



湯明軒(Andrew)

析客數據有限公司 創辦人
中華R軟體學會 祕書長

Table of Contents

- [1 前言](#)
- [2 Python Shell](#)
- [2.1 Python Shell的加強版 IPython](#)
- [3 第一個 Python 指令](#)
- [4 說明文件](#)
- [5 套件安裝與載入](#)
 - [5.1 套件管理程式pip](#)
 - [5.2 套件載入](#)
- [6 變數 Variables](#)
- [7 關鍵字 Keyword](#)
 - [7.1 常數](#)
 - [7.2 型態定義詞](#)
 - [7.3 控制陳述](#)
 - [7.4 運算子](#)
 - [7.5 模組相關](#)
- [8 識別字](#)
 - [8.1 命名習慣](#)
- [9 運算子](#)
 - [9.1 算術運算子](#)
 - [9.2 位移運算子](#)
 - [9.3 位元運算子](#)
 - [9.4 關係運算子](#)
 - [9.5 指派運算子](#)
 - [9.6 分隔符號](#)
- [10 Python內建資料型別](#)
 - [10.1 數值型別 Numeric types](#)
 - [10.2 序列型別](#)
 - [10.2.1 字串 Strings](#)
 - [10.2.2 範圍 Range](#)
 - [10.2.3 串列 List](#)
 - [10.2.4 序對 Tuple](#)
 - [10.2.5 透過「：」控制序列取值](#)
 - [10.3 集合 Set](#)
 - [10.4 字典 Dict](#)
 - [10.5 型別轉換 Type casting](#)
- [11 Python控制敘述](#)
 - [11.1 判斷控制](#)
 - [11.2 迴圈控制](#)
- [12 Python自定義函數](#)
- [13 Jupyter介紹及操作](#)
 - [13.1 Jupyter簡介](#)
 - [13.2 常用快捷鍵](#)
 - [13.3 魔法命令\(Magic Commands\)](#)
 - [13.4 Line magic](#)
 - [13.5 Cell magic](#)
- [14 補充](#)
- [15 Reference](#)

前言

- Why python?
 - 活躍的社群
 - 開放源始碼
 - 深思熟慮的設計
 - 眾多的第三方資源

Python Shell

進入互動模式的方法，是打開終端機，輸入下面的指令：

```
python
```

按下 Enter 鍵後，應該會看到類似下面的內容

```
Python 3.5.2 (...) Type "copyright", "credits" or "license" for more information. >>>
```

這代表 Python 已經啟動，進入了 **Python shell**。

最後一行的 >>> 代表它已經準備接受你的下一個命令。

輸入 `exit()` 並按 Enter，就會退出 **Python shell**，回到原本的提示列。

Python Shell的加強版 IPython

進入互動模式的方法，是打開終端機，輸入下面的指令：

```
ipython
```

按下 Enter 鍵後，應該會看到類似下面的內容

```
Python 3.5.2 |Anaconda 4.2.0 (x86_64)| (...) Type "copyright", "credits" or "license" for more information. IPython  
5.1.0 -- An enhanced Interactive Python. ? -> Introduction and overview of IPython's features. %quickref ->  
Quick reference. help -> Python's own help system. object? -> Details about 'object', use 'object??' for extra  
details. In [1]:
```

最後一行的 In [1]: 代表它已經準備接受你的下一個命令。

練習一下

第一個 Python 指令

我們來讓 Python 執行幾個指令看看。試著輸入 `2 + 3`，然後按 Enter。你會看到下面的結果：

```
>>> 2 + 3
```

這就是 Python 互動模式的標準流程。你問 Python 一個問題，然後它回答你。

Python 畢竟是電腦程式，這種計算當然難不倒它。試試看一些更複雜的計算吧！下面是幾個例子：

- $987 * 543$
- $77 / (9 - 4) * 7$
- $6 ** 8$

它們各代表什麼意思？多試幾個數字，看看你能不能自己找到答案。

考考你

- 假設美金匯率31.3、日圓匯率0.273，那麼520美金可以兌換多少日圓？

說明文件

help(len)?requests.get?requests

套件安裝與載入

套件管理程式pip

- pip 操作命令 需在**CMD(Command Line)**下安裝，而非**python shell**
 - pip list # 看目前系統有安裝哪些套件
 - pip search mysql # 搜尋相關套件
 - pip install package # 安裝套件
 - pip uninstall package # 移除套件
 - pip show --files package # 秀套件檔案列表
 - pip list --outdated # 列出過期套件
 - pip install --upgrade package # 升級

考考你

- 請安裝folium套件

套件載入

- import, from, as
- 大部份時候建議使用import不使用from，環境會比較乾淨
 - 例如下面的requests被import之後，使用上還是要requests.xxx
 - 若使用了from requests import get，使用上可以直接 get，容易與其他同名函數造成混亂
- as只是使用上方便而已，例如：
 - pandas as pd, numpy as np (只是慣例，想怎麼as很隨意的)

In [113]:

```
import requests
res_example = requests.get('http://www.example.com/')
print(res_example.text[:500])

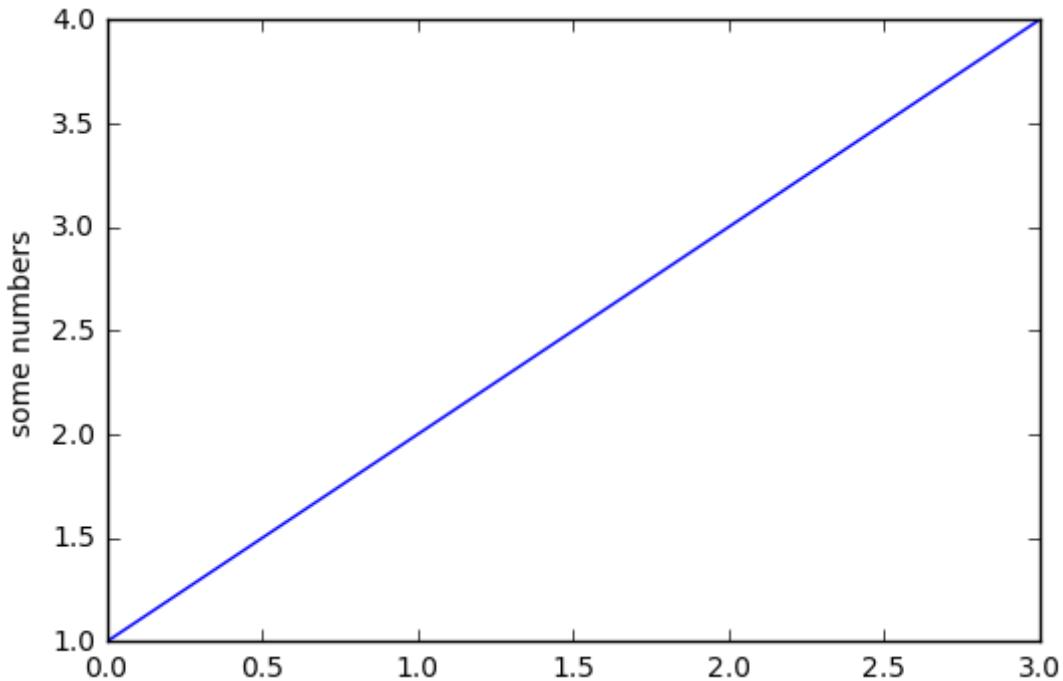
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-
8" />
    <meta name="viewport" content="width=device-width, initial-scale
=1" />
    <style type="text/css">
body {
    background-color: #f0f0f2;
    margin: 0;
    padding: 0;
    font-family: "Open Sans", "Helvetica Neue", Helvetica, Aria
l, sans-serif;

}
div {
    width: 600px;
    margin:
```

