

12장. 웹 스크래핑(웹 크롤링)

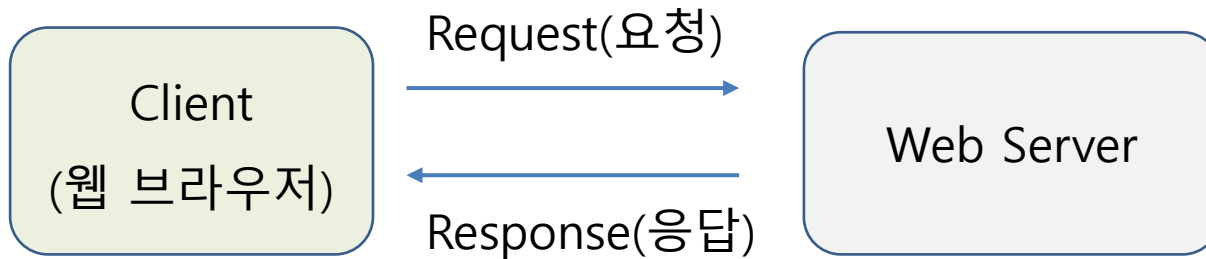


웹 스크래핑 = 웹 크롤링

*웹 Scraping*이란?

인터넷에 있는 웹 페이지를 방문해서 자료를 수집하는 일.
웹 크롤링이라고도 한다.

▶ 웹 서버에 요청하고 응답받기



웹 스크래핑 = 웹 크롤링

▷ requests 모듈 사용

```
import requests

url = "https://www.python.org"
response = requests.get(url)  # url 객체 저장
print(response)
html = response.text  # html 코드 저장
#print(html)

url2 = "https://www.python.org/3"
response = requests.get(url2)
print(response)
```

<Response [200]>

<Response [404]>

정상

페이지 없음

로봇 배제 표준

▷ 로봇 배제 표준

로봇 배제 표준이란?

웹사이트에 로봇이 접근하는 것을 방지하기 위한 규약 robots.txt에 기술하고 있다.

- 로봇에 의한 접근이 허용되는 경우라도 웹 서버에 무리가 갈 만큼 반복적으로 웹 페이지를 요청하는 것과 같이 서비스 안정성을 해칠 수 있는 행위를 하지 않아야 함
- 크롤링(또는 스크래핑)으로 취득한 자료를 임의로 배포하거나 변경하는 등의 행위는 저작권을 침해할 수 있으므로 저작권 규정을 준수해야 함

로봇 배제 표준

▷ 로봇 배제 표준

템플릿 태그	설 명
User-agent: * Disallow: /	모든(*) 로봇에게 루트 디렉터리(/) 이하 모든 문서에 대한 접근을 차단한다.
User-agent: * Allow: /	모든(*) 로봇에게 루트 디렉터리(/) 이하 모든 문서에 대한 접근을 허락한다.
User-agent: * Disallow: /temp/	모든(*) 로봇에게 특정 디렉터리(/temp/)에 대한 접근을 허락한다.

로봇 배제 표준

▷ 로봇 배제 표준

```
← → ↻ google.com/robots.txt

User-agent: *
Disallow: /search
Allow: /search/about
Allow: /search/static
Allow: /search/howsearchworks
Disallow: /sdch
Disallow: /groups
Disallow: /index.html?
Disallow: /?
Allow: /?hl=
Disallow: /?hl=*&
Allow: /?hl=*&gws_rd=ssl$
Disallow: /?hl=*&*&gws_rd=ssl
```

```
← → ↻ python.org/robots.txt

# Directions for robots.  See this URL:
# http://www.robotstxt.org/robotstxt.html
# for a description of the file format.

User-agent: HTTrack
User-agent: puf
User-agent: MSIECrawler
Disallow: /

# The Krugle web crawler (though based on Nutch) is OK.
User-agent: Krugle
Allow: /
Disallow: /~guido/orlijn/
Disallow: /webstats/
```

로봇 배제 표준

▷ 로봇 배제 표준

```
import requests

urls = ["https://www.naver.com/", "https://www.python.org/"]
filename = "robots.txt"

for url in urls:
    file_path = url + filename
    print(file_path)
    resp = requests.get(file_path)
    print(resp.text)
```

웹 스크레이핑 = 웹 크롤링

❖ BeautifulSoup 모듈

HTML과 XML 문서를 파싱하기 위한 파이썬 라이브러리이다.

