Python 網路爬蟲
Web Scraping

Python 網路爬蟲

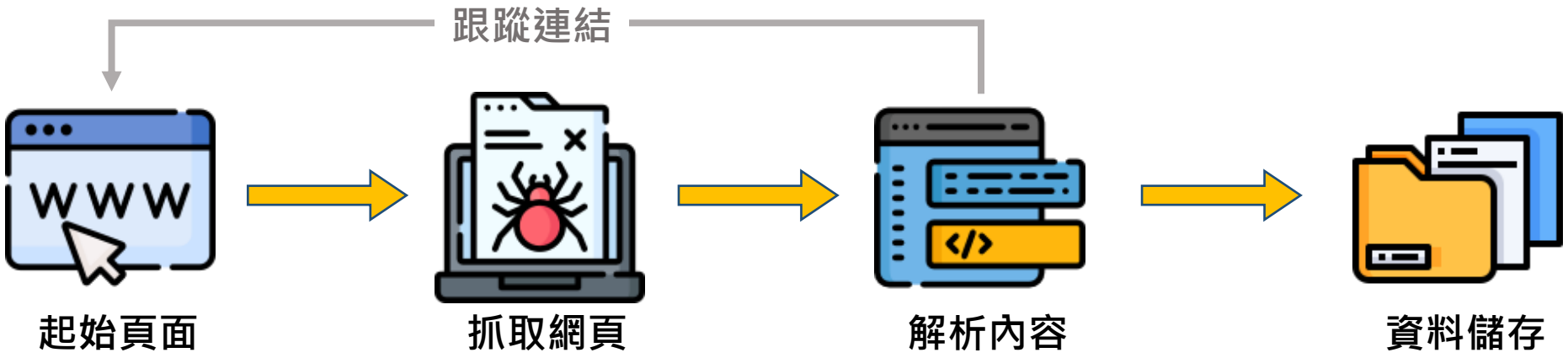
- 網路爬蟲是一種自動瀏覽網頁的程式或腳本。主要目的是從互聯網上下載網頁資料，並提取出有用的信息，或是自動化執行網頁互動 (例如: 高鐵訂票、演唱會訂票) 的任務。
- Python 因為其豐富的套件和框架，使得撰寫網路爬蟲相對容易，因此成為了許多開發者和資料科學家進行網頁數據提取的首選語言。
- 目前 Python 被廣泛使用的爬蟲相關套件工具主要有 :

  Requests、Beautiful Soup、 lxml 、Selenium、Scrapy

Python 網路爬蟲

- 網路爬蟲的工作流程大致可以分為以下幾個步驟：

  1. **起始頁面**：選擇一個或多個起始網址作為爬蟲的入口。
  2. **抓取網頁**：爬蟲訪問這些網址，並從伺服器上下載網頁內容。
  3. **解析內容**：解析網頁內容，通常是HTML或XML格式，提取出有價值的資料，或是進一步的網址連結。
  4. **跟蹤連結**：根據上一步解析內容得到的連結，爬蟲繼續訪問這些頁面，重複抓取和解析的過程。
  5. **儲存資料**：將提取出的資料保存到文件、資料庫或其他儲存系統中供後續使用。



Requests

- **Requests** 是一個使用 Python 語言編寫的非常流行的 HTTP 庫。可以輕鬆地發送 HTTP 請求，並且處理請求Headers、表單數據、文件上傳等複雜情況。
- 使用 **Requests**，可以通過幾行代碼完成 **web 請求**、**處理回應**和**處理異常**。
- **Requests** 的這些特性使得處理網路請求變得簡單而高效。對於進行網絡爬蟲是一個不可或缺的工具。

使用方式

```
response = requests.get(url)
```

URL 的組成結構

- 最簡單的 URL 由三部分組成
  - **安全協定 (Protocol) → 指定了瀏覽器應該使用哪種協議來訪問資源**
  - **域名 (Domain names) → 網站的地址，用來找到伺服器 (通常會對應到某個 IP)**
  - **路徑(Path) → 位於域名後的部分，用來指定網頁或資源的位置**

補充

網域分類類型：com (公司企業或是單位機構)、edu (學校網站或教育相關)、gov (政府機關)

URL 的組成結構

- URL 還可以有兩部分
  - **查詢參數(Protocol)** → 查詢參數位於 URL 的最後部分，以 ? 開頭，用來傳遞數據。例如：?year=2025 表示查詢參數 year 的值是 2025。多個參數可以用 & 分隔，例如：?year=2025&month=10。
  - **片段 (Fragment)** → 以 # 開頭，用來指定頁面中的特定部分。例如：#section1 會跳轉到頁面的特定部分。

一個完整的 URL 可能如下：

```
https://www.example.com/blog/article?year=2025&month=10#comments
```

requests 常用參數

參數	說明
url	請求的 URL
params	URL 查詢參數，通常是一個字典
data	POST 請求的數據，通常是一個字典
json	JSON 格式的請求數據
headers	HTTP 標頭，通常是一個字典
cookies	要發送的 Cookie
auth	認證信息，通常是元組 (用戶名, 密碼)
timeout	等待服務器回應的時間，單位是秒
allow_redirects	是否允許重定向 (默認為 True)
files	文件上傳，通常是字典

補充

requests 除了 get 還有 post、put、delete

response 常用方法和屬性

方法 / 屬性	說明
.status_code	HTTP 狀態碼
.headers	回應的 HTTP 標頭，字典格式
.text	回應內容的文本形式
.content	與 text 類似，但這個屬性提供的是原始的二進制數據。 這在**處理圖片**或其他**二進制內容**時特別有用
.json()	將回應內容解析為 JSON (必須為 JSON 格式)
.url	最終的 URL
.history	重定向紀錄，包含一個或多個 Response
.cookies	回應中的 Cookies，字典格式
.raise_for_status()	如果回應狀態碼是錯誤（4xx 或 5xx），則引發 HTTPError
.elapsed	請求和回應的時間差，`datetime.timedelta`

