python Crawer 網路爬蟲 v.07

製作者：賴志宏
建立日期：2017　最後修改日期: 2021.04.22

Content of Table

1. Introduction

1.1 爬蟲的意義與爬取步驟

- 「網頁爬蟲」（Web Scraping, web crawl）或稱為網頁資料截取（Web Data Extraction）
    - 是一種資料截取技術，可以讓我們直接從網站的HTML網頁取出所需的資料，其過程包含與Web資源進行通訊
    - 剖析文件取出所需資料和整理成資訊，也就是轉換成所需的資料格式。
- Categories
    - one webpage
    - one website
    - multiple websites
- step
    1. 確認要抓取網頁(可能是一個網頁或是一組網頁)的某些部份：
        - Chrome開發人員工具
            - Chrome瀏覽器的開發人員工具可以幫助我們在HTML網頁指出資料的所在，和找出取出此資料的特徵，例如：標籤名稱和屬性值。
    2. 爬取網頁：
        - 靜態網頁：使用HTTP函數庫
            - 與Web伺服器進行HTTP通訊的函數庫，以便取得回應文件的HTML網頁內容，可以使用Requests套件。
        - 互動網頁：
            - 使用selenium (參見另一檔案)
    3. 剖析網頁：
        - 網頁爬蟲工具或函數庫
            - 在取得回應文件後，需要工具或函數庫來剖析文件，以便取出所需的資料，可以使用BeautifulSoup套件。
    4. 資料處理或存檔

In [ ]:
```
# 爬蟲範例

import requests
url = 'https://www.ndhu.edu.tw'
html = requests.get(url)
html.encoding="utf-8"
print(html.text)
```

1.2 爬蟲先備知識

- HTML (見自製投影片)
- CSS (見自製投影片)

1.3 網路禮儀

- 在首頁建立 robots.txt
  - 例如:聯合報網站
    - https://udn.com/robots.txt

robots.txt 線上教學文件

- https://www.seoseo.com.tw/article_detail_602.html
- https://www.awoo.com.tw/blog/robotstxt-crawl/

1.4 網路連線過程

- HTTP 通訊協定
  - HTTP通訊協定的應用程式是一種主從架構（Client-Server Architecture）應用程式，
  - 在客戶端使用URL（Uniform Resource Locations）萬用資源定位器指定連線的伺服端資源，傳送HTTP訊息（HTTP Message）進行溝通，可以請求指定檔案
- HTTP 通訊協定通訊過程
  - Step1：客戶端要求連線伺服端。
  - Step 2：伺服端允許客戶端的連線。
  - Step 3：客戶端送出HTTP請求訊息(request)，內含GET/POST請求取得伺服端的指定檔案。
  - Step 4：伺服端以HTTP回應訊息(response)來回應客戶端的請求，傳回訊息包含請求的檔案內容。

2. 用requests 抓取資料

- 使用requests模組
  - 要安裝 requests 模組:
  - pip install requests 或是 pip3 install requests
  - pip list #可列出目前已安裝的模組

2.1 requests 回應的訊息屬性

- url
  - 網址
- text
  - 網頁原始碼
- raise_for_status
  - 連線訊息詳細內容
- status_code
  - 連線狀態號碼
    - 200 為請求訊息成功
      - requests.codes.ok
    - 400-599 為失敗
    - 404 表示網頁不存在
    - 詳細狀態碼說明
      - https://zh.wikipedia.org/wiki/HTTP%E7%8A%B6%E6%80%81%E7%A0%81

In [ ]:
```
# import requests
url = 'https://www.ndhu.edu.tw'
html = requests.get(url)
html.encoding="utf-8"
if html.status_code == requests.codes.ok:    # 就是 200
    print(html.url)
    print(html.status_code)
    print(requests.codes.ok)
    print('file size:', len(html.text))
```

In [ ]:
```
# 連線失敗時產生的訊號

import requests
url = 'http://aaa.24ht.com.tw/'
html = requests.get(url)
print(html.status_code)
print(html.raise_for_status)    # 連線訊息詳細內容
```

sever header

In [ ]:
```
# get headers of websites' response
# 請注意！標頭名稱區分英文大小寫

import requests
r = requests.get("http://www.google.com")
print(r.headers['Content-Type'])
print(r.headers['Content-Length'])
print(r.headers['Date'])
print(r.headers['Server'])
```

2.2 requests爬取網頁內容

- 取得網頁資料，把所有網頁資料當成一個字串來看
  - import requests模組
  - url = '網址'
  - html = requests.get(url) 取得網頁資料，把所有網頁資料當成一個字串來看
  - html.encoding="utf-8"
  - print(html.text)
- 把網頁內容依分行符號，切割成一行一行字串所組成的串列
  - html = requests.get(url).text.splitlines( ) 把網頁內容依分行符號，切割成一行一行字串所組成的串列
  - for i in range(0, 15):
    - print(html[i])

In [ ]:
```
# 抓取網頁資料
import requests
url = 'https://www.ndhu.edu.tw'
html = requests.get(url)
html.encoding="utf-8"
print(html.text)
```

In [ ]:
```
# 抓取輸入網址的網頁資料
import requests
url = input('Please input URL:')
html = requests.get(url)
html.encoding="utf-8"
print(html.text)
```

抓回來的資料存到串列

In [ ]:
```
# 把網頁內容依分行符號，切割成一行一行字串所組成的串列

import requests
url = 'https://www.ndhu.edu.tw'
html = requests.get(url).text.splitlines( ) # 把網頁內容依分行符號，切割成一行一
for i in range(0, 15):
    print(html[i])
```

2.3 查詢特定網頁文字

- 例如查詢某人是否上榜了

```
if 字串 in html:
   print ("恭喜金榜題名")
else:
   print ("sorry")
```

In [ ]:
```
# _*_ coding: utf-8 _*_
# 程式 9-3 (Python 3 version)

import requests

url = 'http://www.com.tw/exam/check_0001_NO_0_101_0_3.html'
name = input("請輸入要查詢的姓名:")
html = requests.get(url).text
if name in html:
    print("恭喜名列金榜")
else:
    print("不好意思，榜單中找不到{}".format(name))
```

In [ ]:
```
# ch05

import requests
url = 'https://tw.news.yahoo.com/'
html = requests.get(url)
html.encoding="utf-8"

htmllist = html.text.splitlines()
n=0
for row in htmllist:
    if "台灣" in row: n+=1
print("找到 {} 次!".format(n))
```

In [ ]:
```
# check whether the crawl is successful

import requests
url = 'https://www.bbc.com'
html = requests.get(url)
html.encoding="utf-8"
if html.status_code == requests.codes.ok:
    print(html.text)
```

3. 編碼 encoding

- 編碼問題：
  - 在html中會設定編碼方式，例如：
    - 取得網頁的編碼方式 (.encoding)
      - html = requests.get(url)
        print(html.encoding)
  - 設定編碼方式：
    - 若是抓下來之後遇到編碼錯誤，可以用以下方式處理：
      print("encoding: %s" % html.encoding) # 列印編碼方式
      html.encoding = 'big5' # 使用big5碼
      html.encoding = 'utf8' # 使用utf8碼
    之後再列印網頁原始碼：print(html.text)
  - 註：編碼方式：big5 有不同的種類，例如微軟cmd 的編碼為: cp950，而基於香港增補字擴充的編碼則為: big5hkscs

In [ ]:
```
import requests
url = 'https://tw.news.yahoo.com/'
html = requests.get(url)
print(html.encoding)
```

In [ ]:
```
# 使用不對的編碼，會出現亂碼

import requests

r = requests.get("https://www.ndhu.edu.tw/")
print(r.encoding)  # 原來的編碼
r.encoding = 'big5'  # 設定為不正確的編碼
print(r.text)
```

In [ ]:
```
import requests

r = requests.get("https://www.ndhu.edu.tw/")

code = r.encoding
r.encoding = code
print(r.encoding)
print(r.text)
```

4. 開發人員工具 Developer part 1

4.1 [開發人員工具]尋找網頁與原始碼的位置

使用時機

- 在進行網頁資料擷取前，我們需要先分析HTML網頁來找出目標資料的特徵
    - 例如：資料位在哪一個標籤， **標籤是否有唯一的id屬性，有唯一就可以直接搜尋**
    - 如果未在搜尋到標籤的附近，我們可以再次搜尋，或使用走訪方式來處理
    - 換句話說，我們需要分析HTML網頁來找出搜尋和走訪的策略，以便將所需資料擷取出來
- 主要找尋位置的參考依據
    - HTML 標籤
    - id屬性
    - class 屬性

使用方式

- 開啟[開發人員工具]
    - 在Chrome 按下 F12 (或是在網頁上按右鍵/檢查)
- 尋找網頁與原始碼對應的位置
    - 在網頁上取得原始碼的對應位置
        - 選取工具最左邊的箭頭[Select element]，並選取[Elements]標籤，之後在網頁上的位置按一下，原始碼就會移到對應的位置
    - 在原始碼上取得網頁的對應位置
        - 原始碼的HTML標記上上按右鍵/scroll into view

4.2 [開發人員工具]其它常用功能

- 在網頁中按下F5功能鍵，以重新整理網頁
- 常使用的功能
    - 搜尋資料:在原始碼上按Ctrl + F
        - 和CSS的語法一樣:例如搜尋class，可以用[.class名稱]
    - 修改內容(在Elements項目下)
        - 在原始碼中編輯與修改，網頁內容會跟著被改變
        - 若網頁有動態按鈕，選取動態按鈕後，右邊視窗的Style中可修改:hov，例如修改:active, :focus
    - 複製與仿製結構
        - 製作網頁時，可以複製現有的結構(右鍵/copy/copy outerHtml)，供作修改用

