Web Scraping

August 2020

Prepared by Prof. Youn-Sik Hong

복제/수정/배포 금지, All rights reserved by Youn-Sik Hong, 2020

목차

Urllib : download-url. ipynb

- **urllib: URL과 관련 있는 모듈들을 모아놓은 python 패키지**
  - URL: Uniform Resource Locator
  - '**http**://www.python.org', '**ftp**://...'

통신 프로토콜(protocol)

- **urllib 패키지에 포함된 라이브러리**
  - **urllib.request**, **urllib.parse**, **urllib.error** 등

```
from urllib.parse import urlparse
url2 = 'https://en.wikipedia.org/wiki/Python_(programming_language)'
p = urlparse(url2)
```

```
print(p.scheme)
print(p.geturl())
```

웹 페이지 다운로드 및 저장

urllib.request.urlopen

```
import urllib.request as req

url = 'http://www.python.org'

f = req.urlopen(url)
f.read(300)
```

```
with req.urlopen(url) as f:
    print(f.read(300))
```

urllib.request.urlretrieve

```
req.urlretrieve(url, 'test.html')
```

test.html 파일에 저장

URL 한글 처리

url 을 지정할 때 한글이 포함되면

```
url_ko = 'https://ko.wikipedia.org/wiki/파이썬'
req.urlopen(url_ko)
```

```
UnicodeEncodeError: 'ascii' codec can't encode characters
```

문제 해결은 Practice #1에서!!

urllib.parse.quote, unquote 메소드

- string의 특수 문자(한글 문자, 공백문자 등)를 **%xx** 이스케이프 형태로 바꿈

```
import urllib.parse
urllib.parse.quote('파이썬')
```

```
'%ED%8C%8C%EC%9D%B4%EC%8D%AC'
```

```
urllib.parse.unquote('%ED%8C%8C%EC%9D%B4%EC%8D%AC')
```

```
'파이썬'
```

Practice #1

아래 내용에 맞게 코드를 작성하고 실행시켜 보시오.

- '위키피디아 파이썬'의 웹 페이지 url을 찾으시오.

  https://ko.wikipedia.org/wiki/%ED%8C%8C%EC%9D%B4%EC%8D%AC

- 위 이스케이프 문자는 무슨 문자일까? Hint : urllib.parse.unquote 사용

  %ED%8C%8C%EC%9D%B4%EC%8D%AC

- 'https://ko.wikipedia.org/wiki/파이썬'을 urllib.request를 사용하여 웹 페이지를 읽어오려면? Hint : 아래 코드 참조

```
url_prefix = 'https://ko.wikipedia.org/wiki/'
url_ko2 = '파이썬'
url_quo = urllib.parse.quote(url_ko2)
new_url = url_prefix + url_quo
print(new_url)
```

- 변환한 new_url을 사용하여 해당 웹 페이지를 읽어오자.

```
f2 = req.urlopen(new_url)
f2.read(300)
```

우리 동네 날씨 (1/2)

■ 기상청 날씨누리 RSS 서비스 사용

- https://www.weather.go.kr/weather/main.jsp > 생활과 산업 > 서비스 > 인터넷
- **RSS** (RDF Site Summary): 대부분 web-based RSS

RDF: Resource Description Framework

우리 동네 날씨 (2/2) : download-forecast. ipynb

■ 기상청 날씨누리 RSS 서비스 사용 : GET 방식으로 매개변수 전송

- URL 뒤에 "?" 입력
  - 매개변수 작성 : "<key>=<value>"
  - 매개변수가 여러 개일 경우 "&"로 구분
    - 매개변수는 **urlencode** 함수를 사용해 인코딩해야 함

```
import urllib.request
import urllib.parse

#API = "http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"
API = "http://www.kma.go.kr/wid/queryDFSRSS.jsp"
values = {                  # dictionary 자료형 : key-value 쌍
    'zone': '2818582000'
}
params = urllib.parse.urlencode(values)

url = API + "?" + params        # GET 형식으로 매개변수 전송
print("url = ", url)

data = urllib.request.urlopen(url).read()
text = data.decode("utf-8")
print(text)
```

