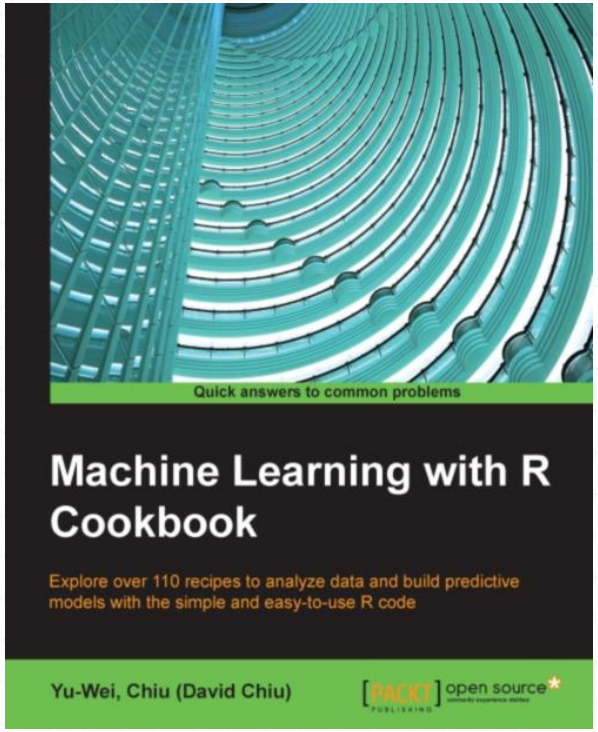
Python 語言基礎課程

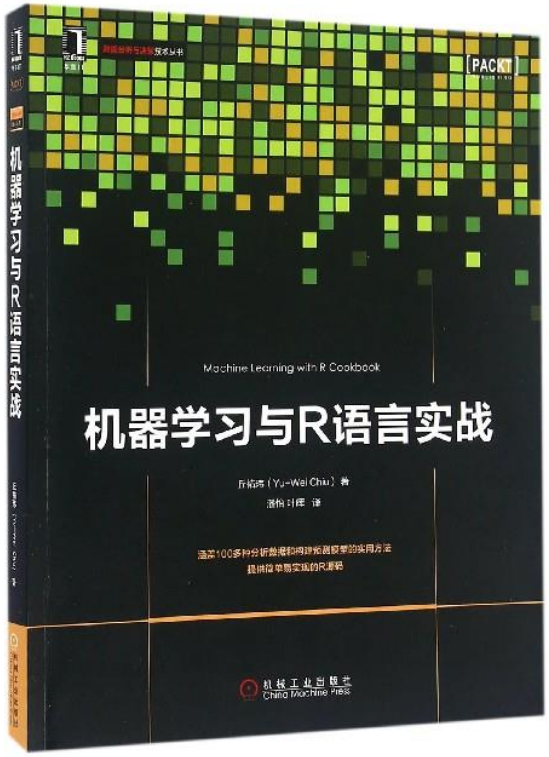
David Chiu

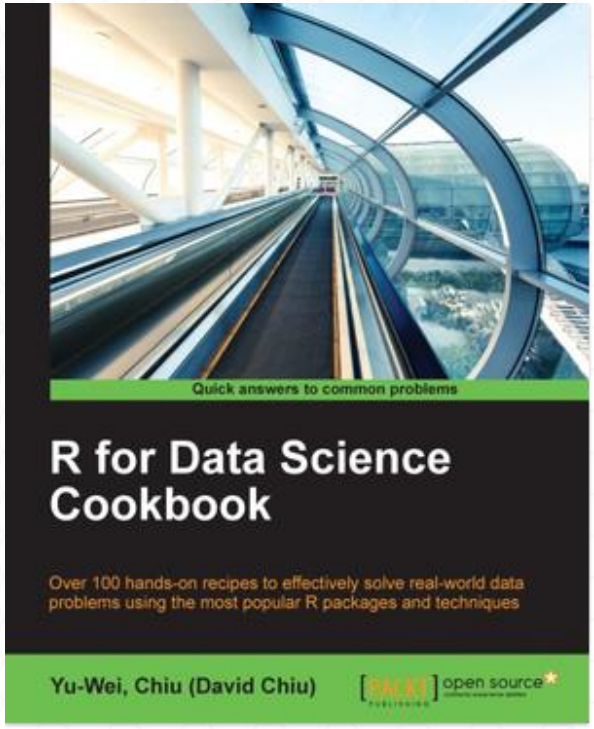
關於我

- 大數軟體有限公司創辦人
- 前趨勢科技工程師
- ywchiu.com
- 大數學堂
  http://www.largitdata.com/
- 粉絲頁
  https://www.facebook.com/largitdata
- R for Data Science Cookbook
  https://www.packtpub.com/big-data-and-business-intelligence/r-data-science-cookbook
- Machine Learning With R Cookbook
  https://www.packtpub.com/big-data-and-business-intelligence/machine-learning-r-cookbook

Machine Learning With R Cookbook (機器學習與R語言實戰) & R for Data Science Cookbook







Author: David (YU-WEI CHIU) Chiu

課程資料

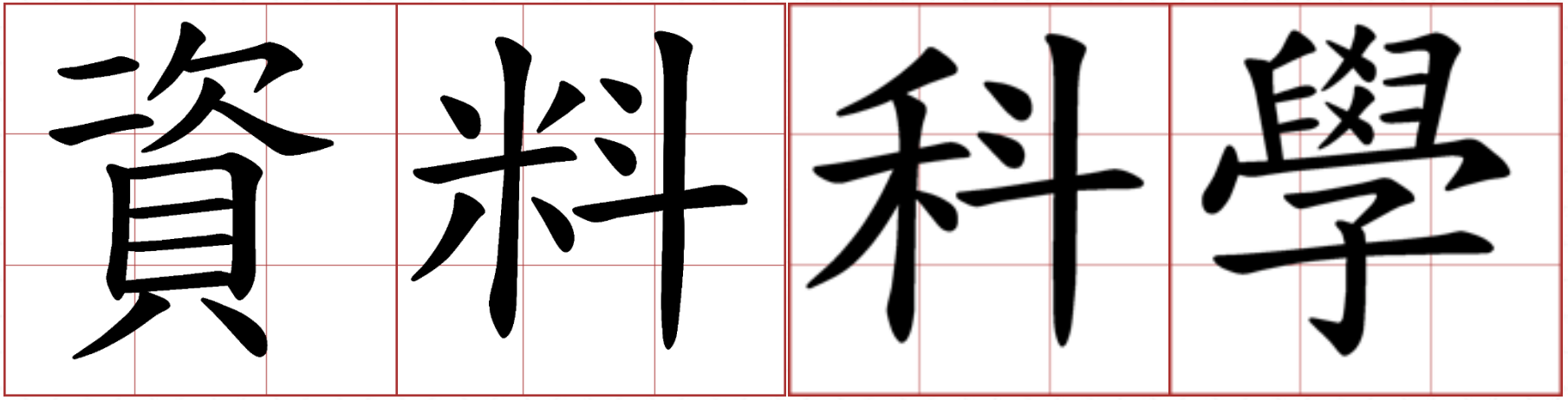
- 所有課程補充資料、投影片皆位於
  - https://github.com/ywchiu/fubonpy

