目次

[Module 1. 開發環境：Anaconda、Jupyter 及爬蟲專案實務 1](#_Toc152459259)

[Module 2. 正規表達式（Regular Expression）說明 16](#_Toc152459260)

[Module 3. HTML基礎與HTTP方法 23](#_Toc152459261)

[Module 4. CSS Selector 36](#_Toc152459262)

[Module 5. Chrome Developer Tool 43](#_Toc152459263)

[Module 6. 套件requests 53](#_Toc152459264)

[Module 7. 套件Beautiful Soup 4 57](#_Toc152459265)

[Module 8. cookie 用於 requests 60](#_Toc152459266)

[Module 9. 案例: PTT\_NBA\_看板主頁與內頁 61](#_Toc152459267)

[Module 10. 套件Selenium（一） 65](#_Toc152459268)

[Module 11. 套件Selenium（二） 68](#_Toc152459269)

[Module 12. 套件Selenium（三） 74](#_Toc152459270)

[Module 13. ActionChains 80](#_Toc152459271)

[Module 14. PyAutoGUI 90](#_Toc152459272)

* 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Anaconda Distribution 的連結



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



圖: 下載後，快速點兩下進行安裝

有安全性警告，可按下執行或同意



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」

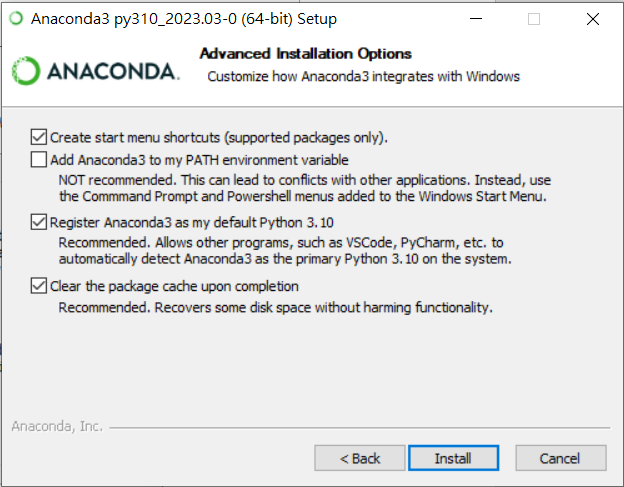


圖: 依需求選擇 (最後一個建議打勾，省空間)，按下「Install」，進行安裝



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



圖：輸入「conda update conda」，更新 conda 本身



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



圖：輸入「conda upgrade --all」，更新當前所有套件



圖：按下「y」後，再按鍵盤「Enter」



圖：更新套件中



圖：套件更新完成



圖：輸入「python」，進入 python 執行環境



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



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 |
| conda create --name web\_scraping python=3.10 notebook ipykernel |

|  |
| --- |
| 進入 conda 環境 |
| conda activate web\_scraping |

|  |
| --- |
| 新增 Kernel |
| python -m ipykernel install --user --name ws --display-name "Python3@ws" |

註:

1. python -m 指的是直接使用模組 (module) 的預設功能。
2. --user 代表將kernel 安裝在個人使用者目錄 (或是個人的家目錄) 當中，而非預設的系統環境。
3. --name ws 與 conda env 的 web\_scraping 沒有直接關係，僅是kernel在系統當中的別名。
4. --display-name "Python3@ws" 是新增Jupyter Notebook時的選項。

|  |
| --- |
| 檢視 Jupyter Notebook kernel |
| jupyter kernelspec list |

|  |
| --- |
| 刪除 Kernel |
| jupyter kernelspec uninstall ws |

注意：

* Jupyter Notebook 建立的 kernel，跟 conda 的 env 概念是不一樣的。
  + Kernel會啟動網頁版本的線上編輯器，並且使用目前 conda env 的Python 版本；而 conda 的 env 單純是環境和套件的管理器。
  + 例如在 python=3.10 的 conda env 裡面安裝kernel，此時該kernel用的python版本就會是 3.10。
  + 無論在任何conda env下，只要啟動 Jupyter Notebook，就會以當下目錄作為主目錄/工作路徑。

|  |
| --- |
| 執行Jupyter Notebook (在虛擬環境web\_scraping下) |
| conda activate web\_scraping  jupyter notebook |

|  |
| --- |
| 刪除Conda 環境 |
| conda remove -n web\_scraping --all |

下載 Chrome Web Driver

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

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

1. 請確認目前你電腦裡面的 chrome 瀏覽器版本

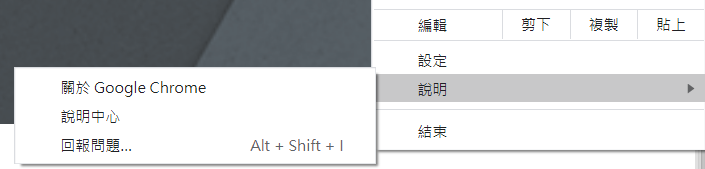


圖: 按下瀏覽器右上方的「」→說明→關於 Google Chrome



圖: 請與下載的ChromeDriver版本一致

1. 下載ChromeDriver檔案，並放到專案資料夾當中

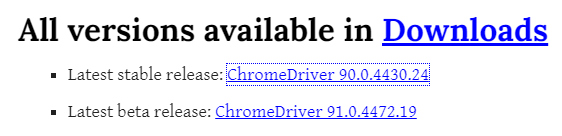


圖: 下載合適的Chrome版本

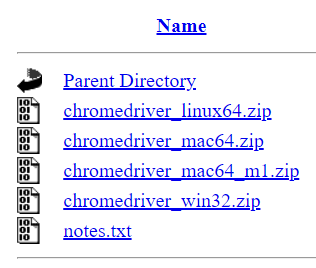


圖: Windows選擇 win32；MacOS選擇對應的晶片版本

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。

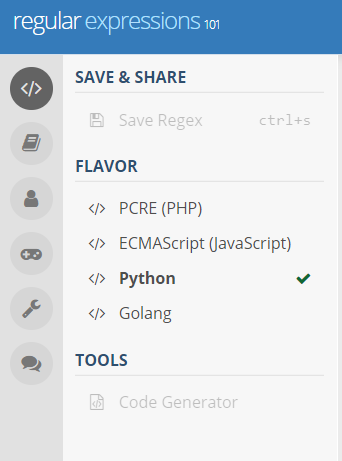
|  |
| --- |
| 說明 |
| 本課程需要安裝的套件指令  **pip install beautifulsoup4 lxml requests selenium webdriver-manager fake-useragent tika pyautogui opencv-python opencv-contrib-python keyboard wget** |

