day1. crawling, scraping

데이터수집하기

1-1 데이터 다운로드하기

- 웹상의 정보를 추출하는 방법
  - urllib 라이브러리
    - http, ftp를 사용해서 데이터를 다운로드
    - urllib : url을 다루는 모듈을 모아놓은 패키지
    - urllib.request 웹사이트에 있는 데이터에 접근하는 기능
- urllib.request 를 이용한 다운로드
  - urllib.request 모듈에 있는 urlretrieve() 를 사용, 직접 파일을 다운로드

웹으로부터 정보를 추출하는 방법

- 파이썬, urlib
  - http, ftp를 사용해 데이터를 다운로드할 때 사용할 수 있는 라이브러리
  - urllib: URL을 다루는 모듈을 모아 놓은 패키지

웹으로부터 정보를 추출하는 방법

- urllib.request 을 이용한 웹사이트에 존재하는 파일을 다운로드

웹으로부터 정보를 추출하는 방법(1)

```
## 1-1_download-png1.py
# 라이브러리 읽어 들이기 --- (※1)
import urllib.request
# URL과 저장 경로 지정하기
url = "http://uta.pw/shodou/img/28/214.png"
savename = "test.png"
# 다운로드 --- (※2)
urllib.request.urlretrieve(url, savename)
print("저장되었습니다...!")
```

웹으로부터 정보를 추출하는 방법(2)

```
## 1-1_download-png2.py
import urllib.request
# URL과 저장 경로 지정하기
url = "http://uta.pw/shodou/img/28/214.png"
savename = "./ch01/test2.png"
# 다운로드 --- (※1)
mem = urllib.request.urlopen(url).read()
# 파일로 저장하기 --- (※2)
with open(savename, mode="wb") as f:
    f.write(mem)
    print("저장되었습니다...!")
```

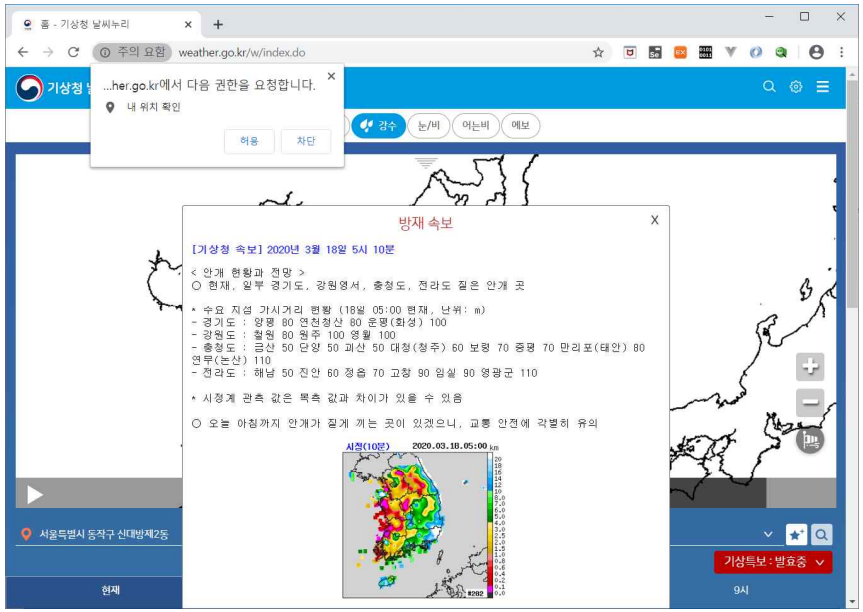
웹으로부터 정보를 추출하는 방법(3)

```
# IP 확인 API로 접근해서 결과 출력하기
# 모듈 읽어 들이기 --- (※1)
import urllib.request
# 데이터 읽어 들이기 --- (※2)
url = "http://api.aoikujira.com/ip/ini"
res = urllib.request.urlopen(url)
data = res.read()
# 바이너리를 문자열로 변환하기 --- (※3)
text = data.decode("utf-8")
print(text)
```

```
[ip]
API_URI=http://api.aoikujira.com/ip/get.php
REMOTE_ADDR=210.2.52.26
REMOTE_HOST=210.2.52.26
REMOTE_PORT=35988
HTTP_HOST=api.aoikujira.com
HTTP_USER_AGENT=Python-urllib/3.7
HTTP_ACCEPT_LANGUAGE=
HTTP_ACCEPT_CHARSET=
SERVER_PORT=80
FORMAT=ini
```

웹으로부터 정보를 추출하는 방법(4)

```
import urllib.request
import urllib.parse
API = "http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"
# 매개변수를 URL 인코딩합니다. --- (※1)
values = {
    'stnId': '108'
}
params = urllib.parse.urlencode(values)
# 요청 전용 URL을 생성합니다. --- (※2)
url = API + "?" + params
print("url=", url)
# 다운로드합니다. --- (※3)
data = urllib.request.urlopen(url).read()
text = data.decode("utf-8")
print(text)
```

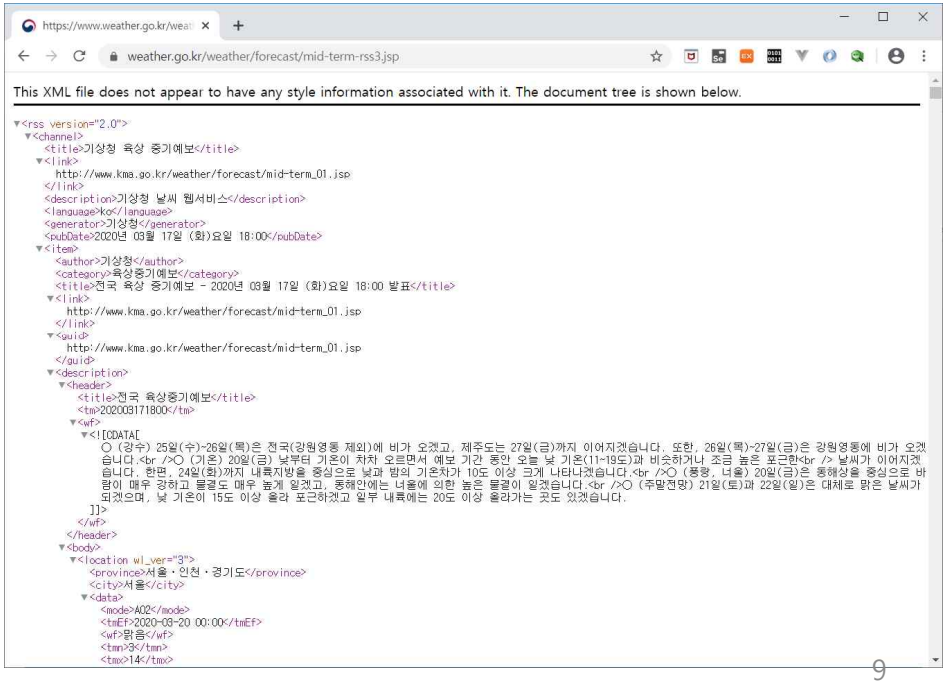


웹으로부터 정보를 추출하는 방법(4)

기상청의 RSS서비스: API(URL)에 지역번호를 지정하면 해당 지역의 기상정보를 제공

http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp

※ [참고] 기상청 RSS: http://kma.go.kr/weather/lifenindustry/sevice_rss.jsp



> python 1-1_download-forecast.py > forecast.html

웹으로부터 정보를 추출하는 방법(4)

매개변수	의미
stnId	기상 정보를 알고 싶은 지역을 지정합니다.

http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=108
http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=109
http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=105
http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=131
http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=133
http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=146
http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=156
http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=143
http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=159
http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=184

지역	지역 번호	지역	지역 번호
전국	108	전라북도	146
서울/경기도	109	전라남도	156
강원도	105	경상북도	143
충청북도	131	경상남도	159
충청남도	133	제주특별자치도	184

참고) GET 요청으로 매개변수 전송하기

- [html 파라미터 전송 서식] GET 요청으로 매개변수 전송하기

http://example.com?key1=v1&key2=v2&key3=v3...

- URL 끝부분에 "?"를 입력
- "<key>=<value>" 형식으로 매개변수를 작성
- 여러 개의 매개변수를 사용할 때는 "&"을 사용해 구분

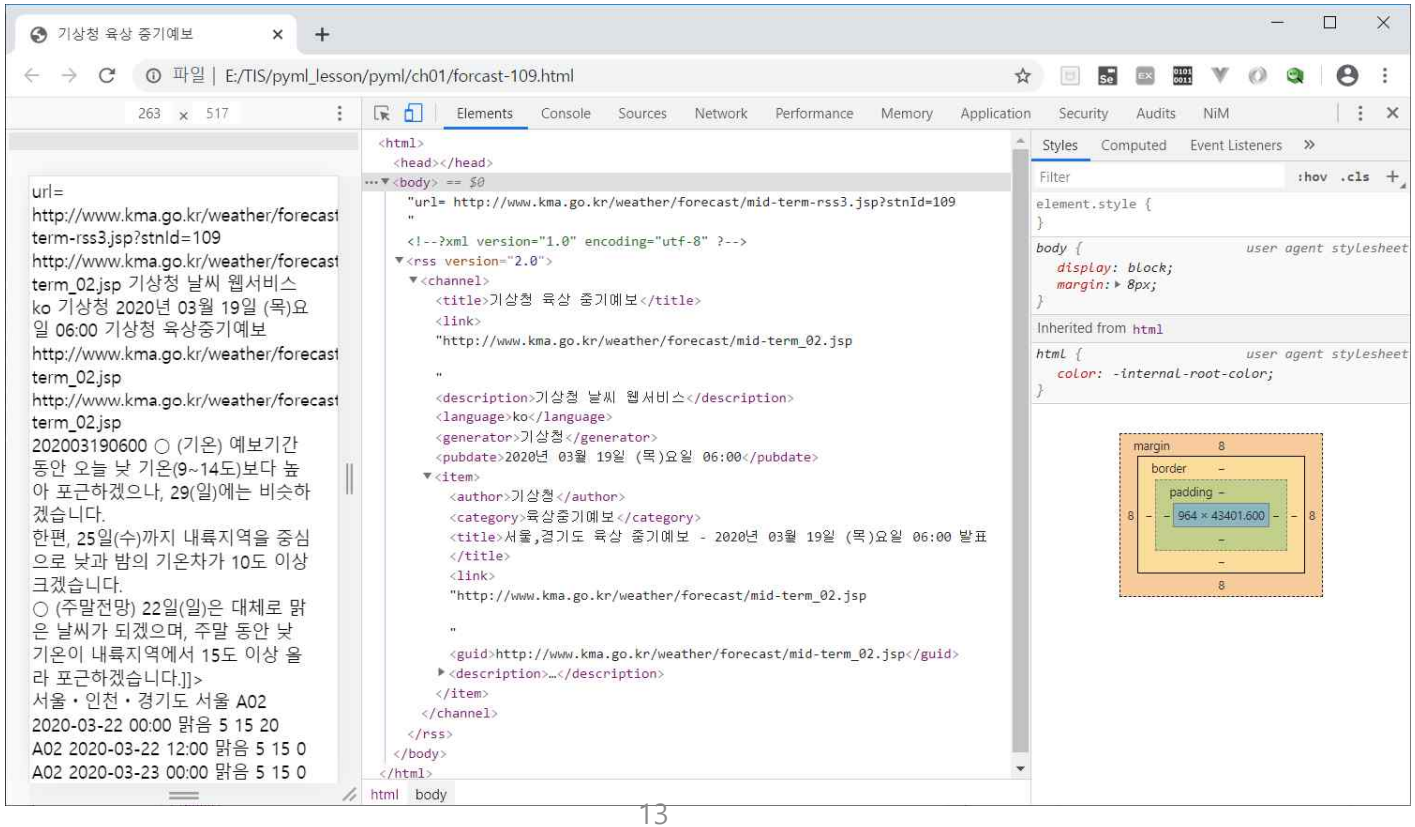
웹으로부터 정보를 추출하는 방법(5)

```
#!/usr/bin/env python3
# 라이브러리를 읽어 들입니다. --- (※1)
import sys
import urllib.request as req
import urllib.parse as parse
# 명령줄 매개변수 추출 --- (※2)
if len(sys.argv) <= 1:
    print("USAGE: download-forecast-argv <Region Number>")
    sys.exit()
regionNumber = sys.argv[1]
# 매개변수를 URL 인코딩합니다. --- (※3)
API = "http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"
values = {
    'stnId': regionNumber
}
params = parse.urlencode(values)
url = API + "?" + params
print("url=", url)
# 다운로드합니다. --- (※3)
data = req.urlopen(url).read()
text = data.decode("utf-8")
print(text)
```


https://velog.io/@city7310/파이썬으로-URL-가지고-놀기

웹으로부터 정보를 추출하는 방법(5)

```
PS E:₩TIS₩pyml_lesson₩pyml> python .₩ch01₩1-1_download-forecast-argv.py 109 >forcast-109.html
PS E:₩TIS₩pyml_lesson₩pyml>
```



1-2 BeautifulSoup 스크레이핑(scraping)

스크레이핑 할 수 있게 하는 파이썬 라이브러리, 원하는 정보를 추출하는 방법을 학습합니다.