5. HTML網頁分析：BeautifulSoup 模組

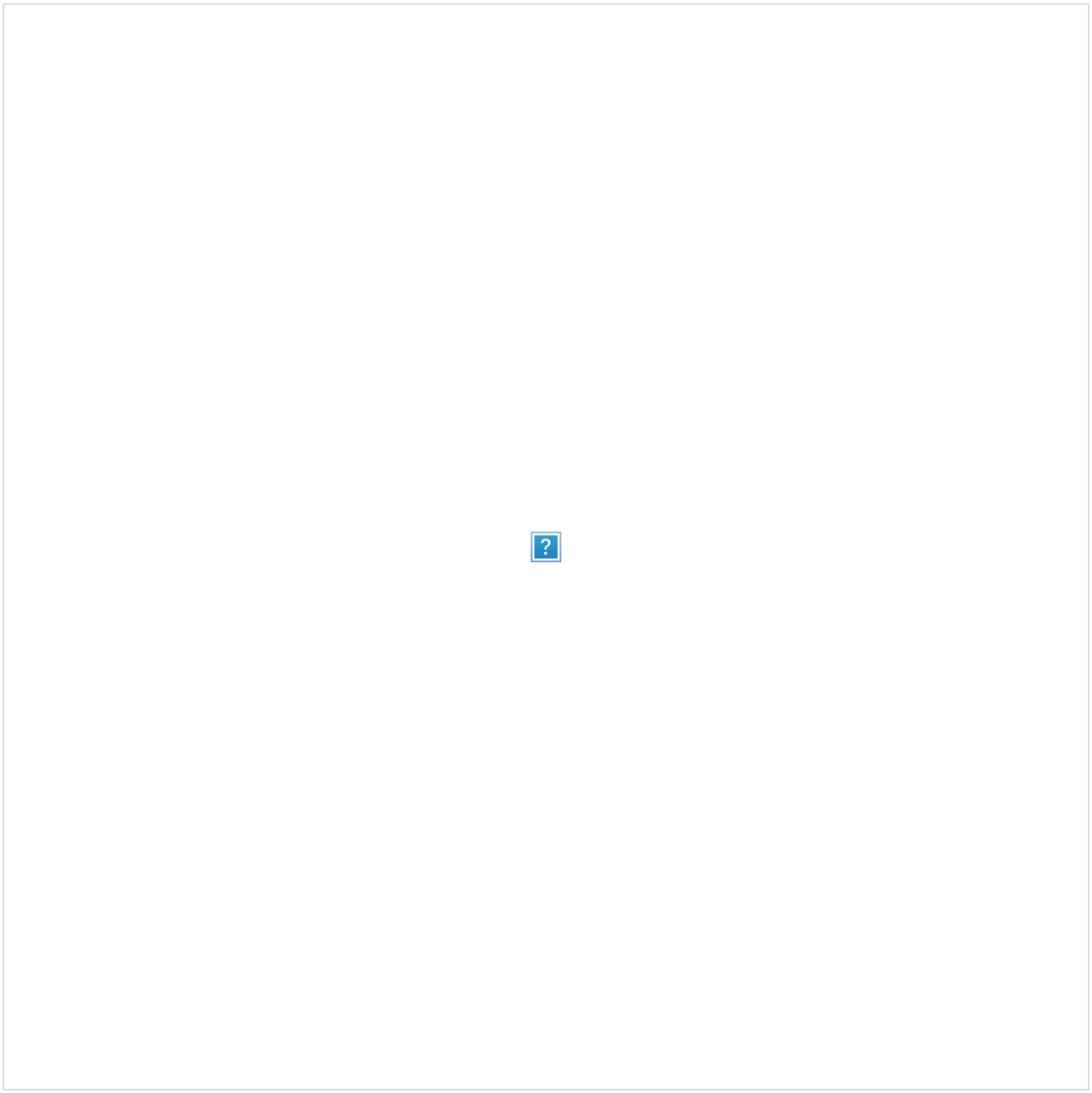
5.0 網頁的架構

In [ ]:

```
<!DOCTYPE html>
<html lang="big5">
 <head>
  <meta charset="utf-8"/>
  <title>測試資料擷取的HTML網頁</title>
 </head>
 <body>
  <!-- Surveys -->
  <div class="surveys" id="surveys">
   <div class="survey" id="q1">
    <p class="question">
     <a href="http://example.com/q1">請問你的性別?</a></p>
    <ul class="answer">
     <li class="response">男 -
      <span class="score selected">20</span></li>
     <li class="response">女 -
      <span class="score">10</span></li>
    </ul>
   </div>
   <div class="survey" id="q2">
    <p class="question">
     <a href="http://example.com/q2">請問你是否喜歡偵探小說?</a></p>
    <ul class="answer">
     <li class="response">喜歡 -
      <span class="score">40</span></li>
     <li class="response">普通 -
      <span class="score selected">20</span></li>
     <li class="response">不喜歡 -
      <span class="score">0</span></li>
    </ul>
   </div>
   <div class="survey" id="q3">
    <p class="question">
     <a href="http://example.com/q3">請問你是否會程式設計?</a></p>
    <ul class="answer">
     <li class="response">會 -
      <span class="score selected">34</span></li>
     <li class="response">不會 -
      <span class="score">6</span></li>
    </ul>
   </div>
  </div>
  <div class="emails" id="emails">
   <div class="question">電子郵件清單資訊: </div>
   abc@example.com
   <div class="survey" data-custom="important">def@example.com</div>
   <span class="survey" id="email">ghi@example.com</div>
  </div>
 </body>
</html>
```

網頁結構圖:

- 這樣的結構又稱作DOM（Document Object Model）
- DOM 形成的樹狀結構中最重要的就是各個節點(node)。
- 在DOM裡面每個元素、文字等都算是一個節點
- 爬蟲其實就是在DOM的節點之間來回的遍歷、尋找目標節點，然後擷取目標內容



5.1 BeautifulSoup 簡介

功能: 進行HTML網頁分析

- 優點
  - 將原先為網頁原始的字串，變為beautifulsoup的物件，可以針對網頁的部分內容(以HTML標籤為單位)進行提取、搜尋等
  - 如 xx.title, xx.find()

- 要安裝 beautifulsoup 模組(目前版本為第4版），若是安裝anaconda，則已經內含，不需要再安裝

  - pip install beautifulsoup4 或是 pip3 install beautifulsoup4
- 要import beautifulsoup 模組

  - from bs4 import BeautifulSoup
  - import requests
  - url = '網址'
  - html = requests.get(url).text 取得網頁資料，把所有網頁資料當成一個字串來看
  - sp = BeautifulSoup(html, "lxml") 解析HTML 之後的結果存在sp中，第二個參數'lxml'為解析html標籤的演算法，也可使用'html.parser'但速度較慢，相容性較差
  - 或是只寫 sp = BeautifulSoup(html)
- prettify() 可以重新整理抓回來的網頁內容，對於缺的資料會補齊，例如缺少右邊的</html>

In [ ]:
```
# get source code

import requests
from bs4 import BeautifulSoup
url = 'https://udn.com/news/index'
html = requests.get(url)
html.encoding = 'utf8'  # 使用utf8碼
sp = BeautifulSoup(html.text, 'lxml')
print(type(sp))
print()
print(sp)
```

In [ ]:
```
# use prettify() function in BeautifulSoup object to format the text
import requests
from bs4 import BeautifulSoup
url = 'https://udn.com/news/index'
html = requests.get(url)
html.encoding = 'utf8'  # 使用utf8碼
sp = BeautifulSoup(html.text, 'html.parser')  # 或是 sp = BeautifulSoup(htm
print(sp.prettify())
```

5.2 BeautifulSoup 搜尋特定HTML標籤

- xx.標籤名稱: 傳回第一個HTML的標籤內容
    - xx.title： 傳回此網頁的標題
    - xx.head
    - xx.a
- find()：傳回第一個符合條件的內容，為單一物件，即HTML標籤物件；沒有找到傳回None
    - 語法
        - find(name, attribute, recursive, text, **kwargs)
            - name參數：指定搜尋的標籤名稱，可以找到第1個符合的HTML標籤，值可以是字串的標籤名稱、正規運算式、清單或函數。
            - attribute參數：搜尋條件的HTML標籤屬性。
            - recursive參數：布林值預設是True，搜尋會包含所有子孫標籤；如為False，搜尋就只限下一層子標籤，不包含再下一層的孫標籤。
            - text參數：指定搜尋的標籤字串內容。
    - 範例
        - .find('標籤名稱')
            - 如.find('img')
                - 傳回第一個img 的tag
            - .find('a')
        - .find('屬性名稱' = '屬性的值')
            - .find(src = "xxx")
            - .find(id="id名稱")
            - .find(class_="class名稱")
                - 由於 class 是 Python 程式語言的保留字，所以 Beautiful Soup 改以 class_ 這個名稱代表 HTML 節點的 class 屬性
                - class_ 也可省略
        - .find('標籤名稱', '屬性名稱' = '屬性的值')
            - .find(img, src = "dog.png")
                - 搜尋第一個 img 標籤，且src = "dog.png"
        - .find(attrs = {'attribute': 'attribure_value'})
            - 上述方法.find('屬性名稱' = '屬性的值')在HTML5 中有一些屬性名稱若直接寫在 Python 的參數中會有一些問題，
                - 例如 data-* 這類的屬性直接寫的話，就會產生錯誤訊息
            - 可以改用此方法
            - .find(attrs = {"ref": "xxx"})
        - .find(tag_name, {'attribute': 'attribure_value'})
            - .find('span', {'id': 'Showtd'}) #找出id為showtd的 \ 內的第一個 html 元素

In [ ]:
```
html_doc = """
<html><head><title>網頁標題</title></head>
<body>
<p class="title"><b>文件標題</b></p>

<p class="story">Once upon a time there were three little sisters; and the
<a href="http://example.com/Rebecca" class="sister1" id="link1">Rebecca</a
<a href="http://example.com/Richard" class="sister2" id="link2">Richard <i
and
<a href="http://example.com/tillie" class="sister1" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>
<img src = "images/pig.png" width = "20" height = "100">

<p class="story">...</p>
</body> </html>
"""
```

In [ ]:
```
from bs4 import BeautifulSoup
sp = BeautifulSoup(html_doc,'lxml')
print(sp.title)
print(sp.head)
print(sp.a)
print(sp.p)
```

