크롤링과 스크레핑
데이터 다운로드하기

웹상의 정보를 추출하는 방법
파이썬은 웹 사이트에 있믐 데이터를 추출하기 위해 “urllib 라이브러리를 사용한다.
이 라이브러리를 이용하면 HTTP 또는 FTP를 사용해 데이터를 다운로드할 수 있다.

urllib은 URL을 다루는 모듈을 모아 놓은 패키지라고 할 수 있다.

그 중에서도 urllib.request모듈은 웹사이트에 있는 데이터에 접근라는 기능을 제공한다. 또한 인증, 리다렉트, 쿠키처럼 인터넷을 이용한 다양한 요청과 처리를 지원한다,

urllib.request를 이용한 다운로드
일단 웹사이트에서 파일을 다운로드하는 방법을 살펴보자.

파일을 다운로드 할 때는 urllib.request모듈에 있는 urlretrieve()함수를 사용한다.
이 함수를 이용하면 직접 파일을 다운로드할 수 있다.

```
# 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...!")
```

urlopen()으로 파일에 저장하는 방법
방금 살펴본 예제에서는 request.urlretrieve()함수를 사용해 파일에 곧 바로 저장했는데, 이번에는 request.urlopen()을 이용하여 데이터를 파이썬 메모리에 올릴 수 있다.

그럼 request.urlopen()을 이용해 메모리 위에 데이터를 올리고, 이후에 파일에 저장해 보자. 이 과정은 데이터를 추출하고 파일로 저장하는 흐름에 따라 진행된다.

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

웹 데이터 추출하기

웹에서 XML 또는 HTML 등의 텍스트 기반 데이터를 다운로드하는 방법을 보자

```
# 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)
```

매개변수를 추가해 요청을 전송하는 방법

이어서 URL에 매개변수를 추가해 요청을 전송하는 방법을 확인해 보자. 이번 예제에서는 기상청의 RSS 서비스를 사용해보자. 기상청 RSS는 URL에 지역번호를 지정하면 해당 지역의 정보를 제공해 준다.

기상청 RSS

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

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

파이썬으로 요청 전용 매개변수를 만들 때는 urllib.parse모듈의 urlencode()함수를 사용해 매개변수를 url인코딩한다.

```
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)
```

```
<rss version="2.0">
▼<channel>
  <title>기상청 육상 중기예보</title>
  <link>http://www.kma.go.kr/weather/forecast/mid-term_01.jsp</link>
  <description>기상청 날씨 웹서비스</description>
  <language>ko</language>
  <generator>기상청</generator>
  <pubDate>2021년 08월 10일 (화)요일 06:00</pubDate>
 ▼<item>
    <author>기상청</author>
    <category>육상중기예보</category>
    <title>전국 육상 중기예보 - 2021년 08월 10일 (화)요일 06:00 발표</title>
    <link>http://www.kma.go.kr/weather/forecast/mid-term_01.jsp</link>
    <guid>http://www.kma.go.kr/weather/forecast/mid-term_01.jsp</guid>
   ▼<description>
     ▼<header>
        <title>전국 육상중기예보</title>
        <tm>202108100600</tm>
       ▼<wf>
          <![CDATA[ ○ (강수) 이번 예보기간 동안 정체전선의 영향으로 남부지방과 제주도를 중심으로 비가 오는 날이 많겠습니다. <br
          /> 한편, 동풍의 영향을 받는 강원영동은 13일(금)부터 17일(화) 사이에 가끔 비가 오겠습니다. <br />○ (기온) 이번 예보기간
          아침 기온은 21~25도, 낮 기온은 27~32도로 어제(9일, 아침최저기온 21~24, 낮최고기온 27~34도)와 비슷하거나 조금 낮겠습니다.
          <br />○ (주말전망) 14일(토)은 전국(수도권 제외)에 비가 오겠고, 15일(일)은 강원영동과 전남권, 경상권, 제주도에 비가 오겠
          습니다. <br /> 아침 기온은 21~25도, 낮 기온은 27~31도가 되겠습니다.<br /><br />* 이번 예보기간에는 북태평양 고기압과 정
          체전선의 위치에 따라 강수의 변동성이 있겠으니, 앞으로 발표되는 기상정보를 참고하기 바랍니다. ]]>
        </wf>
      </header>
     ▼<body>
       ▼<location wl_ver="3">
          <province>서울 · 인천 · 경기도</province>
          <city>서울</city>
         ▼<data>
            <mode>A02</mode>
            <tmEf>2021-08-13 00:00</tmEf>
```

요청 형식 : url= http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=109
URL끝부분에 "?"를 입력하고, "<key>=<value>"형식으로 매개변수를 작성하면 된다.
여러개의 매개변수를 사용할 때는 "&"을 사용해 구분해준다.

