

2018-2

크롤링

3기 이재경

CONTENTS

1. 웹의 구성 요소
2. BeautifulSoup4
3. Selenium
4. Crawling
5. 부록
6. Quest

웹의 구성 요소

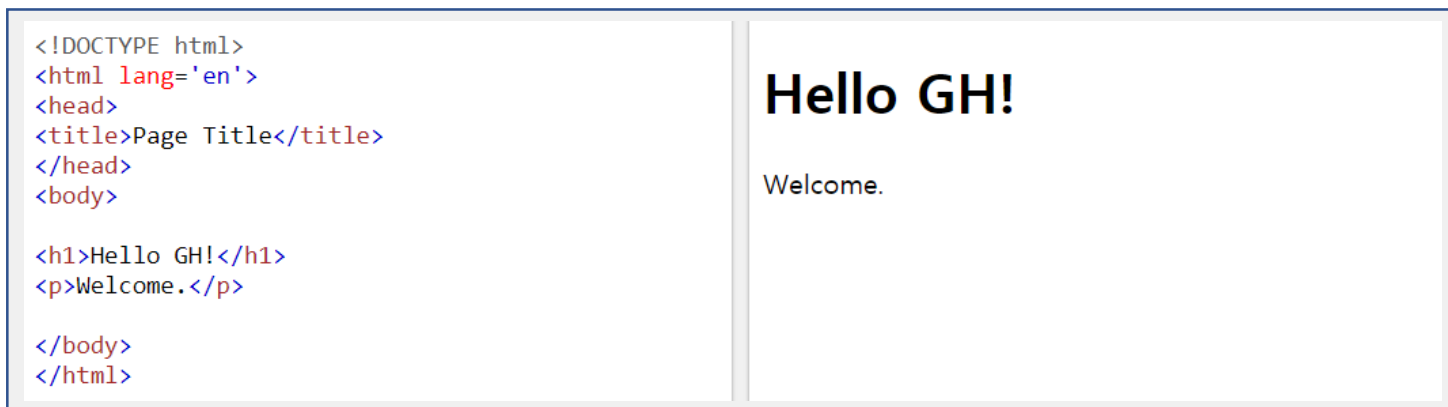
웹이란 **월드 와이드 웹**(World Wide Web, WWW)의 약자로 인터넷에 연결된 클라이언트들이 정보를 공유할 수 있는 공간을 의미

웹 페이지는 크게 세 가지로 이루어져 있습니다. 레이아웃을 잡아주는 **HTML**, 기능을 붙여넣어 주는 **JavaScript**, 화면을 꾸며주는 **CSS**이다.

크롤러를 만들기 위해서는 목표가 되는 웹 페이지의 구조를 이해하고 그 안에서 원하는 데이터를 찾아 내는데 도움이 되고자, 간단하게 웹 페이지의 구성 요소인 HTML, CSS, 그리고 JavaScript에 대해서 이야기 해보고자 한다.

웹의 구성 요소 - HTML

HTML은 **하이퍼텍스트 마크업 언어**(HyperText Markup Language)의 약자로 일반적인 언어와는 다르며, 데이터의 구조나 형식을 지정하는 언어로 요소,태그,속성 등으로 이루어져 있다.



[그림 1-1]

[그림 1-1]은 웹 페이지의 기본 구조입니다. 꺾쇠(<>)로 감싸져 있는 것을 **태그**라고 한다. 태그는 여는 태그<>, 와 닫는 태그 </> 한쌍으로 구성된다.

태그를 만들 때 태그 **이름**과 **속성**으로 태그를 만들 수 있다.

html 태그에는 lang이라는 속성을 가지고 있다.

웹의 구성 요소 - HTML

HTML 태그



[그림 1-2]

HTML에는 어떤 태그들이 있는지 몇가지 알아보겠다.
이런 태그들은 body 태그 안에 포함된다.

p 태그

문단을 나타내는 태그

h 태그

폰트 크기를 설정하는 태그; h 태그는 숫자가 작을수록 폰트가 커지며 1에서 6까지 지원합니다.

ul, ol, li 태그

리스트를 만드는 태그

웹의 구성 요소 - HTML

HTML 태그



```
<!DOCTYPE html>
<html lang='en'>
<head>
<title></title>
</head>
<body>

  <table>
    <tr>
      <td>1</td>
      <td>2</td>
      <td>3</td>
    </tr>
    <tr>
      <td>1</td>
      <td>2</td>
      <td>3</td>
    </tr>
  </table>
  <table>
    <thead>
      <tr>
        <th>11</th>
        <th>22</th>
        <th>33</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>111</td>
        <td>222</td>
        <td>333</td>
      </tr>
    </tbody>
  </table>
</body>
</html>
```

1	2	3
11	22	33

[그림 1-3]

table 태그

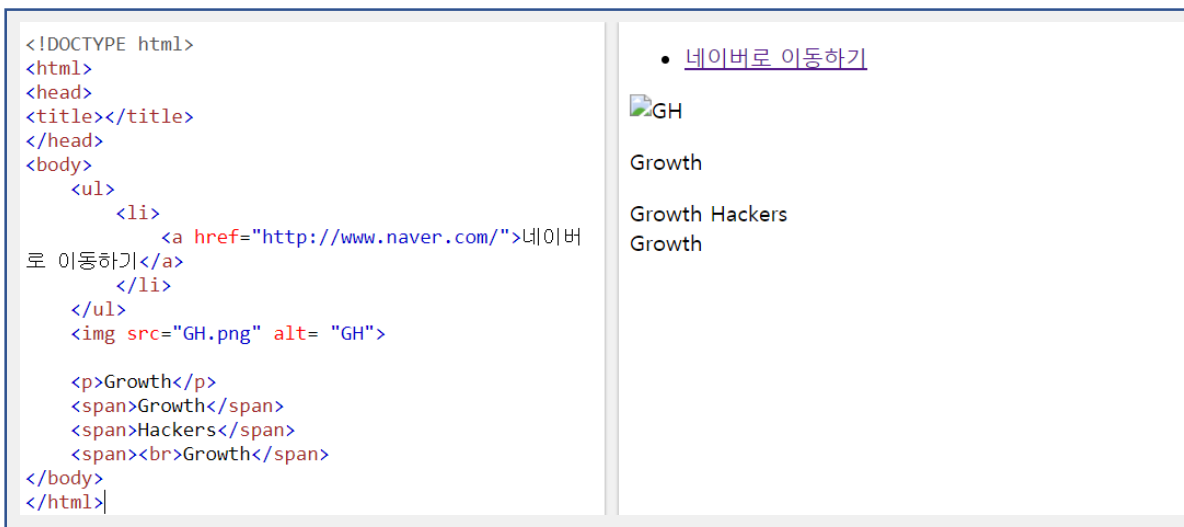
table 태그를 이용하여 표를 표현할 수 있다. 과거에는 table 태그를 이용하여 테이블 뿐만 아니라 페이지의 레이아웃을 잡는 역할로 많이 사용한다.

table 태그는 내부적으로 thead, tbody를 가지고 있을 수 있으며, tr을 이용하여 행을 표현하고 td와 th를 이용하여 각 행의 열을 표현한다.

th태그 와 td태그는 모두 한 행에서 열을 나타내지만 th태그를 줄 경우 열 가운데 정렬과 굵은 글씨체가 된다.

웹의 구성 요소 - HTML

HTML 태그



[그림 1-4]

a 태그

다른 페이지로 이동할 때의 태그

a 태그는 href 속성으로 가지며 이는 이동하게 될 링크입니다.

img 태그

이미지를 띄우게 하는 태그

img는 src와 alt 속성을 가집니다. src를 통하여 이미지를 불러오게 되며, alt를 통하여 이미지가 정상적으로 불러지지 않았을 때, 텍스트로 대체하여 나타냅니다.

img태그는 닫는 태그를 요구하지 않습니다.

span 태그

p 태그와 같이 텍스트를 추가적으로 넣을 수 있습니다.

다만 문단이 바로 나누어 지는 p 태그와 달리 span 태그는 옆으로 나열되게 됩니다.
, </br> 태그를 사용하여 p 태그와 같이 사용할 수 있으며, 이들 또한 굳이 쌍을 이룰 필요는 없습니다.