In [ ]:
```
data1 = sp.find("a")
data2 = sp.find("a", href = "http://example.com/Richard")

print("data1:\n", data1)
print("\ndata2:\n", data2)
```

In [ ]:
```
# class_可省略

print(sp.find("a", class_ = "sister2"))
print(sp.find("a", "sister2"))
```

In [ ]:
```
# find()

from bs4 import BeautifulSoup
sp = BeautifulSoup(html_doc,'lxml')

print("sp.tile ->\n", sp.title)

print("\nsp.find('b') ->\n", sp.find('b')) # <b>文件標題</b>

data1=sp.find("a", {"href":"http://example.com/Richard"})
print('\nsp.find("a", {"href":"http://example.com/Richard"}->\n', data1)

data11=sp.find(id="link2")
print('\nsp.find(id="link2")\n', data11)

data12=sp.find(class_="sister2")
print('\nsp.find(class_="sister2")\n', data12)

data2=sp.find("a", {"id":"link2"})
print('\nsp.find("a", {"id":"link2"}) \n', data2)

data3=sp.find("a", {"class":"sister1"})
print('\nsp.find("a", {"class":"sister1"}) \n', data3)
```

In [ ]:
```
# find() 搜尋第二層

from bs4 import BeautifulSoup
sp = BeautifulSoup(html_doc,'lxml')

data1=sp.find("a", {"href":"http://example.com/Richard"})
print('sp.find("a", {"href":"http://example.com/Richard"}->\n', data1)

data2 = data1.find("img")
print("\n搜尋a下一層的img->\n", data2)
```

In [ ]:
```
from bs4 import BeautifulSoup
sp = BeautifulSoup(html_doc,'lxml')

data1=sp.find("a", class_= "sister1")
print('\nsp.find("a", class_="sister1") \n', data1)

data1=sp.find("a", id = "link2")
print('\nsp.find("a", id = "link2") \n', data1)
```

In [ ]:
```
# ch5_10.py
import bs4

objSoup = bs4.BeautifulSoup(html_doc, 'lxml')
objTag = objSoup.find_all(class_ = 'sister2')
print(objTag[0].attrs)
```

find_all()

- find_all 和 find()的差異
  - 傳回所有符合條件的HTML標籤
  - 傳回值為Result_Set(為一串列)，find()傳回標籤

- find_all()：儲回所有符合條件的內容，為串列
  - .find_all('標籤名稱')
  - .find_all(屬性 = "xxx")
    - .find_all(src = "xxx")
    - .find_all(id = "xxx")
    - .find*all(class* = "xxx") # 注意: class要加上_
    - 註:在 HTML5 中有一些屬性名稱若直接寫在 Python 的參數中會有一些問題，例如 data-* 這類的屬性直接寫的話，就會產生錯誤訊息
    - 可改為另一種格式
      - .find_all(attrs={"tag_name": "value"})
  - .find_all('標籤名稱', 屬性 = "xxx") # 取出HTML tag名稱，且屬性為 "xxx"者，為And運算
    - .find*all('a', class* = "xxx")
    - 也可使用另一種格式.find_all(tag_name, {'attribute', 'attribure_value'})
      - .find_all('span', {'id': 'Showtd'}) #找出id為showtd的 \ 內的所有html 元素
      - .find_all('span', {'class': 'blue'}) #找出class為blue的 \ 內的所有html 元素
      - .find_all('a', {'href':'http://www.ndhu.edu.tw'}) #找出href為'http://www.ndhu.edu.tw'的 \ 內的所有html 元素
  - .find_all(['h1', 'h2']) # 同時取多種的HTML tag, 需要使用中括號[ ]，此為OR的運算

- find_all 的參數
  - find_all(limit = xx, recursive = False/True)
    - limit 限制抓取的數量
    - recursive 是否往下一層

In [ ]:
```
html_doc = """
<html><head><title>網頁標題</title></head>
<body>
<p class="title"><b>文件標題</b></p>

<p class="story">Once upon a time there were three little sisters; and thei
<a href="http://example.com/Rebecca" class="sister1" id="link1">Rebecca</a>
<a href="http://example.com/Richard" class="sister2" id="link2">Richard <i
and
<a href="http://example.com/tillie" class="sister1" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>
<img src = "images/pig.png" width = "20" height = "100">

<span class = "sister2"> test2 </span>

<p class="story">...</p>
</body> </html>
"""
```

In [ ]:
```
from bs4 import BeautifulSoup
sp = BeautifulSoup(html_doc,'lxml')

test1 = sp.find('a')
test2 = sp.find_all('a')

print('.find() -> \n', test1)
print('\n.find_all() -> \n', test2)


print('\n.find() -> \n', type(test1))
print('\n.find_all() -> \n', type(test2))
```

In [ ]:
```
# find_all()

from bs4 import BeautifulSoup
sp = BeautifulSoup(html_doc,'lxml')

data1 = sp.find_all("a")
data2 = sp.find_all("a", href = "http://example.com/Rebecca")
data3 = sp.find_all(href = "http://example.com/Rebecca")
data4 = sp.find_all("a", class_ = "sister2")  # 所有 a標籤，且 class 為 "sist
data5 = sp.find_all(class_ = "sister2")   # 所有 a標籤，且 class 為 "sister2"
data6 = sp.find_all(["a", "img"])  # 所有 a標籤或是 img 標籤，兩者都會取出，是進

print("data1:\n", data1)
print("\ndata2:\n", data2)
print("\ndata3:\n", data3)
print("\ndata4:\n", data4)
print("\ndata5:\n", data5)
print("\ndata6:\n", data6)
```

In [ ]:
```
# class_可省略

print(sp.find_all("a", class_ = "sister2"))
print(sp.find_all("a", "sister2"))
```

In [ ]:
```
# find all tags "a" which classes' names are "sister2"

print('\nsp.find_all("a", {"class":"sister2"}) ->\n', sp.find_all("a", {"cl

print('\nsp.find_all("a", attrs = {"class":"sister2"}) ->\n', sp.find_all('

print('\nsp.find_all({"class":"sister2"}) ->\n', sp.find_all({"class":"sist

print('\nsp.find_all(attrs={"class":"sister2"}) ->\n', sp.find_all(attrs =
```

In [ ]:
```
# 進行 OR 運算，取出網頁元素

a_img = sp.find_all(['a', 'img'])
print(a_img)

sisters = sp.find_all(class_ = ['sister1', 'sister2'])
print('\n', sisters)
```

xx.find_all(True) 取出所有子節點，成為串列

- 注意: xx 必須為字串，此方法只能用在字串或標籤上
- 可以用在find()蒐尋回傳的內容

In [ ]:
```
story = sp.find('p', class_='story')
print(story)
print()

print(story.find_all(True))
```

sp.find_all('a') 取出中網頁中所有的連結，結果存到串列

- 之後就可以使用串列索引來存取連結的內容，例如：
- links[10]的內容可能是 go to google
- links[10].contents 的內容是 "go to google"
- links[10].get('href') 的內容是 "http://www.google.com"

In [ ]:
```
# find_all()得到的串列

from bs4 import BeautifulSoup
sp = BeautifulSoup(html_doc,'lxml')

a_links = sp.find_all('a')
print("a_links = sp.find_all('a') ->\n", a_links) # [<b>文件標題</b>]

print("\na_links[2]:", a_links[2])

print('\nlinks逐一顯示 ->')
for link in a_links:
    print(link)
```

5.3 取得標籤後處理標籤的部分內容

- 取得HTML的內文
- 取得子標籤
- 取得HTML的屬性

取得HTML的內文

- name屬性:取回標籤的名稱
- text屬性：取回網頁的內容中，只取出文字的部份，不含HTML標籤
- string屬性:取回網頁的內容中，只取出文字的部份，不含HTML標籤；但標籤中若還有標籤，取出的內容為空的
- Differences between .text & .string
  - https://stackoverflow.com/questions/25327693/difference-between-string-and-text-beautifulsoup
- contents屬性:取出扣除標籤後的內容，為一串列的資料型態
- get_text()：取回網頁的內容中，只取出文字的部份，不含HTML標籤,與text屬性相同
- getText()：取回網頁的內容中，只取出文字的部份，不含HTML標籤,與text屬性相同

In [ ]:
```
# string vs. text 的差異 1

from bs4 import BeautifulSoup
sp = BeautifulSoup(html_doc,'lxml')

data1=sp.find("a", {"href":"http://example.com/Rebecca"})
print('data1:\n', data1)
print('data1.name:', data1.name)
print('\ndata1.string ->\n', data1.string)
print('\ndata1.text ->\n', data1.text)
print('\ndata1.contents ->\n', data1.contents)
print('\ndata1.getText() ->\n', data1.getText())
```

In [ ]:
```
# string vs. text 的差異 2
# ch. 4_6_3b

from bs4 import BeautifulSoup

html_str = "<div id='msg'>Hello World! <p> Final Test <p></div>"
soup = BeautifulSoup(html_str, "lxml")
tag = soup.div
print("tag.string ->\n", tag.string)        # string屬性
print("\ntag.text ->\n", tag.text)
print(type(tag.text))# text屬性
print('\ntag.contents ->\n', tag.contents)
print('\ndata1.getText() ->\n', tag.getText())

print("\ntag.get_text() ->\n", tag.get_text())    # get_text()函數
print("\n", tag.get_text("-")) # 用 - 隔開文字內容
print("\n", tag.get_text("-", strip=True))
```

取得子標籤

- beautifulsoup物件.子標籤名稱

In [ ]:
```
html_doc = """
<html><head><title>網頁標題</title></head>
<body>
<p class="title"><b>文件標題</b></p>

<p class="story">Once upon a time there were three little sisters; and the
<a href="http://example.com/Rebecca" class="sister1" id="link1">Rebecca</a
<a href="http://example.com/Richard" class="sister2" id="link2">Richard <i
and
<a href="http://example.com/tillie" class="sister1" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>
<img src = "images/pig.png" width = "20" height = "100">

<p class="story">...</p>
</body> </html>
"""
```

In [ ]:
```
# 取得子標籤

data2=sp.find("a", {"class":"sister2"})
print('data2->\n', data2)
print('\ndata2.img->\n', data2.img)
```

