

目次

Module 1. 開發環境：Anaconda、Jupyter 及爬蟲專案實務.....	1
Module 2. 基礎回顧之資料結構.....	32
Module 3. 基礎回顧之流程控制與迴圈.....	39
Module 4. 關於 URL 與字串格式化.....	53
Module 5. 正規表達式 (Regular Expression) 說明.....	55
Module 6. 正規表達式 (Regular Expression) 實作.....	58
Module 7. HTML 基礎與 HTTP 方法	63
Module 8. CSS Selector.....	76
Module 9. Chrome Developer Tool.....	83
Module 10. 套件 requests.....	96
Module 11. 套件 Beautiful Soup 4	100
Module 12. cookie 用於 requests.....	103
Module 13. 案例：PTT_NBA_看板主頁與內頁.....	104
Module 14. 套件 Selenium (一)	108
Module 15. 套件 Selenium (二)	111
Module 16. 套件 Selenium (三)	117
Module 17. ActionChains.....	123
Module 18. PyAutoGUI.....	132

- GitHub 專案連結

https://github.com/telunyang/python_web_scraping

- YouTube 頻道

<https://www.youtube.com/@darreninfo-boatman>

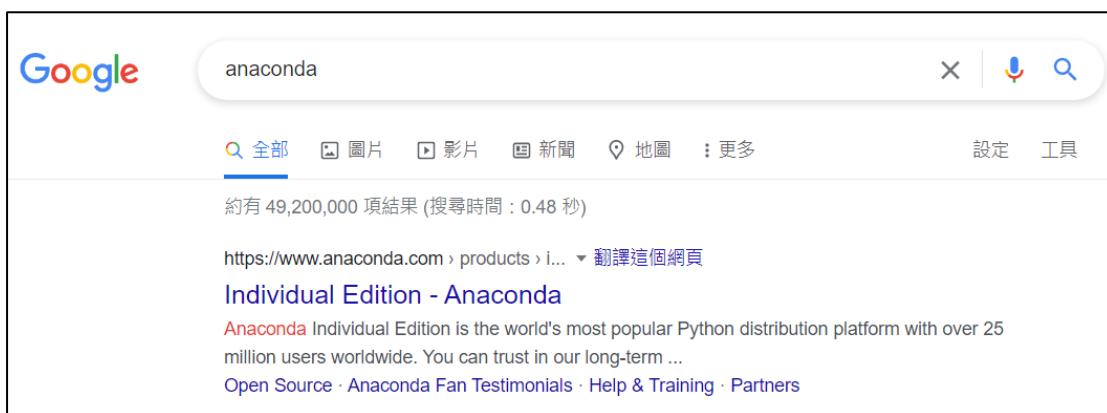
Module 1. 開發環境：Anaconda、Jupyter 及爬蟲專案實務

安裝 Anaconda

參考連結

[1] Anaconda - Individual Edition

<https://www.anaconda.com/products/individual>



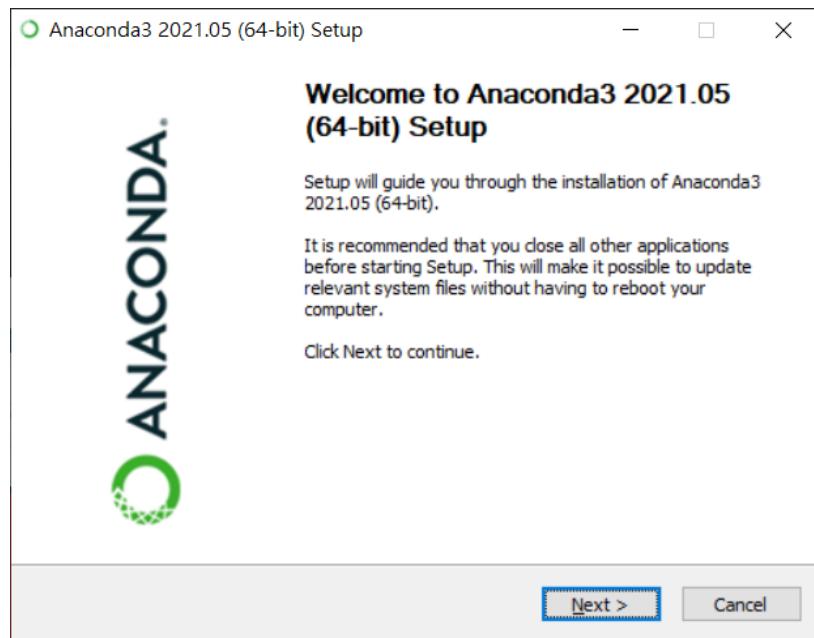
圖：可以先至 google 檢索「anaconda」，找到 Individual Edition 的連結



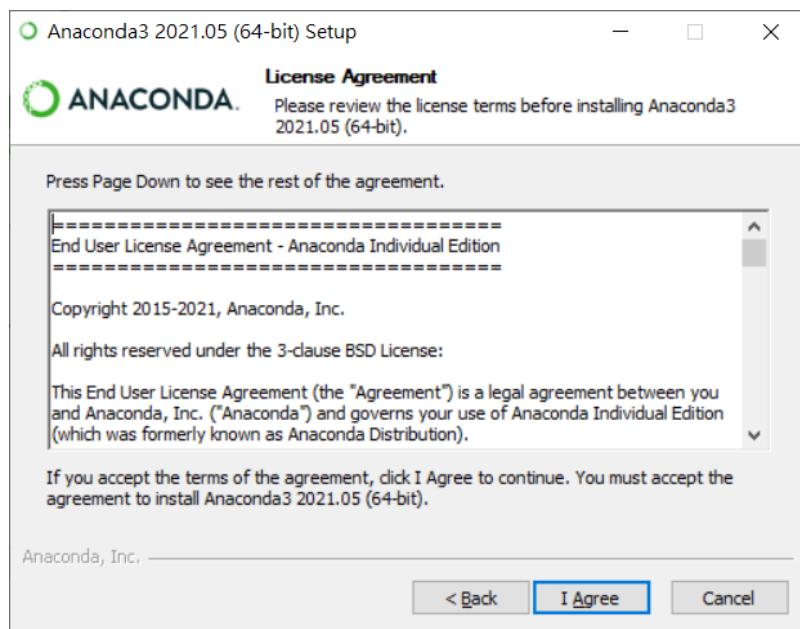
圖：在 Windows 選項下，選擇 64-Bit Graphical Installer，並下載下來



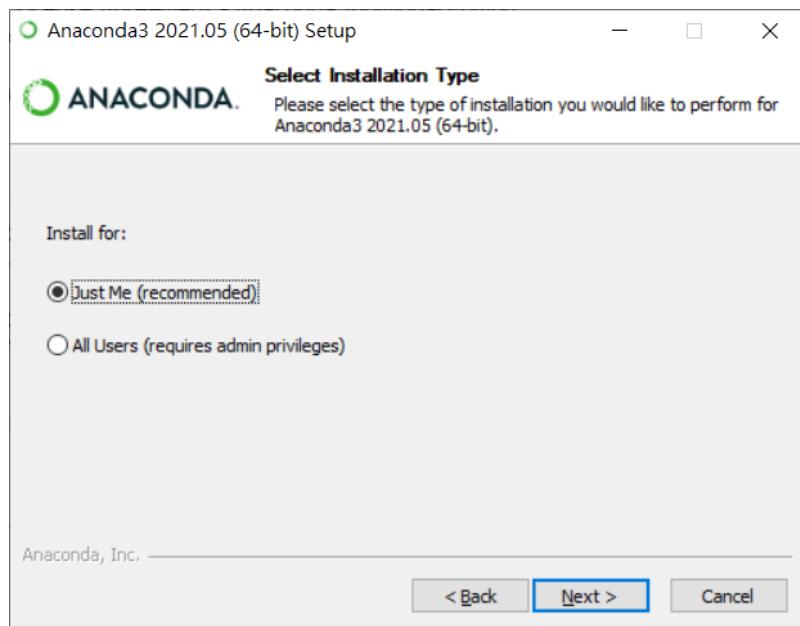
圖：等候下載



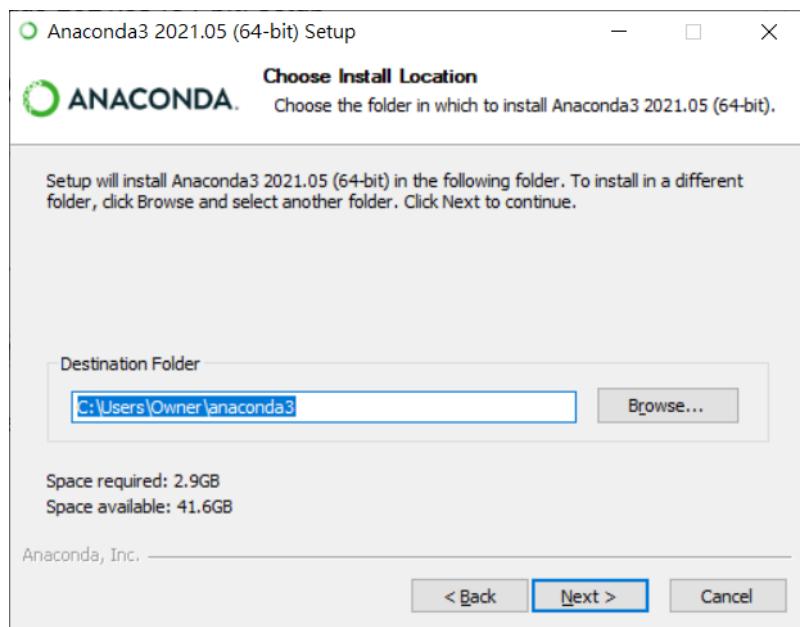
圖：按下「Next」



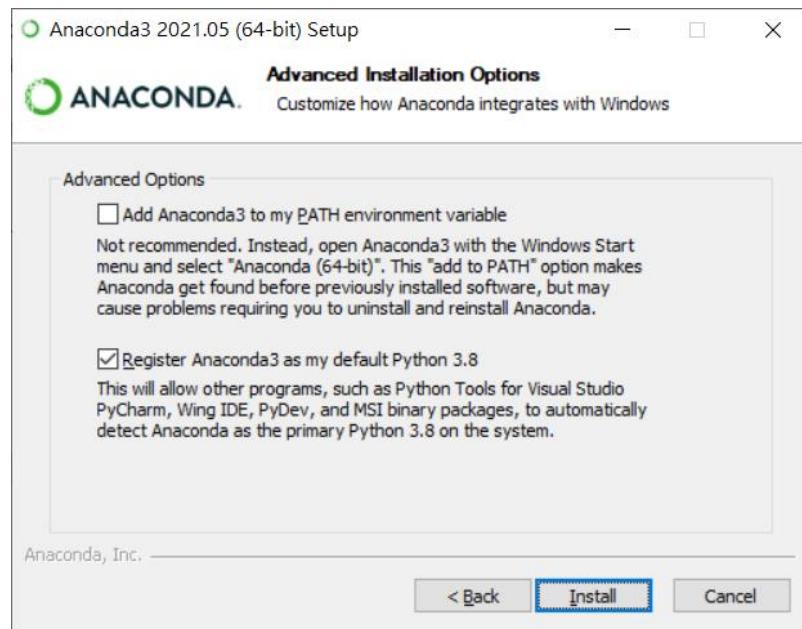
圖：按下「I Agree」



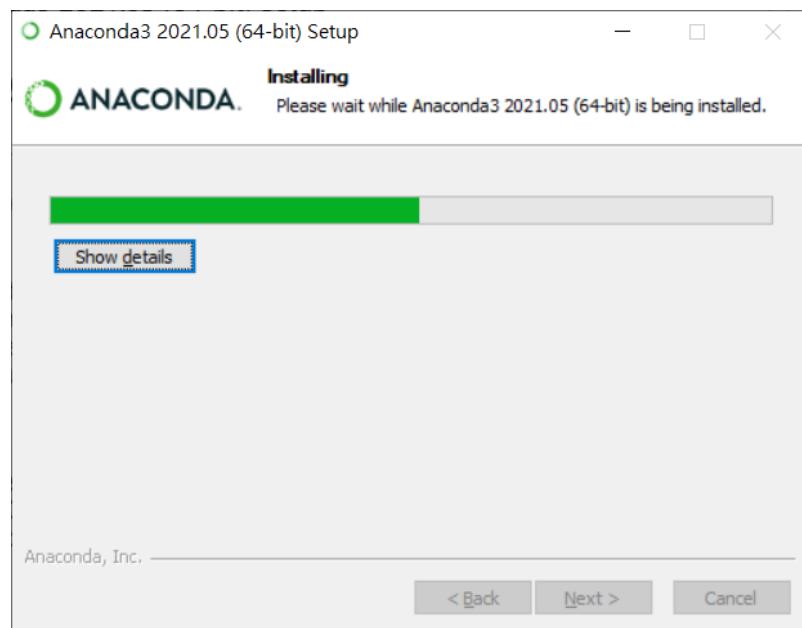
圖：依需求選擇後，按下「Next」



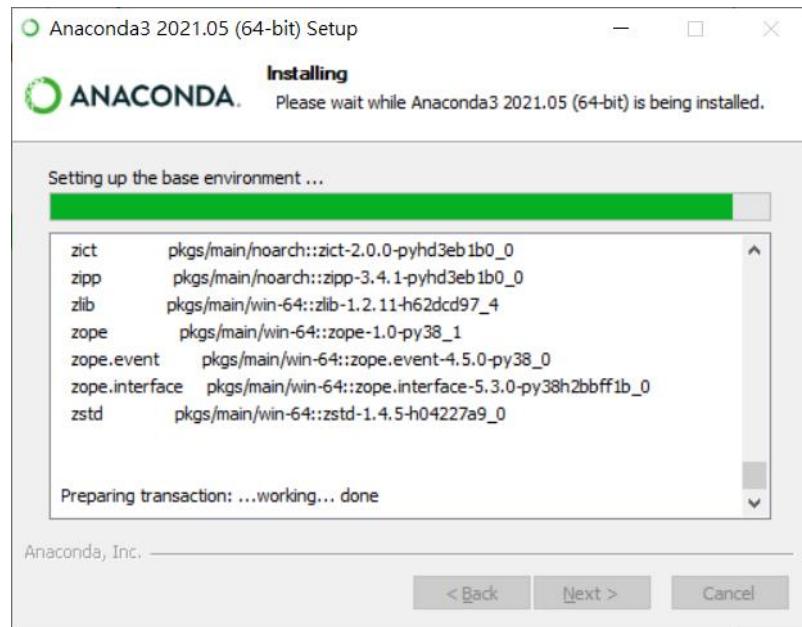
圖：安裝位置會在使用者資料夾中的 anaconda3，依需求設定，按下「Next」



