

目次

Module 1. 開發環境：Anaconda、Jupyter 及爬蟲專案實務.....	1
1-1: 安裝及設定 Anaconda、Jupyter	1
1-2: Jupyter 操作技巧及課程套件安裝.....	19
1-3: 爬蟲專案開發實務分享.....	34
Module 2. 基礎回顧之資料結構.....	35
2-1: 字串物件操作.....	35
2-2: List 物件操作.....	38
2-3: Dict 物件操作.....	40
Module 3. 基礎回顧之流程控制與迴圈.....	42
3-1: 流程控制 (if…elif…else)	42
3-2: 迴圈 (for / while)	43
3-3: 關鍵字與綜合使用 (continue / break)	43
3-3-1 補充: Tuple 的用法.....	45
3-3-2 補充: Set 的用法.....	47
3-3-3 練習: 將 list 當中重複的 dict 去除，並透過指定 dict key 來排序.....	50
Module 4. 關於 URL 與字串格式化.....	55
4-1: 解析 URL (urlparse) 與產生 URL.....	55
4-2: 如何使用 URL 編碼 (quote / unquote)	56
4-3: 字串格式化.....	57
Module 5. 正規表達式 (Regular Expression) 入門.....	58
5-1: 指示符號介紹 (^ / \$ / \d / \w / [?-?] / ...)	59
5-2: 計數符號介紹 ({?} / {?,?} / + / * / ? / ...)	59
5-3: 常用範例推導 (手機號碼表達式、身分證字號表達式、...)	60
Module 6. 正規表達式 (Regular Expression) 進階.....	61
6-1: 具名群組 (Named Group) 介紹.....	61
6-2: 常用函數介紹 (search /.findall / match / group / split / ...)	62
6-3: 具名群組在常用範例上的使用方式 (Email、URL、...)	65
Module 7. HTML 基礎與 HTTP 方法	67
7-1: 常見 HTML 架構介紹.....	67
7-2: 常見 HTTP 方法介紹.....	75
7-3: 細說 GET 與 POST 方法.....	75
Module 8. CSS Selector 與 XPath.....	80
8-1: 概述 CSS Selector 與 XPath.....	80
8-2: 細說 CSS Selector	80

8-3: 細說 XPath	89
Module 9. Chrome Developer Tool.....	92
9-1: 各頁籤常用功能簡介 (Elements / Console / Network / ...) ..	92
9-2: 資策會首頁分析.....	104
9-3: 常用操作流程介紹 (Preserve Log / Clear / ...)	108
Module 10. 套件 requests.....	109
10-1: 觀察理解目標並發出請求 (Request)	110
10-2: 自訂 HTTP Headers、Cookies 及 Session 使用	113
10-3: 解析回應內容 (HTML / JSON)	115
Module 11. 套件 Beautiful Soup 4	118
11-1: 套件介紹及常用功能.....	118
11-2: 解析 HTML，使用 CSS Selector 查找元素	119
11-3: 取出指定內容.....	121
Module 12. 案例實作.....	125
12-1: 分析頁面資訊結構.....	125
12-2: 使用套件解析並取出新聞清單.....	126
12-3: 遞迴取出新聞頁面內容.....	127
Module 13. 案例實作.....	128
13-1: 取出 JSON 內容.....	128
13-2: 解析 JSON 內容.....	128
13-3: 參數取代變換.....	128
Module 14. 套件 Selenium (初階)	131
14-1: 解析 Selenium、WebDriver 與 Browser 連動關係.....	131
14-2: 自訂 Selenium 啟動設定 (移除資訊列、全螢幕顯示、...) ..	131
14-3: 瀏覽器控制與取得網頁原始碼 (get / quit / page_source) ..	132
Module 15. 套件 Selenium (中階)	135
15-1: 進階控制 (alert / frame)	135
15-2: 如何查找頁面元素 (ID / Class / Tag / CSS Selector / ...) ..	136
15-3: 動作控制 (click / send_keys)	138
Module 16. 套件 Selenium (高階)	140
16-1: 等待 (WebDriverWait)	140
16-2: 期待狀況 (Expected Condition)	142
16-3: 根據條件 (By)	144
Module 17. 案例實作.....	146
17-1: 如何使用 Selenium 登入臉書.....	146
17-2: 如何使用 Chrome Developer Tools 分析臉書個人動態牆.....	147
17-3: 如何使用擷取臉書個人動態牆內容.....	149
Module 18. 股票案例實作及視覺化.....	155

18-1: 使用套件取得股價、量等資料.....	155
18-2: 繪製畫出價量及 K 棒.....	157
18-3: 匯出線圖保存.....	160
補充.....	162
Excel 操作.....	162

GitHub 專案超連結

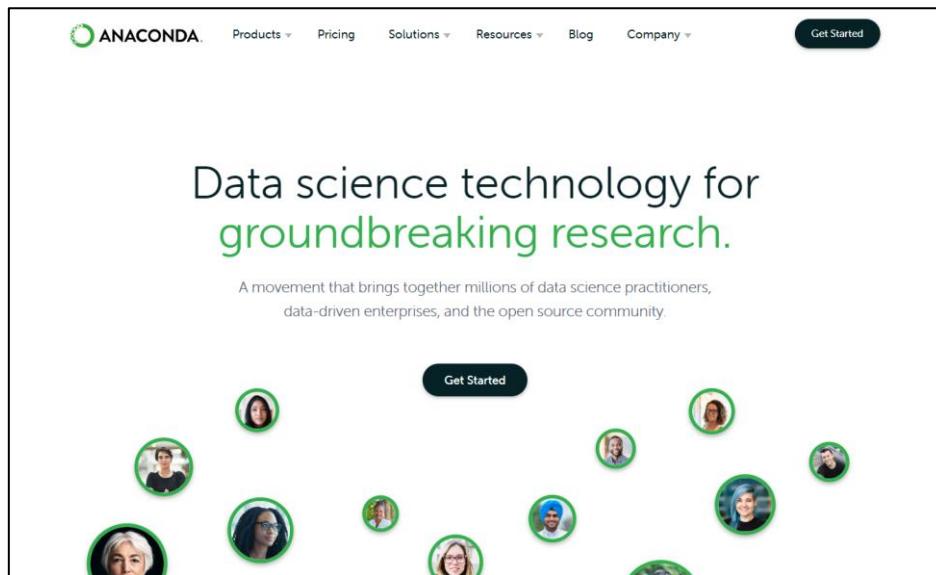
https://github.com/telunyang/python_web_scraping

Module 1. 開發環境：Anaconda、Jupyter 及爬蟲專案實務

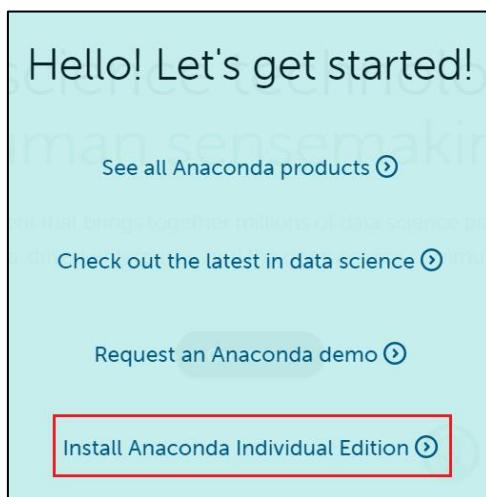
1-1: 安裝及設定 Anaconda、Jupyter

安裝 Anaconda

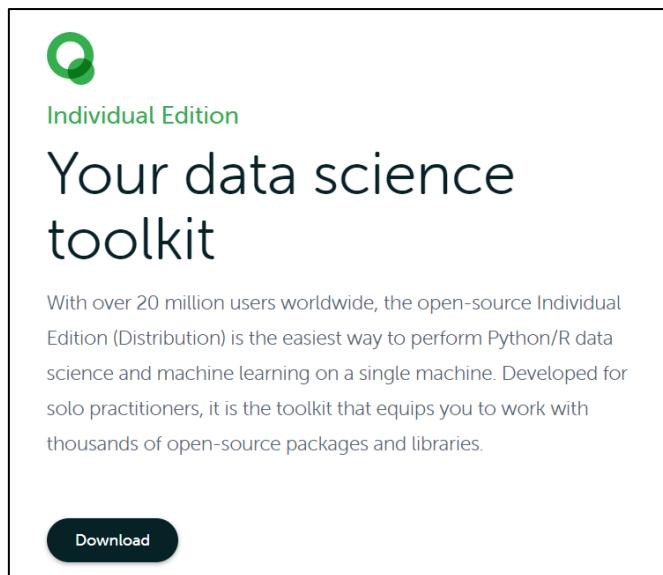
網址：<https://www.anaconda.com/>



圖：進入 Anaconda 首頁，按下右上角「Get Started」



圖：按下「Install Anaconda Individual Edition」



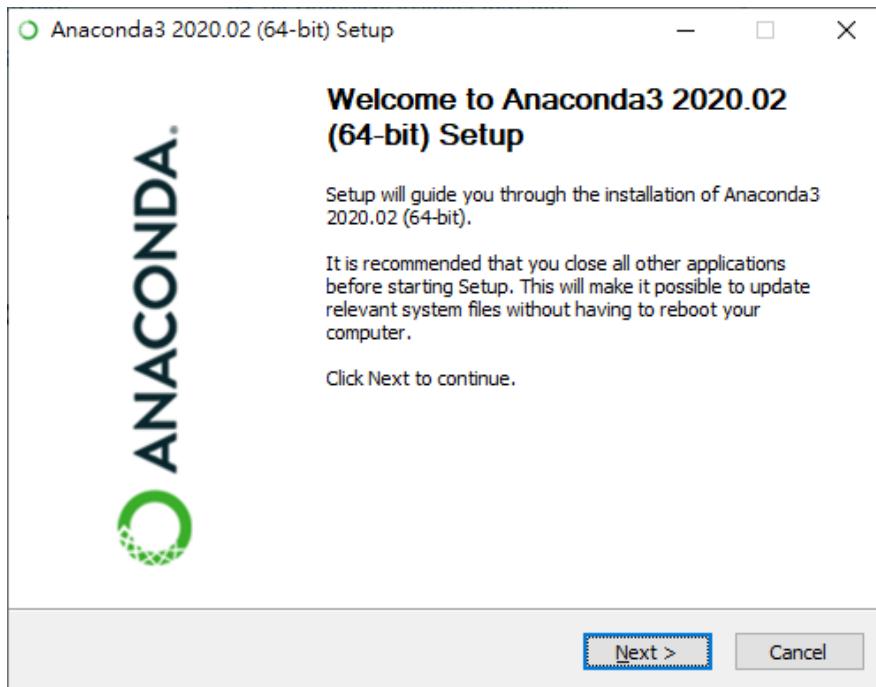
圖：按下左下角「Download」

A screenshot of the "Anaconda Installers" page. It shows three columns of download links: Windows, macOS, and Linux. The Windows column has links for Python 3.7 (64-Bit Graphical Installer, 466 MB) and Python 2.7 (64-Bit Graphical Installer, 413 MB). The macOS column has links for Python 3.7 (64-Bit Graphical Installer, 442 MB) and Python 2.7 (64-Bit Graphical Installer, 637 MB). The Linux column has links for Python 3.7 (64-Bit (x86) Installer, 522 MB) and Python 2.7 (64-Bit (x86) Installer, 477 MB).

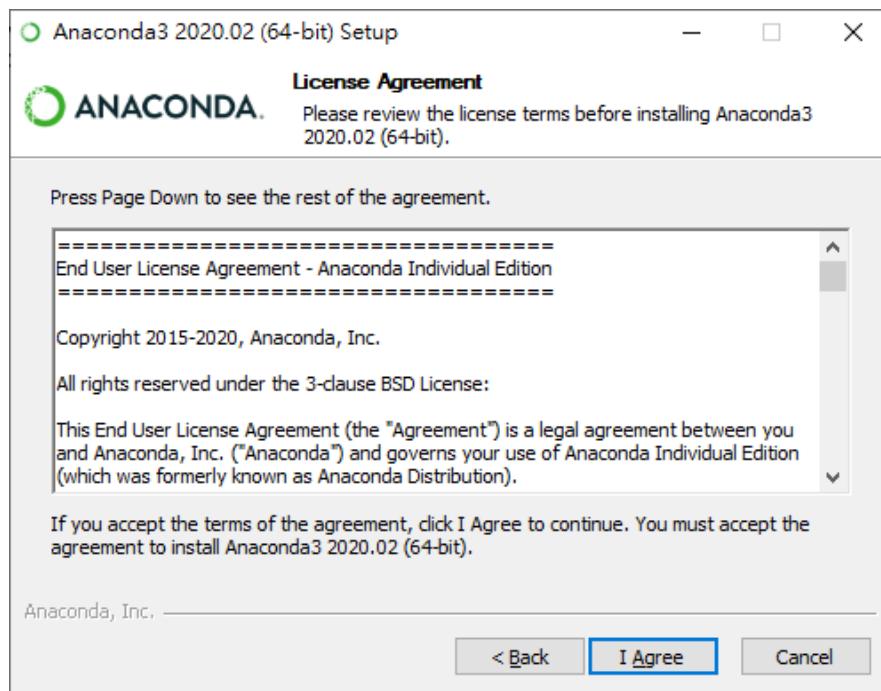
圖：選擇 Windows → Python3.7 → 64-Bit Graphical Installer (466 MB)



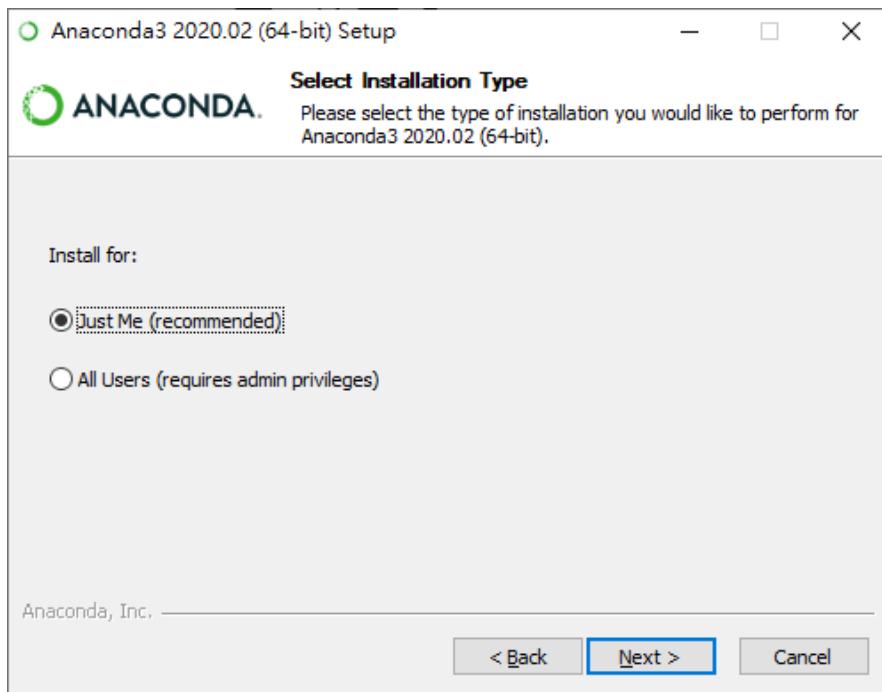
圖：下載後的檔案圖示，連點兩下進行安裝



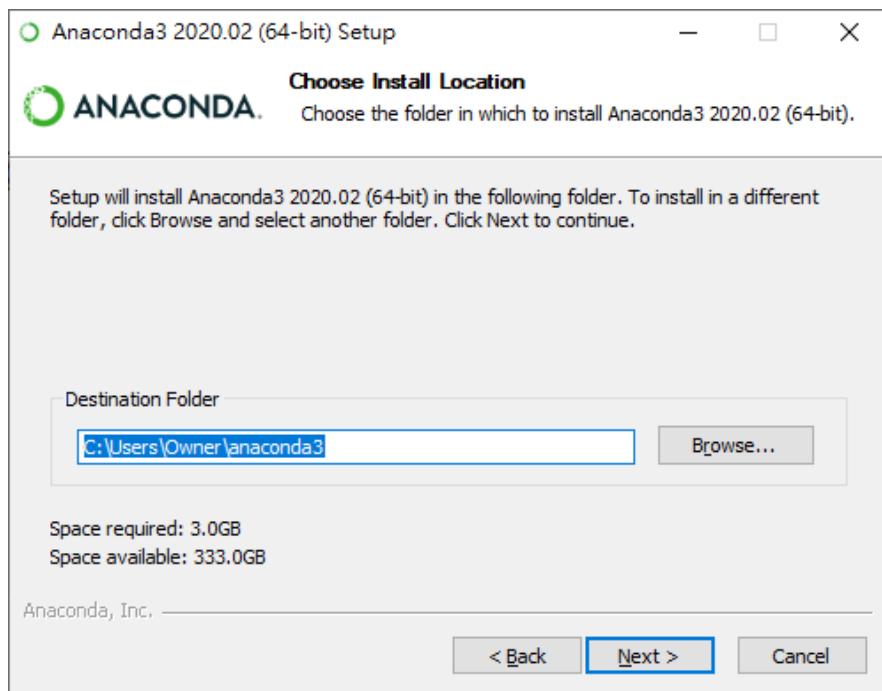
圖：按下「Next」



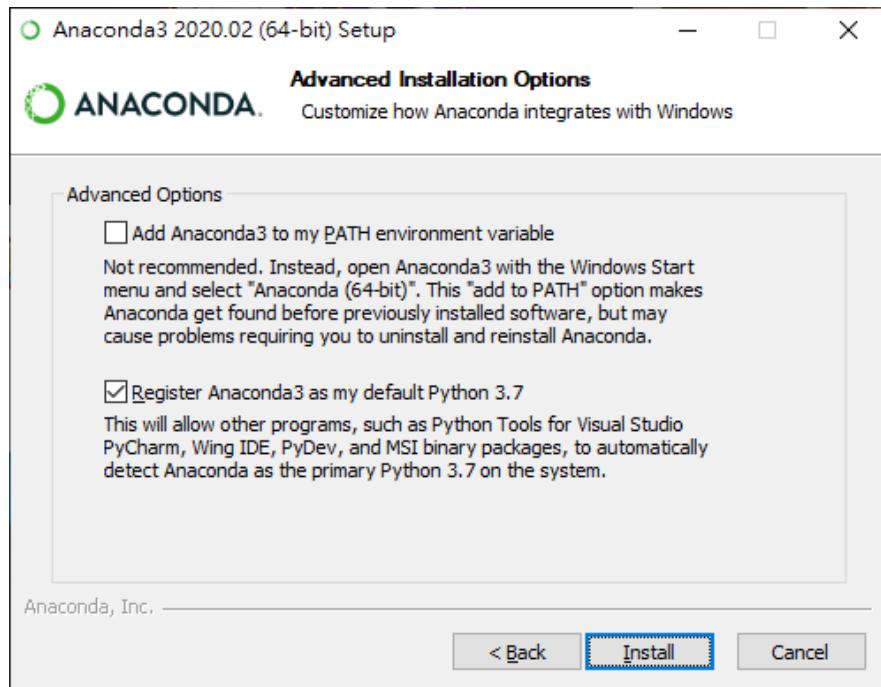
圖：按下「I Agree」



圖：選擇「Just Me」，按下「Next」



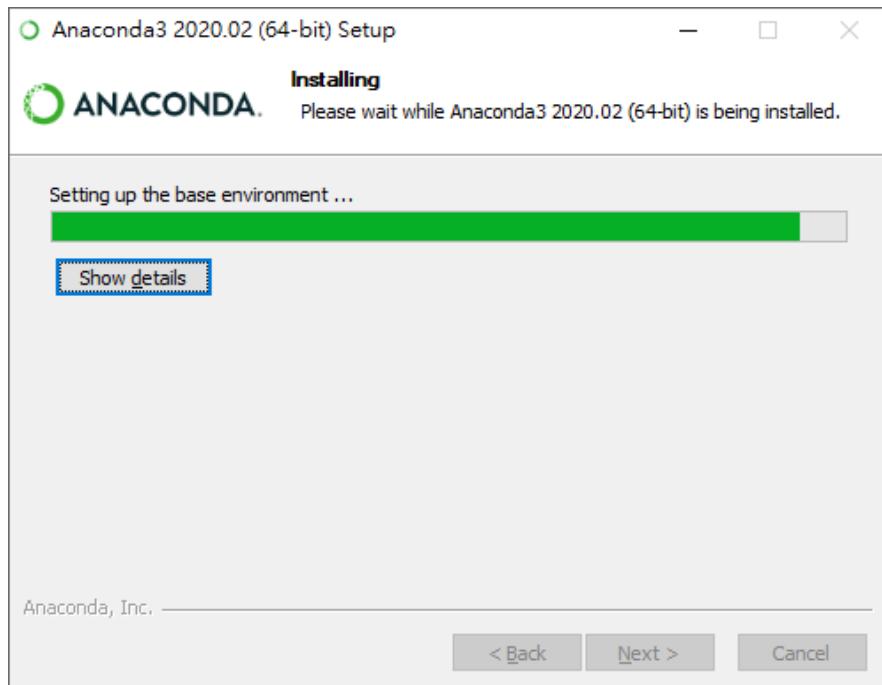
圖：使用預設目的資料夾，按下「Next」(我個人 Windows 帳號是 Owner)



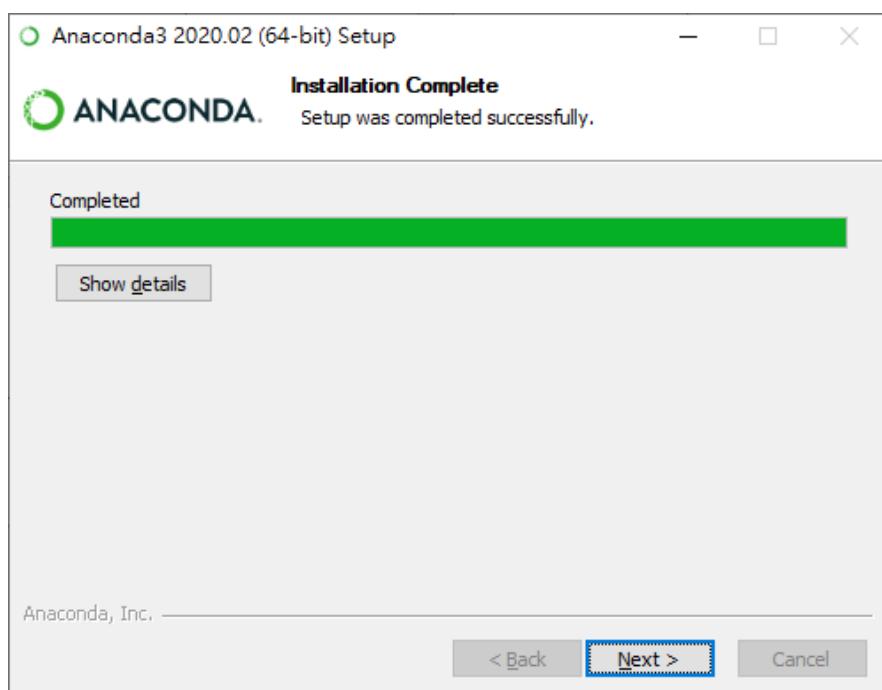
圖：若無額外需求，按下「Install」

說明

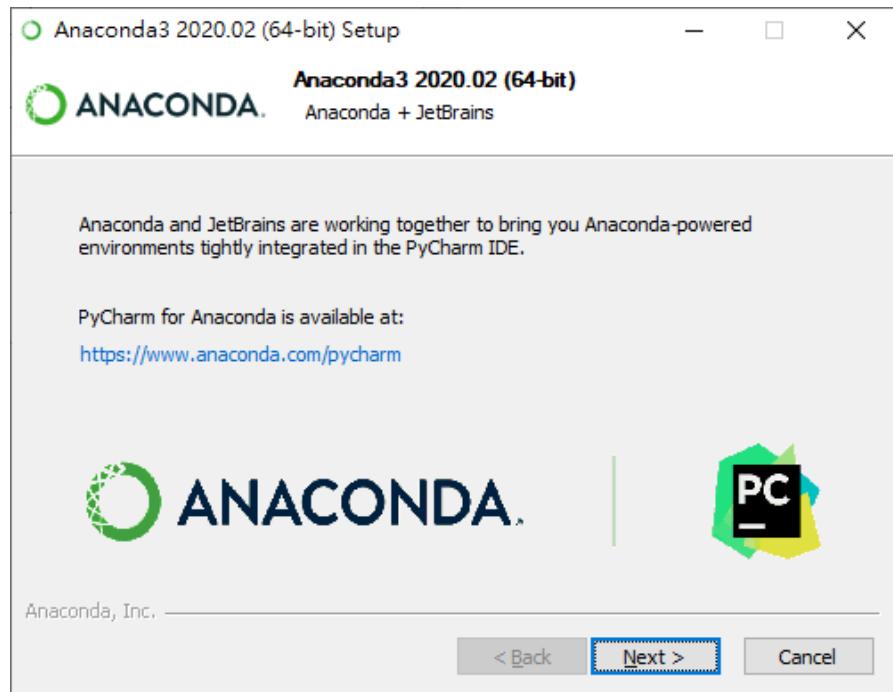
若是想要在 Anaconda(64-bit) 的命令提示字元視窗「以外」的地方使用 conda 指令，例如 Windows 命令提示字元，則需要勾選「Add Anaconda3 to my PATH environment variable」，否則使用預設設定即可（未勾選狀態）；若是之後才有需要用到，也可以直接在系統變數中設定 Anaconda 的執行路徑，亦可使用。



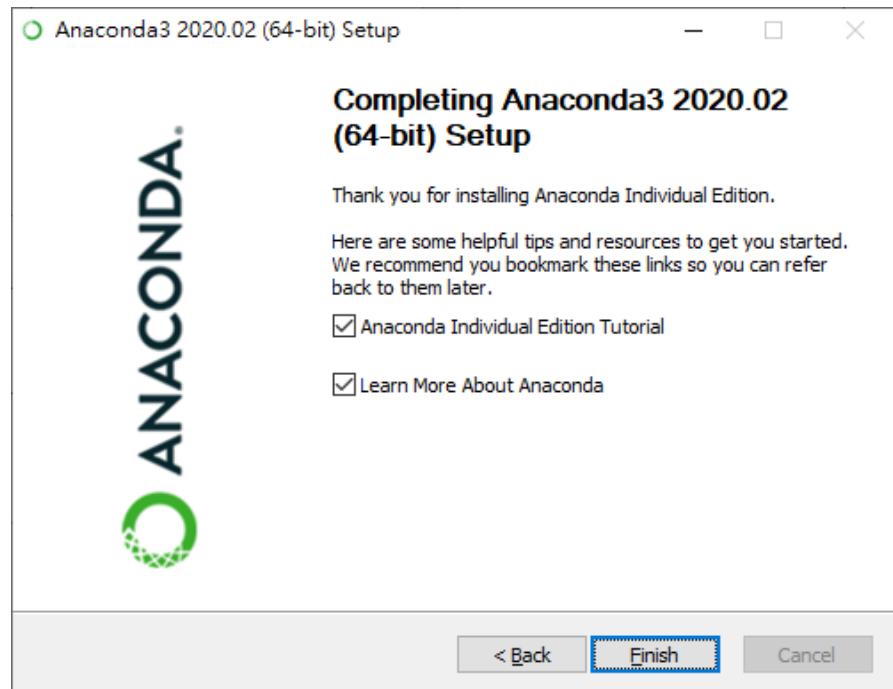
圖：安裝中，請稍候



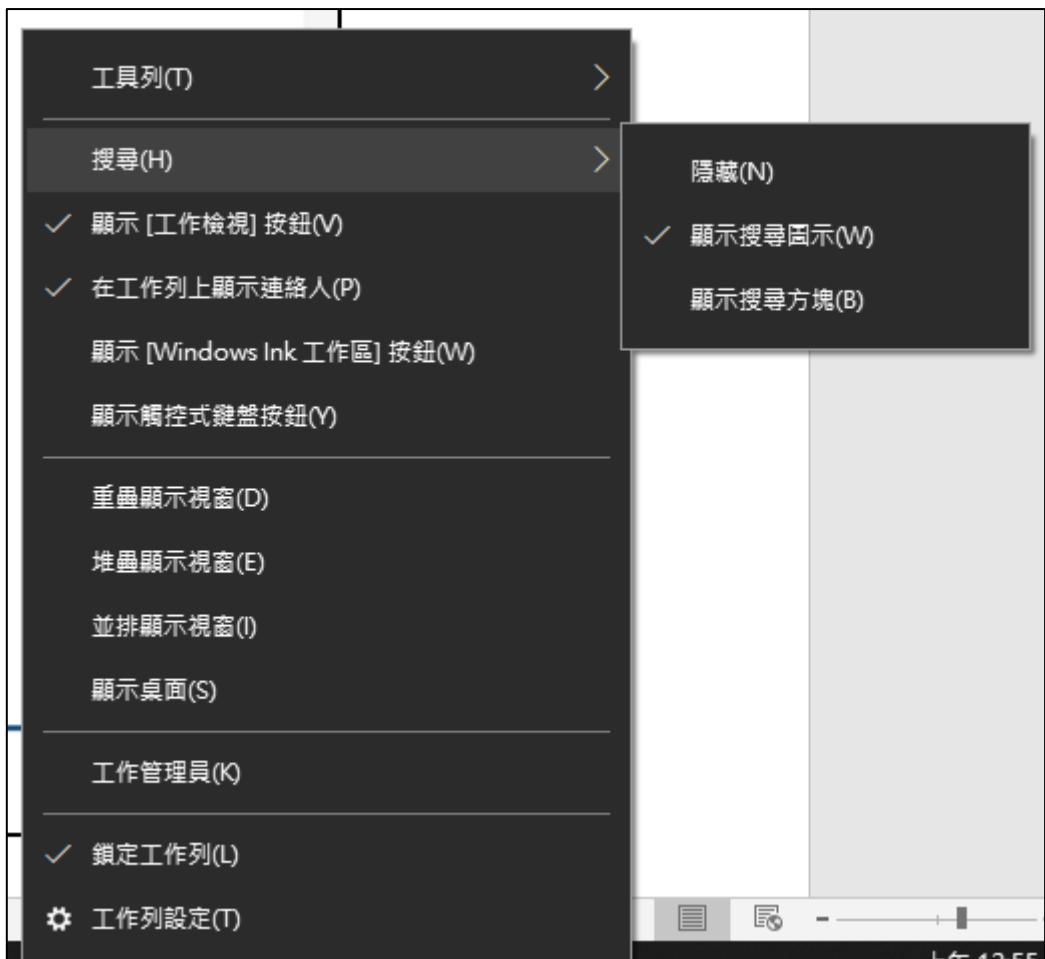
圖：安裝完成，按下「Next」



圖：按下「Next」



圖：上面兩個項目可以取消勾選，按下「Next」



圖：顯示搜尋圖示



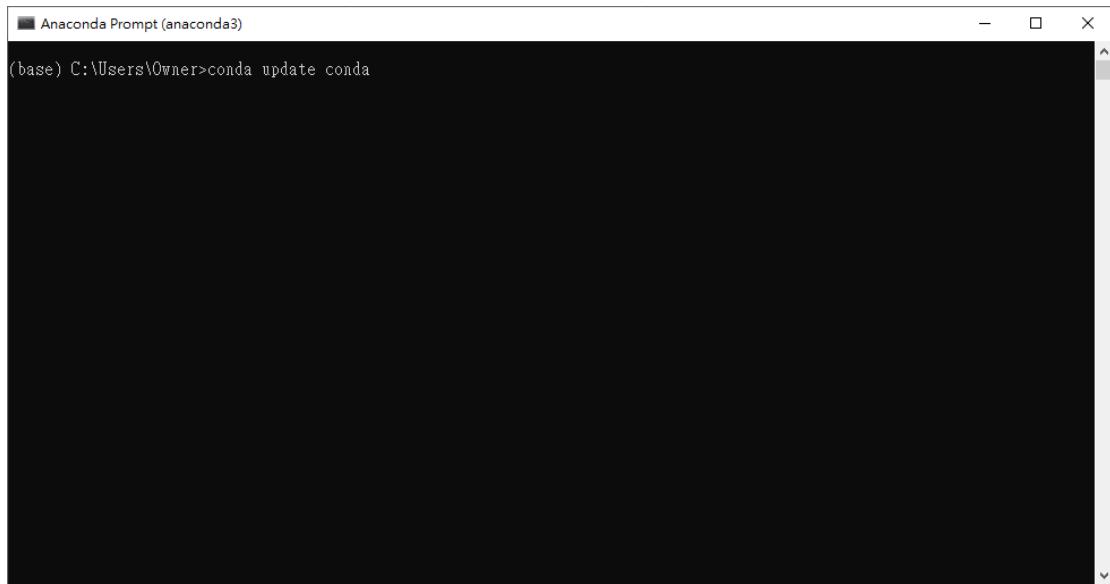
圖：搜尋圖示類似放大鏡，按下搜尋圖示



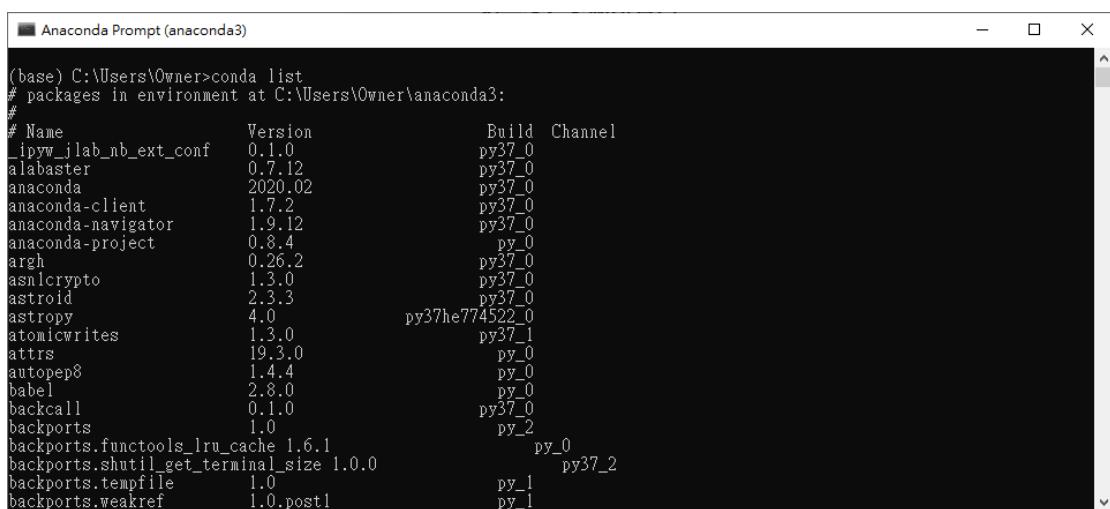
圖：搜尋「anaconda prompt」，按下「Anaconda Prompt (anaconda3)」