Python語言與資料分析

AlphaGO
使用深度學習技術打敗頂尖棋手


Tesla
讓自動駕駛不再是夢想

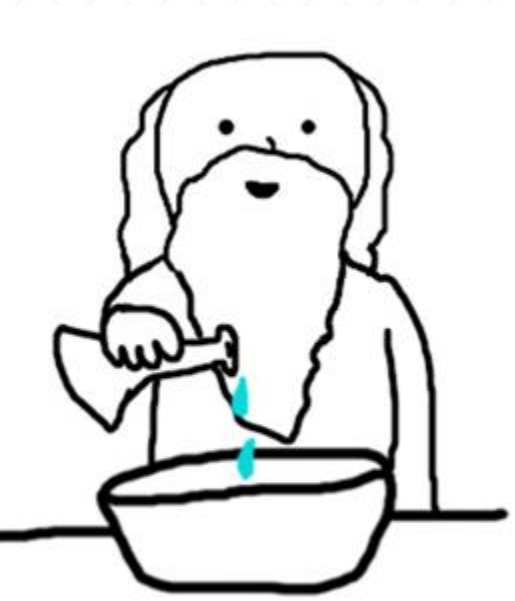

人工智慧
取代重複性高的工作
創造更多想像與新可能



從資料鑒往

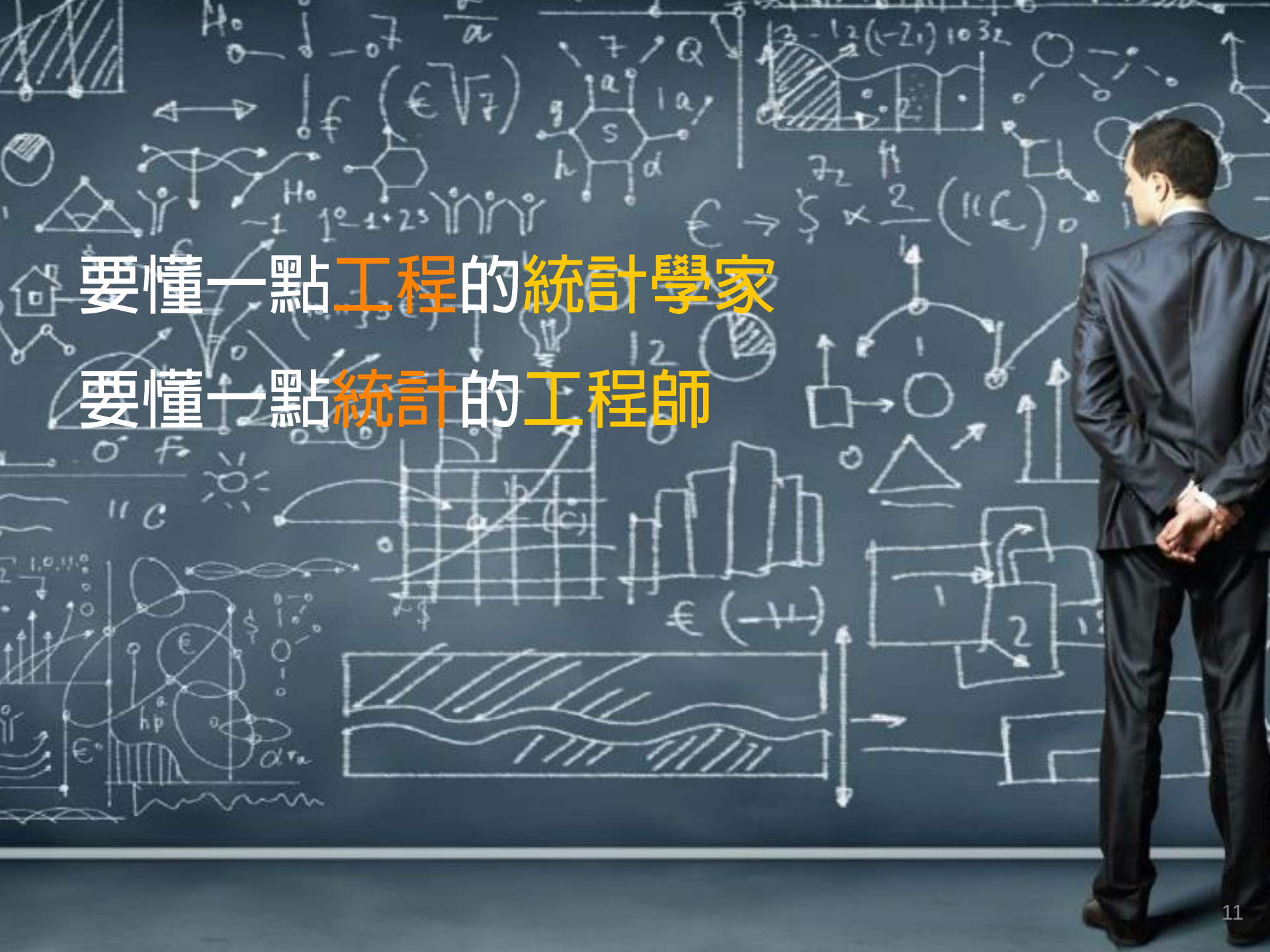
從資料知來

加一點數學統計

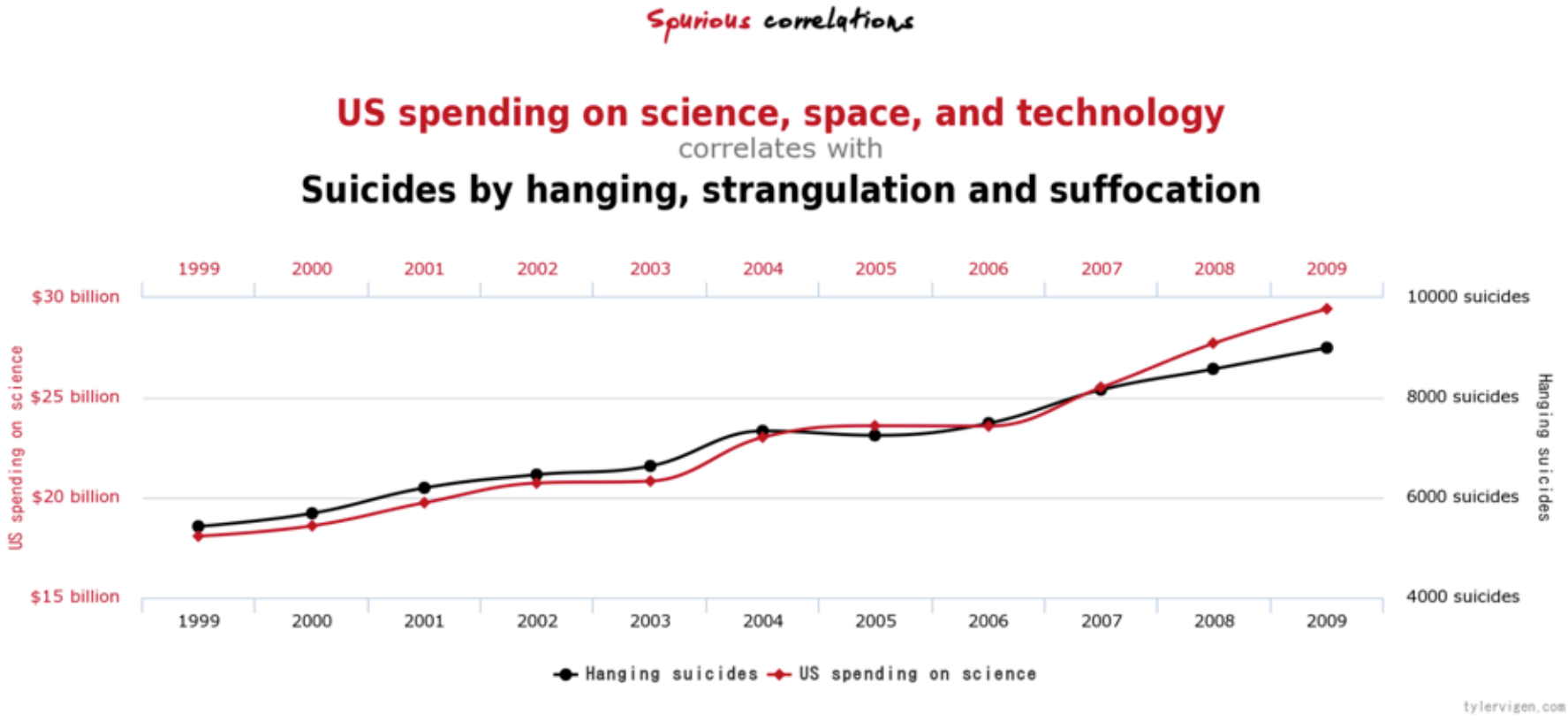


加一點工程

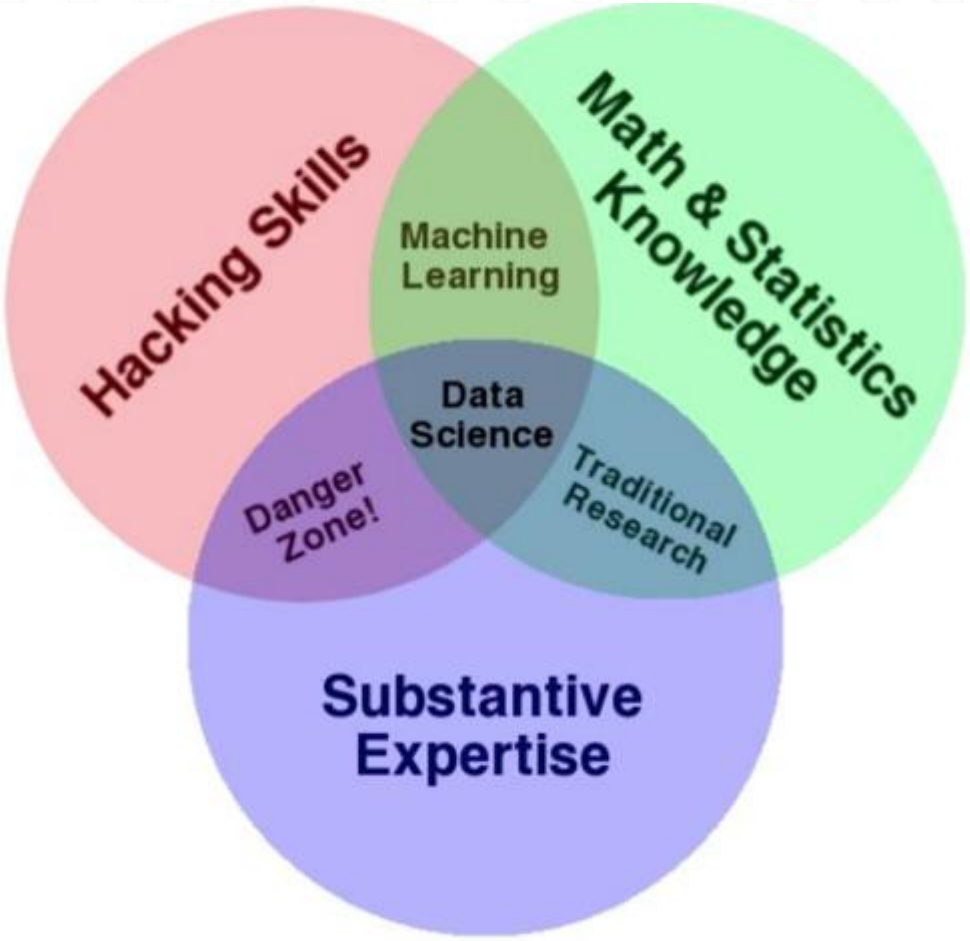
產生資料科學


要懂一點工程的統計學家
要懂一點統計的工程師

只有工程統計還不夠


Spurious correlations
US spending on science, space, and technology
correlates with
Suicides by hanging, strangulation and suffocation
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
$30 billion
$25 billion
$20 billion
$15 billion
US spending on science
10000 suicides
8000 suicides
6000 suicides
4000 suicides
Hanging suicides
Hanging suicides
US spending on science
tylervigen.com

資料科學


Hacking Skills
Math & Statistics Knowledge
Machine Learning
Data Science
Danger Zone!
Traditional Research
Substantive Expertise

資料科學能力

- 統計 (Statistic)

  單變數分析、多變數分析、變異數分析

- 資料處理 (Data Munging)

  抓取資料、清理資料、轉換資料

- 數據視覺化(Data Visualization)

  圖表、商業智慧系統

資料科學家主要工作內容

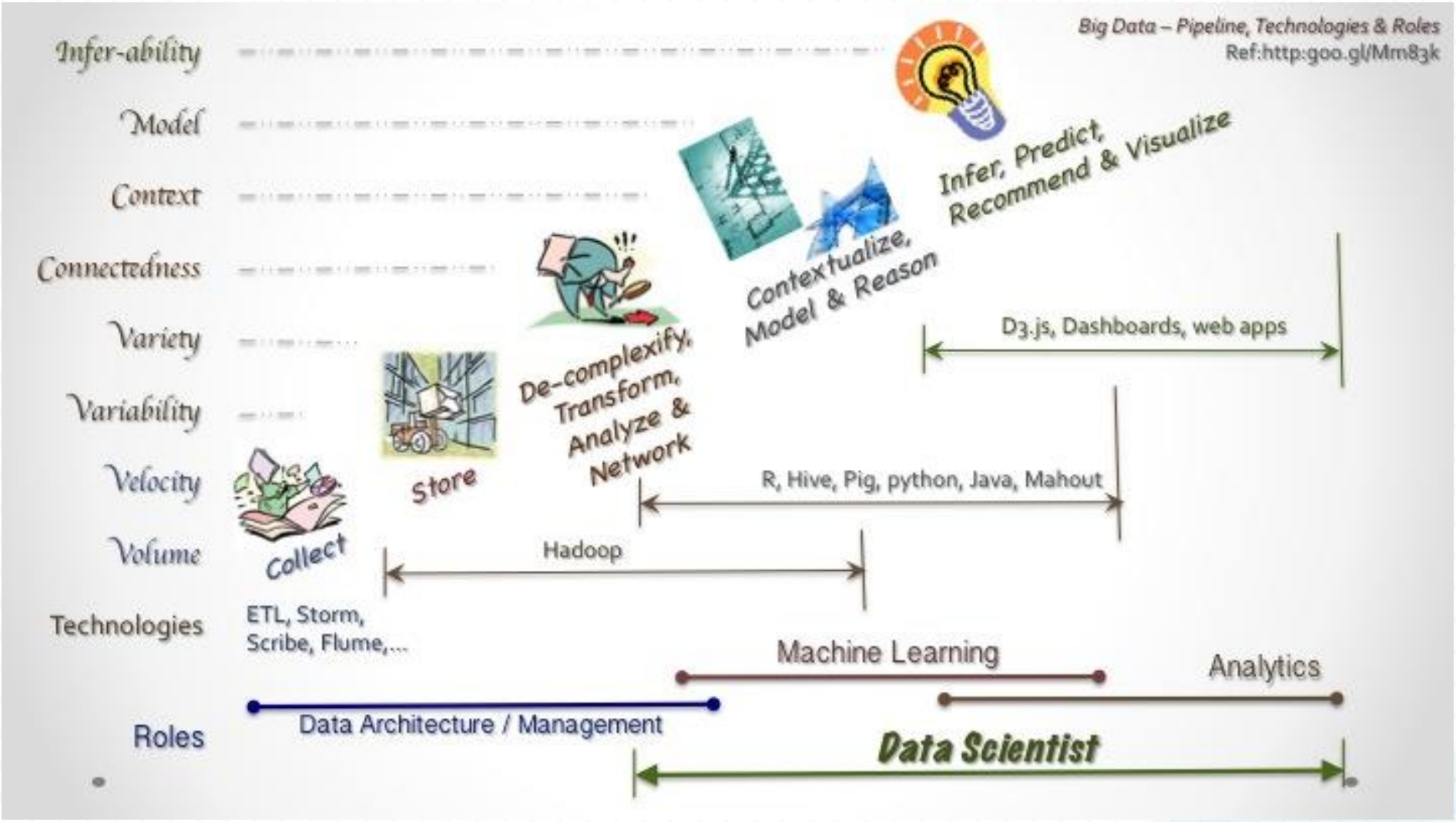
“80% 都在做加總與平均”

工作內容

1. 資料處理 (Data Munging)
2. 資料分析 (Data Analysis)
3. 詮釋結果 (Interpret Result)

真正能用在資料分析的時間很少，必須要能善用工具

資料科學分析步驟


Big Data – Pipeline, Technologies & Roles
Ref:http:goo.gl/Mm8gk
Infer-ability
Model
Context
Connectedness
Variety
Variability
Velocity
Volume
Technologies
Roles
Infer, Predict, Recommend & Visualize
Contextualize, Model & Reason
De-complexify, Transform, Analyze & Network
Store
Collect
D3.js, Dashboards, web apps
R, Hive, Pig, python, Java, Mahout
Hadoop
ETL, Storm, Scribe, Flume,...
Machine Learning
Analytics
Data Architecture / Management
Data Scientist

資料分析工具

Python 語言

Python 語言

- 動態語言 (Dynamic Language)
  - 於執行時期(Runtime)執行程式碼 (不用編譯)
  - Dynamic Type: 函式與變數都不需要宣告類型
- 直譯式語言 (Interpreted Language)
  - 每次執行後可以直接看到結果
- 物件導向語言 (OOP)
- 可執行於多平臺 (Python VM)

Python 語言優點

- 可執行於多平臺 (Python VM)
- 擁有相當多的協力廠商資源
  (資料分析、圖形介面、網頁開發)
- 語法簡潔，編寫快速

簡單易用

JAVA

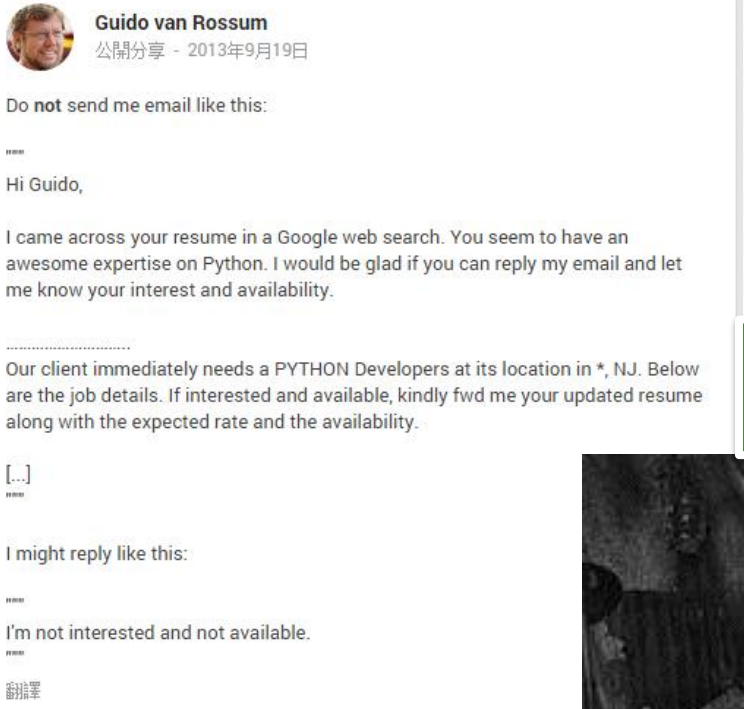

```
class test{
    public static void main(String args[]){
        System.out.println("Hello World");
    }
}
```