매개변수를 명령 줄에서 지정하기
그런데 현재 프로그램은 매개변수를 코드에 입력해야 하므로 다른 지역의 정보를 알아내고 싶을 때는 프로그램을 열고 수정해야 한다. 이런 귀찮은 과정을 거치지 않고 명령줄에서 곧바로 지역 번호를 입력하고 사용하도록 해 보자.

```
#!/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) ###3 URL매개변수를 생성하기 위해 URL인코딩한다.
url = API + "?" + params
print("url=", url)
#### url= http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=108

# 다운로드합니다. --- (※3)
data = req.urlopen(url).read()
text = data.decode("utf-8")
print(text)
```

```
> python download-forecast-argv.py 108
> python download-forecast-argv.py 109
> python download-forecast-argv.py 105
```

BeautifulSoup로 스크레이핑하기
스크레이핑이란 웹 사이트에서 데이터를 추출하고, 원하는 정보를 추출하는 것이다.
최근에는 인터넷 데이터가 많으므로 스크레이핑을 잘하는 것이 중요하다.

파이썬으로 스크리에핑할 때 빼 놓을 수 없는 라이브러리가 바로 "BeautifulSoup"이다.
이 라이브러리를 이용하면 간단한 HTML과 XML에서 정보를 추출할 수 있다.
최근 스크레이핑 라이브러리는 다운로드 기능이 없다.

BeautifulSoup 설치
파이썬 라이브러리를 설치할 때는 pip명령어를 사용한다. pip란 파이썬 패키지를 관리 시스템이다.
파이썬 패키지는 Python Package Index(PyPI)에서 확인할 수 있다. pip를 이용하면 PyPI에 있는 패키지를 명령어 한줄로 설치할 수 있다.

pip으로 BeautifulSoup를 설치할 때는 다음과 같은 명령어를 실행한다,
> pip install beautifulsoup4

BeautifulSoup 기본 사용법
일단은 BeautifulSoup의 기본적인 사용법을 확인해봅시다. 다음 프로그램은 BeautifulSoup를 이용해 분석하는 간단한 예제입니다. 웹사이트로부터 HTML을 가져와서 사용하는 것이 아니라 HTML을 문자로 만들어 사용하고 있다,
그리고 문자 분석을 완료하면 결과를 출력한다.

```
# 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>
"""
# 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)
```

```
결과
h1 = 스크레이핑이란?
p  = 웹 페이지를 분석하는 것
p  = 원하는 부분을 추출하는 것
```

(※3)에서는 BeautifulSoup인스턴스를 생성한다, 이때 첫 번째 매개변수에 HTML을 지정한다. 그리고 두 번째 매개변수에는 분석할 분석기(parser)의 종류를 지정한다. HTML을 분석할 때는 “html.parser”라고 지정한다,
(※4)에서 원하는 부분을 추출한다. 정상적으로 분석했다면 HTML의 구조처럼 루트 요소인 <html>에서 마침표(.)를 사용해 값에 접근한 것이다,
(※5)에서는 string  속성에 잡근해서 요소의 글자부분을 추출한다.

분석할 때 HTML 내부에는 <p>태그가 2개 있는데  soup.html.body.p라고 접근하면 앞쪽에 있는 <p>태그를 추출하게 된다.
이때 첫 번째의 next_sibling에서는 <p>뒤에 줄바꿈 또는 공백이 추출된다.따라서 next_sibling을 한번 더 사용해 2번째 <p>태그를 추출한다.

HTML의 구조를 알고 있다면 쉽게 원하는 요소를 추출할 수 있다. 하지만 루트부터 “html.body…”형태로 HTML구조를 하나하나 적어 나가는 것은 조금 귀찮고 복잡하다.

id로 요소를 찾는 방법
BeautifulSoup는 루트부터 하나하나 요소를 찾는 방법말고도 id속성을 지정해서 요소를 찾는 find()메서드를 제공한다.,

```
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>
"""

# 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)
```

```
#title=스크레이핑이란?
#body=웹 페이지를 분석하는 것
```

여러개의 요소 추출하기 - find_all()메서드
참고로 여러개의 태그를 한 번에 추출하고 싶을 때는 find_all()메서드를 사용한다.
다음코드는 HTML내부에 있는 여러개의 <p>태그를 추출하는 프로그램이다.
<a> 태그는 하이퍼링크 태그이므로, 링크 대상은 href속성으로 지정하고 링크를 설명하는 텍스트는 태그 내부에 입력한다. 다음 코드는 설명 글자와 링크 대상 URL을 추출하는 예이다.

```
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>
"""
# 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)
```

```
naver > http://www.naver.com
daum > http://www.daum.net
```

(※2)에서는 find_all()메서드를 이용해 모든 <a>태그를 추출한다.
(※3)에서는 추출한 모든 요소를 for 구문으로 반복처리 한다. 링크의 href속성은 attrs["href"]처럼 attrs속성에서 추출한다, 또한 내부의 텍스트는 string속성으로 추출한다.