In [114]:

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.show()
```



變數 Variables

有時候，我們會想在很多地方使用同一個東西。但我們要怎麼告訴 Python，我們想要什麼東西？



In [115]:

```
USD = 31.232  
USD * 100
```

Out[115]:

```
3123.2
```

既然是「變」數，代表我們可以改變它的內容：

In [116]:

```
USD = 30.666  
USD * 100
```

Out[116]:

```
3066.6
```

但如果我們問了 Python 不認得的變數名稱，會怎麼樣呢？

In [117]:

```
TWD
```

```
-----  
NameError  
l last)  
<ipython-input-117-ee0ef673ae60> in <module>()  
----> 1 TWD
```

```
Traceback (most recent call
```

```
NameError: name 'TWD' is not defined
```

考考你

- 美金=31.3987、日圓=0.2738，那麼美金乘上日圓+日圓除以美金+美金的平方-日圓開根號等於多少？

關鍵字 Keyword

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

常數

- 關鍵字中的常數有
 - False
 - None
 - True

False 表示邏輯上的假，True 為邏輯上的真。None 是一個特別的值，沒有設定回傳值 (return value) 的函數 (function) 實際上會回傳 None，也就是 null 物件 (object)。

型態定義詞

- 關鍵字中的型態定義詞有
 - class
 - def

class 用於定義類別 (class)，類別為建構物件的模板，Python 的一種類別也是一種型態。def 用於定義函數，函數屬於 'function' 類別。

控制陳述

- 關鍵字中的控制陳述有

- as
- assert
- break
- continue
- del
- elif
- else
- except
- finally
- for
- global
- if
- nonlocal
- pass
- raise
- return
- try
- while
- with
- yield

除了 return 作為函數或方法 (method) 回傳值之用，其他如選擇結構 (selection structure) 的 if-elif-else ，重複結構 (repetition structure) 的 for 、 while ，例外處理 (exception handling) 的 try-except-finally 等。

運算子

- 關鍵字中的運算子 (operator) 有

- and
- not
- or
- is
- in
- lambda

and 、 or 與 not 為邏輯運算子 (logical operator) ， is 用來判斷兩個物件的 id 是否相同， in 用來判斷某個物件是否存在於某個複合資料型態 (compound data type) 之中， lambda 用來建立無名函數 (anonymous function) 。

模組相關

- 關鍵字中跟模組 (module) 相關的有

- from
- import

from 與 import 用來引入模組，或模組中的類別、函數等。

識別字

- 識別字 (identifier) 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。
- 通常識別字會由具有意義的英文單字組成，因此會由字母開始，而非數字 (0-9)、底線符號 (_)，至於單字數量依需要而定。如果單一識別字利用超過一個英文單字組成，各單字之間不可有空格，因為空格會讓編譯器認為是前後分別是兩個不同的識別字。
- 不能拿數字作為識別字的第一個字元

命名習慣

- 連接多英文單字組合的識別字，可用底線符號 (_)，或是大寫駝峰型 (upper camel case) 或是小寫駝峰型 (lower camel case)。
- 大寫駝峰型如
 - SimpleGame
 - MyStory
 - ThreadTestDrive
 - RunThreads
 - TextArea
 - Button_One
 - Midi_Event
 - Short_Message
 - My_Object
 - B_Example
- 小寫駝峰型如
 - actionPerformed
 - setUpNetworking
 - doSomething
 - getField
 - getTitle
 - make_Event
 - turn_Oven_On
 - take_Risk
 - print_List
 - my_Canvas
- 若是只使用一個英文單字，下面是全部用小寫字母的例子
 - clone
 - equals
 - i
 - obj1
 - move
 - paint
 - run
 - play
 - sleep
 - name
 - color

- 首字母大寫的例子

- Zoo
- Animal
- Bird
- Day
- Outer
- Producer
- Server
- Data
- Filename
- Example

- 或全部的字母都大寫，如用在常數 (constant)

- START
- END
- RIGHT
- BEGIN

運算子

- Python 提供多樣、功能完整的運算子 (operator) ，如下列表

+	-	*	**	/	//	%
<<	>>	&		^	~	
<	>	<=	>=	==	!=	

- 另有一些分隔符號 (delimiter)

()	[]	{	}
,	:	.	;	@	=
+=	-=	*=	/=	//=	%=
&=	=	^=	<<=	>>=	**=

算術運算子

Python 的算術運算子 (arithmetic operator) 包含加、減、乘、除、取餘數，皆需兩個運算元 (operand) 構成運算式 (expression) ，如下列表

運算子	功能	範例
+	加	a + b
-	減	a - b
*	乘	a * b
**	指數	a ** b
/	除	a / b
//	整數除法	a // b
%	取餘數	a % b

位移運算子

位移運算子 (shifting operator) 運用在整數資料型態，向右位移等於 n 除以 $\text{pow}(2, n)$ ，向左位移等於 n 乘上 $\text{pow}(2, n)$

運算子	功能	範例
<<	向右位移	a << n
>>	向左位移	a >> n

位元運算子

位元運算子 (bitwise operator) 如下列表

運算子	功能	範例

&	位元且	$a \& b$
	位元包含或	$a b$
\wedge	位元互斥或	$a \wedge b$
\sim	位元相反	$\sim a$

關係運算子

關係運算子 (comparison operator) 需要兩個運算元，如下列表

運算子	功能	範例
<	小於	$a < b$
>	大於	$a > b$
\leq	小於等於	$a \leq b$
\geq	大於等於	$a \geq b$
\equiv	相等	$a \equiv b$
\neq	不相等	$a \neq b$

In []:

```
1 + 1 == 2
```

In []:

```
1 + 1 is 2
```

In []:

```
8 + 7 == 87
```

In []:

```
8 + 9 is not 1
```

In []:

```
"放生" == "棄養"
```

In []:

```
False or True
```

In []:

```
False and True
```

In []:

```
False | True
```

```
In [ ]:
```

```
False & True
```

```
In [ ]:
```

```
bool('Hello world!')
```

```
In [ ]:
```

```
bool('')
```

```
In [ ]:
```

```
bool(0)
```

```
In [ ]:
```

```
bool(1)
```

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合用，會直接將結果儲存到原變數之中，如

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

分隔符號

其他分隔符號有

分隔符號	功能
()	小括弧圍住的運算式會優先計算，函數 (function) 也用小括弧圍住參數列 (parameter list)
[]	序列型態 (sequence type) 的索引符號，或用作定義串列 (list)
{}	用作定義字典 (dictionary)
,	同一行中分隔多個運算式
:	控制陳述條件 (condition) 後的分隔符號
.	用為存取物件的方法 (method) 或屬性 (attribute)
;	可作為單行程式結束的符號，也可不用
@	用作函數或類別 (class) 定義的特殊標記

In []:

```
USD = 31.3987
JPY = 0.2738
USD * JPY + JPY/USD + USD **2 + JPY ** (1/2)
```

In []:

```
JPY ** (1/2)
```

Python內建資料型別

Python 有處理數值、字串、邏輯值、日期與時間等資料的內建型別 Python has a small set of built-in types for handling numerical data, strings, boolean (True or False) values, and dates and time.

- numerical data
 - float (浮點數)
 - int (整數)
- str (字串型別)
- bool (邏輯值型別)
- datetime (日期時間型別)
- None (空無型別)
- unicode

數值型別 Numeric types

內建的數值型別 (numeric types) 共有三種，分別是

型態	描述
int	整數
float	浮點數
complex	複數

可進行以下的計算

計算	描述
$x + y$	x 加 y 之和
$x - y$	x 剪 y 之差
$x * y$	x 乘 y 之積
x / y	x 除 y 之商
$x // y$	x 除 y 之整數商
$x \% y$	x 除 y 之餘數
$-x$	x 取負數
$+x$	x 取正數
$abs(x)$	回傳 x 的絕對值
$int(x)$	轉換 x 為整數
$float(x)$	轉換 x 為浮點數
$complex(re, im)$	轉換 re 為複數的實部， im 為虛部
$c.conjugate()$	回傳 c 的共軛複數
$divmod(x, y)$	回傳 $(x // y, x \% y)$
$pow(x, y)$	x 的 y 次方
$x ** y$	x 的 y 次方

In []:

```
x = 17239871
x ** 6
```

In []:

```
type(x)
```

In []:

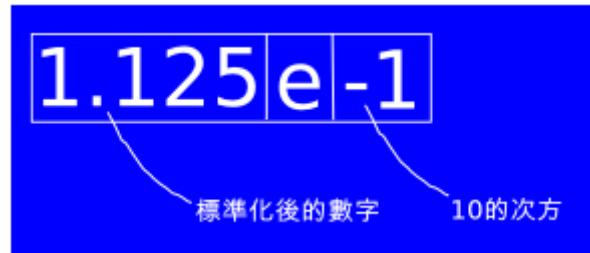
```
3 / 2
```

```
In [ ]:
```

```
type(3/2)
```

```
In [ ]:
```

```
1.125e-1
```



序列型別

內建的序列型別 (sequence types) 共有六種，分別是

型態	描述
str	字串 (string) , 不可變 (immutable)
bytes	字節 (byte) , 不可變
bytearray	字節陣列 (byte array) , 可變 (mutable)
list	串列 (list) , 可變
tuple	序對 (tuple) , 不可變
range	內建函數 range() 回傳的物件 (object) , 常用於 for 迴圈 (for loop)

序列通用的運算有下：

計算	描述
x in s	判斷 x 是否在 s 中
x not in s	判斷 x 是否不在 s 中
s + t	連接 s 及 t
s * n, n * s	將 s 重複 n 次連接 s 本身
s[i]	取得索引值 i 的元素
s[i:j]	取得索引值 i 到 j 的子序列
s[i:j:k]	取得索引值 i 到 j , 間隔 k 的子序列
len(s)	回傳 s 的元素個數
min(s)	回傳 s 中的最小值
max(s)	回傳 s 中的最大值
s.index(i)	取得 s 中第一次出現 i 的索引值
s.count(i)	累計 s 中 i 出現的個數

字串 Strings

字串 (string) 屬於不可變 (immutable) 的序列 (sequence) 型別，字串型態有以下的方法 (method)

方法	描述
<u>str.capitalize()</u> (http://pydoing.blogspot.com/2011/03/python-strcapitalize.html)	回傳將 str 改成首字母大寫，其餘字母小寫的字串
<u>str.center(width[, fillchar])</u> (http://pydoing.blogspot.com/2011/03/python-strcenter.html)	回傳一個將 str 設置字串中央，長度 width 的新字串， fillchar 為填充字元，預設為空格
<u>str.count(sub[, start[, end]])</u> (http://pydoing.blogspot.com/2011/03/python-strcount.html)	計算 sub 出現的次數， start 為起始計算索引值， end 為結束索引值
<u>str.encode(encoding="utf-8", errors="strict")</u> (http://pydoing.blogspot.com/2011/03/python-strencode.html)	回傳 encoding 版本的 bytes 物件
<u>str.endswith(suffix[, start[, end]])</u> (http://pydoing.blogspot.com/2011/03/python-strendswith.html)	判斷 str 是否以 suffix 結尾
<u>str.expandtabs([tabsize])</u> (http://pydoing.blogspot.com/2011/03/python-strexpandtabs.html)	將 tab 符號以 tabsize 的空格數替換
<u>str.find(sub[, start[, end]])</u> (http://pydoing.blogspot.com/2011/03/python-strfind.html)	回傳 sub 第一次出現的索引值
<u>str.format(*args, **kwargs)</u> (http://pydoing.blogspot.com/2011/03/python-strformat.html)	進行格式化字串運算
<u>str.index(sub[, start[, end]])</u> (http://pydoing.blogspot.com/2011/03/python-strindex.html)	回傳 sub 第一次出現的索引值
<u>str.isalnum()</u> (http://pydoing.blogspot.com/2011/03/python-strisalnum.html)	判斷字串中的字元是否都是字母或數字
<u>str.isalpha()</u> (http://pydoing.blogspot.com/2011/03/python-strisalpha.html)	判斷字串中的字元是否都是字母
<u>str.isdecimal()</u> (http://pydoing.blogspot.com/2011/03/python-strisdecimal.html)	判斷字串中所有字元是否是十進位數字
<u>str.isdigit()</u> (http://pydoing.blogspot.com/2011/03/python-strisdigit.html)	判斷字串中所有字元是否是數字

<u>str.isidentifier()</u> <u>(http://pydoing.blogspot.com/2011/03/python-strisidentifier.html)</u>	判斷字串是否可作為合法的識別字
<u>str.islower()</u> <u>(http://pydoing.blogspot.com/2011/03/python-strislower.html)</u>	判斷字串中所有字母字元是否都是小寫字母
<u>str.isnumeric()</u> <u>(http://pydoing.blogspot.com/2011/03/python-strisnumeric.html)</u>	判斷字串中所有字元是否是數字
<u>str.isprintable()</u> <u>(http://pydoing.blogspot.com/2011/03/python-strisprintable.html)</u>	判斷字串中所有字元是否都屬於可見字元
<u>str.isspace()</u> <u>(http://pydoing.blogspot.com/2011/03/python-strisspace.html)</u>	判斷字串是否為空格字元
<u>str.istitle()</u> <u>(http://pydoing.blogspot.com/2011/03/python-stristitle.html)</u>	判斷字串是否適合當作標題
<u>str.isupper()</u> <u>(http://pydoing.blogspot.com/2011/03/python-strisupper.html)</u>	判斷字串中所有字母字元是否都是大寫字母
<u>str.join(iterable)</u> <u>(http://pydoing.blogspot.com/2011/03/python-strjoin.html)</u>	回傳將 str 連結 iterable 各元素的字串
<u>str.ljust(width[, fillchar])</u> <u>(http://pydoing.blogspot.com/2011/03/python-strljust.html)</u>	回傳將 str 在寬度 width 向左對齊的字串，fillchar 為填充字元，預設為空格
<u>str.lower()</u> <u>(http://pydoing.blogspot.com/2011/03/python-strlower.html)</u>	將 str 的英文字母都改成小寫
<u>str.lstrip([chars])</u> <u>(http://pydoing.blogspot.com/2011/03/python-strlstrip.html)</u>	回傳將 str 左邊具有 chars 字元去除的拷貝版本，chars 預設為空格符號
<u>static str.maketrans(x[, y[, z]])</u> <u>(http://pydoing.blogspot.com/2011/03/python-strmaketrans.html)</u>	回傳 x 與 y 配對的 Unicode 編碼字典，若有提供 z，z 中的字元會跟 None 配對
<u>str.partition(sep)</u> <u>(http://pydoing.blogspot.com/2011/03/python-strpartition.html)</u>	以 sep 分割 str 為三個部份，結果回傳具有三個子字串的序對
<u>str.replace(old, new[, count])</u> <u>(http://pydoing.blogspot.com/2011/03/python-strreplace.html)</u>	將 str 中的 old 子字串以 new 代換
<u>str.rfind(sub[, start[, end]])</u> <u>(http://pydoing.blogspot.com/2011/03/python-strrfind.html)</u>	尋找最右邊的 sub，也就是索引值最大的 sub

<u>str.rindex(sub[, start[, end]])</u> http://pydoing.blogspot.com/2011/03/python-strrindex.html	尋找最右邊的 sub，也就是索引值最大的 sub
<u>str.rjust(width[, fillchar])</u> http://pydoing.blogspot.com/2011/03/python-strrjust.html	回傳將 str 在寬度 width 向右對齊的字串，fillchar 為填充字元，預設為空格
<u>str.rpartition(sep)</u> http://pydoing.blogspot.com/2011/03/python-strrpartition.html	以 sep 從最右端分割 str 為三個部份，結果回傳具有三個子字串的序對
<u>str.rsplit([sep[, maxsplit]])</u> http://pydoing.blogspot.com/2011/03/python-strrsplit.html	將 str 從最右端以 sep 分割成子字串，回傳儲存子字串的串列，maxsplit 為子字串最多的數量
<u>str.rstrip([chars])</u> http://pydoing.blogspot.com/2011/03/python-rstrip.html	從 str 的最右端中移除 chars 字元，預設為空白字元
<u>str.split([sep[, maxsplit]])</u> http://pydoing.blogspot.com/2011/03/python-strsplit.html	將 str 以 sep 分割成子字串，回傳儲存子字串的串列，maxsplit 為子字串最多的數量
<u>str.splitlines([keepends])</u> http://pydoing.blogspot.com/2011/03/python-strsplitlines.html	將 str 以新行符號分割成子字串，回傳儲存子字串的串列
<u>str.startswith(prefix[, start[, end]])</u> http://pydoing.blogspot.com/2011/03/python-strstartswith.html	判斷 str 是否以 prefix 開頭
<u>str.strip([chars])</u> http://pydoing.blogspot.com/2011/03/python-rstrip.html	從 str 中移除 chars 字元，預設為空白字元
<u>str.swapcase()</u> http://pydoing.blogspot.com/2011/03/python-strswapcase.html	將 str 中的英文字母進行大小寫轉換
<u>str.title()</u> http://pydoing.blogspot.com/2011/03/python-strtitle.html	將 str 轉換成作為標題的字串
<u>str.translate(map)</u> http://pydoing.blogspot.com/2011/03/python-strtranslate.html	將 str 中的字元以 map 中配對的字元轉換
<u>str.upper()</u> http://pydoing.blogspot.com/2011/03/python-strupper.html	將 str 的英文字母都改成大寫
<u>str.zfill(width)</u> http://pydoing.blogspot.com/2011/03/python-strzfill.html	回傳以 0 填滿 width 的新字串

- 加號運算子可以串接字串 Adding two strings together concatenates them and produces a new string.