웹의 구성 요소 - HTML

HTML 태그

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
  <div>
    <h1>첫 번째 div</h1>
    <span>이재경</span>
  </div>

  <div>
    <h1>두 번째 div</h1>
  </div>
</body>
</html>
```

첫 번째 div

이재경

두 번째 div

[그림 1-4]

div 태그

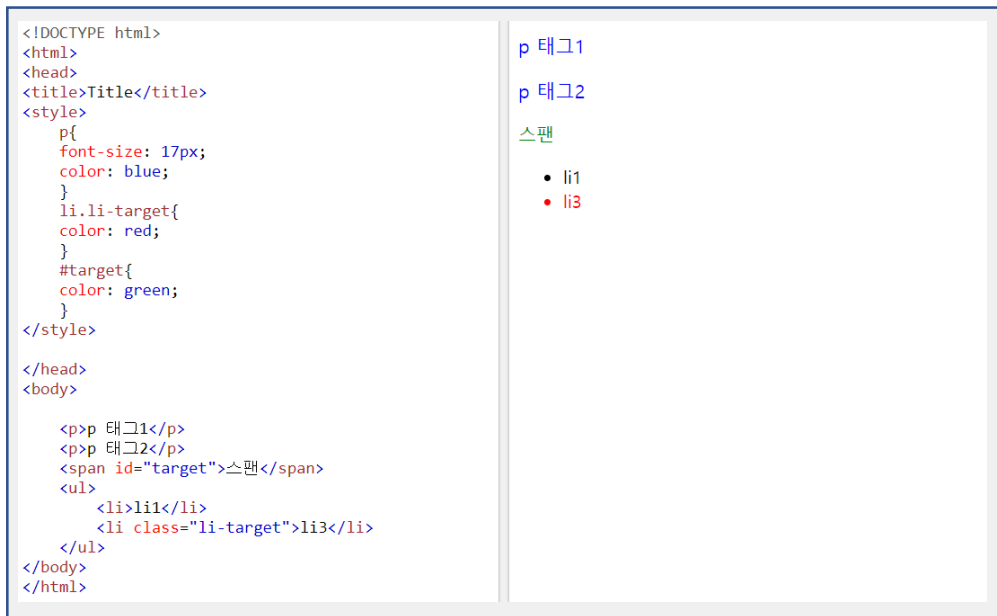
가장 자주 사용하는 태그로, 태그는 특정 기능이 있는 것이 아니라 단순히 영역을 잡아서 레이아웃을 만드는 역할로 사용한다.

div 태그를 사용하면 눈에는 보이지 않지만 그 하위 태그들의 영역을 잡아 주는 역할을 한다.

예전에는 div 태그를 이용하지 않고 table 태그를 이용하여 웹 사이트 구조를 잡았었기에 간혹 table을 중첩해서 만든 사이트들이 존재한다.

웹의 구성 요소 – CSS와 JavaScript

CSS와 JavaScript



[그림 1-4]

CSS(Cascading Style Sheets)은 웹 사이트를 꾸며주는 역할을 한다. CSS를 이용하여 꾸미기 위해 특정 요소에 접근하는 것을 셀렉터(selector)라고 부르며, 특정한 태그를 이용하거나 id와 class라는 속성을 이용하는 방식이 크롤러에서 똑같이 사용 가능하다.

class는 마침표(.)를 붙여 class임을 나타냅니다. class를 이용하면 태그와 상관없이 같은 class를 공유하는 항목에 접근한다.

id는 class와 달리 id값이 고유해야 하기에 중복 사용하지 않기를 권장하고 있다.

JavaScript는 script 태그를 이용하여 웹 사이트에 기능을 넣어줄 수 있다. script 태그는 head에 들어가도 되지만, body의 가장 하단 부분에 넣어주는 것을 권장하고 있다.

Beautifulsoup4

urllib

bs4 (BeautifulSoup4) 모듈은 요청 모듈로 가져온 HTML 코드를 파이썬에서 사용 가능한 객체로 바꿔준다. 따라서 bs4에 대해서 알아보기 앞서 파이썬에서 웹을 요청할 수 있는 라이브러리인 urllib에서 알아야한다.

파이썬에는 urllib라고 하는 요청 모듈이 있으며 이는 내장 모듈이기 때문에 설치할 필요가 없다. urllib은 객체를 만들어 요청하게 된다.

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

url = "http://ghmkt.kr/" #요청을 하고자 하는 웹 사이트입니다.

req = Request(url) #요청 객체를 만드는 부분입니다.
page = urlopen(req) #만들어진 요청 객체를 이용하여 요청하는 부분입니다.

print(page)
print(page.code)
print(page.headers)
print(page.url)
print(page.info().get_content_charset())
print(page.read()) #urllib은 read() 함수를 이용하여 HTML을 바이너리 형태로 가져옵니다.
```

Beautifulsoup4

urllib

`urllib.request.urlopen()` 을 통하여 보게 되는 에러의 대다수는 아래 4 종류에 속한다.

`urllib.error.HTTP Error:HTTP Error 404: Not Found`

:해당 페이지가 없다. 주소 확인이 요망된다.

`urllib.error.HTTP Error:HTTP Error 500: Internal Server Error`

:해당 웹페이지 서버에 이상이 있는 것이므로, 주소 문제가 아니다.

`urllib.error.HTTP Error:HTTP Error 503 : Service Unavailable`

:서버로 접속하는 것을 막아 놓은 것이기에, Selenium으로 해결 가능할 수 있다.

`urllib.error.URLError:<urlopen erro [EMO 11001] getaddrinfo failed>`

1. 사이트가 없어졌다. 2. 방화벽 문제 3. DNS 서버와 접속 문제. 4. 프록시와의 DNS 차이

Beautifulsoup4

bs4

bs4 모듈에 존재하는 **BeautifulSoup()** 함수를 이용하면 문자열을 파이썬에서 사용 가능한 객체로 만들어 준다.
이 함수는 두 개의 인자를 input으로 요구한다. 첫 번째 인자는 대상이 되는 문자열, 두 번째 인자는 파서를 입력해 주어야 한다.

파서란, 원시 코드인 순수 문자열을 객체를 해석할 수 있도록 분석하는 것을 의미하며, **lxml**과 **html.parser**가 있다.

```
from bs4 import BeautifulSoup as bs

html = """<html> <head> </head> <body> <p>test</p> </body></html>"""

soup = BeautifulSoup(html, 'lxml') #bs라는 이름으로 불러왔으니 보통 bs(html, 'lxml')로 쓴다.
print(soup)
print(type(soup))
print()
print(soup.prettify())
```

```
<html> <head> </head> <body> <p>test</p> </body></html>
<class 'bs4.BeautifulSoup'>
```

```
<html>
<head>
</head>
<body>
  <p>
    test
  </p>
</body>
</html>
```

Beautifulsoup4

bs4

```
from bs4 import BeautifulSoup as bs

html = """<html> <head><title class = 'GH' id = "이재경"> Crawling </title> </head> <body> <p>Growth</p>
<p>Hackers</p> <p>GH</p></body></html>"""

soup = BeautifulSoup(html, 'lxml')
tag_p = soup.p

print(tag_p)
print(tag_p.text)
print(tag_p.string)
print(tag_p.name)

tag_title = soup.title

print(); print(tag_title.attrs)
print(tag_title['class'])
print(tag_title['id'])

print(); print(tag_title.get('class'))
print(tag_title.get('class', '없음'))

<p>Growth</p>
Growth
Growth
p

{'class': ['GH'], 'id': '이재경'}
['GH']
이재경

None
없음
```

태그로 접근하는 방법과 속성(attributes)로 접근하는 방법이 있다.

좌측과 같은 태그로 접근하는 방식으로는 첫 번째로 등장하는 태그의 정보만을 가져온다.

속성 데이터는 a 태그의 href 속성, img 태그의 src 속성을 가져오기 위해서 필요하다.

get() 함수를 사용하면 딕셔너리에서 없는 키에 접근했을 때 발생하는 KeyError를 방지 할 수 있다.