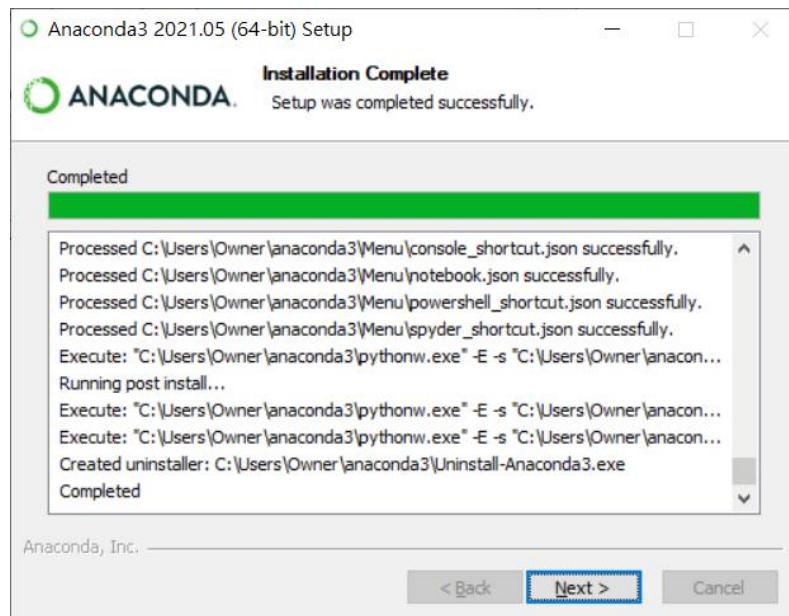
圖：依需求選擇，按下「Install」，進行安裝



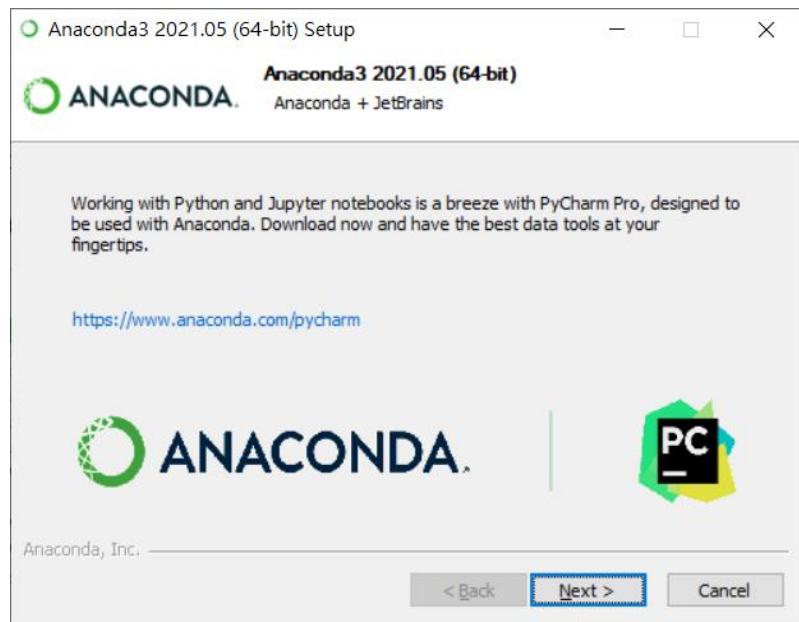
圖：安裝過程，需要一段時間



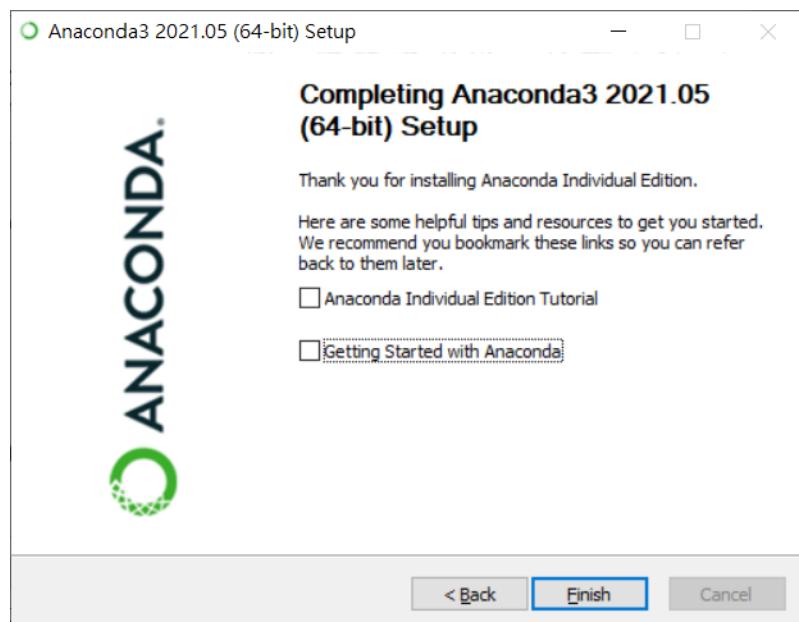
圖：按下「Show details」，會看到安裝過程



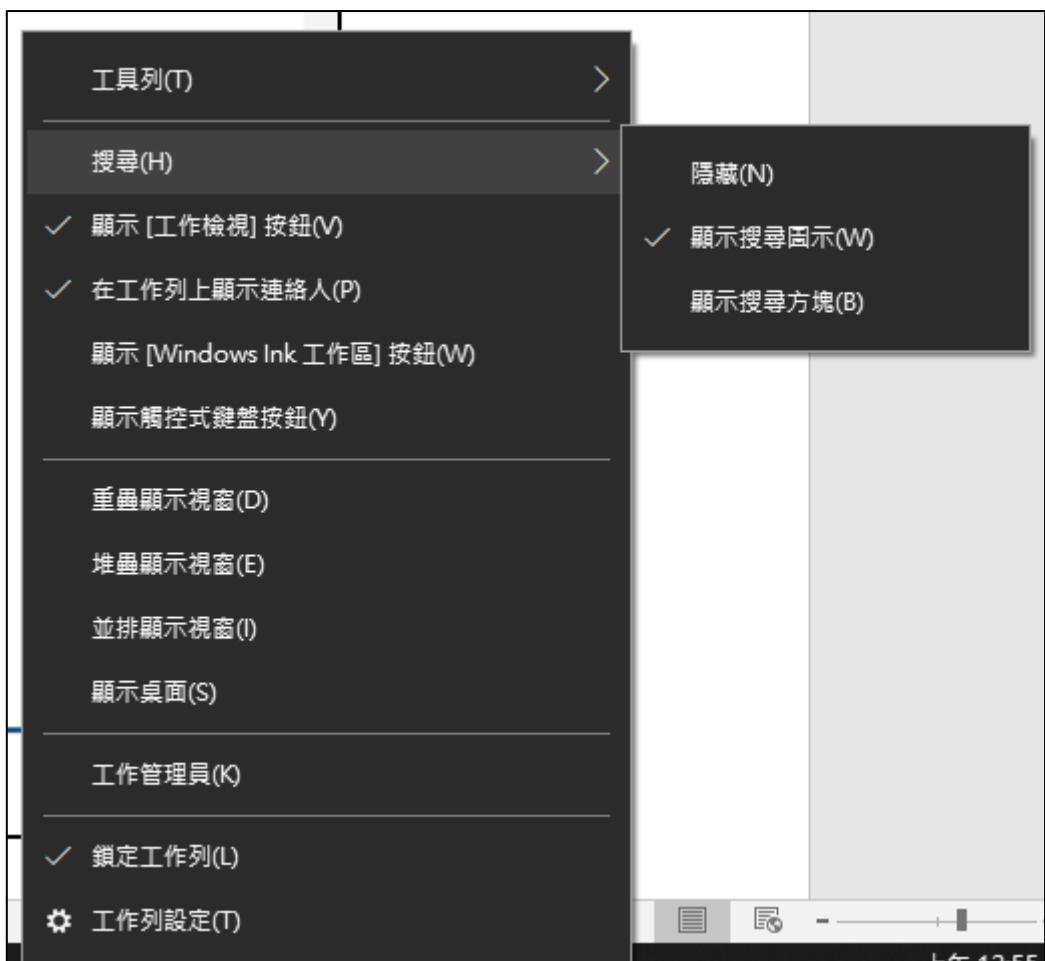
圖：安裝完成後，按下「Next」



圖：按下「Next」



圖：取消勾選圖片中的兩個選項後，按下「Finish」



圖：顯示搜尋圖示



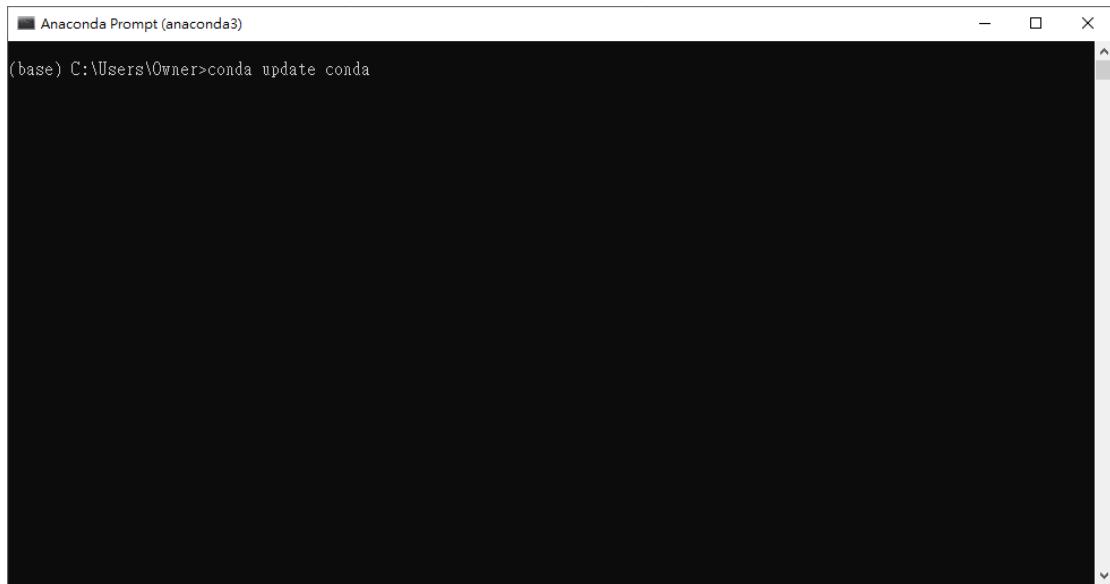
圖：搜尋圖示類似放大鏡，按下搜尋圖示



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

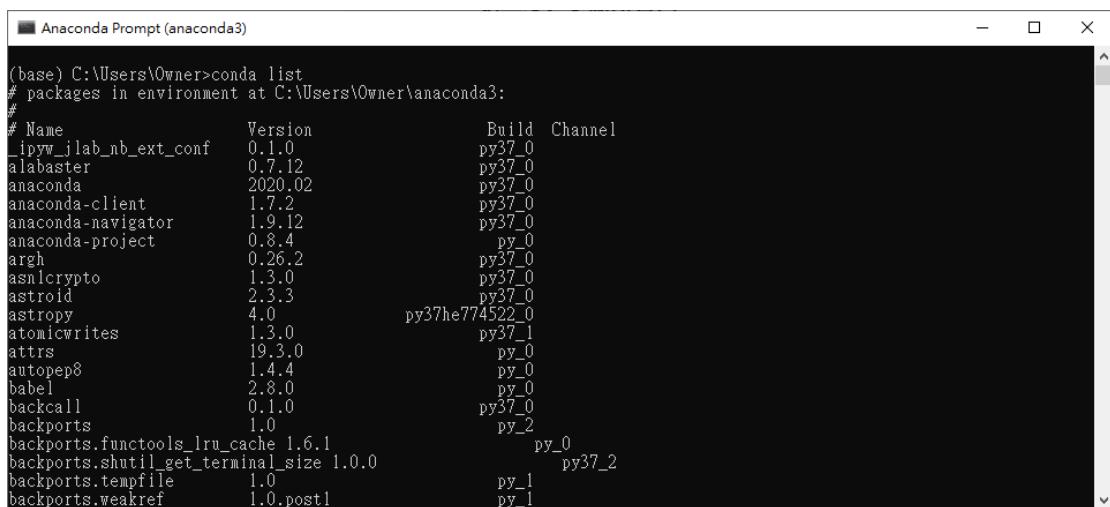


圖：出現 Anaconda Prompt，類似 Windows 的命令提示字元



(base) C:\Users\Owner>conda update conda

圖：輸入「`conda update conda`」本身



```
(base) C:\Users\Owner>conda list
# packages in environment at C:\Users\Owner\anaconda3:
#
# Name           Version        Build  Channel
_ipyw_jlab_nb_ext_conf    0.1.0      py37_0
alabaster          0.7.12     py37_0
anaconda           2020.02    py37_0
anaconda-client     1.7.2      py37_0
anaconda-navigator  1.9.12     py37_0
anaconda-project    0.8.4       py_0
argh                0.26.2     py37_0
asnlibcrypto        1.3.0      py37_0
astroid              2.3.3      py37_0
astropy             4.0        py37he774522_0
atomicwrites        1.3.0      py37_1
attrs                19.3.0     py_0
autopep8            1.4.4      py_0
babel                2.8.0      py_0
backcall             0.1.0      py37_0
backports            1.0        py_2
backports.functools_lru_cache 1.6.1      py_0
backports.shutil_get_terminal_size 1.0.0    py37_2
backports.tempfile    1.0        py_1
backports.weakref     1.0.post1   py_1
```

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

```

■ Anaconda Prompt (anaconda3)

werkzeug          1.0.0           py_0
wheel             0.34.2          py37_0
widgetsnbextension 3.5.1           py37_0
win_inet_pton     1.1.0           py37_0
win_unicode_console 0.5            py37_0
wincertstore      0.2            py37_0
winpty             0.4.3           py_4
wrapt              1.11.2          py37he774522_0
xlrd               1.2.0           py37_0
xlsxwriter        1.2.7           py_0
xlwings            0.17.1          py37_0
xlwt               1.3.0           py37_0
xmltodict         0.12.0          py_0
xz                 5.2.4           h2fa13f4_4
yaml               0.1.7           hc54c509_2
yapf               0.28.0          py_0
zeromq             4.3.1           h33f27b4_3
zict                1.0.0           py_0
zipp                2.2.0           py_0
zlib               1.2.11          h62dcd97_3
zstd               1.3.7           h508b16e_0

(base) C:\Users\Owner>conda upgrade --all

```

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-h62dcd97_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-h62dcd97_3 --> 1.2.11-h62dcd97_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

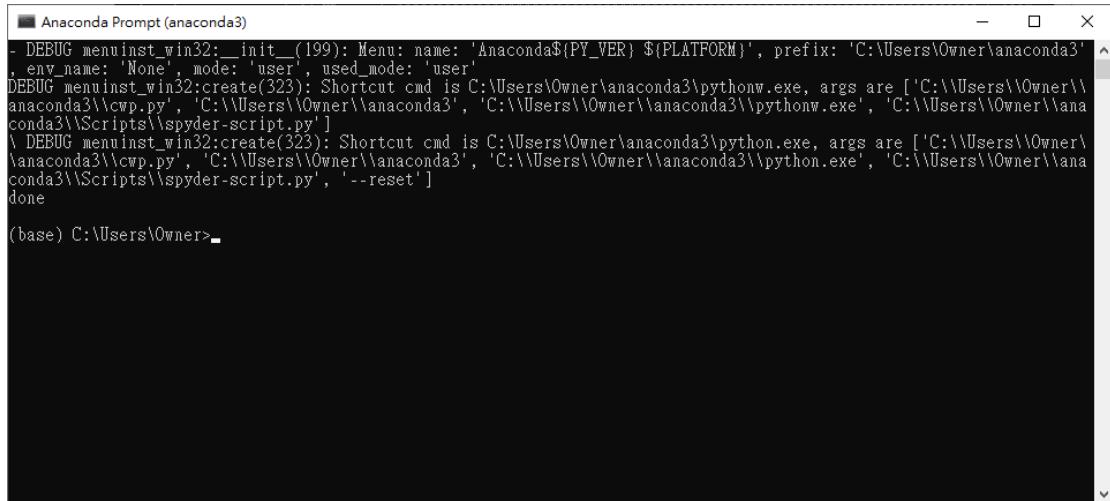
lzo                    2.10-h6df0209_2 --> 2.10-he774522_2

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-appleh | 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-serv | 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

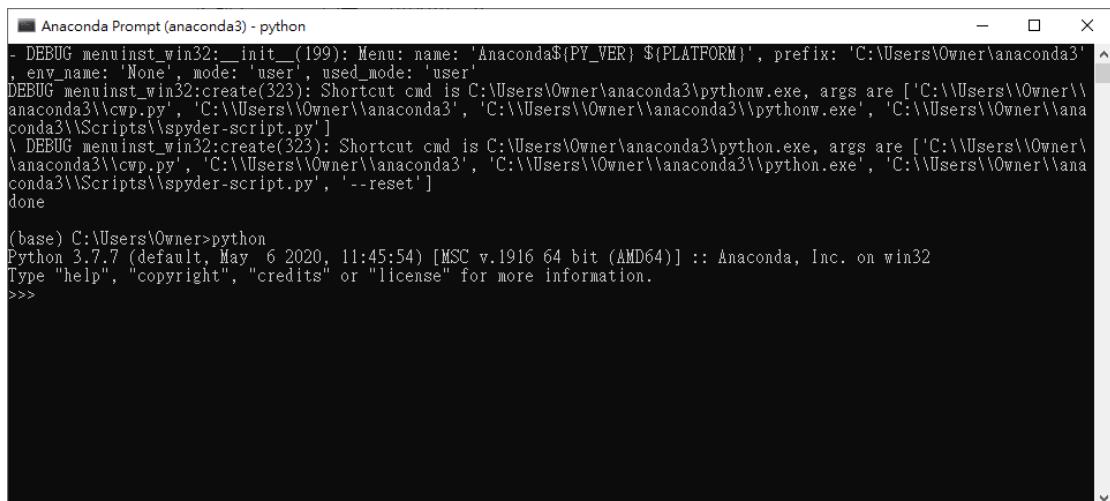


A screenshot of the Anaconda Prompt window titled '(anaconda3)'. The terminal output shows several DEBUG messages from the menuinst_win32 module, indicating the creation of Python-related shortcuts. The messages include:

```
- 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>

圖：套件更新完成

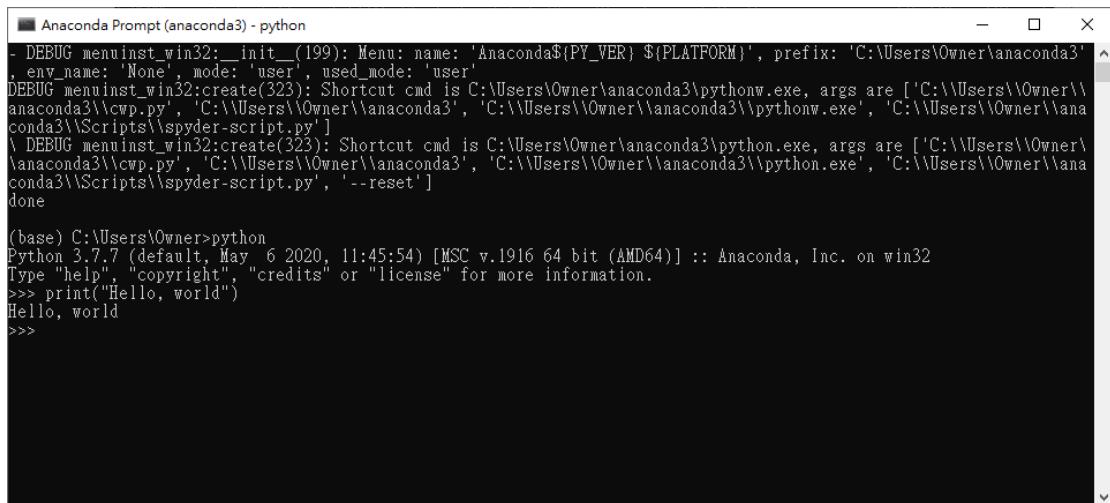


A screenshot of the Anaconda Prompt window titled '(anaconda3) - python'. The terminal output shows the completion of the package update process. It includes DEBUG messages from menuinst_win32 and a standard Python 3.7.7 startup message. The output ends with a prompt for further commands.

```
- 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.
>>>
```

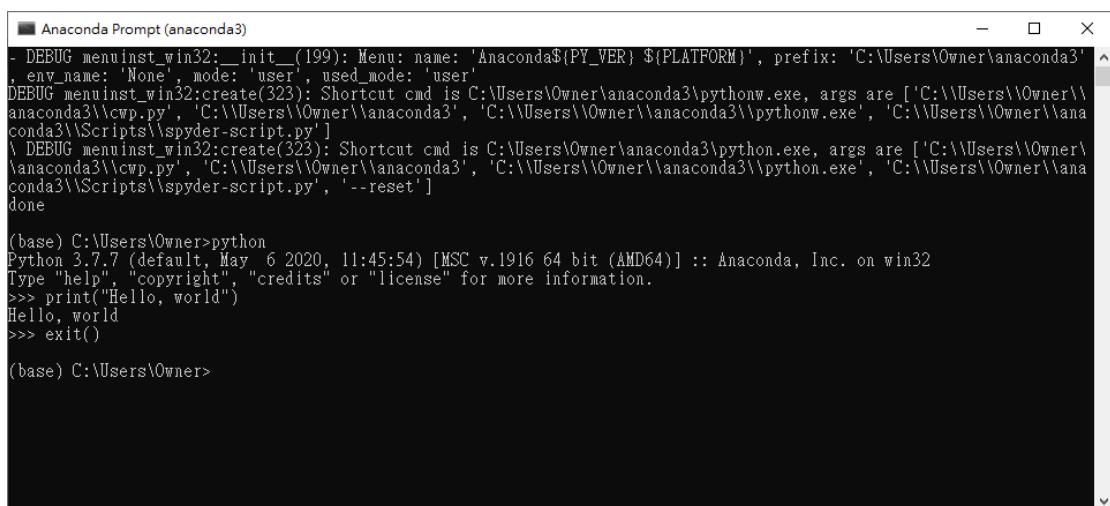
圖：輸入「python」，進入 python 執行環境



```
■ 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\\\\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\\\\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

安裝、切換與刪除 Conda 環境 (Environment)

預設是 (base)，如果有切換環境的需求，例如手上處理著不同 Python 版本和其它相關套件的專案，需要不時切換版本來開發，此時可以建立一到多個 Conda 環境，需要的時候可以切換，不需要的時候可以刪除。

在執行以下的指令前，需要確認目前是否在 Anaconda Prompt 當中，或

是支援 Conda 的 Terminal 環境。終端機顯示預設路徑時，最前面會有 (base)，代表目前正在預設的 Conda 環境當中。

安裝 Conda 環境

範例

```
conda create --name 自訂環境名稱 python=版本號，例如 3.9
```

```
conda create --name web_scraping python=3.9
```

列出所有 Conda 環境

```
conda env list
```

離開當前的 Conda 環境

```
conda deactivate
```

切換 / 啟動 Conda 環境

```
conda activate web_scraping
```

刪除 Conda 環境

```
conda env remove -n web_scraping
```

參考網頁

[1] Conda create environment and everything you need to know to manage conda virtual environment

<https://www.machinelearningplus.com/deployment/conda-create-environment-and-everything-you-need-to-know-to-manage-conda-virtual-environment/>

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）。

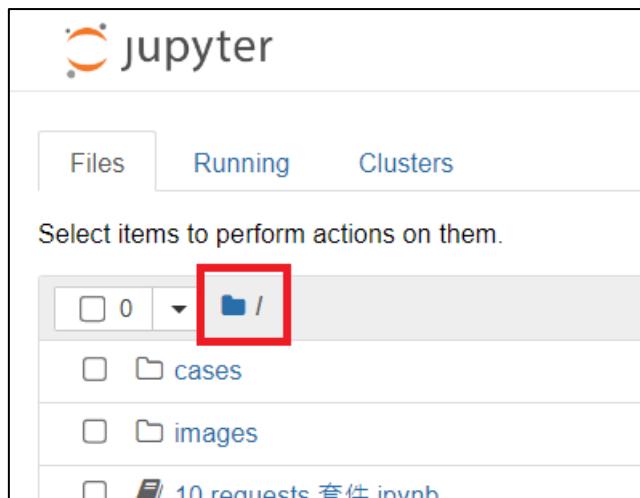
如果我們今天僅是使用 Jupyter 預設的環境，路徑應該在「C:\Users\{你的帳號}」底下，我們只要將專案解壓縮，放至路徑底下即可。

指定路徑 / 目錄下啟動 Jupyter Notebook

開啟 Anaconda Prompt，切換到指定的路徑 / 目錄，再輸入 jupyter notebook 後按下 enter，便能直接從指定位置開啟 Jupyter Notebook。

```
Anaconda Prompt (anaconda3) - jupyter notebook
(base) C:\Users\Owner>cd C:\Users\Owner\notebooks
(base) C:\Users\Owner\notebooks>jupyter notebook
```

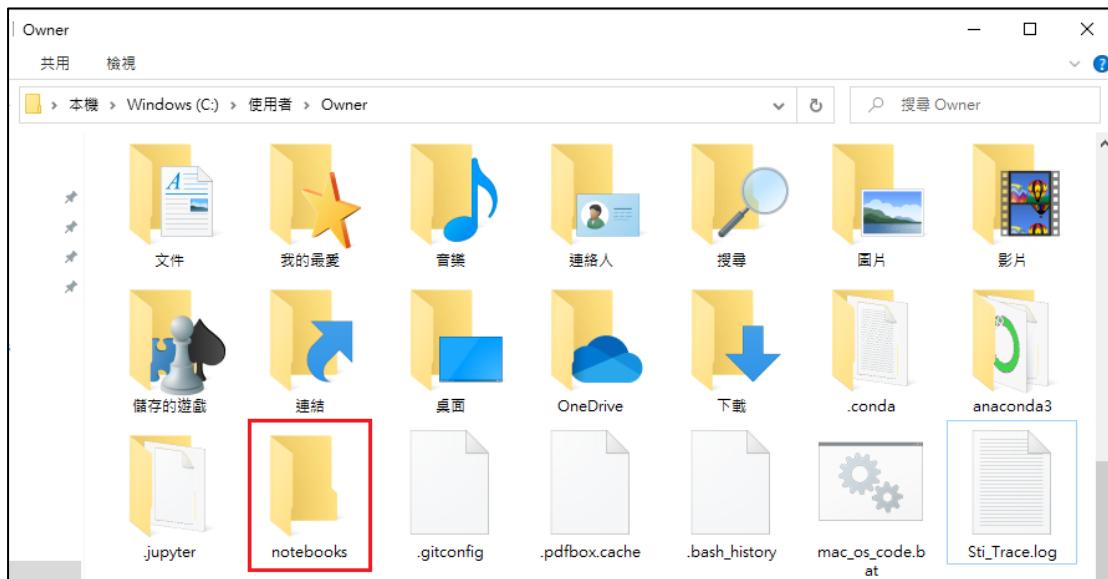
圖：切換路徑，輸入 jupyter notebook 後按下 enter



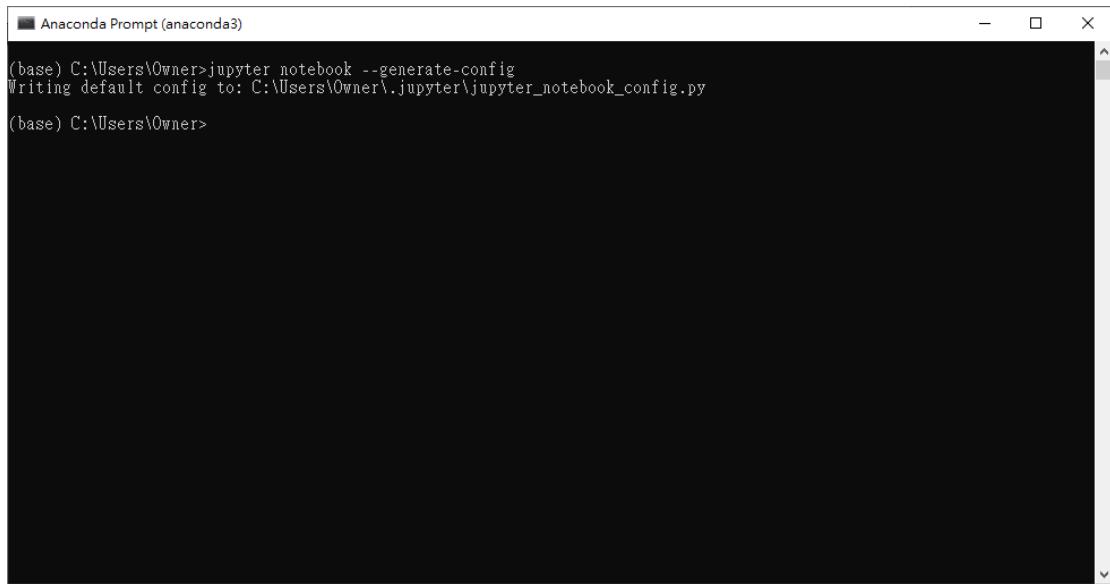
圖：以指定路徑 / 目錄作為主要作業環境

(Optional) 不使用預設設定，建立自訂預設工作目錄

預設在 C:\Users\{你的帳號} 底下，會與其它目錄混在一起，此時可以建立一
啟動 Jupyter Notebook 就直接進入的自訂預設工作目錄。



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



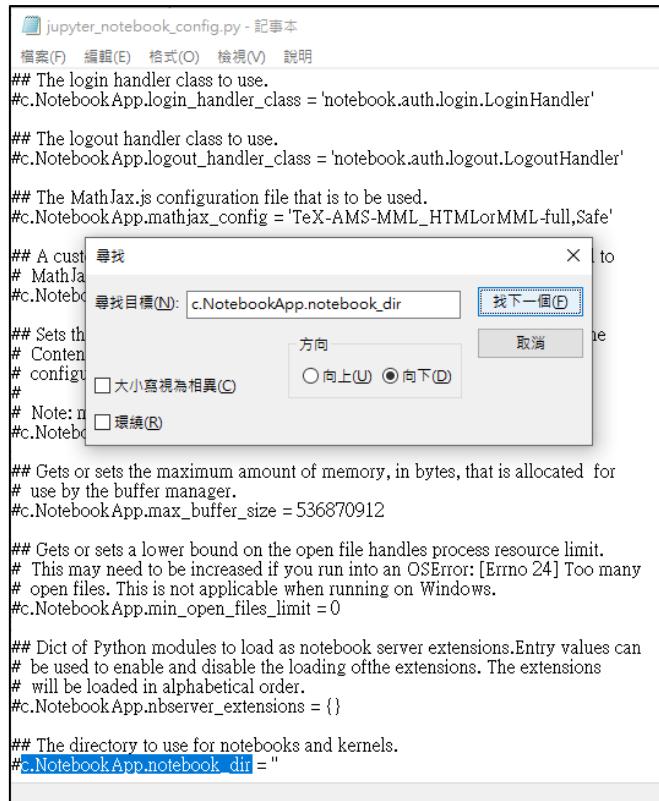
```
Anaconda Prompt (anaconda3)
(base) C:\Users\Owner>jupyter notebook --generate-config
Writing default config to: C:\Users\Owner\.jupyter\jupyter_notebook_config.py
(base) C:\Users\Owner>
```

