

Python Programming

Module: Web Crawling

Web Crawling

- ▶ Web Scrapping
- ▶ 컴퓨터 소프트웨어 기술로 웹사이트들에서 원하는 정보를 추출하는 것
- ▶ 방대한 분량의 웹문서를 사람이 구별하여 모으는 일은 불가능
 - ▶ 소프트웨어가 일하게 하라

Python Web Crawling

Get Simple Text from Web

웹 크롤링을 지원하는 기술

: HTTP Request and Response

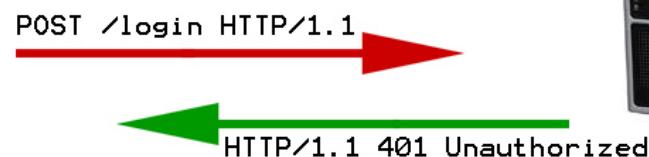
HTTP : 비연결 지향 프로토콜

HTTP MODEL

Client



Client



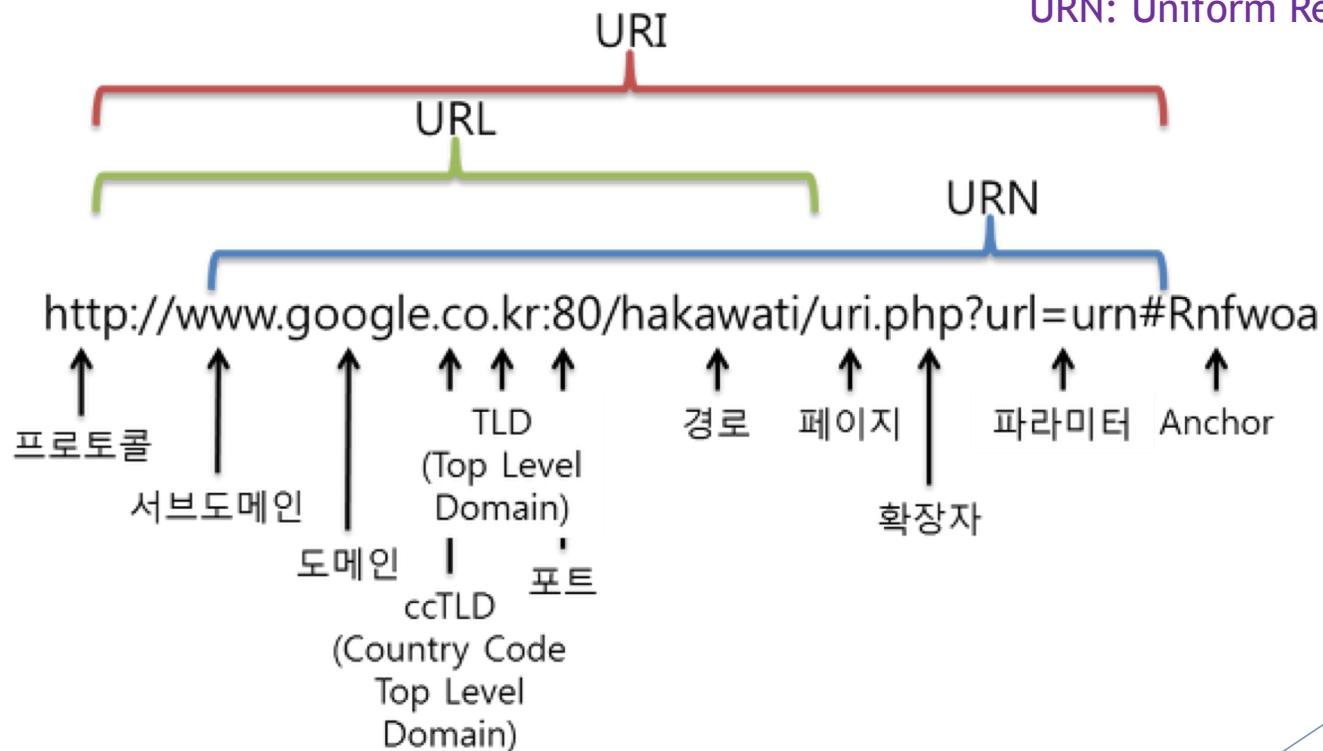
Server



웹 크롤링을 지원하는 기술

: URL and URI

URL : Uniform Resource Locator
URI : Uniform Resource Indicator
URN: Uniform Resource Name



