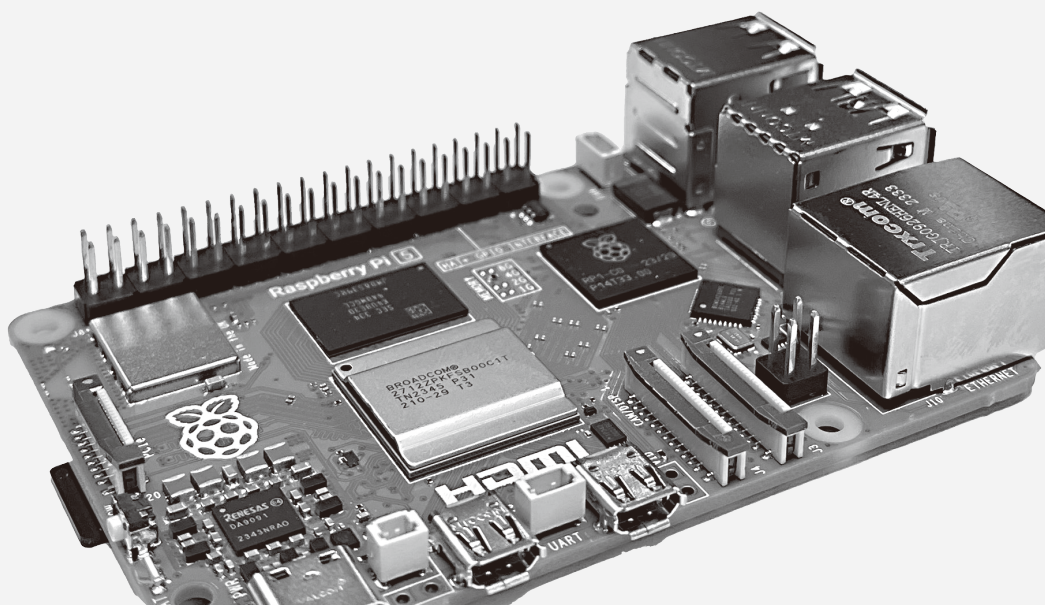


appendix



Python 程式設計基礎

- ▷ A-1 認識 Python 語言
- ▷ A-2 在樹莓派開發 Python 程式
- ▷ A-3 Python 變數與運算子
- ▷ A-4 Python 流程控制
- ▷ A-5 Python 函式與模組
- ▷ A-6 Python 串列與字串



A-1

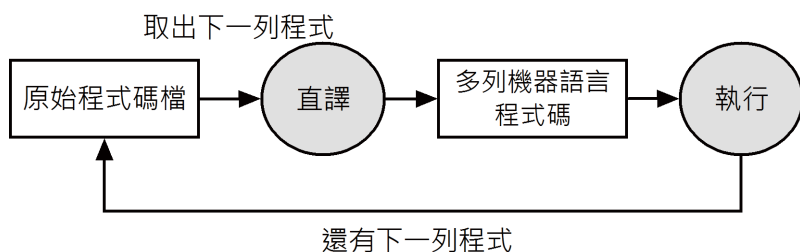
認識 Python 語言

Python 是 Guido Van Rossum 開發的一種通用用途 (General Purpose) 的程式語言，也是擁有優雅語法和高可讀性程式碼的程式語言，可以讓我們開發 GUI 視窗程式、Web 應用程式、系統管理工作、財務分析和大數據資料分析等各種不同的應用程式。

Python 語言有兩個版本：Python 2 和 Python 3，在本書是使用 Python 3 語言。

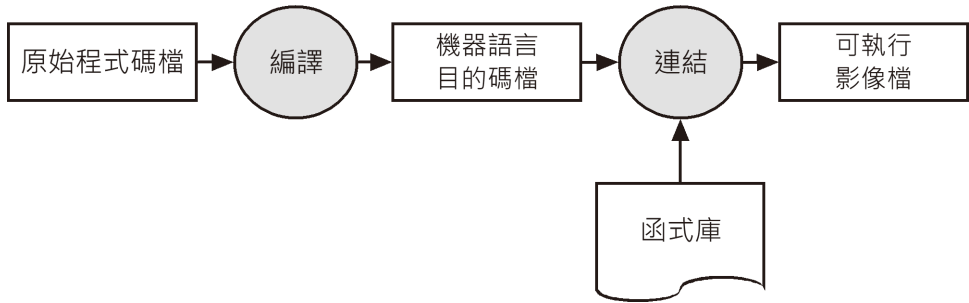
Python 是一種直譯語言

Python 是一種直譯語言 (Interpreted Language)，我們撰寫的 Python 程式是使用「直譯器」(Interpreters) 來執行，直譯器並不會輸出可執行檔案，而是一個指令一個動作，一系列轉換成機器語言後，馬上執行程式碼，如下圖所示：



因為直譯器是一列一系列轉換和執行，所以 Python 語言的執行效率比編譯語言 C 或 C++ 語言還要低，但是非常適合初學者學習程式設計。

C 或 C++ 語言是編譯語言 (Compiled Language)，我們建立的程式碼需要使用編譯器 (Compilers) 來檢查程式碼，如果沒有錯誤，就會翻譯成機器語言的目的碼檔案，如下圖所示：



上述原始程式碼檔案在編譯成機器語言的目的碼檔 (Object Code) 後，因為通常會參考外部程式碼，所以需要使用連結器 (Linker) 將程式使用的外部函式庫連結建立成「可執行映像檔」(Executable Image)，這就是在作業系統可執行的程式檔。

Python 是一種動態型別程式語言

Python 是一種動態型別 (Dynamically Typed) 語言，在 Python 程式碼宣告的變數並不需要預設宣告使用的資料型別，Python 直譯器會依據變數值來自動判斷其資料型別，如下所示：

```
a = 1
b = "Hello World!"
```

上述 Python 程式碼第 1 列的變數 a 是指定成整數 1，所以此變數的資料型別是整數；第 2 列的變數 b 是指定成字串，所以其資料型別是字串。

Python 是一種強型別程式語言

雖然，Python 變數不需要預設宣告使用的資料型別，但是 Python 語言是一種強型別 (Strongly Typed) 的程式語言，並不會自動轉換變數的資料型別，如下所示：

```
# 字串+整數  
v = "計算結果 = " + 100
```

上述 Python 程式碼使用「#」開頭是註解文字，這是一個字串加上整數的運算式，很多程式語言，例如：JavaScript 或 PHP 會自動將整數轉換成字串，但是 Python 語言並不允許自動型別轉換，我們一定需要自行轉換成同一型別，如下所示：

```
# 字串+字串  
v = "計算結果 = " + str(100)
```

上述 Python 程式碼的整數需要先呼叫 `str()` 函式轉換成字串後，才能和之前的字串進行字串連接。

A-2

在樹莓派開發 Python 程式

在 Raspberry Pi OS 可以使用 Thonny、VS Code 或 Geany 建立並執行 Python 程式，在本章的 Python 程式範例可以選擇使用 Geany 和 Thonny 來測試執行（這是 Raspberry Pi OS 預設安裝的 Python 開發環境）。

A-2-1

使用 Geany 建立和執行 Python 程式

Raspberry Pi OS 作業系統內建 Python 3 語言，我們可以馬上使用 Geany 在樹莓派進行 Python 應用程式的開發。

新增 Python 程式檔與輸入程式碼

請執行「選單/軟體開發/Geany」命令啟動 Geany 後，執行「檔案/新增」命令新增程式檔案，然後執行「檔案/另存新檔」命令將新增檔案儲存成 `appa-2-1.py`（別忘了加上副檔名 `.py`），如下圖所示：