使用 API

- API (Application Programming Interface) 就像是一個「橋樑」，讓不同的系統之間可以互相溝通，以 Python 為例我們可以使用 Requests 對 API 發送請求。

  別人創建的 API 通常會提供：

  - **一個網址（endpoint）**
  - **可使用的方法（例如 GET, POST, PUT, DELETE）**
  - **傳入參數的格式（如 JSON）**
  - **回傳資料的格式（通常也是 JSON）只要按照文件規範發出請求，就能取得資料或執行動作。**

Beautiful Soup

- **Beautiful Soup** 提供了一些 Python風格的函式來解析 HTML 和 XML 文件。這個套件非常適合用於網頁**靜態資訊**的抓取，能夠幫助你從網頁中提取所需的資料。
- 使用**Beautiful Soup**，可以進行以下操作：
  - 訪問元素的標籤名、屬性和內容文本。
  - 根據標籤的類別、id或其他屬性找到或過濾特定的HTML元素。

使用方式

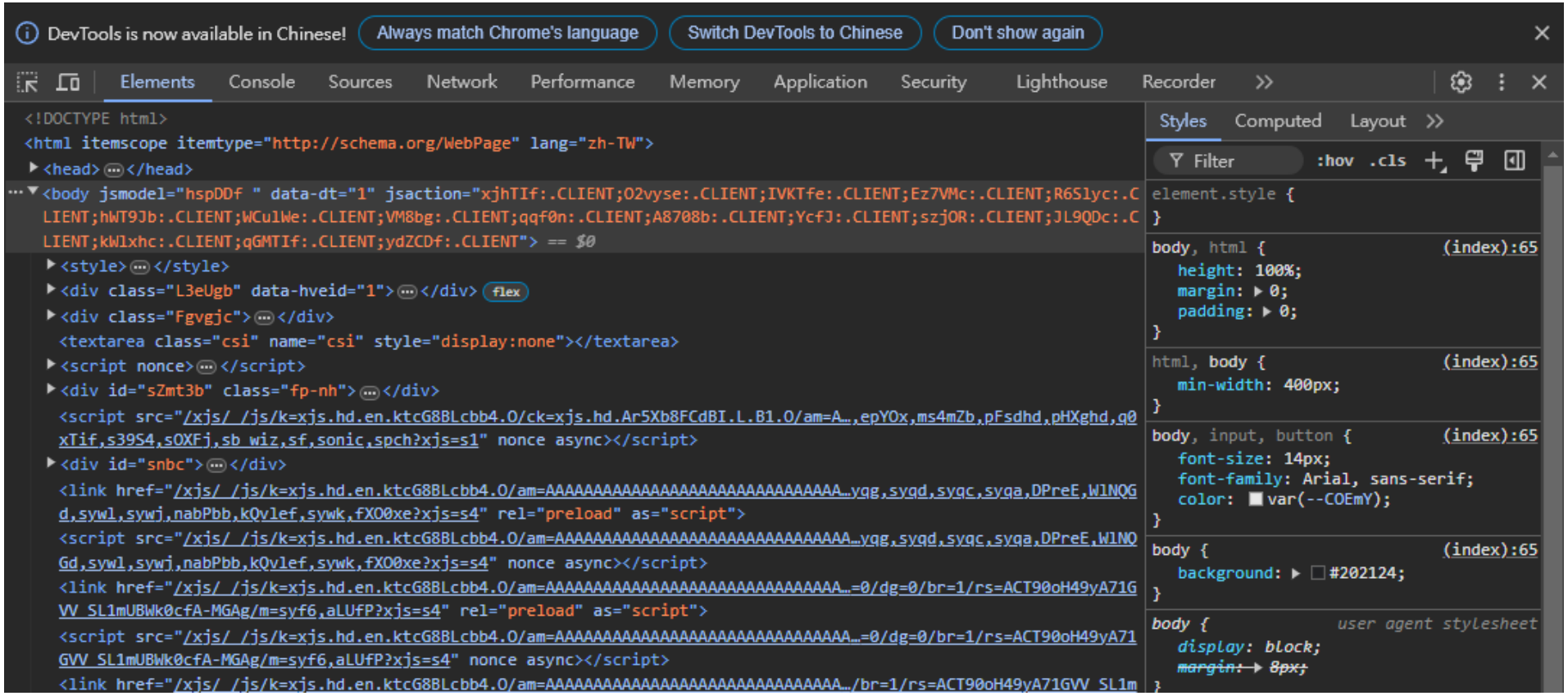
```
soup = BeautifulSoup(response.text, 'lxml')
```

Beautiful Soup 解析器

解析器	使用方法	特性
`html.parser`	`BeautifulSoup(markup, "html.parser")`	• Python的內建標準庫 • 文件容錯稍差
`lxml`	`BeautifulSoup(markup, "lxml")`	• 速度快 • 文件容錯能力強 • 需安裝
`lxml-xml`	`BeautifulSoup(markup, ["lxml-xml"])` `BeautifulSoup(markup, "xml")`	• 速度快 • 唯一支援XML的解析器 • 容錯低、不適合一般網站 • 需要安裝 lxml
`html5lib`	`BeautifulSoup(markup, "html5lib")`	• 依 HTML5 標準重建 DOM • 最好的容錯性 • 速度慢 • 占用記憶體較高

開發者工具 (DevTools)

- 瀏覽器的F12開發者工具提供了一系列功能強大的工具，用於檢查、調整和測試網頁。通過按下F12，可以用來查看網頁的**元素(elements)**



開發者工具 (DevTools)

