웹크롤링과 스크레이핑

안 화 수

크롤링(Crawling)

- **크롤링(Crawling)**

  크롤링(Crawling)이란 프로그램이 웹 사이트를 정기적으로 돌며 정보를 추출하는 기술입니다.
  크롤링하는 프로그램을 "크롤러(Crawler)" 또는 "스파이더(Spider)" 라고 합니다.

  예를 들어, 검색엔진을 구현할 때 사용하는 크롤러는 웹 사이트의 링크를 타고 웹 사이트를 돌아 다닙니다. 그리고 웹 사이트의 데이터를 모아서 데이터베이스에 저장합니다.

스크레이핑(Scraping)

- **스크레이핑(Scraping)**
  스크레이핑(Scraping)이란 웹 사이트에 있는 특정 정보를 추출하는 기술을 의미합니다. 스크레이핑을 이용하면 웹 사이트에 있는 정보를 쉽게 수집할 수 있습니다.

웹에 공개된 정보는 대부분 HTML 형식입니다. 이를 가져와서 데이터베이스에 저장 하려면 데이터 가공이 필요합니다.
광고 등의 불필요한 정보를 제거하고, 필요한 정보만 가져오려면 사이트 구조를 분석해야 합니다. 따라서 스크레이핑이라는 기술은 웹에서 데이터를 추출하는 것뿐만 아니라 그러한 구조를 분석하는 것도 포함됩니다.

또한 최근에는 로그인을 해야만 유용한 정보에 접근할 수 있는 사이트도 많습니다. 이 경우 단순히 URL을 알고 있는 것만으로는 유용한 정보에 접근할 수 없습니다. 따라서 제대로 스크레이핑을 하려면 로그인 해서 필요한 웹 페이지에 접근하는 기술도 알아야 합니다.

웹 데이터 다운로드

- **웹상의 정보를 추출하는 방법**

1. 파이썬은 웹 사이트에 있는 데이터를 추출하기 위해 "urllib" 라이브러리를 사용합니다. 이 라이브러리를 이용하면 HTTP 또는 FTP를 사용해 데이터를 다운로드 할 수 있습니다.

2. urllib 라이브러리는 URL을 다루는 모듈을 모아 놓은 패키지 입니다.

3. 그 중에서도 urllib.request 모듈은 웹 사이트에 있는 데이터에 접근하는 기능을 제공합니다. 또한 인증, 리다이렉트, 쿠키(Cookie) 처럼 인터넷을 이용한 다양한 요청과 처리를 지원합니다.

4. BeautifulSoup 라이브러리를 이용하면 간단하게 HTML과 XML에서 정보를 추출할 수 있습니다.

urllib.request

urllib.request 모듈

- **urllib.request를 이용한 다운로드 : urlretrieve( ) 함수**

파일을 다운로드할 때는 urllib.request 모듈에 있는 urlretrieve( ) 함수를 사용합니다. 이 함수를 사용하면 직접 파일을 다운로드 할 수 있습니다.

download_png1.py

```
import urllib.request as req

# URL과 저장 경로 지정하기
url = https://www.naver.com
#url = "https://t1.daumcdn.net/daumtop_chanel/op/20170315064553027.png"
savename = "naver.html"
#savename = "daum.png"

# 다운로드
req.urlretrieve(url, savename)
print("저장되었습니다...!")
```

urllib.request 모듈

- **urllib.request를 이용한 다운로드 : urlopen( ) 함수**
  이번에는 파일을 다운로드할 때는 urllib.request 모듈에 있는 urlopen( ) 함수를 이용해 메모리 상에 데이터를 올리고, 그 이후에 파일에 저장해 봅니다.

download_png2.py

```
import urllib.request  as req

# URL과 저장 경로 지정하기
url = " https://t1.daumcdn.net/daumtop_chanel/op/20170315064553027.png"
savename = "daumlog.png"

# 다운로드
mem = req.urlopen(url).read()

# 파일로 저장하기 ( w:쓰기모드, b:바이너리 모드 )
with open(savename, mode="wb") as f:
        f.write(mem)
        print("저장되었습니다...!")
```

urllib.request 모듈

- **기상청의 RSS 서비스**

기상청의 RSS서비스는 아래 URL에 지역번호를 지정하면 해당 지역의 기상 정보를 제공해 줍니다.

