

CHAPTER

6

網路爬蟲實作案例

- 6-1 使用 Web API 取得網路資料
- 6-2 網路爬蟲的常見問題
- 6-3 擷取多筆記錄和 HTML 表格資料
- 6-4 擷取多頁面的分頁記錄資料
- 6-5 實作案例：majortests.com 的單字清單
- 6-6 實作案例：批踢踢 PTT BBS 熱門貼文

6-1

使用 Web API 取得網路資料

Web API 是一種 REST API，REST (REpresentational State Transfer) 是架構在 WWW 的 Web 應用程式架構，目前政府機構和各大軟體廠商都提供有付費或免費的 Web API，可以直接撰寫 Python 程式透過 Web API 來取得網路資料。

6-1-1 認識 Web API

Web API (Web Application Programming Interface) 是一種透過 Internet 網際網路來執行其他系統提供的功能，我們就是使用 HTTP 請求來執行其他系統提供的 Web API，如同函數呼叫。

如同瀏覽器輸入 URL 網址來瀏覽網頁，很多公開 API 可以直接在瀏覽器執行 API 來取得網路資料，回應資料大多是 JSON 格式的資料。

☆ Web API 的種類

目前的 Web API 主要可以分為兩種，其簡單說明如下所示：

- ◆ **公開 API (Public/Open API)：**任何人不需註冊帳號就可以使用的 Web API。
- ◆ **認證 API (Authenticated API)：**需要先註冊帳號後，取得認證資料，才能使用 Web API。這些帳號可能需付費或免費註冊，在註冊後，可以得到 API 金鑰 (API Key)，執行 Web API 時，需要提供 API 金鑰的認證資料。

☆ Web API 的認證方式

一般來說，當 Web API 是使用 GET 方法的 HTTP 請求時，有些是公開；有些需要認證；如果是使用 POST 方法，大部分都需要認證。Web API 認證方式主要有 2 種，其說明如下所示：

- ◆ **使用 API 金鑰認證：**當註冊 Web API 帳號取得 API 金鑰後，GET 方法是使用參數來指定認證資料，POST 方法是使用自訂標頭名稱來指定認證資料。
- ◆ **使用帳號和密碼認證：**使用註冊的帳號和密碼進行認證，請參閱 Web API 說明文件，有可能是使用自訂標頭來指定帳號和密碼，或使用 get() 函數的參數來指定帳號和密碼。

6-1-2 直接從網站下載資料

目前很多網站或政府單位的 Open Data 開放資料網站都可以直接下載資料，我們不用撰寫任何 Python 程式碼就可以取得所需資料。

☆ 下載台灣期交所未平倉量

台灣期交所三大法人未平倉量的下載網址，如下所示：

- ◆ <https://www.taifex.com.tw/cht/3/futAndOptDateView>

<ul style="list-style-type: none"> 交易資訊 三大法人 <ul style="list-style-type: none"> 查詢 下載 <ul style="list-style-type: none"> 總表 區分期貨與選擇權二類 <ul style="list-style-type: none"> 依日期 依週別 區分各期貨契約 區分各選擇權契約 選擇權買賣權分計 大額交易者未沖銷部位結構 每日外幣參考匯率查詢 交易歷史資料申請 	<ul style="list-style-type: none"> 本表僅包含本公司指數期貨契約（未含東證期貨、美國道瓊、美國標普500、美國那斯達克100期貨及英國富時100期貨）、選擇權契約、股票期貨契約及股票選擇權契約。 「多方」係指交易者看多指數之交易或未平倉口數，其型態包括買進期貨、買進選擇權買權、賣出選擇權賣權，「空方」係指交易者看空指數之交易或未平倉口數，其型態包括賣出期貨、賣出選擇權賣權、買進選擇權買權。 「契約金額」係以當日交易或未平倉口數依下列方式計算： <ol style="list-style-type: none"> 期貨：以每筆交易價格乘以契約乘數再乘以交易口數後加總。 選擇權：以每筆交易權利金乘以契約乘數再乘以交易口數後加總。 未平倉契約金額以各期貨契約或選擇權序列當日結算價格，乘以契約乘數再乘以未平倉口數後加總。 期交所公告之三大法人交易資訊，係包含自營商、投信及外資，此三類別均是由眾多法人機構集合而成，故無論是多方、空方或多空淨額，僅代表眾多法人機構不同看法合計互抵的結果，而非單一特定法人機構之交易策略，亦不能代表該類法人整體之交易策略。 三大法人交易資訊之公佈，係為提供交易者對當日期貨市場交易概況之參考，期貨商或媒體於引用數據發布公開訊息時，宜特別注意，以免對市場有所誤導。
---	---

區分期貨與選擇權二類下載

日期(起):

日期(迄):

在上述表格輸入日期範圍，按下載鈕，可以下載三大法人未平倉量的 CSV 檔案。

☆ 下載台灣期交所大額交易人未平倉量

台灣期交所下載大額交易人未平倉量的 URL 網址，如下所示：

◆ <https://www.taifex.com.tw/cht/3/largeTraderFutView>

交易資訊

期貨大額交易人未沖銷部位資料下載

期貨大額交易人未沖銷部位資料下載

日期(起): 2022/06/14

日期(迄): 2022/06/15

下載

備註：

- 1.自2008年12月17日起，本表於最後結算日所揭露之未沖銷部位不含當月份到期商品之數量。
- 2."-"表當日無一選到期選擇權契約；"0"表該契約於當日收盤後無未沖銷部位。
- 3.本表下載資料：
(1)當13:45收盤之商品未平倉資訊揭示後，所揭示之當日交易資訊含鉅額交易，惟不含標的證券為國外成分證券ETFs或境外指數ETFs、匯率期貨、東證期貨、(臺幣)黃金期貨之交易量。
(2)當16:15收盤之商品未平倉資訊揭示後，所揭示之當日交易資訊為所有商品(含鉅額交易)之交易量。

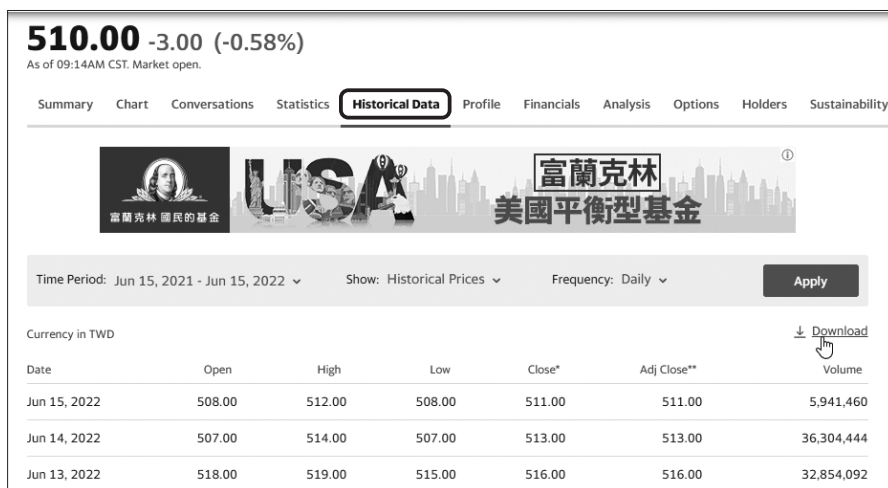
在上述表格輸入日期範圍，按下載鈕，可以下載大額交易人未平倉量的 CSV 檔案。

☆ 下載美國 Yahoo 的股票歷史資料

在美國 Yahoo 財經網站可以下載股票的歷史資料，例如：台積電，其 URL 網址，如下所示：

◆ <https://finance.yahoo.com/quote/2330.TW>

上述網址最後的 2330 是台積電的股票代碼，.TW 是台灣股市，如下圖所示：



請在上述網頁選 **Historical Data** 標籤後，在下方左邊選擇時間範圍，右邊按 **Apply** 鈕顯示股票的歷史資料後，點選下方 **Download** 超連結，可以下載以股票名稱為名的 CSV 檔案。

6-1-3 使用 Web API 取得網路資料

Google 圖書查詢前幾章就介紹過了，是使用 Google Books APIs 查詢圖書資訊，這是一個免費的公開 Web API，其回應資料是 JSON 格式資料，我們準備建立 Python 程式來查詢 Python 圖書資訊。

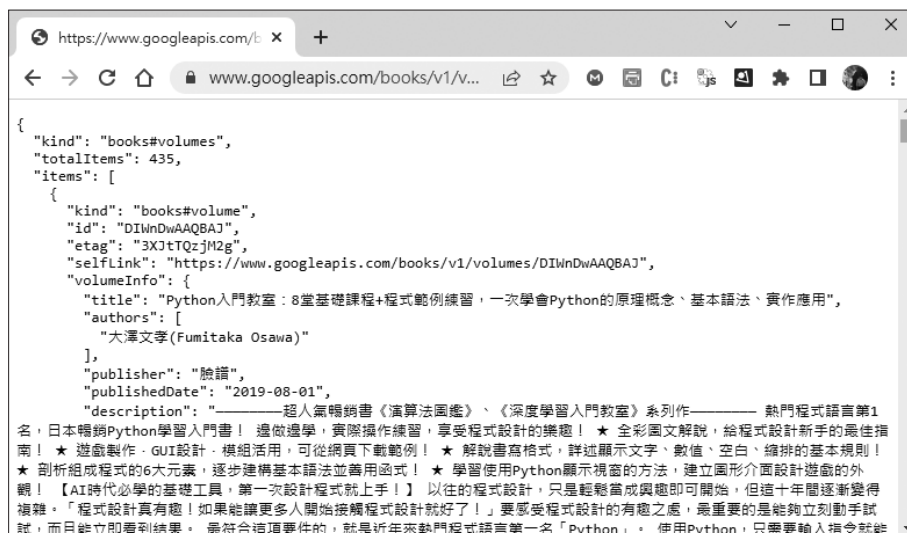
☆ 使用 Google Books APIs

Google Book APIs 可以在線上查詢指定條件的圖書資訊，其 API 格式如下所示：

◆ `https://www.googleapis.com/books/v1/volumes?q=<關鍵字>&maxResults=5&projection=lite`