1. **Elements（元素）**

   這個面板除了顯示當前網頁的 HTML 和 CSS 結構之外。也可以在這裡即時編輯 HTML 和 CSS 來查看修改後的效果，這對於調整樣式和排版非常有用。

2. **Console（控制台）**

   控制台可以用來輸入 JavaScript 代碼並即時執行，還會顯示錯誤和警告信息。這對於調試 JavaScript 代碼非常有幫助。

3. **Sources（來源）**

   這個面板可以查看和調試網頁上的所有 JavaScript 文件。設置斷點、逐步執行代碼，以及查看變量的當前值。

4. **Network（網路）**

   網路面板顯示網頁加載過程中所有的網路請求，包含請求的 URL、方法、狀態碼、加載時間等。這對於分析和優化網頁性能非常重要。

開發者工具 (DevTools)

5. Performance（性能）

性能面板可以用來記錄和分析網頁的性能問題。你可以捕捉一段時間的性能數據，並查看詳細的時間軸，分析哪些部分的加載或渲染過程耗時較長。

6. Memory（記憶體）

這個面板幫助你分析網頁的內存使用情況，檢查是否存在內存洩漏，並優化內存使用。

7. Application（應用程序）

應用程序面板顯示有關網頁的各種存儲信息，例如 Local Storage、Session Storage、Cookies、IndexedDB 等。這對於調試與存儲相關的問題非常有用。

8. Security（安全性）

安全性面板提供有關網頁安全性的資訊，包括 HTTPS 狀態、證書信息等。可以幫助確保網頁的安全性。

認識 HTML 的元素

```
<a href="http://example.com/jarvis" class="sister" id="link1">Jarvis</a>
```

- **<a> ： 標籤(tag)**
- **href、class、id ： 屬性(attribute)**
  - **href**：使用在放置網頁的連結，即當使用者點擊這個連結時，會被導向的網址。
  - **class**：標籤的 CSS 類別名稱，通常用於樣式設定。
  - **id**：標籤的唯一標識符，可以用於 JavaScript 或 CSS 中選擇該元素。
  - 實際顯示在網頁上的文字則是放在標籤內的 `Jarvis`

BS method – find_all() / find()

- **find_all()** → **element_list | []**

  功能：根據**標籤名(tag)**和**屬性(attribute)**來查找所有匹配的元素

- **find()** → **element obj | None**

  功能：根據**標籤名(tag)**和**屬性(attribute)**來查找第一個匹配的元素

Elements (tag / attribute)	用法
Pochita Denji	elements = soup.find_all("a")
Makima	elements = soup.find_all("a", class_="character")
Power	elements = soup.find_all("a", id="link2")
URL	element = soup.find('a', href="http://example.com/Denji")
Pochita	elements = soup.find_all(string="Pochita")

BS method – find_all() / find()

```
# 找到所有的 a 標籤的元素
elements = soup.find_all("a")

# 找到所有的 a 標籤，並且 class 為 character 的元素
elements = soup.find_all("a", class_="character")

# 找到所有的 a 標籤，並且 id 為 link2 的元素
elements = soup.find_all("a", id="link2")

# 找到第一個 a 標籤而且 href 為 http://example.com/Denji 的元素
element = soup.find("a", href="http://example.com/Denji")

# 找到內容為 Pochita 的元素
elements = soup.find(string="Pochita")
```

BS method – select() / select_one()

- **select() → element_list | []**

  功能：根據 CSS 選擇器來選取所有匹配的元素

- **select_one() → element obj | None**

  功能：根據 CSS 選擇器來選取第一個匹配的元素

Elements (tag / attribute)	用法
Pochita Denji	elements = soup.select("a")
Makima	elements = soup.select("a.character")
Power	element = soup.select_one("a#link2")
URL	elements = soup.select('a[href="http://example.com/Denji"]')

BS method – select() / select_one()

```
# 使用標籤選取元素
element = soup.select_one('h1')


# 用 id 選取元素
element = soup.select_one('#link3')


# 用 href 選取元素
element = soup.select_one('a[href="http://example.com/Denji"]')


# 用 class 選取元素
element = soup.select_one("a.character")
```

補充 - CSS 選擇器

```css
p {
    color: green;
}

.red {
    color: #f33;
}

.yellow-bg {
    background: #ffa;
}

.fancy {
    font-weight: bold;
    text-shadow: 4px 4px 3px #77f;
}

.item {
    background-color: #ffa;
}
```

```html
<p class="red">This paragraph has red text.</p>
<p class="red yellow-bg">
    This paragraph has red text and a yellow background.
</p>
<p class="red fancy">This has red text and "fancy" styling.</p>
<p>This is just a regular paragraph.</p>

<p class="item">This paragraph has a yellow background.</p>
```

This paragraph has red text.

This paragraph has red text and a yellow background.

This has red text and "fancy" styling.

This is just a regular paragraph.

This paragraph has a yellow background.

