UNIT 3

python™

Variable, Assignment, Operator

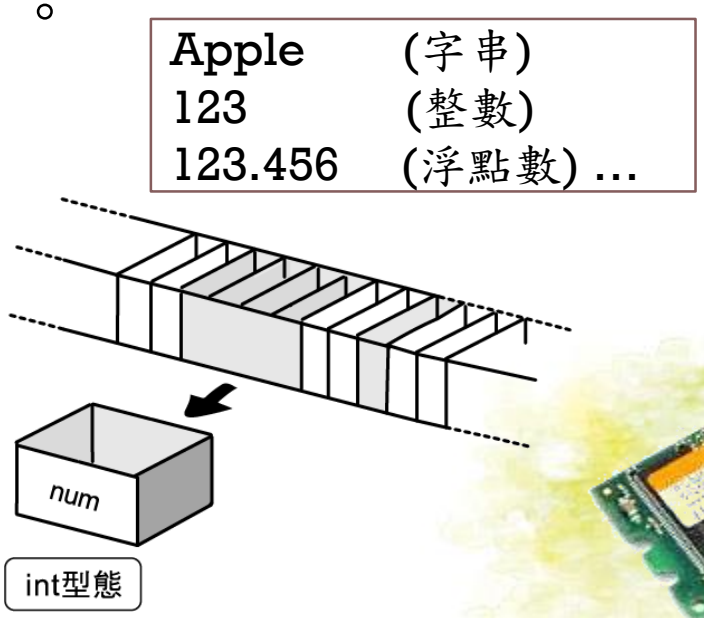
張傑帆 Chang, Jie-Fan

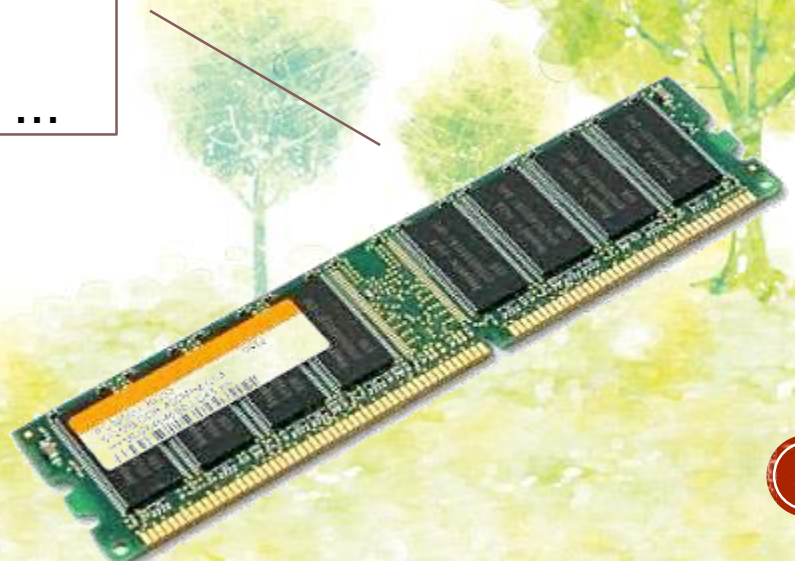
OUTLINE

- 變數
- 賦值
- 運算式、運算子及運算元

變數

- 在電腦裡，所有資訊都會放在記憶體（Random Access Memory, RAM）裡面，然後才進行執行與存取
- 要在程式語言中使用各種資訊或是資料型態，都必須要跟系統取得記憶體空間
- 在Python 裡使用變數就是一種跟系統要記憶體空間的動作。





變數

- 在Python 裡面的所有東西都被稱為物件（Object）
- 物件就好像是生活中各式各樣的東西，如桌椅、水、人、交通工具等等，是一種統稱。
  - 藉由變數來代表各種物件，就好像學生在一個班級裡會有學號，以及人會有身分證字號是一樣的意思。
  - 不同的變數可以代表同一個物件，就好像是學生的學號、名字、綽號、身分證字號、駕照號碼都可以代表同一個人。

