D7：解析靜態網頁與實作靜態資料爬蟲

課程閱讀　本日作業　問題討論　學習心得(完成)

解析靜態網頁與實作靜態資料爬蟲

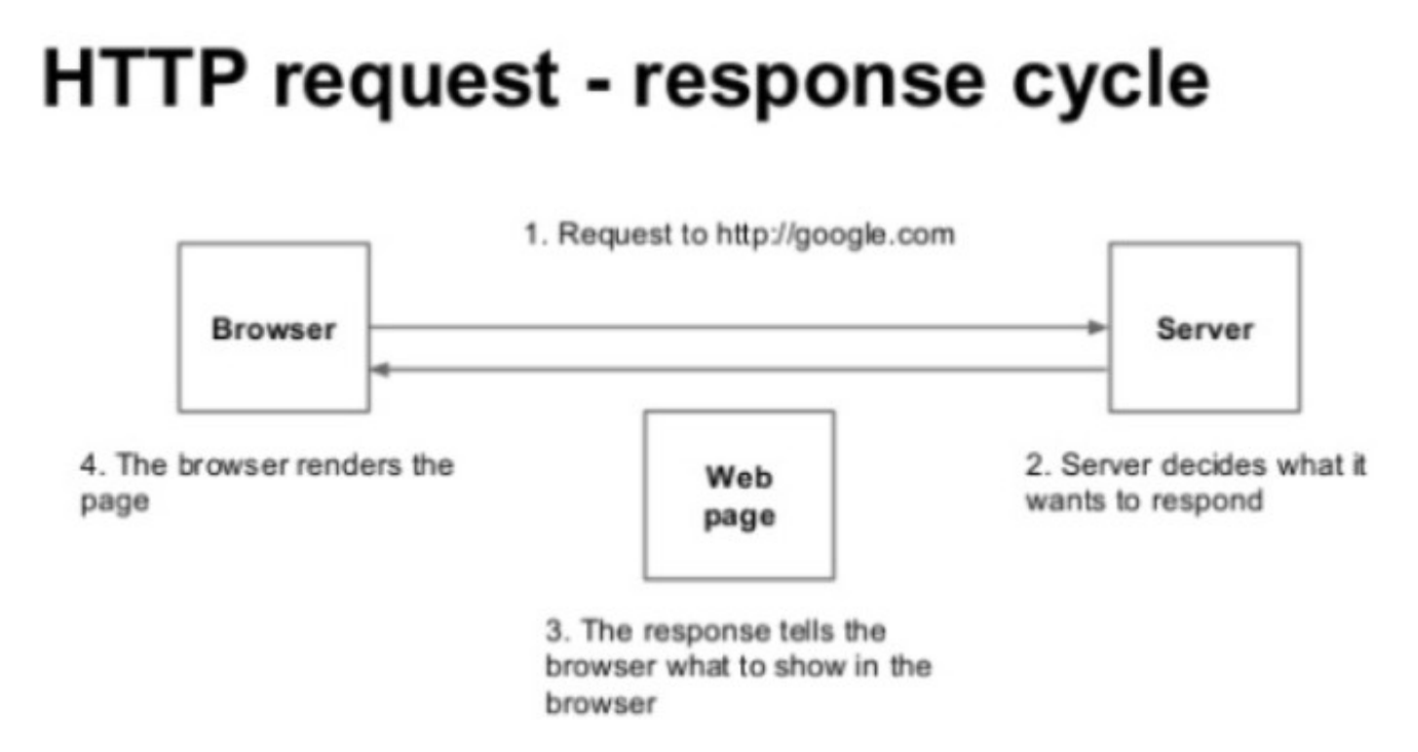
本日知識點目標

- 了解靜態網頁的資料爬蟲策略
- 認識適用於靜態網頁爬蟲的相關套件工具：Request
- 認識適用於靜態網頁爬蟲的相關套件工具：BeatifulSoup

靜態網頁的資料爬蟲策略

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：