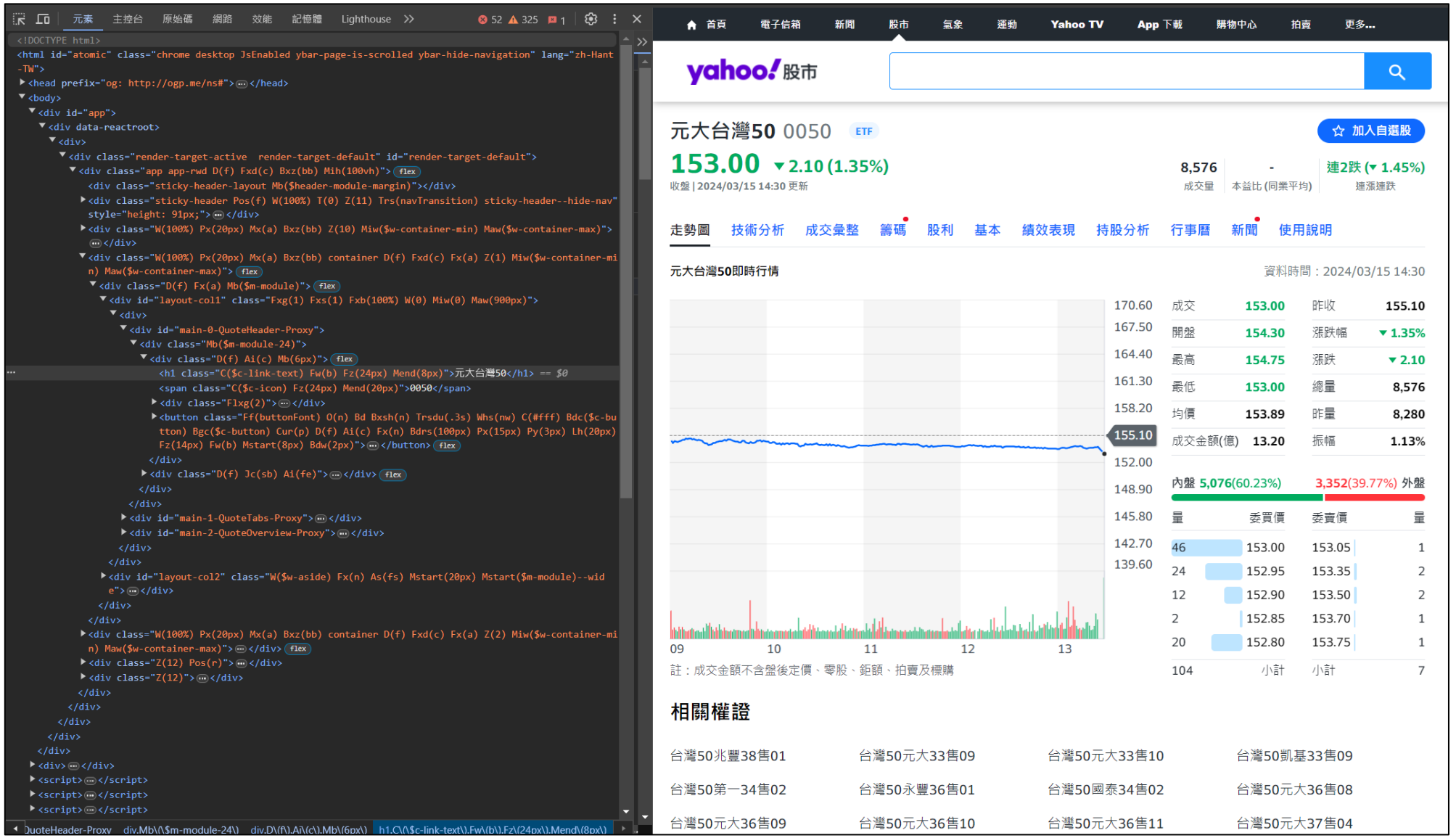
BS method – 模糊搜尋

```
# select() / select_one()


# 模糊搜尋 - 開頭
element = soup.select('[id^="lin"]')


# 模糊搜尋 - 結尾
element = soup.select('[class$="ter"]')


# 模糊搜尋 - 包含
element = soup.select('[id*="ink"]')
```