웹 크롤링을 지원하는 기술

: `urllib`

- ▶ URL handling module
 - ▶ `urllib.request` : URL을 열고 읽어온다
 - ▶ `urllib.error` : `urllib.request`에서 일으키는 예외들을 포함
 - ▶ `urllib.parse` : url을 파싱하는 기술
 - ▶ `urllib.robotparser` : robots.txt 파일을 파싱

`robot.txt` : 검색엔진에게 수집을 제한하는 여러가지 정보를 담고 있는 파일

urllib를 이용한 텍스트 불러오기

```
# urllib로 웹 사이트 불러오기
import urllib.request

page = urllib.request.urlopen("http://www.gutenberg.org/cache/epub/12429/pg12429.txt")
print(page.read(100)) # 100글자만 불러오자
# 일부 깨지는 문자가 발생한다
# read가 완료되면 버퍼가 비어버리므로 다시 불러와야 한다
encoded = page.read().decode("utf-8")
print(encoded)print(encoded[:100]) # 100글자만 받아오자 너무 크다
```

받아온 텍스트를 대상으로 정규식 등을 이용,
분석을 시도한다

Python Web Crawling

with BeautifulSoup4

BeautifulSoup?

- ▶ Python의 대표적인 HTML 파싱 엔진
- ▶ urllib.request는 전달받은 response를 str로 전달하는 역할에 그친다
- ▶ BeautifulSoup은 이 텍스트를 객체 구조로 변환하여 ‘의미있는’ 정보를 추출해 낼 수 있다

설치 : pip install bs4

BeautifulSoup

: 다음 뉴스를 받아와 보자

```
from urllib.request import Request, urlopen
from bs4 import BeautifulSoup

# HTTP GET Request
req = urlopen('http://media.daum.net')

# 먼저 req가 정상 코드를 받아왔는지 확인한다
print(req.getcode()) # HTTP 200은 정상값이다

if req.getcode() == 200:
    # html을 받아오자
    html = req.read()
    print(html) # 인코딩이 깨진다.

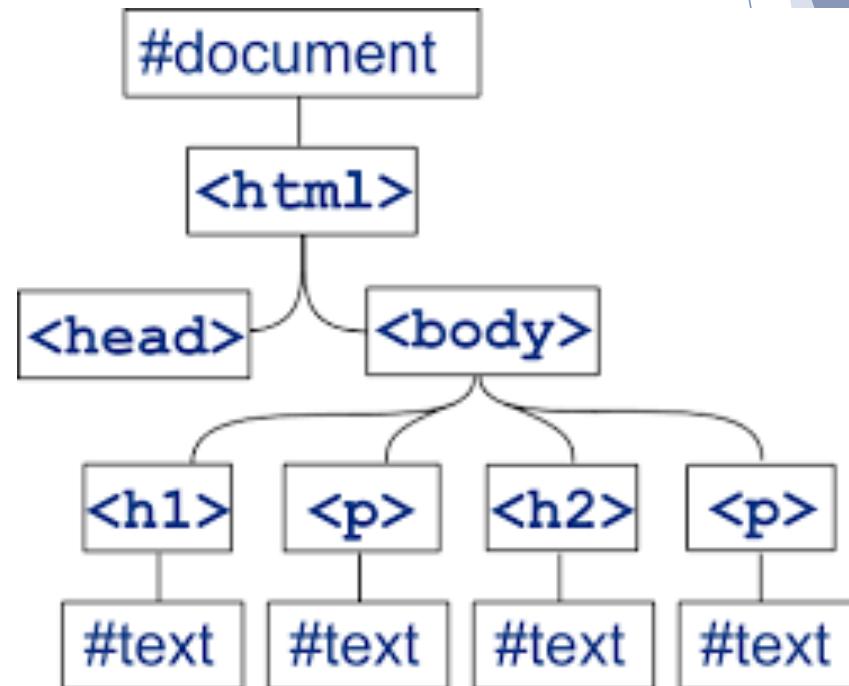
    html = html.decode("utf-8")
    print(html) # 정상이다
else:
    print("HTTP ERROR")

# 이제 soup을 만들어 즐겨보자
soup = BeautifulSoup(html, "html.parser") # soup을 만들었다
# print(soup.prettify())

print(soup.title)
# 타이틀에서 텍스트를 뽑아오자
print(soup.title.text)
```