- 스크레이핑(Scraping):
  - 인터넷에 존재하는 데이터를 웹사이트에서 추출하는 방법
- BeautifulSoup 라이브러리
  - Html, xml을 분석해주는 파이썬 라이브러리
  - Pip 명령어를 사용해서 설치
    - $ pip install beautifulsoup4

BeautifulSoup 스크레이핑(scraping)

- BeautifulSoup 라이브러리
  - HTML의 DOM구조에서 문자열을 추출하는 방식
- 개발자가 할 일
  - **크롤링 하고자 하는 웹 페이지의 DOM을 분석하는 것**

BeautifulSoup

- 설치
  - $ pip install beautifulsoup4
- Package import
  - from bs4 import BeautifulSoup
- HTML 파일열기
  - with open(" example.html") as fp:
  - soup = BeautifulSoup(fp, 'html.parser')

BeautifulSoup example

```
# package import
from bs4 import BeautifulSoup
import os
os.getcwd()
# 1) html 파일 열기
with open("./ch01/example.html") as fp:
  soup = BeautifulSoup(fp, 'html.parser')
soup
```

beautifulsoup-example.py

웹에 있는 소스 가져오기 - urllib

```
# 2) urllib를 통해서 웹에 있는 소스 가져오기
import urllib.request
import urllib.parse

# web_url에 원하는 웹의 URL을 넣어주시면 됩니다.
os.getcwd()
# web_url = "./ch01/example.html"
web_url = "https://news.v.daum.net/v/20200321131028118"
with urllib.request.urlopen(web_url) as response:
    html = response.read()
    soup = BeautifulSoup(html, 'html.parser')
soup
```

beautifulsoup-example.py

웹에 있는 소스 가져오기 - requests

```
# 3) requests를 통해서 웹에 있는 소스 가져오기
import requests

# web_url에 원하는 웹의 URL을 넣어주시면 됩니다.
web_url = "https://news.v.daum.net/v/20200321131028118"
r = requests.get(web_url)
r.status_code
r.headers['content-type']
r.encoding
r.text
```

beautifulsoup-example.py

웹에 있는 소스 가져오기 example

```
<!DOCTYPE html>
<html>
        <head>
                <title>Page title</title>
        </head>
        <body>
        <div>
            <p>a</p>
            <p>b</p>
            <p>c</p>
        </div>
```

example.html

웹에 있는 소스 가져오기 example

```
<div class="ex_class">
        <p>d</p>
        <p>e</p>
        <p>f</p>
    </div>
    <div id="ex_id">
        <p>g</p>
        <p>h</p>
        <p>i</p>
    </div>
            <h1>This is a heading</h1>
            <p>This is a paragraph.</p>
            <p>This is another paragraph.</p>
        </body>
</html>
```

example.html

웹에 있는 소스 가져오기 - find_all

- # 4) find() 및 find_all()함수
- # 함수 인자로는 찾고자 하는 태그의 이름, 속성 기타 등등이 들어간다.
- # find_all(name, attrs, recursive, string, limit, **kwargs)
- # find(name, attrs, recursive, string, **kwargs)

beautifulsoup-example.py

https://twpower.github.io/84-how-to-use-beautiful-soup

웹에 있는 소스 가져오기 example

```
# 출력 결과
[<div>
<p>a</p>
<p>b</p>
<p>c</p>
</div>, <div class="ex_class">
<p>d</p>
<p>e</p>
<p>f</p>
</div>, <div id="ex_id">
<p>g</p>
<p>h</p>
<p>i</p>
</div>]
```

beautifulsoup-example.py

웹에 있는 소스 가져오기 - find_all

```
# 4-1) find_all() : 해당 조건에 맞는 모든 태그들을 가져온다.
with open("./ch01/example.html") as fp:
    soup = BeautifulSoup(fp, 'html.parser')
    all_divs = soup.find_all("div")
    print(all_divs)
```

beautifulsoup-example.py

https://twpower.github.io/84-how-to-use-beautiful-soup

웹에 있는 소스 가져오기 - find

```
# 4-2) find() : 해당 조건에 맞는 하나의 태그를 가져온다.
# 중복이면 가장 첫 번째 태그를 가져온다.
html = "./ch01/example.html"
with open(html) as fp:
    soup = BeautifulSoup(fp, 'html.parser')
    first_div = soup.find("div")
    print(first_div)
```

beautifulsoup-example.py

https://twpower.github.io/84-how-to-use-beautiful-soup

웹에 있는 소스 가져오기 – find_all

```
# 4-3) 태그를 이용해서 가져오기
# 예제 : 모든 <p> 태그들을 가져오기
html = "./ch01/example.html"
with open(html) as fp:
    soup = BeautifulSoup(fp, 'html.parser')
    all_ps = soup.find_all("p")
    print(all_ps)
```

beautifulsoup-example.py

https://twpower.github.io/84-how-to-use-beautiful-soup

웹에 있는 소스 가져오기 – find_all

```
# 4-4) 예제 : 첫번째 <div>태그를 가져오기
html = "./ch01/example.html"
with open(html) as fp:
    soup = BeautifulSoup(fp, 'html.parser')
    first_div = soup.find("div")
    print(first_div)
```

beautifulsoup-example.py

https://twpower.github.io/84-how-to-use-beautiful-soup

웹에 있는 소스 가져오기 – find_all

- # 태그와 속성을 이용해서 가져오기
- # 태그와 속성을 이용할 때 함수의 인자로 원하는 태그를 첫 번째 인자로 그 다음에 속성:값의 형태로 dictionary 형태로 만들어서 넣어주면 된다.
- # find_all('태그명', {'속성명' : '값' ...})
- # find('태그명', {'속성명' : '값' ...})

beautifulsoup-example.py

https://twpower.github.io/84-how-to-use-beautiful-soup

웹에 있는 소스 가져오기 – find 응용

```
# HTML 구조를 이용해 원하는 부분 가져오기
# 예제 4-5) : <div> 태그에서 id속성의 값이 ex_id 인 것을  찾기
html = "./ch01/example.html"
with open(html) as fp:
    soup = BeautifulSoup(fp, 'html.parser')
    ex_id_divs = soup.find('div', {'id' : 'ex_id'})
    print(ex_id_divs)
```

beautifulsoup-example.py

https://twpower.github.io/84-how-to-use-beautiful-soup

웹에 있는 소스 가져오기 – find 응용

```
# 예제 4-6) : id속성의 값이 ex_id인 <div> 태그에서 <p>태그들만 가져오기
with open("example.html") as fp:
    soup = BeautifulSoup(fp, 'html.parser')
    # id=ex_id인 div 태그를 가져와서
    ex_id_divs = soup.find("div", {"id":"ex_id"})
    # 그 태그들 안에서 p 태그를 가져온다.
    all_ps_in_ex_id_divs = ex_id_divs.find_all("p")
    print(all_ps_in_ex_id_divs)
```

beautifulsoup-example.py

https://twpower.github.io/84-how-to-use-beautiful-soup

BeautifulSoup scraping (ex1) (1/2)

```
# 라이브러리 읽어 들이기 --- (※1)
from bs4 import BeautifulSoup
# 분석하고 싶은 HTML --- (※2)
html = """
<html><body>
  <h1>스크레이핑이란?</h1>
  <p>웹 페이지를 분석하는 것</p>
  <p>원하는 부분을 추출하는 것</p>
</body></html>
"""
```

1-2_bs-test1.py

BeautifulSoup scraping (ex1) (2/2)

```
# HTML 분석하기 --- (※3)
soup = BeautifulSoup(html, 'html.parser')
# 원하는 부분 추출하기 --- (※4)
h1 = soup.html.body.h1
p1 = soup.html.body.p
p2 = p1.next_sibling.next_sibling
# 요소의 글자 출력하기 --- (※5)
print("h1 = " + h1.string)
print("p  = " + p1.string)
print("p  = " + p2.string)
```

1-2_bs-test1.py

BeautifulSoup scraping (ex1) (output)

```
PS E:₩TIS₩pyml_lesson₩pyml>
C:/ProgramData/Anaconda3/python.exe
e:/TIS/pyml_lesson/pyml/ch01/1-2_bs-test1.py
h1 = 스크레이핑이란?
p  = 웹 페이지를 분석하는 것
p  = 원하는 부분을 추출하는 것
```

1-2_bs-test1.py

라이브러리 임포트 방법

- import bs4
- from bs4 import BeautifulSoup
- from bs4 import *
- import bs4 as beauty

라이브러리 임포트 방법 (cont)

```
(pyml) PS F:₩tis₩pyml_lesson₩pyml₩ch01> python
Type "help", "copyright", "credits" or "license" for more information.
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__']

>>> import os
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__']

(pyml) PS F:₩tis₩pyml_lesson₩pyml₩ch01> python
>>> from os import listdir
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__']
>>>
```

from os import listdir 은 os패키지로부터 listdir을 import 하라는 명령어

라이브러리 임포트 방법 (cont)

```
>>> os
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'os' is not defined
>>>
>>> os.listdir()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'os' is not defined
>>>
>>> listdir()
['.anaconda', '.android', '.AndroidStudio3.6', '.conda', '.condarc', '.continuum', '.docker', '.dotnet',
'.ipynb_checkpoints', '.ipython', '.jupyter', '.keras', '.matplotlib', '.packettracer', '.pylint.d',
'.python_history', '.vscode', '3D Objects', 'ansel', 'AppData', 'Application Data', 'CiscoPacketTracer6.2sv',
'Contacts', 'Cookies', 'Desktop', 'Documents', 'Downloads', 'Favorites', 'Intel', 'IntelGraphicsProfiles',
'Links', 'Local
```

'from os import listdir'을 하는 순간 해당 이름의 변수나 함수가 없어지고 os 모듈의 listdir로 대체
os라는 모듈명을 사용하거나 os.listdir()과 같이 os 모듈 내의 listdir 함수를 호출할 경우 오류가 발생, listdir()과 같이 해당 함수를 직접 이용하는 방법만 가능

함수 이름만으로도 바로 함수 호출이 가능하므로 프로그래밍할 코드의 수가 적어진다는 장점
그러나 두 번째 방식은 기존에 선언된 변수나 함수와 이름이 충돌할 가능성이 존재함

https://wikidocs.net/3105

Id요소로 찾기(ex2) (1/2)

```
from bs4 import BeautifulSoup

html = """
<html><body>
  <h1 id="title">스크레이핑이란?</h1>
  <p id="body">웹 페이지를 분석하는 것</p>
  <p>원하는 부분을 추출하는 것</p>
</body></html>
"""
```

1-2_bs-test2.py

Id요소로 찾기(ex2) (2/2)

```
# HTML 분석하기 --- (※1)
soup = BeautifulSoup(html, 'html.parser')

# find() 메서드로 원하는 부분 추출하기 --- (※2)
title = soup.find(id="title")
body = soup.find(id="body")

# 텍스트 부분 출력하기
print("#title=" + title.string)
print("#body=" + body.string)
```

1-2_bs-test2.py

Id요소로 찾기 (ex2) (output)

```
PS E:₩TIS₩pyml_lesson₩pyml> &
C:/ProgramData/Anaconda3/python.exe
e:/TIS/pyml_lesson/pyml/ch01/1-2_bs-test2.py
#title=스크레이핑이란?
#body=웹 페이지를 분석하는 것
PS E:₩TIS₩pyml_lesson₩pyml>
```

링크목록 찾기 (ex3)

```
from bs4 import BeautifulSoup
html = """
<html><body>
  <ul>
    <li><a href="http://www.naver.com">naver</a></li>
    <li><a href="http://www.daum.net">daum</a></li>
  </ul>
</body></html>
"""
```

1-2_bs-test2.py

링크목록 찾기 (ex3)

```
# HTML 분석하기 --- (※1)
soup = BeautifulSoup(html, 'html.parser')
# find_all() 메서드로 추출하기 --- (※2)
links = soup.find_all("a")
# 링크 목록 출력하기 --- (※3)
for a in links:
    href = a.attrs['href']
    text = a.string
    print(text, ">", href)
```

1-2_bs-test2.py

DOM 요소의 속성

- DOM(Document Object Model)
  - XML 또는 HTML의 요소에 접근하는 구조
- DOM 요소의 속성
  - 태그 이름 뒤에 있는 각 속성을 지칭

