hci+d lab.

Week 09 • 소셜네트워크 데이터마이닝과 분석

Social Data Mining 01

Joonhwan Lee
human-computer interaction + design lab.

오늘 다룰 내용

- Crawling from websites

1. Crawling from Websites

웹 데이터 수집

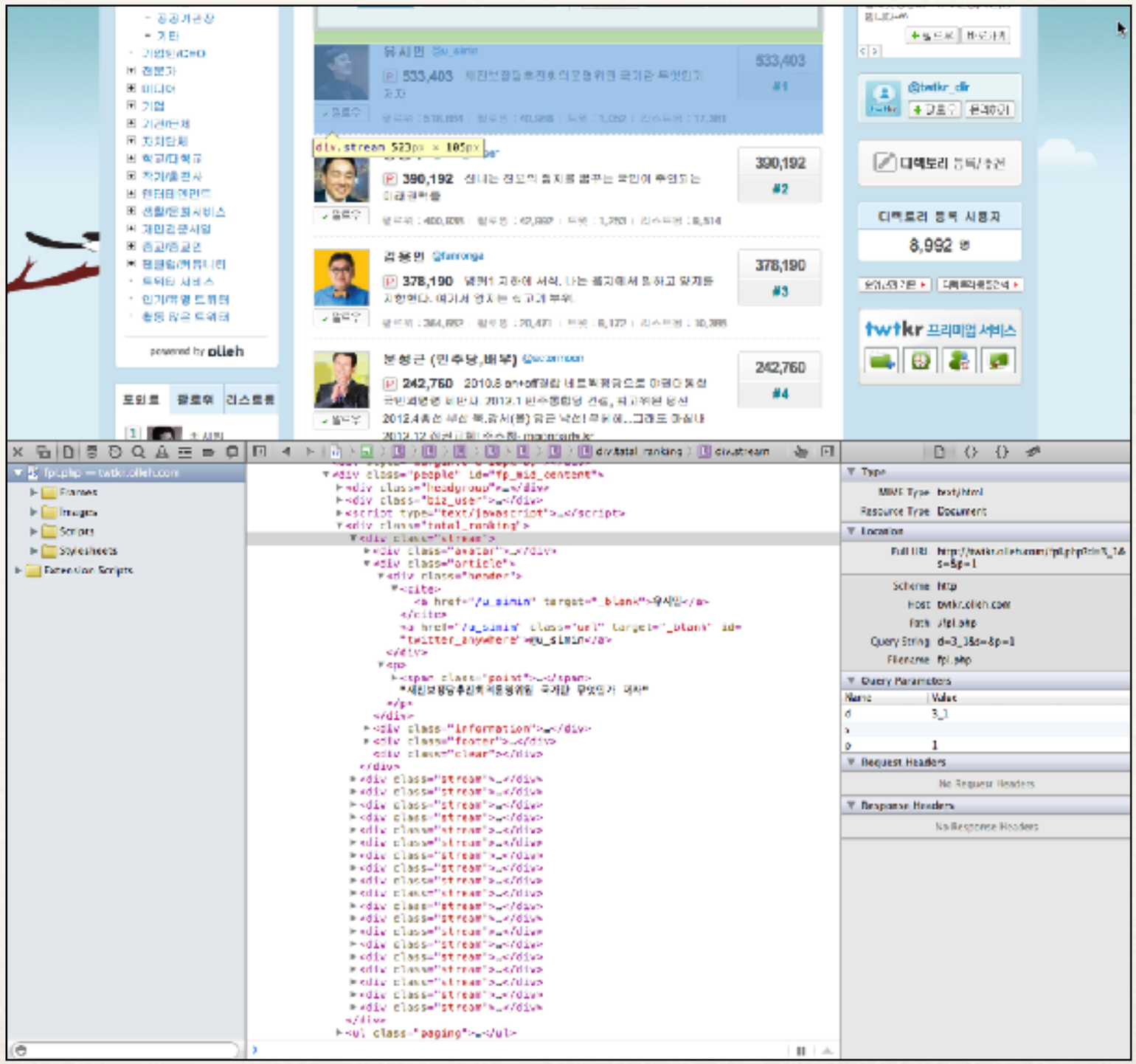
- RQ: 어떤 사람의 트위터 팔로워 구성을 통해 그 사람의 성향을 유추할 수 있을까?
- 예1: A라는 사람의 트위터 팔로워는 모두 500명, 그 중에 30% 정치인, 60%는 연예인 → 연예 정보에 관심이 많은 사람.
- 예2: A라는 사람이 팔로우하는 정치인 중, 보수성향 정치인 10%, 진보성향 정치인 90% → 진보적인 성향을 가진 사람.
- Q1: 팔로우하는 사람의 속성 (연예인인지, 정치인인지, 보수성향의 정치인이지 등)은 어떻게 수집하나..?