**爬蟲專案開發實務分享**

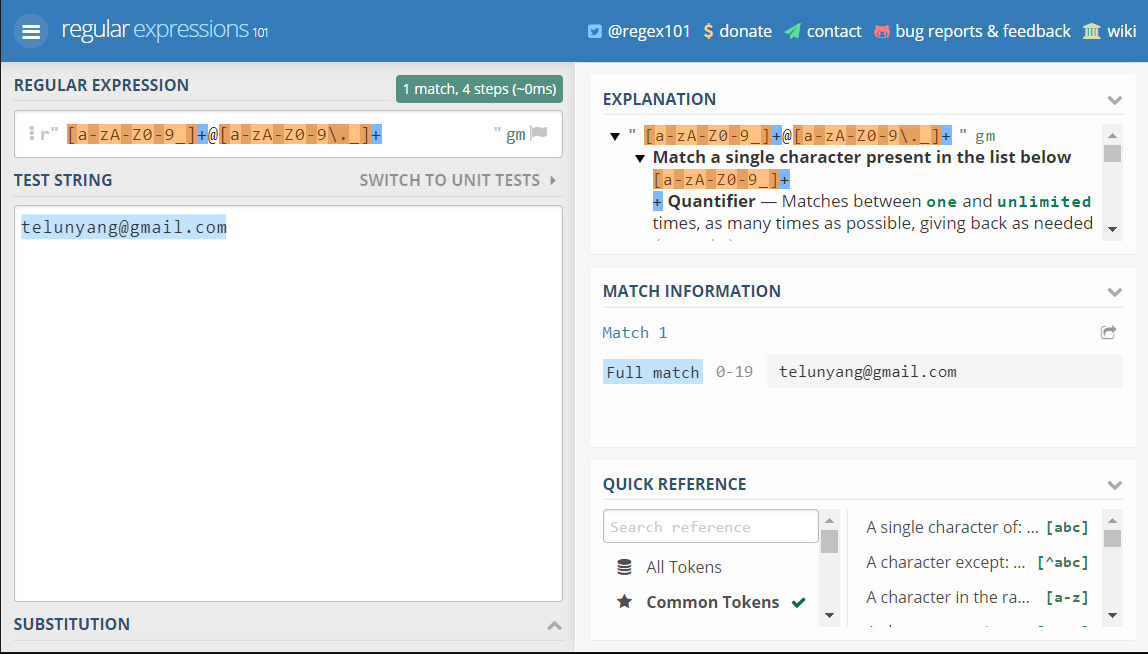
1. 個人 YouTube 頻道（不定時更新）  
   <https://www.youtube.com/@darreninfo-boatman>
2. 「觀察」是爬蟲工作者的重要技能，HTML & CSS selector 的概念要非常熟練；有些人習慣使用 XPATH，由於課程時間安排的關係，不會提到。
3. 被對方的伺服器擋住，無法繼續爬取，相換 IP，可以透過
   1. **Amazon Web Service 的 EC2（虛擬主機）**，透過 Running、Stopped 等過程，提供自動換新的IP給我們。
   2. **連接VPN**，例如 Surfshark，透過切換其它網路環境（俗稱翻牆）來進行；
   3. 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給我們，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；
   4. 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. 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



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 | addbeeeaaccbaa |
| 0個或1個a | a? | addbeeeaaccbaa |
| 0個或更多的a | a\* | addbeeeaaccbaa |
| 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 |
| 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 |

|  |
| --- |
| 常用方法 |
| # 匯入 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]([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 |
|  |
| # 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]) |

|  |
| --- |
| 參考資料 |
| 1. Python3 正则表达式  <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> |