DOM요소의 속성에 대해
그럼 다시 DOM요소의 속성을 추출하는 방법을 확인해보자. 파이썬의 대화형 실행 환경인 REPL을 사용해 동작을 확인해보자. REPL을 실행하려면 명령줄에 "python"라고 입력한다.
<a> 태그라면 href등이 속성이다,

```
from bs4 import BeautifulSoup
soup = BeautifulSoup("<p><a href='a.html'>test</a></p> ",
        "html.parser")

#분석이 제대로 되었는지 확인 --- (※1)
print(soup.prettify())

# <a>태그를 변수 a에 할당
a = soup.p.a

# attrs속성의 자료형 확인  --- (※2)
print(type(a.attrs))

# href 속성이 있는지 확인
print ("href" in a.attrs)
```

```
#href속성값 확인
print(a['href'])
```

(※1)처럼 pretify메서드를 이용하면 제대로 분석되었는지 확인할 수 있다.

(※2)처럼 attrs속성의 자료형을 확인하면 딕셔너리(dict)라는 것을 알 수 있다. 따라서 in연산자를 사용해 원하는 속성이 존재하는지 확인할 수 있다.

urlopen()과 BeautifulSoup 조합하기

BeautifulSoup 인스턴스를 생성하는 방법을 배웠다.,

지금까지 살펴본 예제처럼 HTML문자열을 지정할 수도 있지만 open()함수 또는 urllib.urlopen()함수의 리턴 값을 지정해도 된다.

그럼 urlopen()을 사용해 기상청  RSS에서 특정 내용을 추출해보자.

```
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)
```

```
기상청 육상 중기예보
o (강수) 이번 예보기간 동안 기압골의 영향으로 남부지방과 제주도를 중심으로 비가 오는 날이
많겠습니다.  <br />          한편, 동풍의 영향을 받는 강원영동은 14일(토)부터 17일(화) 사이에 가끔
비가 오겠습니다.          <br />o (기온) 이번 예보기간 아침 기온은 21~25도, 낮 기온은 27~32도로 오늘(
10일, 아침최저기온 21~24, 낮최고기온 27~34도)과 비슷하거나 조금 낮겠습니다.<br />o (주말전망) 14일(
토)은 전국(수도권 제외)에 비가 오겠고, 15일(일)은 강원영동과 전남권, 경상권, 제주도에 비가
오겠습니다. <br />              아침 기온은 21~25도, 낮 기온은 27~32도가 되겠습니다.<br /><br />*
이번 예보기간에는 북태평양고기압과 기압골 위치에 따라 강수의 변동성이 있겠으니, 앞으로 발표되는
기상정보를 참고하기 바랍니다.
```

CSS 선택자 사용하기

BeautifulSoupsms 자바스크립트 라이브러리인 jQuery처럼 css선택자를 지정해서 원하는 요소를 추출하는 기능도 제공한다,

다음은  HTML에서 <h1> 태그와 <li>태그를 추출해 출력하는 코드

```
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>
"""
# HTML 분석하기 --- (※2)
soup = BeautifulSoup(html, 'html.parser')
# 필요한 부분을 CSS 쿼리로 추출하기
# 타이틀 부분 추출하기 --- (※3)
h1 = soup.select_one("div#meigen > h1").string
print("h1 =", h1)
# 목록 부분 추출하기 --- (※4)
li_list = soup.select("div#meigen > ul.items > li")
for li in li_list:
  print("li =", li.string)
```

```
h1 = Rhee 도서
li = 유니티 게임 이펙트 입문
li = 스위프트로 시작하는 아이폰 앱 개발 교과서
li = 모던 웹사이트 디자인의 정석
```

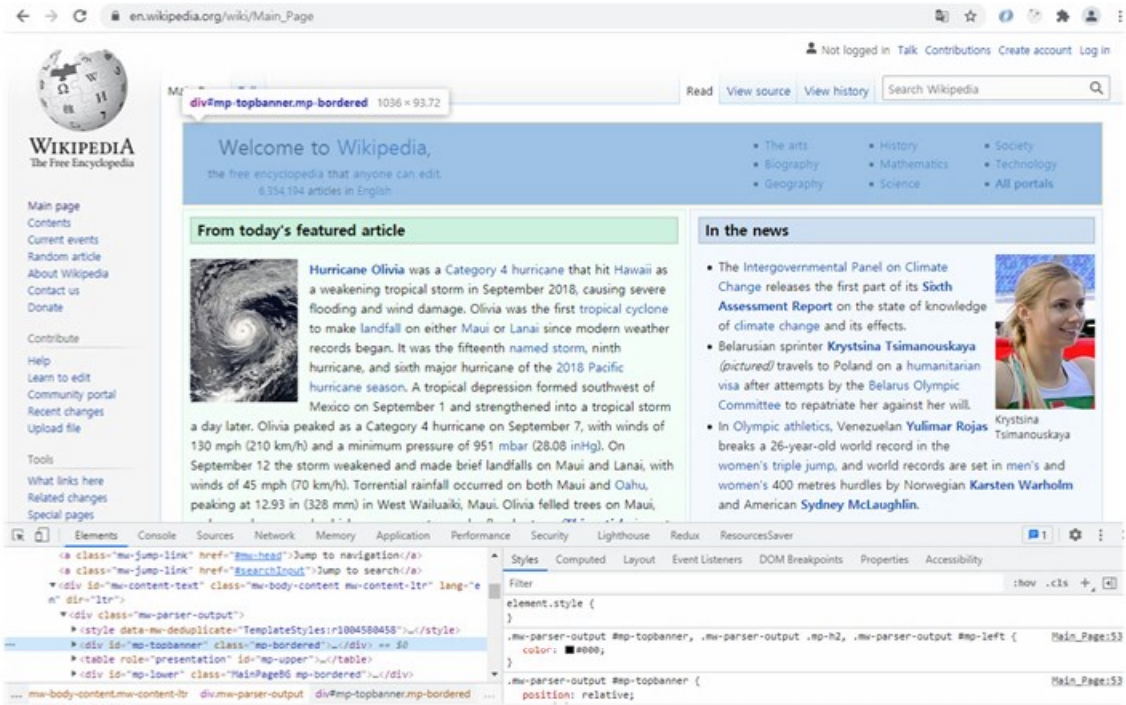
프로그램을 확인해보자,
프로그램의 (※1)에서는 분석 대상 HTML을 지정한다,
(※2)에서는 BeautifulSoup인스턴스를 생성한다, 이때 내부적으로 분석처리가 이뤄진다.
프로그램의 (※3)에서는 select_one()메서드를 사용해 <h1>태그를 추출한다. 이 예에서는 <h1>태그가 하나밖에 없으므로 h1이라고만 지정해도 상관없다. 실제로 웹 사이트에서 추출한 HTML에서 원하는 요소를 선택하려면 css선택자를 조금 더 자세히 지정해야 한다.

