科學計算程式設計實習

Python 變數、賦值、運算子

王昱景 Brian Wang
brian.wang.frontline@gmail.com

變數 Variable

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

- 在 Python 裡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
- 藉由變數來代表各種物件
- 不同的變數可以代表同一個物件
- 只要不是用來當運算子、關鍵字或者特殊符號，基本上都可以拿來當變數名稱

- 雖然變數名稱可以使用中文名，但在程式語言的習慣只會使用英文字母、底線、數字等國際通用的文字與符號來當作變數名稱
- 養成良好的命名習慣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，如此對於各個變數所代表的意義就可以一目了然

- 但也建議變數名稱不要太沉長
- 變數也不能以數字開頭來命名
- something
- myStudent
- stu_bio

Python 關鍵字

- 關鍵字是在 Python 裡有特殊含意的字
- 因此在命名時要特別注意不可使用關鍵字
- 另外 Python 的內建函數也是關鍵字
- 字母有大小寫之分

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	

- 關鍵字依功能分成五大類：
  - 常數：False、None、True
  - 型態定義：class、def
  - 控制陳述：as、assert、break、continue、del、elif、else、except、finally、for、global、if、nonlocal、pass、raise、return、try、while、with、yield

- 運算子：and、not、or、is、in、lambda
- 模組相關：from、import

- 對於變數的命名習慣建議全部使用小寫及底線來完成
- 或是以小寫英文字母開頭，之後每換一個單字，新單字的第一個字母都使用大寫字母
- 函數其命名規則跟變數也是相同的，差別只在於後面會加上一組圓括號
- 沒有特殊需求盡可能不要使用底線開頭來命名變數或函數

賦值 Assignment

- 賦值就是將特定資料值指派給變數
- x = 12
- 程式語言裡面的 “=” 不是數學邏輯，而是賦值的意思
- Python 的變數又可分成全域變數和區域變數

- 全域變數就是整個 Python 檔案都可以使用
- 變數只要賦值一次接下來所有使用到該變數名稱的程式碼變都會指向同一個物件
- 區域變數則是大多出現在迴圈與函數裡
- 如果一個變數首次被賦值的地方是放在被迴圈或函數所涵蓋的區域裡，那此變數的效力範圍就只存在該區域

- Python 屬於直譯式語言，因此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，未必一定要是數值型態的物件
- 可能是字串或者集合型態的資料型態

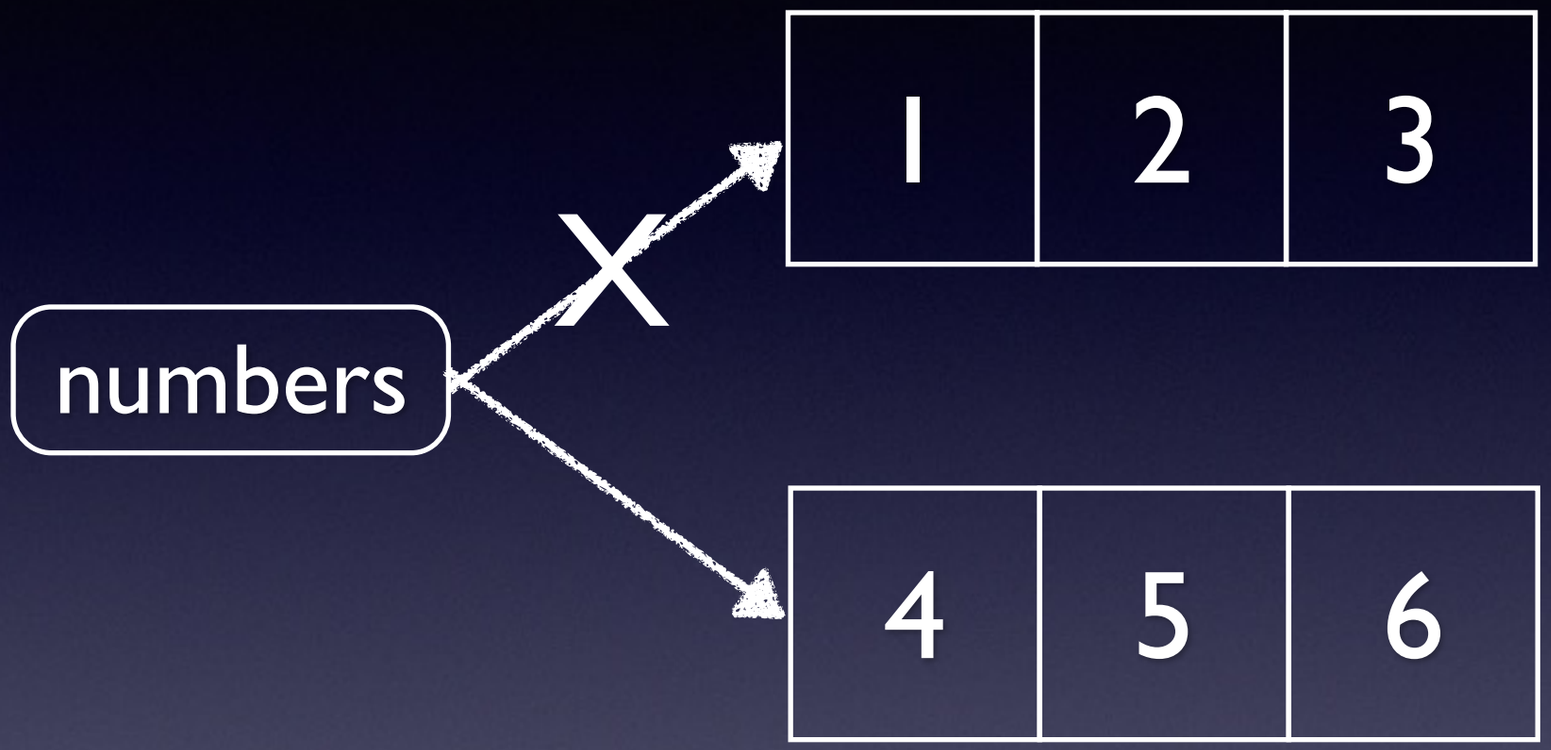
- 如果有多個變數，可以在一行程式碼中一次全賦值

```
integer1 = integer2 = integer3 =10
string1, float1 = “test”, 12.0
print(integer1, integer2, integer3, string1, float1)
```