웹 데이터 수집

http://twtkr.com/fpl.php?d=3_1&n=

웹 데이터 수집

http://twtkr.com/fpl.php?d=3_1&n=



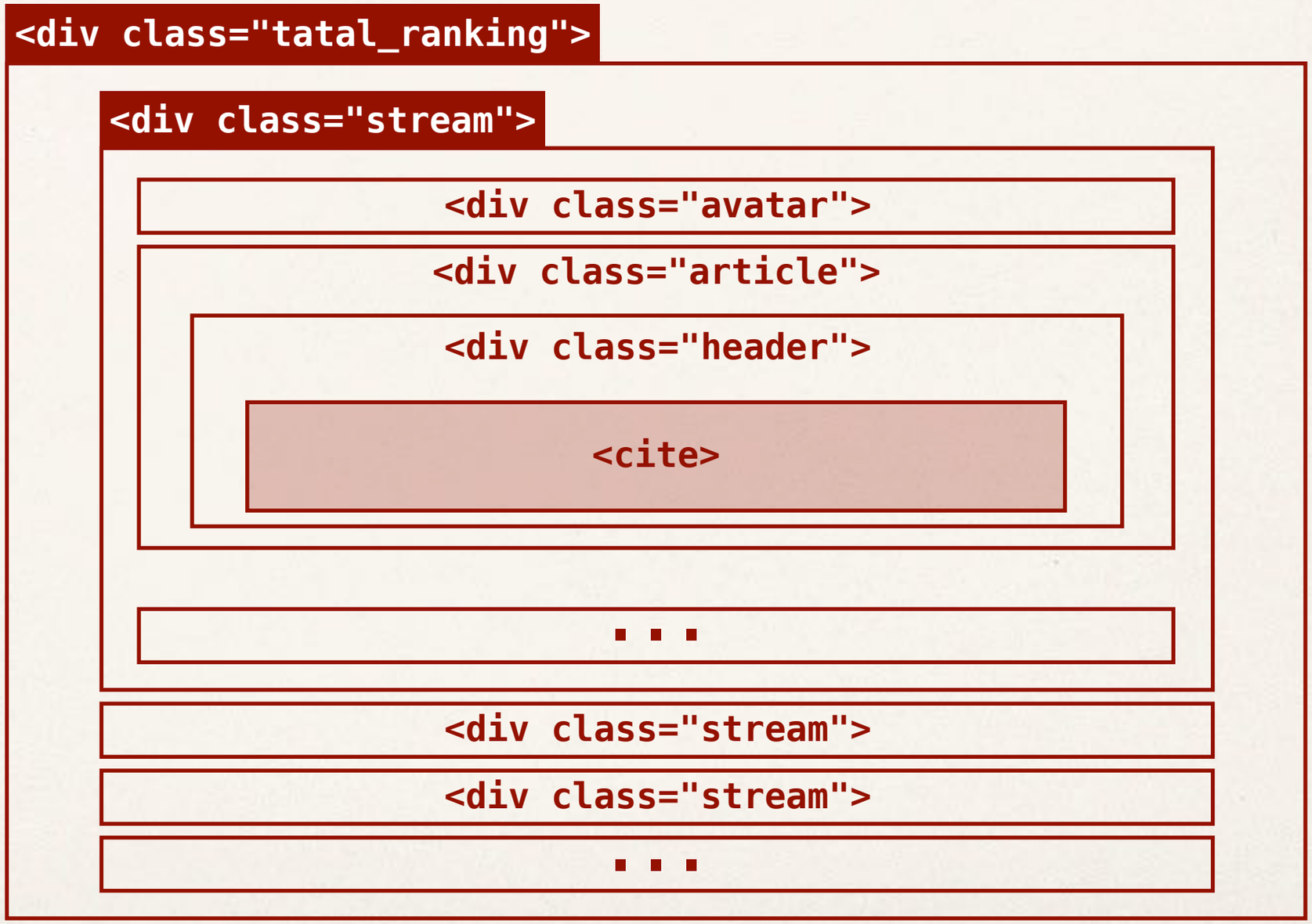
웹 데이터 수집

- 실습 1: 소스코드 분석
  - 수집하려는 웹 페이지의 소스를 분석하여, 필요한 데이터가 담긴 반복되는 패턴블럭을 찾아낸다.
  - 반복되는 패턴블럭의 계층 구조를 찾아내 각각의 요소를 정리한다.
  - 계층 구조 내에서 필요한 요소를 따로 찾아 정리한다.

- twtkr_example.html을 열고 주요한 데이터의 반복되는 패턴 블럭을 찾고, 내부 데이터를 구조화 하시오.

