**觀看網頁的HTML:**  view-source:網址

**HTML三要素 :** <tag attribute=“something”> content </tag>



**標籤通常成對出現 :** <開始標籤>……….<**/**結束標籤>

**\*除了換行 :** <br>

**放在<head></head>內的標籤 :**

* <title> 標題 </title>
* <link CSS >
* <網頁大小>

**放在<body></body>內的標籤 :**

* <div> 大區塊 </div>
* <p> 區塊裡的段落 </p> #要縮排
* <h1~6> 標題(數字越大字越小) </h1~6> #不用<br>即可換行
* <font color=?、size=? 字體=?> 文字 </font>
* <u></u> : 底線
* <i></i> : 斜體
* <b></b> : 粗體
* <img src="./img.jpg"> 圖檔

**CSS :**

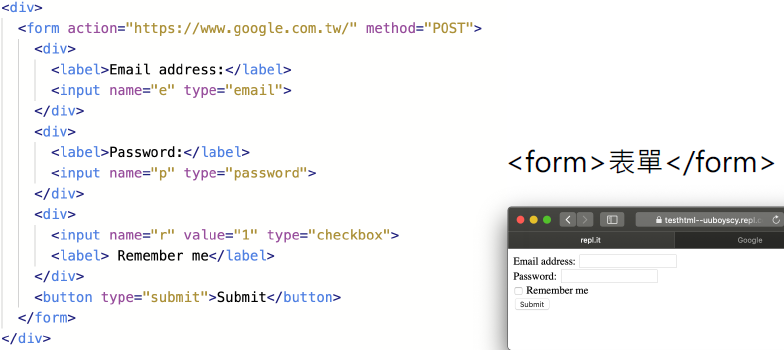
* 為HTML定義樣式（字型、顏色…）
* 語法 :

1. **在style.css頁面定義 :** 選擇器(div、p、.、#…).CSS名稱{CSS語法}
2. **在HTML頁面使用 :**
   * + 1. 在選擇器開始標籤定義class=”CSS名稱”
       2. 或可匯入模板<link href=<https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css>>
       3. 若要用多個CSS,就再KEY一行即可
       4. 多個CSS有相同選擇器，則以最後一行為主

**製作表格**



**製作表單**



**URL結構 :** http(通訊協定)://主機位址(IP)/檔案路徑(…/...)?參數1=XX&參數2=YY

**Http Method**

* **GET :** 有明確網址,按下ENTER即可造訪網頁
* **POST :** 需要送出表單,不管填了什麼送出後網址都一樣,因為我們送出的參數被隱藏了

**Cookie :** 為了辨別用戶身分而儲存在用戶端上的資料

**開發人員工具 （網頁按F12）**

1. **ELEMENT(可觀看HTML)**

**選擇器(點下某一行後最下面出現的,會簡化秀出所有標籤) :**

* id 會被簡化成 **#**
* class 會被簡化成 **.**

1. **NETWORK(可觀看訪問伺服器過程產生的檔案)**

* **General :** 我們訪問此網頁的方式
* **Response Headers :** 網頁伺服器在我們身上綁定的資料
* **Request Headers :** 我們的瀏覽器身份資料
* **Query String Parameters :** 參數
* **Form data :** post時表單輸入的內容

**爬蟲套件**

* **Requests**(第三方套件)和**Urllib**(內建)皆可對網頁提出request
* **BeautifulSoup**(第三方套件)進一步轉換html為物件

**Requests**

* import requests (提出請求)

from urllib import parse (把表單內容轉成字串)

URL=”網址”

#去NETWORK找到Request Headers最下面我們的User-Agent並複製

Head = {‘User-Agent’ : ” 複製文字 ” }

>>> ’ : ”後的空格一定要刪掉!!!!不然會失敗

POST的表單內容 = { ‘key1’ : ’value1’ , ‘key2’ : ’value2’ } >>> 在Form data

>>>通常很大一串,要用迴圈和split()的方式把複製的字串轉換成字典

Cookies = { ‘cookies內容’ : ’名稱’ }

>>> 有cookies的話,可從網址旁邊的鎖頭找到內容和名稱

response = requests.get或post(URL , data = 表單內容, headers=Head ,cookies=Cookies)

>>>不用解碼,但print要加”.text”

**Urllib**

* from urllib import request, parse

URL = ”網址”

Head = {‘User-Agent’ : ” 複製文字 ” , ‘Cookies’ : “cookies內容=名稱” }

表單內容 = { ‘key1’ : ’value1’ , ‘key2’ : ’value2’ }

表單內容 = bytes(parse.urlencode(表單內容) , encoding = 'utf8' )

Req= request.Request(URL , data = 表單內容, headers = Head)

>>>請求認可身分

response=request.urlopen(Req).read().decode('utf8')

>>>要解碼,不然出現的訊息有的會是b(byte)

>>> 會被轉換成JSON

**BeautifulSoup**

* from bs4 import BeautifulSoup

Soup = BeautifulSoup (response.text, ”html.parser” ) ---**Requests**

Soup = BeautifulSoup (response, ”html.parser” ) --- **Urllib**

print ( Soup.prettify() )

>>> 會輸出為整齊的HTML

>>> 表單內容會顯示於”form”底下

* 想找的大標籤 = Soup.findAll (‘ 標籤名如div ’, 屬性如ID= ’ ?? ’ )

或 想找的大標籤 = Soup.findAll (‘ div ’, {“ID” : “??” })