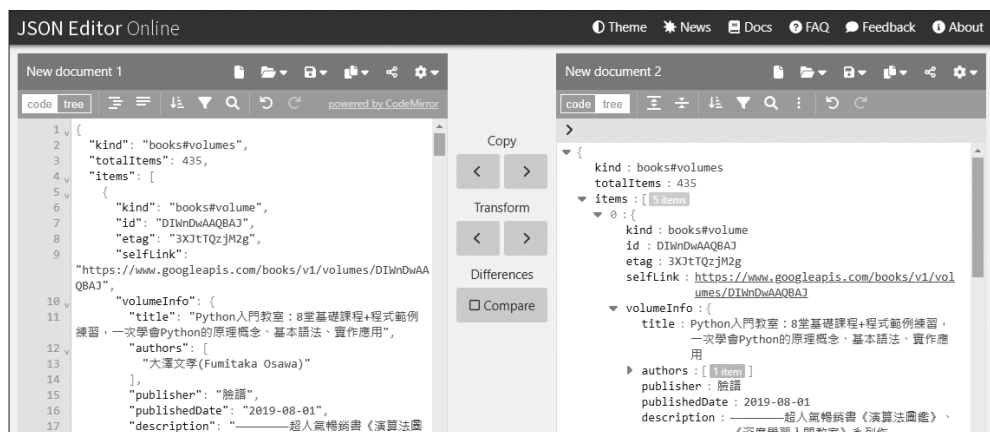
上述網址的 `q` 參數是關鍵字，`maxResults` 是最大搜尋筆數，5 是最多 5 筆圖書，最後 1 個參數是取回精簡圖書資料。例如：查詢 Python 圖書，如下所示：

◆ <https://www.googleapis.com/books/v1/volumes?maxResults=5&q=Python&projection=lite>



上述圖例可以看出這是 JSON 格式的資料，其結構是一個 JSON 物件。我們可以使用線上 JSON 編輯器來顯示 JSON 資料的階層結構，如下所示：

◆ <https://jsoneditoronline.org/>



請按 **Ctrl** + **A** 鍵將瀏覽器的全部 JSON 資料複製至上述圖例的左邊編輯區域，按 Transform 下的 > 鈕，再按 **Transform** 鈕，就可以在右邊看到 JSON 資料的階層結構。

請展開階層結構，totalItems 鍵的值是圖書總數 435，items 鍵的值是 JSON 物件陣列，共有 5 本書（索引值是 0~4），這是每一本圖書的 JSON 物件，在展開後，volumeInfo 鍵的值是圖書資訊，title 鍵值是書名；authors 鍵的值是 JSON 陣列的作者清單。

☆ 將 Google 圖書查詢的 JSON 資料寫入檔案：ch6-1-3.py

Python 程式可以將 Google 圖書查詢 Web API 取回的 JSON 資料，寫入 GoogleBooks.json 檔案，如下所示：

```
import json
import requests

url = "https://www.googleapis.com/books/v1/volumes?maxResults=5&q=Python&projection=lite"
jsonfile = "GoogleBooks.json"
r = requests.get(url)
r.encoding = "utf8"
json_data = json.loads(r.text)
with open(jsonfile, 'w') as fp:
    json.dump(json_data, fp)
```

上述程式碼使用 requests.get() 函數送出 HTTP 請求後，呼叫 json.loads() 函數將讀取資料轉換成字典，然後開啟寫入檔案 GoogleBooks.json，呼叫 json.dump() 函數寫入 JSON 資料至檔案，可以在 Python 程式相同目錄看到建立的 GoogleBooks.json 檔案。

6-2

網路爬蟲的常見問題

網路爬蟲是向 Web 伺服器送出 HTTP 請求後，從回傳 HTML 網頁擷取出所需的資料，本書 Python 程式是使用 requests 或 Selenium 向目標 URL 網址送出 HTTP 請求。不過，因為目前很多網站都內建防爬機制，在連線時可能會遇到一些問題，在這一節我們就來看一看一些常見的網路爬蟲問題。

☆ 判斷是否是 JavaScript 產生的網頁內容

HTML 網頁內容有可能是 JavaScript 程式產生的內容，在進行網路爬蟲的第一步就是判斷是否是 JavaScript 產生的網頁內容，我們可以使用第 3-1-2 節的 Quick JavaScript Switcher 擴充功能來進行判斷。

當判斷不是 JavaScript 產生的網頁內容時，就可以使用第 3-2 節的 requests 向目標 URL 網址送出 HTTP 請求，如果是 JavaScript 產生的網頁內容時，我們需要使用第 3-5 節的 Selenium 向目標 URL 網址送出 HTTP 請求。

☆ 更改 HTTP 標頭偽裝成瀏覽器送出 HTTP 請求

在第 3-4-2 節的 ch3-4-2.py 可以看出如果使用 requests 送出 HTTP 請求，Web 網站是可以知道這是 Python 程式送出的請求，並不是瀏覽器送出。例如：ch6-2.py 送出 HTTP 請求至 momo 網站，如下所示：

```
import requests

URL = "https://www.momoshop.com.tw/search/"

r = requests.get(URL+"searchShop.jsp?keyword=NBA")
if r.status_code == requests.codes.ok:
    r.encoding = "big5"
    print(r.text)
else:
    print("HTTP 請求錯誤..." + url)
```


上述程式碼使用 requests 送出 HTTP 請求，執行結果會看到連線錯誤，如下所示：

```
...  
File ~\anaconda3\lib\site-packages\requests\adapters.py:501 in send  
    raise ConnectionError(err, request=request)  
  
ConnectionError: ('Connection aborted.', ConnectionResetError(10054, '遠端主機已強制關閉一個現存的連線。', None, 10054, None))
```

在第 3-4-2 節已經說明過更改標頭資訊方式，Python 程式可以更改標頭資訊，假裝是從瀏覽器送出 HTTP 請求（Python 程式：ch6-2a.py），如下所示：

```
import requests  
  
URL = "https://www.momoshop.com.tw/search/"  
  
headers = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)'  
           'AppleWebKit/537.36 (KHTML, like Gecko)'  
           'Chrome/63.0.3239.132 Safari/537.36'}  
  
r = requests.get(URL+"searchShop.jsp?keyword=NBA", headers=headers)  
if r.status_code == requests.codes.ok:  
    r.encoding = "big5"  
    print(r.text)  
else:  
    print("HTTP 請求錯誤..." + url)
```

上述程式碼因為更改 HTTP 請求的標頭資訊，其執行結果就可以看到成功取回 HTML 標籤內容。

問題是 momo 網站的商品搜尋結果是 JavaScript 產生的網頁內容，當關掉 JavaScript 後，可以看到商品清單也不見了，如下所示：

◆ <https://www.momoshop.com.tw/search/searchShop.jsp?keyword=NBA>



因為網頁內容是 JavaScript 產生的內容，Python 程式需要改用 Selenium 向目標 URL 網址送出 HTTP 請求（Python 程式：ch6-2b.py），如下所示：

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

URL="https://www.momoshop.com.tw/search/"

driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
driver.implicitly_wait(10)
driver.get(URL+"searchShop.jsp?keyword=NBA")
print("-----")
print(driver.title)
html = driver.page_source
fp = open("NBA.html", "w", encoding="utf8")
fp.write(html)
print("寫入檔案 NBA.html...")
fp.close()
driver.quit()
```

上述程式碼使用 Selenium 送出 HTTP 請求，可以將回傳 HTML 標籤字串儲存成 NBA.html 網頁檔案，第 2273 列就是商品清單的 清單標籤，如下圖所示：

```

2269      <!-- 商品區 -->
2270      <div class="listArea">
2271          <div class="productLoadingTxt" style="display:none">商品載入中...<
2272          <div class="whiteBk" style="display:none"></div>
2273      <ul><li gcode="10041674" is18goods="0"><input type="hidden" name="
2274      </div>
2275  </div>

```

☆ 在多次 HTTP 請求之間加上延遲時間

一般來說，網路爬蟲很可能需要在很短時間內，針對同一個網站密集的送出 HTTP 請求，例如：在 1 秒內送出超過 10 次請求，為了避免被駭客攻擊，目前的 Web 網站大都有預防密集 HTTP 請求的機制。

換句話說，在爬蟲時應避免短時間密集送出 HTTP 請求，而是在每一次請求之間等待幾秒鐘（Python 程式：ch6-2c.py），如下所示：

```

import time
import requests

URL = "http://www.major-tests.com/word-lists/word-list-0{0}.html"

for i in range(1, 10):
    url = URL.format(i)
    r = requests.get(url)
    print(r.status_code)
    print("等待 5 秒鐘...")
    time.sleep(5)

```

上述程式碼匯入 time 模組，在 for/in 迴圈共送出 9 次 HTTP 請求，每次請求之間呼叫 time.sleep(5) 函數暫停幾秒鐘，以此例是參數的 5 秒，也就是每等 5 秒鐘才送出一個 HTTP 請求。

☆ 處理例外情況的 HTML 標籤

當分析 HTML 網頁找到目標 HTML 標籤後，撰寫 Python 爬蟲程式時需要注意一些例外情況來進行特別處理，否則在爬蟲時就有可能中斷在這些例外情況。例如：PTT BBS 的 NBA 版的 HTML 標籤，貼文的標題文字是 `<div class="title">` 下的 `<a>` 標籤，如下圖所示：

```
▼<div class="r-ent"> == $0
  ▶<div class="nrec">...</div>
  ▼<div class="title">
    <a href="/bbs/NBA/M.1655184013.A.21F.html">[花邊] JT成為NBA歷史上單季後賽失誤最
    多的球員</a>
  </div>
  ▶<div class="meta">...</div>
</div>
```

上述 `<div class="r-ent">` 標籤是一篇貼文，位在 `<div class="title">` 下的 `<a>` 標籤是 BBS 貼文的標題文字。如果是一篇已經刪除的貼文，如下圖所示：

```
▼<div class="r-ent"> == $0
  <div class="nrec"></div>
  <div class="title"> (本文已被刪除) [nwd4e9cd] </div>
  ▶<div class="meta">...</div>
</div>
```

上述 `<div class="title">` 標籤就只有文字內容，並沒有 `<a>` 標籤，這是貼文 HTML 標籤的例外情況。當發生這種情況時，我們有兩種處理方式，如下所示：

- ◆ 方法一：使用 `if` 條件判斷 `<div class="title">` 下是否有 `<a>` 標籤，沒有 `<a>` 標籤就跳過不處理，在 `ch6-2e.py` 程式是使用這種方法。
- ◆ 方法二：使用 BeautifulSoup 物件建立替代 `<a>` 標籤，如果沒有就使用替代標籤來代替，在本小節的 `ch6-2d.py` 程式是使用此方法。

Python 程式：ch6-2d.py 是擷取 PPT NBA 版的貼文，首先建立 DELETED 變數，使用 BeautifulSoup 物件建立 <a> 標籤，參數是標籤字串，如下所示：

```
import requests
from bs4 import BeautifulSoup

URL = "https://www.ptt.cc/bbs/NBA/index6503.html"
DELETED = BeautifulSoup("<a href='Deleted'>本文已刪除</a>", "lxml").a
```

上述程式碼建立 <a> 標籤的 BeautifulSoup 物件，最後的 .a 是取得此標籤物件，然後送出 HTTP 請求，如下所示：