在 `appa-2-1.py` 標籤的編輯視窗輸入 Python 程式碼，如下所示：

```
print("Hello World!")
```

上述程式碼在 Geany 可以在 `print()` 函式輸入中文字串，關於中文輸入法的說明，詳見第 4-6 節，在完成 Python 程式碼輸入後，請執行「檔案/儲存」命令儲存 Python 程式。

在 Geany 執行 Python 程式

儲存 Python 程式後，我們就可以在 Geany 執行 Python 程式（執行「檔案/開啟」命令可以開啟存在的 Python 程式），請執行「組建/Execute」命令或按 **F5** 鍵，如下圖所示：



可以開啟終端機視窗看到 Python 程式的執行結果，請按 **Enter** 鍵繼續，如下圖所示：

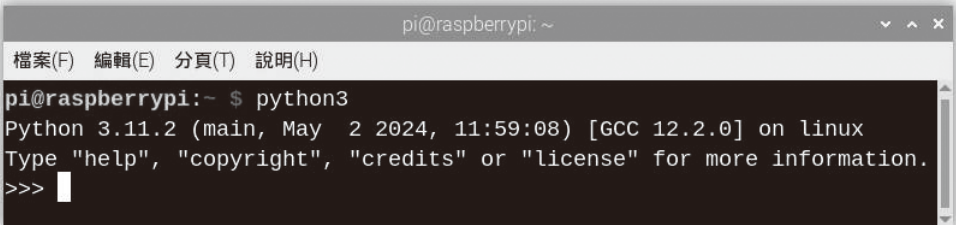


A-2-2

在終端機啟動 Python Shell

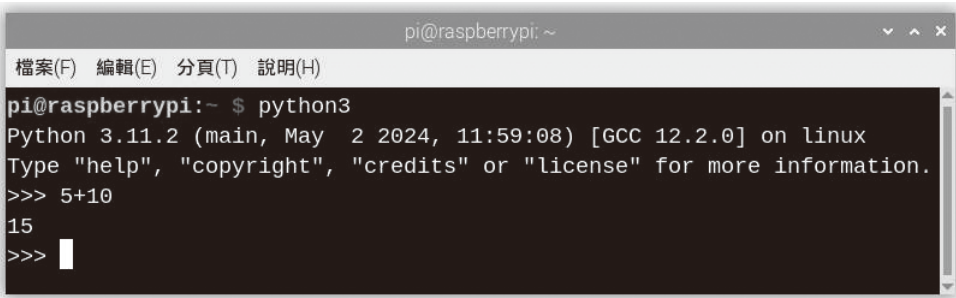
Python 支援互動環境的 Python Shell，我們可以在終端機將其啟動。請使用 `python3` 指令（或 `python` 指令）來啟動 Python Shell，如下所示：

```
$ python3 
```



```
pi@raspberrypi: ~  
檔案(F) 編輯(E) 分頁(T) 說明(H)  
pi@raspberrypi:~ $ python3  
Python 3.11.2 (main, May 2 2024, 11:59:08) [GCC 12.2.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 
```

因為 Python 是直譯語言，在 Python Shell 提供互動模式，可以讓我們在「>>>」提示文字輸入 Python 程式碼來馬上測試執行，例如：輸入 `5+10`，按 鍵，可以馬上看到執行結果 `15`，如下圖所示：



```
pi@raspberrypi: ~  
檔案(F) 編輯(E) 分頁(T) 說明(H)  
pi@raspberrypi:~ $ python3  
Python 3.11.2 (main, May 2 2024, 11:59:08) [GCC 12.2.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 5+10  
15  
>>> 
```

不只如此，我們還可以定義變數 `num = 10`，然後執行 `print()` 函式來顯示變數值，如下所示：

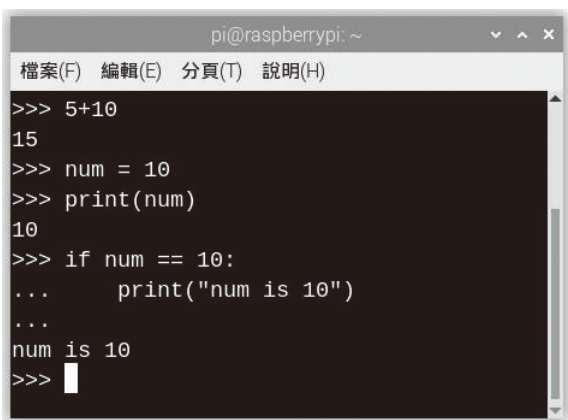
```
num = 10  
print(num)
```

A terminal window titled 'pi@raspberrypi: ~' with a menu bar containing '檔案(F)', '編輯(E)', '分頁(T)', and '說明(H)'. The terminal shows the following commands and output:

```
>>> 5+10
15
>>> num = 10
>>> print(num)
10
>>> 
```

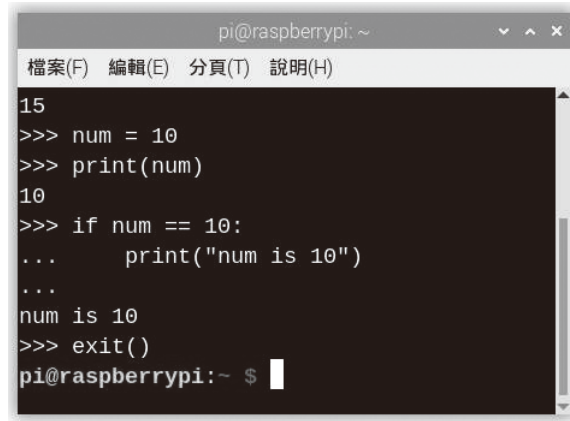
同理，我們可以測試 if 條件，在輸入 if num >= 10: 後，按 鍵，然後縮排 4 個空白字元來輸入 print() 函式，按 2 次 鍵，可以看到執行結果，如下圖所示：

```
if num == 10:
    print("num is 10")
```

A terminal window titled 'pi@raspberrypi: ~' with a menu bar containing '檔案(F)', '編輯(E)', '分頁(T)', and '說明(H)'. The terminal shows the following commands and output:

```
>>> 5+10
15
>>> num = 10
>>> print(num)
10
>>> if num == 10:
...     print("num is 10")
...
num is 10
>>> 
```