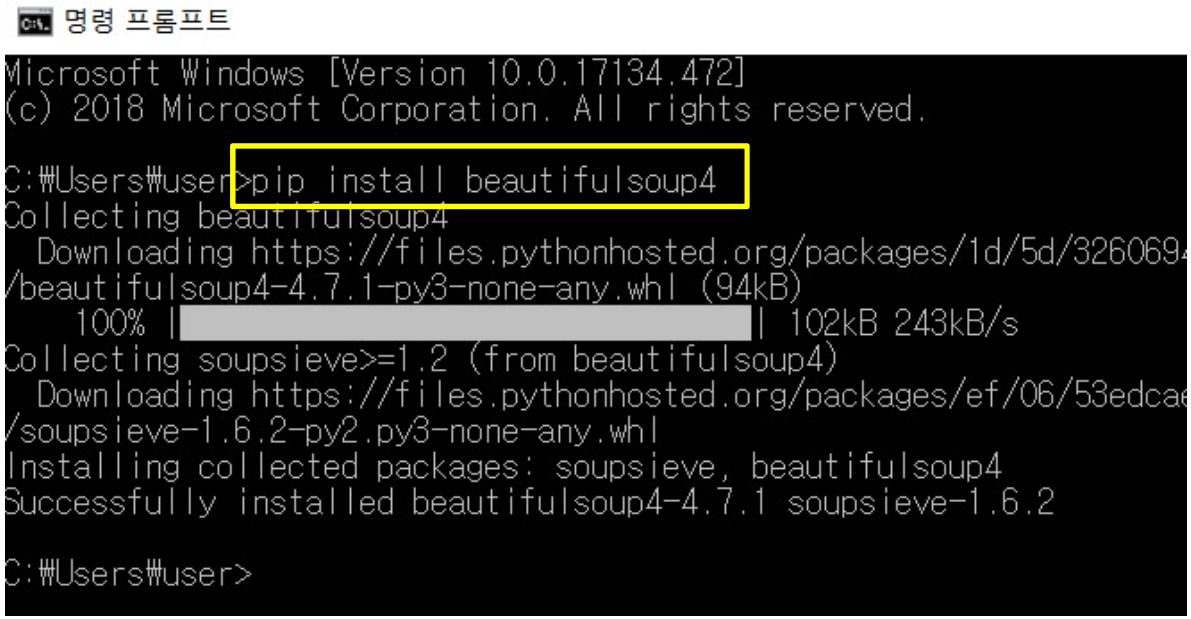
인천시 연수구 송도1동

Practice #2

- **아래 내용에 맞게 코드를 작성하고 실행시켜 보시오.**
  - 본인이 살고 있는 동네 날씨 정보를 구하시오.
  - 중기예보(전국) 정보를 가져오자.
    - **중기예보(전국)**
      http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=108
  - 장기예보(1개월 전망) 정보를 가져오자.
    - **장기예보(1개월 전망)**
      http://www.kma.go.kr/repositary/xml/fct/mon/img/fct_mon1rss_108_20200723.xml
  - RSS 요청 결과는 XML파일로 전달받는다. 아래 XML tag는 무슨 뜻인가?
    - <tmx>, <tmn>, <sky>, <pty>

BeautifulSoup : Web Scraping

- **scraping : 웹 페이지로부터 원하는 정보를 추출**
- **BeautifulSoup : Web scraping을 위해 필요한 라이브러리**
  - HTML 또는 XML로부터 필요한 정보를 쉽게 추출
    - BeautifulSoup 는 다운로드 기능은 없음.
  - 설치 : cmd 창에서 **pip**를 이용