Beautifulsoup4

bs4

```
from bs4 import BeautifulSoup as bs

html = """<html> <head><title> Crawling </title> </head> <body>
<p><span>Growth</span><span>Hackers</span><span>GH</span></p></body></html>"""

soup = BeautifulSoup(html, 'lxml')
tag_p_child = soup.p.children

print('p 자식 태그')
for child in tag_p_child:
    print(child)

tag_span = soup.span
span_parents = tag_span.parents

print('span 부모 태그')
for parent in span_parents:
    print(parent)

next_sibling = tag_span.next_sibling
prev_sibling = next_sibling.previous_sibling

print(next_sibling)
print(prev_sibling)
```

```
p 자식 태그
<span>Growth</span>
<span>Hackers</span>
<span>GH</span>
span 부모 태그
<p><span>Growth</span><span>Hackers</span><span>GH</span></p>
<body>
<p><span>Growth</span><span>Hackers</span><span>GH</span></p></body>
<html> <head><title> Crawling </title> </head> <body>
<p><span>Growth</span><span>Hackers</span><span>GH</span></p></body></html>
<html> <head><title> Crawling </title> </head> <body>
<p><span>Growth</span><span>Hackers</span><span>GH</span></p></body></html>
<span>Hackers</span>
<span>Growth</span>
```

태그들은 중첩되어 사용될 경우 밖에 있는 태그를 상위태그(상위 돔) 또는 **부모 태그**(부모 돔)이라고 부르며, 안에 있는 태그를 하위 태그(하위 돔) 또는 **자식태그**(자식 돔)라고 부른다.

contents와 **children** 속성을 이용하여 자식 태그를 가져올 수 있다. contents 속성을 사용하면 리스트 형태로 자식 태그를 가져온다. children은 contents와 달리 **iterator object** 형태로 반환되기에, 값을 가져오기 위해서는 반복문을 사용해야 한다.

반대로 **parent**와 **parents**를 사용하면 부모 태그로 올라가는 형태로 접근이 가능하다.

parents 속성에 접근하면 **generator object**의 형태로 가져온다. 이 또한 반복문을 이용하면 값을 가져올 수 있다.

형제 태그란 동등한 위치의 태그를 의미합니다. 형제 관계에서는 서로의 태그가 같을 필요는 없다.

Beautifulsoup4

bs4

: `find_all()` 함수를 사용하면 원하는 태그들을 리스트의 형태로 얻을 수 있다.

```
html = """<html> <head><title class = 'GH' id = "이재경"> Crawling </title> </head> <body> <p>Growth</p>
<p>Hackers</p> <p id= "GH">GH</p></body></html>"""
```

```
soup = BeautifulSoup(html, 'lxml')
```

```
print(soup.find_all('p'))
print(soup.find('p'))
print(soup.find_all(id='이재경'))
print(soup.body.find_all(id=False))
print(soup.find_all('p', id='GH'))
print(soup.find_all('title', class_='GH'))
print(soup.find_all('title', 'GH'))
```

```
for name in soup.find_all('title', 'GH'):
    print(name.get_text())
```

```
[<p>Growth</p>, <p>Hackers</p>, <p id="GH">GH</p>]
<p>Growth</p>
[<title class="GH" id="이재경"> Crawling </title>]
[<p>Growth</p>, <p>Hackers</p>]
[<p id="GH">GH</p>]
[<title class="GH" id="이재경"> Crawling </title>]
[<title class="GH" id="이재경"> Crawling </title>]
Crawling
```

앞서서의 방법으로는 우리가 원하는 요소에 접근하기 어렵기에 **find()**와 **find_all()** 함수를 사용해 주어야 한다.

find_all() 함수를 이용하면 우리가 원하는 태그들을 리스트의 형태로 얻을 수 있다.

태그의 종류를 이용하는 동시에 **id**와 **class**를 이용하여 원하는 요소에 접근할 수 있으며 사용 여부로 접근할 수도 있다.

class는 굳이 **class**라고 밝혀줄 필요는 없지만 가독성을 위해 명시해 주는 편이 좋다.

Beautifulsoup4

bs4

```
##select() 함수를 이용하면 find_all() 처럼 리스트로 반환하지만 CSS 셀렉터를 활용하여 원하는 요소에 접근합니다.

html = """<html> <head><title class = 'GH' id = "이재경"> Crawling </title> </head>
<body> <p id = "i" class = "a">Growth</p> <p class= "d">Hackers</p> <p class= "d">GH</p></body></html>"""

soup = BeautifulSoup(html, 'lxml')

print(soup.select('p')) #클래스는 마침표(.), 아이디는 샵(#)으로 접근
print(soup.select('.d'))
print(soup.select('p.d'))
print(soup.select('#i'))
print(soup.select('p#i'))
print(soup.select('body p#i')) #띄어쓰기를 이용하여 자식 태그 표현

[<p class="a" id="i">Growth</p>, <p class="d">Hackers</p>, <p class="d">GH</p>]
[<p class="d">Hackers</p>, <p class="d">GH</p>]
[<p class="d">Hackers</p>, <p class="d">GH</p>]
[<p class="a" id="i">Growth</p>]
[<p class="a" id="i">Growth</p>]
[<p class="a" id="i">Growth</p>]
```

select() 함수를 이용하면 **find_all()**처럼 리스트로 반환합니다. 하지만 **select()**는 CSS셀렉터를 활용하여 원하는 요소에 접근한다.

class는 **마침표(.)**, id는 **샵(#)**으로 접근하는 것만 알면 쉽게 CSS 셀렉터에 접근할 수 있다.

자식 태그를 표현 할 때는 띄어쓰기나 꺾쇠(>)를 사용한다.

Beautifulsoup4

정규 표현식

```
#find_all( )과 find( )를 사용할 때 정규식을 사용하면 좀 더 활용도가 높아집니다

from bs4 import BeautifulSoup as bs
import re

test_str = "test t1sd j test1"

pattern = re.compile('test')

#a = pattern.match(test_str)
#b = pattern.search(test_str)
#c = pattern.findall(test_str)
#d = pattern.finditer(test_str)

print('--match result--') #처음 시작부터 정규식과 일치해야 match가 이루어지며 정규식과 다르면 예러가 난다
print(a)
print(a.group(),a.start(), a.end(),a.span())
print()

print('--search result--') # 문자열에서 일치하는 정규식을 하나 찾아준다
print(b)
print(b.group(),b.start(), b.end(),b.span())
print()

print('--findall result--') #정규식과 매치되는 모든 문자열을 리스트로 반환한다
print(c)
print()

print('--finditer result--') #정규식과 매치되는 모든 문자열을 iterable 객체로 반환하기에 반복문을 이용해야 한다
print(d)
for i in d:
    print(i.group(),i.start(), i.end(),i.span())

--match result--
<_sre.SRE_Match object; span=(0, 4), match='test'>
test <built-in method start of _sre.SRE_Match object at 0x00000284EC02E718> 4 (0, 4)

--search result--
<_sre.SRE_Match object; span=(0, 4), match='test'>
<built-in method start of _sre.SRE_Match object at 0x00000284EC02E3D8> 4 (0, 4)

--findall result--
['test', 'test']

--finditer result--
<callable_iterator object at 0x00000284EC04BE48>
test <built-in method start of _sre.SRE_Match object at 0x00000284EC02E4A8> 4 (0, 4)
test <built-in method start of _sre.SRE_Match object at 0x00000284EC02E238> 16 (12, 16)
```

정규 표현식(regular expression) 또는 **정규식**은 특정한 규칙을 가진 문자열의 집합을 표현하는데 사용하는 형식 언어이다. 정규식을 사용하기 위해서는 **re**(Regular Expression) 모듈을 사용하면 된다.

`find_all()`과 `find()`를 할 때 정규식을 사용하면 정확한 단어가 아니라 특정 단어를 포함하거나, 패턴이 일치하는 요소를 찾을 수 있다.

match() : 문자열의 처음부터 정규식과 매치되는지 조사한다.

search() : 문자열 전체를 검색하여 정규식과 매치되는지 조사한다.

findall() : 정규식과 매치되는 모든 문자열(substring)을 리스트로 리턴한다.
= `find_all()`, `findAll()`

finditer() : 정규식과 매치되는 모든 문자열(substring)을 iterator 객체로 리턴한다.

`match()`, `search()`를 통해 나온 결과물을 `group()`, `start()`, `end()`, `span()`을 이용하여 리턴합니다.
`findall()`과 `finditer()`은 리스트와 객체로 반환하므로 반복문을 이용해야 한다.