# Module 3. HTML基礎與HTTP方法

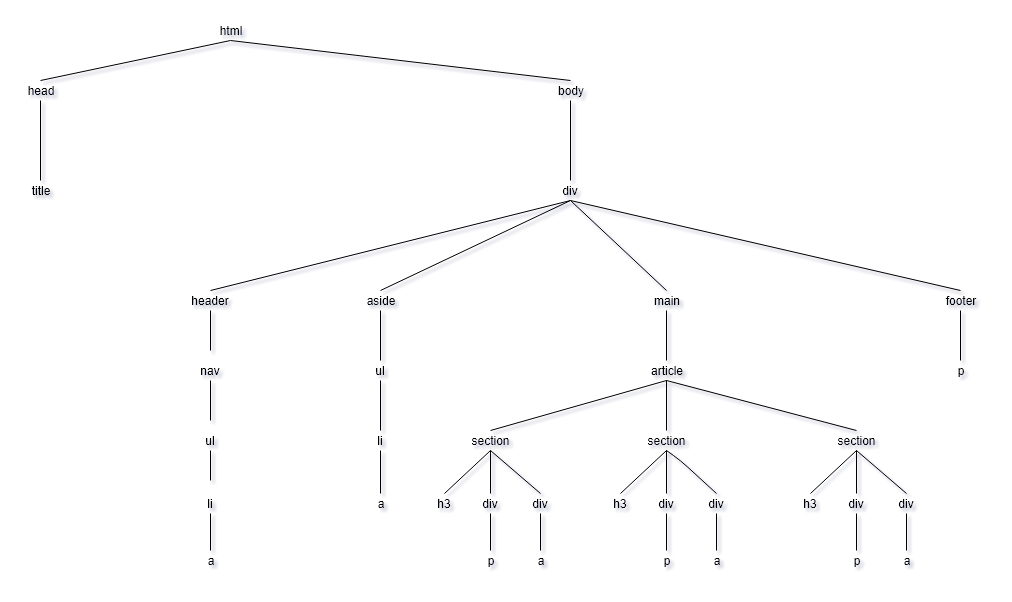
HTML 簡介

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>  <li class="nav-list">  <a class="center link-custom">連結3</a>  </li>  </ul>  </nav>  </header>  <!-- aside 在這裡是左側的選單列表 -->  <aside class="menu">  <ul class="menu-body">  <li class="menu-list">  <a class="center">側欄連結1</a>  </li>  <li class="menu-list">  <a class="center">側欄連結2</a>  </li>  <li class="menu-list">  <a class="center">側欄連結3</a>  </li>  </ul>  </aside>  <!-- main 是主要顯示內容的區域 -->  <main class="content-container">  <article class="article-paragraph">  <section class="episode">  <h3 class="title">HTML parser 開發不求人 - 第三節</h3>  <div class="content">  <p>動態網頁的元素走訪，都需要透過不斷地實作、練習，同時...</p>  </div>  <div class="content-more">  <a class="more-link">More</a>  </div>  </section>  <section class="episode">  <h3 class="title">HTML parser 開發不求人 - 第二節</h3>  <div class="content">  <p>動態網頁的元素走訪，都需要透過不斷地實作、練習，同時...</p>  </div>  <div class="content-more">  <a class="more-link">More</a>  </div>  </section>  <section class="episode">  <h3 class="title">HTML parser 開發不求人 - 第一節</h3>  <div class="content">  <p>動態網頁的元素走訪，都需要透過不斷地實作、練習，同時...</p>  </div>  <div class="content-more">  <a class="more-link">More</a>  </div>  </section>  </article>  </main>  <!-- footer 在這裡是簡單提供網站的基礎資訊 -->  <footer class="footer">  <p>歡迎光臨！本站由 Darren Yang 本人親自虛構…</p>  </footer>  </div>  </body>  </html> |

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：

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

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。 |
| <ul> | <ul><li>清單項目1</li><li>清單項目2</li></ul> | 定義尚未排序的列表。 |
| <li> | <ul><li>清單項目1</li><li>清單項目2</li></ul> | 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</a> | 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https://example.com/resources | 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組資源。 | 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https://example.com/resources/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。

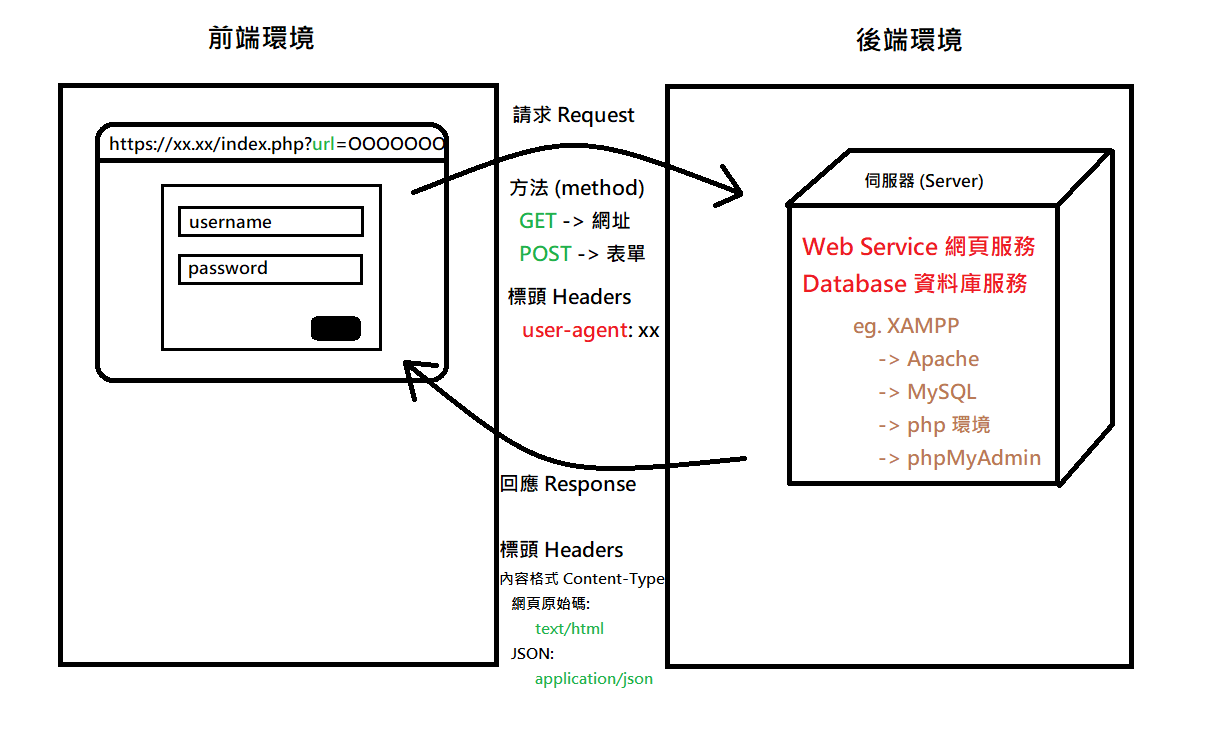


圖: 請求(request)與回應(response)

