基本程式設計

程式設計可分為三大部份，分別是輸入、運算，以及輸出。如下圖所示：

程式經由輸入資料，加以處理產生結果加以輸出。在這些步驟中可能會有錯誤（bugs），此時必須加以除錯（debug），再重新執行，直到產生正確的結果為止。

這三部份環環相扣，而且都很重要，若沒有正確的輸入資料，即使有正確的處理方法，還是會產生錯誤的結果，此即所謂的垃圾進垃圾出（garbage in garbage out，GIGO）。至於輸出結果除了先要求正確外，再要求美觀。最後的處理過程無庸置疑是程式的核心。

有了問題後，開始撰寫解決此問題的程式，需要有代表問題中項目的變數名稱，如我們要計算二個整數的和，則需要代表這二個整數與和的變數名稱，如分別是 a、b 和 total。

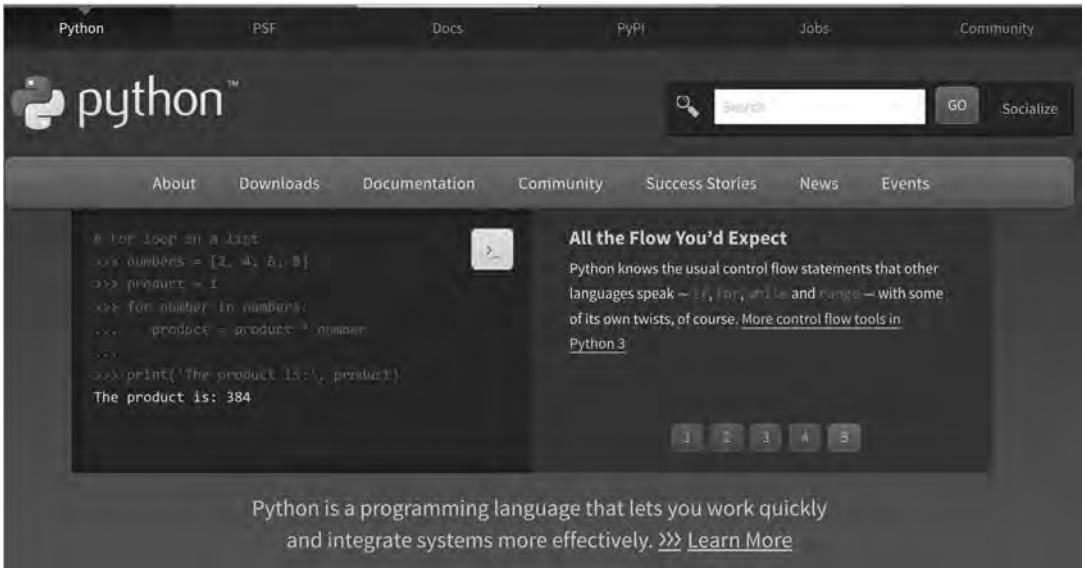
要注意取變數名稱儘量和所要表示的項目相接近，如上述的 total 看起來就是總和的意思，還有要遵循取變數名稱的一些規則，如下所述：