Beautifulsoup4

정규 표현식

```
import re

test_str= """Good afternoon 222"""

pattern = re.compile('[a-z]')
pattern1 = re.compile('[a-z]+')
pattern2 = re.compile('[a-zA-Z]*')
pattern3 = re.compile('[a-zA-Z0-9]')
pattern4 = re.compile('\w+')
pattern5 = re.compile('[^a-z]+')

a = pattern.findall(test_str)
b = pattern1.findall(test_str)
c = pattern2.findall(test_str)
d = pattern3.findall(test_str)
e = pattern4.findall(test_str)
f = pattern5.findall(test_str)

print(a)
print(b)
print(c)
print(d)
print(e)
print(f)

['o', 'o', 'd', 'a', 'f', 't', 'e', 'x', 'n', 'o', 'o', 'n']
['ood', 'afternoon']
['Good', '', 'afternoon', '', '', '', '', '']
['G', 'o', 'o', 'd', 'a', 'f', 't', 'e', 'x', 'n', 'o', 'o', 'n', '2', '2', '2']
['Good', 'afternoon', '222']
['G', ' ', ' ', '222']
```

re.compile()에 들어가는 식을 정규 표현식이라고 한다.
[0-9] 는 0부터 9까지를 의미한다. +는 하나 이상 포함된 것을 계속 찾아주며, *는 앞에 나온 패턴이 하나라도 없어도 된다.

[a-z]는 영어 소문자, [A-Z]는 영어 대문자, [ㄱ-ㅣ가-힣] 는 한글을 의미한다. 대괄호 안의 ^는 대괄호 안의 것을 포함하지 않는 것을 찾아준다.

Beautifulsoup4

정규 표현식

```
import re

test_str = """test tests toast toasts"""

pattern = re.compile('t..t')
pattern1 = re.compile('t...t')
pattern2 = re.compile('t?est\\w+')
pattern3 = re.compile('t?est\\w*')

a = pattern.findall(test_str)
b = pattern1.findall(test_str)
c = pattern2.findall(test_str)
d = pattern3.findall(test_str)

print(a)
print(b)
print(c)
print(d)

['test', 'test']
['toast', 'toast']
['tests']
['test', 'tests']
```

```
import re

test_num = "저의 전화번호는 010-6354-0802 입니다"

pattern = re.compile('[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]')
pattern1 = re.compile('\\d\\d\\d-\\d\\d\\d-\\d\\d\\d\\d')
pattern2 = re.compile('\\d{3}-\\d{4}-\\d{4}')

a = pattern.findall(test_num)
b = pattern1.findall(test_num)
c = pattern2.findall(test_num)

print(a)
print(b)
print(c)

['010-6354-0802']
['010-6354-0802']
['010-6354-0802']
```

마침표를 이용하면 해당 자리를 표현할 수 있다. t. 이라고 하면 t로 시작하는 주 글자를 찾는다.

t?est는 test나 est를 찾는 패턴식이다.

ww는 문자, wd는 숫자를 의미한다. 옆의 { }은 반복 횟수로 ww{3}는 문자를 3자리 씩 묶어서 찾는다.

Selenium

Selenium

Selenium은 웹 브라우저를 컨트롤하는 라이브러리이다.

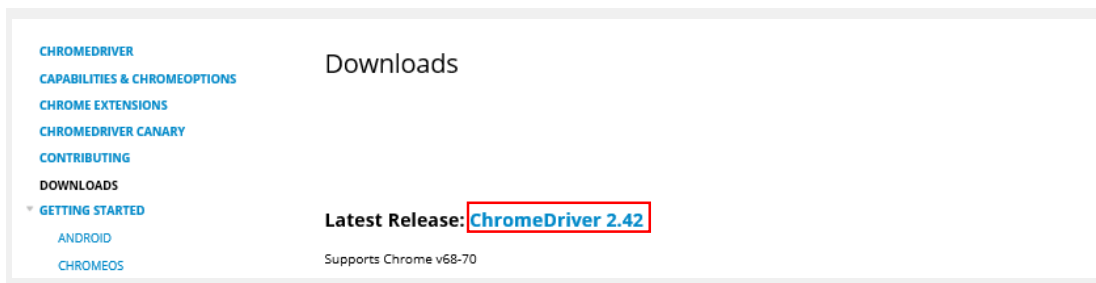
- urllib으로는 제어할 수 없는 JavaScript 요소가 있는 페이지를 다루어야 할 때
- urllib을 통한 접속이 막혀 있는 사이트를 스크래핑 할 때

Selenium 라이브러리 설치

```
pip install selenium
```

Chrome 웹드라이버 설치(물론 사전에 크롬 브라우저가 설치 되어 있어야 한다.)

<https://sites.google.com/a/chromium.org/chromedriver/downloads>



Selenium

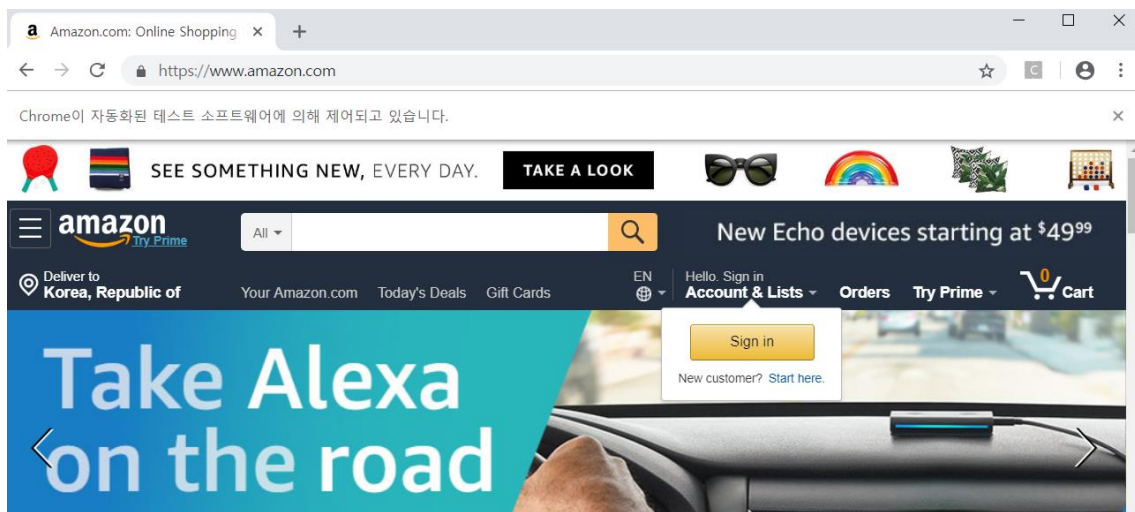
Selenium

```
# import selenium
from selenium import webdriver
import time
import requests
from selenium.webdriver.common.keys import Keys
```

```
path = 'C:/Users/dlwor/chromedriver.exe' # 크롬드라이브 경로 지정 ## 맥과 리눅스는 exe.를 붙이지 않습니다.
```

```
driver = webdriver.Chrome(path)
```

```
driver.get("https://amazon.com")
```



다운 받은 Chrome 웹드라이버 경로를 지정해 준 뒤 실행하게 되면 urllib으로는 접근할 수 없었던 아마존 사이트에도 접근할 수 있다.

Selenium은 주로 웹앱을 테스트하는데 이용하는 프레임워크이다.

눈에 보이는 콘텐츠라면 모두 가져올 수 있다.

직관적이나 속도가 느리다.

bs4와 적절히 섞어서 쓰면 좋다.

