Introduction to Python Web crawler

Python 網路爬蟲簡介

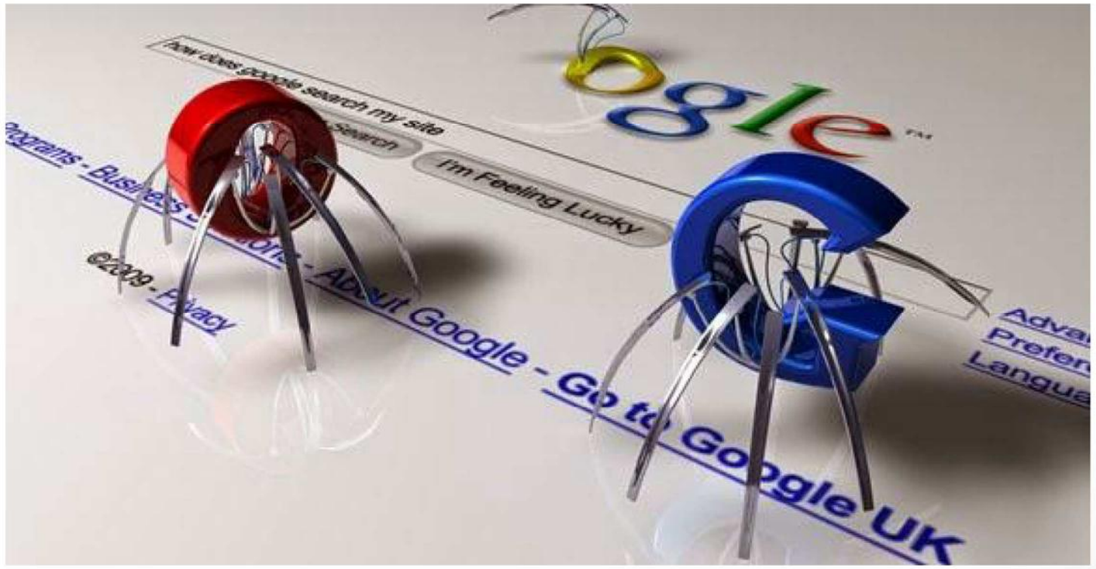
Python Implementation

網路爬蟲簡介

- 網路爬蟲，英文稱做Web Crawler 或 Web Scraping，是一個可以從網路獲取資料的技術，也可以說是一種按照一定的規則，自動的抓取網路資訊的程式或者腳本。
- 它可從全球資訊網上下載網頁，是搜索引擎的資料來源。
- 基本的網路爬蟲從一個初始頁面的網址開始，在抓取網頁的過程中，不斷從當前頁面上抽取新的網址放入佇列，直到滿足系統的停止條件才結束。

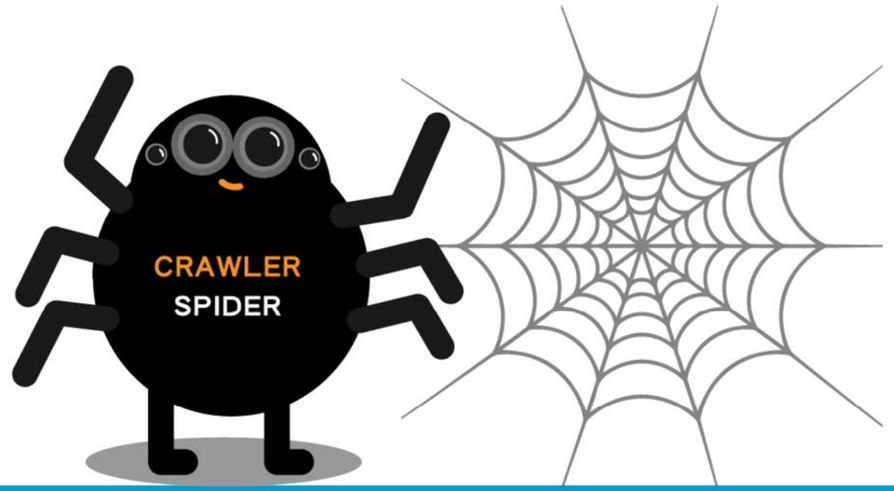
網路爬蟲簡介

- 進行資料探勘時，往往最耗時的步驟並不是對資料進行演算，而是取得資料和處理資料。
- 若沒有合適的資料做為 input，最後只會落得 garbage in ，garbage out 的下場。
- 近年來，**開放資料的風氣盛行，政府和民間都陸續將所蒐集到的資料釋出，成為取得資料相當方便的來源。**
- 但有些資料並不在這類型的網站上等著人下載，因此若能學會基本的網路爬蟲方法，則能將網路上別人的資料變成自己的資料，加以運用。



