5. 웹 스크래핑

5.1. HTTP

우리가 흔히 웹(Web)이라고 부르는 기술은 HTTP라는 방식으로 통신을 한다. HTTP는 하이퍼 텍스트 전송 프로토콜(HyperText Transfer Protocol)의 약자로, 여기서 프로토콜은 '통신 방식'을 의미한다. 인터넷 주소창에 주소가 `http://` 또는 `https://`로 시작하는 것을 보았을 것이다.

HTTP에서는 2대의 컴퓨터가 통신을 한다. 이때 한 쪽은 클라이언트라 하고, 다른 한쪽은 서버라고 한다. 구글, 페이스북, 네이버 등이 서버의 역할을 하고, 사용자는 클라이언트 역할을 한다.

S
(요청)

S
(응답)

HTTP는 항상 클라이언트가 '요청'을 하면 서버가 '응답'을 하는 방식으로 통신이 이뤄진다. 이때 클라이언트는 여러 가지 종류의 요청을 할 수 있다. 대표적인 것이 `get`과 `post`다.

`get` 요청은 서버로부터 글이든 사진이든 무엇을 가져올 때 사용하는 요청이다. 글을 읽거나 파일을 다운로드 받거나 게시판 목록을 볼 때 모두 `get`을 사용한다.

`post`나 서버에 클라이언트가 무엇인가를 보낼 때 사용한다. 로그인 시에 아이디와 비밀번호를 보내거나 게시판에 글을 쓰거나 파일을 업로드 할 때 모두 `post`를 사용한다.

가끔은 공공기관 웹사이트의 경우 과거에 `post` 방식이 더 보안이 좋다는 식으로 잘못 오해를 해서 단순히 게시판 목록 등을 확인할 때도 `post` 방식을 사용하기도 한다.

5.1.1. requests

파이썬에서 HTTP로 통신을 하는 방법에는 여러 가지가 있다. 파이썬에 기본 내장된 라이브러리도 있지만 좀 더 편리한 라이브러리로 `requests`가 있다.

명령창에서 `pip install requests` 로 설치할 수 있다.

주피터 노트북에서 **requests** 를 임포트 한다.

```
import requests
```

5.1.2. get 실습

requests 를 이용해 서버에 요청한다.

`url` 변수에 긁고자 하는 페이지 주소를 저장한다.

```
url = 'https://en.wikipedia.org/wiki/HTML5'
```

`requests.get()` 을 사용해 사이트 정보를 요청해, `res` 에 저장한다.

```
res = requests.get(url)
```

5.1.3. 결과 확인

해당 페이지의 정보가 잘 긁혔는지 HTTP 상태 코드를 확인해보자.

```
res.status_code
```

```
200
```

200이 결과로 나온다면, 성공적으로 요청이 받아들여졌다는 이야기다.

200대의 숫자 외에, 400대의 숫자가 나온다면 서버에 요청하는 과정에서, 클라이언트 에러를 냈다는 것을 의미한다.

반대로, 500대의 숫자가 나온다면, 서버 측에서 에러가 있었다는 것을 의미한다.

5.1.4. 추가 정보

`.encoding` 으로 페이지의 인코딩을 확인할 수 있다.

```
res.encoding
```

```
'UTF-8'
```

`.text` 로 전체 페이지를 HTML 문자열로 볼 수 있다.

```
res.text
```

```
'<!DOCTYPE html>\n<html class="client-nojs" lang="en" dir="ltr">\n<head>\n<me
ta charset="UTF-8"/>\n<title>HTML5 - Wikipedia</title>\n<script>document.docu
mentElement.className = document.documentElement.className.replace( /(^|\\s)c
lient-nojs(\\s|$)/, "$1client-js$2" );</script>\n<script>(window.RLQ=window.R
LQ||[]).push(function(){mw.config.set({"wgCanonicalNamespace":"","wgCanonical
SpecialPageName":false,"wgNamespac ... '
```

5.2. HTTP 상태 코드

자주 나오는 HTTP 상태코드 몇 가지에 대해 알아보자. 기본적으로 HTTP 상태코드는 200, 300 등과 같이 세 자리의 숫자로 구성되어 있으며, 각 숫자의 앞자리는 대분류를 의미한다.