Selenium

Selenium

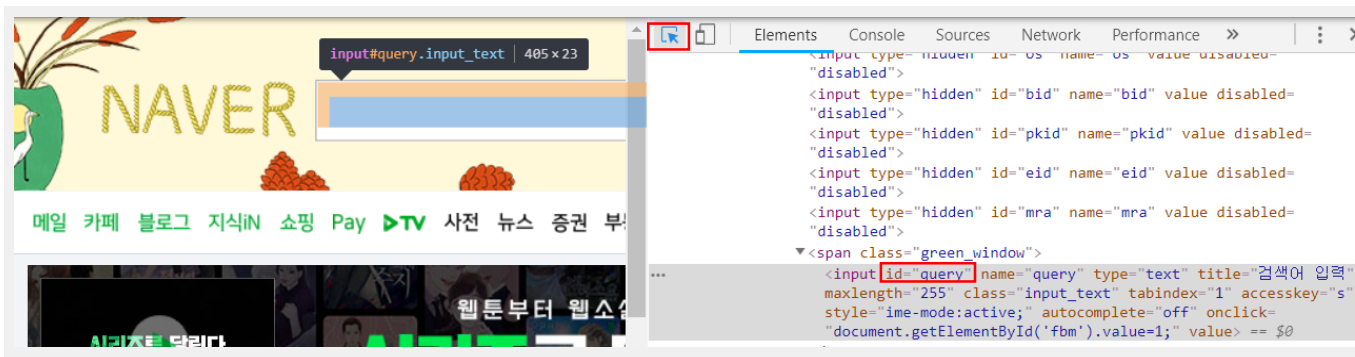
find_element_by_id(id)	id 속성으로 요소 하나 추출	
find_element_by_name(name)	Name 속성으로 요소 하나 추출	
find_element_by_class_name(name)	클래스 이름이 name에 해당되는 요소 하나 추출	find_elements_by_class_name
find_element_by_partial_link(text)	링크의 자식요소에 포함되어 있는 텍스트로 요소 하나 추출	find_elements_by_partial_link
find_element_by_tag_name(name)	태그 이름이 name에 해당하는 요소 하나 추출	find_elements_by_tag_name
find_element_by_link_text(text)	링크 텍스트로 요소 하나 추출	
find_element_by_xpath(query)	xpath를 지정해 요소 하나 추출	find_elements_by_xpath
find_element_by_css_selector(query)	css 선택자 요소 하나 추출	find_elements_by_css_selector

Selenium

Selenium

```
#3초간 쉬는 명령어
time.sleep(3) # = driver.implicitly_wait(3)
# naver에 접속한다
driver.get("https://naver.com")
# 현재 페이지 확인
print(driver.current_url)
# 크롬에서 주소창위에 나오는 이름
print(driver.title)
```


<https://www.naver.com/>
NAVER



```
search = driver.find_element_by_id('query')
```

크롬 드라이버를 통하여 naver에 접속에 접속한다.

F12를 눌러거나 **Ctrl + Shift + I** 를 눌러 Elements를 확인한다.

좌측 상단의  을 누르거나 **Ctrl + Shift + C** 누른 뒤 접근할 위치에 커서 놓는다.

검색어 창의 아이디가 'query' 인 것을 확인하고 **find_element_by_id()**를 활용하여 접근한다.

Selenium

Selenium

```
from selenium.webdriver.common.keys import Keys
```

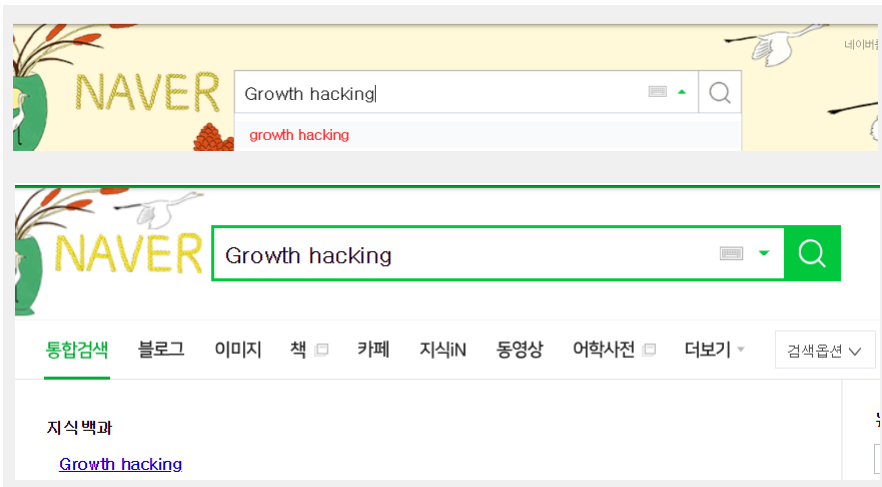
```
search.send_keys("Growth hacking")
```

```
search.submit()
```

사람이 키보드 누르는 것처럼 해주는 모듈을 활용한다.

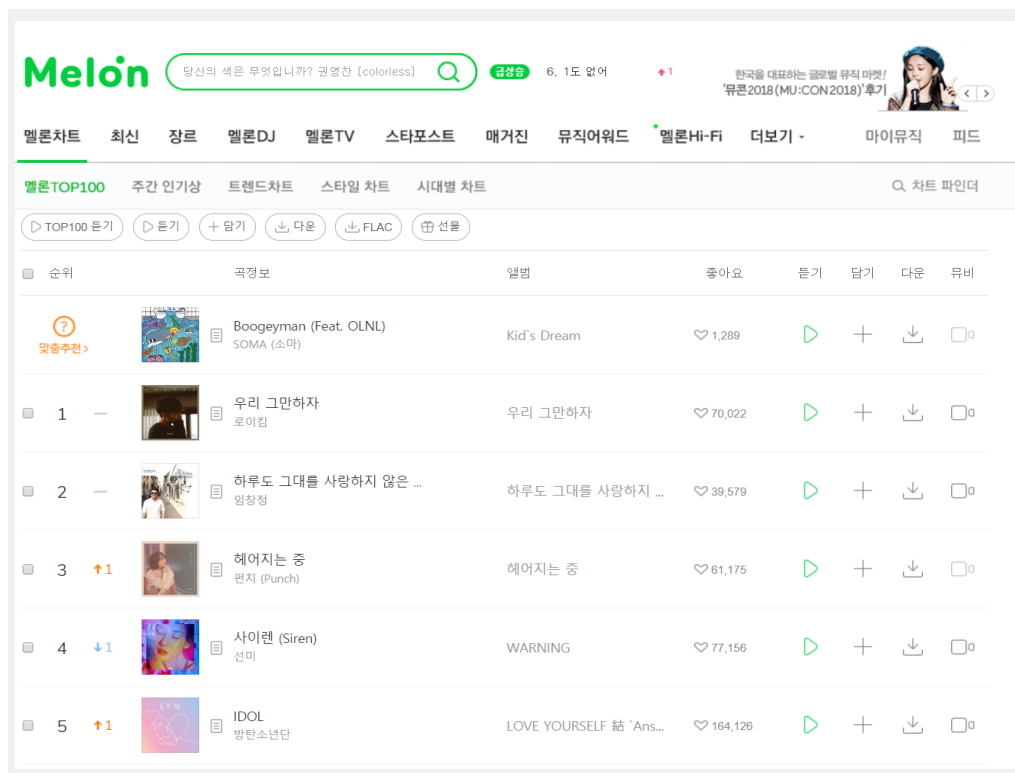
send_keys()를 통해 "Growth hacking" 을 검색창에 입력한다.

submit()를 통해 입력한 문자열을 검색한다.



Crawling

Melon Top 100 차트 가져오기



```
url = "http://www.melon.com/chart/index.htm"
header_ = "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36"
```

```
request = req.Request(url, headers = {'User-Agent':header_})
html = req.urlopen(request)
page = bs(html.read(), 'lxml')
```

Melon Top 100 차트에서 노래에 대한 정보를 긁어 와 보도록 하자

url에 접근하고자 하는 주소를 지정해 준다.

일부 웹서버에서 **user-agent** 헤더를 보내지 않으면 웹로봇으로 인식하고 차단한다. user-agent 헤더는 브라우저 정보로 크롬, 사파리, 익스프로어, 파이어폭스 등 클라이언트 프로그램 정보를 의미한다.