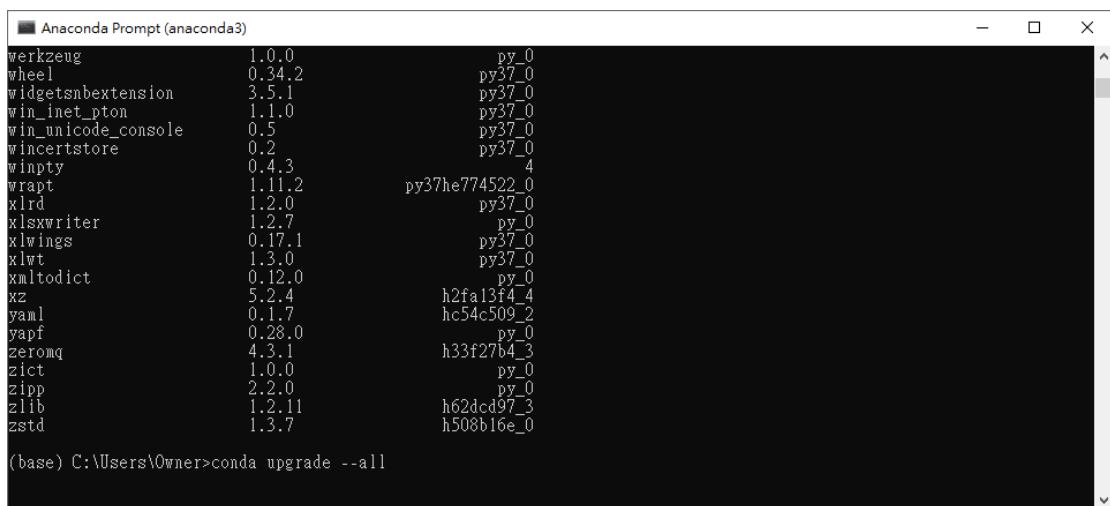
圖：出現 Anaconda Prompt，類似 Windows 的命令提示字元



圖：輸入「`conda update conda`」本身



圖：輸入「`conda list`」，會顯示目前預設安裝的套件（在虛擬環境 base 下）



圖：輸入「`conda upgrade --all`」，更新當前所有套件

```
■ Anaconda Prompt (anaconda3) - conda upgrade --all
sphinxcontrib-qth~          1.0.2-py_0 --> 1.0.3-py_0
sphinxcontrib-ser~          1.1.3-py_0 --> 1.1.4-py_0
sphinxcontrib-web~          1.2.0-py_0 --> 1.2.1-py_0
spyder                      4.0.1-py37_0 --> 4.1.3-py37_0
spyder-kernels              1.8.1-py37_0 --> 1.9.1-py37_0
sqlalchemy                  1.3.13-py37he774522_0 --> 1.3.16-py37he774522_0
sqlite                      3.31.1-he774522_0 --> 3.31.1-h2a8f88b_1
tornado                     6.0.3-py37he774522_3 --> 6.0.4-py37he774522_1
tqdm                        4.42.1-py_0 --> 4.46.0-py_0
wcwidth                     0.1.8-py_0 --> 0.1.9-py_0
werkzeug                    1.0.0-py_0 --> 1.0.1-py_0
xlsxwriter                 1.2.7-py_0 --> 1.2.8-py_0
xlwings                     0.17.1-py37_0 --> 0.19.0-py37_0
xz                           5.2.4-h2fa13f4_4 --> 5.2.5-h62dc97_0
zict                        1.0.0-py_0 --> 2.0.0-py_0
zipp                        2.2.0-py_0 --> 3.1.0-py_0
zlib                         1.2.11-h62dc97_3 --> 1.2.11-h62dc97_4

The following packages will be DOWNGRADED:
anaconda                   2020.02-py37_0 --> custom-py37_1
lzo                          2.10-h6df0209_2 --> 2.10-he774522_2

Proceed ([y]/n)?
```

圖：按下「y」後，再按鍵盤「Enter」

```
■ Anaconda Prompt (anaconda3) - conda upgrade --all
lzo                         2.10-h6df0209_2 --> 2.10-he774522_2

Proceed ([y]/n)? y

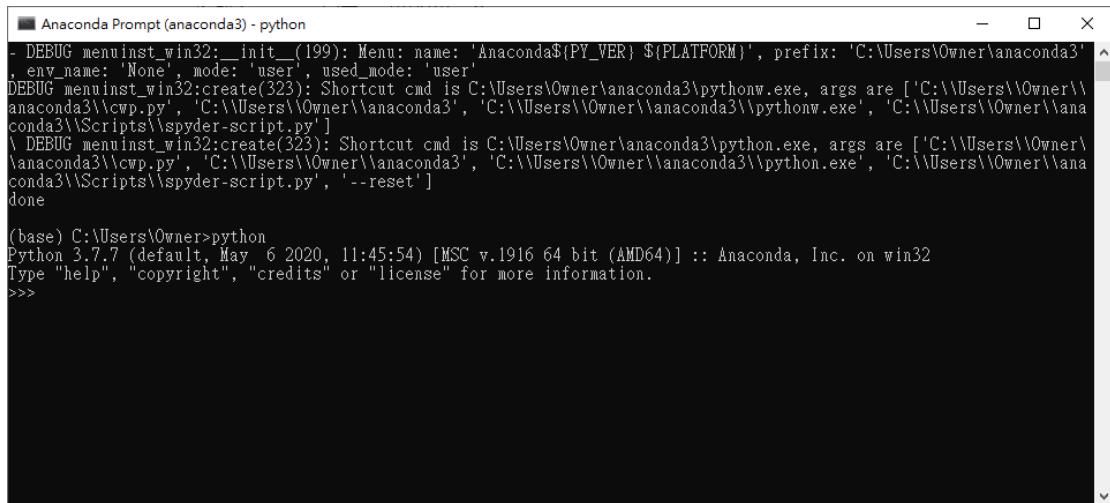
Downloading and Extracting Packages
kiwisolver-1.2.0           | 55 KB    | #####| 100%
seaborn-0.10.1              | 163 KB   | #####| 100%
dask-2.16.0                 | 14 KB    | #####| 100%
jupyter_core-4.6.3          | 85 KB    | #####| 100%
conda-4.8.3                 | 2.8 MB   | #####| 100%
cython-0.29.17              | 1.8 MB   | #####| 100%
prompt_toolkit-3.0.4        | 11 KB    | #####| 100%
fsspec-0.7.1                | 56 KB    | #####| 100%
requests-2.23.0             | 93 KB    | #####| 100%
sphinxcontrib-applehelp      | 27 KB    | #####| 100%
typed-ast-1.4.1             | 141 KB   | #####| 100%
curl-7.69.1                 | 126 KB   | #####| 100%
python-libarchive-c          | 46 KB    | #####| 100%
_anaconda_depends-20         | 6 KB     | #####| 100%
python-language-server       | 94 KB    | #####| 100%
qtawesome-0.7.0             | 726 KB   | #####| 100%
spyder-kernels-1.9.1         | 96 KB    | #####| 100%
sqlalchemy-1.3.16            | 1.5 MB   | #####| 0%
```

圖：更新套件中

```
■ Anaconda Prompt (anaconda3)
- DEBUG menuinst_win32:_init_(199): Menu: name: 'Anaconda${PY_VER} ${PLATFORM}', prefix: 'C:\Users\Owner\anaconda3'
, env_name: 'None', mode: 'user', used_mode: 'user'
DEBUG menuinst_win32:create(323): Shortcut cmd is C:\Users\Owner\anaconda3\pythonw.exe, args are ['C:\Users\Owner\anaconda3\cwp.py', 'C:\Users\Owner\anaconda3\pythonw.exe', 'C:\Users\Owner\anaconda3\Scripts\spyder-script.py']
\ DEBUG menuinst_win32:create(323): Shortcut cmd is C:\Users\Owner\anaconda3\python.exe, args are ['C:\Users\Owner\anaconda3\cwp.py', 'C:\Users\Owner\anaconda3', 'C:\Users\Owner\anaconda3\python.exe', 'C:\Users\Owner\anaconda3\Scripts\spyder-script.py', '--reset']
done

(base) C:\Users\Owner>
```

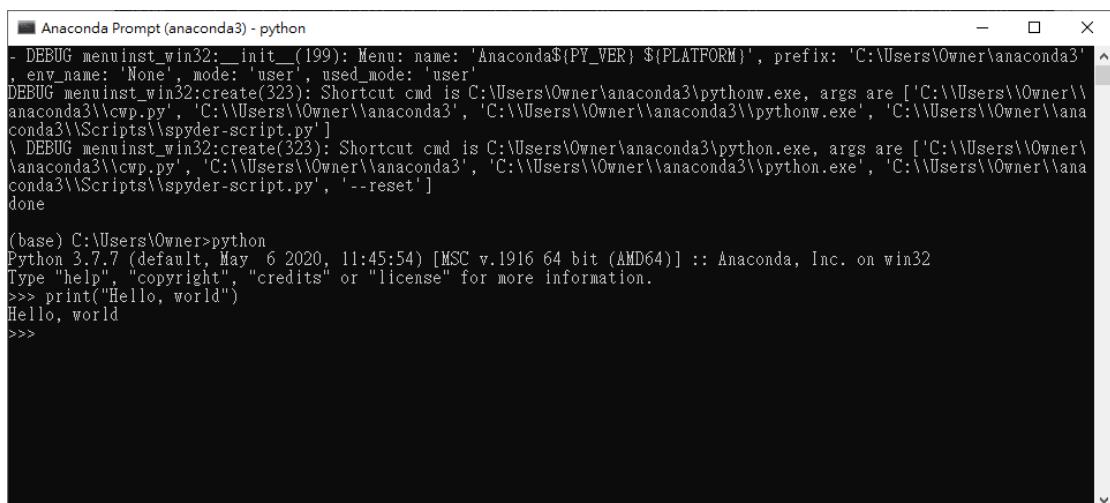
圖：套件更新完成



```
■ Anaconda Prompt (anaconda3) - python
- DEBUG menuinst_win32:_init_(199): Menu: name: 'Anaconda${PY_VER} ${PLATFORM}', prefix: 'C:\Users\Owner\anaconda3'
, env_name: 'None', mode: 'user', used_mode: 'user'
DEBUG menuinst_win32:create(323): Shortcut cmd is C:\Users\Owner\anaconda3\pythonw.exe, args are ['C:\Users\Owner\anaconda3\cwp.py', 'C:\Users\Owner\anaconda3', 'C:\Users\Owner\anaconda3\pythonw.exe', 'C:\Users\Owner\anaconda3\Scripts\spyder-script.py']
\ DEBUG menuinst_win32:create(323): Shortcut cmd is C:\Users\Owner\anaconda3\python.exe, args are ['C:\Users\Owner\anaconda3\cwp.py', 'C:\Users\Owner\anaconda3', 'C:\Users\Owner\anaconda3\python.exe', 'C:\Users\Owner\anaconda3\Scripts\spyder-script.py', '--reset']
done

(base) C:\Users\Owner>python
Python 3.7.7 (default, May 6 2020, 11:45:54) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

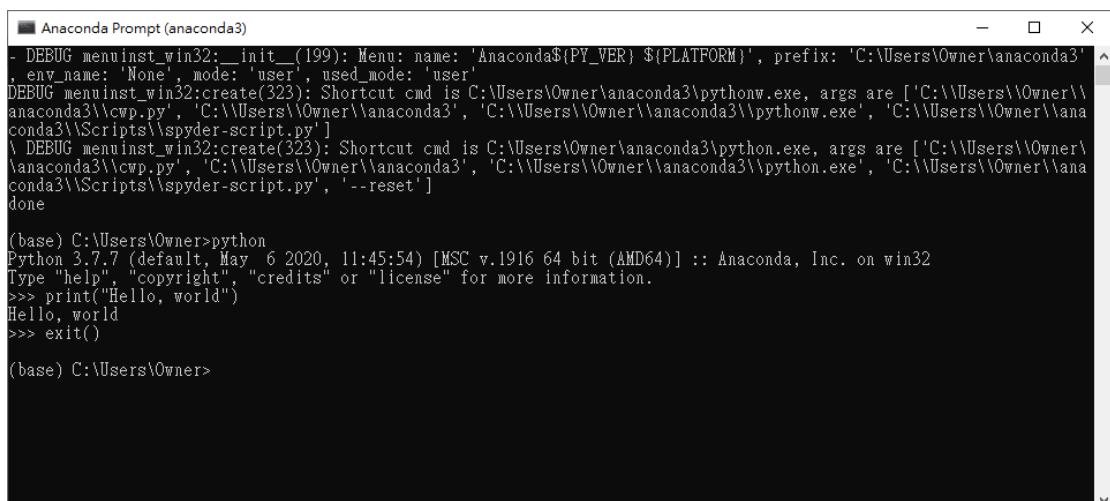
圖：輸入「python」，進入 python 執行環境



```
■ Anaconda Prompt (anaconda3) - python
- DEBUG menuinst_win32:_init_(199): Menu: name: 'Anaconda${PY_VER} ${PLATFORM}', prefix: 'C:\Users\Owner\anaconda3'
, env_name: 'None', mode: 'user', used_mode: 'user'
DEBUG menuinst_win32:create(323): Shortcut cmd is C:\Users\Owner\anaconda3\pythonw.exe, args are ['C:\Users\Owner\anaconda3\cwp.py', 'C:\Users\Owner\anaconda3', 'C:\Users\Owner\anaconda3\pythonw.exe', 'C:\Users\Owner\anaconda3\Scripts\spyder-script.py']
\ DEBUG menuinst_win32:create(323): Shortcut cmd is C:\Users\Owner\anaconda3\python.exe, args are ['C:\Users\Owner\anaconda3\cwp.py', 'C:\Users\Owner\anaconda3', 'C:\Users\Owner\anaconda3\python.exe', 'C:\Users\Owner\anaconda3\Scripts\spyder-script.py', '--reset']
done

(base) C:\Users\Owner>python
Python 3.7.7 (default, May 6 2020, 11:45:54) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, world")
Hello, world
>>>
```

圖：輸入「print("Hello world")」，輸出「Hello world」，python 安裝成功



```
■ Anaconda Prompt (anaconda3)
- DEBUG menuinst_win32:_init_(199): Menu: name: 'Anaconda${PY_VER} ${PLATFORM}', prefix: 'C:\Users\Owner\anaconda3'
, env_name: 'None', mode: 'user', used_mode: 'user'
DEBUG menuinst_win32:create(323): Shortcut cmd is C:\Users\Owner\anaconda3\pythonw.exe, args are ['C:\Users\Owner\anaconda3\cwp.py', 'C:\Users\Owner\anaconda3', 'C:\Users\Owner\anaconda3\pythonw.exe', 'C:\Users\Owner\anaconda3\Scripts\spyder-script.py']
\ DEBUG menuinst_win32:create(323): Shortcut cmd is C:\Users\Owner\anaconda3\python.exe, args are ['C:\Users\Owner\anaconda3\cwp.py', 'C:\Users\Owner\anaconda3', 'C:\Users\Owner\anaconda3\python.exe', 'C:\Users\Owner\anaconda3\Scripts\spyder-script.py', '--reset']
done

(base) C:\Users\Owner>python
Python 3.7.7 (default, May 6 2020, 11:45:54) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, world")
Hello, world
>>> exit()

(base) C:\Users\Owner>
```

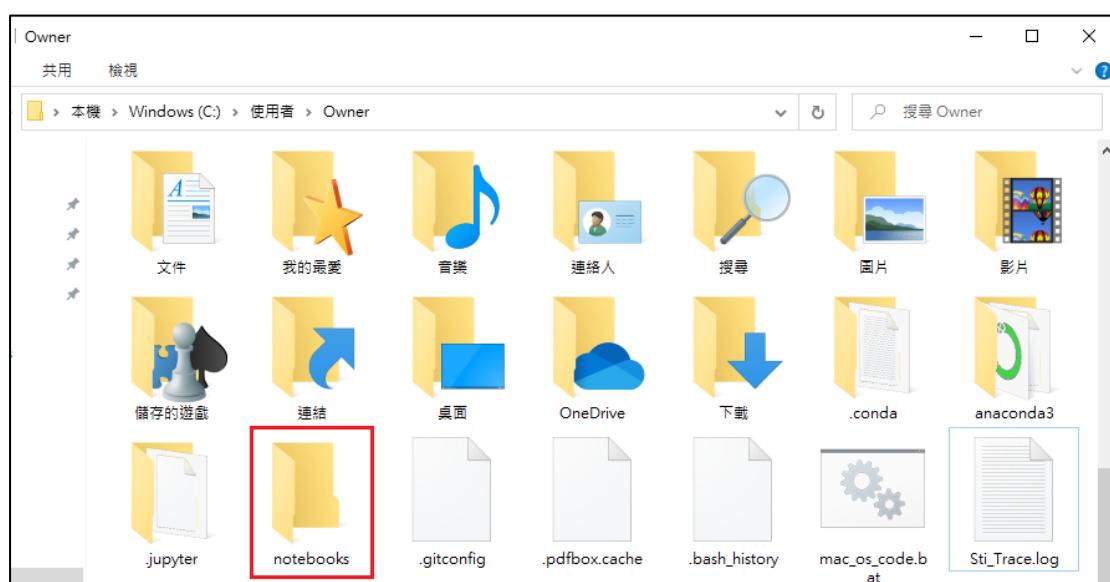
圖：按下「exit()」回到指令輸入的環境

設定 Jupyter

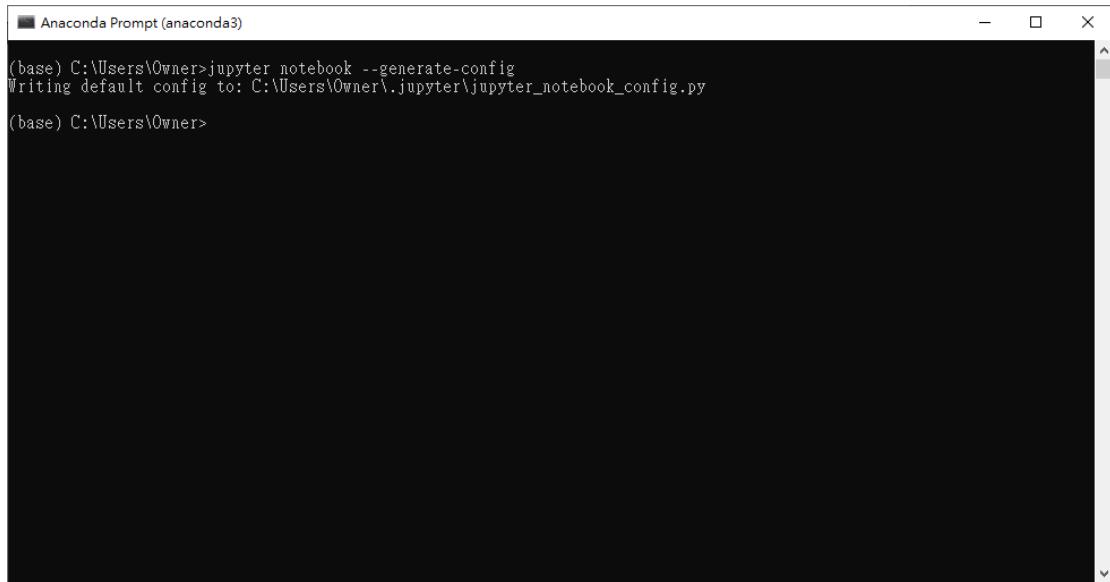
Jupyter 是一個互動式的程式計算環境，協助我們在網頁上，以互動方式撰寫、執行程式碼，並即時看到執行結果，同時創建自己的 Jupyter Notebook 文件（以文件形式來儲存 python 程式碼）。

如果我們今天僅是使用 Jupyter 預設的環境，路徑應該在「C:\Users\{你的帳號}」底下，我們只要將專案解壓縮，放至路徑底下即可（本課程將以預設路徑進行教學）。

若是需要建立屬於自己的工作目錄，而非使用預設設定（預設在 C:\Users\{你的帳號} 底下，會與其它目錄混在一起，不方便管理）。



圖：到 C:\Users\{你的帳號}，新增一個資料夾「notebooks」，並複製路徑



```
(base) C:\Users\Owner>jupyter notebook --generate-config
Writing default config to: C:\Users\Owner\.jupyter\jupyter_notebook_config.py
(base) C:\Users\Owner>
```

圖：輸入「jupyter notebook --generate-config」，建立設定檔

備註
Jupyter 的 config 檔，預設在 C:\Users\{你的帳號}\.jupyter\ 裡面；若是先前已經建立且修改，再輸入時，可以再度產生 config 檔，可以選擇覆蓋過舊的檔案。



圖：使用記事本或編輯器來開啟「jupyter_notebook_config.py」，約 226 行



```
jupyter_notebook_config.py - 記事本
檔案(F) 檔案(E) 檢視(V) 檢視
## The login handler class to use.
#c.NotebookApp.login_handler_class = 'notebook.auth.login.LoginHandler'

## The logout handler class to use.
#c.NotebookApp.logout_handler_class = 'notebook.auth.logout.LogoutHandler'

## The MathJax.js configuration file that is to be used.
#c.NotebookApp.mathjax_config = 'TeX-AMS-MML_HTMLorMML-full,Safe'

## A cust... 寶找 X to
# MathJa... 寶找目標(N): c.NotebookApp.notebook_dir 找下一個(B)
# Content... 方向 取消
# configu...  大小寫視為相異(C)  向上(U)  向下(D)
# Note: n...  環繞(R)

## Gets or sets the maximum amount of memory, in bytes, that is allocated for
## use by the buffer manager.
#c.NotebookApp.max_buffer_size = 536870912

## Gets or sets a lower bound on the open file handles process resource limit.
# This may need to be increased if you run into an OSError: [Errno 24] Too many
# open files. This is not applicable when running on Windows.
#c.NotebookApp.min_open_files_limit = 0

## Dict of Python modules to load as notebook server extensions. Entry values can
## be used to enable and disable the loading of the extensions. The extensions
## will be loaded in alphabetical order.
#c.NotebookApp.nbserver_extensions = {}

## The directory to use for notebooks and kernels.
#c.NotebookApp.notebook_dir = "
```

圖：搜尋「c.NotebookApp.notebook_dir」，並將該行註解「#」移除

程式碼
<pre>## The directory to use for notebooks and kernels. c.NotebookApp.notebook_dir = 'C:\\\\Users\\\\Owner\\\\notebooks'</pre>

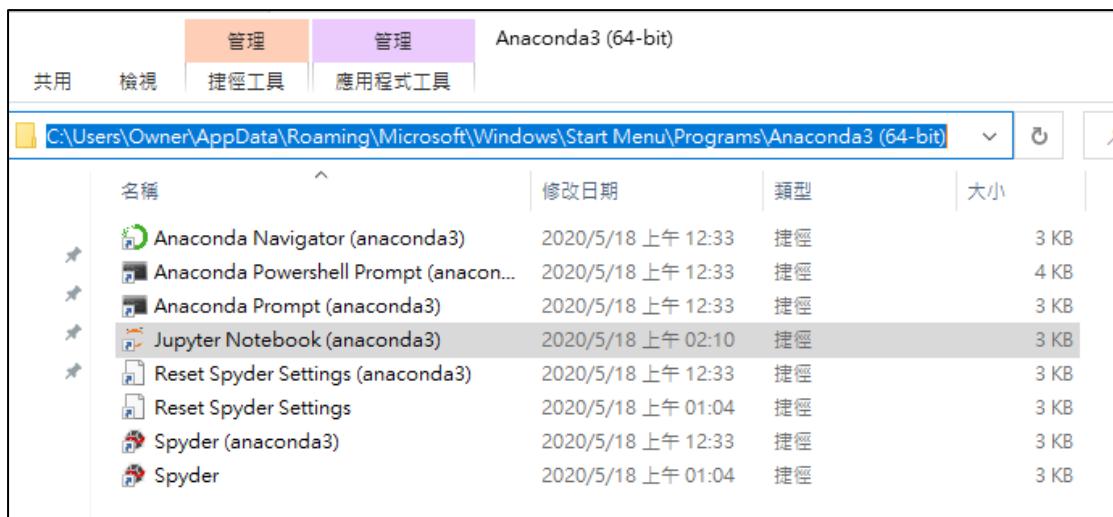
將路徑貼至「」之中，儲存關閉（在 Windows 底下，反斜線需要兩個）。

編輯 Jupyter Notebook 檔案連結

接下來我們要編輯 Jupyter Notebook 檔案連結，有兩種方法：

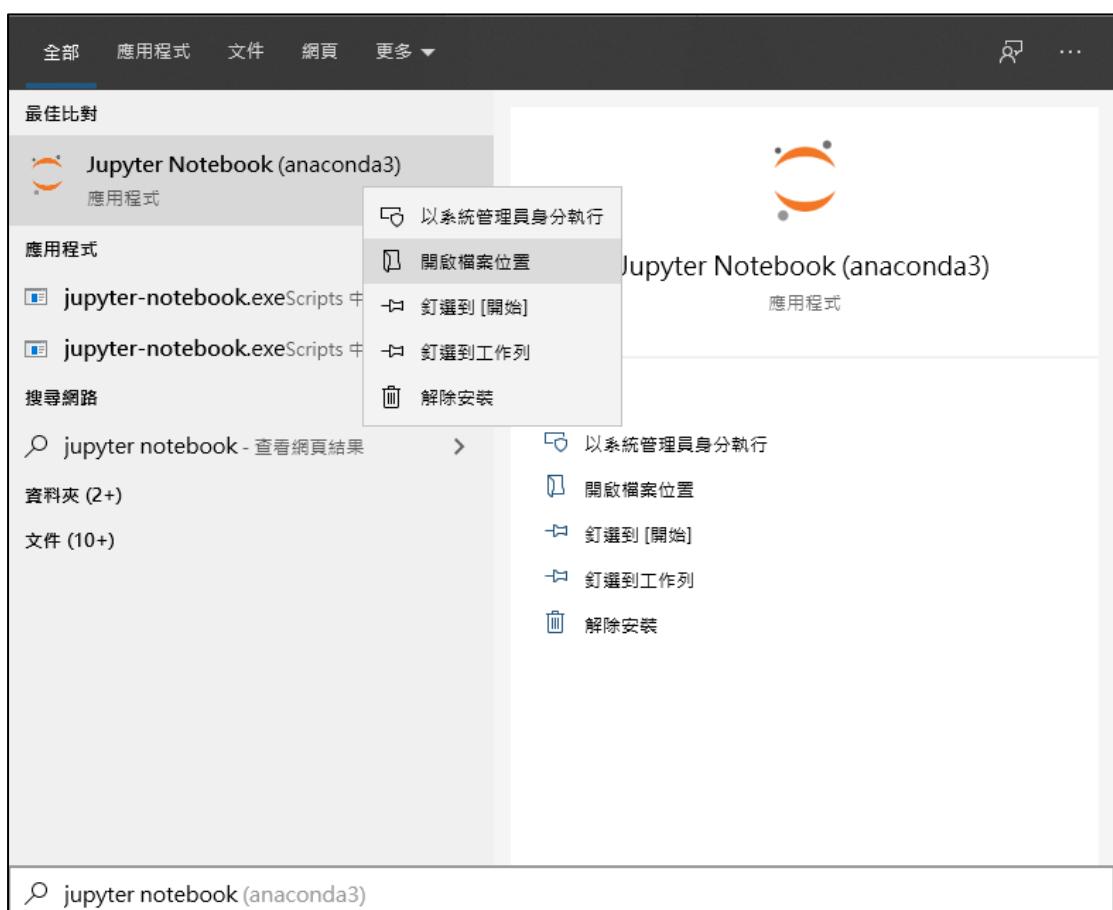
- 到「C:\Users\{你的帳號}\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Anaconda3 (64-bit)」中，找到「Jupyter Notebook

(anaconda3)」。

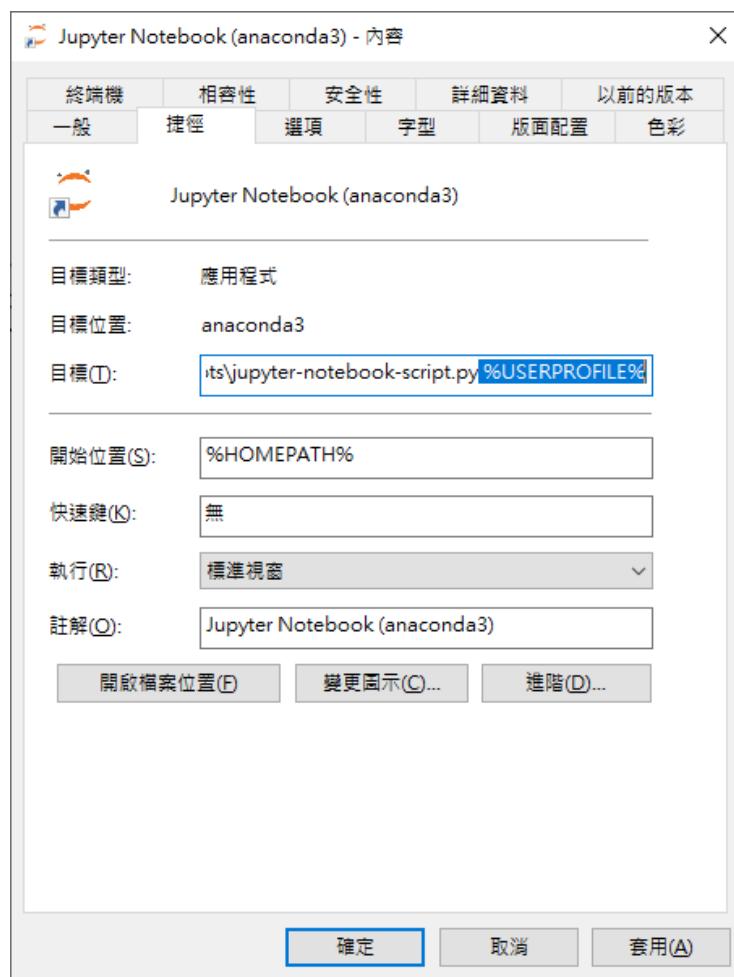
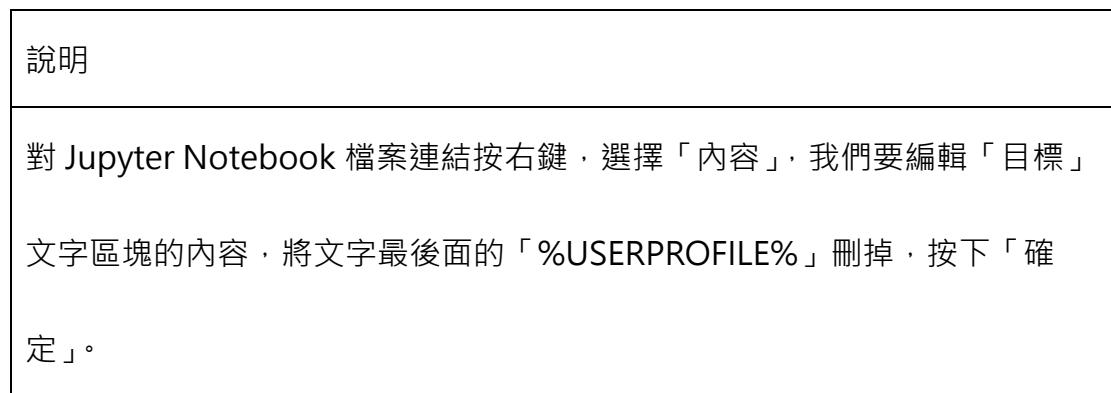


圖：直接在檔案總管的路徑列中，輸入檔案路徑。

- 按下放大鏡圖示，搜尋「Jupyter Notebook (anaconda3)」，然後對搜尋結果的圖示按下右鍵，再按下「開啟檔案位置」。



圖：透過搜尋來取得 Jupyter Notebook 的檔案連結位置

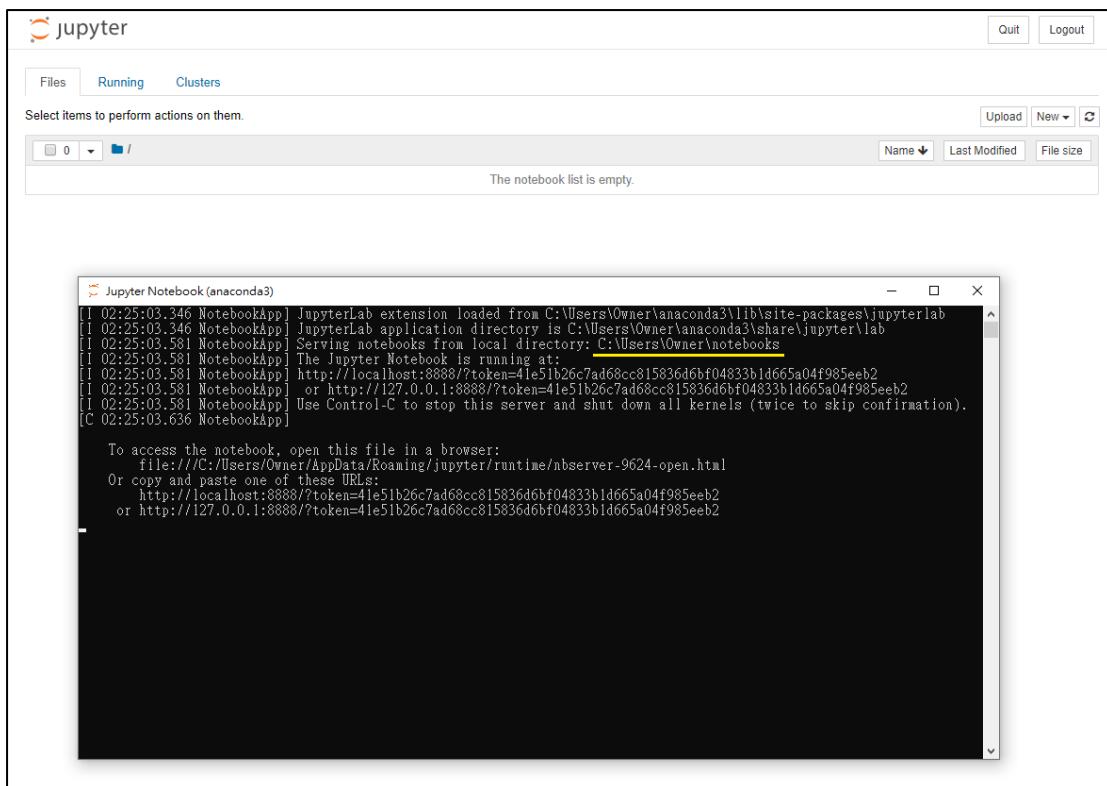


圖：刪除「%USERPROFILE%」的字樣後，按下確定

執行 Jupyter Notebook



圖：搜尋「jupyter notebook」，開啟「Jupyter Notebook (anaconda3)」



圖：執行 Jupyter Notebook 後，會以自訂路徑作為預設工作目錄

開啟 Jupyter Notebook 時，也會啟動 Jupyter Server，到這裡 Anaconda 安裝告一個段落。

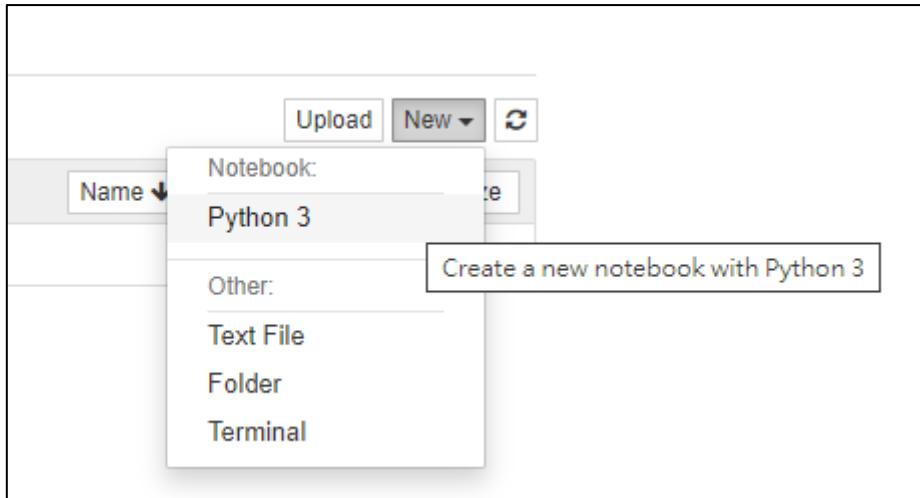
參考資料

jupyter notebook 更改默认工作路径

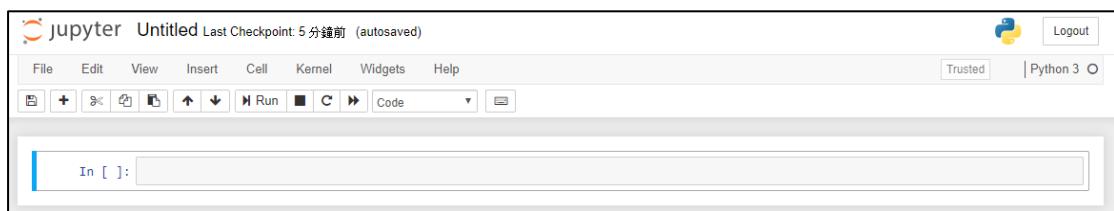
<https://zhuanlan.zhihu.com/p/90269779>

1-2: Jupyter 操作技巧及課程套件安裝

新增 Notebook



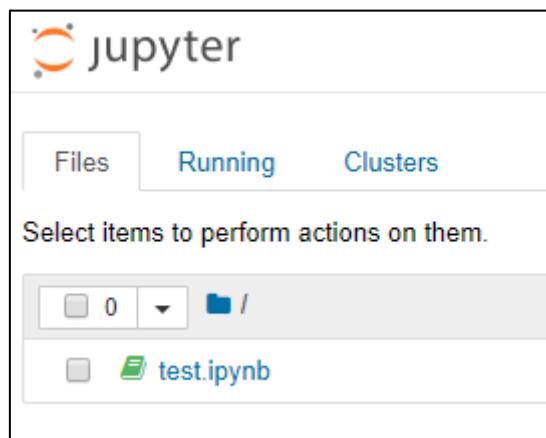
圖：按下 New · 新增 Notebook (選擇 Python3)



圖：此時進入新建的 jupyter notebook document 中

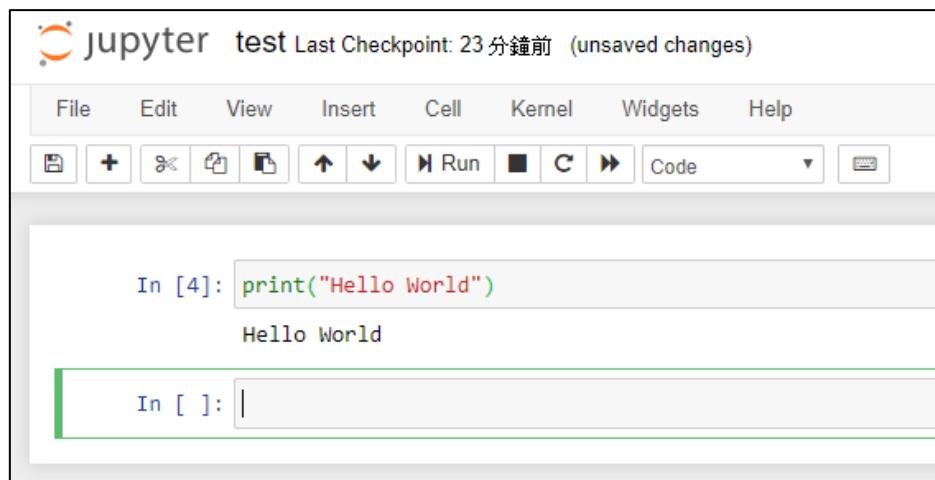


圖：按下「Untitled」，重新命名 Notebook name · 按下「Rename」



圖：工作目錄中，有對應名稱的「.ipynb」檔（Jupyter Notebook 檔）

執行 cell 中的程式碼



圖：在第一個 cell 輸入「`print("Hello World")`」，按下「Shift + Enter」

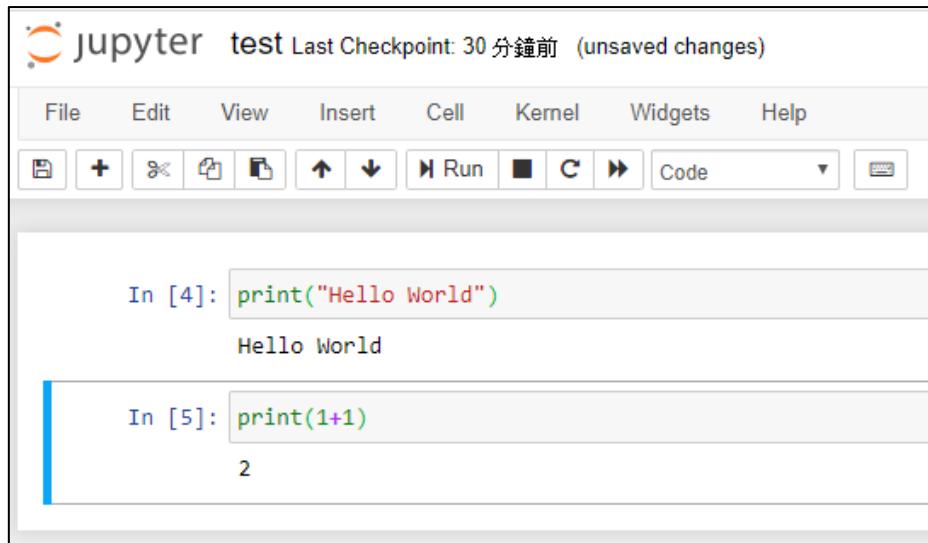
說明

Shift + Enter

執行該 cell，同時新增 cell

Ctrl + Enter

執行該 cell，不新增 cell

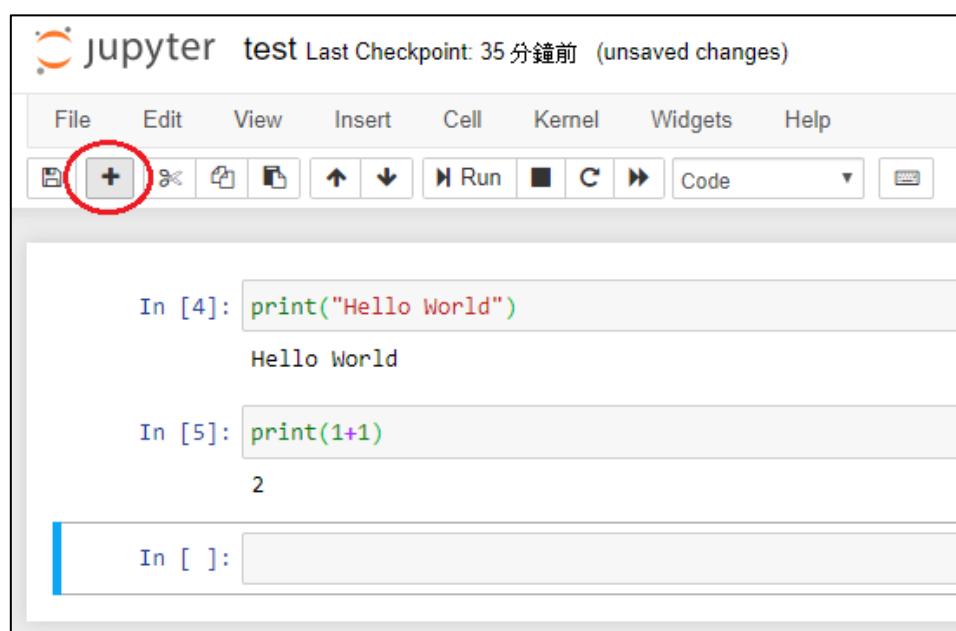