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

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

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

urllib.request 모듈

❖ xml파일 읽기

xml_forecast.py

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

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

if not os.path.exists(savename):
    req.urlretrieve(url, savename)

# BeautifulSoup로 분석하기
xml = open(savename, "r", encoding="utf-8").read()
soup = BeautifulSoup(xml, 'html.parser')
```

urllib.request 모듈

```
# 각 지역 확인하기
info = {}
for location in soup.find_all("location"):
    name = location.find('city').string                # 도시명
    wf = location.find('wf').string                    # 날씨
    tmx = location.find('tmx').string                  # 최고기온
    tmn = location.find('tmn').string                  # 최저기온

    weather = wf + ':' + tmn + ' ~ ' + tmx
    if not (name in info):
        info[name] = []
    info[name].append(weather)                         # info = { name : weather }

# 각 지역의 날씨를 구분해서 출력하기
for name in info.keys():
    print("+", name)
    for weather in info[name]:
        print("| - ", weather)
```

BeautifulSoup

BeautifulSoup모듈

BeautifulSoup 모듈로 스크레이핑하기

1. 스크레이핑이란 웹 사이트에서 데이터를 추출하고, 원하는 정보를 추출하는 것입니다.
   최근에는 인테넷에 데이터가 너무 많으므로 스크레이핑을 잘 활용하는 것이 중요합니다.

2. 파이썬으로 스크레이핑할 때 빼놓을 수 없는 라이브러리가 바로 " BeautifulSoup " 입니다.
   이 라이브러리를 이용하면 간단하게 HTML과 XML에서 정보를 추출할 수 있습니다.

3. 최근 스크레이핑 라이브러리는 다운로드부터 HTML 분석까지 모두 해주는 경우가 많은데,
   BeautifulSoup 라이브러리는 어디까지나 HTML과 XML을 분석해주는 라이브러리 입니다.
   BeautifulSoup 자체에는 다운로드 기능이 없습니다.

BeautifulSoup 모듈 설치 확인

C:₩> pip list

BeautifulSoup 모듈 설치

C:₩> pip install bs4

BeautifulSoup모듈

- **BeautifulSoup 모듈 사용법**

find('태그명') : 특정 태그 1개만 추출하는 역할
find('태그명').string : 특정 태그 안에 있는 텍스트를 추출하는 역할
find('태그명').text : 특정 태그 안에 있는 텍스트를 추출하는 역할
find('태그명').get_text() : 특정 태그 안에 있는 텍스트를 추출하는 역할

find('태그명' , {'class' : 'class명'}) : class 값을 이용해서 추출
find('태그명' , {'id' : 'id명'}) : id명을 이용해서 특정 태그를 구함
find(id = 'id명') : id명을 이용해서 특정 태그를 구함

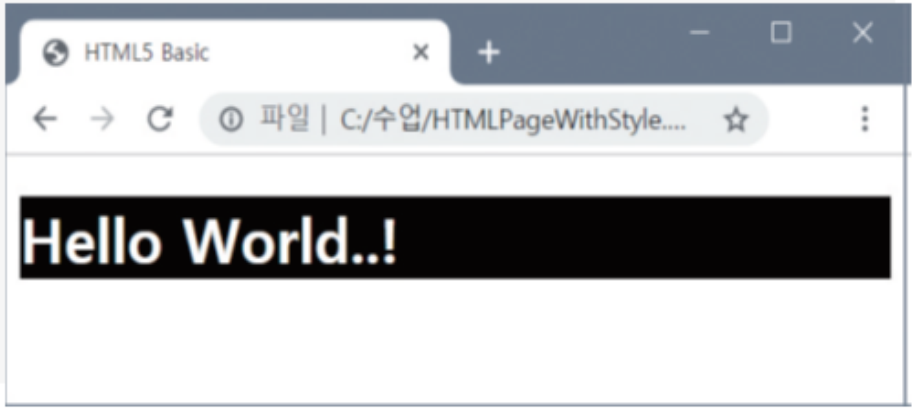
findAll('태그명') : 지정된 모든 태그를 리스트 형태로 추출하는 역할
find_all('태그명') : 지정된 모든 태그를 리스트 형태로 추출하는 역할

HTML 구조

- HTML5 페이지의 구조

CSS 구조

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML5 Basic</title>
    <style>
        h1 {
            color: white;
            background: black;
        }
    </style>
</head>
<body>
    <h1>Hello World..!</h1>
</body>
</html>
```



CSS 구조

```html
<!DOCTYPE html>
<html>
<head>
    <title>CSS3 Selector Basic</title>
    <style>
        .select { color: red; }
    </style>
</head>
<body>
    <ul>
        <li class="select">사과</li>
        <li>바나나</li>
        <li class="select">오렌지</li>
        <li>감</li>
    </ul>
</body>
</html>
```