GET方法

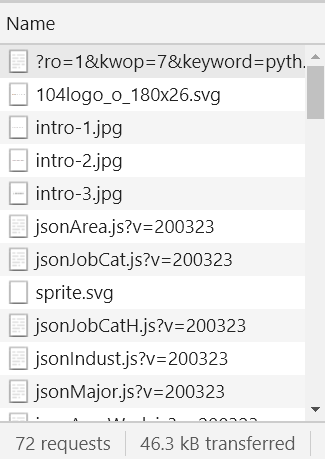
表單資料將以字串方式附加在網址（URI）的後面傳送，在網址尾端，會以「?」符號，開啟跟著表單中的資料，每個欄位間的值，以「&」連接起來。

一般來說，GET 參數（Query String）的格式如下：

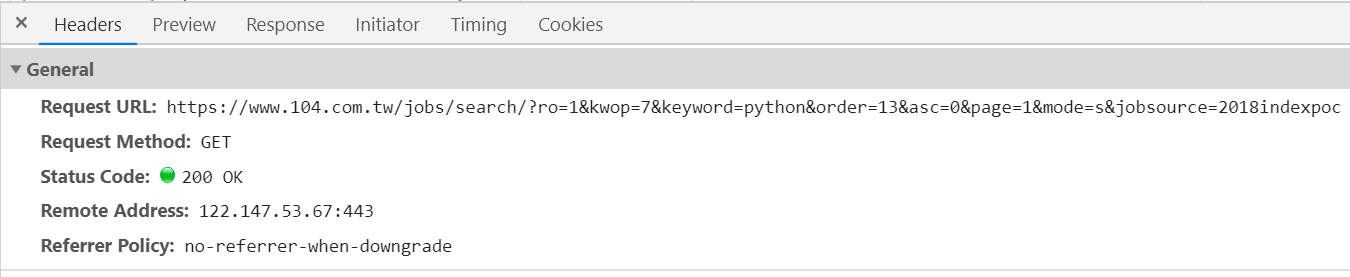
https://www.104.com.tw/jobs/search/?ro=1&kwop=7&keyword=python&order=13&asc=0&page=1&mode=s&jobsource=2018indexpoc

| key（鍵） | value（值） |
| --- | --- |
| ro | 1 |
| kwop | 7 |
| keyword | python |
| order | 13 |
| asc | 0 |
| page | 1 |
| mode | s |
| jobsource | 2018indexpoc |

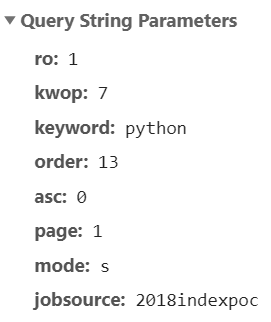
我們將前面的網址放到chrome瀏覽器的網址列中，讀取結束後，按下 Ctrl + Shift + i，再按下 Network，然後 Ctrl + R，重新讀取網址，再從左側的Name欄位裡，選擇最上面（通常最先被讀取的那個）的項目，再選Headers，會看到以下的資訊：



圖：選擇第一個項目



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



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 4. 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。 |
| **.class1.class2** | .name1.name2 | 選擇 class 屬性中同時有 name1 和 name2 的所有元素。 |
| **.class1 .class2** | .name1 .name2 | 選擇作為類名 name1 元素後代的所有類名 name2 元素。 |
| **#id** | #firstname | 選擇 id="firstname" 的元素。 |
| [**\***](https://www.w3school.com.cn/cssref/selector_all.asp) | \* | 選擇所有元素。 |
| [**element**](https://www.w3school.com.cn/cssref/selector_element.asp) | p | 選擇所有 <p> 元素。 |
| [**element.class**](https://www.w3school.com.cn/cssref/selector_element_class.asp) | p.intro | 選擇 class="intro" 的所有 <p> 元素。 |
| [**element,element**](https://www.w3school.com.cn/cssref/selector_element_comma.asp) | div, p | 選擇所有 <div> 元素和所有 <p> 元素。 |
| [**element element**](https://www.w3school.com.cn/cssref/selector_element_element.asp) | div p | 選擇 <div> 元素內的所有 <p> 元素。 |
| [**element > element**](https://www.w3school.com.cn/cssref/selector_element_gt.asp) | div > p | 選擇父元素是 <div> 的所有 <p> 元素。 |
| [**[attribute]**](https://www.w3school.com.cn/cssref/selector_attribute.asp) | [target] | 選擇帶有 target 屬性的所有元素。 |
| [**[attribute=value]**](https://www.w3school.com.cn/cssref/selector_attribute_value.asp) | [target=\_blank] | 選擇帶有 target="\_blank" 屬性的所有元素。 |
| [**[attribute~=value]**](https://www.w3school.com.cn/cssref/selector_attribute_value_contain.asp) | [title~=flower] | 選擇 title 屬性包含單詞 "flower" 的所有元素。 |
| [**[attribute^=value]**](https://www.w3school.com.cn/cssref/selector_attr_begin.asp) | a[href^="https"] | 選擇其 href 屬性值以 "https" 開頭的每個 <a> 元素。 |
| [**[attribute$=value]**](https://www.w3school.com.cn/cssref/selector_attr_end.asp) | a[href$=".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