取得HTML的屬性

- 方法一:beautifulsoup物件['attribe_name']
  - sp['class']
  - sp['id']
- 方法二:.get('屬性名稱')
  - .get('href')
  - .get('src')
- 方法三:.attrs
  - 標籤所有屬性值的字典

In [ ]:
```
from bs4 import BeautifulSoup
sp = BeautifulSoup(html_doc,'lxml')

data1=sp.find("a", {"href":"http://example.com/Rebecca"})
print('data1:\n', data1)
print("\ndata1['href'] ->\n", data1['href'])
print('\ndata1.get("href") ->\n', data1.get('href'))

data3=sp.find("img")
print('\ndata3\n', data3)
print('\ndata3.get("src") ->\n', data3.get("src")) # http://example.com/el
print('\ndata3["src"] ->\n', data3["src"])

data2=sp.find("a", {"class":"sister2"})
print('\ndata2\n', data2)
print('\ndata2.img\n', data2.img)
print('\ndata2.img["src"] ->\n', data2.img["src"])
```

In [ ]:
```
data2=sp.find("a", {"class":"sister2"})
print('\ndata2\n', data2)

print('\ndata2.attrs ->\n', data2.attrs)

print("\ndata2.attrs['href'] ->\n", data2.attrs['href'])
print("\ndata2.attrs['class'] ->\n", data2.attrs['class'])
```

5.4 Get links & images in a website

In [ ]:
```
# get all links in a website

from bs4 import BeautifulSoup
import requests

url = 'http://www.ndhu.edu.tw/'

html = requests.get(url).text
sp = BeautifulSoup(html, 'lxml')
all_links = sp.find_all('a')

for link in all_links:
   href = link.get('href')   #get( )方法會取回完整的html, get_text( ) 方法為只
   if href != None and href.startswith('http://'):
      print(href)
```

In [ ]:
```
# get all links in a website, including http:// and https://

# 範例：取得某一網頁的全部連結
#在命令列中輸入右邊的命令來執行此程式： python get_href.py http://www.ndhu.edu.tw
from bs4 import BeautifulSoup
import requests

url = input('網址:')

html = requests.get(url).text
sp = BeautifulSoup(html, 'lxml')
all_links = sp.find_all('a')

for link in all_links:
   href = link.get('href')   #get( )方法會取回完整的html, get_text( ) 方法為只
   if href != None and (href.startswith('http://') or href.startswith('http
      print(href)
```

In [ ]:
```
# get all address of links and images
# ch05
from bs4 import BeautifulSoup
import requests

url = 'http://www.e-happy.com.tw'
html = requests.get(url)
html.encoding="utf-8"

sp=BeautifulSoup(html.text, 'lxml')
links=sp.find_all(["a","img"]) # 同時讀取 <a> 和 <img>
for link in links:
    href=link.get("href") # 讀取 href 屬性內容
    # 判斷內容是否為非 None，並且開頭文字是 http://
    if  href != None and href.startswith("http://"):
        print(href)
```

In [ ]:
```
# 列出網頁中所有圖片的位置及檔名

# _*_ coding: utf-8 _*_
# 程式 9-6 (Python 3 version)

from bs4 import BeautifulSoup
import requests
import sys
from urllib.parse import urlparse

url = input('請輸入網址，以列出所有圖片的位置及檔名：')
domain = "{}://{}".format(urlparse(url).scheme, urlparse(url).hostname)
html = requests.get(url).text
sp = BeautifulSoup(html, 'html.parser')
all_links = sp.find_all(['a','img'])

for link in all_links:
    src = link.get('src')
    href = link.get('href')
    targets = [src, href]
    for t in targets:
        if t != None and ('.jpg' in t or '.png' in t):
            if t.startswith('http'):
                print(t)
            else:
                print(domain+t)
```

5.5 將抓回來的資料存檔

- text files
  - fp = open('檔名','w')
  - fp.write(要存的檔案內容)
  - fp.close()
- image files

In [ ]:
```python
# 存文字檔
# ch. 4_5_2

import requests

r = requests.get("http://www.flag.tw")

fp = open("flag.txt", "w", encoding="utf8")
fp.write(r.text)
print("寫入檔案flag.txt...")
fp.close()
```

In [ ]:
```python
# 讀檔案

# ch. 4_5_2a

fp = open("flag.txt", "r", encoding="utf8")
str = fp.read()
print("檔案內容:")
print(str)
```

In [ ]:
```python
# 讀檔案:使用 with open
# ch. 4_5_2b

with open("flag.txt", "r", encoding="utf8") as fp:
    str = fp.read()
    print("檔案內容:")
    print(str)
```

In [ ]:
```python
# 讀檔案:使用 xx.readlines()

# ch. 4_5_2c

with open("flag.txt", "r", encoding="utf8") as fp:
    list1 = fp.readlines()
    for line in list1:
        print(line, end="")
```

In [ ]:
```python
# 讀檔案之後用 beautifulsoup 解析 HTML

# ch. 4_6_1b

from bs4 import BeautifulSoup

with open("index.html", "r", encoding="utf8") as fp:
    soup = BeautifulSoup(fp, "lxml")
    print(soup)
```

In [ ]:

```python
# 下載網頁中所有圖片 download all images
# 程式 9-7 (Python 3 version)

from bs4 import BeautifulSoup
import requests
import sys, os
from urllib.parse import urlparse
from urllib.request import urlopen

image_dir = 'download_images'

url = input('請輸入網址，以下載網頁中所有圖片：')

domain = "{}://{}".format(urlparse(url).scheme, urlparse(url).hostname)
html = requests.get(url).text
sp = BeautifulSoup(html, 'html.parser')
all_links = sp.find_all(['a','img'])

for link in all_links:
    src = link.get('src')
    href = link.get('href')
    targets = [src, href]
    for t in targets:
        if t != None and ('.jpg' in t or '.png' in t):
            if t.startswith('http'): full_path = t
            else:                    full_path = domain+t
            print(full_path)

#             image_dir = url.split('/')[-1]
            if not os.path.exists(image_dir): os.mkdir(image_dir)
            filename = full_path.split('/')[-1]
            ext = filename.split('.')[-1]
            filename = filename.split('.')[-2]
            if 'jpg' in ext: filename = filename + '.jpg'
            else:            filename = filename + '.png'
            image = urlopen(full_path)
            fp = open(os.path.join(image_dir,filename),'wb')
            fp.write(image.read())
            fp.close()
```

In [ ]:

```
#ch05
import requests,os
from bs4 import BeautifulSoup
from urllib.request import urlopen

url = 'http://www.tooopen.com/img/87.aspx'

html = requests.get(url)
html.encoding="utf-8"

sp = BeautifulSoup(html.text, 'html.parser')

# 建立 images 目錄儲存圖片
images_dir="images_test/"
if not os.path.exists(images_dir):
    os.mkdir(images_dir)

# 取得所有 <a> 和 <img> 標籤
all_links=sp.find_all(['a','img'])
for link in all_links:
    # 讀取 src 和  href 屬性內容
    src=link.get('src')
    href = link.get('href')
    attrs=[src,href]
    for attr in attrs:
        # 讀取  .jpg 和  .png 檔
        if attr != None and ('.jpg' in attr or '.png' in attr):
            # 設定圖檔完整路徑
            full_path = attr
            filename = full_path.split('/')[-1]  # 取得圖檔名
            print(full_path)
            # 儲存圖片
            try:
                image = urlopen(full_path)
                f = open(os.path.join(images_dir,filename),'wb')
                f.write(image.read())
                f.close()
            except:
                print("{} 無法讀取!".format(filename))
```

In [ ]:

In [ ]:

```
# 威力彩開獎程式 (第一部份) Lotto (part 1)

#ch05
import requests
from bs4 import BeautifulSoup

url = 'http://www.taiwanlottery.com.tw/'
html = requests.get(url)
sp = BeautifulSoup(html.text, 'html.parser')

data1 = sp.select("#rightdown")
#print(data1)

data2 = data1[0].find('div', {'class':'contents_box02'})
#print(data2)

data3 = data2.find_all('div', {'class':'ball_tx'})

print(data3)
```

In [ ]:

```
# 威力彩開獎程式 (完整版) Lotto (full)
#ch05
import requests
from bs4 import BeautifulSoup

url = 'http://www.taiwanlottery.com.tw/'
html = requests.get(url)
sp = BeautifulSoup(html.text, 'html.parser')

data1 = sp.select("#rightdown")
#print(data1)

data2 = data1[0].find('div', {'class':'contents_box02'})
#print(data2)

data3 = data2.find_all('div', {'class':'ball_tx'})
#print(data3)
#
# 威力彩號碼
print("開出順序：",end="")
for n in range(0,6):
    print(data3[n].text,end="  ")

print("\n大小順序：",end="")
for n in range(6,len(data3)):
    print(data3[n].text,end="  ")

## 第二區
red = data2.find('div', {'class':'ball_red'})
print("\n第二區：{}".format(red.text))
```

In [ ]:

```
# 取得中油的每日油價，包含柴油、95和98無鉛汽油，只取有改變的部份
# 取出表格的內容

# _*_ coding: utf-8 _*_
# 程式 9-8 (Python 3 version)

from bs4 import BeautifulSoup
import requests

url = 'https://new.cpc.com.tw/division/mb/oil-more4.aspx'
html = requests.get(url).text
sp = BeautifulSoup(html, 'html.parser')
data = sp.find_all('span', {'id':'Showtd'})
rows = data[0].find_all('tr')

prices = list()
for row in rows:
    cols = row.find_all('td')
    if len(cols[1].text) > 0:
        item = [cols[0].text, cols[1].text, \
                cols[2].text, cols[3].text]
        prices.append(item)
for p in prices:
    print(p)
```