PYTHON


```
print("Hello World")
```

Guido van Rossum – Python 之父

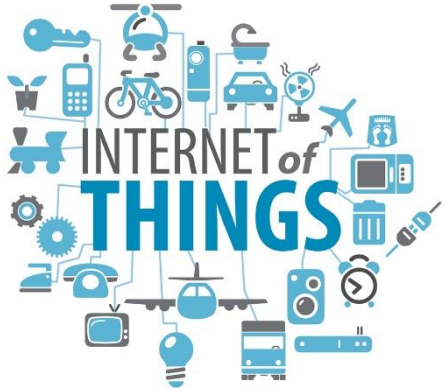
Guido Van Rossum
(https://goo.gl/2kul7Y)



Monty Python
(https://goo.gl/dTjCxR)

豐富的函式庫讓Python 無所不在

物聯網
(http://goo.gl/2j45Nk)



網頁製作
(https://goo.gl/2304w7)

資料分析
(https://goo.gl/2304w7)

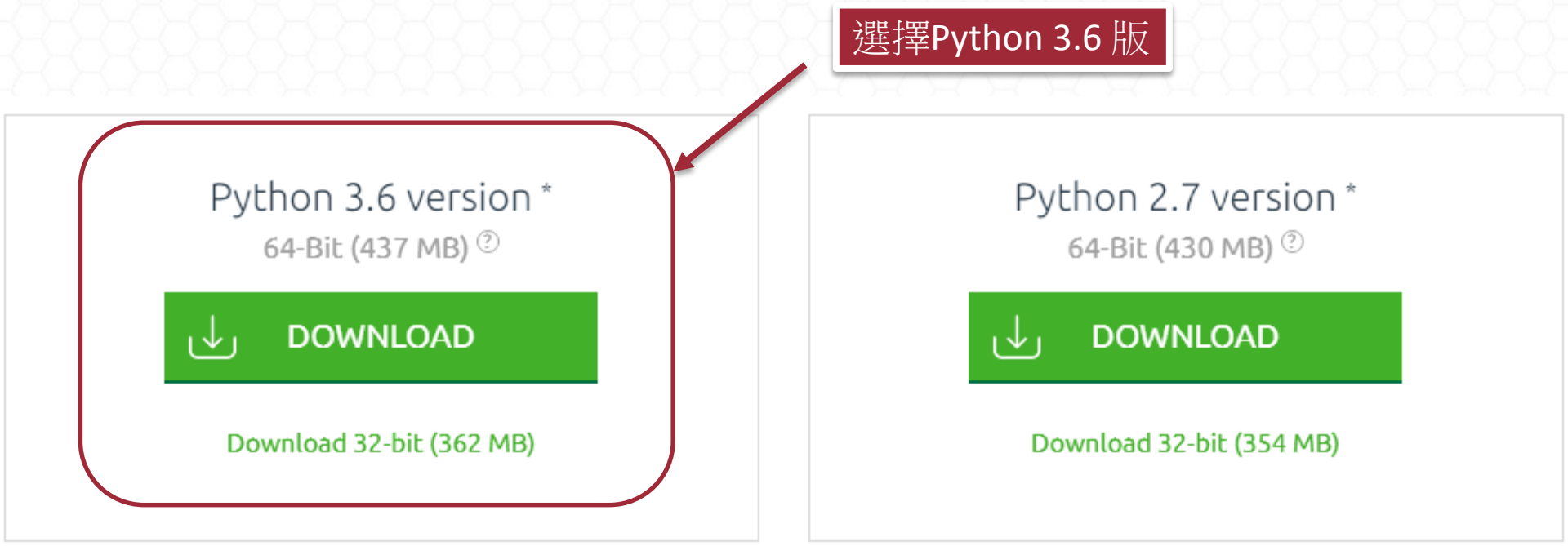
完整的資料分析套件

- 統計科學計算
  - Numpy
  - Scipy
  - statsmodels
- 結構化資料處理與分析
  - Pandas
- 資料探索編輯器
  - Jupyter Notebook
- 深度學習
  - TensorFlow
  - MXNet
- 大數據處理
  - PySpark
- 機器學習
  - Scikit-learn

使用Python 的公司

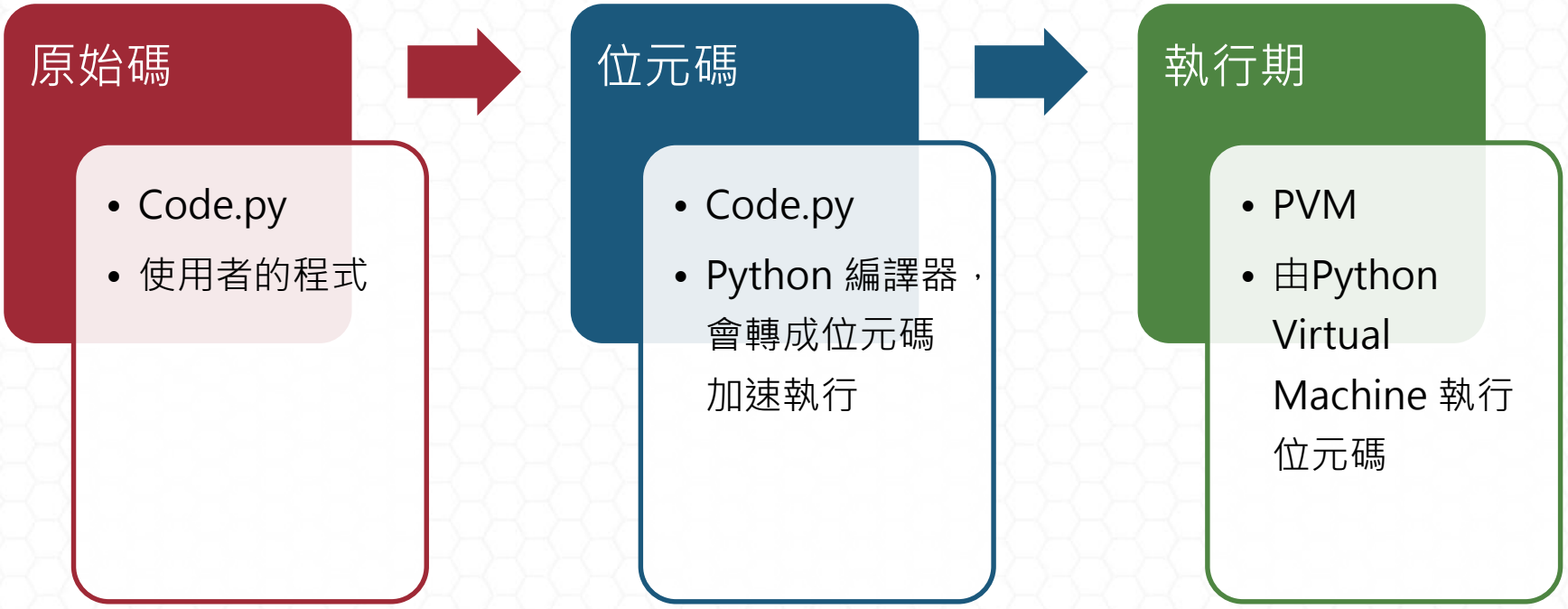
Python 開發工具簡介

安裝Anaconda



https://www.continuum.io/downloads

Python 如何執行


原始碼
• Code.py
• 使用者的程式
位元碼
• Code.py
• Python 編譯器，會轉成位元碼加速執行
執行期
• PVM
• 由Python Virtual Machine 執行位元碼

直譯式語言

- 互動式Shell – 不用編譯，即時看到結果，難以編修程式

```
C:\Users\david>python
>>> print(”hello world”)
hello world
>>> exit()
```

動態語言 (一)

- 不用宣告型態，更簡潔易用

```
>>> a = 2
>>> b = 3
>>> a + b
5
```

動態語言 (二)

```
>>> a = 1+"hello"
>>> print(a)
```

字串與整數無法相加

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-15-b8c992fa3dbd> in <module>()
----> 1 a = 1+"hello"
      2 print a