웹 서버로 부터 HTML 소스코드를 가져온 다음에는 HTML 태그 구조를 해석하기 위한 과정이 필요하다.

HTML 소스 코드를 해석하는 것을 **파싱(parsing)**이라고 부른다.

HTML에서 정보를 추출하기 위해 BeautifulSoup 라이브러리를 사용한다.

▶ BeautifulSoup 설치

pip install BeautifulSoup4

▶ BeautifulSoup 사용

from bs4 import BeautifulSoup

웹 스크레이핑 = 웹 크롤링

- `find()`는 처음 나오는 태그 요소를 찾아옴

```
from bs4 import BeautifulSoup
```

```
html_str = '''
```

```
<html>
```

```
  <body>
```

```
    <ul class = 'item'>
```

```
      <li>인공지능</li>
```

```
      <li>Big Data</li>
```

```
      <li>로봇</li>
```

```
    </ul>
```

```
    <ul class = 'comlang'>
```

```
      <li>Python</li>
```

```
      <li>C/C#</li>
```

```
      <li>Java</li>
```

```
    </ul>
```

```
  </body>
```

```
</html>
```

```
'''
```

```
<ul class="item">
```

```
<li>인공지능</li>
```

```
<li>Big Data</li>
```

```
<li>로봇</li>
```

```
</ul>
```

인공지능

Big Data

로봇

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

```
#print(html)
```

```
first_ul = soup.find('ul') #처음 나오는 ul 태그를 찾음
```

```
print(first_ul)
```

```
print(first_ul.text)
```

웹 스크레이핑 = 웹 크롤링

- `findAll()` 은 모든 태그 요소를 찾아서 리스트로 반환함

```
soup = BeautifulSoup(html_str, "html.parser")
first_ul = soup.find('ul', attrs={'class': 'item'}) #dict = {} 로 찾기
all_li = first_ul.findAll('li') # 모든 li요소를 리스트로 반환
print(all_li)
print(all_li[1])
print(all_li[1].text)

for li in all_li:
    print(li.text)
```

Dictionary 자료구조
{키 : 값}

위키디피아 – 서울 지하철

- 구글에서 'Seoul Subway' 검색 > 위키디피아
<head> 태그의 <title> 스크래핑하기



title 태그 요소: <title>Seoul Metropolitan Subway - Wikipedia</title>
title 태그 이름: title
title 태그 문자열: Seoul Metropolitan Subway - Wikipedia

위키디피아 – 서울 지하철

- 구글에서 '서울 지하철' 검색 > 위키디피아

```
import requests
from bs4 import BeautifulSoup

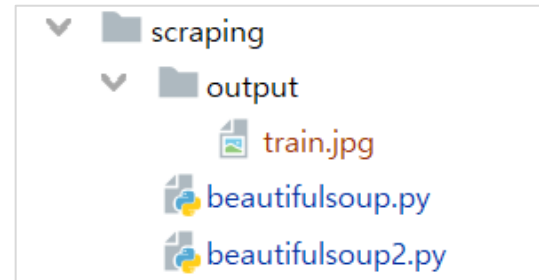
url = "https://en.wikipedia.org/wiki/Seoul_Metropolitan_Subway"
response = requests.get(url)
html = BeautifulSoup(response.text, 'html.parser')

print(html.head)
print("title 태그 요소:", html.title)
print("title 태그 이름:", html.title.name)
print("title 태그 문자열:", html.title.string)

#print(html.body)
```