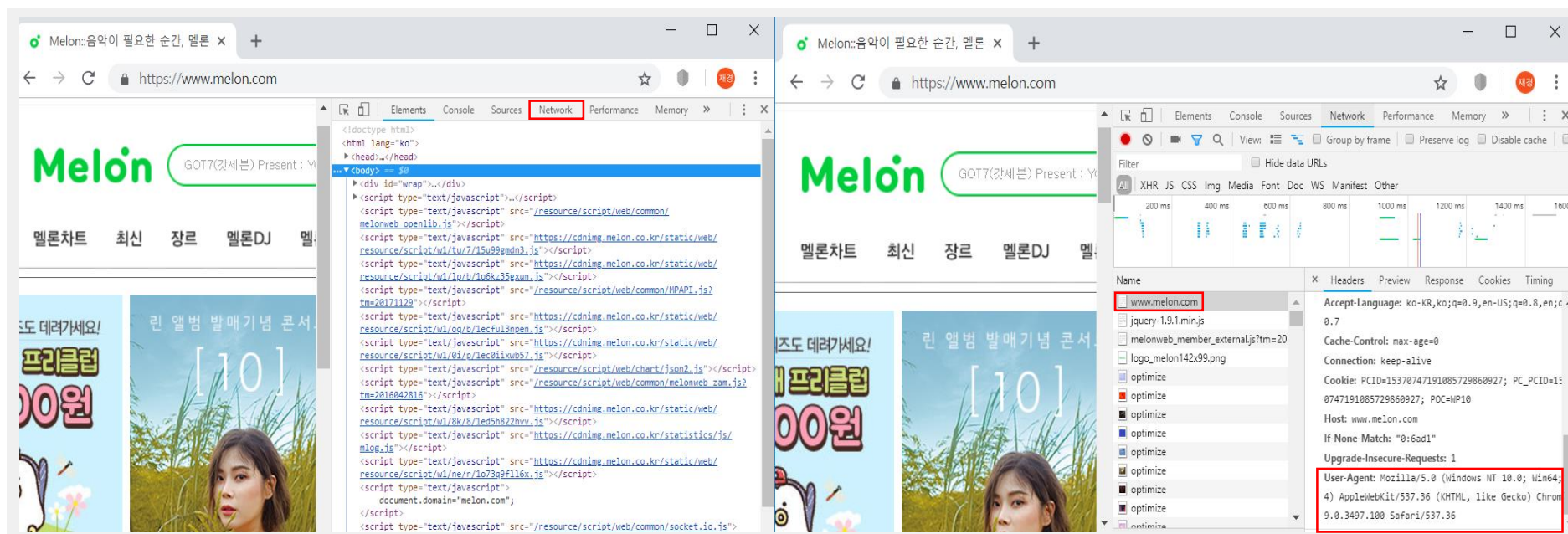
Melon는 그러한 웹서버 중 하나이기에 헤더를 포함시킨 뒤, 요청해야 한다.

가져온 url을 html이란 이름에 지정한 후, 이를 beautifulsoup4를 통해서 정제해 준다.

Crawling

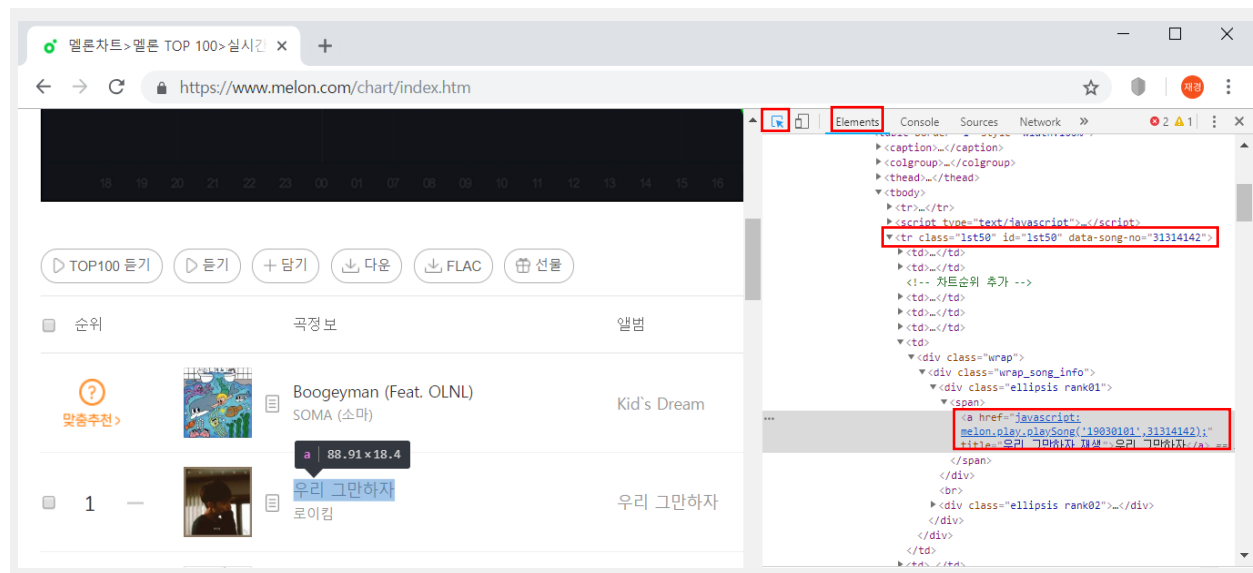
Melon Top 100 차트 가져오기

자신의 user-agent 헤더는 F12 또는 Ctrl + Shift + I 를 누른 뒤, network 탭으로 들어가서 확인 할 수 있다.



Crawling

Melon Top 100 차트 가져오기




```
page.find("div", {'class':'ellipsis rank01'})
```

```
<div class="ellipsis rank01"><span>  
<a href="javascript:melon.play.playSong('19030101',31314142);" title="우리 그만하자 재생">우리 그만하자</a>  
</span></div>
```

```
song_title = page.find("div", {'class':'ellipsis rank01'}).get_text().strip()  
song_artist = page.find("span", {'class':'checkEllipsis'}).get_text().strip()  
  
print(song_title)  
print(song_artist)
```

우리 그만하자
도미킴

Melon Top 100 차트에서 노래에 대한 정보를 긁어 와 보도록 하자

가정 먼저 긁어 오고자 하는 부분을 탐색하기 위해 개발자 탭 상단에 있는  를 누르거나 Ctrl + Shift + C 누른 뒤 접근할 위치에 커서 놓는다.

해당 부분이 어디인지 확인하고 그 상위 태그들을 유심히 관찰하여 전반적인 구조가 어떻게 되어 있는지 인지한다.

Crawling

Melon Top 100 차트 가져오기

```
songlists50 = page.findAll('tr', {'class': 'lst50'})
songlists100 = page.findAll('tr', {'class': 'lst100'})
```

```
for half in [songlists50, songlists100]:
    for songs in half:
        song_artist = songs.find("span", {'class': 'checkEllipsis'}).get_text().strip()

        ac = songs.find('span', {'class': 'checkEllipsis'})
        p = re.compile('[0-9]+')
        song_artist_code = int(p.findall(str(ac))[0])

        song_title = songs.find("div", {'class': 'ellipsis rank01'}).get_text().strip()
        song_title_code = int(songs['data-song-no'])
        song_rank = int(songs.find("span", {'class': "rank "}).get_text())
        song_date = dt_now.strftime('%Y-%m-%d')
        song_hour = dt_now.strftime('%H')
        song_site = "melon"
```

```
melon_list = melon_list.append(
    {'artist': song_artist,
     'artist_code': song_artist_code,
     'title': song_title,
     'title_code': song_title_code,
     'rank': song_rank,
     'date': song_date,
     'hour': song_hour,
     'site': song_site}, ignore_index = True)
```

```
melon_list.to_csv(dt_now.strftime("%Y-%m-%d-%H")+".csv", index=False, encoding = 'euc-kr')
```

클어 오길 바라는 사이트의 구조를 확인하고 자신이 확보하고자 하는 내용들이 어디에 존재하는지 확인한다.

각 항목들에 대한 파악이 되었으면 가장 편리할 것 같은 방식을 선택하여 각 항목의 내용을 가져오도록 하면 된다.

원하는 정보에 접근하는 방식은 다양할 수 있다. 본인이 편한 방식으로 접근하면 된다.

수집해온 정보는 csv파일로 저장하도록 하였다.