5.2.1. 2xx: 성공

200번대 코드는 모두 성공을 뜻한다.

가장 많이 보게 될 200 코드는 요청이 성공적으로 받아들여졌다는 것을 의미한다.

201 등의 여러 상태 코드가 있지만, 거의 사용하지 않는다.

5.2.2. 3xx: 다른 곳에 있음

300번대 상태코드들은 자료가 존재하지만 위치가 변경되었다는 것을 의미한다. 대부분 주소가 자동 전환되기 때문에 크롤링 할 때 크게 신경 쓸 필요는 없다. 이중에서 302 코드와 304 코드를 살펴보자.

302

302 코드는, 요청한 웹사이트 주소가 다른 주소로 옮겨갔다는 것을 의미한다.

304

304 코드는 요청한 자료가 변경되지 않았다는 것을 의미한다. 웹사이트에 접속하면 사이트의 자료를 이용자의 컴퓨터에 보낸다. 하지만 용량이 큰 사진파일의 경우, 매번 보내주기에 비효율적이기 때문에, 한번 사용자의 컴퓨터로 보내고는 그 사진을 사용한다. 이러한 이유로 이미지가 많은 웹사이트에 처음 접속하면 사진이 느리게 뜨지만, 두번째로 들어가면 이미지가 빨리 뜬다. 만약 이미 사용자의 컴퓨터에 자료를 요청하면, 서버는 이미 받아간 파일이라는 의미에서 304 코드를 보여준다.

5.2.3. 4xx: client 문제

문제가 생기는 경우는 400번대 코드와 500번대 코드다. 400번대 코드는, 클라이언트 문제로 사용자가 잘못했을때 뜨는 메시지다. 400번대에서 가장 자주 보게 될 코드는 400, 404, 405 이다.

400

400 코드는 서버에 요청을 잘못했다는 것을 의미한다. 구체적으로 무엇을 잘못했는지는 알 수 없기 때문에, 대체적으로 해결하기 조금 어렵다.

404

404 코드는 요청한 주소가 잘못되었다는 것을 의미한다. 큰 사이트들은 예쁜 화면으로 주소를 찾을 수 없다고 띄우지만, 작은 사이트들은 그저 흰 바탕에 검은 글씨로 *404 not found* 라는 기본 메세지를 띄운다. 404 코드는 주소를 다시 확인해보면 쉽게 해결되는 문제다.

405

405 코드는 **get**, **post** 와 같은 통신 방식이 잘못되었다는 것을 의미한다. 예를 들어**get** 으로 통신해야 하는 주소에 **post** 로 통신하려 했다면, 405 에러를 보게 된다. 반대로,**post**로 통신해야 하는 주소에 **get** 으로 통신한다면, 마찬가지로 405 에러를 보게 된다.

5.2.4. 5xx: server 문제

500번대 코드는 서버 쪽에 문제가 있다는 뜻으로, 웹사이트를 운영하는 쪽에서 문제가 생겼다는 메시지다.

500

500은 internal 서버 에러로, 서버 내부에 에러가 발생했다는 것을 의미한다. 사용자의 잘못으로 발생한 에러가 아닌, 서버 내부의 버그 등으로 생긴 경우에는 사용자 측에서 해결하기 어렵다.

503

503 코드도 가끔 보게 되는데, 주로 서버가 폭주하거나 인터넷 보안 설정 때문에 발생하는 코드다. 500번대 코드는 다양한 상황에서 문제가 발생해 해결하기 어렵지만, 서버에 접속자가 너무 많은 이유라면, 조금 기다렸다가 다시 접속해 해결할 수 있다.

5.3. Encoding

컴퓨터는 문자를 숫자로 이해하는데, 이를 위해 문자를 숫자로 바꾸는 방식을 인코딩이라고 부른다. 예를 들어 'A' 를 숫자 65 로 표현할 수 있다.