1-2_bs-test2.py

REPL 을 사용해서 분석하기 ex(1/3)

(Read **E**valuate **P**rint **L**oop의 줄임말, 파이썬의 실행 콘솔 화면)

```
from bs4 import BeautifulSoup
html = """
<html><body>
  <ul>
    <li>
      <p>
      <a href="http://www.naver.com">
        test
      </a>
      </p></li>
    <li><a href="http://www.daum.net">daum</a></li>
  </ul>
</body></html>
"""
```

1-2_bs-prettify-test.py

REPL 을 사용해서 분석하기 ex(2/3)

```
soup = BeautifulSoup(html, 'html.parser')
# 분석이 제대로 됐는지 확인 --- (※1)
soup.prettify()
a = soup.p.a
# attrs 속석의 자료형 확인 --- (※2)
type(a.attrs)

#href 속성이 있는지 확인
'href' in a.attrs

# href 속성값 확인
a['href']
```

prettify() 메소드를 이용하여 분석이 제대로 되었는지 확인

a.attrs의 type은 dict 즉 딕셔너리

딕셔너리에서 'href' 속성이 있는지 확인

1-2_bs-prettify-test.py

REPL 을 사용해서 분석하기 ex(3/3)

```
>>> type(a.attrs)
<class 'dict'>
>>> #href 속성이 있는지 확인
...
>>> 'href' in a.attrs
True
>>> # href 속성값 확인
...
>>> a['href']
'http://www.naver.com'
>>>
```

1-2_bs-prettify-test.py

urlopen()과 BeautifulSoup 조합하기

```
from bs4 import BeautifulSoup
import urllib.request as req
url = "http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"
# urlopen()으로 데이터 가져오기 --- (※1)
res = req.urlopen(url)
# BeautifulSoup으로 분석하기 --- (※2)
soup = BeautifulSoup(res, "html.parser")
# 원하는 데이터 추출하기 --- (※3)
title = soup.find("title").string
wf = soup.find("wf").string
print(title)
print(wf)
```

1-2_bs-forecast.py

BeautifulSoup html 파싱, 크롤링 – example2

```
 # package import
from bs4 import BeautifulSoup

html = '<td id="td1" class="title">' ₩
       '  <div class="tit3">' ₩
       '    <a href="/movie/bi/mi/basic.nhn?code=161242" title="범죄도시">범죄도시</a>' ₩
       '  </div>' ₩
       '</td>'
```

beautifulsoup-example2.py

https://victorydntmd.tistory.com/245

BeautifulSoup html 파싱, 크롤링 – example2

```
# 1. 조회
def ex1():
    # BeautifulSoup객체생성 ( html문자열, 파싱방법을 지정 )
    bs = BeautifulSoup(html, 'html.parser')
    print(bs, type(bs))

    # a 태그 출력
    tag = bs.a
    print(tag, type(tag))
```

beautifulsoup-example2.py

https://victorydntmd.tistory.com/245

BeautifulSoup html 파싱, 크롤링 – example2

```
# 2. Attribute 값 받아오기
def ex2():
    bs = BeautifulSoup(html, 'html.parser')

    tag = bs.td
    print(tag['class'])      # ['title']      => 리스트
    print(tag['id'])         # td1
    print(tag.attrs)         # {'id': 'td1', 'class': ['title']}    => 딕셔너리

    tag = bs.div
    print(tag['id'])         # id가 없으므로 error
```

beautifulsoup-example2.py

https://victorydntmd.tistory.com/245

BeautifulSoup html 파싱, 크롤링 – example2

```
# 3. Attribute 검색
def ex3():
    bs = BeautifulSoup(html, 'html.parser')

    # div 태그 중, class가 tit3인 태그를 찾는다.
    tag = bs.find('div', attrs={'class': 'tit3'})
    print(tag)

    tag = bs.find('div')
    print(tag)
```

beautifulsoup-example2.py

https://victorydntmd.tistory.com/245

```
    # 없는 태그를 조회할 경우
    tag = bs.find('td', attrs={'class': 'not_exist'})
    print(tag)       # None

    # 전체 태그에 대해 title이 범죄도시인 태그를 찾는다.
    tag = bs.find(attrs={'title': '범죄도시'})
    print(tag)       # <a href="/movie/bi/mi/basic.nhn?code=161242" title="범죄도시">범죄도시
</a>
```

beautifulsoup-example2.py

https://victorydntmd.tistory.com/245

BeautifulSoup htmBeautifulSoup html 파싱, 크롤링 – example2I 파싱, 크롤링 – example2

```
# 4. select(), content() 메서드
def ex4():
    bs = BeautifulSoup(html, 'html.parser')

    # CSS 처럼 셀렉터를 지정할 수 있다.
    tag = bs.select("td div a")[0]
    print(tag)

    text = tag.contents[0]
    print(text)      # 범죄도시
```

beautifulsoup-example2.py

https://victorydntmd.tistory.com/245

BeautifulSoup html 파싱, 크롤링 – example2

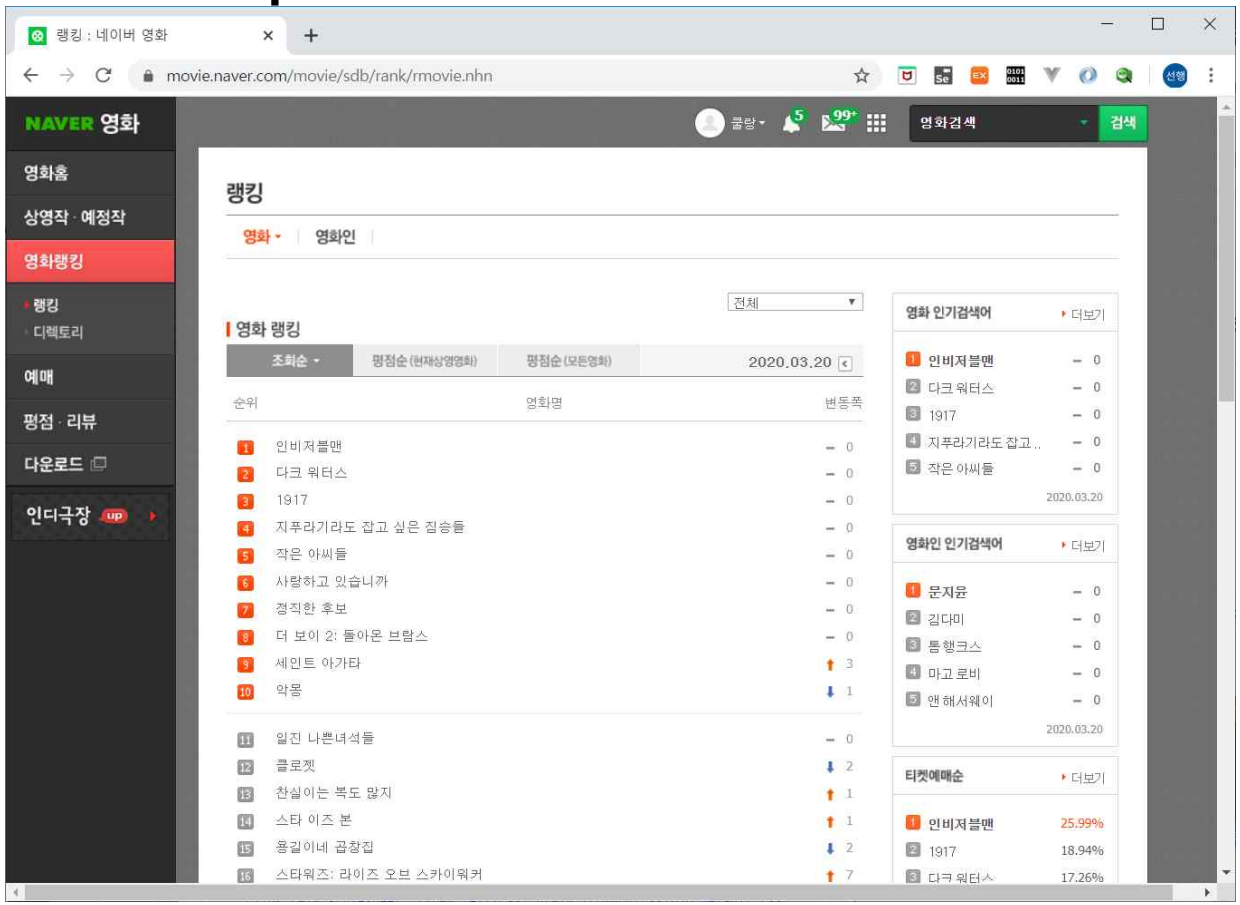
```
# 5. extract() 메서드
def ex5():
    bs = BeautifulSoup(html, 'html.parser')
    tag = bs.select("td")[0]
    print(tag)

    # div요소를 제거
    div_elements = tag.find_all("div")
    for div in div_elements:
        div.extract()
    print(tag)       # <td class="title" id="td1"> </td>
```

beautifulsoup-example2.py

https://victorydntmd.tistory.com/245

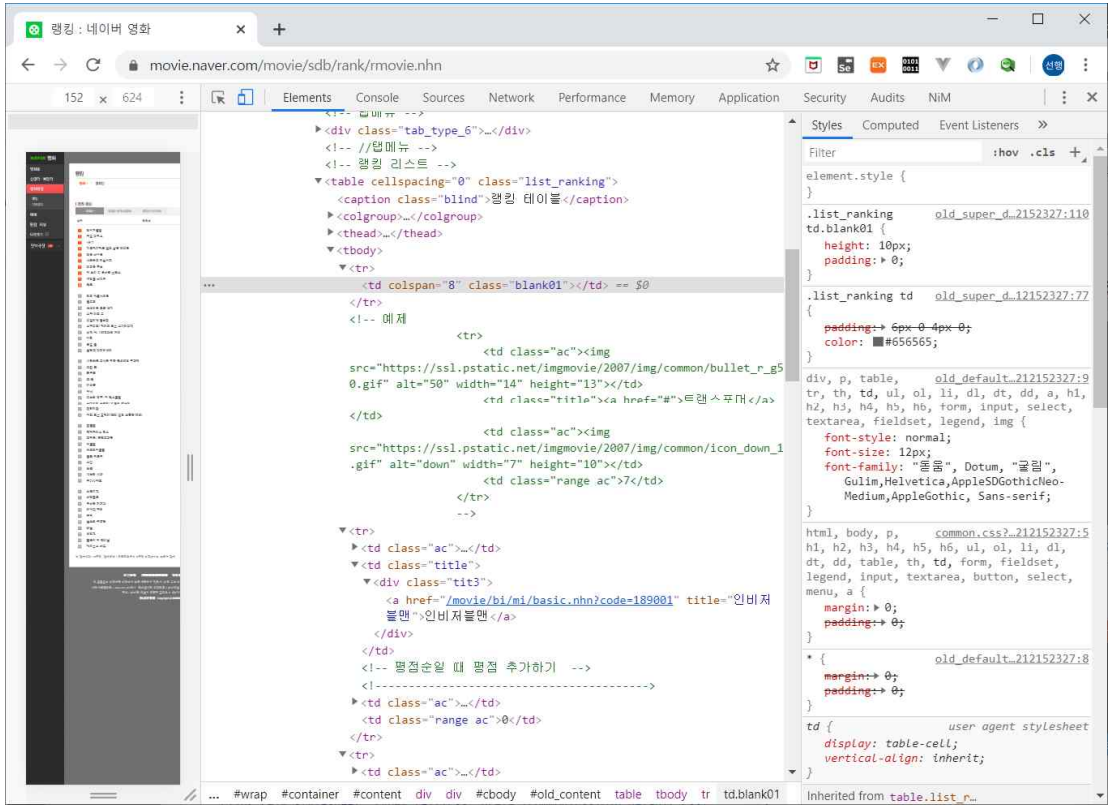
BeautifulSoup - 네이버 영화 크롤링



beautifulsoup-example2.py

https://victorydntmd.tistory.com/245

BeautifulSoup - 네이버 영화 크롤링



beautifulsoup-example2.py

https://victorydntmd.tistory.com/245

BeautifulSoup - 네이버 영화 크롤링

```
                    -->
  ▼<tr> == $0
    ▶<td class="ac">…</td>
    ▼<td class="title">
      ▼<div class="tit3">
          <a href="/movie/bi/mi/basic.nhn?code=189001" title="인비저블맨">인비저블맨</a>
        </div>
      </td>
      <!-- 평점순일 때 평점 추가하기  -->
      <!------------------------------------------>
    ▶<td class="ac">…</td>
      <td class="range ac">0</td>
    </tr>
```

beautifulsoup-example3.py

https://victorydntmd.tistory.com/245

BeautifulSoup - 네이버 영화 크롤링

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

req = Request('http://movie.naver.com/movie/sdb/rank/rmovie.nhn')
res = urlopen(req)
# html = res.read().decode('cp949')
html = res.read().decode('utf-8')