- 사과
- 바나나
- 오렌지
- 감

CSS 구조

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS3 Selector Basic</title>
    <style>
        #header h1 { color: red; }
        #section h1 { color: orange; }
    </style>
</head>
<body>
    <div id="header">
        <h1 class="title">Lorem ipsum</h1>
        <div id="nav">
            <h1>Navigation</h1>
        </div>
    </div>
    <div id="section">
        <h1 class="title">Lorem ipsum</h1>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
    </div>
</body>
</html>
```

Lorem ipsum

Navigation

Lorem ipsum

BeautifulSoup모듈

❖ BeautifulSoup 기본 예제

bs_find01.py

```
from bs4 import BeautifulSoup

html_str = '<html><div>hello</div></html>'
soup = BeautifulSoup(html_str, "html.parser")

print(soup)                    # <html><div>hello</div></html>
print(soup.find("div"))        # <div>hello</div
print(soup.find("div").text)   # hello
```

BeautifulSoup모듈

❖ BeautifulSoup 기본 예제

bs_find02.py

```
from bs4 import BeautifulSoup

html_str = """
<html>
    <body>
        <ul>
            <li>hello</li>
            <li>bye</li>
            <li>welcome</li>
        </ul>
    </body>
</html>
"""
bs_obj = BeautifulSoup(html_str, "html.parser")

ul = bs_obj.find("ul")                    # ul 태그 안에 있는 모든 내용을 추출함
print(ul)
```

BeautifulSoup모듈

❖ BeautifulSoup 기본 예제

bs_find03.py

```
from bs4 import BeautifulSoup

html_str = """
<html>
    <body>
        <ul>
            <li>hello</li>
            <li>bye</li>
            <li>welcome</li>
        </ul>
    </body>
</html>
"""

bs_obj = BeautifulSoup(html_str, "html.parser")

ul = bs_obj.find("ul")                  # ul 태그 안에 있는 모든 내용을 추출함
li = ul.find("li")                      # 첫번째 li 태그 안에 내용만 추출함
print(li)                               # <li>hello</li>
print(li.text)                          # hello
```

BeautifulSoup모듈

❖ BeautifulSoup 기본 예제

bs_findall01.py

```
from bs4 import BeautifulSoup

html_str = """
<html>
    <body>
        <ul>
            <li>hello</li>
            <li>bye</li>
            <li>welcome</li>
        </ul>
    </body>
</html>
"""
bs_obj = BeautifulSoup(html_str, "html.parser")

ul = bs_obj.find("ul")                  # 첫번째 ul 태그를 구해옴
list = ul.findAll("li")                 # 모든 li 태그를 구해옴
print(list)                             # [<li>hello</li>, <li>bye</li>, <li>welcome</li>]
```

BeautifulSoup모듈

❖ BeautifulSoup 기본 예제 : index 번호로 데이터 접근하기

bs_findall02.py

```
from bs4 import BeautifulSoup

html_str = """
<html>
    <body>
        <ul>
            <li>hello</li>
            <li>bye</li>
            <li>welcome</li>
        </ul>
    </body>
</html>
"""
bs_obj = BeautifulSoup(html_str, "html.parser")

ul = bs_obj.find("ul")                  # 첫번째 ul 태그를 구해옴
list = ul.findAll("li")                 # 모든 li 태그를 구해옴
print(list[1])                          # index 1번(두 번째 li태그)을 구해옴  <li>bye</li>
print(list[1].text)                     # 텍스트 값을 구해옴  bye
```

BeautifulSoup모듈

❖ BeautifulSoup 기본 예제 : class속성으로 데이터 접근하기

bs_class.py

```
from bs4 import BeautifulSoup

html_str = """
<html>
    <body>
        <ul class="greet">
            <li>hello</li>
            <li>bye</li>
            <li>welcome</li>
        </ul>
        <ul class="reply">
            <li>ok</li>
            <li>no</li>
            <li>sure</li>
        </ul>
    </body>
</html>
"""
bs_obj = BeautifulSoup(html_str, "html.parser")
ul_reply = bs_obj.find("ul", {"class":"reply"})        # ul 태그에서 class 속성값이 reply 데이터 추출
list = ul_reply.findAll("li")                          # 모든 li 태그를 추출
for li in list:
    print(li.text)                                     # ok no sure  출력
```

BeautifulSoup모듈

- BeautifulSoup 기본 예제 : 속성 값 추출하기

bs_href.py

```
from bs4 import BeautifulSoup

html_str = """
<html>
    <body>
        <ul class="ko">
            <li>
                <a href="https://www.naver.com/">네이버</a>
            </li>
            <li>
                <a href="https://www.daum.net/">다음</a>
            </li>
        </ul>
        <ul class="sns">
            <li>
                <a href="https://www.google.com/">구글</a>
            </li>
            <li>
                <a href="https://www.facebook.com/">페이스북</a>
            </li>
        </ul>
    </body>
</html>
"""
```

BeautifulSoup모듈

- **BeautifulSoup 기본 예제 : 속성 값 추출하기**

