叫 pack() 方法納入版面管理。

接著來認識 Canvas 元件的相關屬性：

- background（或 bg）：背景顏色。
- borderwidth（或 bd）：設框線粗細。
- foreground（或 fg）：前景顏色。

width/height：以 width、height 設定元件的大小。

12-5-1 加入點陣圖

要在 Canvas 元件中載入點陣圖可呼叫 create_image() 方法。語法如下：

```
create_image(position, **options)
```

- position：座標位置 x1、y1。

此處 create_image() 方法無法讀取一般的圖片而是經過處理的 image 物件。所以有兩個步驟：

- 以 PhotoImage() 建構函數去讀取圖片，以 image 物件儲存。
- 再以 create_image() 方法的參數 image 取得。

例如以 Canvas 元件來加入圖片，語法如下：

```
# Canvas 元件繪製圖形
from tkinter import *
wnd = Tk()
wnd.title('Canvas 繪圖 ')
photo = PhotoImage(file = '03.png')
gs = Canvas(wnd)
# 載入圖片
gs.create_image(80, 120, image = photo)
gs.pack()
```

上述的程式碼中，載入圖片以 PhotoImage 物件 photo 來儲存，我們可以載入的圖片格式有「*.png」或「*.gif」的圖片。若載入「*.jpg」格式有時會產生無法識別圖片格式的錯誤。接著建構一個 Canvas 元件名稱為 gs，再透過這個 Canvas 元件呼叫 create_image() 方法，並以參數 image 來取得圖片。請注意，這裡的座標值以 Canvas 元件所指定為主。

12-5-2 以 Canvas 元件繪製幾何圖形

Canvas 元件繪製幾何圖形包括：弧形、扇形、線、圓形或橢圓形、矩形、文字等，接著示範如何利用 Canvas 元件繪製幾何圖形。

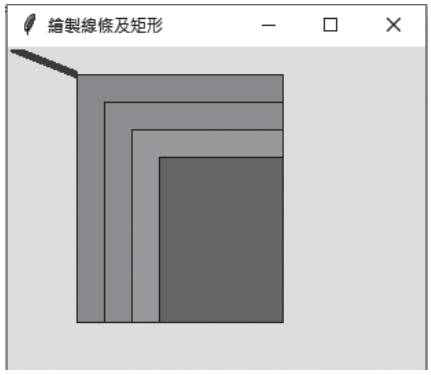
〈以 Canvas 元件繪製幾何圖形〉

檔名 tkCanvas.py

```
from tkinter import *

root = Tk()
root.title(' 繪製線條及矩形 ')
gs = Canvas(root, width = 300, height = 300)
gs.pack()
gs.create_rectangle(
    50, 20, 200, 200, fill = '#AABBFF')
```

```
gs.create_rectangle(
    70, 40, 200, 200, fill = '#AACC69')
gs.create_rectangle(
    90, 60, 200, 200, fill = '#B9C8FF')
gs.create_rectangle(
    110, 80, 200, 200, fill = '#B886D0')

# 左上角
gs.create_line(0, 0, 50, 20,
    fill = '#0E6042', width = 5)

mainloop()
```

執行結果 »



12-6 上機實作演練—以 GUI 實作簡易計算機

本次實做的題目是簡易計算機，提供加減乘除功能，特別的是這個範例將使用 GUI 來呈現計算機介面。

程式說明與輸入步驟

1. 使用 GUI 介面，需有 0~9 及 .（點）鍵、加減乘除 (+-*/) 鍵、Cls 鍵（清除）以及「=」等於鍵。
2. 必須有輸入區及輸出區，輸入區可用按鍵輸入也可用鍵盤輸入。
3. 容許錯誤輸入，輸出錯誤訊息「Infinity」。（例如輸入 : 10/0，輸出 :Infinity.）

輸出計算結果畫面

錯誤輸入時顯示 Infinity 畫面

此範例大致分為下列四個步驟：

1. 建立主視窗
2. 版面配置（建立 Frame）
3. 建置 Label、Entry 與 Button 元件
4. 加入事件處理函數

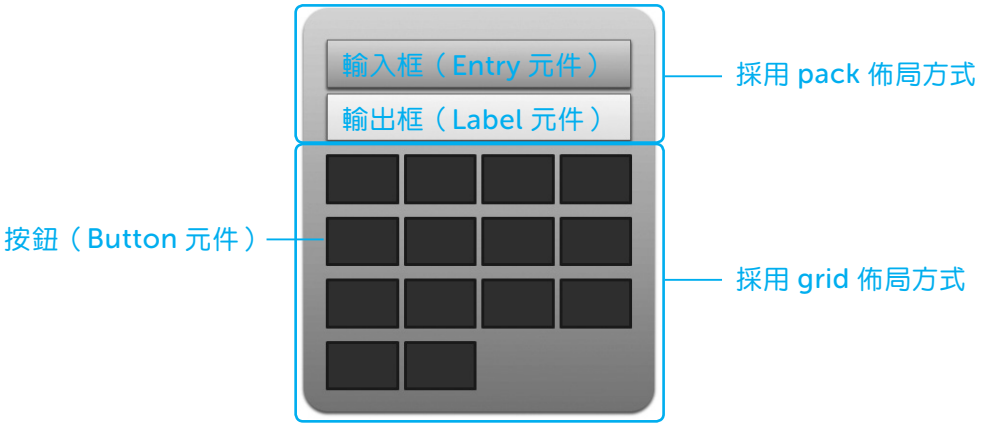
12-6-1 建立主視窗

首先載入 tkinter 模組，並將主視窗名稱命名為「win」，視窗標題設定為「簡易計算機」，程式碼如下：

```
import tkinter as tk
win = tk.Tk()
win.title(" 簡易計算機 ")
…
win.mainloop()
```

12-6-2 版面配置（建立 Frame）

本範例的版面安排如下圖，輸入框與輸出框是上下排列，因此可以使用 pack 方法安排，而 Button 元件是呈現規則狀的表格，適合用 grid 方法來安排各個按鈕元件的位置。



