» 程式設計的相關環境

電腦的組成要素

電腦是由各種設備，如顯示器、鍵盤、滑鼠所組成。這些物理上的設備稱為**硬體**（hardware），除了在運作上不可或缺外，硬體也代表包含機殼在內的物理實體。在硬體當中，電腦運作所需要的五個裝置就稱為**五大單元**（圖 1-1）。

現代的電腦不只是個人電腦（以下統一稱為 PC）與智慧型手機，還有伺服器與路由器等各式機器，無論是哪一種，都是由這裡的五大單元所組成。然而，只有硬體並無法讓電腦運作，還需要 Windows 與 macOS、Android、iOS 等**作業系統**（OS，基礎軟體），以及瀏覽網站時的網頁瀏覽器，播放音樂與相機功能、計算機與記事本、文書作業與試算表等**應用程式**（application）。

硬體之外的部分，就取用英語中 hard 的反義詞 soft 並稱之為**軟體**（software，圖 1-2）。即使硬體相同，導入不同軟體後使用方式就完全不同。

生活中也有一些產品將硬體與軟體結合，例如音樂播放器與數位相機等。**硬體在製作完成後即使發現問題也難以變更，而軟體若有狀況，有時可以透過重新發布修正後程式來進行變更。**

軟體與程式的差異

作業系統與應用程式等軟體是由執行檔（即**程式**）、使用手冊等文件及資料所組成。程式則包含了執行檔與函式庫（參考 **6-2**）。**程式設計指的是「開發程式」，開發程式的人則稱為程式設計師（programmer）。**

圖 1-1 五大單元

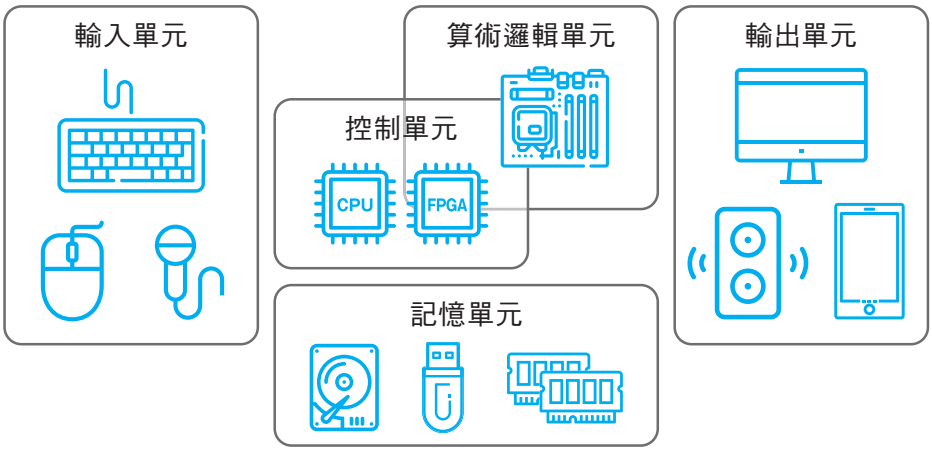


圖 1-2 硬體與各種軟體的關係

Point

- 理解電腦的運作時，如果將五大單元分開思考，較易理解個別功能。
- 軟體包含作業系統與應用程式。
- 程式是軟體的一部分，包含執行檔與函式庫等。

» 程式運作的環境

有 PC 就能使用的應用程式

許多人常用的應用程式有網頁瀏覽器與文書軟體、電子試算表等，這些稱為**桌面應用程式**（desktop application）（圖 1-3）。

使用桌面應用程式除了要將程式儲存於電腦外，還必須儲存許多資料到電腦中才能使用，因此若是要在其他 PC 上使用相同的程式與資料，就必須另行複製與安裝。

這樣一來，桌面應用程式就可以**控制與 PC 連接的硬體**。就像音樂播放軟體能控制喇叭，文書軟體可以使用印表機，要使用硬體，就必須要有桌面應用程式。而桌面應用程式的另一個特徵是不需連接網路也可以使用。

連上網路就能在任意地點使用的應用程式

最近網路上提供的服務越來越多，除了 Facebook 與 Twitter 等社群網路服務，還有像是 Amazon 與樂天等購物網站、Google 與 Yahoo! 等搜尋服務，這些服務都在企業所提供的網路伺服器上運作。