變數

```
>>> 變數=123
>>> print (變數)
123
>>>
```

- 由於3.x 版以後都採用Unicode 編碼，因此理論上只要不是運算子、關鍵字或者特殊符號，基本上都可以拿來當變數名稱，因此中文當然也可以，但不建議。
- 一般只會使用英文字母、底線、數字等國際通用的文字與符號來當作變數名稱。
- 因為Python 使用UTF-8 編碼，所以看得懂中文，但是其他國家的工程師卻未必看得懂，故建議您盡量使用國際通用的英文命名變數，方便國際間資源流通、彼此合作。
- 建議養成良好的命名習慣，這樣可以讓自己或別人在閱讀程式碼時更為快速與便利。
- 盡可能的使用英文單字或縮寫為變數命名，對於變數所代表的意義就可以一目了然，但也不要太冗長。
- **變數也不能以數字開頭來命名。**

變數VARIABLES

- 變數的命名
  - 以英文字母a-z或A-Z或是_為開頭
  - Ex: first-name　　不行
  - Ex: 2x　　不行數字開頭
  - Ex: first_name　　可以
  - Ex: fruit, Fruit　　大小寫視為不同
- Python關鍵字
- 刪除變數
  - del 變數

關鍵字

- 就是在Python 裡面有特殊含意的字。
- 這些字在Python 裡面有其所代表的功能或特殊意義，因此在命名時要特別注意千萬不可以使用到這些字。
- 另外，Python 裡面內建的函數也是關鍵字，例如，print、input等等。以下將Python 的33 個關鍵字列出以供參考。
- 請注意，字母有大小寫之分喔！

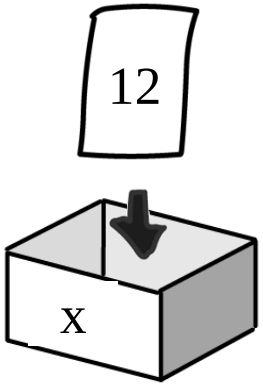
False	assert	del	for	in	or	while
None	break	elif	from	is	pass	with
True	class	else	global	lambda	raise	yield
and	continue	except	if	nonlocal	return	
as	def	finally	import	not	try	

識別字

- 識別字為寫程式時依需求自行定義的名稱
- 包括變數(variable)、函數 (function)、類別 (class) 等，皆為使用自行定義的識別字。
- 除了關鍵字之外，Python 可用任何 Unicode 編碼的字元當作識別字。
- 習慣上識別字的命名仍是以下列為主：
  - 英文字母大寫 A-Z(\u0041-\u005a)
  - 小寫 a-z (\u0061-\u007a)
  - 底線符號 (_, \u005f)
  - 數字 0-9 (\u0030-\u0039)
- 可用一種叫做「駝峰式命名法」的方式來進行命名。

```
studentName
itemPrice
computerScreenSize
book_title
employee_salary
my_student
```

賦值



- 「賦值/指定」是將特定資料值指派給變數
- **「=」不是數學邏輯，而是賦值/指定的意思！**

```
x=12
```

- 上面這個運算式的意思就是將12 這個數值賦予給**x** 這個變數。
- 從這行運算式之後，**x**的內容即裝著12，直到**x** 被賦予其他值為止。
- **Python 的變數又可分成全域變數和區域變數。**

賦值

- 一般而言，Python 屬於直譯式語言的一種。
- 在使用變數時並不需要事先宣告其型態；
- 但是，當一個變數第一次出現時仍必須給予一個初始值。
- 變數所代表的是物件參照，可賦予數值，字串，或者集合型態的資料型態，如set、list 等。
- 也可以在一行程式碼中一次賦值給多個變數，如下所示：

```
01  integer1 = interger2 = integer3 = 10
                    # 多個變數可以同時賦予相同之值做連續指定
02  string1, float1 = 'test', 12.0
                    # 多個不同型態的變數也可以分別賦予不同的值
03  print(integer1, integer2, integer3, string1, float1, sep=',')
04  x, y = y, x          #這樣可以簡單的交換其數值
```

```
>>>
10, 10, 10, 'test', 12.0
>>>
```

存取不存在的變數

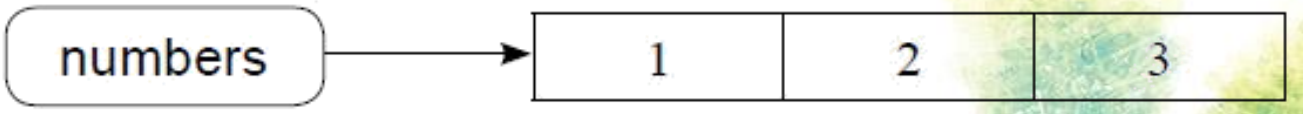
在使用變數前必須要先指定給它值，不然會無法使用且造成錯誤

```
>>> y
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in -toplevel-
    y
NameError: name 'y' is not defined
>>> y = 3
>>> y
3
```

