本章將從基本數學運算開始，一步一步講解變數的使用與命名，接著介紹 Python 的算數運算。

2-1 用 Python 做計算

假設讀者到麥當勞打工，一小時可以獲得 120 元時薪，如果想計算一天工作 8 小時，可以獲得多少工資？我們可以用計算機執行 "120 * 8"，然後得到執行結果。在 Python Shell，可以使用下列方式計算。

```
>>> 120 * 8
960
>>>
```

如果一年實際工作天數是 300 天，可以用下列方式計算一年所得。

```
>>> 120 * 8 * 300
288000
>>>
>>>
```

如果讀者一個月花費是 9000 元，可以用下列方式計算一年可以儲存多少錢。

```
>>> 9000 * 12
108000
>>> 288000 - 108000
180000
>>>
```

上述筆者先計算一年的花費，再將一年的收入減去一年的花費，可以得到所儲存的金額。本章筆者將一步一步推導應如何以程式觀念，處理一般的運算問題。

2-2 認識變數

變數是一個暫時儲存資料的地方，對於 2-1 節的內容而言，如果你今天獲得了調整時薪，時薪從 120 元調整到 125 元，如果想要從重計算一年可以儲存多少錢，你將發現所有的計算將要重新開始。為了解決這個問題，我們可以考慮將時薪設為一個變數，未來如果有調整薪資，可以直接更改變數內容即可。

在 Python 中可以用 "=" 等號設定變數的內容，在這個實例中，我們建立了一個變數 x，然後用下列方式設定時薪。

```
>>> x = 120
>>>
```

如果想要用 Python 列出時薪資料可以使用 print( ) 函數。

```
>>> print(x)
120
>>>
```

如果今天已經調整薪資，時薪從 120 元調整到 125 元，那　我們可以用下列方式表達。

```
>>> x = 125
>>> print(x)
125
>>>
```

註 在 Python Shell 環境，也可以直接輸入變數名稱，即可獲得執行結果。

```
>>> x = 125
>>> x
125
>>>
```

一個程式是可以使用多個變數的，如果我們想計算一天工作 8 小時，一年工作 300 天，可以賺多少錢，假設用變數 y 儲存一年工作所賺的錢，可以用下列方式計算。

```
>>> x = 125
>>> y = x * 8 * 300
>>> print(y)
300000
>>>
```

如果每個月花費是 9000 元，我們使用變數 z 儲存每個月花費，可以用下列方式計算每年的花費，我們使用 a 儲存每年的花費。

```
>>> z = 9000
>>> a = z * 12
>>> print(a)
108000
>>>
```

如果我們想計算每年可以儲存多少錢，我們使用 b 儲存每年所儲存的錢，可以使用下列方式計算。

```
>>> x = 125
>>> y = x * 8 * 300
>>> z = 9000
>>> a = z * 12
>>> b = y - a
>>> print(b)
192000
>>>
```

從上述我們很順利的使用 Python Shell 計算了每年可以儲存多少錢的訊息了，可是上述使用 Python Shell 做運算潛藏最大的問題是，只要過了一段時間，我們可能忘記當初所有設定的變數是代表什麼意義。因此在設計程式時，如果可以為變數取個有意義的名稱，未來看到程式時，可以比較容易記得。下列是筆者重新設計的變數名稱：

- 時薪：**hourly_salary**，用此變數代替 x，每小時的薪資。
- 年薪：**annual_salary**，用此變數代替 y，一年工作所賺的錢。
- 月支出：**monthly_fee**，用此變數代替 z，每個月花費。
- 年支出：**annual_fee**，用此變數代替 a，每年的花費。
- 年儲存：**annual_savings**，用此變數代替 b，每年所儲存的錢。