In [ ]:

```
# 延續上一程式
# 取得中油的每日油價，並寫到網頁中

# _*_ coding: utf-8 _*_
# 程式 9-9 (Python 3 version)

from bs4 import BeautifulSoup
import requests

pre_html = '''
<!DOCTYPE html>
<html>
<head>
<meta charset='utf-8'>
<title>中油油價歷史資料</title>
</head>
<body>
<h2>中油油價歷史資料（取自中油官方網站）</h2>
<table width=600 border=1>
<tr><td>日期</td><td>92無鉛</td><td>95無鉛</td><td>98無鉛</td></tr>
'''

post_html = '''
</table>
</body>
</html>
'''

url = 'http://new.cpc.com.tw/division/mb/oil-more4.aspx'

html = requests.get(url).text
sp = BeautifulSoup(html, 'html.parser')
data = sp.find_all('span', {'id':'Showtd'})
rows = data[0].find_all('tr')

prices = list()
for row in rows:
    cols = row.find_all('td')
    if len(cols[1].text) > 0:
        item = [cols[0].text, cols[1].text, \
                cols[2].text, cols[3].text]
        prices.append(item)

html_body = ''
for p in prices:
    html_body += "<tr><td>{}</td><td>{}</td><td>{}</td><td>{}</td></tr>".\
          format(p[0],p[1],p[2],p[3])
html_file = pre_html + html_body + post_html

fp = open('oilprice.html','w')
fp.write(html_file)
fp.close()
```

In [ ]:

```
# 抓取跑馬燈的圖片

# _*_ coding: utf-8 _*_
# 程式 9-10 (Python 3 version)

from bs4 import BeautifulSoup
```

```
import requests
import sys, os
from urllib.parse import urlparse
from urllib.request import urlopen

post_html = '''
</body>
</html>
'''

url = input('請輸入網址：')
domain = "{}://{}".format(urlparse(url).scheme, urlparse(url).hostname)
html = requests.get(url).text
sp = BeautifulSoup(html, 'html.parser')

pre_html = """
<!DOCTYPE html>
<html>
<head>
<meta charset='utf-8'>
<title>網頁搜集來的資料</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.
  <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap
  <style>
  .carousel-inner > .item > img,
  .carousel-inner > .item > a > img {
      border: 5px solid white;
      width: 50%;
      box-shadow: 10px 10px 5px #888888;
      margin: auto;
  }
  </style>

</head>
<body>
<center><h3>以下是從網頁搜集來的圖片跑馬燈</h3></center>
"""

all_links = sp.find_all(['a','img'])

carousel_part1 = ""
carousel_part2 = ""
picno = 0

for link in all_links:
    src = link.get('src')
    href = link.get('href')
    targets = [src, href]
    for t in targets:
        if t != None and ('.jpg' in t or '.png' in t):
            if t.startswith('http'): full_path = t
            else:                    full_path = domain+t
            print(full_path)
            image_dir = url.split('/')[-1]
            if not os.path.exists(image_dir): os.mkdir(image_dir)
            filename = full_path.split('/')[-1]
            ext = filename.split('.')[-1]
            filename = filename.split('.')[-2]
            if 'jpg' in ext: filename = filename + '.jpg'
```

```
            else:            filename = filename + '.png'
            image = urlopen(full_path)
            fp = open(os.path.join(image_dir,filename),'wb')
            fp.write(image.read())
            fp.close()

            if picno==0:
                carousel_part1 += "<li data-target='#myC' data-slide-to='{}
                carousel_part2 += """
                    <div class='item active'>
                        <img src='{}' alt='{}'>
                    </div>""".format(filename, filename)

            else:
                carousel_part1 += "<li data-target='#myC' data-slide-to='{}
                carousel_part2 += """
                    <div class='item'>
                        <img src='{}' alt='{}'>
                    </div>""".format(filename, filename)
            picno += 1

            html_body = """
            <div id='myC' class='carousel slide' data-ride='carousel'>
                <ol class='carousel-indicators'>
                {}
                </ol>
                <div class='carousel-inner' role='listbox'>
                {}
                </div>
                <a class="left carousel-control" href="#myC" role="button"
                    <span class="glyphicon glyphicon-chevron-left" aria-hi
                    <span class="sr-only">前一張</span>
                </a>
                <a class="right carousel-control" href="#myC" role="button"
                    <span class="glyphicon glyphicon-chevron-right" aria-h
                    <span class="sr-only">後一張</span>
                </a>
            </div>
            """.format(carousel_part1, carousel_part2)

fp = open(os.path.join(image_dir,'index.html'), 'w')
fp.write(pre_html+html_body+post_html)
fp.close()
```

In [ ]:

5.6 使用 CSS 搜尋網頁內容

- select_one()：傳回以CSS選擇器作為運算結果的，類似find()的功能
- select()：傳回以CSS選擇器作為運算結果的所有內容，傳回值為串列，主要操作對象為HTML tage, id和class,類似find_all()的功能
  - HTML tag:
    - .select('tag_name')
      - sp.select('div')
  - id
    - .select('#id_name)
      - sp.select('#link1')
  - class
    - .select('.class_name')
      - sp.select('.class1')
  - select()可以使用複合方式的結構
    - .select('div:nth-of-type(4)') # 抓取第4個div
    - .select('p > a') # 取直接子標籤
    - .select('ul > li:nth-of-type(2)') # 從1開始
    - .select('html head title') # 抓取子孫標籤
    - .select('div a') # 抓取div下的所有a標籤，不管div和a分隔了幾層
    - .select('div > .email) # 取div之下的class為email
    - .select('div ~ .email') # 所有兄弟標籤
    - .select('div + .email') # 下一個兄弟標籤

In [ ]:
```
data1=sp.select("a")      # 取 tag a
print('data1:\n', data1)
print("\ndata1[0]['href'] ->\n", data1[0]['href'])
print('\ndata1[0].get("href") ->\n', data1[0].get('href'))

data2=sp.select_one(".sister2")  # 取 class "sister2"
print("\ndata2->\n", data2)

data3=sp.select("#link1")   # 取 id "link1"
print('\ndata3->\n', data3)
```

In [ ]:
```
my_html = """
<!doctype html>
<html>
<head>
   <meta charset="utf-8">
   <title>洪錦魁著作</title>
   <style>
      h1#author { width:400px; height:50px; text-align:center;
             background:linear-gradient(to right,yellow,green);
      }
          h1#content { width:400px; height:50px;
                 background:linear-gradient(to right,yellow,red);
      }
      section { background:linear-gradient(to right bottom,yellow,gray); }
   </style>
</head>
<body>
<h1 id="author">洪錦魁</h1>
<img src="hung.jpg" width="100">
<section>
   <h1 id="content">一個人的極境旅行 - 南極大陸北極海</h1>
   <p>2015/2016年<strong>洪錦魁</strong>一個人到南極</p>
   <img src="travel.jpg" width="300">
</section>
<section>
   <h1 id="content">HTML5+CSS3王者歸來</h1>
   <p>本書講解網頁設計使用HTML5+CSS3</p>
   <img src="html5.jpg" width="300">
</section>
</body>
</html>
"""
```

In [ ]:
```
# ch5_8.py
import bs4

# htmlFile = myhtml.html', encoding='utf-8')
objSoup = bs4.BeautifulSoup(my_html, 'lxml')
objTag = objSoup.select('#author')
print("資料型態      = ", type(objTag))           # 列印資料型態
print("串列長度      = ", len(objTag))            # 列印串列長度
print("元素資料型態 = ", type(objTag[0]))        # 列印元素資料型態
print("元素內容      = ", objTag[0].getText())    # 列印元素內容
```

In [ ]:
```
# ch5_11.py
import bs4

objSoup = bs4.BeautifulSoup(my_html, 'lxml')
pObjTag = objSoup.select('p')
print("含<p>標籤的串列長度 = ", len(pObjTag))
for pObj in pObjTag:
    print(str(pObj))                # 內部有子標籤<strong>字串
    print(pObj.getText())           # 沒有子標籤
    print(pObj.text)                # 沒有子標籤
```

In [ ]:

```
Example = '''
<!DOCTYPE html>
<html lang="big5">
 <head>
  <meta charset="utf-8"/>
  <title>測試資料擷取的HTML網頁</title>
 </head>
 <body>
  <!-- Surveys -->
  <div class="surveys" id="surveys">
   <div class="survey" id="q1">
    <p class="question">
      <a href="http://example.com/q1">請問你的性別?</a></p>
    <ul class="answer">
     <li class="response">男 -
       <span class="score selected">20</span></li>
     <li class="response">女 -
       <span class="score">10</span></li>
    </ul>
   </div>
   <div class="survey" id="q2">
    <p class="question">
      <a href="http://example.com/q2">請問你是否喜歡偵探小說?</a></p>
    <ul class="answer">
     <li class="response">喜歡 -
       <span class="score">40</span></li>
     <li class="response">普通 -
       <span class="score selected">20</span></li>
     <li class="response">不喜歡 -
       <span class="score">0</span></li>
    </ul>
   </div>
   <div class="survey" id="q3">
    <p class="question">
      <a href="http://example.com/q3">請問你是否會程式設計?</a></p>
    <ul class="answer">
     <li class="response">會 -
       <span class="score selected">34</span></li>
     <li class="response">不會 -
       <span class="score">6</span></li>
    </ul>
   </div>
  </div>
  <div class="emails" id="emails">
    <div class="question">電子郵件清單資訊: </div>
    abc@example.com
    <div class="survey" data-custom="important">def@example.com</div>
    <span class="survey" id="email">ghi@example.com</div>
  </div>
 </body>
</html>
'''
```

In [ ]:
```
# 搜尋<title>標籤，和第3題問題的<div>標籤
# ch.5_3_2

from bs4 import BeautifulSoup

soup = BeautifulSoup(Example, "lxml")
# 搜尋<title>標籤和第3個<div>標籤
tag_title = soup.select("title")
print(tag_title[0].string)
tag_first_div = soup.find("div")
tag_div = tag_first_div.select("div:nth-of-type(3)")
print(tag_div[0])
```

In [ ]:
```
# 使用階層關係搜尋<title>標籤，然後搜尋<div>標籤下所有<a>子孫標籤
# ch.5_3_2a

from bs4 import BeautifulSoup

with open("html/Example.html", "r", encoding="utf8") as fp:
    soup = BeautifulSoup(fp, "lxml")
# 搜尋<title>標籤, 和<div>標籤下的所有<a>標籤
tag_title = soup.select("html head title")
print(tag_title[0].string)
tag_a = soup.select("body div a")
print(tag_a)
```

