appendix

A Anaconda 開發環境與 Python 程式設計

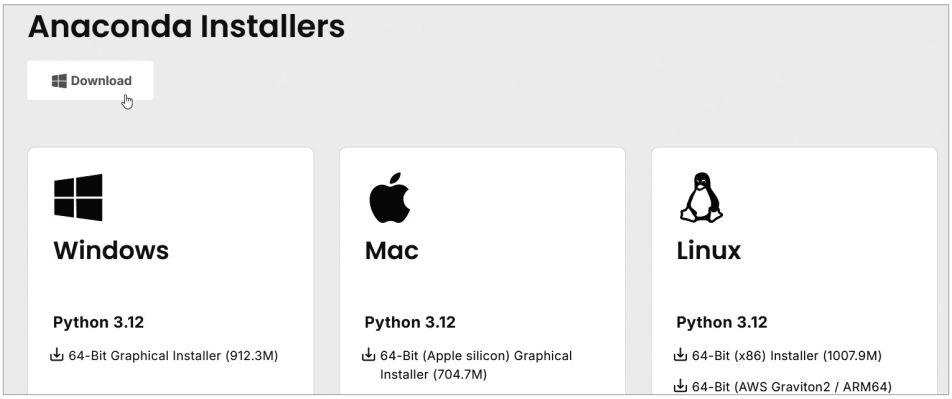
A-1 建立 Anaconda 的 Python 開發環境

Anaconda 整合安裝套件可以在官網免費下載。於本節我們準備在 Windows 作業系統的開發電腦下載和安裝 Anaconda 整合安裝套件。

下載 Anaconda

在 Anaconda 官方網站可以免費下載 Anaconda 整合安裝套件，其網址如下所示：

https://www.anaconda.com/download/success



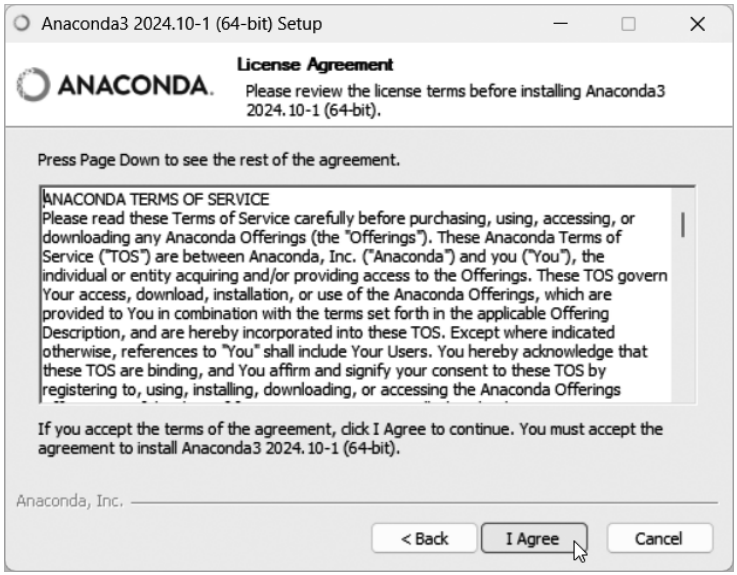
請按 **Download** 鈕下載 Anaconda 安裝程式。本書下載的 Anaconda 安裝程式檔名是：Anaconda3-2024.10-1-Windows-x86_64.exe。

安裝 Anaconda

成功下載 Anaconda 安裝程式後，即可在 Windows 電腦安裝開發環境。筆者是在 Windows 11 作業系統進行安裝（如果已經安裝舊版 Anaconda，請先解除安裝套件），其步驟如下所示：

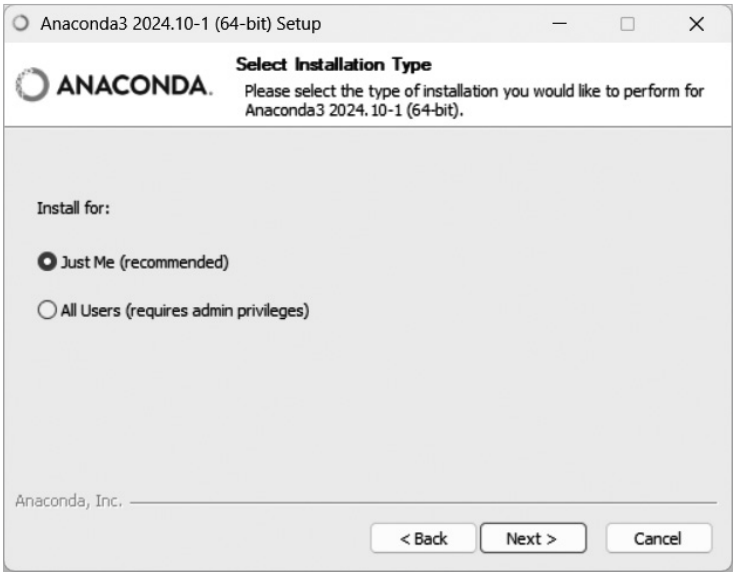
Step 1 請雙擊 **Anaconda3-2024.10-1-Windows-x86_64.exe** 安裝程式檔案，稍等一下，會看到歡迎安裝的精靈畫面。按 **Next** 鈕，可以看到使用者授權書。