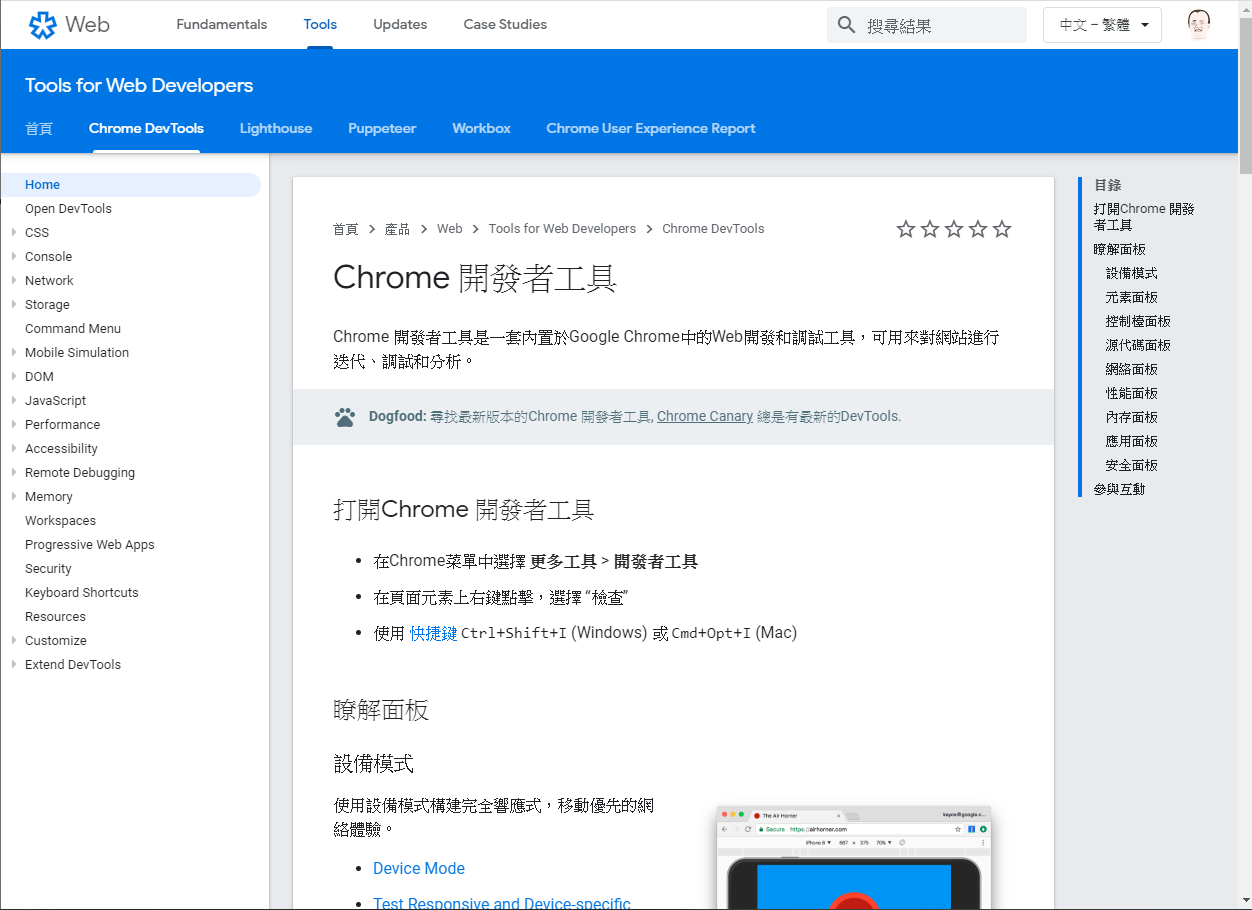
<https://www.w3school.com.cn/cssref/css_selectors.asp>