```
In [4]: print("Hello World")
Hello World

In [5]: print(1+1)
2
```

圖：在第二個 cell 輸入「`print(1+1)`」，按下 `Ctrl + Enter` 的輸出結果

新增 cell



The '+' button in the toolbar is highlighted with a red circle.

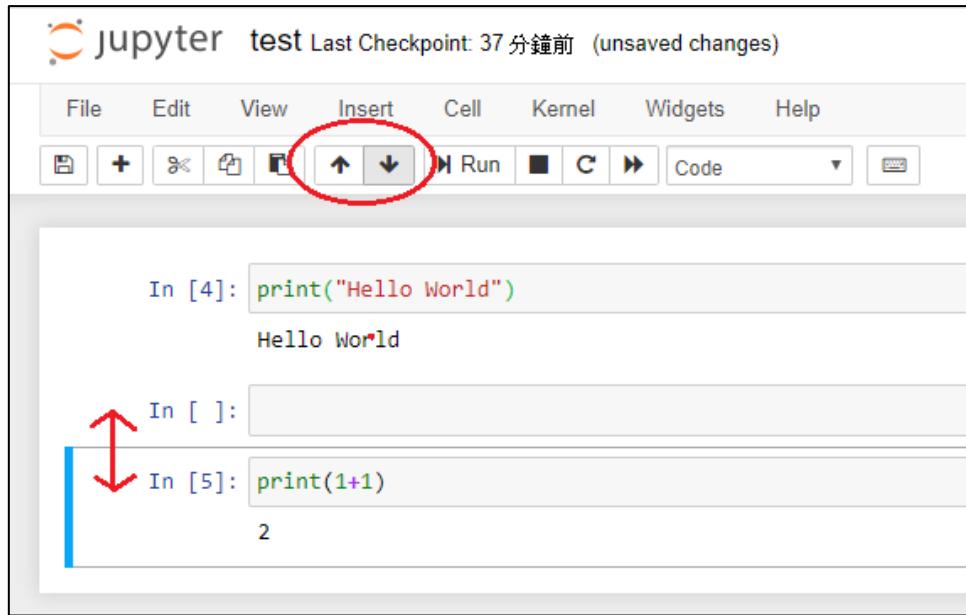
```
In [4]: print("Hello World")
Hello World

In [5]: print(1+1)
2

In [ ]:
```

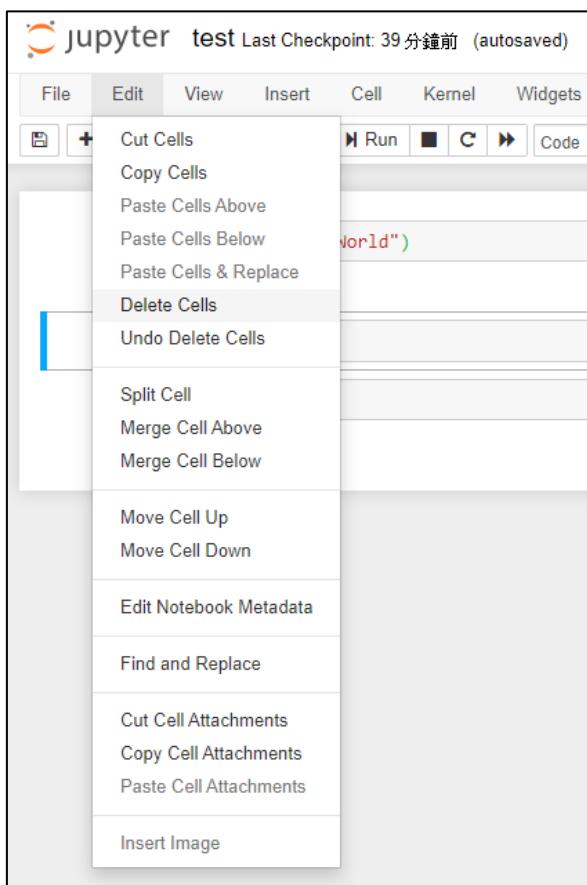
圖：按下「+」符號，會在當前 cell 下新增空白的 cell

移動 cell 位置



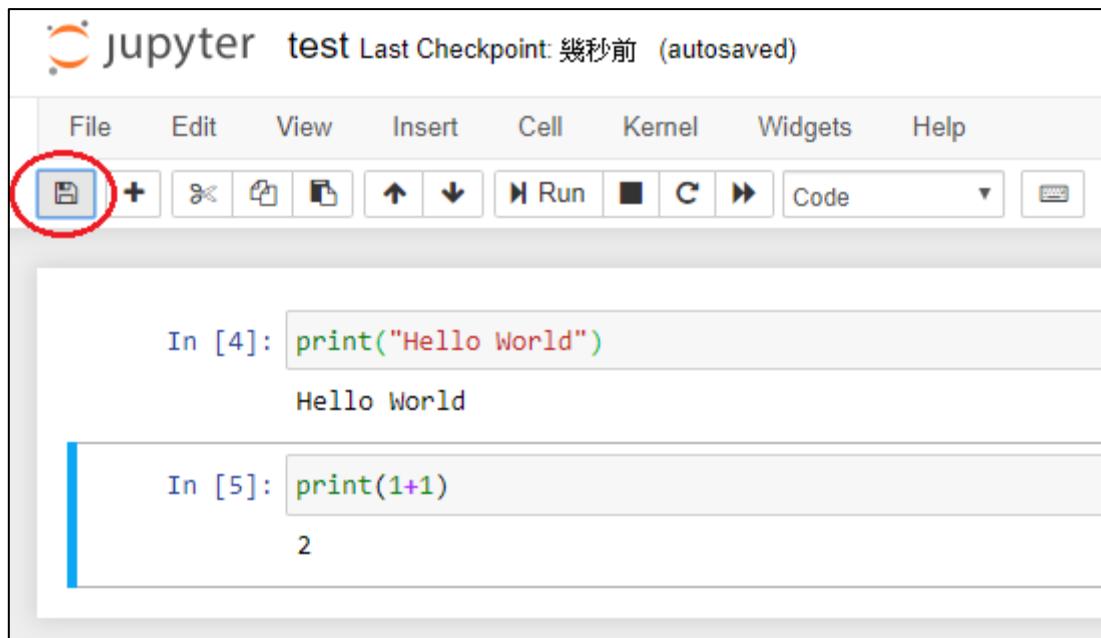
圖：選定 cell 後，按下「↑」或「↓」，可以移動 cell 的位置

刪除 cell



圖：選定 cell 後，按下「Edit」，選擇「Delete Cells」，即可刪除 cell

儲存 notebook

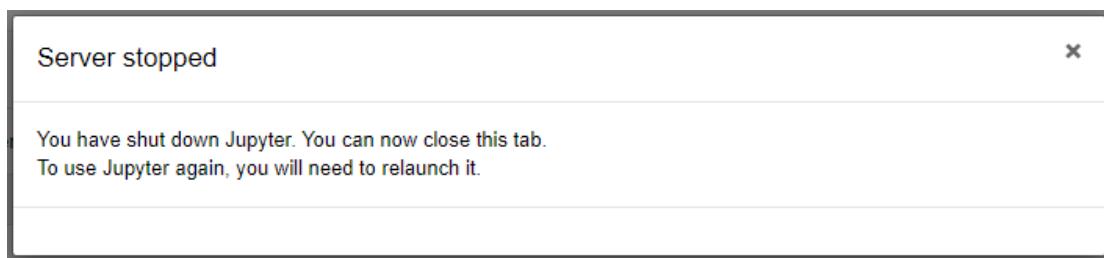


圖：按下儲存鈕，即可儲存 notebook

關閉 Jupyter Server



圖：按下 Quit，即可關閉 Jupyter Server



圖：Jupyter Server 關閉成功的訊息

在 Jupyter Notebook 中安裝套件

我們可以直接在 cell 中，以「！」開頭，讓 Jupyter Notebook 了解到我們輸入

的不是程式碼，而是指令。

```
In [8]: !pip install selenium
Collecting selenium
  Using cached selenium-3.141.0-py2.py3-none-any.whl (904 kB)
Requirement already satisfied: urllib3 in c:\users\owner\anaconda3\lib\site-packages (from selenium) (1.25.8)
Installing collected packages: selenium
Successfully installed selenium-3.141.0
In [ ]:
```

圖：在 Jupyter Notebook 中，安裝套件的參考範例

備註

若是需要反安裝，則輸入「`!pip uninstall -y selenium`」，參數 `-y` 代表過程不需要確認，直接幫你按 `y` 的意思。

```
In [10]: !pip uninstall -y selenium
Found existing installation: selenium 3.141.0
Uninstalling selenium-3.141.0:
  Successfully uninstalled selenium-3.141.0
In [ ]:
```

圖：反安裝套件的畫面

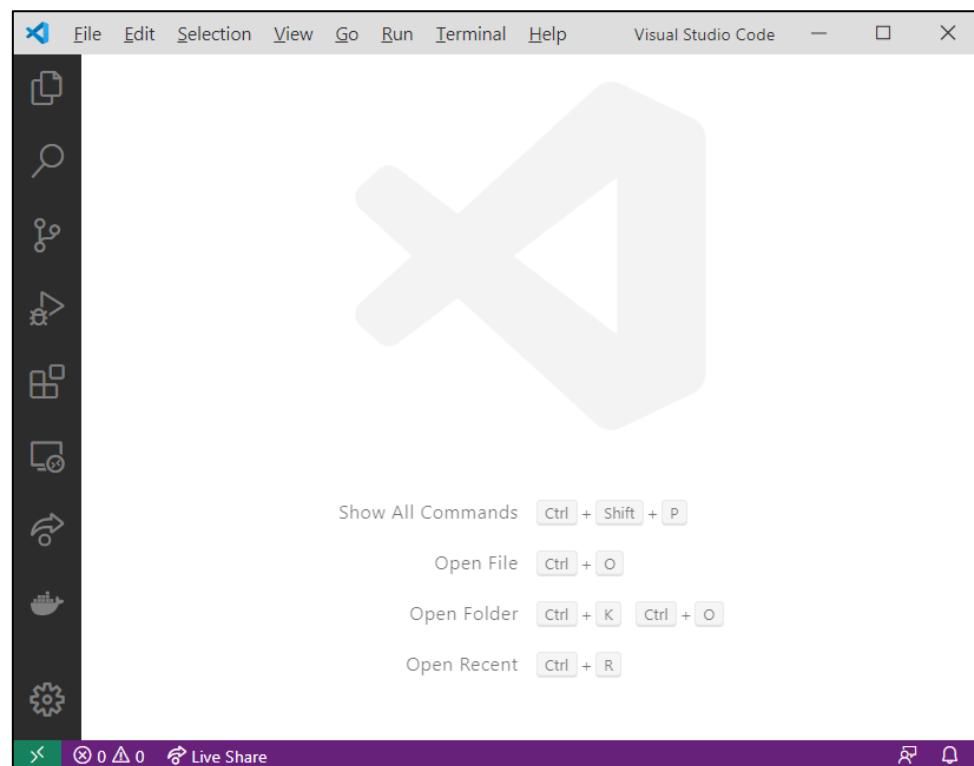
說明

本課程需要安裝的套件指令

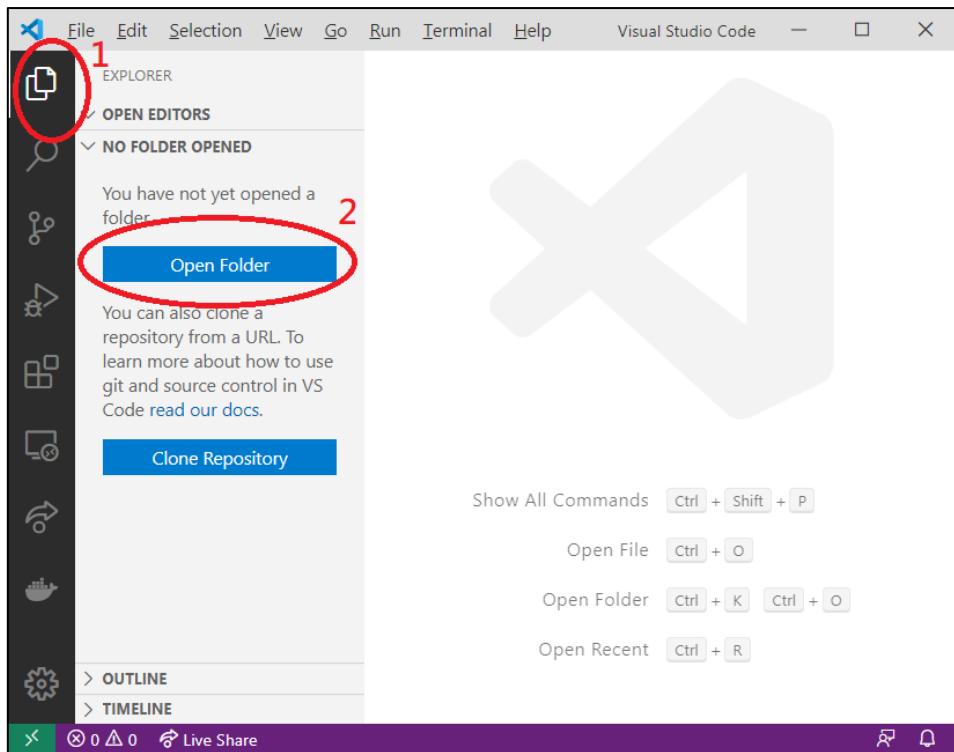
```
pip install selenium beautifulsoup4 requests pandas-datareader
mpl_finance browsermob-proxy
```

透過 Terminal 安裝套件

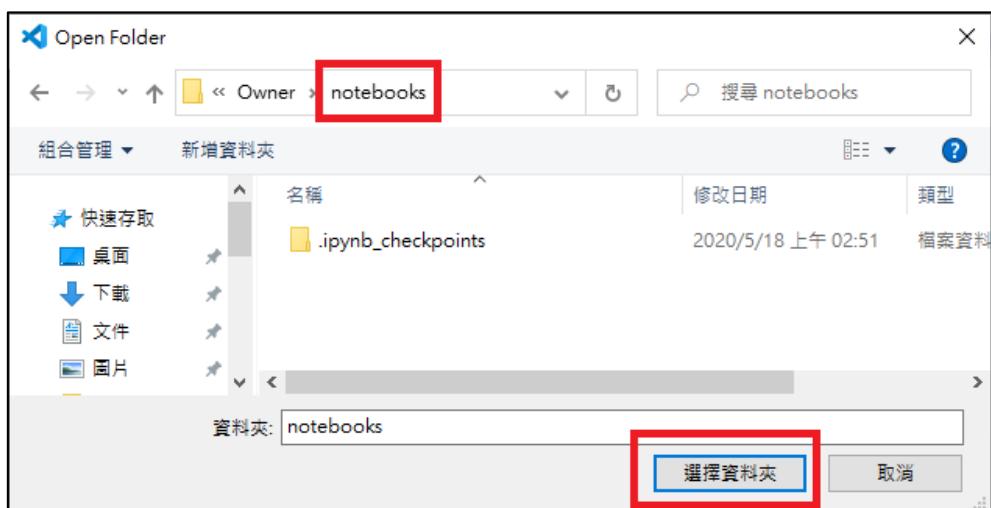
另一種作法，Visual Studio Code (以下簡稱 VS code) 裡面的 Terminal 安裝套件。



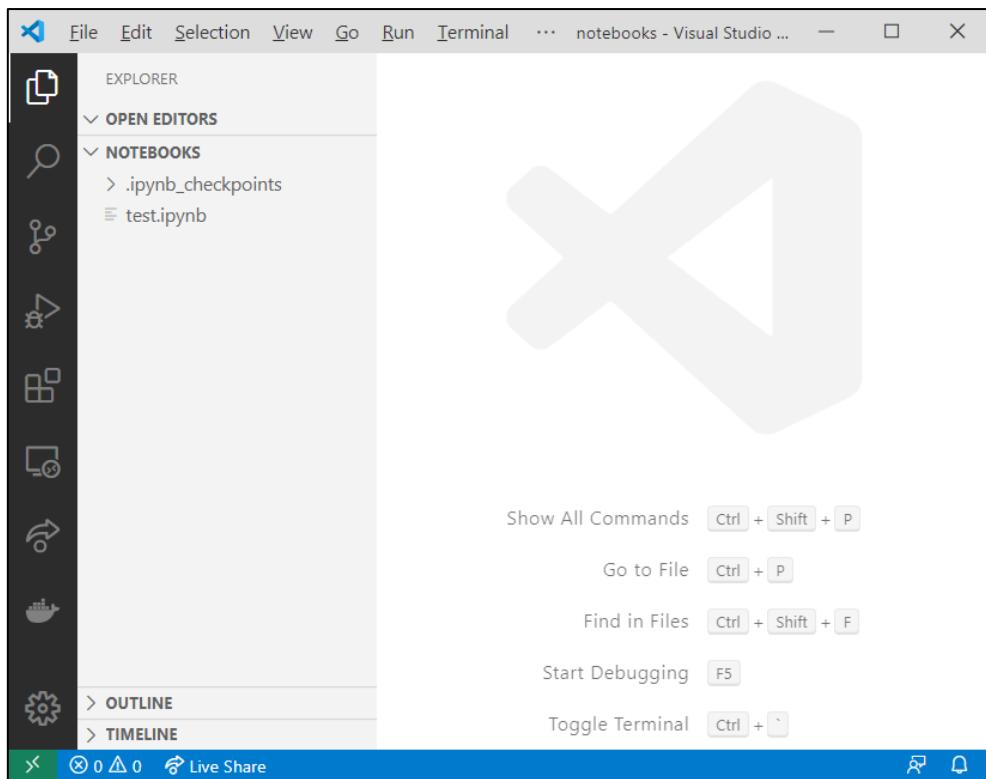
圖：開啟新的 Visual Studio Code



圖：按下開啟資料夾 (Open Folder)



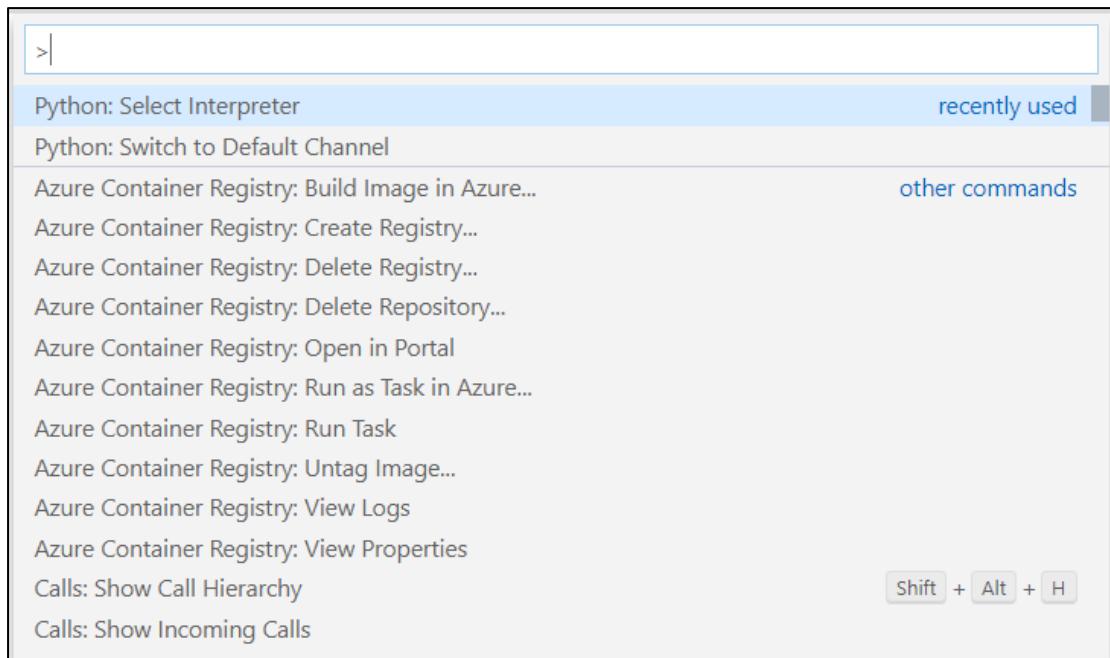
圖：選擇先前 notebooks 資料夾



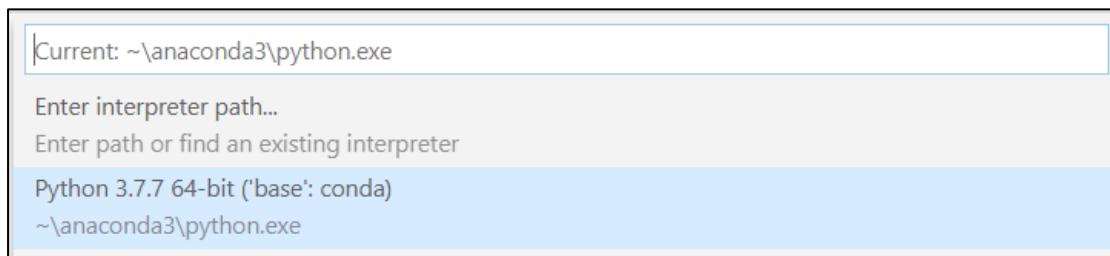
圖：編輯器設定完成

說明

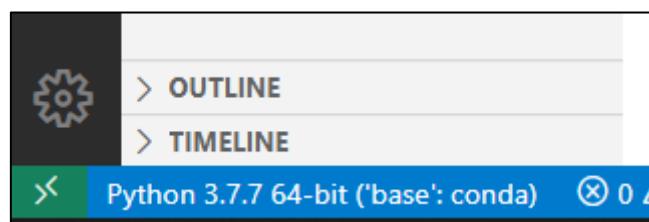
在 Visual Studio Code 輸入 Ctrl + Alt + P，會出現所有指令的列表，請選擇「Python: Select Interpreter」，再選擇「Python 3.7.7 64-bit ('base': conda)」。



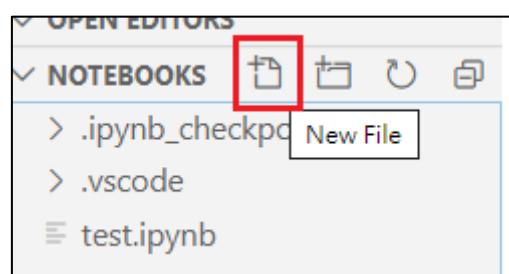
圖：開啟所有指令列表，選擇「Python: Select Interpreter」



圖：選擇先前建立的虛擬環境



圖：此時 Visual Studio Code 左下角



圖：開新檔案，檔案叫作 test.py

The screenshot shows the VS Code interface. On the left is the Explorer sidebar with sections for 'OPEN EDITORS' (containing 'test.py') and 'NOTEBOOKS' (listing '.ipynb_checkpoints', '.vscode', 'test.ipynb', and another 'test.py'). The main area is an editor window titled 'test.py' displaying the Python code: 'print('Hello World!')'.

圖：簡單、任意撰寫一個輸出程式碼，存成 test.py

說明
按下 Ctrl + ~ 來開啟 Terminal，原則上會自動進入 Conda 環境

The screenshot shows the VS Code terminal tab. The output shows the user navigating to their 'notebooks' directory, activating a Conda environment named 'base', and then running the command 'python -V' which returns 'Python 3.7.7'. This indicates the environment has been successfully activated.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Microsoft Windows [版本 10.0.18363.836]
(c) 2019 Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\Owner\notebooks>C:/Users/Owner/anaconda3/Scripts/activate
(base) C:\Users\Owner\notebooks>conda activate base
(base) C:\Users\Owner\notebooks>python -V
Python 3.7.7
(base) C:\Users\Owner\notebooks>
```

圖：開啟 Terminal 後，自動進入 Conda 環境，透過「python -V」確認

The screenshot shows a Jupyter Notebook interface. In the code editor, there is a single cell containing the Python code:

```
1 print('Hello World!')
```

Below the code editor are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is selected, displaying the following terminal session:

```
Microsoft Windows [版本 10.0.18363.836]
(c) 2019 Microsoft Corporation. 著作權所有，並保留一切權利。
C:\Users\Owner\notebooks>C:/Users/Owner/anaconda3/Scripts/activate
(base) C:\Users\Owner\notebooks>conda activate base
(base) C:\Users\Owner\notebooks>python test.py
Hello World!
(base) C:\Users\Owner\notebooks>
```

The line "Hello World!" is highlighted with a red rectangle.