Step 2 按 **I Agree** 鈕同意授權，即可選擇安裝類型。



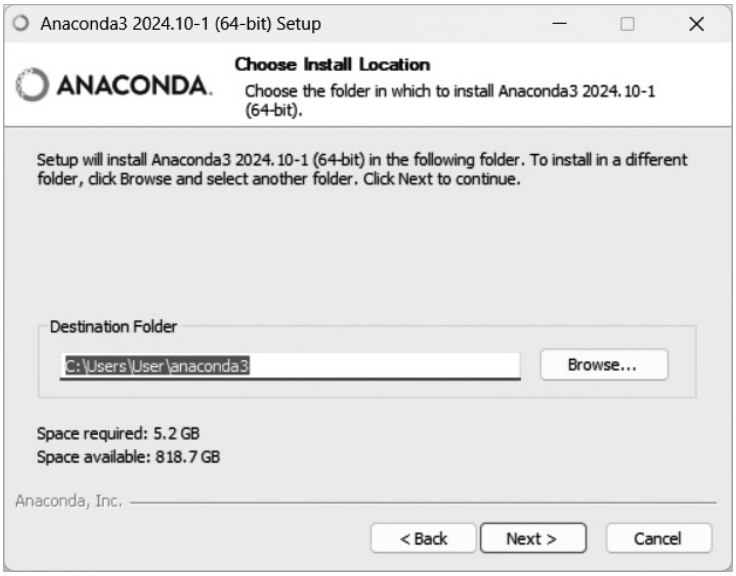
Step 3

預選 **Just Me** 安裝給目前使用者使用（建議），或選 **All Users** 安裝給所有使用者。不用更改，按 **Next** 鈕選擇安裝目錄。

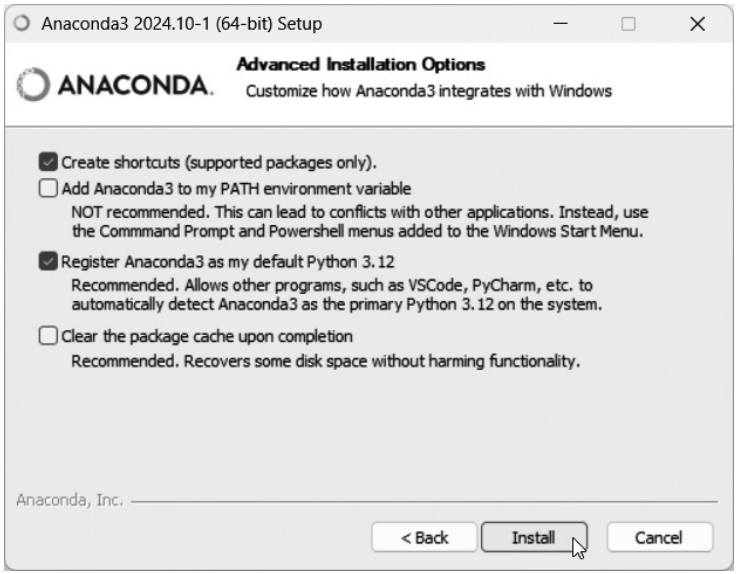


Step 4

按 **Browse** 鈕可更改安裝目錄，不用更改，按 **Next** 鈕勾選所需的進階安裝選項。

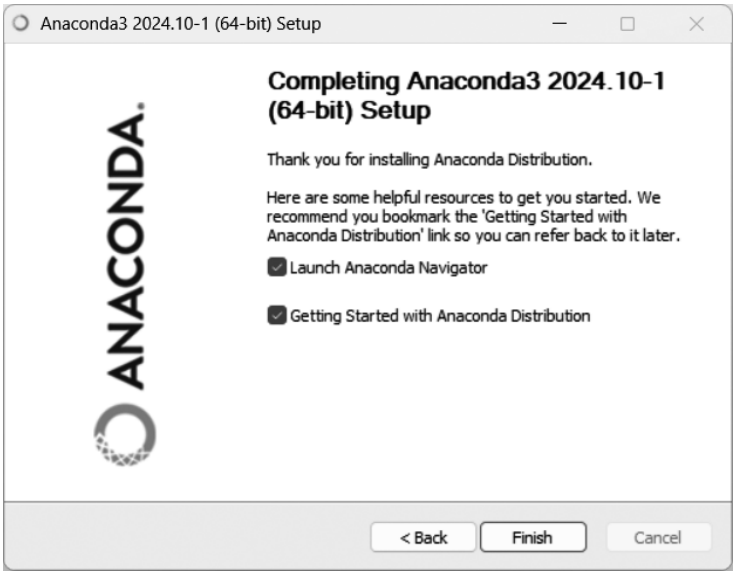


Step 5 預設勾選註冊 Anaconda 為我的預設 Python 3.x 版，不用更改，按 **Install** 鈕開始安裝，會顯示目前的安裝進度。



Step 6 完成安裝後，請按 **Next** 鈕，會看到 Anaconda 廣告畫面；再按 **Next** 鈕，即可看到完成安裝的精靈畫面。

Step 7 按 **Finish** 鈕完成 Anaconda 整合安裝套件的安裝，同時可以看到瀏覽器開啟的相關說明文件。



A-2 變數、資料型態與運算子

「變數」(Variables) 是儲存程式執行期間的暫存資料，其內容為指定資料型態 (Data Types) 的資料。Python 語言的基本資料型態有：整數、浮點數、布林和字串。

基本上，程式是一個資料轉換器，我們可以使用運算子和變數建立運算式來執行所需的程式運算，以便得到所需的執行結果。

A-2-1 使用 Python 變數

程式的變數（Variables）是用來儲存程式執行時的暫存資料。Python 變數不需要先宣告，只需指定變數值，即可建立變數。不過，Python 變數在使用前一定要記得先指定初值（Python 程式：appa-2-1.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），以及浮點數變數 height 和 weight（因為初值 175.5 有小數點）。然後使用 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。

另一種方式是使用「,」號分隔，此時不需要使用 str() 轉換型態，因為 print() 函式會幫我們轉換每 1 個參數的變數型態，如下所示：

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

A-2-2 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（越上面越優先），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
^	位元運算子 XOR
|	位元運算子 OR
in、not in、is、is not、<、<=、>、>=、!=、==	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
