

A photograph showing a person's hands typing on a laptop keyboard. The laptop screen displays a colorful, abstract visualization of a DNA double helix or a similar complex structure. The overall theme of the image is bioinformatics or genetic data analysis.

Python 網路爬蟲實作技術

工研院 Python AI 人工智慧資料分析師 種子師資班

張維元 @ 2018/12/22 (Sat.)

2015

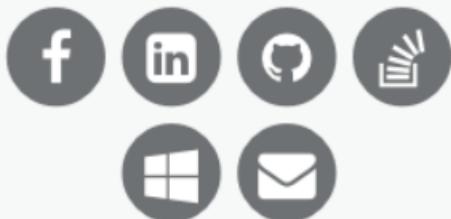


WeiYuan

(@v123582)

Web Development

Data Science



- 全端工程師 + 資料科學家
略懂一點網站前後端開發技術，學過資料探勘與機器學習的皮毛。平時熱愛參與技術社群聚會及貢獻開源程式的樂趣。

site: v123582.tw

line: weiwei63

Outline

- Python 101
- 資料來源與取得
- 認識 HTTP 網站架構與資料溝通方式（上）
- 資料爬蟲 - 靜態網頁篇
- 認識 HTTP 網站架構與資料溝通方式（下）
- 資料爬蟲 - 動態網頁篇
- 實務上的爬蟲應用
- 有了資料之後，然後呢？
- 資料爬蟲的驗收與測驗



Python 101

- +01 Quickstart for Python
 - Introduction
-
- *There is only one way to do it. - Python*
 - Python 的設計哲學是優雅、明確、簡單。
 - Python 2.x 還是 Python 3.x ?
 - Runtime : shell、command、jupyter/ipython、anaconda
 - IDE : PyCharm、Spider(built in anaconda)、Repl.it
 - Python 是一個高階編程語言 (High-level programming language)，直譯式 (Interpreted) 的運作模式提供了快速的開發。核心哲學是「簡單優雅」，盡量用較少的、較簡單易懂的代碼實現需要的功能。提供了非常完善的基礎函式庫，包含網絡，文件，GUI，數據庫，文本等大量內容。此外，Python 可以跟系統有很好的相容性，因此被稱為是膠水語言。Python 有以下幾個缺點：「運行速度慢」、「代碼不能加密」，「無法向下兼容」

- 因為 Python 不能向下相容的特性，導致了 Py2 與 Py3 兩個版本的選擇。在早期會有人建議先不用急著學 Py3，原因是大部分好用的函式庫都還是以 Py2 的版本存在，太新的環境可能會造成無法享受到函式庫豐富的優點。不過以現在來說，大部分有在維護的函式庫也都有提出相應的使用方法。所以以現在來說，我認為以 Py3 為主要版本來學習是比較恰當的。

執行 Python 程式主要有幾種做法：

- **shell** (互動介面) : 在 terminal (終端機、命令提示視窗) 中輸入 python , 可以進入到 python 的互動介面。操作方式是輸入一行程式碼，就會回傳一段結果，因為是一行一行的執行，我們把它稱為是「直譯式 (Interpreted) 」的程式語言。
- **command** (指令) : 把 python 程式碼存成一個 `py` 的檔案，然後在 terminal 輸入 `python <檔名>.py` 然後回傳結果。
- **jupyter/ipython notebook** : jupyter (原本叫做 ipython 後來獨立出來) 是一款把原生的 python 加上一些額外功能的函式庫，其中可以使用 `jupyter notebook` 進入一個網頁版的互動介面。
- **anaconda** : anaconda 把跟資料科學有關的 python 工具在包成一個更大的工具包，裡面包含 python 、常見的套件、還有 Spider IDE 。

在哪裡寫 Python 程式碼？

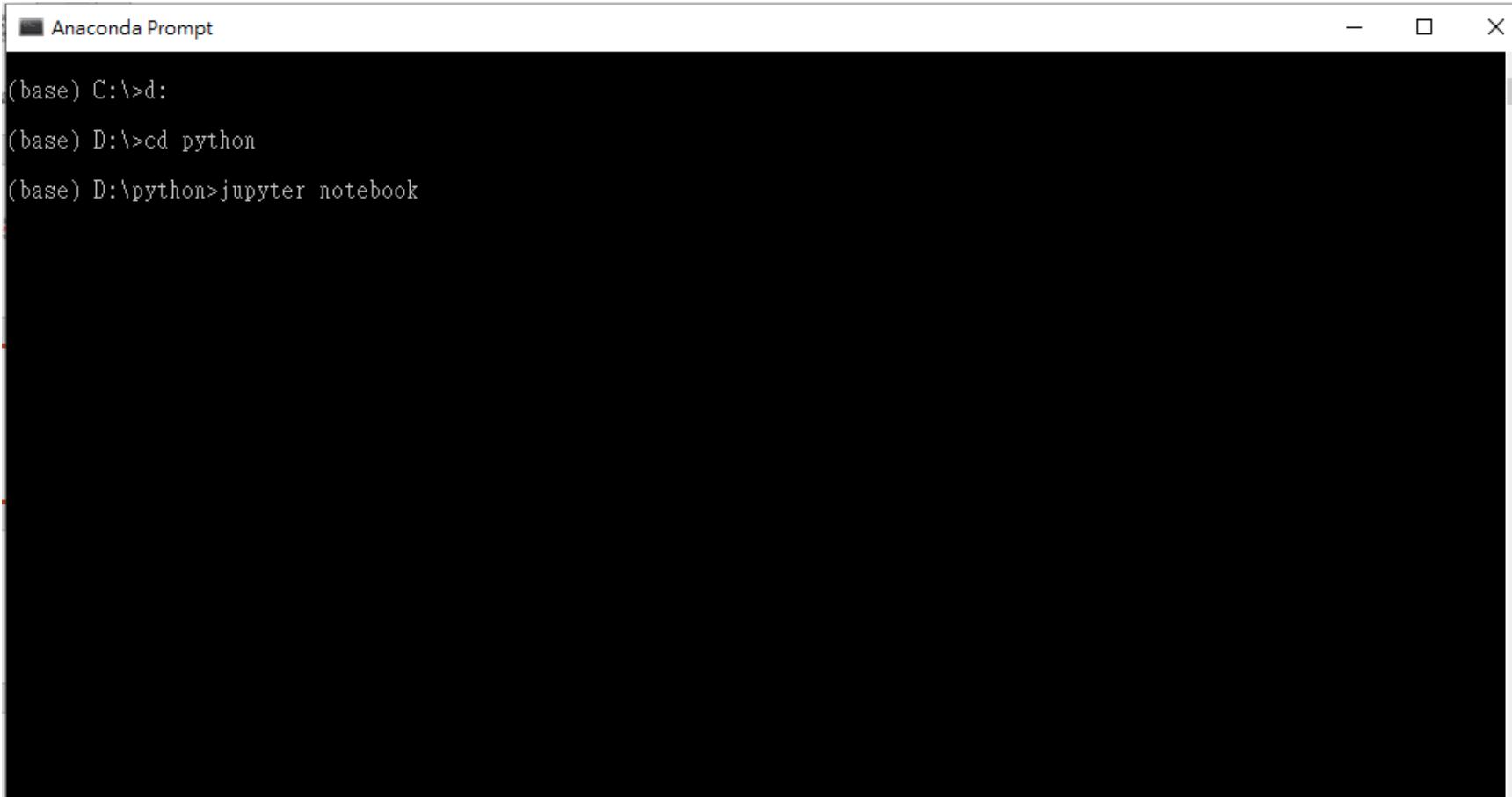
- 除了直接在 shell 或是 jupyter notebook 上，我們還是必須先把程式打好，再透過 python 來執行。那一般我們會在編輯器或是 IDE 上面寫程式。編輯器可以想成是式記事本，反正就是打完一段程式碼，然後存成 .py 副檔名就好，然後再透過 python 指令執行。另外一種選擇是 IDE (Integrated Development Environment，整合開發環境)，他同時包含了寫程式碼的編輯區與執行程式的執行區。
- 常見用來寫 python 的編輯器：sublime text、notepad、Atom
- 常見用來寫 python 的 IDE : PyCharm, Spyder, Rodeo
- 一般來說 IDE 都會有一些額外的功能，像是把你檢查你的程式碼寫得好不好，有沒有壹些語法上的錯誤，幫助開發過程中更方便。不過常見的編輯器，有都會有相對的套件可以使用。有些人會覺得，IDE 包了太多東西，開程式也需要浪費電腦的資源，所以選擇了編輯器進行開發。

相關的開發管理工具

PIP

- pip 是 python 的套件管理工具，簡言之，就是管理套件的。在早期 Python 跟 PIP 是要分別安裝的，不過現在的版本，Python 就會包含 pip，不需要再另外安裝。跟 R 語言不一樣的地方是，Python 的函式庫是在 Python 程式外安裝的，不是在程式內。
- 本次課程採用 [Anaconda](#) 環境
 1. 下載 Anaconda 並安裝
 2. 打開 Anaconda 內的 Jupyter notebook
 3. 在 notebook 內印出 Hello world

1. 打開終端機
2. 到python目錄
3. 執行 jupyter notebook



The screenshot shows a terminal window titled "Anaconda Prompt". The window has standard minimize, maximize, and close buttons at the top right. The terminal's title bar also displays the name "Anaconda Prompt". Inside the terminal, the following command sequence is visible:

```
(base) C:\>d:  
(base) D:\>cd python  
(base) D:\python>jupyter notebook
```

The terminal window has a dark background and light-colored text. A vertical scroll bar is visible on the right side of the terminal area.

開啟成功

http://localhost:8888/tree

Home

Google Jupyter home EasyFlow GP電子簽核系統 TIPTOP GP ERP VIVOTEK ND9441 監視器(首) 監視器(首)10.0.0.6 ASUSTeK COMPUTER INC Advantech WebAccess ...

jupyter

Files Running Clusters

Select items to perform actions on them.