웹 데이터 수집

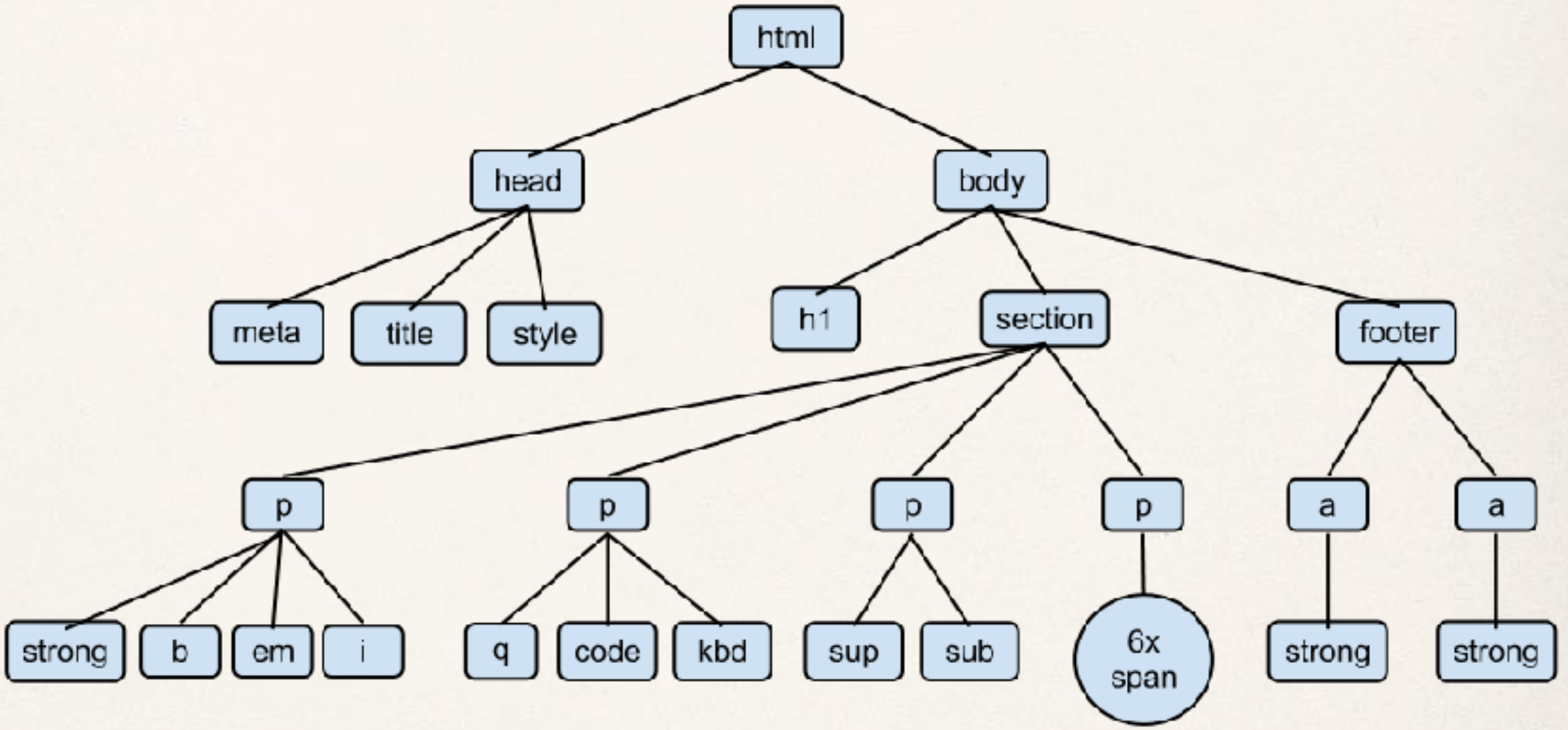
- 실습 1



BeautifulSoup을 이용한 웹페이지 수집 및 분석

- 웹 문서로 부터 특정한 데이터를 추출하기 위해서는 HTML 문서를 읽고 구조를 해석할 수 있는 소프트웨어가 필요.
- BeautifulSoup은 HTML, XML 등을 읽고 해석할 수 있는 소프트웨어 (parser)
  - BS4는 도큐멘트를 파싱한 후 DOM Tree 를 만든다.
- BeautifulSoup 설치
  - pip install beautifulsoup4

HTML Document 와 DOM Tree



HTML Document 와 DOM Tree

The Document

```
<html>
<body>
<h1>Title</h1>
<p>A <em>word</em></p>
</body>
</html>
```

The DOM Tree

```
DOCUMENT
└─ELEMENT: html
   ├─TEXT: '\n'
   ├─ELEMENT: body
   │  ├─TEXT: '\n'
   │  ├─ELEMENT: h1
   │  │  └─TEXT: 'Title'
   │  ├─TEXT: '\n'
   │  ├─ELEMENT: p
   │  │  ├─TEXT: 'A'
   │  │  └─ELEMENT: em
   │  │     └─TEXT: word
   │  └─TEXT: '\n'
   └─TEXT: '\n'
```

BS4를 이용한 HTML Parsing

- BeautifulSoup의 사용

```
> from bs4 import BeautifulSoup
> html_doc = "<html><body><h1>Mr. Belvedere Fan
Club</h1></body></html>"

> soup = BeautifulSoup(html_doc, "html.parser")
> soup
=> <html><body><h1>Mr. Belvedere Fan Club</h1></
body></html>

> print(soup.prettify())