not	邏輯運算子 NOT
and	邏輯運算子 AND
or	邏輯運算子 OR

Python 運算式的多個運算子擁有相同的優先順序，如下所示：

```
3 + 4 - 2
```

上述運算式的「+」和「-」運算子擁有相同優先順序，此時的運算順序是從左至右依序進行運算，即先運算 3+4=7，再運算 7-2=5。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（因此變數 a、b 和 c 的值都是 25）。

A-2-3 基本資料型態

Python 語言的資料型態分為基本型態，與容器型態的串列、字典、集合和元組等。在這一節是基本資料型態的整數、浮點數、布林和字串；容器型態的說明請參閱第 A-5 節。

整數（Integers）

整數資料型態是指變數儲存資料是整數值，沒有小數點，其資料長度可以是任何長度，視記憶體空間而定。例如：一些整數值範例，如下所示：

```
a = 1
b = 100
c = 122
d = 56789
```

Python 變數可以指定成整數值，並執行相關運算（Python 程式：appa-2-3.py），如下所示：

```
x = 5
print(type(x)) # 顯示 "<class 'int'>"
print(x)       # 顯示 "5"
print(x + 1)   # 加法: 顯示 "6"
print(x - 1)   # 減法: 顯示 "4"
print(x * 2)   # 乘法: 顯示 "10"
print(x / 2)   # 除法: 顯示 "2.5"
print(x // 2)  # 整數除法: 顯示 "2"
print(x % 2)   # 餘數: 顯示 "1"
print(x ** 2)  # 指數: 顯示 "25"
x += 1
print(x)  # 顯示 "6"
x *= 2
print(x)  # 顯示 "12"
```

上述程式碼在指定變數 x 值為整數 5 後，依序使用 type(x) 顯示資料型態、執行加法、減法、乘法、除法、整數除法、餘數和指數運算。最後 2 個 x += 1 和 x *= 2 是運算式的簡化寫法，其簡化的運算式如下所示：

```
x = x + 1
x = x * 2
```

浮點數（Floats）

浮點數資料型態是指變數儲存的是整數加上小數，其精確度可達到小數點下 15 位。基本上，整數和浮點數的差異在於小數點，5 是整數，5.0 是浮點數，例如：一些浮點數值的範例，如下所示：

```
e = 1.0
f = 55.22
```

Python 浮點數的精確度只有到小數點下 15 位。同樣地，Python 變數可以指定成浮點數值，與執行相關運算（Python 程式：appa-2-3a.py），如下所示：

```
y = 2.5
print(type(y)) # 顯示 "<class 'float'>"
print(y, y + 1, y * 2, y ** 2) # 顯示 "2.5 3.5 5.0 6.25"
```