```
r = requests.get(URL)
if r.status_code == requests.codes.ok:
    r.encoding = "utf8"
    soup = BeautifulSoup(r.text, "lxml")
    tag_divs = soup.find_all("div", class_="r-ent")
    for tag in tag_divs:
        tag_a = tag.find("a") or DELETED
        print(tag_a["href"])
        print(tag_a.text)
        print(tag.find("div", class_="author").string)
else:
    print("HTTP 請求錯誤..." + url)
```

上述程式碼使用 find_all() 函數找出所有貼文的 <div> 標籤後，使用 for/in 迴圈取出每一篇貼文的標題文字，即 <a> 標籤，如下所示：

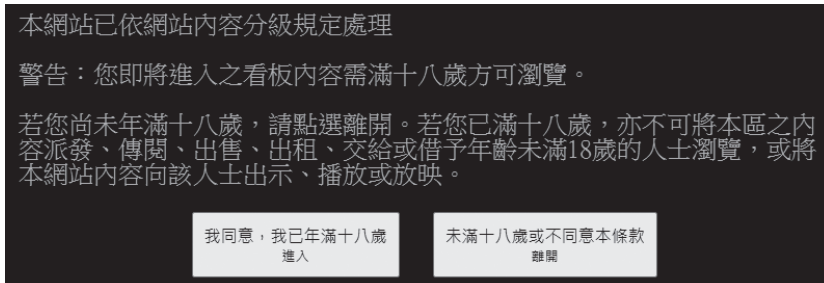
```
tag_a = tag.find("a") or DELETED
```

上述程式碼使用 find() 函數搜尋 <a> 標籤，沒有找到就指定成 DELETED 變數的 <a> 標籤物件，執行結果可以看到顯示「本文已刪除」，而這就是刪除的貼文，如下所示：

```
/bbs/NBA/M.1655178938.A.3E9.html
Re: [BOX ] Celtics 94:104 Warriors
longsea
/bbs/NBA/M.1655179347.A.3CE.html
[討論] Andrew Wiggins 圍巾總冠逐場數據
NukAnah
...
/bbs/NBA/M.1655179729.A.DDC.html
[討論] 總冠 G5
CleanSpurs
Deleted
本文已刪除
-
/bbs/NBA/M.1655180186.A.04C.html
[討論] 合約到期後勇士該續約 Draymond Green 嗎?
justgetup
...
```

☆ 網站內容分級規定

目前很多的網站內容都有分級規定，有些網站在進入前會詢問是否年滿 18 歲，例如：PTT BBS 的 Gossiping 版，如下圖所示：



上述圖例需按 **我同意，我已年滿十八歲 進入** 鈕才能進入網頁。因為 PTT BBS 是使用 Cookie 儲存是否年滿十八歲，所以，Python 程式需要在 requests 請求指定 Cookie，來跳過網站分級規定的畫面（Python 程式：ch6-2e.py），如下所示：

```

import requests
from bs4 import BeautifulSoup

URL = "https://www.ptt.cc/bbs/Gossiping/index.html"

r = requests.get(URL, cookies={"over18": "1"})
if r.status_code == requests.codes.ok:
    r.encoding = "utf8"
    soup = BeautifulSoup(r.text, "lxml")
    tag_divs = soup.find_all("div", class_="r-ent")
    for tag in tag_divs:
        if tag.find('a'): # 是否有 <a> 標籤
            tag_a = tag.find("a")
            print(tag_a["href"])
            print(tag_a.text)
            print(tag.find("div", class_="author").string)
    else:
        print("HTTP 請求錯誤..." + url)

```

上述 `request.get()` 函數的第 2 個參數指定 `cookies` 來跳過網站內容分級規定，`for/in` 迴圈使用 `if` 條件判斷是否找到 `<a>` 標籤，而不是使用 `ch6-2d.py` 的自訂標籤來處理例外情況。

☆ 建立爬蟲目標的 URL 網址

當爬蟲目標的 URL 網址不只一個，而是有很多個 URL 網址時，我們在爬蟲前就需要先建立這些 URL 網址，Python 程式 `ch6-1c.py` 是使用字串 `format()` 函數建立目標的多個 URL 網址。

另一種方式，因為 Python 的 `urllib.parse` 模組是用來處理 URL 網址，我們可以使用此模組的 `urljoin()` 函數來結合建立所需的 URL 網址（Python 程式：`ch6-2f.py`），如下所示：

```

from urllib.parse import urljoin

URL = "http://www.major-tests.com/word-lists/word-list-01.html"
PTT = "https://www.ptt.cc/bbs/movie/index.html"

catalog = ["movie", "NBA", "Gossiping"]

for i in range(1, 5):
    url = urljoin(URL, "word-list-0{0}.html".format(i))
    print(url)
print("-----")
for item in catalog:
    url = urljoin(PTT, "../{0}/index.html".format(item))
    print(url)

```

上述程式碼首先匯入 `urljoin()` 函數，第 1 個 `for/in` 迴圈呼叫 `urljoin()` 函數結合第 1 個參數的 URL 網址和第 2 個參數的檔名，可以建立 `word-list-01.html~word-list-04.html` 的 URL 網址，其執行結果如下所示：

```

http://www.major-tests.com/word-lists/world-list-01.html
http://www.major-tests.com/word-lists/world-list-02.html
http://www.major-tests.com/word-lists/world-list-03.html
http://www.major-tests.com/word-lists/world-list-04.html

```

在第 2 個 `for/in` 迴圈是建立 PPT 各版的 URL 網址，使用串列和「`../`」路徑來取代上一層目錄，其執行結果可以看到建立 `movie`、`NBA` 和 `Gossiping` 版的 URL 路徑，如下所示：

```

https://www.ptt.cc/bbs/movie/index.html
https://www.ptt.cc/bbs/NBA/index.html
https://www.ptt.cc/bbs/Gossiping/index.html

```


6-3 擷取多筆記錄和 HTML 表格資料

在 Python 網路爬蟲常常需要擷取 HTML 網頁的多筆記錄或表格資料，就是一種階層結果的資料擷取，我們可以先取出父標籤後，再一一取出子標籤的記錄資料。

6-3-1 擷取 HTML 網頁的多筆記錄資料（一）

多筆記錄資料的 HTML 標籤是一種階層結構，我們只需找到每筆記錄的父標籤，即 ``、`` 和 `<div>` 等父標籤後，就可以擷取之下多筆記錄，然後再從每一筆記錄再向下一層找出各欄位的標籤。例如：使用 BeautifulSoup 擷取網頁 `` 清單標籤資料的步驟，如下所示：

Step 1 先找出記錄的 `` 父標籤，其所有 `` 子標籤就是記錄集合。

Step 2 重複每筆記錄的 `` 標籤，找出每一筆記錄下一層的欄位標籤。

Yahoo!電影網頁可以查詢目前上映中的電影資料，每一部電影是一筆記錄，在同一頁面擁有多筆電影記錄，其 URL 網址如下所示：

◆ https://movies.yahoo.com.tw/movie_intheaters.html



☆ 分析 HTML 網頁資料

因為關閉 JavaScript 並不會影響目標資料，請使用 Chrome 開發人員工具找出目標資料，可以發現電影清單是 `` 清單標籤，如下所示：

```
<ul class="release_list">
  <li>
    <div class="release_foto"></div>
    <div class="release_info"></div>
  </li>
  ...
</ul>
```

上述外層 `` 標籤（class 屬性值 "release_list"）是多筆記錄的父標籤，每一個 `` 子標籤是一筆記錄，在之下有 2 個 `<div>` 標籤，第 1 個是劇照圖片；第 2 個是電影資訊。首先找出劇照圖片，如下圖所示：

```
▼<li>
  ▼<div class="release_foto">
    ::before
    ▼<a href="https://movies.yahoo.com.tw/movieinfo_main/%E4%B8%8F%E7%B8%85%E7%B4%80...7%95%8C-%E7%B5%B1%E9%9C%B8%E5%A4%A9%E4%B8%8B-jurassic-world-dominion-11809" class="gabtn" data-ga=["'上映中','上映中_上映中第1頁','侏羅紀世界：統霸天下']">
      <div class="type_label movie">院線電影</div>
       == $0
    </a>
  </div>
```

上述 `` 標籤的 data-src 屬性值就是劇照圖片的 URL 網址，如下表所示：

欄位	標籤與屬性
劇照圖片	<code></code> 的 data-src 屬性

然後展開 `<div class="release_info">` 的電影資訊，如下圖所示：

```

▼<div class="release_info">
  ▼<div class="release_info_text">
    ▶<div class="release_movie_name">...</div>
    <div class="release_drama_season"> </div>
    ▶<div class="release_movie_time">...</div>
    ▶<div class="release_text">...</div>
  </div>
  ▶<div class="release_btn_color_btnbox">...</div>
</div>

```

當我們進一步展開之下的 `<div>` 標籤，就可以一一找出中文片名、英文片名、期待度和上映日資料，如下表所示：

欄位	標籤與屬性
中文片名	<code><div></code> (class 屬性值 "release_movie_name") 下的 <code><a></code> 標籤
英文片名	<code><div></code> (class 屬性值 "release_movie_name") 下的 <code><div></code> (class 屬性值 "en") 下的 <code><a></code> 標籤
期待度	<code><div></code> (class 屬性值 "leveltext") 之下的 <code></code> 標籤
上映日	<code><div></code> (class 屬性值 "release_movie_time") 標籤

☆ 擷取 Yahoo!上映中的電影資料：ch6-3-1.py

因為 JavaScript 不會影響目標資料，Python 程式是使用 `requests` 模組的 `get()` 函數來送出 HTTP 請求，首先指定 URL 網址和 HTTP 標頭資訊，如下所示：

```

import requests
from bs4 import BeautifulSoup
import csv, re

URL = "https://movies.yahoo.com.tw/movie_intheaters.html"
headers = {"user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
           "AppleWebKit/537.36 (KHTML, like Gecko)"
           "Chrome/63.0.3239.132 Safari/537.36"}

```

然後是 3 個函數來處理日期、標籤內容和標籤屬性（避免沒有這些內容），在下方的第 1 個 `format_data()` 函數首先判斷標籤是否存在，不存在，就回傳 "N/A"，然後使用正規表達式取出上映日期字串中的日期資料，如下所示：

```

def format_date(date): # 取出上映日期
    if not date: return "N/A"
    pattern = '\d+~\d+~\d+'
    match = re.search(pattern, date.text)
    if match is None:
        return date.text
    else:
        return match.group(0)

def get_text(tag):
    if tag:
        return tag.text.strip()
    else:
        return "N/A"

def get_attrib(tag, attrib):
    if tag:
        return tag[attrib].strip()
    else:
        return "N/A"

```

上述 `get_text()` 和 `get_attrib()` 函數分別取出標籤的文字內容和屬性值，第 1 個參數是標籤物件，`get_attrib()` 函數的第 2 個參數是屬性名稱，if/else 條件判斷標籤物件是否存在，存在才取出標籤內容和屬性值。

在下方建立擷取資料的 `movies` 巢狀串列後，送出 GET 請求取回網頁內容，和使用 `BeautifulSoup` 剖析網頁，如下所示：

