Ask Company

크롤링 차근차근 시작하기

BeautifulSoup4 라이브러리

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

HTTP 응답

웹서버는 대개 HTML, CSS, JavaScript, Image 형식의 응답

HTML 문서는 중첩된 태그로 구성된 계층적인 구조

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Ask Company</title>
    </head>
    <body>
        <h1>Ask Company VOD</h1>
        <ul id="vod_list">
            <li class="vod">파이썬 차근차근 시작하기</li>
            <li class="vod">장고 기본편</li>
        </ul>
        <hr/>
        &copy; 2018 Ask Company
    </body>
</html>
```

DOM 문서

브라우저는 HTML문자열을 DOM Tree로 변환하여, 문서를 표현

```
<!-- 서버로부터 아래 응답을 받는다면
(우리는 이 부분에 집중!!!)
브라우저의 "페이지 소스보기"를 통해 확인
-->

<table>
    <tr>
        <td>테이블 컬럼</td>
    </tr>
</table>
```

```
<!-- 브라우저를 이를 DOM Tree로 다음과 같이 변환
(이때 브라우저 나름의 해석이 들어갑니다.)
브라우저의 "개발자도구"를 통해 확인 -->

<table>
    <tbody>
        <tr>
            <td>테이블 컬럼</td>
        </tr>
    </tbody>
</table>
```

https://developer.mozilla.org/ko/docs/Gecko_DOM_Reference/%EC%86%8C%EA%B0%9C

복잡한 문자열에서 특정 문자열을 가져올려면?

방법1: 정규 표현식을 활용

가장 빠른 처리가 가능하나, 정규표현식 Rule을 만드는 것이 많이 번거롭고 복잡합니다.

때에 따라 필요할 수도 있습니다.

방법2: HTML Parser 라이브러리를 활용

DOM Tree을 탐색하는 방식으로 적용이 쉽습니다.

ex) BeautifulSoup4, lxml, pyquery 등

BeautifulSoup4

HTML/XML Parser

HTML/XML문자열에서 원하는 태그정보를 뽑아냅니다.

설치 : pip install beautifulsoup4

```
from bs4 import BeautifulSoup

html = '''
    <ol>
        <li>NEVER - 국민의 아들</li>
        <li>SIGNAL - TWICE</li>
        <li>LONELY - 씨스타</li>
        <li>I LUV IT - PSY</li>
        <li>New Face - PSY</li>
    </ol>
'''

soup = BeautifulSoup(html, 'html.parser')
for tag in soup.select('li'):
    print(tag.text)
```

다양한 BeautifulSoup4의 Parser

html.parser

BeautifulSoup4 내장 파서

lxml

lxml HTML 파서 사용 (외부 C 라이브러리)

html.parser보다 유연하고 빠른 처리

설치: pip install lxml

```
soup = BeautifulSoup(파싱할문자열, 'html.parser')
```

HTML내에서 태그 찾기

방법1) find를 통해 하나씩 차근차근 찾아들어가기

방법2) 태그 간의 관계를 지정하여 찾기 with CSS Selector

일단 melon 차트 HTML 획득하기

```
>>> import requests
>>> from bs4 import BeautifulSoup

>>> headers = {
        'User-Agent': ('Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) '
                       'AppleWebKit/537.36 (KHTML, like Gecko) '
                       'Chrome/58.0.3029.110 Safari/537.36'),
        'Referer': 'http://www.melon.com',
    }
>>> res = requests.get('http://www.melon.com/chart/index.htm', headers=headers)
>>> html = res.text

>>> res.status_code
200

>>> soup = BeautifulSoup(html, 'html.parser')
```

방법1)

find를 통해 하나씩 차근차근 찾아 들어가기

```
tag_list = []
for tr_tag in soup.find(id='tb_list').find_all('tr'):
    tag = tr_tag.find(class_='wrap_song_info')
    if tag:
        tag_sub_list = tag.find_all(href=lambda value: (value and 'playSong' in
value))
        tag_list.extend(tag_sub_list)

for idx, tag in enumerate(tag_list, 1):
    print(idx, tag.text)
```

CSS Selector 문법의 대표적인 예 (1)

css selector	설명
*	모든 태그
"tag_name"	지정 이름의 모든 태그
"#tag_id"	아이디가 tag_id인 모든 태그
".tag_class"	클래스명 중에 tag_class가 포함된 모든 태그
"tag_name#tag_id"	tag_name 태그 중에 아이디가 tag_id인 모든 태그 (실제 1개)
"tag_name.tag_class"	tag_name 태그 중에 클래스명에 tag_class가 포함하는 모든 태그
"tag_name#tag_id.tag_cls1.tag_cls2"	
"tag_name.tag_cls1.tag_cls2"	
"tag_name1 tag_name2"	tag_name1 태그의 자손 중에서 tag_name2 이름의 모든 태그
"tag_name1 > tag_name2"	tag_name1 태그의 **직계** 자손 중에서 tag_name2 이름의 모든 태그
"tag.tag_cls1 .tag_cls2"	

CSS Selector 문법의 대표적인 예 (2)

css selector	설명
"tag_name[attr]"	tag_name 태그 중에 attr 속성이 정의된 모든 태그
"tag_name[attr=bar]"	tag_name 태그 중에 attr 속성값이 bar인 모든 태그
"tag_name[attr*=bar]"	tag_name 태그 중에 attr 속성값에 bar 문자열이 포함된 모든 태그
"tag_name[attr^=bar]"	tag_name 태그 중에 attr 속성값이 bar 문자열로 시작하는 모든 태그
"tag_name[attr$=bar]"	tag_name 태그 중에 attr 속성값에 bar 문자열이 끝나는 모든 태그

방법2)

CSS Selector 활용

```
tag_list = soup.select('#tb_list tr .wrap_song_info a[href*=playSong]')
for idx, tag in enumerate(tag_list, 1):
    print(idx, tag.text)
```

Tip. CSS Selector 지정할 때

패턴을 너무 타이트하게 지정하시면, HTML 마크업이 조금만 변경되어도 태그를 찾을 수 없게 됩니다.

적절히 최소한의 패턴을 타이트하게 지정해주세요.

이는 다양한 연습을 통해,
감을 익힐 수 밖에 없습니다.

미션

멜론 TOP100 사이트에서 100곡에 대해 아래 정보를 모두 수집해보세요.

- 곡명
- 앨범명
- 가수명
- 랭킹

풀이는 다음 에피소드에서 ☺

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

- Ask Company