Crawling

Melon Top 100 차트 가져오기

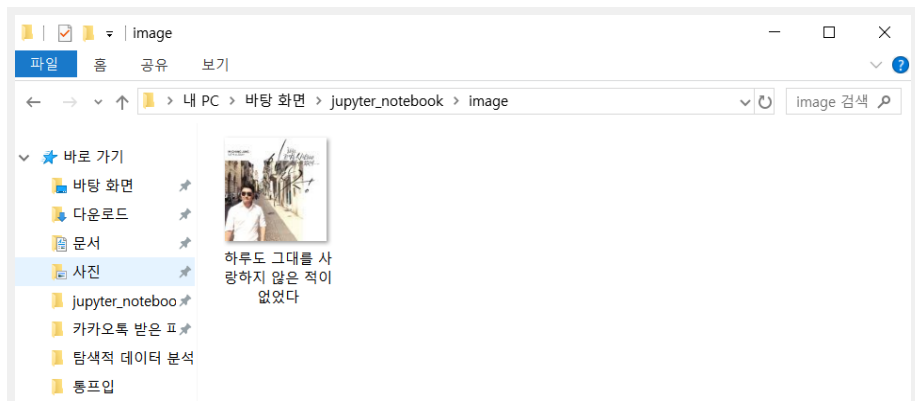
앨범사진 src 가져오기

```
a = []  
for name in page.find("a", {'class': 'image_typeAll'}).children:  
    a.append(name)  
a[1]['src']
```

```
'https://cdnimg.melon.co.kr/cm/album/images/102/05/805/10205805_500.jpg/melon/resize/120/quality/80/optimize'
```

```
urllib.request.urlretrieve(a[1]['src'], "C:/Users/dlwor/Desktop/jupyter_notebook/image/{0}.jpg".format(song_title))
```

```
('C:/Users/dlwor/Desktop/jupyter_notebook/image/하루도 그대를 사랑하지 않은 적이 없었다.jpg',  
<http.client.HTTPMessage at 0x1f09b54fac8>)
```



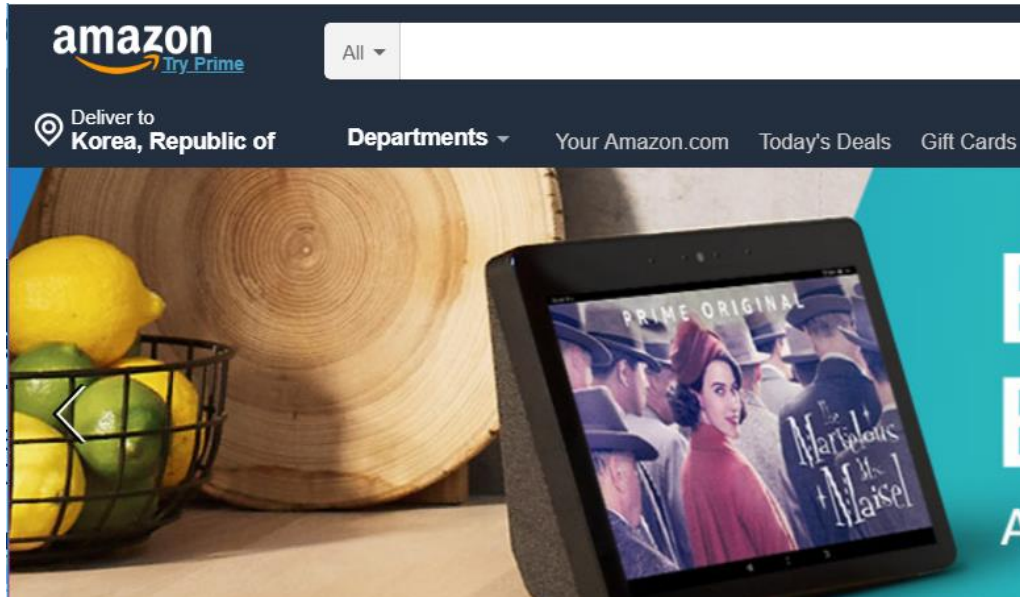
앨범 사진들도 가져와 저장할 수 있다.

앞선 크롤링과 같은 방식으로 가져오고자 하는 부분에 대한 탐색을 한 뒤, 그 요소에 접근하면 된다.

img 태그 안의 src 속성을 가져온 뒤, `urllib.request.urlretrieve()` 함수를 통해서 이미지를 로컬에 저장할 수 있다.

Crawling

Amazon Today's Deal through Selenium



Today's Deals

New deals. Every day. Shop our Deal of the Day, Lightning Deals and more daily deals and limited-time sales. See deals you're watching [here](#), or let the deals come to you by receiving our [daily deals email](#).

Showing 1-24 of 1010 results for 3 Availability Options x

Department

- ☐ Amazon Devices
- ☐ Amazon Video
- ☐ Arts, Crafts & Sewing
- ☐ Automotive & Motorcycle
- ☐ Baby
- ☐ Baby Clothing & Accessories
- ☐ Beauty
- ☐ Books
- ☐ Boys' Fashion
- ☐ Camera & Photo
- ☐ Cell Phones & Accessories
- ☐ Computers & Accessories

[See more](#)

Deal Type

- [Deal of the Day](#)
- [Lightning Deals](#)
- [Savings & Sales](#)



DEAL OF THE DAY

\$26.59

Price: ~~\$136.63~~ (81% off)

Ends in 15:36:00

Arteza Real Brush Pens, 48
Watercolor Markers

Sold by ARTEZA and Fulfilled by Amazon.



DEAL OF THE DAY

\$109.99 - \$229.99

Ends in 15:36:00

Save up to 40% on Anker Nebula
Projector

★★★★☆ 6

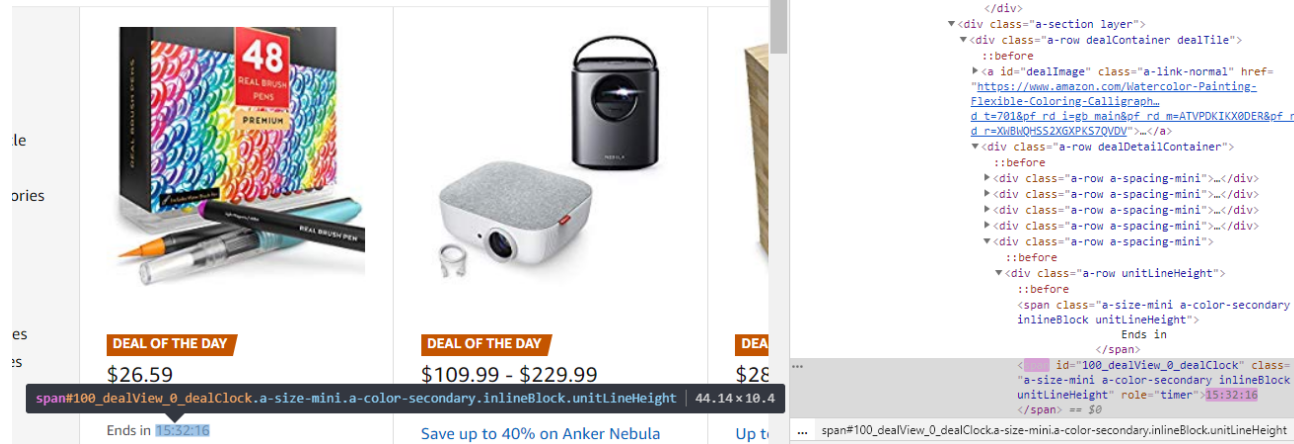
아마존의 Today's Deals 라는 섹션에 들어가게 되면 각 상품에 대한 할인 적용시간이 실시간으로 바뀌어 나타난다.

Crawling

Amazon Today's Deal through Selenium

Deals come to you by receiving our [daily deals email](#).

Deals for 3 Availability Options ✕



DEAL OF THE DAY

\$26.59

span#100_dealView_0_dealClock.a-size-mini.a-color-secondary.inlineBlock.unitLineHeight 44.14 x 10.4

Ends in 15:32:16

DEAL OF THE DAY

\$109.99 - \$229.99

Save up to 40% on Anker Nebula

DEAL OF THE DAY

\$28

Up to 50% off

```
url = "https://www.amazon.com"
header_ = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36"

request = req.Request(url, headers = {'User-Agent':header_})
html = req.urlopen(request)
page = bs(html.read(), 'lxml')

print(html.getcode())

200

left_time = page.find("span", {'id':'100_dealView_1_dealClock'})
print(left_time)

None
```

남아 있는 시간은 JavaScript로 구현되어 있는 동적 요소이기
에 기존의 urllib으로는 정보를 얻어 올 수가 없다.