# 영화 제목에는 <div class="tit3"> 태그
bs = BeautifulSoup(html, 'html.parser')
tags = bs.findAll('div', attrs={'class': 'tit3'})
```

cp949로 decode한 이유는 네이버 영화의 인코딩 타입이 euc-kr 이기 때문
페이지 소스 보기를 열고 charset=euc-kr 을 검색해보세요.

beautifulsoup-example3.py

https://victorydntmd.tistory.com/245

BeautifulSoup - 네이버 영화 크롤링

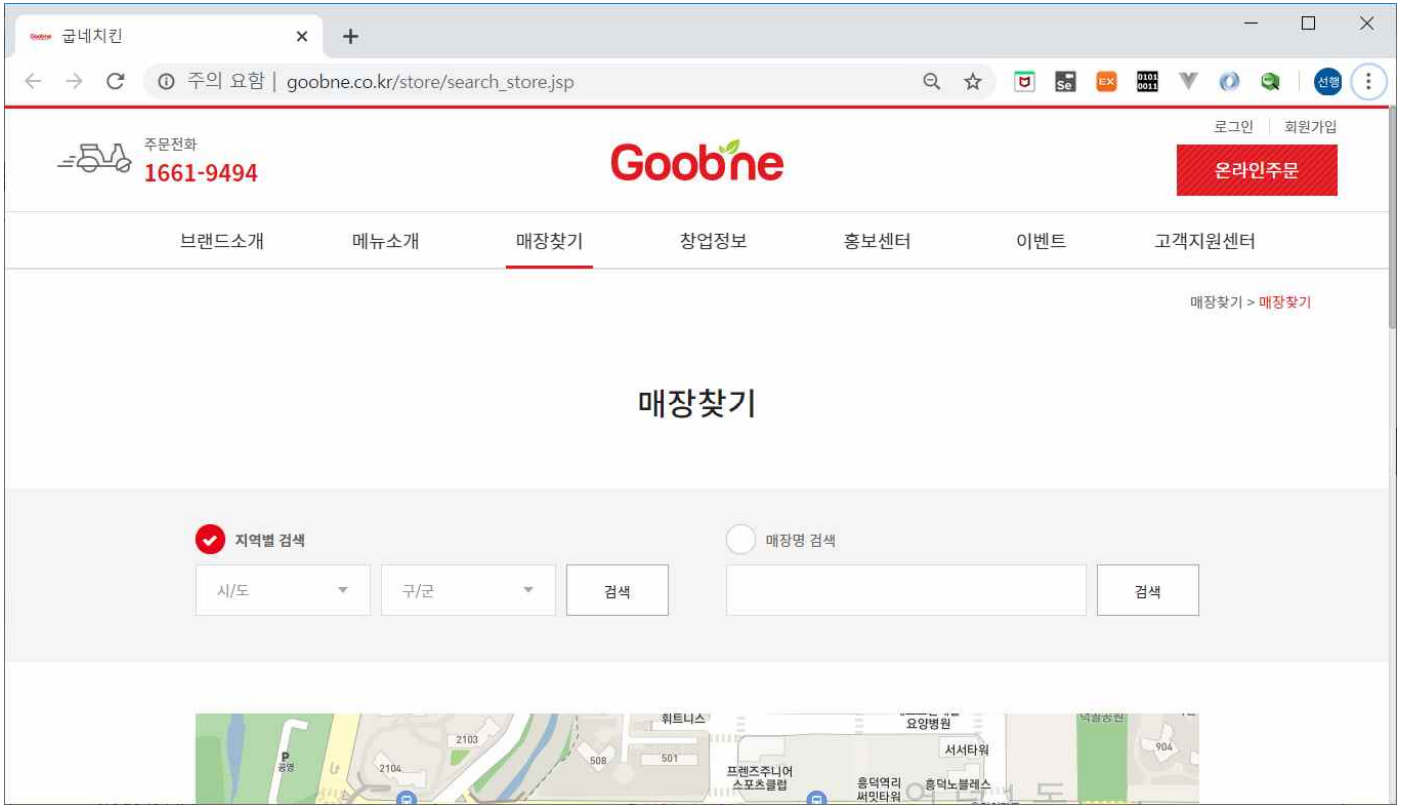
```
for tag in tags :
    # 검색된 태그에서 a 태그에서 텍스트를 가져옴
    print(tag.a.text)

# 인덱스를 주고 싶다면 enumerate를 사용한다.
for index, tag in enumerate(tags):
   print(str(index) + " : " + tag.a.text)
```

beautifulsoup-example3.py

https://victorydntmd.tistory.com/245

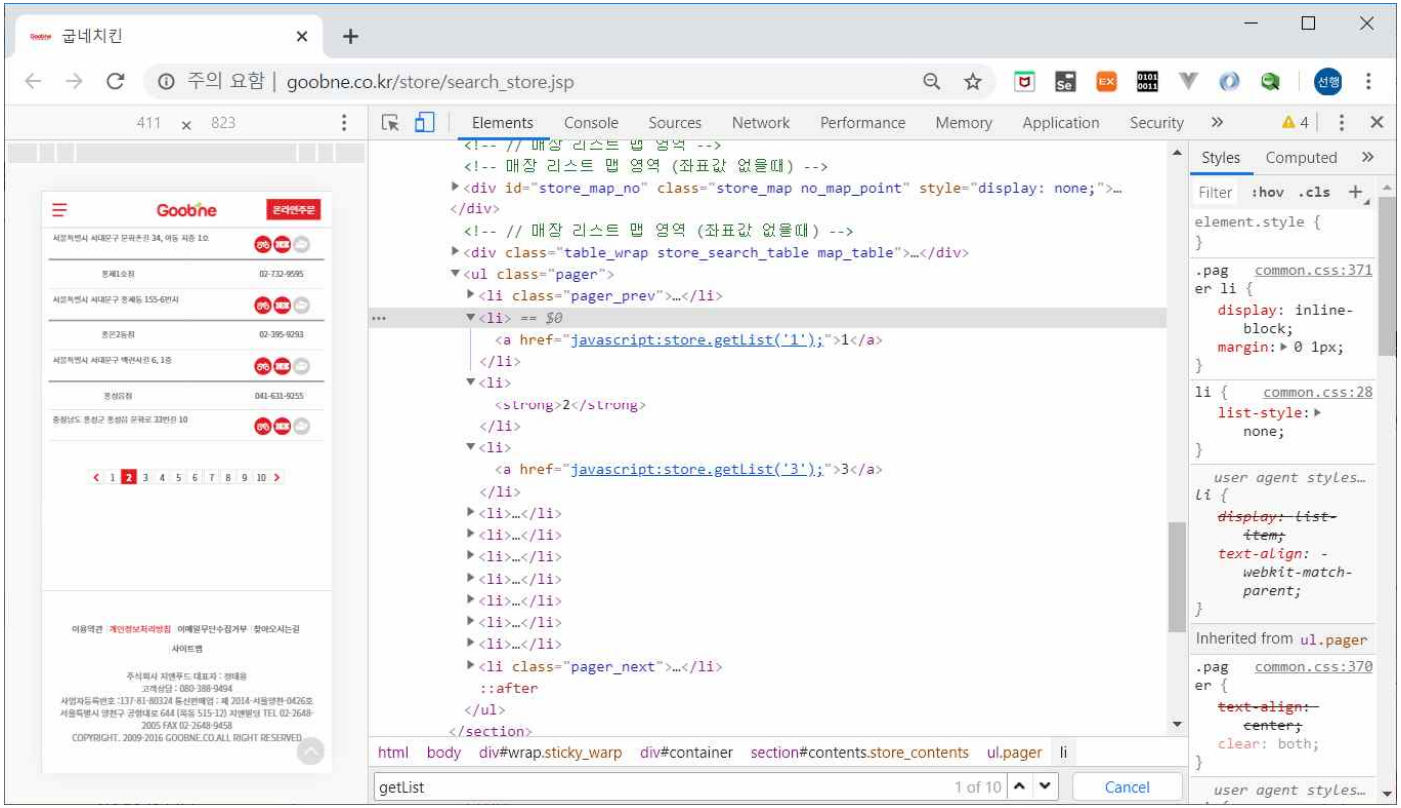
BeautifulSoup - 굽네치킨 매장 정보



beautifulsoup-example3.py

https://victorydntmd.tistory.com/245

BeautifulSoup - 굽네치킨 매장 정보



beautifulsoup-example3.py

https://victorydntmd.tistory.com/245

BeautifulSoup - 굽네치킨 매장 정보

굽네치킨 사이트(링크)의 경우 매장 리스트가 페이징으로 되어 있는데, 페이지를 클릭하면 URL 에서 query string으로 page 번호가 표시되는 것이 아니라 JS 코드로 실행됩니다.

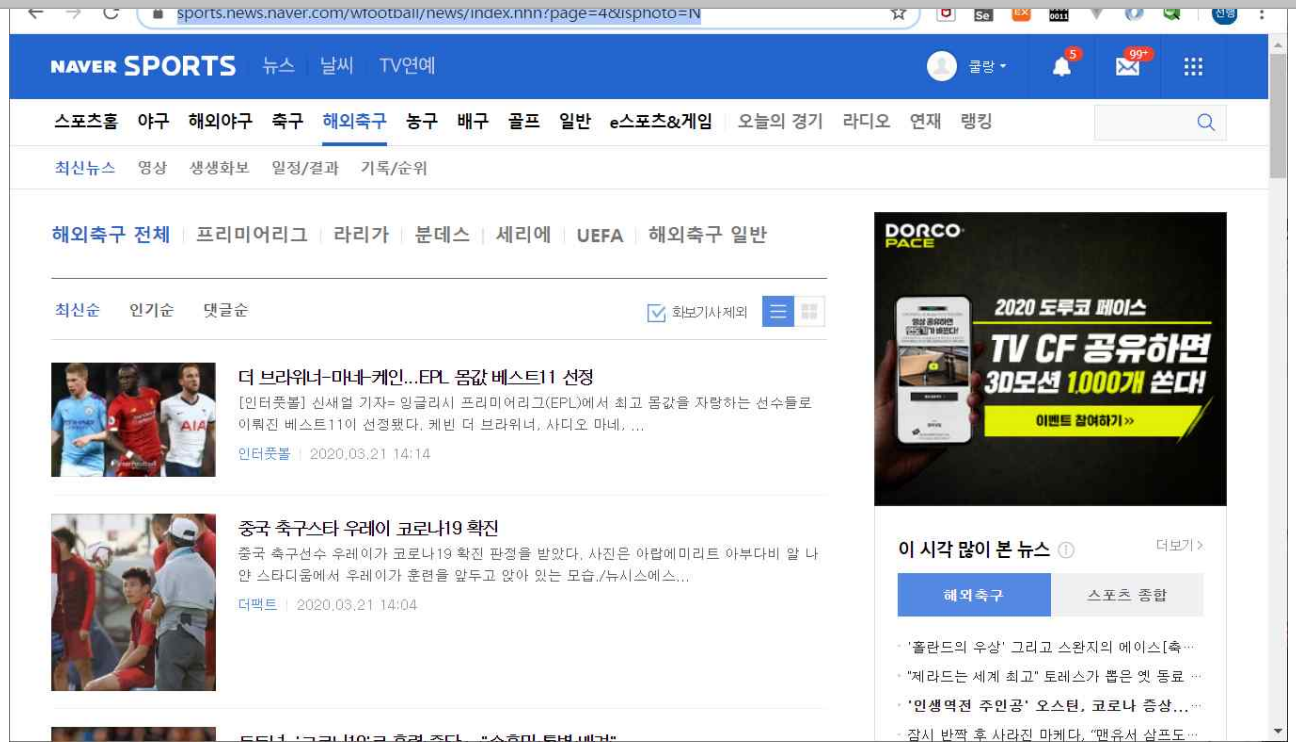
beautifulsoup-example3.py

https://victorydntmd.tistory.com/245

BeautifulSoup - 굽네치킨 매장 정보

https://sports.news.naver.com/wfootball/news/index.nhn?page=4&isphoto=N

Htt request get 방식으로 url에서 페이지 번호가 표시



beautifulsoup-example3.py

https://victorydntmd.tistory.com/245

BeautifulSoup - **굽네치킨 매장 정보**

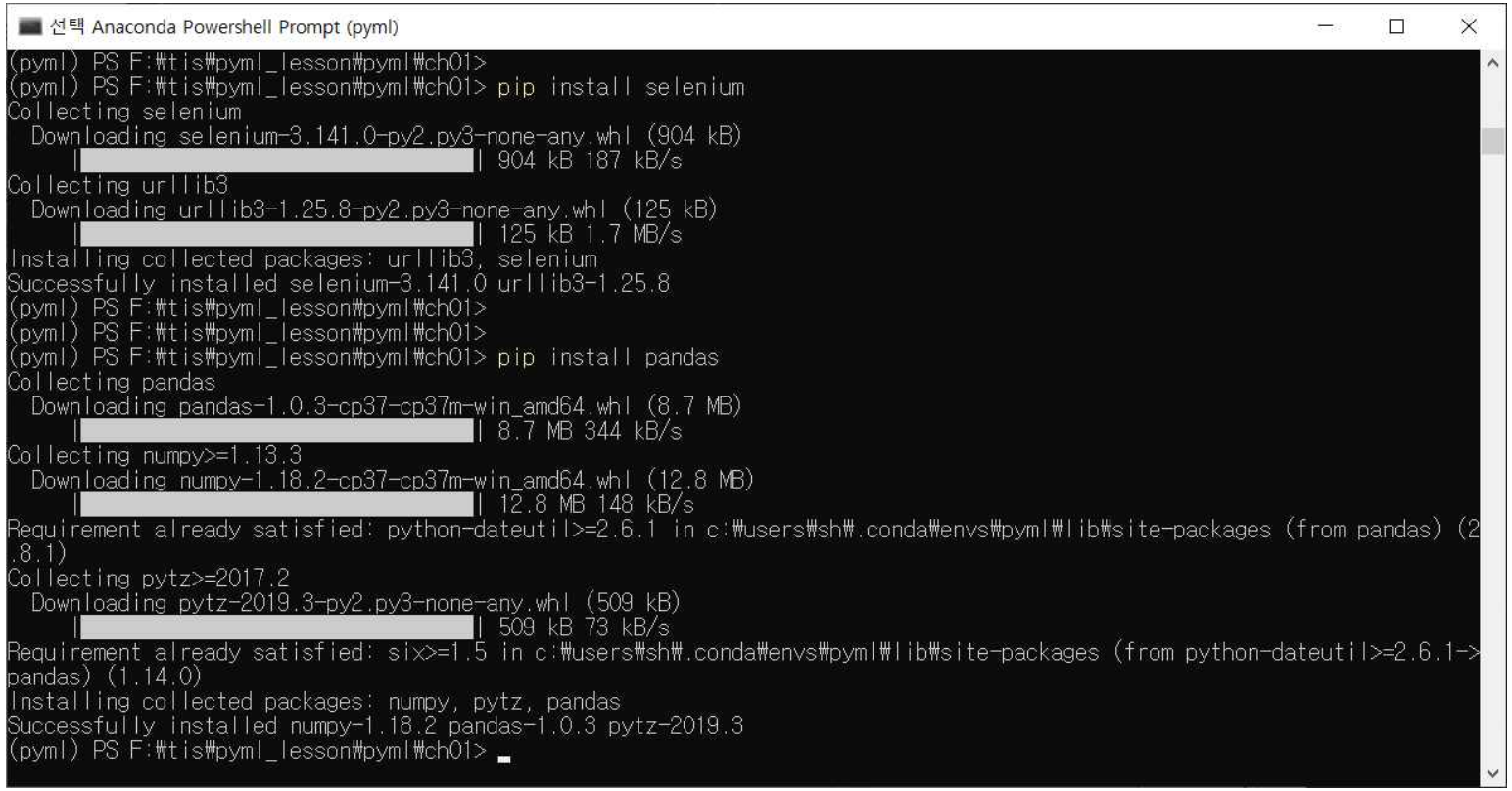
- JS로 실행되는 페이징을 크롤링 하기 위해서는 **selenium 라이브러리**를 사용하여 파이썬 코드에서 직접 화면의 JS를 실행해서 페이지번호를 받아와야함
- selenium은 BeautifulSoup을 설치했을 때와 마찬가지로 패키지설치 후 사용, **webdriver** 인터페이스와 함께 설치 (**링크**)
- https://chromedriver.storage.googleapis.com/index.html?path=2.40/
- > pip3 install selenium
- webdirver를 다운받은 후, 실행파일을 원하는 경로를 선택 (예 C:₩programs₩python₩webdriver)
- **pandas** 는 데이터 분석 라이브러리, 크롤링 결과를 csv 파일로 저장하기 위한 용도로 설치해서 사용
- > pip3 install pandas

※ pip 는 python version2의 명령어이고
pip3은 python version 3의 명령어이다.
Conda의 python 3을 기본 언어로 설치했으므로 pip3을 사용합니다.

beautifulsoup-example3.py

https://victorydntmd.tistory.com/245

pip 설치 - selenium, pasda



beautifulsoup-example3.py

https://victorydntmd.tistory.com/245

BeautifulSoup - 굽네치킨 매장 정보

DOM구조에서, 매장의 이름은 <tbody> 아래에 <tr class="lows">태그 아래에 있는 것을 참고하여 beautifulSoup 구문을 작성

beautifulsoup-example3.py

https://victorydntmd.tistory.com/245

selenium - example