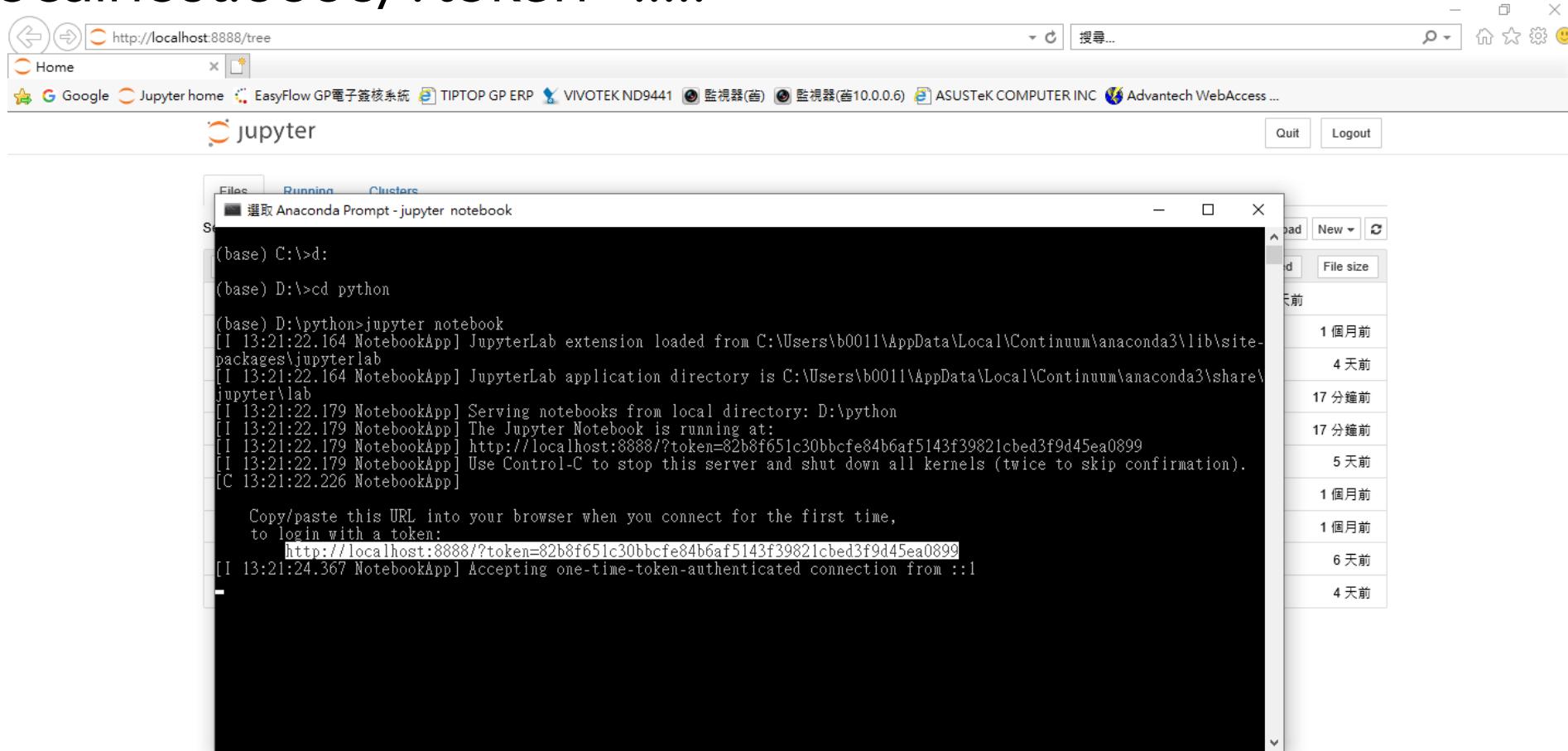
	Name	Last Modified	File size
<input type="checkbox"/> 0	/		
<input type="checkbox"/>	data	4 個月前	
<input type="checkbox"/>	1219.ipynb	1 個月前	
<input type="checkbox"/>	20181222-webCrawler_01.ipynb	4 天前	
<input type="checkbox"/>	20190112-DataAnalysis_01.ipynb	15 分鐘前	
<input type="checkbox"/>	20190113-DataAnalysis_02.ipynb	15 分鐘前	
<input type="checkbox"/>	DAY4_webCrawler_test.ipynb	5 天前	
<input type="checkbox"/>	Untitled.ipynb	1 個月前	
<input type="checkbox"/>	Untitled1.ipynb	1 個月前	
<input type="checkbox"/>	Untitled2.ipynb	6 天前	
<input type="checkbox"/>	week2-day1.ipynb	4 天前	

Upload New

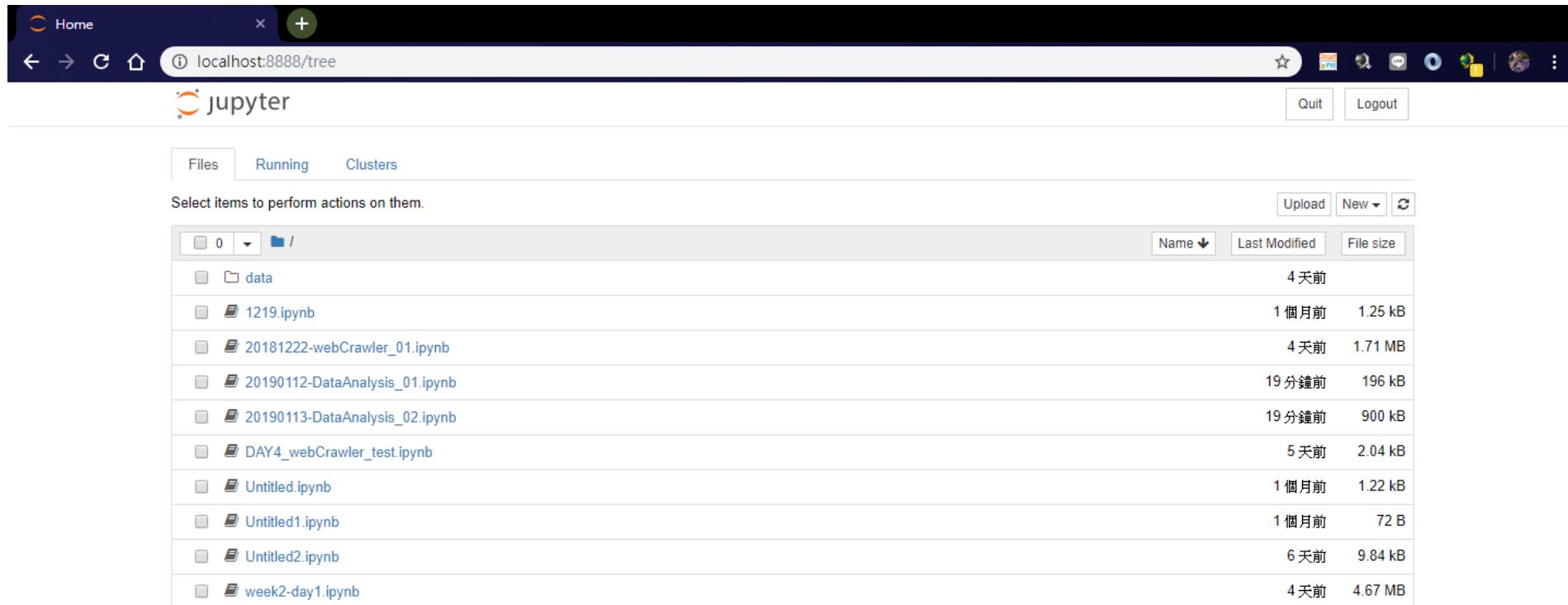


如要切換瀏覽器，複製畫面中的網址

http://localhost:8888/?token=.....



貼到想要使用的瀏覽器的網址列，按下執行



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import this
5
6
```

- 編碼宣告 (Encoding Declaration)
 - 第1行註釋是為了告訴 Linux-based 系統，這是一個 Python 的可執行程式，Windows 系統會忽略這個註釋
 - 第2行註釋是為了告訴 Python 解釋器，按照 UTF-8 編碼讀取程式碼。
 - 第4行是python的彩蛋

```
7 import keyword
8 keyword.kwlist
```

- 第7~10行：載入模組與使用模組內的函式，可使用的語法

```
9 from keyword import kwlist
10 kwlist
```

Common Types & Operator

- Numeric type
 - int, float, bool
- String type
- Container type
 - list, tuple, dict, set

```
1 a, b, c, d = 20, 5.5, True, 4+3j
2
3 print(type(a)) # <class 'int'> 整數
4 print(type(b)) # <class 'float'> 浮點數
5 print(type(c)) # <class 'bool'> 布林
6 print(type(d)) # <class 'complex'> 複雜
7
8
```

arithmetic

- Numeric

bitwise

- Container

comparison

logical

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>><<	Right and left bitwise shift
&	Bitwise 'AND'
^	Bitwise exclusive 'OR' and regular 'OR'
<= <> >=	Comparison operators
< > == !=	Equality operators
= %= /= /= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

- expression = operator + operand (variables)
 - arithmetic, comparison

字串的宣告與使用

```
In [20]: b = 'hello world'  
        b[0], b[1], b[2] #取第N個字元
```

```
Out[20]: ('h', 'e', 'l')
```

```
In [22]: b[0:5] #取第0~5個字元
```

```
Out[22]: 'hello'
```

```
In [23]: b[0:len(b)] #取 b 第0~總字元數
```

```
Out[23]: 'hello world'
```

```
In [24]: b[:] #取 b 全部字元
```

```
Out[24]: 'hello world'
```

```
In [32]: b[::-1] #將字元倒轉
```

```
Out[32]: 'dlrow olleh'
```

替換字元

原本 : The lyrics is not that poor!

新 : The lyrics is good!

替換字元練習1

```
In [40]: s = 'The lyrics is not that poor!'
print(s.replace('not that poor', 'good'))
```

The lyrics is good!

替換字元練習2

```
In [11]: s = 'The lyrics is not that poor!'
print(s[0:s.find('not')] + 'good' + s[s.find('poor')+4:])
```

The lyrics is good!