numbers = [1, 2, 3]

- 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 = [4, 5, 6]



- 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]
- 賦值其實比較像是“指標”，每當亦將值賦予給變數時，就是讓變數指向要賦予的值
- 當執行完 numbers = [4, 5, 6] 之後，剛剛 [1, 2, 3] 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中
- 因此沒有被使用（指向）到的物件，Python 就會自動將其回收

運算式、運算子及運算元

- 大部份的程式碼都是由判斷式及運算式組成
- 運算式是由運算子及運算元所組成
- 運算子就像是運算的種類
- 運算元則是要被用來運算的資料

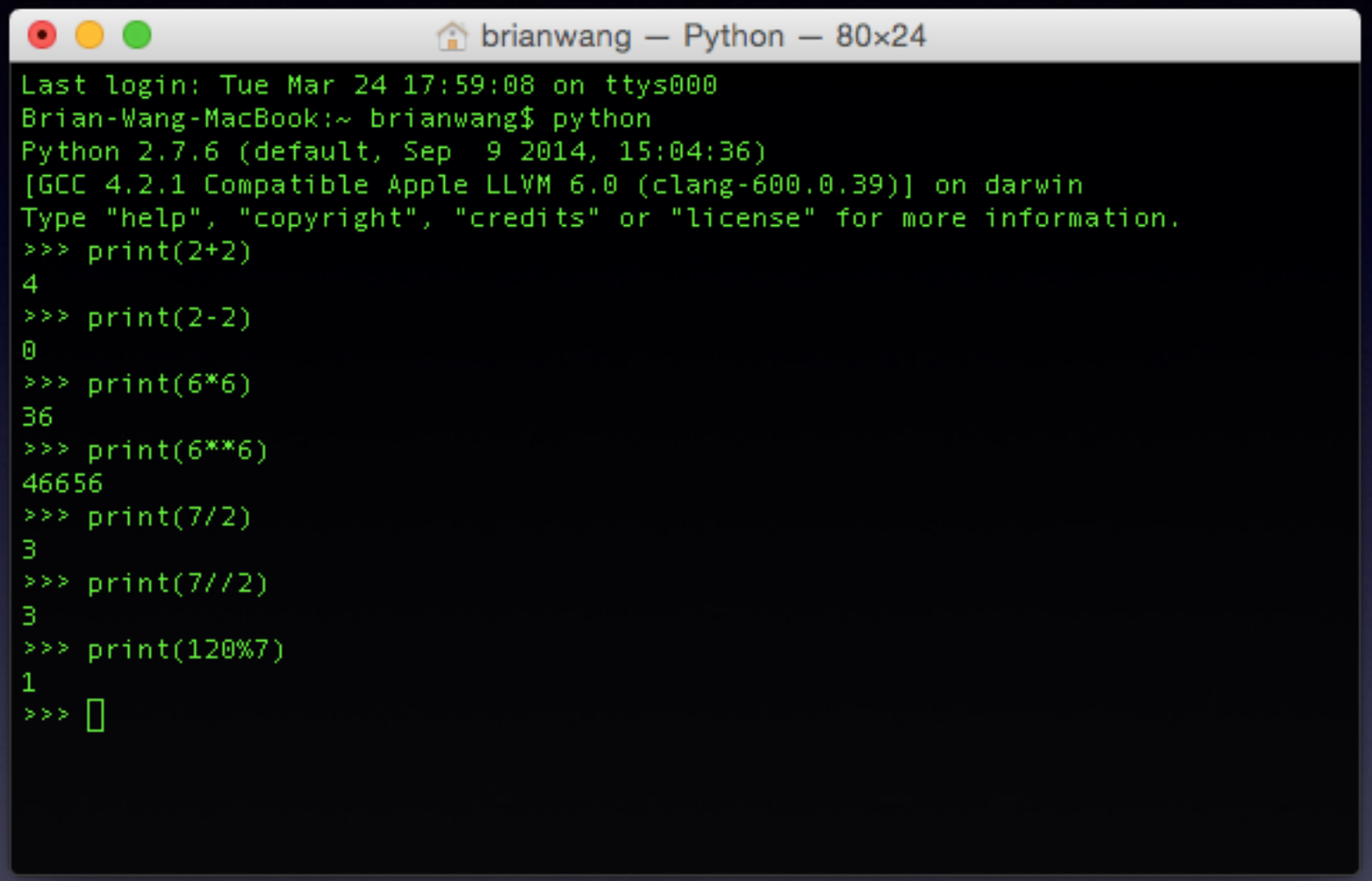
- integer = 1 + 2
- 數字是運算元
- “=”、”+”則是運算子
- 運算元除了是常數以外也可以是變數
- 甚至是其他運算式都可以拿來當運算元

