Python 的 50+ 練習： 資料科學學習手冊

> *網路資料擷取*

數據交點 | 郭耀仁 https://linktr.ee/yaojenkuo

In [1]:

```
import requests
import json
import xml.etree.ElementTree as ET
from bs4 import BeautifulSoup
```

關於網路資料擷取

什麼是網路資料擷取?

> *網路資料擷取是一種透過超文本傳輸協定（Hypertext Transfer Protocol, HTTP）直接從網際網路獲取資料的技術，對比手動透過瀏覽器擷取資料，網路資料擷取通常指的是透過自動化的程式來進行，並將擷取所得資料儲存到資料庫或試算表供後續分析使用。*

來源：https://en.wikipedia.org/wiki/Web_scraping

網路資料擷取的兩個核心任務

1. 傳輸資料。
2. 解析資料。

什麼是傳輸資料?

傳輸資料指的是在網際網路的通訊協定下，瀏覽器（或者一段爬蟲程式）與網頁伺服器之間交換超文本資料的行為，專有技術名詞為超文本傳輸協定（Hypertext Transfer Protocol, HTTP）。

資料傳輸必定是雙向的

1. 瀏覽器（或者一段爬蟲程式）傳輸資料給網頁伺服器：請求（Request）。
2. 網頁伺服器傳輸資料給瀏覽器（或者一段爬蟲程式）：回應（Response）。

網路資料擷取最常使用的請求方法

- GET
- POST

什麼是 GET 請求?

請求網頁伺服器展示指定的資源，應用於取得資料，例如瀏覽 Instagram 相片。

什麼是 POST 請求?

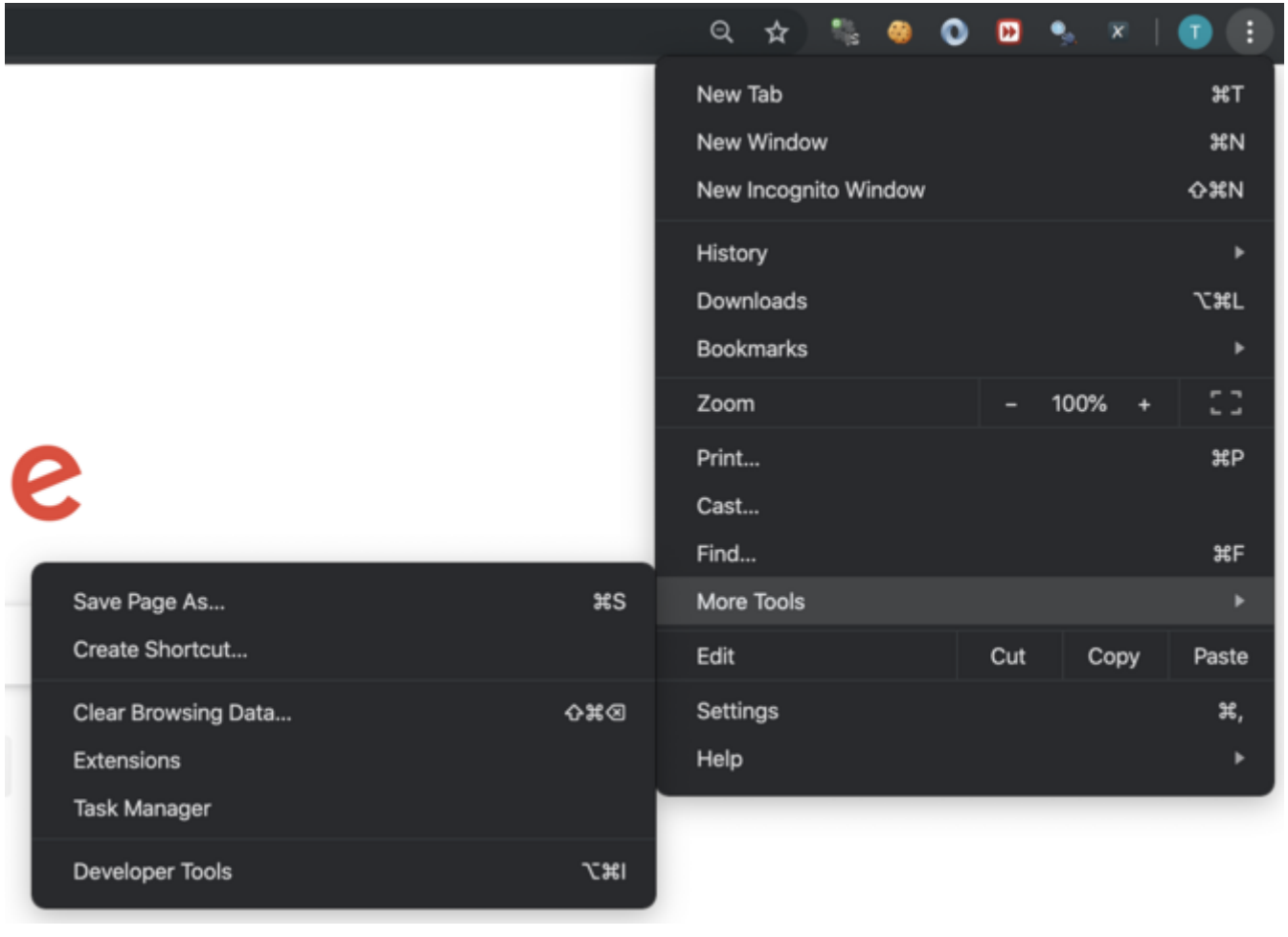
提交指定資源給網頁伺服器，通常伴隨著伺服器狀態的改變，例如分享 Instagram 相片。