# Module 5. 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

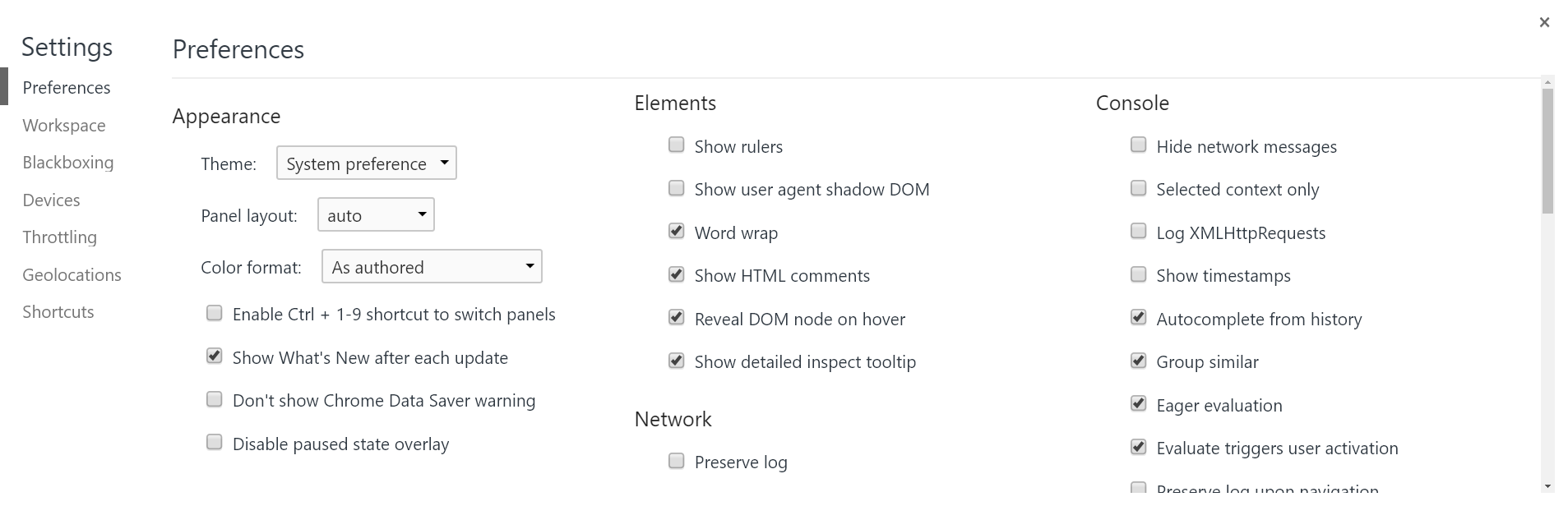
檢查HTML元素

* 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

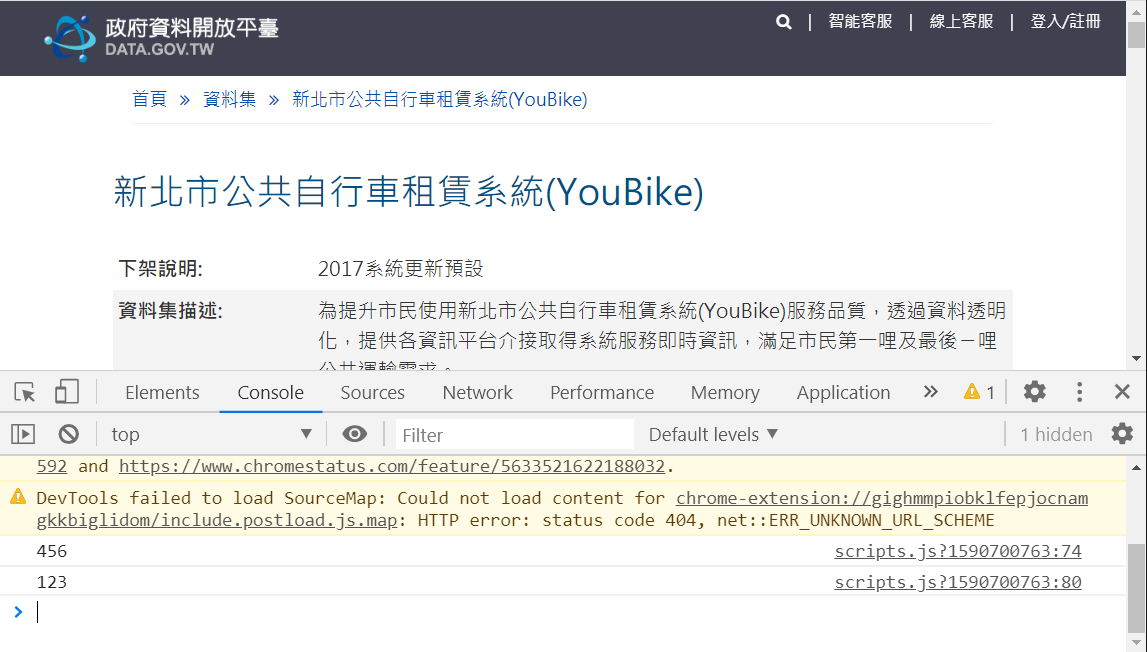
* Ctrl + O 尋找 HTML 當中的檔名
* Ctrl + R 或 F5 刷新頁面
* Ctrl + F5 清除快取後，刷新頁面(重新從伺服器端請求下載 HTML)
* Ctrl + L 清除 Console
* Shift + Enter 在 Console 中斷行(或多行)

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

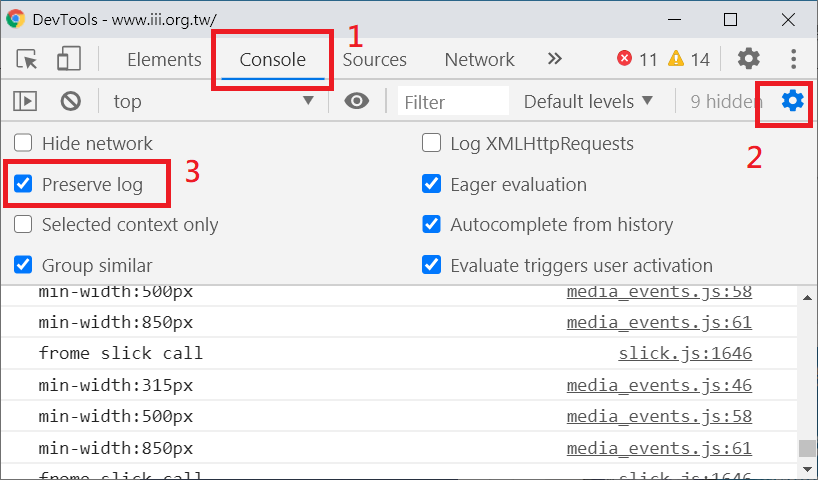


圖：觀看檔案的 Headers 內容

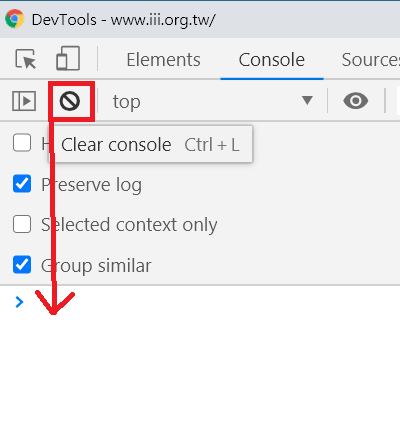
Request Headers (請求標頭，參考[維基百科](https://zh.wikipedia.org/wiki/HTTP%E5%A4%B4%E5%AD%97%E6%AE%B5))

| **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 | Cookie: \_ga=GA1.3.1322956465.1572335045;locale=zh\_TW; \_gid=GA1.3.1110994946.1584940974; \_gat\_gtag\_UA\_141775379\_1=1 |
| User-Agent | 瀏覽器的[瀏覽器身分標識字串](https://zh.wikipedia.org/wiki/%E7%94%A8%E6%88%B7%E4%BB%A3%E7%90%86" \o "用戶代理) | User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:12.0) Gecko/20100101 Firefox/21.0 |