```
bs_obj = BeautifulSoup(html_str, "html.parser")
atag = bs_obj.find("a")          # 첫번째 anchor 태그를 구해옴
print(atag)                      # <a href="https://www.naver.com/">네이버</a>
print(atag['href'])              # https://www.naver.com/
```

BeautifulSoup모듈

BeautifulSoup 기본 사용법

bs_test1.py

```
from bs4 import BeautifulSoup

# 분석하고 싶은 HTML
html = """
<html><body>
  <h1>스크레이핑이란?</h1>
  <p>웹 페이지를 분석하는 것</p>
  <p>원하는 부분을 추출하는 것</p>
</body></html>
"""
# HTML 분석하기 - BeautifulSoup 인스턴스 생성
# HTML을 분석할 때에는 분석기(parser) 종류를 'html.parser' 라고 지정함
soup = BeautifulSoup(html, 'html.parser')

# 원하는 부분 추출하기
h1 = soup.html.body.h1
p1 = soup.html.body.p      # 첫번째 <p>태그를 추출

# p1.next_sibling 에서는 </p> 뒤에 있는 줄바꿈 또는 공백이 추출됨
# next_sibling 을 한번 더 사용해 2번째 <p>태그를 추출합니다.
p2 = p1.next_sibling.next_sibling

# 요소의 글자 출력하기 : text속성으로 요소 사이의 글자 부분을 추출함
print("h1 = " + h1.text)
print("p1  = " + p1. text)
print("p2  = " + p2. text)
```

BeautifulSoup모듈

- **BeautifulSoup 기본 사용법 : id로 요소를 찾는 방법 – find( ) 함수**
  BeautifulSoup는 루트부터 하나하나 요소를 찾는 방법 말고도 id속성을 지정해서 요소를 찾는 find() 메소드를 제공합니다.

bs_test2.py

```
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 : BeautifulSoup 인스턴스 생성
soup = BeautifulSoup(html, 'html.parser')

# find() 메서드로 원하는 부분 추출하기
title = soup.find(id="title")                    # id="title" 인 요소를 구해옴
body  = soup.find(id="body")

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

BeautifulSoup모듈

- BeautifulSoup 기본 사용법 : id로 요소를 찾는 방법

bs_test3.py

```
from bs4 import BeautifulSoup

html = """
<html>
 <head>
  <title>작품과 작가 모음</title>
 </head>
 <body>
  <h1>책 정보</h1>
  <p id="book_title">토지</p>
  <p id="author">박경리</p>

  <p id="book_title">태백산맥</p>
  <p id="author">조정래</p>

  <p id="book_title">감옥으로부터의 사색</p>
  <p id="author">신영복</p>
 </body>
</html>
"""

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

BeautifulSoup모듈

- BeautifulSoup 기본 사용법 : id로 요소를 찾는 방법

```
# id 값이 book_title 인 첫번째 p태그를 구해옴
title = soup.find('p', {"id":"book_title"}).text
print('title:', title)                                  # title: 토지

# id값이 author 인 첫번째 p태그를 구해옴
author = soup.find('p', {"id":"author"}).text
print('author:', author)                                # author: 박경리

# id 값이 book_title 인 모든 p태그를 구해와서 리스트로 리턴
title2 = soup.find_all('p', {"id":"book_title"})
print('title2:', title2)
for t2 in title2:
    print(t2.text)

# id값이 author 인 모든 p태그를 구해와서 리스트로 리턴
author2 = soup.find_all('p', {"id":"author"})
print('author2:', author2)
for a2 in author2:
    print(a2.text)
```

BeautifulSoup모듈

- **urlopen()과 BeautifulSoup – find( ) 함수**
  기상청 RSS에서 특정 내용 추출하기

bs_forecast.py

```
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
res = req.urlopen(url)

# BeautifulSoup으로 분석하기 : BeautifulSoup 인스턴스 생성
soup = BeautifulSoup(res, "html.parser")

# 원하는 데이터 추출하기
# 기상 정보가 있는 xml파일에서 첫번째 title 요소 안에 있는 문자를 구해옴
title = soup.find("title").text
# 기상 정보가 있는 xml파일에서 첫번째 wf 요소 안에 있는 문자를 구해옴
wf = soup.find("wf").text
print(title)
print(wf)
```

BeautifulSoup모듈

- **BeautifulSoup 기본 사용법 : 여러 개의 요소 추출하기 – find_all( ) 함수**
  여러 개의 태그를 한 번에 추출하고 싶을 때는 find_all() 메소드를 사용합니다.

bs_link.py