結束 Python Shell，請輸入 exit() 後，按 鍵，如下圖所示：



```

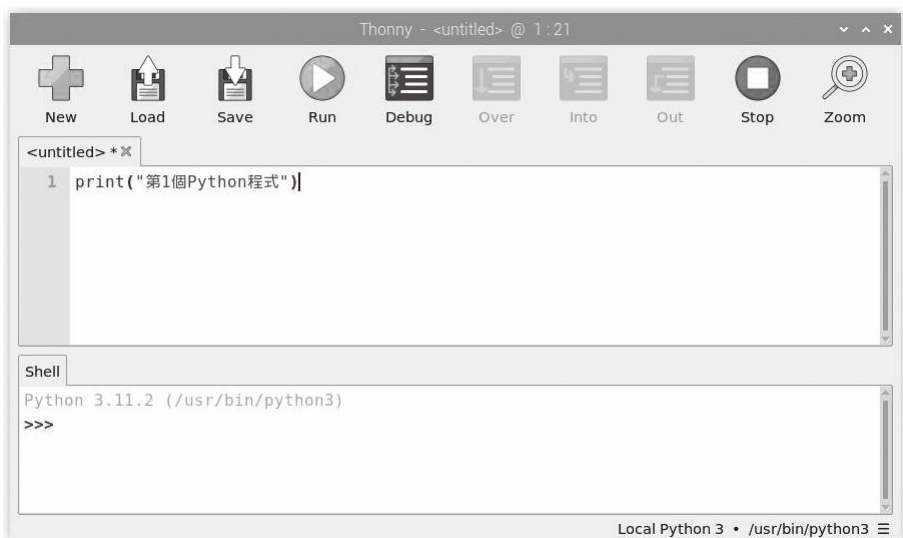
pi@raspberrypi: ~
檔案(F) 編輯(E) 分頁(T) 說明(H)
15
>>> num = 10
>>> print(num)
10
>>> if num == 10:
...     print("num is 10")
...
num is 10
>>> exit()
pi@raspberrypi:~ $

```

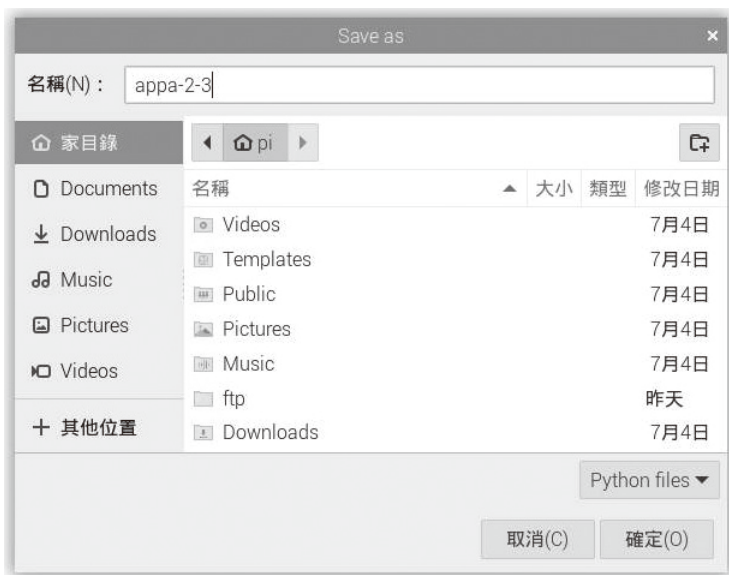
A-2-3 Thonny

Thonny 是一個 Python 語言的整合開發環境，請在 Raspberry Pi OS 執行「選單/軟體開發/Thonny」命令啟動 Thonny。在樹莓派預設是使用 Thonny 精簡介面，請在編輯視窗輸入下列程式碼，如下所示：

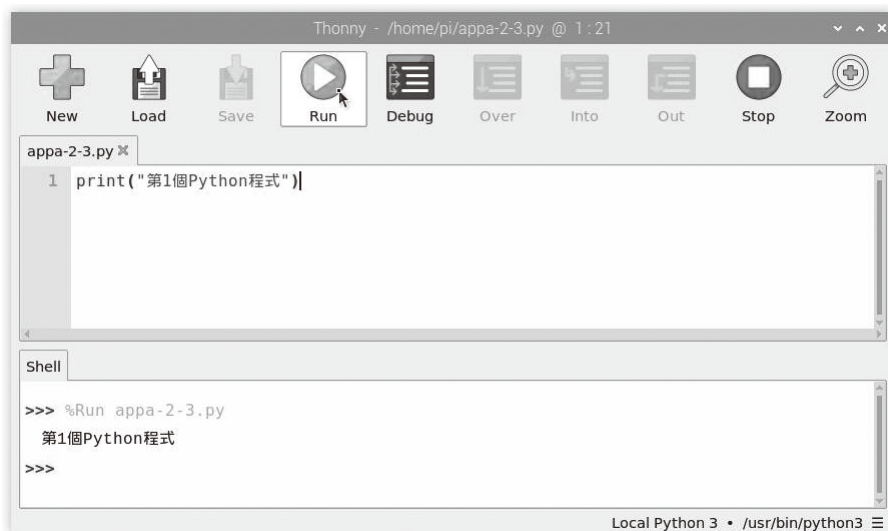
```
print("第1個Python程式!")
```



然後按上方工具列的 **Save** 鈕 (**Load** 鈕可以開啟 Python 程式)，輸入 appa-2-3，按 **Yes** 鈕，再按**確定**鈕儲存成 appa-2-3.py，如下圖所示：



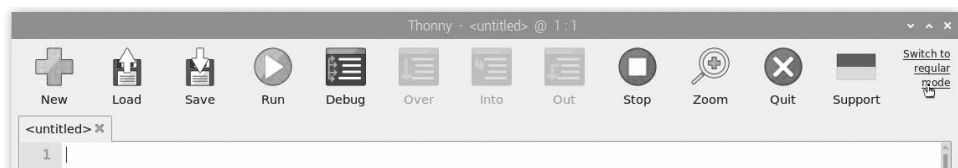
請按 **Run** 鈕執行 Python 程式 (**Stop** 鈕停止執行)，可以在下方框看到執行結果，而下方框就是 Python Shell，如下圖所示：



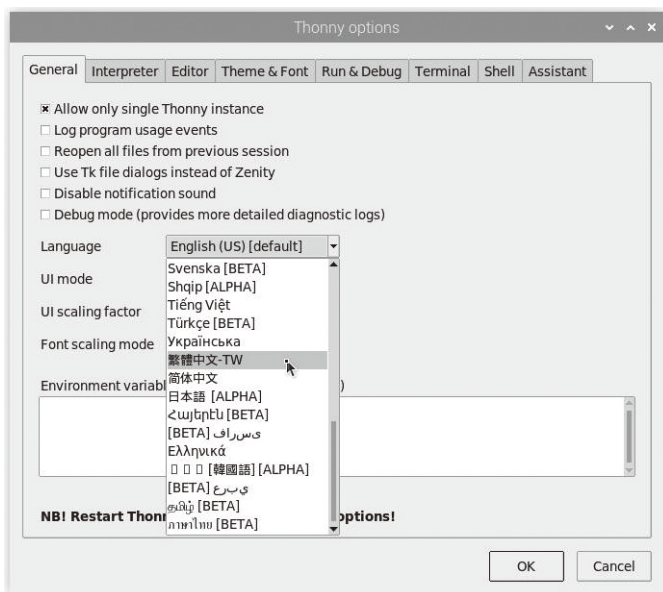
A-2-4 在 Thonny 使用 Python 虛擬環境

Thonny 預設是英文介面，並使用樹莓派 Raspberry Pi OS 預設安裝的 Python 開發環境，如果要使用第 6 章建立的 Python 虛擬環境 ai，在 Thonny 需要設定使用 Python 虛擬環境的 Python 直譯器，其設定步驟如下所示：

Step 1 請執行「選單/軟體開發/Thonny」命令啟動 Thonny 後，點選上方工作列最後的 **Switch to regular mode** 切換成正常模式（目前是精簡模式）。



Step 2 重新啟動 Thonny，執行「Tools/Options...」命令，並在 **General** 標籤的 **Language** 欄選**繁體中文 -TW**，按 **OK** 鈕更改介面語言。