BeautifulSoup : bs-test. ipynb

```
from bs4 import BeautifulSoup
html = '''
<html>
 <body>
  <h1>웹 스크레이핑이란?</h1>
  <p>웹 페이지 분석을 통해</p>
  <p>원하는 내용을 끄집어 냄</p>
 </body>
</html>
'''
soup = BeautifulSoup(html, 'html.parser')    # 파서 종류 지정

h1 = soup.html.body.h1

p1 = soup.html.body.p
p2 = p1.next_sibling
p22 = p2.next_sibling


print("h1 = " + h1.string)
print("p1 = " + p1.string)
print("p2 = " + p2.string)
print("p22 = " + p22.string)
```

p1의 next_sibling은 뒤에 있는 줄바꿈 문자

P2의 next_sibling은 2번째 태그

BeautifulSoup - id로 찾기

```
from bs4 import BeautifulSoup

html = '''
<html>
 <body>
  <h1 id="title">웹 스크레이핑이란?</h1>
  <p id="body">웹 페이지 분석을 통해</p>
  <p id="another">원하는 내용을 끄집어 냄</p>
 </body>
</html>
'''

soup = BeautifulSoup(html, 'html.parser')

title = soup.find(id="title")
body = soup.find(id="body")
body2 = soup.find(id="another")

print("#title = " + title.string)
print("#body = " + body.string)
print("#another = " + body2.string)
```

id를 지정해 요소를 추출

BeautifulSoup – 여러 개 요소 한꺼번에 찾기

```
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>
"""

soup = BeautifulSoup(html, 'html.parser')

links = soup.find_all("a")

for a in links:
    href = a.attrs['href']
    text = a.string
    print(text, ">", href)
```

find_all 메소드

속성(attribute)을 가져올 때

Practice #3

■ 아래 내용에 맞게 코드를 작성하고 실행시켜 보시오.

- 웹 페이지 내용이 다음과 같다.
  - 같은 id 속성 값을 갖는 <p> 태그가 2개 있다.

```
html = '''
<html>
 <body>
  <h1 id="title">웹 스크레이핑이란?</h1>
  <p id="body">웹 페이지 분석을 통해</p>
  <p id="body">원하는 내용을 끄집어 냄</p>
 </body>
</html>
'''
```

- find_all 메소드를 사용하여 <p>태그 중 id='body'인 내용을 모두 가져오도록 슬라이드 12의 예제 코드를 수정해 보자.

날씨 정보 추출 : download-forecast. ipynb

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

url='http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=108'

res = urllib.request.urlopen(url)

soup = BeautifulSoup(res, 'html.parser')

title = soup.find("title").string
wf = soup.find("wf").string
print(title)
print(wf)
```

```
기상청 육상 중기예보
o (강수) 30일(목)~8월 2일(일)은 중부지방(강원영동은 1~2일)에, 3일(월)은 서울.경기도와 강원영서에 비가 오겠습니다.<br />o (기온) 이번 예보기간의 낮 기온은 26~34도로 오늘(22~28도)보다 높겠고, 특히, 경북내륙을 중심으로 33도 이상 올라 덥겠습니다.<br />                8월 1일(토)부터 강원동해안을 중심으로 아침최저기온이 25도 이상으로 열대야가 나타나는 곳이 있겠습니다.<br />o (주말전망) 8월 1일(토)과 2일(일)은 중부지방에 비가 오겠으며, 아침 기온은 23~25도, 낮 기온은 27~34도의 분포를 보이겠습니다.<br /><br />* 31일(금)~8월 2일(일)은 정체전선의 영향으로 중부지방에 천둥.번개를 동반한 강한 비와 함께 많은 비가 올 가능성이 있으니, 앞으로 발표되는 기상정보를 참고하기 바랍니다.<br />* 30일(목)~31일(금) 남부내륙을 중심으로 대기가 불안정하여 소나기가 내릴 가능성이 있습니다.
```

Practice #4

■ 아래 내용에 맞게 코드를 작성하고 실행시켜 보시오.

- RSS 파일 내용(mid-term-rss.jsp?stnId=108)을 확인하시오.

```
<?xml version="1.0" encoding="utf-8" ?>
<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>2020년 07월 27일 (월)요일 18:00</pubDate>
 <item>
<author>기상청</author>
<category>육상중기예보</category>
<title>전국 육상 중기예보 - 2020년 07월 27일 (월)요일 18:00 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>
```

- 오른쪽 코드는 어느 <title>의 내용을 출력했는가?