그리고 프로그램의 (※4) 에서는 select()메서드를 사용해 여러개의 <h1>태그를 추출한다, 그리고 추출한 요소는 리스트 자료형이므로 for구문을 사용해 하나씩 확인한다.

네이버 금융에서 환불 정보 추출하기
네이버 금융지표페이지 : http://finance.naver.com/marketindex/

```
from bs4 import BeautifulSoup
import urllib.request as req
# HTML 가져오기
url = "http://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)
```

```
usd/krw = 1,150.50
```




css 선택자

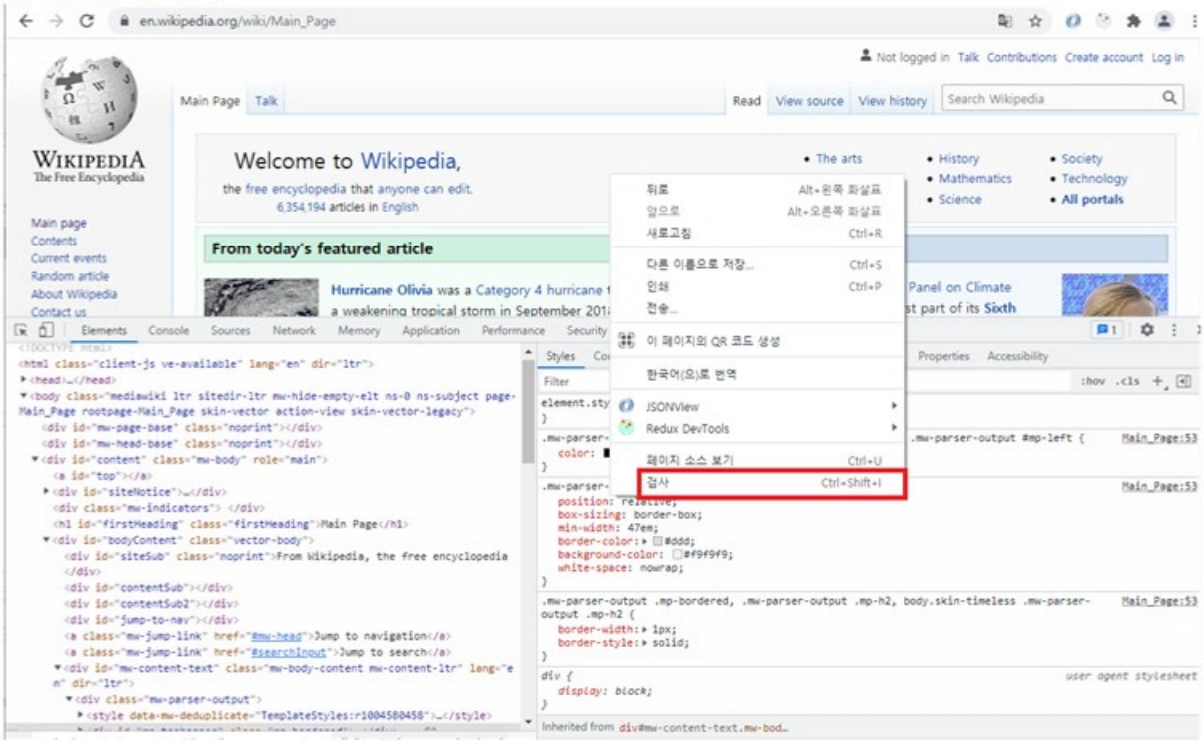
웹 브아우저로 HTML 구조 확인하기

HTML 구조를 확인할 때는 웹브라우저가 제공하는 개발도구를 사용하는 것이 좋다.