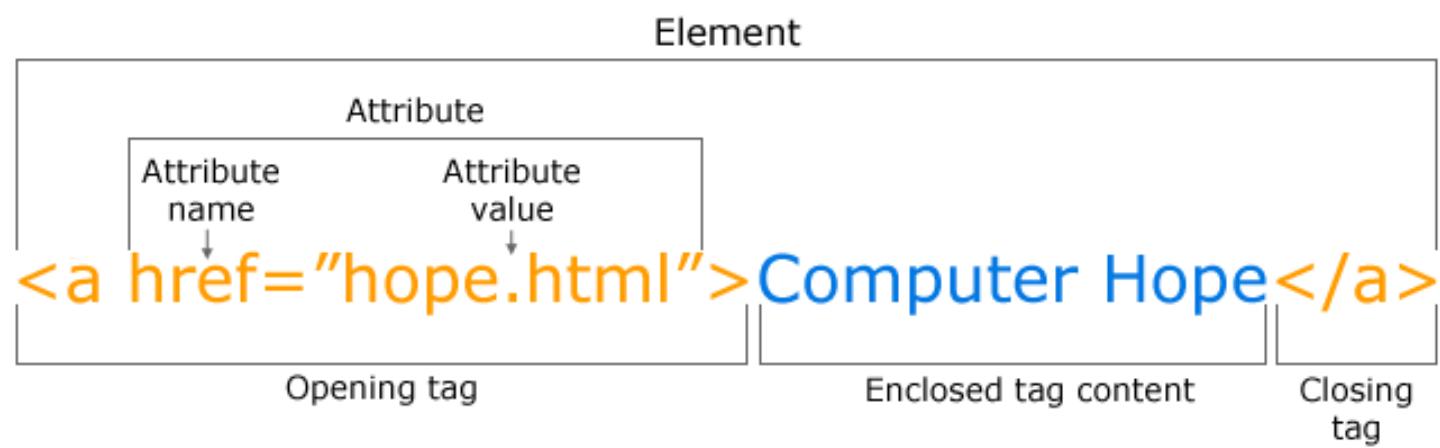
DOM의 간단한 이해

- ▶ HTML은 Document Object Model 이라는 Tree 구조로 구성되어 있다
- ▶ HTML은 XML의 하위 집합
 - ▶ XML과 같이 Contents, Attribute 등을 이용, 내용과 속성에 접근할 수 있다
 - ▶ 단, XML처럼 강한 구속력이 작용하지는 않는다