但是一個視窗裡只能夠使用一種佈局方式，如果要使用多種佈局方式，可以搭配 Frame() 元件將視窗分為多個內部框架。Frame（框架）是一種容器型的視窗元件，建立方式如下：

```
框架名稱 = tk.Frame( 容器名稱 , 參數 )
框架名稱 .pack()
```

例如下面程式建立分置於左右兩邊的框架：（frame.py）

```
01  import tkinter as tk
02  win = tk.Tk()
03  win.title("Frame")
04
05  frm1 = tk.Frame(win, width=100, height=300, bg="green" )
06  frm1.pack(side="right")
07
08  frm2 = tk.Frame(win, width=100, height=300, bg="yellow")
09  frm2.pack(side="left")
10
11  win.mainloop()
```

執行結果 »

「簡易計算機」範例裡使用了兩個框架，上下排列，框架裡還會放置其它元件，所以由元件來控制框架大小即可，並不需要特意去設定框架的寬跟高，程式如下：

```
frame = tk.Frame(win)
frame.pack()
frame1 = tk.Frame(win)
frame1.pack()
```

12-6-3 建置 Label、Entry 與 Button 元件

在上方框架（名稱為 frame）放置了一個 Label 元件與 Entry 元件，Entry 元件設定了背景顏色、字型與邊框大小；Label 元件設定了背景顏色、字型、元件靠右，預設 Label 文字「計算結果」。這兩個元件的寬度都是填滿整個視窗，所以將 pack() 的 fill 屬性設為 x。

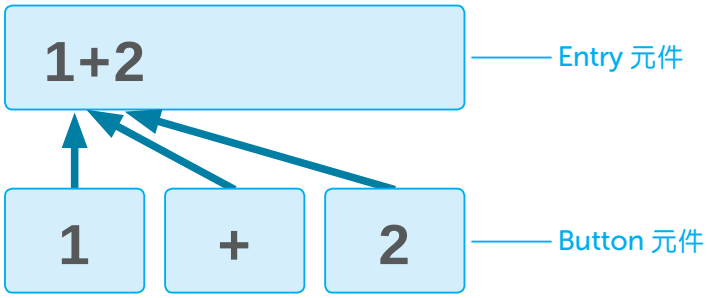
```
entry = tk.Entry(frame, bg="#ffccff", font = "Helvetica 15 bold" ,
borderwidth = 3)
entry.pack(fill="x")
label = tk.Label(frame, bg="#ffff00", font = "Helvetica 15 bold",
anchor="e" , text = " 計算結果 ")
label.pack(fill="x")
```

接下來產生按鈕物件。範例使用的按鈕總共有 17 個按鈕，分別是 0~9 及 .(點)、加 (+) 減 (-) 乘 (*) 除 (/) 鍵、cls 清除鈕、「=」等於鍵。這些按鈕的程式碼幾乎都是重複的，參數也大同小異，我們就可以把它獨立寫成函數，不但可以減少程式撰寫工作，還能增加程式的可讀性，程式碼簡潔也更容易維護。

以下是自訂的 btn 函數，函數指定 7 個參數：root（容器名稱）、text（按鈕文字）、row（grid 佈局的 row 值）、col（grid 佈局的 col 值）、w（按鈕寬度）、colspan（跨欄合併）、command（事件處理函數）。

```
def btn(root, text, row, col, w, colspan, command):
    button = tk.Button(root, text=text, width=w, command=command)
    button.grid(row=row, column=col, padx=5, pady=5, columnspan=colspan)
```

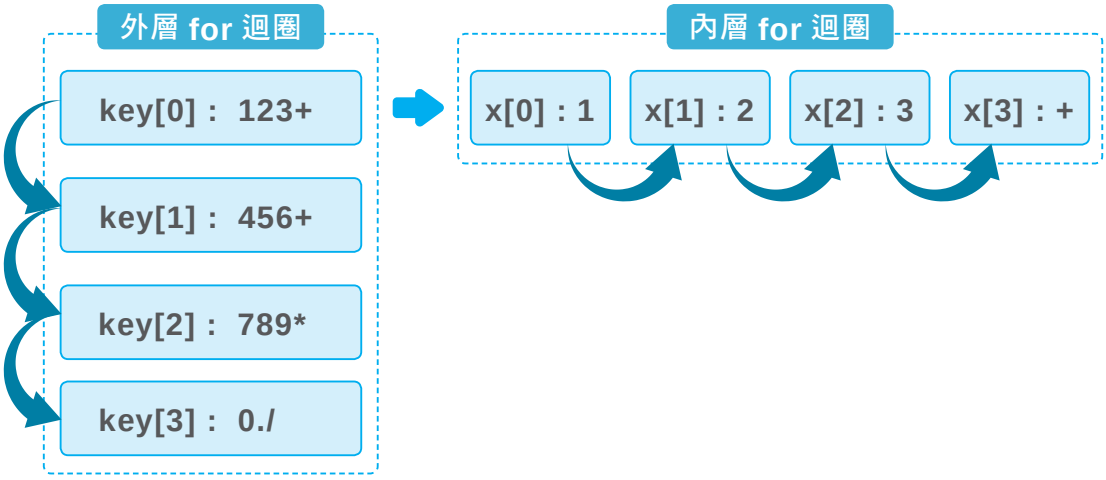
定義好了 btn 函數之後，就可以建立按鈕 Button 元件了。按鈕元件除了「Cls」鍵（清除鍵）與「=」鍵（等於鍵）之外，其餘按鈕都可以在使用者按下按鍵時將按鈕上的文字直接帶入 Entry 元件組合成運算式。



因此這裡使用巢狀 for 迴圈來產生 0~9、.（點）鍵以及加 (+) 減 (-) 乘 (*) 除 (/) 鍵，首先將其定義成 list 串列，for 迴圈就會自動將 list 串列裡的元素依序走訪一遍：

```
key=["123+", "456-", "789*", "0./"]   # 定義 list 串列
for x_index, x in enumerate( key ):
    for y_index, y in enumerate(x):
        btn(frame1, y, x_index, y_index, 6, 1, command=lambda y=y :
    get_input(y))
```