```
In [ ]:
```

```
a = 'I have a pen...'  
b = 'I have a apple...'  
a + b
```

```
In [ ]:
```

```
a * 3
```

特別注意，字串與數值的區別

```
In [ ]:
```

```
'123' * 2
```

```
In [ ]:
```

```
'123' ** 2
```

範圍 Range

range 屬於不可變 (immutable) 的序列 (sequence) 型別，支援序列的以下計算

計算	描述
x in s	判斷 x 是否在 s 中
x not in s	判斷 x 是否不在 s 中
s[i]	取得索引值 i 的元素
len(s)	回傳 s 的元素個數
min(s)	回傳 s 中的最小值
max(s)	回傳 s 中的最大值

```
In [ ]:
```

```
range(9)
```

```
In [ ]:
```

```
5 in range(9)
```

```
In [ ]:
```

```
range(9)[6]
```

```
In [ ]:
```

```
len(range(7))
```

串列 List

串列 (list) 屬於一維、可變 (mutable) 的序列 (sequence) 型態，可進行以下序列通用的計算

計算	描述
<code>x in s</code>	判斷 x 是否在 s 中
<code>x not in s</code>	判斷 x 是否不在 s 中
<code>s + t</code>	連接 s 及 t
<code>s * n, n * s</code>	將 s 重複 n 次連接 s 本身
<code>s[i]</code>	取得索引值 i 的元素
<code>s[i:j]</code>	取得索引值 i 到 j 的子序列
<code>s[i:j:k]</code>	取得索引值 i 到 j，間隔 k 的子序列
<code>len(s)</code>	回傳 s 的元素個數
<code>min(s)</code>	回傳 s 中的最小值
<code>max(s)</code>	回傳 s 中的最大值
<code>s.index(i)</code>	取得 s 中第一次出現 i 的索引值
<code>s.count(i)</code>	累計 s 中 i 出現的個數

由於串列是可變的複合資料型態 (compound data type)，也是 Python 中大量運用的工作型態種類，因此有額外以下的計算

計算	描述
<code>s[i] = x</code>	將索引值 i 的元素指派為 x
<code>s[i:j] = t</code>	將索引值 i 到 j 的元素指派為 t，t 為迭代器
<code>del s[i:j]</code>	刪除索引值 i 到 j 的元素
<code>s[i:j:k] = t</code>	將索引值 i 到 j，間隔 k 的元素指派為 t，t 為迭代器
<code>del s[i:j:k]</code>	刪除索引值 i 到 j，間隔 k 的元素
<u>串列綜合運算 list comprehension</u> http://pydoing.blogspot.com/2011/03/python-listcomprehension.html	運用運算式生成新的串列

串列型態有以下的方法 (method)

方法	描述
<u><code>list.append(x)</code></u> http://pydoing.blogspot.com/2011/03/python-listappend.html	將 x 附加到 list 的最後
<u><code>list.extend(x)</code></u> (http://pydoing.blogspot.com/2011/03/python-listextend.html)	將 x 中的元素附加到 list 的最後

list.count(x) (http://pydoing.blogspot.com/2011/03/python-listcount.html)	計算 list 中 x 出現的次數
list.index(x[, i[, j]]) (http://pydoing.blogspot.com/2011/03/python-listindex.html)	回傳 x 在 list 最小的索引值
list.insert(i, x) (http://pydoing.blogspot.com/2011/03/python-listinsert.html)	將 x 插入 list 索引值 i 的地方
list.pop([i]) (http://pydoing.blogspot.com/2011/03/python-listpop.html)	取出 list 中索引值為 i 的元素，預設是最後一個
list.remove(x) (http://pydoing.blogspot.com/2011/03/python-listremove.html)	移除 list 中第一個 x 元素
list.reverse() (http://pydoing.blogspot.com/2011/03/python-listreverse.html)	倒轉 list 中元素的順序
list.sort([key[, reverse]]) (http://pydoing.blogspot.com/2011/03/python-listsort.html)	排序 list 中的元素

- They can be defined using square brackets [] or using the list type function

In []:

```
a_list = [2, 3, 7, None]
a_list[1] = 'peekaboo'
a_list
```

- 添加與移除串列元素 Adding and removing elements.

In []:

```
a_list.append('dwarf')
a_list
```

- 在特定位置插入元素 Using insert you can insert an element at a specific location in the list.

In []:

```
a_list.insert(1, 'red')
a_list
```

- insert的逆運算是pop，它將特定位置上的元素移出來 The inverse operation to insert is pop, which removes and returns an element at a particular index

In []:

```
a_list.pop(1)
```

In []:

```
a_list
```

- 從後方新增append，remove將第一個發現的移除 Elements can be appended to the end of the list with the append method. Elements can be removed by value using remove, which locates the first such value and removes it from the last.

```
In [ ]:
```

```
a_list.append('peekaboo')  
a_list
```

```
In [ ]:
```

```
a_list.remove('peekaboo')  
a_list
```

- 以in關鍵字查核串列是否包含某個值 You can check if a list contains a value using the in keyword.