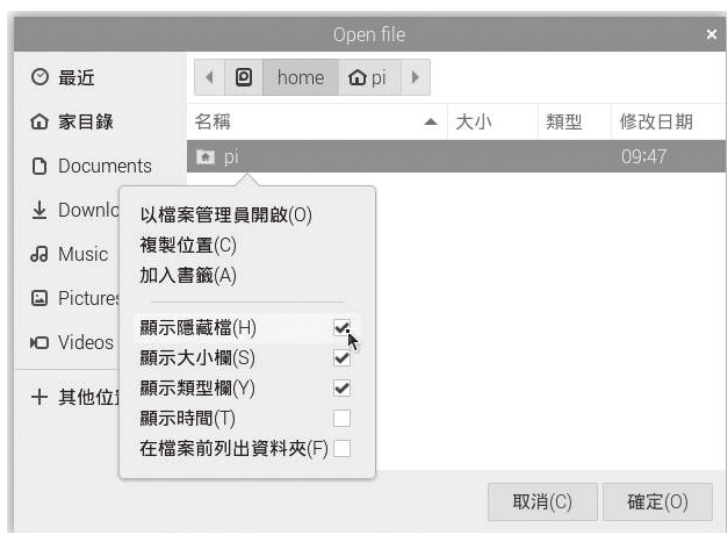
Step
3

請再次重啟 Thonny，執行「工具/選項...」命令，選**直譯器**標籤，在上方選**本地端的 Python 3**，然後在下方 **Python 可執行檔**欄（「/usr/bin/python3」就是預設 Python 開發環境），按欄位後方的 ... 鈕。

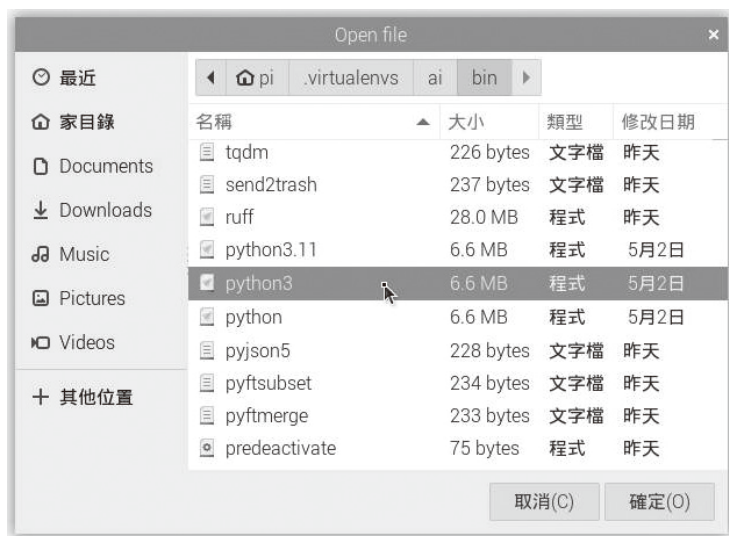


Step
4

請從 + 其他位置切換至「home」目錄，並在 **pi** 目錄上，執行**右鍵顯示快顯功能表的選單**，勾選**顯示隱藏檔**。



Step 5 然後切換至「/home/pi/.virtualenvs/ai/bin」目錄，選 **python3**，按**確定**鈕。



Step 6 確認 Python 可執行檔案的路徑改為 Python 虛擬環境 ai 之後，按**確認**鈕。



現在，Thonny 已經改成中文介面，並且在下方「互動環境」框中，可以看到啟動的是虛擬環境 ai 的 Python 直譯器，如下圖所示：



A-3

Python 變數與運算子

「變數」(Variables) 是儲存程式執行期間的暫存資料，我們可以使用運算子和變數建立運算式來執行運算，以便得到程式執行結果。

A-3-1

使用 Python 變數

變數可以儲存程式執行時的暫存資料，Python 變數並不需要宣告型別，我們只需指定變數值，就可以建立變數。**請注意！**Python 變數在使用前必須指定初值 (Python 程式：appa-3-1.py)，如下所示：

```
grade = 76
height = 175.5
weight = 75.5
```

上述程式碼建立整數變數 `grade`，因為初值是整數，同理，變數 `height` 和 `weight` 是浮點數（因為初值 175.5 有小數點），然後我們可以馬上使用 3 個 `print()` 函式顯示這 3 個變數值，如下所示：

```
print("成績 = " + str(grade))
print("身高 = " + str(height))
print("體重 = " + str(weight))
```

上述 `print()` 函式使用 `str()` 函式將整數和浮點數變數轉換成字串，「+」號是字串連接運算子，在連接字串字面值和轉換成字串的變數值後，就可以輸出 3 個變數的值。

A-3-2 Python 的運算子

Python 提供完整算術 (Arithmetic)、指定 (Assignment)、位元 (Bitwise)、關係 (Relational) 和邏輯 (Logical) 運算子。Python 語言運算子預設的優先順序（愈上面愈優先），如下表所示：

運算子	說明
()	括號運算子
**	指數運算子
~	位元運算子 NOT
+、-	正號、負號
＊、／、／／、％	算術運算子的乘法、除法、整數除法和餘數
+、-	算術運算子加法和減法
<<、>>	位元運算子左移和右移
&	位元運算子 AND

→ 接下頁

運算子	說明
\wedge	位元運算子 XOR
$ $	位元運算子 OR
<code>in</code> 、 <code>not in</code> 、 <code>is</code> 、 <code>is not</code> 、 <code><</code> 、 <code><=</code> 、 <code>></code> 、 <code>>=</code> 、 <code>!=</code> 、 <code>==</code>	成員、識別和關係運算子小於、小於等於、大於、大於等於、不等於和等於
<code>not</code>	邏輯運算子 NOT
<code>and</code>	邏輯運算子 AND
<code>or</code>	邏輯運算子 OR

當 Python 運算式中的多個運算子擁有相同的優先順序時，如下所示：

```
3 + 4 - 2
```

上述運算式的「+」和「-」運算子擁有相同的優先順序，此時的運算順序是從左至右依序的進行運算，即先運算 $3+4=7$ ，然後再運算 $7-2=5$ ，如下圖所示：

$$\begin{array}{l}
 \xrightarrow{\hspace{1cm}} \\
 \boxed{3 + 4} - 2 \\
 7 - 2 \\
 5
 \end{array}$$

不過，Python 語言的多重指定運算式是一個例外，如下所示：

```
a = b = c = 25
```

上述多重指定運算式是從右至左，先執行 $c = 25$ ，然後才是 $b = c$ 和 $a = b$ （所以變數 a 、 b 和 c 的值都是 25），如下圖所示：

$$\begin{array}{l}
 \xleftarrow{\hspace{1cm}} \\
 a = b = \boxed{c = 25} \\
 a = \boxed{b = c} \\
 a = b
 \end{array}$$

A-4

Python 流程控制

Python 的流程控制可以配合條件運算式的條件來執行不同程式區塊 (Blocks)，或重複執行指定區塊的程式碼，流程控制主要分為兩種，如下所示：

- **條件控制**：條件控制是選擇題，分為單選、二選一或多選一，依照條件運算式的結果決定執行哪一個程式區塊的程式碼。
- **迴圈控制**：迴圈控制是重複執行程式區塊的程式碼，擁有一個結束條件可以結束迴圈的執行。

