Python 程式設計

林奇賦　daky1983@gmail.com

Outline

- 網頁抓取與解析
- Requests
- BeautifulSoup

網路爬蟲 (Crawler)

- 網路爬蟲 (Web Crawler) 的應用，最早起源於 Google 搜尋引擎的誕生，算是個古老又貪婪的技術。門檻相當低，只要能送出 HTTP Request 加上正規表示法(Regular Expression) 將網頁原始碼中的資訊解析出來，就算是具備基本的爬蟲功能。
- 爬蟲技術所能創造的商機，當然不僅僅是搜尋引擎而已，像 Google 其實還得再加上搜尋技術才算是真正建立起進入門檻。在 Google 確立了搜尋引擎霸主的地位後，網路爬蟲專家們逐一放棄了將網路上所有的資訊爬下來的野心，轉往利基市場如**比價系統(FindPrice)**、**即時資訊**、**或非web-based的爬蟲 (如 telnet)**，有的則是站在 Google 巨人的肩膀上，從搜尋結果中再爬出更有價值的資訊，將 search engine 當作是爬蟲中的一個子功能加以利用。
- 爬蟲技術的目的在於：**"把別人的資料庫都變成我的資料庫"**

Requests

- HTTP for Humans 給人用的HTTP函式庫
- 在網絡編程中，最最基本的任務包含：
  - 發送請求
  - 登錄
  - 獲取數據
  - 解析數據
  - 反序列化列印內容

安裝與基本用法

- 安裝： pip install requests
- 基本用法：
- import requests
  - requests.get()
  - requests.post()
  - requests.put()
  - requests.delete()
  - requests.patch()

URL 傳參值 / 獲取請求的 URL

```python
import requests

cs_url = 'http://www.so.com/s'
param  = {'ie':'utf-8', 'q':'query'}

r = requests.get(cs_url, params = param)
print(r.url)
```

印出結果： http://www.so.com/s?q=query&ie=utf-8

Timeout 設定

- requests 的超時設置以秒為單位。例如，對請求加參數 timeout = 5 即可設置超時為 5 秒。

```
# a very short timeout is set intentionally
import requests

cs_url = 'http://www.appledaily.com.tw/'
r = requests.get (cs_url, timeout = 0.000001)
```

GET

- GET就是指在網址上指定變數的名稱及變數的值給網 頁伺服器，使用GET是有一個上限的所以較不適合用 來傳送大量的資料或訊號。
- 一個簡單的GET就是在網址尾部加上一個問號？之後 進行宣告傳送的變數。而傳送的格式是：” 變數名稱 = 變數的值” 。
- 若有多個變數需要傳遞則用 & 符號隔開。

Request header

- 藉由 r.request.headers 得到發出要求的標頭

- 如果要改變發出的 header
- my_headers = {‘User-Agent’ : User-Agent值, ’Accept-Encoding‘ : 值 }
- r = requests.get(url, headers = my_headers)

字節模式 / 自動解包

- 網絡上傳輸的數據，很多情況下都是經過壓縮的。經由 requests 發送的請求，當收到的響應內容經過 gzip 或 deflate 壓縮時，requests 會自動為我們解包。我們可以用 Response.content 來獲得以字節形式返回的相應內容

抓取統一發票網頁範例

- EX09_01.py

POST

- POST方法是將要傳送的資訊放在message-body中
- 使用POST方法就不用擔心資料大小的限制，可以防止 使用者操作瀏覽器網址，表單的資料被隱藏在 message-body中，因此，在大多數的情況下，使用 POST方法將表單資料傳到Web Server端是更有效的方法。

POST 表單

- POST 方法可以將一組用戶數據，以表單的形式發送到遠端服務器。遠端服務器接受後，依照表單內容做相應的動作。
- 調用 requests 的 POST 方法時，可以用 data 參數接收一個 Python 字典結構。requests 會自動將 Python 字典序列化為實際的表單內容。例如：

```
import requests
cs_url    = '網址'
my_data   = {
    'key1' : 'value1',
    'key2' : 'value2'
}
r = requests.post(cs_url, data = my_data)
```

POST範例 捷運站查詢

- 利用Chrome瀏覽器的內建Network功能觀察 POST 的參數有哪些
- 將 POST 的資料使用 python 打包後送至動態網頁中
- EX09_02.py

BeautifulSoup

- 安裝：
  - pip install BeautifulSoup4
- 載入：
  - from bs4 import BeautifulSoup
- 使用方法：
  - 用BeautifulSoup類別，傳入欲解析的網頁原始碼，即可產生出 BeautifulSoup 的物件，藉由此物件呼叫需要使用的方法即可快速找到所需要的資料。