```
In [ ]:
```

```
'dwarf' in a_list
```

- 串列串接與結合 Concatenating and combining lists

```
In [ ]:
```

```
[4, None, 'foo'] + [7, 8, (2, 3)]
```

```
In [ ]:
```

```
x = [4, None, 'foo']  
x.extend([7, 8, (2, 3)])  
x
```

- 注意+串接是相對昂貴的運算，因為新串列必須創立且物件要複製。用extend添加元素到現存串列比較好，尤其是當涉及大形串列物件時 Note that list concatenation is a comparatively expensive operation since a new list must be created and the objects copied over. Using extend to append elements to an existing list, especially if you are building up a large list, is usually preferable.

```
In [ ]:
```

```
#everything = []  
#for chunk in list_of_lists:  
#    everything.extend(chunk)  
  
#everything = []  
#for chunk in list_of_lists:  
#    everything = everything + chunk
```

- 上面比下面快
- 排序 Sorting

```
In [ ]:
```

```
a = [7, 2, 5, 1, 3]
a.sort()
a
```

- sort函數有許多有用的選項，例如下例可依字串長度(當作key)進行排序 Sort has a few options that will occasionally come in handy. One is the ability to pass a secondary sort key, i.e. a function that produces a value to use to sort the objects.

```
In [ ]:
```

```
b = ['saw', 'small', 'He', 'foxes', 'six']
b.sort(key=len)
b
```

- 二元搜尋與有序的串列 Binary search and maintaining a sorted list
- bisect.bisect尋找維持串列有序的新元素插入位置 bisect.bisect finds the location where an element should be inserted to keep it sorted

```
In [ ]:
```

```
import bisect
c = [1, 2, 2, 2, 3, 4, 7]
bisect.bisect(c, 2)
```

```
In [ ]:
```

```
bisect.bisect(c, 5)
```

- bisect.insort不光找尋插入位置，還實際將新元素插入 bisect.insort actually inserts the element into that location.

```
In [ ]:
```

```
bisect.insort(c, 6)
c
```

- Slicing

```
In [ ]:
```

```
seq = [7, 2, 3, 7, 5, 6, 0, 1]
seq[1:5]
```

- 注意與上面的細微差異 While element at the start index is included, the stop index is not included, so that the number of elements in the result is stop - start

```
In [ ]:
```

```
seq[3:4] = [6, 3]
seq
```

```
In [ ]:
```

```
seq[:5]
```

```
In [ ]:
```

```
seq[3:]
```

```
In [ ]:
```

```
seq[-4:]
```

```
In [ ]:
```

```
seq[-6:-2]
```

```
In [ ]:
```

```
seq[::2]
```

```
In [ ]:
```

```
seq[::-1]
```

序對 Tuple

序對 (tuple) 屬於不可變 (mutable) 的序列 (sequence) 型別，可進行以下序列通用的計算

計算	描述
x in s	判斷 x 是否在 s 中
x not in s	判斷 x 是否不在 s 中
s + t	連接 s 及 t
s * n, n * s	將 s 重複 n 次連接 s 本身
s[i]	取得索引值 i 的元素
s[i:j]	取得索引值 i 到 j 的子序列
s[i:j:k]	取得索引值 i 到 j，間隔 k 的子序列
len(s)	回傳 s 的元素個數
min(s)	回傳 s 中的最小值
max(s)	回傳 s 中的最大值
s.index(i)	取得 s 中第一次出現 i 的索引值
s.count(i)	累計 s 中 i 出現的個數

透過「：」控制序列取值

```
In [ ]:
```

```
a = 'hello, world'  
type(a)
```

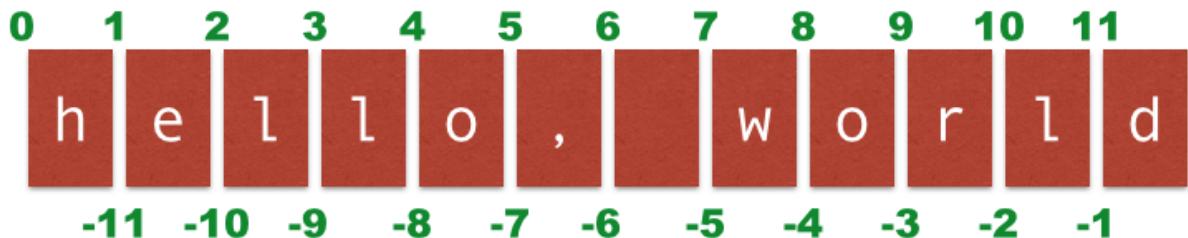
```
In [ ]:
```

```
a[10]
```

```
In [ ]:
```

```
a[-7]
```

- 位置這樣算



```
In [ ]:
```

```
a[2:5]
```

```
In [ ]:
```

```
a[4:]
```

```
In [ ]:
```

```
a[:5]
```

```
In [ ]:
```

```
a[-4:-2]
```

```
In [ ]:
```

```
a[0] + a[4] + a[7] + " " + a[4] + a[-2:]
```

考考你

- 請用前面定義的「`a = 'hello, world'`」，透過字串取值及字串串接的技巧，輸出「how old」字串

集合 Set

內建的集合型別 (set types) 的字面常數使用大括弧圍起來，其物件屬於複合資料型別 (compound data type)，也就是說單一集合型別物件可以包含多個元素，但沒有重複的元素。

以無順序的方式存放物件 **An unordered collection object for other unique objects**

- 例如以下 s1
 - $s1 = \{1, 1, 1, 2, 2, 3, 3, 4, 5\}$
- $s1$ 實際等於 $s2$
 - $s2 = \{1, 2, 3, 4, 5\}$
- 集合型態的物件可進行以下的運算：

計算	描述
$x \text{ in } s$	判斷 x 是否在 s 中
$x \text{ not in } s$	判斷 x 是否不在 s 中
$s1 \& s2$	且運算，取得 $s1$ 與 $s2$ 的交集
$s2 s2$	或運算，取得 $s1$ 與 $s2$ 的聯集
$s1 \wedge s2$	對稱差運算，取得 $s1$ 與 $s2$ 的對稱差集
$s1 - s2$	差運算，取得 $s1$ 與 $s2$ 的差集
$s1 < s2$	判斷 $s1$ 是否為 $s2$ 的真子集
$s1 \leq s2$	判斷 $s1$ 是否為 $s2$ 的子集
$s1 > s2$	判斷 $s2$ 是否為 $s1$ 的真子集
$s1 \geq s2$	判斷 $s2$ 是否為 $s1$ 的子集
$\text{len}(s)$	回傳 s 的元素個數
$\text{min}(s)$	回傳 s 中的最小值， s 中的元素必須是相同型態
$\text{max}(s)$	回傳 s 中的最大值， s 中的元素必須是相同型態

- 集合型態物件的運算大都有相對應的方法 (method)：

方法	描述
<u>$s1.\text{intersection}(s2)$</u> (http://pydoing.blogspot.com/2011/03/python-setintersection.html)	等於 $s1 \& s2$
<u>$s1.\text{union}(s2)$</u> (http://pydoing.blogspot.com/2011/03/python-setunion.html)	等於 $s1 s2$
<u>$s1.\text{symmetric_difference}(s2)$</u> (http://pydoing.blogspot.com/2011/03/python-setsymmetricdifference.html)	等於 $s1 \wedge s2$
<u>$s1.\text{difference}(s2)$</u> (http://pydoing.blogspot.com/2011/03/python-setdifference.html)	等於 $s1 - s2$
<u>$s1.\text{issubset}(s2)$</u> (http://pydoing.blogspot.com/2011/03/python-setissubset.html)	等於 $s1 \leq s2$

s1.issuperset(s2) (http://pydoing.blogspot.com/2011/03/python-setissuperset.html)	等於 $s1 \geq s2$
s1.isdisjoint(s2) (http://pydoing.blogspot.com/2011/03/python-setisdisjoint.html)	判斷 $s1$ 與 $s2$ 是否無交集，若無交集，回傳 True
s.copy() (http://pydoing.blogspot.com/2011/03/python-setcopy.html)	回傳 s 的拷貝

- 由於 set 型態是可變的，因此有額外兩個新增與刪除元素的方法：

方法	描述
s.add(e) (http://pydoing.blogspot.com/2011/03/python-setadd.html)	增加 e 為 s 的元素
s.remove(e) (http://pydoing.blogspot.com/2011/03/python-setremove.html)	從 s 中刪除元素 e

- 可將集合視為無鍵的字典物件 You can think of them like dicts, but keys only, no values.
- 兩種方式創建集合：set函數與大括弧{}運算子 A set can be created in two ways: via the set function or using a set literal with curly braces.

In [118]:

```
set([2, 2, 2, 1, 3, 3])
```

Out[118]:

```
{1, 2, 3}
```

In [119]:

```
{2, 2, 2, 1, 3, 3}
```

Out[119]:

```
{1, 2, 3}
```

- 集合支援許多數學上的集合運算子 Sets support mathematical set operations like union, intersection, difference, and symmetric difference.

In [120]:

```
a = {1, 2, 3, 4, 5}
b = {3, 4, 5, 6, 7, 8}
a | b # union (or)
```

Out[120]:

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [121]:

```
a & b # intersection (and)
```

Out[121]:

```
{3, 4, 5}
```

```
In [122]:
```

```
a - b # difference
```

```
Out[122]:
```

```
{1, 2}
```

```
In [123]:
```

```
a ^ b # symmetric difference (xor)
```

```
Out[123]:
```

```
{1, 2, 6, 7, 8}
```

```
In [124]:
```

```
a_set = {1, 2, 3, 4, 5}  
{1, 2, 3}.issubset(a_set)
```

```
Out[124]:
```

```
True
```

```
In [125]:
```

```
a_set.issuperset({1, 2, 3})
```

```
Out[125]:
```

```
True
```

```
In [126]:
```

```
{1, 2, 3} == {3, 2, 1}
```

```
Out[126]:
```

```
True
```

- Python集合物件集合物件方法 : a.add(x), a.remove(x), a.union(b), a.intersection(b), a.difference(b), a.symmetric_difference(b), a.issubset(b), a.issuperset(b), a.isdisjoint(b)

```
In [127]:
```

```
tup = 4, 5, 6  
tup
```

```
Out[127]:
```

```
(4, 5, 6)
```

- 值組中還有值組，稱為巢狀值組 Creating a tuple of tuples

```
In [128]:
```

```
nested_tup = (4, 5, 6), (7, 8)  
nested_tup
```

```
Out[128]:
```

```
((4, 5, 6), (7, 8))
```

- 透過tuple函數可將序列或迭代物件轉為值組 Any sequence or iterator can be converted to a tuple by invoking tuple.

```
In [129]:
```

```
tuple([4, 0, 2])
```

```
Out[129]:
```

```
(4, 0, 2)
```

```
In [130]:
```

```
tup = tuple('string')  
tup
```

```
Out[130]:
```

```
('s', 't', 'r', 'i', 'n', 'g')
```

- Python索引從0開始 Zero-Based Numbering.

```
In [131]:
```

```
tup[0]
```

```
Out[131]:
```

```
's'
```

- 值組是不可更改的 Tuple is immutable.

```
In [132]:
```

```
tup = tuple(['foo', [1, 2], True])  
tup  
# tup[2] = False # TypeError: 'tuple' object does not support item assignment
```

```
Out[132]:
```

```
('foo', [1, 2], True)
```

```
In [133]:
```

```
tup[1].append(3)  
tup
```

```
Out[133]:
```

```
('foo', [1, 2, 3], True)
```

- 值組可用+運算子串接成較長的值組 Tuples can be concatenated using the + operator to produce longer tuples

In [134]:

```
(4, None, 'foo') + (6, 0) + ('bar',)
```

Out[134]:

```
(4, None, 'foo', 6, 0, 'bar')
```

- 值組可用*運算子複製成較長的值組 Multiplying a tuple by an integer, as with lists, has the effect of concatenating together that many copies of the tuple.

In [135]:

```
('foo', 'bar') * 4
```

Out[135]:

```
('foo', 'bar', 'foo', 'bar', 'foo', 'bar', 'foo', 'bar')
```

- 解構值組 Unpacking tuples.

In [136]:

```
tup = (4, 5, 6)
a, b, c = tup
c
```

Out[136]:

```
6
```

- Python的變數交換方式 Pythonic way to swap variables.

In [137]:

```
x, y = 1, 2 # now x is 1, y is 2
x, y = y, x # Pythonic way to swap variables; now x is 2, y is 1
x
```

Out[137]:

```
2
```

In [138]:

```
y
```

Out[138]:

```
1
```

- count與index是值組的兩個方法 Count and index are two methods for tuple instances.