```
In [34]: print(s[0:s.find('not')]) # 取第0字元到'not'為止
```

The lyrics is

```
In [39]: print(s[s.find('poor')+4:]) #找 'poor' 的第4位元以後
```

!

替換字元練習3：%s 格式化字串 · 取%()內的字元

```
In [2]: s = 'The lyrics is not that poor!'
print('%s%s%s' % (s.split('not')[0], 'good', s.split('poor')[1]))
```

The lyrics is good!

```
In [42]: print(s.split('not'))
```

['The lyrics is ', ' that poor!']

```
In [43]: print(s.split('not')[0])
```

The lyrics is

```
In [45]: print(s.split('poor'))
```

['The lyrics is not that ', '!']

```
In [3]: print(s.split('poor')[1])
```

!

拆解網址：<https://www.trivago.com.tw:3000/search#taipei>

一個網址的結構大概如圖所示，試著用 **Python** 分別把每個值取出來。

```
In [8]: url = 'https://www.trivago.com.tw:3000/search#taipei'

protocol = url.split(':')[0]
domain = url.split(':')[1].split(':')[0]
port = url.split(':')[2].split('/')[0]
path = url.split('#')[0].replace('protocol', '').replace(protocol, '').replace(domain, '').replace(port, '').replace('/', '').replace('/', '')
fragment = url.split('#')[1]

print('protocol : %s' % (protocol)) # https
print('domain : %s' % (domain)) # www.trivago.com.tw
print('port : %s' % (port)) # 3000
print('path : %s' % (path)) # search
print('fragment : %s' % (fragment)) # taipei

protocol : https
domain : www.trivago.com.tw
port : 3000
path : search
fragment : taipei
```

```
In [50]: url.split(':') # split 切割字元
```

```
Out[50]: ['https', '//www.trivago.com.tw', '3000/search#taipei']
```

```
In [48]: url.split(':')[1] # split 切割字元
```

```
Out[48]: ['https', 'www.trivago.com.tw:3000/search#taipei']
```

```
In [56]: url.split('#')[0].replace(':', '').replace('/', '') # replace取代字元
```

```
Out[56]: 'httpswww.trivago.com.tw3000search'
```

- 容器型態 (Container type/Compound types)

- 列表 (list) [] => 有序且可變的
- 字組 (tuple) () => 有序但不可變的 (string 是一種特例)
- 字典 (dict) {} => Key - Value 對應的組合
- 集合 (Set) {} => 無序且包含不可以重複元素的組合

列表 list

```
In [18]: b = [1, 2, 3, 4, 5, 6]
print(b[0])
print(b[2])
b[0] = 999
print(b[0:5])
print()
print(b[0:len(b)])
print(b[:])
print()
print(b[1:5])
print(b[::-1])
```

```
1
3
[999, 2, 3, 4, 5]
```

```
[999, 2, 3, 4, 5, 6]
[999, 2, 3, 4, 5, 6]
```

```
[2, 3, 4, 5]
[6, 5, 4, 3, 2, 999]
```

list 可以相加、可以乘數、不可減與除

```
In [20]: L = [1, 2, 3, 4]  
G = [3, 4, 5, 6]
```

```
In [28]: print(L+G)  
[1, 2, 3, 4, 3, 4, 5, 6]
```

```
In [25]: print(L*2 + G*2)  
[1, 2, 3, 4, 1, 2, 3, 4, 3, 4, 5, 6, 3, 4, 5, 6]
```

```
In [26]: print(L-G)
```

```
-----  
TypeError                                     Traceback (most recent call last)  
<ipython-input-26-bf6df738304c> in <module>()  
----> 1 print(L-G)  
  
TypeError: unsupported operand type(s) for -: 'list' and 'list'
```

```
In [27]: print(L/2)
```

```
-----  
TypeError                                     Traceback (most recent call last)  
<ipython-input-27-59f175ab2203> in <module>()  
----> 1 print(L/2)  
  
TypeError: unsupported operand type(s) for /: 'list' and 'int'
```

list 相關函式

```
In [40]: my_list = [1]
my_list.append(2)
print(my_list)
```

```
[1, 2]
```

```
In [41]: my_list = [1]
my_list.append(3)      # append() 在串列後端加入 <item> 項目
print(my_list)
```

```
[1, 3]
```

```
In [43]: my_list.extend([4])
print(my_list)
```

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

```
In [44]: my_list.extend([5])  # extend() 從最新的list不斷增加
print(my_list)
```

```
[1, 3, 4, 5]
```

```
In [57]: my_list.insert(0, 6)  # 在 <position> 位置插入 <item> 項目
print(my_list)
```

```
[6, 1, 3, 4, 5]
```

```
In [58]: my_list.pop()      # 刪除並回覆串列的最後項目
print(my_list)
```

```
[6, 1, 3, 4]
```

```
In [60]: my_list.remove(1)  # 刪除第一次出現的 <item> 項目
print(my_list)
```

```
[6, 3, 4]
```

```
In [61]: my_list.reverse() # 反向排序  
print(my_list)  
[4, 3, 6]
```

```
In [62]: my_list.reverse() # 反向排序  
print(my_list)  
[6, 3, 4]
```

```
In [71]: my_list.sort() # 由小到大排序  
print(my_list)  
[3, 4, 6]
```

```
In [77]: my_list.count(3) # 回覆 <item> 項目出現的次數  
Out[77]: 1
```

```
In [78]: len(my_list)  
Out[78]: 3
```

```
In [76]: sum(my_list)  
Out[76]: 13
```

- 動動手練習

第一題：

[1, 2, 3, 4, 5, 6, 7, 8, 9, 0] 變成 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

第二題：

[1, 2, 3, 4, 5, 6, 7, 8, 9, 0] 變成 [2, 3, 4, 5, 6, 7, 8, 9, 0, 1]

字典 dictionary

每個字典項目為鍵與值的配對 (以冒號分隔)

```
In [80]: my_dict = {'Mark': 1, 'test': 0}
my_dict['Mary'] = 2          # 利用方括號語法來修改值
print(my_dict)

{'Mark': 1, 'test': 0, 'Mary': 2}
```

```
In [81]: my_dict['Tom']           # 只有鍵，沒有值就會出現錯誤
```

```
-----
KeyError                                 Traceback (most recent call last)
<ipython-input-81-4edd0c92b934> in <module>()
----> 1 my_dict['Tom']

KeyError: 'Tom'
```

```
In [82]: del my_dict['test']      # 刪除鍵，同時刪除值
print(my_dict)

{'Mark': 1, 'Mary': 2}
```

```
In [84]: my_dict.items()         # 列出項目
```

```
Out[84]: dict_items([('Mark', 1), ('Mary', 2)])
```

```
In [85]: my_dict.keys()         # 列出鍵
```

```
Out[85]: dict_keys(['Mark', 'Mary'])
```

```
In [86]: my_dict.values()       # 列出值
```

```
Out[86]: dict_values([1, 2])
```

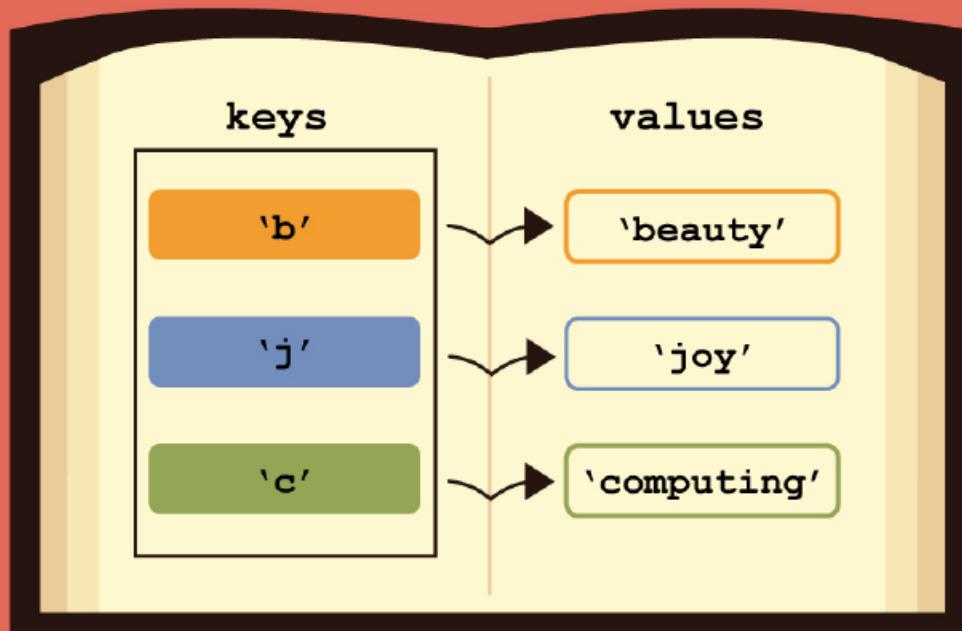
整合不同字典

```
In [91]: dic1={1:10, 2:20}  
dic2={3:30, 4:40}  
dic3={5:50, 6:60}  
  
dic = dict(list(dic1.items()) + list(dic2.items()) + list(dic3.items()))  
print(dic)
```

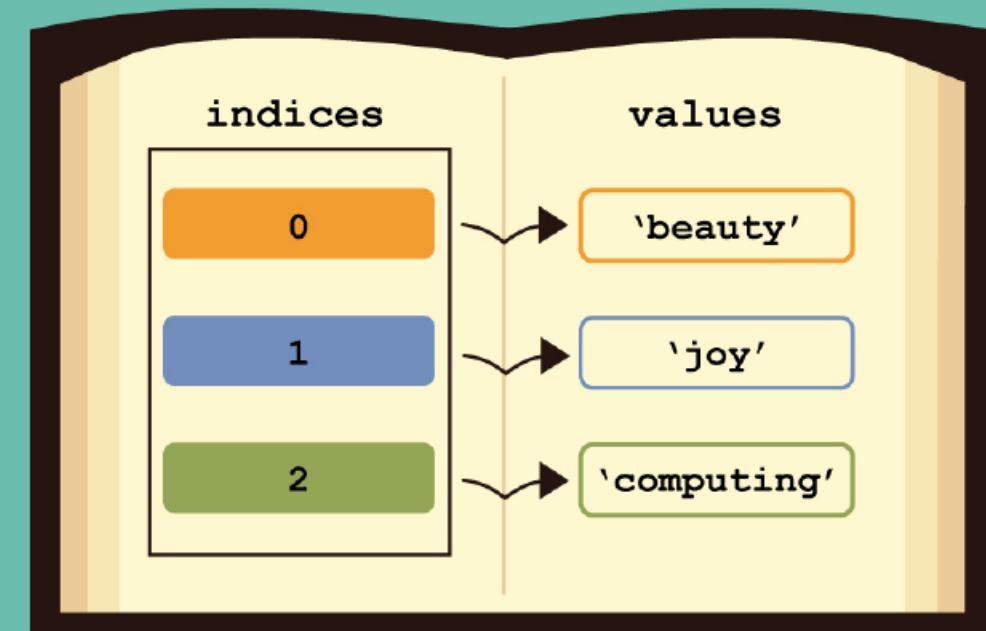
```
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```



dictionaries



lists



What it truth?

a	b	a and b	a or b
False	False	False	False
False	True	False	True
True	False	False	True
True	True	True	True

CONSTRUCTED TABLE.