```

movies = [{"中文片名", "英文片名", "期待度", "海報圖片", "上映日"}]
r = requests.get(URL, headers=headers)
if r.status_code == requests.codes.ok:
    soup = BeautifulSoup(r.text, 'lxml')
    tag_ul = soup.find("ul", class_="release_list")
    rows = tag_ul.find_all("li")

```

上述 if/else 條件判斷請求是否成功，成功，就呼叫 find() 函數使用 class 屬性值 "release_list" 定位 標籤，然後呼叫 find_all() 函數取出之下的所有 子標籤。在下方使用 for/in 迴圈取得每一筆記錄的欄位資料，如下所示：

```
for row in rows:
    name_div = row.find("div",class_="release_movie_name")
    cht_n = get_text(name_div.find("a"))
    eng_n = get_text(name_div.find("div",class_="en").find("a"))
    expect = get_text(row.find("div",class_="leveltext").find("span"))
    photo_div = row.find("div",class_="release_foto")
    poster_url = get_attrib(photo_div.find("img"),"data-src")
    date = row.find('div',class_='release_movie_time')
    release_date = format_date(date)
    movie= [cht_n,eng_n,expect,poster_url,release_date]
    movies.append(movie)
```

上述 for/in 迴圈依序呼叫 get_text() 或 get_attrib() 函數來取得中文片名、英文片名、期待度、劇照圖片網址和上映日期，在建立成 movie 串列後，呼叫 append() 函數新增至 movies 巢狀串列。在下方的 with/as 程式區塊是將 movies 巢狀串列寫入 CSV 檔案 movies.csv，如下所示：

```
else:
    print("HTTP 請求錯誤...")

with open("movies.csv", "w+",newline="",encoding="utf-8") as fp:
    writer = csv.writer(fp)
    for item in movies:
        writer.writerow(item)
```

Python 程式的執行結果可以在 Python 程式相同目錄看到 movies.csv 檔案（因為有中文內容，請使用匯入方式匯入 Excel），如下圖所示：

	A	B	C	D	E
1	中文片名	英文片名	期待度	海報圖片	上映日
2	露草	Tsuyukusa	43%	https://movies.yahoo.com.tw/zt/w420/h/o/production/movies/March20	2022/6/10
3	我的夏日大戀習	One Summer Story	60%	https://movies.yahoo.com.tw/zt/w420/h/o/production/movies/May2022	2022/6/10
4	餘命10年	The Last 10 Years	98%	https://movies.yahoo.com.tw/zt/w420/h/o/production/movies/May2022	2022/6/10
5	黎巴嫩的天空	1982	100%	https://movies.yahoo.com.tw/zt/w420/h/o/production/movies/May2022	2022/6/10
6	熟女解放中	The Book of Delights	100%	https://movies.yahoo.com.tw/zt/w420/h/o/production/movies/May2022	2022/6/10
7	侏羅紀世界：統霸天下	Jurassic World Dominion	98%	https://movies.yahoo.com.tw/zt/w420/h/o/production/movies/April2022	2022/6/8
8	PINA	PINA	80%	https://movies.yahoo.com.tw/zt/w420/h/o/production/movies/May2022	2022/6/3
9	終極夜路	Gasoline Alley	80%	https://movies.yahoo.com.tw/zt/w420/h/o/production/movies/May2022	2022/6/2
10	進擊的地才	The Novice	75%	https://movies.yahoo.com.tw/zt/w420/h/o/production/movies/April2022	2022/6/2
11	們	Men	100%	https://movies.yahoo.com.tw/zt/w420/h/o/production/movies/April2022	2022/6/2

6-3-2 擷取 HTML 網頁的多筆記錄資料（二）

Ashion 範本商務網站是一個使用 JavaScript 程式碼產生網頁內容的範本購物網站，我們需啟用 JavaScript，即將 Quick JavaScript Switcher 切換成小綠點，才能夠成功顯示商品清單，每一個方框是一個商品資料，其 URL 網址如下所示：

◆ <https://fchart.github.io/Ashion/>



Fit micro corduroy shirt
★★★★★
\$ 59.0



Tropical Kimono
★★★★★
\$ 49.0 \$ 59.0



Contrasting sunglasses
★★★★★
\$ 59.0



Water resistant backpack
★★★★★
\$ 49.0 \$ 59.0

☆ 分析 HTML 網頁資料

因為關閉 JavaScript 會影響目標資料，請開啟 JavaScript 後，使用 Chrome 開發人員工具找出目標資料，可以發現商品資料是位在 <section> 標籤下的 <div> 巢狀標籤，如下所示：

```
<section class="product spad">
  <div class="product __ item">
  </div>
  ...
</section>
```

上述外層 `<section>` 標籤 (class 屬性值 "product spad") 是多筆記錄的父標籤，在之下有多層 `<div>` 標籤，我們可以找出每一個 `<div>` 子標籤 (class 屬性值 "product__item") 是一筆記錄，記錄的欄位資料如下表所示：

欄位	標籤與屬性
產品名稱	<code><h6></code> 下的 <code><a></code> 標籤
產品圖片	<code><div></code> (class 屬性值 "product__item__pic") 的 <code>data-setbg</code> 屬性值
價格	<code><div></code> (class 屬性值 "product__price") 標籤

☆ 擷取 Ashion 範本商務網站的商品資料：ch6-3-2.py

因為 JavaScript 會影響目標資料，所以 Python 程式是使用 Selenium 來送出 HTTP 請求，如下所示：

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup
import json

URL = "https://fchart.github.io/Ashion/"
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
driver.implicitly_wait(10)
driver.get(URL)

soup = BeautifulSoup(driver.page_source, "lxml")
```

上述程式碼送出 GET 請求取回網頁內容後，使用 BeautifulSoup 剖析網頁，即可在下方定位各記錄的 `<section>` 父標籤，如下所示：

```

sec = soup.find("section", class_="product spad")
items = sec.find_all("div", class_="product __item")
print(len(items))

```

上述 find() 函數使用 class 屬性值 "product spad" 定位 <section> 標籤，然後呼叫 find_all() 函數取出之下所有 <div> 子標籤，和顯示記錄數。在下方建立 products 空串列後，使用 for/in 迴圈取出每一筆記錄的欄位資料來建立成 Python 字典，如下所示：

```

products=[]
for item in items:
    tag = item.find("h6").find("a")
    title = tag.text.strip() if tag else "N/A"
    tag = item.find("div", class_="product __item __pic")
    img = tag["data-setbg"].strip() if tag else "N/A"
    tag = item.find("div", class_="product __price")
    price = tag.text.strip() if tag else "N/A"
    print(title)
    products.append({
        "title": title,
        "image": URL+img,
        "price": price
    })

```

上述 for/in 迴圈取出每筆記錄的 <div> 標籤後，依序找到之下的商品名稱、商品圖片和定價的標籤物件，並且改用單行選擇條件判斷如果物件存在，才取出標籤或屬性值，最後建立成字典且新增至 products 串列。在下方將 products 字典串列寫入 products.json 檔案，如下所示：

```

driver.quit()
with open("products.json", "w", encoding="utf-8") as fp: # 寫入 JSON 檔案
    json.dump(products,fp,indent=2,
        sort_keys=True,
        ensure_ascii=False)

```


Python 程式的執行結果可以在 Python 程式相同目錄看到 products.json 檔案，顯示共有 8 筆記錄和商品名稱，如下所示：

```
8
Buttons tweed blazer
Flowy striped skirt
Cotton T-Shirt
Slim striped pocket shirt
Fit micro corduroy shirt
Tropical Kimono
Contrasting sunglasses
Water resistant backpack
```

6-3-3 擷取 HTML 表格資料






在實務上，網路上的很多金融數據都是 HTML 表格資料，所以這一節我們特別針對 HTML 表格標籤來說明如何擷取此類資料。例如：使用 BeautifulSoup 擷取 <table> 標籤資料的步驟，如下所示：

Step 1 先找出記錄的 <table> 父標籤，其所有 <tr> 子標籤就是記錄集合。

Step 2 重複每筆記錄，找出每一筆記錄下一層的 <td> 和 <th> 欄位標籤。

在台灣銀行網站可以查詢即時匯率資料，這是使用 HTML 表格顯示的匯率資料，如下所示：

◆ <https://rate.bot.com.tw/xrt?Lang=zh-TW>

幣別	現金匯率		即期匯率		遠期匯率	歷史匯率
	本行買入	本行賣出	本行買入	本行賣出		
 美金 (USD)	29.34	30.01	29.69	29.79	查詢	查詢
 港幣 (HKD)	3.632	3.836	3.758	3.818	查詢	查詢
 英鎊 (GBP)	34.59	36.71	35.6	36	查詢	查詢
 澳幣 (AUD)	20.16	20.94	20.45	20.65	查詢	查詢
 加拿大幣 (CAD)	22.45	23.36	22.85	23.05	查詢	查詢
 新加坡幣 (SGD)	20.81	21.72	21.3	21.48	查詢	查詢
 瑞士法郎 (CHF)	28.97	30.17	29.65	29.9	查詢	查詢
 日圓 (JPY)	0.2117	0.2245	0.219	0.223	查詢	查詢
 南非幣 (ZAR)	-	-	1.821	1.901	查詢	查詢

☆ 分析 HTML 網頁資料

因為關閉 JavaScript 並不會影響目標資料，請使用 Chrome 開發人員工具找出目標資料，可以看出匯率資料是 `<table>` 表格標籤，如下所示：

```
<table title="牌告匯率" class="table .." summary="此表格是牌告匯率..">
...
</table>
```

我們可以直接使用 Chrome 開發人員工具取得定位 `<table>` 標籤的 CSS 選擇器，如下所示：

```
#iellandabove > div > table
```

然後從 `<table>` 標籤開始取得之下所有 `<tr>` 子標籤的記錄，即可取出所有欄位的 `<td>` 和 `<th>` 子標籤。

☆ 擷取台灣銀行的即時匯率資料：ch6-3-3.py

因為 JavaScript 不會影響目標資料，Python 程式是使用 `requests` 模組的 `get()` 函數來送出 HTTP 請求，如下所示：

```
import requests
from bs4 import BeautifulSoup
import csv

url = "https://rate.bot.com.tw/xrt?Lang=zh-TW"
csvfile = "xrt.csv"
r = requests.get(url)
r.encoding = "utf8"
soup = BeautifulSoup(r.text, "lxml")
```

上述程式碼送出 GET 請求取回網頁內容後，使用 `BeautifulSoup` 剖析網頁，即可在下方定位各記錄的 `<table>` 父標籤，如下所示：

```
tag_table = soup.select_one("#iellandabove > div > table")
rows = tag_table.find_all("tr")
```

上述 `select_one()` 函數使用 CSS 選擇器定位 `<table>` 標籤後，即可呼叫 `find_all()` 函數取出之下所有 `<tr>` 子標籤。在下方 `for/in` 迴圈取出每一筆 `<tr>` 記錄標籤的欄位資料來存入 CSV 檔案，如下所示：

```
with open(csvfile,'w+',newline='',encoding="big5") as fp:
    writer = csv.writer(fp)
    for row in rows:
        lst = []
        for cell in row.find_all(["td", "th"]):
            lst.append(cell.text.replace("\n","").
                        replace("\r","").
                        strip())
        writer.writerow(lst)
```