```
In [139]:
```

```
a = (1, 2, 2, 2, 3, 4, 2)
a.count(2)
```

```
Out[139]:
```

```
4
```

```
In [140]:
```

```
a.index(3)
```

```
Out[140]:
```

```
4
```

```
In [141]:
```

```
c
```

```
Out[141]:
```

```
6
```

字典 Dict

可能是Python最重要的資料結構 Dict: dict is likely the most important built-in Python data structure.

- 常稱為雜湊圖和關聯矩陣 A more common name for it is hash map or associative array.
- 可長可短的鍵值對，鍵與值均須為Python物件 It is a flexibly-sized collection of key-value pairs, where key and value are Python objects.
- 用大括弧{}以及冒號:來分隔鍵與值，以創建字典物件 One way to create one is by using curly braces {} and using colons to separate keys and values
- 建立字典變數可利用大括弧，裡頭以 key:value 為配對的資料項目，每一筆資料再以逗號區隔開，例如
 - `d1 = {1:"a", 2:"b"}`
- 上述字典型態的變數 d1 有兩筆資料，第一筆資料的 key 為 1 ， value 為 "a" ，第二筆資料的 key 為 2 ， value 為 "b" 。
- 使用字典須注意， key 必須是不可變的 (immutable) 資料型態，如數字、字串 (string) 等， value 沒有限制，因此有需要的話，使用串列 (list) 或字典皆可。
- 也可以利用字典型態的建構子 (constructor) 建立物件，如下
 - `d2 = dict(1="a", 2="b")`
 - `d3 = dict({1:"a", 2:"b"})`
 - `d4 = dict(zip((1, 2), ("a", "a")))`
 - `d5 = dict([[2, "b"], [1, "a"]])`

以上 d2 、 d3 、 d4 、 d5 所得到的字典物件，全都會與 d1 相同。

- 字典物件可進行以下的運算

計算	描述
<code>d[key]</code>	從 d 中取得 key 的 value
<code>d[key] = value</code>	將 d 的 key 指定為 value
<code>del d[key]</code>	刪除 d 中 key 所指定的 value
<code>key in d</code>	判斷 key 是否在 d 中
<code>key not in d</code>	判斷 key 是否不在 d 中
<code>iter(d)</code>	回傳由 d 的 key 建立的迭代器
<code>len(d)</code>	回傳 d 的配對資料個數

- 字典物件有以下的方法 (method)

方法	描述
<code>dict.clear()</code> (http://pydoing.blogspot.com/2011/03/python-dictclear.html)	清空 dict 的所有配對資料
<code>dict.copy()</code> (http://pydoing.blogspot.com/2011/03/python-dictcopy.html)	回傳 dict 的拷貝
<code>classmethod dict.fromkeys(seq[, value])</code> (http://pydoing.blogspot.com/2011/03/python-dictfromkeys.html)	由 seq 中的元素構成 key ，每個 key 都給相同的 value 值

<u>dict.get(key[, default])</u> <u>(http://pydoing.blogspot.com/2011/03/python-dictget.html)</u>	從 dict 中取得 key 的 value，若無此 key 則回傳 default，default 預設為 None
<u>dict.items()</u> <u>(http://pydoing.blogspot.com/2011/03/python-dictitems.html)</u>	回傳 dict_items 物件，使 key:value 儲存為序對，然後依序儲存在 dict_items 物件中
<u>dict.keys()</u> <u>(http://pydoing.blogspot.com/2011/03/python-dictkeys.html)</u>	回傳 dict_items 物件，使 key 依序儲存在 dict_items 物件中
<u>dict.pop(key[, default])</u> <u>(http://pydoing.blogspot.com/2011/03/python-dictpop.html)</u>	將 key 的 value 從 dict 移除，若無此 key，回傳 default
<u>dict.popitem()</u> <u>(http://pydoing.blogspot.com/2011/03/python-dictpopitem.html)</u>	從 dict 移除任意一組 key:value
<u>dict.setdefault(key[, default])</u> <u>(http://pydoing.blogspot.com/2011/03/python-dictsetdefault.html)</u>	如果 key 在 dict 中，回傳 value 值，反之，將 key=default 加入 dict 之中
<u>dict.update([other])</u> <u>(http://pydoing.blogspot.com/2011/03/python-dictupdate.html)</u>	將 dict 以 other 更新
<u>dict.values()</u> <u>(http://pydoing.blogspot.com/2011/03/python-dictvalues.html)</u>	回傳 dict_items 物件，使 value 依序儲存在 dict_items 物件中

In [142]:

```
d1 = {'a' : 'some value', 'b' : [1, 2, 3, 4]}
d1
```

Out[142]:

```
{'a': 'some value', 'b': [1, 2, 3, 4]}
```

- 元素存取、插入與指定的語法同串列和值組 Elements can be accessed and inserted or set using the same syntax as accessing elements of a list or tuple.

In [143]:

```
d1[7] = 'an integer'
d1
```

Out[143]:

```
{7: 'an integer', 'b': [1, 2, 3, 4], 'a': 'some value'}
```

```
In [144]:
```

```
d1['b']
```

```
Out[144]:
```

```
[1, 2, 3, 4]
```

```
In [145]:
```

```
'b' in d1
```

```
Out[145]:
```

```
True
```

```
In [146]:
```

```
d1[5] = 'some value'  
d1['dummy'] = 'another value'  
d1
```

```
Out[146]:
```

```
{7: 'an integer',  
'b': [1, 2, 3, 4],  
'dummy': 'another value',  
5: 'some value',  
'a': 'some value'}
```

```
In [147]:
```

```
del d1[5]  
d1
```

```
Out[147]:
```

```
{7: 'an integer',  
'b': [1, 2, 3, 4],  
'dummy': 'another value',  
'a': 'some value'}
```

```
In [148]:
```

```
ret = d1.pop('dummy')  
ret
```

```
Out[148]:
```

```
'another value'
```

```
In [149]:
```

```
print(d1.keys())  
d1.values()
```

```
dict_keys([7, 'b', 'a'])
```

```
Out[149]:
```

```
dict_values(['an integer', [1, 2, 3, 4], 'some value'])
```

- 可以update方法將字典合併 One dict can be merged into another using the update method.

```
In [150]:
```

```
d1.update({'b' : 'foo', 'c' : 12})  
d1
```

```
Out[150]:
```

```
{7: 'an integer', 'c': 12, 'b': 'foo', 'a': 'some value'}
```

- 常以2值組產生字典 Since a dict is essentially a collection of 2-tuples, it should be no shock that the dict type function accepts a list of 2-tuples

```
In [151]:
```

```
mapping = dict(zip(range(5), reversed(range(5))))  
mapping
```

```
Out[151]:
```

```
{0: 4, 1: 3, 2: 2, 3: 1, 4: 0}
```

- 常用下列方法設定預設值 It's very common to have logic like the following to set default value.

```
if key in some_dict: value = some_dict[key] else: value = default_value
```

- dict的get和pop方法可以設定預設值，因此上述if-else程式撰寫方法可以簡單表示如下 The dict methods get and pop can take a default value to be returned, so that the above if-else block can be written simply as follows.
- get方法若沒有發現key則回傳空無物件None，然而pop方法未發現key時會發出例外情況訊息 get by default will return None if the key is not present, while pop will raise an exception.

```
value = some_dict.get(key, default_value)
```

```
In [152]:
```

```
words = ['apple', 'bat', 'bar', 'atom', 'book']  
by_letter = {}  
for word in words:  
    letter = word[0]  
    if letter not in by_letter:  
        by_letter[letter] = [word]  
    else:  
        by_letter[letter].append(word)  
  
by_letter
```

```
Out[152]:
```

```
{'a': ['apple', 'atom'], 'b': ['bat', 'bar', 'book']}
```

- by_letter.setdefault(letter, []).append(word)可以取代上面if-else判斷敘述

```
In [153]:
```

```
words = ['apple', 'bat', 'bar', 'atom', 'book']
by_letter = {}
for word in words:
    letter = word[0]
    by_letter.setdefault(letter, []).append(word)

by_letter
```

```
Out[153]:
```

```
{'a': ['apple', 'atom'], 'b': ['bat', 'bar', 'book']}
```

- 內建collections模組有一好用的類別defaultdict，可以傳入一個Python型別或函數以產生字典中每個槽的預設值 The built-in collections module has a useful class, defaultdict, which makes this even easier. One is created by passing a type or function for generating the default value for each slot in the dict.

```
In [154]:
```

```
from collections import defaultdict
```

- 空字典，預設的值儲存容器為串列

```
In [155]:
```

```
by_letter = defaultdict(list)
by_letter
```

```
Out[155]:
```

```
defaultdict(list, {})
```

- 以首字為keys，開始加字

```
In [156]:
```

```
for word in words:
    by_letter[word[0]].append(word)
```

```
In [157]:
```

```
by_letter
```

```
Out[157]:
```

```
defaultdict(list, {'a': ['apple', 'atom'], 'b': ['bat', 'bar', 'book']})
```

- 字典的值可以是任何的Python物件，但其鍵必須為不可更改的物件，例如：前述的純量型別(整數、浮點數與字串)，或是值組，這種性質稱為可雜湊性 While the values of a dict can be any Python object, the keys have to be immutable objects like scalar types (int, float, string) or tuples (all the objects in the tuple need to be immutable, too). The technical term here is hashability.
- 可以hash函數檢查任一物件是否為可雜湊的 You can check whether an object is hashable (can be used as a key in a dict) with the hash function.