這種在網路環境下才能運作的應用程式，就稱為**網路應用程式**。使用網路應用程式時，會**需要網頁瀏覽器等軟體**（圖 1-4）。

能讓智慧型手機發揮最大功能的應用程式

最近有很多人透過智慧型手機收集資訊，而不是透過 PC。在手機運作的應用程式，就是**手機應用程式**。

利用智慧型手機所具備的 GPS 功能、相機、網路、感測器等硬體，許多應用程式應運而生，例如遊戲。

» 轉換為電腦能處理的格式

用於程式設計的檔案

人類可以理解使用日語或英語等自然語言所寫的文章，而製作設計稿時如果使用圖表，也會變得更加容易理解且更直觀。但是電腦並無法直接處理文字資料與設計稿，因此我們必須將想要處理的內容轉換為電腦能夠理解的語言（機器語言）（圖 2-1）。

人類要使用機器語言相當困難，因此我們會使用容易轉換成機器語言的程式語言，來取代日常中的自然語言。軟體的開發，就是依照程式語言語法建立**原始碼**。

接下來我們需要將以程式語言編寫的原始碼，轉換成電腦可處理的機器語言程式，其檔案格式就是**執行檔**。

這個**編寫原始碼，建立程式的作業流程**，就叫做程式設計。程式設計有時也包含製作設計稿、測試程式是否正確運作，以及去除程式錯誤（bug）的除錯（debug）程序。

如何轉換為程式

要將原始碼轉換為程式，有**編譯器**與**直譯器**兩種方法（圖 2-2）。編譯器是**事前就一次將所有原始碼轉換為程式，執行時處理的是程式**。就像是翻譯文章一樣，事先把內容轉換好，執行時就能更快速地處理。

而直譯器的做法，是**在執行的同時一邊轉換原始碼**，就像是口譯一樣，將對方所說的話從旁傳遞出去，在處理上較為費時，不過發生意外狀況時可以簡單地稍作修正並再次執行。

圖 2-1 人與電腦擅長的語言

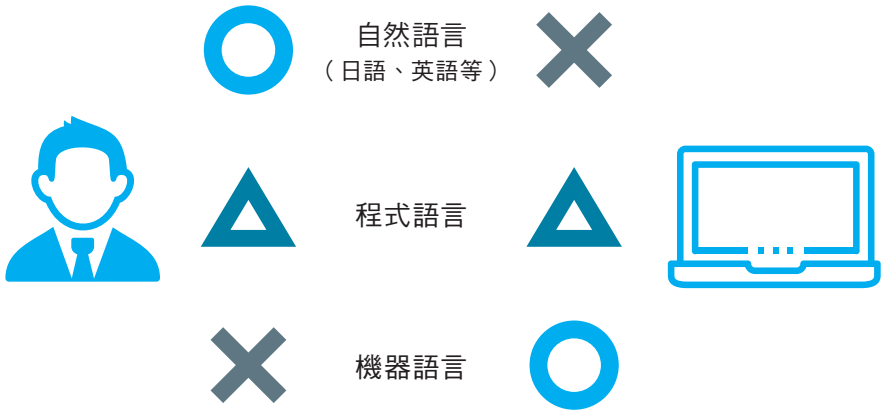
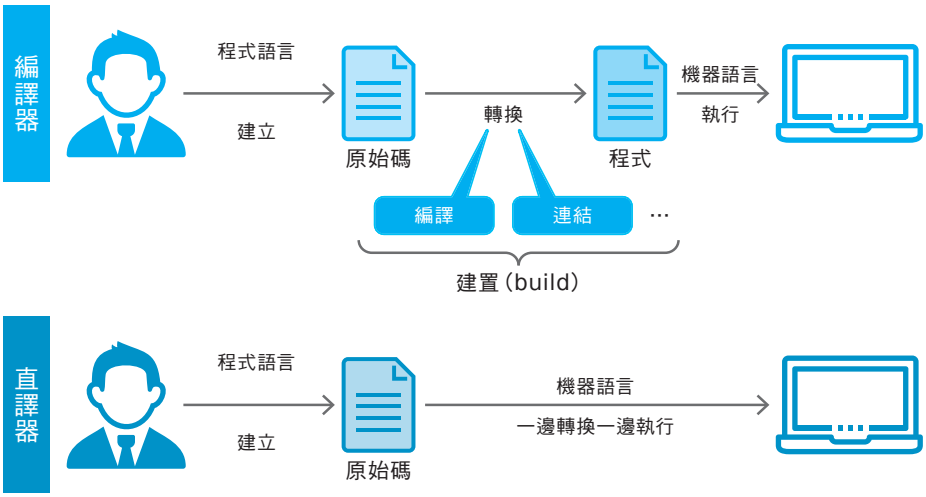


圖 2-2 編譯器與直譯器



Point

- 電腦無法理解人類擅長的自然語言，人類也難以理解電腦處理資料所使用的機器語言，因此才會使用程式語言。
- 以電腦執行程式語言編寫的原始碼時，有編譯器與直譯器這兩種方式。