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



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

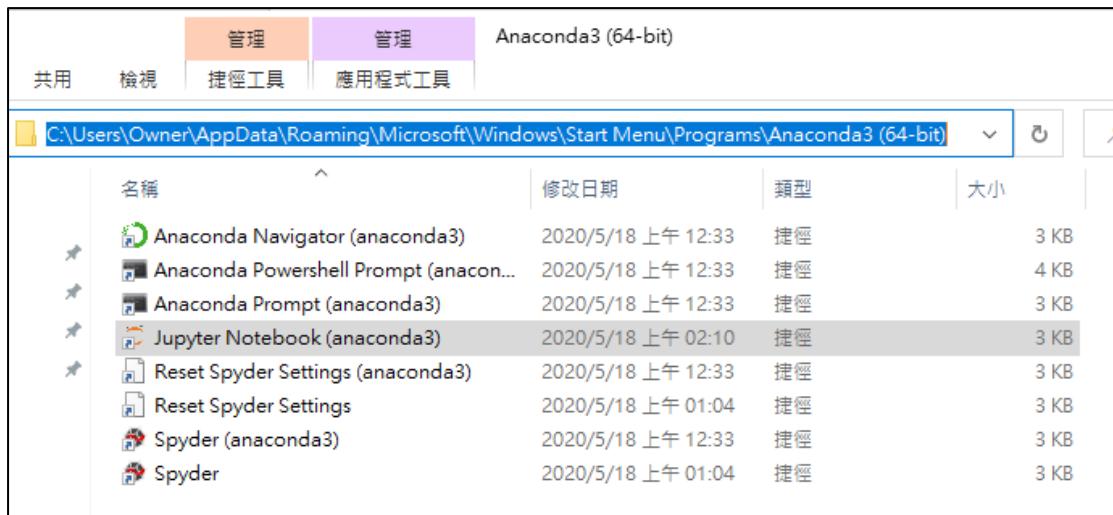
程式碼
<pre> ## The directory to use for notebooks and kernels. c.NotebookApp.notebook_dir = 'C:\\\\Users\\\\Owner\\\\notebooks' </pre>

將路徑貼至「」之中，儲存關閉（在 Windows 底下，反斜線需要兩個）。

接下來我們要編輯 Jupyter Notebook 檔案連結，有兩種方法：

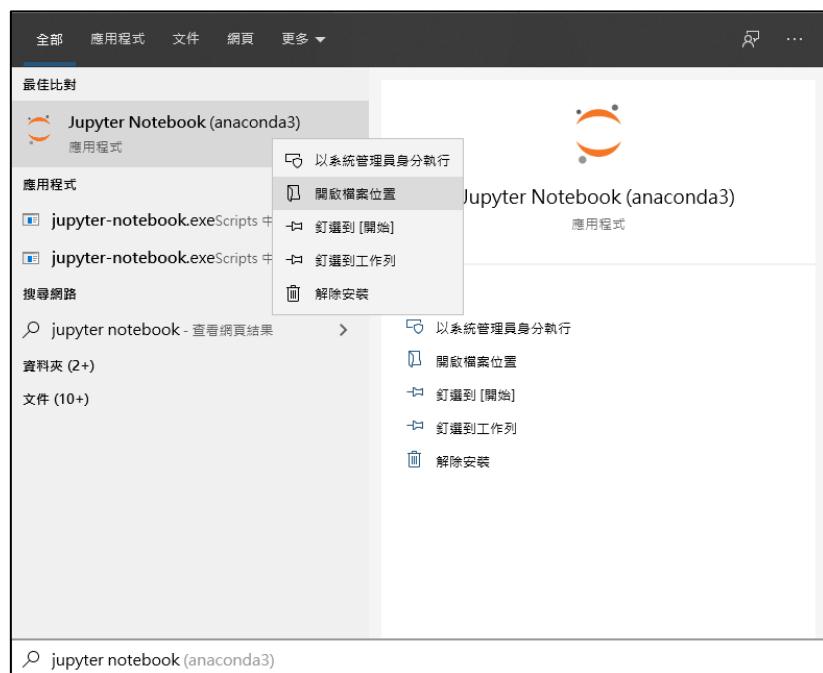
- 到「C:\Users\{你的帳號}\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Anaconda3 (64-bit)」中，找到「Jupyter Notebook

(anaconda3)」。



圖：直接在檔案總管的路徑列中，輸入檔案路徑。

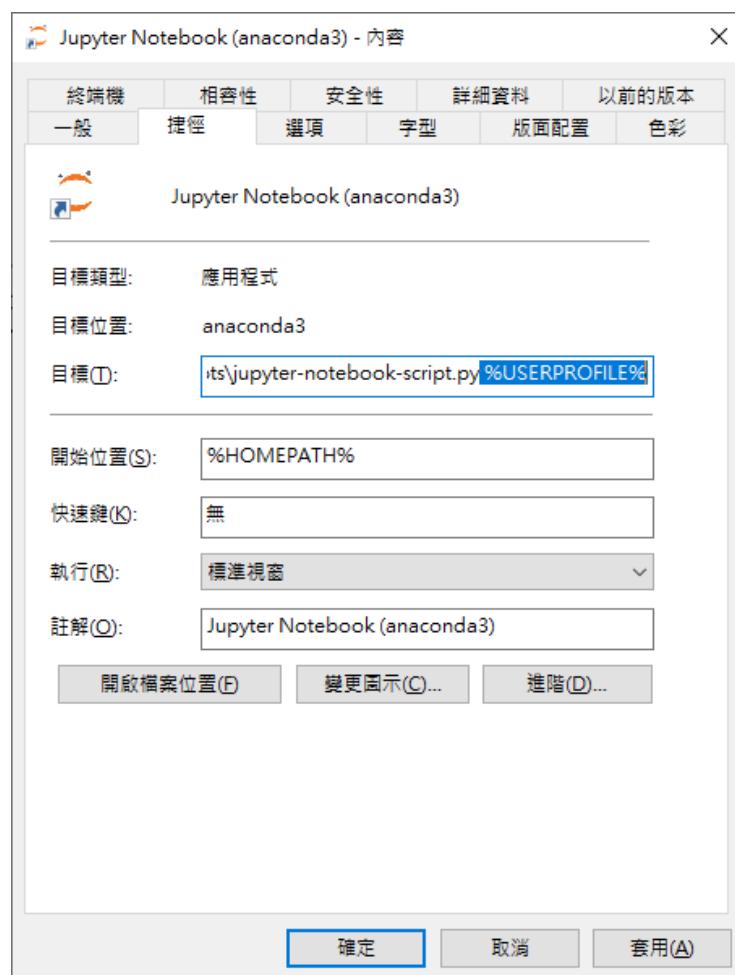
- 按下放大鏡圖示，搜尋「Jupyter Notebook (anaconda3)」，然後對搜尋結果的圖示按下右鍵，再按下「開啟檔案位置」。



圖：透過搜尋來取得 Jupyter Notebook 的檔案連結位置

說明

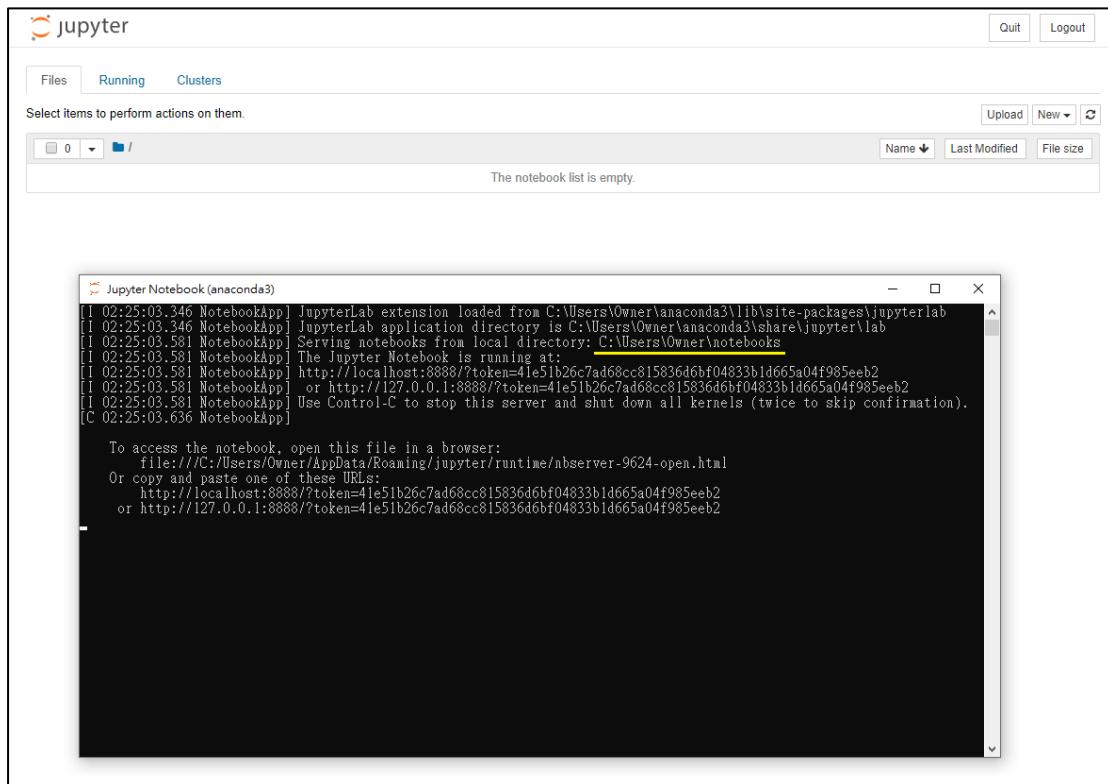
對 Jupyter Notebook 檔案連結按右鍵，選擇「內容」，我們要編輯「目標」文字區塊的內容，將文字最後面的「%USERPROFILE%」刪掉，按下「確定」。



圖：刪除「%USERPROFILE%」的字樣後，按下確定



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



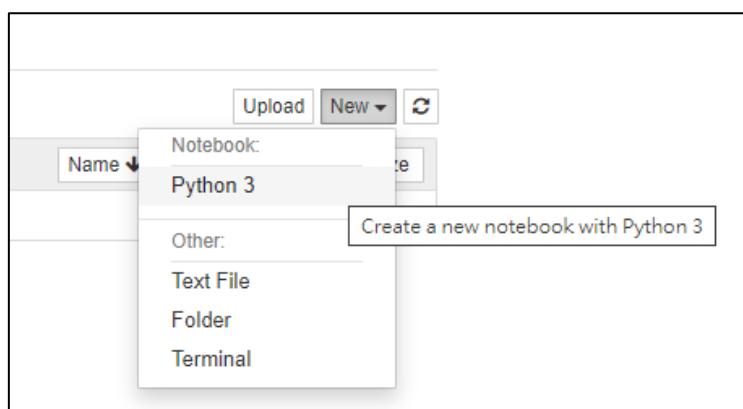
圖：執行 Jupyter Notebook 後，會以自訂路徑作為預設工作目錄

參考資料

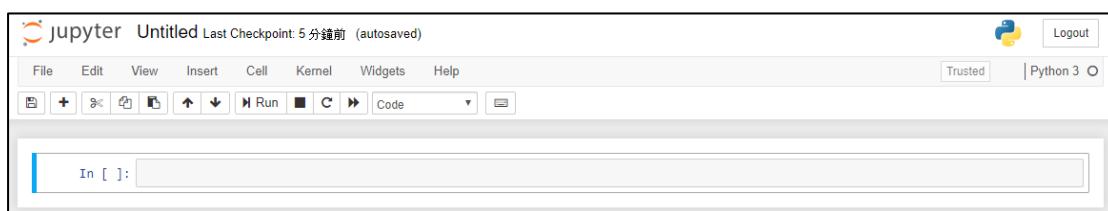
jupyter notebook 更改默认工作路径

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

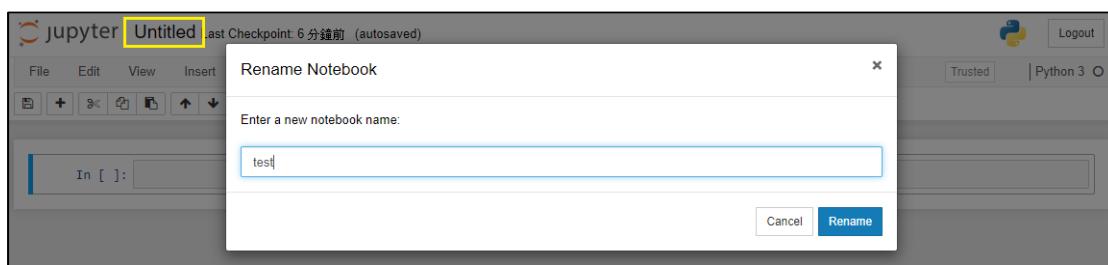
新增 Notebook



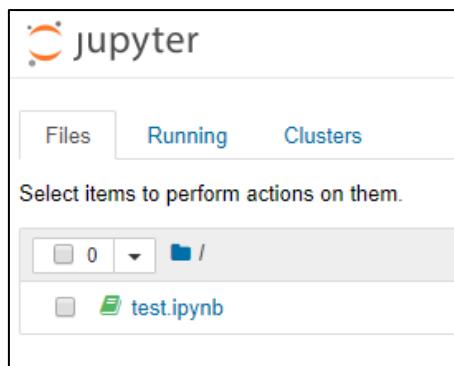
圖：按下 New · 新增 Notebook (選擇 Python3)



圖：此時進入新建的 jupyter notebook document 中

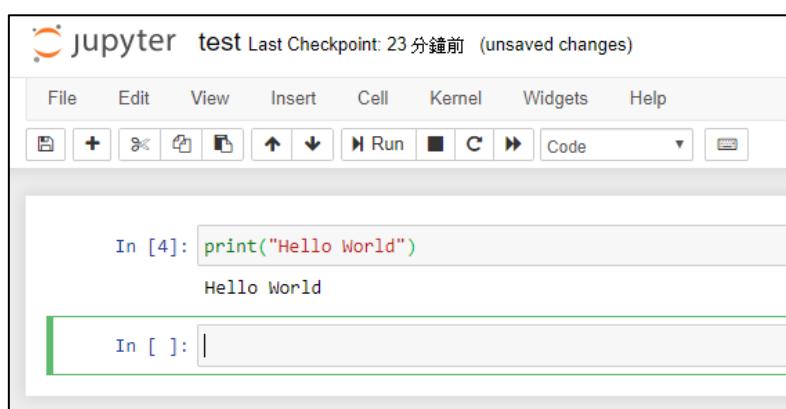


圖：按下「Untitled」，重新命名 Notebook name，按下「Rename」



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，同時向下新增 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.

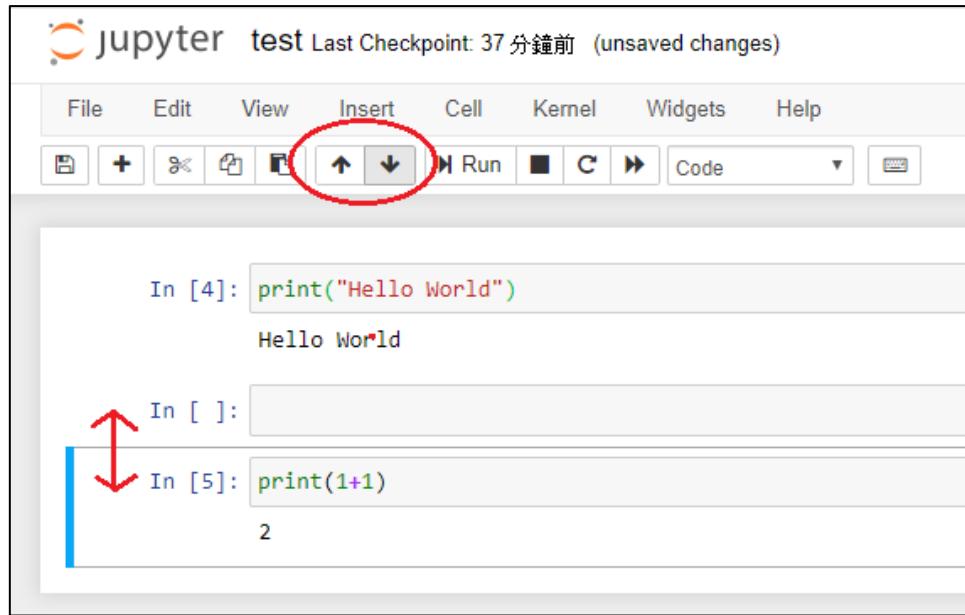
```
In [4]: print("Hello World")
Hello World

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

In [ ]:
```

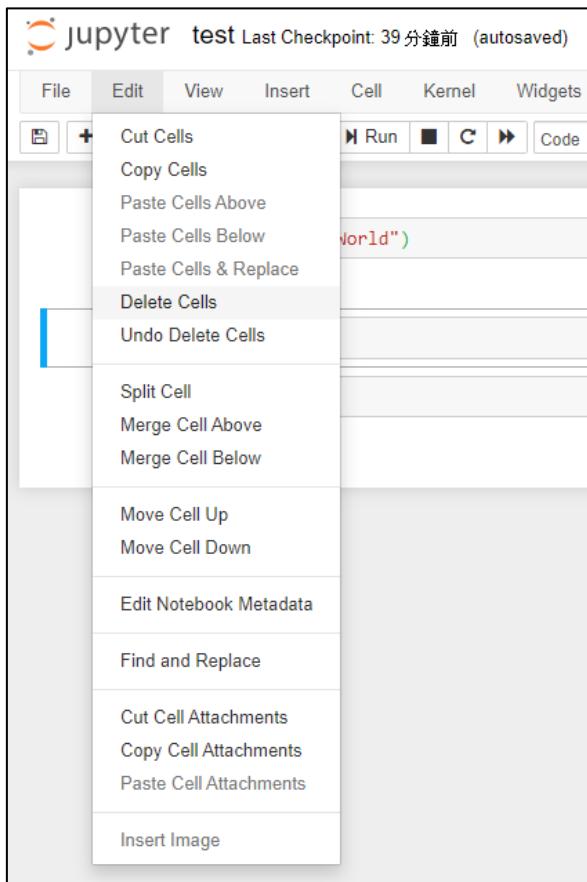
圖：按下「+」符號，會在當前 cell 下新增空白的 cell

移動 cell 位置



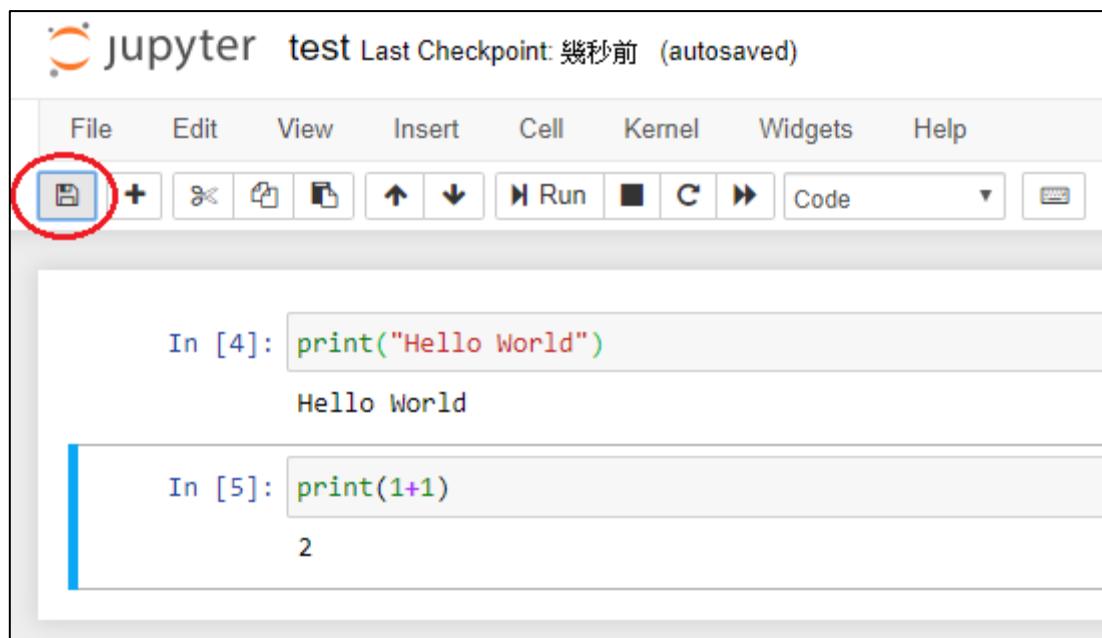
圖：選定 cell 後，按下「↑」或「↓」，可以移動 cell 的位置

刪除 cell



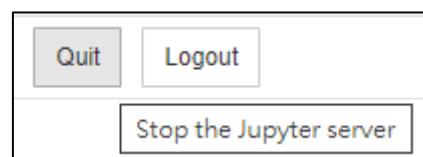
圖：選定 cell 後，按下「Edit」，選擇「Delete Cells」，即可刪除 cell