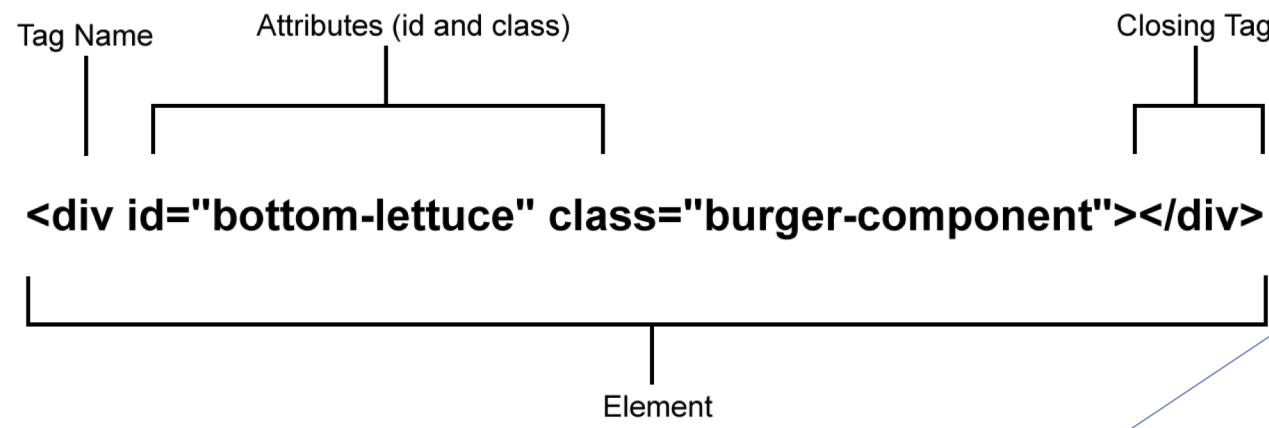
HTML에 대한 간단한 이해

Breakdown of an HTML Tag



CSS class와 id

- ▶ HTML 태그는 id와 class를 가질 수 있다
 - ▶ id : 페이지 내 유일한 식별값
 - ▶ class : 중복 사용될 수 있는 스타일 값. 여러 개를 조합할 수도 있다



Understanding CSS Selectors

: Basic form

Selector	Example	Example description	CSS
<u>.class</u>	.intro	Selects all elements with class="intro"	1
<u>#id</u>	#firstname	Selects the element with id="firstname"	1
<u>*</u>	*	Selects all elements	2
<u>element</u>	p	Selects all <p> elements	1
<u>element,element</u>	div,p	Selects all <div> elements and all <p> elements	1
<u>element element</u>	div p	Selects all <p> elements inside <div> elements	1
<u>element>element</u>	div>p	Selects all <p> elements where the parent is a <div> element	2
<u>element+element</u>	div+p	Selects all <p> elements that are placed immediately after <div> elements	2

Understanding CSS Selectors

: Attribute Selectors

Selector	Example	Example description	CSS
<u>[attribute]</u>	[target]	Selects all elements with a target attribute	2
<u>[attribute=value]</u>	[target=_blank]	Selects all elements with target="_blank"	2
<u>[attribute~=value]</u>	[title~=flower]	Selects all elements with a title attribute containing the word "flower"	2
<u>[attribute =value]</u>	[lang =en]	Selects all elements with a lang attribute value starting with "en"	2
<u>[attribute^=value]</u>	a[src^="https"]	Selects every <a> element whose src attribute value begins with "https"	3
<u>[attribute\$=value]</u>	a[src\$=".pdf"]	Selects every <a> element whose src attribute value ends with ".pdf"	3
<u>[attribute*=value]</u>	a[src*="w3schools"]	Selects every <a> element whose src attribute value contains the substring "w3schools"	3

Understanding CSS Selectors

: Extended

Selector	Example	Example description	CSS
<code>:before</code>	<code>p:before</code>	Insert content before the content of every <code><p></code> element	2
<code>:after</code>	<code>p:after</code>	Insert content after every <code><p></code> element	2
<code>:first-child</code>	<code>p:first-child</code>	Selects every <code><p></code> element that is the first child of its parent	2
<code>:last-child</code>	<code>p:last-child</code>	Selects every <code><p></code> element that is the last child of its parent	3
<code>:nth-child(n)</code>	<code>p:nth-child(2)</code>	Selects every <code><p></code> element that is the second child of its parent	3
<code>:nth-last-child(n)</code>	<code>p:nth-last-child(2)</code>	Selects every <code><p></code> element that is the second child of its parent, counting from the last child	3
<code>:nth-of-type(n)</code>	<code>p:nth-of-type(2)</code>	Selects every <code><p></code> element that is the second <code><p></code> element of its parent	3
<code>:nth-last-of-type(n)</code>	<code>p:nth-last-of-type(2)</code>	Selects every <code><p></code> element that is the second <code><p></code> element of its parent, counting from the last child	3

페이지 분석

: 개발자 도구의 도움이 필요해(Chrome의 경우)