上述程式碼呼叫 `open()` 函數開啟檔案後，使用第 1 層 `for/in` 迴圈取出每筆記錄的 `<tr>` 標籤後，搜尋之下所有 `<td>` 和 `<th>` 標籤的欄位，然後使用第 2 層 `for/in` 迴圈取出各儲存格的資料後，寫入一筆記錄至 CSV 檔案：`xrt.csv`，其執行結果可以看到 CSV 檔案內容（Excel 請使用匯入方式開啟），如下圖所示：

	A	B	C	D	E	F	G
1	幣別	幣別	現金匯率		即期匯率		遠期匯率
2		本行買入	本行賣出	本行買入	本行賣出		本行買入
3	美金 (USD)	29.34	30.01	29.69	29.79	查詢	查詢
4	港幣 (HKD)	3.632	3.836	3.758	3.818	查詢	查詢
5	英鎊 (GBP)	34.59	36.71	35.6	36	查詢	查詢
6	澳幣 (AUD)	20.16	20.94	20.45	20.65	查詢	查詢
7	加拿大幣 (CAD)	22.45	23.36	22.85	23.05	查詢	查詢
8	新加坡幣 (SGD)	20.81	21.72	21.3	21.48	查詢	查詢
9	瑞士法郎 (CHF)	28.97	30.17	29.65	29.9	查詢	查詢
10	日圓 (JPY)	0.2117	0.2245	0.219	0.223	查詢	查詢
11	南非幣 (ZAR)	-	-	1.821	1.901	查詢	查詢
12	瑞典幣 (SEK)	2.55	3.07	2.89	2.99	查詢	查詢

上述圖例是從網頁擷取的表格資料，這些資料有些混亂和多了很多欄位，我們可以進一步使用 Python 字串函數和 Pandas 執行資料清理。

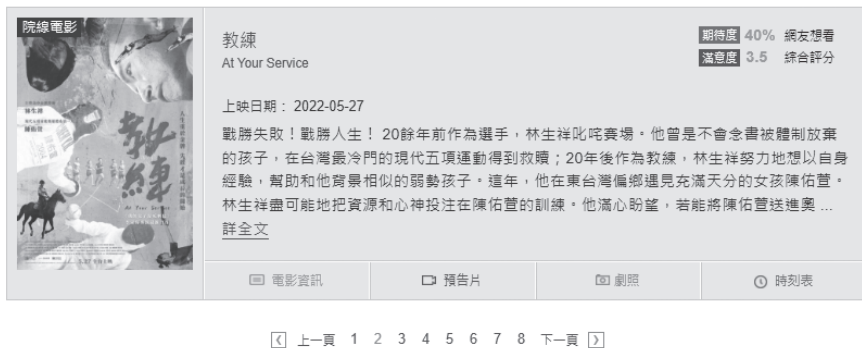
6-4 擷取多頁面的分頁記錄資料

在第 6-3 節是擷取單一頁面的多筆記錄資料，如果資料不只多筆，還有很多分頁時，我們需要切換分頁來擷取多頁面的分頁記錄資料。

6-4-1 使用 URL 網址參數擷取多頁面的資料

在第 6-3-1 節 Yahoo! 目前上映中的電影資料是多分頁網頁的多筆記錄，URL 網址的 page 參數是分頁數，2 是第 2 頁，如下所示：

◆ https://movies.yahoo.com.tw/movie_intheaters.html/?page=2



上述圖例在網頁最後是切換分頁的頁碼和上一頁/下一頁超連結。

☆ 分析 HTML 網頁資料

Yahoo! 電影資料的 HTML 網頁分析和第 6-3-1 節相同。在電影清單最後可以看到下一頁超連結，其 HTML 標籤是 `` 標籤下的 `<a>` 標籤，如下所示：

```
<li class="nexttxt">
<a href="http://movies.yahoo.com.tw/movie_intheaters.html?page=3"
rel="next">下一頁</a>
</li>
```

上述 `<a>` 標籤的 `href` 屬性值是下一頁的 URL 網址。如果是最後一頁，因為已經沒有下一頁，所以不是 `<a>`；而是 `` 標籤，同時在 `` 標籤的 `class` 屬性值多了 `"disabled"`，如下所示：

```
<li class="nexttxt disabled"><span>下一頁</span></li>
```

☆ 使用 URL 參數擷取 Yahoo!上映中的電影資料：ch6-4-1.py

Python 程式是直接修改 `ch6-3-1.py`，首先在 URL 網址指定頁碼參數 `page`，如下所示：

```
URL = "https://movies.yahoo.com.tw/movie_intheaters.html/?page={0}"
```

在建立擷取資料的 `all_movies` 巢狀串列後，使用 `for/in` 迴圈執行 10 次，共擷取 10 頁分頁，這是使用 `URL.format()` 函數建立各分頁的 URL 網址，如下所示：

```
all_movies = [["中文片名", "英文片名", "期待度", "海報圖片", "上映日"]]
for page in range(1, 11):
    url = URL.format(page)
    print("抓取: 第" + str(page) + "頁 網路資料中...")
    r = requests.get(url, headers=headers)
    if r.status_code == requests.codes.ok:
        soup = BeautifulSoup(r.text, 'lxml')
```

上述程式碼送出 GET 請求取回網頁內容後，使用 `BeautifulSoup` 剖析網頁，即可在下方定位各記錄的 `` 父標籤，和取得所有 `` 子標籤，然後在第 2 層 `for/in` 迴圈取得每一筆記錄的欄位資料，如下所示：

```
movies = []
tag_ul = soup.find("ul", class_="release_list")
rows = tag_ul.find_all("li")
for row in rows:
    name_div = row.find("div", class_="release_movie_name")
    cht_n = get_text(name_div.find("a"))
    eng_n = get_text(name_div.find("div", class_="en").find("a"))
    expect = get_text(row.find("div", class_="leveltext").find("span"))
```



```

photo_div = row.find("div",class_="release_foto")
poster_url = get_attrib(photo_div.find("img"),"data-src")
date = row.find('div',class_='release_movie_time')
release_date = format_date(date)
movie= [cht_n,eng_n,expect,poster_url,release_date]
movies.append(movie)
all_movies = all_movies + movies

```

上述 for/in 迴圈依序取得中文片名、英文片名、期待度、劇照圖片網址和上映日期，在建立 movie 串列後，使用「+」加法運算子新增至 all_movies 巢狀串列。在下方 if 條件檢查是否有下一頁，以便判斷是否已經擷取完所有分頁（因為可能不到 10 頁），如下所示：

```

if soup.find("li", class_="nexttxt disabled"):
    break    # 已經沒有下一頁
print("等待 5 秒鐘...")
time.sleep(5)
else:
    print("HTTP 請求錯誤...")

```

上述程式碼使用 time.sleep() 函數等待 5 秒鐘後，就可以繼續執行迴圈擷取下一分頁的電影清單，最後將 all_movies 巢狀串列寫入 CSV 檔案 all_movies.csv。

Python 程式的執行結果顯示正在擷取各分頁電影資料的訊息文字，在完成後，可以在 Python 程式相同目錄看到 all_movies.csv 檔案，如下所示：

```

抓取：第 1 頁 網路資料中...
等待 5 秒鐘...
抓取：第 2 頁 網路資料中...
等待 5 秒鐘...
...
抓取：第 7 頁 網路資料中...
等待 5 秒鐘...
抓取：第 8 頁 網路資料中...

```

6-4-2 使用下一頁按鈕擷取多頁面的資料

Yahoo! 電影目前上映中的電影清單最後可以看到下一頁超連結按鈕，Python 程式除了使用 URL 參數 page 擷取分頁資料外，還可以找出下一頁按鈕的下一頁超連結來取得下一頁分頁的 URL 網址。

因為 ch6-4-1.py 和 ch6-4-2.py 的差異不大，筆者只準備說明其差異處，首先初始 URL 網址沒有參數，就是第 1 頁 page=1，如下所示：

```
URL = "https://movies.yahoo.com.tw/movie_intheaters.html/?page=1"
```

原來 for/in 迴圈是固定擷取 10 頁資料，改成 while 無窮迴圈來擷取分頁資料（結束條件是沒有下一頁超連結），如下所示：

```
...
page = 1
while True:
    print("抓取: 第" + str(page) + "頁 網路資料中...")
    page = page + 1
    r = requests.get(URL, headers=headers)
    if r.status_code == requests.codes.ok:
        ...
```

上述 page 變數記錄目前擷取的分頁數，擷取記錄和欄位部分和 ch6-4-1.py 相同，只有在最後新增擷取下一頁的 URL 網址，第 1 個 if 條件判斷是否是最後 1 頁，如果是，就跳出 while 無窮迴圈，如下所示：

```
...
if soup.find("li", class_="nexttxt disabled"):
    break # 已經沒有下一頁
nextPage = soup.find("li", class_="nexttxt")
if nextPage:
    URL = nextPage.find("a")["href"]
    print("等待 5 秒鐘...")
    time.sleep(5)
```



```

else:
    break
...

```

上述程式碼使用 `find()` 函數定位下一頁的 `` 標籤後，`if/else` 條件判斷此標籤是否存在，存在，就取得之 `<a>` 標籤的 `href` 屬性值，現在，URL 變數值就成為下一頁的 URL 網址，可以繼續擷取下一頁分頁的電影資料。

6-4-3 使用 Selenium 擷取下一頁表格資料

NBA 商品資料是 JavaScript 程式碼產生的網頁內容，這是使用分頁表格顯示的多筆商品資料，其 URL 網址如下所示：

◆ https://fchart.github.io/ML/nba_items.html

NBA 商品資料		
商品編號	商品名稱	價格
1	【NBA】NIKE 青年版 塞爾提克 Kyrie Irving 厄文 城市版球衣 青年球衣 運動背心 籃球球衣(NBA球衣)	1,190
2	【NBA】NIKE 籃網隊 Jeremy Lin DRY TEE 短袖上衣 短袖T恤 運動上衣 運動服 短T 健身 慢跑(NBA上衣)	690
3	【NBA】NIKE 青年版 公鹿隊 字母哥 Antetokounmpo 運動短T 迅速排汗 健身 基本款(Youth)	590
4	【NBA】NIKE 波士頓 塞爾提克隊 Kyrie Irving 球衣 厄文 運動背心 籃球球衣 球迷版(064403-103)	2,680
5	【NBA】NIKE 休士頓 火箭隊 Chris Paul 球衣 CP3 運動背心 籃球球衣(NBA球衣)	1,580
6	【NBA】NIKE 青年版 球衣 火箭隊 PAUL CP3 保羅船長 籃球球衣 籃球背心 球迷版(Youth)	1,071
7	【NBA】NIKE Westbrook 雷霆隊 DRY 快速排汗 短袖上衣 短袖T恤 運動上衣 運動服 短T 健身 慢跑(NBA上衣)	990
8	【NBA】NIKE 青年版 勇士隊 短袖上衣 DRI-FIT 快速排汗 短袖T恤 健身 慢跑(WZ2B711D1-WARRIORS)	612
9	【NBA】NIKE 休士頓 火箭隊 James Harden 球衣 大鬍子 哈登 運動背心 籃球球衣 球迷版(NBA球衣)	1,390
10	【NBA】NIKE U NK ELT CREW 籃球運動襪 blue(NBA籃球襪)	349