In [ ]:
```
# 搜尋特定標籤下的直接子標籤，可以同時使用nth-of-type()找出是第幾個標籤，也可以使用id
# ch.5_3_2b

from bs4 import BeautifulSoup

with open("Example.html", "r", encoding="utf8") as fp:
    soup = BeautifulSoup(fp, "lxml")
# 搜尋指定標籤下的直接子標籤
tag_a = soup.select("p > a")
print(tag_a)
tag_li = soup.select("ul > li:nth-of-type(2)")
print(tag_li)
tag_span = soup.select("div > #email")
print(tag_span)
```

In [ ]:

```
# 使用CSS選擇器「~」搜尋之後的所有兄弟標籤，「+」是只有下一個兄弟標籤，首先使用find()函
# ch.5_3_2c

from bs4 import BeautifulSoup

with open("Example.html", "r", encoding="utf8") as fp:
    soup = BeautifulSoup(fp, "lxml")
# 搜尋兄弟標籤
tag_div = soup.find(id="q1")
print(tag_div.p.a.string)
print("-----------")
tag_div = soup.select("#q1 ~ .survey")
for item in tag_div:
    print(item.p.a.string)
print("-----------")
tag_div = soup.select("#q1 + .survey")
for item in tag_div:
    print(item.p.a.string)
```

In [ ]:

```
# 搜尋指定class和id屬性值的HTML標籤，前2個select()函數是搜尋id屬性值q1，和<span>標
# ch.5_3_2d

from bs4 import BeautifulSoup

with open("Example.html", "r", encoding="utf8") as fp:
    soup = BeautifulSoup(fp, "lxml")
# 搜尋class和id屬性值的標籤
tag_div = soup.select("#q1")
print(tag_div[0].p.a.string)
tag_span = soup.select("span#email")
print(tag_span[0].string)
tag_div = soup.select("#q1, #q2")  # 多個id屬性
for item in tag_div:
    print(item.p.a.string)
print("-----------")
tag_div = soup.find("div")  # 第1個<div>標籤
tag_p = tag_div.select(".question")
for item in tag_p:
    print(item.a["href"])
tag_span = soup.select("[class~=selected]")
for item in tag_span:
    print(item.string)
```

In [ ]:
```
# 搜尋HTML標籤是否擁有指定屬性，或進一步指定屬性值來執行搜尋
# ch.5_3_2e

from bs4 import BeautifulSoup

with open("Example.html", "r", encoding="utf8") as fp:
    soup = BeautifulSoup(fp, "lxml")
# 搜尋特定屬性值的標籤
tag_a = soup.select("a[href]")
print(tag_a)
tag_a = soup.select("a[href='http://example.com/q2']")
print(tag_a)
tag_a = soup.select("a[href^='http://example.com']")
print(tag_a)
tag_a = soup.select("a[href$='q3']")
print(tag_a)
tag_a = soup.select("a[href*='q']")
print(tag_a)
```

In [ ]:
```
# select_one()函數和select()函數的使用方式相同，不過，此函數只會傳回符合的第1筆標籤，
# ch.5_3_2f

from bs4 import BeautifulSoup

with open("Example.html", "r", encoding="utf8") as fp:
    soup = BeautifulSoup(fp, "lxml")
# 使用select_one()方法搜尋標籤
tag_a = soup.select_one("a[href]")
print(tag_a)
```

5.7 爬取其它HTML文件

- 爬取項目清單文件
- 爬取自定義清單文件
- 爬取表格文件
- 爬取項目清單文件
- 爬取相鄰位置的節點(HTML標籤)

In [ ]:
```
ch5_2_1 = '''
<!doctype html>
<html>
<head>
   <meta charset="utf-8">
   <title>ch5_2_1.html</title>
</head>
<body>
<h1>台灣旅遊景點排名</h1>
<ol type="a">
   <li>故宮博物院</li><li>日月潭</li><li>阿里山</li>
</ol>
<h2>台灣夜市排名</h2>
<ol type="A">
   <li>士林夜市</li><li>永康夜市</li><li>逢甲夜市</li>
</ol>
<h2>台灣人口排名</h2>
<ol type="i">
   <li>新北市</li><li>台北市</li><li>桃園市</li>
</ol>
<h2>台灣最健康大學排名</h2>
<ol type="I">
   <li>明志科大</li><li>台灣體院</li><li>台北體院</li>
</ol>
</body>
</html>
'''
```

In [ ]:
```
# 爬取項目清單文件
# ch5_13_1.py
import requests, bs4

objSoup = bs4.BeautifulSoup(ch5_2_1, 'lxml')
titleobj = objSoup.find_all('h2')                # h2標題
print(titleobj[2].text)

itemobj = objSoup.find('ol', type='I')           # type='I'
items = itemobj.find_all('li')
for item in items:
    print(item.text)
```

In [ ]:
```
ch5_2_2 = '''
<!doctype html>
<html>
<head>
   <meta charset="utf-8">
   <title>ch5_2_2.html</title>
</head>
<body>
<h1>國家首都資料表</h1>
<dl>
   <dt>Washington</dt>
      <dd>美國首都</dd>
   <dt>Tokyo</dt>
      <dd>日本首都</dd>
   <dt>Paris</dt>
      <dd>法國首都</dd>
</dl>
</body>
</html>
'''
```

In [ ]:
```
# 爬取自定義清單文件
# ch5_13_2.py
import requests, bs4

objSoup = bs4.BeautifulSoup(ch5_2_2, 'lxml')

mycity = []
cityobj = objSoup.find('dl')
cities = cityobj.find_all('dt')
for city in cities:
    mycity.append(city.text)                # mycity串列

mycountry = []
countryobj = objSoup.find('dl')
countries = countryobj.find_all('dd')
for country in countries:
    mycountry.append(country.text)          # mycountry串列

print("國家 = ", mycountry)
print("首都 = ", mycity)
data = dict(zip(mycountry, mycity))
print(data)                                 # 字典顯示結果
```

In [ ]:
```
ch5_2_3 = '''
<!doctype html>
<html>
<head>
   <meta charset="utf-8">
   <title>ch5_2_3.html</title>
</head>
<body>
<table border="1">
   <thead><!--建立表頭 -->
      <tr><th colspan="3">聯合國水資源中心</th></tr>
      <tr><th>河流名稱</th><th>國家</th><th>洲名</th></tr>
   </thead>
   <tbody><!-- 建立表格本體 -->
      <tr><td>長江</td><td>中國</td><td>亞洲</td></tr>
      <tr><td>尼羅河</td><td>埃及</td><td>非洲</td></tr>
      <tr><td>亞馬遜河</td><td>巴西</td><td>南美洲</td></tr>
   </tbody>
   <tfoot><!-- 建立表尾 -->
      <tr><td colspan="3">製表2017年5月30日</td></tr>
   </tfoot>
</table>
</body>
</html>
'''
```

In [ ]:
```
# 爬取表格文件

# ch5_13_3.py
import requests, bs4
objSoup = bs4.BeautifulSoup(ch5_2_3, 'lxml')

myriver = []                            # 河川
tableobj = objSoup.find('table').find('tbody')
tables = tableobj.find_all('tr')
for table in tables:
    river = table.find('td')
    myriver.append(river.text)

mycountry = []                          # 國家
for table in tables:
    countries = table.find_all('td')
    country = countries[1]
    mycountry.append(country.text)

print("國家 = ", mycountry)
print("河川 = ", myriver)
data = dict(zip(mycountry, myriver))
print(data)                             # 字典顯示結果
```

5.8 爬取相對位置的節點(HTML標籤)

- find_next_sibling() # 下一個兄弟節點
- find_previous_sibling() # 上一個兄弟節點
- find_next_siblings() # 之後的所有兄弟節點
- find_previous_siblings() # 之前的所有兄弟節點
- parent() # 父節點

In [ ]:
```
import requests, bs4
objSoup = bs4.BeautifulSoup(ch5_2_3, 'lxml')

data1 = objSoup.find('td')
print(data1)

print('\nnext_sibling()->', data1.find_next_sibling())
print('\nnext_siblings()->', data1.find_next_siblings())
print('\n', data1.parent())
print('\n', data1.parent.find_next_sibling())
```

In [ ]:
```
# ch5_13_5.py
import requests, bs4

objSoup = bs4.BeautifulSoup(ch5_2_3, 'lxml')

myriver = []                                        # 河川
mystate = []                                        # 洲名
tableobj = objSoup.find('table').find('tbody')
tables = tableobj.find_all('tr')

for table in tables:
    countries = table.find_all('td')
    country = countries[1]                          # 國家節點
    river = country.find_previous_sibling('td') # 前一個節點
    myriver.append(river.text)
    state = country.find_next_sibling('td')     # 下一個節點
    mystate.append(state.text)

print("洲名 = ", mystate)
print("河川 = ", myriver)
data = dict(zip(mystate, myriver))
print(data)                                         # 字典顯示結果
```

5.9 網頁資料的修改

In [ ]:
```
# 更改HTML標籤名稱和屬性：Ch5_6.py

from bs4 import BeautifulSoup

soup = BeautifulSoup("<b class='score'>Joe</b>", "lxml")
tag = soup.b
tag.name = "p"
tag["class"] = "question"
tag["id"] = "name"
print(tag)
del tag["class"]
print(tag)
```

In [ ]:
```
# 修改HTML標籤的文字內容：Ch5_6a.py

from bs4 import BeautifulSoup

soup = BeautifulSoup("<b class='score'>Joe</b>", "lxml")
tag = soup.b
tag.string = "Mary"
print(tag)
```

In [ ]:
```
# 新增HTML標籤和文字內容：Ch5_6b.py

from bs4 import BeautifulSoup
from bs4.element import NavigableString

soup = BeautifulSoup("<b></b>", "lxml")
tag = soup.b
tag.append("Joe")
print(tag)
new_str = NavigableString(" Chen")
tag.append(new_str)
print(tag)
new_tag = soup.new_tag("a", href="http://www.example.com")
tag.append(new_tag)
print(tag)
```