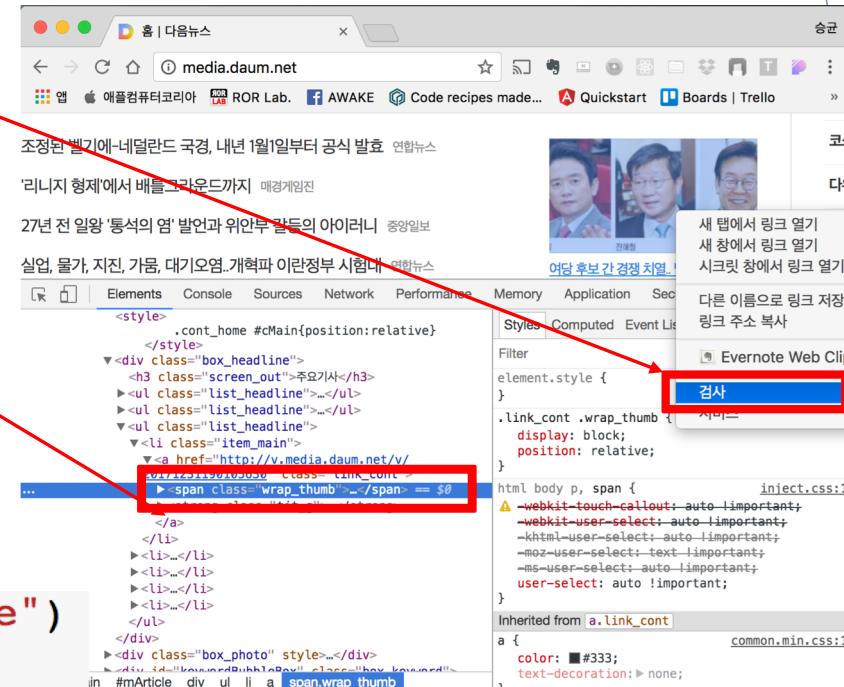
1. 원하는 요소에서 우클릭
검사 메뉴를 실행한다

2. selector를 찾아낸다

```
articles = soup.select("#mArticle")
print(articles)
```

3. Selector 획득

#mArticle



BeautifulSoup

: CheatSheet

Basic

```
# https://www.crummy.com/software/BeautifulSoup/bs4/doc/
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc, 'html.parser')

soup.title # <title>The Dormouse's story</title>
soup.title.name # u'title'
soup.title.string # u'The Dormouse's story'
soup.title.parent.name # u'head'

#various finder
css_soup.select("p.strikeout.body") # css finder
soup.p # <p class="title"><b>The Dormouse's story</b></p>
soup.p['class'] # u'title'
soup.a # <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>
soup.find_all('a') # [<a ..>, ..]
soup.find(id="link3") # <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
for link in soup.find_all('a'):
    print(link.get('href')) # http://example.com/elsie, # http://example.com/lacie
```

<http://akul.me/blog/2016/beautifulsoup-cheatsheet/>

Make soup

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(open("index.html"))
soup = BeautifulSoup("<html>data</html>")
```

Output