- 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（布林運算子）
  - 其他運算子

算術運算子

- 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


brianwang — Python — 80×24
Last login: Tue Mar 24 17:59:08 on ttys000
Brian-Wang-MacBook:~ brianwang$ python
Python 2.7.6 (default, Sep 9 2014, 15:04:36)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.39)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print(2+2)
4
>>> print(2-2)
0
>>> print(6*6)
36
>>> print(6**6)
46656
>>> print(7/2)
3
>>> print(7//2)
3
>>> print(120%7)
1
>>>

- “ ** ” 就是次方，也就是所謂的指數
- “ // ” 是表示整數除法，也就是執行完除法之後的結果直接做無條件捨去小數部分
- “ % ” 百分號被稱為“模數”，是拿來取餘數用的

邏輯運算子

- and 為邏輯且，前後兩個運算元 (operand) 都為真，運算式 (expression) 才為真，
- or 為邏輯或，前後兩個運算元有一個為真，運算式就為真
- not 為邏輯否，也就是真變成假，假變成真

以下程式示範 and 的使用

```
1   a = 12
2   b = 22
3
4   if a > b and a != b:
5       print("a > b")
6
7   if a < b and a != b:
8       print("a < b")
```

由於 a < b 且 a != b ，所以印出 "a < b"

以下程式示範 or 的使用

```
1   a = 12
2   b = 22
3
4   if a > b or a != b:
5       print("a > b or a != b")
6
7   if a > b or a != b:
8       print("a < b or a != b")
```

由於是用 or 連接兩個運算式， a 與 b 既然不相等，於是兩個字串都會被印出

以下程式示範 not 的使用

```
1   a = 12
2   b = 0
3
4   if a:
5       print("a is true")
6
7   if b:
8       print("b is true")
9
10  if not a:
11      print("a is false")
12
13  if not b:
14      print("b is false")
```

- 對 Python 而言，所有 0 的值會是假，也就是 False ，所有非 0 值會是真，也就是 True ，所以

```
4   if a:
```

- 條件 (condition) 會成立， if 陳述 (statement) 底下的區塊 (block) 會被執行。同樣的

```
13   if not b:
```

- 條件也會成立

比較運算子

- 比較運算子有可稱為關係運算子
- 是將兩個運算單元拿來相互比較以得知兩者之間的關係
- “=”所代表的意義是賦值，所以原本相等的意義就以”==”雙等號來表示
- 經過比較運算子運算後所得的結果則是布林值

運算子	功能	範例
<	小於	a < b
>	大於	a > b
<=	小於等於	a <= b
>=	大於等於	a >= b
==	相等	a == b
!=	不相等	a != b