What is truth?

False

None

0

0.0

''

b ''

[]

{}

()

True

Non-null references and not otherwise false

Non-zero numbers

Non-empty strings

Non-empty containers

Other empty containers with
non-literal empty forms are also
considered false.

Flow Control

- if - elif - else
- while
- for in

```
1 if condition:  
2     ...  
3 elif condition:  
4     ...  
5 else:  
6     ...  
7  
8 a = ... if condition else ...  
9
```

Flow Control

- if - elif - else
- while
- for in

```
1 while condition:  
2     ....
```

Flow Control

- if - elif - else
- while
- for in

```
1 for i in [...]:  
2     ...  
3  
4 a = [i for i in [...]] # list  
5 b = (i for i in [...]) # generator
```

[1, 2, 3, 4, 5, 6, 7, 8, 9, 0] 變成 [2, 4, 6, 8, 0, 1, 3, 5, 7, 9]

```
In [11]: xa = [1,2,3,4,5,6,7,8,9,0]
x1=[]
x2=[]

for i in xa:
    if i % 2 == 0 :      # 如果除 2 後餘數為 0，就是 True
        x1.append(i)
    else:
        x2.append(i)

print(x1 + x2)
```

[2, 4, 6, 8, 0, 1, 3, 5, 7, 9]

有沒有 == 0 的判斷式，結果有差別！

```
In [10]: xa = [1,2,3,4,5,6,7,8,9,0]
x1=[]
x2=[]

for i in xa:
    if i % 2:          # 如果除 2 後餘數為 0，就是 False
        x1.append(i)
    else:
        x2.append(i)

print(x1 + x2)
```

[1, 3, 5, 7, 9, 2, 4, 6, 8, 0]

使用 if not 的比較式，也可以達到原本結果

```
In [12]: xa = [1,2,3,4,5,6,7,8,9,0]
x1=[]
x2=[]

for i in xa:
    if not i % 2 : # 如果不是被 2 整除，就是 True
        x1.append(i)
    else:
        x2.append(i)

print(x1 + x2)
```

[2, 4, 6, 8, 0, 1, 3, 5, 7, 9]

在數學中，用 $n!$ 來表示階乘，例如， $4! = 1 \times 2 \times 3 \times 4 = 24$ 。請計算 $3!, 4!, 5!$ 。

for 迴圈

```
In [15]: nums = int(input('請輸入數字: '))
d = []

for i in range(nums):
    d.append(i+1)
print(d)

answer = 1

for num in d:
    answer = answer * num
print(d[-1], '!-', answer)

請輸入數字: 4
[1, 2, 3, 4]
4 != 24
```

for 迴圈，另一種簡單寫法

```
In [16]: nums = int(input('請輸入數字: '))
sum = 1
total = 0
for i in range(1, nums+1):
    sum *= i
    total += sum
print(nums, '!-', answer)

請輸入數字: 4
4 != 24
```

while 迴圈

```
In [17]: nums = int(input('請輸入數字: '))

def fact(n):
    r = 1
    i = 1
    while i <= n:
        r *= i
        i += 1
    return r

total = fact(nums)
print(nums, '!-', total)

請輸入數字: 4
4 != 24
```

計算字符出現次數

```
In [24]: a ='Please count the character here.'
d={}

for i in a:
    if i in d:
        d[i] = d[i]+1
    else:
        d[i]=1
print(d)

{'P': 1, 'l': 1, 'e': 6, 'a': 3, 's': 1, ' ': 4, 'c': 3, 'o': 1, 'u': 1, 'n': 1, 't': 3, 'h': 3, 'r': 3, '.': 1}
```

找到字符所在的位置

```
In [3]: s ='Here are UPPERCASE and lowercase chars.'
d = {} # 空集合：無序且包含不可以重複元素的組合

for count, c in enumerate(s):
    d.setdefault(c, [])
    d[c].append(count+1)

print(d)

{'H': [1], 'e': [2, 4, 8, 27, 32], 'r': [3, 7, 28, 37], ' ': [5, 9, 19, 23, 33], 'a': [6, 20, 30, 36], 'U': [10], 'P': [11, 12], 'E': [13, 18], 'R': [14], 'C': [15], 'A': [16], 'S': [17], 'n': [21], 'd': [22], 'l': [24], 'o': [25], 'w': [26], 'c': [29, 34], 's': [31, 38], 'h': [35], '.': [39]}
```

補充：輸入字符，回傳位置

```
In [5]: str_1='Here are UPPERCASE and lowercase chars.'
char_1=str(input('請輸入尋找的字母:'))
count=1
str_list=list(str_1)
for each_char in str_list:
    count+=1
    if each_char==char_1:
        print(each_char,count-1)
```

```
請輸入尋找的字母:e
e 2
e 4
e 8
e 27
e 32
```

range()

```
In [6]: for i in range(1, 3):
    print(i)

1
2
```

zip()

```
In [24]: for i, j in zip(['a', 'b', 'c'], [1, 2, 3]): # zip 接受一系列可迭代的對象作為參數，將對象中對應的元素打包
    print(i, j)

a 1
b 2
c 3
```

enumerate()

```
In [26]: for i,j in enumerate(['a', 'b', 'c']): # enumerate 傳回一個次序及值成組tuple的迭代
    print(i, j)

0 a
1 b
2 c
```

補充：zip 傳入參數的長度不等，則返回list的長度和參數中長度最短的對象

```
In [22]: a = [1, 2, 3]
b = [4, 5, 6]
c = [7, 8, 9, 0]
zip1 = zip(a, b) # zip 接受一系列可迭代的對象作為參數，將對象中對應的元素打包成一個個 tuple (元組)，然後返回由這些 tuples 組成的 list (列表)
print(list(zip1))

zip2 = zip(a, c)
print(list(zip2))

[(1, 4), (2, 5), (3, 6)]
[(1, 7), (2, 8), (3, 9)]
```



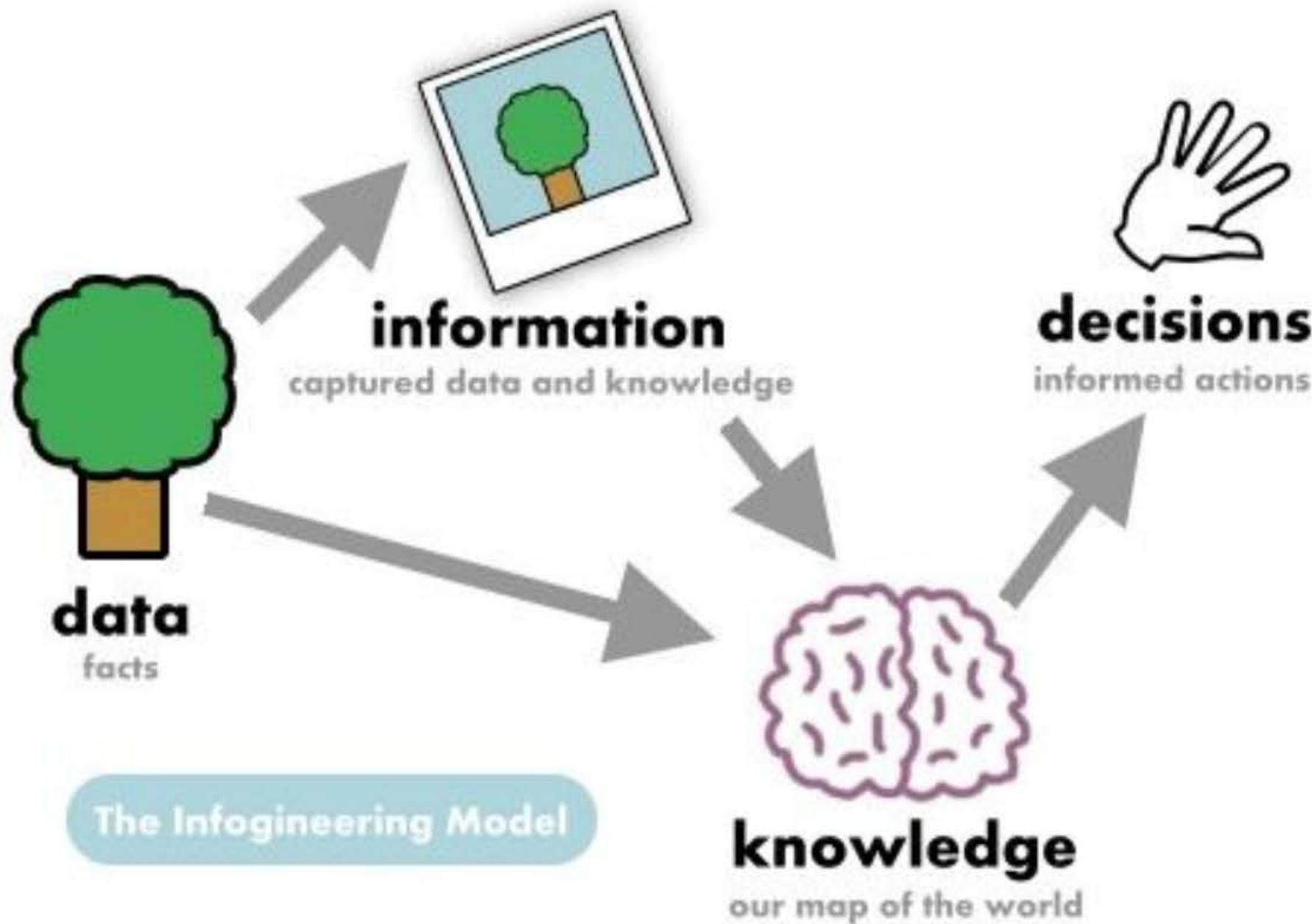
資料來源與取得

Outline

- 知識管理 101 - 從資料到儲存
- 常見的公開資料來源
- 如何使用 Python 存取資料
- API 資料來源與 Request 串接存取

知識管理 101 - 從資料到儲存

- 根據維基百科中，對於資料的定義如下：「資料是指未經過處理的原始記錄。一般而言，資料缺乏組織及分類，無法明確的表達事物代表的意義，它可能是一堆的雜誌、一大疊的報紙、數種的開會記錄或是整本病人的病歷紀錄。」