1、 變數名稱以英文字母或底線開頭，不可以其它數字或符號字元。

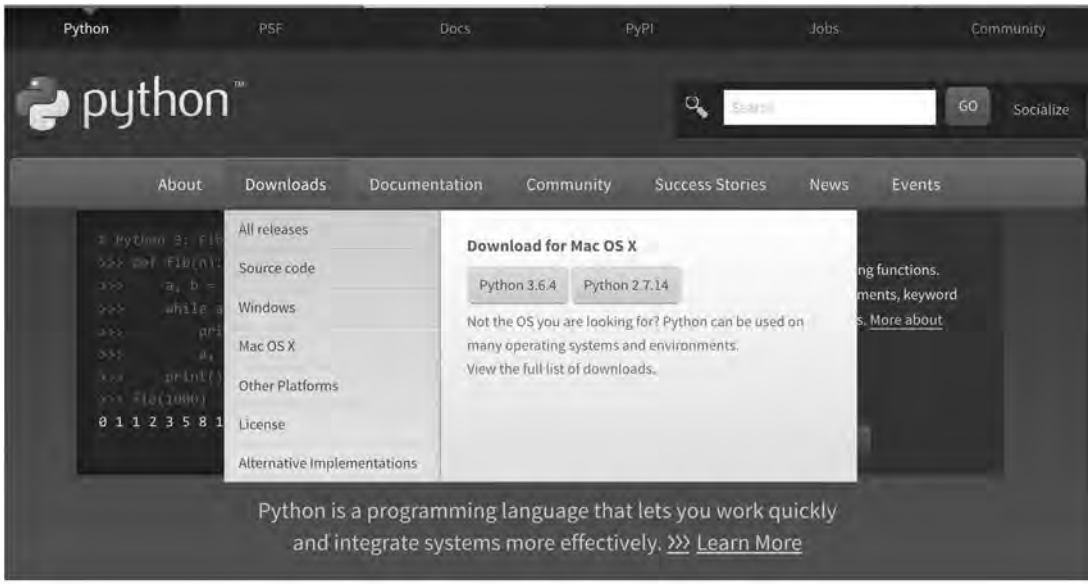
2、 接下來可以是數字或底線。

工欲善其事，必先利其器。所以在未進入撰寫程式前必需下載 Python 的直譯器軟體，茲說明如下。

請先到 **www.python.org** 的網站，會出現以下的畫面：



接著，選取 Downloads 選單下適當的選項，如下圖所示：



筆者是選取 **Mac OS X** 和 **Python 3.6.4**，就可以將 Python 的直譯器下載到你的應用程式中。

從應用程式中選取 Python 的圖樣，如下所示：

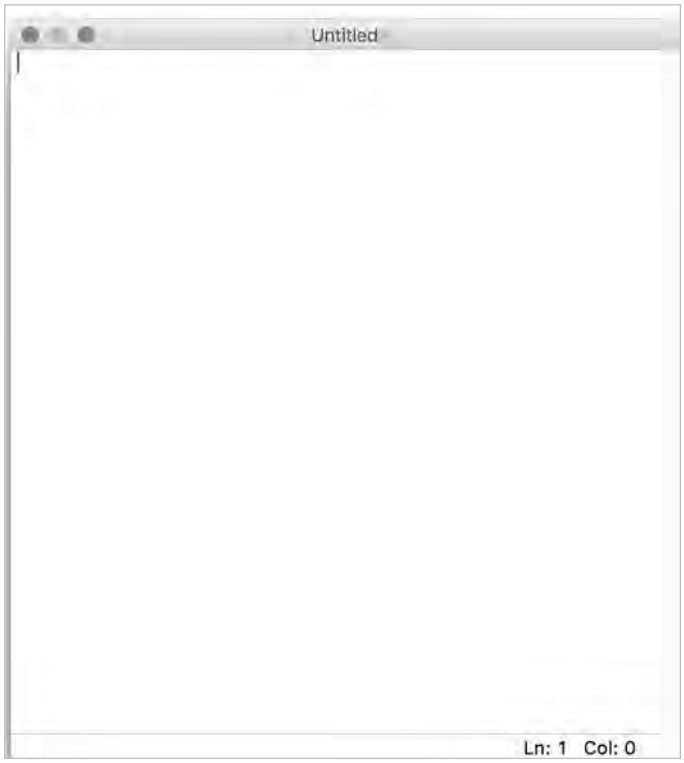
按此圖樣二下，就會得到以下的畫面：

從中選取左上角的 IDLE 圖樣，就可以撰寫程式了，如下圖所示：

凡是在本書看到的程式中，若前面有>>>符號，則是在上述 IDLE 的模式下執行的。

這個模式下的程式無法儲存，當重新再開啓 IDLE 時，原先的程式就不見了，此時，若想保存程式內容，你也可以在 File 選單下選取 New File 的選項，如下圖所示：

此時就可以撰寫程式了，當撰寫完之後，會出現以下的畫面讓你撰寫程式。



程式撰寫完之後，選取 Run -> Run Module 的選項執行之。在執行之前系統會請你將此檔案儲存起來。

本書中，若程式左邊有行號，則是在此檔案的模式下進行的，就讓我們從輸出來加以解說。

1-1 輸出函式

Python 的輸出函式為 print 函式，將結果顯示於螢幕上。請看以下範例：

範例程式：

```
1  a = 123
2  b = 123.456
3  c = 'Python'
4
5  print(a)
6  print(b)
7  print(c)
8
9  print('a =', a)
10 print('b =', b)
11 print('c =', c)
```