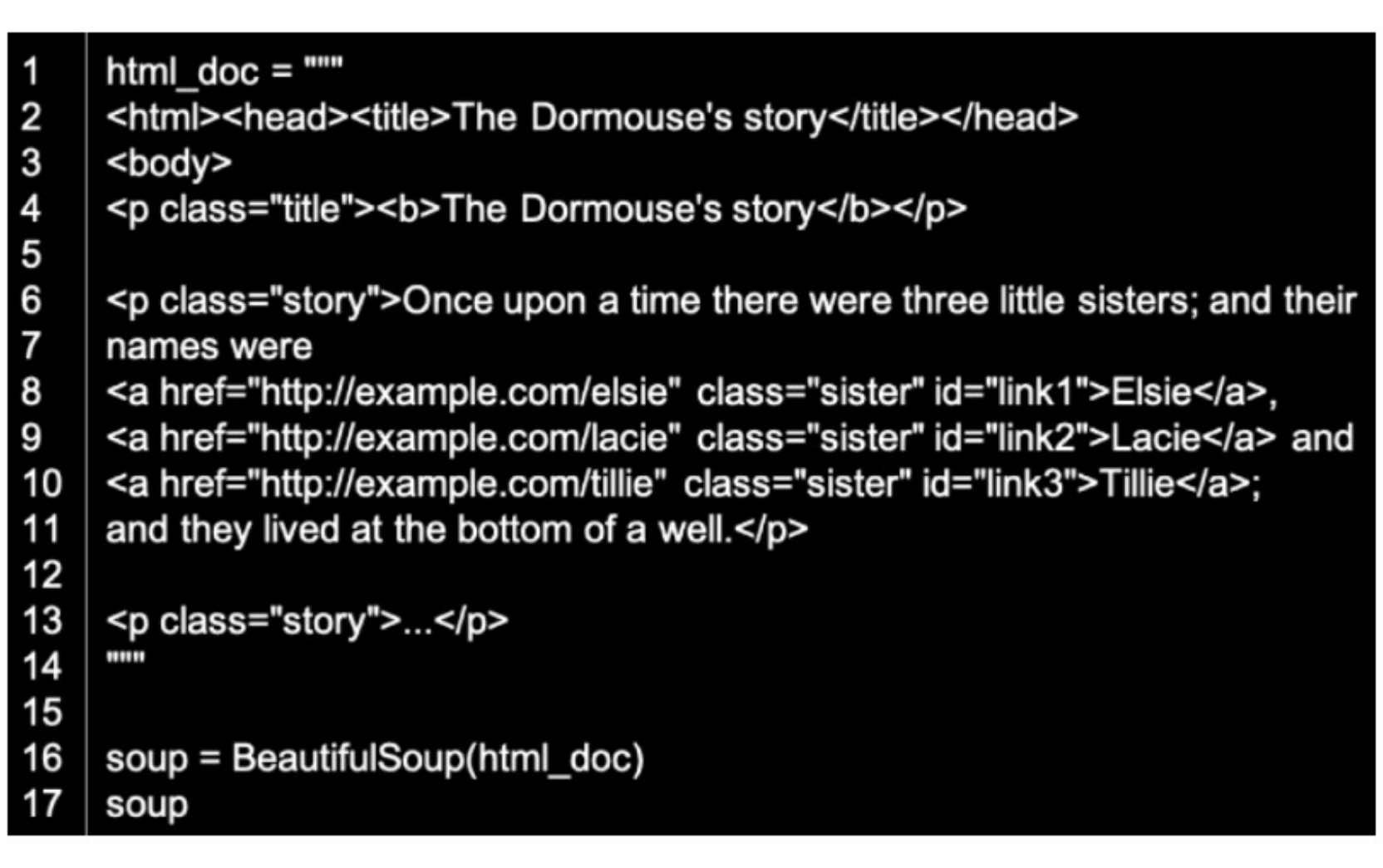
- 模擬 Request
- 攔截 Response
- 從 Response 整理資料



先看範例：模擬 Request & 攔截 Response

```
import requests
# 引入函式庫
r = requests.get('https://github.com/timeline.json')
# 想要爬資料的目標網址
response = r.text
# 模擬發送請求的動作
```