```
import time
from selenium import webdriver

# 드라이버 파일 위치
path = "C:/programs/chromedriver/chromedriver.exe"
 #조금만 기다리면 selenium으로 제어할 수 있는 브라우저 새창이 뜬다
driver = webdriver.Chrome(path)

#webdriver가 google 페이지에 접속하도록 명령
driver.get('https://www.google.com')

time.sleep(5)  #  5초 정지

#webdriver를 종료하여 창이 사라진다
driver.close()
```

Selenium-test.py

pandas csv save - example

```
import pandas as pd

data = [[1,2,3,4], [5,6,7,8]]

dataframe = pd.DataFrame(data)

dataframe.to_csv("./ch01/pandas-save.csv", header=False, index=False)
```

pandas-save.py

CSS 선택자 사용하기

BeautifulSoup는 jQuery처럼 CSS 선택자를 지정해서 원하는 요소를 추출하는 기능도 제공

메서드	설명
soup.select_one(<선택자>)	CSS 선택자로 요소 하나를 추출합니다.
soup.select(<선택자>)	CSS 선택자로 요소 여러 개를 리스트로 추출합니다.

1-2_bs-select.py

CSS 선택자 사용하기 - 1/2

```
from bs4 import BeautifulSoup

# 분석 대상 HTML --- (※1)
html = """
<html><body>
<div id="meigen">
  <h1>위키북스 도서</h1>
  <ul class="items">
    <li>유니티 게임 이펙트 입문</li>
    <li>스위프트로 시작하는 아이폰 앱 개발 교과서</li>
    <li>모던 웹사이트 디자인의 정석</li>
  </ul>
</div>
</body></html>
"""
```

1-2_bs-select.py

CSS 선택자 사용하기 2/2

```
# HTML 분석하기 --- (※2)
soup = BeautifulSoup(html, 'html.parser')
#----------------------------------------
# 필요한 부분을 CSS 쿼리로 추출하기
#----------------------------------------
# 1) 타이틀 부분 추출하기 --- (※3)
h1 = soup.select_one("div#meigen > h1").string
print("h1 =", h1)

# 2) 목록 부분 추출하기 --- (※4)
li_list = soup.select("div#meigen > ul.items > li")
for li in li_list:
  print("li =", li.string)
```

select 는 리스트 자료형를 생성합니다

1-2_bs-select.py

CSS 선택자 사용하기 - output

```
(pyml) PS F:₩tis₩pyml_lesson₩pyml₩ch01> python .₩1-2_bs-select.py
h1 = 위키북스 도서
li = 유니티 게임 이펙트 입문
li = 스위프트로 시작하는 아이폰 앱 개발 교과서
li = 모던 웹사이트 디자인의 정석

(pyml) PS F:₩tis₩pyml_lesson₩pyml₩ch01>
```

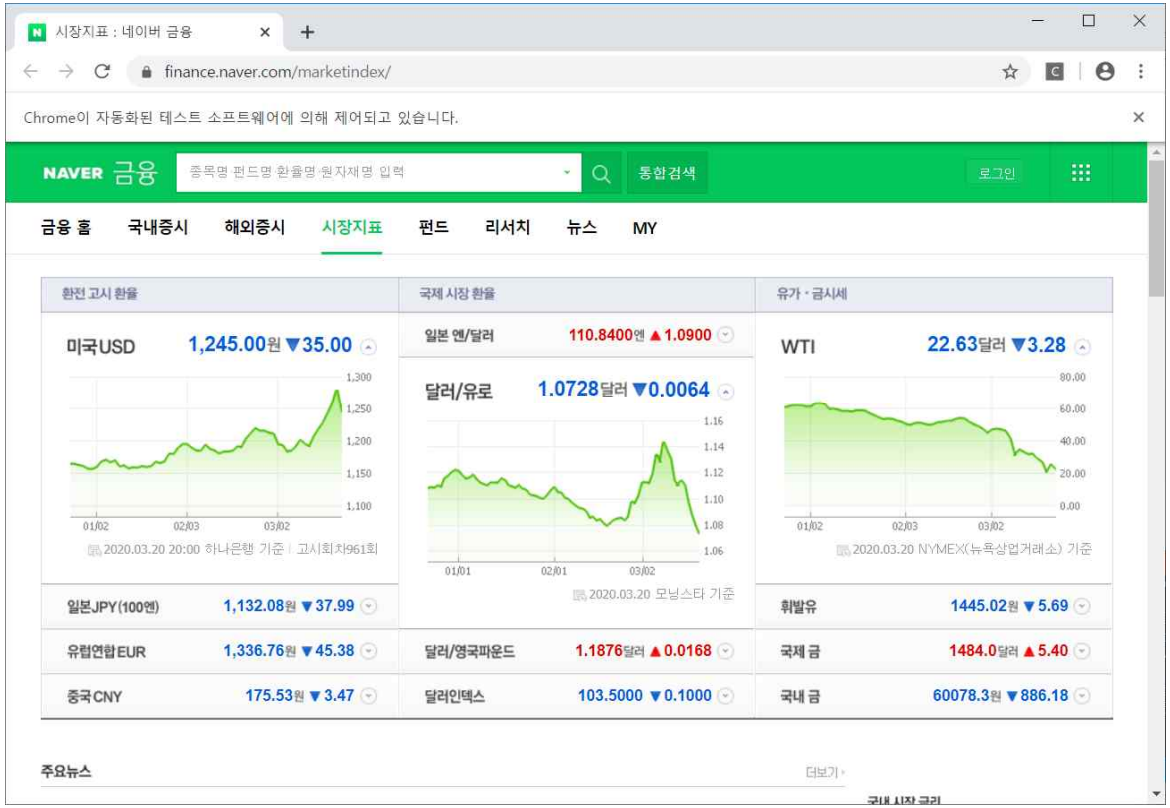
1-2_bs-select.py

네이버금융 환율정보 추출하기

다양한 금융 정보가 공개되어 있는 "네이버 금융" 에서 원/달러 환율 정보를 추출합니다.

1-2_bs-usd.py

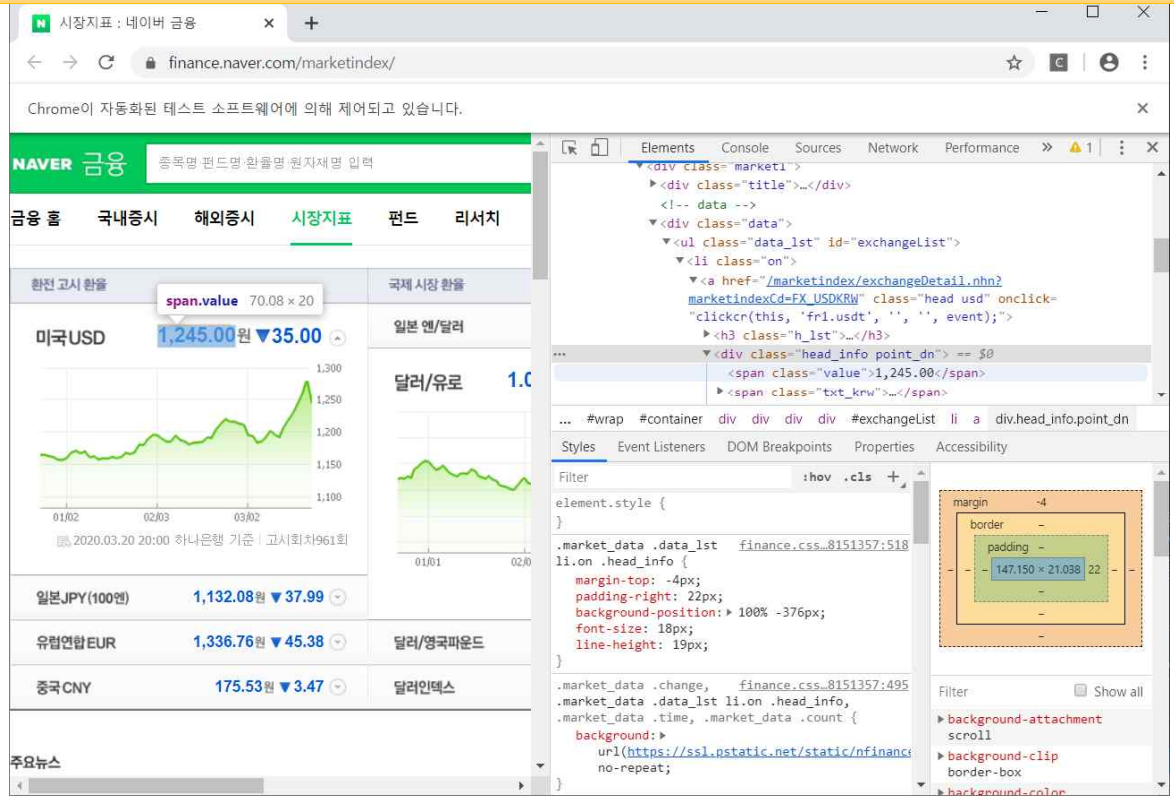
네이버금융 환율정보 추출하기



1-2_bs-usd.py

네이버금융 환율정보 추출하기

Debugger를 실행해서 소스를 확인합니다



1-2_bs-usd.py

네이버금융 환율정보 추출하기

원/달러 환율을 표시하는 코드는 head_info 클래스 다음 value 클래스 의 string