1
2
3
4
5
下一頁 >
最後一頁 »

位在上述分頁 HTML 表格下方有下一頁鈕，按此按鈕，可以切換下一頁分頁的 HTML 表格，因為這是 JavaScript 按鈕，並無法取得下一頁的 URL 網址，Python 程式只能使用 Selenium 模擬按下一頁鈕操作來切換分頁。

☆ 分析 HTML 網頁資料

因為關閉 JavaScript 會影響目標資料，請開啟 JavaScript 後，使用 Chrome 開發人員工具找出目標資料，可以看出商品資料是 `<table>` 表格標籤，如下所示：

```
<table class="table table-dark" id="our-table">
...
</table>
```

我們可以使用 Chrome 開發人員工具取得定位 `<table>` 標籤的 CSS 選擇器，如下所示：

```
#our-table
```

然後從 `<table>` 標籤開始取得之下所有 `<tr>` 子標籤的記錄，即可取出所有欄位的 `<td>` 和 `<th>` 子標籤。接著使用開發人員工具找出下一頁鈕是 `<button>` 標籤，擁有 `class` 屬性值 `"nextbtn"`。

☆ 使用 Selenium 擷取下一頁的表格資料：ch6-4-3.py

因為 JavaScript 會影響目標資料，所以 Python 程式是使用 Selenium 來送出 HTTP 請求，如下所示：

```
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup
import time, csv

URL="https://fchart.github.io/ML/nba_items.html"

driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
driver.implicitly_wait(10)
driver.get(URL)
pages_remaining = True
page_num = 1
while pages_remaining:
    soup = BeautifulSoup(driver.page_source, "lxml")
```

上述 `pages_remaining` 變數判斷是否還有下一頁，`page_num` 是目前的頁碼，然後使用 `while` 迴圈取出全部分頁的 HTML 表格資料，首先建立 BeautifulSoup 物件剖析 HTML 標籤字串。在下方呼叫 `select_one()` 函數取得 `<table>` 標籤，然後取得所有表格列的 `<tr>` 標籤，如下所示：

```
tag_table = soup.select_one("#our-table")
rows = tag_table.find_all("tr")

csvfile = "NBA_Products" + str(page_num) + ".csv"

with open(csvfile, 'w+', newline='', encoding="utf8") as fp:

    writer = csv.writer(fp)
```

上述程式碼建立各分頁的 CSV 檔名後，開啟 CSV 檔案來寫入資料。在下方第一層 `for/in` 迴圈取出所有標題列和資料列，第 2 層 `for/in` 迴圈取得所有儲存格，和將儲存格資料新增至 `lst` 串列，如下所示：

```
for row in rows:

    lst = []

    for cell in row.find_all(["td", "th"]):

        lst.append(cell.text.replace("\n", "").

                    replace("\r", "").

                    strip())

    writer.writerow(lst)

print("儲存頁面:", page_num)
```

上述 `writer.writerow()` 函數將此列資料串列寫入 CSV 檔案。在下方 `try/catch` 例外處理使用 Selenium 模擬按下一頁鈕操作，和處理沒有找到分頁 `<a>` 標籤時的例外情況，如下所示：

```
try:

    next_link = driver.find_element(By.CLASS_NAME, "nextbtn")

    if next_link:

        next_link.click()

        time.sleep(5)

        page_num = page_num + 1

    else:
```



```

        pages_remaining = False
    except Exception:
        pages_remaining = False
driver.close()

```

上述 `find_element()` 函數取得按鈕的標籤物件（沒有找到即丟出例外）後，`if/else` 條件判決是否有此物件，如果有，就呼叫 `click()` 函數模擬按下按鈕，然後等待 5 秒來載入網頁後，即可擷取下一頁 HTML 表格資料。

Python 程式的執行結果除了顯示目前截取和儲存的頁面編號外，可以在 Python 程式相同目錄看到各分頁 `NBA_Products1~17.csv` 檔案。

6-5

實作案例：majortests.com 的單字清單

majortests.com 網站 (<https://www.majortests.com/>) 是留學考試的模擬測驗和教學資源網站，如下圖所示：

The screenshot shows the homepage of **majortests.com**. At the top, there is a navigation menu with links: SAT, GRE, GMAT, Word Lists, MAT, Essays, and Topics. The main heading is "Practice Tests and Resources for High School, College and Graduate Tests". Below this, there are four main sections:

- SAT ***: Over 50 practice tests covering these SAT test sections:
 - [SAT Math](#)
 - [SAT Critical Reading](#)
 - [SAT Writing](#)
 Get Started >
- GRE ***: Over 50 practice tests covering these GRE test sections:
 - [GRE Math](#)
 - [GRE Verbal](#)
 - [GRE Essays](#)
 - [GRE Vocabulary](#)
 Get Started >
- GMAT ***: Over 30 practice tests covering these GMAT test sections:
 - [GMAT Verbal](#)
 - [GMAT Math](#)
 - [GMAT Essay](#)
 Get Started >
- Miller Analogies Test ***: A separate section for the Miller Analogies Test.

At the bottom right, there is a section for **Word Lists**, which includes "1500 essential words to build the vocabulary you need".

上述網頁游標所在的方框是英文 1500 個基本單字，我們準備抓取此 URL 網址的英文單字清單。

☆ 步驟一：識別出目標的 URL 網址

網路爬蟲的第一步是識別出目標的單一 URL 網址，或多個 URL 網址清單，以網站 `majortests.com` 基本 1500 單字來說，前 9 頁的 URL 網址如下所示：

```
http://www.majortests.com/word-lists/word-list-01.html
http://www.majortests.com/word-lists/word-list-02.html
http://www.majortests.com/word-lists/word-list-03.html
...
http://www.majortests.com/word-lists/word-list-09.html
```

上述 URL 網址清單的 HTML 網頁檔名只有最後 1 個數字不同，我們可以建立函數來產生這一系列 URL 清單，如下所示：

```
URL = "http://www.majortests.com/word-lists/word-list-0{0}.html"

def generate_urls(url, start_page, end_page):
    urls = []
    for page in range(start_page, end_page+1):
        urls.append(url.format(page))
    return urls
```

上述 URL 變數是 URL 網址的範本字串，「{0}」符號是準備使用 `format()` 函數來替代的數字，`generate_urls()` 函數的第 1 個參數是 URL 網址的範本字串，第 2 個參數是起始數字，第 3 個是結束數字，使用 `for/in` 迴圈產生和回傳 URL 網址清單。

☆ 步驟二：送出 HTTP 請求取得網路資源

在識別出目標的 URL 網址後，我們就可以使用 Quick JavaScript Switcher 擴充功能來進行判斷，因為目錄資料並不是 JavaScript 產生，可以使用 `requests` 送出 HTTP 請求，如下所示：

```
def get_resource(url):
    headers = {"user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
               "AppleWebKit/537.36 (KHTML, like Gecko)"
               "Chrome/63.0.3239.132 Safari/537.36"}
    return requests.get(url, headers=headers)
```

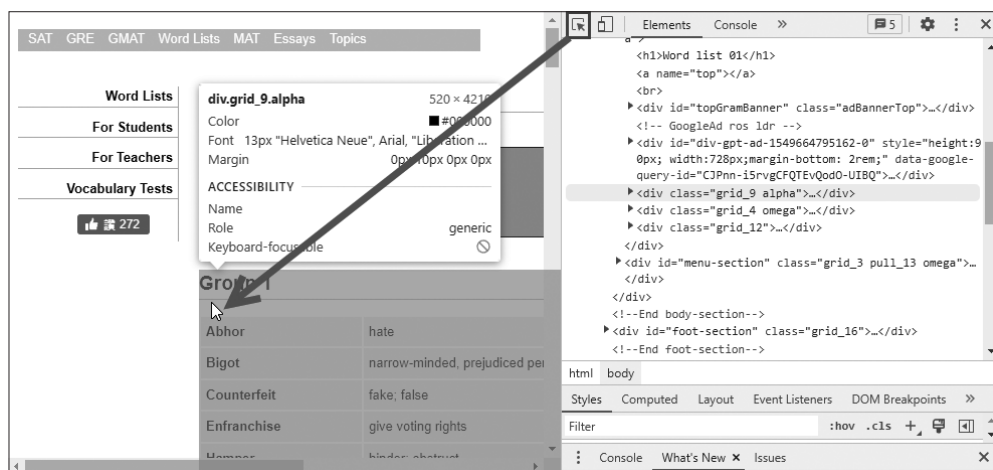
上述 `get_resource()` 函數使用自訂標頭和參數的 URL 網址來送出 HTTP 請求，可以取得回應的 HTML 網頁。

☆ 步驟三：分析 HTML 網頁找出爬蟲的目標標籤

在實際剖析 HTML 網頁前，我們需要先分析 HTML 網頁來找出爬蟲的目標標籤，例如：第 1 頁的單字清單，如下所示：

◆ <http://www.majortests.com/word-lists/word-list-01.html>

請啟動 Chrome 瀏覽上述網址後，按 **F12** 鍵開啟開發人員工具，如下圖所示：



上述每一頁 HTML 網頁是 Group1~10 個群組的表格單字，請點選上方工具列的第 1 個圖示，然後移至單字表格，可以看到是 `<div>` 標籤，如下所示：

```

<div class="grid_9 alpha">
<h3>Group 1</h3><a name="1"></a>
<table class="wordlist">
<tbody>
<tr><th>Abhor</th><td>hate</td></tr>
<tr><th>Bigot</th><td>narrow-minded, prejudiced person</td></tr>
...
<tr><th>Remuneration</th><td>payment for work done</td></tr>
<tr><th>Talisman</th><td>lucky charm</td></tr>
</tbody>
</table>
.....