透過瀏覽器的操作，我們以使用者介面進行資料的傳輸

- 在瀏覽器輸入網址（Uniform Resource Locator, URL）。
- 填寫表單然後提交。
- 與其他的 UI 元件互動，例如下拉式選單、滑桿或複選框等。

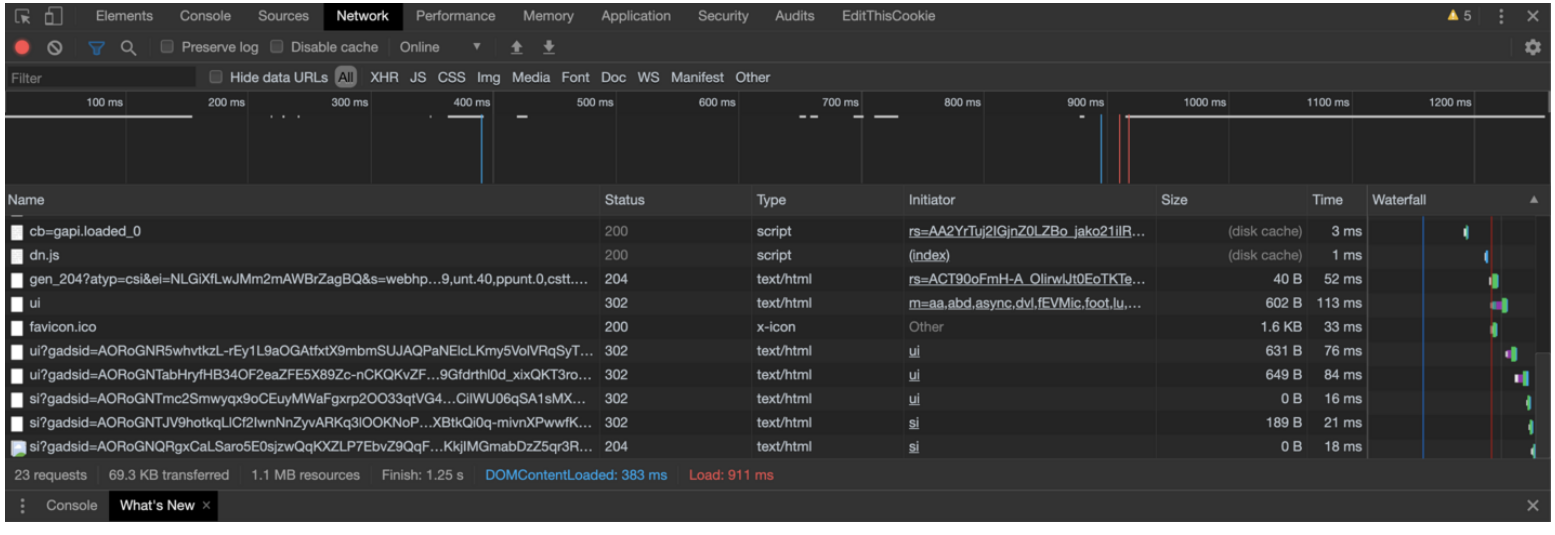
開發人員工具

我們可以透過瀏覽器的開發人員工具（Developer Tools）觀察資料的傳輸


New Tab ⌘T
New Window ⌘N
New Incognito Window ⇧⌘N
History
Downloads ⌥⌘L
Bookmarks
Zoom − 100% +
Print... ⌘P
Cast...
Find... ⌘F
More Tools
Edit Cut Copy Paste
Settings ⌘,
Help
Save Page As... ⌘S
Create Shortcut...
Clear Browsing Data... ⇧⌘⌫
Extensions
Task Manager
Developer Tools ⌥⌘I

點選 Network 觀察資料傳輸的細節

1. 開發人員工具。
2. 點選 Network 頁籤。
3. 重新整理網頁。


Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
EditThisCookie
Preserve log
Disable cache
Online
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
100 ms
200 ms
300 ms
400 ms
500 ms
600 ms
700 ms
800 ms
900 ms
1000 ms
1100 ms
1200 ms
Name
Status
Type
Initiator
Size
Time
Waterfall
cb=gapi.loaded_0 200 script rs=AA2YrTuj2IGjnZ0LZBo_jako21ilR… (disk cache) 3 ms
dn.js 200 script (index) (disk cache) 1 ms
gen_204?atyp=csi&ei=NLGiXfLwJMm2mAWBrZagBQ&s=webhp…9,unt.40,ppunt.0,cstt.… 204 text/html rs=ACT90oFmH-A_OlirwlJt0EoTKTe… 40 B 52 ms
ui 302 text/html m=aa,abd,async,dvl,fEVMic,foot,lu,… 602 B 113 ms
favicon.ico 200 x-icon Other 1.6 KB 33 ms
ui?gadsid=AORoGNR5whvtkzL-rEy1L9aOGAtfxtX9mbmSUJAQPaNElcLKmy5VolVRqSyT… 302 text/html ui 631 B 76 ms
ui?gadsid=AORoGNTabHryfHB34OF2eaZFE5X89Zc-nCKQKvZF…9Gfdrthl0d_xixQKT3ro… 302 text/html ui 649 B 84 ms
si?gadsid=AORoGNTmc2Smwyqx9oCEuyMWaFgxrp2OO33qtVG4…CilWU06qSA1sMX… 302 text/html ui 0 B 16 ms
si?gadsid=AORoGNTJV9hotkqLICf2lwnNnZyvARKq3lOOKNoP…XBtkQi0q-mivnXPwwfK… 302 text/html si 189 B 21 ms
si?gadsid=AORoGNQRgxCaLSaro5E0sjzwQqKXZLP7EbvZ9QqF…KkjIMGmabDzZ5qr3R… 204 text/html si 0 B 18 ms
23 requests
69.3 KB transferred
1.1 MB resources
Finish: 1.25 s
DOMContentLoaded: 383 ms
Load: 911 ms
Console
What's New