賦值

- 當建立numbers 這個變數，並給予[1, 2, 3] 這個值，就好像建立出一個叫numbers 的框框，它會指向一組具有
[1, 2, 3] 值的list

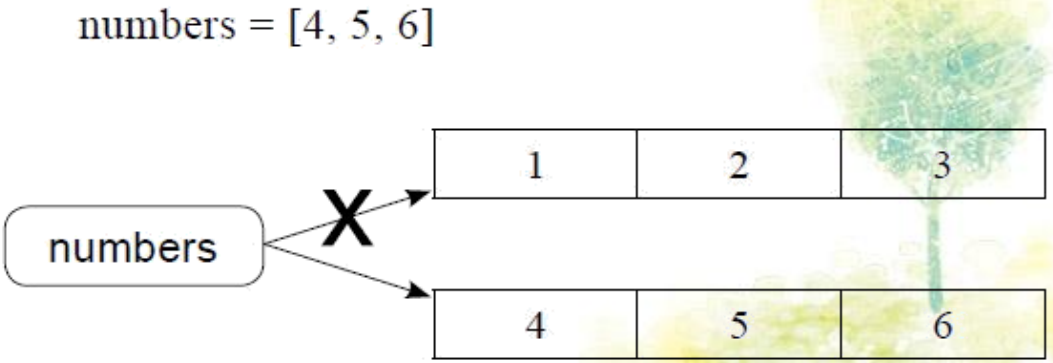
numbers = [1, 2, 3]



numbers 賦值時在記憶體中的可能排列方式

賦值

- 如果再執行numbers=[4, 5, 6]
- 就會建立產生另一組值為[4, 5, 6] 的list
- 並且讓numbers 這個變數指向[4, 5, 6]
- Python 的賦值其實比較像是「指標」，每當要將值賦予給變數時，就是讓變數指向要賦予的值



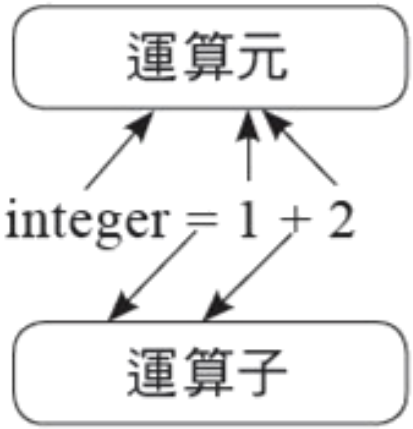
numbers 再次賦值時在記憶體中的可能排列方式

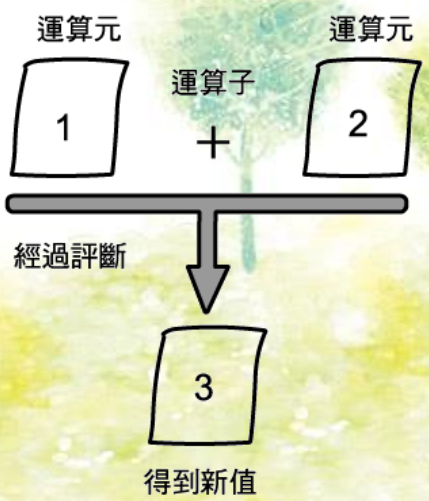
賦值

- 執行完numbers = [4, 5, 6] 之後
- 剛剛[1, 2, 3] 的這個list 就沒有變數指向它
- 這個list 就在垃圾回收機制的候選回收名單之中。
- 新增物件需要跟**系統**索取記憶體空間
- 沒有被使用(指向) 到的物件，Python 就會自動將其回收，以清理出更多可用的記憶體空間。

運算式、運算子及運算元

- 大部分的程式碼都是由判斷式及運算式組成的。
- 但此時先不談判斷式，我們先了解一下運算式。
- 1 + 1 = 2 這種國小就有教過的式子就是運算式。
- 運算式是由運算子及運算元所組成，
- 運算子就像是這個運算的種類（如+, = 等）
- 運算元則是要被用來運算的資料（如1, 2, integer 等）