```
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 : BeautifulSoup 인스턴스 생성
soup = BeautifulSoup(html, 'html.parser')

# find_all() 메서드로 추출하기 : 모든 <a>태그 추출
links = soup.find_all("a")

# 링크 목록 출력하기
for a in links:
    href = a.attrs['href']       # href 속성으로 속성값을 구해옴
    text = a.string              # <a>태그 사이의 글자를 구해옴
    print(text, ">", href)
```

BeautifulSoup모듈

CSS 선택자 사용하기

BeautifulSoup는 자바스크립트 라이브러리인 jQuery처럼 CSS선택자를 지정해서 원하는 요소를 추출하는 기능도 제공합니다.

메소드	설명
soup.select_one(선택자)	CSS 선택자로 요소 하나를 추출
soup.select(선택자)	CSS 선택자로 요소 여러 개를 리스트로 추출

BeautifulSoup모듈

bs_select.py

```
from bs4 import BeautifulSoup

# 분석 대상 HTML
html = """
<html><body>
<div id="meigen">
  <h1>중앙 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
soup = BeautifulSoup(html, 'html.parser')

# 필요한 부분을 CSS 쿼리로 추출하기
# 타이틀 부분 추출하기
h1 = soup.select_one("div#meigen > h1").string
print("h1 =", h1)

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

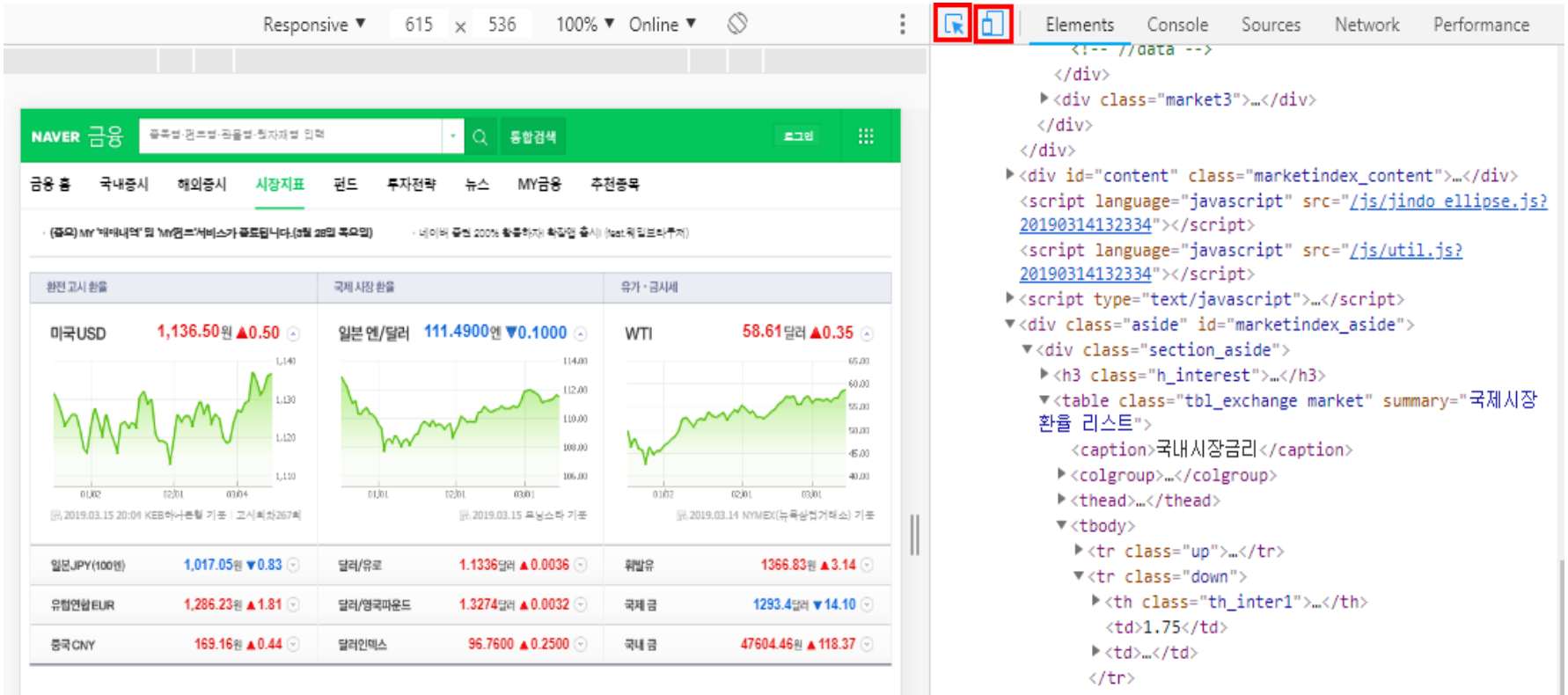
BeautifulSoup모듈

네이버 금융에서 시장지표 추출하기

다양한 금융 정보가 공개되 있는 "네이버 금융"에서 원/달러 환율 정보를 추출하기

네이버 금융의 시장 지표 페이지 : http://finance.naver.com/marketindex

크롬 웹브라우저로 네이버 금융 사이트로 접속 후 개발자 도구 메뉴 선택 (단축키 : F12)



BeautifulSoup모듈

네이버 금융에서 시장지표 추출하기

bs_usd.py

```
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 : 원-달러
price = soup.select_one("div.head_info > span.value").string
print("usd/krw =", price)

# 원하는 데이터 추출하기 : 엔-달러
yen = soup.select_one("a.head.jpy_usd  span.value").string
print("yen/usd =", yen)

# 원하는 데이터 추출하기 : WTI
wti = soup.select_one("a.head.wti  span.value").string
print("wti =", wti)
```

BeautifulSoup모듈

- **기상청의 날씨 정보 구해오기**

  기상청의 날씨 정보(기온, 습도) 구해오는 예제

weather.py

```
import requests                                  # 웹 페이지의 HTML을 가져오는 모듈
from bs4 import BeautifulSoup                    # HTML을 파싱하는 모듈

# 웹 페이지를 가져온 뒤 BeautifulSoup 객체로 만든다.
response =
requests.get('http://www.weather.go.kr/weather/observation/currentweather.jsp')