```
# HTML
soup.prettify() #pretty print
str(soup) # non-pretty print

# String
soup.get_text() #all text under the element
```

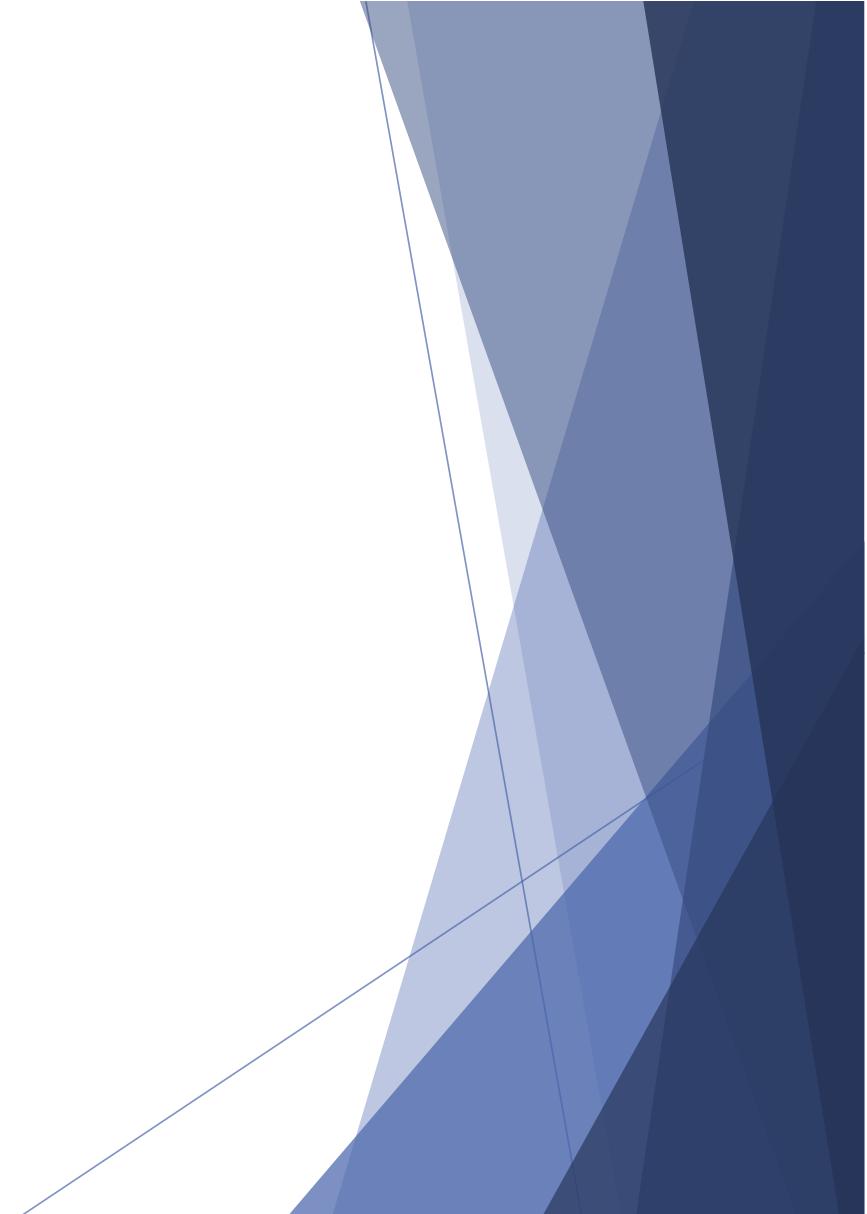
Search

```
-----
# css selector
-----
css_soup.select("p.strikeout.body")
soup.select("p nth-of-type(3)") # 3rd child
soup.select("head > title")
soup.select("p > a:nth-of-type(2)")
soup.select("p > #link1") # direct child
soup.select("#link1 ~ .sister") # sibling
soup.select('a[href]') # existence of an attribute
soup.select_one(".sister")