儲存 notebook

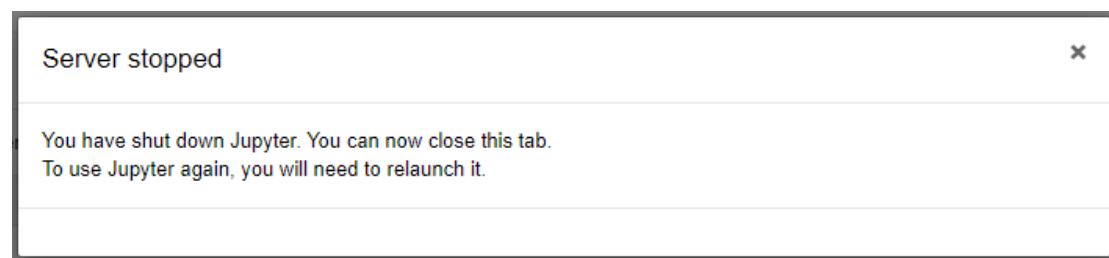


圖：按下儲存鈕，即可儲存 notebook

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

切換 Jupyter Kernel

Kernel 是 Python 的編譯器 (Compiler)，編譯器的主要功能，簡單來

說，就是將我們撰寫的原始碼，轉換成機器看得懂的語言，進而執行我們自訂的程式流程，預設是 IPython。

檢視 Jupyter Notebook kernel

jupyter kernelspec list

注意：

1. Jupyter Notebook 建立的 kernel，跟 conda 的 env 不一樣。一般來說，想在 Jupyter Notebook 裡面用到 conda 的 env，需要在 conda env 另外安裝獨立的 Jupyter Notebook，才能用到該 conda 的 env。
2. 安裝完 Anaconda 後的內建 Jupyter Notebook，是獨立工具，與 conda 另行安裝的 env **沒有直接關係**。

安裝 Kernel

```
python -m ipykernel install --user --name web_scraping --display-name  
web_scraping
```

```

Anaconda Prompt (anaconda3)
(base) C:\Users\Owner>jupyter kernelspec list
Available kernels:
python3    C:\Users\Owner\anaconda3\share\jupyter\kernels\python3

(base) C:\Users\Owner>python -m ipykernel install --user --name web_scraping --display-name web_scraping
Installed kernelspec web_scraping in C:\Users\Owner\AppData\Roaming\jupyter\kernels\web_scraping

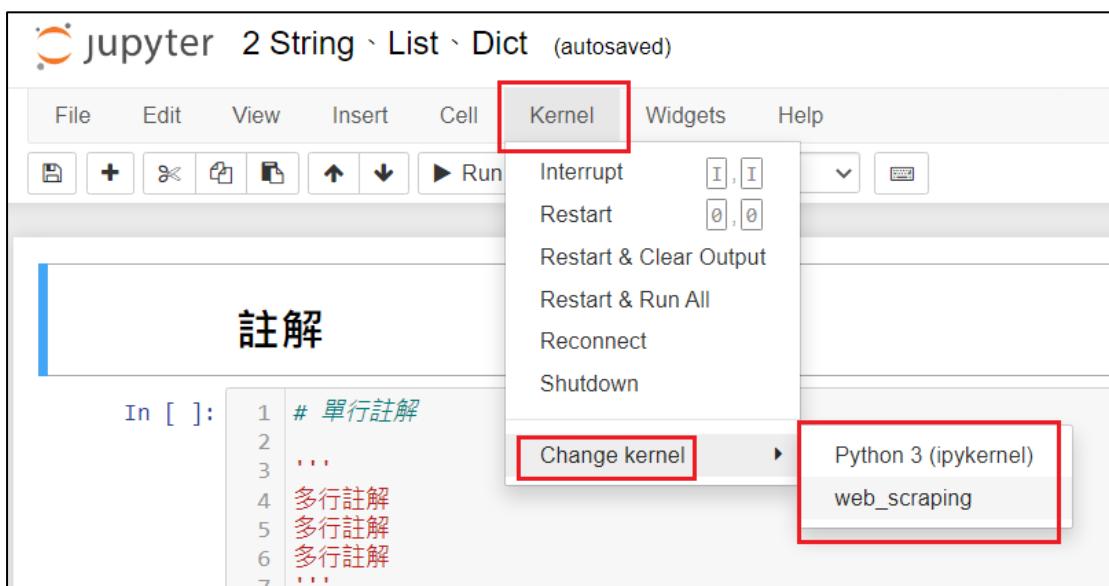
(base) C:\Users\Owner>jupyter kernelspec list
Available kernels:
web_scraping    C:\Users\Owner\AppData\Roaming\jupyter\kernels\web_scraping
python3         C:\Users\Owner\anaconda3\share\jupyter\kernels\python3

(base) C:\Users\Owner>

```

圖：安裝完 Jupyter Notebook Kernel，再次檢視，會發現新增的 Kernel

開啟自行新增或既有的 ipynb 檔案，按下 Kernel，移到 Change Kernel，再選擇自行安裝的 Conda 環境。



圖：切換 Jupyter Notebook Kernel

下載 Chrome Web Driver (課程後面會由 WebDriverManager 取代)

1. 請先下載 ChromeDriver (註：電腦要先安裝 Google Chrome 瀏覽器)

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

2. 請確認目前你電腦裡面的 chrome 瀏覽器版本



圖：按下瀏覽器右上方的「⋮」→說明→關於 Google Chrome



圖：請與下載的 ChromeDriver 版本一致

3. 下載 ChromeDriver 檔案，並放到專案資料夾當中



圖：下載合適的 Chrome 版本

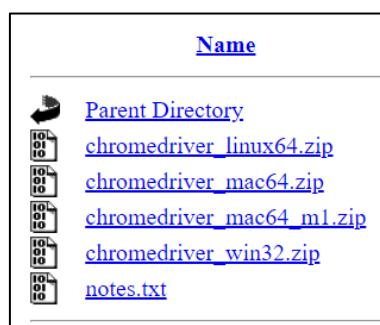


圖: Windows 選擇 win32 ; MacOS 選擇對應的晶片版本

安裝課程所需套件

可以考慮先切換到自訂的 Conda 環境，例如 web_scraping，再透過 pip 安裝套件。

切換 Conda 環境

```
conda activate web_scraping
```

說明

本課程需要安裝的套件指令

```
pip install beautifulsoup4 lxml requests selenium webdriver-manager fake-useragent tika  
pyautogui opencv-python opencv-contrib-python keyboard
```

補充

```
pip install mplfinance browsermob-proxy pymysql
```

爬蟲專案開發實務分享

1. 個人 YouTube 頻道 (不定時更新)

<https://www.youtube.com/channel/UCUqT6-mTPkQkCyGjlbm3IMA>

2. 「觀察」是爬蟲工作者的重要技能，HTML & CSS selector 的概念要非常熟練；有些人習慣使用 XPATH，由於課程時間安排的關係，不會提到。

3. 被對方的伺服器擋住，無法繼續爬取，相換 IP，可以透過

(a)、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

(b)、**連接 VPN**，例如 Surfshark，透過切換其它網路環境（俗稱翻牆）來

進行；

(c)、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（例如 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，有時候可以搭配 PC 平台的手機螢幕控制工具，結合課堂上

提到的 PyAutoGUI，透過網路請求的品質，來決定是否切換；

(d)、可以整合免費的 **Proxy Pool** 來取得臨時可用的 IP，持續對網頁進行

請求，如果更付費的話，品質更好。

4. 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

註解

單行註解

...

多行註解

多行註解

多行註解

...

....

這也是多行註解

這也是多行註解

....

變數

...

變數：值(如字串、數值等)的容器

變數命名方式：

原則上是「英文、數字和底線(_)」的組合，

例如 my_name、getName、age，

但不能是「數字」開頭

...

my_age = 18

123_my_age = 18

String

- 「字串」(string) 如同一串字，同時由兩個以上的字元 (character) 組成。
- 字串需要用「兩個單引號」或「兩個雙引號」包起來，例如 '星期日' 或 "星期日"。
- 包住字串的兩個引號一定要一樣，不能一個單引號、一個雙引號，要同時都是單引號，或是同時都是雙引號。

如果包住的字只有一個，則稱之為「字元」(character)。

字串變數初始化

string01 = "1,2,3,4"

```
print(string01)

# 替換字串

...
用法
string.replace(str1, str2)
說明
將 s 中的 str1 替換成 str2
...
string02 = "Alex"
string02 = string02.replace('ex', 'len')
print(string02)
```

```
# 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
...
string03 = "      ___ccc___"
print(string03)
print(string03.strip())
```

```
# 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())
```

```
# 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())
```

List

- 稱為「串列」或「列表」，跟其它程式語言當中的「陣列」(Array)很類似。
- 使用中括號 [] 將資料儲存起來。
- 可以想像成一個七天份的藥盒子，從星期天開始。
- 例如【'星期天', '星期一', '星期二', '星期三', '星期四', '星期五', '星期六'】。
- 這個藥盒子內，有各自存取的代號(位置)，例如 0 代表「星期天」，1 代表「星期一」，2 代表「星期二」，依此類推。
- 這個存取的代號 0、1、2…等等，稱之為「索引」(index)。
- 可以儲存各種資料型態。

```
# 初始化一個 list
```

```
ids = [1, 2, 3, 4, 5, 6]  
print(ids)
```

```
# 新增元素在 list 尾端
```

```
ids.append(7)  
print(ids)
```

```
# 修改索引位置的元素
```

```
ids[4] = 99  
print(ids)
```

```
# 刪除索引位置的元素
```

```
ids.pop(4)  
print(ids)
```

```
# 新增元素 9 在指定索引 1，其餘元素往後移
```

```
ids.insert(1, 9)
print(ids)
```

移除指定的值

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

補充：字串分割成 list

...

用法

string.split()

說明

默認以空格、換行字元分割字串 s，返回 list

...

```
string04 = "1,2,3,4"
```

```
list04 = string04.split(',')  
print(list04)
```

補充：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

...

```
list05 = ['古道', '西風', '瘦馬']
```

```
string05 = '-'.join(list05)
```

```
print(string05)
```

```
string05 = ''.join(list05)
```

```
print(string05)
```

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

Slicing

```
# 初始化一個字串
myStr = 'Hello,World!'

...
透過冒號 ( :) 進行分割
...

# 從頭取到 5 ( 不包含第 5 個元素，實際為 0 ~ 4 )
print( myStr[0:5] ) # 索引從 0 開始，到 5 - 1 的索引位置

# 從頭取到 5 ( 不包含第 5 個元素，實際為 0 ~ 4 )
print( myStr[:5] ) # 冒號前面留空，效果跟 [0:5] 一樣

# 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，實際為 7 ~ 8 )
print( myStr[7:9] )

...
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] )

# 取得檔案全名
string06 = "/home/darren/ebook.txt"
list06 = string06.split("/")
print(list06[-1])

# 搜尋字串
string07 = 'believe'
result07 = string07.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(result07)

```

Dict

- 稱為「字典」，跟其它程式語言的「物件」(Object) 很類似。
- 透過大括號 { } 來存放資料。
- 結構以 key:value 來儲存資料，key 可以想像成一個變數。
- 一個字典可以放置多組 key:value，每組之間用逗號 (,) 分隔。
- 例如 {'name': 'Darren', 'age': 18, 'height': 172, 'weight': 100}
- 其中 'name' 就是字典其中的 key，而 'Darren' 就是此 key 對應的 value。
- 與 List 不同的是，Dict 通常是透過 key 來存取 value，沒有特定的 index。

```

# 初始化 dict
dict01 = {"蘋果": 100, "橘子": 20, "水梨": 50}
print(dict01)

```

...

也可以透過排版，寫成：

```

dict01 = {
    "蘋果": 100,
    "橘子": 20,
    "水梨": 50
}
...

```

```

# 印出蘋果的價格
print(dict01["蘋果"])

```

```

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

```

```

# 刪除 水梨
del dict01["水梨"]
print(dict01)

```


Module 3. 基礎回顧之流程控制與迴圈

參考資料

Python 基礎教學

<https://www.runoob.com/python/python-tutorial.html>

```
if ... else; if ... elif ... else
```

if 敘述

```
num = 10
```

```
...
```

可以用以下符號來進行條件判斷

== (等於)

> (大於)

>= (大於等於)

< (小於)

<= (小於等於)

!= (不等於)

```
...
```

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

if else 敘述

```
name = 'apple'
```

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

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")  
else:  
    print("Not found")
```

補充: True (真) 和 False (假、偽)

```
is_available = True  
  
if is_available == True:  
    print("真")  
else:  
    print("假")
```

也可以不用加「 == True」

```
if is_available:  
    print("真")  
else:  
    print("假")
```

while; for

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

for 迴圈 01

...

用法

```
range(n, m-1)
```

說明

會走訪 n 到 m-1 的數字

...

```
for i in range(5, 8):
    print(i, end = ",")
```

for 迴圈 02

...

用法

```
range(n, m-1, 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 = ",")
```

```
break; continue; pass
```

break

...

說明

當偵測到字母 t 時，就會強制結束迴圈

...

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

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(char)，

但不會結束迴圈，仍然會進入下一圈繼續執行

...

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

```

# 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 char in 'content':
    if char == 't':
        pass
    print(char, end="")

```

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

```

# 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)

```

透過指定索引輸出值

```

print( myTuple01[1] )
print( myTuple02[1] )

```

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))
```

```
# 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)
```

```
# 用 for 迴圈逐一輸出資料
```

```
for value in myTuple01:
    print(value)
```

```
# 用 len 計算 tuple 資料個數
```

```
print( len(myTuple02) )
```

```
# 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)
```

```
# 只能從記憶體刪除整個 tuple 變數
```

```
del myTuple01
print(myTuple01)
```

```
def
```

```
...
```

函式（其它程式語言常取名為 `function`）

說明

將經常使用的程式碼打包起來，變成一種模組，每執行一次函式，相當於把模組的程式碼全部執行過。

更進階的用法，可以參考這裡：

[1] [Python 教學]5 個必知的 Python Function 觀念整理

<https://www.learncodewithmike.com/2019/12/python-function.html>

...

'''不回傳 值'''

定義函式

```
def show_name():
    print("Doraemon")
```

執行函式

```
show_name()
```

'''回傳 值'''

定義函式

```
def get_name():
    name = "Doraemon"
    return name
# 也可以寫成 return "Doraemon"
```

執行函式

```
result = get_name()
```

輸出回傳結果

```
print(result)
```

'''傳遞變數'''

定義函式

```
def get_greeting(name):
    return "Hello, " + name
```

執行函式

```
result = get_greeting("Darren")
```

輸出回傳結果

```
print(result)

'''全域變數 & 區域變數'''

# 全域變數
name = "Bill"

# 定義函式
def set_name():
    # 區域變數
    name = "Doraemon"

# 執行函式
set_name()

# 輸出結果
print(name)
```

```
'''全域變數 & 區域變數'''

name = "Bill"

# 定義函式
def set_name():
    ...

    想在函式內部使用、修改外部變數的值，  

    必須在變數前加上 global 關鍵字，  

    但是不能在加上 global 的時候進行變數初始化
    ...

    global name
    name = "Doraemon"

# 執行函式
set_name()

# 輸出結果
print(name)
```

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)

基本概念

- 一筆筆讀資料：for 迴圈
- 總共有幾筆資料：len()
- 添加一筆資料：add()
- 添加多筆資料：update() (參數放的是 list)
- 查詢陣列中有沒有我要的資料：in (只回傳 true 或 false)
- 刪除元素：discard() or remove()
- 清空：clear()、del

```
# 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)
```

```
# 一筆筆讀資料 : for 迴圈
```

```
for data in mySet02:  
    print(data)
```

```
# 總共有幾筆資料: len()
```

```
print( len(mySet02) )
```

```
# 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] )
```

```
# 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)
```

```
# 添加多筆資料: update()
```

```
mySet01.update(["甲", "乙", "丙"])
mySet02.update(["子", "丑", "寅"])
```

```
print(mySet01)
print(mySet02)
```

```
# 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")
```

```
# 刪除元素: discard() or remove()
```

```
mySet01.discard("丙")
print(mySet01)
```

```
mySet02.remove("丑")
print(mySet02)
```

```
# 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)
```

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

```
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)
```

輸出結果

```
[{'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'}]
```

...

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)
```

輸出結果

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

```
('id', '380512239')
)
}

...
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)
```

輸出結果

```
[
    {'id': '380512238', 'link': 'https://stickershop.line-scdn.net/stick
ershop/v1/sticker/380512238/android/sticker.png'},
```

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

輸出結果

```
[  
{'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'}  
]
```

Module 4. 關於 URL 與字串格式化

產生網址

```
# 匯入套件
from urllib import parse

# 產生 url
string = 'https://www.104.com.tw/jobs/search/?'
query = {
    "keyword": "python",
    "order": 1,
    "jobsource": "2018indexpoc",
    "ro": 0}
result = string + parse.urlencode(query)
print(result)
```

...

憲法法庭 - 首頁

<https://cons.judicial.gov.tw/index.aspx>

憲法法庭 - 解釋

<https://cons.judicial.gov.tw/judcurrent.aspx?fid=2195>

設定 URL 範例 - 釋字第 791 號解釋

<https://cons.judicial.gov.tw/docdata.aspx?fid=100&id=310972>

...

#最簡單的方式：透過 formatted string

```
no = 310972
result = f"https://cons.judicial.gov.tw/docdata.aspx?fid=100&id={no}"
print(result)
```

...

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

...

```
# 編碼 parse.quote()
result01 = parse.quote('a=1&b=2')
print(result01)

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

字串格式化

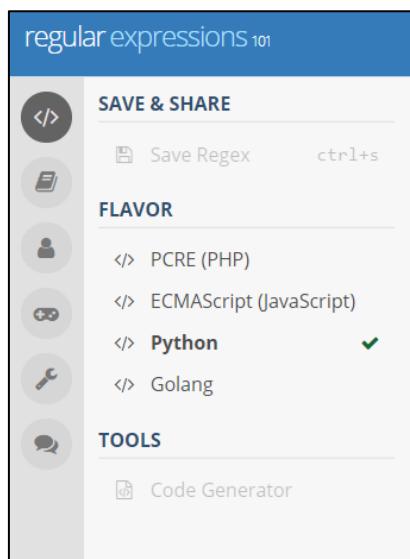
```
# 百分比 (%)
print('整數: %d' % 20) # 格式化整數
print('浮點數: %f' % 1.111) # 預設保留 6 位小數
print('浮點數: %.1f' % 1.111) # 取 1 位小數
print('My name is %s' % 'Darren') # 格式化字串

# str.format() => 用法: '{}'.format()
name = "Darren"
age = 18
str01 = "name: {}, age: {}".format(name, age)
print(str01)

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

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

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, the pattern is shown as a string of characters: "[a-zA-Z0-9_]+@[a-zA-Z0-9_.]+\.". The 'TEST STRING' field contains the email '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 '@', then another single character from the list [a-zA-Z0-9_.] one or more times. 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 'A single character of: ... [abc]' and 'A character except: ... [^abc]'. The top right of the interface includes links for '@regex101', '\$ donate', 'contact', 'bug reports & feedback', and 'wiki'.

圖：使用正規表達式，來判斷字串是否符合文字格式或條件

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]	abcdef
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\$
位於文字字元(\w)邊界的字元	\b ex. \bD	DarrenYang
非位於文字字元(\w)邊界的字元	\B ex. \Ba	DarrenYang
配對卻不在群組裡顯示	John (?:Cena)	John Cena

說明	正規表達式	範例
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

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

參考資料

1. Python3 正則表达式

<https://www.runoob.com/python3/python3-reg-expressions.html>

2. 正則表達式-全型英數中文字、常用符號 unicode 對照表

<https://blog.typeart.cc/%E6%AD%A3%E5%89%87%E8%A1%A8%E9%81%94%E5%BC%8F-%E5%85%A8%E5%9E%8B%E8%8B%B1%E6%95%B8%E4%B8%AD%E6%96%87%E5%AD%97%E3%80%81%E5%B8%B8%E7%94%A8%E7%AC%A6%E8%99%9Funicode%E5%B0%8D%E7%85%A7%E8%A1%A8/>

3. 匹配中文字符的正則表達式： [/u4e00-/u9fa5]

<https://www.itread01.com/content/1513168876.html>

4. 【Regular Expression】正向環視、反向環視

<https://toyo0103.blogspot.com/2017/01/regular-expression.html>

常用方法

汇入 regex 套件

import re

search

...

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...

...

regex01 = r'[a-zA-Z](\d{12})\d{8}'

string01 = "A123456789, S299888777"

match01 = re.search(regex01, string01)

print(match01)

print(match01[0])

```
print(match01[1])  
  
...  
補充：  
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))
```

```
#.findall  
...  
說明  
re.findall 會將所有配對到的字串  
回傳成一個 list  
...  
regex02 = r'[0-9]+'  
string02 = "0911111111, 0922222222, 0933333333"  
listMatch02 = re.findall(regex02, string02)  
print(listMatch02)  
print(listMatch02[0])  
print(listMatch02[2])
```

```
#.finditer  
...  
說明  
re.finditer 會將所有配對到的字串  
以迭代的方式呈現，若沒有配對到，則回傳 None  
...  
regex03 = r'[0-9]+'  
string03 = "0911111111, 0922222222, 0933333333"  
iterableMatch03 = re.finditer(regex03, string03)  
if iterableMatch03 != None:  
    for match in iterableMatch03:
```

```
print(match[0])
```

```
# match
```

```
'''
```

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`

```
'''
```

```
regex04 = r'2[0-9]{3}\/[0-1]?[0-9]{1}\/([0-3]?[0-9])'
string04 = "2022/06/30"
match04 = re.match(regex04, string04)
print(match04)
print(match04[0])
print(match04[1])
```

```
# split
```

```
'''
```

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`

```
'''
```

```
regex06 = r'\d'
string06 = "One1Two2Three3Four4"
listMatch06 = re.split(regex06, string06)
print(listMatch06)
```

```
# sub
```

```
'''
```

說明

`re.sub(regex, replace_string, test_string)`
將 `regex` 所代表的文字，改成 `replace_string`，文字來源是 `test_string`

```
'''
```

```

regex07 = r"\D"
string07 = "5-20 #1314"
strResult = re.sub(regex07, "", string07)
print(strResult)

```

環視

名稱	語法	說明
正向環視	(?=)	這位置右邊要出現什麼
正向環視否定	(?!)	這位置右邊不能出現什麼
反向環視	(?<=)	這位置左邊要出現什麼
反向環視否定	(?<!)	這位置左邊不能出現什麼

環視 (例如去除中文字旁邊的空白)

```

regex08 = r"\s(?! [a-zA-Z])" # 也可以寫成 r"(?![a-zA-Z])\s"
string08 = "一 天 一 蘋 果 醫 生 遠 離 我。An apple a day keeps the
doctor away."
strResult = re.sub(regex08, ' ', string08)
print(strResult)

```

環視 (加入千分位)

```

regex09 = r'(?=<\d)(?= (\d{3})+\b)'
string09 = '1234567890'
strResult = re.sub(regex09, ',', string09)
print(strResult)

```

具名群組

```
...
補充：  
除了 .group(n) 以外，  
還可以用 key 來代替 n。  
...  
  