```
▼<li class="on">
  ▼<a href="/marketindex/exchangeDetail.nhn?
  marketindexCd=FX_USDKRW" class="head usd" onclick=
  "clickcr(this, 'fr1.usdt', '', '', event);">
    ▶<h3 class="h_lst">…</h3>
    ▼<div class="head_info point_dn">
        <span class="value">1,245.00</span> == $0
      ▶<span class="txt_krw">…</span>
        <span class="change"> 35.00</span>
        <span class="blind">하락</span>
      </div>
      ::after
    </a>
```

1-2_bs-usd.py

네이버금융 환율정보 추출하기

다양한 금융 정보가 공개되어 있는 "네이버 금융" 에서 원/달러 환율 정보를 추출합니다.

```
from bs4 import BeautifulSoup
import urllib.request as req
# HTML 가져오기
# url = "http://info.finance.naver.com/marketindex/"
url = "https://finance.naver.com/marketindex/"
res = req.urlopen(url)
# HTML 분석하기
soup = BeautifulSoup(res, "html.parser")
# 원하는 데이터 추출하기 --- (※1)
price = soup.select_one("div.head_info > span.value").string
print("usd/krw =", price)
```

1-2_bs-usd.py

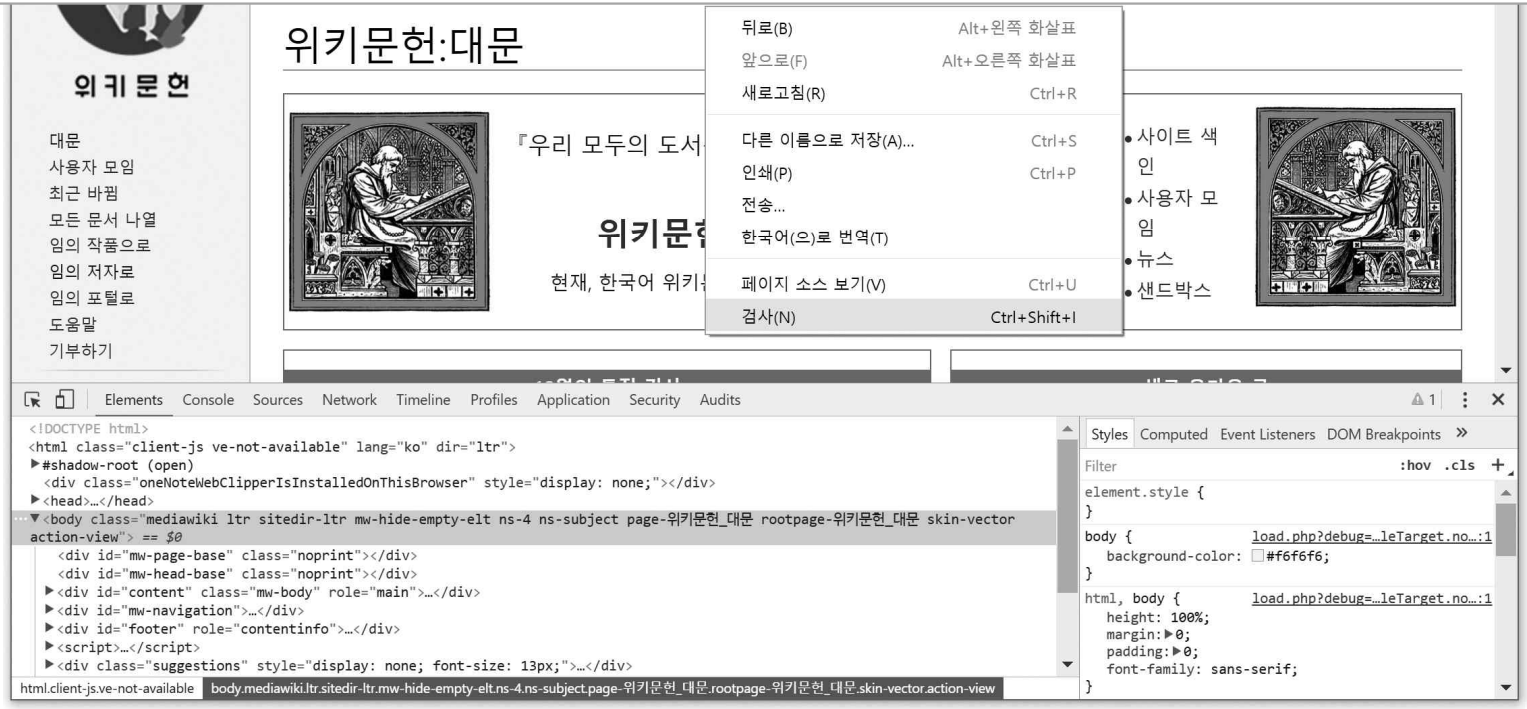
1-3. CSS 선택자

HTML 구조를 확인할 때는 CSS요소를 사용하는 것이 편리합니다.

웹브라우저에서 HTML구조확인하기

- 웹 브라우저가 제공하는 개발자 도구를 사용

1. 구글 크롬(Google Chrome)에서 분석하고 싶은 웹 페이지 위에 마우스 오른쪽 버튼 클릭
2. [검사] 선택



1-2_bs-usd.py

웹브라우저에서 HTML구조확인하기

원하는 요소 선택하기

3. 개발자 도구 왼쪽 위에 있는 요소 선택 아이콘 클릭
4. 페이지에서 조사하고 싶은 요소 클릭

1-2_bs-usd.py

웹브라우저에서 HTML구조확인하기

5. 태그를 선택한 상태로 마우스 오른쪽 버튼 클릭
6. 팝업 메뉴에서 [Copy > Copy selector] 클릭
7. 선택한 요소의 CSS 선택자가 클립보드에 복사됨

1-2_bs-usd.py

1-3. CSS 선택자

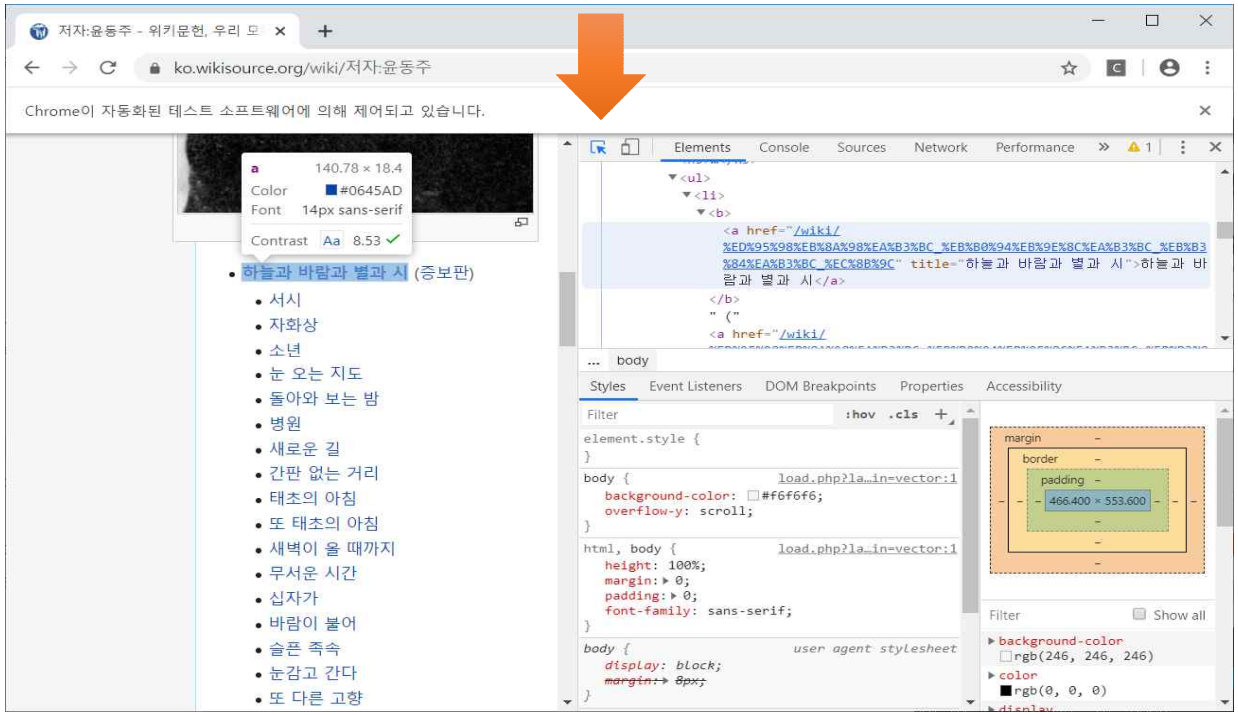
위키문헌에서 윤동주 작가 작품목록 가져오기

- 위키 문헌에 공개돼 있는 윤동주 작가의 작품 목록을 프로그램을 통해 가져오기
- https://ko.wikisource.org/wiki/저자:윤동주

1-3_sel-dongju.py

위키문헌에서 윤동주 작가 작품목록 가져오기

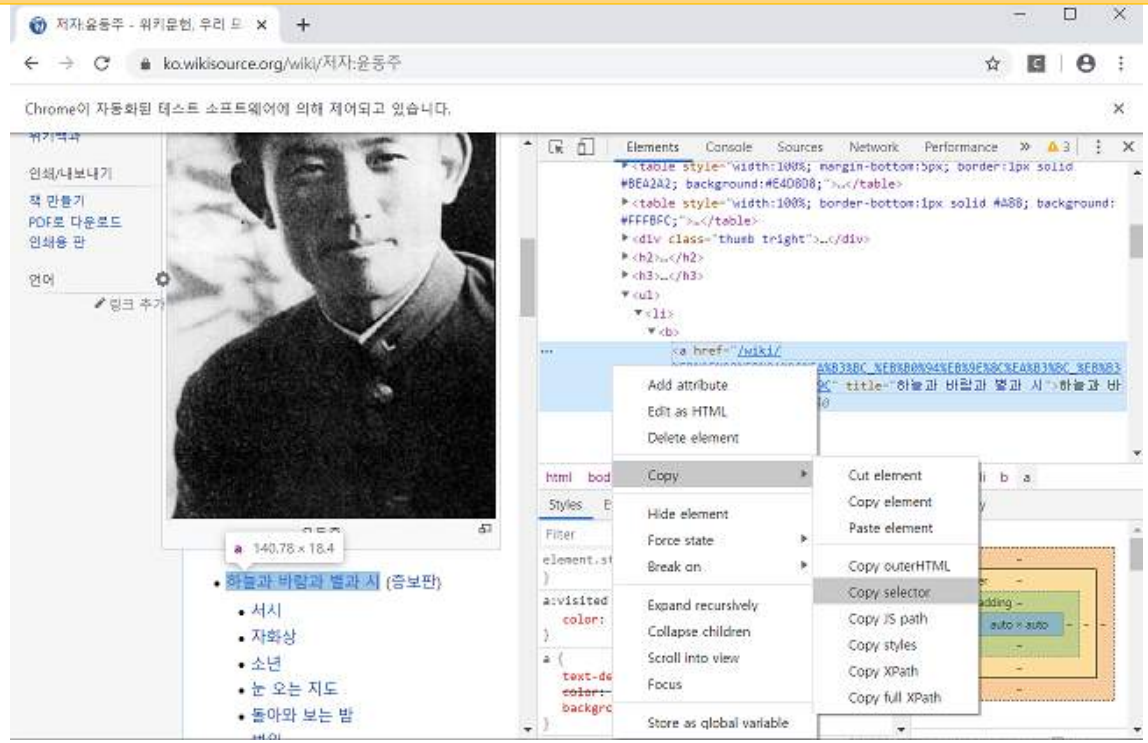
F12 개발자도구 > select an element page to inspect



1-3_sel-dongju.py

위키문헌에서 윤동주 작가 작품목록 가져오기

Elements창에서 마우스오른쪽 클릭 > copy>copy selecter



1-3_sel-dongju.py

위키문헌에서 윤동주 작가 작품목록 가져오기

- #mw-content-text > div > ul:nth-child(6) > li > b > a

- nth-child(n)은 n번째에 있는 요소를 의미
- nth-child(6)은 6번째에 있는 태그라는 의미

1-3_sel-dongju.py

file: src/ch1/sel-dongju.py