> heading = soup.find_all(“h1")
=> [<h1>Mr. Belvedere Fan Club</h1>]

> heading[0].get_text()
=> 'Mr. Belvedere Fan Club'
```

BS4를 이용한 HTML Parsing

- find_all 의 사용법
  - `find_all("h1")`
    - <h1>~</h1> 태그 안의 내용
  - `find_all("div")`
    - ~ 태그 안의 내용
  - `find_all("div", class_="footer")`
    - ~ 태그 안의 내용
  - `find_all("div", id="footer")`
    - ~ 태그 안의 내용
  -
```
divs = soup.find_all("div", class_="header")
for div in divs:
    if div.a["href"] == "twitter_anywhere":
```
    - ~ 태그 안의 내용

BS4를 이용한 HTML Parsing

- find_all의 사용법
  - find_all이 반환하는 값은 array (한 페이지에 같은 요소가 여럿 있을 것을 가정하므로...)
  - 따라서 find_all이 수집한 데이터를 처리하기 위해서는 for-loop 등의 iterator 를 사용한다.

```
id_list = []
divs = soup.find_all("div", class_="header")
for div in divs:
    if div.a["href"] == "twitter_anywhere":
        id_list.append(div.a.text)
```

실습 2: twitter 아이디와 사용자 이름 수집

- twtkr_example.html 파일을 읽어 트위터 아이디와 사용자 이름을 수집해 보자. 수집된 id 에서 @ 기호를 삭제하여 출력한다.
- 예: u_simin, 유시민
- (참고) HTML 파일 불러오는 방법

```
with open("data/twtkr_example.html") as
file:
    html_doc = file.read()
```