Python 的程式區塊是程式碼縮排相同數量的空白字元，一般是 4 個空白字元，換句話說，相同縮排的程式碼屬於同一個程式區塊。

A-4-1

條件控制

Python 條件控制敘述是使用條件運算式，配合程式區塊建立的決策敘述，可以分為 3 種：單選 (if)、二選一 (if/else) 或多選一 (if/elif/else)。

if 單選條件敘述

if 條件敘述是一種是否執行的單選題，只是決定是否執行程式區塊內的程式碼，如果條件運算式的結果為 True，就執行程式區塊的程式碼。Python 語言的程式區塊是相同縮排的多列程式碼，習慣用法是縮排 4 個空白字元。

例如：判斷氣溫決定是否加件外套的 if 條件敘述 (Python 程式：appa-4-1.py)，如下所示：

```
t = int(input("請輸入氣溫 => "))
if t < 20:
    print("加件外套!")
print("今天氣溫 = " + str(t))
```

上述程式碼使用 input() 函式輸入字串，然後呼叫 int() 函式轉換成整數值，當 if 條件敘述的條件成立，才會執行縮排的程式敘述。更進一步，我們可以活用邏輯運算式，當氣溫在 20~22 度之間時，顯示「加一件薄外套！」訊息文字，如下所示：

```
if t >= 20 and t <= 22:
    print("加一件薄外套!")
```

if/else 二選一條件敘述

單純 if 條件只能選擇執行或不執程式區塊的單選題，但如果是排它情況的兩個執行區塊，只能二選一，我們可以加上 else 關鍵字，依條件決定執行哪一個程式區塊。

例如：學生成績以 60 分區分是否及格的 if/else 條件敘述（Python 程式：appa-4-1a.py），如下所示：

```
s = int(input("請輸入成績 => "))
if s >= 60:
    print("成績及格!")
else:
    print("成績不及格!")
```

上述程式碼因為成績有排它性，60 分以上為及格分數，60 分以下為不及格。

if/elif/else 多選一條件敘述

Python 多選一條件敘述是 if/else 條件的擴充，在之中新增 elif 關鍵字來新增一個條件判斷，就可以建立多選一條件敘述。在輸入時，別忘了輸入在條件運算式和 else 之後的「:」冒號。

例如：輸入年齡值來判斷不同範圍的年齡，小於 13 歲是兒童，小於 20 歲是青少年，大於等於 20 歲是成年人，因為條件不只一個，所以需要使用多選一條件敘述（Python 程式：appa-4-1b.py），如下所示：

```
a = int(input("請輸入年齡 => "))
if a < 13:
    print("兒童")
elif a < 20:
    print("青少年")
else:
    print("成年人")
```

上述 if/elif/else 多選一條件敘述從上而下如同階梯一般，一次判斷一個 if 條件，如果為 True，就執行程式區塊，並且結束整個多選一條件敘述；如果為 False，就進行下一次判斷。

A-4-2

迴圈控制

Python 語言迴圈控制提供 for 計數迴圈（Counting Loop），和 while 條件迴圈。

for 計數迴圈

在 for 迴圈的程式敘述中擁有計數器變數，計數器可以每次增加或減少一個值，直到迴圈結束條件成立為止。基本上，如果已經知道需重複執行幾次，就可以使用 for 計數迴圈來重複執行程式區塊。

例如：在輸入最大值後，可以計算出 1 加至最大值的總和（Python 程式：appa-4-2.py），如下所示：

```
m = int(input("請輸入最大值 =>"))
s = 0
for i in range(1, m + 1):
    s = s + i
print("總和 = " + str(s))
```

上述 for 計數迴圈需要使用內建 range() 函式，此函式的範圍不包含第 2 個參數本身，所以，1~m 範圍是 range(1, m + 1)。

for 迴圈與 range() 函式

基本上，for 計數迴圈一定需要使用 range() 函式來產生指定範圍的計數值，這是 Python 內建函式，可以有 1、2 和 3 個參數，如下所示：

- 擁有 1 個參數的 range() 函式：此參數是終止值（並不包含終止值），預設的起始值是 0，如下表所示：

range() 函式	整數值範圍
range(5)	0~4
range(10)	0~9
range(11)	0~10

例如：建立計數迴圈顯示值 0~4，如下所示：

```
for i in range(5):
    print("range(5)的值 = " + str(i))
```

- 擁有 2 個參數的 range() 函式：第 1 參數是起始值，第 2 個參數是終止值（並不包含終止值），如下表所示：

range() 函式	整數值範圍
range(1, 5)	1~4
range(1, 10)	1~9
range(1, 11)	1~10

例如：建立計數迴圈顯示值 1~4，如下所示：

```
for i in range(1, 5):
    print("range(1,5)的值 = " + str(i))
```

- 擁有 3 個參數的 range() 函式：第 1 參數是起始值，第 2 個參數是終止值（不含終止值），第 3 個參數是間隔值，如下表所示：

range() 函式	整數值範圍
range(1, 11, 2)	1、3、5、7、9
range(1, 11, 3)	1、4、7、10
range(1, 11, 4)	1、5、9
range(0, -10, -1)	0、-1、-2、-3、-4...-7、-8、-9
range(0, -10, -2)	0、-2、-4、-6、-8

例如：建立計數迴圈從 1~10 顯示奇數值，如下所示：

```
for i in range(1, 11, 2):
    print("range(1,11,2)的值 = " + str(i))
```

while 條件迴圈

while 迴圈敘述需要在程式區塊自行處理計數器變數的增減，迴圈是在程式區塊開頭檢查條件，條件成立才允許進入迴圈執行。例如：使用 while 迴圈來計算階層函數值（Python 程式：appa-4-2a.py），如下所示：

```
m = int(input("請輸入階層數 =>"))
r = 1
n = 1
while n <= m:
    r = r * n
    n = n + 1
print("階層值! = " + str(r))
```

上述 while 迴圈的執行次數是直到條件 False 為止，假設 m 輸入 5，就是計算 5! 的值，變數 n 是計數器變數。如果符合 $n \leq 5$ 條件，就進入迴圈執行程式區塊，迴圈結束條件是 $n > 5$ ，在程式區塊不要忘了更新計數器變數 $n = n + 1$ 。

A-5

Python 函式與模組

Python「函式」(Functions) 是一個獨立程式單元，可以將大工作分割成一個個小型工作，我們可以重複使用之前建立的函式或直接呼叫 Python 語言的內建函式。

A-5-1

函式

函式名稱如同變數是一種識別字，其命名方式和變數相同，程式設計者需要自行命名。在函式的程式區塊之中，可以使用 return 關鍵字回傳函式值，並結束函式執行。函式的參數 (Parameters) 列是函式的使用介面，在呼叫時，我們需要傳入對應的引數 (Arguments)。

定義函式

在 Python 程式以 `def` 關鍵字來建立沒有參數列和回傳值的 `print_msg()` 函式 (Python 程式：appa-5-1.py)，如下所示：

```
def print_msg():  
    print("歡迎學習Python程式設計!")
```

上述函式名稱是 `print_msg`，在名稱後的括號中定義傳入的參數列；如果函式沒有參數，就是空括號，在空括號後不要忘了輸入「:」冒號。

Python 函式如果有回傳值，我們需要使用 `return` 關鍵字來回傳值。例如：判斷參數值是否在指定範圍的 `is_valid_num()` 函式，如下所示：

```
def is_valid_num(no):  
    if no >= 0 and no <= 200.0:  
        return True  
    else:  
        return False
```