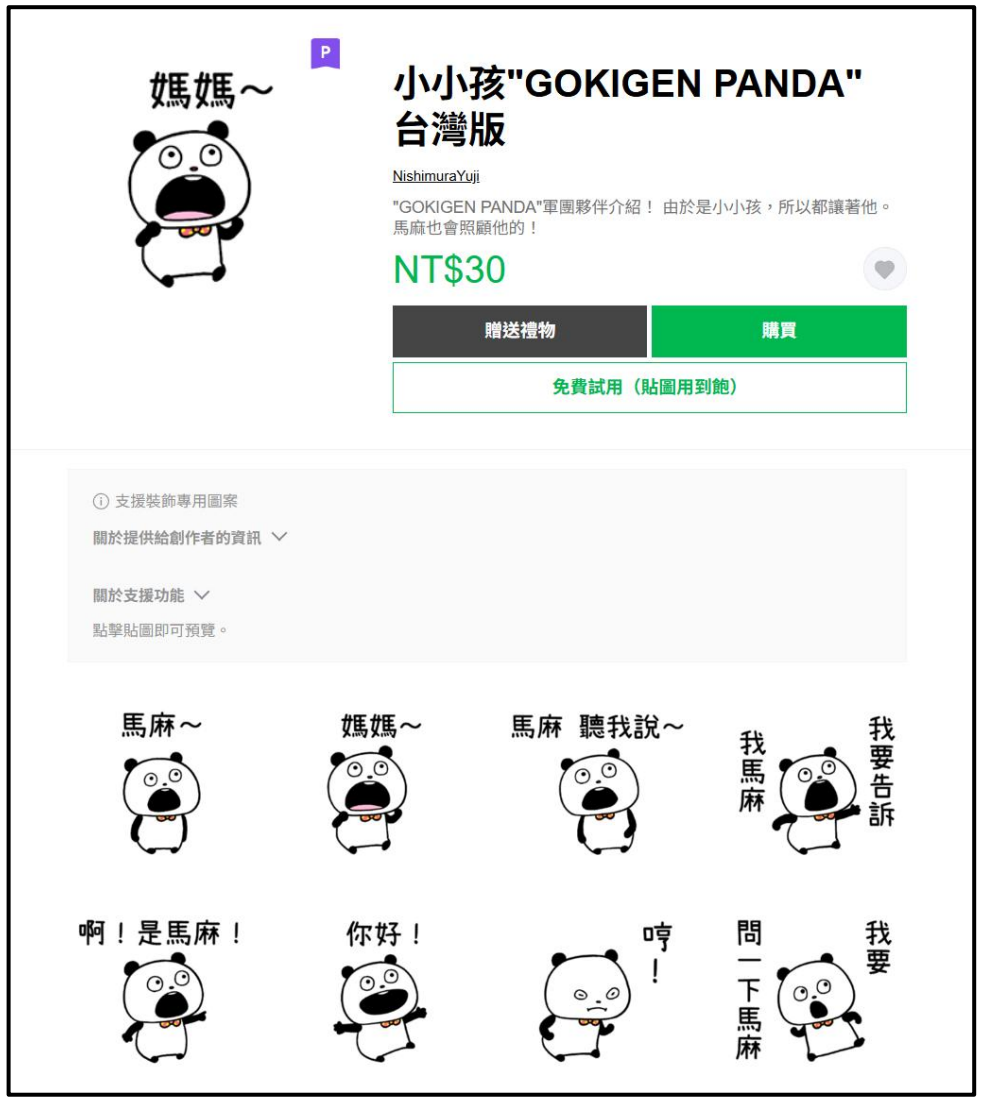
element method/attribute

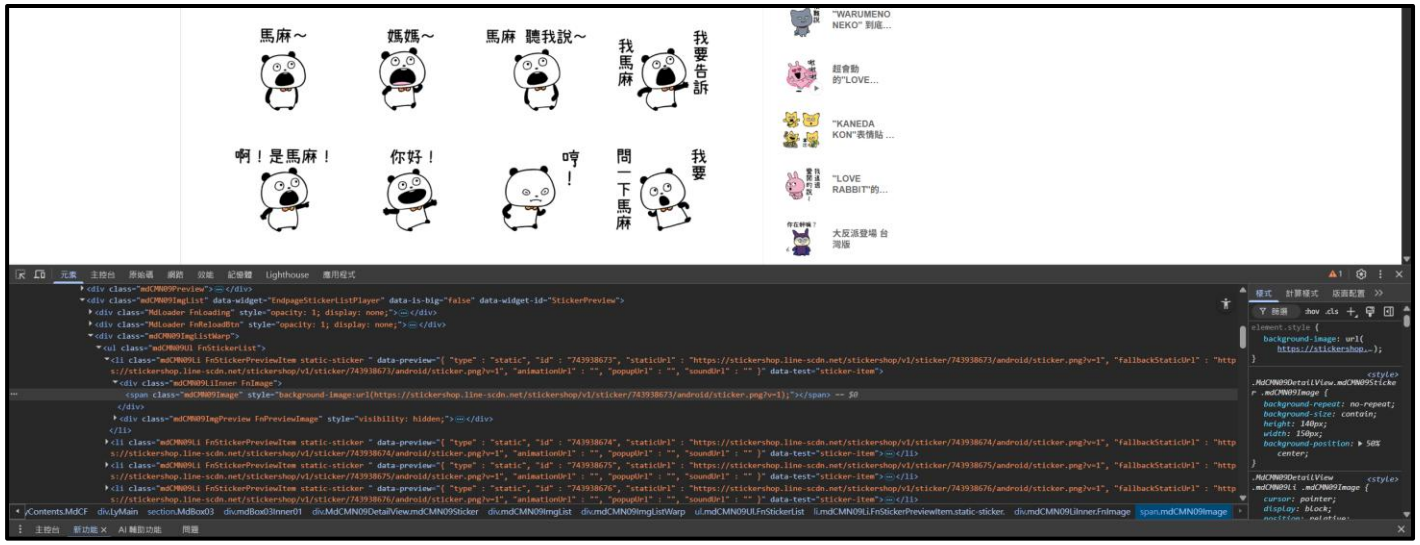
方法 / 屬性	說明
.find(name)	搜尋並返回第一個匹配指定標籤名稱的元素。如果找不到，返回 None。
.find_all(name)	搜尋並返回所有匹配指定標籤名稱的元素，返回值為一個列表。
.get_text()	獲取元素內的所有文字內容，返回值為一個字符串。
.attrs	獲取元素的所有屬性，返回值為一個字典。
.parent	獲取當前元素的父元素。
.children	以生成器形式獲取當前元素的所有子元素。
.descendants	以生成器形式獲取當前元素的所有子孫元素。
.previous_sibling	獲取當前元素的前一個兄弟元素。
.next_sibling	獲取當前元素的下一個兄弟元素。
.previous_element	獲取文檔中當前標籤的上一個元素。
.next_element	獲取文檔中當前標籤的下一個元素。

爬蟲範例練習 – 股市資訊爬取



爬蟲範例練習 – LINE圖片爬取





正則表達式 Regex expression

- 正則表達式（Regular Expression，簡稱 regex）是一種用於匹配符合指定格式字串的表示法。在處理文本數據時，提供了靈活且強大的方式來搜尋、編輯和操作。正則表達式在許多編程語言中都被廣泛應用，例如 Python、JavaScript、Golang、Perl .....等等。

Python 使用 regex 方式

```
import re

data_list = re.findall(pattern, str)

new_text = re.sub(pattern, replace_str, str)
```

正則表達式 Regex expression

函式	用途	回傳類型
re.match()	從字串開頭開始比對	Match 物件或 None
re.search()	搜尋整個字串，找到第一個符合的	Match 物件或 None
re.findall()	回傳所有符合的結果	list
re.sub()	取代符合的部分	str

```python
import re
text = "Name: John, Age: 30"
match = re.search(r'Name: (\w+), Age: (\d+)', text)

if match:
    print(match.group(0))  # 整個匹配字串: Name: John, Age: 30
    print(match.group(1))  # 第一組: John
    print(match.group(2))  # 第二組: 30
```