上述程式碼指定變數 y 的值為 2.5 後，顯示資料型態和執行相關數學運算。

布林（Booleans）

Python 語言的布林（Boolean）資料型態可以使用 True 和 False 關鍵字來表示，如下所示：

```
x = True
y = False
```

除了使用 True 和 False 關鍵字，下列變數值也視為 False，如下所示：

- 0、0.0：整數值 0 或浮點數值 0.0。
- []、()、{}：容器型態的空串列、空元組和空字典。
- None：關鍵字 None。

在實作上，當運算式使用關係運算子（==、!=、<、>、<=、>=）或邏輯運算子（not、and、or）時，其運算結果就是布林值。首先是邏輯運算子（Python 程式：appa-2-3b.py），如下所示：

```
a = True
b = False
print(type(a)) # 顯示 "<class 'bool'>"
print(a and b) # 邏輯AND: 顯示 "False"
print(a or b)  # 邏輯OR: 顯示"True"
print(not a)   # 邏輯NOT: 顯示 "False"
```

上述程式碼指定變數為布林值後，依序執行 AND、OR 和 NOT 運算。接著是 2 個變數比較的關係運算子（Python 程式：appa-2-3c.py），如下所示：

```
a = 3
b = 4
print(a == b)  # 相等: 顯示 "False"
print(a != b)  # 不等: 顯示 "True"
print(a > b)   # 大於: 顯示 "False"
print(a >= b)  # 大於等於: 顯示 "False"
print(a < b)   # 小於: 顯示 "True"
print(a <= b)  # 小於等於: 顯示 "True"
```

字串（Strings）

Python「字串」(Strings) 並不能更改其中內容，所有字串的變更都是建立一個全新字串。Python 字串是使用「'」單引號或「"」雙引號括起的一序列 Unicode 字元，如下所示：

```
s1 = "學習Python語言程式設計"
s2 = 'Hello World!'
```

上述程式碼的變數是字串資料型態。Python 語言並沒有字元型態，當引號括起的字串只有 1 個時，即為字元，如下所示：

```
ch1 = "A"
ch2 = 'b'
```

上述程式碼是字元。當在 Python 程式建立字串後，我們就可以顯示字串、計算字串長度、連接 2 個字串和格式化顯示字串內容（Python 程式：appa-2-3d.py），如下所示：

```
str1 = 'hello'     # 使用單引號建立字串
str2 = "python"    # 使用雙引號建立字串
print(str1)        # 顯示 "hello"
print(len(str1))   # 字串長度: 顯示 "5"
str3 = str1 + ' ' + str2  # 字串連接
print(str3)        # 顯示 "hello python"
str4 = '%s %s %d' % (str1, str2, 12)  # 格式化字串
print(str4)        # 顯示 "hello python 12"
```

上述程式碼建立字串變數 str1 和 str2 後，使用 print() 函式顯示字串內容，len() 函式計算字串有幾個英文或中文字元。我們可以使用加法「+」連接字串，或使用類似 C 語言 printf() 函式的格式字串來建立字串內容，其格式字元「%s」是字串，「%d」是整數，「%f」是浮點數。

Python 字串物件提供一些好用的方法來處理字串（Python 程式：appa-2-3e.py），如下所示：

```
s = "hello"
print(s.capitalize())  # 第1個字元大寫: 顯示 "Hello"
print(s.upper())       # 轉成大寫: 顯示 "HELLO"
print(s.rjust(7))      # 右邊填空白字元: 顯示 "  hello"
print(s.center(7))     # 置中顯示: 顯示 " hello "
print(s.replace('l', 'L'))  # 取代字串: 顯示 "heLLo"
print('  python '.strip())  # 刪除空白字元: 顯示 "python"
```

A-3 流程控制

Python 流程控制可以搭配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 程式區塊是程式碼縮排相同數量的空白字元，一般是使用 4 個空白字元。也就是說，相同縮排的程式碼屬於同一個程式區塊。

A-3-1 條件控制

Python 條件控制敘述是使用條件運算式，搭配程式區塊建立的決策敘述，可以分為三種：單選 (if)、二選一 (if/else) 或多選一 (if/elif/else)。

if 單選條件敘述：appa-3-1.py

if 條件敘述是一種是否執行的單選題，只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，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。我們可以進一步活用邏輯運算式，當氣溫在 20~22 度之間時，顯示「加一件簿外套 !」訊息文字，如下所示：

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

if/else 二選一條件敘述：appa-3-1a.py

單純 if 條件只能選擇執行或不執行程式區塊的單選題；而若是排它情況的兩個執行區塊，只能二選一，則可加上 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，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!")
```

上述程式碼的成績具有排它性，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：appa-3-1b.py

Python 多選一條件敘述是 if/else 條件的擴充，在之中加上 elif 關鍵字來新增一個條件判斷，即可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，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，則進行下一個條件判斷。

單行條件敘述：appa-3-1c.py

Python 語言並不支援「條件運算式」(Conditional Expressions)，我們可以使用單行 if/else 條件敘述來代替，其語法如下所示：