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Python 2.x v.s. Python 3.x

- *Python 2.x is legacy, Python 3.x is the present and future of the language*

- *Python 3.x*
  - 更簡潔的語法，Unicode 支援與強大內建函式庫
  - 尚在持續更新與開發中

- *Python2.x*
  - 支援主要作業環境
  - 第三方函式庫主要支援版本

Python 2 與 Python 3 的差異

- Python 3 的 Print

```
>>> print('hello world')
hello world
```

- Python 2 的 Print

```
>>> print 'hello world'
hello world
```

可以使用 from __future__ import print_function 取代print

從命令列執行Python

- 互動式Shell – 即時看到結果，難以編修程式

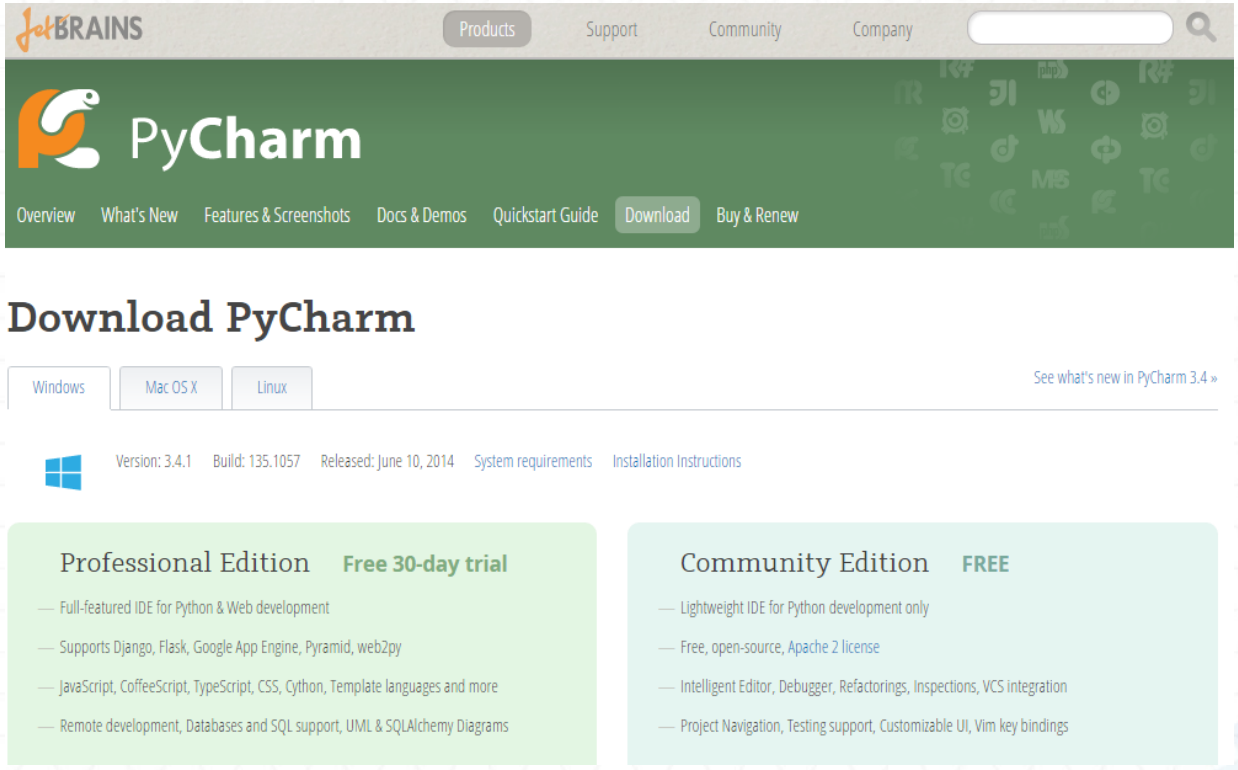
```
C:\Users\david>python
```

- 從檔案執行Python – 無法即時看到結果，容易編修程式

```
C:\Users\david>python test.py
```

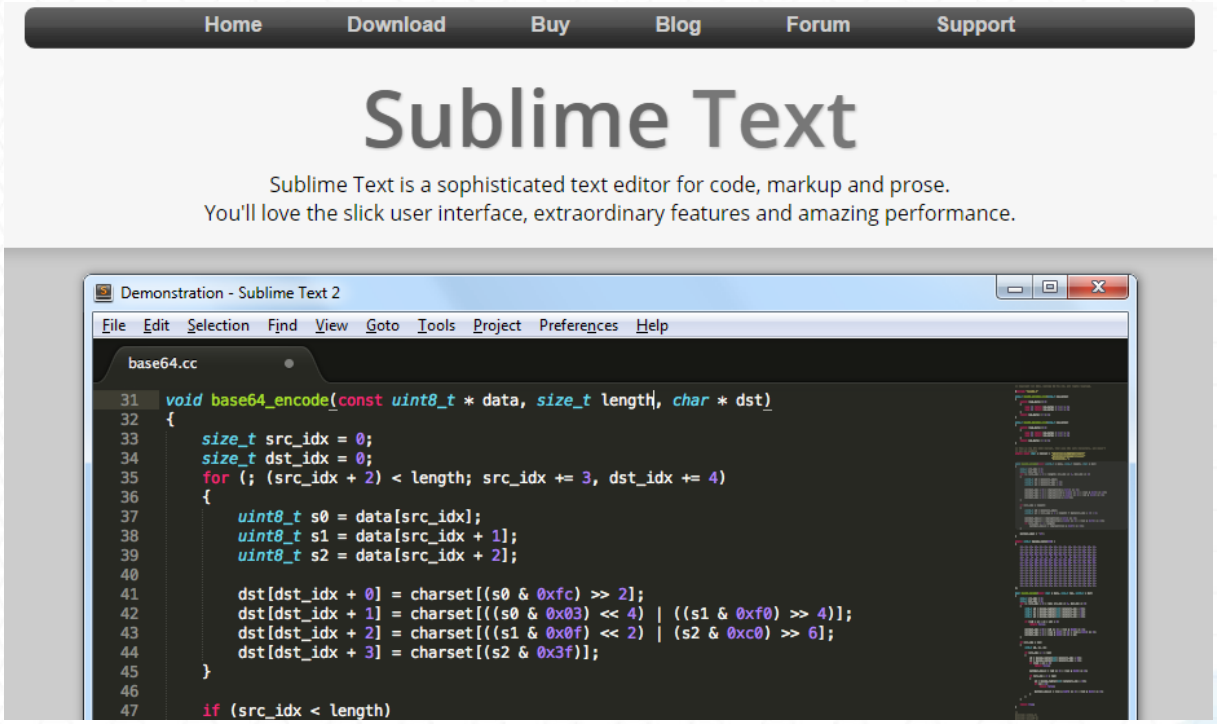
Python IDE - PyCharm

- 建議使用該工具做專案開發
  - https://www.jetbrains.com/pycharm/download/



Sublime Text

- MAC 使用者建議使用該工具做專案開發
  - https://www.sublimetext.com/

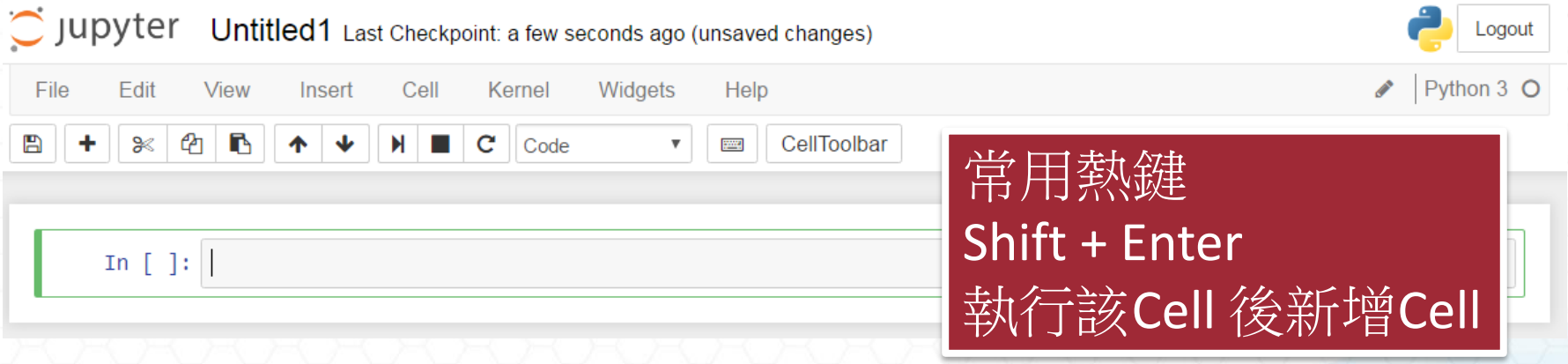


Jupyter Notebook

- 使用於教學與資料探索，不適合開發大型專案

啟用 Jupyter (jupyter Notebook)

- 在命令列下打:
  - jupyter notebook
  - 自動開啟瀏覽器後便可瀏覽 (預設為localhost:8888)
- 可匯出.ipynb, .py 各種不同格式檔案
- 瀏覽快捷鍵 Help -> Keyboard Shortcuts



數字 (Number)

算數還需倚賴計算機?

- 透過Python 可以直接計算數字

```
3 + 2 * 8
```

數字類型

整數Integers

- 整數，有分正數或負數
  - 例如：2和-2

浮點數Floating point Numbers

- 有小數點，或可以使用指數E
  - 例如：2.0和-2.1及4E2

例子	數字類型
1, -5, 1000	整數
1.2 ， -0.5, 2e2, 3E2	浮點數

基本算術

```
# 相加
2+1
# 相減
2-1
# 相乘
2*2
# 相除
3/2
```

在Python 2會得到 1
(整數除以整數得到整數)

但若是在Python 3輸入3/2則會得到1.5

進階算數功能

- 指數運算

```
2**3
```

- 開根號

```
4**0.5
```

- 先乘除後加減

```
2+10*10+3
```

- 使用小括號指定運算順序

```
(2+10)*(10+3)
```

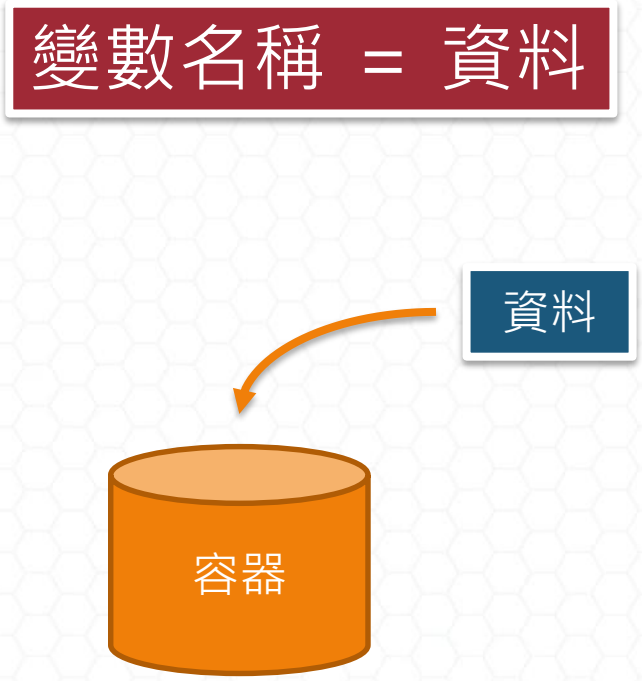
可以如何創建變數？

■ 使用等號

```
a = 5
# Python會將a視為5
a + a
```

■ 可重新命名

```
# 重新賦值
a = 10
a
# 重新定義 a
a = a + a
a
```



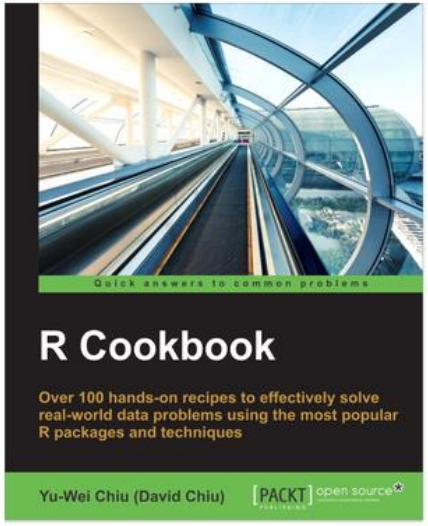
計算台幣價格

price = 25.6

exchange_rate = 32.33

ntd_price= price * exchange_rate

ntd_price



https://goo.gl/wPVmSK

字串 (STRING)

Python 讓文字處理再簡單不過!

- 有沒有試過?
  - 從上千筆記錄將所有英文文字變成大寫
  - 將百篇文章中多餘的空白拿掉
  - 根據特定符號切割文字

http://goo.gl/CC8Qcx

創建字串

- 字串類別 (str) 是內建的

```
a = "hello world"
print(a.__str__)
```

- 使用單引號或雙引號

```
# 使用單引號標註一段話
'This is also a string'
# 使用雙引號標註一段話
"String built with double quotes"
```

因為單引號停止了字串
因此可以使用雙引號和單引號
的組合表達完整字串

```
' I'm using single quotes, but will create an error'
```

SyntaxError: invalid syntax

```
"Now I'm ready to use the single quotes inside a string!"
```

Print 函式

- # 印出字詞

'Hello world'

- # 中文也沒問題

'寶寶心理苦，但寶寶不說'

- # 用Print列出所有結果

```
print('Hello world 1')
print('Hello world 2')
print('Use \n to print a new line')
print('\n')
print('See what i mean?')
```

換行與多行文字處理

- # 使用 \ 作為換行符號

```
a = 'hi this is a l\
ong text'
print(a)
```

- # 使用三個single quote 做多行處理

```
a = '''hi this is a l
ong text'''
print(a)
```

Print – 特殊字元

- \n – 換行字元

```
print('Here is a new line \n and here is the second line')
```

- \t – 以表格、或表格形式排列資料

```
print('Here is a new line \t and here is the second line')
```

字串索引 (1/2)

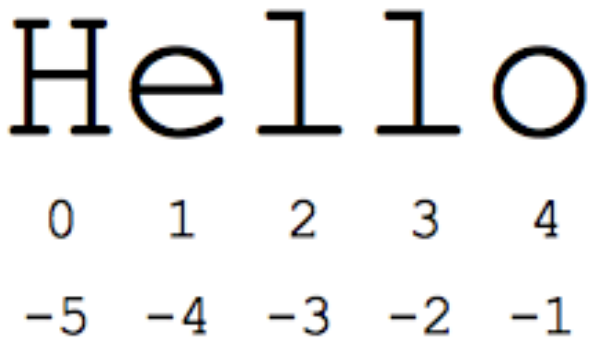
```
s = 'Hello'
print(s)
```

- Python的編制索引為從0開始

```
s[0]
s[1]
```

- 也可使用負號從後面開始索引

```
s[-1]
s[-2]
```



字串索引 (2/3)

- 用：來選定起始點或迄點等範圍

```
#  取得從第一個索引之後的所有元素
s[1:]
s #  並不會更改原字串

#  取到索引為第 3 個的元素
s[:3]
#  取得所有元素
s[:]
```

代表抓取從第0~第3個元素
但不包括第3個元素本身

字串索引 (3/3)

```
#  for(int i=0; i< len(s); i++)
s[::1]
#  for(int i=0; i< len(s); i+=2)
s[::2]
#  for(int i=len(s) -1 ; i > 0; i--)
s[::-1]
```

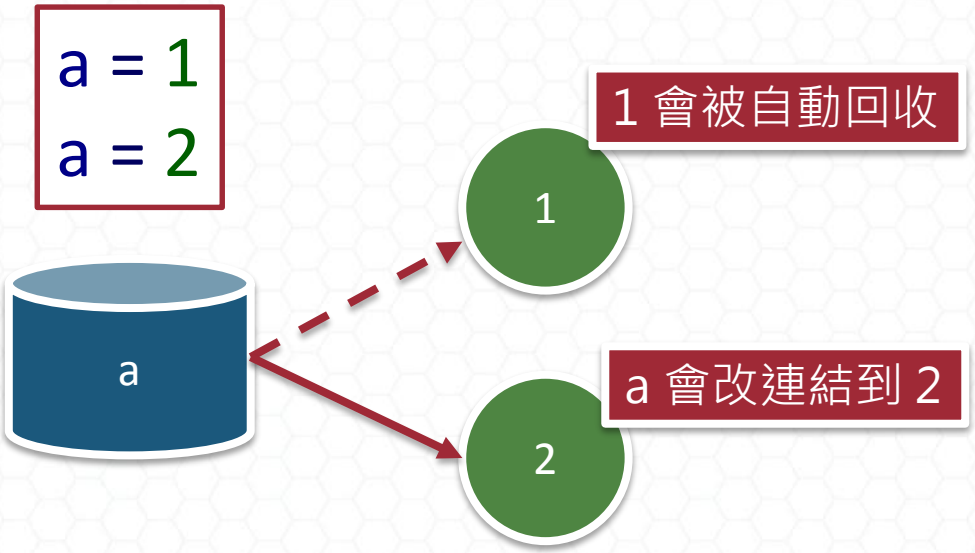
字串特性 (1/3)

- 不變性：一旦創建一個字符串，其內的元件不能被改變或替換

```
s
#  無法修改裡面的內容
s[0] = 'x'
---------------------------------------------------
TypeError           Traceback (most recent call last)
<iphtyhon-input-49-3a9c668aa5ab> in <module>()
      1 # #  Let's try to change the first letter to 'x'
------>2 s[0] = 'x''x'
TypeError: 'str' object does not support item assignment
```

不變性

- 整數、浮點數、字串、Tuple 是不可變
- 當物件不再被使用時，Python 會自動做垃圾收集 (Garbage Collection) 釋放記憶體空間。



字串特性 (2/3)

- 使用+做字串的連接：

```
s
s + 'concatenate me!'
s = s + 'concatenate me!'
print(s)
s
```

字串特性 (3/3)

- 可用乘號重複創建同一元素：

```
letter = "?"
letter
letter *10
```

字串的其他功能

- .upper( ) 大寫

```
s.upper()
```

- .lower( )：小寫

```
s.lower()
```

- .split( )：分隔

```
s.split()
s.split('W')
```

- 使用函數len()來檢查的字串的長度：

```
len('Hello World')
```

可以用dir(s) 做功能的查詢

輸出格式 (PRINT FORMATTING)

當我想要用樣版替代部分文字時

```
name = 'David'
bank = ‘Cathay'

print(bank + "is " + name + "'s favorite")
```

有沒有更好的做法?