外層 for 迴圈第一次執行時會帶入 list 串列的第一個元素「123+」，內層的 for 迴圈則建立「1」、「2」、「3」、「+」四個按鈕元件，請參考以下示意圖。



各位可以發現外層 for 迴圈循序取出 list 串列元素（key[0]~key[3]），索引值（index）與 grid 佈局的 row 屬性位置相對應；內層 for 迴圈循序取出「123+」字串元素 (x[0]~x[3])，索引值與 column 屬性位置對應。因此使用了 enumerate 函數取出 for 迴圈的索引值直接帶入 grid 的 row 屬性與 column 屬性。

我們稍後再來介紹按鍵的事件處理函數，先將「Cls」按鈕與「=」按鈕完成，語法如下所示。

```
btn(frame1, 'Cls', 3, 3, 6, 1, command=clear)
btn(frame1, '=', 5, 0, 20, 4, command=calc)
```

12-6-4 加入事件處理函數

範例裡有三個事件處理函數，分別是取得按鍵文字的 get_input() 函數、清除輸入的 clear() 函數與負責計算的 calc() 函數，直接在 Button 元件的 command 設定事件處理函數就可以了，例如要指定 clear() 函數給「Cls」按鈕，可以這樣表示：

在這裡加上 clear 函數

```
btn(frame1, 'Cls', 3, 3, 6, 1, command=clear)
```

要特別注意的是這裡不能直接呼叫函數，也就是不能寫成「clear()」，如果這樣寫的話，程式執行到此就會直接執行函數，而不會等到點擊按鈕時才去呼叫函數。因此這裡必須使用回呼函數（callback function）模式，把函數當成引數來傳遞。

回呼函數是不能帶入參數，如果要帶入參數，必須使用 lambda 運算式，lambda 是一種簡單的運算表示方法，用來建立匿名函數（anonymous function），格式如下：

```
lambda 參數：運算式
```

例如以下是一個具名的加法函數 sum()：

```
def sum(x):
    return x+1

print(sum(2))     # 回傳計算結果 3
```

用 lambda 來表示的話，只要改寫成：

```
sum = lambda x : x+1
print(sum(2))     # 回傳計算結果 3
```

lambda 運算式中的參數，是在程式執行的時候取值，而不是在定義的時候，舉例來說：

```
y=5
sum = lambda x : x + y
y=10
print(sum(2))     # 回傳計算結果 12
```

直覺會認為 y=5 帶入 lambda 運算式所以 sum(2) 得到的結果應該是 7，事實上執行結果卻是 12，這是因為程式執行到第 3 行時變數 y 的值已經指定為 10 了。有一個變通的方式，就是先將參數值指定給 lambda 的預設參數（default parameter），如此一來就能先取得參數值，例如：

```
y=5
sum = lambda x, y=y : x + y
y=10
print(sum(2))     # 回傳計算結果 7
```

回到簡易計算機範例，範例中使用 for 迴圈建立的按鈕必須將按鈕上的字傳給事件處理函數使用，因此使用 lambda 運算式，變數 y 的值必須在 for 迴圈執行時就給定，所以這裡使用了 lambda 的預設參數，程式如下：

```
btn(frame1, y, x_index, y_index, 6, 1, command=(lambda y=y :
get_input(y)))
```

12-6-5 捕捉錯誤訊息

當按下「=」按鈕必須計算運算式的結果，並且容許錯誤輸入，如果輸入錯誤的話就輸出錯誤訊息「Infinity」。要捕捉錯誤訊息，最簡單的方式就是進入程式之後，以 try...except... 捕捉執行階段的錯誤，格式如下：

```
try:
    # 主程式
except:
    # 例外處理程序 (exception handlerexcept)
```

錯誤發生時，就會執行 except 區塊裡的例外處理程序。接著我們來看一下 calc() 函數：

```
01  def calc():
02      try:
03          input = entry.get()              # 取得 entry 元件輸入的內容
04          output = eval(input)             # 取得運算結果
05          entry.delete(0, "end")           # 清除 entry 元件輸入的內容
06          label.config(text = output)      # 在 label 元件顯示文字
07      except:
08          label.config(text = "Infinity")  # 在 label 元件顯示錯誤訊息
```