» 人與電腦能理解的語言

可由電腦直接處理的低階語言

語言可以透過「較接近電腦還是人類」為基準進行分類（圖 2-3）。電腦可以直接處理的只有機器語言，由於電腦是以二進位的方式處理資料，機器語言會是 0 和 1 的排列，但有時也會採用十六進位的方式，讓人類較能理解。

只是人類要理解十六進位依然相當困難，因此開始使用組合語言。組合語言**與機器語言是一對一的關係，可以寫得像英文一樣**，因此人類在閱讀上較為容易。

將組合語言寫出的原始碼轉換為機器語言，就叫做組譯（assemble），而進行轉換的程式就稱為組譯器（assembler）。有些人也會將組合語言的英文說為 assembler language。

這些較貼近電腦的語言，如機器語言和組合語言，就稱為**低階語言**（低級語言）。

人類較易閱讀的高階語言

人類並不是不能閱讀組合語言，但在製作大型程式，需要撰寫大量文字時，使用組合語言將不便於執行。而機器語言的寫法會因硬體而異，如果使用機器語言，想在其他製造商的電腦上運作程式時就必須重新改寫原始碼。

因此人類開始思考，使用語法上便於人類讀寫的程式語言編寫原始碼，再將其轉換為機器語言，這種更貼近人類的語言，就稱為**高階語言**（高級語言）。使用這些語言**寫下原始碼後，要轉換（移植）至其他硬體時**也更容易（圖 2-4）。

最近更出現支援**跨平台**的語言，可以讓程式直接在其他硬體與作業系統上執行。

圖 2-3 高階語言和低階語言

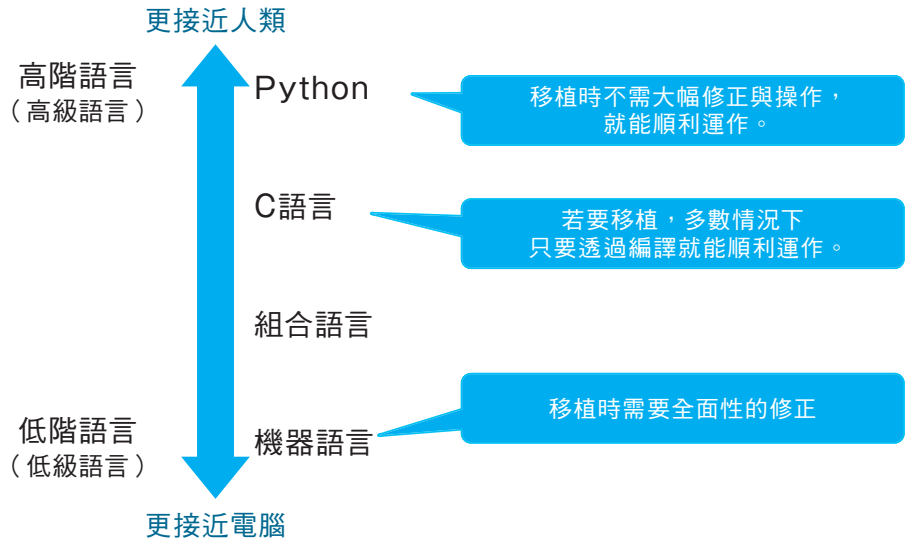
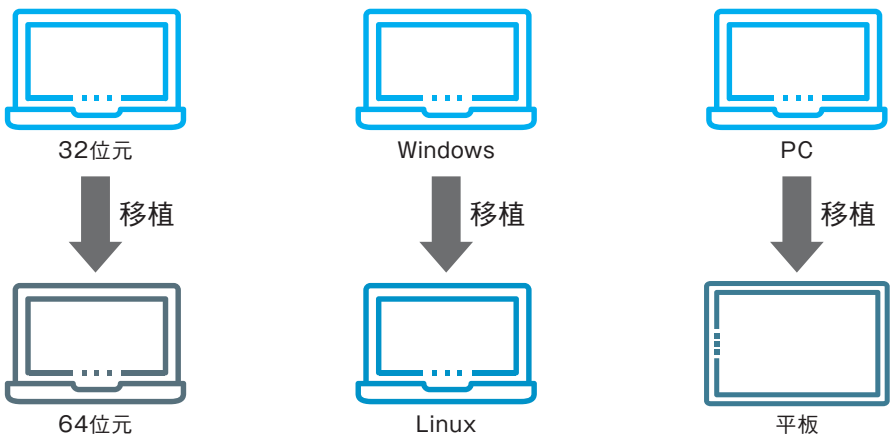