資料來源與取得

1. 檔案

資料會包成檔案提供下載，格式可能包含常用的標準格式，例如「CSV」、「JSON」等等通用的格式。如果是已經有提供制式的格式的話，相對容易處理，一般的程式語言或是商業軟體都具備讀取的功能。不過還有一些很常見的資料格式，像是「XLS」、「PDF」，處理上就不是這麼容易，需要更多的工具協助才可以。

資料來源與取得

2. 開放接口 (API)

API (Application Program Interface , 應用程序接口) 提供程式化的連接的接口，讓工程師/分析師可以選擇資料中要讀取的特定部分，而不需要把整批資料事先完整下載回來。API 一般都是直接連接到一個資料庫，而資料庫內儲存的都是即時更新最新版本的資料。

資料來源與取得

3. 網頁爬蟲

最後一種也是很常出現資料的地方，就是網頁上。我們常常會發現我們的資料並不是一個特定的檔案，也沒有 API 可以使用。僅出現在網頁上。這樣的話，就只能自己寫爬蟲，把自己想用的資料從網頁上爬下來。

資料該怎麼存？資料格式解密

CSV (Comma Separated Values) 逗號分隔值，是一種常見的資料格式，使用逗號將不同欄位做為分隔。可以使用一般的文字編輯器以原始格式開啟，也可以使用 excel 或 number 等試算表軟體以表格方式開啟。一般格式如下，第一列會記錄格式，第二列開始記錄資料。

```
1
2 firstName, lastName
3 John, Doe
4 Anna, Smith
5 Peter, Jones
6
```

資料該怎麼存？資料格式解密

CSV (Comma Separated Values) 逗號分隔值，是一種常見的資料格式，使用逗號將不同欄位做為分隔。可以使用一般的文字編輯器以原始格式開啟，也可以使用 excel 或 number 等試算表軟體以表格方式開啟。一般格式如下，第一列會記錄格式，第二列開始記錄資料。

- 優點：結構單純、人機皆可讀、檔案小
- 缺點：結構鬆散（未限定編碼、不一定有欄位名稱）、格式誤判、換行問題

資料該怎麼存？資料格式解密

- JSON (JSON stands for JavaScript Object Notation)

JavaScript 物件格式，是一種延伸自 JavaScript 物件來儲存和交換簡單結構的輕量級純文字資料交換格式。一般格式如下，每一筆資料都會用 “{資料屬性：資料數值}” 的格式紀錄，也可以是巢狀資料。

```
1  {"employees": [  
2      { "firstName":"John", "lastName":"Doe" },  
3      { "firstName":"Anna", "lastName":"Smith" },  
4      { "firstName":"Peter", "lastName":"Jones" }  
5  ]}  
6
```

資料該怎麼存？資料格式解密

- JSON (*JSON stands for JavaScript Object Notation*)
JavaScript 物件格式，是一種延伸自 JavaScript 物件來儲存和交換簡單結構的輕量級純文字資料交換格式。一般格式如下，每一筆資料都會用 “{資料屬性：資料數值}” 的格式紀錄，也可以是巢狀資料。
- 優點：可以存放結構較複雜的資料、大部份瀏覽器都支援
- 缺點：檔案較大（不過比XML小）、不一定適合轉換成表格型式

資料該怎麼存？資料格式解密

XML (eXtensible Markup Language) 可延伸標記式語言，是一種標記式語言，處理包含各種資訊的資料等。

```
1 <employees>
2   <employee>
3     <firstName>John</firstName> <lastName>Doe</lastName>
4   </employee>
5   <employee>
6     <firstName>Anna</firstName> <lastName>Smith</lastName>
7   </employee>
8   <employee>
9     <firstName>Peter</firstName> <lastName>Jones</lastName>
10  >
11    </employee>
12  </employees>
```

資料該怎麼存？資料格式解密

XML (eXtensible Markup Language) 可延伸標記式語言，是一種標記式語言，處理包含各種資訊的資料等。

- 優點：可以存放結構較複雜的資料、大多瀏覽器可幫忙排版成較易讀格式
- 缺點：檔案較大（比 JSON 更大）、不一定適合轉換成表格型式



「思考目標」→「找資料」→「整理資料」
→「儲存資料」→「應用資料」

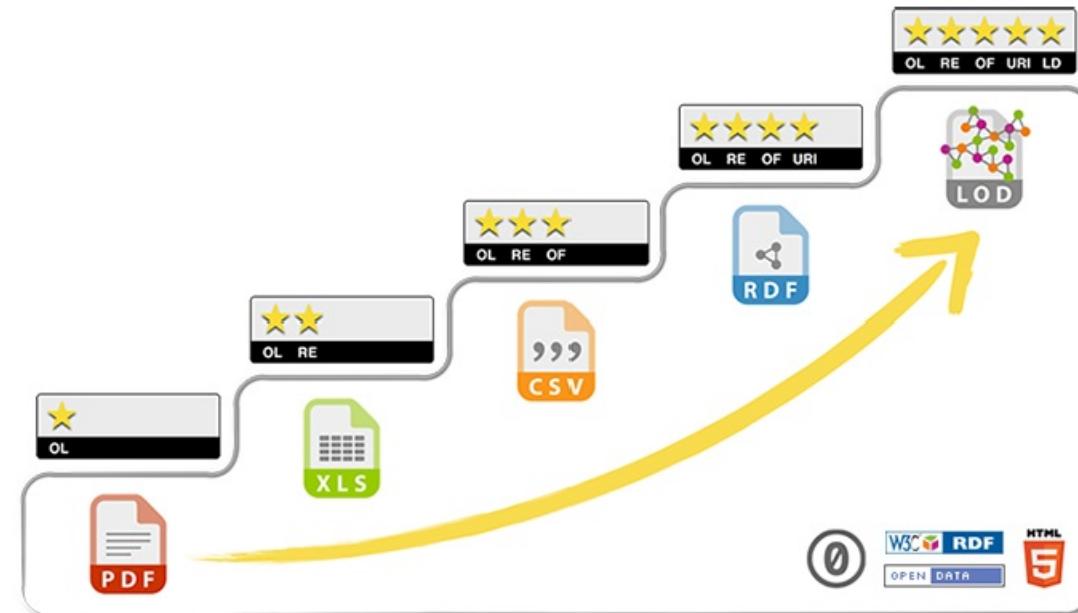
#討論：用檔案方式來保管資料是一個好的方法嗎？會不會遇到什麼問題？

什麼是開放資料？

- 開放資料 (英語：Open data) 指的是一種經過挑選與許可的資料。這種資料不受著作權、專利權，以及其他管理機制所限制，可以開放給社會公眾，任何人都可以自由出版使用，不論是要拿來出版或是做其他的運用都不加以限制。

可得性與可讀性

重新使用與散播



什麼是開放資料？

- 開放資料 (英語：Open data) 指的是一種經過挑選與許可的資料。這種資料不受著作權、專利權，以及其他管理機制所限制，可以開放給社會公眾，任何人都可以自由出版使用，不論是要拿來出版或是做其他的運用都不加以限制。

可得性與可讀性

重新使用與散播



常見資料來源

- 台灣政府資料開放平台：<https://data.gov.tw/>
- 台北市開放資料平台：<https://data.taipei/index>
- 新北市開放資料平台：<http://data.ntpc.gov.tw/>
- 台中市開放資料平台：<http://opendata.taichung.gov.tw/>
- 桃園市開放資料平台：<http://data.tycg.gov.tw/>
- 台南市開放資料平台：<http://data.tainan.gov.tw/>
- 高雄市開放資料平台：<https://data.kcg.gov.tw/>

常見資料來源

- 世界經濟貿易合作組織資料庫：<https://data.oecd.org/>
- 世界銀行開放資料：<https://data.worldbank.org.cn/>
- 世界衛生組織：<http://apps.who.int/gho/data/node.home>
- Google BigQuery 公開資料集：
<https://cloud.google.com/bigquery/public-data/>
- Google 開放資料搜索：<http://www.google.com/publicdata>
- 亞馬遜開放資料：<https://aws.amazon.com/cn/datasets/>

如何使用 Python 存取資料

下載檔案 => 開檔/讀檔 => 可讀與解析

到網址下載 csv 檔案_公車時刻表

```
In [28]: import urllib.request  
res = "http://opendata.hccg.gov.tw/dataset/432257df-491f-4875-8b56-dd814aee5d7b/resource/de014c8b-9b75-4152-9fc6-f0d499cefbe4/download/2015030511.csv"  
urllib.request.urlretrieve(res, './data/51.csv') # 在python 路徑，先建立 data資料夾
```

