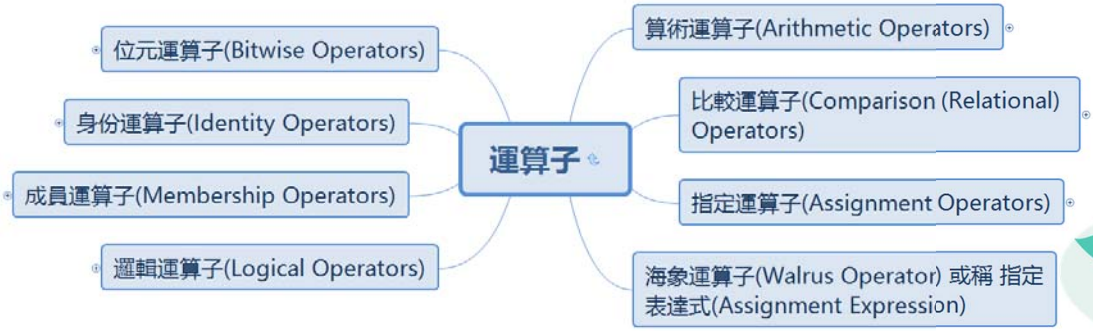
Python 運算子

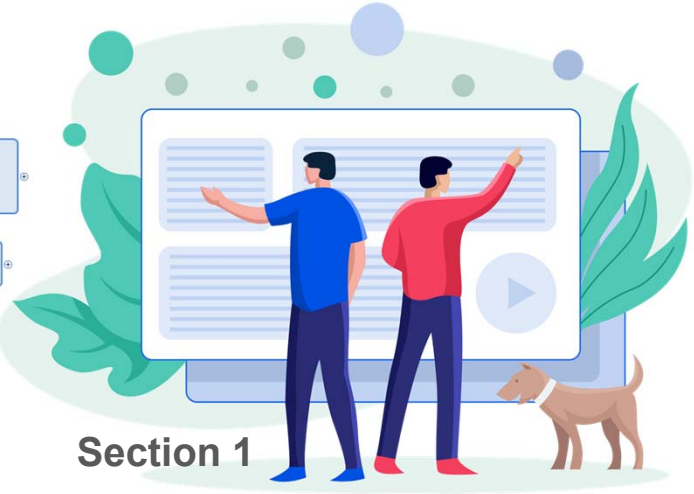
學習目標

- 比較運算子(Comparison (Relational) Operators)
- 邏輯運算子(Logical Operators)
- 身份運算子(Identity Operators)
- 位元運算子(Bitwise Operators)

Python比較與邏輯運算子

- 使用比較運算子(Comparison (Relational) Operators)
- 使用邏輯運算子(Logical Operators)





比較運算子 -1

運算子	說明	舉例 假設 x=2
==	等於	x == 2 (True)
!=	不等於	x != 2 (False)
<	小於	x < 5 (True)
>	大於	x > 5 (False)
<=	小於等於	x <= 2 (True)
>=	大於等於	x >= 2 (True)

- 比較運算的結果為 True/False (注意第一個字母大寫，如果是小寫，就代表是另一個變數，而不是 布林 的結果)
- 留意 == 與 = 的不同點
- ==：比較運算子
- =：指定運算子

比較運算子 -2

- 比較運算可以任意的串在一起
- x < y <= z 相當於 x < y and y <= z，差別在於 y 只評估一次
  - 如果 x 沒有小於 y，則不會評估 z 的值
- 範例
  - >>> 4 <= 6 > 7
    False
  - >>> 5 < 6 > 3
    True
  - >>> 5 > 6 > 3
    False
- Python 常會直接使用數學公式的特性與想法
- Python允許多個比較運算子可以一起比較，而且也顧及到效率：在運算完所有的比較運算子之前，如果已確認結果，就不會花時間做不需要的比較運算

邏輯運算子 -1

運算子	說明	舉例　假設 x=True, y=False
and	假如兩個運算元都是非零，則結果為真	(x and y) is False
or	假如有任一個運算元是非零，則結果為真	(x or y) is True
not	反向運算，真變假，假變真	not (x and y) is True

- 邏輯運算子(Logical operator)的對象是布林(Boolean)，所以又可以稱為「布林運算子」
- Python的「邏輯運算子」不使用符號，僅使用文字；符號(& | ...等)是另一種運算子，稱為位元運算子，對象是二進制的bit位元