圖 2-4 移植範例



Point

- 雖然我們稱之為「低階語言（低級語言）」、「高階語言（高級語言）」，但並非代表語言的水準高低。
- 移植到其他硬體時，高階語言的轉換手續較少。

» 讓電腦記憶資料

資料的儲存場所——變數

在程式中指定值的儲存位置有兩種方法，就是**變數**與**常數**（圖 3-9）。

在數學方程式中使用 x 與 y 等符號取代想求得的值，這些符號就是變數。代表值是會改變的，而進行程式設計時，如果要將**執行時值會改變的各種資料存到記憶體**，也會使用變數。

必須多次進行複雜計算時，只要事前計算共同的部分並儲存計算結果，就能重複運用在計算過程中，提升效率，這時就要**預留過程中資料的儲存位置並予以命名**。這樣一來，只要指定該名稱，就能讀取所儲存的資料值。

上述的例子並不需要改寫資料值，不過進行重複的處理時，也可能會需要變更資料值。以計算九九乘法為例，比起寫下 1 到 9 的全部數字，在變數中儲存 1 到 9 的值，處理資料時才代入不同數字，程式也會更簡潔易讀（圖 3-10）。

儲存後就不能變更資料的常數

變數中儲存的資料是可以變更的，也就是說，值可能會一直改變，如果不看變數內容，就不知道儲存了什麼資料。而某個開發人員所儲存的值也可能會在其他處理中受到更改，這代表有些程式內容**會有較高的機會發生錯誤**。

相較於此，常數**的資料一旦儲存就不能改寫**（圖 3-11），常數與變數一樣，可以在多個地方使用相同的值，不需要重複書寫。使用常數時，在變更資料的瞬間就會發生錯誤，不只修正時容易找出問題，只看名稱也能知道是什麼樣的值。

圖 3-9 變數與常數

圖 3-10 在重複性的處理中使用常數

不使用變數

```
print("%d * %d = %d" % (1, 1, 1 * 1))
print("%d * %d = %d" % (1, 2, 1 * 2))
print("%d * %d = %d" % (1, 3, 1 * 3))
…
print("%d * %d = %d" % (9, 7, 9 * 7))
print("%d * %d = %d" % (9, 8, 9 * 8))
print("%d * %d = %d" % (9, 9, 9 * 9))
```

使用變數

```
for i in range(1, 10):  ←使用變數 i
      for j in range(1, 10):    ←使用變數 j
          print("%d * %d = %d" % (i, j, i * j))
```

【執行結果】

```
1 * 1 =  1
1 * 2 =  2
1 * 3 =  3
   ⋮
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
```

圖 3-11 常數的使用範例

```
PI = 3.14          ←圓周率
ROOT_DIR = '/'     ←系統的根目錄
```

Point

- 使用變數可以暫時存入值，之後也能變更內容。
- 使用常數時，存入值之後就無法變更，可以避免使用變數時錯誤變更資料值的情況。

» 依序處理資料

將堆積的資料依序處理

想到在陣列中存入與取出資料的不便，就會想找其他方法，能在處理時盡可能不移動資料。因此經常會使用一種方式，**是取出或存入資料時，只限定從開頭或是結尾單向操作**。

取出資料時從最後存入的資料開始，這種結構就稱為**堆疊**（Stack）（圖 3-38）。如同字面上的意思，堆疊就像是在箱子中堆疊物品，取出時必須從上方依序取出的方法，由於最後存入的資料會最先取出，因此也稱為「**LIFO**（Last In first Out）」。在 **4-16** 所介紹的「深度優先搜尋」中，經常會使用堆疊這種資料結構。

將堆疊運用於陣列時，會記憶陣列中最後元素的位置。這樣一來要放入新增資料或刪除資料時，就知道位置在哪裡，因此處理上相當迅速。

另外，將資料放入堆疊就稱為 Push，取出就稱為 Pop。

依資料存入順序依序處理

將存入資料依序取出的資料結構就稱為**佇列**（queue）（圖 3-39）。這個詞彙在英文中有「排隊」的意思，就像打撞球時擊球一樣，新增至這一側的資料，會從另一側被取出。最先放入的資料會最先被取出，因此也稱為「**FIFO**（First In First Out）」。在 **4-16** 所介紹的「廣度優先搜尋」中，經常會使用佇列。