每一個檔案都是一組完整的請求與回覆

- Request
    - Headers
    - Method
- Response
    - Headers
    - Body

傳輸資料的不同格式

- XHR(XMLHttpRequest)。
- Doc(HTML documents)。
- 其他（JS、CSS、Img...等）。

可以藉由「暫時」關閉瀏覽器 JavaScript 來猜測資料格式

- 關閉 JavaScript 後資料消失，檢查 **XHR**。
- 關閉 JavaScript 後資料仍然存在，檢查 **Doc**。

「暫時」關閉 Chrome 瀏覽器 JavaScript 的外掛

JavaScript Switcher

幾個範例網站

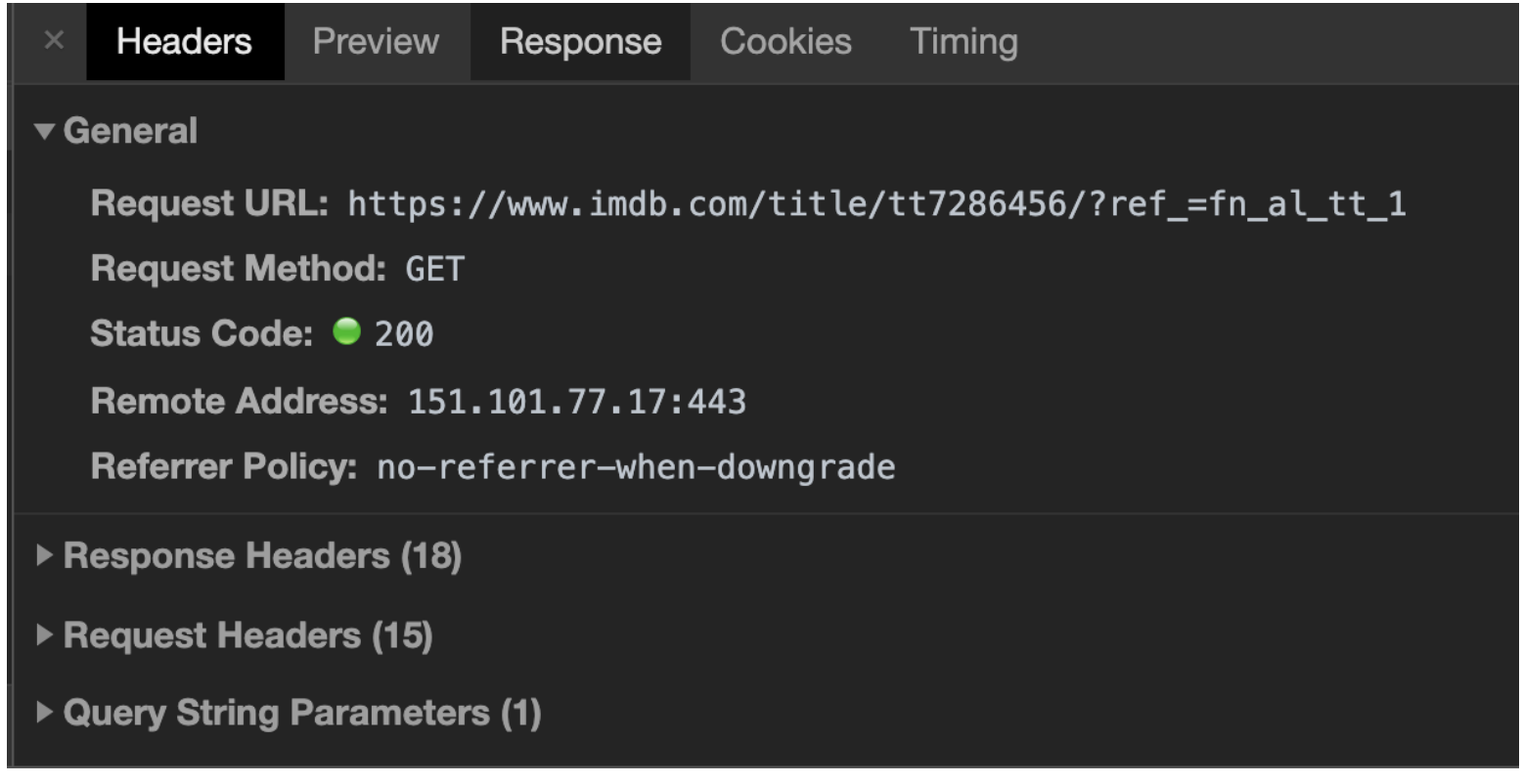
- https://ecshweb.pchome.com.tw/search/v3.3/：關閉 JavaScript 後資料消失，檢查 **XHR**。
- https://emap.pcsc.com.tw/：關閉 JavaScript 後資料消失，檢查 **XHR**。
- https://www.imdb.com/：關閉 JavaScript 後資料仍然存在，檢查 **Doc**。