# 身分證字號  
regex08 = r'[A-Z](?P<gender>[12])\d{8}'  
string08 = "A100000001"  
match08 = re.match(regex08, string08)  
  
# 完整配對的文字  
print(match08[0])  
print(match08.group(0))  
print(match08.group())  
  
# 具名(類似 key)所代表的值，也可以用索引代號來取得  
print(match08.group('gender'))  
print(match08['gender'])  
print(match08[1])
```

Module 7. HTML 基礎與 HTTP 方法

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>
          </ul>
        </nav>
      </header>
      <div class="content">
        <h1>歡迎來到我的家</h1>
        <p>這是一個簡單的 HTML 範例，展示了如何建立一個網站的基本結構。</p>
        <ul>
          <li>首頁</li>
          <li>連結 1</li>
          <li>連結 2</li>
        </ul>
      </div>
    </div>
  </body>
</html>
```

```

        </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 --&gt;
&lt;aside class="menu"&gt;
    &lt;ul class="menu-body"&gt;
        &lt;li class="menu-list"&gt;
            &lt;a class="center"&gt;側欄連結 1&lt;/a&gt;
        &lt;/li&gt;
        &lt;li class="menu-list"&gt;
            &lt;a class="center"&gt;側欄連結 2&lt;/a&gt;
        &lt;/li&gt;
        &lt;li class="menu-list"&gt;
            &lt;a class="center"&gt;側欄連結 3&lt;/a&gt;
        &lt;/li&gt;
    &lt;/ul&gt;
&lt;/aside&gt;

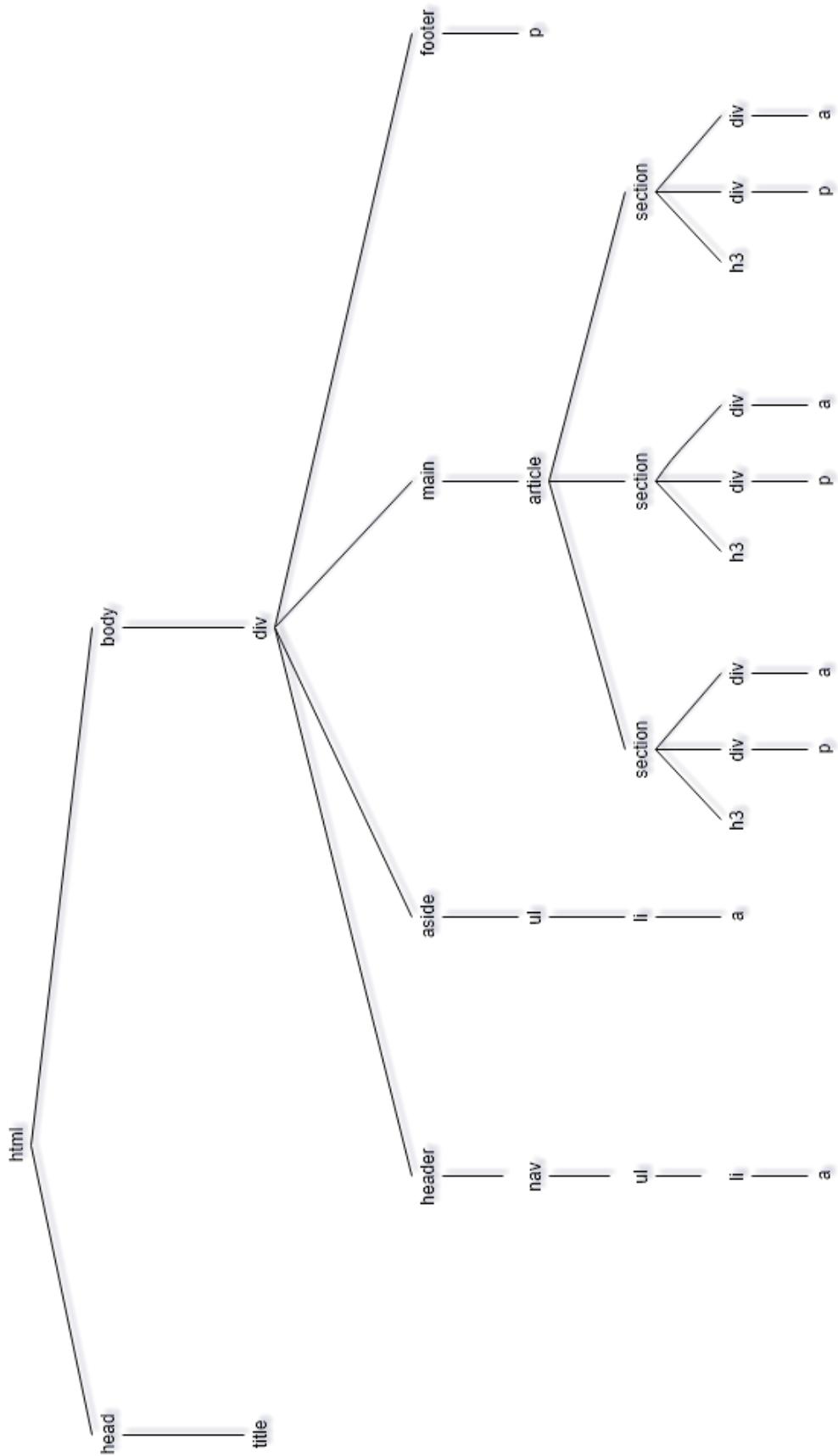
<!-- main 是主要顯示內容的區域 --&gt;
&lt;main class="content-container"&gt;
    &lt;article class="article-paragraph"&gt;
        &lt;section class="episode"&gt;
            &lt;h3 class="title"&gt;HTML parser 開發不求人 - 第三節&lt;/h3&gt;
            &lt;div class="content"&gt;
                &lt;p&gt;動態網頁的元素走訪，都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

```

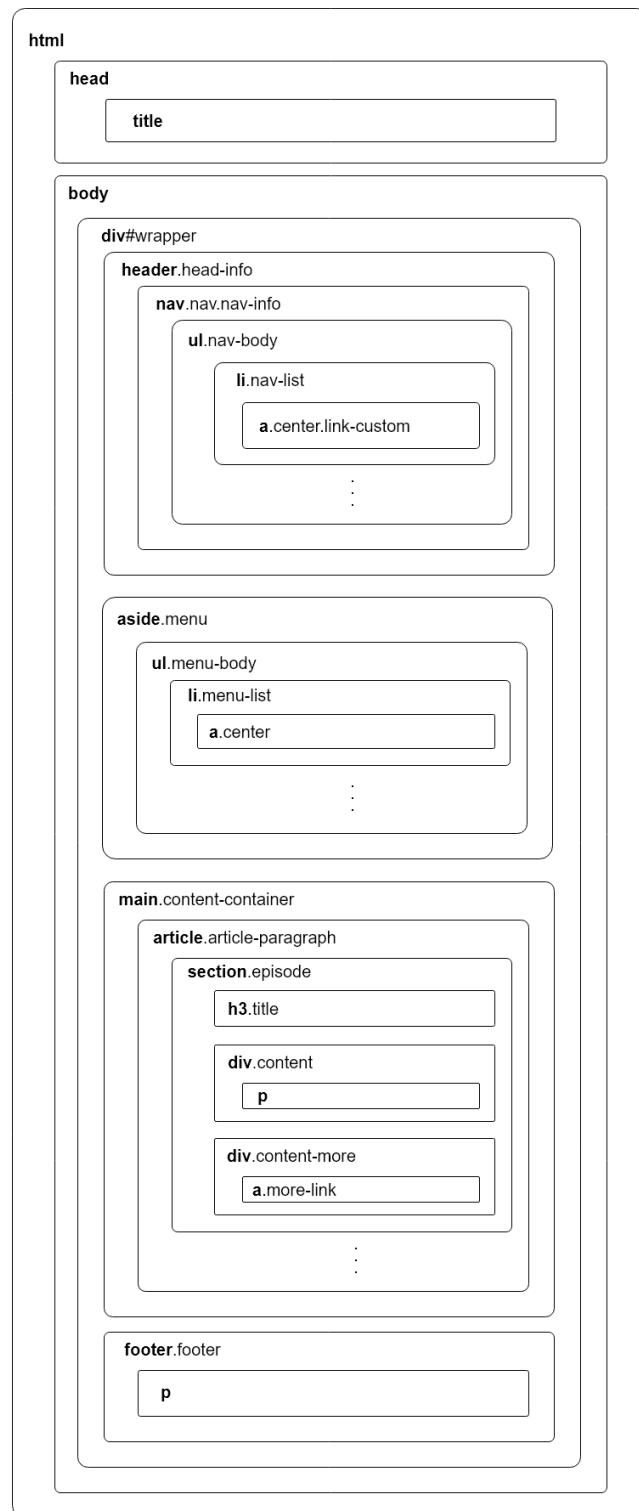
```
<div class="content">
    <p>動態網頁的元素走訪，都需要透過不斷地實作、練習，同時...</p>
</div>
<div class="content-more">
    <a class="more-link">More</a>
</div>
</section>
<section class="episode">
    <h3 class="title">HTML parser 開發不求人 - 第一節</h3>
    <div class="content">
        <p>動態網頁的元素走訪，都需要透過不斷地實作、練習，同時...</p>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

常見 HTTP 方法介紹

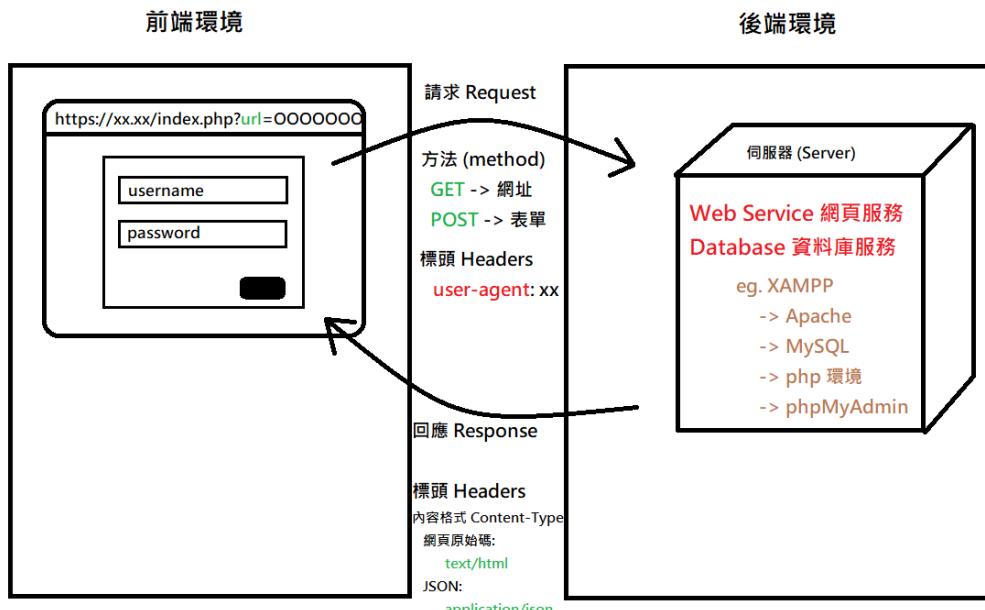
HTTP 請求方法在 RESTful API 中的典型應用				
資源	GET	PUT	POST	DELETE
一組資源的 URI，比如 <code>https://example.com/resource</code> s	列出 URI，以及該資源組中每個資源的詳細資訊（後者可選）。	使用給定的一組資源替換目前整個資源。	在本組資源中建立/追加一個新的資源。該操作往往返回新資源的 URL。	刪除整組資源。
單個資源的 URI，比如 <code>https://example.com/resource</code> s/142	取得指定的資源的詳細資訊，格式可以自選一個合適的網路媒體類型（比如：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。

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。



圖：請求(request)與回應(response)

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

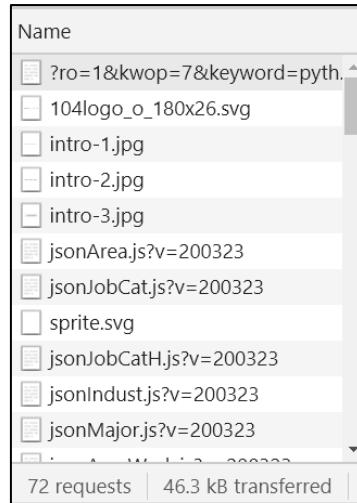
「？」符號，開啟跟著表單中的資料，每個欄位間的值，以「&」連接起來。

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&jobsourc=2018indexpoc>

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

A screenshot of the Headers tab in browser developer tools, focusing on the General section. It displays the following information:

- 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

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

A screenshot of the Headers tab in browser developer tools, focusing on the Query String Parameters section. It lists the following parameters:

- ro:** 1
- kwop:** 7
- keyword:** python
- order:** 13
- asc:** 0
- page:** 1
- mode:** s
- jobsouce:** 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。
- **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

概述 CSS Selector

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(元素)選擇器」、「類別選擇器」、「ID 選擇器」、「屬性選擇器」。

* : 通用選擇器(Universal selector)

將樣式套用於全部元素

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

element : 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>
```

element.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

#id : 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>
```

element#id : 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

element1, element2 ,element3,... : 群組選擇器(Grouped selector)

對 element1 、 element2 元素套用相同樣式

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

<div>Black</div>
<p>Black</p>
```

element element : 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>
```

element > element : 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>
```

element1 + element2 : 相鄰元素選擇器(Adjacent sibling selector)

若元素為 element2，且前面有 element1 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

.class

.intro

選擇 class="intro" 的所有元素。

補充說明

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

補充說明

選擇器	例子	例子描述
<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。
<u>[attribute]</u>	[target]	選擇帶有 target 屬性的所有元素。
<u>[attribute=value]</u>	[target=_blank]	選擇帶有 target="_blank" 屬性的所有元素。
<u>[attribute~=value]</u>	[title~=flower]	選擇 title 屬性包含單詞 "flower" 的所有元素。
<u>[attribute^=value]</u>	a[href^="https"]	選擇其 href 屬性值以 "https" 開頭的每個 <a> 元素。

補充說明

選擇器	例子	例子描述
<u>[attribute\$=value]</u>	a[href\$=".pdf"]	選擇其 href 屬性以 ".pdf" 結尾的所有 <a> 元素。

參考資料：

CSS 選擇器參考手冊

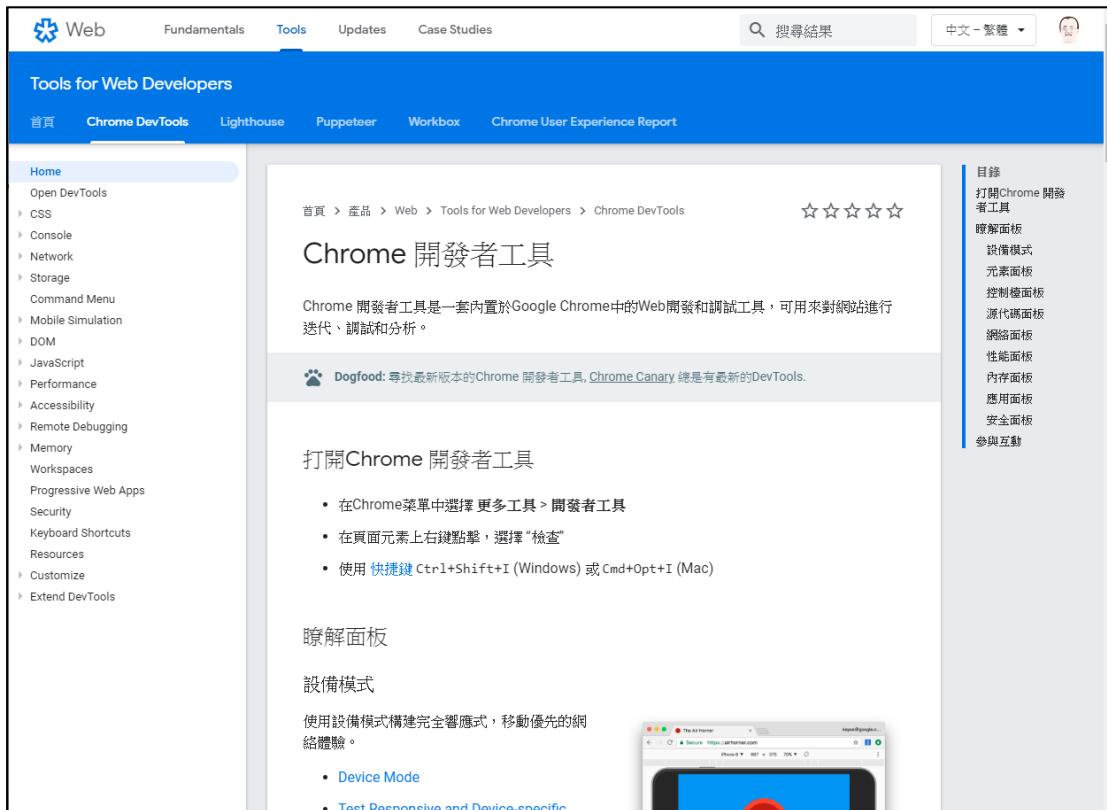
https://www.w3school.com.cn/cssref/css_selectors.asp

Module 9. Chrome Developer Tool

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

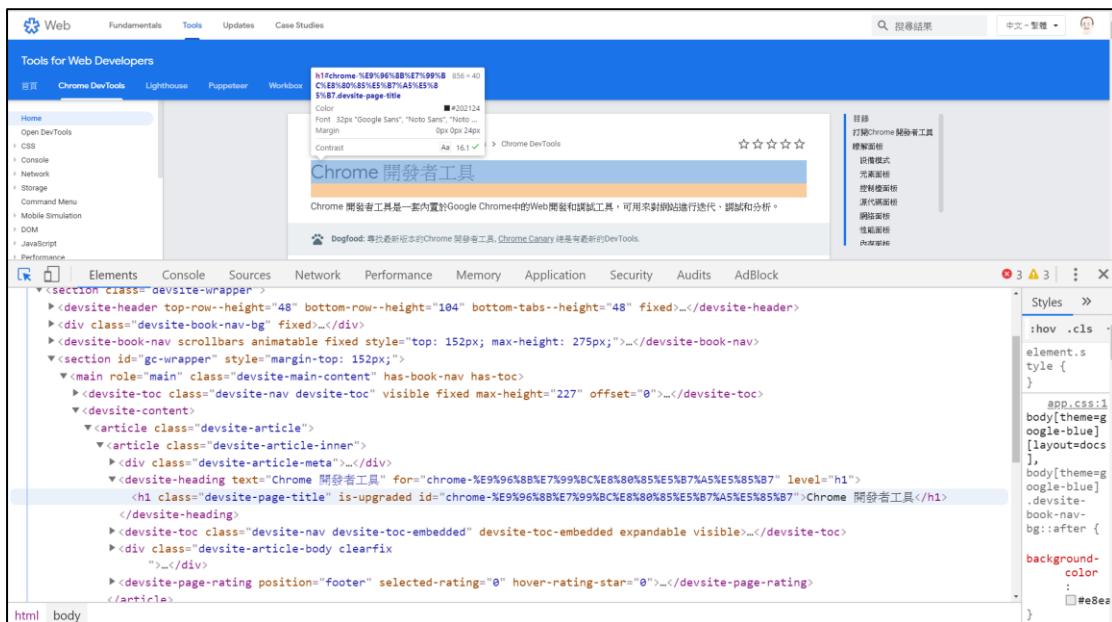
開啟開發工具(dock)

- F12

Elements 面板

檢查 HTML 元素

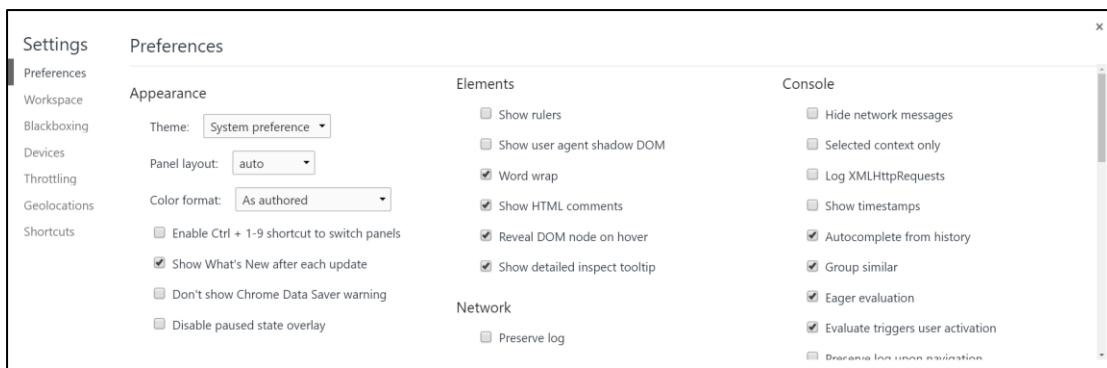
- Ctrl + Shift + C (追縱滑鼠移過網頁元素所在位置的狀態)
- 網頁內容任意處按滑鼠右鍵→檢查