```
from bs4 import BeautifulSoup
import urllib.request as req


# 뒤의 인코딩 부분은 "저자:윤동주"라는 의미입니다.
# 따로 입력하지 말고 위키 문헌 홈페이지에 들어간 뒤에 주소를 복사해서 사용하세요.
url ="https://ko.wikisource.org/wiki/
        %EC%A0%80%EC%9E%90:%EC%9C%A4%EB%8F%99%EC%A3%BC"
res = req.urlopen(url)
soup = BeautifulSoup(res, "html.parser")

# #mw-content-text 바로 아래에 있는
# ul 태그 바로 아래에 있는
# li 태그 아래에 있는
# a 태그를 모두 선택합니다.
a_list = soup.select("#mw-content-text > div > ul > li a")

for a in a_list:
    name = a.string
    print("-", name)
```

위키문헌에서 윤동주 작가 작품목록 가져오기

```
(pyml) PS F:₩tis₩pyml_lesson₩pyml₩ch01> python .₩1-3_sel-dongju.py
- 하늘과 바람과 별과 시
- 증보판
- 서시
- 자화상
- 소년
- 눈 오는 지도
- 돌아와 보는 밤
- 병원
- 새로운 길
- 간판 없는 거리
- 태초의 아침
- 또 태초의 아침
- 새벽이 올 때까지
- 무서운 시간
- 십자가
```

1-3_sel-dongju.py

1-3. CSS 선택자

선택자 기본 서식

서식	설명
*	모든 요소를 선택합니다.
<요소 이름>	요소 이름을 기반으로 선택합니다.
.<클래스 이름>	클래스 이름을 기반으로 선택합니다.
#<id 이름>	id 속성을 기반으로 선택합니다.

선택자들의 관계를 지정하는 서식

서식	설명
<선택자>, <선택자>	쉼표로 구분된 여러 개의 선택자를 모두 선택합니다.
<선택자> <선택자>	앞 선택자의 후손 중 뒤 선택자에 해당하는 것을 모두 선택합니다.
<선택자> > <선택자>	앞 선택자의 자손 중 뒤 선택자에 해당하는 것을 모두 선택합니다.
<선택자> + <선택자>	같은 계층에서 바로 뒤에 있는 요소를 선택합니다.
<선택자1> ~ <선택자2>	선택자1부터 선택자2까지의 요소를 모두 선택합니다.

1-3_sel-dongju.py

1-3. CSS 선택자

선택자 기본 서식

서식	설명
<요소>[<속성>]	해당 속성을 가진 요소를 선택합니다.
<요소>[<속성>=<값>]	해당 속성의 값이 지정한 값과 같은 요소를 선택합니다.
<요소>[<속성>~=<값>]	해당 속성의 값이 지정한 값을 단어로 포함(띄어쓰기로 구분해서 완전히 포함)하고 있다면 선택합니다[8].
<요소>[<속성>|=<값>]	해당 속성의 값으로 시작하면 선택합니다(이때 하이픈 기호(※)로 구분해서 확인합니다).
<요소>[<속성>^=<값>]	해당 속성의 값이 지정한 값으로 시작하면 선택합니다.
<요소>[<속성>$=<값>]	해당 속성의 값이 지정한 값으로 끝나면 선택합니다.
<요소>[<속성>*=<값>]	해당 속성의 값이 지정한 값을 포함하고 있다면 선택합니다.

선택자들의 관계를 지정하는 서식

서식	설명
<요소>:root	루트 요소
<요소>:nth-child(n)	n번째 자식 요소

1-3_sel-dongju.py

1-3. CSS 선택자

선택자들의 관계를 지정하는 서식

BeatutifulSoup에서는 nth-of-typ(n) 서식만 지원함.

서식	설명
<요소>:nth-last-child(n)	뒤에서부터 n번째 자식 요소
<요소>:nth-of-type(n)	n번째 해당 종류의 요소
<요소>:first-child	첫 번째 자식 요소
<요소>:last-child	마지막 번째 자식 요소
<요소>:first-of-type	첫 번째 해당 종류의 요소
<요소>:last-of-type	마지막 번째 해당 종류의 요소
<요소>:only-child	자식으로 유일한 요소
<요소>:only-of-type	자식으로 유일한 종류의 요소
<요소>:empty	내용이 없는 요소
<요소>:lang(code)	특정 언어로 code를 지정한 요소
<요소>:not(s)	s 이외의 요소
<요소>:enabled	활성화된 UI 요소
<요소>:disabled	비활성화된 UI 요소
<요소>:checked	체크돼 있는 UI 요소를 선택합니다.

1-3_sel-dongju.py

CSS 선택자로 추출 연습하기

HTML.에서 특정 요소를 선택해서 추출하기 -> "Numbers 요소를 추출"

```
<ul id="bible">
  <li id="ge">Genesis</li>
  <li id="ex">Exodus</li>
  <li id="le">Leviticus</li>
  <li id="nu">Numbers</li>
  <li id="de">Deuteronomy</li>
</ul>
```

books.html

CSS 선택자로 추출 연습하기

```
from bs4 import BeautifulSoup
fp = open("books.html", encoding="utf-8")
soup = BeautifulSoup(fp, "html.parser")

# CSS 선택자로 검색하는 방법
sel = lambda q : print(soup.select_one(q).string)
sel("#nu")                          #(※1)
sel("li#nu")                         #(※2)
sel("ul > li#nu")                   #(※3)
sel("#bible #nu")                    #(※4)
sel("#bible > #nu")                 #(※5)
sel("ul#bible > li#nu")              #(※6)
sel("li[id='nu']")                  #(※7)
sel("li:nth-of-type(4)")             #(※8)

# 그 밖의 방법
print(soup.select("li")[3].string)    #(※9)
print(soup.find_all("li")[3].string)  #(※10)
```

0부터 세므로 0, 1,2,3 즉 4번째 요소를 선택해서 문자열을 출력합니다

1-3_sel-books.py

CSS 선택자로 과일과 야채 선택하기

HTML.에서 특정 요소를 선택해서 추출하기 -> "아보카도"를 추출

```
<html>
<body>
<div id="main-goods" role="page">
  <h1>과일과 야채</h1>
  <ul id="fr-list">
    <li class="red green" data-lo="ko">사과</li>
    <li class="purple" data-lo="us">포도</li>
    <li class="yellow" data-lo="us">레몬</li>
    <li class="yellow" data-lo="ko">오렌지</li>
  </ul>
```

fruits-vegetables.html

CSS 선택자로 과일과 야채 선택하기

HTML.에서 특정 요소를 선택해서 추출하기 -> "아보카도"를 추출

```
  <ul id="ve-list">
    <li class="white green" data-lo="ko">무</li>
    <li class="red green" data-lo="us">파프리카</li>
    <li class="black" data-lo="ko">가지</li>
    <li class="black" data-lo="us">아보카도</li>
    <li class="white" data-lo="cn">연근</li>
  </ul>
</div>
</body>
</html>
```

fruits-vegetables.html

CSS 선택자로 과일과 야채 선택하기

HTML.에서 특정 요소를 선택해서 추출하기 -> "아보카도"를 추출

```
  <ul id="ve-list">
    <li class="white green" data-lo="ko">무</li>
    <li class="red green" data-lo="us">파프리카</li>
    <li class="black" data-lo="ko">가지</li>
    <li class="black" data-lo="us">아보카도</li>
    <li class="white" data-lo="cn">연근</li>
  </ul>
</div>
</body>
</html>
```

fruits-vegetables.html

CSS 선택자로 과일과 야채 선택하기

HTML.에서 특정 요소를 선택해서 추출하기 -> "아보카도"를 추출

```
from bs4 import BeautifulSoup
fp = open("fruits-vegetables.html", encoding="utf-8")
soup = BeautifulSoup(fp, "html.parser")

# CSS 선택자로 추출하기
print(soup.select_one("li:nth-of-type(8)").string)                    #(※1)
print(soup.select_one("#ve-list > li:nth-of-type(4)").string)         #(※2)
print(soup.select("#ve-list > li[data-lo='us']")[1].string)           #(※3)
print(soup.select("#ve-list > li.black")[1].string)                   #(※4)
```

1) 모든 태그 중에서 8번째 요소를 선택

2) id가 ve-list인 요소 아래에 속한 li 태그에서 4번째 것을 선택

3) id가 ve-list인 요소 아래에 속한 data-lo='us'인 li 요소에서 0, 1 즉, 2번째 것을 선택

4) id가 ve-list인 요소 아래에 속한 클래스 black으로 지정된 li 요소에서 0, 1 즉, 2번째 것을 선택

1-3_sel-avocado.py

CSS 선택자로 과일과 야채 선택하기

HTML.에서 특정 요소를 선택해서 추출하기 -> "아보카도"를 추출

```
# find 메서드로 추출하기 ----                    (※5)
cond = {"data-lo":"us", "class":"black"}
print(soup.find("li", cond).string)

# find 메서드를 연속적으로 사용하기 ---          (※6)
print(soup.find(id="ve-list")
          .find("li", cond).string)
```

5) data-lo us이고class black인 li 태그를 찾는다

6) id가 ve-list 인 것을 찾고 이가운데 5)의 조건을 갖고 있는 태그를 찾는다.

1-3_sel-avocado.py

정규 표현식과 함께 조립하기

```
from bs4 import BeautifulSoup
import re # 정규 표현식을 사용할 때 --- (※1)
html = """
<ul>
  <li><a href="hoge.html">hoge</li>
  <li><a href="https://example.com/fuga">fuga*</li>
  <li><a href="https://example.com/foo">foo*</li>
  <li><a href="http://example.com/aaa">aaa</li>
</ul>
"""
soup = BeautifulSoup(html, "html.parser")
# 정규 표현식으로 href에서 https인 것 추출하기 --- (※2)
li = soup.find_all(href=re.compile(r"^https://"))
for e in li: print(e.attrs['href'])
```

2) regex 라이브러리를 포함시키고
정규식이 https:// 문자열을 시작으로 하는 li구문을
찾아서 href 속성값 출력

1-3_sel-avocado.py

1-4. 링크에 있는 것을 한꺼번에 내려받기

어떤 페이지에 있는 모든 이미지, 페이지 등을 한꺼번에 내려 받는 방법

CSS 선택자로 과일과 야채 선택하기

urljoin - URL을 기반으로 상대 경로를 정대경로로 변환

```
from urllib.parse import urljoin
base = "http://example.com/html/a.html"
print( urljoin(base, "b.html") )
print( urljoin(base, "sub/c.html") )
print( urljoin(base, "../index.html") )
print( urljoin(base, "../img/hoge.png") )
print( urljoin(base, "../css/hoge.css") )
```

1-4_cr-path.py

CSS 선택자로 과일과 야채 선택하기

http://, // 등으로 시작하는 path가 주어지면 base URL 정보를 무시하고 path 지정한 URL을 리턴

```
from urllib.parse import urljoin
base = "http://example.com/html/a.html"
print( urljoin(base, "/hoge.html") )
print( urljoin(base, "http://otherExample.com/wiki") )
print( urljoin(base, "//anotherExample.org/test") )
```