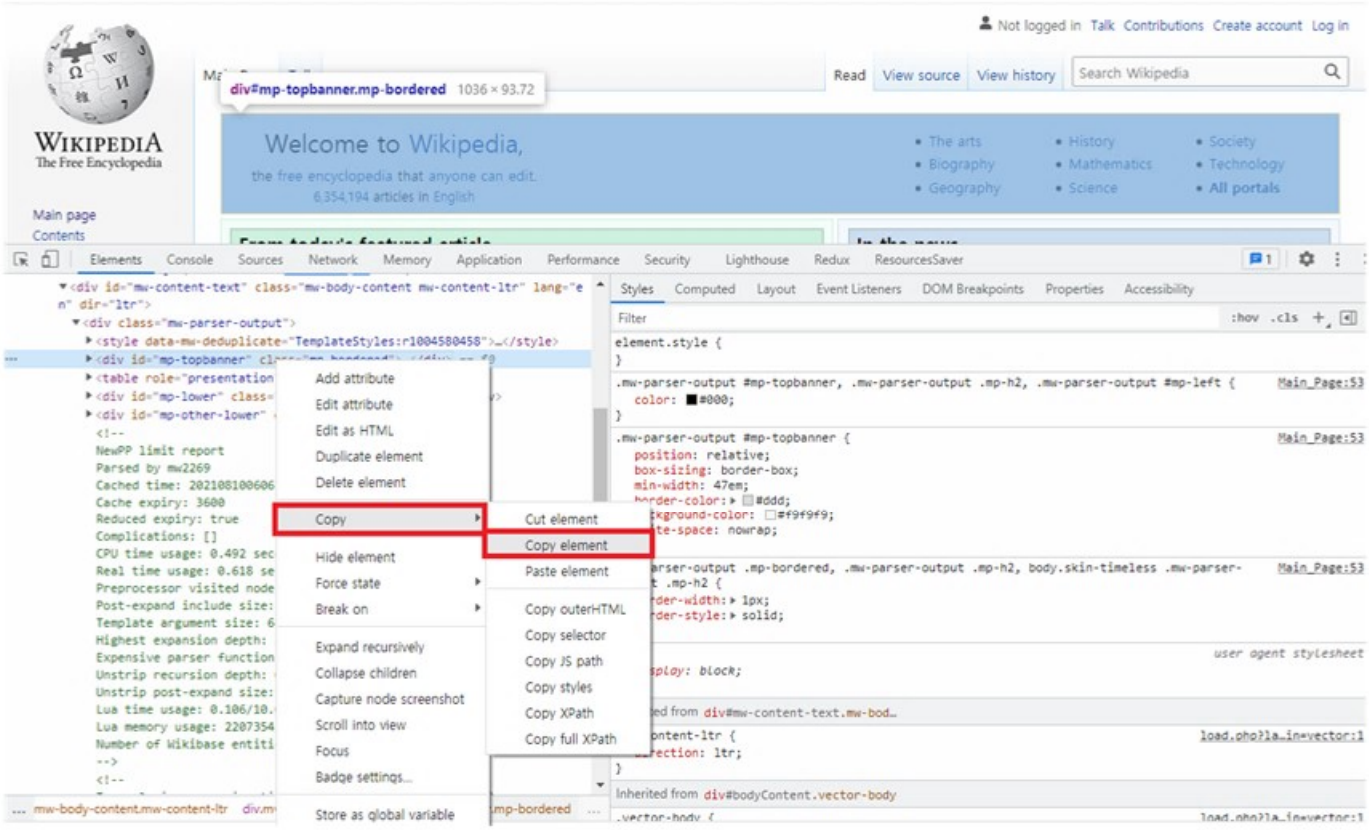
구글 크롬에서 분석하고 싶은 웹페이지 위에 마우스 오른쪽 버튼을 클릭한면 컨텍스트 메뉴가 나타난다.
여기서 검사를 선택하면 개발자 도구가 열린다.