先看範例：從 Response 整理資料



```
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>
"""

soup = BeautifulSoup(html_doc)
soup
```

靜態爬蟲的處理

以上的範例我們將程式分成兩個階段：

1. 模擬 Request & 攔截 Response
2. 從 Response 整理資料

這邊分別利用了「request」和「BeautifulSoup」這兩個函式庫進行

Requests Library

Requests 是一個 Python HTTP 庫，該項目的目標是使 HTTP 請求更簡單，更人性化。

其主要工作作為負責網頁爬蟲中的 HTTP Request & Respone 的部分。

BeautifulSoup Library

Beautiful Soup 是一個 Python 包，功能包括解析HTML、XML文件、修復含有未閉合標籤等錯誤的文件。這個擴充包為待解析的頁面建立一棵樹，以便提取其中的資料。

其主要工作作為負責網頁爬蟲中的 解析資料 的部分。

BeautifulSoup

在這樣的網頁語法中，我們可以利用 BeautifulSoup 挑出我們想要的部分

soup = BeautifulSoup(html_doc)

```
<html>
      <head>
           <title>The story</title>
      </head>
<body>
      <p class="title-class" id="title-id"><b>The Dormouse's story</b></p>
      <p class="content">
            <a href="http://example.com/link1" class="link" id="link1">A</a>
            <a href="http://example.com/link2" class="link" id="link2">B</a>
            <a href="http://example.com/link3" class="link" id="link3">C</a>
      </p>
</body>
```

\# 利用標籤
soup.title #
soup.title.name # title
soup.title.text # The story

```
<html>
     <head>
           <title>The story</title>
     </head>
<body>
     <p class="title-class" id="title-id"><b>The Dormouse's story</b></p>
     <p class="content">
          <a href="http://example.com/link1" class="link" id="link1">A</a>
          <a href="http://example.com/link2" class="link" id="link2">B</a>
          <a href="http://example.com/link3" class="link" id="link3">C</a>
     </p>
</body>
```

\# 取出屬性
soup.p['class'] # [title-class]
soup.a['id'] # title-id

```
<html>
      <head>
            <title>The story</title>
      </head>
<body>
      <p class="title-class" id="title-id"><b>The Dormouse's story</b></p>
      <p class="content">
            <a href="http://example.com/link1" class="link" id="link1">A</a>
            <a href="http://example.com/link2" class="link" id="link2">B</a>
            <a href="http://example.com/link3" class="link" id="link3">C</a>
      </p>
</body>
```

\# 利用 find 方法
soup.find(id='titile-id')
soup.find('p', class_="content")
soup.find_all('a', class_="link")

```
<html>
      <head>
            <title>The story</title>
      </head>
<body>
      <p class="title-class" id="title-id"><b>The Dormouse's story</b></p>
      <p class="content">
            <a href="http://example.com/link1" class="link" id="link1">A</a>
            <a href="http://example.com/link2" class="link" id="link2">B</a>
            <a href="http://example.com/link3" class="link" id="link3">C</a>
      </p>
</body>
```

BeautifulSoup 語法範例

soup = BeautifulSoup(html_doc) => 將原始的 HTML 字串轉成物件

\# 利用標籤
soup.title # <dom> => 取出第一個 title 標籤的物件
soup.title.name # title => 取出第一個 title 標籤的物件的標籤名稱
soup.title.text # The story => 取出第一個 title 標籤的物件的文字