網路爬蟲簡介

- 理想狀態下， 爬蟲並不是必須品， 每個網站都應該提供API以(半)結構化的格式共用資料。(CSV格式、JSON格式)
- 然而，並不是所有的資料都能這麼方便地以CSV或者JSON載入工作環境，有時候資料散落在網路不同的角落裡。
- 此外，並不是每一個網站都會建置 API（Application Programming Interface）讓你很省力地把資料帶回家。
- 現實情況中，只有部份網站提供了API，而且它們通常會限制可以抓取的資料，以及訪問這些資料的頻率。
- 此外，對於網站開發者而言，維護前端介面比維護後端API介面優先順序更高。
- 因此，我們不能僅僅依賴於API 去訪問我們所需的資料， 而是應該學習一些爬蟲技術的相關知識。



網路爬蟲的用途

網路爬蟲是收集相關網路資料的利器，因此需要大量網路資料的情形就很適合爬蟲，用途主要如下：

- 資料收集、資料分析流程進行
- 做為網頁搜尋引擎 (Google、Yahoo)的網頁收集器
- 即時、非即時的輿情分析
- 社群發文的留言分析
- 科學研究：線上人類行為，線上社群演化，人類動力學研究，計量社會學，複雜網絡，數據挖掘等領域

網路爬蟲是否合法

- 網路爬蟲目前處於早期發展階段，「允許哪些行為」尚處於摸索之中。以目前的情況來看，如果抓取資料的行為僅限於個人使用， 則應不存在問題；而如果資料用於轉載，那麼抓取的資料類型就非常關鍵了。
- 世界各地法院的一些案件可以協助我們了解哪些網路爬蟲行為是允許的。
  - 在Feist Publications, Inc. 起訴Rural Tel寫phone Service Co. 的案件中， 美國聯邦最高法院裁定抓取並轉載真實資料（比如電話清單）是允許的。
  - 在澳大利亞， Telstra Corporation Limited 起訴Phone Directories Company Pty Ltd 這一類似案件中， 則裁定只有擁有明確作者的資料，才可以獲得版權。
  - 在歐盟的ofir.dk起訴home.dk一案中， 最終裁定定期抓取和深度連結是允許的。
  - 這些案件告訴我們， 當抓取的資料是現實生活中的真實資料（比如營業位址、 電話清單） 時，是允許轉載的。
  - 但是如果是原創資料（比如，意見和評論），通常就會受到版權限制，而不能轉載。
- 當要抓取某個網站的資料時，請記住自己是該網站的訪客，理應約束自己的抓取行為， 否則他們可能會封鎖IP，甚至採取更進一步的法律行動。

用什麼語言寫爬蟲？

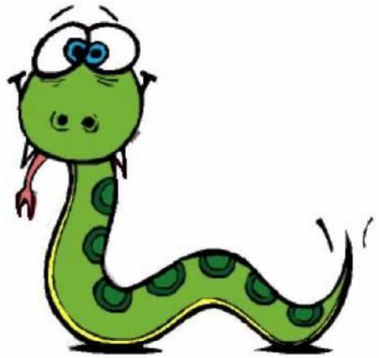
Coding. Crawler

Java

```
public class CrawlerExample {

    public static void main(String[] args) throws IOException {
        PrintWriter textFile = null;
        try {
            textFile = new PrintWriter("result.txt");
            System.out.println("Enter the URL you wish to crawl..");
            System.out.print("@> ");
            String myUrl = new Scanner(System.in).nextLine();

            String response = getContentByUrl(myUrl);

            Matcher matcher = Pattern
              .compile("href=[\"'](.[^\"']+)[\"']").matcher(response);
            while (matcher.find()) {
                String url = matcher.group(1);
                System.out.println(url);
                textFile.println(url);
            }
        } finally {
            if(textFile != null) {
                textFile.close();
            }
        }
    }

    private static String getContentByUrl(String myUrl)
            throws IOException {
        URL url = new URL(myUrl);
        URLConnection urlConnection = url.openConnection();
        BufferedReader in = null;
        StringBuilder response = new StringBuilder();
        try {
            in = new BufferedReader(new InputStreamReader
              (urlConnection.getInputStream()));
            String inputLine;
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
        } finally {
            if(in != null) {
                in.close();
            }
        }
        return response.toString();
    }
}
```