```

在分析上述 HTML 標籤後，我們可以得到分析結果如下所示：

- ◆ 每一頁的單字表格都是 <div> 標籤的子標籤，因為有 10 個群組，所以共有 10 個 <table> 子標籤。
- ◆ 每一個 <table> 表格標籤是一個群組的單字，class 屬性都是 "wordlist"。
- ◆ 每一個 <tr> 標籤是一個單字，<th> 是單字；<td> 是單字說明。

☆ 步驟四：使用爬蟲工具剖析 HTML 網頁

在分析找出爬蟲的目標 HTML 標籤後，我們就可以使用 BeautifulSoup 來剖析 HTML 網頁，如下所示：

```

from bs4 import BeautifulSoup

def parse_html(html_str):
    return BeautifulSoup(html_str, "lxml")

```

上述 parse_html() 函數使用 BeautifulSoup 剖析參數的 HTML 字串，可以回傳 BeautifulSoup 物件。

☆ 步驟五：取出所需的資料

現在，我們可以建立爬蟲函數來進行 URL 網址清單的資料擷取，一一取出 HTML 網頁表格中的英文單字。

Python 函數：web_scraping_bot()

web_scraping_bot() 函數是使用 for/in 迴圈擷取 URL 網址清單中的每一個 URL 網址，如下所示：

```
import time

def web_scraping_bot(urls):
    eng_words = []

    for url in urls:
        file = url.split("/")[1]
        print("抓取: " + file + " 網路資料中...")
        r = get_resource(url)
        if r.status_code == requests.codes.ok:
            soup = parse_html(r.text)
            words = get_words(soup, file)
            eng_words = eng_words + words
            print("等待5秒鐘...")
            time.sleep(5)
        else:
            print("HTTP請求錯誤...")

    return eng_words
```

上述 eng_words 變數儲存擷取的單字清單，for/in 迴圈一一取出 URL 網址來呼叫 get_resource() 函數送出 HTTP 請求，if/else 條件判斷是否請求成功，如果成功，就呼叫 parse_html() 函數剖析 HTML 標籤字串，使用 get_words() 函數取出單字，和將取出單字新增至 eng_words 清單，在等待 5 秒鐘後，再送出下一個 URL 網址的 HTTP 請求。

Python 函數：get_words()

函數 `get_words()` 共有 2 個參數，第 1 個參數是 BeautifulSoup 物件，第 2 個參數是 HTML 網頁的檔案名稱，如下所示：

```
def get_words(soup, file):
    words = []
    count = 0

    for wordlist_table in soup.find_all(class_="wordlist"):
        count += 1
        for word_entry in wordlist_table.find_all("tr"):
            new_word = []
            new_word.append(file)
            new_word.append(str(count))
            new_word.append(word_entry.th.text)
            new_word.append(word_entry.td.text)
            words.append(new_word)

    return words
```

上述巢狀迴圈的外層 `for/in` 迴圈找出所有 `class` 屬性值是 "wordlist" 的 `<table>` 標籤，`count` 是群組計數，內層 `for/in` 迴圈找出所有 `<tr>` 標籤，每一個單字的資料依序是檔名、群組編號、單字 (`<th>`) 和說明 (`<td>`)，最後回傳每一頁 HTML 網頁檔案擷取的單字清單。

☆ 步驟六：儲存取出的資料

函數 `web_scraping_bot()` 可以取出所有 URL 網址清單中的單字清單，在完成後，呼叫 `save_to_csv()` 函數儲存成 CSV 檔案，如下所示：

```
import csv

def save_to_csv(words, file):
    with open(file, "w+", newline="", encoding="utf-8") as fp:
        writer = csv.writer(fp)
        for word in words:
            writer.writerow(word)
```


上述程式碼開啟 utf-8 編碼的文字檔案，然後將取的單字清單寫入 CSV 檔案。

☆ 步驟七：建立主程式執行網路爬蟲

最後，我們可以建立主程式執行上述函數來執行爬蟲作業，如下所示：

```
if __name__ == "__main__":
    urls = generate_urls(URL, 1, 9)
    # print(urls)
    eng_words = web_scraping_bot(urls)
    for item in eng_words:
        print(item)
    save_to_csv(eng_words, "words.csv")
```



請注意！當 Python 直譯器執行 Python 程式的主程式時，Python 程式需要使用 if 條件判斷 `__name__` 特殊變數值是否是 `"__main__"`，如果是，位在此 if 程式區塊的程式碼就是主程式執行的 Python 程式碼。

在主程式首先呼叫 `generate_urls()` 函數建立 URL 網址清單後，呼叫 `web_scraping_bot()` 抓取英文單字，成功後，使用 `for/in` 迴圈顯示英文單字的清單，最後呼叫 `save_to_csv()` 函數儲存成 CSV 檔案 `words.csv`。

完整 Python 程式：`/ch6-5/word_list_crawler.py` 的執行結果，可以看到每間隔 5 秒送出 1 次 HTTP 請求，在完成後顯示取得的英文單字清單，我們可以使用 Excel 檢視擷取的英文單字清單，如下圖所示：

	A	B	C	D
4	word-list-01.html	1	Enfranchise	give voting rights
5	word-list-01.html	1	Hamper	hinder; obstruct
6	word-list-01.html	1	Kindle	to start a fire
7	word-list-01.html	1	Noxious	harmful; poisonous; lethal
8	word-list-01.html	1	Placid	calm; peaceful
9	word-list-01.html	1	Remuneration	payment for work done
10	word-list-01.html	1	Talisman	lucky charm
11	word-list-01.html	2	Abrasive	rough; coarse; harsh
12	word-list-01.html	2	Bilk	cheat; defraud
13	word-list-01.html	2	Covert	hidden; undercover
14	word-list-01.html	2	Engender	cause
15	word-list-01.html	2	Hangar	storage area (like garage) for a plane
16	word-list-01.html	2	Knotty	complex; difficult to solve
17	word-list-01.html	2	Nuance	something subtle; a fine shade of meaning

6-6

實作案例：批踢踢 PTT BBS 熱門文章

批踢踢 PTT BBS 是國內著名的 BBS 討論空間，區分成多個看板的討論區，例如：NBA 看板 <https://www.ptt.cc/bbs/NBA/index.html>，如下圖所示：



上述網頁顯示貼文清單，我們準備建立 Python 爬蟲程式取得特定看板今天所發文章的相關資訊，和顯示熱門文章。

☆ 步驟一：識別出目標的URL網址

批踢踢 PTT BBS 的網址是固定開頭，在最後使用 index.html 結尾，不過，不同看板的路徑並不相同，如下所示：

```
URL = "https://www.ptt.cc"
# TOPIC = "Gossiping"
TOPIC = "NBA"
```

上述 URL 變數是批踢踢 PTT BBS 的基底網址，TOPIC 是看板名稱，我們可以輕鬆組合出特定看板的 URL 網址，如下所示：

```
url = URL + "/bbs/" + TOPIC + "/index.html"
```

例如：NBA 和 Gossiping 看板的網址，如下所示：

```
https://www.ptt.cc/bbs/NBA/index.html
```

```
https://www.ptt.cc/bbs/Gossiping/index.html
```

☆ 步驟二：送出 HTTP 請求取得網路資源

在識別出目標的 URL 網址後，我們可以使用 Quick JavaScript Switcher 擴充功能來進行判斷，因為目錄資料並不是 JavaScript 產生，可以使用 requests 送出 HTTP 請求，如下所示：

```
def get_resource(url):  
    headers = {"user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"  
              "AppleWebKit/537.36 (KHTML, like Gecko)"  
              "Chrome/63.0.3239.132 Safari/537.36"}  
    return requests.get(url, headers=headers, cookies={"over18": "1"})
```

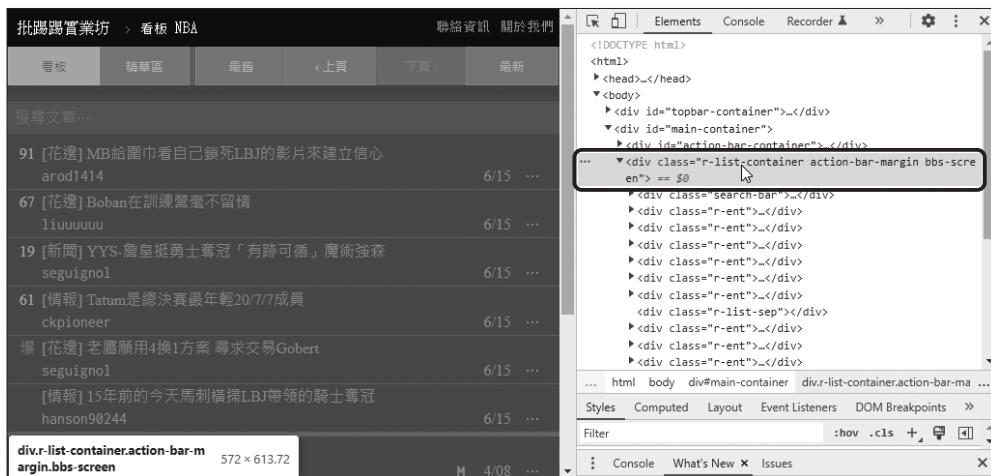
上述 get_resource() 函數使用自訂標頭、內容分級的 Cookie 和參數的 URL 網址來送出 HTTP 請求，可以取得網路資源的 HTML 網頁。

☆ 步驟三：分析 HTML 網頁找出爬蟲的目標標籤

在實際剖析 HTML 網頁前，我們需要先分析 HTML 網頁找出爬蟲的目標標籤，例如：NBA 看板，如下所示：

◆ <https://www.ptt.cc/bbs/NBA/index.html>

請啟動 Chrome 瀏覽上述網址後，按 **F12** 鍵開啟開發人員工具，如下圖所示：



上述 HTML 網頁是文章清單，請點選上方工具列的第 1 個圖示，然後移至文章標題清單，可以看到 `<div>` 標籤下是多個 `<div class="r-ent">` 標籤，如下所示：

```

<div class="r-list-container" ...>
  <div class="r-ent">...</div>
  <div class="r-ent">...</div>
  <div class="r-ent">...</div>
  .....
</div>
.....

```

上述每一個 `<div class="r-ent">` 標籤是一篇文章的標題文字，其標籤內容如下所示：

```

<div class="r-ent">
  <div class="nrec"><span class="hl f3">45</span></div>
  <div class="mark"></div>
  <div class="title">
    <a href="/bbs/NBA/M.1517280599.A.598.html">...</a>
  </div>
  <div class="meta">

```

```

<div class="date"> 1/30</div>

<div class="author">Rambo</div>

</div>

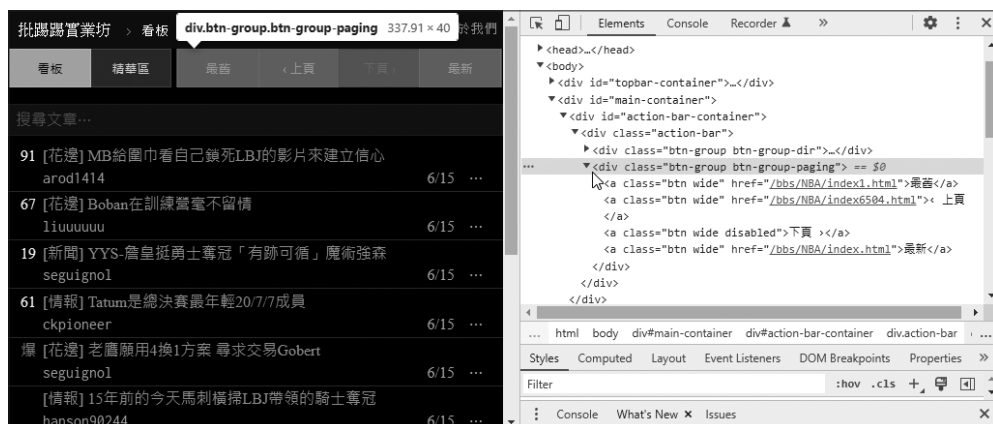
</div>