上述函式使用 2 個 `return` 關鍵字來回傳值，回傳 `True` 表示合法，`False` 為不合法。再來看一個執行運算的 `convert_to_f()` 函式，如下所示：

```
def convert_to_f(c):  
    f = (9.0 * c) / 5.0 + 32.0  
    return f
```

上述函式使用 `return` 關鍵字回傳函式的執行結果，即運算式的運算結果。

函式呼叫

在 Python 程式碼呼叫函式是使用函式名稱加上括號中的引數列。因為 `print_msg()` 函式沒有回傳值和參數列，所以呼叫函式只需使用函式名稱加上空括號，如下所示：

```
print_msg()
```

函式如果擁有回傳值，在呼叫時可以使用指定敘述來取得回傳值，如下所示：

```
f = convert_to_f(c)
```

上述程式碼的變數 `f` 可以取得 `convert_to_f()` 函式的回傳值。如果函式回傳值是 `True` 或 `False`，例如：`is_valid_num()` 函式，我們可以在 `if` 條件敘述呼叫函式作為判斷條件，如下所示：

```
if is_valid_num(c):  
    print("合法!")  
else:  
    print("不合法")
```

上述條件使用函式回傳值作為判斷條件，可以顯示數值是否合法。

A-5-2 使用 Python 模組

Python 語言之所以擁有強大的功能，這都是因為有眾多標準和網路上現成模組來擴充程式功能，我們可以匯入 Python 模組來直接使用模組提供的函式，而不用自己撰寫相關函式。

匯入模組

我們可以使用 `import` 關鍵字匯入模組，例如：匯入名為 `random` 的模組，然後直接呼叫此模組的函式來產生亂數值（Python 程式：appa-5-2.py），如下所示：

```
import random
```

上述程式碼匯入名為 `random` 的模組後，我們就可以呼叫模組的 `randint()` 函式，馬上產生指定範圍之間的整數亂數值，如下所示：

```
target = random.randint(1, 100)
```

上述程式碼可以產生 1~100 之間的整數亂數值。

模組的別名

在 Python 程式檔匯入模組，除了使用模組名稱來呼叫函式，也可以使用 `as` 關鍵字替模組取一個別名，然後改用別名來呼叫函式（Python 程式：appa-5-2a.py），如下所示：

```
import random as R

target = R.randint(1, 100)
```

上述程式碼在匯入 `random` 模組時，使用 `as` 關鍵字取了別名 `R`，所以，我們可以使用別名 `R` 來呼叫 `randint()` 函式。

匯入模組的部分名稱

當我們使用 `import` 關鍵字匯入模組時，預設是匯入全部的內容，但在實務上，我們可能只需使用到模組的 1 或 2 個函式或物件，此時可以使用 `from/import` 程式敘述來匯入模組的部分名稱，例如：在第 6-6-2 節匯入 `openai` 模組的 `OpenAI` 類別，如下所示：

```
from openai import OpenAI
```

上述程式碼只匯入 `openai` 模組的 `OpenAI` 類別。由於使用 `from/import` 程式敘述匯入的變數、函式或類別是匯入到目前的程式檔案，成為目前程式檔案的範圍，因此在使用時不需要使用模組名稱來指定所屬的模組，直接使用 `OpenAI()` 即可（而不是 `openai.OpenAI()`），如下所示：

```
api_key = "<API-KEY>"
client = OpenAI(api_key=api_key)
```

A-6

Python 串列與字串

Python 語言的字串 (Strings) 並不能更改其內容，所有字串的變更都是建立一個全新的字串。而串列 (Lists) 類似其他程式語言的陣列 (Arrays)，這是一種有序的資料結構，中文譯名有清單、串列和列表等。

A-6-1

字串

Python 語言的字串 (Strings) 是使用「`'`」單引號或「`"`」雙引號括起的一序列 Unicode 字元。

建立字串

我們可以使用指定敘述指定變數值為一個字串，如下所示：

```
str1 = "學習Python語言程式設計"  
str2 = 'Hello World!'  
ch1 = "A"
```

上述前 2 列程式碼是建立字串，最後 1 列是字元（只有 1 個字元的字串）。此外，我們也可以使用物件方式建立字串（Python 資料型別都是物件），如下所示：

```
name1 = str()  
name2 = str("陳會安")
```

上述第 1 列程式碼是建立空字串，第 2 列建立內容為 "陳會安" 的字串物件。

補充說明

在 Python 語言中，已有名為 `str()` 的內建函式，若變數名稱也命名為 `str`，當 Python 程式同時使用 `str()` 函式時，直譯器會認為 `str` 式指變數，而不是 `str()` 函式，所以會產生錯誤。在替變數命名時，變數名稱建議不要取為 `str`；同理，因為有內建的字元函式 `chr()`，所以變數名稱也建議不要命名為 `chr`。

輸出字串內容

Python 可以使用 `print()` 函式輸出字串內容，如下所示：

```
print("str1 = " + str1)  
print("str2 = " + str2)
```

上述 `print()` 函式使用字串連接運算式輸出字串變數，因為輸出的是字串，不需要呼叫 `str()` 函式轉換成字串型別。同理，我們也可以直接輸出字串變數，如下所示：

```
print(str1)
print(str2)
```

走訪字串的每一個字元

字串是一序列的 Unicode 字元，我們可以使用 `for` 迴圈走訪顯示每一個字元，正式的說法是迭代 (Iteration)，如下所示：

```
str3 = 'Hello'
for e in str3:
    print(e)
```

上述 `for` 迴圈中，位在 `in` 關鍵字後的是字串 `str3`，每執行一次 `for` 迴圈，就從字串第 1 個字元開始，取得一個字元指定給變數 `e`，並且移至下一個字元，直到取出字串的最後 1 個字元為止，其操作如同從字串的第 1 個字元走訪至最後 1 個字元。

使用索引運算子取得字元

Python 字串可以使用「`[]`」索引運算子取出指定位置的字元，索引值是從 0 開始 (此外，索引值也可以是負值)，如下所示：

```
str1 = 'Hello'
print(str1[0])    # H
print(str1[1])    # e
print(str1[-1])   # o
print(str1[-2])   # l
```

上述程式碼依序顯示字串 str1 的第 1 和第 2 個字元，而 -1 是最後 1 個，-2 是倒數第 2 個。

連接運算子

算術運算子的「+」加法使用在字串時是連接運算子，可以連接字串，我們已經使用大量連接運算子在 print() 函式建立輸出內容，如下所示：

```
str2 = " World!"  
str3 = str1 + str2
```

上述程式碼使用連接運算子連接字串 str1 和 str2。

重複運算子

算術運算子的「*」乘法使用在字串時是重複運算子，可以重複第 2 個運算元次數的字串，如下所示：

```
str1 = 'Hello'  
str4 = str1 * 3
```

上述程式碼可以重複 3 次 str1 的內容。

成員運算子

Python 字串可以使用成員運算子 in 和 not in 來檢查字串是否存在其他字串之中，如下所示：

```
str5 = "Welcome!"  
print("come" in str5)      # True  
print("come" not in str5)  # False
```