圖：檢查元素

補充說明

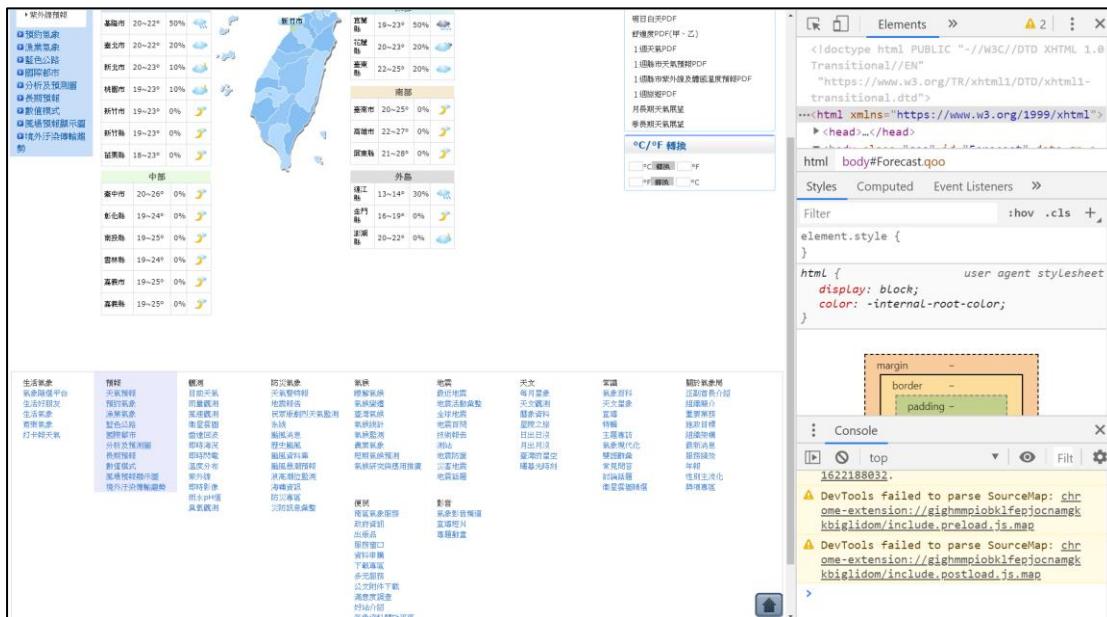
開啟 Chrome 開發者工具以後，按下 F1，可以看到一些偏好設定，方便我們設定開發工具，例如顯示外觀、模擬裝置、自訂地理位置、快捷鍵等。



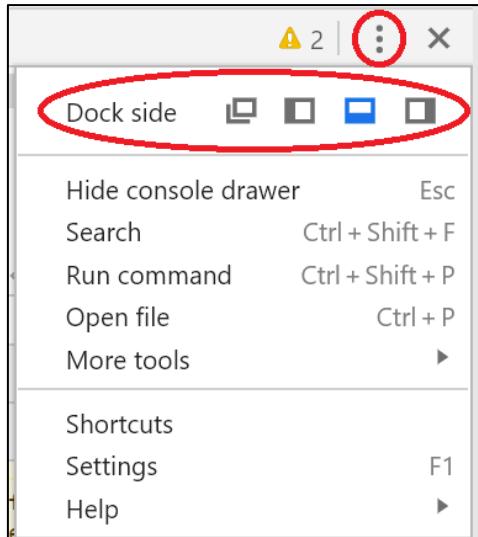
圖：Chrome 開發者工具偏好設定

開啟開發工具後，常用快速鍵：

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

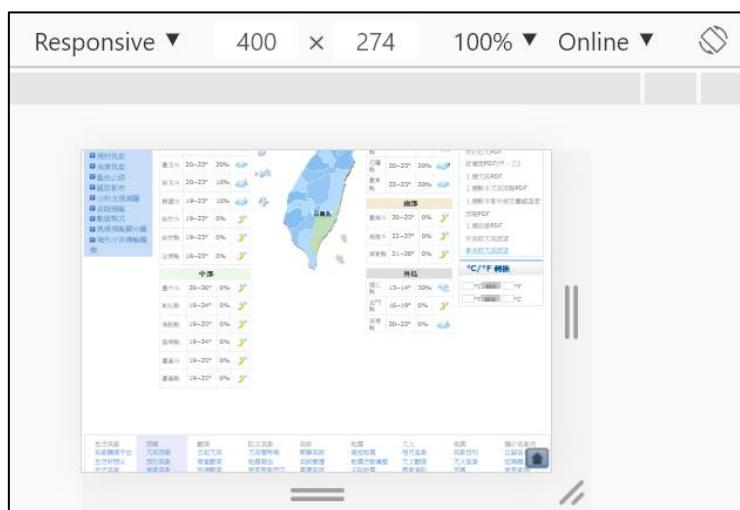


圖：切換 dock side，從下方到右側

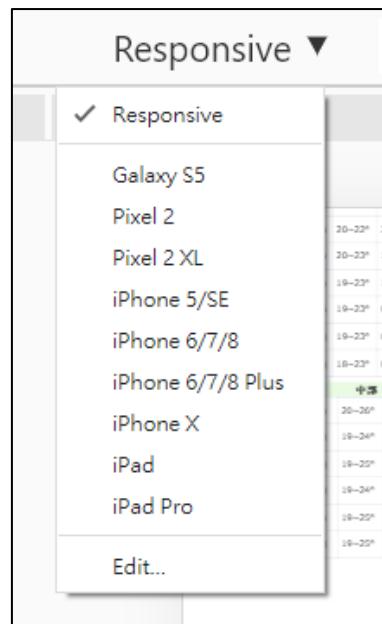


圖：按下三個點的圖示，也可以選擇 dock side

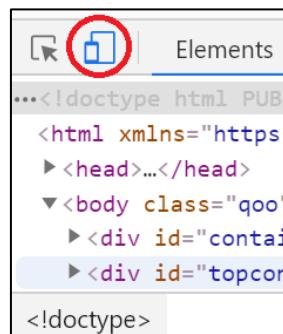
- **Ctrl + Shift + M** 開啟模擬裝置模式(切換裝置工具欄)



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



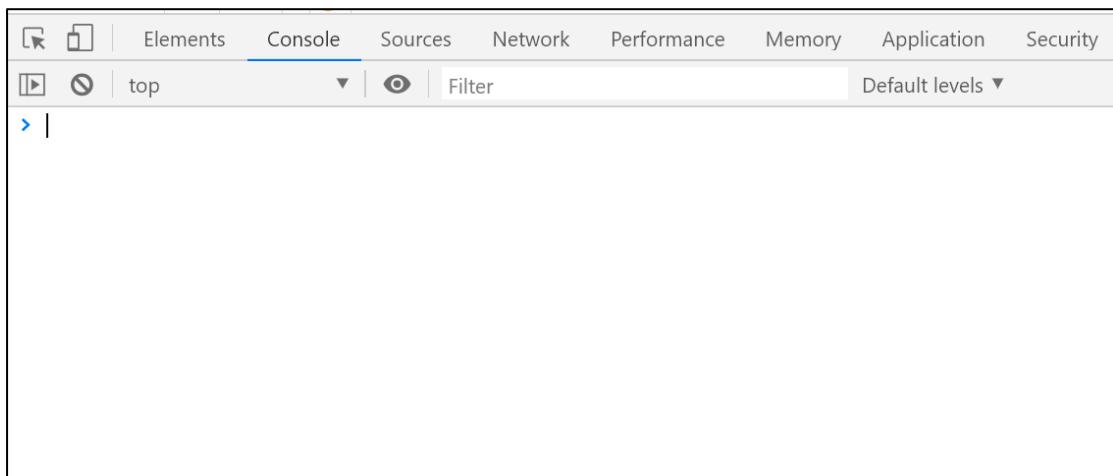
圖：等同按下切換裝置工具欄

- 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

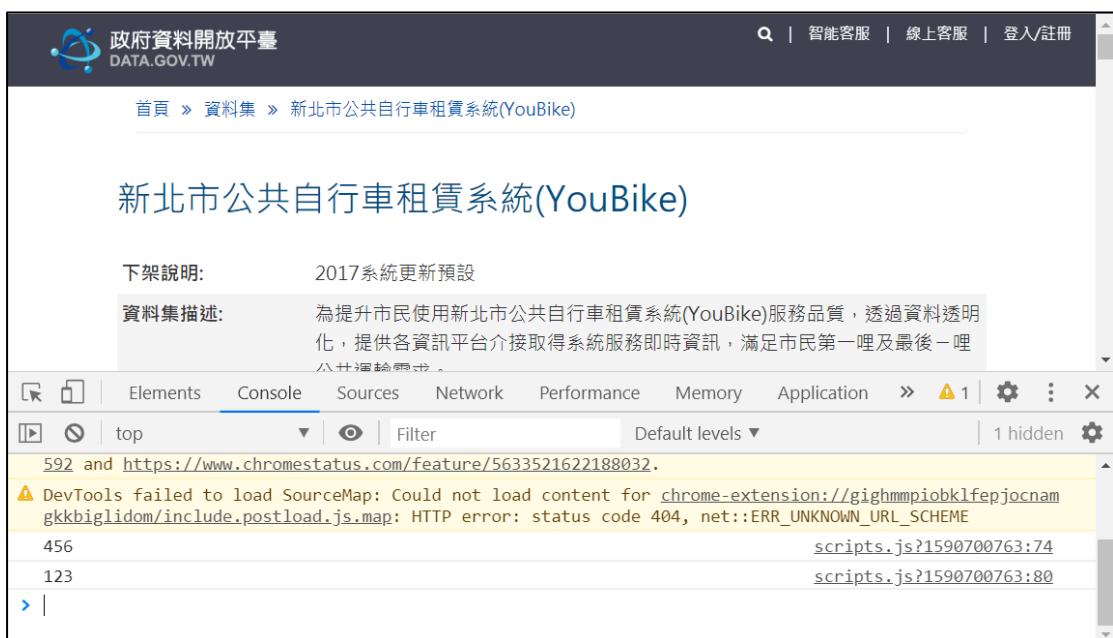
- Shift + Enter 在 Console 中斷行(或多行)

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



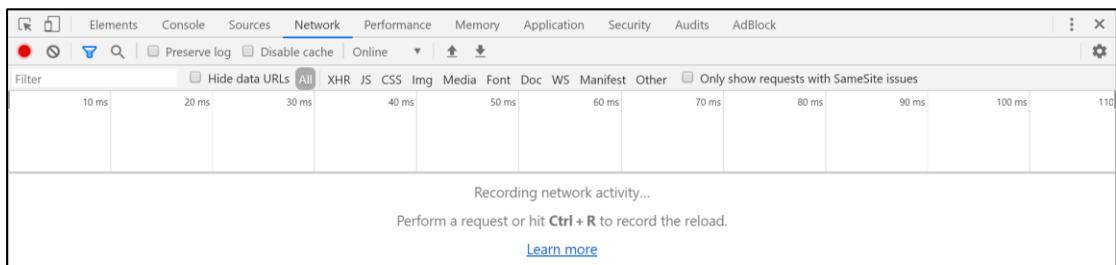
圖：可以看到目前網站的情況。

Network 面板

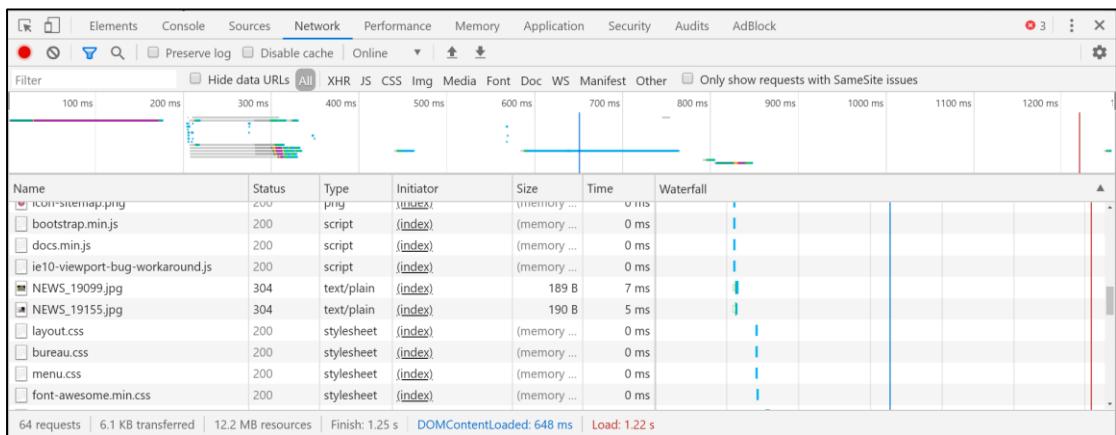
Network 面板會顯示出所有網路請求的詳細訊息記錄，包括狀態、資源類型、

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。



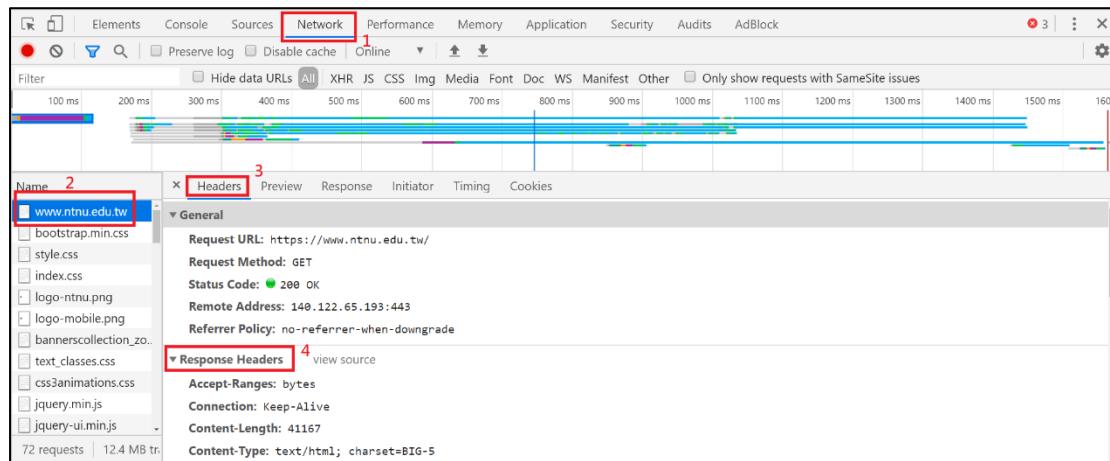
(圖) Network 面板會記錄任何的網路活動



(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



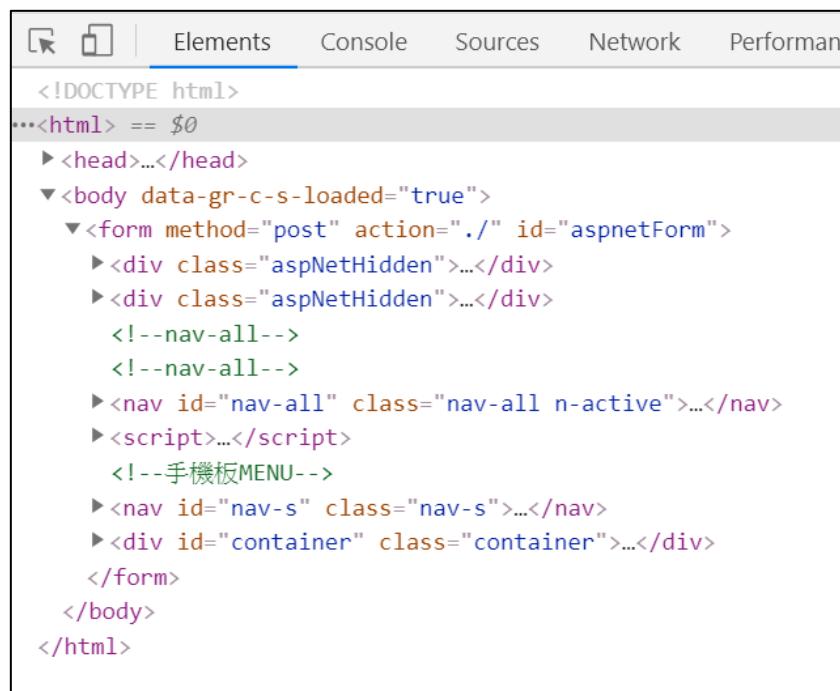
圖：觀看檔案的 Headers 內容

Request Headers (請求標頭，參考[維基百科](#))

標頭欄位	說明	範例
Cookie	之前由伺服器通過 Set-Cookie 傳送的一 個 超文字傳輸協定 Cookie	<p>Cookie:</p> <pre>_ga=GA1.3.1322956465.1572335045;loc ale=zh_TW;</pre>

標頭欄位	說明	範例
		<pre>_gid=GA1.3.1110994946.1584940974; _gat_gtag_UA_141775379_1=1</pre>
User-Agent	瀏覽器的瀏覽器身 分標識字串	<pre>User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/21.0</pre>

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



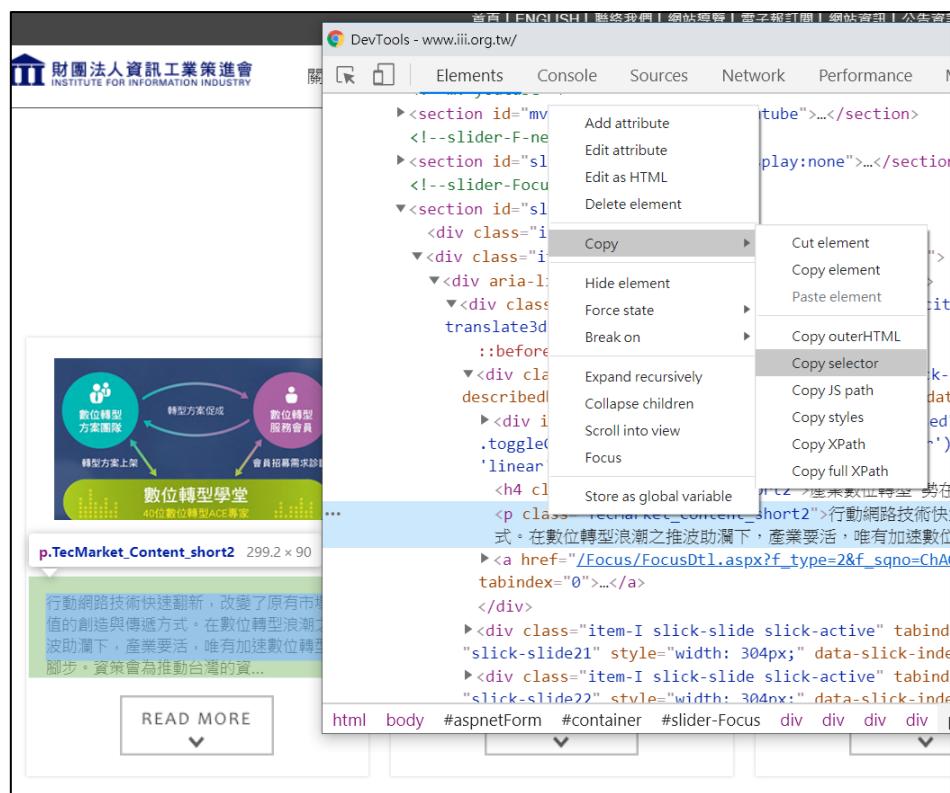
圖：焦點報導有三個元素



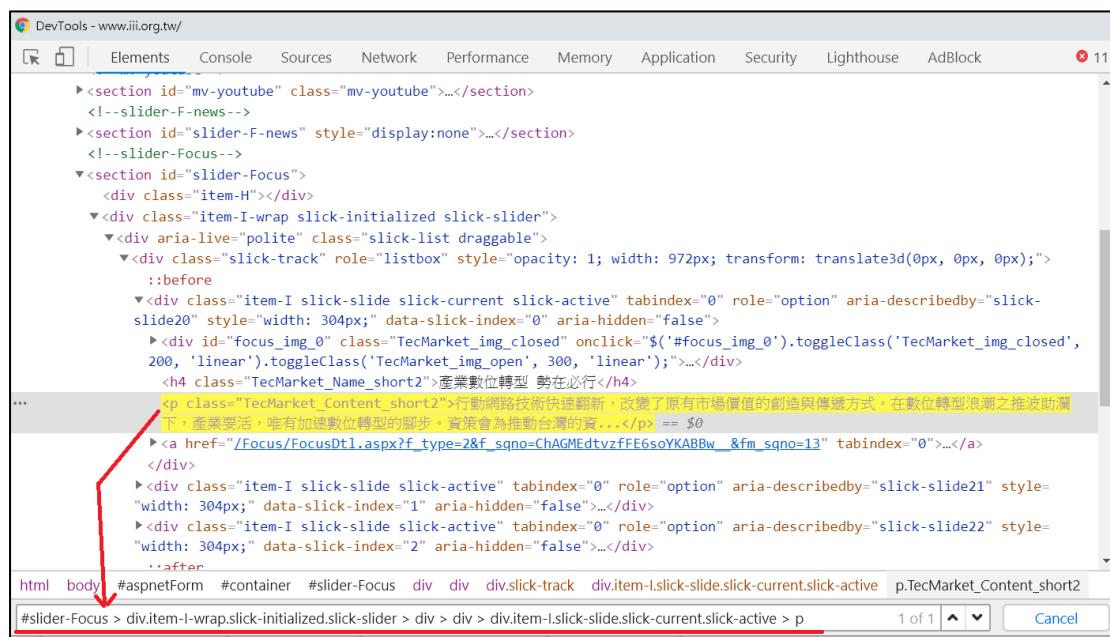
圖：報導包在 `div.item-I-wrap.slick-initialized.slick-slider` 裡面



圖：若是想要取得文字區塊的選擇器

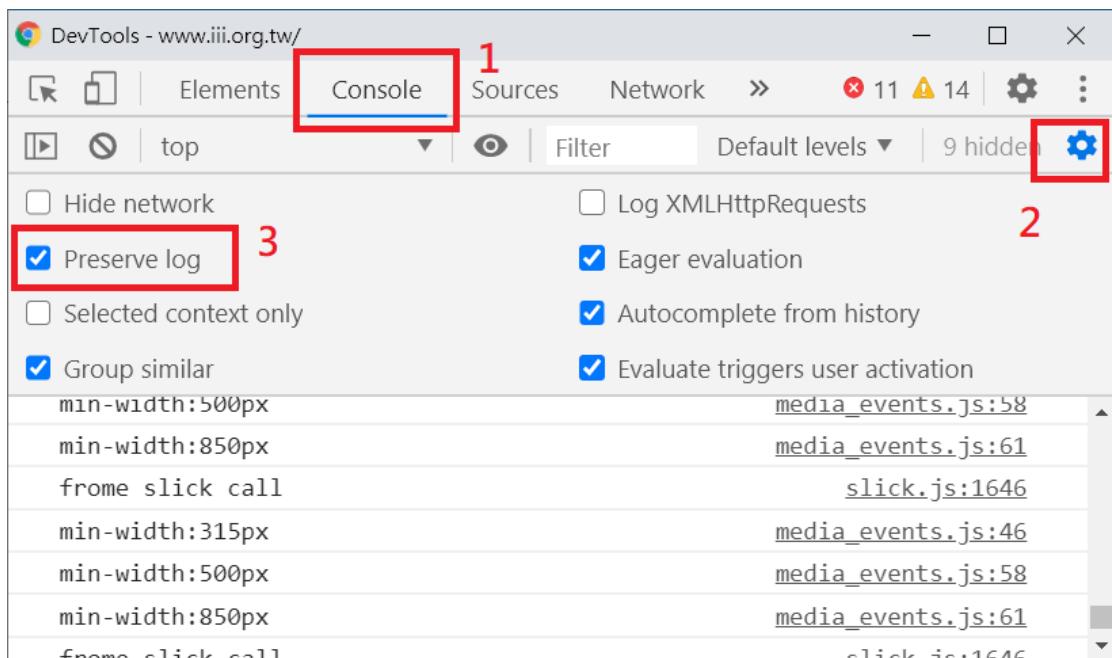


圖：對著 Elements 面板的元素區塊按右鍵→Copy→Copy Selector

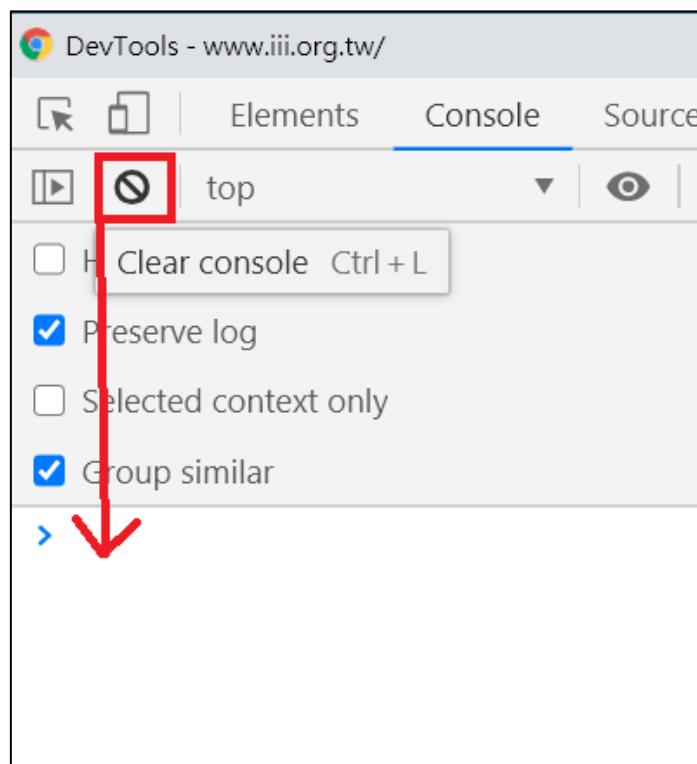


圖：對 elements 面板按 Ctrl+F，將剛才的 CSS selector 貼上，得到驗證

常用操作流程介紹 (Preserve Log / Clear / ...)