```
變數 = 變數1 if 條件運算式 else 變數2
```

上述指定敘述的「=」號右邊是單行 if/else 條件敘述，如果條件成立，就將變數指定成變數 1 的值；否則指定成變數 2 的值。例如：12/24 制的時間轉換運算式，如下所示：

```
hour = hour-12 if hour >= 12 else hour
```

上述程式碼開始是條件成立指定的變數值或運算式，接著是 if 加上條件運算式，最後 else 之後是不成立時指定的值。所以，當條件為 True，hour 變數值為 hour-12；False 則為 hour。

A-3-2 迴圈控制

Python 迴圈控制敘述提供 for 計數迴圈（Counting Loop），和 while 條件迴圈。

for 計數迴圈：appa-3-2.py

在 for 迴圈的程式敘述中擁有計數器變數，此計數器可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，計算出從 1 加至最大值的總和，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 計數迴圈需搭配內建 range() 函式使用，此函式的範圍不包含第 2 個參數本身，所以，1~m 的範圍是 range(1, m + 1)。

for 迴圈與 range() 函式

Python 的 for 計數迴圈需使用 range() 函式來產生指定範圍的計數值，這是 Python 內建函式，可以傳入 1、2 或 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（不含終止值），其預設的起始值是 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 個參數是起始值，第 2 個參數是終止值（不含終止值），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 個參數是起始值，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：appa-3-2a.py

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 迴圈來計算階層函式值，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為 False 為止。假設 m 輸入 5，即計算 5! 的值；變數 n 是計數器變數。如果符合 n <= 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-4 函式、模組與套件

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。

Python 之所以擁有強大的功能，都是因為有眾多標準和網路上現成模組 (Modules) 與套件 (Packages) 來擴充程式功能。我們可以匯入 Python 模組與套件來直接使用其提供的函式，而不用自己撰寫相關函式。

A-4-1 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。在函式的程式區塊中，可使用 return 關鍵字傳回函式值，同時結束函式的執行。函式的參數 (Parameters) 列是函式的使用介面，在呼叫時，需要傳入對應的引數 (Arguments)。

定義函式：appa-4-1.py

在 Python 程式建立無參數列和傳回值的 print_msg() 函式，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 函式如果有傳回值，需要使用 return 關鍵字來傳回值。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 關鍵字來傳回值，傳回 True 表示合法；False 為不合法。接著是一個執行運算的 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 關鍵字傳回函式的執行結果，即運算式的運算結果。

函式呼叫：appa-4-1.py

Python 程式碼呼叫函式是使用函式名稱加上括號中的引數列。由於 print_msg() 函式沒有傳回值和參數列，呼叫函式只需使用函式名稱加上空括號，如下所示：

```
print_msg()
```

函式如果有傳回值，在呼叫時可以使用指定敘述來取得傳回值，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() 函式的傳回值。如果函式傳回值為 True 或 False，例如：is_valid_num() 函式，則可在 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")
```

上述條件使用函式傳回值作為判斷條件，可以顯示數值是否合法。

A-4-2 使用 Python 模組與套件

Python 模組是單一 Python 程式檔案，即副檔名 .py 的檔案；套件是一個目錄內含多個模組的集合，且在根目錄通常包含 Python 檔案 __init__.py。

為了方便說明，當本書 Python 程式匯入 Python 模組與套件後，不論是呼叫模組的物件方法或函式，都會統一使用函式來說明。

匯入模組或套件：appa-4-2.py

Python 程式是使用 import 關鍵字匯入模組或套件，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，如下所示：

```
import random
```

上述程式碼匯入名為 random 的模組後，即可呼叫模組的 randint() 函式，產生指定範圍之間的整數亂數值，如下所示：

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

上述程式碼產生 1~100 之間的整數亂數值。

模組或套件的別名：appa-4-2a.py

在 Python 程式檔匯入模組或套件，除了使用模組或套件名稱來呼叫函式，也可使用 as 關鍵字替模組取一個別名，然後使用別名來呼叫函式，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，因此可以使用別名 R 來呼叫 randint() 函式。

匯入模組或套件的部分名稱：appa-4-2b.py

當 Python 程式使用 import 關鍵字匯入模組後，匯入的模組預設是全部內容。但實務上，我們可能只會使用到模組的 1 或 2 個函式或物件，此時請使用 form/import 程式敘述匯入模組的部分名稱，例如：在 Python 程式匯入 BeautifulSoup 模組（在 Python 開發環境需安裝 beautifulsoup4 套件 4.13.3 版和 lxml 套件 5.3.1 版），如下所示：

```
from bs4 import BeautifulSoup
```