如果現在使用上述變數重新設計程式，可以得到下列結果。

```
>>> hourly_salary = 125
>>> annual_salary = hourly_salary * 8 * 300
>>> monthly_fee = 9000
>>> annual_fee = monthly_fee * 12
>>> annual_savings = annual_salary - annual_fee
>>> print(annual_savings)
192000
>>>
```

相信經過上述說明，讀者應該了解變數的基本意義了。

2-3 認識程式的意義

延續上一節的實例，如果我們時薪改變、工作天數改變或每個月的花費改變所有輸入與運算皆要重新開始，而且每次皆要重新輸入程式碼，這是一件很費勁的事，同時很可能會常常輸入錯誤，為了解決這個問題，我們可以使用 Python Shell 開啟一個檔案，將上述運算儲存在檔案內，這個檔案就是所謂的程式。未來有需要時，再開啟重新運算即可。

程式實例 ch2_1.py：使用程式計算每年可以儲存多少錢，下列是整個程式設計。

```
1  # ch2_1.py
2  hourly_salary = 125
3  annual_salary = hourly_salary * 8 * 300
4  monthly_fee = 9000
5  annual_fee = monthly_fee * 12
6  annual_savings = annual_salary - annual_fee
7  print(annual_savings)
```

執行結果

```
========== RESTART: D:/Python/ch2/ch2_1.py ==========
192000
>>>
```

未來我們時薪改變、工作天數改變或每個月的花費改變，只要適度修改變數內容，就可以獲得正確的執行結果。

2-4 認識註解的意義

上一節的程式 ch2_1.py，儘管我們已經為變數設定了有意義的名稱，其實時間一久，常常還是會忘記各個指令的內涵。所以筆者建議，設計程式時，適度的為程式碼加上註解。在 1-9 節已經講解註解的方法，下列將直接以實例說明。

程式實例 ch2_2.py：重新設計程式 ch2_1.py，為程式碼加上註解。

```
1  # ch2_2.py
2  hourly_salary = 125                          # 設定時薪
3  annual_salary = hourly_salary * 8 * 300      # 計算年薪
4  monthly_fee = 9000                           # 設定每月花費
5  annual_fee = monthly_fee * 12                # 計算每年花費
6  annual_savings = annual_salary - annual_fee  # 計算每年儲存金額
7  print(annual_savings)                        # 列出每年儲存金額
```

執行結果 與 ch2_1.py 相同。

相信經過上述註解後，即使再過 10 年，只要一看到程式應可輕鬆瞭解整個程式的意義。

2-5 Python 變數與其它程式語言的差異

許多程式語言變數在使用前是需要先宣告，Python 對於變數的使用則是可以在需要時，再直接設定使用。有些程式語言在宣告變數時，需要設定變數的資料型態，Python 則不需要設定，它會針對變數值的內容自行設定資料型態。

2-6 變數的命名原則

Python 對於變數的命名，使用有一些規則要遵守，否則會造成程式錯誤。

- 必須由英文字母、_（底線）或中文字開頭，建議使用英文字母。
- 變數名稱只能由英文字母、數字、_（底線）或中文字所組成。
- 英文字母大小寫是敏感的，例如：Name 與 name 被視為不同變數名稱。
- Python 系統保留字（或稱關鍵字）或 Python 內建函數名稱不可當作變數名稱。

註 雖然變數名稱可以用中文字，不過筆者不建議使用中文字，也許是怕將來有相容性的問題。

下列是不可當作變數名稱的 Python 系統保留字。

and	as	assert	break	class	continue
def	del	elif	else	except	False
finally	for	from	global	if	import
in	is	lambda	none	nonlocal	not
or	pass	raise	return	True	try
while	with	yield			