上面第 4 行使用了 eval() 函數將字串當成運算式並返回計算結果，eval() 的格式如下：

```
eval(String)
```

例如：

```
eval("2 + 2")
```

其輸出結果為 4，再舉一個例子：

```
eval("(5+3)*10 -8")
```

其輸出結果為 72，再舉一個例子：

```
eval("1+4*3/2")
```

其輸出結果為 7.0。

本章範例就說明至此，完整程式碼如下：

〈簡易計算機（GUI）〉

檔名 Review_calculator.py

```
# -*- coding: utf-8 -*-
"""
程式名稱：簡易計算機 (GUI 介面 )
題目要求：
1. 使用 GUI 介面 , 需有 0~9. 鍵、加減乘除 (+-*/) 鍵、Cls 鍵 ( 清除 )
2.input 區塊可用按鍵輸入也可用鍵盤輸入
3. 容許錯誤輸入，輸出錯誤訊息「Infinity」. ( 例如輸入： 10/0，輸出：Infinity.)
"""

def btn(root, text, row, col, w, colspan, command):
    button = tk.Button(root, text=text, width=w, command=command)
    button.grid(row=row, column=col, padx=5, pady=5, columnspan=colspan)

def get_input(argu):
    entry.insert("end",argu)            # 將按鈕文字輸入 entry 元件

def calc():
    try:
        input = entry.get()             # 取得 entry 元件輸入的內容
        output = eval(input)            # 取得運算結果
        entry.delete(0, "end")          # 清除 entry 元件輸入的內容
        label.config(text = output)     # 在 label 元件顯示文字
    except:
        label.config(text = "Infinity")  # 在 label 元件顯示錯誤訊息

def clear():
    entry.delete(0, "end")
    label.config(text = "")

import tkinter as tk
```

```
31  win = tk.Tk()
32  win.title(" 簡易計算機 ")
33
34  frame = tk.Frame(win)
35  frame.pack()
36  frame1 = tk.Frame(win)
37  frame1.pack()
38
39  entry = tk.Entry(frame, bg="#ffccff", font = "Helvetica 15 bold",
    borderwidth = 3)
40  entry.pack(fill="x")
41  label = tk.Label(frame, bg="#ffff00", font = "Helvetica 15 bold",
    anchor="e" , text = " 計算結果 ")
42  label.pack(fill="x")
43
44
45  key=["123+", "456-", "789*", "0./"]
46  for x_index, x in enumerate(key):
47      for y_index, y in enumerate(x):
48          btn(frame1, y, x_index, y_index, 6, 1, command=(lambda
            y=y : get_input(y)))
49
50  btn(frame1, 'Cls', 3, 3, 6, 1, command=clear)
51  btn(frame1, '=', 5, 0, 20, 4, command=calc)
52
53  win.mainloop()
```

Chapter 12 重點回顧

1. Graphical User Interface （圖形使用者介面，簡稱 GUI）是指使用圖形方式顯示使用者操作介面。
2. tkinter 套件是一種內建標準模組，允許使用者在不同的作業平台下建構 GUI 介面。
3. GUI 介面的最外層是一個視窗物件，用來容納視窗元件如標籤、按鈕等。
4. 主視窗設定完成之後，必須在程式最後使用 mainloop() 方法讓程式進入循環偵聽模式來偵測使用者觸發的「事件（Event）」，直到關閉視窗為止。
5. tkinter 提供 3 種佈局方法：pack、grid 及 place。
   (1) pack 方法預設以由上而下方式擺放元件。
   (2) grid 是以表格方式擺放元件。
   (3) place 方法是透過元件在視窗中的絕對位置與相對位置來指定位置。
6. Label 元件的功能是用來顯示文字，它是一個非互動式的元件。
7. 如果想變更 Label 元件的文字可以指定文字變數給 textvariable 屬性。
8. Entry 元件可以讓使用者輸入資料，它是單行模式，想要輸入多行就要使用 Text 元件。如果要輸入 Entry 元件的預設值，可以使用 insert 方法。如果要刪除 Entry 元件裡的文字，可以使用 delete 方法。
9. Text 元件用來儲存或顯示多行文字，包括純文字或格式文件，Text 元件也可以被用作文字編輯器。
10. 想在已建立的文字方塊設定文字內容，必須呼叫 insert() 方法。
11. 當建立 Text 元件後，如果要變更元件的參數設定，可以使用 config 方法。
12. 選項元件有兩種：Checkbutton（核取方塊）提供多選的功能；Radiobutton（單選按鈕）只能從多個項目中擇一，無法多選。
13. Checkbutton 是從列出的項目當中做不同的選擇，可以通通不選，也能同時選取，或者只挑選幾個。
14. 捲軸常被使用在文字區域（Text）、清單方塊（Listbox）或是畫布（Canvas）等元件。
15. 對話方塊（Dialog）是一種使用者與程式間互動的介面，tkinter 套件中的對話方塊包括：messagebox、simpledialog、filedialog、colorchooser。

16. 訊息方塊主要目的就是以簡便的訊息與使用者互動。方法概分兩大類，詢問和顯示。詢問「ask」為開頭。顯示以「show」開頭。
17. simpledialog（簡單型方塊）有三個方法：處理字串的 askstring() 方法；用於整數的 askinteger() 方法；只能採浮點數的 askfloat() 方法。
18. filedialog（檔案方塊）本身是 tkinter 套件的模組，有兩個方法與它有關 askopenfile() 方法：開啟檔案。asksaveasfile() 方法：儲存檔案。
19. colorchooser（調色方塊）元件提供色彩的選擇，askcolor() 方法可產生標準對話方塊，以調色盤供顏色的選擇。
20. 要產生功能表就得使用 Menu 元件。不過 Menu 元件只能產生功能表的骨架，還必須配合 Menu 元件的相關方法。
21. Menu 元件中的 add() 方法之參數 type 可用來指定功能表的種類，包括：command、cascade(submenu)、checkbutton、radiobutton 或 separator。
22. Canvas 元件（畫布控制元件）可以繪圖，包括線條、幾何圖形或文字等。
23. 利用 Canvas 元件進行圖形繪製有兩種座標系統：
    (1) Windows 座標系統，以螢幕的左上角為原點 (x = 0, y = 0)。
    (2) Canvas 元件的座標系統，依指定位置進行繪製。
24. 除非特別指定，繪製的物件會以 Canvas 元件的座標系統為主。所有元件都要加入主視窗物件，再呼叫 pack() 方法納入版面管理。
25. 要在 Canvas 元件中載入點陣圖可呼叫 create_image() 方法。此處 create_image() 方法無法讀取一般的圖片而是經過處理的 image 物件。
    步驟 1：以 PhotoImage() 建構函數去讀取圖片，以 image 物件儲存，我們可以載入的圖片格式有「*.png」或「*.gif」的圖片。
    步驟 2：再以 create_image() 方法的參數 image 取得。
26. Canvas 元件繪製幾何圖形包括：弧形、扇形、線、圓形或橢圓形、矩形、文字等。
27. lambda 是一種簡單的運算表示方法，用來建立匿名函數（anonymous function）。
28. lambda 運算式中的參數，是在程式執行的時候取值，而不是在定義的時候。
29. 要捕捉錯誤訊息，最簡單的方式就是進入程式之後，以 try...except... 捕捉執行階段的錯誤。