運算式、運算子及運算元

- 運算子可以分成：
  - 算術運算子
  - 指定運算子
  - 比較運算子
  - 邏輯運算子(又稱布林運算子)
  - 其他運算子

算術運算子

- 就是一般數學式中常用的加減乘除

運算子	意義
+	加法運算子，執行兩運算元之加法。
-	減法運算子，執行兩運算元之減法。
*	乘法運算子，執行兩運算元之乘法。
**	指數運算子。
/	除法運算子，執行兩運算元之除法。
//	整數除法運算子，出來的結果會自動取整數
%	模數運算子，取餘數用

算術運算子

```
01  print(2 + 2)          # 加法運算子
02  print(2 - 2)          # 減法運算子
03  print(6 * 6)          # 乘法運算子
04  print(6 ** 6)         # 指數運算子
05  print(7 / 2)          # 除法運算子
06  print(7 // 2)         # 整數除法運算子
07  print(120 % 7)        # 模數運算子
```

```
>>>
4
0
36
46656
3.5
3
1
>>>
```

算術運算子

- **範例1**

  到市場並拿出了5000 精靈幣，「這個是我們世界的貨幣，挑選一套可在比賽當天穿的服裝採買一番吧！」

- **程式碼**

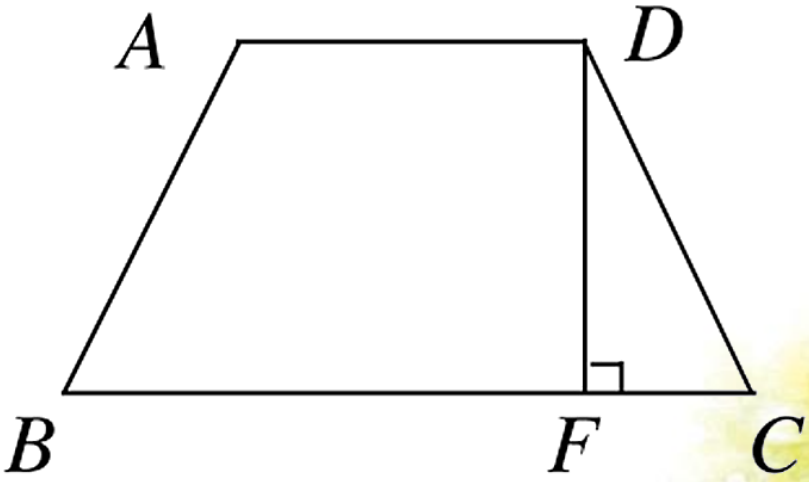
```
01  elfCoins = 5000        # 一開始有5000 精靈幣
02  elfCoins = elfCoins - (1000 + 1500)
        # 買了價值1000 精靈幣的上衣和1500 精靈幣的披風
03  elfCoins = elfCoins - 2000 * 0.75
        # 2000 精靈幣的帽子和鞋子組合，特價75 折
04  print(“ 還剩下”, elfCoins, “ 精靈幣”)
        # 印出剩餘精靈幣的數量
```

- **執行結果**

```
>>>
還剩下 1000.0 精靈幣
>>>
```

小練習

- 請算出此梯型面積
- 上底：10cm
- 下底：15cm
- 高：7cm



關係運算子

- 示範程式碼

```
01  print( 2 == 2 )        # 相等
02  print( 2 != 2 )        # 不相等
03  print( 6 > 4 )         # 大於
04  print( 4 < 6 )         # 小於
05  print( 6 >= 3)         # 大於等於
06  print( 6 <= 3)         # 小於等於
```

- 執行結果

```
>>>
True
False
True
True
True
False
>>>
```

關係運算子

- 比較運算子可以用在判斷變數的值是否在一個範圍之內，例如：
- 示範程式碼

```
01  integer = 7
02  print( 5 < integer <= 10)
```

- 執行結果

```
>>>
True
>>>
```

關係運算子

- 除了and、or、not 這三個關鍵字被拿來當作邏輯運算子外，還有is 與in 這兩個關鍵字也是。
- is 是判斷兩個運算元是否「本質上」相等，並回傳bool 值
- 而所謂「本質上」的相等，若指的是變數，則代表是否指向同一個物件。
- 故兩個變數可以有完全相同的值，但是卻指向不同的物件，例如：