```
$ python3 cr-path.py
http://example.com/html/b.html
http://example.com/html/sub/c.html
http://example.com/index.html
http://example.com/img/hoge.png
http://example.com/css/hoge.css
```

1-4_cr-path2.py

재귀적으로 HTML페이지를 처리하는 방법

http://, // 등으로 시작하는 path가 주어지면 base URL 정보를 무시하고 path 지정한 URL을 리턴

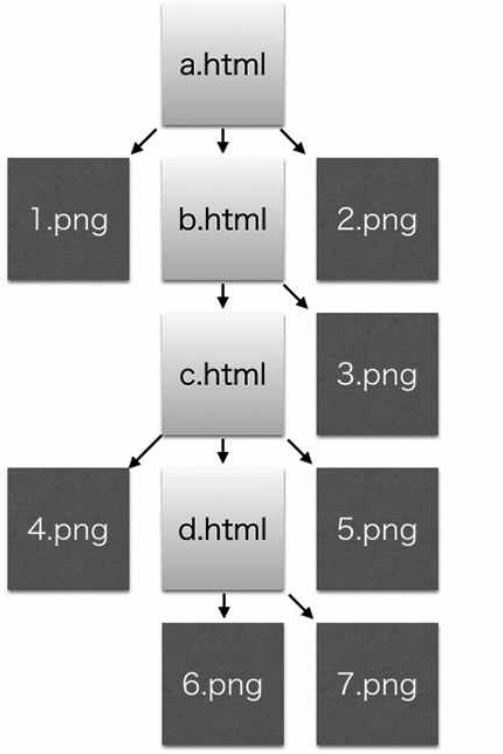
```
from urllib.parse import urljoin
base = "http://example.com/html/a.html"
print( urljoin(base, "/hoge.html") )
print( urljoin(base, "http://otherExample.com/wiki") )
print( urljoin(base, "//anotherExample.org/test") )
```

```
$ python cr-path2.py
http://example.com/hoge.html
http://otherExample.com/wiki
http://anotherExample.org/test
```

1-4_cr-path2.py

재귀적으로 HTML페이지를 처리하는 방법

- "a.html"에서 "b.html"로 링크 이동하고, "b.html"에서 "c.html"로 링크 이동하는 경우
- "a.html"에서 링크를 통해 이동하는 페이지를 모두 다운로드하고, "c.html"을 다운받지 않으면 중간에 링크가 잘리는 문제가 발생
- "a.html"을 분석하면 "b.html"도 함께 분석. 또한 "c.html"에서 "d.html"로 링크를 통해 이동하는 경우가 있다면 "c.html"도 분석해야 함
- HTML을 다운로드하고 싶다면 재귀적으로 HTML을 분석해야 함



1-4_cr-path2.py

재귀적으로 HTML페이지를 처리하는 방법

- 함수를 이용한 재귀 처리를 사용
- 재귀 처리는 프로그래밍 기법 중 하나로서 어떤 함수 내부에서 해당 함수 자신을 호출하는 것을 의미

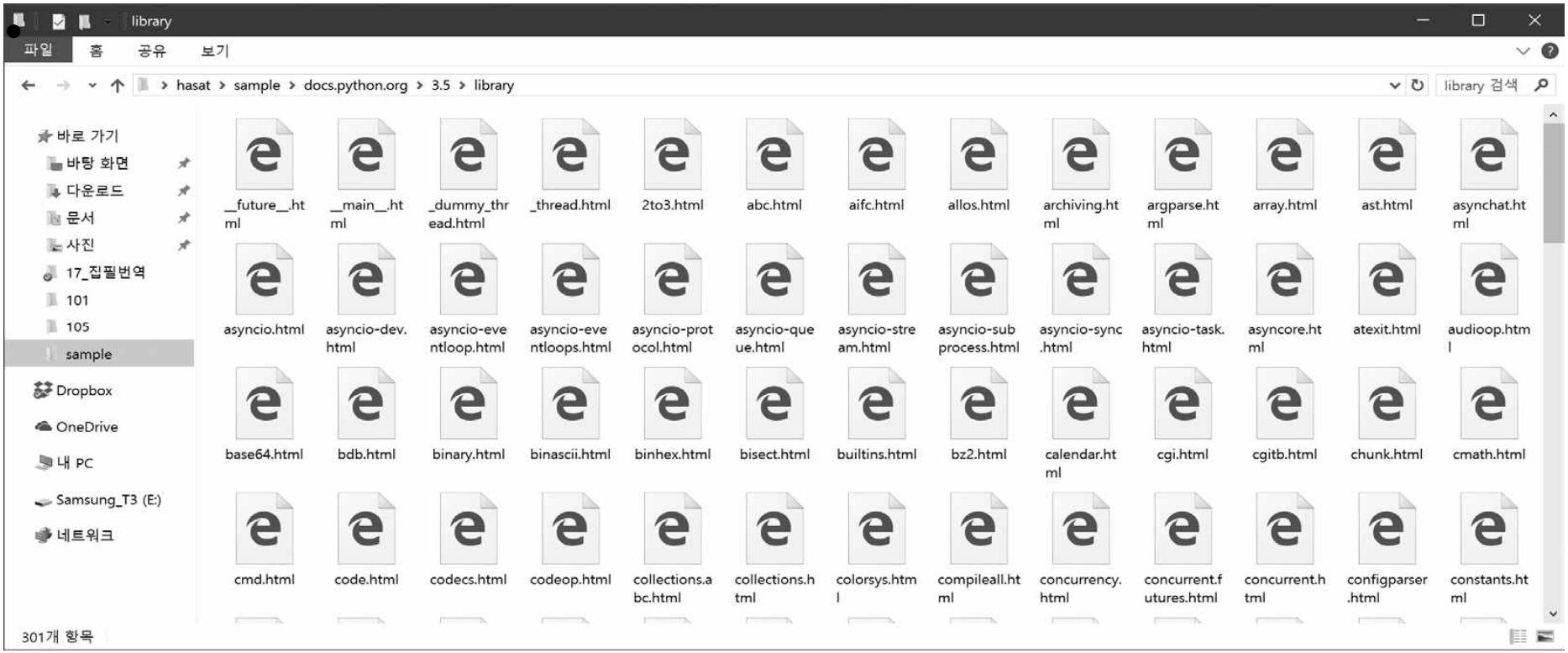
1. HTML을 분석
2. 링크를 추출
3. 각 링크 대상에 다음과 같은 처리를 진행
4. 파일을 다운로드
5. 파일이 HTML이라면 재귀적으로 1.로 돌아가서 순서를 처음부터 실행

1-4_cr-path2.py

1-4. 링크에 있는 것을 한꺼번에 내려받기

재귀적으로 HTML페이지를 처리하는 방법

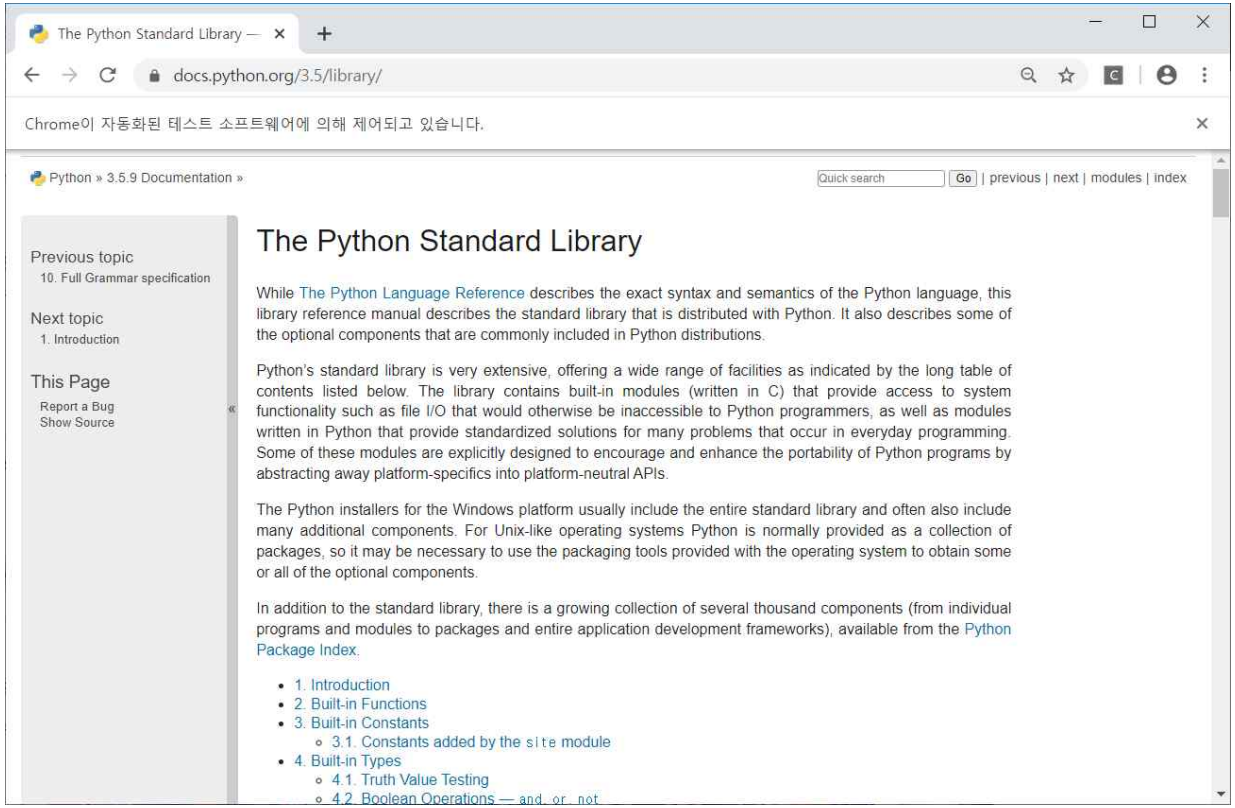
- 프로그램을 실행하면 다음과 같이 사이트 내부의 파일 또는 HTML 등을 모두 다운로드



1-4_cr-path2.py

재귀적으로 HTML페이지를 처리하는 방법

- 다운로드 받을 페이지는 파이썬 매뉴얼 페이지



1-4_cr-path2.py

재귀적으로 HTML페이지를 처리하는 방법

```
# 모듈 읽어 들이기 --- (※1)
from bs4 import BeautifulSoup
from urllib.request import *
from urllib.parse import *
from os import makedirs
import os.path, time, re
```

재귀적으로 HTML페이지를 처리하는 방법

```
# 이미 처리한 파일인지 확인하기 위한 변수 --- (※2)
proc_files = {}

# HTML 내부에 있는 링크를 추출하는 함수 --- (※3)
def enum_links(html, base):
    soup = BeautifulSoup(html, "html.parser")
    links = soup.select("link[rel='stylesheet']") # CSS
    links += soup.select("a[href]") # 링크
    result = []
    # href 속성을 추출하고, 링크를 절대 경로로 변환 --- (※4)
    for a in links:
        href = a.attrs['href']
        url = urljoin(base, href)
        result.append(url)
    return result
```

1-4_cr-path2.py

재귀적으로 HTML페이지를 처리하는 방법

```
# 파일을 다운받고 저장하는 함수 --- (※5)
def download_file(url):
    o = urlparse(url)
    savepath = "./" + o.netloc + o.path
    if re.search(r"/$", savepath): # 폴더라면 index.html
        savepath += "index.html"
    savedir = os.path.dirname(savepath)
    # 모두 다운됐는지 확인
    if os.path.exists(savepath): return savepath
    # 다운받을 폴더 생성
    if not os.path.exists(savedir):
        print("mkdir=", savedir)
        makedirs(savedir)
```

```
# 파일 다운받기 --- (※6)
  try:
      print("download=", url)
      urlretrieve(url, savepath)
      time.sleep(1) # 1초 휴식 --- (※7)
      return savepath
  except:
      print("다운 실패: ", url)
      return None
```

urlretrieve() 실제로 파일을 다운로드 하는 부분

1-4_cr-path2.py

재귀적으로 HTML페이지를 처리하는 방법

```
# HTML을 분석하고 다운받는 함수 --- (※8)
def analyze_html(url, root_url):
    savepath = download_file(url)
    if savepath is None: return
    if savepath in proc_files: return # 이미 처리됐다면 실행하지 않음 --- (※9)
    proc_files[savepath] = True
    print("analyze_html=", url)
    # 링크 추출 --- (※10)
    html = open(savepath, "r", encoding="utf-8").read()
    links = enum_links(html, url)
    for link_url in links:
        # 링크가 루트 이외의 경로를 나타낸다면 무시 --- (※11)
        if link_url.find(root_url) != 0:
            if not re.search(r".css$", link_url): continue
        # HTML이라면
        if re.search(r".(html|htm)$", link_url):
            # 재귀적으로 HTML 파일 분석하기
            analyze_html(link_url, root_url)
            continue
        # 기타 파일
        download_file(link_url)
```

1-4_cr-path2.py

재귀적으로 HTML페이지를 처리하는 방법

```
if __name__ == "__main__":
    # URL에 있는 모든 것 다운받기 --- (※12)
    url = "https://docs.python.org/3.5/library/"
    analyze_html(url, url)
```

1-4_cr-path2.py