Chapter 12 學習評量

選擇題

(　　) 1. 對於 tkinter 套件的描述，何者有誤？
(A) Python 標準函數有所支援
(B) 只適用於 Windows 作業平台
(C) 使用 import 敘述匯入此套件
(D) 用於 GUI 介面。

(　　) 2. 產生主視窗物件後，要呼叫哪一個方法來清除它？
(A) destroy()　(B) title() 方法
(C) mainloop()　(D) invoke() 方法。

(　　) 3. 某個元件呼叫 grid() 方法，若參數「row = 1, column = 0」表示？
(A) 第一列、第一欄　(B) 第一列、第二欄
(C) 第二列、第一欄　(D) 第二列、第二欄。

(　　) 4. Button 元件的哪一個屬性是用來處理按下按鈕的回應？
(A) 屬性 command　(B) 屬性 state
(C) 屬性 cursor　(D) 屬性 justify。

(　　) 5. filedialog 對話方塊的兩個方法中，哪一個參數可指定檔案類型？
(A) filetypes　(B) initaldir　(C) initalfile　(D) title。

(　　) 6. Menu 元件建立功能表，哪一個方法用來產生主功能表的項目？
(A) add_radiobutton()　(B) config()
(C) add_command()　(D) add_cascade() 方法。

(　　) 7. Canvas 元件，要讀取一般圖片，要呼叫哪一個方法？
(A) creat_arc()　(B) create_line()
(C) create_bitmap()　(D) create_image()。

填充題

1. 主視窗設定完成之後，必須在程式最後使用_____方法讓程式進入循環偵聽模式。
2. 所謂_____是由使用者的動作或系統所觸發的訊號。
3. tkinter 提供了 3 種佈局方法：_____、_____以及_____。
4. _____元件的功能是用來顯示文字，它是一個非互動式的元件。
5. 當使用者按下按鈕時會觸發_____事件，呼叫對應的事件處理函數。
6. 如果想變更按鈕元件的文字可以指定文字變數給_____屬性。
7. _____元件可以讓使用者輸入資料，它是單行模式。
8. Text 元件如果要要禁止使用者編輯，請將_____參數值設為「tk.DISABLED」。
9. 選項元件有兩種：_____和_____。
10. 通常_____最主要的功能就是提供訊息。

簡答題

1. 何謂事件？試簡述之。

2. 請說明 tkinter 有哪幾種佈局方式。

3. 要改變 button 元件上的文字內容或變更屬性（文字顏色、底色、寬、高、字型等），有哪兩種做法？

4. 請問 Entry 元件及 Text 元件都可以讓使用者輸入資料，但兩者間有何最大的不同點？

5. 選項元件有哪兩種？兩者在功能上有何不同？

6. 捲軸（Scrollbar）通常被使用在哪幾種元件中？請至少舉出兩種。

7. 試簡述使用訊息方塊的目的及訊息方塊的種類。

8. 試簡述簡單型對話方塊（simpledialog）有哪三個方法。

9. 試簡述 filedialog（檔案對話方塊）有哪幾種方法。

10. Canvas 元件具有畫布功能，能藉由滑鼠的移動做基本繪製；它有哪兩種座標系統？

附錄

學習評量解答

Chapter 1

選擇題

1. (D)　2. (B)　3. (D)　4. (B)　5. (C)

填充題

1. 低階；高階
2. 機器語言
3. 編譯器；直譯器
4. 流程圖
5. Conda
6. Spyder
7. 縮排

簡答題

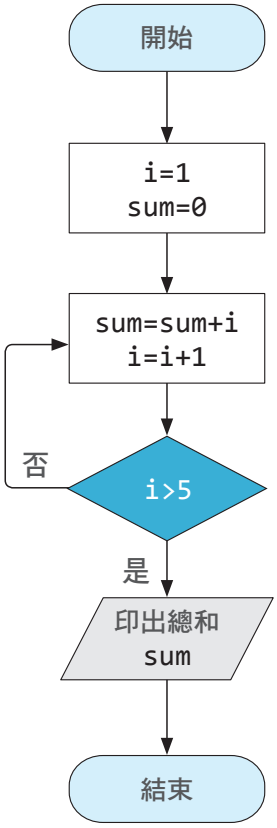
1. Python 的註解有兩種，單行註解跟多行註解。單行註解符號是「#」，在「#」以後的文字都會被當成註解。多行註解是以三個引號包住註解文字，引號可以是成對的三個雙引號。

2. 編譯：編譯器會先檢查整個程式，完全沒有語法錯誤之後，再連結相關資源輸出可執行檔。編譯完成的可執行檔是可以直接執行的檔案，每一次執行時，不需再翻譯，所以執行速度較快。缺點是編譯過程中發生錯誤時，必須回到程式碼找出錯誤的地方加以更正，再重新編譯、連結、產生可執行檔，開發過程會比較不便。編譯式的語言如 C、FORTRAN、COBOL 等等。

   直譯：Python 就是屬於直譯式的語言，直譯顧名思義就是一邊解讀原始碼，一邊執行，當錯發生時會停止執行並顯示錯誤行數與原因，對程式開發來說會比較方便，也因為它不產生執行檔，每一次執行都必須經過直譯才能執行，因此執行效率會比編譯式稍差。直譯式的語言如 HTML、JavaScript、Python 等等。

3. 步驟 1：指定 i=1、sum=0
   步驟 2：sum 的值 +i （sum=sum+i）
   步驟 3：i 的值 +1 （i=i+1）
   步驟 4：如果 i 大於 5，演算法結束，否則，返回重新執行步驟 2。

4.