正則表達式 Regex expression

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]	abcdXYZ0123
一個字元，不在某個範圍內: a-z	[^a-z]	abcdXYZ0123
一個字元，在某個範圍內: a-z or A-Z	[a-zA-Z]	abcdXYZ0123
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^$

正則表達式 Regex expression

說明	正規表達式	範例
配對 a 或 b	a|b	addbeeeaaccbaa
0個或1個d	ad?	addbeeeaaccbaa
0個或更多d	ad*	addbeeeaaccbaa
1個或更多的a	a+	aaa, aabbbaaa
完整3個a	a{3}	a, aaa, aabaa, aaaaa
3個以上的 a	a{3,}	aa, aaa, aabaa, aaaaa
3個到6個之間的 a	a{3,6}	a, aaaaa, aa, aaaaaaaa
字串的開始	^ ex. ^Chainsaw	Chainsaw let devil saw
字串的結束	$ ex. saw$	Chainsaw let devil saw

正則表達式 Regex expression

說明	正規表達式	範例
位於文字邊界的字元	\b ex. \bis	My name is Jarvis Lu
非位於文字邊界的字元	\B ex. \Bis	My name is Jarvis Lu
避開特殊字元	\ ex. \.tw	xxx_tw@yahoo.com.tw
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, 4096, 2019, 3000, 1789, 1776, 1024, 8192
配對卻不在群組裡顯示	Denji (?:Pochita)	Denji Pochita, Denji Makima
正向環視 （這位置右邊要出現什麼）	Denji(?= Pochita)	Denji Pochita, Denji Makima
正向環視否定 （這位置右邊不能出現什麼）	Denji(?! Pochita)	Denji Pochita, Denji Makima
反向環視 （這位置左邊要出現什麼）	(?<=Denji)Pochita	Denji Pochita, Makima Pochita

BS method – 模糊搜尋

```
# find_all() / find()
# 使用正規表達式
import re
# 模糊搜尋 - 開頭
element = soup.find_all(id=re.compile(r'^link'))


# 模糊搜尋 - 結尾
element = soup.find_all(class_=re.compile(r'ter$'))


# 模糊搜尋 - 包含
element = soup.find_all(id=re.compile(r'link'))
```

Selenium

- selenium 是使用 Python 進行網路爬蟲時，必備的函式庫之一，透過 selenium 可以模擬出使用者在瀏覽器的所有操作行為 (點擊按鈕、輸入帳號密碼、捲動捲軸...等)。
- 除了爬蟲的應用，也常作為「自動化測試」使用的工具，在網站開發完成後，透過自動化的腳本測試所有功能是否正常。

當使用 requests 無法爬取網頁資訊時，通常都會改用 selenium 來處理 動態網頁 或 需要 JavaScript 生成的網頁內容。

Selenium – 套件安裝

- Selenium 安裝 → pip install selenium
- 查看特定套件的版本 → pip show <package_name>

Python 3.7 之後的版本(pip 版本必須在 19 以上)，執行 Selenium 安裝時，預設會安裝 Selenium 4 的最新版本

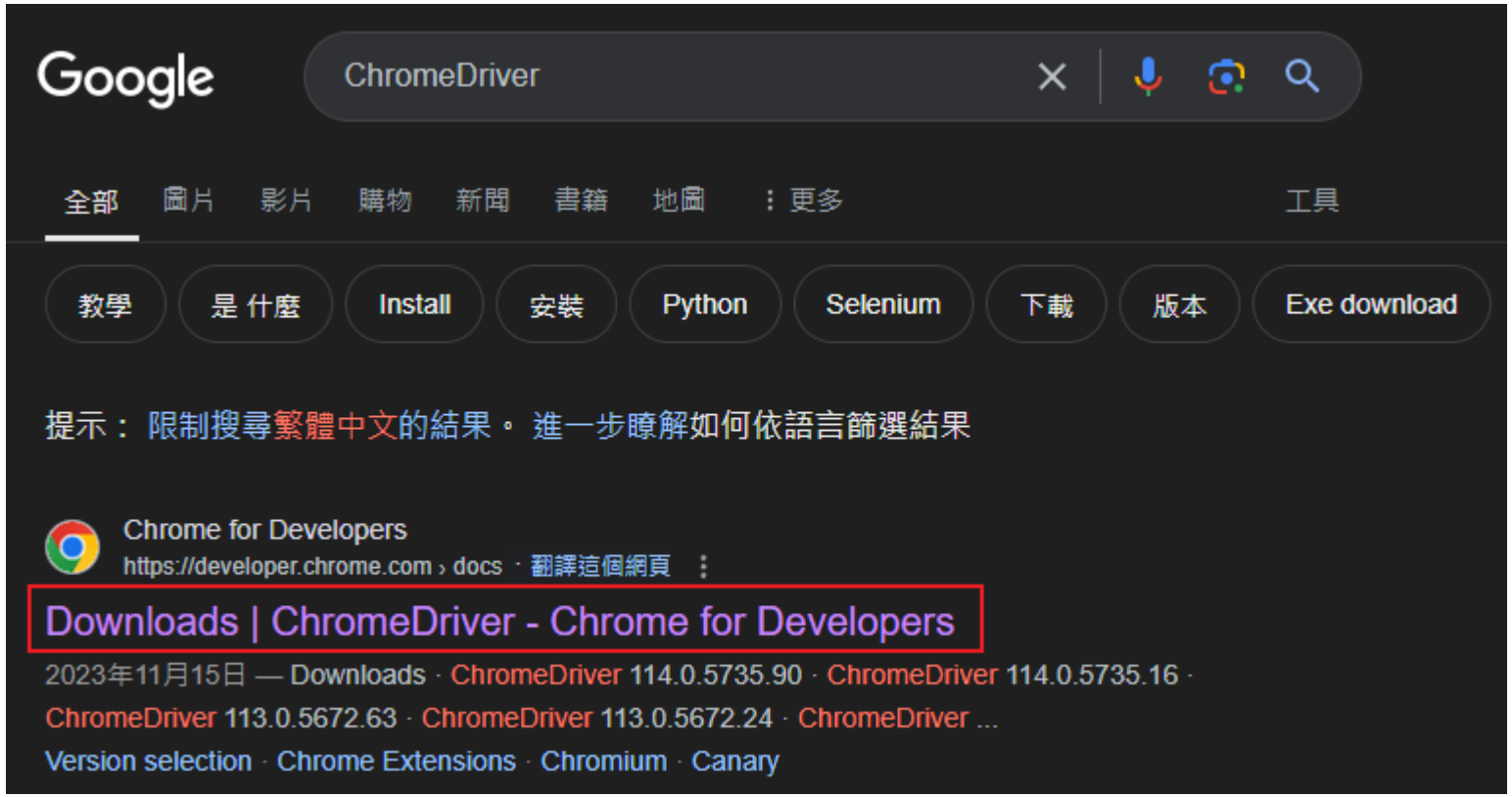
- 查看 pip 版本 → pip --version
- 更新 pip 版本 → python pip install --upgrade