**常用操作流程介紹 （Preserve Log / Clear / …）**



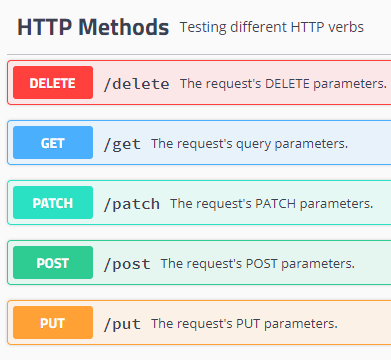
圖：勾選Preserve log，縱然頁面刷新，過去的log依然會保存起來



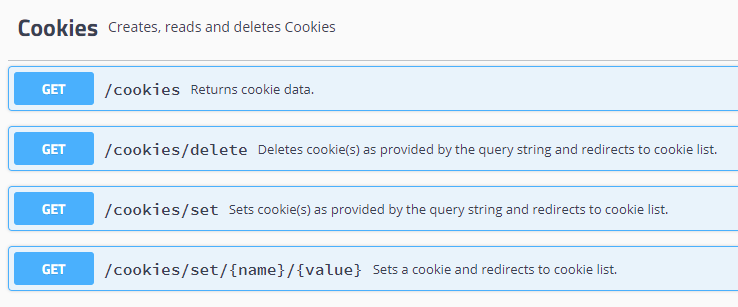
圖：按下 Clear Console，或是Ctrl+L，即可清除log

# Module 6. 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://greenlife.epa.gov.tw/categories/restaurant  # 預設連結 https://greenliving.epa.gov.tw/newPublic/APIs/Restaurant4  # 分頁連結 https://greenliving.epa.gov.tw/newPublic/APIs/Restaurant4/1  # 請求網址  url = 'https://greenliving.epa.gov.tw/newPublic/APIs/Restaurant4/1'  res = requests.get(url)  # 將 json 轉成物件  obj = json.loads(res.text) # 或使用 obj = res.json()  # 輸出對應節點的文字  print(obj['Result'])  print(obj['RowsCount'])  print(obj['PageIndex'])  print("=" \* 50)  # 輸出部分節點的文字  for o in obj['Detail']:      if o['Latitude'] == 0.0 or o['Longitude'] == 0.0: continue      print(f"Id: {o['Id']}")      print(f"Name: {o['Name']}")      print(f"Address: {o['Address']}")      print(f"Latitude: {o['Latitude']}")      print(f"Longitude: {o['Longitude']}")      print("=" \* 50) |

# Module 7. 套件Beautiful Soup 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 8. 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 9. 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="noreferrer noopener nofollow"]')) > 0:          soup\_.select\_one('span.f2 + a[href][rel="noreferrer noopener 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 = 6504  # 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="noreferrer noopener nofollow"]')) > 0:          soup\_.select\_one('span.f2 + a[href][rel="noreferrer noopener 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()      # 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 10. 套件Selenium（一）

**解析Selenium、WebDriver與Browser連動關係**

* Selenium 是一種 web automatic testing 工具 (套件)，藉此操作 WebDriver，協助開發者操作網頁表單資料、點選按鈕或連結、取得網頁內容並進行檢驗。
* 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ChromeDriver, EdgeDriver, FirefoxDriver等。
* Browser 經由 WebDriver 啟動，讓 Selenium 進行操作，完成網頁自動化的工作。



圖：還操控瀏覽器的位置與大小



圖：drirver.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 11. 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



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  from selenium.webdriver.chrome.service import Service  from selenium import webdriver  # 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 exptected\_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(3)  # 刷新頁面  driver.refresh()  # 睡個幾秒  sleep(3)  # 刷新頁面  driver.refresh() |

|  |
| --- |
| 關閉瀏覽器 |
| # 關閉瀏覽器  driver.quit() |

# Module 12. 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，否則一直等到時間截止，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不存在於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  '''  # 開啟用於自動控制的瀏覽器 (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  '''  # 開啟用於自動控制的瀏覽器  driver = webdriver.Chrome()  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 ( 透過 .get\_attribute('屬性') 來取得屬性的值 )      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  '''  # 開啟用於自動控制的瀏覽器  driver = webdriver.Chrome()  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() |