```
title = soup.find("title").string
print(title)
```

- <language>, <category> 등으로 수정해서 실행시켜보자.

XML parsing 모듈 사용(1/2)

■ 슬라이드 15의 코드를 XML 파싱 모듈을 사용하는 코드로 변경

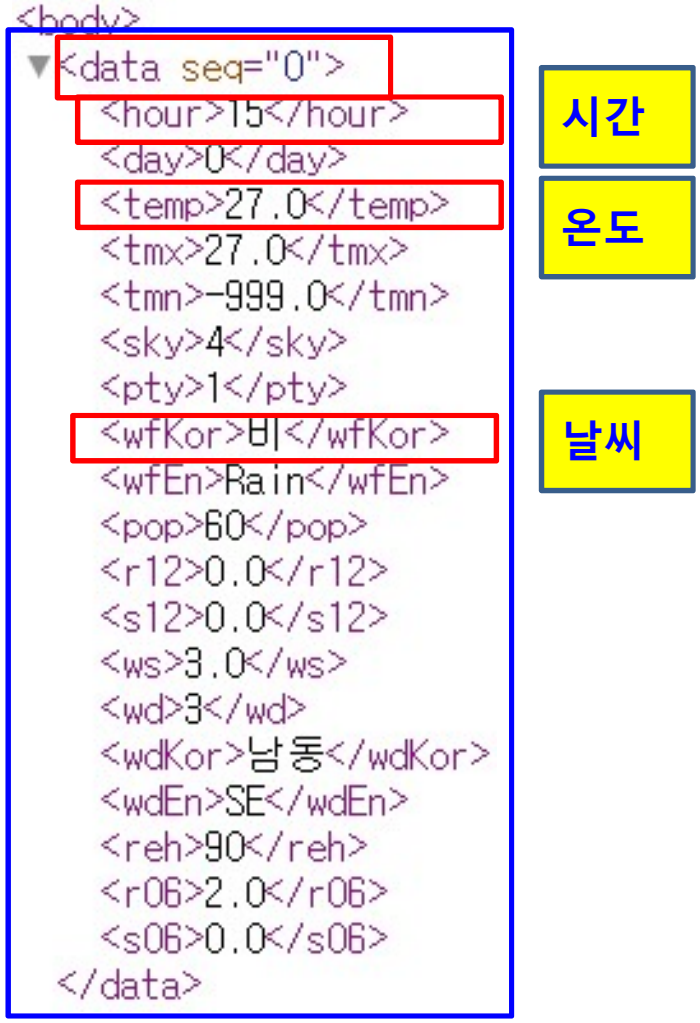
```
from xml.etree import ElementTree    # xml parsing module

url='http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=108'
res = urllib.request.urlopen(url)

tree = ElementTree.parse(res) # parsing the XML file
root = tree.getroot()         # get the root element

title = root.find('channel/title').text
print(title)

link = root.find("channel/link").text
language = root.find("channel/language").text
category = root.find("channel/item/category").text
title2 = root.find("channel/item/title").text
print(link)
print(language)
print(category)
print(title2)
```

어떤 title 내용이 출력될까?

XML parsing 모듈 (2/3)

■ 우리 동네 현재 날씨와 현재 온도 정보를 추출해 보자!

연수구
송도 1동



시간

온도

날씨

XML parsing 모듈 (3/3)

■ 우리 동네 현재 날씨와 현재 온도 정보를 추출해 보자!