圖：執行 test.py，輸出 Hello World，代表環境沒問題

說明

由於我們會用到 selenium，所以可以透過「**pip install selenium**」來安裝套件。

```
(base) C:\Users\Owner\notebooks>pip -V
pip 20.1.1 from c:\users\owner\anaconda3\lib\site-packages\pip (python 3.7)

(base) C:\Users\Owner\notebooks>pip install selenium
Collecting selenium
  Downloading selenium-3.141.0-py2.py3-none-any.whl (904 kB)
    |██████████| 904 kB 111 kB/s
Requirement already satisfied: urllib3 in c:\users\owner\anaconda3\lib\site-packages (from selenium) (1.25.8)
Installing collected packages: selenium
Successfully installed selenium-3.141.0
(base) C:\Users\Owner\notebooks>
```

圖：安裝 selenium 套件

下載 ChromeDriver – WebDriver for Chrome

網址：<https://chromedriver.chromium.org/>

操作 selenium 時，會需要啟動瀏覽器來協助我們進行自動化操作，在這裡我們使用 Chrome 作為自動化工具。要注意的地方，在於**下載的 ChromeDriver 要跟你所使用的電腦安裝 Chrome 版本一致**。



圖：確認版本

All versions available in Downloads

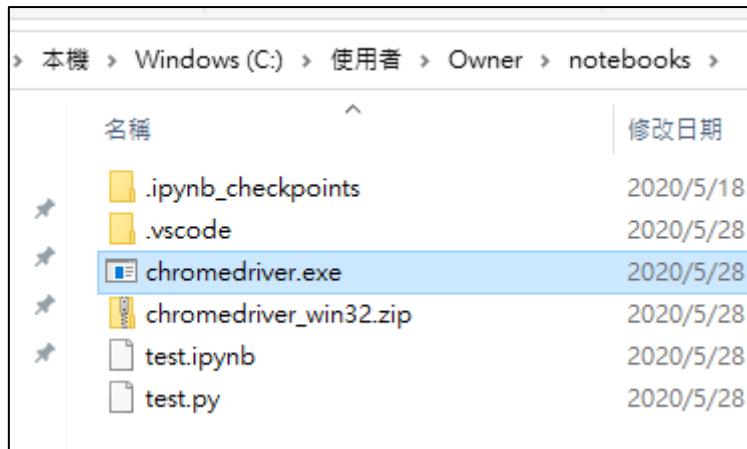
- Current stable release: [ChromeDriver 83.0.4103.39 \(we are skipping 82\)](#)
- Previous stable release: [ChromeDriver 81.0.4044.138](#)
- Beta release of ChromeDriver 84 coming soon

圖：選擇對應的版本，按下連結

Index of /83.0.4103.39/

Name	Last modified	Size	ETag
Parent Directory		-	
chromedriver_linux64.zip	2020-05-05 20:53:36	4.98MB	4e9d74f71a97470e59c1c0d311be49f6
chromedriver_mac64.zip	2020-05-05 20:53:38	6.87MB	640d3c63b3e8e7899f4a3aa6eebd22f4
chromedriver_win32.zip	2020-05-05 20:53:39	4.54MB	437630bbad9193f71af596bda21155ae
notes.txt	2020-05-05 20:53:43	0.00MB	6af5124d67e594649991106abd058e5b

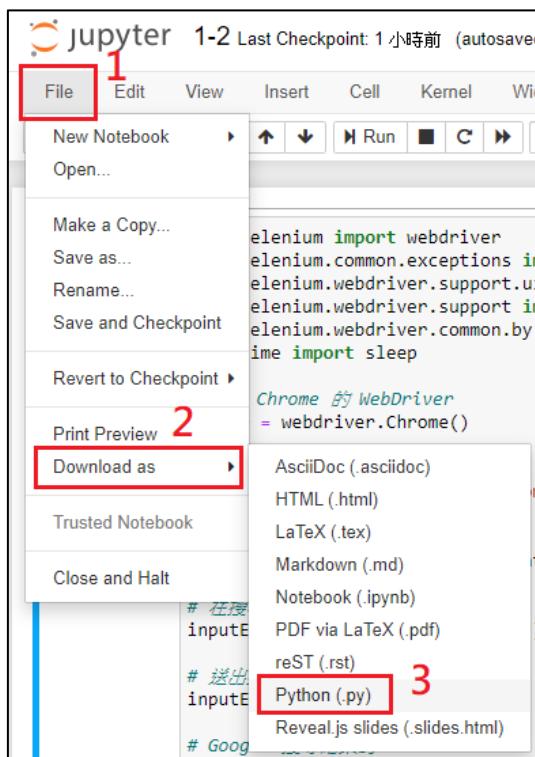
圖：下載到 notebooks 資料夾，並解壓縮



圖：解壓縮的 chromedriver.exe，要跟程式碼放在一起

測試 selenium

1. 使用 Jupyter：開啟 1-2.ipynb
2. 使用 VS code 的 Terminal：透過 1-2.ipynb 匯出 py 檔來執行



圖：匯出 py 檔到專案目錄 (notebooks) 下

備註

使用 Terminal 時，記得確認 Conda 環境有沒有成功啟用

1-3: 爬蟲專案開發實務分享

- 個人的 Web Scraping 播放清單 (不定期更新)

https://www.youtube.com/playlist?list=PLV4FeK54eNbyQhivk_2Mxw5Fz78HtlynG

- 「觀察」是爬蟲工作者的重要技能，HTML & CSS selector 的概念要非常熟練。
- 被對方的伺服器擋住，無法繼續爬取，可以透過 Amazon Web Service 的 EC2 (虛擬主機)，透過 Running、Stopped 等過程，提供自動換新的 IP 紿我們，或是連接 VPN，透過其它網路環境來進行；若是人在咖啡廳，臨時被擋，需要換 IP，可以切換到手機網路 (手機當作無線基地台)，開啟飛航模式，過個幾秒 (10 秒後) 再閉關飛航模式，此時網路服務供應商便會提供新的 IP 紿我們，便可繼續爬取資料。
- 建議：爬取資料時，每經過一個階段 (可能是網站換頁前後、網頁動態生成資料之間)，各給一個「隨機」的 sleep 時間，例如 1 到 3 秒，或是以自身經驗，在攻防之中，取得平衡，設定一組比較不會被擋的隨機數。

Module 2. 基礎回顧之資料結構

2-1: 字串物件操作

```
In [1]:  
# 初始化  
string01 = "1,2,3,4"  
print(string01)  
1,2,3,4  
  
In [2]:  
# 字串分割  
'''  
用法  
string.split()  
說明  
默認以空格、換行字元分割字串 s，返回 list  
'''  
list01 = string01.split(',')  
print(list01)  
['1', '2', '3', '4']  
  
In [3]:  
# 將 list 元素合併成字串  
'''  
用法  
string.join(seq)  
說明  
以 string 為分隔符，將 seq 中的元素串起來成為一個新的字串  
'''  
string02 = '-'.join(list01)  
print("將 list 元素合併成字串的結果: {}".format(string02))  
  
string02 = ''.join(list01)  
print("將 list 元素合併成字串的結果: {}".format(string02))  
將 list 元素合併成字串的結果: 1-2-3-4  
將 list 元素合併成字串的結果: 1234  
  
In [4]:  
# 搜尋字串  
string03 = 'believe'
```

```
result03 = string03.find('lie')
'''

用法
string.find(str)
說明
返回 s 第一次在字串 s 中出現的 index，若找不到則返回-1
'''

print(result03)
2
In [5]:
```

```
# 替換字串
'''

用法
string.replace(str1, str2)
說明
將 s 中的 str1 替換成 str2
'''

string04 = "Alex"
string04 = string04.replace('lex', 'llen')
print(string04)
Allen
In [6]:
```

```
# 字串變成小寫
'''

用法
string.lower()
說明
將字串 s 裡的字母全部改成小寫
'''

print("CAR".lower())
car
In [7]:
```

```
# 字串變成大寫
'''

用法
string.upper()
說明
將字串 s 裡的字母全部改成大寫
```

```
'''  
print("good".upper())  
GOOD  
In [8]:  
# 去除字串左邊空格  
'''  
  
用法  
string.lstrip()  
說明  
去除字串 s 左邊的空格  
'''  
  
string05 = "      ____ccc____"  
print("原先的字串: {}".format(string05))  
print("去除左邊空格後: {}".format(string05.lstrip()))  
原先的字串:      ____ccc____  
去除左邊空格後: ____ccc____  
In [9]:  
# 去除字串右邊空格  
'''  
  
用法  
string.rstrip()  
說明  
去除字串 s 右邊的空格  
'''  
  
string06 = "____ccc____"  
print("原先的字串: {}".format(string06))  
print("去除左邊空格後: {}".format(string06.rstrip()))  
原先的字串: ____ccc____  
去除左邊空格後: ____ccc____  
In [10]:  
# 去除字串兩側空格  
'''  
  
用法  
string.strip()  
說明  
去除字串 s 左、右兩邊的空格  
'''  
  
string07 = "      ____ccc____"  
'''
```

```
print("原先的字串: {}".format(string07))
print("去除左邊空格後: {}".format( string07.strip() ))
原先的字串:      ____ccc_____
去除左邊空格後: ____ccc_____
```

2-2: List 物件操作

In [1]:

```
# 初始化一個 list
ids = [1, 2, 3, 4, 5, 6]

print("目前 list 內容: {}".format(ids))
目前 list 內容: [1, 2, 3, 4, 5, 6]
```

In [2]:

```
# 新增元素在 list 尾端
ids.append(7)

print("目前 list 內容: {}".format(ids))
目前 list 內容: [1, 2, 3, 4, 5, 6, 7]
```

In [3]:

```
# 修改索引位置的元素
ids[4] = 99

print("目前 list 內容: {}".format(ids))
目前 list 內容: [1, 2, 3, 4, 99, 6, 7]
```

In [4]:

```
# 刪除索引位置的元素
ids.pop(4)

print("目前 list 內容: {}".format(ids))
目前 list 內容: [1, 2, 3, 4, 6, 7]
```

In [5]:

```
# 新增元素在指定索引，其餘元素往後移
ids.insert(1, 9)
```

'''

用法

```
list.insert(index, element)
```

說明

在指定的 `index` 處，新增 `element`，原先位置的索引往後移

'''

```
print("目前 list 內容: {}".format(ids))
```

```
目前 list 內容: [1, 9, 2, 3, 4, 6, 7]
```

In [6]:

```
# 移除指定的值
```

```
ids.remove(9)
```

```
print("目前 list 內容: {}".format(ids))
```

```
目前 list 內容: [1, 2, 3, 4, 6, 7]
```

序列物件分割 (Slicing)

- 序列物件包含 String、List 及 Tuple 等，我們可透過其順序取得元素
- 中括號為常見物件的索引值取值 (Indexing) 或分割之語法
 - 索引值取值指的是取出一個值
 - 分割指的是取出一部分值

索引值 :	0	1	2	3	4	5	6	7	8	9	10	11
	'H'	'e'	'l'	'l'	'o'	' '	'W'	'o'	'r'	'l'	'd'	'!'
索引值 :	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

In [7]:

```
# 初始化一個字串
```

```
myStr = 'Hello,World!'
```

In [8]:

'''

透過冒號 (:) 進行分割

'''

```
# 從頭取到 5 (不包含第 5 個元素)
```

```
print( myStr[:5] ) # 冒號前面留空
```

```
# 從 7 取到尾
```

```
print( myStr[7:] ) # 冒號後面留空
```

```
# 從 7 取到 9 (不包含第 9 個元素)
```

```
print( myStr[7:9] )
```

```
Hello
```

```

orld!
or
In [9]:
'''

使用負號

'''

# 從倒數第 10 取到倒數第 7 (不包含倒數第 7 的元素)
print( myStr[-10:-7] )

# 從倒數第 5 取到尾
print( myStr[-5:] )

# 從頭取到倒數第 7 (不包含倒數第 7 的元素)
print( myStr[:7] )
Hello
orld!
Hello

```

2-3: Dict 物件操作

```

In [1]:
# 初始化 dict {水果名稱: 價格}
dict01 = {"蘋果": 100, "橘子": 20, "水梨": 50}
print(dict01)
{'蘋果': 100, '橘子': 20, '水梨': 50}

```

```

In [2]:
# 印出蘋果的價格
print(dict01["蘋果"])
100

```

```

In [3]:
# 修改橘子的價格
dict01["橘子"] = 30
print(dict01["橘子"])
30

```

```

In [4]:
# 刪除 水梨
del dict01["水梨"]
print(dict01)

```

```
{'蘋果': 100, '橘子': 30}
```

In [5]:

'''格式化成字串來輸出'''

```
# 1. 將所有 keys 轉成 list
```

```
list_keys = list(dict01.keys())
```

```
# 2. 將所有 values 轉成 list
```

```
list_values = list(dict01.values())
```

```
# 3. 使用迴圈來輸出格式化字串
```

```
for i in range(len(list_keys)):
```

```
    print("%s 的價格是 %d 元"%( list_keys[i], list_values[i] ))
```

蘋果 的價格是 100 元

橘子 的價格是 30 元

Module 3. 基礎回顧之流程控制與迴圈

3-1: 流程控制 (if...elif...else)

In [1]:

```
# if 敘述  
num = 10
```

```
if num > 5:  
    print("num 大於 5")  
num 大於 5
```

In [2]:

```
# if else 敘述  
name = 'apple'  
  
if name == 'apple':  
    print('名稱是 apple')  
else:  
    print("名稱不是 apple")  
名稱是 apple
```

In [3]:

```
# if elif else  
name = 'darren'  
  
if name == "alex":  
    print("名稱: alex")  
elif name == "bill":  
    print("名稱: bill")  
elif name == "carl":  
    print("名稱: carl")  
elif name == "darren":  
    print("名稱: darren")  
else:  
    print("Not found")  
名稱: darren
```

3-2: 迴圈 (for / while)

In [1]:

```
# while 迴圈
count = 1
while count <= 5:
    print(count, end="")
    count = count + 1
12345
```

In [2]:

```
# for 迴圈 01
'''
用法
range(n, m)
說明
會走訪 n 到 m-1 的數字
'''
for i in range(5, 8):
    print(i, end = ", ")
5, 6, 7,
```

In [3]:

```
# for 迴圈 02
'''
用法
range(n, m, step)
說明
以每step為間距，走訪 n 到 m-1 的數字
'''
for i in range(5, 20, 2):
    print(i, end = ", ")
5, 7, 9, 11, 13, 15, 17, 19,
```

3-3: 關鍵字與綜合使用 (continue / break)

In [1]:

```
# break
'''
```

說明

當偵測到字母 t 時，就會強制結束迴圈

```
'''  
for string in 'content':  
    if string == 't':  
        break  
    print(string, end="")  
con
```

In [2]:

```
# continue
```

說明

當偵測到字母 t 時，

會跳過本次迴圈剩下的程式碼 `print(string)`，

但不會結束迴圈，仍然會進入下一圈繼續執行

```
'''  
for string in 'content':  
    if string == 't':  
        continue  
    print(string, end="")  
conen
```

In [3]:

```
# pass
```

說明

當偵測到字母 t 時，會忽略該條件，繼續像正常迴圈一樣運行程序

備註

有時候寫 `pass`，是為了將某塊或某行列入 `to-do`

```
'''  
for string in 'content':  
    if string == 't':  
        pass  
    print(string, end="")  
content
```

3-3-1 補充: Tuple 的用法

補充: Tuple 的用法

特點

- 有序號、索引概念
- 允許重覆
- 不可更改組的資料
- 使用 for 迴圈讀出一筆筆的資料

In [1]:

```
# Tuple 初始化: 第一種
myTuple01 = ("人", "帥", "得體")
print(myTuple01)
```

```
# Tuple 初始化: 第二種
myTuple02 = "哆", "啦", "A", "夢"
print(myTuple02)
('人', '帥', '得體')
('哆', '啦', 'A', '夢')
```

In [2]:

```
# 透過指定索引輸出值
print( myTuple01[1] )
print( myTuple02[1] )
帥
啦
```

In [3]:

```
# 複數變數修改值
a = 10
b = 20
print("交換前: a = {}, b = {}".format(a, b))
a, b = b, a
print("交換後: a = {}, b = {}".format(a, b))
交換前: a = 10, b = 20
交換後: a = 20, b = 10
```

In [4]:

```
# list 可以修改指定索引的值
```

```
myList = ["人", "帥", "任性"]
myList[2] = "真好"
print(myList)

# tuple 不可以修改指定索引的值
myTuple = ("人", "帥", "任性")
myTuple[2] = "真好"
print(myTuple)
['人', '帥', '真好']

-----
-----
TypeError                                     Traceback (most recen
t call last)
<ipython-input-4-7703a10a6556> in <module>
      6 # tuple 不可以修改指定索引的值
      7 myTuple = ("人", "帥", "任性")
----> 8 myTuple[2] = "真好"
      9 print(myTuple)

TypeError: 'tuple' object does not support item assignment
In [5]:
# 用 for 迴圈逐一輸出資料
for value in myTuple01:
    print(value)
人
帥
得體
In [6]:
# 用 len 計算 tuple 資料個數
print( len(myTuple02) )
4
In [7]:
# 因為無法修改 tuple 的資料，所以也無法指定索引進行刪除
del myTuple01[1]
print(myTuple01)
```

```

TypeError                                         Traceback (most recen
t call last)
<ipython-input-7-edd2587ce8f4> in <module>
      1 # 因為無法修改 tuple 的資料，所以也無法指定索引進行刪除
----> 2 del myTuple01[1]
      3 print(myTuple01)

TypeError: 'tuple' object doesn't support item deletion
In [8]:
# 只能從記憶體刪除整個 tuple 變數
del myTuple01
print(myTuple01)
-----
-----
NameError                                         Traceback (most recen
t call last)
<ipython-input-8-fbef4f7af234> in <module>
      1 # 只能從記憶體刪除整個 tuple 變數
      2 del myTuple01
----> 3 print(myTuple01)

NameError: name 'myTuple01' is not defined

```

3-3-2 補充: Set 的用法

補充: Set 的用法

特點

- 無序號、索引概念
- 無索引(隨機出現)
- 沒有重覆(一樣的只會出現一次)

```

In [1]:
# Set 初始化: 第一種
mySet01 = {"網路", "爬蟲", "真好玩"}

```

```

print(mySet01)

# Set 初始化: 第二種 (裡面放 tuple)
mySet02 = set( ("網路", "爬蟲", "真好玩") )

print(mySet02)
{'真好玩', '爬蟲', '網路'}
{'真好玩', '爬蟲', '網路'}

In [2]:
# 一筆筆讀資料 : for 迴圈
for data in mySet02:
    print(data)
真好玩
爬蟲
網路

In [3]:
# 總共有幾筆資料: len()
print( len(mySet02) )
3

In [4]:
# 因為沒有序號、索引的概念，所以無法透過指定索引輸出
print( mySet01[0] )
-----
-----
-----
TypeError                                     Traceback (most recent call last)
<ipython-input-4-a7aab546c07f> in <module>
      1 # 因為沒有序號、索引的概念，所以無法透過指定索引輸出
      2 print( mySet01[0] )

TypeError: 'set' object is not subscriptable

In [5]:
# 添加一筆資料: add()
mySet01.add("嗎?")
mySet02.add("對不對?")

print(mySet01)
print(mySet02)

```

```

# 此時新增重複的，資料不會增加
mySet01.add("真好玩")
print(mySet01)
{'真好玩', '爬蟲', '網路', '嗎?'}
{'對不對?', '真好玩', '爬蟲', '網路'}
{'真好玩', '爬蟲', '網路', '嗎?'}

In [6]:

# 添加多筆資料: update()
mySet01.update(["甲", "乙", "丙"])
mySet02.update(["子", "丑", "寅"])

print(mySet01)
print(mySet02)
{'丙', '爬蟲', '乙', '嗎?', '真好玩', '網路', '甲'}
{'爬蟲', '真好玩', '丑', '寅', '子', '網路', '對不對?'}

In [7]:

# 查詢陣列中有沒有我要的資料: in (只回傳 true 或 false)
print("甲" in mySet01)
print("甲" in mySet02)

if "乙" in mySet01:
    print("有資料")
else:
    print("找不到資料")
True
False
有資料

In [8]:

# 刪除元素: discard() or remove()
mySet01.discard("丙")
print(mySet01)

mySet02.remove("丑")
print(mySet02)
{'爬蟲', '乙', '嗎?', '真好玩', '網路', '甲'}
{'爬蟲', '真好玩', '寅', '子', '網路', '對不對?'}

In [9]:

```

```

# 清空: clear()、del
''''.clear() 會清空 set 變數，但變數依然存在，所以會印出空 set'''
mySet01.clear()
print(mySet01)

''' del 會將 set 變數從記憶體中刪除，所以刪除完後，變數會變成未宣告
的狀態'''
del mySet02
print(mySet02)
set()

-----
-----
NameError                                         Traceback (most recent
t call last)
<ipython-input-9-b6276d9e9fa5> in <module>
6 ''' del 會將 set 變數從記憶體中刪除，所以刪除完後，變數會變
成未宣告的狀態'''
7 del mySet02
----> 8 print(mySet02)

NameError: name 'mySet02' is not defined

```

3-3-3 練習: 將 list 當中重複的 dict 去除，並透過指定 dict key 來排序

```

In [1]:
import pprint

'''

流程 1
'''


# 假設我們有 3 個 dict，每個 dict 都是 LINE 官方貼圖(靜態圖片，無
動畫、無聲音)
dict01 = {

```

```

    "link": "https://stickershop.line-scdn.net/stickershop/v1/sticker/380512238/android/sticker.png",
    "id": "380512238"
}

dict02 = {
    "link": "https://stickershop.line-scdn.net/stickershop/v1/sticker/380512238/android/sticker.png",
    "id": "380512238"
}

dict03 = {
    "link": "https://stickershop.line-scdn.net/stickershop/v1/sticker/380512239/android/sticker.png",
    "id": "380512239"
}

# 接下來，我們把這三個 dict，都放到一個 list 當中
listLineStickers = []
listLineStickers.append(dict01)
listLineStickers.append(dict02)
listLineStickers.append(dict03)

# 檢視一下當前內容
pprint.pprint(listLineStickers)
[{'id': '380512238',
 'link': 'https://stickershop.line-scdn.net/stickershop/v1/sticker/380512238/android/sticker.png'},
 {'id': '380512238',
 'link': 'https://stickershop.line-scdn.net/stickershop/v1/sticker/380512238/android/sticker.png'},
 {'id': '380512239',
 'link': 'https://stickershop.line-scdn.net/stickershop/v1/sticker/380512239/android/sticker.png'}]

```

In [2]:

```

...
流程 2
...

```

```
# 建立一個 set 物件，準備 add 所有 tuple，這些 tuple 裡面都有 dict_items 物件
_set = set()
```

```
'''
```

一、dict.items()

說明：

items() 方法把字典中每一對 key 和 value 組成一個 tuple
例如：

```
dict_items([
    ('link', 'https://stickershop.line-scdn.net/stickershop/v1/sticker/318800558/android/sticker.png'),
    ('id', '318800558')
])
```

二、tuple(dict.items())

說明：

1. 將 dict_items 格式轉成 tuple，目前是為了「讓 set 可以使用 .add() 方法，來去除重複」。
2. 之所以要將轉換格式，是因為 tuple 可以被新增到 set 當中，dict 和 dict_items 不行。
3. tuple 是可以雜湊的(hashable)，可雜湊代表「雜湊值不可變動」，不可變動才能拿來判斷是否相同或比較 (equal or compare)。
4. 可變動的資料型態，例如 list 可以 append()、remove()，或是像 dict 等透過指定 key 來新增修改、刪除資料的格式。

例如：

```
([
    ('link', 'https://stickershop.line-scdn.net/stickershop/v1/sticker/318800558/android/sticker.png'),
    ('id', '318800558')
])
```

```
# 將放置 LINE 貼圖的 dict 各別轉換成為 dict_items 物件，再各別轉換成 tuple，最後新增到 Set 當中
```

```
for dictLineSticker in listLineStickers:
    dict_items = dictLineSticker.items()
```

```
_tuple = tuple(dict_items)
_set.add(_tuple)

pprint.pprint(_set)
{('link',
 'https://stickershop.line-scdn.net/stickershop/v1/sticker/380512238/android/sticker.png'),
 ('id', '380512238')),
 (('link',
 'https://stickershop.line-scdn.net/stickershop/v1/sticker/380512239/android/sticker.png'),
 ('id', '380512239'))}
```

In [3]:

'''

流程 3

'''

```
# 新增 list，準備將去掉重複的 dict 資料各別 append 進去
listResult = []
```

'''

三、dict(t)

說明：

原先的 `tuple(dict.items())` 的結果，透過 `dict()` 轉型，變成原先 dict 的 key-value 格式

例如：

```
{
    'id': '318800558',
    'link': 'https://stickershop.line-scdn.net/stickershop/v1/sticker/318800558/android/sticker.png'
}
```

'''

```
# 此時 set 應該已經去除重複的 tuple，此時將 tuple 各別轉回原本的 dict，並寫入新的 list 當中
```

```
for _tuple in _set:
    dictLineSticker = dict(_tuple)
    listResult.append(dictLineSticker)
```

```
pprint.pprint(listResult)
[{'id': '380512239',
 'link': 'https://stickershop.line-scdn.net/stickershop/v
1/sticker/380512239/android/sticker.png'},
 {'id': '380512238',
 'link': 'https://stickershop.line-scdn.net/stickershop/v
1/sticker/380512238/android/sticker.png'}]
```

In [4]:

```
'''
```

流程 4

```
'''
```

```
# 使用 sorted，並指定每個 dict 當中的 id 索引進行排序
listResult = sorted(listResult, key=lambda myDict: myDict
['id'], reverse=False)
```

```
pprint.pprint(listResult)
[{'id': '380512238',
 'link': 'https://stickershop.line-scdn.net/stickershop/v
1/sticker/380512238/android/sticker.png'},
 {'id': '380512239',
 'link': 'https://stickershop.line-scdn.net/stickershop/v
1/sticker/380512239/android/sticker.png'}]
```

Module 4. 關於 URL 與字串格式化

4-1: 解析 URL (urlparse) 與產生 URL

In [10]:

```
# 汇入套件
from urllib import parse

# urlparse
url = 'https://docs.python.org/3.7/search.html?q=parse&check_keywords=yes&area=default'
parseResult = parse.urlparse(url)
print("分析 URLs 成為 components:\n{}".format(parseResult))
分析 URLs 成為 components:
ParseResult(scheme='https', netloc='docs.python.org', path='/3.7/search.html', params='', query='q=parse&check_keywords=yes&area=default', fragment='')
```

In [11]:

```
# 印出 scheme
print(parseResult.scheme)
https
```

In [12]:

```
# 印出 query
print(parseResult.query)
q=parse&check_keywords=yes&area=default
```

In [13]:

Attribute	Index	Value	Value if not present
scheme	0	URL scheme specifier	<code>scheme</code> parameter
netloc	1	Network location part	empty string
path	2	Hierarchical path	empty string
params	3	Parameters for last path element	empty string
query	4	Query component	empty string
fragment	5	Fragment identifier	empty string
username		User name	<code>None</code>
password		Password	<code>None</code>
hostname		Host name (lower case)	<code>None</code>
port		Port number as integer, if present	<code>None</code>

圖：命名元組 (named tuple)

```
# 產生 url
string = 'https://www.104.com.tw/jobs/search/?'
query = {
    "ro": 1,
    "kwop": 7,
    "keyword": "python",
    "order": 13,
    "asc": 0,
    "page": 1,
    "mode": "s",
    "jobsource": "2018indexpoc"}
result = string + parse.urlencode(query)
print(result)
https://www.104.com.tw/jobs/search/?ro=1&kwop=7&keyword=p
ython&order=13&asc=0&page=1&mode=s&jobsource=2018indexpoc
```

4-2: 如何使用 URL 編碼 (quote / unquote)

```
In [ ]:
# 汇入套件
from urllib import parse
'''

說明
url 只允許部分 ASCII 的字元（數字與部分符號），
其它的字元（例如中文字）是不符合 url 標準的，
這時候 url 若有其它字元
便要進行編碼
'''
```

```
In [2]:
# quote
result01 = parse.quote('a=1&b=2')
print(result01)
a%3D1%26b%3D2
```

```
In [3]:
```

```
# unquote
result02 = parse.unquote('a%3D1%26b%3D2')
print(result02)
a=1&b=2
```

4-3: 字串格式化

In [1]:

```
# 百分比(%)
print('%d' % 20) # 格式化整數
print('%f' % 1.11) # 預設保留 6 位小數
print('.1f' % 1.11) # 取 1 位小數
print('My name is %s' % 'Darren') # 格式化字串
20
1.110000
1.1
My name is Darren
```

In [2]:

```
# str.format => 作法: '{}'.format()
name = "Darren"
age = 18
str01 = "My name is {} and my age is {}".format(name, age)
print(str01)
My name is Darren and my age is 18
```

In [3]:

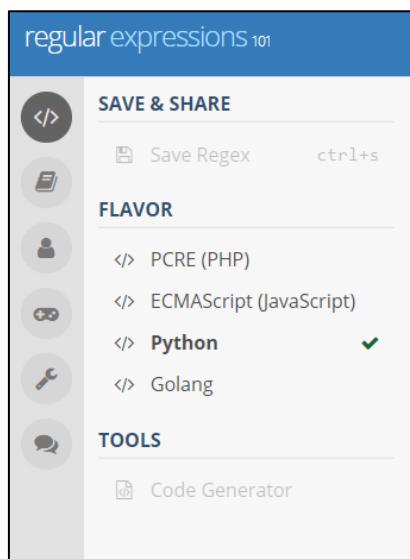
```
# f-string(又作 formatted string literals, version >= 3.6)
name = "Darren"
age = 18
str02 = f"My name is {name} and my age is {age}"
print(str02)
My name is Darren and my age is 18
```

In [4]:

```
# 樣板字串(Template String)
from string import Template
tpl = Template('$who likes $what')
print(tpl.substitute(who='Darren', what='Python'))
Darren likes Python
```

Module 5. 正規表達式 (Regular Expression) 入門

正規表達式 (Regular Expression) 是用來配對、過濾、替換文字的一種表示法。請先進入「<https://regex101.com/>」頁面，我們之後測試正規表達式，都會透過這個網頁的功能。正規表達式是需要大量練習才能了解的知識，希望大家都能透過頻繁地練習，慢慢感受到正規表達式在文字處理上的便捷。



圖：選擇 FLAVOR 為 Python

A screenshot of the regex101.com main interface. In the 'REGULAR EXPRESSION' field, there is a pattern: ^[a-zA-Z0-9_]+@[a-zA-Z0-9_.]+\..+gm. The 'TEST STRING' field contains 'telunyang@gmail.com'. The 'EXPLANATION' panel shows the breakdown of the regex: it matches a single character from the list [a-zA-Z0-9_] one or more times, followed by '@', another single character from the list [a-zA-Z0-9_.] one or more times, followed by '.', and finally one or more characters. The 'MATCH INFORMATION' panel shows 'Match 1' with the full match 'telunyang@gmail.com'. The 'QUICK REFERENCE' panel provides links to common regex concepts like 'Single character', 'Character except', and 'Character range'.

圖：使用正規表達式，來判斷字串是否符合文字格式或條件

5-1: 指示符號介紹 (^ / \$ / \d / \w / [?-?] / ...)

5-2: 計數符號介紹 ({?} / {?,?} / + / * / ? / ...)

下面表格為快速參考的範例：

說明	正規表達式	範例
一個字元: a, b or c	[abc]	abcdef
一個字元 · 除了: a, b or c	[^abc]	abcde f
一個字元 · 在某個範圍內: a-z	[a-z]	abcd0123
一個字元 · 不在某個範圍內: a-z	[^a-z]	abcd0123
一個字元 · 在某個範圍內: a-z or A-Z	[a-zA-Z]	abcdXYZ0123
避開特殊字元	\ ex. \?	?
任何單一字元	.	任何字元
任何空白字元 (\f\r\n\t\v)	\s	空格、換行、換頁等
任何非空白字元 (不是 \f\r\n\t\v)	\S	非空格、非換行、非換頁等
任何數字	\d	10ab
任何非數字	\D	10ab
任何文字字元	\w	10ab*AZ\\$
任何非文字字元	\W	10ab*AZ\\$
以群組的方式配對 · 同時捕捉被配對的資料	(...) ex. (1[0-9]{3} 20[0-9]{2})	1992, 2019, 1789, 1776, 1024, 3000, 4096, 8192
配對 a 或 b	a b	addbeeeeaccbaa
0 個或 1 個 a	a?	addbeeeeaccbaa
0 個或更多的 a	a*	addbeeeeaccbaa
1 個或更多的 a	a+	aaa, aaaaa
完整 3 個 a	a{3}	aaa, aaaaa
3 個以上的 a	a{3,}	aa, aaa, aaaaa
3 個到 6 個之間的 a	a{3,6}	aaa, aaaaaa, aaaa, aaaaaaaa
字串的開始	^ ex. ^Darren	^ DarrenYang
字串的結束	\$ ex. Yang\$	DarrenYang\$
位於邊界的字元	\b ex. \bD	DarrenYang
非位於邊界的字元	\B ex. \Ba	DarrenYang

說明	正規表達式	範例
配對卻不在群組裡顯示	John(?:Cena)	John Cena
正向環視 (這位置右邊要出現什麼)	John(?=Cena)	John Cena
正向環視否定 (這位置右邊不能出現什麼)	Johnnie(?!Cena)	Johnnie Walker
反向環視 (這位置左邊要出現什麼)	(?<=Johnnie) Walker	Johnnie Walker
反向環視否定 (這位置左邊不能出現什麼)	(?<!John) Walker	Johnnie Walker

5-3: 常用範例推導 (手機號碼表達式、身分證字號表達式、...)

用途	正規表達式	範例
E-mail	[a-zA-Z0-9_]+@[a-zA-Z0-9\._]+	darren@darreninfo.cc telunyang@gmail.com
英文名字	[a-zA-Z]+	Darren Alex
身分證	[a-zA-Z](1 2)[0-9]{8}	A123456789
網址	https?:\/\/[0-9a-zA-Z_-]+(\.[a-zA-Z0-9]+)+(\/?[0-9a-zA-Z_-]+)+=\d+	https://darreninfo.cc/?page_id=10
實數與小數	[0-9]{1,4}\.[0-9]+	9487.94
手機	(?:0 886-?)9\d{2}-?\d{6}	0912345678
市內電話	0[2-8-]+[0-7]+[0-9]	02-66316666

Module 6. 正規表達式 (Regular Expression) 進階

6-1: 具名群組 (Named Group) 介紹

與一般的群組差異，在於提供群組一個名稱，而非以 Group 1、Group 2 的概念存在。

MATCH INFORMATION		
Match 1		
Full match	0-32	abcd{111:333,444:555}cdefg123456
Group 1.	5-20	111:333,444:555
Group 2.	26-32	123456

圖：原先的群組配對結果

- 格式：

`(?P<value> regex_string)`

- 正規表達式範例：

`[a-zA-Z]+\{(?P<value>.*?)\}[a-zA-Z]+(?P<number>[0-9]+)`

- 字串範例：

`abcd{111:333,444:555}cdefg123456`

REGULAR EXPRESSION

```
^r" [a-zA-Z]+ \{ (?P<value>.*?) \} [a-zA-Z]+ (?P<number>[0-9]+)
```

TEST STRING

```
abcd{111:333,444:555}cdefg123456
```

圖：建立了 value 與 number 兩個具名群組

MATCH INFORMATION

Match 1

Full match	0-32	abcd{111:333,444:555}cdefg123456
Group `value`	5-20	111:333,444:555
Group `number`	26-32	123456

圖：原先的 Group 1、Group 2，擁有了名稱

6-2: 常用函數介紹 (search / findall / match / group / split / ...)

```
In [4]:
# 汇入 regex 套件
import re
In [9]:
# search
regex01 = r'[a-zA-Z]{1|2}\d{8}'
string01 = "A123456789, S299888777"
match01 = re.search(regex01, string01)
'''

說明
re.search 會將整個字串進行搜尋，  

但只會比對到第一組，
```

```

match[0]是 regex 所代表的整個完整比對的字串 ,
match[1]是第一組()中的內容 ,
match[2]是第二組()中的內容...
'''

print(match01)
print(match01[0])
print(match01[1])
<re.Match object; span=(0, 10), match='A123456789'>
A123456789
1
In [10]:


match.group() 或 match.group(0) 是 regex 所代表的整個完整比對的
字串 ,
match.group(1)是第一組()中的內容 ,
match.group(2)是第二組()中的內容...
'''

print(match01.group(0))
print(match01.group(1))
A123456789
1
In [11]:


#.findall
regex02 = r'[0-9]+'
string02 = "0911111111, 0922222222, 0933333333"
listMatch02 = re.findall(regex02, string02)
'''

說明
re.findall 會將所有配對到的字串
回傳成一個 list
'''

print(listMatch02)
print(listMatch02[0])
print(listMatch02[2])
['0911111111', '0922222222', '0933333333']
0911111111
0933333333
In [17]:

```

```
# match
regex03 = r'2[0-9]{3}\/[0-1]?[0-9]{1}\/([1-3]?[0-9])'
string03 = "2020/06/10"
match03 = re.match(regex03, string03)
'''

說明
re.match 與 re.search 的差別，  

在於 match 會從字串的「開頭」開始比對，  

比對不到，便回傳 None
'''

print(match03)
print(match03[0])
print(match03[1])
<re.Match object; span=(0, 10), match='2020/06/10'>
2020/06/10
10
```

In [18]:

```
'''  
使用 match.group(n) 的格式
'''
```

```
print(match03.group())
print(match03.group(1))
2020/06/10
10
```

In [19]:

```
# group
regex04 = r'(1[0-9]{3}),?\s(2[0-9]{3})'
string04 = "1992, 2019"
match04 = re.match(regex04, string04)
'''
```

說明

`match.group()` 會輸出完整比對到的字串
`match.group(n)` 輸出第 n 個()比對到的內容
'''

```
print(match04.group())
print(match04.group(1))
print(match04.group(2))
1992, 2019
```

1992

2019

In [20]:

```
# split
regex05 = r'\d'
string05 = "One1Two2Three3Four4"
listMatch05 = re.split(regex05, string05)
'''
說明
re.split 類似 string.split('separator') ,
只是用正規表達式來作為 separator ,
並回傳 list
'''
print(listMatch05)
['One', 'Two', 'Three', 'Four', '']
```

6-3: 具名群組在常用範例上的使用方式 (Email、URL、...)