In [ ]:
```
# 插入HTML標籤和清除標籤內容：Ch5_6c.py

from bs4 import BeautifulSoup

soup = BeautifulSoup("<p><b>One</b></p>", "lxml")
tag = soup.b
new_tag = soup.new_tag("i")
new_tag.string = "Two"
tag.insert_before(new_tag)
print(soup.p)
new_string = soup.new_string("Three")
tag.insert_after(new_string)
print(soup.p)
tag.clear()
print(soup.p)
```

In [ ]:
```
# 取代HTML標籤：Ch5_6d.py

from bs4 import BeautifulSoup

soup = BeautifulSoup("<p><b>One</b></p>", "lxml")
tag = soup.b
new_tag = soup.new_tag("i")
new_tag.string = "Two"
tag.replace_with(new_tag)
print(soup.p)
```

6. 正規化表示式：regular expression

參考我的正規表示式

https://www.tutorialspoint.com/python3/python_reg_expressions.htm

https://docs.python.org/3.6/library/re.html?highlight=regular%20expression

In [ ]:
```
- 「正規運算式」（Regular Expression）是一個範本字串，可以用來進行字串比對，
- 在正規運算式的範本字串中，每一個字元都擁有特殊意義，這是一種小型的字串比對語言。
- 正規運算式直譯器或稱為引擎能夠將定義的正規運算式範本字串和字串變數進行比較，引擎傳回布林
- True表示字串符合範本字串的定義；False表示不符合。
```

測試正規表示式 的網站:

pythex

整數數字要用正規表達式描述，可以用 [0123456789]+ 其中的中括號 [與] 會框住一群字元，用來代表字元群，加號 + 所代表的是重複 1 次或以上， 因此，該表達式就可以描述像 323421 這樣的數字。

在正規表達式中，為了更方便撰寫，可以用 [0-9]+ 表達同樣的概念， 其中的 0-9 其實就代表了 0123456789 等字元。甚至可以再度縮短，以 [\d]+ 代表， 其中的 \d 就代表數字所成的字元集合。

Special characters

\ escape special characters
. matches any character
^ matches beginning of string
$ matches end of string
[5b-d] matches any chars '5', 'b', 'c' or 'd'
[^a-c6] matches any char except 'a', 'b', 'c' or '6'
R|S matches either regex R or regex S
() creates a capture group and indicates precedence

Quantifiers

- 0 or more (append ? for non-greedy)
- 1 or more (append ? for non-greedy)
  ? 0 or 1 (append ? for non-greedy)
  {m} exactly mm occurrences
  {m, n} from m to n. m defaults to 0, n to infinity
  {m, n}? from m to n, as few as possible

Special sequences

\A start of string
\b matches empty string at word boundary (between \w and \W)
\B matches empty string not at word boundary
\d digit
\D non-digit
\s whitespace: [\t\n\r\f\v]
\S non-whitespace
\w alphanumeric: [0-9a-zA-Z_]
\W non-alphanumeric
\Z end of string
\g matches a previously defined group

Special sequences

(?iLmsux) matches empty string, sets re.X flags
(?:...) non-capturing version of regular parentheses
(?P...) matches whatever matched previously named group
(?P=) digit
(?#...) a comment; ignored
(?=...) lookahead assertion: matches without consuming
(?!...) negative lookahead assertion
(?<=...) lookbehind assertion: matches if preceded
(?<!...) negative lookbehind assertion
(?(id)yes|no) match 'yes' if group 'id' matched, else 'no'

常用之正規表示法範例

- 開頭要1~9
  [1-9]

- 0~9的數字出現 0個或多個
  \d*

- 西元生日格式
  /^\d{4}-\d{2}-\d{2}$/
  範例："1996-08-06"

- 身分證字號
  /^[A-Z]\d{9}$/
  範例："A123456789"

- 手機號碼
  /^09\d{8}$/
  範例："0912345678"

- 不包含小寫母音的字串
  範例：/^[^aeiou]*$/
  "hEllO","ApplE"

- gmail 信箱
  /^.*@gmail\.com$/
  範例："test@gmail.com"

- email信箱
  /^\w+((-\w+)|(\.\w+))*\@[A-Za-z0-9]+((\.|-)[A-Za-z0-9]+)*\.[A-Za-z0-9]+$/

- 四則運算算式
  /^[0-9\+\-*\/]*$/
  範例："1+2*3"

- 價格 : 0 或是正整數
  /^(0|[1-9]\d*)$/

- 網址
  /((http[s]{0,1}|ftp):\/\/[a-zA-Z0-9\.\-]+\.([a-zA-Z]{2,4})(:\d+)?(\/[a-zA-Z0-9\.\-~!@#$%^&*+?:_\/=<>]*)?)/g
  或是
  ((http[s]{0,1}|ftp):\/\/[a-zA-Z0-9\.\-_\/]+\.([a-zA-Z]{2,4})(:\d+)?(\/[a-zA-Z0-9\.\-~!@#$%^&*+?:_\/=<>]*)?)

- 載入 re 模組
- 搜尋字串有兩種使用方法：
  1. 回傳結果物件 = re.方法(正規表示式,搜尋字串)
     regex = r"([a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+.[a-zA-Z0-9-.]+)"
     emails = re.findall(regex,要搜尋的內容)
  2. 回傳結果物件 = 正規表示法物件.方法(搜尋字串)
     regex = re.compile('[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+.[a-zA-Z0-9-.]+')
     emails = regex.findall(要搜尋的內容)
- 搜尋 beautifulsoup物件
  - re.compile('正規表示式')
  - 例如 sp.find*all(class*=re.compile('sister'))

In [ ]:
```
# ch05
import re
pat = re.compile('[a-z]+')

m = pat.search('3tem12po')
print(m) # <_sre.SRE_Match object; span=(1, 4), match='tem'>

if not m==None:
    print(m.group())  # tem
    print(m.start())  # 1
    print(m.end())    # 4
    print(m.span())   # (1,4)
```

In [ ]:
```
# ch05

import re
pat = re.compile('[a-z]+')

m = pat.findall('tem12po')
print(m)  # ['tem', 'po']
```

In [ ]:
```
# ch5_7_5.py
import bs4
import re

htmlFile = "<h1 class='boldtext'>深智數位</h1>"
objSoup = bs4.BeautifulSoup(htmlFile, 'lxml')
tag = objSoup.find('h1', class_=re.compile('text'))
print(tag)
print(tag.text)
```

In [ ]:
```python
# 結合findall 搜尋非字串的 beautifulsoup物件
# 搜尋所有sister開頭的class,包含 sister1 & sister2

import re
from bs4 import BeautifulSoup
sp = BeautifulSoup(html_doc,'lxml')

# reg = re.compile('sister')
class_sister = sp.find_all(class_=re.compile('sister'))

# print(sp)
print(class_sister)
```

In [ ]:
```python
#ch05
# 以字串的方式進行搜尋，非搜尋beautifulsoup物件

import requests,re
regex = re.compile('[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+')
url = 'https://www.csie.ndhu.edu.tw/professors-zh_tw/'
html = requests.get(url)
emails = regex.findall(html.text)
for email in emails:
    print(email)
```

In [ ]:
```python
# _*_ coding: utf-8 *_*
# 程式 9-4 (Python 3 version)

import requests, re

regex = r"([a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+)"
url = 'https://www.csie.ndhu.edu.tw/professors-zh_tw/'

html = requests.get(url).text

emails = re.findall(regex,html)
for email in emails:
    print(email)
```

7. 使用 get 或 post 來傳遞參數

- httpbin提供HTTP請求/回應的測試服務，類似Echo服務，可以將我們送出的HTTP請求，
- 自動將送出的請求資料以JSON格式回應，支援HTTP方法GET和POST等，其網址是：http://httpbin.org

7.1 使用get 傳遞參數

- 語法
  - requests.get(url, params={key: value}, args)
- 常用參數
  - params = 字典 # 送出RESTful API的HTTP請求
    - 即網址後的查詢參數
    - requests.get(網址?參數1=值&參數2=值&參數3=值) # 參數會以&隔開
  - headers # 一般用來傳遞 user-agent
  - auth = (帳號, 密碼)
  - timeout # 指定等待的回應時間不超過timeout參數的時間
  - cookies
- 詳細參數的參考資料
  - https://www.w3schools.com/python/ref_requests_get.asp

7.1.1 params 參數

- requests.get(url, params = 字典)
  - 送出RESTful API的HTTP請求
  - 即網址後的查詢參數
  - requests.get(網址?參數1=值&參數2=值&參數3=值) # 參數會以&隔開

In [ ]:
```python
# request.get() 的 params 參數

import requests

# 方法一
html = requests.get("https://www.google.com.tw/search?q=5g&tbm=isch")

# 方法二
data = {'q': '火星文', 'tbm': 'isch'}
html = requests.get("https://www.google.com.tw/search?", params=data)

print(html.text)
```

In [ ]:
```python
# 使用get 傳參數

# python 大數據特訓班 鄧文淵 chap.2
import requests
payload = {'key1': 'value1', 'key2': 'value2'}
# 將查詢參數加入 GET 請求中
html = requests.get("http://httpbin.org/get", params=payload)
print(html.url) # http://httpbin.org/get?key1=value1&key2=value2
```

In [ ]:
```
import requests

# data = {'name': '陳會安', 'score': 95}
# r = requests.get("https://www.google.com.tw/search?q=5g+AI", params=data
r = requests.get("https://www.google.com.tw/search?q=5g+AI")
print(r.text)
```

In [ ]:
```
# 使用get 傳參數

# python 大數據特訓班 鄧文淵 chap.2
import requests
payload = {'q': '火星文', 'tbm': 'isch'}
# 將查詢參數加入 GET 請求中
html = requests.get("http://www.google.com/search", params=payload)
print(html.url)
print("*" * 50)
print(html.text) # http://httpbin.org/get?key1=value1&key2=value2
```