```
#url = "http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=2818582000"
res2 = urllib.request.urlopen(url)

tree2 = ElementTree.parse(res2)
root2 = tree2.getroot()

title = root2.find("channel/item/title").text
print(title)

ref = "channel/item/description/body/data"

print('시간대별 날씨 예보')
last_index = len(root2.findall(ref))
for data in root2.findall(ref):
    hour = data.find("hour").text
    temp = data.find("temp").text
    wf = data.find("wfKor").text
    print("시간="+hour, ",₩t온도="+temp, ",₩t예보="+wf)
```

XML은 tag가 계층 구조로 정의

추출하려는 데이터가
어느 tag에 정의되어 있는지
tag 경로를 지정 하는 게 중요

목차

CSS Selector (1/3)

서식	설명
*	모든 원소를 선택
<원소 이름>	해당 원소 이름을 기준으로 선택
.<클래스 이름>	해당 클래스 이름을 기준으로 선택
#<id 이름>	해당 id 속성을 기준으로 선택

서식	설명
<원소>:root	root
<원소>:nth-child(n)	n번째 자식 원소
<원소>:nth-of-type(n)	n번째 해당 type의 원소
<원소>:first-child	첫번째 자식 원소
<원소>:last-child	마지막 자식 원소

CSS Selector 기초 (1/2) : css-select. ipynb

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(html, "html.parser")

sel = lambda q: print(soup.select_one(q).string)
pr = lambda x: print(x)

sel('#nu')
#sel('li#nu')
#sel('ul > li#nu')
#sel('ul li#nu')

sel('#bible > #nu')
#sel('#bible #nu')
#sel('ul#bible > li#nu')
#sel('ul#bible li#nu')

sel("li[id='nu']")
#sel("li:nth-of-type(4)")

#pr(soup.select("li"))
#pr(soup.select("li")[3])
pr(soup.select("li")[3].string)

#pr(soup.find_all("li"))
#pr(soup.find_all("li")[3])
pr(soup.find_all("li")[3].string)
```

```
html = """
<html><body>
<div id="meigen">
 <ul id="bible">
  <li id="ge">Genesis(창세기)</li>
  <li id="ex">Exodus(출애굽기)</li>
  <li id="le">Leviticus(레위기)</li>
  <li id="nu">Numbers(민수기)</li>
  <li id="de">Deuteronomy(신명기)</li>
 </ul>
</div>
</body></html>
"""
```

히브리어 성경 5권

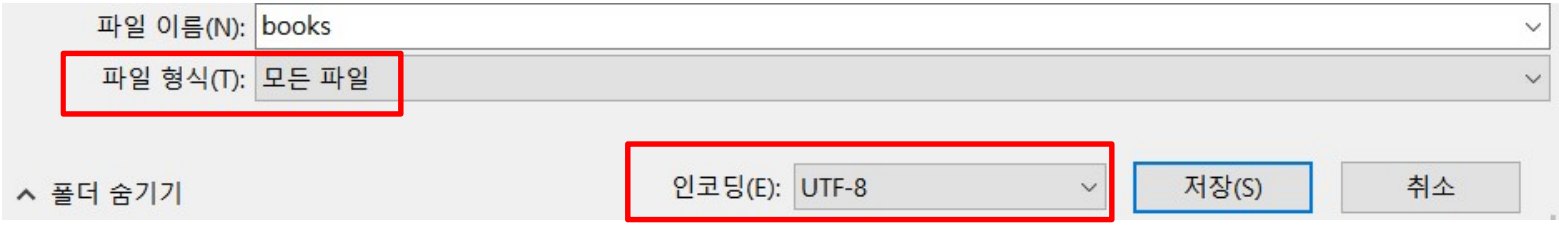
12가지 CSS selector 출력 결과는?

CSS Selector 기초 (2/2) : css-select. ipynb, books.html

```
<html><body>
<div id="meigen">
 <ul id="bible">
  <li id="ge">Genesis(창세기)</li>
  <li id="ex">Exodus(출애굽기)</li>
  <li id="le">Leviticus(레위기)</li>
  <li id="nu">Numbers(민수기)</li>
  <li id="de">Deuteronomy(신명기)</li>
 </ul>
 </div>
</body></html>
```

메모장 사용 파일 저장
파일 이름 : books.html
인코딩 : UTF-8

파일 저장 경로 –
ipnyb 파일과 같은 경로에 저장




```
fp = open('books.html', 'r', encoding='utf-8')
```

```
soup = BeautifulSoup(fp, "html.parser")
```

Practice #5

- **아래 내용에 맞게 코드를 작성하고 실행시켜 보시오.**
  - Genesis(창세기) 만을 출력
  - Exodus(출애굽기) 만을 출력
  - Leviticus(레위기) 만을 출력
  - Deuteronomy(신명기) 만을 출력
  - 히브리어 성경 5권을 순서대로 모두 출력
  - 히브리어 성경을 순서대로 앞 3권을 출력
  - 슬라이드 22의 코드를 수정.