上述程式碼匯入 BeautifulSoup 模組後，就可建立 BeautifulSoup 物件，如下所示：

```
html_str = "<p>Hello World!</p>"
soup = BeautifulSoup(html_str, "lxml")
print(soup)
```

請注意！form/import 程式敘述匯入的變數、函式或物件是匯入到目前的程式檔案，成為目前程式檔案的範圍，所以使用時不需要透過模組名稱來指定所屬的模組，直接使用 BeautifulSoup 即可。

A-5 容器型態

Python 語言支援的容器型態有：串列、字典和元組，容器型態如同一個放東西的盒子，我們可以將項目或元素等東西儲存在盒子中。

A-5-1 串列

Python 語言的「串列」(Lists) 類似其他程式語言的「陣列」(Arrays)，中文譯名有串列、清單和列表等。不同於不能更改的字串型態，串列允許更改 (Mutable) 內容，我們可以新增、刪除、插入和更改串列的項目 (Items)。

串列的基本使用：appa-5-1.py

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，如下所示：

```
ls = [6, 4, 5]    # 建立串列
print(ls, ls[2])  # 顯示 "[6, 4, 5] 5"
print(ls[-1])     # 負索引從最後開始: 顯示 "5"
ls[2] = "py"      # 指定字串型態的項目
print(ls)         # 顯示 "[6, 4, 'py']"
ls.append("bar")  # 新增項目
print(ls)         # 顯示 "[6, 4, 'py', 'bar']"
ele = ls.pop()    # 取出最後項目
print(ele, ls)    # 顯示 "bar [6, 4, 'py']"
```

上述程式碼首先建立 3 個項目的串列 ls，然後使用索引取出第 3 個項目 (索引從 0 開始)，負索引 -1 則是最後 1 個。接著更改串列項目成字串，再使用 append() 方法在串列的最後新增項目，而 pop() 方法可以取出最後 1 個項目。

切割串列：appa-5-1a.py

Python 串列可以在「[]」方括號中使用「:」符號的語法，即指定開始和結束索引來切割串列成子串列，如下所示：

```
nums = list(range(5))  # 建立一序列的整數串列
print(nums)            # 顯示 "[0, 1, 2, 3, 4]"
print(nums[2:4])       # 切割索引2~4(不含4): 顯示 "[2, 3]"
print(nums[2:])        # 切割索引從2至最後: 顯示 "[2, 3, 4]"
print(nums[:2])        # 切割從開始至索引2(不含2): 顯示 "[0, 1]"
print(nums[:])         # 切割整個串列: 顯示 "[0, 1, 2, 3, 4]"
print(nums[:-1])       # 使用負索引切割: 顯示 "[0, 1, 2, 3]"
nums[2:4] = [7, 8]     # 使用切割來指定子串列
print(nums)            # 顯示 "[0, 1, 7, 8, 4]"
```

走訪串列：appa-5-1b.py

Python 程式是使用 for 迴圈走訪並顯示串列的每一個項目，如下所示：

```
animals = ['cat', 'dog', 'bat']
for animal in animals:
    print(animal)
```

上述 for 迴圈可以一一取出串列的每一個項目並顯示出來，其執行結果如右所示：

```
cat
dog
bat
```

如果需要顯示串列各項目的索引值，則使用 enumerate() 函式，如下所示：

```
animals = ['cat', 'dog', 'bat']
for index, animal in enumerate(animals):
    print(index, animal)
```

上述 enumerate() 函式有 2 個回傳值，第 1 個 index 為是索引值，其執行結果如右所示：

```
0 cat
1 dog
2 bat
```

串列推導：appa-5-1c.py

串列推導（List Comprehension）是一種建立串列的簡潔語法，我們可以在「[]」方括號中使用 for 迴圈產生串列項目，如果需要，還可以加上 if 條件子句篩選出所需的項目，如下所示：

```
list1 = [x for x in range(10)]
```

上述程式碼的第 1 個變數 x 是串列項目，這是使用之後的 for 迴圈來產生項目，以此例是 0~9，可以建立串列：[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]。

不只如此，方括號第 1 個 x 可以是變數，也可以是運算式，例如：使用 x+1 產生項目，如下所示：

```
list2 = [x+1 for x in range(10)]
```

上述程式碼可以建立串列：[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]。如果需要，還可以在 for 迴圈後加上 if 條件子句，例如：只顯示偶數項目，如下所示：

```
list3 = [x for x in range(10) if x % 2 == 0]
```

上述程式碼在 for 迴圈後是 if 條件子句，可判斷 x % 2 的餘數是否為 0，也就是只顯示值是 0 的項目，即偶數項目，可以建立串列：[0, 2, 4, 6, 8]。同樣地，我們也能使用運算式來產生項目，如下所示：

```
list4 = [x*2 for x in range(10) if x % 2 == 0]
```