以下為整數型態 (integral type) 的例子

```
1  a = 12
2  b = 22
3
4  if a < b:
5      print("a < b")
6
7  if a <= b:
8      print("a <= b")
9
10 if a > b:
11     print("a > b")
12
13 if a >= b:
14     print("a >= b")
15
16 if a == b:
17     print("a == b")
18
19 if a != b:
20     print("a != b")
```

關係運算子所構成的運算式結果為 True 或 False ，如果為 True ， if 陳述 (statement) 底下縮排 (indentation) 的區塊 (block) 便會被執行

以下為浮點數型態 (floating-point type) 的例子

```
1   a = 22.0
2   b = 22.0
3
4   if a < b:
5       print("a < b")
6
7   if a <= b:
8       print("a <= b")
9
10  if a > b:
11      print("a > b")
12
13  if a >= b:
14      print("a >= b")
15
16  if a == b:
17      print("a == b")
18
19  if a != b:
20      print("a != b")
```

以下為真假字面常數 (literal) 的例子

```python
a = True
b = False

if a < b:
    print("a < b")

if a <= b:
    print("a <= b")

if a > b:
    print("a > b")

if a >= b:
    print("a >= b")

if a == b:
    print("a == b")

if a != b:
    print("a != b")
```

以下為字串 (string) 的例子

```
1  a = "A"
2  b = "a"
3
4  if a < b:
5      print("a < b")
6
7  if a <= b:
8      print("a <= b")
9
10 if a > b:
11     print("a > b")
12
13 if a >= b:
14     print("a >= b")
15
16 if a == b:
17     print("a == b")
18
19 if a != b:
20     print("a != b")
```

這裡比較 A 與 a 在 Unicode 編碼中的順序，
A 排在 a 的前面，所以 A 的編碼值較小

- is 用來判斷兩個物件 (object) 是否相等，也就是判斷由內建函數 (function) id() 回傳的 id 號碼是否一樣

```
a = 12
b = 12
c = 22

if a is b:
    print("a is b")

if a is c:
    print("a is c")
```

- 由此例可見出， a 與 b 雖然是不同的整數變數 (variable) ，但內含數值 (value) 相同
- 對 Python 而言，等於只建立一個整數 12 的物件，而 a 與 b 共同連結到這個物件
- 這樣的情況對字串 (string) 也受用，因為字串是不可變的 (immutable) ，因此相同內容的字串也會是相同的物件

- is 也可以和 not 連用，可判斷兩個是否為不同物件

```
1  a = 12
2  b = 12
3  c = 22
4
5  if a is not b:
6      print("a is not b")
7
8  if a is not c:
9      print("a is not c")
```

- 但是 is 與 is not 的使用需注意一點，對於可變 (mutable) 的資料型態，例如串列 (list) 就算有相同的內容，實際上仍會是不同的物件，例如

```
a = []
b = []

if a is b:
    print("a is b")

if a is not b:
    print("a is not b")
```

- 這是因為不可變的資料型態 (data type)，一旦建立該物件後，內容不可再做變更
- 因此 Python 處理這方面只建立單一的物件，以節省記憶體空間
- 而可變的資料型態，由於內容可以再做變更，因此無論建立多少相同內容的物件，這都會在記憶體上是不同的物件

指派運算子

- Python 最基本的指派運算子 (assignment operator) 為單一個等號 = ，這是用來將等號右邊的值拷貝給給左邊的變數 (variable) 資料
- 等號也可以跟其他運算子 (operator) 合用，會直接將結果儲存到原變數之中

運算子	功能	範例
=	指派	a = b
+=	相加同時指派	a += b
-=	相減同時指派	a -= b
*=	相乘同時指派	a *= b
**=	取指數同時指派	a **= b
/=	相除同時指派	a /= b
//=	整數相除同時指派	a //= b
%=	取餘數同時指派	a %= b
&=	位元且同時指派	a &= b
^=	位元互斥或同時指派	a ^= b
|=	位元包含或同時指派	a |= b
<<=	向左位移同時指派	a <<= b
>>=	向右位移同時指派	a >>= b

- 大體上可與指派運算子連用的為算術運算子 (arithmetic operator) 及位元運算子 (bitwise operator) ，以下程式示範算術運算子的使用

```
a = 1

a += 2
print(a)

a -= 1
print(a)

a *= 22
print(a)

a **= 3
print(a)

a /= 7
print(a)

a //= 5
print(a)

a %= 3
print(a)
```

- 以下程式示範與位元運算子的連用

```
a = 248

a &= 7
print(a)

a |= 192
print(a)

a ^= 63
print(a)

a <<= 3
print(a)

a >>= 5
print(a)
```