5. (1) 輸入資料（Input）：0 個或多個輸入。
   (2) 輸出結果（Output）：至少 1 個以上的輸出結果。
   (3) 明確性：描述的處理程序必須是明確，不能模稜兩可。
   (4) 有限性：必須在有限的步驟內完成工作，不可以有無窮迴圈。
   (5) 正確性：可以正確地解決問題。

6. (1) 程式碼簡潔易讀；(2) 跨平台；(3) 自由 / 開放原始碼；
   (4) 多重思維的程式語言；(5) 擴充能力強的膠水語言

7. 隨著程式碼的規模日益龐大，現在軟體工程的重點就在可讀性與維護性，而適時使用「註解」就是達到這兩項重點的最主要方法。註解不僅可以幫助其他的程式設計師閱讀程式內容，在日後程式維護時，清楚的註解也能省下不少維護成本。

Chapter 2

選擇題

1. (A)　2. (D)　3. (B)

填充題

1. 保留字
2. help()
3. 靜態型別；動態型別
4. True；False
5. %；format

簡答題

1.

fileName01	有效
$result	無效，不能有 $
2_result	無效，第一個字元不能是數字
number_item	有效

2. 整數（int）：100　　浮點數（float）：25.3　　布林值（bool）：True

3.
```
01  user_name = input(" 請輸入姓名：")
02  score = input(" 請輸入數學成績：")
03  print("%s 的數學成績：%5.2f" % (user_name,float(score)))
```

4. (1) 不需要理會引數資料型態，一律用 {} 表示
   (2) 可使用多個引數，同一個引數可以多次輸出，位置可以不同

5. int()：強制轉換為整數資料型態
   float()：強制轉換為浮點數資料型態
   str()：強制轉換為字串資料型態

Chapter 3

選擇題

1. (B)　2. (A)　3. (C)　4. (D)　5. (C)

填充題

1. 運算子；運算元
2. 單一指派；複合指派
3. and；or；not
4. 快捷運算
5. 由左到右

簡答題

1. 8
2. 16
3. 因為 15 的二進位表示法為 1111，10 的二進位表示法為 1010，兩者執行 AND 運算後，結果為 $(1010)_2$，也就是 $(10)_{10}$。
4. NOT 作用是取 1 的補數（complement），也就是 0 與 1 互換。例如 a=12，二進位表示法為 00001100，取 1 的補數後，由於所有位元都會進行 0 與 1 互換，因此運算後的結果得到 -13。
5. 在等號關係是 "==" 運算子，至於 "=" 則是指派運算子，這種差距很容易造成程式碼撰寫時的疏忽，請多加留意。
6. 200.0；-59.2；-1.0
7.

```
sales_list=[60,80,55]
cake = int(input(" 請輸入購買的蛋糕數量："))
Cookies = int(input(" 請輸入購買的餅乾數量："))
coffee = int(input(" 請輸入購買的咖啡數量："))
total = cake*sales_list[0] + Cookies*sales_list[1] + coffee*sales_list[2]
print(' 購買總金額為：', total)
```

Chapter 4

選擇題

1. (D)　2. (C)　3. (B)　4. (B)　5. (A)

填充題

1. for；while
2. range()
3. break
4. continue
5. 變數初始值；迴圈條件式；調整變數增減值

簡答題

1.
```
N = int(input(" 請輸入一個數值 : "))
print('False' if N%3!=0 else 'True')
```

2.
```
sum = 0
i = 1
while i <= 100:
    sum += i
    i += 1
print(sum)
```

3.
```
sum = 0
for i in range(101):
    sum+=i
print(sum)
```

4.
```
sum=0
n1=1
n=int(input(" 請輸入任一整數 :"))
for i in range(1,n+1):
    for j in range(1,i+1):
        n1*=j; # n! 的值
    sum+=n1;# 1!+2!+3!+..n!
    n1=1
print("1!+2!+3!+...+{0}!={1}".format(n,sum))
```

5. 1
   4
   7
   10
   13

6.
```
sum=0
index=0
while index <= 50:
    sum=sum+index
    index += 2
print ('1~50 偶數總和 ', sum)
```

Chapter 5

選擇題

1. (A)　2. (C)　3. (B)

填充題

1. 字串
2. 「=」指派
3. \
4. format()
5. 切片
6. Unicode
7. replace()

簡答題

1.
```
words=input(" 請輸入字串 .")
words_lower=list(words.lower())

result =""
for w in words_lower:
    if ord(w) in range(97, 123):
        result+=w

result_l = len(result)
print(" 共有 {} 個英文字母 , 字母是 {}".format(result_l,result))
```

2.
```
strA = input(" 請輸入字串：")
revletters = strA[::-1]
print(" 原字串：{}\n 反轉後：{}".format(strA,revletters))
```

3.
```
with open("twisters.txt" , "r" ) as f:
    story=f.read()    # 讀出檔案內容

words="Peter"
sc=story.count(words)
print("{} 出現了 {} 次 ".format(words,sc))
```

4. (1) 透過索引值（index）取某個字元
   (2) 使用切片（slice）方法取某段字串
   (3) 使用 split() 函數分割字串

Chapter 6

選擇題

1. (D)　2. (C)　3. (C)　4. (D)　5. (D)

填充題

1. 內建函數；自訂函數
2. 形式參數；實際引數
3. 位置；關鍵字
4. 傳值
5. global

簡答題

1. sorted 函數與 sort () 方法都是排序，兩者功能大同小異，都有 reverse 與 key 參數，差別在於 sort() 方法只支援 list 串列資料進行排序，要注意的是 sort() 方法沒有回傳值，會直接排序序列的內容。

2. 自訂函數與 lambda() 有何不同？先以一個簡例做解說。

```
01  def result(x, y): # 自訂函數
02      return x+y
03  result = lambda x, y : x + y  #lambda() 函數
```

請注意，自訂函數時，函數名稱 result，可作為呼叫 lambda() 函數的變數名稱。定義函數時，函數主體有多行指令；但是 lambda() 函數只能有一行運算式。另外自訂函數有名稱，lambda() 函數無名稱，必須指定一個變數來儲存運算結果，再以變數名 result 來呼叫 lambda() 函數，依其定義傳入參數。在 result () 函數中以 return 指令回傳；lambda() 的運算結果由變數 result 儲存。