5.3.1. ASCII와 Latin-1

현재 쓰이는 대부분의 인코딩은 1967년 미국에서 만든 ASCII에 바탕을 두고 있다. ASCII는 7비트 인코딩으로 모두 128개의 글자를 표현할 수 있다. ASCII만으로는 알파벳과 숫자, 몇 가지 문장부호만을 표현할 수 있었기에 이를 8비트로 확장한 ISO/IEC 8859 인코딩이 나타

난다. ISO 8859는 8비트로 ASCII보다 128개의 문자를 더 나타낼 수 있다. ISO 8859-1은 이 128개에 서유럽에서 사용되는 문자들을 추가한 것으로 Latin-1이라고도 하며 현재도 널리 쓰이고 있다.

5.3.2. EUC-KR과 CP949

한글의 경우에는 8비트로도 부족하기 때문에 16비트 인코딩을 사용했다. 가장 오래된 표준인 KS X 1001은 한글을 낱자로 조합하는 조합형 방식 대신 "가", "닭"과 같이 완성된 글자들 각각에 숫자를 붙이는 완성형 방식을 사용했다. 한글 완성형은 현대 한국어에서 자주 사용하는 2,350자만을 포함시켰다. KS X 1001과 ASCII를 포함하는 인코딩을 **EUC-KR**이라고 한다.

EUC-KR은 한글을 완전히 쓸 수 없었기 때문에 여러 문제가 있었다. 그래서 마이크로소프트는 윈도95부터 EUC-KR을 확장하여 현대 한글 11,172글자를 모두 포함한 **CP949**를 사용하였다. 현재는 EUC-KR이라고 해도 일반적으로 CP949를 가리킨다. 일부에서는 MS949라는 이름으로 부르기도 한다.

5.3.3. 유니코드와 UTF-8

한국에는 EUC-KR이 있듯이 일본에도 EUC-JP, SHIFTJS 등의 인코딩이 있다. 국가마다 서로 다른 인코딩이 존재하기 때문에 동일한 텍스트도 국가에 따라 다른 글자로 보이는 문제가 있다. 이러한 문제를 해결하기 위해 유니코드(unicode)라는 전세계 공통의 문자집합이 만들어졌다. 유니코드에는 한글, 알파벳, 한자는 물론이고 사라진 고대 문자까지 총 139종 136,755 글자가 포함되어 있다.

유니코드에는 UTF-8, UTF-16 등 몇 가지 인코딩이 존재한다. UTF-8의 경우 알파벳만 사용하면 ASCII와 똑같아진다는 특징이 있어 널리 사용된다. 리눅스와 맥은 UTF-8을 기본 인코딩으로 사용한다. 한 조사에 따르면 전세계 웹사이트의 98%가 UTF-8을 사용하고 있다.

5.3.4. 주의점

윈도에서는 기본 인코딩이 CP949이고 리눅스와 맥에서는 UTF-8이기 때문에 운영체제 간에 파일을 주고 받을 때는 인코딩에 문제가 생길 수 있다.

5.4. 다운로드

이번 강의에서는 인터넷에서 이미지 혹은 파일을 다운 받는 방법을 알아보자.

5.4.1. 이미지 다운로드

이미지를 우클릭 후 "이미지 주소 복사"를 선택한다. 해당 이미지 주소를 `url` 변수에 할당한다.

```
url = 'https://www.python.org/static/img/python-logo.png'
```

url에 접속한다.

```
res = requests.get(url)
```

응답의 내용을 파일에 저장한다.

```
with open('logo.jpg', 'wb') as f:
    f.write(res.content)
```

5.5. HTML

웹 문서는 HTML이라는 문서 형식을 사용한다. HTML은 DOC나 HWP와 비슷한 문서 형식이지만 전용 프로그램으로 열지 않아도 서식 정보를 사람이 확인할 수 있다는 특징이 있다.