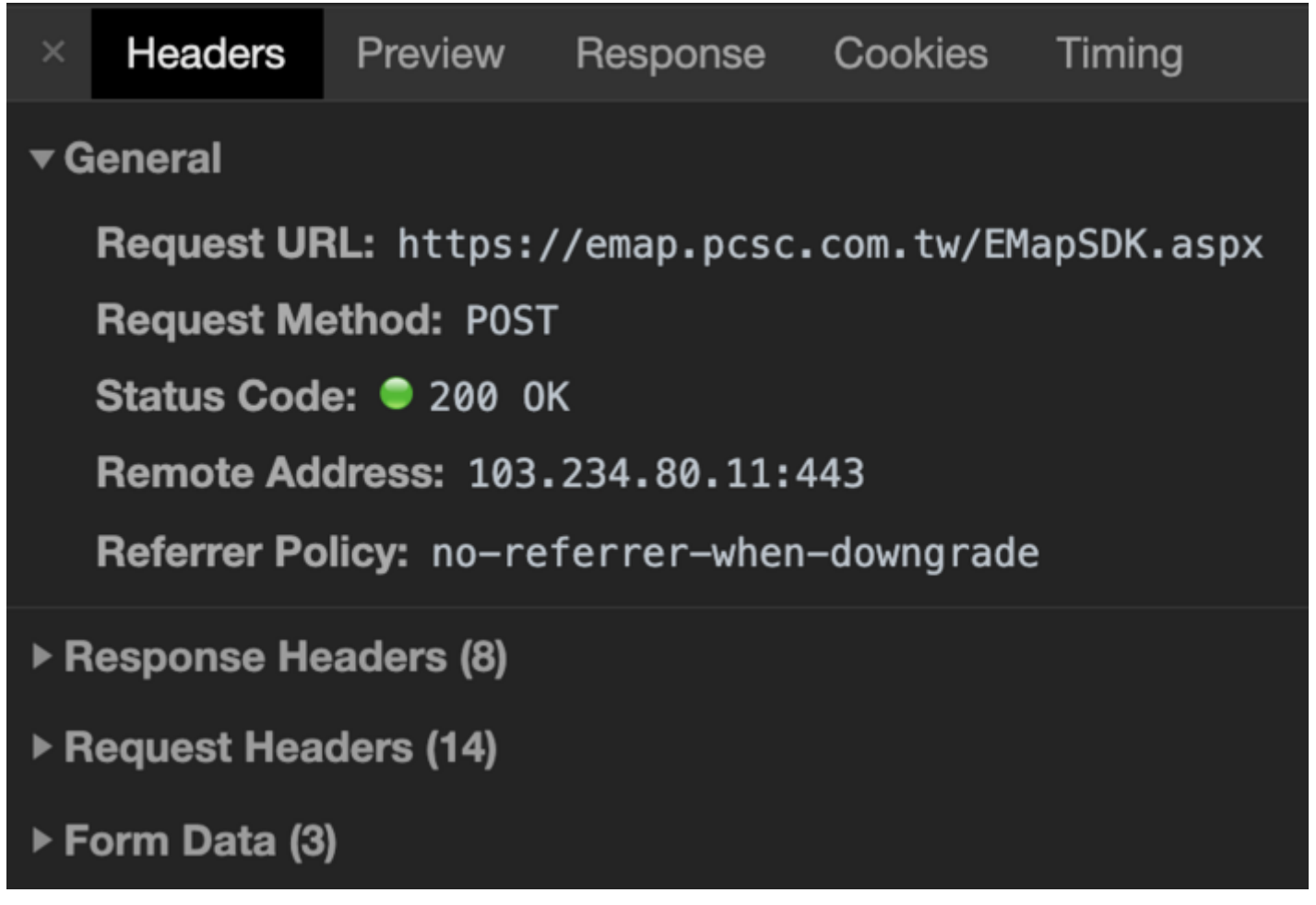
—但找到資料，可以檢查傳輸的細節

- Headers
    - General（包含資源索取的網址、請求方式等。）
    - Response Headers（關於回應的描述。）
    - Request Headers（關於請求的描述。）
    - Query String Parameters（不一定會有。）
    - Form Data（不一定會有，通常伴隨 POST 請求方法。）

—但找到資料，可以檢查傳輸的細節（續）

- Preview（回應在瀏覽器上渲染的樣子。）
- Response（回應的原始資料。）
- Cookies（不一定會有。）


Headers
Preview
Response
Cookies
Timing
General
Request URL: https://www.imdb.com/title/tt7286456/?ref_=fn_al_tt_1
Request Method: GET
Status Code: 200
Remote Address: 151.101.77.17:443
Referrer Policy: no-referrer-when-downgrade
Response Headers (18)
Request Headers (15)
Query String Parameters (1)


Headers
Preview
Response
Cookies
Timing
General
Request URL: https://emap.pcsc.com.tw/EMapSDK.aspx
Request Method: POST
Status Code: 200 OK
Remote Address: 103.234.80.11:443
Referrer Policy: no-referrer-when-downgrade
Response Headers (8)
Request Headers (14)
Form Data (3)

如何在 Python 程式與網頁伺服器之間傳輸資料?

透過第三方模組 Requests。

關於 Requests

什麼是 Requests?