원하는 요소 선택하기

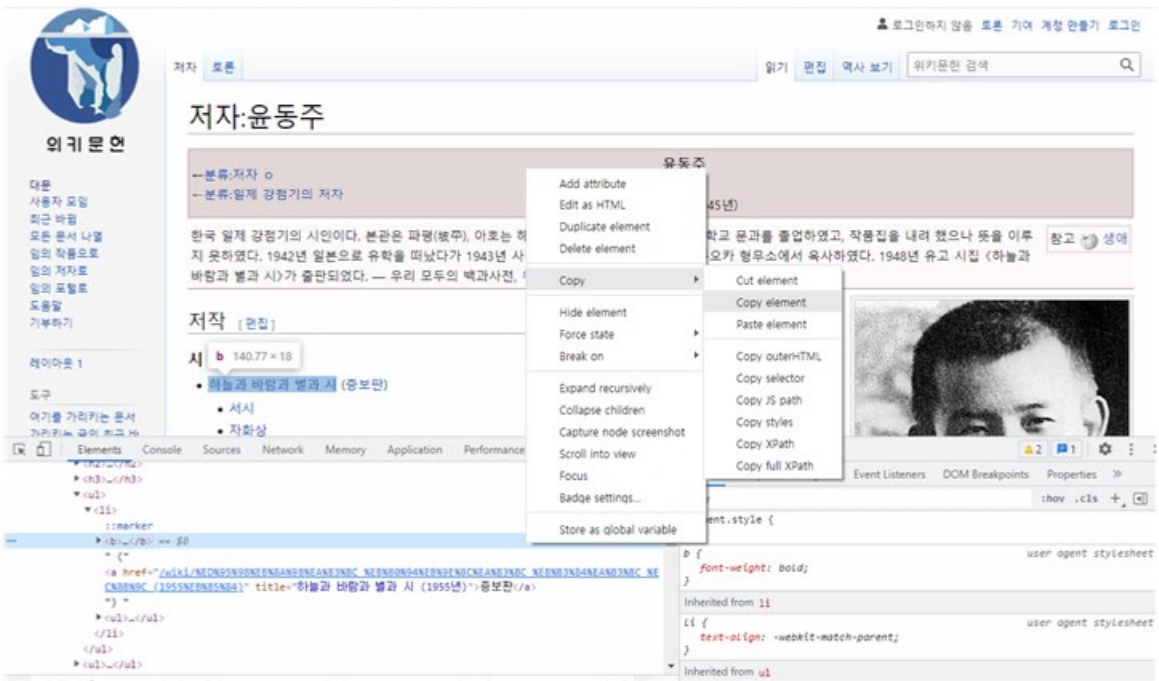
웹 브라우저에서 개발자 도구를 열었다면 개발자 도구 왼쪽 위에 있는 요소 선택 아이콘을 클릭하고 페이지에서 조사하고 싶은 요소를 클릭한다. 이렇게 하면 개발자 도구에 표시되는 요소구조 표시 부분에서 해당 DOM 요소를 확인할 수 있다.



개발자 도구에서 HTML구조를 확인했으면 태그를 선택한 상태로 마우스 오른쪽 버튼을 클릭한다.
그리고 팝업 메뉴에서 [ Copy > Copy selector ]를 클릭하면 요소의 CSS 선택자가 클립보드에 복사된다.
이 기능을 스크레이핑할 때 굉장히 유용한 기능이다.




위키 문헌에 공개돼 있는 윤동주 작가의 작품 목록 가져오기
그럼 방금 설명한 방법을 사용한 예로 위키 문헌( https://ko.wikisource.org/wiki/)에 공개돼 있는 윤동주 작가의 작품 목록을 프로그램을 통해 가져와보자.
일단 윤동주 작가 페이지를 웹브라우저에서 연다.
그리고 마우스 오른쪽 버튼을 클릭하고 [검사]를 눌러 개발자 도구를 띄운다.



참고로 “하늘과 바람과 별과 시”라는 것은 작품집인데, 위키문헌 사이트의 구조가 단순하기 때문에 선택자만으로는 작품을 구분할 수 없다. “ul태그 내부에 또는 ul태그가 있다면 작품집”이라는 형태로 구분해야 하는데 간단한 연습이므로 작품집과 직접 구분없이 모두 가져와 보겠다.

다음과 같이 css선택자가 복사된다.
#mw-content-text > ul:nth-child(7) > ul > li a
ul:nth-child(n)이라는 것은 n번째에 있는 요소를 의미한다. 따라서 ul:nth-child(7)이라는 것은 7번째에 있는 태그라는 의미이다. 현재 페이지를 조금 더 분석하면 #mw-content-text 내부에 있는  ul태그는 모두 작품과 관련된 태그이다. 따라서 따로 구분할 필요 없으므로 생략해도 된다.
그러나 BeautifulSoup 는 nth-child를 지원하지 않는다.

```
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)
```

```
- 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
- 바람이 불어
```

css선택자를 자세히 알아보기
속성 선택자

| 속성 | 풀이 |
|---|---|
| h1[class] | class 속성이 있는 모든 h1 |
| img[alt] | alt속성이 있는 모든 이미지 |
| *[title] | title 속성을 갖는 모든 요소 |

가상 클래스 선택자

| 속성 | 풀이 |
|---|---|
| a:link | 링크된 글자를 보고만 있을 때의 스타일을 주는 선택자 |
| a:visited | 링크된 글자를 눌러 해당 페이지에 갔다가 돌아온 경우의 스타일을 주는 선택자 |
| a:hover | 링크 걸린 글자에 마우스가 닿았을 경우의 스타일을 주는 선택자 |
| a:active | 링크 걸린 글자가 활성화되었을 경우의 스타일을 주는 선택자. (클릭했다가 돌아왔거나 클릭하려다 만 경우) |
| a:focus | 링크 걸린 글자에 포커스가 생길 경우의 스타일을 주는 선택자.(탭키 등으로 포커스가 나타날 경우) |