# attribute value
soup.select('a[href="http://example.com/elsie"]') # exact attribute
soup.select('a[href^="http://example.com/]') # negative match
soup.select('a[href$="tillie"]') # end match
soup.select('a[href*=".com/el"]') # middle match
```

Real Life Example

네이버 최신 영화 리스트

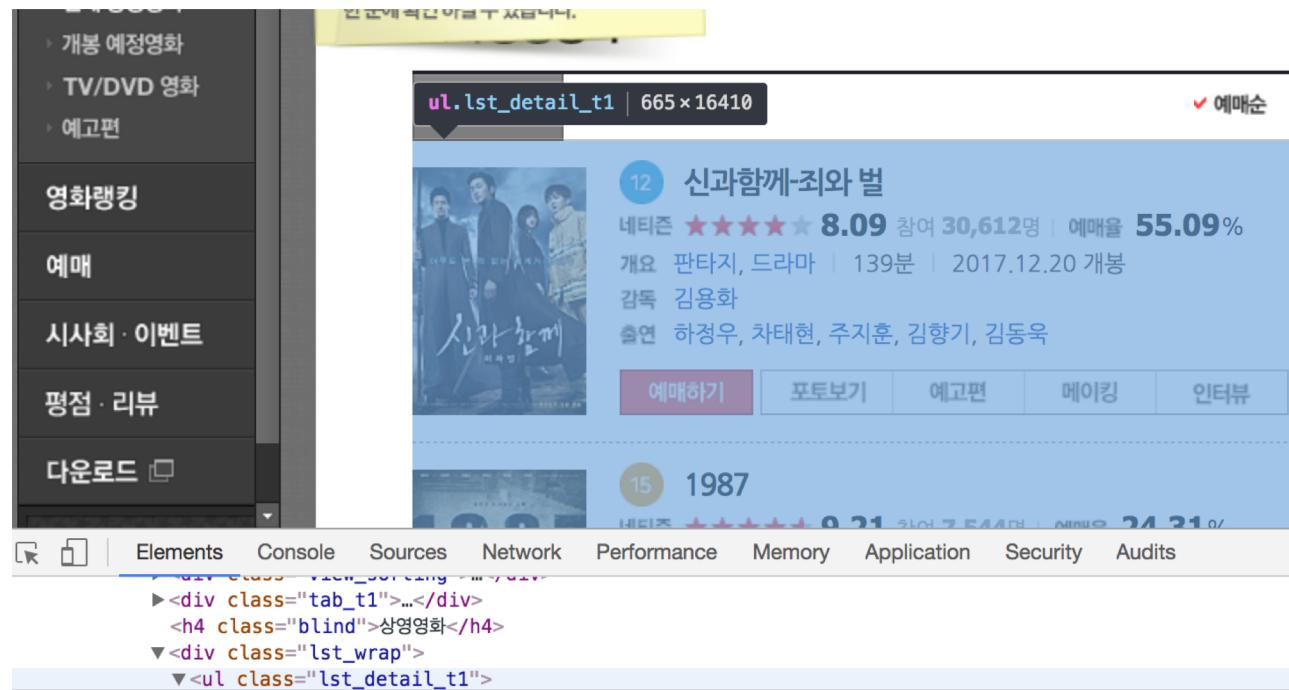


수집을 위한 준비

- ▶ URL 준비
 - ▶ Naver 홈 > 영화 > 상영작-예정작
 - ▶ <http://movie.naver.com/movie/running/current.nhn>
- ▶ HTML을 불러온다

```
from urllib.request import Request, urlopen
from bs4 import BeautifulSoup
request = Request("http://movie.naver.com/movie/running/current.nhn")
response = urlopen(request)
html = response.read()
# print(html[: 256]) # 코드가 깨진다... 인코딩을 변경하자
html = html.decode("utf-8")
# print(html[:256]) # 코드가 정상적이다
# 이제 BeautifulSoup으로 html을 정제하자
bs = BeautifulSoup(html, "html.parser")
print(bs.title)
print(bs.title.name)
# 이쁘게 표현해보자|print(bs.prettify() [:1024])
```

영화 리스트의 Selector를 찾아보자



찾아낸 셀렉터 : class = “lst_detail_t1”인 ul

```
currents = bs.find("ul", attrs = {"class": "lst_detail_t1"})
```

수집 로직의 작성

```
for child in currents:  
    try:  
        titles = child.find("dt", attrs={"class": "tit"})  
        for title in titles:  
            try:  
                if title.name == "a":  
                    print(title.text, title['href'])  
            except:  
                pass  
    except:  
        pass
```

수집 로직의 작성

: Renewal Version

Using select method

```
from urllib.request import Request, urlopen
from bs4 import BeautifulSoup

request = Request("http://movie.naver.com/movie/running/current.nhn")
# print(dir(request))
# print(request.host)

response = urlopen(request)
html = response.read().decode("utf-8")
# print(html[:256])

# 이제 BeautifulSoup으로 html을 정제하자
bs = BeautifulSoup(html, "html.parser")

# print(bs.select(".lst_detail_t1"))
current_movies = bs.select(".lst_detail_t1 > li")
# print(type(current_list))

# print(current_list[0])

# for info in current_list:
#     print(info.prettify())

for movie in current_movies:
    titles = movie.select(".lst_dsc > .tit > a")