Selenium - Webdriver

- Webdriver 是用來執行並操作瀏覽器的 API 介面，不同的瀏覽器(Chrome、Edge、Firefox...等)都會有各自對應的驅動程式 (driver)，Selenium 會透過 Webdriver 來直接對瀏覽器進行操作，並執行自動化作業，就如同使用者在操作。
- 在下載 Webdriver 之前要先檢查目前使用的瀏覽器版本 :
  - Chrome : 右上角選單 → 設定 → 關於 Chrome

Selenium – ChromeDriver 下載

1. 網頁搜尋 ChromeDriver



Selenium – ChromeDriver 下載

2. 如果瀏覽器版本為 115 以上，就點選 Chrome for Testing

3. 依照版本選擇 Channel

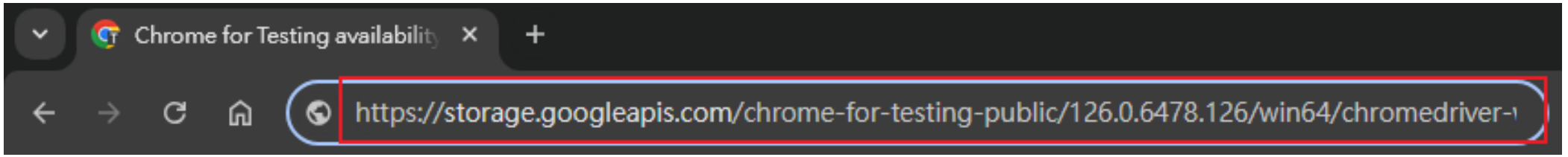
Channel	Version	Revision	Status
Stable	126.0.6478.126	r1300313	✓
Beta	127.0.6533.26	r1313161	✓
Dev	128.0.6559.0	r1319638	✓
Canary	128.0.6583.0	r1323996	✓

Selenium – ChromeDriver 下載

4. 複製要下載的 chromedriver 網址

chromedriver	**win32**	https://storage.googleapis.com/chrome-for-testing-public/126.0.6478.126/win32/chromedriver-win32.zip	200
chromedriver	**win64**	https://storage.googleapis.com/chrome-for-testing-public/126.0.6478.126/win64/chromedriver-win64.zip	200
chrome-headless-shell	**linux64**	https://storage.googleapis.com/chrome-for-testing-public/126.0.6478.126/linux64/chrome-headless-shell-linux64.zip	200

5. 將網址貼上網址列後送出



Selenium – 使用 webdriver

```
from selenium import webdriver


url = "https://www.google.com.tw/"
driver = webdriver.Chrome()
driver.get(url)


driver.quit() # 關閉瀏覽器
```

Selenium – webdriver 屬性與方法

driver 的屬性與方法	說明
`current_url`	取得目前頁面的 URL
`title`	取得目前頁面的標題
`page_source`	取得完整 HTML 原始碼
`save_screenshot(path)`	擷取整頁截圖
`get_cookies()`	取得所有 cookie
`add_cookie(cookie_dict)`	新增一筆 cookie
`quit()`	關閉整個瀏覽器與驅動程式
`execute_script(script)`	執行 JavaScript 程式碼

Selenium – execute_script()

- driver.execute_script() 是 Selenium 的強大功能之一，可以直接在瀏覽器中執行 JavaScript，用來抓取動態資料、滾動頁面、點擊隱藏元素等。

```
# 滾動到網頁底部
driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
time.sleep(3)  # 等待網頁載入完成
```

```
driver.execute_script("window.open('about:blank', '_blank');") # 開啟分頁
new_tabs = driver.window_handles
driver.switch_to.window(new_tabs[-1]) # 切換至新分頁
driver.get(url)
```

Selenium – webdriver 的搜尋方法

- **find_elements() → element_list | []**

  功能：

  用來定位多個網頁元素。返回一個包含所有符合條件的元素列表。如果沒有找到任何符合條件的元素，返回一個空列表。

- **find_element() → element obj | NoSuchElementException error**

  功能：

  用來定位單一網頁元素。當找到符合條件的第一個元素時，即返回該元素。如果沒有找到任何符合條件的元素，會拋出 NoSuchElementException 異常

Selenium – webdriver 的定位方法

定位方法	說明
`find_element(By.ID, 'id')`	透過元素的 id 屬性來定位元素。
`find_element(By.NAME, 'name')`	透過元素的 name 屬性來定位元素。
`find_element(By.CLASS_NAME, 'class_name')`	透過元素的 class 屬性來定位元素。
`find_element(By.TAG_NAME, 'tag_name')`	透過元素的 tag 名稱來定位元素，例如 div, span, a 等。
`find_element(By.LINK_TEXT, 'link_text')`	透過完全匹配的超連結來定位 <a> 元素。
`find_element(By.PARTIAL_LINK_TEXT, 'partial_link')`	透過部分匹配的超連結來定位 <a> 元素。
`find_element(By.XPATH, 'xpath')`	透過 XPath 表達式來定位元素。 適用於複雜的定位需求。
`find_element(By.CSS_SELECTOR, 'css_selector')`	透過 CSS 選擇器來定位元素。