字串

```
#使用 %s 印出格式化的字串：

x = 'String'
print('Place my variable here: %s' %(x))

x = 13.3
print('Place my variable here: %s' %(x))
```

浮點數

- 用 **%n1.n2f** 來表示數字

```
print('Floating point number: %1.2f' %(13.145))

print('Floating point number: %1.10f' %(13.145))

print('Floating point number: %25.3f' %(13.145))
```

%s 與 %r

- %s是str() ； %r是repr()，在python中皆可用作字串的轉換：

```
print('Here is a number: %s. Here is a string:
%s' %(123.1,'hi'))

print('Here is a number: %r. Here is a string:
%r' %(123.1,'hi'))
```

字串Format 處理

```
# 帶有 format 的 String
raw = '\n\t this is a \t line with format \t\n'
print(raw)

# 使用 r 取得raw string
raw = r'\n\t this is a \t line with format \t\n'
print(raw)

# 使用 repr 取得string 中的表示
raw = repr('\n\t this is a \t line with format \t\n')
print(raw)
```

使用.format() 方法'

```
# 根據樣式替代
print('This is a string with an {p}'.format(p='insert'))

# 根據樣式替代多字串
print('One: {p}, Two: {p}, Three: {p}'.format(p='Hi!'))

# 根據多樣式替代
print('Object 1: {a}, Object 2: {b}, Object 3:
{c}'.format(a=1,b='two',c=12.3))

# 根據位置替代
print('One: {0}, Two: {1}, One: {0}'.format(1,2))
```

比起%s這是比較好的方法

資料輸入 (INPUT)

跟Python 產生互動吧?

要怎麼跟 Python 產生互動呢?

http://goo.gl/qakBy9

使用input函式

```
>>> print(“Hello from python”)
Hello from python
>>> input("How are you?")
How are you?"Cool"
'Cool’
>>> answer = input("How are you?")
How are you?"cool"
>>> answer
'cool'
```

使用print 產生訊息

使用input 取得輸入

將輸入放置於answer中

使用raw_input

- answer = input("how are you?")

字串需要用雙引號包覆

- answer = raw_input("how are you?")

字串不需要用雙引號包覆

串列(LIST)

記錄多人的資料

http://goo.gl/TkRhfz

有沒有一個容器能同時記錄多個人的資料?

- 電話
- 姓名
- 性別

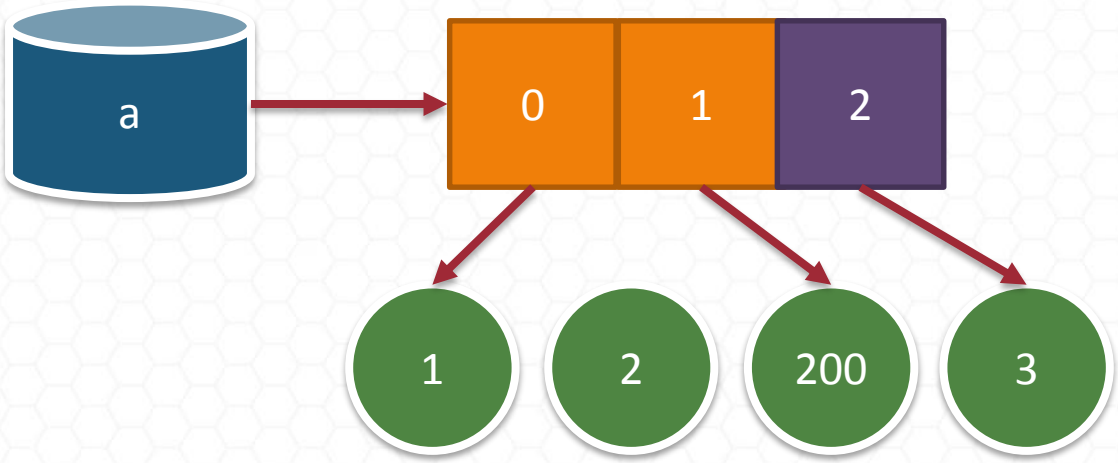
Python List

- 剛剛介紹到的數值與字串單為單一值，如果要存放多個值時，可以用設麼資料結構?
  - List
- 其他語言都有陣列的概念(Array)，Python 的List 比較接近於ArrayList
- 宣告List 的方式
  - a = []
  - a = list()

可變性

- 串列(List) 與字典(Dictionary) 屬於可變型態

```
a = [1,2]
a.append(3)
a[1] = 200
print(a)
```



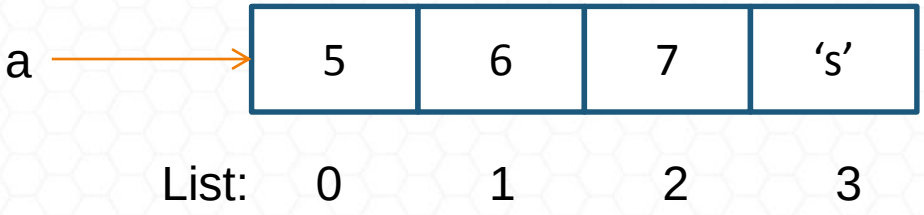
List

- 可以在list 之中存放不同類型的物件

```
my_list = [1,2,3]

my_list = ['A string',23,100.232,'o']

len(my_list)
```

List 基本操作

```
a = [5, 6, 7, 's'] # a = [5 6 7 ‘s’]
print(a[0]) # 5
print(a[2:4]) # [7, ‘s’]
print(a[-1]) # ‘s’
print(a[-2]) # 7
print(a[2:]) # [7, ‘s’]
print(a[::2]) # [5, 7]
print(a[::-1]) # [‘s’, 7, 6, 5]
print(len(a)) # 4
print([min(a), max(a)]) # [5, ‘s’]
```



List 元素增減

```
a = [5, 6, 7, 8]
a.pop() # a = [5, 6, 7]
a.append(2) # a = [5, 6, 7, 2]
a.sort() # a = [2, 5, 6, 7]
a.reverse() # a = [7, 6, 5, 2]
```

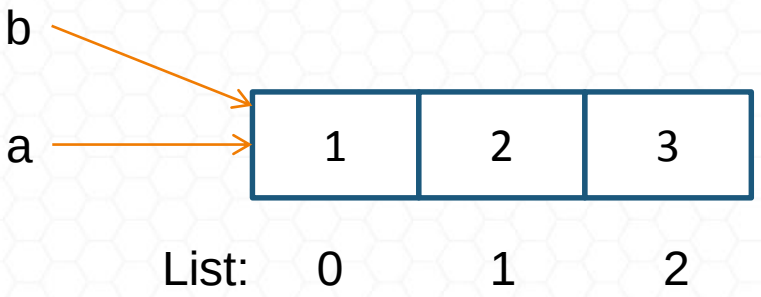
套用List 於字串中

```
>>> list('a')
['a']
>>>hello =  list('hello world')
>>>print(hello)
['h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r',
'l', 'd']
>>> 'e' in hello
True
>>> 'a' in hello
False
```

指定(Assignment)陣列

```
a = [1, 2, 3]
b = a
a[1] = 2000
b
```

[1, 2000, 3]



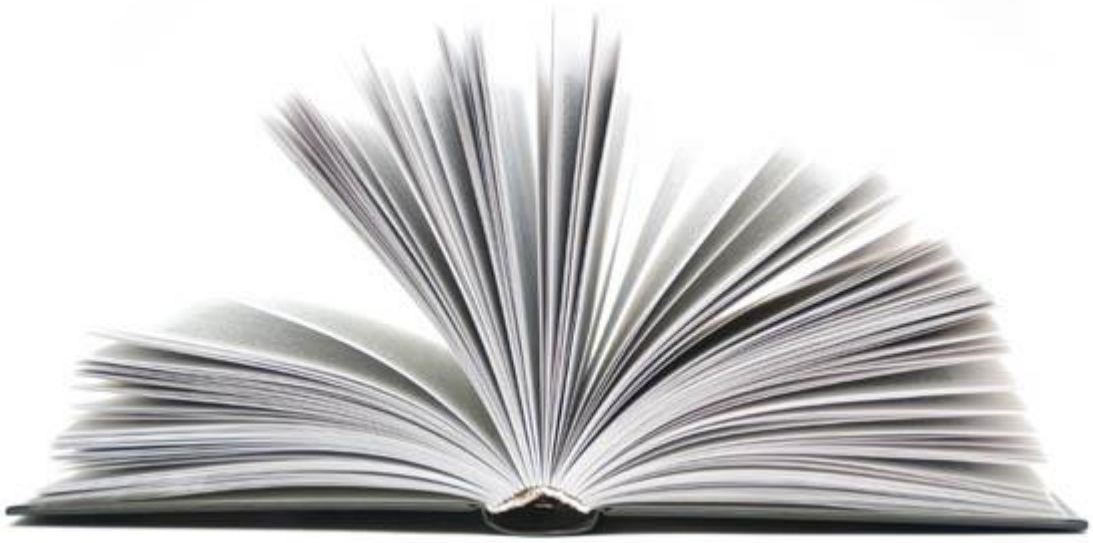
複製List – 解決方法

```
a = [1, 2, 3]

import copy
aa = copy.deepcopy(a) # Deep Copy
a[1] = 2000
print(aa)
```

字典(DICTIONARY)

從書中快速找到想要的資料?

- 建立索引頁或許是個比較好的方法



http://goo.gl/37cpjL

Python 字典(Dictionary)

- 其他語言有相同的操作
  - Java: HashMap, HashTable…
  - C++: hashmap
  - C#: Dictionary…
- dic = {key : value}
- 其中key 為唯一不重複值

Dictionary 範例

```
# 建立字典
dic = {'a':100, 'b':"yes", 'c':0.98}
print(dic)

# 取得keys
print(dic.keys())

# 取得values
print(dic.values())

# 取得單一key 所對到的內容
print(dic['a'])

# 取得單一key 所對到的內容 (有預設值, 比較推薦的方法)
print(dic.get('a'))
```

Dictionary 範例(二)

```
# 增加資料到字典中
dic['d'] = 'new'
print(dic)

# 更新字典內的元素
dic.update({'e':123})
print(dic)

# 遍歷整個字典資料
for rec in dic:
    print(rec, dic[rec])
```

元組(TUPLES)

Tuple

- 跟List不同，在Tuple 內的值無法被更改
  - 在記憶體中有固定大小
- 建立Tuple 的方式
  - tuple1 = (1,2,3)
  - tuple2 = 1,2,3
  - tuple3 = tuple([1,2,3])
- 檢視tuple 跟 list 的差異
  - dir(tuple)
  - dir(list)

當希望資料能維持一致性，而內容值不能被修改時(例如資料庫回傳資料)，可以使用Tuple。

建立 Tuples

```
# 可以混和不同資料型態
t = ('one',2)

# 可以檢查Tuple 的長度
len(t)

# 根據位置取用值
t[0]
t[-1]
```

Tuple 方法

```
# 使用 .index 取用資料所在位置
t.index('one')

# 使用 .count 計算資料出現次數
t.count('one')
```

不變性 Immutability

```
t[0]= ‘change’
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-93def5f9b4bd> in <module>()
----> 1 t[0]= 'change'

TypeError: 'tuple' object does not support item assignment

t.append('nope’)
--------------------------------------------------------------------------- AttributeError
Traceback (most recent call last)
<ipython-input-15-799b3447c4d9> in <module>()
----> 1 t.append('nope')

AttributeError: 'tuple' object has no attribute 'append'
```

Packing 與 Unpacking

```
>>> a,b = 1,2
>>> 1
1
>>> a,b = 1,2
>>> a
1
>>> b
2
>>> c = a,b
>>> c
(1, 2)
>>> d = 1,2
>>> c ==d
True
```

刪除變數

- 使用del 可以刪除變數

```
>>> del a
>>> del b
```

- 如果要交換a,b 元素，要怎麼做?

```
>>> c = a
>>> a = b
>>> b = c
>>> del c
```

使用tuple 交換變數

```
>>> a,b = b,a
>>> a
2
>>> b
1
```

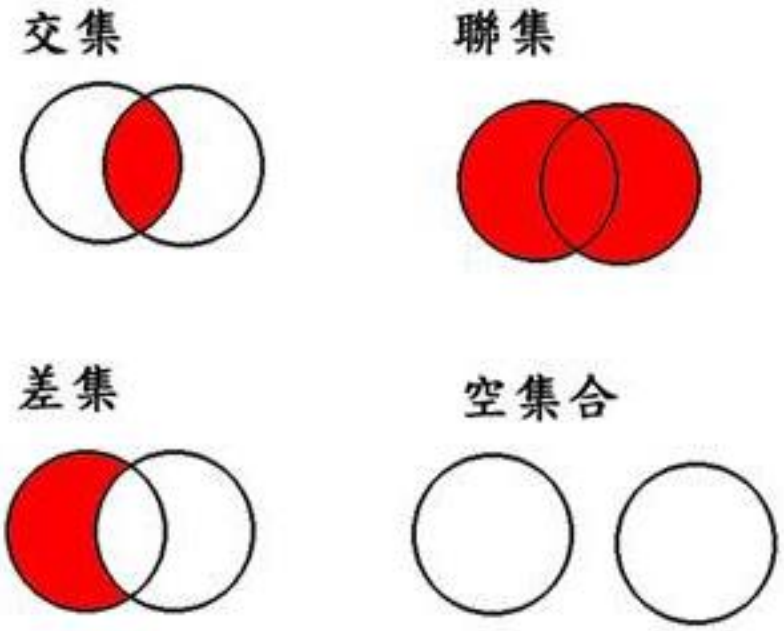
交換變數

神奇!?