BeautifulSoup

- 使用soup.標籤名稱可以獲得html標籤中第一個符合的標籤內容：
  - Ex: soup.span 會得到 統一發票號碼獎中獎號碼
- 使用.string 或是 呼叫get_text()函式都可以獲得標籤內的內容：
  - Ex: soup.span.string 會得到 統一發票號碼獎中獎號碼
  - Ex: soup.span.get_text() 得到結果同上
- 如果有多筆相同名稱的標籤都需要取得內容時，可以使用findAll() 函式來達到目的：
  - Ex: soup.findAll('span',{'class': 't18Red'})

findAll or find_all 函式

- 使用get(tag)函式可以得到該tag的屬性
  - Ex: for i in soup.findAll(’span’,{‘class’: t18Red’}):

    print(i.get(‘class’)) #會得到 [‘t18Red’]，以list型態回傳
- 也可以使用正規表示式搜索標籤內容
  - Ex: print(soup.findAll(href=re.compile("http://.*")))
- 使用tag的多個屬性值進行搜尋：
  - Ex: soup.findAll(tag,attrs={‘名稱1’:'值1',‘名稱2’:'值2'})
- 對搜索的結果的個數進行限制： limit=n
  - Ex: print(soup.findAll(href=re.compile("http://.*"), limit = 2))

findAll or find_all 函式

- 同時找尋多個標籤的內容可以傳入list型態
  - Ex: print(soup.findAll([‘a’,‘span’]))　#同時取a與span的內容
- 除了讀取屬性之外亦可修改屬性內的值
  - Ex: tag = soup.findAll('a')[0]

    tag["href"] = "/index.html”

    print(tag)
- 另外還有 find() 函式可以使用，與 findAll 用法相同，不同之處在於find函式得到的結果為findAll回傳的第一個值的內容

recursive 參數

- 呼叫tag的 find_all()方法時，會檢查當前所有tag的子孫節點，如果只想搜尋tag的直接子節點，可以使用參數 recursive = Fasle

```
<html>
 <head>
  <title>
   The Dormouse's story
  </title>
 </head>
...
```

```
soup.html.find_all("title")
# [<title>The Dormouse's story</title>]

soup.html.find_all("title", recursive=False)
# []
```

像調用 find_all()一樣調用tag

- find_all可以用下面的簡寫方式撰寫，兩個例子的程式碼倆倆等價。

```
soup.find_all("a")
soup("a")
```

```
soup.title.find_all(text=True)
soup.title(text=True)
```

下面範例操作的html原始碼

```
html_doc = """
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

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

CSS選擇器

- BeautifulSoup 支持大部分的CSS選擇器，在 Tag 或 BeautifulSoup 對象的 .select() 方法中傳入字串參數，即可使用CSS選擇器的語法找到tag。
- 對於熟悉CSS選擇器語法的人來說這是個非常方便的方法

```
soup.select("title")
# [<title>The Dormouse's story</title>]
soup.select("p:nth-of-type(3)")
# [<p class="story">...</p>]
```

CSS選擇器

- 透過tag標籤逐層查找：

```
soup.select("body a")
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
# <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

soup.select("html head title")
 # [<title>The Dormouse's story</title>]
```

CSS選擇器

- 找到某個tag標籤下的直接子標籤：

```
soup.select("head > title")
# [<title>The Dormouse's story</title>]

soup.select("p > a")
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
# <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

soup.select("p > a:nth-of-type(2)")
# [<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>]

soup.select("p > #link1")
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>]

soup.select("body > a")
# []
```

CSS選擇器

- 找到兄弟節點標籤：

```
soup.select("#link1 ~ .sister")
# [<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

soup.select("#link1 + .sister")
# [<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>]
```

CSS選擇器

- 通過CSS的類別名來查找：

```
soup.select(".sister")
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
# <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

選擇 class包含 sister的所有元素
soup.select("[class~=sister]")
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
# <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

CSS選擇器

- 通過tag的id查找：

```
soup.select("#link1")
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>]

soup.select("a#link2")
# [<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>]
```

CSS選擇器

- 通過是否存在某個屬性來查找：

```
soup.select('a[href]')
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
# <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

CSS選擇器

- 通過屬性的值來查找：

```
soup.select('a[href="http://example.com/elsie"]')
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>]

soup.select('a[href^="http://example.com/"]')
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
# <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

soup.select('a[href$="tillie"]')
# [<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

soup.select('a[href*=".com/el"]')
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>]
```

抓取統一發票範例

- 問題1, (下面這種、分隔的字串要怎麼分割出來)
  - '82267055、72762106、06820335'
- 問題2, 擷取到的內容似乎包含兩期對獎的號碼，要怎麼限制抓取的是某一組 (ex. 最新一期的)

中文URL的編碼/解碼

- import urllib.parse
- urllib.parse.quote(str)
  - 此方法可將str中的字串轉為url編碼
- urllib.parse.unquote(str)
  - 將url碼解碼

Homework 7

- 抓取yahoo!電影的某部電影, 例如:
  - https://tw.movies.yahoo.com/movieinfo_main.html/id=5644
- 需要抓取的資訊如下:
  - 電影名稱 (中英)
  - 上映日期
  - 類　　型
  - 片　　長
  - 導　　演
  - 演　　員
  - 發行公司
  - 官方網站
  - 劇情介紹
- 將擷取出來的資料存檔，檔名: **編號.txt** ，以這部電影為例存檔為**5644.txt**

Selenium

- http://selenium-python.readthedocs.io/index.html

Drivers

Chrome:	https://sites.google.com/a/chromium.org/chromedriver/downloads
Edge:	https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/
Firefox:	https://github.com/mozilla/geckodriver/releases
Safari:	https://webkit.org/blog/6900/webdriver-support-in-safari-10/