수도 클래스 선택자

| 속성 | 풀이 |
|---|---|
| :first-letter | 요소의 첫 글자 |
| :first-line | 요소의 첫 줄 |
| :first-child | 같은 요소 중 첫 번째 요소 |
| :last-child | 같은 요소 중 마지막 번째 요소 |
| :before | 요소의 맨 앞에 배치하는 마크업에는 없는 가상 요소 |
| :after | 요소의 맨 뒤에 배치하는 마크업에는 없는 가상 요소 |

css 선택자로 추출 연습하기

그럼 css선택자를 사용해 실제로 HTML코드를 보고, 특정 요소를 선택해서 추출하는 프로그램을 생각해본다.

books.html

```
<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>
```

이러한 HTML에서 Numbers요소를 추출하는 경우를 생각해 보자.

```
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)
```

```
Numbers
Numbers
Numbers
Numbers
Numbers
Numbers
Numbers
Numbers
Numbers
Numbers
```

(※1)방법은 가장 기본적인 방법으로 id 속성이 nu인 것을 추출한다.
(※2)방법은 거기에 <li>태그라는 것을 추가로 지정한 것이다.
(※3)방법은 <ul>태그의 자식이라는 것을 추가한 것이다.
(※4)방법은 id속성을 사용해 #bible아래의 #nu를 선택하게 한 것이다.
(※5)방법은 (※4)방법에서 태그들이 직접작인 부모 자식관계를 가지고 있다는 것을 나타낸 것이다,
(※6)방법은 id가 bible인 <ul>태그 바로 아래에 있는 id가 nu인 <li>태그를 선택한 것이다.
(※7)방법은 속성 검색을 사용해 id가 nu 인<li> 태그를 지정하는 것이다.
(※8)방법은 4번째 <li>태그를 추출하는 것이다.
(※9)와 (※10)방법은 각각 select()와 find_all()메서드를 사용한다. 두가지 모두 <li>태그를 모두 추출한다.

css 선택자로 과일과 야채 선택해보기

fruits-vegetables.html

```
<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>
  <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>
```

이중에 “아보카도”를 추출하고 싶다면 어떤 선택자를 사용해야 하나?

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

```
# CSS 선택자로 추출하기
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)
# 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)
```

```
아보카도
아보카도
아보카도
아보카도
아보카도
```

(※2) 방법은 야채를 나타내는 id가 ve-list인 요소바로 아래에 있는 <li>태그 중에 4번째 요소를 추출한 것이다.

(※3) 방법은 select()메서드를 사용해 id가 ve-list인 요소 바로 아래에 있는 <li>태그 중에서 data-lo속성이 “us"인 것을 모두 추출하고, 그중에서 [1]인 요소(0부터 세므로 2번째 요소)를 선택하는 것이다.

현재 html에서는 data-li가 “us"인 것은 파프리카와 아보카도 뿐이기 때문에 이런 방법을 사용할 수 있는 것이다.

(※4) 방법도 select()메서드를 사용해 class속성이 “black"인 요소 가운데[1]의 요소를 선택하는 것이다.

class 속성이 "black"인 것은 가지와 아보카도뿐이다.

(※5) 방법은 find()메서드를 사용하는 것이다. find()메서드는 객체를 사용해 여러개의 조건을 한번에 지정할 수 있다는 것이 특징이다. 현재 코드에서는 data-lo속성이 “us", 속성이 ” black"인 것이 특장이다.

(※6) 방법은 find()메서드를 두 번 조합해서 사용하는 방법이다. find()메서드를 연속으로 사용하면 이전에 추출한 요소에 추가적인 조건을 지정할 수 있다. 따라서 id가 “ve-list”인 요소를 추출하고 거기서 “li"태그이며 특정 조건을 만족하는 것을 추출한다.

링크에 있는 것을 한꺼번에 내려 받기

한꺼번에 다운 받는 데 필요한 처리 내용
일단 <a>태그의 링크 대상이 상대 경로일 수 있다. 그래서 릹크 대상이 HTML일 경우, 해당 HTML의 내용에 추가적인 처리를 해야 한다. 그리고 링크를 재귀적으로 다운 받아야 한다. 이번 절에서는 링크에 있는 것을 한꺼번에 다운 받는 기법을 알아보자.