輸出結果：

```
123
123.456
Python
a = 123
b = 123.456
c = Python
```

程式中的 = 符號是指定運算子（assignment operator），表示將右邊的數值或經過運算後數值，指定給左邊的變數，如：

a = 123

表示將 123 指定給 a 變數。此稱為運算式（expression），在 Python 也稱為敘述（statement）。其它指定運算式相同。

從輸出結果中得知，這二種的輸出差異在於，後者在輸出答案時有加上一些訊息，使得結果更易閱讀。簡易地的說，以單引號或雙引號括住的字串皆會照印出來。

還有一點要說明的是，print 函式的輸出都會跳行。若要不跳行，則可以加上 end 的參數，如下所示：

範例程式：

```
1  a = 123
2  b = 123.456
3  c = 'Python'
4
5  print(a, end = '  ')
6  print(b, end = '  ')
7  print(c)
```

輸出結果：

```
123 123.456 Python
```

當我們加上 end = ' ' 後，前兩個 print 的輸出結果皆不會跳行，輸出在同一行上。注意，end 後面單引號內有空一格，也可以空多格。當然，也可以是其它字元如 *。如下範例所示：

範例程式：

```
1  a = 123
2  b = 123.456
3  c = 'Python'
4
5  print(a, end = '*')
6  print(b, end = '*')
7  print(c)
```

輸出結果：

```
123*123.456*Python
```

1-1-1 format 格式化輸出

除此之外，為了輸出的美觀，Python 提供了格式化的輸出。計有 format 和 % 兩種格式。如下範例所示：

範例程式：

```
1  a = 123
2  b = 123.456
3  c = 'Python'
4
5  print(format(a, '5d'))
6  print(format(b, '10.2f'))
7  print(format(c, '10s'))
8  print()
9
10 print(format(a, '<5d'))
11 print(format(b, '<10.2f'))
12 print(format(c, '>10s'))
```

輸出結果：

```
  123
    123.46
Python

123
123.46
    Python
```

程式中的 print() 主要的用意在於跳一空白行。其中 5d 的 5 表示有 5 個欄位寬，d 表示是一整數。10.2f 中的 f 表示對應的是浮點數，10.2 表示後面有 2 位小數點，

有 10 個欄位寬，這 10 個欄位寬包含小數點。10s 的 s 表示印出的是字串，而且其欄位寬為 10。

從輸出結果得知，Format 格式化輸出，字串的預設輸出是向左靠齊，而整數和浮點數是向右靠齊。還好，我們可以利用 > 和 < 的符號做為向右和向左靠齊的機制。如輸出結果所示。若要置中，則可使用 ^ 符號，請讀者自行測試之。

1-1-2 % 的格式化輸出

除了 format 的格式化輸出外，Pyhton 還提供了一個更方便的格式化輸出的利器，那就是 %，其語法如下：

print(‘%格式化字符’%(variable_list))

以 % 為開頭，後接格式化字符，它可為 d、f、s，這是第一個參數，並以單引號或雙引號括住，接下來是 % 再加上以小括號，括住每一個格式化字符所對應的變數。如下範例所示：

範例程式：

```
1  a = 123
2  b = 123.456
3  c = 'Python'
4
5  print('%5d'%(a))
6  print('%10.2f'%(b))
7  print('%10s'%(c))
8  print()
9
10 print('%-5d'%(a))
11 print('%-10.2f'%(b))
12 print('%-10s'%(c))
```

輸出結果：

```
  123
    123.46
    Python
```

```
123
123.46
Python
```

從輸出結果得知，程式中的

print('%5d'%(a))

表示 a 以 5d 的格式印出，注意，% 的符號不可省略喔！其餘的依此類推。在 % 的格式化中，不管整數、浮點數或是字串的輸出，皆是向右靠齊，若要向左靠齊則必需加上負的符號 '-' 才可。所以

print('%-5d'%(a))

則是將結果靠左對齊。如輸出結果所示。

為了能讓讀者看出欄位寬的作用，在下一範例程式中加入兩條直線做為輸出結果的界線，以方便檢視，如下範例程式所示：

範例程式：

```
1  a = 123
2  b = 123.456
3  c = 'Python'
4
5  print('|%5d|'%(a))
6  print('|%10.2f|'%(b))
7  print('|%10s|'%(c))
8
9  print('|%-5d|'%(a))
10 print('|%-10.2f|'%(b))
11 print('|%-10s|'%(c))
```