下列是不可當作變數名稱的 Python 系統內建函數，若是不小心將系統內建函數名稱當作變數，程式本身不會錯誤，但是原先函數功能會喪失。

abs( )	all( )	any( )	apply( )	basestring( )
bin( )	bool( )	buffer( )	bytearray( )	callable( )
chr( )	classmethod( )	cmp( )	coerce( )	compile( )
complex( )	delattr( )	dict( )	dir( )	divmod( )
enumerate( )	eval( )	execfile( )	file( )	filter( )
float( )	format( )	frozenset( )	getattr( )	globals( )
hasattr( )	hash( )	help( )	hex( )	id( )
input( )	int( )	intern( )	isinstance( )	issubclass( )
iter( )	len( )	list( )	locals( )	long( )
map( )	max( )	memoryview( )	min( )	next( )
object( )	oct( )	open( )	ord( )	pow( )
print( )	property( )	range( )	raw_input( )	reduce( )
reload( )	repr( )	reversed( )	round( )	set( )
setattr( )	slice( )	sorted( )	staticmethod( )	str( )
sum( )	super( )	tuple( )	type( )	unichr( )
unicode( )	vars( )	xrange( )	zip( )	_import( )

實例 1：下列是一些不合法的變數名稱。

```
sum,1          # 變數不可有 ","
3y             # 變數不可由阿拉伯數字開頭
x$2            # 變數不可有 "$" 符號
and            # 這是系統保留字不可當作變數名稱
```

實例 2：下列是一些合法的變數名稱。

```
SUM
_fg
x5
總和
```

實例 3：下列 3 個代表不同的變數。

SUM

Sum

sum

2-7 基本數學運算

2-7-1 四則運算

Python 的四則運算是指加（+）、減（-）、乘（*）和除（/）。

實例 1：下列是加法與減法運算實例。

```
>>> x = 5 + 6            # 將5加6設定給變數x
>>> print(x)
11
>>> y = x - 10           # 將x減10設定給變數y
>>> print(y)
1
>>>
```

實例 2：乘法與除法運算實例。

```
>>> x = 5 * 9            # 將5乘以9設定給變數x
>>> print(x)
45
>>> y = 9 / 5            # 將9除以5設定給變數y
>>> print(y)
1.8
>>>
```

2-7-2 餘數和整除

餘數（mod）所使用的符號是 "%"，可計算出除法運算中的餘數。整除所使用的符號是 "//"，是指除法運算中只保留整數部分。

實例 1：餘數和整除運算實例。

```
>>> x = 9 % 5            # 將9除以5的餘數設定給變數x
>>> print(x)
4
>>> y = 9 // 2           # 將9除以2的整數結果設定給變數y
>>> print(y)
4
>>>
```

2-7-3 次方

次方的符號是 " ** "。

實例 ch1：平方、次方的運算實例。

```
>>> x = 3 ** 2           # 將3的平方設定給變數x
>>> print(x)
9
>>> y = 3 ** 3           # 將3的3次方設定給變數y
>>> print(y)
27
>>>
```

2-7-4 Python 語言控制運算的優先順序

Python 語言碰上計算式同時出現在一個指令內時，除了括號 "（"、"）" 最優先外，其餘計算優先次序如下。

1. 次方
2. 乘法、除法、求餘數（%）、求整數（//），彼此依照出現順序運算。
3. 加法、減法，彼此依照出現順序運算。

實例 1：Python 語言控制運算的優先順序的應用。

```
>>> x = ( 5 + 6 ) * 8 - 2
>>> print(x)
86
>>> y = 5 + 6 * 8 - 2
>>> print(y)
51
>>>
```

2-8 指派運算子

常見的指派運算子如下：

運算子	實例	說明
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
%=	a %= b	a = a % b
//=	a //= b	a = a // b
**=	a **= b	a = a ** b

實例 1：指派運算子的實例說明。

```
>>> x = 10
>>> x += 5
>>> print(x)
15
>>> x = 10
>>> x -= 5
>>> print(x)
5
>>> x = 10
>>> x *= 5
>>> print(x)
50
>>> x = 10
>>> x /= 5
>>> print(x)
2.0
>>> x = 10
>>> x %= 5
>>> print(x)
0
>>> x = 10
>>> x //= 5
>>> print(x)
2
>>> x = 10
>>> x **= 5
>>> print(x)
100000
>>>
```