soup = BeautifulSoup(response.content, 'html.parser')

# <table id="weather_table">을 찾는다.
table = soup.find('table', {'id': 'weather_table'})
```

BeautifulSoup모듈

- 기상청의 날씨 정보 구해오기

```
data = []  # 데이터를 저장할 리스트 생성
for tr in table.find_all('tr'):          # 모든 <tr> 태그를 찾아서 반복(각 지점의 데이터를 가져옴)
    tds = list(tr.find_all('td'))        # 모든 <td> 태그를 찾아서 리스트로 만듦
    # (각 날씨 값을 리스트로 만듦)

    for td in tds:                                  # <td> 태그 리스트 반복(각 날씨 값을 가져옴)
        if td.find('a'):                            # <td> 안에 <a> 태그가 있으면(지점인지 확인)
            point = td.find('a').text               # <a> 태그 안에서 지점을 가져옴
            temperature = tds[5].text  #<td>태그 리스트의 여섯 번째(인덱스 5)에서 기온을 가져옴
            humidity = tds[9].text     # <td>태그 리스트의 열 번째(인덱스 10)에서 습도를 가져옴
            data.append([point, temperature, humidity]) # data 리스트에 지역, 기온, 습도를 추가

print(data)

print('[ 지역 , 기온 , 습도 ]')
for w in data:
    print(w)
```

로그인이 필요한 사이트에서 정보 구하기

requests 모듈

- requests 모듈
  request 모듈은 로그인이 필요한 사이트에서 쿠키와 세션 정보를 구해오기 위해서 사용되는 모듈입니다.

- requests 모듈 설치 확인
  c:₩> pip list

- requests 모듈 설치
  c:₩> pip install requests

- 네이버나 다음 같은 검색 포털 사이트들은 검색 프로그램(로봇)이 쉽게 로그인할 수 없게 보안적으로 막혀 있습니다.

- 한빛 출판네트워크
  - 로그인 페이지 : http://www.hanbit.co.kr/member/login.html
  - 마이페이지 : http://www.hanbit.co.kr/myhanbit/myhanbit.html

requests 모듈

- requests 모듈
  requests 모듈을 이용해서 현재 시간에 대한 테이터를 추출하고, 추출한 데이터를 텍스트 형식과 바이너리 형식으로 출력하는 예제

requests_test.py

```
import requests

r = requests.get("http://api.aoikujira.com/time/get.php")

# 텍스트 형식으로 데이터 추출하기
text = r.text
print(text)

# 바이너리 형식으로 데이터 추출하기
bin = r.content
print(bin)
```

requests 모듈

- requests 모듈
  requests 모듈을 이용해서 바이너리 데이터인 이미지를 다운로드 받아서 저장하는 예제

requests_png.py

```
import requests

r = requests.get("http://wikibook.co.kr/wikibook.png")

# 바이너리 형식으로 데이터 저장하기 ( w:쓰기모드, b:바이너리모드 )
with open("test.png", "wb") as f:
    f.write(r.content)
print("saved")
```