Python

```
if __name__ == '__main__':
    with open("result.txt","wt") as textFile:
        print("Enter the URL you wish to crawl..")
        myUrl = input("@> ")
        for i in re.findall("href=[\"'](.[^\"']+)[\"']",
            urllib.request.urlopen(myUrl).read().decode(), re.I):
            print(i)
            textFile.write(i+'\n')
```



Python為網路爬蟲主流之因 – 相關支援套件眾多

- 對Linux和Windows跨平台都有不錯的支援
- 可使用Numpy，Pandas等進行科學計算與數值分析整合
- 可使用Matplotlib與Mayavi2進行2d與3d的視覺化處理
- 可使用Networkx進行複雜網路的處理
- 可使用Rpy與R語言介面進行統計相關的處理
- 可使用Django進行網站的快速開發

資料來源

- 跨資料分析能夠結合不同的知識，進而找出資料間無法明顯發覺的關連性。跨資料分析牽涉到不同領域的背景，也需要處理、串聯不同來源的資料集，是一個具有挑戰的工作。尤其當資料來自許多不同的單位，用不同的格式發布。這造成資料搜集更加費時費工，有效地善用工具將可以節省人工成本的損耗。

- 資訊就像是經過主廚精心烹調的料理，而資料就像是原料一樣。有好的資料價值，一定要有是適合的資料。「找資料」->「整理資料」->「用資料」，是在分析前的標準程序。仰賴於網路科技的普及，搜尋引擎已經覆蓋了大量的網路資源。有效地使用搜尋引擎是找到的資料的基本功。除此之外，隨著開放資料的議題興起，現在也有越來越多的官方或非官方組織將其資料公開讓大家使用。

資料來源

- **幾個常見的公開資料單位：**
  - 政府資料開放平台
  - 行政院主計總處
  - 中選會選舉資料庫
  - Google Public Data
  - World Bank 世界銀行

資料取得

- 知道資料在哪裡之後，接下來就是如何取得資料。一般來說，資料常見的發佈方式有幾下三種：
  - 檔案
  - API
  - 靜態/動態網頁

資料取得

◆ 檔案：

- 資料會包成檔案提供下載，格式主要為「CSV」、「JSON」、「XML」等等。如果是已經有提供制式的格式的話，相對容易處理，一般的程式語言或是商業軟體都具備讀取的功能。

◆ API：

- API（Application Program Interface，應用程序接口），提供程式化的連接的接口，讓工程師/分析師可以選擇資料中要讀取的特定部分，而不需要把整批資料事先完整下載回來。
- API 一般都是直接連接到一個資料庫，而資料庫內儲存的都是即時更新最新版本的資料。
- API 主要分為兩個動作：1. 使用者呼叫查詢； 2. 伺服器根據需求回傳。
- 呼叫的方式有 POST 或是 GET；回傳一般也會使用像是 JSON 的格式。

◆ 靜態/動態網頁：

- 最後一種也是很常出現資料的地方，就是網頁上。我們常常會發現我們的資料並不是一個特定的檔案，也沒有 API 可以使用。他就是穩穩地出現在網頁上。這樣的話，就只能自己寫爬蟲，把自己想用的資料從網頁上爬下來。