或 想找的大標籤 = Soup.select (‘ div [ID= “ ?? “] td [align= “ ?? “]’) >>>特定標籤底下的標籤

或 想找的大標籤 = Soup.select (‘ div#?? ’)

# find和select\_one只會找出第一個

# findAll和select會列出所有符合條件的標籤在一個list

>>> print會顯示整個標籤下的樹狀結構

* 進一步分析標籤內的標籤 = 想找的大標籤[ list內第幾個位子 ].find ( ‘標籤名’ )

或 print( 想找的大標籤[ list內第幾個位子 ].標籤名 )

>>> print只會找出第一個標籤

標籤內的第二個標籤 = 想找的大標籤[ list內第幾個位子 ].標籤名.next\_sibling

\*標籤內的第二個標籤 = 想找的大標籤[ list內第幾個位子 ].標籤名.next\_siblings

>>> 則會顯示接下來所有的標籤的generator,要用for迴圈print出來

* 標籤中間夾雜的文字 = 進一步分析標籤內的標籤.text或string

屬性的value = 進一步分析標籤內的標籤[‘屬性’] >>>當字典用

>>> 如果是查網址,print出來都會缺少"http://主機位址",要自己在print時加上去

* 前後幾頁也想一起爬 :

1. 觀看網址有無規律性,先設一個變數,並用”%S”取代URL內的文字,在URL最後接上%(變數)
2. 或是SELECT出”上一頁”的按鈕標籤內的網址,作為新的URL
3. 以迴圈完成,必要時最後加上TRY&EXCEPT以免例外情形(如: 連結已被刪)
4. 可適時隨機等待時間,防止被識別為程式 :

time.sleep(random.randint(3,6))

* 進入文章並存取文字檔 :

1. 用OS模組創建資料夾
2. 善用.split(' ')[?]區分文字
3. 必要時最後加上TRY&EXCEPT以免文字格式不符

* 下載圖片

1. from urllib import request

request.urlretrieve（圖片網址,存放目錄）

* **Session** :

1. 也是一種連線請求方式,像是一個使用者,可一直攜帶著cookies,就不用在造訪下一頁時又要再輸一次cookies資料
2. ss = requests.session()

ss.cookies[ ‘cookies內容’ ] = ’ cookies名稱’

res = ss.get(URL , headers = Head)

1. 用爬蟲造訪含cookies的網頁的方法 :
2. 先用get方式造訪接受前的網址
3. 在element裡面,找到接受cookies的按鈕的html,會說明要input一些hidden的資訊,以及button submit出去的資訊,這些都是詢問cookies的post data,他們的name跟value要select出來放進字典

<form action="/ask/over18" method="post">

<input type="hidden" name="from" value="/bbs/Gossiping/index.html">

<button class="btn-big" type="submit" name="yes" value="yes">)

1. 而form就代表詢問cookies的表單網址,需要select出action的內容並跟"http://主機位址"組合成完整的網址
2. 再次用post的方式,夾帶剛剛的post data造訪被詢問cookies的網址
3. 再用get方式造訪接受後的網址(此時已不用攜帶post data)

**動態網頁**

* 比如一個網頁滑到最底部會自動產生新內容(EX: FB),或是即時更新的匯率網站,因為觸發了特定條件,瀏覽器自動幫我們用GET或POST提出request,並獲得**JSON資料結構**回傳
* request的網址可在network的XHR找到疑似的request URL及form data,其中的參數可以視我們需要爬取的量再修改
* 用此URL得到的respond為**JSON資料結構**,要轉換成字典(每個字典即是新的文章),觀察其中的key和value,可能會發現到跟文章網址有關聯的ID,就可以更改URL爬取更大量文章標題或圖片了

**Selenium**

* 可以利用程式腳本操作瀏覽器(像開外掛)，程式執行後會自動開啟一個Chrome瀏覽器
* 語法

1. pip install selenium
2. 更新版本(限COLAB或LINUX) :

!apt-get update

!apt install chromium-chromedriver

1. from selenium.webdriver import Chrome
2. 設定無頭模式參數(不想顯示螢幕的話)(但限colab)

chrome\_options = webdriver.ChromeOptions()  >>> 啟用設定

chrome\_options.add\_argument('--headless') >>> 增加參數: 啟用無頭模式

chrome\_options.add\_argument('--no-sandbox')  >>> 增加參數: 以最高權限執行

1. chrome\_options.add\_argument("user-agent="+UserAgent)  >>> 增加參數: 如果有UserAgent的話
2. **方法一 :** driver = Chrome(options=chrome\_options) # 有設定參數再打

**方法二 :** driver = Chrome('./chromedriver') >>>要用自己下載的chromedriver的話再打(需存放在與python檔同路徑!!)

1. driver.get(url)
2. 開始使用網頁內容

**方法一 :** soup = BeautifulSoup(driver.page\_source, 'html.parser')

**方法二(模擬手動) :**

driver.find\_element\_by\_class\_name(' ??? ').click() >>>定位按鈕標籤位置並點擊

driver.find\_element\_by\_tag\_name(' ??? ').send\_keys(' 攝影 ') >>>找到搜尋的標籤並在表單輸入內容

driver.find\_element\_by\_xpath(' XPath複製內容 ').click() >>>前往頁面(直接在標籤的部分點右鍵copy XPath)

time.sleep(5) >>>怕跑太快可以睡一下

1. cookie = driver.get\_cookies() >>>得到cookies

for c in cookie:

ss.cookies.set(c['name'], c['value']) >>>設定cookies到ss後接下來就可以進行後續操作(爬html等)

1. driver.close() >>>關掉瀏覽器