파이썬으로 로그인하기

- login_getmileage.py

```
import requests
from bs4 import BeautifulSoup

# 아이디와 비밀번호 지정하기[자신의 것을 사용]
USER = "guardian23"
PASS = " "

# 세션 시작하기
session = requests.session()

# 파이썬 프로그램으로 로그인하기
login_info = {
    "m_id": USER,           # 아이디 지정
    "m_passwd": PASS        # 비밀번호 지정
}
url_login = "http://www.hanbit.co.kr/member/login_proc.php"
res = session.post(url_login, data=login_info)          # URL에 post 요청을 수행
```

파이썬으로 로그인하기

```
# 로그인이 완료되면 마이페이지에 접근
url_mypage = "http://www.hanbit.co.kr/myhanbit/myhanbit.html"
res = session.get(url_mypage)

# BeautifulSoup 인스턴스 생성하기 - 텍스트 형식으로 데이터 추출
soup = BeautifulSoup(res.text, "html.parser")

# 마일리지와 이코인 가져오기
mileage = soup.select_one(".mileage_section1 span").get_text()
ecoin = soup.select_one(".mileage_section2 span").get_text()
print("마일리지: " + mileage)
print("이코인: " + ecoin)
```

selenium

웹 브라우저 원격 조작 : selenium

- 웹 브라우저 원격조작에 사용하는 selenium

자바스크립트를 많이 사용하는 웹 사이트의 경우에는 앞에서 소개한 requests 모듈로 필요한 데이터를 스크레이핑 하기 힘들다. 이런 경우에 웹 브라우저를 원격 조작할때 사용하는 도구로 selenium 이 있습니다.

일반적으로 selenium 은 웹 애플리케이션 테스트를 자동화할 때 사용하지만, 스크레이핑을 할 때도 유용하게 사용할 수 있습니다. selenium 을 이용하면 자동으로 URL을 열고 클릭할 수 있으며, 스크롤하거나, 문자를 입력하는 등의 다양한 조작을 자동화 할 수 있습니다.

또한 화면을 캡쳐해서 이미지로 저장하거나 HTML의 특정 부분을 추출하는 것도 가능하고, 구글 크롬, 파이어폭스, 인터넷익스플로러, 오페라 등의 다양한 웹브라우저를 원격으로 조작할 수 있습니다.

웹 브라우저 원격 조작 : selenium

- selenium + chrome 환경구축

1. selenium 설치

c:₩> pip install selenium

2. 브라우저 Driver 설치

1) selenium을 사용하기 위해서는 브라우저 별로 driver를 다운로드 해야 한다.
2) selenium의 기능중에서는 컴퓨터가 직접 웹 브라우저를 띄운 후 코드를 입력 후에 동작시킬 수 있도록 webdriver라는 api를 제공한다.
3) 컴퓨터가 webdriver api로 브라우저를 직접 제어할 수 있도록 도와주는 driver를 설치해주어야 한다. 크롬, 엣지, 파이어폭스, 사파리에서 driver를 제공한다.
4) 자신의 운영체제와 웹브라우저 버전에 맞는 드라이버 파일을 다운로드 받도록 하자. 다운로드 받은 파일을 압축 해제 후 아래 위치에 저장한다.

C:₩downloads₩chromedriver.exe

Chrome 버전 확인


설정 검색
Chrome 정보
Chrome
Chrome이 최신 버전입니다.
버전 105.0.5195.102(공식 빌드) (64비트)
Chrome 도움말 보기
문제 신고
Chrome
Copyright 2022 Google LLC. All rights reserved.
Chrome은 Chromium 오픈소스 프로젝트를 비롯한 여러 오픈소스 소프트웨어를 기반으로 제작된 브라우저입니다
서비스 약관
새 탭 Ctrl+T
새 창 Ctrl+N
새 시크릿 창 Ctrl+Shift+N
방문 기록
다운로드 Ctrl+J
북마크
글꼴 크기 - 100% +
인쇄 Ctrl+P
전송...
찾기... Ctrl+F
도구 더보기
수정 잘라내기 복사 붙여넣기
설정
도움말
종료
Chrome 정보
새로운 기능
고객센터
문제 신고하기... Alt+Shift+I

웹 브라우저 원격 조작 : selenium

- selenium 다운로드

1.chrome
https://sites.google.com/chromium.org/driver/downloads?authuser=0

2.firefox
https://github.com/mozilla/geckodriver/releases

3.edge
https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/

4.Safari
https://webkit.org/blog/6900/webdriver-support-in-safari-10/

웹 브라우저로 google사이트 접속하기

- selenium01.py

```
# selenium 모듈에서 webdriver를 불러온다
from selenium import webdriver

# 다운로드 받아 압축을 해제한 driver 파일 경로를 path 변수에 할당한다
path = “c:/downloads/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')
```