```

在分析上述文章標題的 HTML 標籤後，我們可以得到分析結果如下所示：

- ◆ 看板清單的每一篇文章標題是一個 `<div class="r-ent">` 標籤，在其 `<div>` 子標籤是此文章的相關資訊。
- ◆ `<div class="nrec">` 子標籤是推文數，位在 `` 子標籤的數值，如果推文太多，其值可能是'爆'。
- ◆ `<div class="title">` 子標籤是文章標題，在 `<a>` 子標籤是標題文字和連接此篇文章的 URL 網址。
- ◆ `<div class="meta">` 子標籤是日期和作者資訊，分別位在 `<div class="date">` 和 `<div class="author">` 子標籤。

因為批踢踢 PTT BBS 的看板是分頁顯示文章標題清單，在上一頁也有可能今天的文章，我們需要找出上一頁的超連結來繼續向上一頁擷取文章資料，上一頁按鈕是位在上方工具列，如下圖所示：



上述工具列按鈕是 `<div>` 標籤，如下所示：

```
<div class="btn-group btn-group-paging">
  <a class="btn wide" href="/bbs/NBA/index1.html">最舊</a>
  <a class="btn wide" href="/bbs/NBA/index6504.html">< 上頁</a>
  <a class="btn wide disabled">下頁 ></a>
  <a class="btn wide" href="/bbs/NBA/index.html">最新</a>
</div>
```

上述 `class` 屬性值 `"btn-group btn-group-paging"` 是工具列，第 2 個 `<a>` 標籤是上一頁鈕，我們需要取出此 `<a>` 標籤的 `href` 屬性來繼續處理上一頁的 HTML 網頁，可以看到網址是相對路徑，沒有 `https://www.ptt.cc`。

☆ 步驟四：使用爬蟲工具剖析 HTML 網頁

在分析找出爬蟲的目標 HTML 標籤後，我們可以使用 BeautifulSoup 來剖析 HTML 網頁，如下所示：

```
from bs4 import BeautifulSoup

def parse_html(r):
    if r.status_code == requests.codes.ok:
        r.encoding = "utf8"
        soup = BeautifulSoup(r.text, "lxml")
    else:
        print("HTTP 請求錯誤..." + url)
        soup = None

    return soup
```

上述 `parse_html()` 函數使用 BeautifulSoup 剖析參數的 Response 物件，並且判斷是否請求成功，和指定編碼 `utf8`，可以回傳 BeautifulSoup 物件。

☆ 步驟五：取出所需的資料

現在，我們可以建立爬蟲函數進行文章的資料擷取，取出今天的所有文章資料。

Python 函數：web_scraping_bot()

web_scraping_bot() 函數首先擷取參數 url 的 BBS 看板的第一頁，如下所示：

```
import time

def web_scraping_bot(url):
    articles = []
    print("抓取網路資料中...")
    soup = parse_html(get_resource(url))
    if soup:
        # 取得今天日期，去掉開頭 '0' 符合 PTT 的日期格式
        today = time.strftime("%m/%d").lstrip('0')
```

上述if條件判斷是否有剖析的 BeautifulSoup 物件，如果有，先建立今天的日期，因為 PTT 日期沒有開頭的 '0'，所以刪除此字元，然後呼叫 get_articles() 函數取得文章資料，參數是 BeautifulSoup 物件和今天日期，如下所示：

```
# 取得目前頁面的今日文章清單
current_articles, prev_url = get_articles(soup, today)
while current_articles:
    articles += current_articles
    print("等待 2 秒鐘...")
    time.sleep(2)
    # 剖析上一頁繼續尋找是否有今日的文章
    soup = parse_html(get_resource(URL + prev_url))
    current_articles, prev_url = get_articles(soup, today)

return articles
```

上述 `articles` 串列儲存擷取的文章資料，`while` 迴圈判斷是否有取得文章，如果有，就需要找前一頁，首先將取得的文章資料新增至 `articles` 串列，在等待 2 秒鐘後，使用取得的 `prev_url` 變數（相對路徑），`URL + prev_url` 建立前一頁的 URL 網址，然後重複呼叫 `get_resource()`、`parse_html()` 和 `get_articles()` 函數來取得文章資料，直到沒有今天文章為止，即可結束 `while` 迴圈。

Python 函數：get_articles()

函數 `get_articles()` 共有 2 個參數，第 1 個是 BeautifulSoup 物件，第 2 個是今天日期，如下所示：

```
def get_articles(soup, date):
    articles = []
    # 取得上一頁的超連結
    paging_div = soup.find("div", class_="btn-group btn-group-paging")
    paging_a = paging_div.find_all("a", class_="btn")
    prev_url = paging_a[1]["href"]
```

上述 `articles` 變數儲取得的文章，首先找到上方工具列的 `<div>` 標籤，然後找出所有 `<a>` 標籤，第 2 個 `<a>` 標籤的 `href` 屬性是上一頁的 URL 網址，這是相對路徑的網址。然後，取出文章清單的 `<div>` 標籤，如下所示：

```
tag_divs = soup.find_all("div", class_="r-ent")
for tag in tag_divs:
    # 判斷文章的日期
    if tag.find("div", class_="date").text.strip() == date:
        push_count = 0 # 取得推文數
        push_str = tag.find("div", class_="nrec").text
        if push_str:
            try:
                push_count = int(push_str) # 轉換成數字
```




```
except ValueError: # 轉換失敗,可能是 '爆' 或 'X1','X2'
    if push_str == '爆':
        push_count = 99
    elif push_str.startswith('X'):
        push_count = -10
```

上述程式碼取出所有文章的 `<div>` 標籤後, `for/in` 迴圈一一取出每一篇文章的 `<div>` 標籤, 巢狀 `if` 條件的外層在找出日期後, 判斷是否是今天, 如果是, 接著取出 `<div>` 標籤的推文數, 內層 `if` 條件判斷是否有推文數, 如果有, 轉換成數字, 如果轉換失敗, 就使用 `if/elif` 條件判斷可能的值。

在下方 `if` 條件判斷標題的 `<a>` 標籤是否存在, 如果存在, 就一一取出文章資料的 URL 網址、標題文字和作者, 如下所示：

```
# 取得貼章的超連結和標題文字
if tag.find("a"): # 有超連結,表示文章存在
    href = tag.find("a")["href"]
    title = tag.find("a").text
    author = tag.find("div", class_="author").string
    articles.append({
        "title": title,
        "href": href,
        "push_count": push_count,
        "author": author
    })

return articles, prev_url
```

上述程式碼是將取得的文章資料建立成字典, 然後新增至 `articles` 串列, 回傳值有 2 個, 依序是文章串列和上一頁的 URL 網址。

☆ 步驟六：儲存取出的資料

函數 `web_scraping_bot()` 可以取出所有文章資料的串列，每一篇文章是一個字典，在完成後，就呼叫 `save_to_json()` 函數儲存成 JSON 檔案，如下所示：

```
import json

def save_to_json(articles, file):
    print("今天總共有: " + str(len(articles)) + " 篇文章")
    threshold = MAX_PUSH
    print("熱門文章(> %d 推): " % (threshold))
    for item in articles:    # 顯示熱門文章清單
        if int(item["push_count"]) > threshold:
            print(item["title"], item["href"], item["author"])
    with open(file, "w", encoding="utf-8") as fp: # 寫入 JSON 檔案
        json.dump(articles,fp,indent=2,sort_keys=True,ensure_ascii=False)
```

上述程式碼首先顯示今日的文章資訊，共有幾篇文章，推文數超過 50 的有幾篇，然後開啟 utf-8 編碼的文字檔案，將串列的 JSON 資料寫入 JSON 檔案。

☆ 步驟七：建立主程式執行網路爬蟲

最後，我們可以建立主程式執行上述函數來執行爬蟲作業，如下所示：

```
if __name__ == '__main__':
    url = URL + "/bbs/" + TOPIC + "/index.html"
    print(url)
    articles = web_scraping_bot(url)
    for item in articles:
        print(item)
    save_to_json(articles, "articles.json")
```

上述程式碼首先建立看板的 URL 網址後，呼叫 `web_scraping_bot()` 函數擷取文章資料，在成功後，使用 `for/in` 迴圈顯示文章串列，最後呼叫 `save_to_json()` 函數儲存成 JSON 檔案 `articles.json`。

完整 Python 程式：`/ch6-5/ppt_crawler.py` 的執行結果，可以看到每間隔 2 秒送出 HTTP 請求，在完成後顯示取得的文章清單，最後是今日文章和熱門文章的相關資訊，如下所示：

```
今天總共有: 13 篇文章
熱門文章(> 50 推):
[花邊] MB 給圍巾看自己鎖死 LBJ 的影片來建立信心 /bbs/NBA/M.1655255463.A.E1B.html arod1414
[花邊] Boban 在訓練營毫不留情 /bbs/NBA/M.1655256546.A.22F.html liuuuuuu
[情報] Tatum 是總決賽最年輕 20/7/7 成員 /bbs/NBA/M.1655257043.A.9A0.html ckpioneer
[花邊] 老鷹願用 4 換 1 方案 尋求交易 Gobert /bbs/NBA/M.1655257579.A.5FC.html seguignol
[外絮] 圍巾:打騎士會特別準備,想證明他們錯過 /bbs/NBA/M.1655224542.A.5D1.html arod1414
[情報] 爵士隊想要用 Gobert 換來球星、潛力股、籤 /bbs/NBA/M.1655250939.A.F41.html thnlkj0665
Re: [花邊] 假K湯說他被勇士主場永桶 /bbs/NBA/M.1655251985.A.68E.html love1500274
```

★ 學習評量 ★

- ❶ 請說明什麼是 Web API？網路爬蟲的常見問題有哪些？
- ❷ 當網站內容有分級規定，請使用 PTT 為例，說明 Python 爬蟲程式如何解決這種問題？
- ❸ 在第 6-5 節的 Python 爬蟲程式只能抓取前 9 頁的中階英文單字清單，請修改程式可以抓取全部 15 頁的英文中階和進階單字。
- ❹ 請參考第 6-5 節建立 Python 爬蟲程式，可以從 W3School 網站取出 HTML 標籤說明清單，並且輸出成 CSV 檔案，其 URL 網址如下：

```
https://www.w3schools.com/tags/default.asp
```

- ❺ 請參考第 6-6 節批踢踢 PTT 的 Python 爬蟲程式，建立可跨不同看板討論區的 Python 爬蟲程式，我們準備爬出多個看板的熱門文章，和找出每日各看板前 10 篇熱門文章的推文數。
- ❻ 請參考第 6-3-2 節和 6-6 節在各大網路商城找出一類有興趣的商品，然後建立 Python 爬蟲程式來擷取不同電商網站的商品資料。