Crawling

Amazon Today's Deal through Selenium

Deals come to you by receiving our [daily deals email](#).

Deals for 3 Availability Options ✕

DEAL OF THE DAY

\$26.59

span#100_dealView_0_dealClock.a-size-mini.a-color-secondary.inlineBlock.unitLineHeight | 44.14 x 10.4

Ends in 15:28:37

DEAL OF THE DAY

\$109.99 - \$229.99

Save up to 40% on Anker Nebula

DEAL OF THE DAY

\$28

Up to 50% off

```
path = 'C:/Users/dlwor/chromedriver.exe'
driver = webdriver.Chrome(path)

driver.get("https://www.amazon.com/gp/goldbox/")

html = driver.page_source
soup = BeautifulSoup(html, 'lxml')
left_time_sel = soup.find("span", {'id': '100_dealView_1_dealClock'})
print(left_time_sel.text)
```

15:28:39

따라서 이는 Selenium을 통해 해당 사이트의 javascript를 웹 드라이버를 통해 실행한 뒤 정보를 가져와야 한다.

Selenium을 통해 해당 페이지에 접속한 뒤 **page_source**를 통해 해당 페이지의 html을 가지고 온 뒤 beautifulsoup을 활용하여 원하는 요소를 가져올 수 있다.

Crawling

크롤링의 활용

크롤링은 차후에 이루어질 수 있는 많은 분석들의 재료를 구하는데 있어 유용하다.

여러 웹에서 작성된 글이나 댓글 등의 텍스트 정보를 긁어와 자연어 처리를 거친 후 감성 분석 또는 트렌드 분석에 활용될 수 있다.

이미지를 이용한 데이터들을 분석하는 비정형적 분석에서도 크롤링을 통해 필요한 이미지들을 확보할 수 있다.

공공데이터 포털처럼 오픈 API를 통해 제공하는 데이터를 용이하게 받아 올 수 있다.

분석을 위한 데이터 수집의 목적 외에도 많은 서비스들이 크롤링을 활용하여 서비스를 제공하며, 검색 엔진이나 가격 비교 서비스와 같은 서비스들이 대표적이다.

부록

Selenium으로 요소 조종하기

<code>clear()</code>	글자 입력란에 글자를 지움	<code>id</code>	요소의 id속성
<code>click()</code>	요소를 클릭	<code>location</code>	요소의 위치
<code>get_attribute(name)</code>	Name에 해당되는 값을 추출	<code>parent</code>	부모 요소
<code>is_displayed()</code>	요소가 화면에 출력되는지 확인	<code>rect</code>	크기와 위치정보를 가진 딕셔너리 자료형을 리턴
<code>is_selected()</code>	체크박스 등의 요소가 선택된 상태인지 확인	<code>screenshot_as_base64</code>	BASE64로 스크린샷을 추출
<code>is_enabled()</code>	요소가 활성화되어 있는지 확인	<code>screenshot_as_png</code>	PNG형식으로 스크린샷 추출
<code>screenshot(filename)</code>	스크린샷을 찍는다.	<code>size</code>	요소의 크기
<code>send_keys(value)</code>	키를 입력한다.	<code>tag_name</code>	태그 이름
<code>submit()</code>	입력 양식을 전송한다.	<code>text</code>	요소의 내부글자
<code>value_of_css_property(name)</code>	Name에 해당하는 CSS속성의 값을 추출		

부록

Selenium 드라이버 조작

add_cookie(cookie_dict)	쿠키 값을 딕셔너리 형식으로 지정	get_screenshot_as_base64()	Base64형식으로 스크린샷을 추출
back() / forward()	이전 페이지 또는 다음 페이지로 이동	get_screenshot_as_png()	PNG형식으로 스크린샷을 추출
close()	브라우저를 닫는다	get_window_position(windowHandle = "current")	브라우저의 위치를 추출
current_url	현재 url을 추출	get_window_size(windowHandle = "current")	브라우저의 크기를 추출
delete_all_cookies()	모든 쿠키를 제거	implicitly_wait(sec)	대기시간을 토 단위로 지정해 대기
delete_cookie(name)	특정 쿠키를 제거	quit()	selenium자체를 종료
execute(command, params)	브라우저의 고유 명령어를 실행	save_screenshot(filename)	스크린 샷을 저장
excute_async_script(script,*args)	비동기 처리하는 자바스크립트를 실행	set_page_load_timeout(time_to_wait)	페이지를 읽는 타임아웃 시간을 지정
execute_script(script,*args)	동기 처리하는 자바스크립트를 실행	set_script_timeout(time_to_wait)	스크립트의 타임아웃 시간을 지정
get(url)	url로 브라우저 이동	set_window_position(x,y>windowHandle='current')	브라우저의 위치를 지정
get_cookie(name)	특정 쿠키 값을 추출	set_window_size(가로,세로>windowHandle='current')	브라우저의 크기를 지정
get_cookies	모든 쿠키 값을 딕셔너리 형식으로 추출	title	현재페이지의 타이틀을 추출
get_log(type)	로그를 추출(browser/drvirt/client/server)		

Quest

네이버 기사 크롤링



네이버 정치 일반 기사 페이지에 들어가서 등장하는 기사들의 url을 저장하고 그 중 첫번째 기사의 제목과 기사를 불러오는 코드이다.

Quest

네이버 기사 크롤링

```
import pandas as pd
from pandas import DataFrame
import numpy as np

data = [titles, texts]

newData = np.transpose(data)

df = DataFrame(newData, columns = ["기사 제목", "내용"])

df
```

	기사 제목	내용
0	"두달 만에 둘러짚"...정우연이 밝힌 '김윤옥 명목가발' 사건	김윤옥 여사 차에 두고 있고 있었다 주장영커집권이 유력했던 대선 후보의 부...
1	여년 수준이라더니 결국 한미합권 축소...北 의식한 '몸 낮추기'	영커공연 준비는 이렇게 끝났고 이번에는 한미연합훈련은 어떻게 하는 건지 살...
2	[뉴스브리핑] '4인 선거구' 무산 반발...서울시의회 총출	인 선거구 무산 반발서울시의회 총출의장석을 점거한 바른미래당 시의원들을 자...
3	[비하인드 뉴스] 아파트 매각 작전? 국회 교원위 또...	영커저희 뉴스에서는 오래전부터 근로라는 말 대신 노동 노동자라는 표현을 쓰...
4	속전속결 합의...윤상 "현승일, 책임감 느끼는 것 같았다"	영커오늘 남북은 판문점에서 공언협상을 보였습니다 우리측에서 윤상 단장이...
5	'민주화 정신' 추가...기본권 강화	영상 플레이어 영커 청와대가 오는 일 발의할 대통령 개헌안 ...
6	한미연합훈련 '여년대로 두 달'...北, '축소' 발표	영커한미연합훈련 일정이 오늘일 공식 발표했습니다 그런데 우리와 미국의 발표내...
7	MB, 대선 직후 능인선원 주지 지광스님에게 손수 전화 "고맙다"	서울신문경찰 현금 의원 수수 구속영장에 적시불고대학원 불린 청탁 뇌물지광도 ...
8	강창일 의원 "피해자 동의 없으면 강간" 형법 개정안 발의	성범죄성립요건국제기준준용형법개정안에서년이상강화 ...

네이버 정치 기사 페이지서 다섯페이지를 넘어가며 등장하는 기사들의 제목과 내용을 가져오기!!

pandas를 통해서 정리할 필요는 없으며 제목과 내용만 가져올 수 있으면 된다.

*페이지를 넘기는 것은 페이지 하단에 존재하는 페이지 숫자를 접근하면 된다.

"<https://news.naver.com/main/list.nhn?mode=LS2D&mid=shm&sid2=269&sid1=100&date=20180928&page=> 의 패턴을 활용하여 각 페이지에 접근해도 좋다.

하지만 urllib으로는 접근이 어려워 selenium을 이용한다면 `find_element_by_link_text("페이지 숫자").click()` 을 사용하면 비교적 쉽게 접근할 수 있다.

네이버 정치 일반 기사 url : "<http://news.naver.com/main/list.nhn?mode=LS2D&mid=shm&sid1=100&sid2=269>

*ppt에 등장하는 모든 코드는 같이 첨부한 ipynb 파일에 있습니다.