讀檔案

```
In [ ]: fh = open('./data/51.csv', newline='', encoding='utf-8')  
f = fh.read() # read() 會一次讀取所有的檔案內容  
fh.close() # close() 將檔案關閉以節省資源，必要!!
```

解析檔案內容

```
In [31]: import csv  
reader = csv.reader(f.split('\n'), delimiter=',')  
for row in reader:  
    print(row)
```

```
['\ufeff序號', '路線編號', '起訖站', '平/假日', '站名', '班次1', '班次2', '班次3', '班次4', '班次5', '班次6', '班次7', '班次8', '班次9', '班次10', '班次11', '是否單邊停靠']  
[1, '51', '香山區公所一火車站(先經中山路)', '平日', '香山區公所', '06:30', '07:50', '09:10', '10:30', '12:00', '13:20', '14:40', '16:10', '18:10', '19:40', '21:10', '否']  
[2, '51', '香山區公所一火車站(先經中山路)', '平日', '香山國小', '06:32', '07:52', '09:12', '10:32', '12:02', '13:22', '14:42', '16:12', '18:12', '19:42', '21:12', '否']  
[3, '51', '香山區公所一火車站(先經中山路)', '平日', '聖德宮', '06:33', '07:53', '09:13', '10:33', '12:03', '13:23', '14:43', '16:13', '18:13', '19:43', '21:13', '否']  
[4, '51', '香山區公所一火車站(先經中山路)', '平日', '頂埔社區', '06:34', '07:55', '09:15', '10:35', '12:05', '13:24', '14:45', '16:15', '18:15', '19:45', '21:15', '否']  
[5, '51', '香山區公所一火車站(先經中山路)', '平日', '大鵬新城', '06:36', '07:57', '09:17', '10:37', '12:07', '13:26', '14:47', '16:17', '18:17', '19:47', '21:17', '單邊停靠']  
[6, '51', '香山區公所一火車站(先經中山路)', '平日', '台貿四村', '06:38', '07:59', '09:19', '10:39', '12:09', '13:28', '14:49', '16:19', '18:19', '19:49', '21:19', '單邊停靠']
```

列出班次 1 的各站時間

In [32]:

```
import csv

bus1 = []
reader = csv.reader(f.split('\n'), delimiter=',')

for rows in reader:
    if rows:
        bus1.append(rows[5])

print(bus1)
```

['班次1', '06:30', '06:32', '06:33', '06:34', '06:36', '06:38', '06:39', '06:41', '06:42', '06:43', '06:44', '06:45', '06:46', '06:47', '06:48', '06:49', '06:50', '06:51', '06:53', '06:55', '06:57', '06:58', '07:00', '07:01', '07:03', '07:05', '07:07', '07:09', '07:10', '07:12', '07:14', '07:16', '07:18', '07:20', '07:21', '07:23', '07:24', '07:25', '07:00', '07:02', '07:03', '07:04', '07:06', '07:08', '07:09', '07:11', '07:12', '07:13', '07:14', '07:15', '07:16', '07:17', '07:18', '07:19', '07:20', '07:21', '07:23', '07:25', '07:27', '07:28', '07:30', '07:31', '07:33', '07:35', '07:37', '07:39', '07:40', '07:42', '07:44', '07:46', '07:48', '07:49', '07:51', '07:53', '07:54', '07:55']

到網址下載 zip 檔案_天氣預報

```
In [39]: import urllib.request  
import zipfile  
res = "http://opendata.cwb.gov.tw/govdownload?dataid=F-D0047-093&authorizationkey=rdec-key-123-45678-011121314"  
urllib.request.urlretrieve(res,'./data/F-D0047-093.zip') # 檔案下載儲存路徑  
f = zipfile.ZipFile('./data/F-D0047-093.zip') # 解壓縮  
f.extractall('./data/F-D0047-093') # 解壓縮後檔案路徑
```

```
urllib.request.urlretrieve(url, filename=None, reporthook=None, data=None)
```

filename：指定了保存本地路徑（如果參數未指定，urllib會生成一個臨時文檔保存數據。） reporthook：一個回調函數，當連接上服務器、以及相應的數據塊傳輸完畢時會觸發該回調，我們可以利用這個回調函數來顯示當前的下載進度。 data：指post導服務器的數據，該方法返回一個包含兩個元素的(filename, headers) 元組，filename 表示保存到本地的路徑，header表示服務器的響應頭

讀資料夾中特定檔案

```
In [43]: fh = open("./data/F-D0047-093/64_72hr_CH.xml", "r",encoding='utf-8')  
xml = fh.read()  
  
fh.close()
```

使用到未載入的模組時，會出現執行錯誤

解析檔案內容

```
In [44]: import xmltodict
d = dict(xmltodict.parse(xml))
print(d)

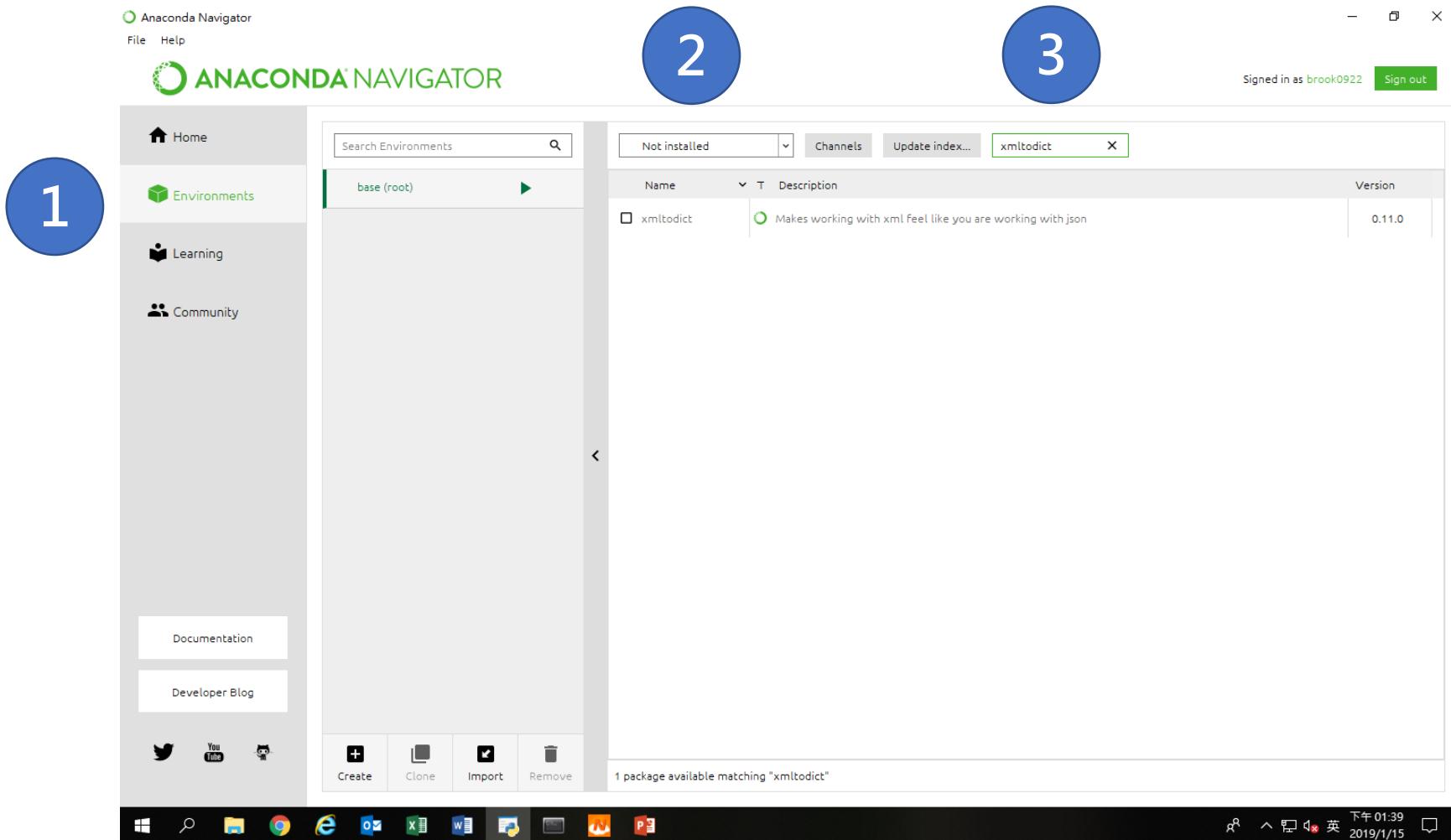
-----
ModuleNotFoundError: Traceback (most recent call last)
<ipython-input-44-768ecd8292df> in <module>()
      1 # 解析檔案內容
      2
----> 3 import xmltodict
      4 d = dict(xmltodict.parse(xml))
      5 print(d)

ModuleNotFoundError: No module named 'xmltodict'
```