Selenium – 版本差異

因 Selenium 4 和 Selenium 3 使用方法有差異，

在 安裝 以及 使用網路上的程式碼時 須特別留意

- 元素定位
  - (新) Selenium 4 →

```
# 需要先匯入 By 這個類
from selenium.webdriver.common.by import By
driver.find_element(By.CLASS_NAME, "className")
driver.find_element(By.ID, "xx")
```

  - (舊) Selenium 3 →

```
driver.find_elements_by_class_name("className")
driver.find_element_by_id("xx")
```

Selenium – element 的屬性和方法

屬性 / 方法	說明
text	獲取元素的文字內容。
tag_name	獲取元素的標籤名稱。
size	獲取元素的尺寸。
location	獲取元素的位置。
is_displayed()	判斷元素是否顯示在網頁上。
is_enabled()	判斷元素是否可用。
is_selected()	判斷元素是否被選中（主要用於選項和複選框）。
click()	點擊元素。
send_keys(*value)	向元素輸入文字。
clear()	清空元素的內容（適用於輸入框）。
submit()	提交表單。
get_attribute(name)	獲取元素的屬性值。
find_element(by, value)	在當前元素內查找子元素。
find_elements(by, value)	在當前元素內查找多個子元素。

Selenium – Explicit Waits

- Explicit Waits 是 Selenium 中的一種等待機制，用來解決網頁元素加載速度與自動化測試腳本執行速度不同步的問題。當我們需要等待某個條件成立後再繼續執行後續操作時，就會用到 Explicit Waits。

```
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

favoritenumber = WebDriverWait(driver, 10).until(
   EC.presence_of_element_located((By.CSS_SELECTOR, "span[id='favoritenumber']"))
).text
```

PyAutoGUI

- PyAutoGUI 是 Python 的跨平台 (Windows、macOS、Linux) 圖形化介面自動化工具，可控制滑鼠、鍵盤、螢幕截圖與影像辨識 (需搭配 OpenCV) 。
- 常見應用：重複操作自動化、表單填寫、截圖處理、簡易 GUI 測試、影像定位點擊等。

PyAutoGUI – 套件安裝

- PyAutoGUI 安裝 → `pip install pyautogui`
- OpenCV 安裝 → `pip install opencv-python`

通常在使用 PyAutoGUI 都會搭配截圖功能進行影像辨識

所以建議一起安裝 opencv 套件

PyAutoGUI – 螢幕與座標

方法	回傳	說明
`size()`	`(w, h)`	目前主桌面解析度
`position()`	`(x, y)`	目前滑鼠座標
`onScreen(x, y)`	`bool`	座標是否在螢幕內

方法	回傳	說明
`screenshot(path=None, region=None)`	`PIL.Image`	全螢幕或區域截圖
`locateOnScreen(image, confidence)`	`Box or None`	螢幕上尋找單一匹配
`locateAllOnScreen(image, confidence)`	迭代器	尋找所有匹配
`locateCenterOnScreen(image, confidence)`	`(x, y)`	找到中心點

PyAutoGUI – 在 cmd 執行

- 在 cmd 執行 python -m pyautogui 可以進入互動模式，幫助我們定位程式要執行的範圍或是起點

  終端機畫面上會即時顯示:

  - 滑鼠目前的位置 (X, Y 座標)
  - 滑鼠所在像素的 RGB 顏色值

```
(base) E:\Program\web_scraping>python -m pyautogui
Press Ctrl-C to quit.
X: 1192 Y:  894 RGB: (255, 255, 255)
```

PyAutoGUI – 滑鼠控制

方法	說明
`moveTo(x, y, duration)`	移動到絕對座標
`moveRel(xOffset, yOffset, duration)`	相對移動
`click(x, y, clicks, interval, button)`	點擊（可指定座標/次數/鍵）
`doubleClick() / rightClick()`	快捷點擊
`dragTo(x, y, duration, button)`	拖曳到座標
`scroll(clicks)`	滑鼠滾輪（上正下負）

PyAutoGUI – 鍵盤控制

方法	主要參數	回傳
write(text, interval)	str, float	None
press(key)	str	None
keyDown(key) / keyUp(key)	str	None
hotkey(*keys)	多鍵	None

在使用自動化控制時，可以搭配內建的 keyboard 模組 強制結束程式

```
import keyboard

while True:
    # 要執行的程式碼
    if keyboard.is_pressed('Q'):
        print("退出程式")
        break
```

影片下載 yt-dlp

- yt-dlp是一個功能強大、持續維護的命令列工具，主要用來**下載 YouTube 及其他影音網站的影片、音訊與字幕**。它是知名工具 youtube-dl 的分支，在效能、網站支援度與功能上都有大幅改進。

```
from yt_dlp import YoutubeDL
ydl_opts = {
    'outtmpl': '%(title)s.%(ext)s',  # 輸出檔名樣式
    'format': 'bestvideo+bestaudio/best',  # 選最佳畫質
}
url = 'https://www.instagram.com/p/DQtxUy_ExSt/'  # 替換成想下載的影片網址
with YoutubeDL(ydl_opts) as ydl:
    ydl.download([url])
```