網路爬蟲 Web Crawler

- 網路爬蟲是用在沒有以檔案或是 **API** 釋出資料集的情況下。這個時候就只能捲起袖子，自己想要的資料自己爬！

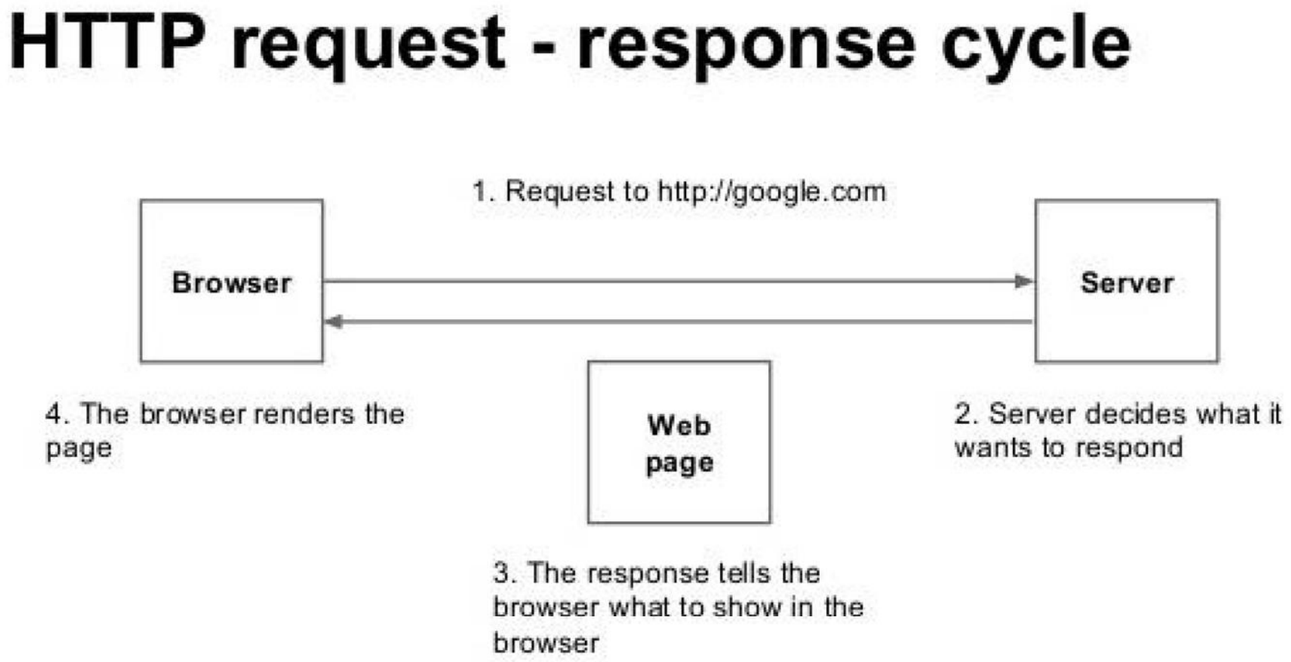
- 網路爬蟲，簡單來說，就是模擬使用者的行為，把資料做一個攔截的動作。基本上可以簡化為：

[模擬 Request] -> [攔截 Response] -> [從 Response 整理資料] -> [done!]

網路爬蟲 Web Crawler - 靜態網頁

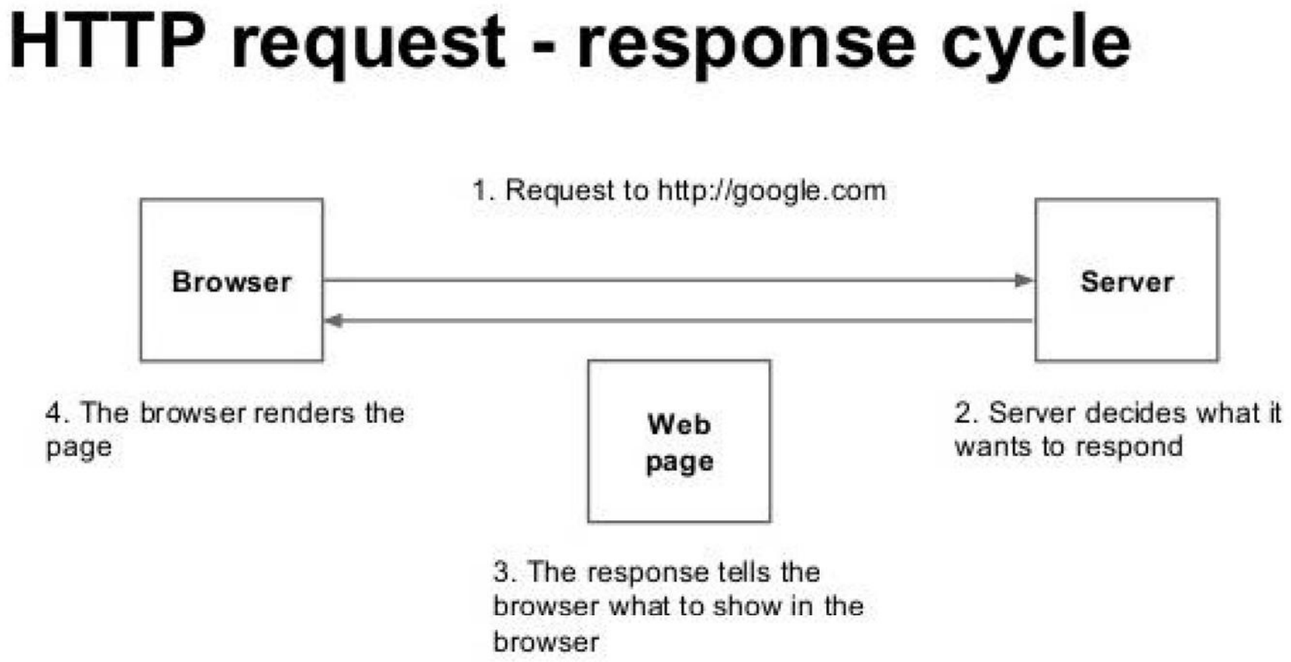
- 所謂的靜態網頁，表示網頁是在 **Server-side** 就已經產生回來的，所以你看的網頁上的資料是固定的（除非重新要求 **Server-side**）。