Practice #6

■ 아래 내용에 맞게 코드를 작성하고 실행시켜 보시오.

- 예제 html 페이지가 다음과 같다.

```
html2 = """
<html>
<body>
<div id="nlpbooks">
  <h1>자연어처리 관련 도서</h1>
  <ul class="items">
    <li>머신러닝, 딥러닝 실전개발 입문</li>
    <li>Natural Language Processing in Action</li>
    <li>Applied Text Analysis with Python</li>
    <li>자연어처리 딥러닝 캠프</li>
  </ul>
</div>
</body>
</html>
"""
```

- 4권의 도서 제목을 출력하는 코드를 완성하시오.

```
soup2 =  BeautifulSoup(html2, 'html.parser')

tag = soup2.select_one("div#nlpbooks > h1")
pr(tag)

li_list = soup2.select(                                    )
for i, li in enumerate(li_list):
    print(i+1, '번째 도서=', li.string)
```

CSS Selector 고급(1/2) : css-select2. ipynb, vegetables.html

```
from bs4 import BeautifulSoup

fp = open("vegetables.html", encoding="utf-8")
soup = BeautifulSoup(fp, "html.parser")

check_null = lambda x: print(x.string) if x  else print('not found')

t = soup.select_one("li:nth-of-type(3)")
check_null(t)
t2 = soup.select_one("ul#fruits-list > li:nth-of-type(3)")
check_null(t2)
t3 = soup.select_one("ul#vegetable-list > li:nth-of-type(3)")
check_null(t3)
```

```
<html><body>
<div id="main-goods" role="page">
  <h1>여러 가지 과일과 채소</h1>
  <ul id="fruits-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="vegetable-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>
```

한 개의 list에만 적용 가능.
따라서 인덱스는 원소가 4개이므로
1,2,3,4 만 가능

2 개의 리스트 중 첫 번째 리스트
즉, fruits-list

CSS Selector 고급(2/2) : css-select2. ipynb, vegetables.html

어떤 값이 출력될까?

```
sel = lambda q : print(soup.select_one(q).string)
sel_1st = lambda q : print(soup.select(q)[0].string)
sel_2nd = lambda q : print(soup.select(q)[1].string)

sel("ul#fruits-list > li:nth-of-type(3)")
sel_1st("ul#fruits-list > li[data-lo='us']")
sel_2nd("ul#fruits-list > li[data-lo='us']")
sel_1st("#vegetable-list > li.black")
sel_2nd("#vegetable-list > li.black")

cond = {"data-lo":"us", "class":"black"}
cond2 = {"data-lo":"us", "class":"red green"}
print(soup.find("li", cond).string)
print(soup.find("li", cond2).string)

print(soup.find(id="vegetable-list")
          .find("li",cond).string)
```

```
<html><body>
<div id="main-goods" role="page">
  <h1>여러 가지 과일과 채소</h1>
  <ul id="fruits-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="vegetable-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>
```