7.1.2 爬蟲的偽裝：user-agent

- 使用網頁時傳送的 user-agent
  - 在開發者工具中點選 Network標籤，之後重新整理網頁，再點選Name中的第一筆資料，再點選 Header標籤，即可再下方看到 user-agent
  - 或是從以下網頁可以看到瀏覽器傳送出去的user-agent
    - http://httpbin.org/user-agent
- 使用爬蟲時傳送的user-agent
  - 如下程式，所以可以看出不同
  - 所以若是需要要，網站伺服器是可以擋掉網路爬蟲
  - 爬蟲若是被此伺服器擋住，可以做以下處理
- 語法
  - 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'}
  - r = requests.get(url, headers=url_headers)

In [ ]:
```
import requests
url = 'http://httpbin.org/user-agent'
html = requests.get(url)
html.encoding="utf-8"
print(html.text)
```

In [ ]:
```
# 爬蟲的偽裝 - 加上瀏覽器的user-agent
url = "http://httpbin.org/user-agent"
r = requests.get(url)
print(r.text)
print("----------------------")
url_headers = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) Ap
r = requests.get(url, headers=url_headers)
print(r.text)
```

7.1.3 傳遞帳號密碼

requests.get(url, auth = (帳號, 密碼))

In [ ]:
```
import requests

url = "https://api.github.com/user"

r = requests.get(url, auth=('hueyan@ms2.hinet.net', '********'))
if r.status_code == requests.codes.ok:
    print(r.headers['Content-Type'])
    print(r.json())
else:
    print("HTTP請求錯誤...")
```

In [ ]:
```

```

7.1.4 timeout()

- 設定連線的時間，以免佔用太多的server連線時間

In [ ]:
```
import requests

try:
    r = requests.get("http://www.google.com", timeout=0.03)
    print(r.text)
except requests.exceptions.Timeout as ex:
    print("錯誤: HTTP請求已經超過時間...\n" + str(ex))
```

模擬人類行為，間隔時間後再送出請求

- 網頁爬蟲很可能需要在很短時間內，針對同一網站密集的送出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，目前的網站大都有預防密集請求的機制。
- 使用 time.sleep(秒數)

In [ ]:
```
import time
import requests

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

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

7.1.5 存取 cookie

- Cookies英文原義是小餅乾
- 可以在瀏覽器保留使用者的瀏覽資訊
- Cookies是儲存在瀏覽器電腦，不會浪費Web伺服器的資源

In [ ]:
```
# 取得 cookie
# 如果HTTP請求的回應內容有Cookie，我們可以使用cookies屬性來取出Cookie值
# 此網站沒有，只是顯示格是

r = requests.get("http://example.com/")
v = r.cookies["cookie_name"]
print(v)
```

In [ ]:
```
# 傳送 cookie
# 送出Cookie的HTTP請求

url = "http://httpbin.org/cookies"
cookies = dict(name='Joe Chen')
r = requests.get(url, cookies=cookies)
print(r.text)
```

7.2 requests.post( ) 傳遞參數

In [ ]:
```
# 使用 post

# python 大數據特訓班 鄧文淵 chap.2
import requests
payload = {'key1': 'value1', 'key2': 'value2'}
# 將查詢參數加入 POST 請求中
html = requests.post("http://httpbin.org/post", data=payload)
print(html.text)
```

8. 其它

開發人員工具 Developer part 2

* Network：可以看要求HTML的request和response的訊息
* Network / all HTML的request和response的所有訊息
* Network / XHR 看部份網頁更新時使用的傳遞訊息

網頁內使用局部改變網頁內容，但未改變網址的抓取方式

In [ ]:

使用chrome的XHR抓取每日盤後成交資訊 (參考：碁峰 大數據學習特訓班 ch.07)

步驟：
1. 以chrome開啟網頁 (如證交所的日成交資訊：https://www.twse.com.tw/zh/page/tradi
2. 按F12 (或是在網頁上按右鍵選「檢視」) 開啟開發者工具，選擇Network / XHR
3. 在網頁中選擇股票及要觀看的時間 (例如台積電，2018年10月)
4. 在開發者工具中選取[STOCK_DAY?response=json&...]，複製Request URL:後面的網址
5. 在Chrome中開啟新的頁面，貼上網址後按ENTER，即可看到json格式的盤後資訊

檢查檔案是否更新

In [ ]:

```
'''
方法：將md5碼儲存起來，定時建立新的md5碼，並和已儲存的md5碼比較，
若是不相同表示網頁已有更新，則需要重新抓取網頁
'''
# hashlib package


import hashlib
```

In [ ]:

```
#ch06
import hashlib,os,requests

url = "http://opendata.epa.gov.tw/ws/Data/REWXQA/?\
$orderby=SiteName&$skip=0&$top=1000&format=json"

# 讀取網頁原始碼
html=requests.get(url).text.encode('utf-8-sig')
# 判斷網頁是否更新
md5 = hashlib.md5(html).hexdigest()
if os.path.exists('old_md5.txt'):
    with open('old_md5.txt', 'r') as f:
        old_md5 = f.read()
    with open('old_md5.txt', 'w') as f:
        f.write(md5)
else:
    with open('old_md5.txt', 'w') as f:
        f.write(md5)

if md5 != old_md5:
    print('資料已更新...')
else:
    print('資料未更新，從資料庫讀取...')
```

In [ ]:

```
# old_md5.text

1dbdca789f60e93e236550d24810d78d
```

In [ ]:

```python
# ch06 pm25.py

import sqlite3,ast,hashlib,os,requests
from bs4 import BeautifulSoup

conn = sqlite3.connect('DataBasePM25.sqlite') # 建立資料庫連線
cursor = conn.cursor() # 建立 cursor 物件

# 建立一個資料表
sqlstr='''
CREATE TABLE IF NOT EXISTS TablePM25 ("no" INTEGER PRIMARY KEY AUTOINCREMEI
NOT NULL UNIQUE ,"SiteName" TEXT NOT NULL ,"PM25" INTEGER)
'''
cursor.execute(sqlstr)

url = "http://opendata.epa.gov.tw/ws/Data/REWXQA/?\
$orderby=SiteName&$skip=0&$top=1000&format=json"
# 讀取網頁原始碼
html=requests.get(url).text.encode('utf-8-sig')

# 判斷網頁是否更新
md5 = hashlib.md5(html).hexdigest()
old_md5 = ""

if os.path.exists('old_md5.txt'):
    with open('old_md5.txt', 'r') as f:
        old_md5 = f.read()
with open('old_md5.txt', 'w') as f:
    f.write(md5)

if md5 != old_md5:
    print('資料已更新...')
    sp=BeautifulSoup(html,'html.parser')
    # 將網頁內轉換為 list,list 中的元素是 dict
    jsondata = ast.literal_eval(sp.text)
    # 刪除資料表內容
    conn.execute("delete from TablePM25")
    conn.commit()

    n=1
    for site in jsondata:
        SiteName=site["SiteName"]
        PM25=0 if site["PM2.5"] == "" else int(site["PM2.5"])
        print("站名:{}   PM2.5={}".format(SiteName,PM25))
        # 新增一筆記錄
        sqlstr="insert into TablePM25 values({},'{}',{})" .format(n,SiteNar
        cursor.execute(sqlstr)
        n+=1
        conn.commit() # 主動更新
else:
    print('資料未更新，從資料庫讀取...')
    cursor=conn.execute("select *  from TablePM25")
    rows=cursor.fetchall()
    for row in rows:
        print("站名:{}   PM2.5={}".format(row[1],row[2]))

conn.close()  # 關閉資料庫
```

In [ ]:

工作排程自動下載

In [ ]:

In [ ]:
```
# ch06 pm25_autorun.py

import sqlite3,ast,requests,os
from bs4 import BeautifulSoup

cur_path=os.path.dirname(__file__) # 取得目前路徑
conn = sqlite3.connect(cur_path + '/' + 'DataBasePM25.sqlite') # 建立資料庫連
cursor = conn.cursor() # 建立 cursor 物件

# 建立一個資料表
sqlstr='''
CREATE TABLE IF NOT EXISTS TablePM25 ("no" INTEGER PRIMARY KEY AUTOINCREMEN
NOT NULL UNIQUE ,"SiteName" TEXT NOT NULL ,"PM25" INTEGER)
'''
cursor.execute(sqlstr)

url = "http://opendata.epa.gov.tw/ws/Data/REWXQA/?\
$orderby=SiteName&$skip=0&$top=1000&format=json"
# 讀取網頁原始碼
html=requests.get(url).text.encode('utf-8-sig')

print('資料已更新...')
sp=BeautifulSoup(html,'html.parser')
# 將網頁內轉換為 list,list 中的元素是 dict
jsondata = ast.literal_eval(sp.text)
# 刪除資料表內容
conn.execute("delete from TablePM25")
conn.commit()

n=1
for site in jsondata:
    SiteName=site["SiteName"]
    PM25=0 if site["PM2.5"] == "" else int(site["PM2.5"])
    print("站名:{}   PM2.5={}".format(SiteName,PM25))
    # 新增一筆記錄
    sqlstr="insert into TablePM25 values({},'{}',{})" .format(n,SiteName,PM
    cursor.execute(sqlstr)
    n+=1
    conn.commit() # 主動更新

conn.close()  # 關閉資料庫
```

In [ ]:

參考資料：

- 爬蟲教學網站:
  - 大數學堂 http://www.largitdata.com/course_list/
- python 初學特訓班，碁峰 chap.5-6
- python 程式設計實務：從出學到活用(16堂課） chap.9-10
- WebDriver API官網
- Python大數據特訓班：鄧文淵 chap.2.3
- 資料科學學習手札33:基於Python的網路資料採集實戰（1）
  - 連結
- 數據科學學習手札47:基於Python的網絡數據採集實戰（2）
  - 連結
- 資料科學學習手札50: 基於Python的網路資料採集-selenium篇（上）
  - 連結
- Python 資料科學與人工智慧（陳允傑，旗標） ch.4-ch.7 （有投影片，資料很詳細）
- 洪錦魁（2019）。Python網路爬蟲王者歸來–大數據擷取、清洗、儲存與分析。台北：旗標。