輸出結果：

```
|  123|
|    123.46|
|    Python|
|123  |
|123.46    |
|Python    |
```

這輸出結果更能看出欄位寬和向左靠齊的作用。

上述的整數之輸出結果皆以十進位的方式輸出。在 format 格式化中也可以使用 x、o、b 分別以十六進位、八進位和二進位的方式輸出結果。如下所示：

範例程式：

```
1  a = 123
2
3  print(format(a, '5x'))
4  print(format(a, '5o'))
5  print(format(a, '5b'))
```

輸出結果：

```
   7b
  173
1111011
```

但在 % 格式化輸出中，就沒有二進位的機制加以輸出結果。$(123)_{10} = (173)_8 = (1111011)_2$。如下所示：

範例程式：

```
1  a = 123
2
3  print('%5x'%(a))
4  print('%5o'%(a))
```

輸出結果：

```
   7b
  173
```

行文至此，格式化的輸出的優點是啥呢？可以將輸出結果加以美化，以及更易參閱。我們以範例來說明：

進階控制流程

此章的進階控制流程其實就是前面二章的應用，也就是將選擇敘述與迴圈敘述混搭來完成你的工作。利用迴圈敘述加上選擇敘述，好比如虎添翼般的更具有威力，請參閱以下的範例程式。

4-1 亂數產生器

試撰寫一程式產生 10 個亂數，程式如下所示：

範例程式：

```
1  import random
2  for i in range(1, 11):
3      randNum = random.randint(1, 100)
4      print('%4d'%(randNum), end = '')
```

輸出結果：

```
   8  41  69   7  12  76   8  21  23  71
```

若要檢視所產生的亂數中有多少個是偶數或是奇數，此時就必需藉助選擇敘述來判斷。如下範例程式所示：

範例程式：

```
1  import random
2  even = 0
3  odd = 0
4  for i in range(1, 11):
5      randNum = random.randint(1, 100)
6      print(randNum, end = ' ')
7      if randNum % 2 == 0:
8          even += 1
9      else:
10         odd += 1
11 print('\neven = %d, odd = %d'%(even, odd))
```

輸出結果：

```
23  41  14   8   2  10  21  34  85  41
even = 5, odd = 5
```

此程式較上一程式多了以下的敘述：

範例程式：

```
1  if randNum % 2 == 0:
2       even += 1
3  else:
4       odd += 1
```

以及用來印出偶數和奇數個數的 print 敘述：

print('\neven = %d, odd = %d'%(even, odd))

再產生多一點的亂數，今利用亂數產生器產生 100 個亂數，然後判斷這些亂數有多少個是 3 的倍數，5 的倍數，和 7 的倍數，則程式如下所示：

範例程式：

```
1  #Version 1
2  import random
3  times3 = 0
4  times5 = 0
5  times7 = 0
6  others = 0
7  for i in range(1, 101):
8      flag = False
9      randNum = random.randint(1, 100)
10     print(randNum, end = ' ')
11     if randNum % 3 == 0:
12         times3 += 1
13         flag = True
14     if randNum % 5 == 0:
15         times5 += 1
16         flag = True
```

```
17      if randNum % 7 == 0:
18          times7 +=1
19          flag = True
20      if flag == False:
21          others += 1
22  print('\ntimes3 = %d, times5 = %d , times7 = %d'%(times3,
23        times5, times7))
24  print('others = %d'%(others))
```

輸出結果：

```
85 42 10 27 12 89 96 42 35 86 65 46 15 59 30 42 87 22 82 77 78 85 53
10 91 75 93 10 50 65 4 5 3 10 8 11 9 38 7 54 94 24 27 51 2 81 8 31
24
37 87 82 69 44 76 60 86 90 23 16 19 18 76 81 23 43 89 15 44 10 31 23
23 99 61 94 67 100 71 71 90 69 15 28 21 54 85 7 22 76 75 83 25 84 53
2 78 91 79 75
times3 = 37, times5 = 25 , times7 = 12
others = 42
```

此程式和上一程式差不多都使用選擇敘述來判斷所產生的亂數是 3 或 5 或 7 的倍數。程式雖然可正確的執行，但其輸出結果並不怎麼美觀，以下程式將針對此點做了一些改善，請參閱以下範例程式：