> *Requests 是 Python 的超文本傳輸協定第三方模組，標榜簡潔且人性化。*

來源：https://requests.readthedocs.io/en/master/

以 `import` 指令載入 Requests

In [2]:

```
import requests
```

如果環境中沒有安裝 Requests，載入時會遭遇 `ModuleNotFoundError`

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'requests'
```

如果遭遇 `ModuleNotFoundError` 可以在終端機使用 `pip install` 指令安裝

```
pip install requests
```

可以透過兩個屬性檢查版本號與安裝路徑

- `__version__` 屬性檢查版本號。
- `__file__` 屬性檢查安裝路徑。

In [3]:

```
print(requests.__version__)
print(requests.__file__)
```

```
2.25.1
/Users/kuoyaojen/opt/miniconda3/envs/pyds/lib/python3.
8/site-packages/requests/__init__.py
```

使用 Requests 的兩個函數傳輸資料

- `get(request_url, params=query_str_params)`：對 `request_url` 發送帶有 `query_str_params` 的 GET 請求。
- `post(request_url, data=form_data)`：對 `request_url` 發送帶有 `form_data` 的 POST 請求。

In [4]:

```
request_url = "https://ecshweb.pchome.com.tw/search/v3.3/all/results"
query_str_params = {
    'q': 'macbook',
    'page': 1,
    'sort': 'rnk/dc'
}
response = requests.get(request_url, params=query_str_params)
```

In [5]:

```
request_url = "https://emap.pcsc.com.tw/EMapSDK.aspx"
form_data = {
    'commandid': 'SearchStore',
    'city': '台北市',
    'town': '大安區'
}
response = requests.post(request_url, data=form_data)
```

In [6]:

```
request_url = "https://www.imdb.com/title/tt10048342"
response = requests.get(request_url)
```

常用的 `Response` 實例屬性與方法

- `status_code` 屬性來驗證超文本傳輸協定的傳輸狀態。
- `text` 屬性以 Python 的文字型態儲存回應內容。
- `json` 方法以 Python 的資料結構儲存回應內容。

依照不同資料格式解析

JSON 格式：使用 `Response` 實例的 `json` 方法建立 Python 資料結構 `dict` 或 `list`。

依照不同資料格式解析（續）

XML 格式：利用標準模組 `xml` 的解析器將 `Response` 實例的 `text` 屬性轉換成 `element.tree` 實例。

依照不同資料格式解析（續）

HTML 格式：利用第三方模組 Beautiful Soup4 的解析器將 `Response` 實例的 `text` 屬性轉換成 `soup` 實例。

JSON 格式

什麼是 JSON?

> *JavaScript Object Notation (JSON)是依照 JavaScript 物件語法的資料格式，雖然是以 JavaScript 語法為基礎，但可獨立使用，且許多程式設計環境亦可讀取、解析並產生 JSON。JSON 可能是物件或字串。當你想從 JSON 中讀取資料時，JSON 可作為物件；當要跨網路傳送 JSON 時，就會是字串。JSON 物件可儲存於其自有的檔案中，基本上就是副檔名為 `.json` 的文字檔案。*

來源：mozilla.org

如何驗證回應是否為 JSON 格式?

在開發人員工具 > Network 檢視 Preview 或者 Response。

JSON 格式資料的兩種外型

- 鍵值外型：與 Python `dict` 相似。
- 陣列外型：與 Python `list` 相似。

看看 JSON 格式資料

https://ecshweb.pchome.com.tw/search/v3.3/：關閉 JavaScript 後資料消失，檢查 **XHR**。

In [7]:

```
request_url = "https://ecshweb.pchome.com.tw/search/v3.3/all/results"
query_str_params = {
    'q': 'macbook',
    'page': 1,
    'sort': 'rnk/dc'
}
response = requests.get(request_url, params=query_str_params)
```

使用 Response 實例的 `json()` 方法解析 JSON 格式資料為鍵值外型

In [8]:

```
json_format = response.json()
print(type(json_format))
print(json_format.keys())
```

```
<class 'dict'>
dict_keys(['QTime', 'totalRows', 'totalPage', 'range',
'cateName', 'q', 'subq', 'token', 'isMust', 'prods'])
```

事實上 `response` 實例的 `json()` 方法是使用了標準模組 `json` 的 `loads` 函數

In [9]:

```
json_format = json.loads(response.text)
print(type(json_format))
print(json_format.keys())
```

```
<class 'dict'>
dict_keys(['QTime', 'totalRows', 'totalPage', 'range',
'cateName', 'q', 'subq', 'token', 'isMust', 'prods'])
```

JSON 格式資料的陣列外型

In [10]:

```
print(type(json_format['prods']))
print(len(json_format['prods']))
```

```
<class 'list'>
20
```

XML 格式

什麼是 XML?

> *XML (Extensible Markup Language) 是一種標記語言，它使用沒有預先定義名稱或功能的標籤來儲存資料，並且能透過 XPath 語法取得所需資料。*

來源：https://developer.mozilla.org/en-US/docs/Web/XML

如何驗證回應是否為 XML 格式?

在開發人員工具 > Network 檢視 Response 的第一個標籤是否有明確宣告 XML 格式。

```
<?xml version="1.0"?>...
```

XML 格式資料的外型

由開發者自行定義的標籤組成有階層的樹狀結構。

看看 XML 格式資料

https://emap.pcsc.com.tw/：關閉 JavaScript 後資料消失，檢查 XHR。

In [11]:

```
request_url = "https://emap.pcsc.com.tw/EMapSDK.aspx"
form_data = {
    'commandid': 'SearchStore',
    'city': '台北市',
    'town': '大安區'
}
response = requests.post(request_url, data=form_data)
```

XML 格式：利用標準模組 `xml` 的解析器將 `Response` 實例的 `text` 屬性轉換成 `element.tree` 實例。

使用 `ET` 的 `fromstring` 函數轉換文字為「可解析」的實例

In [12]:

```
root = ET.fromstring(response.text)
print(type(root))
```

```
<class 'xml.etree.ElementTree.Element'>
```

透過 XPath 取得指定標籤儲存的資料

> *XPath 指的是 XML 路徑語言，透過該語言可以指定 XML 文件中指定的部分、標籤。*

來源：https://developer.mozilla.org/en-US/docs/Web/XPath

In [13]:

```
# The XPath for POIName
poinames = [e.text for e in root.findall('.//POIName')]
print(poinames)
```

```
['大台', '大信', '大敦', '中廣', '仁安', '仕吉', '台科一',
'永康', '禾光', '立仁', '光忠', '吉孝', '吉忠', '合旺', '合
維', '安居', '安松', '佑安', '技安', '辛亥', '卓聯', '和平
東', '和安', '和金', '和泰', '和樂', '延吉', '昇隆', '東門',
'欣安和', '欣隆昌', '花市', '金信', '金華', '長星', '阿波羅',
'信中', '信安', '信義', '信興', '建安', '建忠', '建南', '建
綸', '恆安', '科技站', '科建', '科興', '師大', '泰利', '國
館', '崇光', '康福', '教育大學', '統合', '統家', '統領', '通
化', '頂好', '頂安', '頂東', '喜悅', '富陽', '復忠', '復昌',
'復維', '敦仁', '敦禾', '敦安', '敦信', '敦南', '敦頂', '敦
隆', '敦維', '敦親', '森美', '華電', '愛國', '新北科', '新東
帝', '新泰順', '新國聯', '溫州', '溫東', '瑞升', '瑞安', '義
村', '誠安', '福亭', '鳳翔', '樂安', '樂利', '樂和', '樂隆',
'黎元', '豫銘', '錢忠', '靜安', '龍和', '龍延', '龍門', '龍
泉', '龍淵', '龍普', '濟南', '臨江', '臨通', '豐安', '懷生',
'羅鑫', '麟光', '鑫忠孝', '鑫泰', '鑫通', '鑫富民', '鑫復']
```

In [14]:

```
# The XPath for Address
addresses = [e.text for e in root.findall('.//Address')]
print(addresses)
```

```
['台北市大安區羅斯福路三段283巷14弄16號1樓', '台北市大安區信義路三段33號', '台北市大安區敦化南路二段63巷7號1樓', '台北市大安區仁愛路三段25-1號27號', '台北市大安區仁愛路四段27巷1號', '台北市大安區忠孝東路四段223巷42號', '台北市大安區基隆路四段43號1樓', '台北市大安區永康街43號', '台北市大安區和平東路二段63號1樓', '台北市大安區安和路二段74巷1號', '台北市大安區復興南路一段107巷5弄1號1樓', '台北市大安區忠孝東路四段299號', '台北市大安區延吉街72號', '台北市大安區復興南路二段151巷41號', '台北市大安區四維路170巷8號1樓', '台北市大安區安居街33號', '台北市大安區安東街50之2號50之3號50之4號', '台北市大安區忠孝東路三段217巷1弄2號', '台北市大安區和平東路三段97號', '台北市大安區辛亥路二段57號', '台北市大安區羅斯福路四段1號1樓卓聯大樓', '台北市大安區和平東路一段129之1號', '台北市大安區和平東路三段230號', '台北市大安區和平東路一段91號', '台北市大安區和平東路一段169號', '台北市大安區和平東路三段228巷45號1樓', '台北市大安區延吉街237號', '台北市大安區敦化南路二段238號', '台北市大安區信義路二段198巷6號1樓', '台北市大安區安和路一段47號', '台北市大安區基隆路二段142之1號及142之2號', '台北市大安區建國南路一段274號', '台北市大安區金山南路二段18號1樓', '台北市大安區金華街140號1樓', '台北市大安區基隆路三段85號', '台北市大安區忠孝東路四段222號224號1樓', '台北市大安區信義路三段101號', '台北市大安區大安路一段218號', '台北市大安區信義路四段265巷12弄1號', '台北市大安區信義路四段32號', '台北市大安區
```

敦化南路一段187巷29號', '台北市大安區忠孝東路三段249號', '台北市大安區建國南路二段151巷6之8號', '台北市大安區仁愛路四段151巷33號忠孝東路四段216巷32弄19號21號', '台北市大安區永康街2巷12號1樓', '台北市大安區復興南路二段203號', '台北市大安區建國南路一段28號30號', '台北市大安區復興南路二段271巷2號1樓', '台北市大安區師大路87號', '台北市大安區仁愛路四段266巷15弄22號', '台北市大安區光復南路240巷25號', '台北市大安區復興南路一段135巷5號', '台北市大安區永康街12之2號1樓', '台北市大安區敦南街38號', '台北市大安區忠孝東路四段181巷7弄11之1號11之2號', '台北市大安區忠孝東路四段216巷27弄1號1樓', '台北市大安區忠孝東路四段205巷7弄5號1樓', '台北市大安區通化街26之8號', '台北市大安區仁愛路四段79號1號', '台北市大安區大安路一段67號1樓', '台北市大安區大安路一段43號', '台北市大安區復興南路二段82-1及82-2號', '台北市大安區和平東路三段298號300號1樓', '台北市大安區光復南路98之3號98之5號', '台北市大安區通化里光復南路616號', '台北市大安區復興南路二段17號', '台北市大安區忠孝東路四段148號部份', '台北市大安區敦化南路二段265巷6號1樓', '台北市大安區安和路一段86號', '台北市大安區仁愛路四段122巷50號1樓', '台北市大安區敦化南路一段236巷13號', '台北市大安區忠孝東路四段101巷7號', '台北市大安區敦化南路二段182號', '台北市大安區東豐街43號45號1樓', '台北市大安區辛亥路二段171巷8號', '台北市大安區信義路三段65號1樓', '台北市大安區通化街177號', '台北市大安區愛國東路75號', '台北市大安區新生南路一段3號B1樓', '台北市大安區敦化南路二段99號1樓', '台北市大安區泰順街13號', '台北市大安區光復南路180巷12號', '台北市大安區羅斯福路三段245號', '台北市大安區和平東路一段266號', '台北市大安區杭州南路二段91號', '台北市大安區瑞安街182號', '台北市大安區忠孝東路三段160號', '台北市大安區敦化南路一段247巷12號1樓', '台北市大安區羅斯福路二段31

號1樓', '台北市大安區忠孝東路四段216巷68號', '台北市大安區樂業街71號73號', '台北市大安區樂利路76號78號1樓', '台北市大安區樂利路29號29-1號', '台北市大安區敦化南路二段331巷14號', '台北市大安區臥龍街188巷1號', '台北市大安區大安路二段102號', '台北市大安區忠孝東路四段26巷5號', '台北市大安區光復南路262號', '台北市大安區和平東路二段197號199號1樓', '台北市大安區師大路59巷13號', '台北市大安區和平東路二段38之1號40號1樓', '台北市大安區羅斯福路三段193號1樓', '台北市大安區和平東路二段118巷33號', '台北市大安區敦化南路一段233巷25號', '台北市大安區濟南路三段12號1樓', '台北市大安區安和路二段67號', '台北市大安區通安街64號1樓', '台北市大安區東豐街9號', '台北市大安區忠孝東路三段248巷9號', '台北市大安區羅斯福路三段29號31號1樓', '台北市大安區臥龍街252號及252-1號', '台北市大安區忠孝東路四段313號1樓', '台北市大安區古風里泰順街64號1樓', '台北市大安區信義路四段294巷7號1樓', '台北市大安區忠孝東路四段181巷40弄22號1樓', '台北市大安區信義路三段178號1樓']

HTML 格式

什麼是 HTML?

> *HTML（HyperText Markup Language，超文本標記語言）是打造網頁的基石，是表述並定義網頁架構的標記語言。HTML 包含了一系列的元素（elements），而元素包含了預先定義功能以及名稱的標籤與內容，並透過標籤來控制內容的呈現樣貌。*

來源：https://developer.mozilla.org/en-US/docs/Glossary/HTML

如何驗證回應是否為 HTML 格式?

在開發人員工具 > Network 檢視 Response 的第一個標籤是否有明確宣告 HTML 格式。