上述程式碼可以檢查字串 "come" 是否存在 str5 字串之中。

使用關係運算子進行字串比較

如同整數和浮點數，字串也一樣可以使用關係運算子進行 2 個字串的比較，如下所示：

```
"green" == "glow"  
"green" != "glow"  
"green" > "glow"  
"green" >= "glow"  
"green" < "glow"  
"green" <= "glow"
```

切割運算子

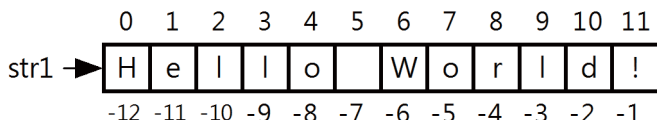
Python 語言不只可以使用「[]」索引運算子取出指定索引位置的字元，更進一步，索引運算子還可以是「切割運算子」(Slicing Operator)，可以從原始字串切割出所需的子字串，其語法如下所示：

```
str1[start:end]
```

上述 [] 語法中使用「:」冒號分隔 2 個索引位置，可以取回字串 str1 從索引位置 start 到 end-1 之間的字串。如果沒有 start，就是從 0 開始；沒有 end 就是到字串的最後 1 個字元。例如：本節範例字串 str1 的字串內容，如下所示：

```
str1 = 'Hello World!'
```

上述字串的索引位置值可以是正值，也可以是負值，如下圖所示：



現在，就讓我們看一些切割字串的範例，如下表所示：

切割字串	索引值範圍	取出的子字串
str1[1:3]	1~2	"el"
str1[1:5]	1~4	"ello"
str1[:7]	0~6	"Hello W"
str1[4:]	4~11	"o World!"
str1[1:-1]	1~(-2)	"ello World"
str1[6:-2]	6~(-3)	"Worl"

字元函式

Python 內建字元函式用來處理 ASCII 碼，其說明如下表所示：

字元函式	說明
ord()	回傳參數字元的 ASCII 碼
chr()	回傳參數 ASCII 碼的字元

字串函式

Python 內建字串函式可以取得字串長度、字串中的最大和最小字元，其說明如下表所示：

字串函式	說明
len()	回傳參數字串的長度
max()	回傳參數字串的最大字元
min()	回傳參數字串的最小字元

字串方法

Python 字串物件提供數種方法來檢查字串內容、搜尋子字串和轉換字串內容。因為是字串方法，我們需要使用物件變數加上「.」句號來呼叫，如下所示：

```
str1 = 'welcome to python'
print(str1.islower())
```

上述程式碼建立字串 `str1` 後，呼叫 `islower()` 方法檢查內容是否都是小寫英文字母。此外，字串方法不只可以使用在字串變數，也可以直接在字串字面值來呼叫（因為都是物件），如下所示：

```
print("2024".isdigit())
```

字串物件提供的相關方法，其說明如下表所示：

字串方法	說明
<code>isalnum()</code>	如果字串內容是英文字母或數字，回傳 <code>True</code> ，否則為 <code>False</code>
<code>isalpha()</code>	如果字串內容只有英文字母，回傳 <code>True</code> ，否則為 <code>False</code>
<code>isdigit()</code>	如果字串內容只有數字，回傳 <code>True</code> ，否則為 <code>False</code>
<code>isidentifier()</code>	如果字串內容是合法的識別字，回傳 <code>True</code> ，否則為 <code>False</code>
<code>islower()</code>	如果字串內容是小寫英文字母，回傳 <code>True</code> ，否則為 <code>False</code>
<code>isupper()</code>	如果字串內容是大寫英文字母，回傳 <code>True</code> ，否則為 <code>False</code>
<code>isspace()</code>	如果字串內容是空白字元，回傳 <code>True</code> ，否則為 <code>False</code>
<code>endswith(str1)</code>	如果字串內容是以參數字串 <code>str1</code> 結尾，回傳 <code>True</code> ，否則為 <code>False</code>
<code>startswith(str1)</code>	如果字串內容是以參數字串 <code>str1</code> 開頭，回傳 <code>True</code> ，否則為 <code>False</code>
<code>count(str1)</code>	回傳字串內容出現多少次參數字串 <code>str1</code> 的整數值
<code>find(str1)</code>	回傳字串內容出現參數字串 <code>str1</code> 的最小索引位置值，若沒有找到則回傳 <code>-1</code>
<code>rfind(str1)</code>	回傳字串內容出現參數字串 <code>str1</code> 的最大索引位置值，若沒有找到則回傳 <code>-1</code>
<code>capitalize()</code>	回傳只有第 1 個英文字母為大寫的字串
<code>lower()</code>	回傳小寫英文字母的字串
<code>upper()</code>	回傳大寫英文字母的字串
<code>title()</code>	回傳字串中每 1 個英文字的第 1 個英文字母大寫的字串
<code>swapcase()</code>	回傳英文字母大寫變小寫，小寫變大寫的字串
<code>replace(old, new)</code>	將字串中參數 <code>old</code> 的舊子字串取代成參數 <code>new</code> 的新字串

A-6-2

串列

Python 串列 (Lists) 是使用「[]」方括號括起的多個項目，每一個項目使用「,」逗號分隔。

建立串列

我們可以使用指定敘述指定變數值是一個串列，串列的項目可以是相同資料型別，也可以是不同資料型別，如下所示：

```
list1 = []  
list2 = [1, 2, 3, 4, 5]  
list3 = [1, 'Hello', 3.5]
```

上述第 1 列程式碼是建立空串列，第 2 個串列項目都是整數，第 3 個串列的項目是不同資料型別。我們也可以使用物件方式來建立串列，如下所示：

```
list4 = list()  
list5 = list(["tom", "mary", "joe"])  
list6 = list("python")
```

上述第 1 列程式碼是建立空串列，第 2 列建立參數字串項目的串列，最後是將字串中的每一個字元分割建立成串列。

建立巢狀串列

Python 串列的元素可以是另一個串列，換句話說，我們可以建立巢狀串列，相當於是其他程式語言的多維陣列，如下所示：

```
list7 = [1, ["tom", "mary", "joe"], [1, 3, 5]]
```

上述串列的第 1 個項目是整數，第 2 個和第 3 個項目是另一個字串和整數型別的串列。

輸出串列項目

Python 可以使用 `print()` 函式輸出串列項目，如下所示：

```
print(list2)
print("list3 = " + str(list3))
```

上述 `print()` 函式可以直接輸出串列變數的內容（`print()` 函式會自動轉換成字串），我們也可以呼叫 `str()` 函式建立字串連接運算式來輸出串列項目。

使用索引運算子存取串列項目

Python 串列可以使用「`[]`」索引運算子存取指定位置的項目，不只可以取出，也可以更改項目（但字串只能取出字元，並不能更改字元），索引值是從 0 開始，也可以是負值。首先是取出項目，如下所示：

```
list1 = [1, 2, 3, 4, 5, 6]
print(list1[0]) # 1
print(list1[1]) # 2
print(list1[-1]) # 6
print(list1[-2]) # 5
```

上述程式碼依序顯示串列 `list1` 的第 1 個和第 2 個項目，`-1` 是最後 1 個，`-2` 是倒數第 2 個。更改串列項目就是使用指定敘述「`=`」等號，例如：更改第 2 個項目成為 10（索引值是 1），如下所示：

```
list1[1] = 10
```