In [3]:

```
# 汇入套件
import re
```

In [4]:

```
# named group
regex01 = r'[a-zA-Z]+\{(?:<value>.*?)\}[a-zA-Z]+(?:<number>[0-9]+)'
string01 = "abcd{111:333,444:555}cdefg123456"
match01 = re.match(regex01, string01)
'''
說明
除了 .group(n) 以外 ,
還可以做用 name ,
作為 key 來存取 group()
'''
print(match01.group())
print(match01.group('value'))
print(match01.group('number'))
print(match01['value'])
```

```
print(match01['number'])
print(match01[1])
print(match01[2])
abcd{111:333,444:555}cdefg123456
111:333,444:555
123456
111:333,444:555
123456
111:333,444:555
123456
```

In [5]:

```
# E-mail
'''

說明
透過 named group 取得 username、domain name
'''

regex02 = r'(?P<username>[a-zA-Z0-9]+)@(?P<domain_name>[0-
9A-Za-z]+\.[0-9A-Za-z]+)'
string02 = "telunyang@gmail.com"
match02 = re.match(regex02, string02)
print(match02.group())
print(match02.group('username'))
print(match02.group('domain_name'))
telunyang@gmail.com
telunyang
gmail.com
```

In [6]:

```
# url
regex03 = r'https?:\/\/(?P<domain_name>[a-zA-Z.]+)\/(?P<path>
[a-zA-Z]+)\/'
string03 = "https://www.iiiedu.org.tw/location/"
match03 = re.match(regex03, string03)
print(match03.group())
print(match03.group('domain_name'))
print(match03.group('path'))
https://www.iiiedu.org.tw/location/
www.iiiedu.org.tw
location
```

Module7. HTML 基礎與 HTTP 方法

7-1: 常見 HTML 架構介紹

HTML 簡介

HTML (Hyper Text Markup Language , 超文字標記語言) 是用來產生 Web 網頁的語言。HTML 裡面的標籤 (tags , 例如 ul 、 li 、 a 、 p 、 div 等 , 以及 HTML5 增加的 section 、 article 、 aside 、 nav 、 footer 等) , 這些標籤會告訴瀏覽器何時顯示、顯示什麼、如何顯示。

建立 HTML 檔案

範例

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no" />
    <title>歡迎來到我的家</title>
  </head>
  <body>
    <div id="wrapper">
      <!-- nav 在這裡是網頁上緣導覽列 -->
      <header class="head-info">
        <nav class="nav nav-info">
          <ul class="nav-body">
            <li class="nav-list">
              <a class="center link-custom">首頁</a>
            </li>
            <li class="nav-list">
              <a class="center link-custom">連結 1</a>
            </li>
            <li class="nav-list">
```

```

        <a class="center link-custom">連結 2</a>
    </li>
    <li class="nav-list">
        <a class="center link-custom">連結 3</a>
    </li>
</ul>
</nav>
</header>

<!-- aside 在這裡是左側的選單列表 -->
<aside class="menu">
    <ul class="menu-body">
        <li class="menu-list">
            <a class="center">側欄連結 1</a>
        </li>
        <li class="menu-list">
            <a class="center">側欄連結 2</a>
        </li>
        <li class="menu-list">
            <a class="center">側欄連結 3</a>
        </li>
    </ul>
</aside>

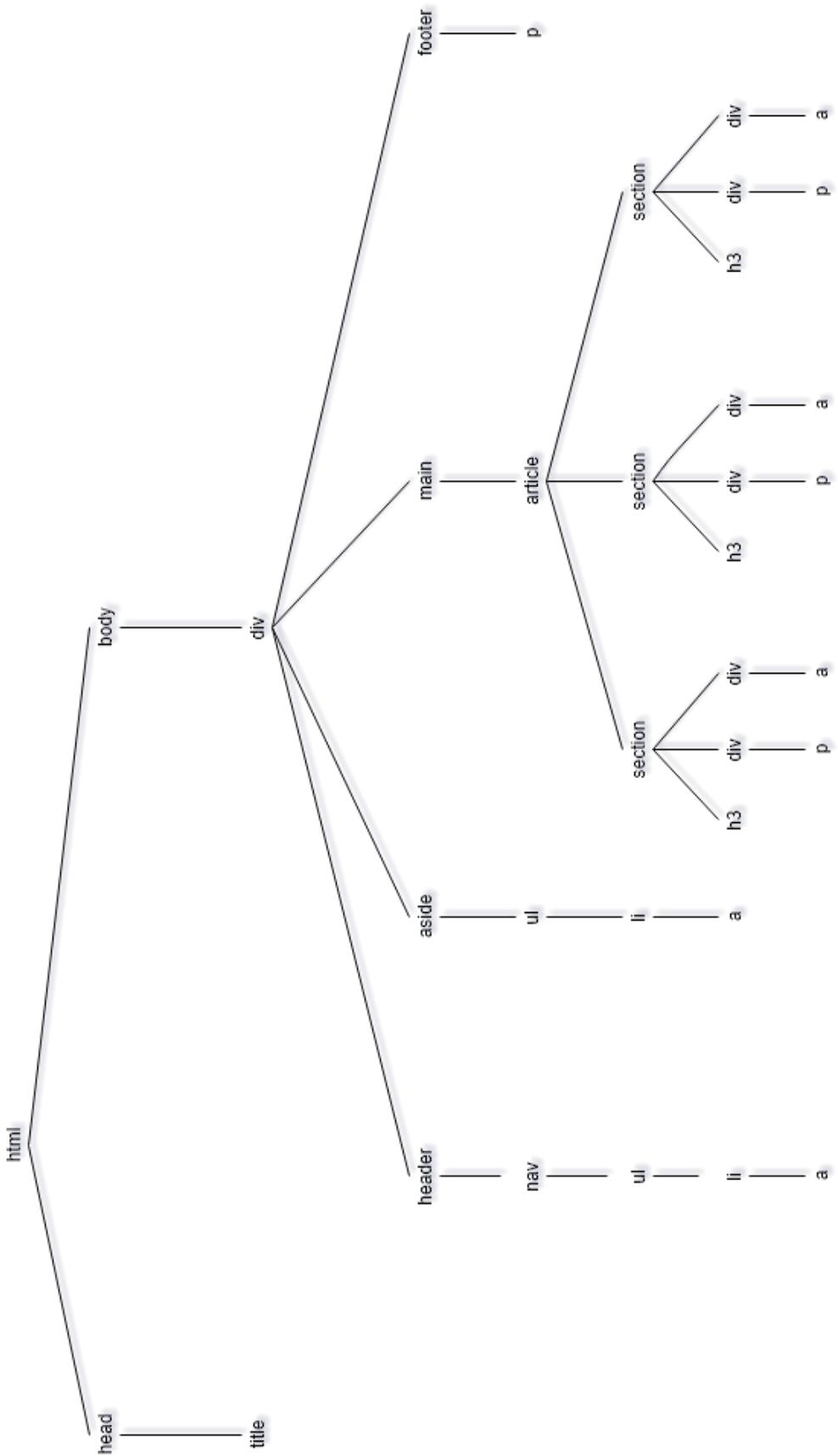
<!-- main 是主要顯示內容的區域 -->
<main class="content-container">
    <article class="article-paragraph">
        <section class="episode">
            <h3 class="title">HTML parser 開發不求人 - 第三節</h3>
            <div class="content">
                <p>動態網頁的元素走訪，都需要透過不斷地實作、練習，同時...</p>
            </div>
            <div class="content-more">
                <a class="more-link">More</a>
            </div>
        </section>
        <section class="episode">
            <h3 class="title">HTML parser 開發不求人 - 第二節</h3>

```

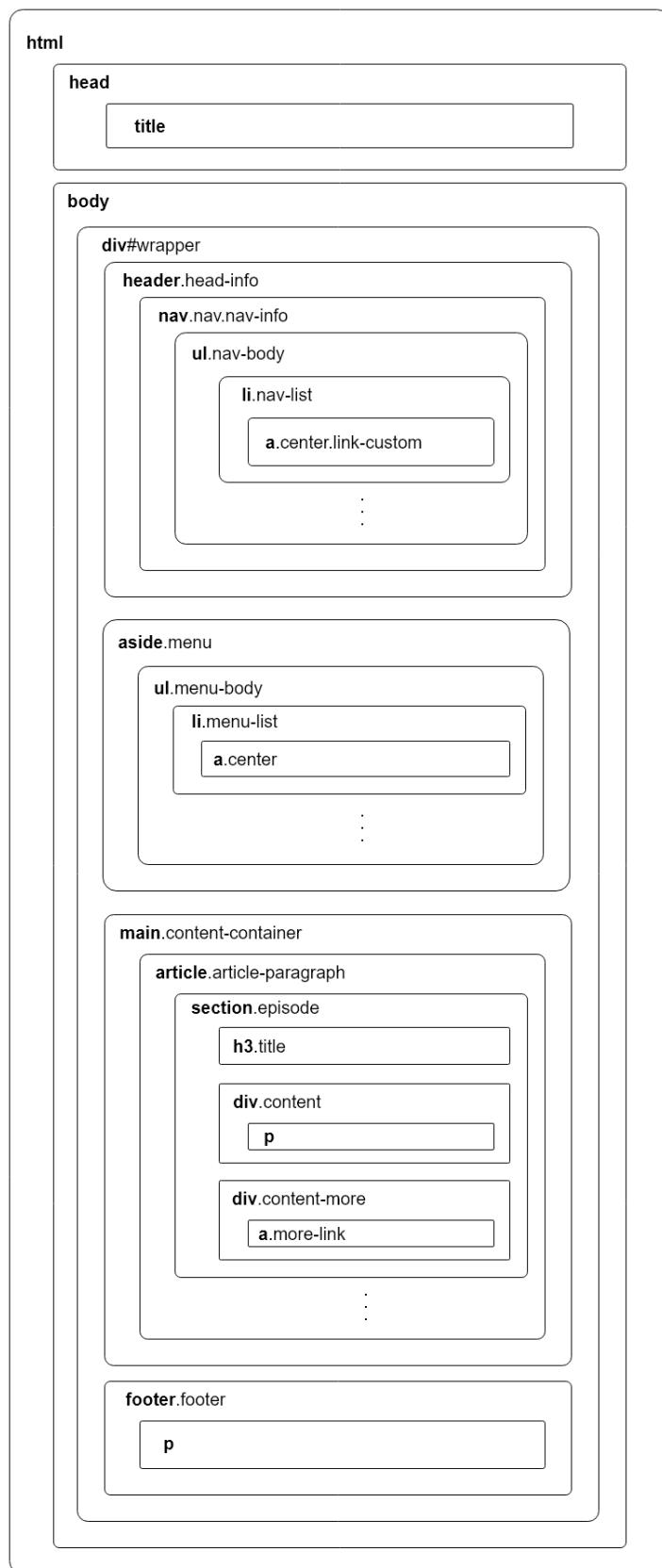
```
<div class="content">
    <p>動態網頁的元素走訪，都需要透過不斷地實作、練習，同時...</p>
</div>
<div class="content-more">
    <a class="more-link">More</a>
</div>
</section>
<section class="episode">
    <h3 class="title">HTML parser 開發不求人 - 第一節</h3>
    <div class="content">
        <p>動態網頁的元素走訪，都需要透過不斷地實作、練習，同時...</p>
    </div>
    <div class="content-more">
        <a class="more-link">More</a>
    </div>
    </section>
</article>
</main>

<!-- footer 在這裡是簡單提供網站的基礎資訊 --&gt;
&lt;footer class="footer"&gt;
    &lt;p&gt;歡迎光臨！本站由 Darren Yang 本人親自虛構...&lt;/p&gt;
&lt;/footer&gt;
&lt;/div&gt;
&lt;/body&gt;
&lt;/html&gt;</pre>
```

基本 HTML 階層，大致上長這個樣子：



若是網頁排版上的設定，應該長這個樣子：



標籤介紹

介紹範例裡頭的標籤。

標籤	範例	說明
<!DOCTYPE>	<!DOCTYPE html><html>...</html>	定義文件的格式。
<head>	<head><title>我的網站</title></head>	定義有關這個文件的資訊，至少會與<title></title>一起使用。
<title>	<title>我的網站</title>	定義文件的標題。
<header>	<header>頁首資訊</header>	文件的頁首資訊。
<nav>	<nav>導覽列</nav>	導覽列，定義導覽連結。
	清單項目 1清單項目 2	定義尚未排序的列表。
	清單項目 1清單項目 2	定義一個項目列表。
<!-- ... -->	<!-- 這裡是註解 -->	HTML 文件中的註冊。
<a>	<a>友站連結	定義超連結。
<main>	<main>放置主要內容</main>	具體說明文件的主要內容。
<section>	<section>放置需要區隔的資訊</section>	定義文件的部分內容。
<article>	<article>文章內容</article>	定義一篇文章。
<aside>	<aside>側欄資料</aside>	定義在網頁側邊的內容。
<div>	<div>放置需要區隔的資訊</div>	類似<section>，定義文件的部分內容。
<h1> 到 <h6>	<h3>某個主題或是需要明顯標註的資訊</h3>	定義 HTML 的標題/標頭。
<p>	<p>文章當中的段落</p>	定義文字段落。
<footer>	<footer>頁尾資訊</footer>	文件的頁尾資訊。

補充說明

網頁當中，也有表格元素，叫作「table」，它的格式通常如下：

<table>

```

<thead>
  <tr>
    <th>標題 1</th>
    <th>標題 2</th>
    <th>標題 3</th>
  </tr>
</thead>
<tbody>
  <tr>
    <td>第 1 行的第 1 個表格</td>
    <td>第 1 行的第 2 個表格</td>
    <td>第 1 行的第 3 個表格</td>
  </tr>
  <tr>
    <td>第 2 行的第 1 個表格</td>
    <td>第 2 行的第 2 個表格</td>
    <td>第 2 行的第 3 個表格</td>
  </tr>
  <tr>
    <td>第 3 行的第 1 個表格</td>
    <td>第 3 行的第 2 個表格</td>
    <td>第 3 行的第 3 個表格</td>
  </tr>
</tbody>
<tfoot>
  <tr>
    <td>表格尾端第 1 個表格</td>
    <td>表格尾端第 2 個表格</td>
    <td>表格尾端第 3 個表格</td>
  </tr>
</tfoot>
</table>

```

它跟其它元素一樣，可能會有自己的 CSS 設定，例如 id、class 等。

其它標籤的使用方式，可以參閱 w3school 的介紹：

<https://www.w3schools.com/tags/>

加入超連結

範例

```
<a href="https://www.ntu.edu.tw" target="_blank">國立臺灣大學</a>
```

a 是一種 html tag/element，也可視為一種物件，href 與 target 是 a 的屬性。

自訂屬性

在 HTML 5 的規範下，可以用「**data-***」的格式，來自訂屬性，以便 jQuery 可以透過自訂屬性來取得自訂的資料，例如「**data-item-id='11'**」、「**data-user-name='Darren Yang'**」、「**data-height='1024'**」、「**data-width='768'**」、「**data-what-you-may-call-it='Iron man'**」

範例

```
<a href=" https://www.iiiedu.org.tw/" target="_blank" data-item-id="5566" data-user-name="Darren Yang" data-tmp-path="/tmp">資策會數位教育研究所</a>
```

確定屬性後，便能透過元素擷取的套件，來取得屬性資訊。

7-2: 常見 HTTP 方法介紹

HTTP 請求方法在 RESTful API 中的典型應用				
資源	GET	PUT	POST	DELETE
一組資源的 URI，比如 <code>https://example.com/resource</code> s	列出 URI，以及該資源組中每個資源的詳細資訊（後者可選）。	使用給定的一組資源替換目前整個資源。	在本組資源中建立/追加一個新的資源。該操作往往返回新資源的 URL。	刪除整組資源。
單個資源的 URI，比如 <code>https://example.com/resource</code> s/142	取得指定的資源的詳細資訊，格式可以自選一個合適的網路媒體類型（比如：XML、JSON 等）	替換/建立指定的資源。並將其追加到相應的資源組中。	把指定的資源當做一個資源組，並在其下建立/追加一個新的元素，使其隸屬於目前資源。	刪除指定的元素。

7-3: 細說 GET 與 POST 方法

舉個例子，如果 HTTP 代表現在我們現實生活中寄信的機制，那麼信封的撰寫格式就是 HTTP。我們姑且將信封外的內容稱為 http-header，信封內的書信稱為 message-body，那麼 HTTP Method 就是你要告訴郵差的寄信規則。

假設 GET 表示信封內不得裝信件的寄送方式，如同是明信片一樣，你可以把要傳遞的資訊寫在信封(http-header)上，寫滿為止，價格比較便宜；然而 POST 就是信封內有裝信件的寄送方式（信封有內容物），不但信封可以寫東西，信封內(message-body) 還可以置入你想要寄送的資料或檔案，價格較貴。

使用 GET 的時候我們直接將要傳送的資料以 Query String (一種 Key/Vaule 的編碼方式) 加在我們要寄送的地址(URL)後面，然後交給郵差傳送。使用 POST 的時候則是將寄送地址(URL)寫在信封上，另外將要傳送的資料寫在另一張信紙後，將信紙放到信封裡面，交給郵差傳送。

GET 方法

表單資料將以字串方式附加在網址 (URI) 的後面傳送，在網址尾端，會以「？」符號，開啟跟著表單中的資料，每個欄位間的值，以「&」連接起來。

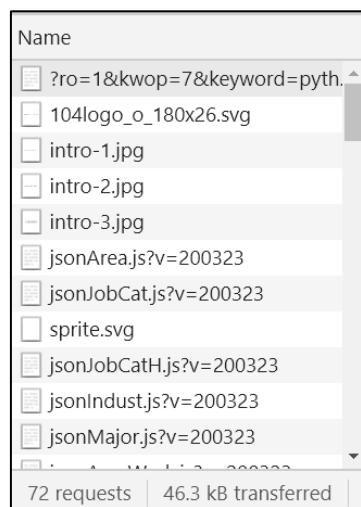
一般來說，GET 參數 (Query String) 的格式如下：

`https://www.104.com.tw/jobs/search/?ro=1&kwop=7&keyword=python&order=13&asc=0&page=1&mode=s&jobsource=2018indexpoc`

key (鍵)	value (值)
ro	1
kwop	7
keyword	python
order	13
asc	0
page	1
mode	s

key (鍵)	value (值)
jobsource	2018indexpoc

我們將前面的網址放到 chrome 瀏覽器的網址列中，讀取結束後，按下 Ctrl + Shift + i，再按下 Network，然後 Ctrl + R，重新讀取網址，再從左側的 Name 欄位裡，選擇最上面（通常最先被讀取的那個）的項目，再選 Headers，會看到以下的資訊：



圖：選擇第一個項目

Headers	Preview	Response	Initiator	Timing	Cookies
General <pre>Request URL: https://www.104.com.tw/jobs/search/?ro=1&kwop=7&keyword=python&order=13&asc=0&page=1&mode=s&jobsouce=2018indexpoc Request Method: GET Status Code: 200 OK Remote Address: 122.147.53.67:443 Referrer Policy: no-referrer-when-downgrade</pre>					

圖：看到 Headers 裡面的 General，明確指出 Request Method 是 GET

▼ Query String Parameters

ro: 1
kwop: 7
keyword: python
order: 13
asc: 0
page: 1
mode: s
jobsource: 2018indexpoc

圖：移到最下面，可以看到 Query String 的參數

POST 方法

我們用網頁表單的格式來說明：

格式

```
<form name = "myForm"  
action = "資料處理程式的 URI"  
method = "POST"  
enctype = "application/x-www-form-urlencoded 或是 multipart/form-data")  
>  
.....  
</form>
```

說明

- **name** 是指 form 的名稱，例如：myForm。
- **action** 是指該 form 被使用者送出之後，負責接收與處理資料的程式之 URI (Uniform Resource Identifiers)。如果省略不寫的話，會以當前所在的 URI 來取代。

說明

- `method` 用來規範該 form 被送出時，所採用的 HTTP method，預設值是 GET。
 - POST 方法是將資料包裝在 HTTP 標頭內 (message-body 當中) 傳送給 web server。
 - 使用 GET method 所能傳遞的資料有限 (連同 URI 共 255 字元)，在需要上傳大量資料或檔案時，會使用 POST method。
- `enctype` 用以規範該 form 被送出時，所採用的 content type。可用的值有兩種：application/x-www-form-urlencoded (預設值) 與 multipart/form-data。
 - 當您打算透過表單來上傳檔案時，請務必將 enctype 設為 multipart/form-data，同時 method 也要設為 POST 才行。

Module 8. CSS Selector 與 XPath

8-1: 概述 CSS Selector 與 XPath

CSS selectors 定義了 CSS 規則該套用在哪些網頁元素上，可以粗略分成「標籤選擇器」、「類別選擇器」、「ID 選擇器」、「通用選擇器」、「屬性選擇器」；XPath 即為 XML 路徑語言 (XML Path Language)，它是一種用來確定 XML 文檔中某部分位置的語言；XPath 基於 XML 的樹狀結構，提供在資料結構樹中找尋節點的能力。

8-2: 細說 CSS Selector

* : 通用選擇器(Universal selector)

將樣式套用於全部元素

```
<style type="text/css">  
  * { margin: 0px; padding: 0px; }  
</style>
```

e : 元素選擇器(Element type selector)

將樣式套用於指定元素

```
<style type="text/css">
```

```
  div { color: Red; }
```

```
p { color: Blue; }  
</style>
```

```
<div> Red </div>  
<p> Blue </p>
```

.class : 類別選擇器(Class selector)

將樣式套用於具指定類別的元素

```
<style type="text/css">  
    .Red { color: Red; }  
</style>  
  
<div class="Red"> Red </div>  
  
<p class="Red"> Red </p>
```

e.class : 元素類別選擇器(Class selector)

當元素 e 具指定類別 class，則套用樣式

```
<style type="text/css">  
    div.Red { color: Red; }  
</style>  
  
<div class="Red"> Red </div>
```

```
<p class="Red"> Black </p>
```

注意：元素 e 和.中間不可以有空白字元

#eid : ID 選擇器(ID selector)

將樣式套用於具指定 ID 的元素

```
<style type="text/css">
```

```
  #Red { color: Red; }
```

```
 </style>
```

```
<div id="Red"> Red </div>
```

e#eid : 元素 ID 選擇器(ID selector)

當元素 e 具指定 ID eid，則套用樣式

```
<style type="text/css">
```

```
  div#Red { color: Red; }
```

```
 </style>
```

```
<div id="Red"> Red </div>
```

```
<p id="Red"> Black </p>
```

註：元素 e 和#中間不可以有空白字元

e1, e2 [,e3,...] : 群組選擇器(Grouped selector)

對 e1、e2 元素套用相同樣式

```
<style type="text/css">  
    div, p { color: Black; }  
</style>  
  
<div>Black</div>  
  
<p>Black</p>
```

e d : 後代選擇器(Descendant selector)

若元素為 d，且為元素 e 的子元素或子孫元素，則套用樣式

```
<style type="text/css">  
    div span { color: Black; }  
</style>  
  
<div>  
  
    <span>Black</span>  
  
    <p>  
        <span>Black</span>  
    </p>  
  
</div>
```

e > c : 子元素選擇器(Child selector)

若元素 c 為元素 e 的子元素，則套用樣式

```
<style type="text/css">  
    div > span { color: Blue; }  
</style>  
  
<div>  
  
    <span>Blue</span>  
  
    <p>  
        <span>Black</span>  
    </p>  
  
</div>
```

e1 + e2 : 相鄰元素選擇器(Adjacent sibling selector)

若元素為 e2，且前面有 e1 元素，則套用樣式

```
<style type="text/css">  
    div + p { color: Blue; }  
</style>  
  
<div>
```

```

<p>Black</p>

<div>Black</div>

<p>Blue</p>

</div>

```

補充說明		
選擇器	例子	例子描述
<u>.class</u>	.intro	選擇 class="intro" 的所有元素。
.class1.class2	.name1.name2	選擇 class 屬性中同時有 name1 和 name2 的所有元素。
.class1.class2	.name1.name2	選擇作為類名 name1 元素後代的所有類名 name2 元素。
<u>#id</u>	#firstname	選擇 id="firstname" 的元素。
<u>*</u>	*	選擇所有元素。
<u>element</u>	p	選擇所有 <p> 元素。
<u>element.class</u>	p.intro	選擇 class="intro" 的所有 <p> 元素。
<u>element, element</u>	div, p	選擇所有 <div> 元素和所有 <p> 元素。
<u>element element</u>	div p	選擇 <div> 元素內的所有 <p> 元素。
<u>element > element</u>	div > p	選擇父元素是 <div> 的所有 <p> 元素。

補充說明

選擇器	例子	例子描述
<u><i>element + element</i></u>	div + p	選擇緊跟 <code><div></code> 元素的首個 <code><p></code> 元素。
<u><i>element1 ~ element2</i></u>	p ~ ul	選擇前面有 <code><p></code> 元素的每個 <code></code> 元素。
<u><i>[attribute]</i></u>	[target]	選擇帶有 <code>target</code> 屬性的所有元素。
<u><i>[attribute=value]</i></u>	[target=_blank]	選擇帶有 <code>target="_blank"</code> 屬性的所有元素。
<u><i>[attribute~=value]</i></u>	[title~=flower]	選擇 <code>title</code> 屬性包含單詞 "flower" 的所有元素。
<u><i>[attribute =value]</i></u>	[lang =en]	選擇 <code>lang</code> 屬性值以 "en" 開頭的所有元素。
<u><i>[attribute^=value]</i></u>	a[href^="https"]	選擇其 <code>src</code> 屬性值以 "https" 開頭的每個 <code><a></code> 元素。
<u><i>[attribute\$=value]</i></u>	a[href\$=".pdf"]	選擇其 <code>src</code> 屬性以 ".pdf" 結尾的所有 <code><a></code> 元素。
<u><i>[attribute*=value]</i></u>	a[href*="w3schools"]	選擇其 <code>href</code> 屬性值中包含 "abc" 子串的每個 <code><a></code> 元素。
<u><i>:active</i></u>	a:active	選擇活動連結。
<u><i>::after</i></u>	p::after	在每個 <code><p></code> 的內容之後插入內容。
<u><i>::before</i></u>	p::before	在每個 <code><p></code> 的內容之前插入內容。

補充說明

選擇器	例子	例子描述
<u>:checked</u>	input:checked	選擇每個被選中的 <input> 元素。
<u>:default</u>	input:default	選擇預設的 <input> 元素。
<u>:disabled</u>	input:disabled	選擇每個被禁用的 <input> 元素。
<u>:empty</u>	p:empty	選擇沒有子元素的每個 <p> 元素（包括文本節點）。
<u>:enabled</u>	input:enabled	選擇每個啟用的 <input> 元素。
<u>:first-child</u>	p:first-child	選擇屬於父元素的第一個子元素的每個 <p> 元素。
<u>::first-letter</u>	p::first-letter	選擇每個 <p> 元素的首字母。
<u>::first-line</u>	p::first-line	選擇每個 <p> 元素的首行。
<u>:first-of-type</u>	p:first-of-type	選擇屬於其父元素的首個 <p> 元素的每個 <p> 元素。
<u>:focus</u>	input:focus	選擇獲得焦點的 input 元素。
<u>:fullscreen</u>	:fullscreen	選擇處於全屏模式的元素。
<u>:hover</u>	a:hover	選擇滑鼠指標位於其上的連結。
<u>:in-range</u>	input:in-range	選擇其值在指定範圍內的 input 元素。
<u>:indeterminate</u>	input:indeterminate	選擇處於不確定狀態的 input 元素。

補充說明

選擇器	例子	例子描述
<u>:invalid</u>	input:invalid	選擇具有無效值的所有 input 元素。
<u>:lang(<i>language</i>)</u>	p:lang(it)	選擇 lang 屬性等於 "it" (義大利) 的每個 <p> 元素。
<u>:last-child</u>	p:last-child	選擇屬於其父元素最後一個子元素每個 <p> 元素。
<u>:last-of-type</u>	p:last-of-type	選擇屬於其父元素的最後 <p> 元素的每個 <p> 元素。
<u>:link</u>	a:link	選擇所有未訪問過的連結。
<u>:not(<i>selector</i>)</u>	:not(p)	選擇非 <p> 元素的每個元素。
<u>:nth-child(n)</u>	p:nth-child(2)	選擇屬於其父元素的第二個子元素的每個 <p> 元素。
<u>:nth-last-child(n)</u>	p:nth-last-child(2)	同上，從最後一個子元素開始計數。
<u>:nth-of-type(n)</u>	p:nth-of-type(2)	選擇屬於其父元素第二個 <p> 元素的每個 <p> 元素。
<u>:nth-last-of-type(n)</u>	p:nth-last-of-type(2)	同上，但是從最後一個子元素開始計數。
<u>:only-of-type</u>	p:only-of-type	選擇屬於其父元素唯一的 <p> 元素的每個 <p> 元素。
<u>:only-child</u>	p:only-child	選擇屬於其父元素的唯一子元素的每個 <p> 元素。
<u>:optional</u>	input:optional	選擇不帶 "required" 屬性的 input 元素。

補充說明

選擇器	例子	例子描述
<u>:out-of-range</u>	input:out-of-range	選擇值超出指定範圍的 input 元素。
<u>::placeholder</u>	input::placeholder	選擇已規定 "placeholder" 屬性的 input 元素。
<u>:read-only</u>	input:read-only	選擇已規定 "readonly" 屬性的 input 元素。
<u>:read-write</u>	input:read-write	選擇未規定 "readonly" 屬性的 input 元素。
<u>:required</u>	input:required	選擇已規定 "required" 屬性的 input 元素。
<u>:root</u>	:root	選擇文檔的根項目。
<u>::selection</u>	::selection	選擇使用者已選取的元素部分。
<u>:target</u>	#news:target	選擇當前活動的 #news 元素。
<u>:valid</u>	input:valid	選擇帶有有效值的所有 input 元素。
<u>:visited</u>	a:visited	選擇所有已訪問的連結。

參考資料：

CSS 選擇器參考手冊

https://www.w3school.com.cn/cssref/css_selectors.asp

8-3: 細說 XPath

xpath 屬性定位

xpath 可以通過元素的 id, name, class 這些屬性定位，如下：

```
driver.find_element_by_xpath("//*[@id='kw']").send_keys("by_xpath")
driver.find_element_by_xpath("//*[@name='wd']").send_keys("by_xpath")
driver.find_element_by_xpath("//*[@class='s_ipt']").send_keys("by_xpath")
```

xpath 其他屬性定位

```
driver.find_element_by_xpath("//*[@autocomplete='off']").send_keys("by_xpath")
```

path 標籤

同一個屬性同名較多的時候，可以指定標籤，定位更準，如下：

```
driver.find_element_by_xpath("//input[@id='kw']").send_keys("by_xpath")
```

xpath 層級

如果一個元素的屬性不是很明顯，可以通過父元素來找它，如果父元素的屬性

也不是很明顯，就通過父元素的父元素，即爺爺元素來找，如下

```
# 通過父元素定位
driver.find_element_by_xpath("//span[@id='s_kw_wrap']/input").send_keys("by_xpath")
# 通過爺爺元素定位
driver.find_element_by_xpath("//form[@id='form']/span/input").send_keys("by_xpath")
```

xpath 索引

如果一個元素它的兄弟元素跟它標籤一樣，這時候無法通過層級定位，可以通過索引定位，如下：

```
driver.find_element_by_xpath("//select[@id='nr']/option[1]").click()  
driver.find_element_by_xpath("//select[@id='nr']/option[2]").click()  
driver.find_element_by_xpath("//select[@id='nr']/option[3]").click()
```

這裡索引是從 1 開始的，跟 python 的索引不一樣。

xpath 邏輯運算

xpath 支援與(and)、或(or)、非(not)，如下：