상대경로를 전개하는 방법
그럼 일단 첫 번째 문제부터 살펴본다.
일단<a> 태그의 href 속성에 링크 대상이 "../img/hoge.png"처럼 상대 경로로 적혀 있다고 하자.
<a>태그가 상대 경로로 주어졌을 때 대상에 있는 것을 다운받으려면 상대경로를 절대 경로로 변환해야 한다.
상대 경로를 전개할 때는 urllib.parse.urljoin을 사용한다. 실제로 프로그램을 확인해보자.

```
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") )
```

```
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
```

결과를 보면 기본 URL을 기반으로 상대 경로를 절대 경로로 변환한다는 것을 알 수 있다, 이터럼 상대 경로를 절대 경로로 변환하는 urljoin()함수의 사용법을 살펴 본다.

urljoin(base, path)
이 함수는 첫 번째 매개변수로 기본 URL, 두번째 매개변수로 상대 경로를 지정한다.
만약 상대 경로(path 매개변수)가 http:..등으로 시작한다면 기본  URL(base 매개변수)을 무시하고, 두 번째 매개변수에 지정한 URL을 리턴한다.

```
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") )
```

```
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
```

재귀적으로 HTML 페이지를 처리하는 방법
"a.html"에서 "c.html"로 링크 이동라고 "b/.html"에서 "c.html"로 링크를 타고 이동하는 경우를 생각해보자.
이때 "a.html"에서 링크를 통해  이동하는 페이지를 모두 다운로드하고, "c.html"을 다운받지 않으면 중간에 링크가 잘리는 문제가 발생한다, 따라서 "a.html"을 분석하면 "b.html"도 함께 분석해야 한다. 또한 "c.html"에서

“d.html"로 링크를 통해 이동하는 경우가 있다면 "c.html"도 분석해야 한다, 따라서 HTML을 다운로드하고 싶다면 재귀적으로 HTML을 분석해야 한다.

이러한 구조의 데이터를 처리하면 함수를 이용한 재귀 처리를 사용하면 된다. 재귀처리는 프로그래밍 기법 중 하나로서 어떤 함수 내부에서 해당 함수 자신을 호출하는 것을 의미한다.
이러한 기법을 이용하면 연결된 모든 HTML페이지를 다운받을 수 있다.

순서를 정리하면 다음과 같다.
(1) HTML을 분석한다.
(2) 링크를 추출한다.
(3) 각 링크 대상에 다음과 같은 처리를 한다.
(4) 파일을 다운받는다.
(5) 파일이 HTML이라면 재귀적으로 (1) 로 돌아가서 순서를 처음부터 실행한다.

모든 페이지를 한꺼번에 다운 받는 프로그램
일단 이번 절에서는 웹에 있는 파이썬 문서중에서 library폴더 아래에 있는 모든 것을 다운 받아보자

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

(※1)에서는 필요한 모듈을 읽어 들인다.
인터넷에서 데이터를 내려 받기 위해 urllib.request, URL분석을 위한 urllib.parse, 폴더 생성을 위한 os, 경로와 관련된 os.path, 스립을 위한 time, 정규표현식을 위한 re모듈을 읽어 들인다.
(※2)에서는 전역 변수proc_files를 초기화한다. 이는 이미 분석한 HTML파일인지 판별하기 위한 변수이다.
HTML링크구조는 a.html에서 b.html로 이동하는 링크가 있으면 b.html에서 a.html로 이동하는 변수가 있을 수

있다. 이때 따로 처리를 하지 않으면 무한 루프에 빠져서 처리가 종료되지 않는다. 따라서 이러한 변수를 사용해 HTML에 두 번 이상 처리를 반복하지 않게 만들어야 한다.
(※3)의 emul_links()함수에서는 HTML을 분석하고, 링크를 추출한다. <a> 태그로 링크,<link>태그로 스타일시트의 경로를 찾는데 두가지 모두 BeautifulSoup의 select() 메서드를 사용했다.
(※4)에서는 링크 태그의 HREF속성에 적혀 있는url을 추출하고, 절대 경로로 변환한다.
(※5)부터 인터넷에 있는 파일을 다운 받는 부분이다. URL을 기반으로 파일명을 결정하고, 필요하다면 폴더를 생서한다. 실제로 다운로드하는 부분은 (※6)이다.
여기서 파일을 다운로드할 때는 urlretrieve()함수를 사용한다.,
(※7)에서 일시적으로 처리를 멈추는 기능을 제공하는데, 이는 파일을 다운로드하는 웹 서버에 부하를 주지않기 위한 예의라고 할 수 있다.
(※8)의 analyze_html()함수에는 HTML파일을 분석하고, 링크에 있는 것을 다운 받는다.
(※9)는 같은 파일에 처리를 반복하지 않게 확인하는 부분이다.
(※2)에서 선언했던 변수를 사용한다.
프로그램의 (※10)에서는 링크를 추출한다. 이 프로그램에서는 (※6)에서 본 것처럼 urlretrive()함수로 다운받고, 이미 다운받은 파일을 읽어 들이는 처리를 진행한다.
프로그램의 (※11)부분에서는 링크 대상을 확인해서 링크가 해당 사이트가 아닌 경우 다운로드하지 않게 만든다. css파일을 다운로드하지 않으면 레이아웃이 깨질 수 있으므로 css파일은 예외적으로 다운로드하게 조건문을 설정했다.
또한 프로그램의 (※12)에서는 어떤 사이트를 다운로드할지 지정한다,
__name__에는 모듈 이름이 들어오게 되는데, 모듈이 아닌 경우에 __main__이 들어온다. 따라서 이 스크립트를 직접 적으로 실행하는 경우에만 __name__에 __main__이 들어와서 처리를 진행한다.