上述程式碼可以建立串列：[0, 4, 8, 12, 16]。

A-5-2 字典

Python 的「字典」(Dictionaries) 是一種儲存鍵值資料的容器型態。我們可以使用鍵 (Key) 來取出或更改值 (Value)，也可使用鍵來新增或刪除項目。對比於其他程式語言，就是結合陣列 (Associative Array)。

字典的基本使用：appa-5-2.py

Python 字典 (Dictionaries) 是使用大括號「{}」定義成對的鍵和值 (Key-value Pairs)，每一對使用「,」逗號分隔，其中的鍵和值是使用「:」冒號分隔，如下所示：

```
d = {"cat": "white", "dog": "black"}  # 建立字典
print(d["cat"])       # 使用Key取得項目: 顯示 "white"
print("cat" in d)     # 是否有Key: 顯示 "True"
d["pig"] = "pink"     # 新增項目
print(d["pig"])       # 顯示 "pink"
print(d.get("monkey", "N/A"))  # 取出項目+預設值: 顯示 "N/A"
print(d.get("pig", "N/A"))     # 取出項目+預設值: 顯示 "pink"
del d["pig"]          # 使用Key刪除項目
print(d.get("pig", "N/A"))     # "pig"不存在: 顯示 "N/A"
```

上述程式碼建立字典變數 d 後，使用鍵 "cat" 取出值，然後透過 in 運算子檢查是否有此鍵值。接著新增 "pig" 鍵值 (如果鍵值不存在，就是新增) 並顯示此鍵值。最後使用 get() 方法使用鍵取出值，如果鍵值不存在，就傳回第 2 個參數的預設值，而 del 是刪除項目。

走訪字典：appa-5-2a.py

如同串列，Python 程式一樣是使用 for 迴圈以鍵來走訪字典，如下所示：

```
d = {"chicken": 2, "dog": 4, "cat": 4, "spider": 8}
for animal in d:
    legs = d[animal]
    print(animal, legs)
```

上述程式碼建立字典變數 d 後，使用 for 迴圈走訪字典的所有鍵，可以顯示各種動物有幾隻腳，其執行結果如下所示：

```
chicken 2
dog 4
cat 4
spider 8
```

如果需要同時走訪字典的鍵和值，則需使用 items() 方法，如下所示：

```
d = {"chicken": 2, "dog": 4, "cat": 4, "spider": 8}
for animal, legs in d.items():
    print("動物: %s 有 %d 隻腳" % (animal, legs))
```

上述 for 迴圈是走訪 d.items()，可以傳回鍵 animal 和值 legs，其執行結果如下所示：

```
動物: chicken 有 2 隻腳
動物: dog 有 4 隻腳
動物: cat 有 4 隻腳
動物: spider 有 8 隻腳
```

字典推導：appa-5-2b.py

字典推導（Dictionary Comprehension）是一種建立字典的簡潔語法，我們可以在「{}」大括號中使用 for 迴圈產生字典項目，如果需要，還可以加上 if 條件子句來篩選出所需的項目，如下所示：

```
d1 = {x:x*x for x in range(10)}
```

上述程式碼的第 1 個 x:x*x 是字典項目，其位在「:」前的是鍵，之後的是值。這是使用之後的 for 迴圈來產生項目，以此例是 0~9，可以建立字典：{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}。

不只如此，我們還可以在 for 迴圈後加上 if 條件子句，例如：只顯示奇數的項目，如下所示：

```
d2 = {x:x*x for x in range(10) if x % 2 == 1}
```

上述程式碼在 for 迴圈後是 if 條件子句，可判斷 x % 2 的餘數是否為 1，也就是只顯示值是 1 的項目，即奇數項目，可以建立字典：{1: 1, 3: 9, 9: 81, 5: 25, 7: 49}。

A-5-3 元組

「元組」(Tuple) 是一種類似串列的容器型態，簡單地說，元組是一個唯讀串列：一旦 Python 程式指定元組的項目，就不再允許更改。Python 元組是使用「()」小括號來建立，每一個項目使用「,」逗號分隔 (Python 程式：appa-5-3.py)，如下所示：

```
t = (5, 6, 7, 8)  # 建立元組
print(type(t))    # 顯示 "<class 'tuple'>"
print(t)          # 顯示 "(5, 6, 7, 8)"
print(t[0])       # 顯示 "5"
print(t[1])       # 顯示 "6"
print(t[-1])      # 顯示 "8"
print(t[-2])      # 顯示 "7"
for ele in t:     # 走訪項目
    print(ele, end=" ")  # 顯示 "5, 6, 7, 8"
```