```
driver.find_element_by_xpath("//*[@id='kw' and @autocomplete='off']").send_keys("by_xpath")
```

xpath 模糊匹配

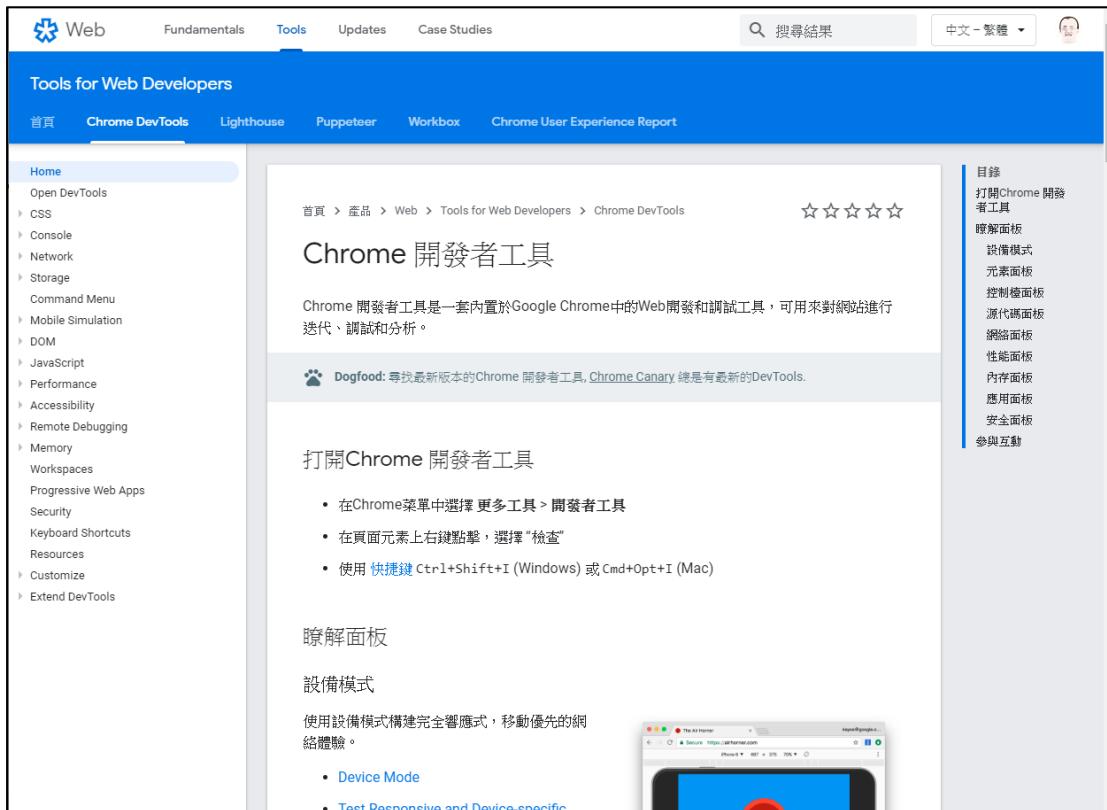
```
# 匹配文字  
driver.find_element_by_xpath("//*[contains(text(), 'hao123')]").click()  
# 匹配屬性  
driver.find_element_by_xpath("//*[contains(@id, 'kw')]").click()  
# 匹配以什麼開頭  
driver.find_element_by_xpath("//*[starts-with(@id, 's_kw_')]").click()  
# 匹配以什麼結尾  
driver.find_element_by_xpath("//*[ends-with(@id, 'kw_wrap')]").click()
```

Module 9. Chrome Developer Tool

9-1: 各頁籤常用功能簡介 (Elements / Console / Network / ...)

Chrome 開發者工具是內建於 Google Chrome 中的 Web 開發和測試工具

網址：<https://developers.google.com/web/tools/chrome-devtools?hl=zh-tw>



圖：Chrome 開發者工具的說明網頁

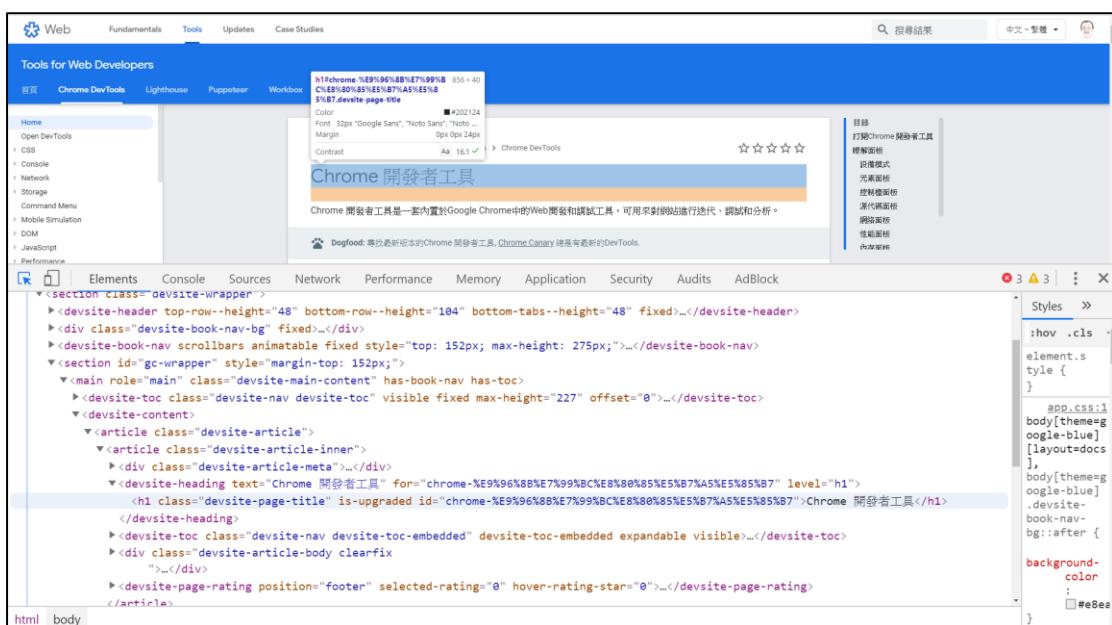
開啟開發工具(dock)

- F12
- Ctrl + Shift + I

Elements 面板

檢查 HTML 元素

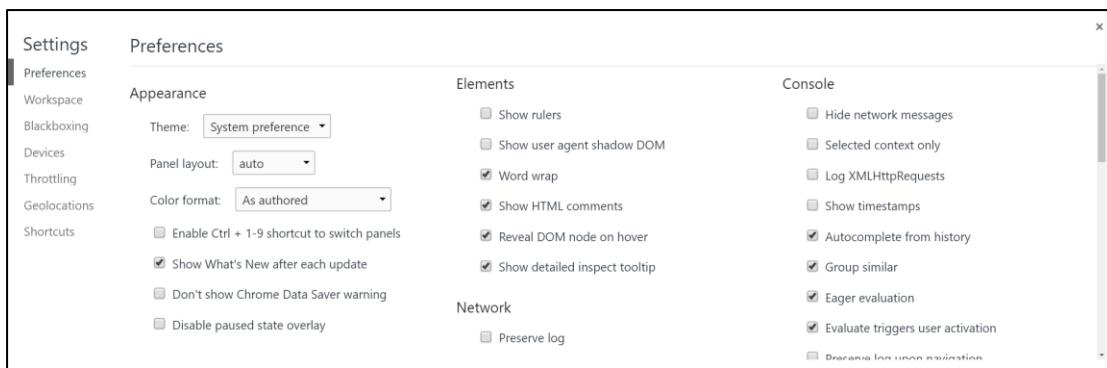
- Ctrl + Shift + C (追縱滑鼠移過網頁元素所在位置的狀態)
- 網頁內容任意處按滑鼠右鍵→檢查



圖：檢查元素

補充說明

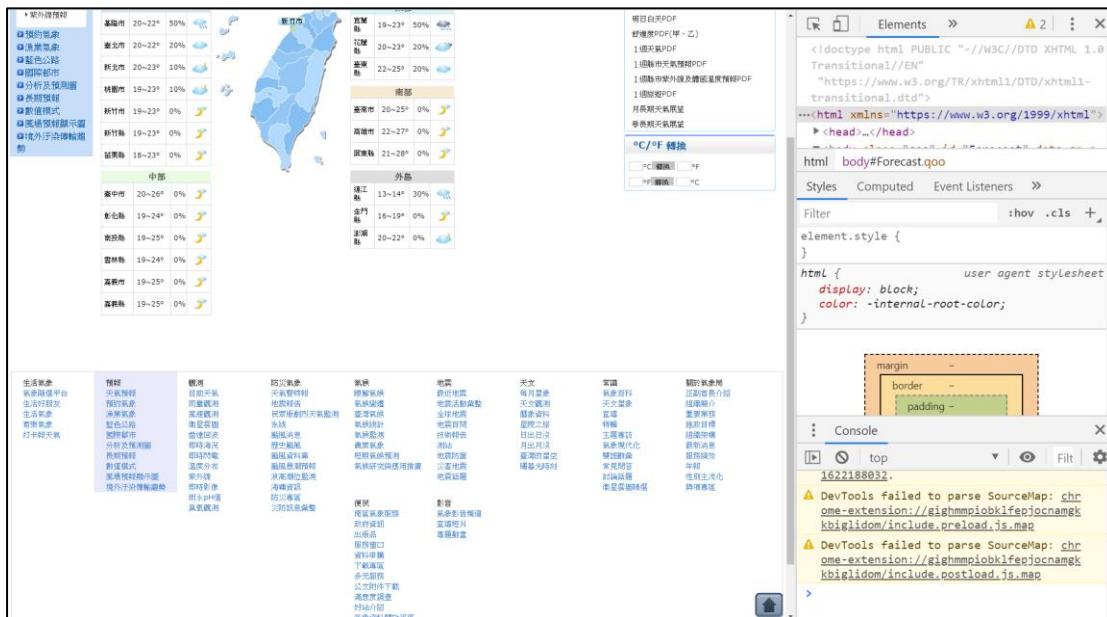
開啟 Chrome 開發者工具以後，按下 F1，可以看到一些偏好設定，方便我們設定開發工具，例如顯示外觀、模擬裝置、自訂地理位置、快捷鍵等。



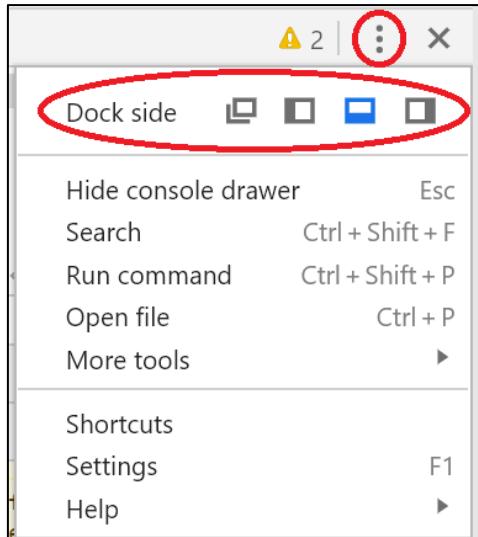
圖：Chrome 開發者工具偏好設定

開啟開發工具後，常用快速鍵：

- **Ctrl + Shift + D** 切換檢查元素的 dock side

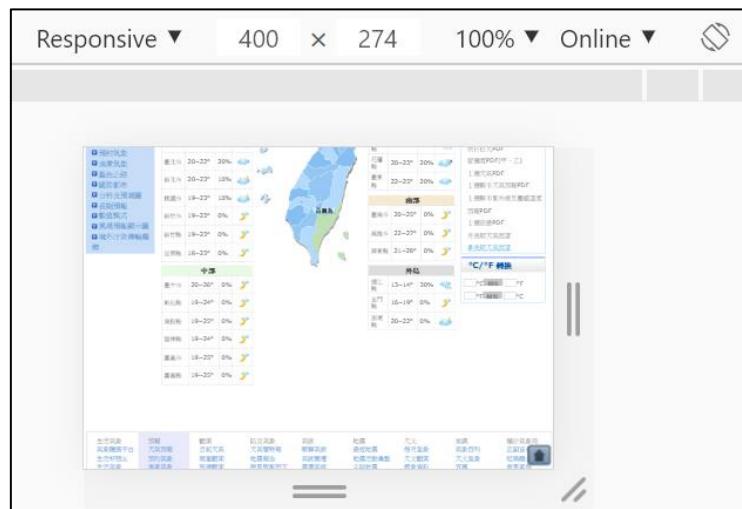


圖：切換 dock side，從下方到右側

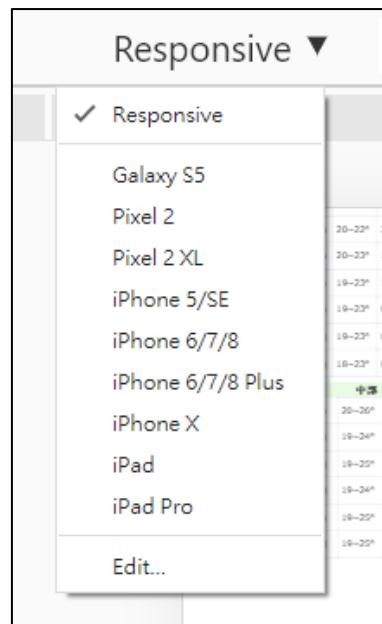


圖：按下三個點的圖示，也可以選擇 dock side

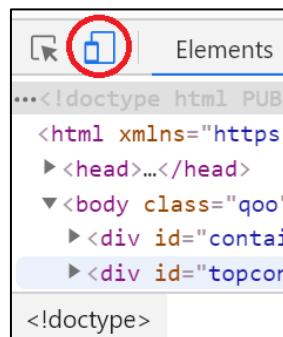
- **Ctrl + Shift + M** 開啟模擬裝置模式(切換裝置工具欄)



圖：可選擇不用的行動裝置，或自訂寬高，來顯示網頁



圖：選擇裝置來觀看網頁

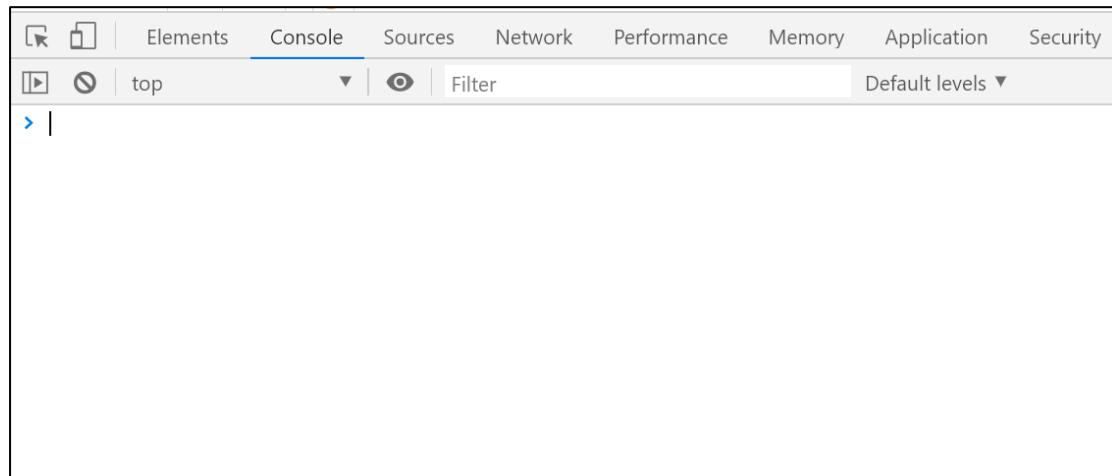


圖：等同按下切換裝置工具欄

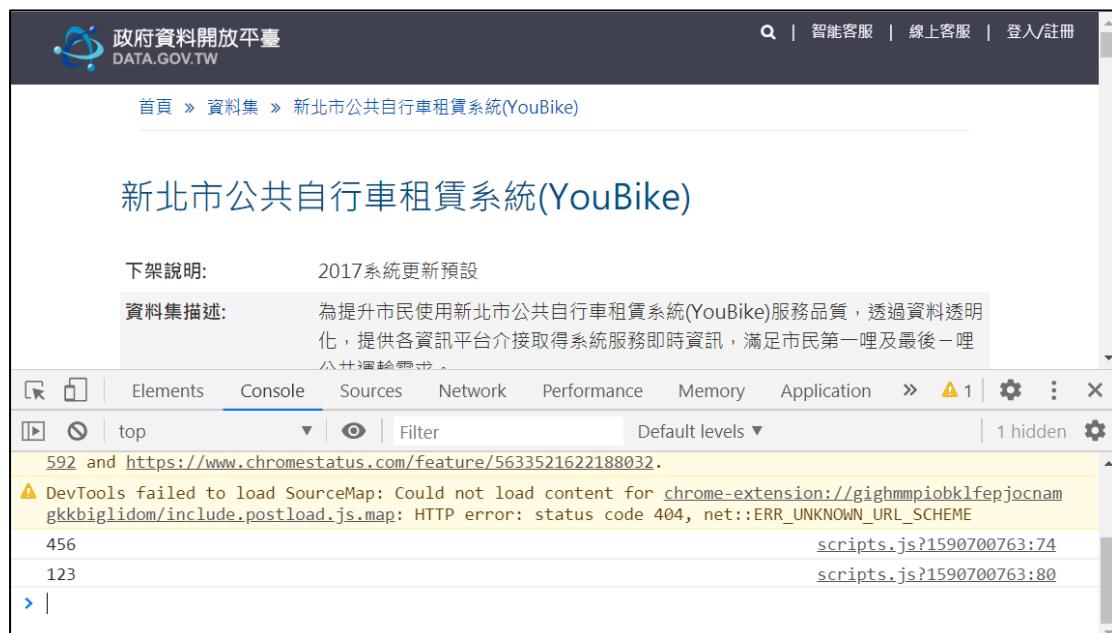
- Ctrl + O 尋找 HTML 當中的檔名
- Ctrl + R 或 F5 刷新頁面
- Ctrl + F5 清除快取後，刷新頁面(重新從伺服器端請求下載 HTML)
- Ctrl + L 清除 Console
- Shift + Enter 在 Console 中斷行(或多行)

Console 面板

我們可以使用 Console 面板，了解目前網頁執行的狀況。



圖：Console 面板

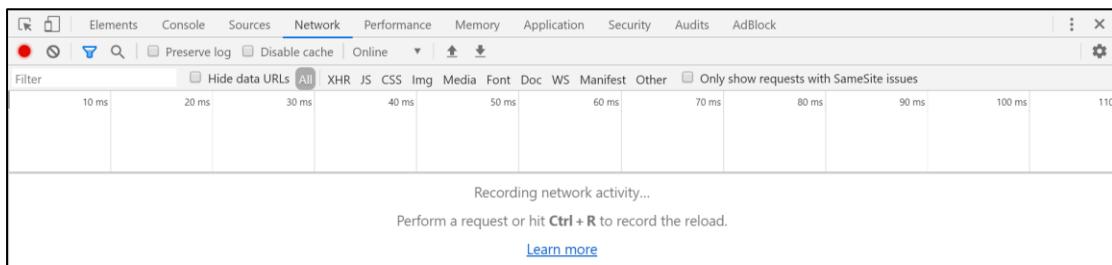


圖：可以看到目前網站的情況。

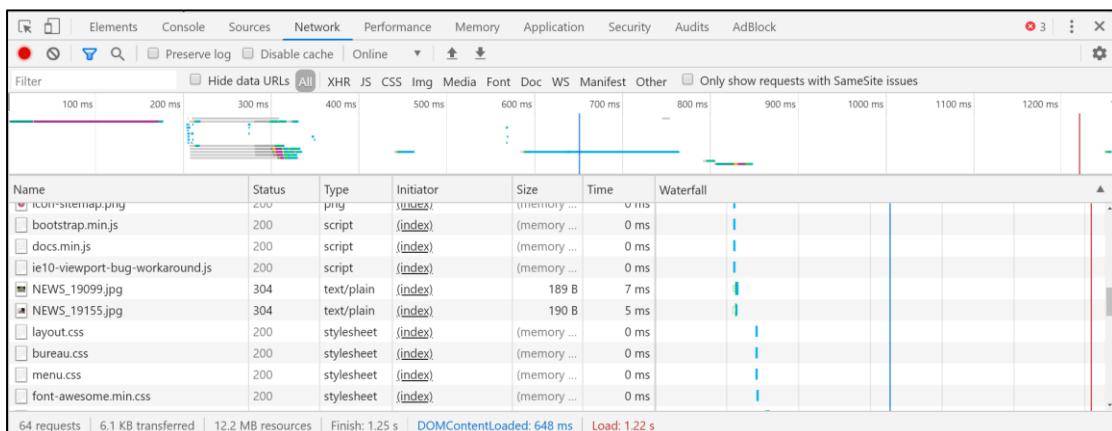
Network 面板

Network 面板會顯示出所有網路請求的詳細訊息記錄，包括狀態、資源類型、

大小、所需時間、HTTP request header 和 response header 等等，明確找出哪些請求比預期還要耗時，並加以調整，是優化網頁的重要工具。



(圖) Network 面板會記錄任何的網路活動



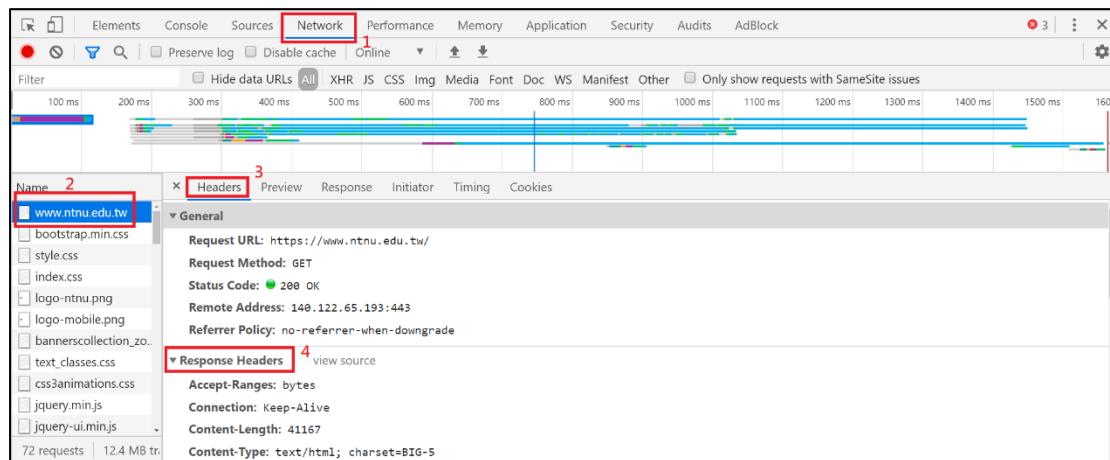
(圖) 記錄網頁讀取的資訊與下載順序

我們可以透過 Headers，來了解網頁請求的狀況。開啟 Headers 的流程為：

1. 開啟 Network 面板
2. Ctrl + R 或是 F5 刷新頁面

3. 點選左側的檔案名稱

4. 點選 Headers



圖：觀看檔案的 Headers 內容

最後，讓我們來看一下 Request Headers 與 Response Headers。

(下列表格參考維基百科)

Request Headers (請求標頭)

標頭欄位	說明	範例
Accept	能夠接受的回應內容類型（Content-Types）。	Accept: text/plain
Accept-Encoding	能夠接受的編碼方式列表。參考 HTTP 壓縮	Accept-Encoding: gzip, deflate
Accept-Language	能夠接受的回應內容的自然語言列表。	Accept-Language: en-US

標頭欄位	說明	範例
Authorization	用於超文字傳輸協定的認證資訊	Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Cache-Control	用來指定在這次的請求/回應鏈中的所有快取機制都必須遵守的指令	Cache-Control: no-cache
Connection	該瀏覽器想要優先使用的連接類型	Connection: keep-alive Connection: Upgrade
Cookie	之前由伺服器通過 Set-Cookie 傳送的一個 超文字傳輸協定 Cookie	Cookie: _ga=GA1.3.1322956465.1572335045;locale =zh_TW; _gid=GA1.3.1110994946.1584940974; _gat_gtag_UA_141775379_1=1
Content-Length	以 八位位元組陣列 (8 位元的位元組) 表示的請求體的長度	Content-Length: 348
Content-Type	請求多媒體類型 (用於 POST 和 PUT 請求中)	Content-Type: application/x-www-form-urlencoded

標頭欄位	說明	範例
Host	伺服器的域名(用於虛擬主機)，以及伺服器所監聽的埠號。如果所請求的埠是對應的服務的標準埠，則埠號可被省略。	Host: en.wikipedia.org:80 Host: en.wikipedia.org
Origin	發起一個針對 跨來源資源共享 的請求（要求伺服器在回應中加入一個『存取控制-允許來源』('Access-Control-Allow-Origin') 欄位）。	Origin: http://www.example-social-network.com
Pragma	每次發出請求，確認有沒有新的檔案；若有 Cache-Control，則直接看 Cache-Control。	Pragma: no-cache
Referer	表示瀏覽器所存取的前一個頁面，正是那個頁面上的某個連結將瀏覽器帶到了目前所請求的這個頁面。	Referer: http://en.wikipedia.org/wiki/Main_Page
User-Agent	瀏覽器的瀏覽器身分標識字串	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/21.0

標頭欄位	說明	範例
Upgrade	要求伺服器升級到另一個協定。	Upgrade: HTTP/2.0, SHTTP/1.3, IRC/6.9, RTA/x11

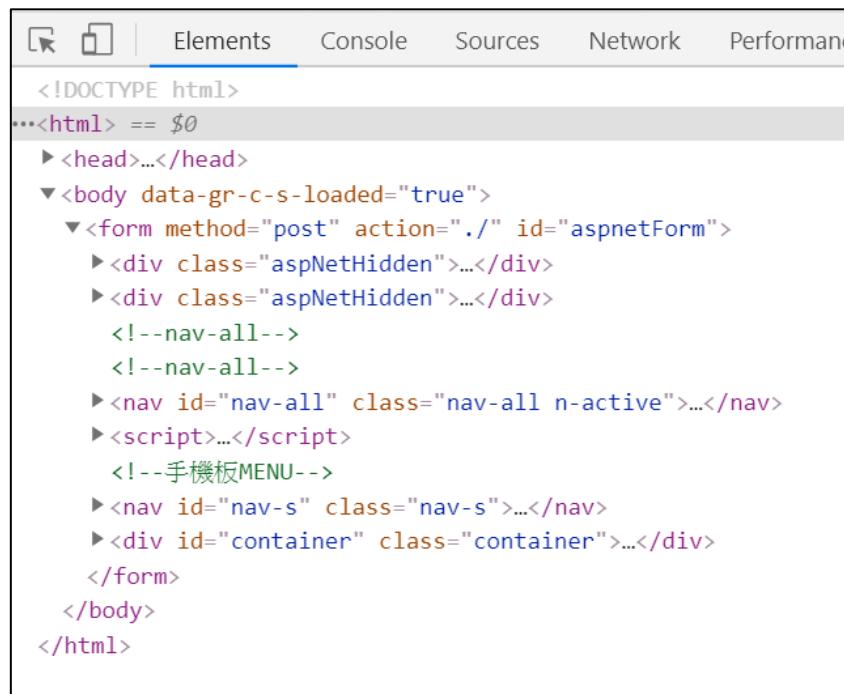
Response Headers (回應標頭)

標頭名	說明	例子
Access-Control-Allow-Origin	指定哪些網站可參與到跨來源資源共享過程中	Access-Control-Allow-Origin: *
Accept-Ranges	這個伺服器支援哪些種類的部分內容範圍	Accept-Ranges: bytes
Age	這個物件在代理快取中存在的時間，以秒為單位	Age: 12
Allow	對於特定資源有效的動作。針對HTTP/405這一錯誤代碼而使用	Allow: GET, HEAD
Cache-Control	向從伺服器直到客戶端在內的所有快取機制告知，它們是否可以快取這個物件。其單位為秒	Cache-Control: max-age=3600
Connection	針對該連接所預期的選項	Connection: close
Content-Disposition	一個可以讓客戶端下載檔案並建議檔名的頭部。檔名需要用雙引號包裹。	Content-Disposition: attachment; filename="fname.ext"
Content-	在資料上使用的編	Content-Encoding: gzip

標頭名	說明	例子
Encoding	碼類型。	
Content-Language	內容所使用的語言	Content-Language: da
Content-Length	回應訊息體的長度，以 位元組（8 位元為一位元組）為單位	Content-Length: 348
Content-Location	所返回的資料的一個候選位置	Content-Location: /index.htm
Content-Type	目前內容的 MIME 類型	Content-Type: text/html; charset=utf-8
Date	此條訊息被傳送時的日期和時間	Date: Tue, 15 Nov 1994 08:12:31 GMT
ETag	對於某個資源的某個特定版本的一個識別碼，通常是一個訊息雜湊	ETag: "737060cd8c284d8af7ad3082f209582d"
Expires	指定一個日期/時間，超過該時間則認為此回應已經過期	Expires: Thu, 01 Dec 1994 16:00:00 GMT
Last-Modified	所請求的物件的最後修改日期	Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT
Location	用來進行重新導向，或者在建立了某個新資源時使用。	Location: http://www.w3.org/pub/WWW/People.html
Pragma	每次發出請求，確認有沒有新的檔案；若有 Cache-Control，則直接看 Cache-Control。	Pragma: no-cache
Refresh	用於設定可定時的	Refresh: 5;

標頭名	說明	例子
	重新導向跳轉。	url=http://www.w3.org/pub/WWW/People.html
Server	伺服器的名字	Server: Apache/2.4.1 (Unix)
Set-Cookie	HTTP cookie	Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1
Status	用來說明目前這個超文字傳輸協定回應的狀態。	Status: 200 OK

9-2: 資策會首頁分析



```

<!DOCTYPE html>
...<html> == $0
  > <head>...</head>
  > <body data-gr-c-s-loaded="true">
    > <form method="post" action="/" id="aspnetForm">
      > <div class="aspNetHidden">...</div>
      > <div class="aspNetHidden">...</div>
      > <!--nav-all-->
      > <!--nav-all-->
      > <nav id="nav-all" class="nav-all n-active">...</nav>
      > <script>...</script>
      > <!--手機板MENU-->
      > <nav id="nav-s" class="nav-s">...</nav>
      > <div id="container" class="container">...</div>
    > </form>
  > </body>
</html>

```

圖：檢查元素，可看得出 nav 元素分成一般與手機版本

```

▼<div id="container" class="container">
  <!--ContentPlaceHolder1-->
  ><section id="slider-AD" style="max-height: 850px; margin-top: 0px;">...</section>
  <!--information-news-->
  ><section id="slider-I-news" style="margin-top: 0px;">...</section>
  <!--mv-youtube-->
  ><section id="mv-youtube" class="mv-youtube">...</section>
  <!--slider-F-news-->
  ><section id="slider-F-news" style="display:none">...</section>
  <!--slider-Focus-->
  ><section id="slider-Focus">...</section>
  <!--link-others-->
  ><section id="link-others">...</section>
  <!--link-favorite-->
  ><section id="link-favorite">...</section>
  <!-- 輪播圖 -->
  <!-- Swiper JS -->
  <script src="dist/js/swiper.jquery.min.js"></script>
  <!-- Initialize Swiper -->
  ><script>...</script>
  <!--about-iii-->
  ><section id="link-about">...</section>

```

圖：網站由上而下分成幾個 sections



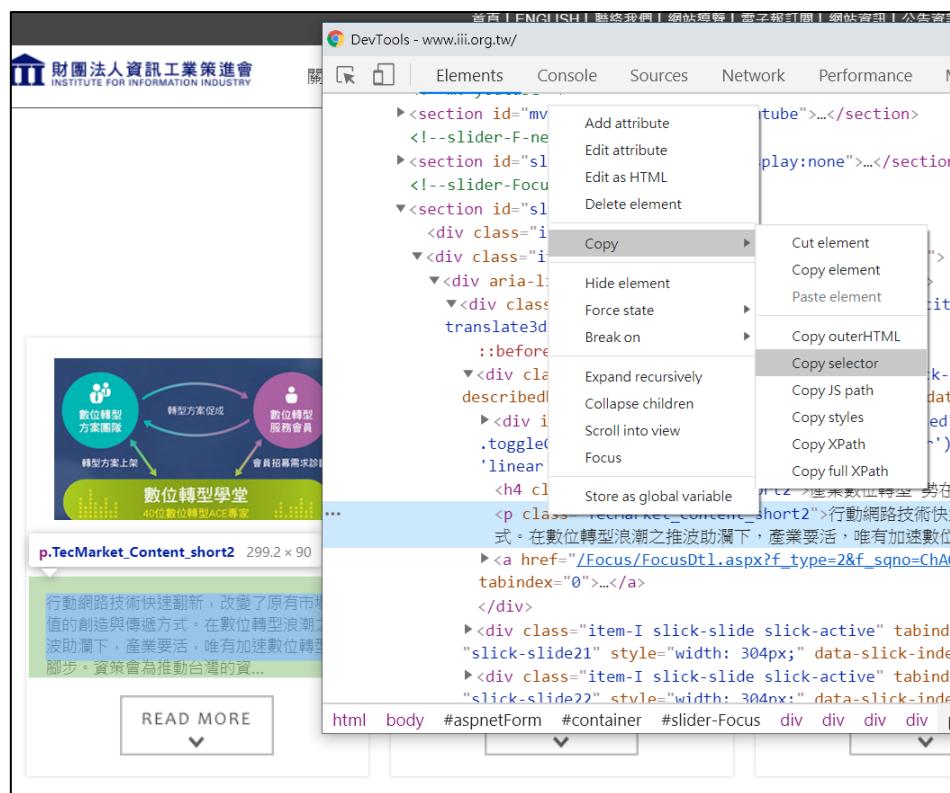
圖：焦點報導有三個元素



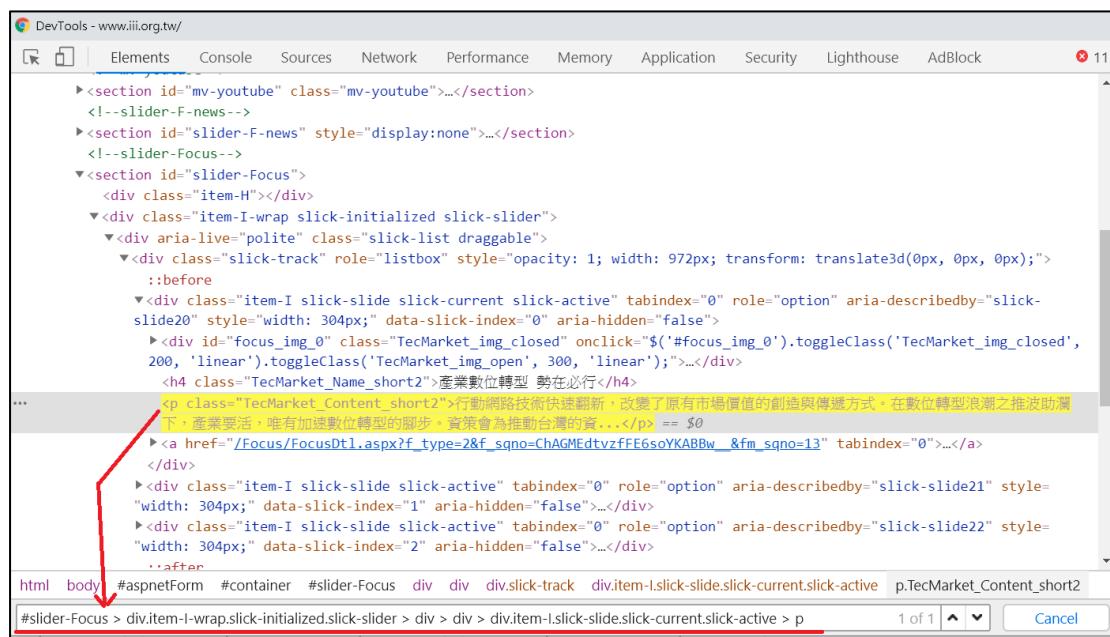
圖：報導包在 div.item-I-wrap.slick-initialized.slick-slider 裡面



圖：若是想要取得文字區塊的選擇器

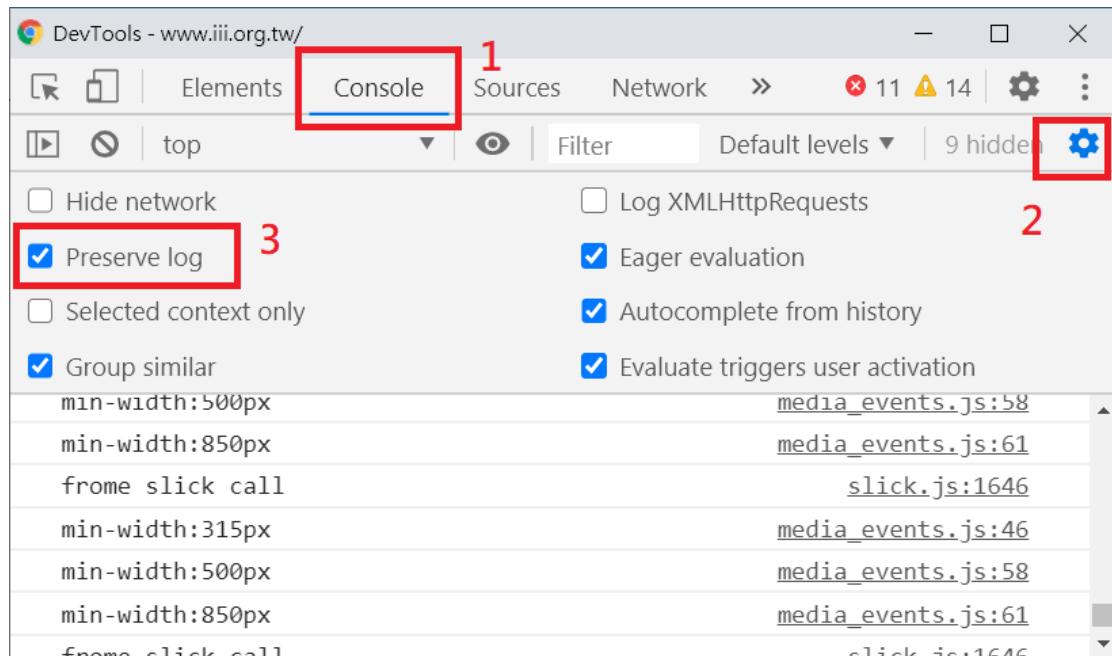


圖：對著 Elements 面板的元素區塊按右鍵→Copy→Copy Selector

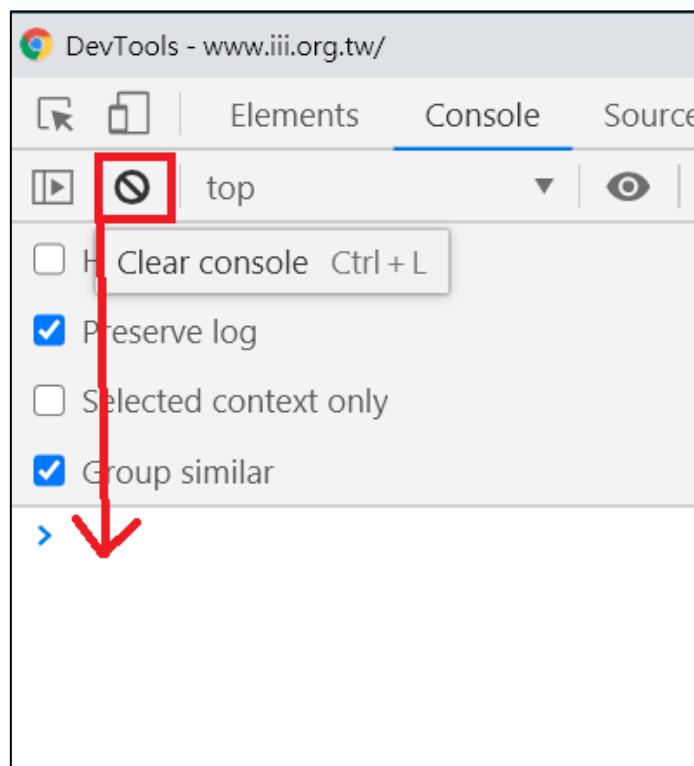


圖：對 elements 面板按 Ctrl+F，將剛才的 CSS selector 貼上，得到驗證

9-3: 常用操作流程介紹 (Preserve Log / Clear / ...)



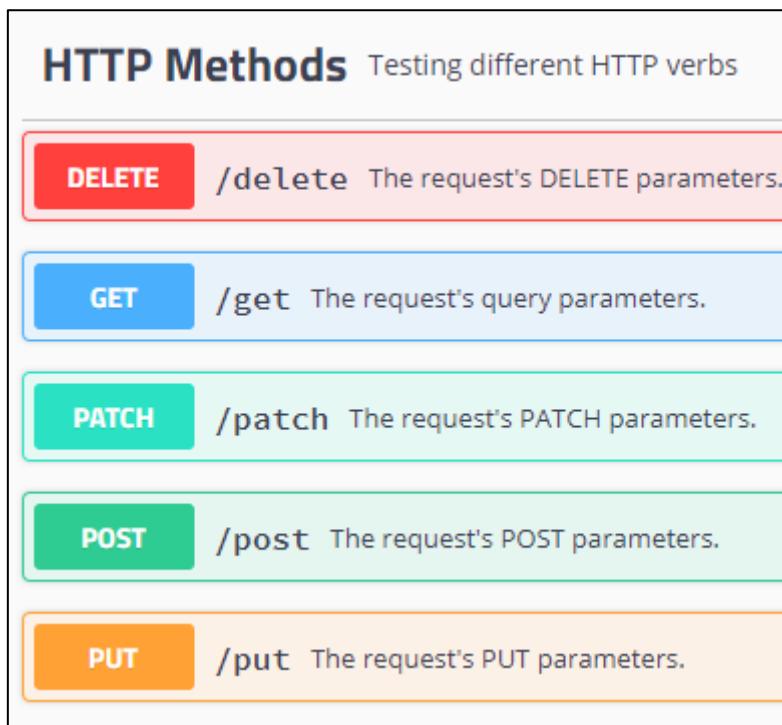
圖：勾選 Preserve log，縱然頁面刷新，過去的 log 依然會保存起來



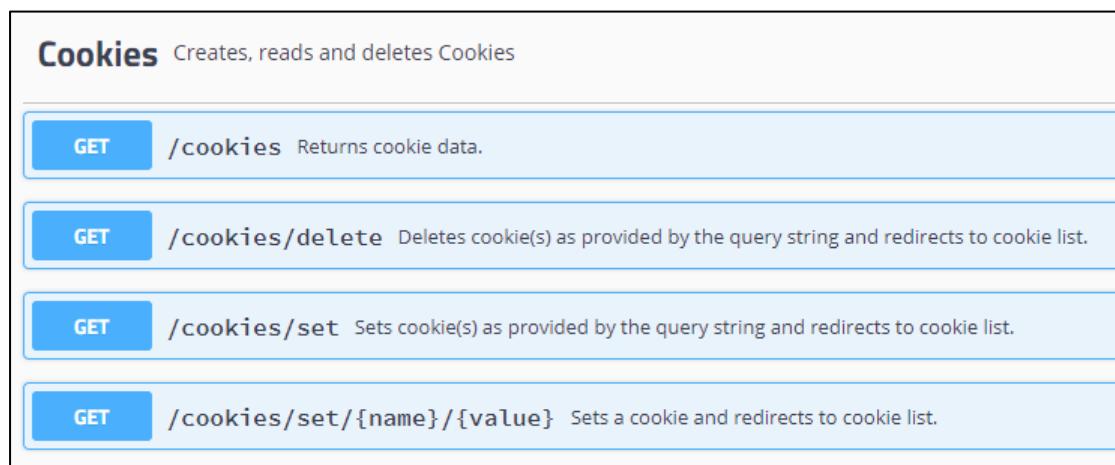
圖：按下 Clear Console，或是 Ctrl+L，即可清除 log

Module 10. 套件 requests

<https://httpbin.org/> 是一個專門拿來測試 HTTP Request 的網路服務，只要依照文件發動 HTTP Request 到指定的路徑，就會將它收到的內容以 JSON 格式回傳，在測試 API 行為時非常好用。



圖：支援的 HTTP 方法



圖：增刪修 cookies 的操作方法

10-1: 觀察理解目標並發出請求 (Request)

In [1]:

```
# 引入 requests 模組
import requests

# 使用 GET 方式下載普通網頁
r = requests.get('https://httpbin.org/get')

# 伺服器回應的狀態碼
print(r.status_code)

# 檢查狀態碼是否 OK
if r.status_code == requests.codes.ok:
    print("OK")

# 輸出網頁 HTML 原始碼
print(r.text)

200
OK
{
    "args": {},
    "headers": {
        "Accept": "*/*",
        "Accept-Encoding": "gzip, deflate",
        "Host": "httpbin.org",
        "User-Agent": "python-requests/2.24.0",
        "X-Amzn-Trace-Id": "Root=1-5fad5383-679ff26c439bb8ff3d09daa5"
    },
    "origin": "114.43.211.109",
    "url": "https://httpbin.org/get"
}
```

In [3]:

```
# GET 方法的 query string
my_params = {
```

```
'key1': 'value1',
'key2': 'value2'
}

# 將 query string 加入 GET 請求中
r = requests.get('https://httpbin.org/get', params = my_params)

# 觀察 URL
print(r.url)

# 輸出網頁 HTML 原始碼
print(r.text)
https://httpbin.org/get?key1=value1&key2=value2
{
    "args": {
        "key1": "value1",
        "key2": "value2"
    },
    "headers": {
        "Accept": "*/*",
        "Accept-Encoding": "gzip, deflate",
        "Host": "httpbin.org",
        "User-Agent": "python-requests/2.24.0",
        "X-Amzn-Trace-Id": "Root=1-5fad5493-3e3b60b2195f9bec3d1c42c4"
    },
    "origin": "114.43.211.109",
    "url": "https://httpbin.org/get?key1=value1&key2=value2"
}
```

In []:

'''
如果你有 GitHub 帳號，
可以透過這個 Web API，
來取得個人資料
'''

```
# 需要帳號登入的網頁
r = requests.get('https://api.github.com/user', auth=('帳號', '密碼'))

# 輸出網頁 HTML 原始碼
print(r.text)

In [7]:
# POST 方法的 form data
my_data = {
    'key1': 'value1',
    'key2': 'value2'
}

# 將 form data 加入 POST 請求中
r = requests.post('https://httpbin.org/post', data = my_data)

# 輸出網頁 HTML 原始碼
print(r.text)
{
    "args": {},
    "data": "",
    "files": {},
    "form": {
        "key1": "value1",
        "key2": "value2"
    },
    "headers": {
        "Accept": "*/*",
        "Accept-Encoding": "gzip, deflate",
        "Content-Length": "23",
        "Content-Type": "application/x-www-form-urlencoded",
        "Host": "httpbin.org",
        "User-Agent": "python-requests/2.24.0",
        "X-Amzn-Trace-Id": "Root=1-5fad54de-319fbc201ae876af2b8fad45"
    },
    "json": null,
}
```

```
"origin": "114.43.211.109",
"url": "https://httpbin.org/post"
}

In [ ]:

# 要上傳的檔案 (變數名稱為 my_filename)
my_files = {
    'my_filename': open('turingcerts.jpg', 'rb')
}

# 將檔案加入 POST 請求中
r = requests.post('https://httpbin.org/post', files = my_files)

# 輸出網頁 HTML 原始碼
print(r.text)
```

In [8]:

```
'''  
如果伺服器傳回的網頁資料中含有 cookies , requests 也可以輕鬆取出 c  
ookies 的資料  
'''
```

```
# 含有 cookie 的內容
r = requests.get("https://www.wine-searcher.com/")

# 印出其中一個 cookie
print(r.cookies['_pxhd'])
415fa67947b951ada65358e82085bbe355f751b5cb889ab39a4fb583
fd2a7cd:ef775ae0-24fb-11eb-b1d7-b98207575929
```

10-2: 自訂 HTTP Headers、Cookies 及 Session 使用

In [2]:

```
# 汇入 requests 套件
import requests

# 自訂標頭
```

In [2]:

```
my_headers = {'user-agent': 'my-request/1.0.0'}
```

```
# 將自訂標頭加入 GET 請求中
r = requests.get('https://httpbin.org/get', headers = my_headers)
```

```
# 輸出網頁 HTML 原始碼
print(r.text)
{
    "args": {},
    "headers": {
        "Accept": "*/*",
        "Accept-Encoding": "gzip, deflate",
        "Host": "httpbin.org",
        "User-Agent": "my-request/1.0.0",
        "X-Amzn-Trace-Id": "Root=1-5fa00510-4d6c922478f6013f077ca3f0"
    },
    "origin": "114.43.211.202",
    "url": "https://httpbin.org/get"
}
```

In [4]:

```
# 自訂 cookie 格式
# jar = requests.cookies.RequestsCookieJar()
# jar.set("first_cookie", "hello", domain="httpbin.org")
# jar.set("second_cookie", "world", domain="httpbin.org")
```

```
# 自訂 cookie 格式
my_cookies = {
    "first_cookie": "hello",
    "second_cookie": "world"
}
```

```
# 將 cookie 加入 GET 請求
r = requests.get('https://httpbin.org/get', cookies = my_cookies)
```

```
# 輸出網頁 HTML 原始碼
print(r.text)
{
    "args": {},
    "headers": {
        "Accept": "*/*",
        "Accept-Encoding": "gzip, deflate",
        "Cookie": "first_cookie=hello; second_cookie=world",
        "Host": "httpbin.org",
        "User-Agent": "python-requests/2.24.0",
        "X-Amzn-Trace-Id": "Root=1-5fad5577-05db96474ef1b1a8586343ee"
    },
    "origin": "114.43.211.109",
    "url": "https://httpbin.org/get"
}
```

In [5]:

```
# 開啟 session
session = requests.Session()

# 透過 session 建立 cookie
session.get('https://httpbin.org/cookies/set/myCookieName/super_mario')

# 印出所有 cookies
print(session.cookies)

# 印出自訂的 cookie
print(session.cookies['myCookieName'])
<RequestsCookieJar[<Cookie myCookieName=super_mario for httpbin.org/>]>
super_mario
```

10-3: 解析回應內容 (HTML / JSON)

In [1]:

```
# 汇入套件
import requests, json

# 參考網址: https://data.taipei/#/application
r = requests.get('https://data.taipei/opendata/datalist/apiAccess?scope=resourceAquire&rid=35aa3c53-28fb-423c-91b6-2c22432d0d70&limit=100&offset=0');

# 將 json 轉成物件
obj = json.loads(r.text)

# 輸出對應節點的文字
print(obj['result']['results'][0]['ShowGroupName'])
print(obj['result']['results'][0]['Location'])
財團法人台北市文化基金會
台北市文化基金會（台北偶戲館樓上）

In [2]:
```

```
print("=" * 50)

# 輸出部分節點的文字
for index, result in enumerate(obj['result']['results']):
    if index == 5:
        break
    print(f"Location: {result['Location']}, ShowGroupName: {result['ShowGroupName']}")
=====
Location: 台北市文化基金會（台北偶戲館樓上）, ShowGroupName: 財團法人台北市文化基金會
Location: 中華民國象棋文化協會名揚分會, ShowGroupName: 中華民國象棋文化協會名揚分會
Location: 中華民國象棋文化協會名揚分會, ShowGroupName: 中華民國象棋文化協會名揚分會
Location: 名揚象棋會館, ShowGroupName: 中華民國象棋文化協會名揚分會
Location: 中華民國象棋文化協會名揚分會會館, ShowGroupName: 中華民國象棋文化協會名揚分會
```

```
In [3]:
```

```
# 印出當前 headers
```

```
print(r.headers)
{'Server': 'nginx', 'Date': 'Thu, 12 Nov 2020 15:32:48 GMT',
 'Content-Type': 'application/json', 'Content-Length': '337775',
 'Connection': 'keep-alive', 'Set-Cookie': 'PHPSES
SID=7u6bm7728ua09cuhou4o483nuq; path=/', 'Expires': 'Thu,
19 Nov 1981 08:52:00 GMT', 'Cache-Control': 'no-store, no-
cache, must-revalidate', 'Pragma': 'no-cache', 'Access-Con
trol-Allow-Origin': '*', 'Access-Control-Allow-Headers': 'X-Requeste
d-With, Content-Type, Accept, Origin, Authorizat
ion', 'Access-Control-Allow-Methods': 'GET, POST, PUT, DEL
E, PATCH, OPTIONS', 'X-Content-Type-Options': 'nosniff,
nosniff', 'X-XSS-Protection': '1;mode=block, 1;mode=block
', 'X-Frame-Options': 'SAMEORIGIN, SAMEORIGIN', 'Content-S
ecurity-Policy': 'frame-ancestors https://www.tgos.tw, fra
me-ancestors https://www.tgos.tw'}
```

In [4]:

```
# 印出 content-type
print(r.headers['Content-Type'])
application/json
```

Module 11. 套件 Beautiful Soup 4

11-1: 套件介紹及常用功能

套件介紹

Beautiful Soup 是一個 HTML parser，將 Document 轉換成一個樹狀結構，

提供簡單的函式來走訪、搜尋、修改分析此樹狀結構，支持 CSS 選擇器。

常用功能

我們主要用 BeautifulSoup 套件來作為網站解析的工具。

- find() 方法
- find_all() 方法
- select() 方法

In [1]:

```
import requests as rq
from bs4 import BeautifulSoup
import pprint as pp

# PTT NBA 版
url = "https://www.ptt.cc/bbs/NBA/index.html"

# 用 requests 的 get 方法把網頁抓下來
response = rq.get(url)

# 指定 lxml 作為解析器
soup = BeautifulSoup(response.text, "lxml")
```

In [2]:

```

# 第一個 <a></a>
print(soup.find("a"))
print("----分隔線----")
<a href="/bbs/" id="logo">批踢踢實業坊</a>
----分隔線----
In [ ]:

# 全部 <a></a>，此時回傳 list
pp pprint(soup.find_all("a"))
print("----分隔線----")
In [4]:

# 指定 list 某個元素的 html
print(soup.find_all("a")[2])
<a class="right small" href="/about.html">關於我們</a>

```

11-2: 解析 HTML，使用 CSS Selector 查找元素

```

# 汇入套件
from bs4 import BeautifulSoup
import requests

# PTT NBA 板
url = "https://www.ptt.cc/bbs/NBA/index.html"

# 用 requests 的 get 方法把網頁抓下來
response = requests.get(url)

# 指定 lxml 作為解析器
soup = BeautifulSoup(response.text, "lxml")
In [ ]:

# 搜尋所有 div，類別名稱為 r-ent，回傳為 list
posts = soup.find_all("div", class_ = "r-ent")

# 印出 list 內容
print(posts)

# 檢視物件的型態，在這裡是一個 ResultSet

```

```
print(type(posts))  
In [6]:  
'''  
一般我們使用迴圈將裡面的每一個元素再抓出來，  
準備收集作者 id  
'''  
  
# 建立一個空的 list 來放置作者 id  
author_ids = []  
  
# 搜尋 class name 為 r-ent 的 div 集合  
posts = soup.find_all("div", class_ = "r-ent")  
  
# 透過迭代方式一個一個將 author 摘取回來  
for post in posts:  
    # .extend() 是加入「資料集合」到 list 的尾端  
    author_ids.extend(post.find("div", class_ = "author"))  
  
print(author_ids)  
['wwf1588', 'pneumo', 'yoyoruru', 'hanson90244', 'thnlkj06  
65', 'FAYeeeeeee', 'azlbf', 'jackie0414', 'ivo88114', 'cl  
oud72426', 'ivo88114', 'asdfgh0920', 'LBJALA', 'ericf129',  
'rial', 'carotyao', 'huan0', 'Vedan', 'Vedan', 'Acetoxy',  
'namie810303']  
In [ ]:  
'''  
以下透過 CSS selector 取得元素，  
回傳格式為 list  
'''  
  
# 輸出 title  
print(soup.select('title'))  
  
# 輸出 a，  
print(soup.select('a'))  
  
# 透過 class 名稱取得元素  
print(soup.select("a.board"))  
  
# 透過 id 名稱取得元素
```

```
print(soup.select("#logo"))

In [8]:  
# 透過 attribute 取得元素  
print(soup.select('a[class="btn wide"]'))  
[<a class="btn wide" href="/bbs/NBA/index1.html">最舊</a>, <a class="btn wide" href="/bbs/NBA/index6500.html">< 上頁</a>, <a class="btn wide" href="/bbs/NBA/index.html">最新</a>]
```

11-3: 取出指定內容

```
# 汇入套件  
from bs4 import BeautifulSoup  
import requests

# PTT NBA 版  
url = "https://www.ptt.cc/bbs/NBA/index.html"

# 用 requests 的 get 方法把網頁抓下來  
response = requests.get(url)

# 指定 lxml 作為解析器  
soup = BeautifulSoup(response.text, "lxml")

In [2]:  
# 取得單一節點的文字內容 (select 會回傳 list)  
print(soup.select('title')[0].get_text())  
print(soup.select('a')[0].get_text())  
看板 NBA 文章列表 - 批踢踢實業坊  
批踢踢實業坊

In [3]:  
# 透過迭代取得所有 a 的文字內容  
for a in soup.select('a'):  
    print(a.get_text())
批踢踢實業坊  
看板 NBA  
關於我們
```

聯絡資訊

看板

精華區

最舊

< 上頁

下頁 >

最新

[花邊] 唐西奇斥資 270 萬美元 入手達拉斯豪宅

搜尋同標題文章

搜尋看板內 thn1kj0665 的文章

[討論] Aldridge 為什麼都不會想去強隊拿冠軍

搜尋同標題文章

搜尋看板內 seabox 的文章

[公告] 板規 v6.8

搜尋同標題文章

搜尋看板內 Vedan 的文章

[公告] 第一次被退文，可在三天後刪除退文

搜尋同標題文章

搜尋看板內 Vedan 的文章

[情報] Playoffs Schedule 19-20

搜尋同標題文章

搜尋看板內 Acetoxy 的文章

[公告] 季後賽條款於 10/14 零時解除

搜尋同標題文章

搜尋看板內 namie810303 的文章

In [4]:

```
# 透過迭代取得所有 a 的屬性 href
for a in soup.select('a'):
    if a.has_attr('href'):
        print(a['href'])

    else:
        print()
        print("=" * 50)
        print(f"連結[{a.get_text()}] 沒有 href 屬性")
        print("=" * 50)
        print()

/bbs/
/bbs/NBA/index.html
```

/about.html
/contact.html
/bbs/NBA/index.html
/man/NBA/index.html
/bbs/NBA/index1.html
/bbs/NBA/index6506.html

=====連結[下頁 >] 沒有 href 屬性=====

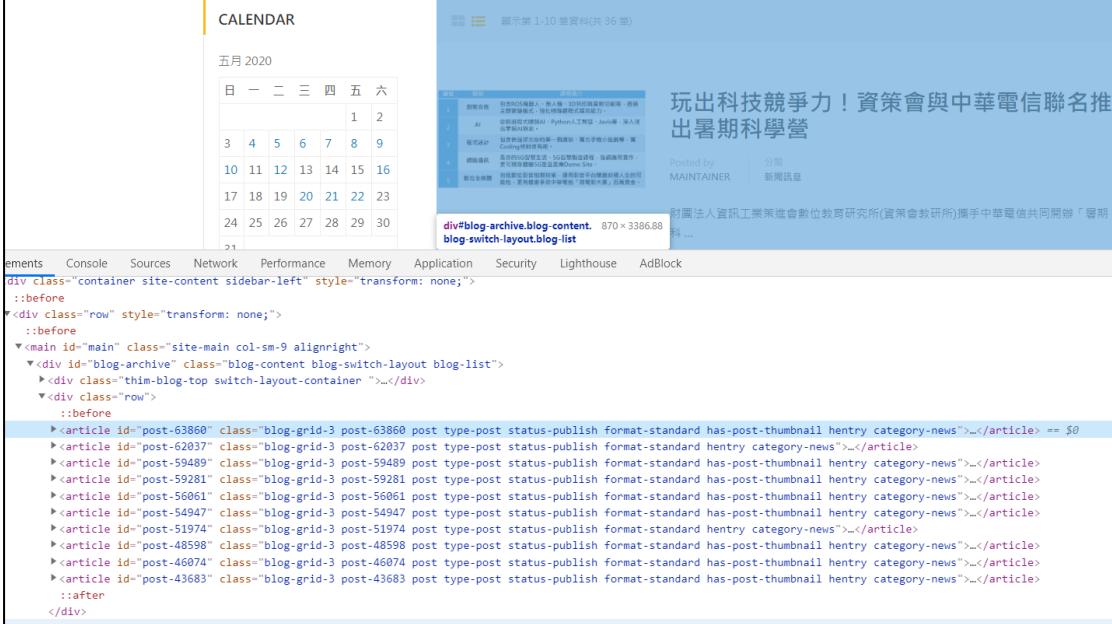
/bbs/NBA/index.html
/bbs/NBA/M.1605187585.A.CF9.html
/bbs/NBA/search?q=thread%3A%5B%E8%8A%B1%E9%82%8A%5D+%E5%94%90%E8%A5%BF%E5%A5%87%E6%96%A5%E8%B3%87270%E8%90%AC%E7%B8%8E%E5%85%83+%E5%85%A5%E6%89%8B%E9%81%94%E6%8B%89%E6%96%AF%E8%B1%AA%E5%AE%85
/bbs/NBA/search?q=author%3Athnlkj0665
/bbs/NBA/M.1605194928.A.0DE.html
/bbs/NBA/search?q=thread%3A%5B%E8%A8%8E%E8%AB%96%5D+Aldridge%E7%82%BA%E4%BB%80%E9%BA%BC%E9%83%BD%E4%B8%8D%E6%9C%83%E6%83%B3%E5%8E%BB%E5%BC%B7%E9%9A%8A%E6%8B%BF%E5%86%A0%E8%BB%8D
/bbs/NBA/search?q=author%3Aseabox
/bbs/NBA/M.1558698194.A.1DC.html
/bbs/NBA/search?q=thread%3A%5B%E5%85%AC%E5%91%8A%5D+%E6%9D%BF%E8%A6%8Fv6.8
/bbs/NBA/search?q=author%3AVedan
/bbs/NBA/M.1583725468.A.281.html
/bbs/NBA/search?q=thread%3A%5B%E5%85%AC%E5%91%8A%5D+%E7%AC%4%B8%80%E6%AC%A1%E8%A2%AB%E9%80%80%E6%96%87%EF%BC%8C%E5%8F%AF%E5%9C%A8%E4%B8%89%E5%A4%A9%E5%BE%8C%E5%88%AA%E9%99%A4%E9%80%80%E6%96%87
/bbs/NBA/search?q=author%3AVedan
/bbs/NBA/M.1597695044.A.F65.html
/bbs/NBA/search?q=thread%3A%5B%E6%83%85%E5%A0%B1%5D+Playoff+Schedule+19%E2%80%9320
/bbs/NBA/search?q=author%3AAcetoxy

/bbs/NBA/M.1602599945.A.C45.html
/bbs/NBA/search?q=thread%3A%5B%E5%85%AC%E5%91%8A%5D+%E5%A
D%A3%E5%BE%8C%E8%B3%BD%E6%A2%9D%E6%AC%BE%E6%96%BC10%2F14%
E9%9B%B6%E6%99%82%E8%A7%A3%E9%99%A4
/bbs/NBA/search?q=author%3Anamie810303

Module 12. 案例實作

以 iiiedu 新聞訊息為例。

12-1: 分析頁面資訊結構



The screenshot shows a news archive page for May 2020. The left sidebar contains a calendar for May. The main content area displays a list of news articles. Each article is represented by a card with a thumbnail, title, and a brief summary. The first article's details are visible in the browser's developer tools:

```
div#blog-archive.blog-content.870x3386.88  
blog-switch-layout.blog-list
```

The developer tools also show the DOM structure, which consists of multiple `article` elements, each representing a news item.

圖：列表由若干個 `article` 組成



This screenshot shows the same news archive page, but the first article has been selected, highlighting its content. The developer tools show the DOM structure for this specific article, which includes header, content, author information, and meta-data.

```
iv#blog-archive.blog-content.blog-switch-layout.blog-list  
<div class="thim-blog-top switch-layout-container">...</div>  
<div class="row">  
  :before
```

The highlighted part of the code corresponds to the first article's content, showing how most of the article's information is contained within a single `article` element.

圖：以第一則為例，大部分資訊包在 `article` 裡面

12-2: 使用套件解析並取出新聞清單

In [1]:

```
# 匯入套件
from bs4 import BeautifulSoup
import requests
```

In [2]:

```
# 取得新聞列表
url = "https://www.iiiedu.org.tw/category/news/"
```

```
# 用 requests 的 get 方法把網頁抓下來
response = requests.get(url)
```

```
# 指定 lxml 作為解析器
soup = BeautifulSoup(response.text, "lxml")
```

In [3]:

```
# 取得 a 的文字
for a in soup.select('article div.entry-content header.entry-header h2.entry-title a'):
    print(a.get_text())
台南 AI 人才培訓 深耕在地產業
解決科技人才斷層 經濟部工業局攜手各界響應 2020 SEMICON Taiwan 攜手
人才循環大聯盟培育高科技人才
資策會協辦全國臨床診療技能競賽 虛擬診療訓練系統 助疫情期間培訓不中斷
台南 AI 培訓 跨域學習獲大廠聘用
網站維護通知
2020 台北遊戲開發者論壇熱鬧登場！ 與 Twitch 官方首度直播合作，精彩演
講線上看
資策會聯手彰化縣推科技反毒！新科技反毒行動巡迴車開跑
宅生活發威！VR 虛擬博物館為疫情寒冬帶來正能量
新冠擋不了！跨國怎麼學 AI？研華科技與資策會共辦「臺泰 AI 學院」
全民防疫 E 起來 臺灣智慧學習產業鏈不缺席
```

12-3: 遞迴取出新聞頁面內容

```
In [1]:  
# 汇入套件  
from bs4 import BeautifulSoup  
import requests  
  
# 取得新聞列表  
url = "https://www.iiiedu.org.tw/category/news/"  
  
# 用 requests 的 get 方法把網頁抓下來  
response = requests.get(url)  
  
# 指定 lxml 作為解析器  
soup = BeautifulSoup(response.text, "lxml")  
  
In [2]:  
# 建立 list 來放置新聞連結  
list_news = []  
  
# 取得 a 的連結  
for a in soup.select('article div.entry-content header.entry-header h2.entry-title a'):  
    list_news.append(a['href'])  
  
# 走訪每一個 a link，並印出網頁內文  
for index, link in enumerate(list_news):  
    res = requests.get(link)  
    soup_link = BeautifulSoup(res.text, "lxml")  
    print(soup_link.select('article div.entry-content')[0].get_text())  
    print("=" * 50)
```

Module 13. 案例實作

以 臺北市資料大平台 為例，我們只要修改 offset 的值，從 0 改成 100, 200, ...。

13-1: 取出 JSON 內容

13-2: 解析 JSON 內容

13-3: 參數取代變換

```
In [1]:  
# 匯入套件  
import requests, json  
  
In [2]:  
# 參考網址: https://data.taipei/#/application  
r = requests.get('https://data.taipei/opendata/datalist/apiAccess?scope=resourceAquire&rid=35aa3c53-28fb-423c-91b6-2c22432d0d70&limit=100&offset=0');  
  
# 將 json 轉成物件  
obj = json.loads(r.text)  
  
# 輸出對應節點的文字  
print(obj['result']['results'][0]['ShowGroupName'])  
print(obj['result']['results'][0]['Location'])  
財團法人台北市文化基金會  
台北市文化基金會（台北偶戲館樓上）  
  
In [3]:  
print("=" * 50)  
  
# 輸出部分節點的文字  
for index, result in enumerate(obj['result']['results']):  
    if index == 5:  
        break  
    print(f"Location: {result['Location']}, ShowGroupName: {result['ShowGroupName']}")
```

```
=====
Location: 台北市文化基金會（台北偶戲館樓上）， ShowGroupName: 財團法人台北市文化基金會
Location: 中華民國象棋文化協會名揚分會， ShowGroupName: 中華民國象棋文化協會名揚分會
Location: 中華民國象棋文化協會名揚分會， ShowGroupName: 中華民國象棋文化協會名揚分會
Location: 名揚象棋會館， ShowGroupName: 中華民國象棋文化協會名揚分會
Location: 中華民國象棋文化協會名揚分會會館， ShowGroupName: 中華民國象棋文化協會名揚分會
```

In [4]:

```
# 參考網址: https://data.taipei/#/application
r = requests.get('https://data.taipei/opendata/datalist/apiAccess?scope=resourceAquire&rid=35aa3c53-28fb-423c-91b6-2c22432d0d70&limit=100&offset=100');

# 將 json 轉成物件
obj = json.loads(r.text)

# 輸出對應節點的文字
print(obj['result']['results'][0]['ShowGroupName'])
print(obj['result']['results'][0]['Location'])
台北市基督徒救世會
網路活動
```

In [5]:

```
print("=" * 50)

# 輸出部分節點的文字
for index, result in enumerate(obj['result']['results']):
    if index == 5:
        break
    print(f"Location: {result['Location']}, ShowGroupName: {result['ShowGroupName']}")
```

```
=====
Location: 網路活動, ShowGroupName: 台北市基督徒救世會
Location: 網路活動, ShowGroupName: 台北市基督徒救世會
Location: 台北市青島西路 7 號 4 樓, ShowGroupName: 台北女青年會
```

```
Location: 臺中市屯區藝文中心演藝廳, ShowGroupName: 童歡藝術有限公司
```

```
Location: 臺北市中山區新生北路二段 58 巷 36 號四樓, ShowGroupName: 紙文物維護學會
```

```
In [6]:
```

```
# 參考網址: https://data.taipei/#/application
r = requests.get('https://data.taipei/opendata/datalist/apiAccess?scope=resourceAquire&rid=35aa3c53-28fb-423c-91b6-2c22432d0d70&limit=100&offset=300');

# 將 json 轉成物件
obj = json.loads(r.text)

# 輸出對應節點的文字
print(obj['result']['results'][0]['ShowGroupName'])
print(obj['result']['results'][0]['Location'])
易禧創意
易禧創意設計中心
```

```
In [7]:
```

```
print("=" * 50)

# 輸出部分節點的文字
for index, result in enumerate(obj['result']['results']):
    if index == 5:
        break
    print(f"Location: {result['Location']}, ShowGroupName: {result['ShowGroupName']}")
=====
Location: 易禧創意設計中心, ShowGroupName: 易禧創意
Location: 台北市中山區中山北路二段 18 號, ShowGroupName: 台北之家
```

Module 14. 套件 Selenium (初階)

14-1: 解析 Selenium 、 WebDriver 與 Browser 連動關係

- Selenium 是一種 web automatic testing 工具，操作網頁表單資料、點選按鈕或連結、取得網頁內容並進行檢驗。selenium 是一個套件，可以藉此操作 WebDriver 。
- WebDriver 是用來執行並操作瀏覽器的一個 API 介面，程式透過呼叫 WebDriver 來直接對瀏覽器進行操作，實作則決定於所選用的瀏覽器 driver ，例如有 FirefoxDriver, ChromeDriver, InternetExporeDriver 等 。
- Browser 經由 WebDriver 啟動，讓 Selenium 進行操作，完成網頁自動化的工作 。

14-2: 自訂 Selenium 啟動設定 (移除資訊列、全螢幕顯示、...)

In [2]:

```
# 決入相關套件
from selenium import webdriver
from time import sleep

'''
selenium 啓動 Chrome 的進階配置參數
參考網址 : https://stackoverflow.max-everyday.com/2019/12/se
lenium-chrome-options/
'''

# 啟動瀏覽器工具的選項
options = webdriver.ChromeOptions()
```

```

# options.add_argument("--headless") #不開啟實體
瀏覽器背景執行
options.add_argument("--start-maximized") #最大化視
窗
options.add_argument("--incognito") #開啟無痕模
式
options.add_argument("--disable-popup-blocking") #禁用彈出
攔截

# 使用 Chrome 的 WebDriver
driver = webdriver.Chrome(options = options)

# 螢幕最大化
driver.maximize_window()

# 開啟 Google 首頁
driver.get("https://tw.yahoo.com")

# 休眠幾秒
sleep(5)

# 關閉瀏覽器
driver.quit()

```

方法	說明
get_window_position()	取得瀏覽器視窗左上角位置
set_window_position(x, y)	設定瀏覽器視窗左上角位置
get_window_size()	取得瀏覽器視窗大小
set_window_size(x, y)	設定瀏覽器視窗大小
maximize_window()	將瀏覽器視窗最大化
minimize_window()	將瀏覽器視窗最小化

圖：還操控瀏覽器的位置與大小

14-3: 瀏覽器控制與取得網頁原始碼 (get / quit /

```
page_source )
```

In []:

```
# 汇入相關套件
from selenium import webdriver
from time import sleep

'''
selenium 啓動 Chrome 的進階配置參數
參考網址：https://stackoverflow.max-everyday.com/2019/12/sele
nium-chrome-options/
'''

# 啟動瀏覽器工具的選項
options = webdriver.ChromeOptions()
# options.add_argument("--headless") #不開啟實體
瀏覽器背景執行
options.add_argument("--start-maximized") #最大化視
窗
options.add_argument("--incognito") #開啟無痕模
式
options.add_argument("--disable-popup-blocking") #禁用彈出
攔截

# 使用 Chrome 的 WebDriver
driver = webdriver.Chrome(options = options)

# 開啟 數位時代 首頁
driver.get("https://www.bnnext.com.tw/")

# 取得檢視原始碼的內容
html = driver.page_source

# 印出 html
print(html)

# 關閉瀏覽器
driver.quit()
```

屬性	說明
name	瀏覽器名稱
title	目前開啟網頁之標題
current_url	目前開啟網頁之 URL
page_source	目前開啟網頁之原始碼
session_id	網頁連線 id
capabilities	瀏覽器功能設定

圖 : driver.page_source 外，還有其它可以使用

Module 15. 套件 Selenium (中階)

15-1: 進階控制 (alert / frame)

```
In [ ]:  
from selenium import webdriver  
from selenium.common.exceptions import TimeoutException  
from selenium.webdriver.support.ui import WebDriverWait  
from selenium.webdriver.support import expected_conditions  
as EC  
from selenium.webdriver.common.by import By  
from time import sleep  
  
# 使用 Chrome 的 WebDriver  
driver = webdriver.Chrome()  
  
# 開啟網頁  
driver.get("http://crptransfer.moe.gov.tw/")  
  
# 跳出 alert 視窗  
driver.execute_script("window.alert('這是我們自訂的彈跳視窗')")  
  
# 等個幾秒  
sleep(3)  
  
# 點選彈出裡面的確定按鈕  
driver.switch_to.alert.accept()  
  
# 等個幾秒  
sleep(3)  
  
# 關閉瀏覽器  
driver.quit()
```

15-2: 如何查找頁面元素 (ID / Class / Tag / CSS Selector / ...)

方法	說明
find_element(by, value)	使用 by 指定之方法取得第一個符合 value 的元素
find_element_by_class_name(name)	傳回符合指定 class 名稱之元素
find_elements_by_class_name(name)	傳回符合指定 class 名稱之元素串列
find_element_by_css_selector(selector)	傳回符合指定 CSS 選擇器名稱之元素
find_elements_by_css_selector(selector)	傳回符合指定 CSS 選擇器名稱之元素串列
find_element_by_id(id)	傳回符合指定 id 之元素
find_elements_by_id(id)	傳回符合指定 id 之元素串列
find_element_by_link_text(text)	傳回符合指定超連結文字之元素
find_elements_by_link_text(text)	傳回符合指定超連結文字之元素串列
find_element_by_partial_link_text(text)	傳回符合部分指定超連結文字之元素
find_elements_by_partial_link_text(text)	傳回符合部分指定超連結文字之元素串列
find_element_by_name(name)	傳回符合指定元素名稱之元素
find_elements_by_name(name)	傳回符合指定元素名稱之元素串列
find_element_by_tag_name(tag)	傳回符合指定標籤名稱之元素
find_elements_by_tag_name(tag)	傳回符合指定標籤名稱之元素串列

圖：列出一些參考用的方法

In [14] :

```
from selenium import webdriver
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions
as EC
from selenium.webdriver.common.by import By
from time import sleep

# 使用 Chrome 的 WebDriver
driver = webdriver.Chrome()

# 開啟網頁
driver.get("http://crptransfer.moe.gov.tw/")

# 尋找網頁中的搜尋框
inputElement = driver.find_element_by_name("SN")

# 在搜尋框中輸入文字
inputElement.send_keys("人帥真好")
```

```

# 睡個幾秒
sleep(2)

# 送出搜尋
inputElement.submit()

# 搜尋結果的 CSS Selector
cssSelector = "body > table > tbody > tr:nth-child(1) > td
> main > article > div > table > tbody > tr:nth-child(2) >
td"

try:
    # 等待網頁搜尋結果
    WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.CSS_SELECTOR, cssSelector)))

    # 取得第一頁搜尋結果 (型態為 list)
    result = driver.find_elements_by_css_selector(cssSelector)

    # 輸出想要爬取的文字
    print(result[0].text)

    # 睡個幾秒
    sleep(3)

    # 關閉瀏覽器
    driver.quit()

except TimeoutException:
    print('等待逾時！')
20200320172
4710763
20200320172

```

15-3: 動作控制 (click / send_keys)

In [9]:

```
from selenium import webdriver
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions
as EC
from selenium.webdriver.common.by import By
from time import sleep

# 使用 Chrome 的 WebDriver
driver = webdriver.Chrome()

# 開啟網頁
driver.get("https://www.104.com.tw/jobs/main/")

# 尋找網頁中的搜尋框
inputElement = driver.find_element_by_id("ikeyword")

# 在搜尋框中輸入文字
inputElement.send_keys("python")

# 睡個幾秒
sleep(2)

# 按鈕選擇器
cssSelectorBtn = "button.btn.btn-primary.js-formCheck"

try:
    # 等待網頁搜尋結果
    WebDriverWait(driver, 10).until(EC.visibility_of_element_located((By.CSS_SELECTOR, cssSelectorBtn)))

    # 按鈕 Web Element (型態為 list)
    btn = driver.find_elements_by_css_selector(cssSelectorBtn)
```

```
# 按下按鈕
btn[0].click()

# 睡個幾秒
sleep(3)

# 關閉瀏覽器
driver.quit()

except TimeoutException:
    print('等待逾時！')
```

Module 16. 套件 Selenium (高階)

16-1: 等待 (WebDriverWait)

等待有分幾種：

- 強制等待
 - 通常泛指 sleep() 函式
- 隱性等待
 - 設置了一個最長等待時間，如果在規定時間內網頁加載完成，則執行下一步，否則一直等到時間截止，然後執行下一步。注意這裡有一個弊端，那就是程序會一直等待整個頁面加載完成。
- 顯性等待
 - 配合 until() 和 until_not() 方法，就能夠根據判斷條件而進行靈活地等待了。它主要的意思就是：如果條件成立了，則執行下一步，否則繼續等待，直到超過設置的最長時間，直到拋出 TimeoutException 。

In [2]:

```
# 汇入套件
from selenium import webdriver
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions
as EC
from selenium.webdriver.common.by import By
from time import sleep
```

In [8]:

```
'''  
強制等待  
'''  
  
# 使用 Chrome 的 WebDriver  
driver = webdriver.Chrome()  
  
# 走訪網址  
driver.get('https://tw.yahoo.com/')  
  
# 印出網址  
print(driver.current_url)  
  
# 強制等待 3 秒再往下一步 (下一行程式)  
sleep(3)  
  
# 關閉瀏覽器  
driver.quit()  
https://tw.yahoo.com/
```

In [9]:

```
'''  
隱性等待  
'''  
  
# 使用 Chrome 的 WebDriver  
driver = webdriver.Chrome()  
  
# 走訪網址  
driver.get('https://tw.yahoo.com/')  
  
# 印出網址  
print(driver.current_url)  
  
# 最高等 30 秒  
driver.implicitly_wait(30)  
  
# 關閉瀏覽器  
driver.quit()  
https://tw.yahoo.com/
```

In [3]:

```

''''
顯性等待
''''

# 使用 Chrome 的 WebDriver
driver = webdriver.Chrome()

# 走訪網址
driver.get('https://www.youtube.com/?gl=TW')

try:
    # 滿足條件 (10 秒內找到元素) , 則往下一步
    WebDriverWait(driver, 10).until(EC.presence_of_element
_located( (By.LINK_TEXT, '首頁') ) )

    # 印出首頁連結
    print(driver.find_element_by_link_text('首頁').get_attribute('href'))

except TimeoutException:
    print('等待逾時！')
finally:
    # 關閉瀏覽器
    driver.quit()

```

<https://www.youtube.com/>

16-2: 期待狀況 (Expected Condition)

通常與 WebDriverWait 配合使用，動態等待頁面上元素出現或者消失。

- title_is
 - 判斷當前頁面的 title 是否精確等於預期
- title_contains
 - 判斷當前頁面的 title 是否包含預期字符串

- **presence_of_element_located**
 - 判斷某個元素是否被加到了 dom 樹里，並不代表該元素一定可見
- **visibility_of_element_located**
 - 判斷元素是否可見。可見代表元素非隱藏，並且元素的寬和高都不等於 0
- **presence_of_all_elements_located**
 - 判斷是否至少有 1 個元素存在於 DOM tree 中。舉個例子，如果頁面上有 n 個元素的 class 都是'col-md-3'，那麼只要有 1 個元素存在，這個方法就返回 True
- **text_to_be_present_in_element**
 - 判斷某個元素中的 text 是否包含了預期的字串
- **text_to_be_present_in_element_value**
 - 判斷某個元素中的 value 屬性是否包含了預期的字串
- **frame_to_be_available_and_switch_to_it**
 - 判斷該 frame 是否可以 switch 進去，如果可以的話，返回 True 並且 switch 進去，否則返回 False
- **invisibility_of_element_located**
 - 判斷某個元素中是否存在於 DOM tree 或不可見
- **element_to_be_clickable**
 - 判斷某個元素中是否可見並且是 enable 的，這樣的話才叫 clickable

- staleness_of
 - 等某個元素從 dom 樹中移除，注意，這個方法也是返回 True 或 False
- element_to_be_selected
 - 判斷某個元素是否被選中了，一般用在下拉列表
- element_selection_state_to_be
 - 判斷某個元素的選中狀態是否符合預期
- element_located_selection_state_to_be
 - 跟上面的方法作用一樣，只是上面的方法傳入定位到的 element，而這個方法傳入 locator
- alert_is_present
 - 判斷頁面上是否存在 alert，這是個老問題，很多同學會問到

16-3: 根據條件 (By)

- By.ID = "id"
- By.XPATH = "xpath"
- By.LINK_TEXT = "link text"
- By.PARTIAL_LINK_TEXT = "partial link text"
- By.NAME = "name"
- By.TAG_NAME = "tag name"

- By.CLASS_NAME = "class name"
- By.CSS_SELECTOR = "css selector"

Module 17. 案例實作

17-1: 如何使用 Selenium 登入臉書

```
In [1]:  
  
# 匯入套件  
from selenium import webdriver  
from selenium.common.exceptions import TimeoutException  
from selenium.webdriver.support.ui import WebDriverWait  
from selenium.webdriver.support import expected_conditions  
as EC  
from selenium.webdriver.common.by import By  
from time import sleep  
  
'''  
selenium 啓動 Chrome 的進階配置參數  
參考網址：https://stackoverflow.max-everyday.com/2019/12/selenium-chrome-options/  
'''  
  
# 啟動瀏覽器工具的選項  
options = webdriver.ChromeOptions()  
# options.add_argument("--headless") #不開啟實體  
瀏覽器背景執行  
options.add_argument("--start-maximized") #最大化視  
窗  
options.add_argument("--incognito") #開啟無痕模  
式  
options.add_argument("--disable-popup-blocking") #禁用彈出  
攔截  
  
# 使用 Chrome 的 WebDriver (含 options)  
driver = webdriver.Chrome( options = options )  
  
# 走訪網址  
driver.get('https://www.facebook.com/')  
  
# 輸入帳號  
inputEmail = driver.find_element_by_id('email')
```

```

inputEmail.send_keys("你的帳號")

# 輸入密碼
inputPwd = driver.find_element_by_id('pass')
inputPwd.send_keys("你的密碼")

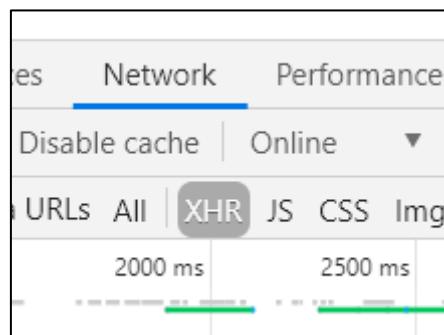
# 按下登入/送出鈕
btnSubmit = driver.find_element_by_css_selector('button[type="submit"] [name="login"]')
btnSubmit.click()

# 強制等待 10 秒
sleep(10)

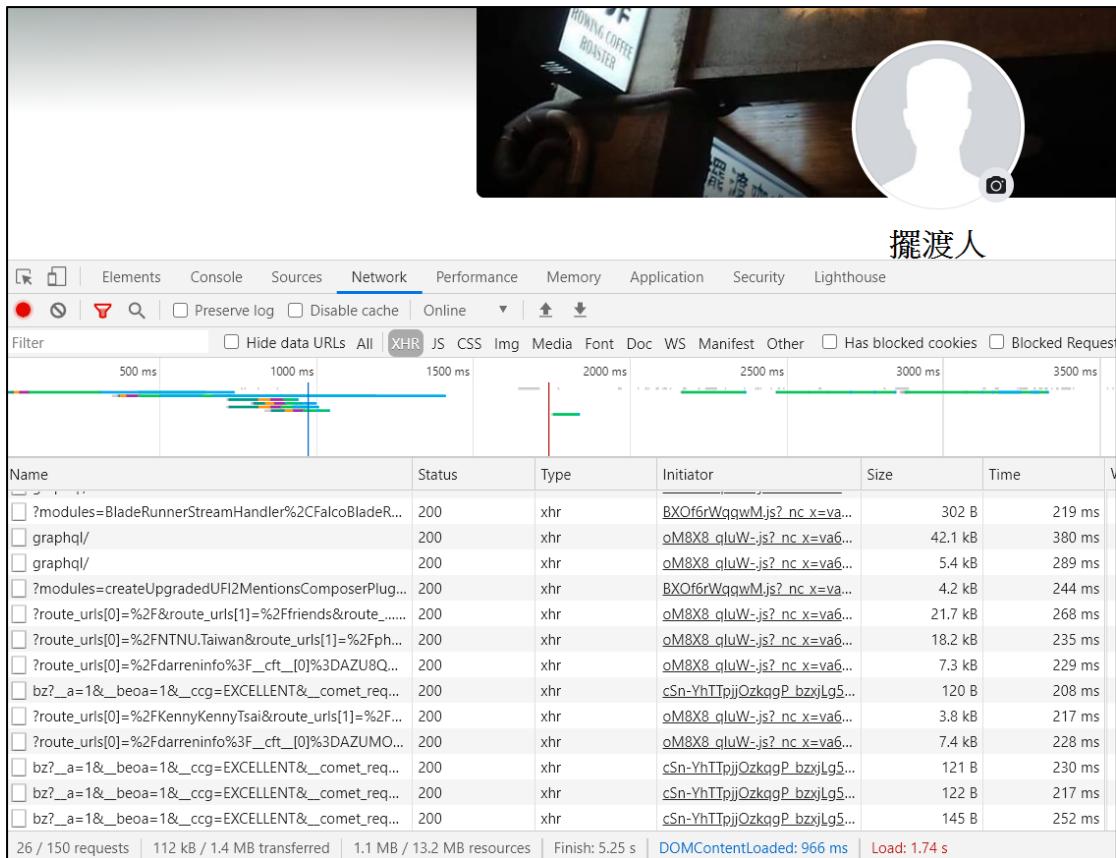
# 關閉瀏覽器
driver.quit()

```

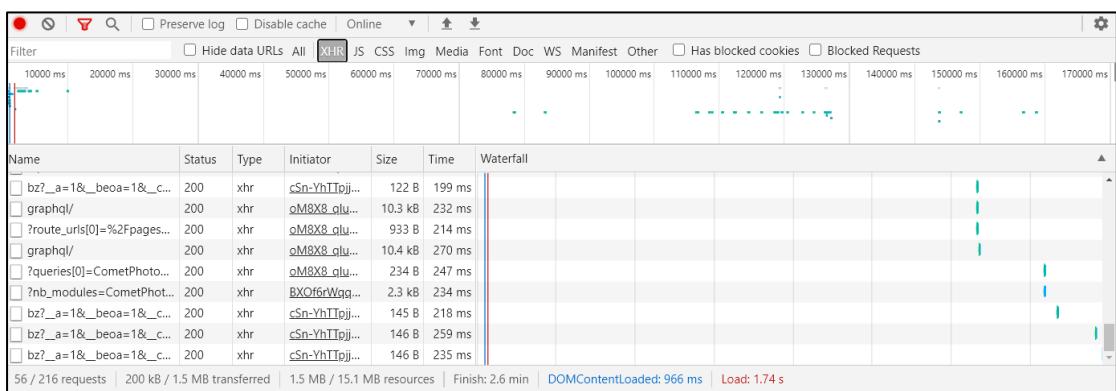
17-2: 如何使用 Chrome Developer Tools 分析臉書個人動態牆



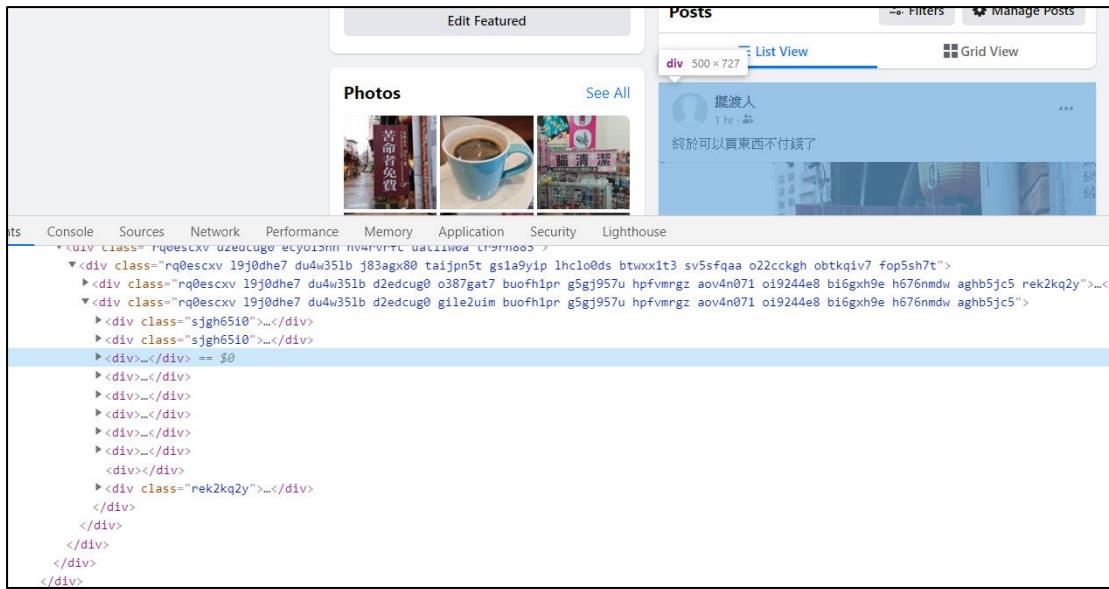
圖：進入 Network 面板，選擇 XHR (XMLHttpRequest)



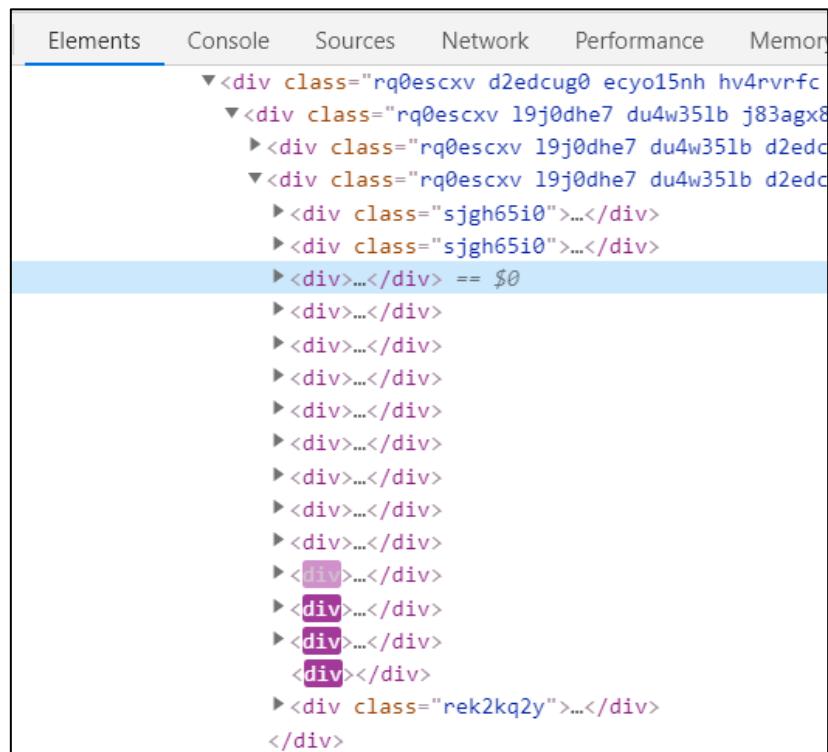
圖：當我們重新整理頁面的時候，Finish 時間如圖片



圖：當我們往下滾動，發現網頁元素會動態生成，Finish 時間會持續成長



圖：當我們選定某個 div，看似目前有 7 個左右



圖：往下繼續滾動的時候，元素確實自動生成了

17-3: 如何使用擷取臉書個人動態牆內容

In [3]:

```

# 匯入套件
from selenium import webdriver
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions
as EC
from selenium.webdriver.common.by import By
from time import sleep
import json

'''
selenium 啓動 Chrome 的進階配置參數
參考網址：https://stackoverflow.max-everyday.com/2019/12/se lenium-chrome-options/
'''

# 啟動瀏覽器工具的選項
options = webdriver.ChromeOptions()
# options.add_argument("--headless") #不開啟實體
瀏覽器背景執行
options.add_argument("--start-maximized") #最大化視
窗
options.add_argument("--incognito") #開啟無痕模
式
options.add_argument("--disable-popup-blocking") #禁用彈出
攔截

# 使用 Chrome 的 WebDriver
driver = webdriver.Chrome(options = options)

# 螢幕最大化
driver.maximize_window()

# 放置 FB 個人發文的基本資訊
listPost = []

'''

以下定義了一些方法，作為爬蟲流程
'''
```

```

# 前往 FB
def begin():
    # 走訪網址
    driver.get('https://www.facebook.com/')

# 登入
def login():
    # 輸入帳號
    inputEmail = driver.find_element_by_id('email')
    inputEmail.send_keys("你的帳號")

    # 輸入密碼
    inputPwd = driver.find_element_by_id('pass')
    inputPwd.send_keys("你的密碼")

    # 按下登入/送出鈕
    btnSubmit = driver.find_element_by_css_selector('button[type="submit"] [name="login"]')

    btnSubmit.click()

    # 強制等待
    sleep(3)

# 走訪個人頁
def visit():
    driver.get('https://www.facebook.com/darreninfo/')

# 滾動頁面
def scroll():
    # 瀏覽器內部的高度
    innerHeightOfWindow = 0

    # 當前捲動的量(高度)
    totalOffset = 0

    # 每捲一次，休息幾秒

```

```

sleepingSecond = 1

# 在捲動到沒有元素動態產生前，持續捲動
while totalOffset <= innerHeightOfWindow:
    # 每次移動高度
    totalOffset += 300;

    # 捲動的 js code
    js_scroll = "(function (){{window.scrollTo({{top: {}, behavior: 'smooth' }});}})()".format(totalOffset)

    # 執行 js code
    driver.execute_script(js_scroll)

    # 強制等待
    sleep(sleepingSecond)

    # 透過執行 js 語法來取得捲動後的高度
    innerHeightOfWindow = driver.execute_script('return window.document.documentElement.scrollHeight;');

    # 強制等待
    sleep(sleepingSecond)

    # 印出捲動距離
    print("innerHeightOfWindow: {}, totalOffset: {}".format(innerHeightOfWindow, totalOffset))

    # 為了實驗功能，捲動超過一定的距離，就結束程式
    if totalOffset > 8000:
        break

# 分析元素內容
def parse():
    # 確認當前動個人態牆的發文數量
    cssSelectorPost = 'div[data-ad-comet-preview="message"] span[dir="auto"]'

```

```

try:
    # 顯性等待個人發文的元素出現
    #WebDriverWait(driver, 30).until(EC.visibility_of_
element_located((By.CSS_SELECTOR, cssSelectorPost) ) )

    # 取得元素
    div = driver.find_elements_by_css_selector(cssSelec-
torPost)

    # 將每個發文的內容擷取出來
    for index, elm in enumerate(div):
        print("index: {}".format(index))
        print("text: {}".format(elm.text))
        print()

    # 將資料新增到 list 當中
    listPost.append({
        "index": index,
        "text":elm.text
    })

    # 將放置發文的 list ,以 JSON 格式存入檔案
    fp = open("FB.json", "w",encoding='UTF-8')
    fp.write( json.dumps(listPost, ensure_ascii=False)
)
    fp.close()

except TimeoutException:
    print('等待逾時！')

# 關閉瀏覽器
def end():
    # 關閉瀏覽器
    driver.quit()

# 主程式

```

```
if __name__ == '__main__':
    begin()
    login()
    visit()
    scroll()
    parse()
    end()
```

Module 18. 股票案例實作及視覺化

18-1: 使用套件取得股價、量等資料

網址：<https://www.lfd.uci.edu/~gohlke/pythonlibs/>

TA-Lib: a wrapper for the TA-LIB Technical Analysis Library.

[TA_Lib-0.4.18-cp38-cp38-win_amd64.whl](#)
[TA_Lib-0.4.18-cp38-cp38-win32.whl](#)
[TA_Lib-0.4.18-cp37-cp37m-win_amd64.whl](#)
[TA_Lib-0.4.18-cp37-cp37m-win32.whl](#)
[TA_Lib-0.4.18-cp36-cp36m-win_amd64.whl](#)
[TA_Lib-0.4.18-cp36-cp36m-win32.whl](#)
[TA_Lib-0.4.17-cp35-cp35m-win_amd64.whl](#)
[TA_Lib-0.4.17-cp35-cp35m-win32.whl](#)
[TA_Lib-0.4.17-cp34-cp34m-win_amd64.whl](#)
[TA_Lib-0.4.17-cp34-cp34m-win32.whl](#)
[TA_Lib-0.4.17-cp27-cp27m-win_amd64.whl](#)
[TA_Lib-0.4.17-cp27-cp27m-win32.whl](#)

圖：依電腦規格選擇檔案下載

說明

其中 cp37 表示 python3.7，win32 表示 32 位，amd64 表示 64 位，可以自行下載對應的檔案，並放置專案資料夾（例如 notebooks）；本案例使用 `TA_Lib-0.4.18-cp37-cp37m-win_amd64.whl`。

```
Microsoft Windows [版本 10.0.18363.836]
(c) 2019 Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\Owner\notebooks>C:/Users/Owner/anaconda3/Scripts/activate

(base) C:\Users\Owner\notebooks>conda activate base

(base) C:\Users\Owner\notebooks>pip install TA_Lib-0.4.18-cp37-cp37m-win_amd64.whl
Processing c:\users\owner\notebooks\ta_lib-0.4.18-cp37-cp37m-win_amd64.whl
Installing collected packages: TA-Lib
Successfully installed TA-Lib-0.4.18

(base) C:\Users\Owner\notebooks>
```

圖：至 Terminal 安裝下載好的套件

```
In [ ]:

''''

載入預設套件
'''

# basic
import numpy as np
import pandas as pd

# get data
import pandas_datareader as pdr

# visual
import matplotlib.pyplot as plt
import mpl_finance as mpf
import seaborn as sns

# time
import datetime as datetime

In [2]:

# 使用 pandas_datareader 抓取某個股票，這邊使用台積電(2330)
start = datetime.datetime(2020,1,1)

# 取得 data frame
df_2330 = pdr.DataReader('2330.TW', 'yahoo', start=start)
```

```
# 預覽 data frame
```

```
df_2330
```

	High	Low	Open	Close	Volume	Adj Close
Date						
2020-01-02	339.0	332.5	332.5	339.0	31754120.0	328.295837
2020-01-03	345.0	335.5	344.0	339.5	41811268.0	328.780060
2020-01-06	334.5	332.0	333.0	332.0	45343057.0	321.516846
2020-01-07	333.0	326.5	332.5	329.5	50879181.0	319.095795
2020-01-08	333.0	325.0	325.0	329.5	37567748.0	319.095795
...
2021-03-22	598.0	586.0	588.0	593.0	34089033.0	593.000000
2021-03-23	603.0	593.0	597.0	594.0	29374037.0	594.000000
2021-03-24	582.0	571.0	571.0	576.0	107253959.0	576.000000
2021-03-25	581.0	570.0	572.0	575.0	67833354.0	575.000000
2021-03-26	592.0	580.0	582.0	590.0	36829858.0	590.000000

297 rows × 6 columns

18-2: 繪製畫出價量及 K 棒

In []:

```
'''
```

載入預設套件

```
'''
```

```
# basic
```

```
import numpy as np
```

```
import pandas as pd
```

```
# get data
```

```
import pandas_datareader as pdr
```

```
# visual
import matplotlib.pyplot as plt
import mpl_finance as mpf
import seaborn as sns

#time
import datetime as datetime
In [2]:
# 使用pandas_datareader 抓取某個股票，這邊使用台積電(2330)
start = datetime.datetime(2020,1,1)

# 取得 data frame
df_2330 = pdr.DataReader('2330.TW', 'yahoo', start=start)

# 預覽 data frame
df_2330
```

	High	Low	Open	Close	Volume	Adj Close
Date						
2020-01-02	339.0	332.5	332.5	339.0	31754120.0	328.295837
2020-01-03	345.0	335.5	344.0	339.5	41811268.0	328.780060
2020-01-06	334.5	332.0	333.0	332.0	45343057.0	321.516846
2020-01-07	333.0	326.5	332.5	329.5	50879181.0	319.095795
2020-01-08	333.0	325.0	325.0	329.5	37567748.0	319.095795
...
2021-03-22	598.0	586.0	588.0	593.0	34089033.0	593.000000
2021-03-23	603.0	593.0	597.0	594.0	29374037.0	594.000000
2021-03-24	582.0	571.0	571.0	576.0	107253959.0	576.000000
2021-03-25	581.0	570.0	572.0	575.0	67833354.0	575.000000
2021-03-26	592.0	580.0	582.0	590.0	36829858.0	590.000000

297 rows × 6 columns

In [3]:

```
# 設定 index 格式
df_2330.index = df_2330.index.format(formatter=lambda x:
x.strftime('%Y-%m-%d'))

fig = plt.figure(figsize=(24, 8))

ax = fig.add_subplot(1, 1, 1)

# 設定 x 軸標籤
ax.set_xticks(range(0, len(df_2330.index), 10))
ax.set_xticklabels(df_2330.index[::-1])

# 使用mpf.candlestick2_ochl來繪製 K 線圖
mpf.candlestick2_ochl(ax, df_2330['Open'], df_2330['Close'],
'df_2330['High'],
```

```
df_2330['Low'], width=0.6, colorup='r',
colordown='g', alpha=0.75);
```



圖：預覽圖表

18-3: 匯出線圖保存

In [2]:

```
'''
載入預設套件
'''

# basic
import numpy as np
import pandas as pd

# get data
import pandas_datareader as pdr

# visual
import matplotlib.pyplot as plt
import mpl_finance as mpf
import seaborn as sns

#time
import datetime
```

In [3]:

```
# 使用pandas_datareader抓取某個股票，這邊使用台積電(2330)
start = datetime.datetime(2020,1,1)
```

```

# 取得 data frame
df_2330 = pdr.DataReader('2330.TW', 'yahoo', start=start)

# 設定 index 格式
df_2330.index = df_2330.index.format(formatter=lambda x:
x.strftime('%Y-%m-%d'))

fig = plt.figure(figsize=(24, 8))

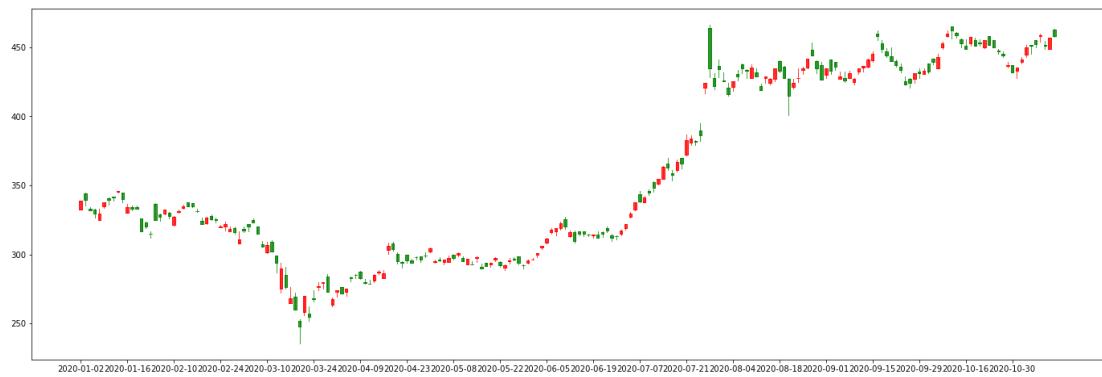
ax = fig.add_subplot(1, 1, 1)

# 設定 x 軸標籤
ax.set_xticks(range(0, len(df_2330.index), 10))
ax.set_xticklabels(df_2330.index[::10])

# 使用mpf.candlestick2_ochl 來繪製 K 線圖
mpf.candlestick2_ochl(ax, df_2330['Open'], df_2330['Close'],
                     df_2330['High'], df_2330['Low'], width=0.6, colorup='r',
                     colordown='g', alpha=0.75)

# 將輸出的圖片結果，另存新檔
plt.savefig("2330.png")

```



圖：預覽剛才另存的圖表

補充

Excel 操作

安裝 openpyxl

```
!pip install openpyxl
```

新增、讀取、儲存 excel

```
In [ ]:  
from openpyxl import load_workbook  
from openpyxl import Workbook  
  
# 動態新增檔案  
# workbook = Workbook()  
# worksheet = workbook.create_sheet("students", 0)  
  
# 讀取已存在的 excel 檔案  
importFile = "read.xlsx"  
workbook = load_workbook(filename = importFile)  
  
# 顯示所有工作表  
print(workbook.sheetnames)  
  
# 取得主要的 sheet  
worksheet = workbook['students']  
  
# 新增標題  
worksheet['C1'] = 'phone_number'  
  
# 新增資料  
worksheet['C2'] = "0911111111"  
worksheet['C3'] = "0922222222"  
worksheet['C4'] = "0933333333"  
worksheet['C5'] = "0944444444"  
  
# 自訂學生清單 dictionaries in list
```

```
listStudents = [
    {"name": "Eric", "age": 48, "phone_number": "0955555555"
} ,
    {"name": "Fox", "age": 27, "phone_number": "0966666666"
} ,
]

# 將學生清單寫入 excel (各別寫入名單的尾端)
position = 6
for student in listStudents:
    worksheet['A' + str(position)] = student["name"]
    worksheet['B' + str(position)] = student["age"]
    worksheet['C' + str(position)] = student["phone_number"]
]
position += 1

# 儲存 workbook
exportFile = "read_export.xlsx"
workbook.save(exportFile)

# 關閉 workbook
workbook.close()
```