3. 10 次

4.
```
01  n = int(input(" 請輸入存入本金："))
02  y = int(input(" 請輸入期間 ( 年 )："))
03  r = 0.02
04  total= round(n * (1 + r/12) ** (12*y))
05  print(total)
```

5. 定義函數時要有「形式參數」來準備接收資料，而呼叫函數要有「實際引數」進行資料的傳遞工作。
   (1) 形式參數：定義函數時，用來接收實際引數所傳遞的資料，進入函數主體執行指令或運算。
   (2) 實際引數：程式中呼叫函數時，將資料傳遞給自訂函數。

6. Python 函數的引數分為位置引數與關鍵字引數，位置引數就是依照參數位置傳入引數，如果函數定義了 3 個參數，就要帶入 3 個引數，缺一不可，它具有順序性，不可錯亂。關鍵字引數就是藉由關鍵字來傳入引數，只要必要的參數都有指定，關鍵字引數的位置並不一定要依照參數順序。

7. Python 的引數傳遞以不可變和可變物件來運作：
   (1) 不可變物件 （如數值、字串）傳遞引數時，接近於「傳值」。
   (2) 可變物件（如串列）傳遞引數時，以「傳址」處理。簡單來說，如果可變物件被修改內容值，因為佔用同一位址，會連動影響函數外部的值。

8. 變數依其有效範圍分為全域變數與區域變數：
   (1) 全域變數：定義在函數外的變數，其有效範圍適用於整個檔案 (*.py)。
   (2) 區域變數：適用於所宣告的函數或流程控制的程式區塊，離開此範圍就會結束其生命週期。

9. 一個函數或副程式，是由自身所定義或呼叫的，稱為遞迴，至少要定義 2 種條件：
   (1) 一個可以反覆執行的遞迴過程
   (2) 一個跳出執行過程的出口

Chapter 7

選擇題

1. (B)　2. (C)　3. (A)　4. (C)　5. (D)

填充題

1. tuple
2. del
3. + ; *
4. key ; value
5. get()

簡答題

1.
```
with open("twisters.txt", "r") as f:
    story=f.read()    # 讀出檔案內容

story=story.replace("?", " ");
story=story.replace("\n", " ");
story_list=story.split()

words={}
for i in range(len(story_list)):
    if story_list[i] not in  words:
        words[story_list[i]]=story_list.count(story_list[i])

print(words)
```

2. {'a': 1, 'b': 9, 'c': 4, 'd': 25, 'e': 16}

3.

資料型態	tuple	list	dict	set
中文名稱	序對	串列	字典	集合
使用符號	()	[]	{}	{}
具順序性	有序	有序	無序	無序
可變 / 不可變	不可	可	可	可
舉例	(1, 2, 3)	[1,2,3]	{'word1':'apple'}	{1, 2, 3}

4. [4, 5, 6, 7, 8, 9, 10]

5. {'name': 'Tom', 'age': 18, 'city': ' 高雄 ', 'hobby': ' 籃球 '}

Chapter 8

選擇題

1. (D)　2. (A)

填充題

1. 套件
2. __init__.py
3. import
4. 別名.函數名稱
5. __main__

簡答題

1.
```
import datetime
def isVaildDate(yy,mm,dd):
    try:
        return datetime.date(yy,mm,dd)
    except:
        return " 日期錯誤 "

print(isVaildDate(2017,2,30))
```

2. 如果要一次匯入多個套件，則必須以逗點 (,) 隔開不同的套件名稱，語法如下：

```
import 套件名稱 1, 套件名稱 2, ..., 套件名稱 n
```

例如：同時匯入 Python 標準模組的數學和亂數模組。

```
import math, random
```

3.
```
import 套件名稱 as 別名
```

4. Python 的標準函數庫裡面非常多好用的模組，可以讓我們省下不少程式開發的時間，當程式裡會同時匯入多個模組，這時函數名稱就有可能會重複，好在 Python 提供命名空間機制，它就像是一個容器，將模組資源限定在模組的命名空間內，避免不同模組之間同名衝突的問題。

5. (1) math 模組提供了 C 函數庫中底層的浮點數運算的函數。
(2) random 模組提供了隨機選擇的工具。
(3) datetime 模組有許多操作日期以及時間的函數，並支援時區換算。
(4) 更具閱讀性的 pprint 套件。
(5) time 和 calendar 模組可以用於格式化日期和時間，也定義了一些與時間和日期相關的函數。
(6) os 模組是作業系統相關模組。
(7) sys 模組包含與 Python 直譯器相關的屬性與函數

Chapter 9

選擇題

1. (D)　2. (D)　3. (C)

填充題

1. 二進位檔　2. 檔案系統
3. 檔案物件　4. r
5. readline()　6. b

簡答題

1. 檔案如果依儲存方式來分類，可以分文字檔與二進位檔兩種。
2. 「絕對路徑」指的是一個絕對的位置，並不會隨著現在目錄的改變而改變。「相對路徑」就是相對於現在目錄的路徑表示法，因此「相對路徑」所指到的檔案或目錄，會隨著現在目錄的不同而改變。
3. 首先必須以 open() 方法開啟指定的檔案，接著使用檔案物件所提供的 read()、readline() 或 readlines() 方法從檔案讀取資料，最後再以 close() 方法關閉檔案。

Chapter 10

選擇題

1. (B)　2. (C)　3. (A)　4. (B)　5. (D)
6. (B)　7. (D)

填充題

1. 語法
2. 執行期間錯誤
3. 邏輯錯誤
4. NameError
5. finally

簡答題

1. 程式錯誤的三種類型：語法錯誤、執行期間錯誤、邏輯錯誤。
2. 當程式執行時，產生了不是程式設計人員原先預期的結果，這種情況下，Python 直譯器會接手管理，並終止程式的運作。
3. LookupError：當映射或序列型別的鍵或索引無效時所引發。
   OSError：操作系統函數發生錯誤時所引發。
   NameError：是指名稱沒有定義的錯誤。
   RuntimeError：執行期間錯誤。
4. 比較常見的例外處理的時機為程式與外部的人員或裝置進行互動的 I/O 行為時，例如要求使用者輸入數字或輸入字串，或是與外部資料庫連結時，就算是所撰寫的程式碼完全正確，但有可能因為網路線路問題或是資料庫連結過程發生異常或網路中斷，都會造成程式發生錯誤，引發系統丟出例外，並強迫中斷程式的執行。