佇列會記憶陣列中開頭的元素與最後元素，新增資料時會接續最後位置繼續登錄，刪除時則從開頭的元素開始取出。

另外，放入資料到佇列稱為 Enqueue，取出資料稱為 Dequeue。

圖 3-38 堆疊

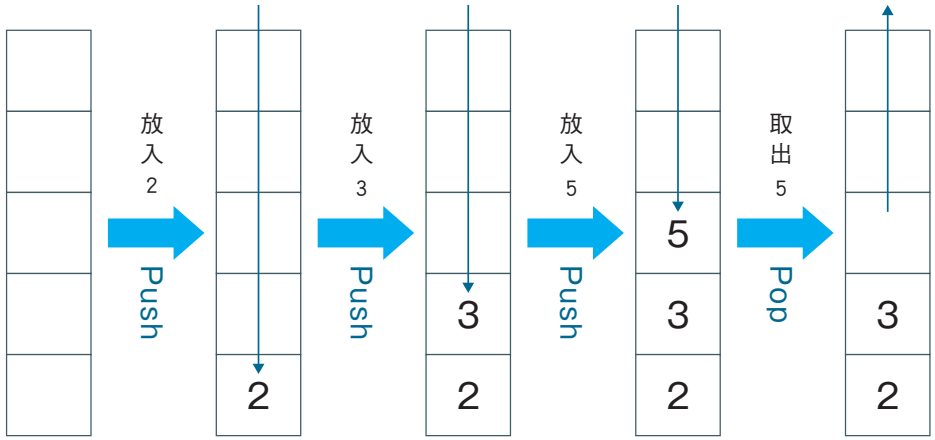
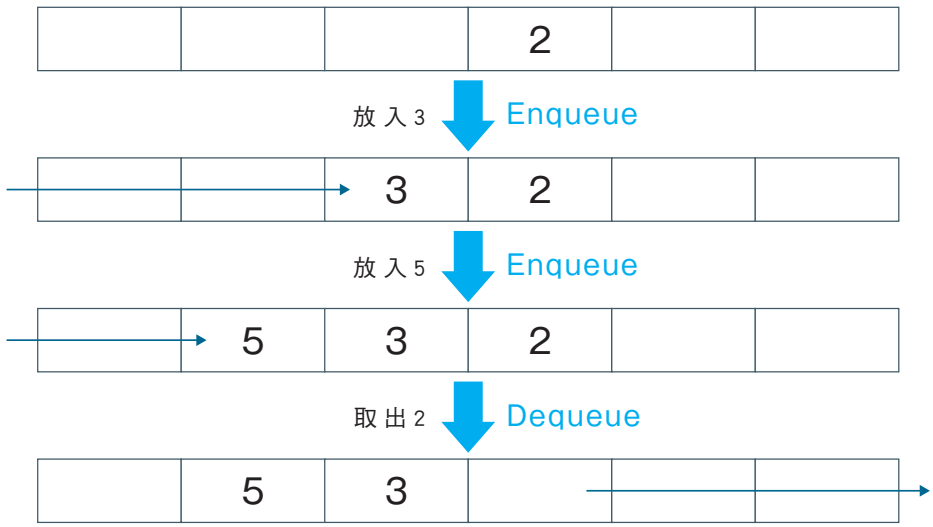


圖 3-39 佇列



Point

- 取出時從最後放入的資料開始，這種資料結構就稱為堆疊，在深度優先搜尋時經常使用。
- 取出時從最先放入的資料開始，這種資料結構就稱為佇列，在廣度優先搜尋時經常使用。

小試身手

實際執行程式

如果不實際輸入、操作，就難以理解輸入與執行的方法。此外，從實踐中得知處理大約需要花多少時間，以及發生錯誤時要如何應對也相當重要。

請一定要實際輸入原始碼，看看輸出後的結果如何！

這裡將要介紹如何在 Web 瀏覽器執行 Python 程式，會需要 Google 的帳戶，但不需要特別安裝其他軟體。

❶ 請連上「Google Colaboratory」（https://colab.research.google.com），選擇「新增筆記本」。

❷ 在輸入欄位輸入以下的原始碼。

❸ 按下輸入欄位左側的執行按鈕，執行輸入的原始碼。

```
for i in range(1, 51):
    if (i % 3 == 0) and (i % 5 == 0):
        print('FizzBuzz')
    elif i % 3 == 0:
        print('Fizz')
    elif i % 5 == 0:
        print('Buzz')
    else:
        print(i)
```

如果發生錯誤，請確認是否有輸入錯誤（縮排的位置不正確、少打了「:」、誤用全形字元等）。另外，縮排可以使用 2 個空白字元、4 個空白字元，以及 tab 鍵等，這幾種方式都沒問題，但格式必須一致。

在開發程式的過程中，發生錯誤以及輸入錯誤都是難以避免的，發生錯誤可以再調整，不必太過擔心。