웹 사이트를 이미지로 캡처하기

- selenium_capture.py

```
from selenium import webdriver

path = "c:/downloads/chromedriver.exe"
driver = webdriver.Chrome(path)

# 3초 대기하기
driver.implicitly_wait(3)

# URL 읽어 들이기
driver.get("https://www.nate.com")

# 화면을 캡처해서 저장하기
driver.save_screenshot("website.png")

# 브라우저 종료하기
driver.quit()
```

자바스크립트 실행해보기

- selenium_js.py

```
from selenium import webdriver

path = "c:/downloads/chromedriver.exe"

# 조금만 기다리면 selenium으로 제어할 수 있는 브라우저 새창이 뜬다
driver = webdriver.Chrome(path)

driver.implicitly_wait(3)

# 웹 페이지 열기
driver.get("https://www.google.com")

# 자바스크립트 실행하기
r = driver.execute_script("return 100 + 50")
print(r)
```

웹 브라우저로 검색하기

- selenium_search.py

```
from selenium import webdriver

path = "c:/downloads/chromedriver.exe"

# selenium의 webdriver로 크롬 브라우저를 실행한다
driver = webdriver.Chrome(path)

# google에 접속한다
driver.get("http://www.google.co.kr")

# 검색 입력 부분에 커서를 올리고 검색 입력 부분에 다양한 명령을 내리기 위해
# elem 변수에 할당한다
elem = driver.find_element_by_name("q")

# 검색어 입력창에 default로 값이 있을 수 있어 비운다
elem.clear()

# 검색어를 입력한다
elem.send_keys("Selenium")

# 검색을 실행한다
elem.submit()
```

nate.com 자동 로그인

```
from selenium import webdriver

path = 'c:/downloads/chromedriver.exe'

driver = webdriver.Chrome(path)

# 3초 대기
driver.implicitly_wait(3)

# nate.com 페이지 불러오기
driver.get('https://www.nate.com')

# 아이디 / 비번 입력 양식의 태그 구하기
driver.find_element_by_id('ID').send_keys('guardian23@nate.com')
driver.find_element_by_id('PASSDM').send_keys('비밀번호')

# 로그인 버튼 클릭
driver.find_element_by_id('btnLOGIN').click()
```

네이버 자동 로그인
(자동 로그인 방지에 걸림)

```
from selenium import webdriver
import time

path = 'c:/downloads/chromedriver.exe'

driver = webdriver.Chrome(path)

# 3초 대기
driver.implicitly_wait(3)

# naver.com 페이지 불러오기
driver.get('https://www.naver.com')

# 로그인 버튼을 찾고 클릭합니다
login_btn = driver.find_element_by_class_name('link_login')
login_btn.click()
time.sleep(1)

# 아이디 / 비번 입력 양식의 태그 구하기
driver.find_element_by_id('id').send_keys('아이디')
driver.find_element_by_id('pw').send_keys('비밀번호')

# 로그인 버튼 클릭
driver.find_element_by_id('log.login').click()
```

네이버 자동 로그인 (1/2)

❖ 로그인 자동 입력 방지에 걸리지 않는 방법

pyperclip 라이브러리를 이용해서 아이디와 비밀번호를 클립보드에 저장한 뒤, Ctrl + v 키로 붙여넣기로 처리

```
# pyperclip 모듈 설치
# pip install pyperclip

from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
import pyperclip

path = 'c:/downloads/chromedriver.exe'

driver = webdriver.Chrome(path)
driver.get('https://www.naver.com')
time.sleep(1)

# 로그인 버튼을 찾고 클릭합니다
login_btn = driver.find_element_by_class_name('link_login')
login_btn.click()
time.sleep(1)
```

네이버 자동 로그인 (2/2)

```
# id, pw 입력할 곳을 찾습니다.
tag_id = driver.find_element_by_name('id')
tag_pw = driver.find_element_by_name('pw')
tag_id.clear()
time.sleep(1)

# id 입력
tag_id.click()
# pyperclip.copy('아이디')
pyperclip.copy('giduck23')
tag_id.send_keys(Keys.CONTROL, 'v')
time.sleep(1)

# pw 입력
tag_pw.click()
pyperclip.copy('비밀번호')
tag_pw.send_keys(Keys.CONTROL, 'v')
time.sleep(1)

# 로그인 버튼을 클릭합니다
login_btn = driver.find_element_by_id('log.login')
login_btn.click()
```