```
<!DOCTYPE html>...
```

HTML 格式資料的外型

由預先定義的標籤組成有階層的樹狀結構。

看看 HTML 格式資料

https://www.imdb.com/：關閉 JavaScript 後資料仍然存在，檢查 Doc。

In [15]:

```
request_url = "https://www.imdb.com/title/tt0111161"
response = requests.get(request_url)
```

HTML 格式：利用第三方模組 Beautiful Soup 的解析器將`Response` 實例的 `text` 屬性轉換成 `soup` 實例。

什麼是 Beautiful Soup?

> *Beautiful Soup 是 Python 的第三方模組，可以讓使用者快速解析 HTML 格式資料，從中擷取出使用者需要的資料。*

來源：https://www.crummy.com/software/BeautifulSoup/bs4/doc/

以 `from bs4 import BeautifulSoup` 指令載入函數

In [16]:

```
from bs4 import BeautifulSoup
```

如果環境中沒有安裝 Beautiful Soup，載入時會遭遇 `ModuleNotFoundError`

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'bs4'
```

如果遭遇 `ModuleNotFoundError` 可以在終端機使用 `pip install` 指令安裝

```
pip install beautifulsoup4
```

使用 Beautiful Soup 的 `BeautifulSoup` 函數轉換文字為「可解析」的實例

In [17]:

```
soup = BeautifulSoup(response.text)
print(type(soup))
```

```
<class 'bs4.BeautifulSoup'>
```

使用 CSS 選擇器取得指定標籤中的內容或屬性

> *CSS 選擇器是 CSS 規則的一部分，它能讓開發者選定要調整哪個（或哪些）HTML 元素的樣式，在爬蟲程式中我們利用 CSS 選擇進行「選定」但是不做「調整」，這是與網頁前端工程最大的不同。*

來源：https://developer.mozilla.org/en-US/docs/Glossary/CSS_Selector

CSS 基本選擇器包含

- 類型選擇器 `elementname` 例如 `h1`。
- Class 選擇器 `.classname` 例如 `.poster`。
- ID 選擇器 `#idname` 例如 `#main`。
- ...等。

非網頁前端工程師不易快速判斷 CSS 選擇器

- 在「欲選擇」的網頁元素按右鍵 > Inspect 觀察 CSS 選擇器該如何宣告。
- 有時可以透過 Chrome 瀏覽器外掛 Selector Gadget 輔助。

In [18]:

```
# The CSS Selector for title
title = soup.select('h1')[0].text.strip()
print(title)
```

```
刺激1995
```

In [19]:

```
# The CSS Selector for rating
rating = float(soup.select('a div span')[0].text)
print(rating)
```

```
9.3
```

In [20]:

```
# The CSS Selector for poster
poster = soup.select('img')[0].get('src')
print(poster)
```

```
https://m.media-amazon.com/images/M/MV5BMDFkYTc0MGEtZmN
hMC00ZDIzLWFmNTEtODM1ZmRlYWMwMWFmXkEyXkFqcGdeQXVyMTMxOD
k2OTU@._V1_QL75_UX190_CR0,0,190,281_.jpg
```