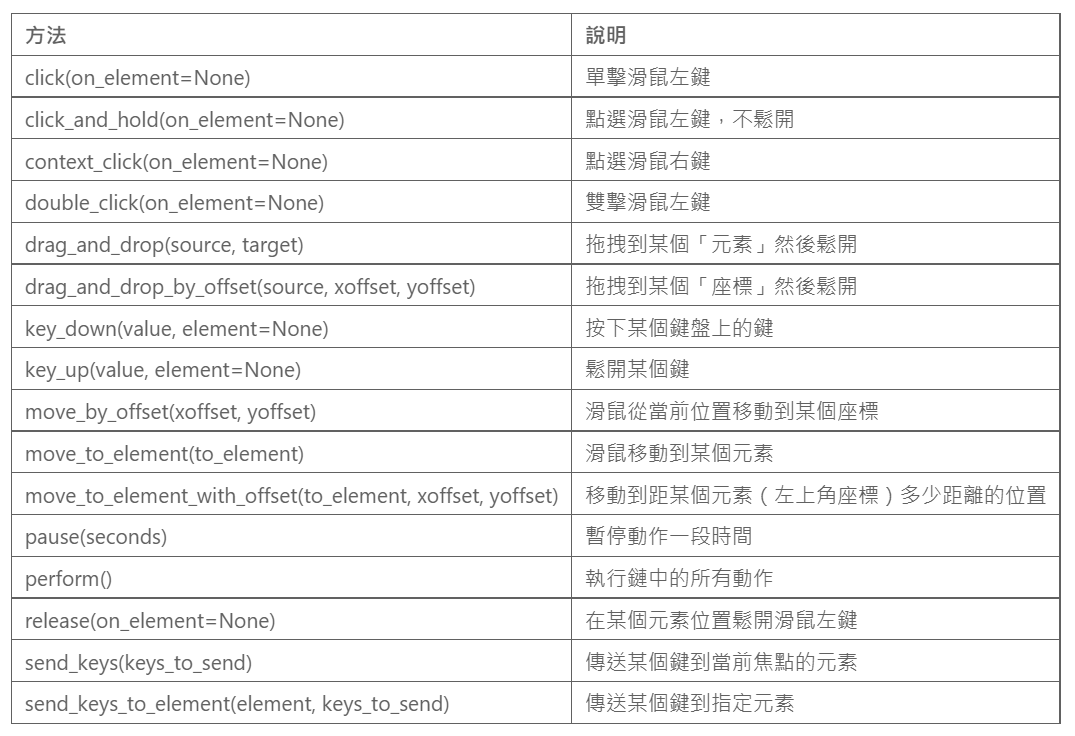
# Module 13. ActionChains

**匯入工具**

加入行為鍊 ActionChains (在 WebDriver 中模擬滑鼠移動、點繫、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**



**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>   1. python selenium滑鼠鍵盤操作（ActionChains）   <https://www.796t.com/article.php?id=94198>   1. 行為鏈   <https://python-selenium-zh.readthedocs.io/zh_CN/latest/7.2%20%E8%A1%8C%E4%B8%BA%E9%93%BE/>   1. ActionChains In Selenium   <https://medium.com/@kavidhanda/actionchains-in-selenium-cde43dee0111>   1. 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  ) |

|  |
| --- |
| 範例 1: 對特定座標連續點擊 |
| '''  範例 1: 對特定座標連續點擊  補充:  先在 console 面版中，輸入下列程式碼，  會在滑鼠一次移動後，顯示座標。  document.onmousemove = function(e){      var x = e.pageX;      var y = e.pageY;      e.target.title = "X is "+x+" and Y is "+y;  };  參考連結:  [1] Pop Cat  https://popcat.click/  [2] Is there a way to tell Chrome web debugger to show the current mouse position in page coordinates?  https://stackoverflow.com/questions/12888584/is-there-a-way-to-tell-chrome-web-debugger-to-show-the-current-mouse-position-in  '''  # 前往頁面  driver.get('https://popcat.click/')  # 建立行為鍊  ac = ActionChains(driver)  # 移到指定座標  '''  從 0,0 開始，若先前已移動，則進行相對位移，數值要用負號  例如 ac.move\_by\_offset(-50, -80)  '''  ac.move\_by\_offset(870, 350)  # 暫停一下  ac.pause(3)  # 點擊一下  for i in range(2000):      ac.click()  # 執行  ac.perform()  # 睡一下  sleep(3)  # 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(1)  # 放置第一個  ac.drag\_and\_drop(dragger, items[0])  # 暫停一下  ac.pause(1)  # 放置第二個  ac.click\_and\_hold(dragger).release(items[1])  # 暫停一下  ac.pause(1)  # 放置第三個  ac.click\_and\_hold(dragger).move\_to\_element(items[2]).release()  # 暫停一下  ac.pause(1)  # 放置第四個  ac.click\_and\_hold(dragger).move\_by\_offset(400, 150).release()  # 執行  ac.perform()  # 睡一下  sleep(3)  # 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(1)  # 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(1)  # 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(3)  # 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(1)  # 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)  # 註: 點住移動，只能在 iframe 範圍內，超出的話，span 會放不掉  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 14. 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](https://opencv.org/releases/)

**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 |
| 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') | 點繫滑鼠 |
| 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** |
| --- | --- |
| pyautogui.typewrite('Hello world!\n', interval=secs\_between\_keys) | 在遊標所在位置輸入文字 |
| pyautogui.typewrite(['a', 'b', 'c', 'left', 'backspace', 'enter', 'f1'], interval=secs\_between\_keys) | 在游標所在位置分別輸入鍵盤按鍵 |
| pyautogui.write('Hello world!', interval=0.25) | 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 |
| pyautogui.hotkey('ctrl', 'c') | 組合鍵 ctrl + c (複製) |
| pyautogui.hotkey('ctrl', 'v') | 組合鍵 ctrl + v (貼上) |
| pyautogui.hotkey('ctrl', 'alt', 'delete') | 組合鍵 ctrl + alt + delete |
| pyautogui.keyDown(key\_name) | 按下指定鍵 |
| pyautogui.keyUp(key\_name) | 放開指定鍵 |
| pyautogui.press('enter') | 按下 enter |

* 訊息框功能

| **方法** | **說明** |
| --- | --- |
| pyautogui.alert('This displays some text with an OK button.') | 警示訊息 |
| 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，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', 'altleft', 'altright', 'apps', 'backspace','browserback', 'browserfavorites', 'browserforward', 'browserhome','browserrefresh', 'browsersearch', 'browserstop', 'capslock', 'clear','convert', 'ctrl', 'ctrlleft', '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', 'hanguel', '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', '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-pyautogui-%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(0.01)          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=0.01)                # 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 - 魔術鋼琴鏄  https://html5.gamedistribution.com/rvvASMiM/44fbc5e5c6e54ac9985d1c81f1fb8121/index.html  '''  # 設定每一個動作，都暫停若干秒  pyautogui.PAUSE = 0.1  # 打擊點  listPoint = [      [620, 600],      [720, 600],      [820, 600],      [920, 600]  ]  # 點擊  def click():      for \_ in listPoint:          # 取得指定打擊點的 RGB 值          coord = pyautogui.pixel( \_[0], \_[1] )          # coord[0] 代表 Red，coord[1] 代表 Green，coord[2] 代表 Blue          if coord[0] == 0 and coord[1] == 0:              pyautogui.mouseDown(\_[0], \_[1])              pyautogui.mouseUp(\_[0], \_[1])    #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 |