邏輯運算子 -2

運算子	說明	注意事項
x or y	if x is false, then y, else x	(1)
x and y	if x is false, then x, else y	(2)
not x	if x is false, then True, else False	(3)

- 注意事項：
  - (1) 快速運算，只有 x 結果為假才會評估 y 的值
  - (2) 快速運算，當 x 結果為真時才會評估 y 的值
  - (3) “not” 運算子的優先順序較低，所以 “not a == b” 會被解譯為 “not (a == b)”，且 “a == not b” 是語法錯誤
  - (4) 在做邏輯的運算時，如果已知道運算結果，那麼未做的邏輯運算就不會繼續花電腦資源去做計算

Operator precedence 運算子優先順序 (上方較優先)

Operator 運算子	Description 描述說明
(expressions...), [expressions...], {key: value...}, {expressions...}	括號、元組、串列、字典、集合表達
x[index], x[index:index], x(arguments...), x.attribute	Subscription (index), slicing, call, attribute reference
await x	Await expression
**	算數運算子：次方 Exponentiation
+x, -x, ~x	正號, 負號, 算數運算子 NOT
*, @, /, //, %	乘法、矩陣乘法、除法、地板除法、取餘數
+, -	算數運算子
<<, >>	位元運算子 Shifts
&	位元運算子 AND
^	位元運算子XOR
|	位元運算子OR
in, not in, is, is not, <, <=, >, >=, !=, ==	成員運算子、身分運算子、比較運算子
not x	邏輯運算子 NOT
and	邏輯運算子AND
or	邏輯運算子 OR
if – else	條件表達式
lambda	匿名函式表達式
:=	指定敘述

Examples

- a < 10 and b > 30　　相當於　　(a < 10) and (b > 30)

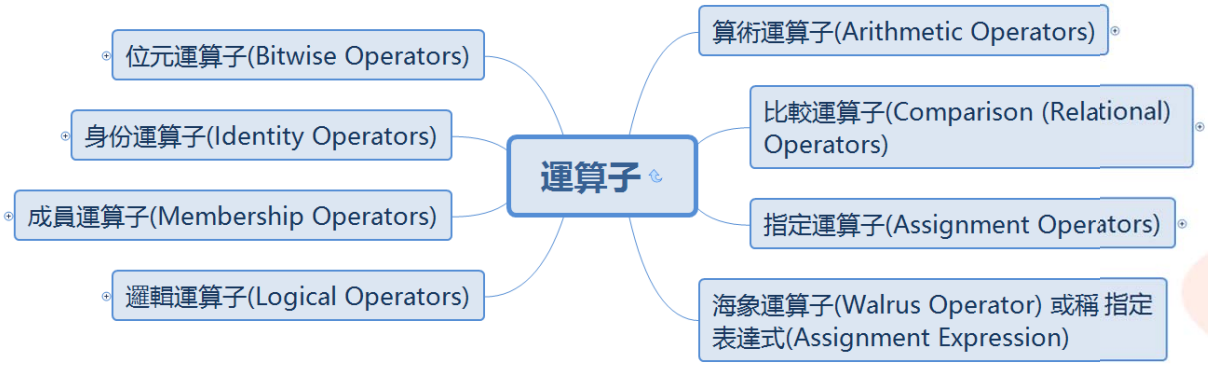
```
>>> 20 + 4 * 10
60
>>> (20 + 4) * 10
240
>>> 2 * 3 ** 4 * 5
810
>>> 2 * 3 ** (4 * 5)
6973568802
```

比較與邏輯運算子 (Demo)

- 使用 Python 互動式環境做比較運算、邏輯運算

Python身分運算子

使用身分運算子(Identity Operators)



身份運算子

運算子	說明	舉例
is	如果運算子兩邊的變數都指向同一個物件，則結果為真，否則結果為假	x is y，如果 id(x) == id(y)，結果為 True
is not	如果運算子兩邊的變數都指向同一個物件，則結果為假，否則結果為真	x is not y，如果 id(x) != id(y)，結果為 True