集合(SET)

集合

- 當要從兩堆資料中找到之間的相同處與相異處時可以使用集合



http://goo.gl/xRiuwd

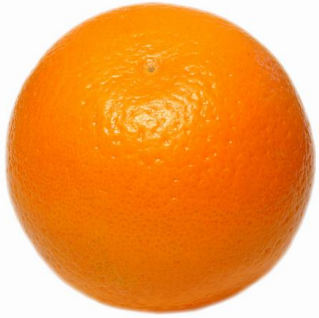
Sets

```
x = set()

# 使用 add() 增添資料
x.add(1)
x
x.add(2)
x

# 可以將一個list 轉為 set
l = [1,1,2,2,3,4,5,6,1,1]
set(l)
```

Python 運算式

比較兩個變數

- 如何比較兩個變數之間的異同?



Operator

http://goo.gl/lLnFJT

http://goo.gl/gyxYUl

簡單比較動作

```
# 比較大小
a = 5
b = 3
a > b
b > a
```

```
# 比較是否相同
a = True
b = False
a == b
a != b
```

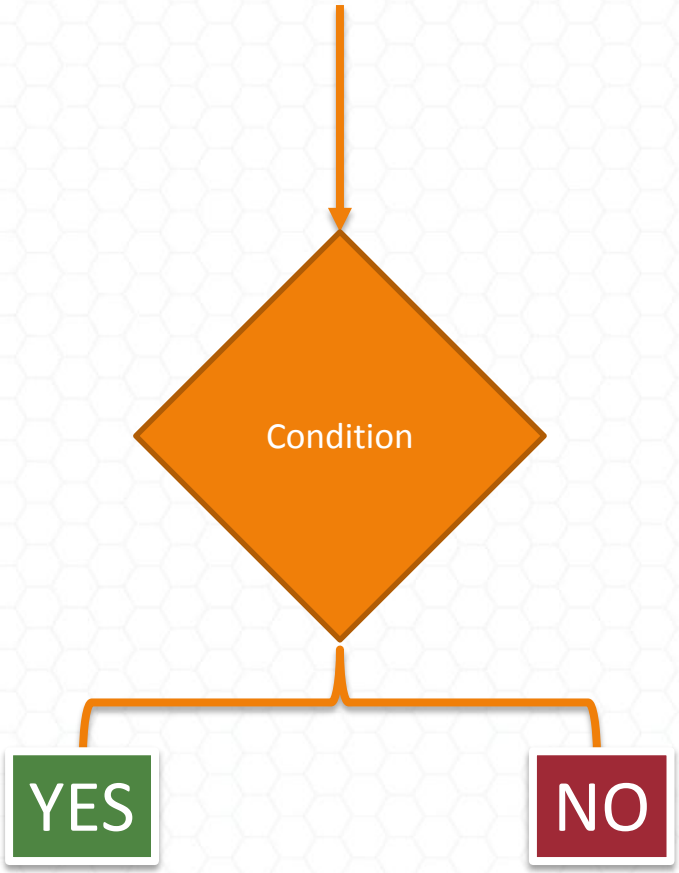
簡單比較動作 (2)

```
#將結果指回c
c = a == b
print(c)

# 使用 Not
print(not a)
```

```
# 比對not a 與 b
c =(not a== b)
print(c)

# 比對None
a = None
b = None
print(a ==b)
```

Python 陳述(Statement)

對不同選擇做抉擇



陳述式

■ 其他語言的陳述式

```
if (a>b){
    a = 2;
    b = 4;
}
```

◆ 使用 {} 表述陳述式
◆ 使用 ; 表示結尾

■ Python 語言陳述式

```
if a>b:
    a = 2
    b = 4
```

◆ 使用 : 表述陳述式
◆ 使用indent

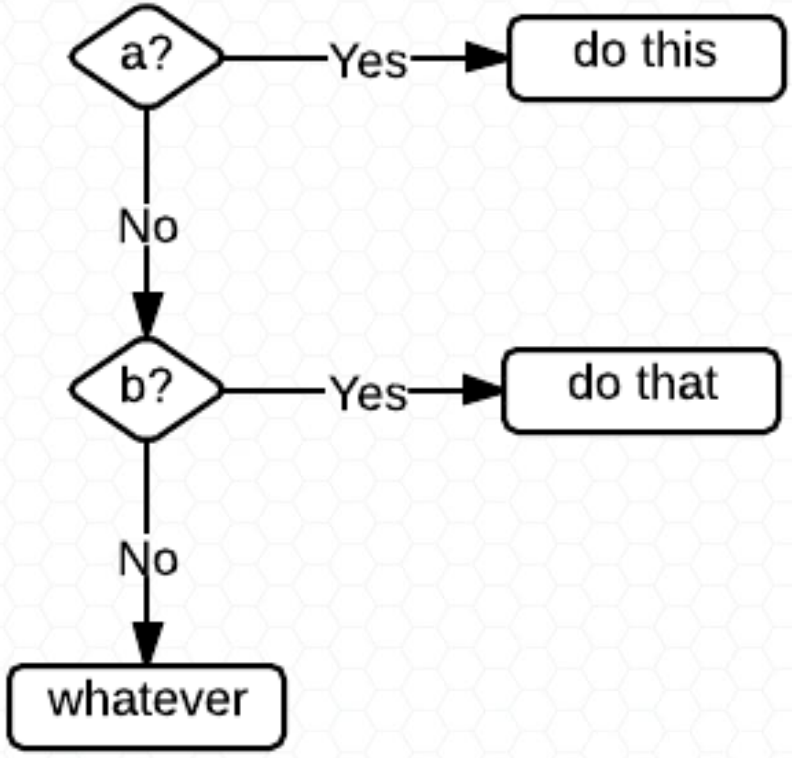
程式縮排 Indentation

- 使用 tab 或空白去區隔執行區塊(替代傳統的{})
- Pep8規定使用4個空白
  http://legacy.python.org/dev/peps/pep-0008/#indentation
- Google 使用2個空白

IF, ELIF, ELSE 陳述

把決策過程轉換成程式碼

```
if a:
    do this
elif b:
    do that
else:
    whatever
```



http://goo.gl/j5UCtT

第一個例子

```
if True:
    print('It was true!')

x = False

if x:
    print('x was True!')
else:
    print('I will be printed in any case where x is not true')
```

If…elif…else

```
loc = 'Bank'

if loc == 'Auto Shop':
    print('Welcome to the Auto Shop!')
elif loc == 'Bank':
    print('Welcome to the bank!')
else:
    print('Where are you?')
```

SWITCH…CASE

- Python 沒有Switch…Case，但可以使用字典模擬：

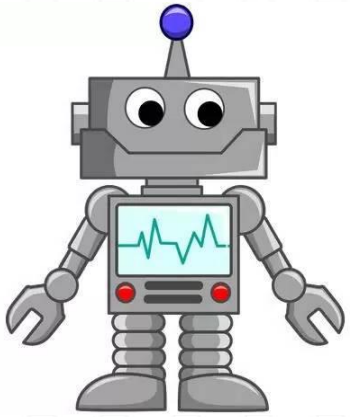
- Python 版的switch…case (使用字典):

```
def func(x):
    return {
        'a': 1,
        'b': 2,
        }.get(x, 0)
```

For 迴圈 (For Loop)

For 迴圈

- 就像讓個機器人重複執行同樣的動作數次



1	2	3	4	5

http://goo.gl/emhFyz

For 迴圈例子

```
# 印出 list 中的每個元素
l = [1,2,3,4,5,6,7,8,9,10]
for num in l:
        print(num)

# range(1,11) 可以產生跟 l 一樣的list
print range(1,11)
for num in  range(1,11):
        print(num)
```

For 迴圈例子 (2)

```
# 從1加到10
list_sum = 0
for num in l:
        list_sum = list_sum + num
print(list_sum)

# 可以用 += 簡寫
list_sum = 0
for num in l:
        list_sum += num
print(list_sum)
```

For 迴圈操作字串與Tuple

```
# 取出每一個單字
for letter in 'This is a string.':
    print(letter)

# 取出Tuple 中每一個元素
tup = (1,2,3,4,5)
for t in tup:
    print(t)
```

使用For 迴圈操作Tuple

```
l = [(2,4),(6,8),(10,12)]

for tup in l:
    print(tup)

# Unpacking
for (t1,t2) in l:
    print(t1)
```

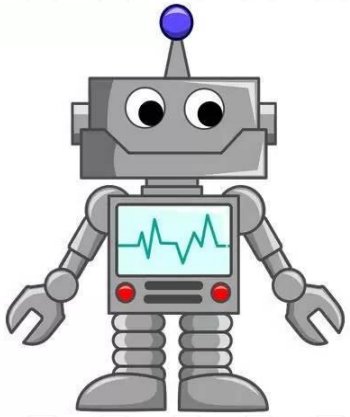
使用For 迴圈操作字典

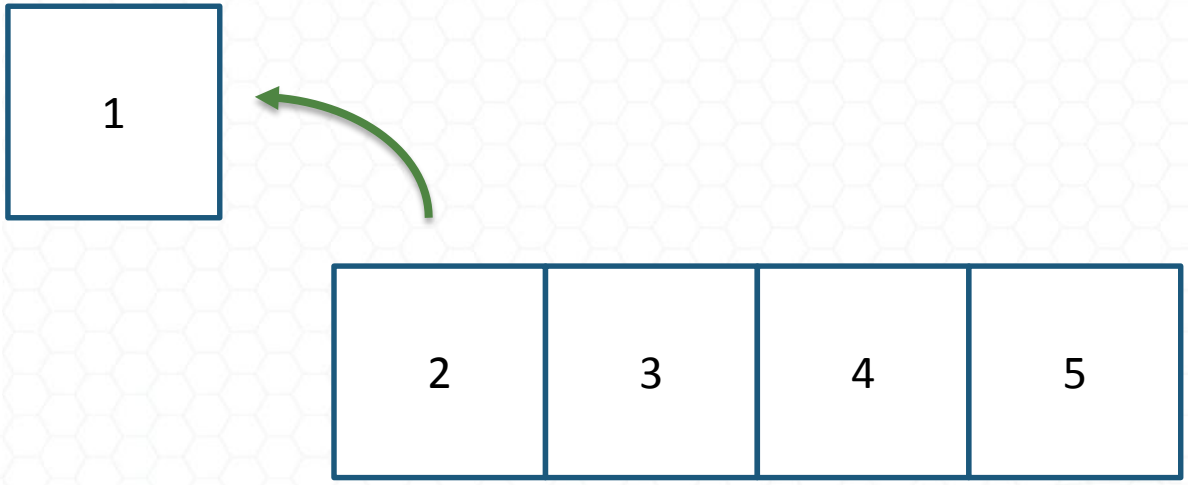
```
d = {'k1':1,'k2':2,'k3':3}
for item in d:
    print(item)

# python 2
for k, v in d.items():
    print(k, v)

# python 3
for k, v in d.items():
    print((k, v))
```

迭代器 (Iterator)

Iterator

- 如果要一一取出內容物件呢?