- 這樣時候，我們可以來解析一下網頁、瀏覽器與伺服器的流程如下圖



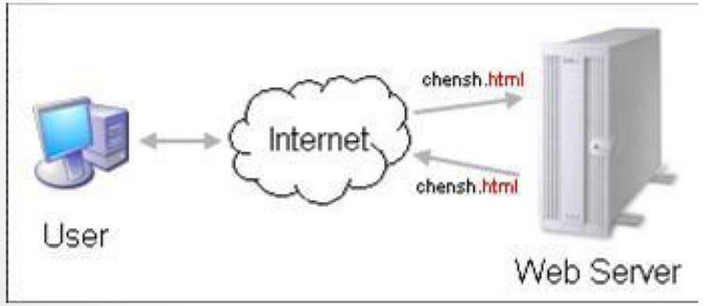
網路爬蟲 Web Crawler - 靜態網頁形成及溝通過程

1. 使用者（**Client-side**）發出請求，稱為是 **Request**。
2. 伺服器（**Server-side**）收到請求，根據請求處理後回應，稱為是 **Response**。
3. 產生的回應如果是純資料的話，屬於 **API** 的一種；如果是網頁的話，就會回傳一個包含 **HTML** 標籤的網頁格式。
4. 瀏覽器接收包含 **HTML** 標籤的網頁格式，呈現網頁給使用者。

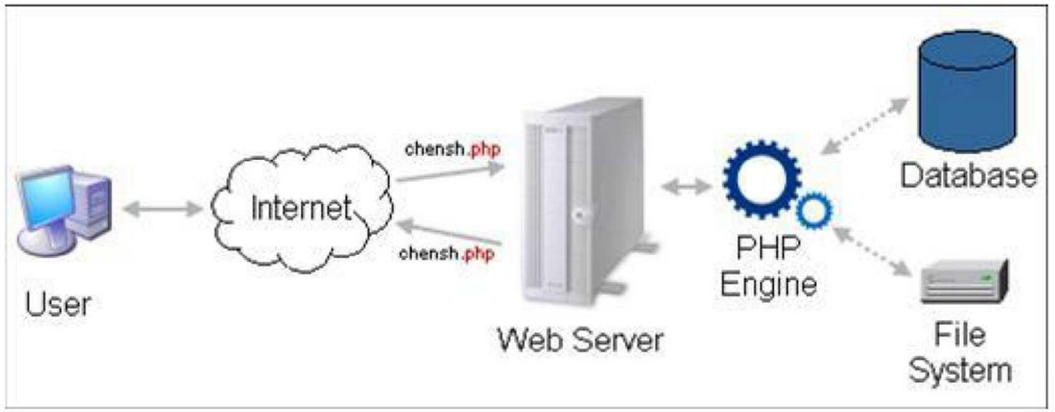


網路爬蟲 Web Crawler – 動態網頁

- 動態網頁有別於靜態網頁產生資料的方式。靜態網頁是透過每一次使用者請求，後端會產生一次網頁回傳，所以請求與回傳是一對一的，有些人把他們稱為同步。
- 在動態網頁的話，是透過 Ajax 的技術，來完成非同步的資料傳輸。換句話說，就是在網頁上，任何時間點都可以發送請求給後端，後端只回傳資料，而不是回傳整個網頁。
- 這樣一來，就不是一對一的關係，在處理資料上就會比較麻煩。所以我們換個角度，原本是模擬瀏覽器的動作，現在我們直接模擬人的操作。



靜態網頁



動態網頁

網頁資料擷取

從網站擷取資料的主要方式有以下幾種：

- 經由網站，直接下載以某種格式儲存的原始資料檔案，主要以CSV 或JSON格式為主
- 經由網站提供的專屬API來抓取資料，主要以CSV 或JSON格式為主
- 經由HTTP爬取網頁資料，並且在本地端進行解析，抽取出想要的部份 (網路爬蟲)

Q & A