關係運算子

- 示範程式碼

```
01  list1 = [1, 'test']      # 將一樣的值
02  list2 = [1, 'test']      # 賦予到兩個不同的變數上面
03
04  print(list1 == list2)  # 測試list1 跟list2 的值是否相等
05  print(list1 is list2)    # 測試list1 跟list2 的指向目標物件是否為同一個
06
07  list1 = list2            # 將list1 指向list2 所指向的物件
08
```

關係運算子

- 示範程式碼

```
09  print(list1 == list2)   # 測試list1 跟list2 的值是否相等
10  print(list1 is list2)    # 測試list1 跟list2 的指向目標物件是否為同一個
11
12  string1 = 'test'         # 同樣將一樣的值
13  string2 = 'test'         # 賦予到兩個不同的變數上面
14
15  print(string1 == string2)    # 測試string1 跟string2 的值是否相等
16  print(string1 is string2)    # 測試string1 跟string2 指向目標物件是否為同一個
```

關係運算子

- 執行結果

```
>>>
True
False
True
True
True
True
>>>
```

關係運算子

- 在將test 這個內容賦值給string1 跟string2 時，照理說string1 跟string2 只有值一樣，所指向的物件應該是不同的
- 但是Python 為了節省記憶體空間，因此會將使用比較簡單的資料型態且其內容相同的不同變數，在短時間內指向同一個物件。
- 這便造成上面list 在測試時所指向物件雖然不同，string卻相同的情形。

關係運算子

- in 的功能則是判斷一個元素是否為一集合的元素，適用範圍包含字串、集合、清單、序對和字典。
- 除了字串外，其他幾個資料型態都屬於容器資料型態，類似集合的概念，所以可以使用in 來判斷
- 但是字串也可以使用in，原因是因為字串可以視為由一連串長度為1 的字串所組成的長字串
- 如此一來，字串也就具有類似集合的概念了，例如：

關係運算子

```
01  tuple1 = (1, 2, '3', '4', [1, 2])   # 將多種不同型態的元素賦值給tuple1
02  list1 = [5, 6, '7', '8']            # 將多種不同型態的元素賦值給list1
03  dictionary1 = {9 : 'b', 0 : 'd'}     # 將9、0 當作key 值，對應的值為b、d
04  set1 = {'x', 'y', 'z'}               # 將'x', 'y', 'z' 當作元素賦值給set1
05  string1 = 'abcdefghijklmn'           # 建立string 字串
06
07                                       # 開始測試in
08
09  print(1 in tuple1, 3 in tuple1, 9 not in tuple1)
10  print(5 in list1, 7 in list1, 'a' not in list1)
11  print('b' in dictionary1, 9 in dictionary1)
12  print('x' in set1, 10 in set1)
13  print('z' in string1, 'c' in string1)
```

```
>>>
True False True
True False True
False True
True False
False True
>>>
```

關係運算子

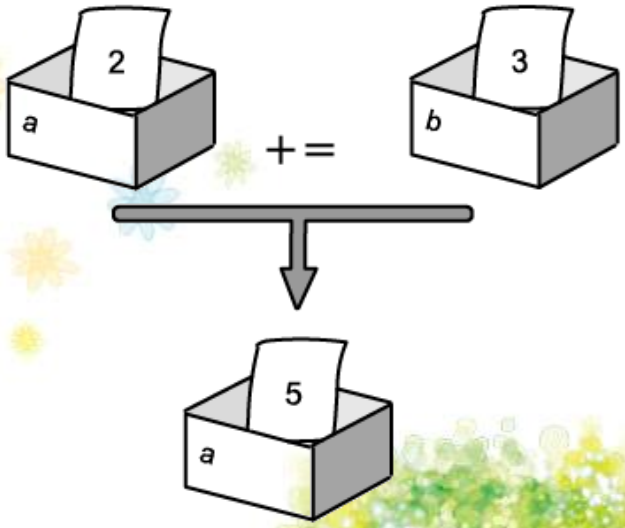
- **範例**

一個巷口，兩側都是牛肉麵店。山珍麵店的老闆說：「我這邊的牛肉麵原價一碗500 精靈幣，現在打55 折給你。」而海味麵店的老闆則說：「我這邊的牛肉麵一碗400 精靈幣，再打6折給你。」小明認為山珍麵店比較便宜，因為它打了55 折。小美說:「別急，讓我來算給你看。」

```
01 land = 500 * 0.55        # 山珍麵店打折後的價格
02 sea = 400 * 0.6          # 海味麵店打折後的價格
03 print(land < sea)        # 判斷山珍麵店打折後的價格是否低於海味麵店打折後的價格
```

```
>>>
False
>>>
```