```
In [158]:
```

```
hash('string')
```

```
Out[158]:
```

```
-483641919220361367
```

```
In [159]:
```

```
hash((1, 2, (2, 3)))
```

```
Out[159]:
```

```
1097636502276347782
```

```
In [160]:
```

```
# hash((1, 2, [2, 3])) # fails because lists are mutable
```

- 可將list轉換為tuple再設定為key To use a list as a key, an easy fix is to convert it to a tuple:

```
In [161]:
```

```
d = {}
d[tuple([1, 2, 3])] = 5
d
```

```
Out[161]:
```

```
{(1, 2, 3): 5}
```

型別轉換 Type casting

```
In [162]:
```

```
x = 6666666
y = 'hello'
z = True
```

```
In [163]:
```

```
type(x)
```

```
Out[163]:
```

```
int
```

```
In [164]:
```

```
type(y)
```

```
Out[164]:
```

```
str
```

```
In [165]:
```

```
type(z)
```

```
Out[165]:
```

```
bool
```

```
In [166]:
```

```
str(x)
```

```
Out[166]:
```

```
'6666666'
```

```
In [167]:
```

```
str(z)
```

```
Out[167]:
```

```
'True'
```

```
In [168]:
```

```
int(z)
```

```
Out[168]:
```

```
1
```

考考你

- 經過上面三行的執行，變數x, y, z各別有何改變？

Python控制敘述

判斷控制

if, elif, and else

```
In [169]:
```

```
if True:  
    print("YES")
```

```
YES
```

```
In [170]:
```

```
if False:  
    print("YES")  
else:  
    print("NO")
```

```
NO
```

In [171]:

```
if 8+7==87:  
    print("Orz")  
elif 8+9==87:  
    print("666")  
else:  
    print("NO")
```

NO

考考你

- 練習：
 - 令x='晴天'
 - 如果x是雨天，print("帶雨傘")，反之print("戴墨鏡")

In [172]:

```
a = [1, 2, 3]  
if a:  
    print ('I found something!')
```

I found something!

- 空序列被視為假 Empty sequences (lists, dicts, tuples, etc.) are treated as False if used in control flow.

In [173]:

```
b = []  
if not b:  
    print ('Empty!')
```

Empty!

In [174]:

```
x = -99  
if x < 0:  
    print ('It is negative')
```

It is negative

In [175]:

```
x = 99  
if x < 0:  
    print ('It is negative')  
elif x == 0:  
    print ('Equal to zero')  
elif 0 < x < 5:  
    print ('Positive but smaller than 5')  
else:  
    print ('Positive and larger than or equal to 5')
```

Positive and larger than or equal to 5

迴圈控制

In [176]:

```
x=0
while x < 5:
    print (x, "is less than 5")
    x += 1
```

```
0 is less than 5
1 is less than 5
2 is less than 5
3 is less than 5
4 is less than 5
```

In [177]:

```
for x in range(5):
    print (x, "is less than 5")
```

```
0 is less than 5
1 is less than 5
2 is less than 5
3 is less than 5
4 is less than 5
```

In [178]:

```
for x in range(10):
    if x==3:
        continue # go immediately to the next iteration
    if x==5:
        break # quit the loop entirely
    print (x)
```

```
0
1
2
4
```

Python自定義函數

- Python程式碼最主要的組織與再利用方式就是函數 Functions are the primary and most important method of code organization and reuse in Python.
- 函數以def關鍵字定義之，並以return關鍵字回傳函數輸出 Functions are declared using the def keyword and returned from using the return keyword.

In [179]:

```
def my_function(x, y, z=1.5):
    if z > 1:
        return z * (x + y)
    else:
        return z / (x + y)
```

- 如果函數未以return敘述回傳，輸出將為空無物件 If the end of a function is reached without encountering a return statement, None is returned.
- 函數參數分為位置引數與關鍵字引數，上例中何引數為前者？何引數為後者？ Each function can have some number of positional arguments and some number of keyword arguments. Keyword arguments are most commonly used to specify default values or optional arguments. In the above function, x and y are positional arguments while z is a keyword argument.
- 關鍵字引數必須接續在位置引數(如果有的話)之後，關鍵字引數的順序可以任意，只須記住引數的名稱即可 The main restriction on function arguments is that the keyword arguments must follow the positional arguments (if any). You can specify keyword arguments in any order; this frees you from having to remember which order the function arguments were specified in and only what their names are.

In [180]:

```
my_function(5, 6, z=0.7)
```

Out[180]:

```
0.06363636363636363
```

In [181]:

```
my_function(3.14, 7, 3.5)
```

Out[181]:

```
35.49
```

- Functions can access variables in two different scopes: global and local. An alternate and more descriptive name describing a variable scope in Python is a namespace. Any variables that are assigned within a function by default are assigned to the local name-space. The local namespace is created when the function is called and immediately populated by the function's arguments. After the function is finished, the local name-space is destroyed (with some exceptions, see section on closures below).

In [182]:

```
def func():
    a = []
    for i in range(5):
        a.append(i)
```

In [183]:

```
func() # return nothing because of local scoping
# a # NameError: name 'a' is not defined
```

In [184]:

```
a = []
def func():
    for i in range(5):
        a.append(i)
```

```
In [185]:
```

```
func()  
a
```

```
Out[185]:
```

```
[0, 1, 2, 3, 4]
```

```
In [186]:
```

```
a
```

```
Out[186]:
```

```
[0, 1, 2, 3, 4]
```

Jupyter介紹及操作



Open source, interactive data science and scientific computing across over 40 programming languages.

Jupyter簡介

- 前身是IPython Notebook
- Jupyter的名稱來源於JULia, PYThon, R
- 讀音和木星（Jupiter）是類似的
- 原創者 Fernando Perez 博士
 - [Fernando 在台灣介紹 IPython 的影片 \(http://youtu.be/ZD5n9s8PGtI\)](http://youtu.be/ZD5n9s8PGtI)
- 非常適合用來做筆記、報表與簡報(特別厲害)

常用快捷鍵

- 完整表：Help > Keyboard Shortcuts
- 開啟快捷鍵搜索欄：Ctrl + Shift + P
- 註解整行(段)Code：Ctrl + /
- 開啟說明文件：於函數後方加上問號
 - 例如：sum?
- 透過Alt鍵，使用多光標功能(multiple cursors)

```
In [ ]: x = [
    'one'
    'two'
    'three'
    'four'
    'five'
]
```

- Run Cell：Ctrl + Enter (留在原地)、Shift + Enter (跳到下一格)
- Add Cell Below (Command Mode)：B (大小寫無關)
- Delete Cell (Command Mode)：DD

魔法命令(Magic Commands)

- 查看所有：%lsmagic

In [188]:

```
%lsmagic
```

Out[188]:

```
Available line magics:
%alias %alias_magic %autocall %automagic %autosave %bookmark %
%cat %cd %clear %colors %config %connect_info %cp %debug %dhi
%st %dirs %doctest_mode %ed %edit %env %gui %hist %history %
%killbgscripts %ldir %less %lf %lk %ll %load %load_ext %loadp
%y %logoff %logon %logstart %logstate %logstop %ls %lsmagic %
%lx %macro %magic %man %matplotlib %mkdir %more %mv %notebook
%page %pastebin %pdb %pdef %pdoc %pfile %pinfo %pinfo2 %po
%pd %pprint %precision %profile %prun %psearch %psource %pushd
%pwd %pycat %pylab %qtconsole %quickref %recall %rehashx %r
%eload_ext %rep %rerun %reset %reset_selective %rm %rmdir %run
%save %sc %set_env %store %sx %system %tb %time %timeit %
%unalias %unload_ext %who %who_ls %whos %xdel %xmode
```

Available cell magics:

```
%%! %%HTML %%SVG %%bash %%capture %%debug %%file %%html %%ja
%vascript %%js %%latex %%perl %%prun %%pypy %%python %%python2
%%python3 %%ruby %%script %%sh %%svg %%sx %%system %%time
%%timeit %%writefile
```

Automagic is ON, % prefix IS NOT needed for line magics.