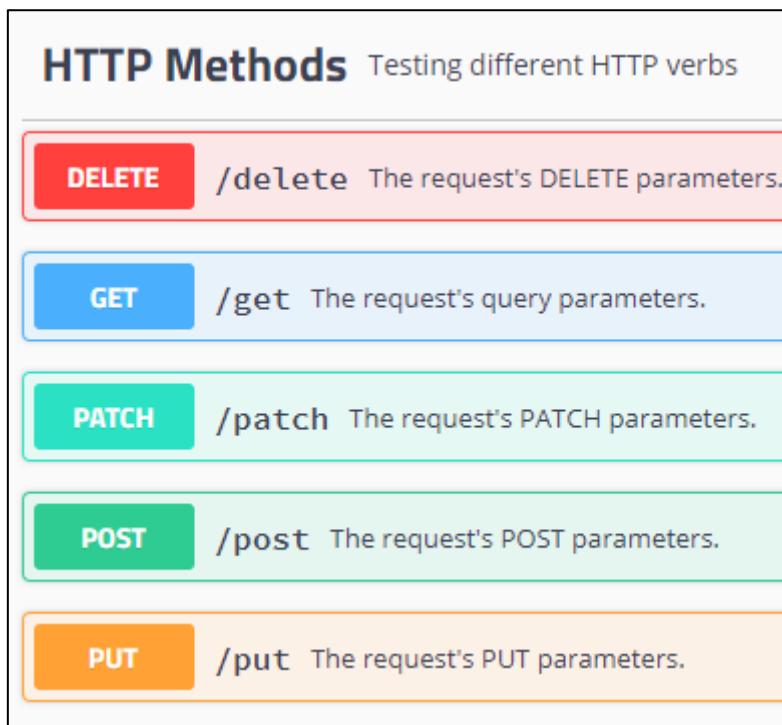
圖：勾選 Preserve log，縱然頁面刷新，過去的 log 依然會保存起來



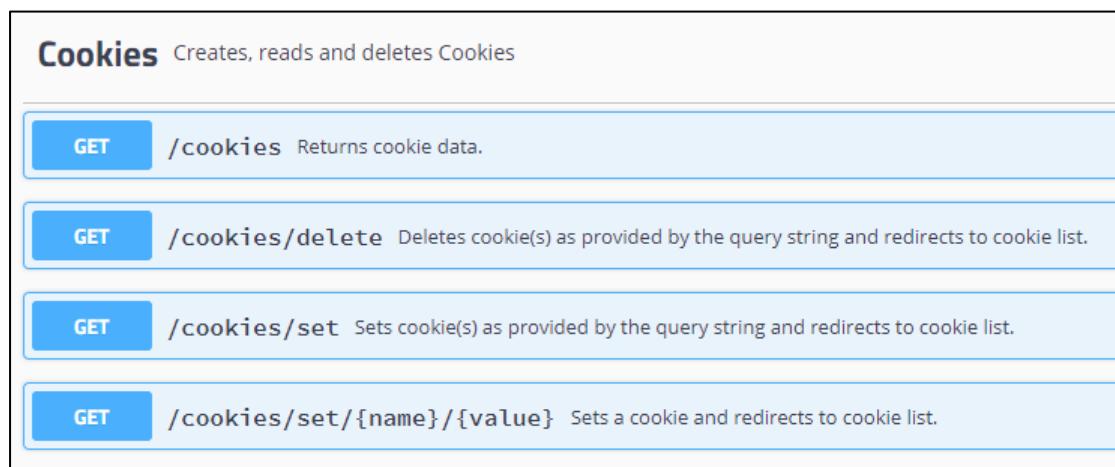
圖：按下 Clear Console，或是 Ctrl+L，即可清除 log

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

```
requests 套件
```

```
# 使用 requests 工具
```

```
import requests
```

```
# 使用 json 工具
```

```
import json
```

```
# 使用 GET 方式下載普通網頁
```

```
res = requests.get('https://httpbin.org/get')
```

```
# 伺服器回應的狀態碼
```

```
# 參考網頁: https://reurl.cc/2DRpan
```

```
print(res.status_code)
```

```
# 回傳資料的編碼
```

```
print(res.encoding)
```

```
# 指定回傳資料的編碼
```

```
# response.encoding = 'utf-8'
```

```
# 輸出網頁 HTML 原始碼
```

```
print(res.text)
```

```
# GET 方法的 query string
```

```
my_params = {
```

```
    'key1': 'value1',
```

```
    'key2': 'value2'
```

```
}
```

```
# 將 query string 加入 GET 請求中
```

```
res = requests.get('https://httpbin.org/get', params = my_params)
```

```
# 觀察 URL
```

```
print(res.url)
```

```

# 輸出網頁 HTML 原始碼
print(res.text)

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

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

# 輸出網頁 HTML 原始碼
print(res.text)

# 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
res = requests.post('https://httpbin.org/post', files = my_files)

# 輸出網頁 HTML 原始碼
print(res.text)

# 自訂標頭
my_headers = {
    'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.82 Safari/537.36'
}

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

# 輸出網頁 HTML 原始碼
print(res.text)

# 自訂 cookie 格式
my_cookies = {

```

```

    "first_cookie": "hello",
    "second_cookie": "world"
}

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

# 輸出網頁 HTML 原始碼
print(res.text)

# 參考網址: https://data.taipei/#/application
url = 'https://data.taipei/api/v1/dataset/d706f428-b2c7-4591-9ebf-
9f5cd7408f47?scope=resourceAquire'
res = requests.get(url)

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

# 輸出對應節點的文字
print(obj['result']['limit'])
print(obj['result']['offset'])

# 輸出部分節點的文字
for o in obj['result']['results']:
    print(f"_id: {o['_id']}")
    print(f"餐廳名稱: {o['餐廳名稱']}") 
    print(f"餐館電話: {o['餐館電話']}") 
    print(f"餐館地址: {o['餐館地址']}") 
    print("=" * 50)

```

Module 11. 套件 BeautifulSoup 4

套件介紹及常用功能

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 (取得單一元素)
- `find_all()` 方法 (取得元素集合)
- `select_one()` 方法 (取得單一元素)
- `select()` 方法 (取得元素集合)

BeautifulSoup 基本用法

`soup.select()` :

回傳的結果是元素集合 (list 型態，BeautifulSoup ResultSet)

`soup.select_one()` :

回傳的結果是單一元素 (BeautifulSoup Result)

Beautifulsoup 套件

...

參考網頁

[1] Python 使用 Beautiful Soup 抓取與解析網頁資料，開發網路爬蟲教學
<https://blog.gtwang.org/programming/python-beautiful-soup-module-scrape-web-pages-tutorial/2/>

...

```
import requests as req
from bs4 import BeautifulSoup as bs
from pprint import pprint

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

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

# 指定 lxml 作為解析器
soup = bs(res.text, "lxml")

# 第一個 <a></a>
print(soup.find("a"))

# 全部 <a></a>，此時回傳 list
print(soup.find_all("a"))

# 指定 list 某個元素的 html
print(soup.find_all("a")[2])

# 取得 id 為 logo 的元素
logo = soup.find(id = "logo")
print(logo)

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

...
以下透過 CSS selector 取得元素，  
回傳格式為 list
...
# 輸出 title
```

```

print(soup.select_one('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_one("#logo"))

# 透過 attribute 取得元素
print(soup.select('a[class="board"]'))

# 取得單一節點的文字內容 (select_one 會回傳單一 bs element 物件，select 會回傳 list)
print(soup.select_one('title').get_text())
print(soup.select('a')[0].get_text())

# 透過迭代取得所有 a 的文字內容
for a in soup.select('a'):
    print(a.get_text())

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

```

Module 12. cookie 用於 requests

以 PTT Gossiiping (八卦版) 為例

```
import requests as req
from bs4 import BeautifulSoup as bs

# PTT Gossiiping (八卦版)
url = "https://www.ptt.cc/bbs/Gossiping/index.html"

# 首頁網址
prefix = 'https://www.ptt.cc'

# 設定 cookie
my_cookies = {
    "over18": "1"
}

# 用 requests 的 get 方法把網頁抓下來
res = req.get(url, cookies = my_cookies)

# 指定 lxml 作為解析器
soup = bs(res.text, "lxml")

# 顯示連結列表
for a in soup.select('div.r-ent > div.title > a'):
    print(a.get_text())
    print(prefix + a['href'])
```

Module 13. 案例: PTT_NBA_看板主頁與內頁

取得 PTT NBA 列表

```
# 匯入套件
from bs4 import BeautifulSoup as bs
import requests as req
from pprint import pprint

# 取得新聞列表
url = "https://www.ptt.cc/bbs/NBA/index.html"

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

# 指定 lxml 作為解析器
soup = bs(res.text, "lxml")

# 建立 list 來放置列表資訊
list_posts = []

# 取得 列表 的文字與超連結
for a in soup.select('div.r-ent div.title a[href]:'):
    print(a.get_text())
    print(a['href']) # 或是 a.get('href')

# 加入列表資訊
list_posts.append({
    'title': a.get_text(),
    'link': 'https://www.ptt.cc' + a['href']
})

# 走訪每一個 a link，整合網頁內文
for index, obj in enumerate(list_posts):
    res_ = req.get(obj['link'])
    soup_ = bs(res_.text, "lxml")

    # 去掉 div.article-metaline (作者、標題、時間...等)
    for div in soup_.select('div[class^="article-metaline"]'):
        div.decompose()
```

```

# 去掉實際文章網址超連結（判斷去掉元素是否存在）
if len(soup_.select('span.f2 + a[href][rel="nofollow"]')) > 0:
    soup_.select_one('span.f2 + a[href][rel="nofollow"]').decompose()

# 去掉 span.f2 (發信 IP、文章網址、編輯...等)
for span in soup_.select('span.f2'):
    span.decompose()

# 去掉 div.push (推文：推、→、噓)
for div in soup_.select('div.push'):
    div.decompose()

# 取得實際需要的內容（類似 JavaScript 的 innerHTML）
html = soup_.select_one('div#main-content').decode_contents()
# html = str(soup_.select_one('div#main-content')) # 類似
JavaScript outerHTML

# 預覽列表和內文
# print(obj['title'])
# print(obj['link'])
# print(html)
# print("=" * 50)

# 整合到列表資訊的變數當中
list_posts[index]['content'] = html

# 預覽所有結果
# pprint(list_posts)

```

思考

- 如何取得多個分頁的內容？
 - 觀察分頁數字在網址的呈現方式
 - 將觀察到的分頁數字嵌入對應的網址當中

```

# 清空放置列表資訊的變數
list_posts.clear()

# 起始頁數
init_page = 6503

# 最新頁數
latest_page = 6505

# 在已經知道分頁數的情況下
for page in range(init_page, latest_page + 1):

    # 取得新聞列表
    url = f"https://www.ptt.cc/bbs/NBA/index{page}.html"

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

    # 指定 lxml 作為解析器
    soup = bs(res.text, "lxml")

    # 取得 列表 的文字與超連結
    for a in soup.select('div.r-ent div.title a[href]'):

        # 加入列表資訊
        list_posts.append({
            'title': a.get_text(),
            'link': 'https://www.ptt.cc' + a['href']
        })

    # 走訪每一個 a link，整合網頁內文
    for index, obj in enumerate(list_posts):
        res_ = req.get(obj['link'])
        soup_ = bs(res_.text, "lxml")

        # 去掉 div.article-metaline (作者、標題、時間...等)
        for div in soup_.select('div[class^="article-metaline"]'):
            div.decompose()

```

```
# 去掉實際文章網址超連結（判斷去掉元素是否存在）
if len(soup_.select('span.f2 + a[href][rel="nofollownoopener noreferrer"]')) > 0:
    soup_.select_one('span.f2 + a[href][rel="nofollownoopener noreferrer"]').decompose()

# 去掉 span.f2 (發信 IP、文章網址、編輯...等)
for span in soup_.select('span.f2'):
    span.decompose()

# 去掉 div.push (推文: 推、→、噓)
for div in soup_.select('div.push'):
    div.decompose()

# 取得實際需要的內容 (類似 JavaScript 的 innerHTML)
html = soup_.select_one('div#main-content').decode_contents()
# html = str(soup_.select_one('div#main-content')) # 類似
JavaScript outerHTML

# 整合到列表資訊的變數當中
list_posts[index]['content'] = html

# 預覽所有結果
pprint(list_posts)
```

Module 14. 套件 Selenium (一)

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。

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

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

selenium 套件

...

參考網頁：

[1] 下載 Chrome Web Driver

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

...

操作 browser 的 API

```
from selenium.webdriver.chrome.service import Service
```

```
from selenium import webdriver
```

匯入套件

```
from bs4 import BeautifulSoup as bs
```

強制等待 (執行期間休息一下)

```
from time import sleep
```

使用 Chrome 的 WebDriver

```
my_service = Service(executable_path=".chromedriver.exe")
```

```
driver = webdriver.Chrome(service=my_service)
```

開啟 104 人力行銀 首頁

```
driver.get("https://www.104.com.tw/jobs/main/")
```

```
# 取得檢視原始碼的內容 (page_source 取得的 html，是動態的、使用者操作過後的結果)
html = driver.page_source

# 印出 html (也可以跟 BeautifulSoup 整合)
# print(html)

# 指定 lxml 作為解析器
soup = bs(html, "lxml")

# 取得 上方選單 元素
ul = soup.select_one('ul.header_persona-list')

# 顯示內文
print(ul.get_text())

# 休眠幾秒
sleep(3)

# 關閉瀏覽器
driver.quit()
```

Module 15. 套件 Selenium (二)

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

註: 新的 webdriver 版本已不支援 `find_element(s)_by_xxx()` 系列的語法

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

圖：取得網頁元素方法

匯入自動測試工具相關套件
...
匯入套件
...
操作 browser 的 API
<code>from selenium.webdriver.chrome.service import Service</code>
<code>from selenium import webdriver</code>

```
# 處理逾時例外的工具

from selenium.common.exceptions import TimeoutException

# 面對動態網頁，等待某個元素出現的工具，通常與 expected_conditions 搭配

from selenium.webdriver.support.ui import WebDriverWait

# 搭配 WebDriverWait 使用，對元素狀態的一種期待條件，若條件發生，則等待結束，往下一行執行

from selenium.webdriver.support import expected_conditions as EC

# 期待元素出現要透過什麼方式指定，通常與 EC、WebDriverWait 一起使用

from selenium.webdriver.common.by import By

# 強制等待（執行期間休息一下）

from time import sleep

# 整理 json 使用的工具

import json

# 執行 command 的時候用的

import os

...

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

my_options = webdriver.ChromeOptions()
# my_options.add_argument("--headless")          #不開啟實體瀏覽器背景執行
my_options.add_argument("--start-maximized")      #最大化視窗
my_options.add_argument("--incognito")            #開啟無痕模式
my_options.add_argument("--disable-popup-blocking") #禁用彈出攔截
my_options.add_argument("--disable-notifications") #取消 chrome 推播通知
my_options.add_argument("--lang=zh-TW")           #設定為正體中文


# 使用 Chrome 的 WebDriver

my_service = Service(executable_path="./chromedriver.exe")
driver = webdriver.Chrome(
    options = my_options,
    service = my_service
)

```

在瀏覽器中執行自訂 JavaScript 程式

```

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

# 跳出 alert 視窗 (在 chrome 裡面執行 javascript 語法)
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()

```

輸入文字，送出表單

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.CSS_SELECTOR, 'input#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.presence_of_element_located(
            (By.CSS_SELECTOR, cssSelector)
        )
    )

    # 取得第一頁搜尋結果
    element = driver.find_element(By.CSS_SELECTOR, cssSelector)

    # 輸出想要爬取的文字
    print(element.text)
    print(element.get_attribute('innerText')) # 另一種寫法

    # 睡個幾秒
    sleep(3)
except TimeoutException:
    print('等待逾時！')
```

輸入文字，按下送出鈕

```
# 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.CSS_SELECTOR, 'input#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:
    # 等待元素
    WebDriverWait(driver, 10).until(
        EC.presence_of_element_located(
            (By.CSS_SELECTOR, cssSelectorBtn)
        )
    )

    # 取得按鈕元素
    btn = driver.find_element(By.CSS_SELECTOR, cssSelectorBtn)

    # 按下按鈕
    btn.click()

    # 睡個幾秒
    sleep(3)
except TimeoutException:
    print('等待逾時！')
```

刷新頁面 (類似 F5 或 Ctrl + R)

```
# 開啟網頁
driver.get("https://reurl.cc/jR725D")

# 睡個幾秒
sleep(5)

# 刷新頁面
driver.refresh()

# 睡個幾秒
sleep(5)

# 刷新頁面
driver.refresh()
```

關閉瀏覽器

```
# 關閉瀏覽器
driver.quit()
```

Module 16. 套件 Selenium (三)

等待 (WebDriverWait)

等待有分幾種：

- 強制等待
 - 通常泛指 sleep() 函式
- 隱性等待 (implicitly_wait)
 - 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，或是能夠取得指定的元素 (透過 find_element*)，則執行下一步，否則一直到時間截止，然後拋出例外。
- 顯性等待 (WebDriverWait)
 - 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 。

期待狀況/條件 (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 樹裡，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

元素定位策略/方式 (By)

- By.ID = "id"
- By.CSS_SELECTOR = "css selector"
- 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"

匯入自動測試工具相關套件

...

參考網址：

[1] Webdriver Manager for Python

<https://pypi.org/project/webdriver-manager/>

...

匯入套件

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.common.exceptions import TimeoutException,
NoSuchElementException
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
```

強制等待

```
...
強制等待
...

# 自動取得 Chrome 的 WebDriver
driver = webdriver.Chrome(
    service = Service(ChromeDriverManager().install())
)

try:
    # 走訪網址
    driver.get('https://tw.yahoo.com/')

    # 強制等待 3 秒
    sleep(3)

    # 印出網址
    print(driver.current_url)
except:
    print("程式出錯!")
finally:
    # 關閉瀏覽器
    driver.quit()
```

隱性等待

```
...
隱性等待
...

# 自動取得 Chrome 的 WebDriver
driver = webdriver.Chrome(
    service = Service(ChromeDriverManager().install())
)

try:
    # 最多等 15 秒
    driver.implicitly_wait(15)
```

```
# 走訪網址
driver.get('https://tw.yahoo.com/')

# 取得元素
element = driver.find_element(By.CSS_SELECTOR, 'a#header-logo')

# 印出超連結
print(element.get_attribute('href'))
except NoSuchElementException:
    print("找不到元素!")
finally:
    # 關閉瀏覽器
    driver.quit()
```

顯性等待

...