指派運算子



- 指派運算子是將運算過後的結果儲存在某個變數中
- 使用復合指派運算子的好處是可以讓運算式變簡單
- 如下表所示

運算子	例子	意義
=	a = b	將b的值賦予a
+=	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

範例

使用復合指定運算子
把計算剩餘精靈幣的程式加以簡化的方法吧

```
01 elfCoins = 5000          # 一開始有5000 精靈幣
02 cost = 0                 # 預設總支出為0
03 cost += 1000 + 1500      # 買了價值1000 精靈幣的上衣和1500 精靈幣的披風
04 cost += 2000 * 0.75      # 2000 精靈幣的帽子和鞋子組合，特價75 折
05 sea = 400                # 海味麵店的牛肉麵原價
06 sea *= 0.6               # 牛肉麵打折後的價格
07 cost += sea * 2          # 買了兩碗海味麵店的牛肉麵
08 elfCoins -= cost         # 剩餘的精靈幣 = 原來的精靈幣數量 - 總支出
09 print("還剩下", elfCoins, "精靈幣")  # 印出剩餘精靈幣的數量
```

```
>>>
還剩下 520.0 精靈幣
>>>
```

小練習

- 請列出一半徑為100公分的圓形之圓週率，半徑，圓周長及圓面積

- Hint:
- import math
- math.pi

```
>>>
圓週率： 3.141592653589793
半徑： 100
圓周長： 628.3185307179587
圓面積： 31415.926535897932
>>>
```

其他運算子

- 除了上述所提到的四大類運算子之外，還有一些比較少用，且無法歸類到上述四大類運算子之中，故另外以下表為各位介紹。

運算子名稱	運算子	意義
逗號運算子	,	分隔變數、資料集裡的元素等等
分號運算子	;	分隔運算式
點號運算子	.	存取類別、模組的方法或屬性
小括弧運算子	( )	定義tuple、函式/方法呼叫
中括弧運算子	[ ]	定義list、序列形態的索引符號
大括弧運算子	{ }	定義dict
冒號運算子	:	控制條件後的分隔符號或辭典元素之配對

其他運算子

```
01  import math                 # import math 模組
02  a, b = (10, 20)             # 定義a 為10 而b 為20
03  x = 3; y = 4; z = 5         # 定義x 為3，y 為4，z 為5。
04                              # 這兩種定義方式都可以同時定義多個變數
05  print(math.pi)              # 呼叫math 的pi 這個屬性，也就是圓周率
06  tuple1 = (1, 2, 3, 4)       # 定義tuple1 為(1, 2, 3, 4)
07  list1 = [5, 6, 7, 8]        # 定義list1 為[5, 6, 7, 8]
08  dict1 = {'a' : 1, 'b' : 2}  # 定義dict1 中'a' 為1，'b' 為2
09  print(list1[:3])            # 印出list1 裡面第一個元素到第三個元素
```

```
>>>
3.141592653589793
[5, 6, 7]
>>>
```

運算子的優先順序

運算子	說明
(…)、[…]、{…}	tuple、list、dict
a [i]、a [i:j]、a.b、a (…)、a.b (…)	內含中小括號之呼叫
+k、-k、~k	正負數及補數
a * b、a / b、a % b、a // b	乘法、除法、取餘數、整數除法
a + b、a – b	加法、減法
a << b、a >> b	位移運算
a & b	AND位元運算
a ^ b	XOR位元運算
a | b	OR位元運算

運算子的優先順序

運算子	說明
<、>、<=、>=、==、<>、!=、is、is not、in、not in	比較運算、本體測試、序列元素關係測試
not a	not邏輯運算
a and b	and邏輯運算
a or b	or邏輯運算