範例程式：

```
1   #Version 2
2   import random
3   times3 = 0
4   times5 = 0
5   times7 = 0
6   others = 0
7   count = 1
8   for i in range(1, 101):
9       flag = False
10      randNum = random.randint(1, 100)
11      if count % 10 != 0:
12          print('%5d'%(randNum), end = ' ')
13      else:
```

```
14          print('%5d'%(randNum))
15      count += 1
16
17      # Calculate times3, times5, and times7
18      if randNum % 3 == 0:
19          times3 += 1
20      if randNum % 5 == 0:
21          times5 += 1
22      if randNum % 7 == 0:
23          times7 +=1
24      if flag == False
25          others += 1
26  print('\ntimes3 = %d, times5 = %d , times7 = %d'%(times3, times5, times7))
27  print('others = %d'%(others))
```

輸出結果：

```
   43    51    44    41    82    76    50     8    36    43
   58    68    85    77    11    50    39    98    32    29
   55    69    53    33    82    53    23    19    12    22
   36    26    80    96    79    32    69    38    69    95
   81    90     1    86    20    32    44    73    83    52
   95    50    45    20    40    56    51    84    34    56
   10    88    68    25    17    82    28    58     6    54
   48    79     2    36    60    29    75    81    36    35
   68    91    25    44    62    92    74    66    37    89
   67    85     5    59    53     6    36    40    45    98

times3 = 27, times5 = 23, times7 = 9
others = 48
```

我們將這 100 個亂數以一列印出 10 個亂數，以下是其對應的片段程式碼：

範例程式：

```
1   count = 1
2   for i in range(1, 101):
3       randNum = random.randint(1, 100)
4       if count % 10 != 0:
5           print('%5d'%(randNum), end = ' ')
6       else:
7           print('%5d'%(randNum))
8       count += 1
```

程式中以 count 變數來控制每一列要印出 10 個亂數。當 count 可被 10 整除時，則要跳行，所以其對應的 print 敘述不必加 end = ''。

4-2 定數迴圈與不定數迴圈

當迴圈有固定執行的次數時，我們稱之為定數迴圈。而不定數迴圈，表示沒有固定的迴圈執行次數，使用者隨時可以以另一種方式來中止迴圈的執行。

例如，產生 10 次的 1 到 49 的亂數，此為定數迴圈，因為我們固定它執行 10 次。程式如下所示：

範例程式：

```
1  import random
2  count = 1
3  while count <= 10:
4      for i in range(1, 7):
5          randNum = random.randint(1, 49)
6          print('%3d'%(randNum), end = ' ')
7      print()
8      count += 1
9  print('Over')
```

輸出結果：

```
 23  37  49  40  49  34
 30   1  22  23  20  44
 12  39   5  30  43  33
 20  45  42   2  16  42
 10  13  42  11  43  36
 44  47  16  37  46  23
 34   3   2  10  36  23
 38  39  23  44  27  41
 19   6  37  42  27   9
  3  44  26   1   6   1
Over
```

此程式為多重迴圈，在外迴圈的 while 用來控制產生多少次的亂數，以 count 變數來輔助。而在內迴圈的 for 則產生六個 1~49 的亂數。

我們現在以不定數迴圈來實作之。程式中以交談式的方式詢問使用者是否要再繼續產生六個 1~49 的亂數。如以下範例程式所示：

範例程式：

```
1  import random
2  again = 1
3  while again == 1:
4      for i in range(1, 7):
5          randNum = random.randint(1, 49)
6          print('%3d'%(randNum), end = ' ')
7      print()
8      again = eval(input('continue:1 or quit:0 ---->'))
```

輸出結果：

```
22  12   8   6   3  11
continue:1 or quit:0 ---->1
 20   1  46  42  43  28
continue:1 or quit:0 ---->1
 27  45  49   9  40   8
continue:1 or quit:0 ---->1
 30  18  21  10  35  34
continue:1 or quit:0 ---->1
 19  14  43  34  33  14
continue:1 or quit:0 ---->0
Over
```

程式中以 again 變數來控制程式是否繼續產生六個 1~49 的亂數。以交談式的方式引導使用者輸入一數值，並指定給 again 變數，若輸入 1，則表示繼續產生六個 1~49 的亂數，若為 0，則結束迴圈的執行。