In [21]:

```
# The CSS Selector for cast
request_url = "https://www.imdb.com/title/tt0111161/fullcredits"
response = requests.get(request_url)
soup = BeautifulSoup(response.text)
cast = [e.text.strip() for e in soup.select('.primary_photo+ td a')]
print(cast)
```

```
['Tim Robbins', 'Morgan Freeman', 'Bob Gunton', 'Willia
m Sadler', 'Clancy Brown', 'Gil Bellows', 'Mark Rolsto
n', 'James Whitmore', 'Jeffrey DeMunn', 'Larry Brandenb
urg', 'Neil Giuntoli', 'Brian Libby', 'David Proval',
'Joseph Ragno', 'Jude Ciccolella', 'Paul McCrane', 'Ren
ee Blaine', 'Scott Mann', 'John Horton', 'Gordon Green
e', 'Alfonso Freeman', 'V.J. Foster', 'John E. Summer
s', 'Frank Medrano', 'Mack Miles', 'Alan R. Kessler',
'Morgan Lund', 'Cornell Wallace', 'Gary Lee Davis', 'Ne
il Summers', 'Ned Bellamy', 'Joe Pecoraro', 'Harold E.
Cope Jr.', 'Brian Delate', 'Don McManus', 'Donald Zin
n', 'Dorothy Silver', 'Robert Haley', 'Dana Snyder', 'J
ohn D. Craig', 'Ken Magee', 'Eugene C. DePasquale', 'Bi
ll Bolender', 'Ron Newell', 'John R. Woodward', 'Chuck
Brauchler', 'Dion Anderson', 'Claire Slemmer', 'James K
isicki', 'Rohn Thomas', 'Charlie Kearns', 'Rob Reider',
'Brian Brophy', 'Paul Kennedy', 'James Babson', 'Dennis
Baker', 'Fred Culbertson', 'Richard Doone', 'Shane Grov
e', 'Rita Hayworth', 'David Hecht', 'Alonzo F. Jones',
'Gary Jones', 'Sergio Kato', 'Michael Lightsey', 'Georg
```

```
e Macready', 'Robin J. Meyer', 'Christopher Page', 'Nei
l Riddaway', 'Brad Spencer', 'Jodiviah Stepp', 'Mark A.
Strain']
```

In [22]:

```
# The CSS Selector for characters
characters = [e.text.strip() for e in soup.select('.character a:nth-child(1)')]
print(characters)
```

```
['Andy Dufresne', "Ellis Boyd 'Red' Redding", 'Warden N
orton', 'Heywood', 'Captain Hadley', 'Tommy', 'Bogs Dia
mond', 'Brooks Hatlen', '1946 D.A.', 'Skeet', 'Jigger',
'Floyd', 'Snooze', 'Ernie', 'Guard Mert', 'Guard Trou
t', "Andy Dufresne's Wife", 'Glenn Quentin', '1946 Judg
e', '1947 Parole Hearings Man', 'Fresh Fish Con', 'Hung
ry Fish Con', 'New Fish Guard', 'Fat Ass', 'Tyrell', 'L
aundry Bob', 'Laundry Truck Driver', 'Laundry Leonard',
'Rooster', 'Pete', 'Guard Youngblood', 'Projectionist',
'Hole Guard', 'Guard Dekins', 'Guard Wiley', 'Moresby B
atter', '1954 Landlady', '1954 Food-Way Manager', '1954
Food-Way Woman', '1957 Parole Hearings Man', 'Ned Grime
s', 'Mail Caller', 'Elmo Blatch', 'Elderly Hole Guard',
'Bullhorn Tower Guard', 'Man Missing Guard', 'Head Bull
Haig', 'Bank Teller', 'Bank Manager', 'Bugle Editor',
'1966 D.A.', 'Duty Guard', '1967 Parole Hearings Man',
'1967 Food-Way Manager', 'Con', 'Old Man on Bus', 'Poli
ce Officer', 'Con', 'Inmate', 'Gilda Mundson Farrell',
'Bank Teller', 'Inmate', 'Convict', 'Inmate II', 'Con',
'Ballin Mundson', "Robin - Andy's Waitress", 'Traffic
(driver)', 'Con', '1957 Parole Hearings Guard', 'New Fi
sh Con', 'Yard Inmate']
```

如果對網路資料擷取很有興趣，建議再進修網頁工程

- HTML/CSS/JavaScript 基礎。
- 後端語言以及框架。
- 資料庫基礎。

重點統整

- 網路資料擷取的兩個核心任務
    - 傳輸資料。
    - 解析資料。
- 使用 Requests 的兩個函數傳輸資料
    - `get(request_url, params=query_str_params)`：對 `request_url` 發送帶有 `query_str_params` 的 GET 請求。
    - `post(request_url, data=form_data)`：對 `request_url` 發送帶有 `form_data` 的 POST 請求。

重點統整（續）

- 依照不同資料格式解析：
    - JSON 格式：使用 Response 實例的 `json()` 方法建立 Python 資料結構 `dict` 或 `list`
    - XML 格式：利用標準模組 `xml` 的解析器將 Response 實例的 `text` 屬性轉換成 `element.tree` 實例。
    - HTML 格式：利用第三方模組 Beautiful Soup4 的解析器將 Response 實例的 `text` 屬性轉換成 `soup` 實例。
- XML 格式：透過 XPath 取得指定標籤儲存的資料。
- HTML 格式：使用 XPath 或 CSS 選擇器取得指定標籤中的內容或屬性。