http://goo.gl/emhFyz

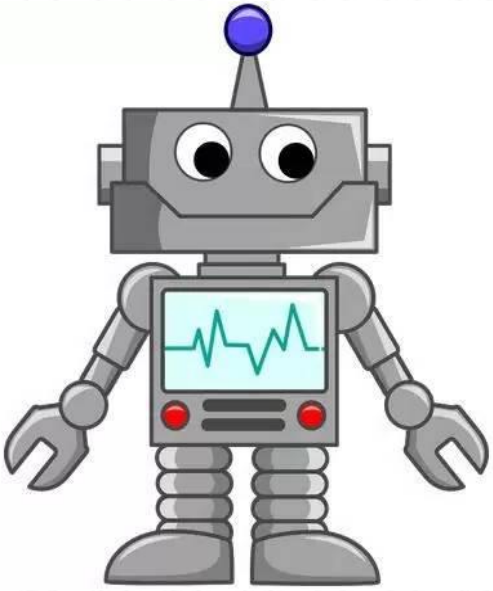
next() & iter()

```
s = [1,2,3,4,5]
s_iter = iter(s)
print(next(s_iter))
print(next(s_iter))
```

While 迴圈 (While Loop)

while 迴圈

- 如何讓機器人無限次不斷重複執行同樣的動作



http://goo.gl/emhFyz

while 迴圈

```
x = 0

while x < 10:
        print('x is currently: ',x)
        print('x is still less than 10, adding 1 to x')
        x+=1
```

可以配合else 使用

```
x = 0

while x < 10:
        print('x is currently: ',x)
        print('x is still less than 10, adding 1 to x')
        x+=1
else:
        print('All Done!')
```

break, continue, pass (1/3)

```
x = 0

while x < 10:
        print('x is currently: ',x )
        print(' x is still less than 10, adding 1 to x' )
        x+=1
        if x ==3:
                print('x==3')
        else:
                print('continuing...' )
                continue
```

Break 暫停迴圈
Continue 持續執行迴圈
Pass 略過這一迴圈

break, continue, pass (2/3)

```
x = 0

while x < 10:
    print 'x is currently: ',x
    print ' x is still less than 10, adding 1 to x'
    x+=1
    if x ==3:
        print('Breaking because x==3' )
        break
    else:
        print('continuing...')
        continue
```

無窮迴圈

```
# 絕對不能執行這段程式碼
while True:
        print('Uh Oh infinite Loop!')
```

檔案操作(FILE)

操作檔案

- 透過檔案，我們可以保存與讀取資料

1, David, M
2, Mary, F
3, John, M

寫入檔案

```
#將Hello World 寫入檔案中
fid = open('test.txt', 'w')
fid.write('Hello\nWorld')
fid.close()

# 使用with 自動關檔
with open('test.txt', 'w') as f:
        f.write('Hello\nWorld')
```

讀取檔案

```
# 一行一行讀取檔案
with open('test.txt', 'r') as fid:
    for line in fid:
        print("Line: " + line.strip())

# 讀取檔案中所有內容
with open('test.txt', 'r') as fid:
    s = fid.read()
print("line",s)
```

計算行數

```
#計算行數
k = 0
with open('1.tsv') as fid:
    for line in fid:
        k = k + 1
print(k)

#或簡寫為:
print(len([ line for line in open('test.txt')]))
```

方法簡介(Method)

Python 函式

- Python 物件導向語言 (OOP)
- 在Python 中，任何東西都是物件
- 每個物件都有自己的函式(方法)，封裝對物件的操作，減少使用者花費時間撰寫重覆的程式碼

參數

object.method(arg1,arg2,etc...)

使用Help 與 dir 查詢套件與函式

```
>>> import sys
>>> help(sys)
```

使用help 查詢文件

```
>>> dir(sys)
```

使用dir 表列可用屬性與方法

```
>>> help(sys.exit)
>>> ?sys.exit
```

不確定該方法的功能? 使用help

內建函式範例

```
l = [1,2,3,4,5]
dir(l.count)

l.append(2)
peint(l)
l.count(2)

help(l.count)
?l.count
```

函式簡介(FUNCTION)

函式

- 希望能寫一次，但在不同情境可以重複呼叫好幾次
- Python 定義函式的語句: def
  - Defining a function

```
def say_hello():
    print("hello world")
```

定義簡單函式

```
def addNum(a, b):
    return(a+b) #Return

print addNum(1,4)
```

得到回傳結果

函式範例

```
def add_num(num1,num2):
      return(num1+num2)
add_num(4,5)

result = add_num(4,5)
print(result)
print(add_num('one','two'))
```

可以使用+

- 加總兩個數字
- 合併兩個字串

接受多個參數

```
def addall(x, *args):
    res = x
    for y in args:
        res += y
    return(res)

addall(1,2,3,4,5)
```

接受多個 key/value 參數

```
def make_two_lists(**kwargs):
    keys,values = [],[]
    for k,v in kwargs.items():
        keys.append(k)
        values.append(v)
    return([keys,values])

make_two_lists(david='M', Mary = 'F', John='M')
```

Lambda 運算式
(Lambda expression)

不想寫落落長的函式

- Lambda 可以使用了單一陳述句建立簡單匿名函式，簡單方便，用完即丟。

比較Lambda 與函式

■ 一般函式

```
def square(num):
    result = num**2
    return(result)
square(2)

def square(num):
    return(num**2)
square(2)
```

■ Lambda 函式

```
def square(num): return num**2
square(2)

lambda num: num**2

square = lambda num: num**2
square(2)
```

Lambda 範例

```
# 取偶數
even = lambda x: x%2==0
even(3)
even(4)

# 取第一個元素
first = lambda s: s[0]
first('hello')
```

```
# 字串倒排
rev = lambda s: s[::-1]
rev('hello')

# 兩個元素相加
adder = lambda x,y : x+y
adder(2,3)
```

巢狀陳述與範圍
(Nested Statement and scope)

變數範圍

- 在Python 中每個變數都被存在各字的命名空間(namespace)，不同命名空間變數的可視範圍皆不相同

```
x = 25

def printer():
    x = 50
    return(x)
print(x )
print(printer())
```

範圍搜尋順序

- LEGB 規則
  - Local
  - Enclosing functions
  - Global (模組)
  - Built-in (Python)

LEGB 範例

```
f = lambda x:x**2  # x is local

name = 'This is a global name'

def greet():
    # Enclosing function
    name = 'Sammy'  # Enclosing function locals
    def hello():
        print('Hello '+name )

    hello()

greet()
print(name) # Global
len # Built-in function
```

本地變數

```
x = 50

def func(x):
    print('x is', x)
    x = 2
    print('Changed local x to', x )

func(x)
print('x is still', x)
```

Global

```
x = 50

def func():
    global x
    print('This function is now using the global x!' )
    print('Because of global x is: ', x)
    x = 2
    print('Ran func(), changed global x to', x )

print('Before calling func(), x is: ', x)
func()
print('Value of x (outside of func()) is: ', x )
```

讓變數變成Global

錯誤與例外
(Errors and Exceptions)

錯誤訊息

- 寫程式出錯是不可避免的，但該怎麼處理錯誤情況呢？

IOError

```
with open('testfile','r')  as f:
    f.write('Test write this')
```

```
---------------------------------------------------------------------------
IOError                                   Traceback (most recent call last)
<ipython-input-18-8a78d5e09a11> in <module>()
      1 with open('testfile','r')  as f:
----> 2     f.write('Test write this')

IOError: File not open for writing
```

IOError

處理 IOError

```
try:
    f = open('testfile','w')
    f.write('Test write this')
except IOError: #處理IOError
    print("Error: Could not find file or read data")
else:
    print("Content written succesfully")
f.close()
```

除以0 的時候的處理：Try & Except

```
dividend = raw_input("Dividend: ")
divisor = raw_input("Divisor: ")

try:
    print(float(dividend) / float(divisor))
except ZeroDivisionError as detail:
    print("ZeroDvisionError", detail)
```

想像是try catch – 針對除以零時的對應動作

Try & Except

```
dividend = raw_input("Dividend: ")
divisor = raw_input("Divisor: ")
```

與上面例子的不同?

```
try:
    print(float(dividend) / float(divisor))
except:
    print(¨Error¨)
```

```
Dividend: "qoo"
Divisor: 3
error
```

Finally

```
dividend = raw_input("Dividend: ")
divisor = raw_input("Divisor: ")

try:
    print(float(dividend) / float(divisor))
except:
    print("Error")
finally:
    print('final block')
```

最後執行的區塊

物件 (Object)

物件

- Python 是物件導向語言 (OOP)
- 在Python 中，任何東西都是物件
- 每個物件都有:
  - id
  - type
  - value

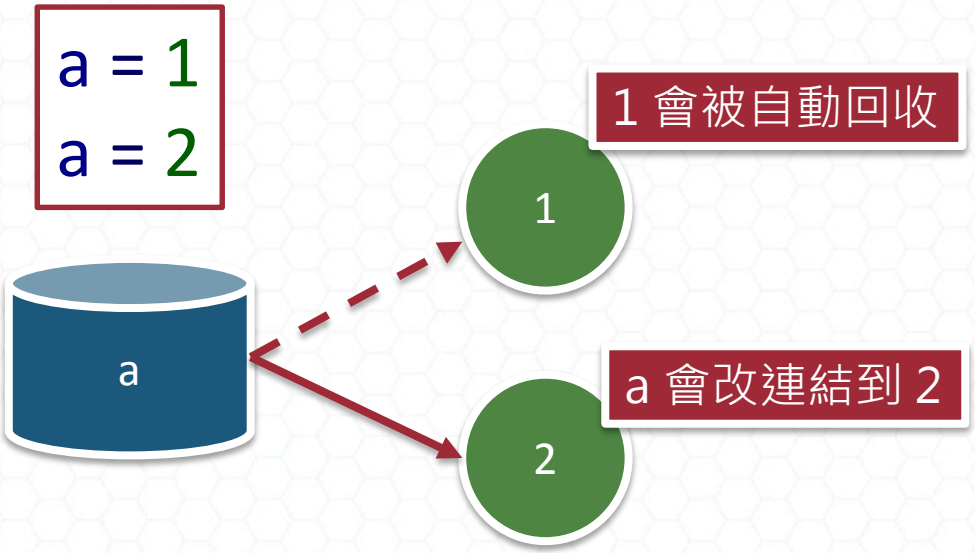
```
a = 1
print(id(a))
print(type(a))
print(a)
```

Python 所有東西都是物件

```
print(type(1))
print(type([]))
print(type(()))
print(type({}))
```

不變性

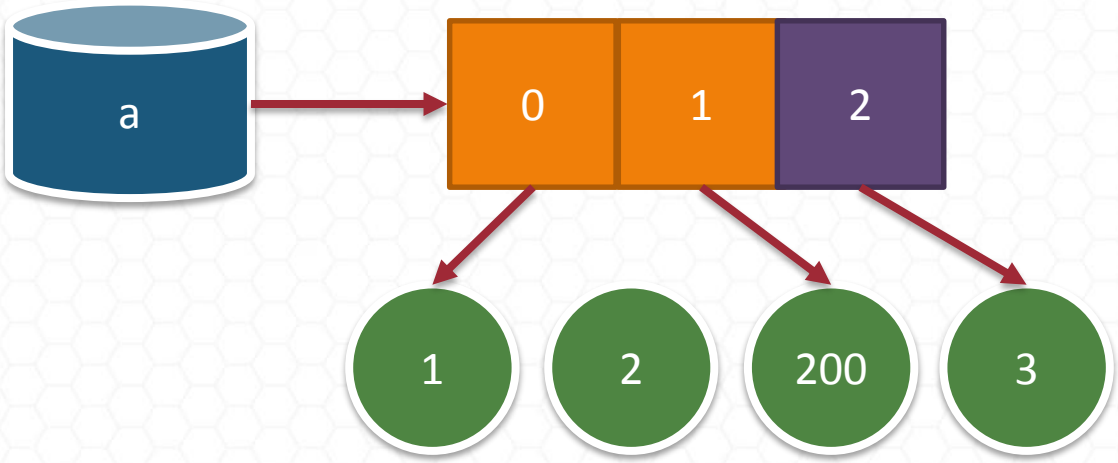
- 整數、浮點數、字串、Tuple 是不可變
- 當物件不再被使用時，Python 會自動做垃圾收集 (Garbage Collection) 釋放記憶體空間。



可變性

- 串列(List) 與字典(Dictionary) 屬於可變型態

```
a = [1,2]
a.append(3)
a[1] = 200
print(a)
```



可變與不可變

- 不可變的資料在賦值時會產生新id
- 可變的資料在賦值時會對應到原id

```
a = [1,2,3]
b = [1,2,3]
print(id(a), id(b))

a = 1
b = 1
print(id(a), id(b))
```

類別 (Class)

class

- 使用者可以使用class 定義自己的類別
- 從類別我們可以建立實例(instance)

```
class Sample(object):
    pass

x = Sample() # new instance

print(type(x))
```

屬性與方法

- 物件包含屬性(attribute) 與方法(method)
  - 屬性: 物件的特性
  - 方法: 對物件的操作
  - 例如: 可以建立一隻”狗”的物件，包含”品種”為其屬性，”叫” 則為狗的方法