웹에서 직접 데이터 수집

- 항상 저장된 페이지에서 파일을 수집할 수 없음.
- 실시간으로 웹페이지에 접속해서 저장된 페이지를 수집해야 함.
- 인터넷에 접속하여 페이지의 소스코드를 받아 처리하기 위해서는 다음과 같은 명령어를 사용.
  - ```
    import urllib.request
    with urllib.request.urlopen("http://
    twtkr.com/fpl.php?d=3&n=20") as url:
        doc = url.read()
    ```
    ```

Page Iterator

- 앞선 예제의 페이지 버튼을 눌러 다음 페이지로 이동.

  - page1: http://twtkr.com/fpl.php?d=3_1&n=
  - page2: http://twtkr.com/fpl.php?d=3_1&s=&p=2&n=20
  - page3: http://twtkr.com/fpl.php?d=3_1&s=&p=3&n=20
  - page4: http://twtkr.com/fpl.php?d=3_1&s=&p=4&n=20

- 공통점?

Page Iterator

- http://twtkr.com/fpl.php?d=3_1&s=&p=3&n=20
  - http://twtkr.com/fpl.php 뒤에 붙어 있는 문자와 숫자는 패러미터.
  - p=3: page number
  - n=20: page당 표시할 데이터의 수
  - d=3_1: ?????
  - s=: ?????

Page Iterator

- http://twtkr.com/fpl.php?**d=1_3**&n=20&**p=3**

Page Iterator

- 전체 페이지를 자동으로 수집하려면..?
  - twtkr의 정치인 페이지의 경우, 1 페이지 부터 54 페이지까지 존재함.
  - 불필요한 패러미터를 삭제한 후 loop 를 사용하여 page 자동 전환

```
base_url = "http://twtkr.com/fpl.php?
d=3_1&n=20&p="

for i in range(1, 55):
    print(base_url + str(i))
```

실습 3: Page Iterator 사용하여 모든 아이디 수집

- twtkr의 디렉토리에서 정치인의 트위터 아이디와 이름을 모두 수집하는 프로그램을 작성하시오.
  - 주소: http://twtkr.com/fpl.php?d=3_1&s=&p=1&n=20
  - Step 1: url 정리 및 공통 요소 추출
  - Step 2: page iterator 작성
  - Step 3: 실습 2에서 작성한 코드 사용하여 트위터 아이디 및 이름 수집.
  - Step 4: 화면에 표시. (출력)

Page Iterator 수정

- 페이지가 끝나는 시점을 자동으로 알고 싶다.
  - 페이지를 수집할 때, 모든 페이지의 끝 번호를 알수 있을까?
  - twtkr에서도 카테고리마다 페이지 숫자가 모두 다름
    - 정치인: 54페이지, 연예인: 29페이지
  - 끝나는 페이지의 특징을 찾아보자.

화살표가 없다!

Page Iterator 수정

- 화살표를 그려주는 태그
  - `<li class="btn btn_next">`
  - 맨 마지막 페이지에는 해당 태그가 존재하지 않음. 이는 다음과 같이 표현 가능.

```
len(soup.find_all("li", class_="btn
btn_next”)) == 0
```

- 따라서 현재 불러온 페이지에 <li class="btn btn_next"> 가 존재하지 않고, <ul class="paging"> 아래 있는 페이지 숫자가 현재 URL에 사용한 페이지와 같으면 현재 페이지가 마지막 페이지.

실습 4: 페이지 자동 종료

- **if len(sp.find_all("li", class_="btn btn_next")) == 0:**

- **uls = sp.find_all("ul", class_="paging")**
  for ul in uls:
  lis = ul.find_all('li')
  last_page_no = int(lis[len(lis)-1].text)

- 위의 코드를 이용하여 has_no_more_pages() 함수를 만든 후에,

```
while not_lastpage:
    ...
    if has_no_more_pages(soup, pageno):
        not_lastpage = False
```

를 이용하여 자동으로 페이지 이동하며 데이터 수집.

Questions?