Chapter 11

選擇題

1. (C)　2. (D)　3. (D)

填充題

1. 由上而下；模組化
2. 類別
3. 封裝；繼承；多型
4. public
5. 實體化
6. self
7. 兩個下底線
8. 衍生
9. 覆寫
10. has_a

簡答題

1. 在傳統程式設計的方法中，主要是以「結構化程式設計」為主，就是「由上而下設計」 與「模組化設計」。每一個模組會個別完成特定功能，主程式則組合每個模組後，完成最後要求的功能。不過一旦主程式要求功能變動時，則可能許多模組內的資料與程式碼都需要同步變動，而這也是結構化程式設計無法有效使用程式碼的主因。「物件導向程式設計」能以一種生活化、可讀性更高的設計觀念來進行程式開發工作，並且所開發出來的程式也較容易擴充、修改及維護，以彌補「結構化程式設計」的不足。

2. 所謂封裝是利用「類別」來實作「抽象化資料型態」。所謂「抽象化」，是讓使用者只能接觸到這些方法（函數），而無法直接使用資料。資料抽象化的目的是方便於日後的維護，當應用程式的複雜性愈高，資料抽象化做得愈好，愈能提高程式的再利用性和閱讀性。另外，抽象化也符合了資訊隱藏的意義，這就是「封裝」的主要作用。

3. 簡單的說，多型最直接的定義就是讓具有繼承關係的不同類別物件，可以呼叫相同名稱的成員函數，並產生不同的反應結果。

4. 當建立物件時，__init__() 這個特殊方法會為物件進行初始化的工作，例如設定初始值或進行資料庫連線等工作。

5. 通常宣告類別後，會將類別實體化為物件，並將物件指派給變數，再透過這個變數來存取物件。在 Python 程式語言有一項重要特性，那就是每個東西都是物件，所以寫程式時，也可以在不需要將物件指派給變數的情況下去使用物件，這就是一種稱為匿名物件的程式設計技巧。

Chapter 12

選擇題

1. (B)　2. (A)　3. (C)　4. (A)　5. (A)

6. (D)　7. (D)

填充題

1. mainloop()
2. 事件
3. pack；grid；place
4. Label
5. click
6. textvariable
7. Entry
8. state
9. Checkbutton；Radiobutton
10. messagebox

簡答題

1. 所謂「事件」是由使用者的動作或系統所觸發的訊號。舉例來說，當使用者按下按鍵時會觸發 click 事件，系統會呼叫指定的事件處理函數來回應此一事件。
2. tkinter 提供了 3 種佈局方法：pack、grid 以及 place。
3. 透過 textvariable 參數指定文字變數、透過 config 方法更改文字內容或屬性值。
4. Entry 元件可讓使用者輸入資料，為單行模式，想要輸入多行要使用 Text 元件。
5. 選項元件有兩種：Checkbutton（核取方塊）和 Radiobutton（單選按鈕）。Checkbutton 提供多選功能；Radiobutton 只能從多個項目中擇一，無法多選。
6. 捲軸通常被使用在文字區域、清單方塊或是畫布等。
7. 訊息方塊主要目的就是以簡便的訊息與使用者互動。概分兩大類，詢問和顯示。詢問「ask」為開頭，伴隨 2 ～ 3 個按鈕來產生互動行為。顯示以「show」開頭，只會顯示一個「確定」鈕。
8. simpledialog（簡單型對話方塊）有三個方法：處理字串的 askstring() 方法；用於整數的 askinteger() 方法；只能採浮點數的 askfloat() 方法。
9. filedialog（檔案對話方塊）本身是 tkinter 套件的模組，以開啟檔案和儲存檔案最有密切關係。askopenfile() 方法：開啟檔案；asksaveasfile() 方法：儲存檔案。
10. (1) Windows 座標系統，以螢幕的左上角為原點（x = 0, y = 0）。
    (2) Canvas 元件的座標系統，依指定位置進行繪製。

達人必學 Python 程式設計

2018 年 7 月初版
書　號 PB371

國家圖書館出版品預行編目（CIP）

達人必學 Python 程式設計 / 勁樺科技編著 . --
初版 .-- 新北市：台科大圖書，2018.07
面 ；　公分
ISBN 978-986-455-680-9（平裝）
1.Python（電腦程式語言）
312.32P97　　107007706

編 著 者　勁樺科技
責任編輯　沈育卿
版面構成　魏怡茹
封面設計　柯進祥

出 版 者　台科大圖書股份有限公司
門市地址　242 新北市新莊區中正路 649-8 號 8 樓
電　　話　02-2908-0313
傳　　真　02-2908-0112
網　　址　www.tiked.com.tw
電子郵件　service@tiked.com.tw

有著作權　侵害必究

▶ 本書受著作權法保護。未經本公司事前書面授權，不得以任何方式（包括儲存於資料庫或任何存取系統內）作全部或局部之翻印、仿製或轉載。

▶ 書內圖片、資料的來源已盡查明之責，若有疏漏致著作權遭侵犯，我們在此致歉，並請有關人士致函本公司，我們將作出適當的修訂和安排。

郵購帳號　19133960
戶　　名　台科大圖書股份有限公司
※ 郵撥訂購未滿 1500 元者，請付郵資：本島地區 100 元 / 外島地區 200 元
客服專線　0800-000-599
網路購書　www.tiked.com.tw

各服務中心專線
・總　公　司　02-2908-5945　　・台北服務中心　02-2908-5945
・台中服務中心　04-2263-5882　　・高雄服務中心　07-555-7947

線上讀者回函
歡迎給予鼓勵與建議
www.tiked.com.tw/PB371