顯性等待

...

```
# 自動取得 Chrome 的 WebDriver
driver = webdriver.Chrome(
    service = Service(ChromeDriverManager().install())
)

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

    # 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()
```

Module 17. ActionChains

匯入工具

加入行為鍊 ActionChains (在 WebDriver 中模擬滑鼠移動、點擊、拖曳、按右鍵出現選單，以及鍵盤輸入文字、按下鍵盤上的按鈕等)

```
from selenium.webdriver.common.action_chains import ActionChains
```

基本用法

方法	說明
click(on_element=None)	單擊滑鼠左鍵
click_and_hold(on_element=None)	點選滑鼠左鍵，不鬆開
context_click(on_element=None)	點選滑鼠右鍵
double_click(on_element=None)	雙擊滑鼠左鍵
drag_and_drop(source, target)	拖拽到某個「元素」然後鬆開
drag_and_drop_by_offset(source, xoffset, yoffset)	拖拽到某個「座標」然後鬆開
key_down(value, element=None)	按下某個鍵盤上的鍵
key_up(value, element=None)	鬆開某個鍵
move_by_offset(xoffset, yoffset)	滑鼠從當前位置移動到某個座標
move_to_element(to_element)	滑鼠移動到某個元素
move_to_element_with_offset(to_element, xoffset, yoffset)	移動到距某個元素（左上角座標）多少距離的位置
pause(seconds)	暫停動作一段時間
perform()	執行鏈中的所有動作
release(on_element=None)	在某個元素位置鬆開滑鼠左鍵
send_keys(keys_to_send)	傳送某個鍵到當前焦點的元素
send_keys_to_element(element, keys_to_send)	傳送某個鍵到指定元素

寫法

- 鍊式

```
ActionChains(driver).move_to_element( web_element ).click( web_element ).perform()
```

或是

```
action_chains = ActionChains(driver)

action_chains.move_to_element( web_element ).click( web_element ).perform()
```

- 分步

```
action_chains = ActionChains(driver)

action_chains.move_to_element( web_element )

action_chains.click( web_element )

action_chains.perform()
```

補充: 切換目前到 iframe 當中

- 取得網頁上的 iframe

```
iframe = driver.find_element(By.CSS_SELECTOR, "iframe#game-iframe")
```

- 切換到 iframe 當中

```
driver.switch_to.frame(iframe)
```

- 回到主框架

```
driver.switch_to.default_content()
```

參考連結

1. Selenium 的 ActionChains Api 介面詳解

<https://www.796t.com/article.php?id=325399>

2. python selenium 滑鼠鍵盤操作 (ActionChains)

<https://www.796t.com/article.php?id=94198>

3. 行為鍊

https://python-selenium-zh.readthedocs.io/zh_CN/latest/7.2%20%E8%A1%8C%E4%B8%BA%E9%93%BE/

4. ActionChains In Selenium

<https://medium.com/@kavidhanda/actionchains-in-selenium-cde43dee0111>

5. Selenium 4.1.0 documentation - selenium.webdriver.common.action_chains

https://www.selenium.dev/selenium/docs/api/py/webdriver/selenium.webdriver.common.action_chains.html

匯入工具

...

匯入工具

備註：每次執行以下任一範例前，都要執行一次"匯入工具"的 cell

...

操作 browser 的 API

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
```

期待元素出現要透過什麼方式指定，通常與 EC、WebDriverWait 一起使用

```
from selenium.webdriver.common.by import By
```

加入行為鍊 ActionChain (在 WebDriver 中模擬滑鼠移動、點擊、拖曳、按右鍵
出現選單，以及鍵盤輸入文字、按下鍵盤上的按鈕等)

```
from selenium.webdriver.common.action_chains import ActionChains
```

加入鍵盤功能 (例如 Ctrl、Alt 等)

```
from selenium.webdriver.common.keys import Keys
```

強制等待 (執行期間休息一下)

```

from time import sleep

# 啟動瀏覽器工具的選項
my_options = webdriver.ChromeOptions()
# my_options.add_argument("--headless") #不開啟實體瀏覽器背景執行
my_options.add_argument("--start-maximized") #最大化視窗
my_options.add_argument("--incognito") #開啟無痕模式
my_options.add_argument("--disable-popup-blocking") #禁用彈出攔截
my_options.add_argument("--disable-notifications") #取消通知

# 使用 Chrome 的 WebDriver
driver = webdriver.Chrome(
    options = my_options,
    service = Service(ChromeDriverManager().install())
)

```

範例 1：對特定座標連續點擊

...

範例 1：對特定座標連續點擊

參考連結：

[1] Crazy Game - Planet Clicker 2

<https://www.crazygames.com/game/planet-clicker-2>

[2] 擴充功能 coordinates

<https://chrome.google.com/webstore/detail/coordinates/bpf1bjmbfccblbhlcmlgkajdpoiepmkd>

...

前往頁面

```
driver.get('https://www.crazygames.com/game/planet-clicker-2')
```

建立行為鍊

```
ac = ActionChains(driver)
```

```
# 移到指定座標 (從 0,0 開始)
ac.move_by_offset(928, 369)

# 點擊一下
ac.click()

# 暫停一下
ac.pause(15)

# 若先前已移動，則進行相對位移 (從 928, 369 開始)
ac.move_by_offset(-256, -60) #(672, 309)

# 點擊一下
for i in range(2000):
    ac.click()

# 執行
ac.perform()

# 睡一下
sleep(5)

# 關閉 web driver
driver.quit()
```

範例 2: 拖曳網頁元素 (使用 drag_and_drop)

...

範例 2: 拖曳網頁元素

參考連結:

[1] Mootools Drag and Drop example

<http://sahitest.com/demo/dragDropMooTools.htm>

...

前往頁面

```
driver.get('http://sahitest.com/demo/dragDropMooTools.htm')

# 取得被拖曳的來源元素
dragger = driver.find_element(By.CSS_SELECTOR, "div#dragger")

# 目標元素（放置的區域，共 4 個）
items = driver.find_elements(By.CSS_SELECTOR, "div.item")

# 建立行為鍊
ac = ActionChains(driver)

# 暫停一下
ac.pause(2)

# 放置第一個
ac.drag_and_drop(dragger, items[0])

# 暫停一下
ac.pause(2)

# 放置第二個
ac.click_and_hold(dragger).release(items[1])

# 暫停一下
ac.pause(2)

# 放置第三個
ac.click_and_hold(dragger).move_to_element(items[2]).release()

# 暫停一下
ac.pause(2)

# 放置第四個
ac.click_and_hold(dragger).move_by_offset(400, 150).release()

# 執行
ac.perform()
```

```
# 睡一下  
sleep(5)  
  
# 關閉 web driver  
driver.quit()
```

範例 3: 組合熱鍵 (全選 + 複製 + 貼上)

...

範例 3: 組合熱鍵 (全選 + 複製 + 貼上)

參考連結:

[1] Display Text Input Fields

https://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_input_type_text

...

前往頁面

```
driver.get('https://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_input_type_text')
```

取得網頁上的 iframe

```
iframe = driver.find_element(By.CSS_SELECTOR, "iframe#iframeResult")
```

切換到 iframe 當中

```
driver.switch_to.frame(iframe)
```

回到主框頁

```
# driver.switch_to.default_content()
```

取得第一個文字欄位

```
inputText01 = driver.find_element(By.CSS_SELECTOR, "input#fname")
```

取得第二個文字欄位

```
inputText02 = driver.find_element(By.CSS_SELECTOR, "input#lname")
```

```
# 建立行為鍊
ac = ActionChains(driver)

# 在第一個文字欄位當中輸入萬用字元
ac.key_down(Keys.SHIFT,
inputText01).send_keys('12345').key_up(Keys.SHIFT).send_keys('67890')

# 暫停一下
ac.pause(2)

# 全選與複製第一個文字欄位當中的所有字元
ac.key_down(Keys.CONTROL,
inputText01).send_keys('ac').key_up(Keys.CONTROL)

# 暫停一下
ac.pause(2)

# 在第二個文字欄位當中貼上文字
ac.key_down(Keys.CONTROL,
inputText02).send_keys('v').key_up(Keys.CONTROL)

# 執行
ac.perform()

# 睡一下
sleep(5)

# 關閉 web driver
driver.quit()
```

範例 4: 移動 Slider (by a handle)

...

範例 4: 移動 Slider (by a handle)

參考連結：

[1] jQuery UI - Slider

<https://jqueryui.com/slider/>

...

前往頁面

```
driver.get('https://jqueryui.com/slider/')
```

取得網頁上的 iframe

```
iframe = driver.find_element(By.CSS_SELECTOR, "iframe.demo-frame")
```

切換到 iframe 當中

```
driver.switch_to.frame(iframe)
```

拿到把手元素

```
span = driver.find_element(By.CSS_SELECTOR, 'span.ui-slider-handle.ui-corner-all.ui-state-default')
```

建立行為鍊

```
ac = ActionChains(driver)
```

暫停

```
ac.pause(2)
```

點住不放，並向右移動，移到指定的座標（註：目前在 iframe 當中，使用的座標是 iframe 內部座標）

```
ac.click_and_hold(span).move_by_offset(577, 16).release()
```

執行

```
ac.perform()
```

睡一下

```
sleep(5)
```

關閉 web driver

```
driver.quit()
```

Module 18. PyAutoGUI

控制電腦的鍵盤、滑鼠，對圖形化介面進行操作的工具

安裝套件

一般安裝

```
pip install pyautogui
```

影像處理

```
pip install opencv-python
```

```
pip install opencv-contrib-python
```

Optional: 有影像處理的需求，則需要安裝 [OpenCV](#)

使用方法

- 滑鼠功能

方法	說明
pyautogui.position()	取得滑鼠游標座標
pyautogui.size()	目前螢幕大小
pyautogui.onScreen(x, y)	判斷游標座標是否在螢幕範圍內
pyautogui.PAUSE = 2.5	設定 PyAutoGUI 每一個動作間隔 2.5 秒

方法	說明
pyautogui.moveTo(x, y, duration=num_seconds)	移動滑鼠游標到 x,y
pyautogui.moveRel(xOffset, yOffset, duration=num_seconds)	在目前游標座標偏移相對位置
pyautogui.dragTo(x, y, duration=num_seconds, button='left')	按住滑鼠左鍵 · 拖曳滑鼠到 x,y
pyautogui.dragRel(xOffset, yOffset, duration=num_seconds, button='left')	按住滑鼠左鍵 · 在目前游標座標拖曳滑鼠到相對位置
pyautogui.click(x=moveToX, y=moveToY, clicks=num_of_clicks, interval=secs_between_clicks, button='left')	點擊滑鼠
pyautogui.rightClick(x=moveToX, y=moveToY)	在指定座標按下滑鼠右鍵
pyautogui.middleClick(x=moveToX, y=moveToY)	在指定座標按下滑鼠中間鍵
pyautogui.doubleClick(x=moveToX, y=moveToY)	在指定座標連點兩下滑鼠左鍵
pyautogui.tripleClick(x=moveToX, y=moveToY)	在指定座標連點三下滑鼠左鍵
pyautogui.scroll(amount_to_scroll, x=moveToX, y=moveToY)	滑鼠游標移到 x,y · 而後滾動 amount_to_scroll 次; 正值向上 · 負值向下
pyautogui.mouseDown(x=moveToX, y=moveToY, button='left')	在指定座標按下滑鼠按鍵
pyautogui.mouseUp(x=moveToX, y=moveToY, button='left')	在指定座標放開滑鼠按鍵

- 鍵盤功能

方法	說明
<code>pyautogui.typewrite('Hello world!\n', interval=secs_between_keys)</code>	在遊標所在位置輸入文字
<code>pyautogui.typewrite(['a', 'b', 'c', 'left', 'backspace', 'enter', 'f1'], interval=secs_between_keys)</code>	在游標所在位置分別輸入鍵盤按鍵
<code>pyautogui.write('Hello world!', interval=0.25)</code>	每 0.25 秒輸入一次文字 · 但要先 手機將焦點移到文字欄位當中
<code>pyautogui.hotkey('ctrl', 'c')</code>	組合鍵 <code>ctrl + c</code> (複製)
<code>pyautogui.hotkey('ctrl', 'v')</code>	組合鍵 <code>ctrl + v</code> (貼上)
<code>pyautogui.hotkey('ctrl', 'alt', 'delete')</code>	組合鍵 <code>ctrl + alt + delete</code>
<code>pyautogui.keyDown(key_name)</code>	按下指定鍵
<code>pyautogui.keyUp(key_name)</code>	放開指定鍵
<code>pyautogui.press('enter')</code>	按下 <code>enter</code>

- 訊息框功能

方法	說明
<code>pyautogui.alert('This displays some text with an OK button.')</code>	警示訊息

方法	說明
pyautogui.confirm('This displays text and has an OK and Cancel button.')	確認訊息 (會回傳值)
pyautogui.prompt('This lets the user type in a string and press OK.')	提示訊息 (會回傳值)

- 截圖功能

方法	說明
pyautogui.screenshot()	回傳 Pillow or PIL Image 物件
pyautogui.screenshot('foo.png')	回傳 Pillow or PIL Image 物件，並儲存檔案
pyautogui.locateOnScreen('looksLikeThis.png')	若畫面上有跟照片相同的圖案，回傳它的 Pillow or PIL Image 物件，並顯示它的 x,y
pyautogui.locateAllOnScreen('looksLikeThis.png')	若畫面上有跟照片相同的圖案，回傳它的 Pillow or PIL Image 物件，此方法會回傳全部圖片的 left, top, width, height
pyautogui.locateCenterOnScreen('looksLikeThis.png')	若畫面上有跟照片相同的圖案，則取得該圖案在螢幕當中的中心點座標
pyautogui.center(pyautogui.locateOnScreen('looksLikeThis.png'))	取得 Pillow or PIL Image 物件的中心點座標

鍵盤表 (key_name 名稱)

```
[ '\t', '\n', '\r', ' ', '!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', ':', ';', '<', '=' , '>', '?', '@'
```

```
', '[', '\\', ']', '^', '_', '`', 'a', 'b', 'c', 'd', 'e', 'f',
', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r',
', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '{', '|', '}', '~',
', 'accept', 'add', 'alt', 'altright', 'apps', 'backspace',
'browserback', 'browserfavorites', 'browserforward',
'browserhome', 'browserrefresh', 'browsersearch', 'browserstop',
'capslock', 'clear', 'convert', 'ctrl', 'ctrlleft',
't', 'ctrlright', 'decimal', 'del', 'delete', 'divide', 'down',
', 'end', 'enter', 'esc', 'escape', 'execute', 'f1', 'f10',
'f11', 'f12', 'f13', 'f14', 'f15', 'f16', 'f17', 'f18', 'f19',
', 'f2', 'f20', 'f21', 'f22', 'f23', 'f24', 'f3', 'f4', 'f5',
', 'f6', 'f7', 'f8', 'f9', 'final', 'fn', 'hangul', 'hangul',
', 'hanja', 'help', 'home', 'insert', 'junja', 'kana', 'kanji',
'launchapp1', 'launchapp2', 'launchmail', 'launchmediaselect',
'left', 'modechange', 'multiply', 'nexttrack', 'nonconvert',
', 'num0', 'num1', 'num2', 'num3', 'num4', 'num5', 'num6',
', 'num7', 'num8', 'num9', 'numlock', 'pagedown', 'pageup',
', 'pause', 'pgdn', 'pgup', 'playpause', 'prevtrack', 'print',
't', 'printscreen', 'prntscrn', 'prtsc', 'prtscr', 'return',
', 'right', 'scrolllock', 'select', 'separator', 'shift', 'shiftleft',
', 'shiftright', 'sleep', 'space', 'stop', 'subtract',
', 'tab', 'up', 'volumedown', 'volumemute', 'volumeup', 'win',
', 'winleft', 'winright', 'yen', 'command', 'option', 'optionleft',
', 'optionright']
```

參考網頁

[1] Welcome to PyAutoGUI's documentation!

<https://pyautogui.readthedocs.io/en/latest/index.html>

[2] PyAutoGUI - Installation

<https://pyautogui.readthedocs.io/en/latest/install.html>

[3] Python : 用 PyAutoGUI 來操控滑鼠及鍵盤

<https://yang10001.yia.app/wp/2021/03/06/python-%E7%94%A8-pyautoqui-%E4%BE%86%E6%93%8D%E6%8E%A7%E6%BB%91%E9%BC%A0%E5%8F%8A%E9%8D%B5%E7%9B%A4/>

[4] Python 自動化工具 – PyAutoGUI 釋放你的雙手

<http://13.231.129.69/2020/08/13/python-%E8%87%AA%E5%8B%95%E5%8C%96%E5%B7%A5%E5%85%B7-pyautogui-%E9%87%8B%E6%94%BE%E4%BD%A0%E7%9A%84%E9%9B%99%E6%89%8B/>

[5] PyAutoGUI : 使用 Python 控制電腦

<https://yanwei-liu.medium.com/pyautogui-%E4%BD%BF%E7%94%A8python%E6%93%8D%E6%8E%A7%E9%9B%BB%E8%85%A6-662cc3b18b80>

[5] 【PYTHON】pyautogui 如何增加每秒的點選次數 ?

<https://www.796t.com/post/ZTVueDA=.html>

[6] pyautogui 文檔 (五) : 截圖及定位功能

<https://www.itread01.com/content/1556528332.html>

參考網頁

[1] GitHub: keyboard

<https://github.com/boppreh/keyboard>

[2] GitHub: mouse

<https://github.com/boppreh/mouse>

安裝套件

...

安裝套件

...

`!pip install pyautogui opencv-python opencv-contrib-python keyboard`

匯入工具

```
...
匯入工具
...
import pyautogui
from time import sleep, time
import keyboard
from IPython.display import clear_output
```

協助了解滑鼠游標定位

```
...
請在 Terminal 當中執行
取得滑鼠座標和游標上的色碼，協助我們定位
補充：必須在 Terminal 當中執行，請見 18.py
...
pyautogui.displayMousePosition()
```

取得目前滑鼠座標

```
...
取得目前滑鼠座標
...
while True:
    #自動清除此格的文字輸出
    clear_output(wait=True)

    #取得滑鼠游標的座標，並加以輸出
    x, y = pyautogui.position()
    print(f"x={x}, y={y}")

    # 睡一下，以免吃光記憶體
    sleep(0.1)
```

記錄滑鼠軌跡

```
...
記錄滑鼠軌跡
...

# 記錄軌跡用的變數
listMove = []

# 記錄軌跡
def record():
    # 監聽事件
    while True:
        if keyboard.is_pressed('ctrl'): # 按下 Ctrl 錄製軌跡，放開就不進行錄製
            x, y = pyautogui.position()
            listMove.append(f'{x},{y}')
            sleep(0.1)
        elif keyboard.is_pressed('alt'): # 按下 alt，結束程式
            break

    # 寫入檔案
    with open("move.txt", "w", encoding="utf-8") as file:
        file.write( '\n'.join(listMove) )

...
主程式
...

if __name__ == "__main__":
    record()
```

讀取滑鼠軌跡，並模擬滑鼠游標移動

```
...
```

讀取滑鼠軌跡，並模擬滑鼠遊標移動

```
...
# 模擬移動
def move():
    # 讀取軌跡檔
    with open("move.txt", "r", encoding="utf-8") as file:
        # 逐行讀取
        for line in file:
            # 切割文字，找出 x,y
            x, y = line.strip().split(",")
            # 移動（設定移動時間）
            pyautogui.moveTo( int(x), int(y), duration=0.1)

            # 提前結束
            if keyboard.is_pressed('alt'):
                break
...
主程式
...
if __name__ == "__main__":
    move()
```

確認畫面上的圖片物件是否存在

```
...
每秒判斷桌面上的圖片(座標)是否存在
補充：請先開小算盤(計算機)
...
while True:
    # 睡一下
    sleep(0.1)
#自動清除此格的文字輸出
```

```
clear_output(wait=True)

# 取得 Box 物件
btn_2_location = pyautogui.locateOnScreen('images/2.png', confidence=0.9)

# 輸出
print(btn_2_location)
```

自動使用小算盤(計算機)

...

自動使用計算機

補充：此時計算機應該明顯出現在桌面上，不然會有判斷錯誤的問題

...

```
# 設定每一個動作，都暫停若干秒
```

```
pyautogui.PAUSE = 0.5
```

```
# 比對所有圖片，取得顯示在桌面的圖片物件 (pillow / PIL Image)
```

```
btn_0_location = pyautogui.locateOnScreen('images/0.png',
confidence=0.9)
```

```
btn_1_location = pyautogui.locateOnScreen('images/1.png',
confidence=0.9)
```

```
btn_2_location = pyautogui.locateOnScreen('images/2.png',
confidence=0.9)
```

```
btn_3_location = pyautogui.locateOnScreen('images/3.png',
confidence=0.9)
```

```
btn_4_location = pyautogui.locateOnScreen('images/4.png',
confidence=0.9)
```

```
btn_5_location = pyautogui.locateOnScreen('images/5.png',
confidence=0.9)
```

```
# 取得每一個圖片的中心點
```

```
btn_0_point = pyautogui.center(btn_0_location)
```

```
btn_1_point = pyautogui.center(btn_1_location)
```

```
btn_2_point = pyautogui.center(btn_2_location)
btn_3_point = pyautogui.center(btn_3_location)
btn_4_point = pyautogui.center(btn_4_location)
btn_5_point = pyautogui.center(btn_5_location)

# 按下 5201314
pyautogui.click(btn_5_point.x, btn_5_point.y)
pyautogui.click(btn_2_point.x, btn_2_point.y)
pyautogui.click(btn_0_point.x, btn_0_point.y)
pyautogui.click(btn_1_point.x, btn_1_point.y)
pyautogui.click(btn_3_point.x, btn_3_point.y)
pyautogui.click(btn_1_point.x, btn_1_point.y)
pyautogui.click(btn_4_point.x, btn_4_point.y)
```

打擊遊戲

...

打擊遊戲

範例網址: <https://taiko.bui.pm/>

請搭配 18.py，取得參考用的座標、色碼

注意：每台電腦的顏色配置可能不同，要自行找到合適的色碼

Red, Green, Blue

橘色: (243, 71, 40)

藍色: (101, 189, 187)

連打: (243, 181, 0)

打擊區座標

(x=519, y=455)

...

設定每一個動作，都暫停若干秒

pyautogui.PAUSE = 0.1

```

# 打擊點
listPoint = [
    [519, 455]
]

# 點擊
def click():
    for list_ in listPoint:
        # 每次點擊都會重新取得指定打擊點的 RGB 值
        coord = pyautogui.pixel( list_[0], list_[1] )

        # coord[0] 代表 Red , coord[1] 代表 Green , coord[2] 代表 Blue
        if coord[0] == 243 and coord[1] == 71: # 若 R = 243, G = 71
            pyautogui.press('j') # 右邊內側橘色
        elif coord[0] == 101 and coord[1] == 189: # 若 R = 101, G = 189
            pyautogui.press('k') # 右邊外側藍色
        elif coord[0] == 243 and coord[1] == 181: # 若 R = 243, G = 181
            pyautogui.press('d') # 左邊外側藍色
            pyautogui.press('f') # 左邊內側橘色
            pyautogui.press('j') # 右邊內側橘色
            pyautogui.press('k') # 右邊外側藍色

    ...

主程式區域
...

if __name__ == "__main__":
    while True:
        # 當我們按下 ctrl 鍵時，自動點擊，按下 alt 則結束迴圈
        if keyboard.is_pressed('ctrl'):
            click()
        elif keyboard.is_pressed('alt'):
            break

```