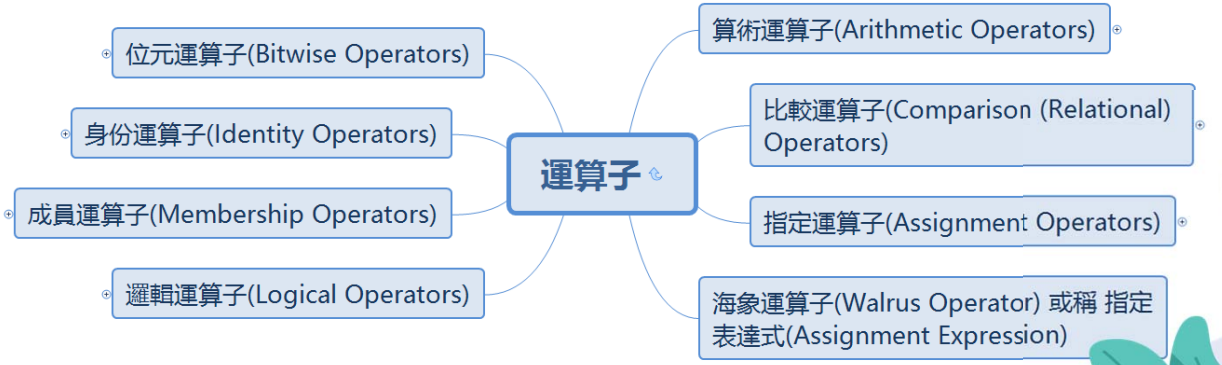
- Python 內建的函數 id() 傳回一個唯一的整數值代表記憶體位址，用來辨識物件
- Identity 運算子比較兩物件的記憶體位址是否相同

身分運算子 (Demo)

- 使用 Python 互動式環境做身份運算

Python位元運算子

使用位元運算子(Bitwise Operators)



位元運算子 -1

- 位元運算是一個位元一個位元逐一運算(bit-by-bit operation)

p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

位元運算子 -2

運算子	說明	舉例
&	位元 AND 運算	(a&b) = 12，二進位為 0000 1100
|	位元 OR 運算	(a|b) = 61，二進位為 0011 1101
^	位元 XOR 運算	(a^b) = 49，二進位為 0011 0001
~	1的補數運算 (Ones Complement Operator)	(~a) = -61, 二進位為 1100 0011， 求二的補數得到 -61
<<	位元左移運算 (邏輯移位)	a<<2 = 240，二進位為 1111 0000
>>	位元右移運算 (算數移位)	a>>2 = 15，二進位為 0000 1111

假設 a=60, b=13

位元運算子 -3

AND

A (60)	0	0	1	1	1	1	0	0
B (13)	0	0	0	0	1	1	0	1
A & B	0	0	0	0	1	1	0	0

OR

A (60)	0	0	1	1	1	1	0	0
B (13)	0	0	0	0	1	1	0	1
A | B	0	0	1	1	1	1	0	1

XOR

A (60)	0	0	1	1	1	1	0	0
B (13)	0	0	0	0	1	1	0	1
A ^ B	0	0	1	1	0	0	0	1

位元運算子 (Demo)

- 使用 Python 互動式環境做位元運算

本章重點精華回顧

- 熟悉 Python 四種種運算子
  - 比較運算 (Comparison (Relational) Operators)
    針對數值或是變數數值的比較
  - 邏輯運算 (Logical Operators)
    對象是True或False的邏輯比較
  - 身份運算 (Identity Operators)
    針對變數的記憶體位址的比對
  - 位元運算 (Bitwise Operators)
    針對二進制bits值的比較

Lab: Python的運算子

- Lab01: 使用互動式環境熟悉運算子
  - 比較運算、邏輯運算、身份運算、位元運算

Lab01: 使用互動式環境熟悉運算子

- 啟動Python互動式執行環境，做以下練習。
  - >>> a, b = 7, 5
  - >>> a, b
  - >>> a > b, a >= b
  - >>> a < b, a <= b
  - >>> a == b, a != b
  - >>> 8 > 7 > 6
  - >>> 6 < 8 > 7
  - >>> a, b = 0, 9
  - >>> a, b
  - >>> a > 10 and b < 15
  - >>> a > 10 or b <15
  - >>> not a
  - >>> a and b
  - >>> a or b
  - >>> not b
  - >>> x = 5
  - >>> id(x)
  - >>> y = 5
  - >>> id(y)
  - >>> x == y
  - >>> x is y
  - >>> id(x) == id(y)
  - >>> y = 20
  - >>> id(y)
  - >>> x == y
  - >>> x is y
  - >>> id(x) == id(y)
  - >>> 19 & 11
  - >>> 19 | 11
  - >>> 19 ^ 11
  - >>> ~11
  - >>> 11 >> 2
  - >>> 11 << 4