2-9 Python 等號的多重指定使用

使用 Python 時，可以一次設定多個變數等於某一數值。

實例 1：設定多個變數等於某一數值的應用。

```
>>> x = y = z = 10
>>> print(x)
10
>>> print(y)
10
>>> print(z)
10
>>>
```

Python 也允許多個變數同時指定不同的數值。

實例 2：設定多個變數，每個變數有不同值。

```
>>> x, y, z = 10, 20, 30
>>> print(x, y, z)
10 20 30
>>>
```

當執行上述多重設定變數值後，甚至可以執行更改變數內容。

實例 3：將 2 個變數內容交換。

```
>>> x, y = 10, 20
>>> print(x, y)
10 20
>>> x, y = y, x
>>> print(x, y)
20 10
>>>
```

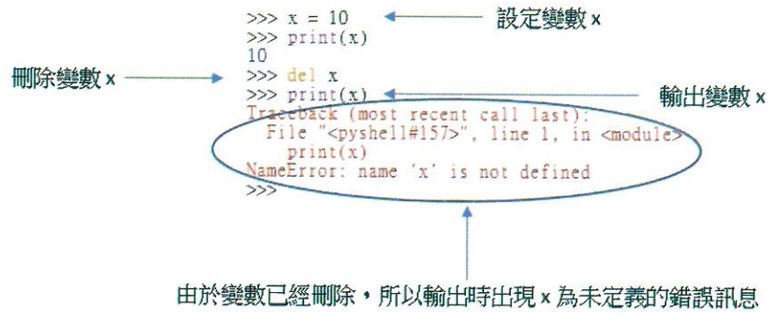
上述原先 x, y 分別設為 10, 20，但是經過多重設定後變為 20, 10。

2-10 刪除變數

程式設計時，如果某個變數不再需要，可以使用 del 指令將此變數刪除，相當於可以收回原變數所佔的記憶體空間，以節省記憶體空間。刪除變數的格式如下：

```
del 變數名稱
```

實例 1：驗證變數名稱回收後，將無法再使用。此例，嘗試輸出已刪除的變數，然後程式出現錯誤訊息。



2-11 Python 的斷行

2-11-1 一行有多個敘述

在 Python 是允許一行有多個敘述，彼此用 ";" 隔開即可，儘管 Python 有提供此功能，不過筆者不鼓勵如此撰寫程式碼。

程式實例 ch2_3.py：一行有多個敘述的實例。

```
1  # ch2_3.py
2  x = 10
3  print(x)
4  y = 20;print(y)        # 一行有2個敘述不過不鼓勵這種寫法
```

執行結果

```
========== RESTART: D:/Python/ch2/ch2_3.py ==========
10
20
>>>
```

2-11-2 將一個敘述分成多行

在設計大型程式時，常會碰上一個敘述很長，需要分成 2 行或更多行撰寫，此時可以在敘述後面加上 "\" 符號，Python 解譯器會將下一行的敘述視為這一行的敘述。特別注意，在 "\" 符號右邊不可加上任何符號或文字，即使是註解符號也是不允許。

另外，也可以在敘述內使用小括號，如果使用小括號，就可以在敘述右邊加上註解符號。

程式實例 ch2_4.py：將一個敘述分成多行的應用。

```
1  # ch2_4.py
2  a = b = c = 10
3  x = a + b + c + 12
4  print(x)
5  # 續行方法1
6  y = a +\
7      b +\
8      c +\
9      12
10 print(y)
11 # 續行方法2
12 z = ( a +      # 此處可以加上註解
13       b +
14       c +
15       12 )
16 print(z)
```

執行結果

```
========== RESTART: D:\Python\ch2\ch2_4.py ==========
42
42
42
>>>
```