方法一：終端機輸入指令

pip install xmltodict

方法二：從 ANACONDA 操作畫面載入模組



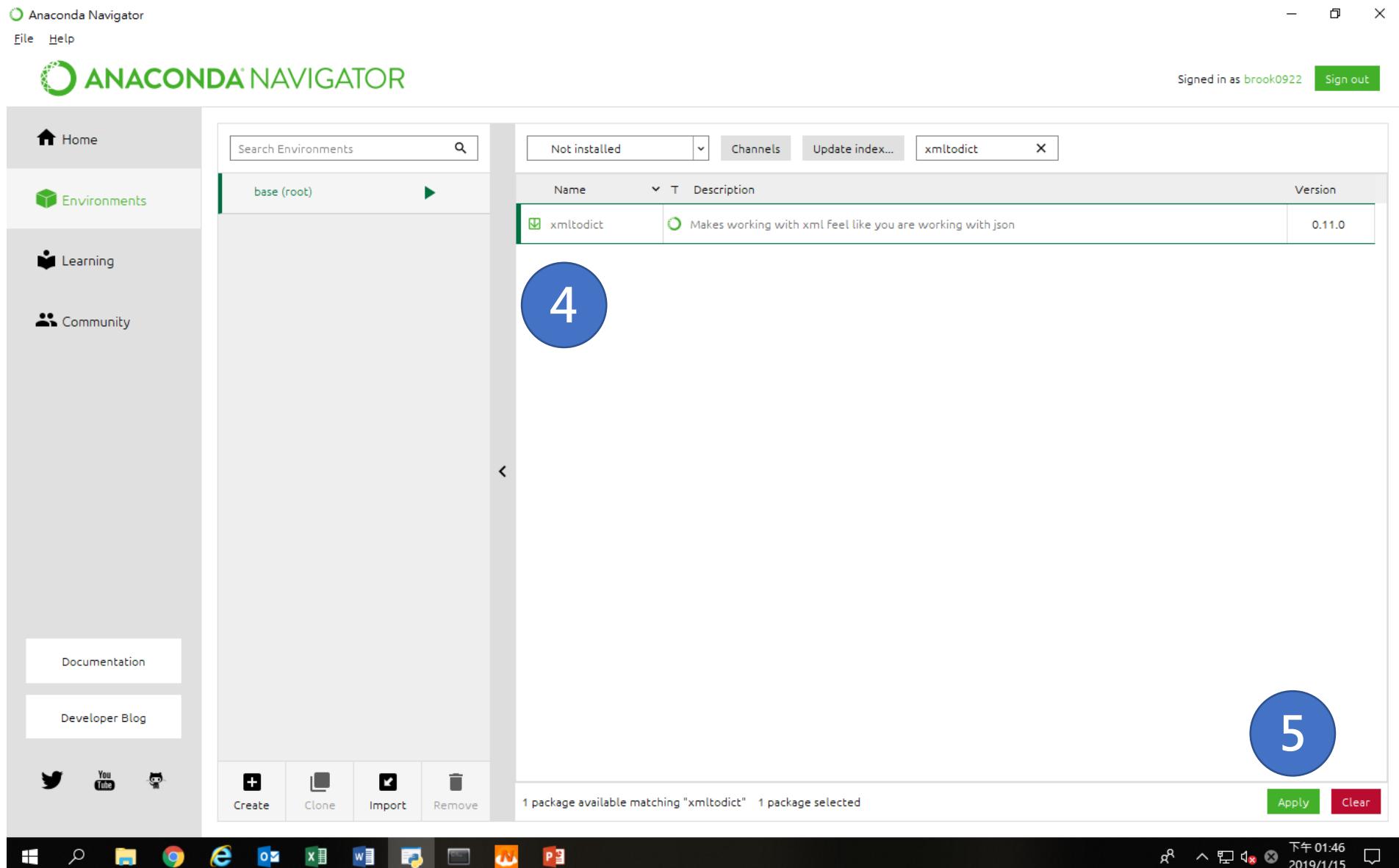
方法一：ANACONDA 終端機輸入指令

(base) C:\>d:

(base) D:\>cd python

(base) D:\python>pip install xmltodict

方法二：選擇模組，Apply



The screenshot shows the Anaconda Navigator interface. On the left, there's a sidebar with icons for Home, Environments, Learning, and Community. Below these are links for Documentation and Developer Blog, along with social media icons for Twitter, YouTube, and GitHub.

The main area has a search bar at the top labeled "Search Environments" with a magnifying glass icon. Below it, a dropdown menu shows "Not installed". A search term "xmltodict" is entered in the search bar. To the right of the search bar are buttons for "Channels", "Update index...", and "x".

The central part of the screen displays a table with columns for Name, Description, and Version. One row is highlighted, showing "xmltodict" with the description "Makes working with xml feel like you are working with json" and the version "0.11.0".

A large blue circle with the number "4" is overlaid on the search bar area. Another blue circle with the number "5" is overlaid on the "Apply" button at the bottom right of the table.

At the very bottom, a taskbar shows various application icons, and the system tray indicates the date and time as "下午 01:46 2019/1/15".

Name	Description	Version
xmltodict	Makes working with xml feel like you are working with json	0.11.0

1 package available matching "xmltodict" 1 package selected

Apply Clear

方法二：Apply

Anaconda Navigator

File Help

Signed in as brook0922 Sign out

ANACONDA NAVIGATOR

Home Environments Learning Community Documentation Developer Blog

Search Environments base (root)

Not installed Channels Update index... xmltodict

Name Description Version

Install Packages

13 packages will be installed

Name	Unlink	Link	Channel
1 xmltodict	-	0.11.0	defaults
2 *certifi	2018.8.24	2018.11.29	defaults
3 *conda	4.5.11	4.5.12	defaults
4 *cryptography	2.3.1	2.4.2	defaults
5 *curl	7.61.0	7.63.0	defaults
6 *krb5	-	1.16.1	defaults
7 *libcurl	7.61.0	7.63.0	defaults

* indicates the package is a dependency of a selected package

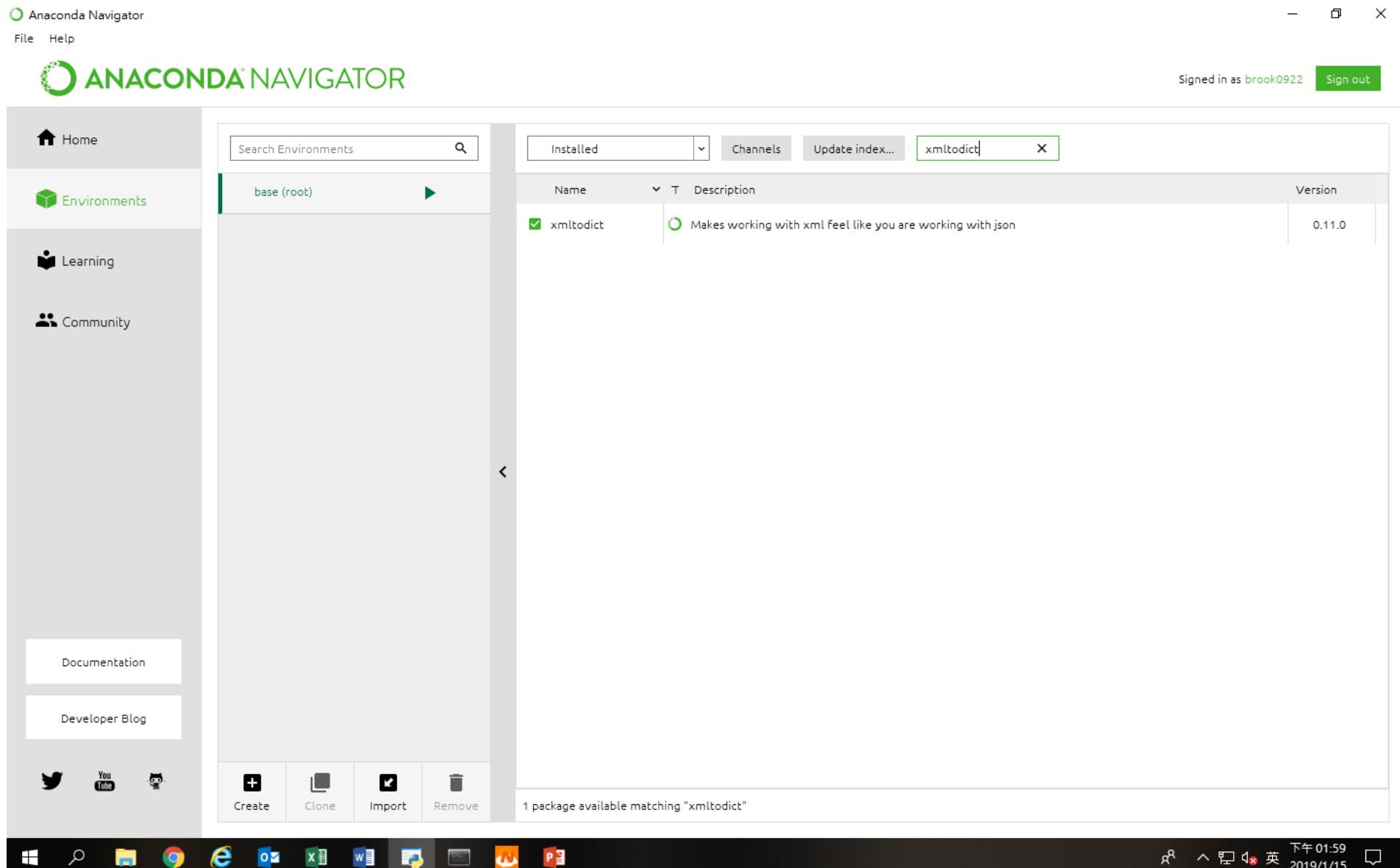
Cancel Apply

1 package available matching "xmltodict" 1 package selected

Apply Clear

6

方法二：Installed 清單中找的到就安裝成功了



開檔/讀檔 - CSV

```
1 import csv  
2  
3 with open('eggs.csv', 'rb') as csvfile:  
4     spamreader = csv.reader(csvfile)  
5     for row in spamreader:  
6         print(row)  
7  
8
```

開檔/讀檔 - JSON

```
1  
2  
3 import json  
4  
5 d = json.loads('["foo", {"bar": ["baz", null, 1.0, 2]}]')  
6 print(d)  
7  
8
```