不只如此，我們還可以改成不同資料型別的項目，例如：更改第 3 個項目為字串，如下所示：

```
list1[2] = "陳會安"
```

走訪串列的每一個項目

我們可以使用 for 迴圈走訪並顯示串列的每一個項目，如下所示：

```
for e in list1:  
    print(e, end=" ")
```

上述 for 迴圈可以一一取出串列每一個項目和顯示。

存取和走訪巢狀串列

如果是多層巢狀串列，我們需要使用多個索引值來存取指定項目，如下所示：

```
list2 = [[2, 4], ["tom", "mary", "joe"], [1, 3, 5]]
```

上述巢狀串列有 2 層，第 1 層有 3 個項目，每一個項目是另一個串列，所以存取指定項目需要使用 2 個索引，例如：取得第 2 個項目中的第 1 個項目，和更改第 3 個項目中的第 2 個項目，如下所示：

```
list2[1][0]  
list2[2][1] = 7
```

同理，因為巢狀串列有兩層，我們需要使用 2 層 for 迴圈來走訪每一個項目，如下所示：

```
for e1 in list2:  
    for e2 in e1:  
        print(e2, end=" ")
```

補充說明

存取串列項目的索引值如果超過範圍，Python 直譯器會顯示 `index out of range` 索引超過範圍的 `IndexError` 錯誤訊息，如下所示：

```
list2[0][2] = 6
```

`print()` 函式顯示索引值超過範圍的串列項目，如下所示：

```
print(list[7])
```

上述程式碼會產生 `TypeError` 錯誤訊息，因為項目根本不存在，所以也不會知道是什麼型別。

在串列新增項目

Python 串列是一個容器，我們可以很容易地插入、新增和刪除串列項目。範例串列 `list1` 原來有 2 個項目，如下所示：

```
list1 = [1, 5]
```

我們可以呼叫串列物件的 `append()` 方法新增單一項目，如下所示：

```
list1.append(7)
```

上述方法新增項目 7，現在的串列是：`[1, 5, 7]`。如果需要同時新增多個項目，請使用 `extend()` 方法，如下所示：

```
list1.extend([9, 11, 13])
```

上述方法擴充串列的項目，一次新增 3 個項目，現在的串列是：`[1, 5, 7, 9, 11, 13]`。

在串列插入項目

在串列新增項目是新增在串列的最後，我們也可以使用 `insert()` 方法在指定的索引位置插入 1 個項目，繼續之前的 `list1` 串列，如下所示：

```
list1.insert(1, 3)
```

上述方法是在第 1 個參數的索引值插入第 2 個參數的項目，即插入整數 3 在第 2 個項目，現在的串列是：`[1, 3, 5, 7, 9, 11, 13]`。

刪除串列項目

我們可以使用 `del` 關鍵字刪除指定索引的串列項目，繼續之前的 `list1` 串列，如下所示：

```
del list1[2]
```

上述程式碼刪除第 3 個項目，現在的串列是：`[1, 3, 7, 9, 11, 13]`。除了 `del` 關鍵字，我們也可以使用 `pop()` 方法刪除並回傳最後 1 個項目，如下所示：

```
e1 = list1.pop()
```

上述方法刪除最後 1 個項目並回傳，所以變數 `e1` 就是最後 1 個項目 13，現在的串列是：`[1, 3, 7, 9, 11]`。如果 `pop()` 方法有指定參數值，即刪除並回傳指定索引值的項目，如下所示：

```
e2 = list1.pop(1)
```

上述方法刪除索引值 1 的第 2 個項目並回傳，所以變數 `e2` 就是第 2 個項目 3，現在的串列是：`[1, 7, 9, 11]`。如果需要刪除指定項目（不是索引），我們可以使用 `remove()` 方法，如下所示：

```
list1.remove(9)
```

上述方法刪除項目 9，現在的串列是：[1, 7, 11]。

連接運算子

算術運算子的「+」加法使用在串列，就是連接 2 個串列，即合併串列，如下所示：

```
list1 = [2, 4]
list2 = [6, 8, 10]
list3 = list1 + list2
```

上述程式碼使用串列連接運算子連接串列 list1 和 list2，故串列 list3 的項目是：[2, 4, 6, 8, 10]。

重複運算子

當算術運算子的「*」乘法使用在串列，可以建立重複第 2 個運算元次數的串列，繼續之前的 list1 串列，如下所示：

```
list4 = list1 * 3
```

上述程式碼可以重複 3 次 list1 的項目，故串列 list4 的項目是：[2, 4, 2, 4, 2, 4]。

成員運算子

成員運算子 in 和 not in 也可以使用在串列，用來檢查串列是否存在指定的項目，繼續之前的 list1 和 list2 串列，如下所示：

```
print(8 in list2)      # True
print(2 not in list1)  # False
```

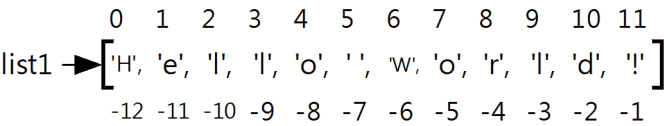
上述程式碼可以檢查項目 8 是否存在 list2 串列，項目 2 是否不存在 list1 串列。

切割運算子

切割運算子可以從原始串列切割出所需的子串列，其語法和字串的切割運算子相同，筆者就不重複說明。本節範例串列的項目是字串內容 'Hello World!' 的每一個字元，如下所示：

```
list1 = list('Hello World!')
```

上述程式碼建立的串列項目為：['H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!']，項目的索引位置值可以是正值，也可以是負值，如下圖所示：



現在，就讓我們看一些切割串列的範例，如下表所示：

切割串列	索引值範圍	取出的子串列
list1[1:3]	1~2	['e', 'l']
list1[1:5]	1~4	['e', 'l', 'l', 'o']
list1[:7]	0~6	['H', 'e', 'l', 'l', 'o', ' ', 'W']
list1[4:]	4~11	['o', ' ', 'W', 'o', 'r', 'l', 'd', '!']
list1[1:-1]	1~(-2)	['e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd']
list1[6:-2]	6~(-3)	['W', 'o', 'r', 'l']

串列函式

Python 語言內建串列函式可以取得串列長度的項目個數、排序串列、加總串列項目、取得串列中的最大和最小項目等。常用串列函式的說明，如下表所示：

串列函式	說明
len()	回傳參數串列的長度，即項目個數
max()	回傳參數串列的最大項目
min()	回傳參數串列的最小項目
list()	回傳參數字串、元組、字典和集合轉換成的串列
enumerate()	回傳 enumerate 物件，其內容是串列索引和項目的元組 (Tuple)
sum()	回傳參數串列項目的總和
sorted()	回傳排序參數串列的一個全新串列

串列方法

Python 串列物件提供數種方法來新增、插入、刪除和搜尋項目，或是排序和反轉串列等。常用串列方法的說明，如下表所示：

串列方法	說明
append(item)	新增參數 item 項目至串列的最後
extend(list)	新增參數 list 串列項目至串列的最後
insert(index, item)	在串列參數 index 位置插入參數 item 項目
pop(index)	刪除並回傳串列參數 index 索引的項目，如果沒有參數，就是最後 1 個項目
remove(item)	刪除串列第 1 個找到的參數 item 項目
count(item)	回傳串列中等於參數 item 項目的個數
index(item)	回傳串列第 1 個找到參數 item 項目的索引，如果項目不存在，就會產生 ValueError 錯誤
sort()	排序串列的項目
reverse()	反轉串列的項目