建構子__init__

- 物件是藉由建構子建立物件
- 建構子方法是 __init__()
- 使用者可以在類別中使用def __init__() 定義建構子

```
class Dog(object):
    def __init__(self,breed):
        self.breed = breed

sam = Dog(breed='Lab')
frank = Dog(breed='Huskie')
sam.breed
frank.breed
```

定義屬性

self.attribute = something

類別屬性

- 類別屬性是公開的，類別以外的地方也可以存取

```
class Dog(object):
    species = 'mammal' # 類別屬性
    def __init__(self,breed,name):
        self.breed = breed
        self.name = name
sam = Dog('Lab','Sam')
sam.name
sam.species
Dog.species # 類別屬性是公開的
```

DRY 原則

- Don’t Repeat Yourself
- 同樣的動作應該被放置於Function 中
- 同樣的Function 應該被放置於Class 中

方法 (METHOD)

方法

- 類別內的方法是封裝(Encapsulation) 的重要概念
- 可以想成是針對物件屬性的操作

```
class Circle(object):
    pi = 3.14
    def __init__(self, radius=1):
        self.radius = radius
    def area(self):
        return self.radius * self.radius * Circle.pi
    def setRadius(self, radius):
        self.radius = radius
    def getRadius(self):
        return self.radius
```

```
c = Circle()
c.setRadius(2)
print(c.getRadius())
print(c.area())
```

繼承 (Inheritance)

繼承

- 當要建立的新類別與已經存在的類別有許多共通的屬性與方法，可以繼承該已存在類別
- 被繼承的類別稱為父類別(superclass)
- 繼承者為子類別 (subclass)
- 優點為重用(reuse)已存在的程式碼，並且降低程式的複雜度

建立Animal 類別

```
class Animal(object):
    def __init__(self):
        print("Animal created")

    def whoAmI(self):
        print("Animal")

    def eat(self):
        print("Eating")

a = Animal()
a.eat()
```

建立Dog 類別繼承Animal 類別

```
class Dog(Animal):
    def __init__(self):
        Animal.__init__(self)
        print "Dog created"

    def whoAmI(self):
        print "Dog"

    def bark(self):
        print "Woof!"

d = Dog()
d.whoAmI()
d.eat()
d.bark()
```

繼承Animal 類別

改寫Animal 內的方法

可以使用 Animal 內定義的方法

特殊方法
(Special Methods)

特殊方法

- 每個物件都有基本方法如: __init__(), __str__(), __len__() 以及 __del__()
- 特殊方法通常都以底線(e.g. __init__) 包覆
- 在定義類別時可以修改這些特殊方法

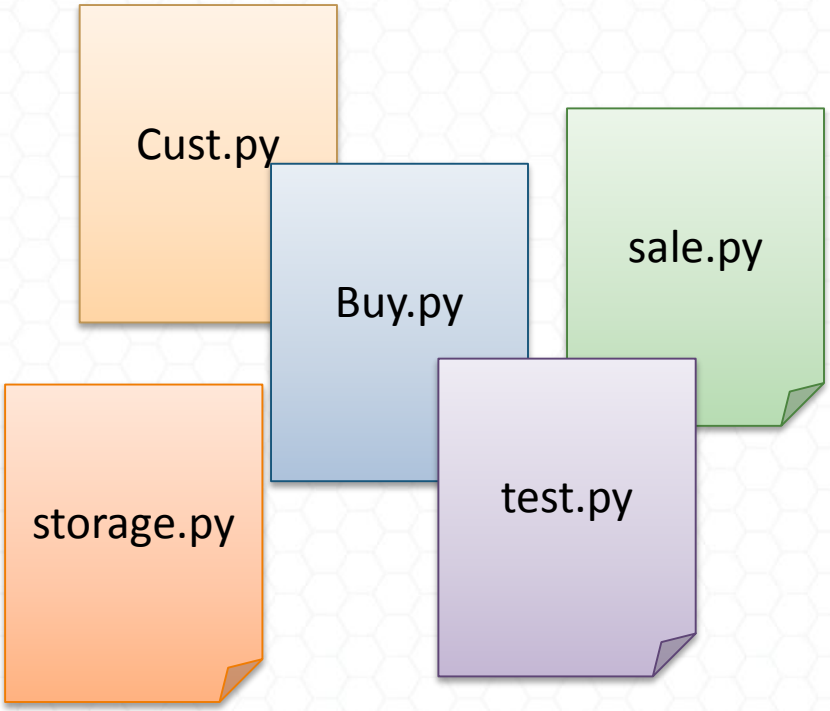
覆寫特殊方法

```
class Book(object):
    def __init__(self, title, author, pages):
        print("A book is created")
        self.title = title
        self.author = author
        self.pages = pages

    def __str__(self):
        return("Title:%s , author:%s, pages:%s " %(self.title, self.author,
self.pages))

    def __len__(self):
        return(self.pages)

    def __del__(self):
        print("A book is destroyed")
```

```
book = Book("Python Rocks!", "Jose Portilla", 159)

#Special Methods
print(book)
print(len(book))
del book
```

模組 (Module)

如何有效整理程式碼?



模組

- Python 模組只是以.py 做為副檔名的Python 程式
- 通常模組內包覆多組函式可供使用者使用
- 使用者可以透過import 指令匯入模組
- 所有模組只會被載入一次，重複 import 不會導致函式重複被宣告

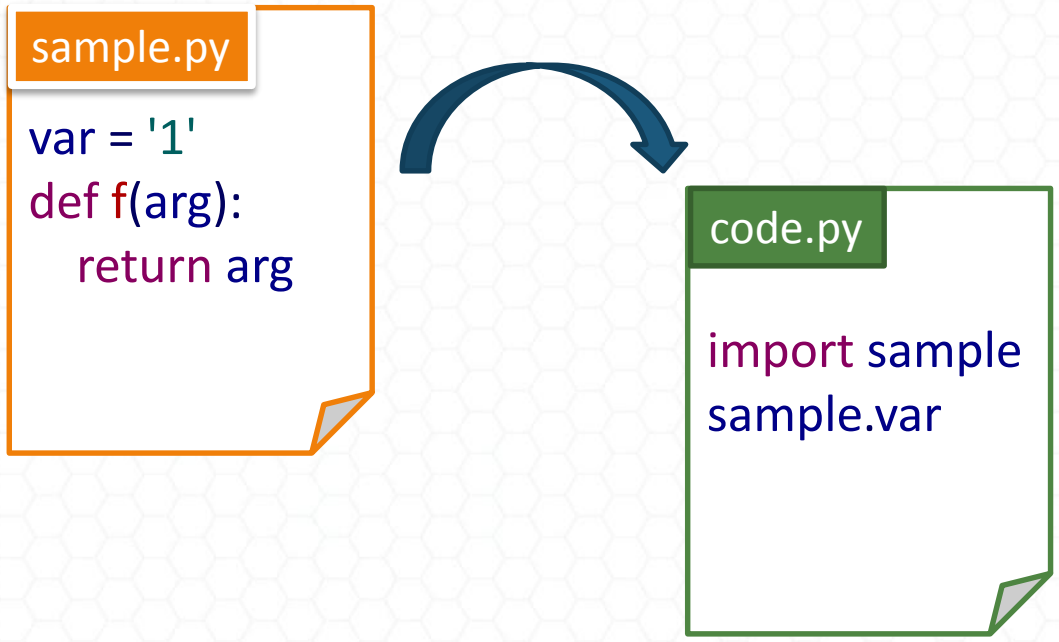
使用內建模組

```
# 引用模組
import math
math.ceil(2.4)

# 探索模組內容物
print(dir(math))
help(math.ceil)
```

使用者定義模組
(User-Defined Module)

如果想使用自己定義好的一堆函式?

- 使用者可以自己開發模組，只要將模組定義為<module>.py，之後便可以透過import <module>來使用模組



使用者自定義模組

```
import test
print(test.b)
print(test.hello('David'))

dir(test)
```

test.py

```
var = 'I am an variable'

def hello(name):
        return "hello," + name
```

模組名稱空間
(Module Namespace)

怎麼避免函式名稱衝突

- 假設貓跟狗都有.eat 這個函式，Python 可以使用命名空間來避免衝突

cat.eat()

dog.eat()

https://goo.gl/1nMU0t

模組引入的方式

```
# 最安全的存取方式
import test
print(test.hello('David'))

# 有跟hello 名字衝突的風險
from test import hello
print(hello('David'))
```

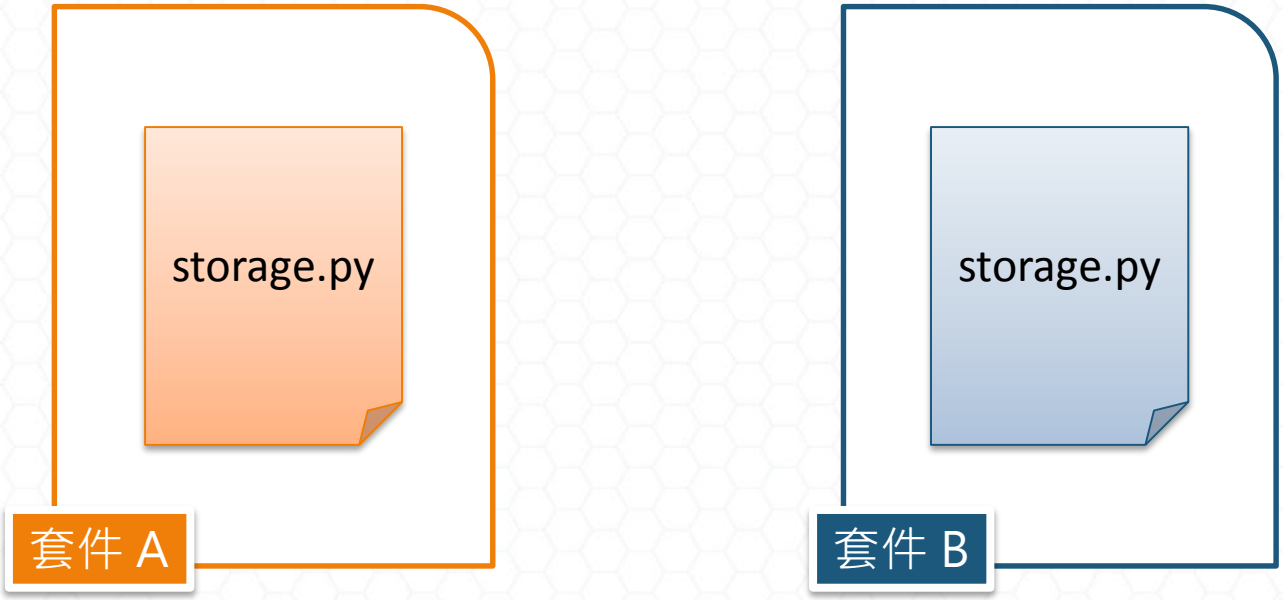
模組引入的方式(2)

```
# 有跟h 名字衝突的風險
from test import hello as h
print(h('David'))

# 最不推薦的方法
from test import *
print(b)
```

套件 (Package)

如何避免模組名字衝突

- 假設兩個不同使用者都同樣想要取storage 當模組名稱?



套件

- 套件就是目錄而已
- 必須在目錄下加上__init__.py (空檔案) 來啟用套件

```
child/          # 套件名稱
   __init__.py  # 使用__init__.py 指名為套件
   test.py      # 套件內的模組
```

引用套件

```
# 呼叫方法一
from child import test
print(test.hello('david'))

# 呼叫方法二
import child
print(child.test.hello('david'))

# 呼叫方法三
from child.test import hello
print(hello('david'))
```

引用第三方套件

Python 讓你變救世主

http://goo.gl/jbuZk4

安裝python 套件

- 下載原始碼，解壓縮後，找到setup.py處執行：
  - python setup.py install
- 套件管理工具
  - easy_install
  - pip (主流套件管理工具，可以反安裝套件)

安裝pip: Python 程式語言的套件管理程式

- **Ubuntu 使用者**

sudo apt-get install curl

sudo apt-get install python-pip

- **Linux(non-ubuntu) & Mac 使用者**

curl http://python-distribute.org/distribute_setup.py | sudo python

curl https://raw.github.com/pypa/pip/master/contrib/get-pip.py | sudo python

- **Windows 使用者**

下載 https://raw.github.com/pypa/pip/master/contrib/get-pip.py

執行 python get-pip.py

Pip 簡易操作

- 套件安裝

pip install requests

pip install install http://example.com/virtualenv-1.6.4.zip

pip install git+https://github.com/simplejson/simplejson.git

- 套件升級

pip install -U requests

- 套件移除

pip uninstall requests

Pip 進階指令

- 搜尋套件 [關鍵字: requests]

pip search requests

- 列出pip 說明

pip help

- 列出所有已安裝套件

pip freeze

建立套件安裝需求檔requirements.txt

- pip freeze > requirements.txt
- pip install -r requirements.txt

THANK YOU