    for title in titles:
        print(title.text, title['href'])
```

Real Life Example

Naver API Call

Naver API

- ▶ 네이버에서 제공하는 각종 기능을 개발자에게 공개
- ▶ <https://developers.naver.com/>
- ▶ 지도, 뉴스, 검색 등 다양한 기능들을 외부에 공개하고 있으며, 네이버 사용자 관련된 기능들이 일부 오픈되어 있다



API 사용을 위한 준비

- ▶ API 사용을 위한 앱을 만들어야 한다

Products

Documents

Application

NAVER D2

Support

내 애플리케이션

애플리케이션 등록

Clova Platform Console β API 제휴 신청

Application 등록

우리는 일단 file://dummy.py로 두자

애플리케이션 이름
py-crawl| ✓

- 네이버 아이디로 로그인할 때 사용자에게 표시되는 이름이므로 가급적 10자 이내의 간결한 이름을 사용해주세요.
- 40자 이내의 영문, 한글, 숫자, 공백문자, "-", "_"만 입력 가능합니다.

사용 API
선택하세요. ▼ ✓
검색 X

환경 추가
WEB 설정 X ^
웹 서비스 URL (최대 10개)
file:///dummy.py + ✓

- 테스트 품 우측 끝의 '+' 버튼을 누르면 행이 추가되며, '-' 버튼을 누르면 행이 삭제됩니다.
- http와 https는 구분하지 않습니다.
- www는 빼고 입력해 주세요. 예) http://naver.com
- 서브 도메인이 있으면 대표 도메인만 입력해 주세요. (예: http://naver.com)
- 하이브리드 앱은 location.href 객체 출력 값을 입력하면 됩니다. (예: file:///로컬URL)

Client ID와 Client Secret 확인

애플리케이션 정보

Client ID	_DA93V_MBJrbEfDGz90k
Client Secret 보기

매뉴얼의 확인

Documents > 서비스 API > 검색

서비스 API

데이터랩

검색

단축 URL

이미지캡차

음성캡차

네이버 공유하기

모바일 앱 연동

네이버 오픈메인

2. API 기본 정보

메서드	인증	요청 URL	출력 포맷
GET	-	https://openapi.naver.com/v1/search/news.xml	XML
GET	-	https://openapi.naver.com/v1/search/news.json	JSON

3. 요청 변수 (request parameter)

요청 변수	타입	필수 여부	기본값	설명
query	string	Y	-	검색을 원하는 문자열로서 UTF-8로 인코딩한다.
display	integer	N	10(기본값), 100(최대)	검색 결과 출력 건수 지정
start	integer	N	1(기본값), 1000(최대)	검색 시작 위치로 최대 1000까지 가능

기본적인 수집

```
import urllib.request

client_id = "_DA93V_MBjrbEfDGz90k"
client_secret = "A3AGuyY2iH"

enc_text = urllib.parse.quote("북핵")
url = "https://openapi.naver.com/v1/search/news.json?query={}".format(enc_text)

request = urllib.request.Request(url)
request.add_header("X-Naver-Client-Id", client_id)
request.add_header("X-Naver-Client-Secret", client_secret)
response = urllib.request.urlopen(request)
rescode = response.getcode()

print(rescode)

if (rescode == 200):
    response_body = response.read()
    print(response_body.decode('utf-8'))
else:
    print("Error Code: %d" % rescode)
```

JSON

▶ JavaScript Object Notation

- ▶ 최근에는 API 결과값을 전송할 때 XML보다 JSON을 더 많이 쓴다
- ▶ 경량이고, 구조가 단순하여 쉽게 처리할 수 있다
- ▶ 문자형, 수치형, 논리값, 배열, 사전 등 다양한 데이터형 지원

```
{  
    hey: "guy",  
    anumber: 243,  
    - anobject: {  
        whoa: "nuts",  
        - anarray: [  
            1,  
            2,  
            "thr<h1>ee"  
        ],  
        more: "stuff"  
    },  
    awesome: true,  
    bogus: false,  
    meaning: null,  
    japanese: "明日がある。",  
    link: http://jsonview.com,  
    notLink: "http://jsonview.com is great"  
}
```

수집 데이터를 json으로 가공

: 수집 로직의 개선

```
if (rescode == 200):
    response_body = response.read()

    json_rt = response_body.decode("utf-8")
    py_rt = json.loads(json_rt)

    news_list = py_rt['items']

    for news in news_list:
        print("title: {title} @ {pubDate}".format_map(news))

    # =====
    for news in news_list:
        news['title']
        print("title: {title} @ {pubDate}".format_map(news))

else:
    print("Error Code: %d" % rescode)
```

Happy Crawling ☺