HTML에서 서식은 다음과 같이 태그(tag)를 이용해서 표시한다.

```
<u>하이</u>
```

꺽쇠(`<>`)로 감싼 내용을 태그라고 하는데 `u` 태그는 밑줄(underline)을 가리킨다. 그리고 `<u>` 는 시작 태그, `</u>` 는 끝 태그이다. 즉, 위의 표현은 '하이'에 밑줄을 치라는 뜻이 된다.

자주 사용되는 태그로는 다음과 같은 것들이 있다.

- a: 링크
- div: 구역
- span: 문단 내에서 같은 서식이 적용된 영역
- h1, h2, ..: 제목1, 제목2, ...
- strong: 강조
- td: 표의 칸

링크의 경우 보통 아래와 같은 형식으로 사용한다.

```
<a href="http://google.com">구글</a>
```

위의 HTML은 '구글'이라는 글자에 링크를 건다. `href` , `class` 와 같은 것들을 속성

(attribute)이라고 하는데, 말 그대로 태그에 구체적인 속성을 표시한다. `href` 는 링크를 클릭했을 때 이동할 주소를 가리킨다.

웹 스크래핑에 특히 중요한 속성이 `id` 와 `class` 다. 이 둘은 태그의 고유 아이디와, 태그의 서식 유형을 지정할 때 사용한다. 이 속성들을 이용하면 여러 태그 중에 원하는 태그를 지목할 수 있다.

5.6. lxml

lxml 은 HTML을 해석하고 원하는 태그를 쉽게 찾아주는 라이브러리다.

라이브러리가 없다면 `pip install lxml` 로 설치할 수 있다.

```
import lxml.html
```

5.6.1. HTML 분석

HTML을 문자열 형태로 가져와 `lxml.html.fromstring()` 에 넘기면, HTML 문자열을 분석한 요소가 생긴다.

```
url = 'https://en.wikipedia.org/wiki/HTML5'
res = requests.get(url)
element = lxml.html.fromstring(res.text)
```

제목 뽑아보기

웹페이지의 제목을 뽑아보자. 먼저 제목에 해당하는 태그가 어떤 속성을 가지고 있는지 확인할 필요가 있다. 최근의 웹 브라우저들은 개발자 도구를 내장하고 있어, 웹 문서에서 각 요소들을 확인해볼 수 있도록 한다.

크롬을 기준으로 설명하자. 제목을 우클릭 후 **검사**를 클릭하면 제목이 강조되어 표시되면서 다음과 같은 HTML을 확인할 수 있다.

```
<h1 id="firstHeading" class="firstHeading">HTML5</h1`
```

위의 HTML은 제목1(`h1`)으로 `id` 와 `class` 속성이 모두 `firstHeading` 이다. 이런 속성을 이용해 위의 태그를 찾을 수 있다.

5.6.2. cssselect

먼저 `pip install cssselect` 로 `cssselect` 라이브러리를 설치한다. 이 라이브러리는 직

접 사용하지 않으므로 import를 할 필요가 없다.

이제 `class` 가 `firstHeading` 인 태그를 찾아보자.

```
heading = element.cssselect('.firstHeading')
```

다른 속성과 달리 클래스와 아이디는 약어로 표시를 할 수 있다. `.firstHeading` 은 클래스가 `firstHeading` 임을 나타낸다.

`#firstHeading` 은 아이디가 `firstHeading` 을 나타낸다.

```
heading   # 확인
```

```
[<Element h1 at 0x1cc43d63cc8>]
```

리스트 안에 `h1 Element` 가 들어있는 것을 확인할 수 있다. Element는 하나의 태그로 둘러쌓인 영역을 가리킨다.

5.6.3. 결과 확인

결과물을 확인해보자.

`heading` 의 첫번째 항목에 `.text_content()` 를 적용해 텍스트를 추출해보자.

```
title = heading[0].text_content()
```

```
title   # 확인
```

```
'HTML5'
```

위키피디아의 제목이 뽑혔다.