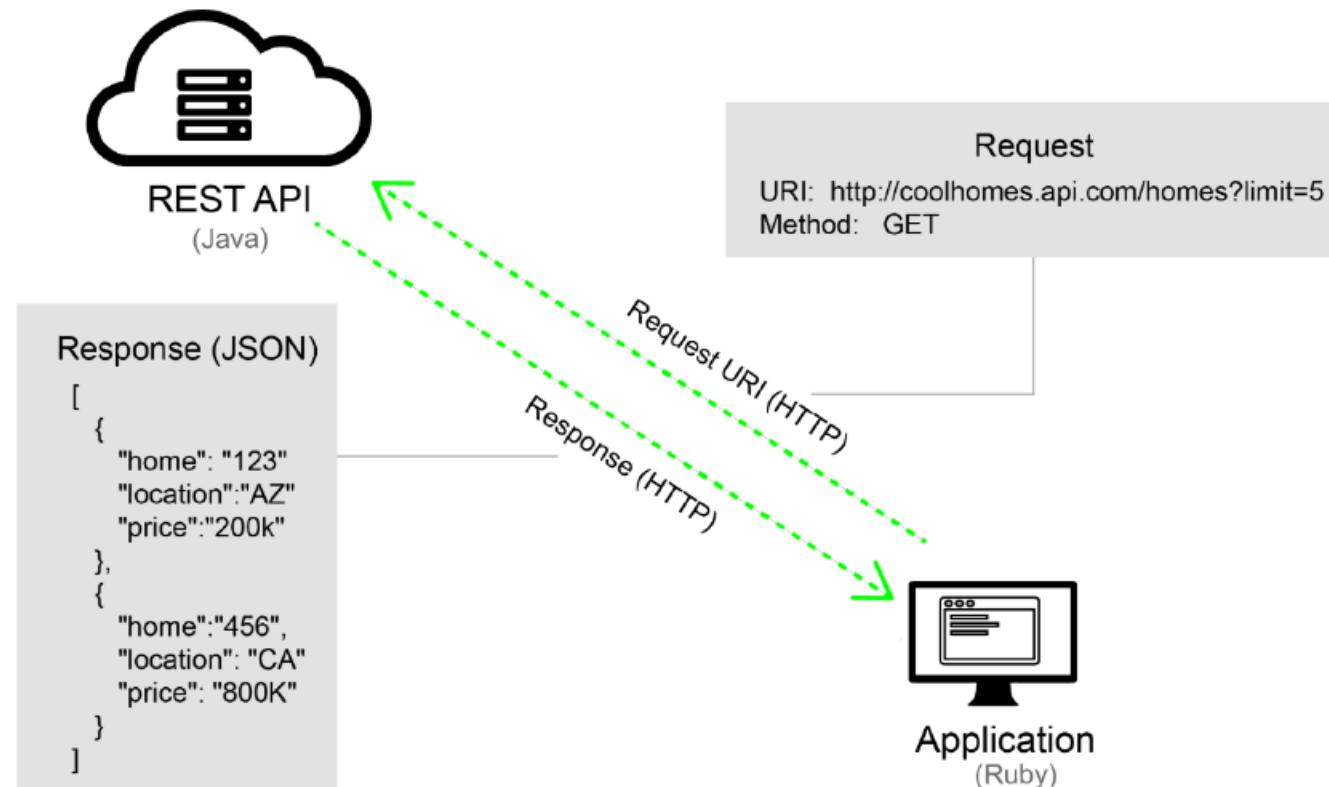
開檔/讀檔 - XML

```
1  
2 import xml.etree.ElementTree as ET  
3 tree = ET.parse('country_data.xml')  
4 root = tree.getroot()  
5  
6 import xmltodict  
7 d = dict(xmltodict.parse(xml))  
8
```

#動手實作

- ① 取出班次一的每一個時間
- ② 將班次一的每一個時間用一種資料型態保存
- ③ 將班次一到五與其所有時間用一種資料型態個別保存
- ④ 依據上述動作將班次一到班次五都存在同一個變數中
- ⑤ 計算班次一到班次五個別的開車時間與平均開車時間

API 資料來源與 Request 串接存取



Preparation

① 安裝 postman google extension:

<https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop/related?hl=zh-TW>

② 安裝 jsonview google extension:

<https://chrome.google.com/webstore/detail/jsonview/chklaanhfefbnpoihckbnefhakgolnmc/related?hl=zh-TW>

<https://chrome.google.com/webstore>

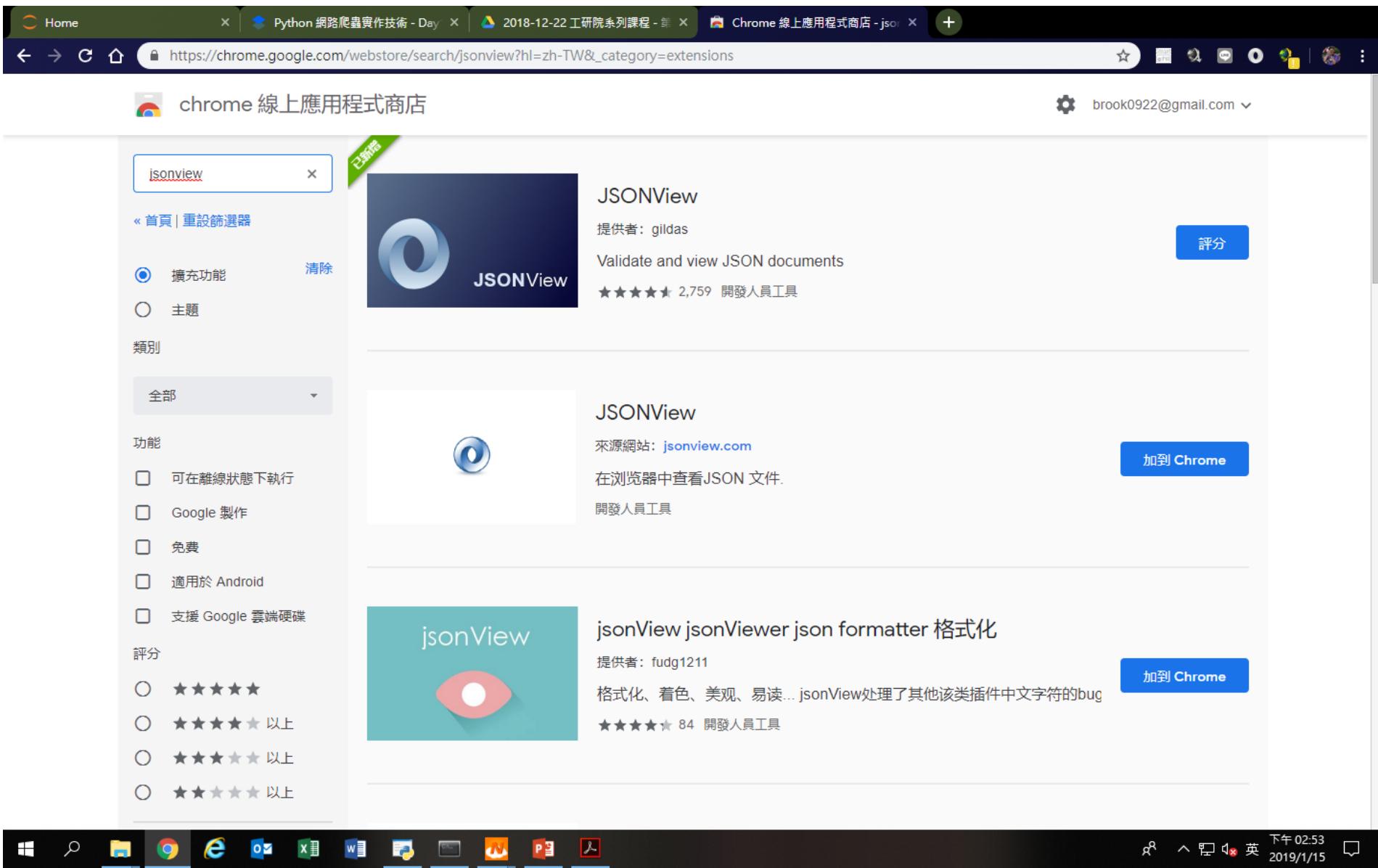
The screenshot shows the Chrome Web Store interface. On the left, there's a sidebar with filters for '擴充功能' (Extensions), '主題' (Themes), and '類別' (Categories). Below these are dropdown menus for '全部' (All) and '功能' (Features), with checkboxes for '可在離線狀態下執行' (Offline execution), 'Google 製作' (Made by Google), '免費' (Free), '適用於 Android' (Android compatible), and '支援 Google 雲端硬碟' (Google Cloud Storage compatible). There are also sections for '評分' (Rating) with radio buttons for 5-star, 4.5-star, 4-star, 3.5-star, and 3-star reviews.

The main content area features a large promotional image for the 'EARTH VIEW' extension, which displays a stunning aerial view of a waterfall. To the right of the image, the text reads: 'EARTH VIEW Experience a beautiful image from Google Earth every time you open a new tab.' Below this, there are five small circular navigation dots.

Below the main image, there's a section titled '為你推薦' (Recommended for you) with four featured extensions: 'LINE it!' (green background), 'MEGA Chrome Extension' (red background), 'Full Page Screenshots Capture. Edit. Save.' (blue background), and 'Booking.com for Chrome™' (dark blue background). Each card includes the extension name, its logo, and a star rating with the number of reviews.

At the bottom of the page, there's a navigation bar with icons for Home, Python 網路爬蟲實作技術 - Day, 2018-12-22 工研院系列課程, Chrome 線上應用程式商店, and a search bar containing the URL https://chrome.google.com/webstore/category/extensions?hl=zh-TW. The status bar at the bottom right shows the date and time: 下午 02:51 2019/1/15.

安裝 : jsonview



Request Library

```
1 import requests  
2 # 引入函式庫  
3 r = requests.get('https://github.com/timeline.json')  
4 # 想要爬資料的目標網址  
5 response = r.text  
6 # 模擬發送請求的動作  
7  
8
```

#Note：安裝一下 JSONView extension

101

#Example

```
1 import requests
2 # 引入函式庫
3 r = requests.get('https://www.dcard.tw/_api/forums')
4 # 想要爬資料的目標網址
5 response = r.text
6 # 模擬發送請求的動作
7
8
```

#Example

```
1  
2 import json  
3 data = json.loads(response)  
4  
5 for d in data:  
6     print(d['title'])  
7  
8
```

#動手實作

- ① 請告訴我這個 API 一次會回傳幾筆資料？每一筆資料包含哪些欄位？
- ② 取出每一筆資料的「標題」、「貼文時間」、「留言人數」、「按讚人數」
- ③ 計算熱門/非熱門文章的「平均留言人數」與「平均按讚人數」
- ④ 請問 Dcard 總共有幾個看板？
- ⑤ 請問最熱門的看板是哪一個？