Practice #7

■ 아래 내용에 맞게 코드를 작성하고 실행시켜 보시오

```
fp = open("sell-items.html", encoding="utf-8")
soup2 = BeautifulSoup(fp, "html.parser")

soup2.select('p')
soup2.select('.price')
soup2.select('span.price')
soup2.select('#fruits2')
soup2.select('p > span')
soup2.select('p span')
soup2.select('p.name > span.price')
soup2.select('h1 .name > span.store')
soup2.select("a[href='http://test1']")
soup2.select('p > span.price')[0].text

prices = soup2.select('p > span.price')
for price in prices:
    print(price.text)
soup2.select('a')[0]['href']
```

```
<html><body>
    <h1> 시장
        <p id='fruits1' class='name' title='바나나'> 바나나
            <span class = 'price'> 3000원 </span>
            <span class = 'inventory'> 500개 </span>
            <span class = 'store'> 가가가 </span>
            <a href = 'http://test1'> url1 </a>
        </p>

        <p id='fruits2' class='name' title='귤'> 귤
            <span class = 'price'> 2000원 </span>
            <span class = 'inventory'> 100개 </span>
            <span class = 'store'> 나나나</span>
            <a href = 'http://test2'> url2 </a>
        </p>
        <p id='fruits3' class='name' title='파인애플'> 파인애플
            <span class = 'price'> 5000원 </span>
            <span class = 'inventory'> 10개 </span>
            <span class = 'store'> 가가가</span>
            <a href = 'http://test1'> url1 </a>
        </p>
    </h1>
</body></html>
```

파일 이름 : sell-items.html

목차

Naver 금융에서 환율 정보 추출 (1/2)

■ 시장 지표 웹 사이트 : http://finance.naver.com/marketindex

■ Cursor를 "환전 고시 환율" ➔ 오른쪽 버튼 ➔ 페이지 소스 보기

```
<div class="head_info point_up">
    <span class="value">1,119.00</span>
    <span class="txt_krw"><span class="blind">원</span></span>
    <span class="change">6.00</span>
    <span class="blind">상승</span>
</div>
```

환율 정보 추출 (2/2) : exchange-rate. ipynb

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

url = "http://finance.naver.com/marketindex/"
res = req.urlopen(url)

soup = BeautifulSoup(res, "html.parser")

#prices = soup.select_one("div.head_info > span.value").string
#prices = soup.select("div.head_info > span.value")
prices = soup.select("span.value")

print("달러: " + prices[0].string)
print("엔: " + prices[1].string)
print("유로: " + prices[2].string)

for price in prices:
    print(price, "원")
```

시인 윤동주의 시 목록 가져오기 (1/3)

- **웹브라우저로 HTML 구조 확인**
  - Chrome → 오른쪽 버튼 → [검사]
- **구글에서 "저자 윤동주" 검색**
  - 개발자 도구 > "Elements" 탭
  - Elements 클릭 ➔ DOM 구조

시인 윤동주의 시 목록 가져오기 (2/3)

- **작품 목록에서 첫 번째 요소(“하늘과 바람과 별과 시” 클릭**
  - <b><a ... >하늘과 바람과 별과 시 ...
  - 태그 선택 > 마우스 오른쪽 버튼 > Copy > Copy Selector
    - CSS Selector를 클립보드에 복사
  - #mw-content-text > div > ul:nth-child(6) > li > b > a

```
▼<ul>
  ▼<li>
    ▼<b>
      <a href="/wiki/%ED%95%98%EB%8A%98%EA%B3%BC_%EB%B0%94%EB%9E%8C%EA%B3%BC_%EB%B3%84%EA%B3%BC_%EC%8B%9C" title="하늘과 바람과 별과 시">하늘과 바람과 별과 시</a>
    </b>
    " ("
    <a href="/wiki/%ED%95%98%EB%8A%98%EA%B3%BC_%EB%B0%94%EB%9E%8C%EA%B3%BC_%EB%B3%84%EA%B3%BC_%EC%8B%9C_(1955%EB%85%84)" title="하늘과 바람과 별과 시 (1955년)">증보판</a>
    ")
    "
```

저작 [편집]

시 [편집]

- **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
  - 슬픈 족속
  - 눈감고 간다
  - 또 다른 고향
  - 길
  - 별 헤는 밤

시인 윤동주의 시 목록 가져오기 (3/3) : poem-list. ipynb

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

quote_name = urllib.parse.quote('저자:윤동주')
print(quote_name)

homeaddr = 'https://ko.wikisource.org/wiki/'
url = homeaddr + quote_name
print(url)

res = req.urlopen(url)
soup = BeautifulSoup(res, "html.parser")

a_list = soup.select("#mw-content-text > div > ul > li a")

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