上述程式碼建立元組變數 t 後，顯示型態名稱，並在顯示元組內容後，使用索引取出指定的項目，最後使用 for 迴圈走訪元組的每個項目。

A-6 類別與物件

Python 是一種物件導向程式語言，事實上，Python 所有內建資料型態都是物件，包含：模組和函式等也都是物件。

A-6-1 定義類別和建立物件

物件導向程式是使用物件來建立程式，每一個物件儲存資料 (Data) 和提供行為 (Behaviors)，透過物件之間的通力合作來完成程式的功能。

定義類別：appa-6-1.py

類別 (Class) 是物件的模子，也是藍圖，我們需要先定義類別，才能依據類別的模子來建立物件。Python 語言是使用 class 關鍵字來定義 Student 類別，如下所示：

```
class Student:
    def __init__(self, name, grade):
        self.name = name
        self.grade = grade

    def displayStudent(self):
        print("姓名 = " + self.name)
        print("成績 = " + str(self.grade))

    def whoami(self):
        return self.name
```

上述程式碼使用 class 關鍵字定義類別，在之後的是類別名稱 Student，然後是「:」冒號。接著才是類別定義的程式區塊 (Function Block)。

一般來說，類別擁有儲存資料的「資料欄位」(Data Field) 和定義行為的方法 (Methods)，並且擁有一個特殊名稱的方法稱為「建構子」(Constructors)，其名稱一定是「__init__」。

類別建構子

類別建構子是每一次使用類別建立新物件時，就會自動呼叫的方法。Python 類別的建構子名為「__init__」，不能更名，在 init 前後是 2 個「_」底線，如下所示：

```
def __init__(self, name, grade):
    self.name = name
    self.grade = grade
```

上述建構子的寫法和 Python 函式相同，在建立新物件時，可以使用參數來指定資料欄位 name 和 grade 的初值。

建構子和方法的 self 變數

在 Python 類別建構子和方法的第 1 個參數是 self 變數，這是一個特殊變數，絕對不可以忘記此參數，其功能相當於 C# 和 Java 語言的 this 關鍵字。

請注意！self 不是 Python 語言的關鍵字，只是約定俗成的變數名稱。self 變數的值是參考呼叫建構子或方法的物件，以建構子 __init__() 方法來說，參數 self 的值是參考新建立的物件，如下所示：

```
self.name = name
self.grade = grade
```

上述程式碼 self.name 和 self.grade 就是指定新物件資料欄位 name 和 grade 的值。

資料欄位

類別的資料欄位，亦稱為成員變數（Member Variables）。在 Python 類別定義資料欄位並不需要特別語法，只要使用 self 開頭存取的變數，就是資料欄位。在 Student 類別的資料欄位有 name 和 grade，如下所示：

```
self.name = name
self.grade = grade
```

上述程式碼是在建構子指定資料欄位的初值，沒有使用特別語法，其 name 和 grade 即為類別的資料欄位。

方法

類別的方法就是 Python 函式，只是其第 1 個參數一定是 self 變數，而且在存取資料欄位時，不要忘了使用 self 變數來存取（因為有 self 才是存取資料欄位），如下所示：

```
def displayStudent(self):
    print("姓名 = " + self.name)
    print("成績 = " + str(self.grade))
```

使用類別建立物件

在定義類別後，我們可以使用類別來建立物件，也稱為實例（Instances）。同一類別可以如同工廠生產一般地建立多個物件，如下所示：

```
s1 = Student("陳會安", 85)
```

上述程式碼建立物件 s1，Student() 就是呼叫 Student 類別的建構子方法，擁有 2 個參數來建立物件。然後可以使用「.」運算子呼叫物件方法，如下所示：

```
s1.displayStudent()
print("s1.whoami() = " + s1.whoami())
```

同樣的語法可以用來存取物件的資料欄位，如下所示：

```
print("s1.name = " + s1.name)
print("s1.grade = " + str(s1.grade))
```

A-6-2 隱藏資料欄位

Python 類別定義的資料欄位和方法預設可以在其他 Python 程式碼中存取這些資料欄位，與呼叫這些方法，對比其他物件導向程式語言就是 public 公開成員。

如果資料欄位需要隱藏，或方法只能在類別中呼叫，即不是類別對外的使用介面，則需使用 private 私有成員。在 Python 資料欄位和方法名稱只需使用 2 個「_」底線開頭，就表示是私有（Private）資料欄位或方法（Python 程式：appa-6-2.py），如下所示：

```
def __init__(self, name, grade):
    self.name = name
    self.__grade = grade
```

上述建構子的 __grade 資料欄位是隱藏的資料欄位。我們也可建立只有在類別中呼叫的私有方法（Private Methods），如下所示：

```
def __getGrade(self):
    return self.__grade
```

上述方法名稱是 __getGrade()，這個方法只能在定義類別的程式碼呼叫，呼叫時記得一樣要加上 self，如下所示：

```
print("成績 = " + str(self.__getGrade()))
```