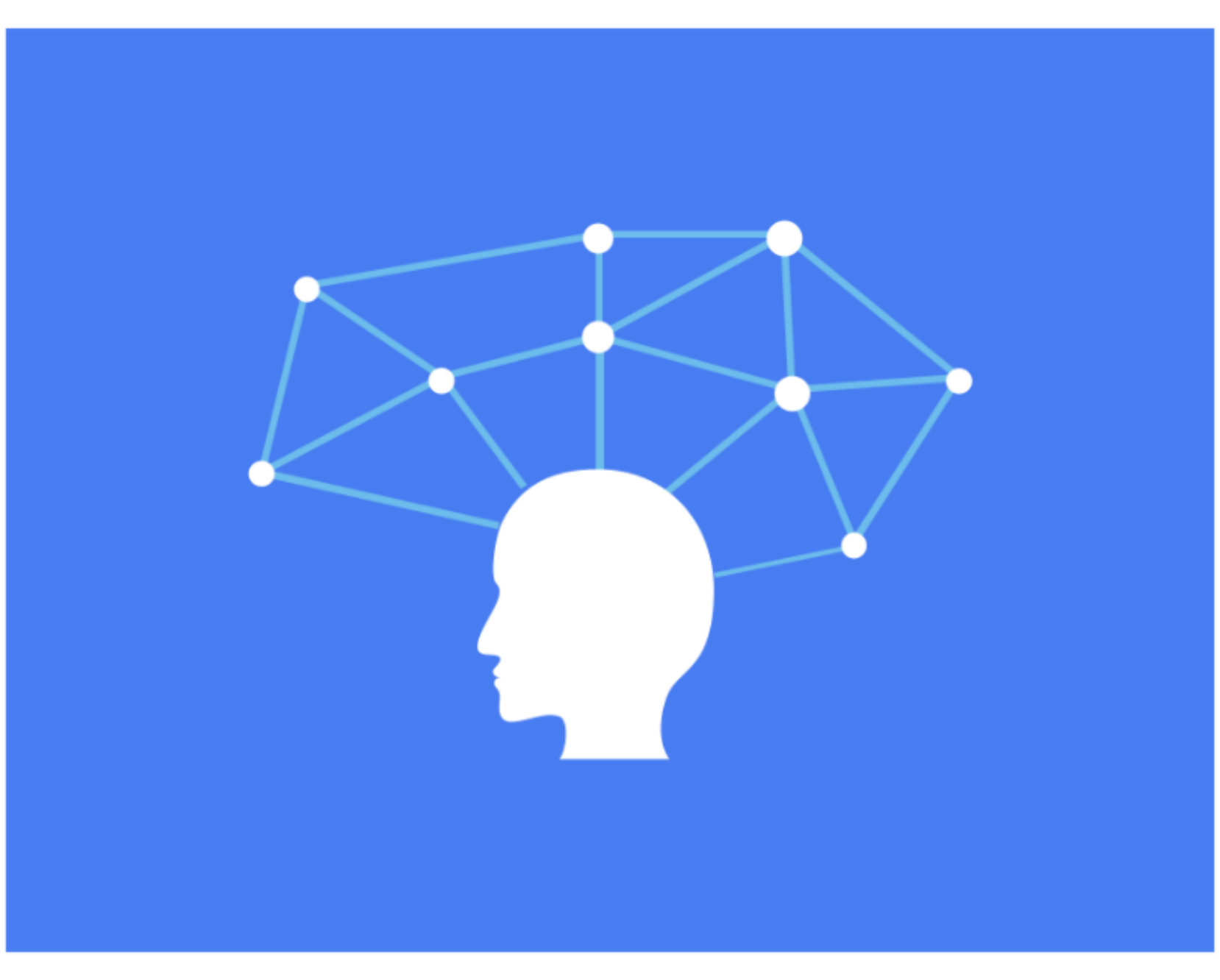
\# 取出屬性
soup.p['class'] # [title-class] => 取出第一個 p 標籤的物件中的 class 屬性
soup.a['id'] # title-id => 取出第一個 p 標籤的物件中的 id 屬性

\# 利用 find 方法
soup.find(id='titile-id') => 取出第一個 id = title-id 的物件
soup.find('p', class_="content") => 取出第一個 class = content 的 p 標籤物件
soup.find_all('a', class_="link") => 取出所有 class = link 的 a 標籤物件

從存取資料到解析內容

以上的範例我們將程式分成兩個階段：

重要知識點複習



- 了解靜態網頁的資料爬蟲策略
- 認識適用於靜態網頁爬蟲的相關套件工具：Request & BeatifulSoup

解題時間

下一步：完成作業