위키디피아 – 서울 지하철

- '서울 지하철' > 이미지 파일 PC에 저장하기



```
Elements Console Sources Network Performance Memory Application Security
<caption class="infobox-title">Seoul Metropolitan Subway</caption>
<tbody>
  <tr>
    <td colspan="2" class="infobox-image">
      <a href="/wiki/File:Seoul-metro-2009-20180916-103548.jpg" class="image">
        
      </a>
    </td>
  </tr>
</tbody>
```

위키디피아 – 서울 지하철

```
import requests
from bs4 import BeautifulSoup

url = "https://en.wikipedia.org/wiki/Seoul_Metropolitan_Subway"
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

# 지하철 사진 경로 출력
target_img = soup.find('img', attrs={'alt': 'Seoul-metro-2009-20180916-103548.jpg'})
print(target_img)

# 소스 사진 읽어오기
target_img_src = target_img.get('src') # 이미지 경로 가져와서 저장
print("이미지 파일 경로:", target_img_src)

target_img_response = requests.get('http:' + target_img_src) # 이미지 경로 url 저장

# 바이너리 파일 모드로 파일에 쓰기
with open("./output/train.jpg", 'wb') as f:
    f.write(target_img_response.content) # 이미지 - content 속성 사용
print("이미지 파일로 저장했습니다.")
```

네이버에서 웹 크롤링하기

✓ Naver에서 필요한 정보 추출하기



```
▼<div class="group_flex">
  ▶<div class="logo_area"></div>
  ▼<div class="service_area">
    ▼<a id="NM_set_home_btn" href="https
      "네이버를 시작페이지로"
      ::after
    </a>
```

네이버에서 웹 크롤링하기

- Naver에서 필요한 정보 추출하기

```
import requests
from bs4 import BeautifulSoup

response = requests.get("http://www.naver.com")    # 응답 객체 생성
html = BeautifulSoup(response.text, 'html.parser') # html 문서로 파싱

div = html.find('div', {'class' : 'service_area'})
first_a = div.find('a')
#print(first_a)
print(first_a.text)
```


네이버에서 웹 크롤링하기

실습 문제

네이버 시작 페이지의 우측 상단의 링크 중에서 '주니어네이버'를 추출하세요.
(파일이름 : naver_begin_a.py)

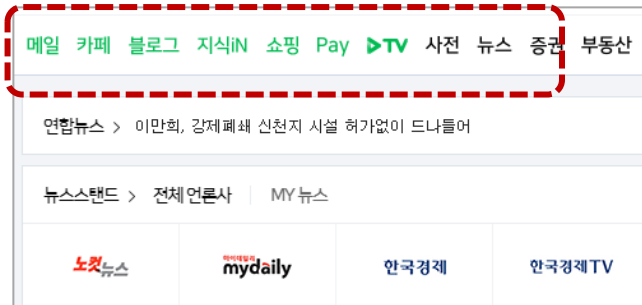
👉 실행 결과
주니어네이버

```
import requests
from bs4 import BeautifulSoup

response = requests.get("https://www.naver.com/")
soup = BeautifulSoup(response.text, 'html.parser')
div = soup.find('div', attrs={'class' : 'service_area'})
all_a = div.findAll('a')
print(all_a[1].text)
```

네이버에서 웹 크롤링하기

✓ Naver 메뉴 가져오기



```
▼<div class="group_nav">
  ▼<ul class="list_nav type_fix"> == $0
    ▼<li class="nav_item">
      ▼<a href="https://mail.naver.com/" class="nav"
        <i class="ico_mail"></i>
        "메일"
      </a>
```

```
import requests
from bs4 import BeautifulSoup

response = requests.get("https://www.naver.com/")
soup = BeautifulSoup(response.text, "html.parser")
ul = soup.find('ul', attrs={'class': 'list_nav type_fix'})
lis = ul.findAll('li')
for li in lis:
    a = li.find('a')
    print(a.text)

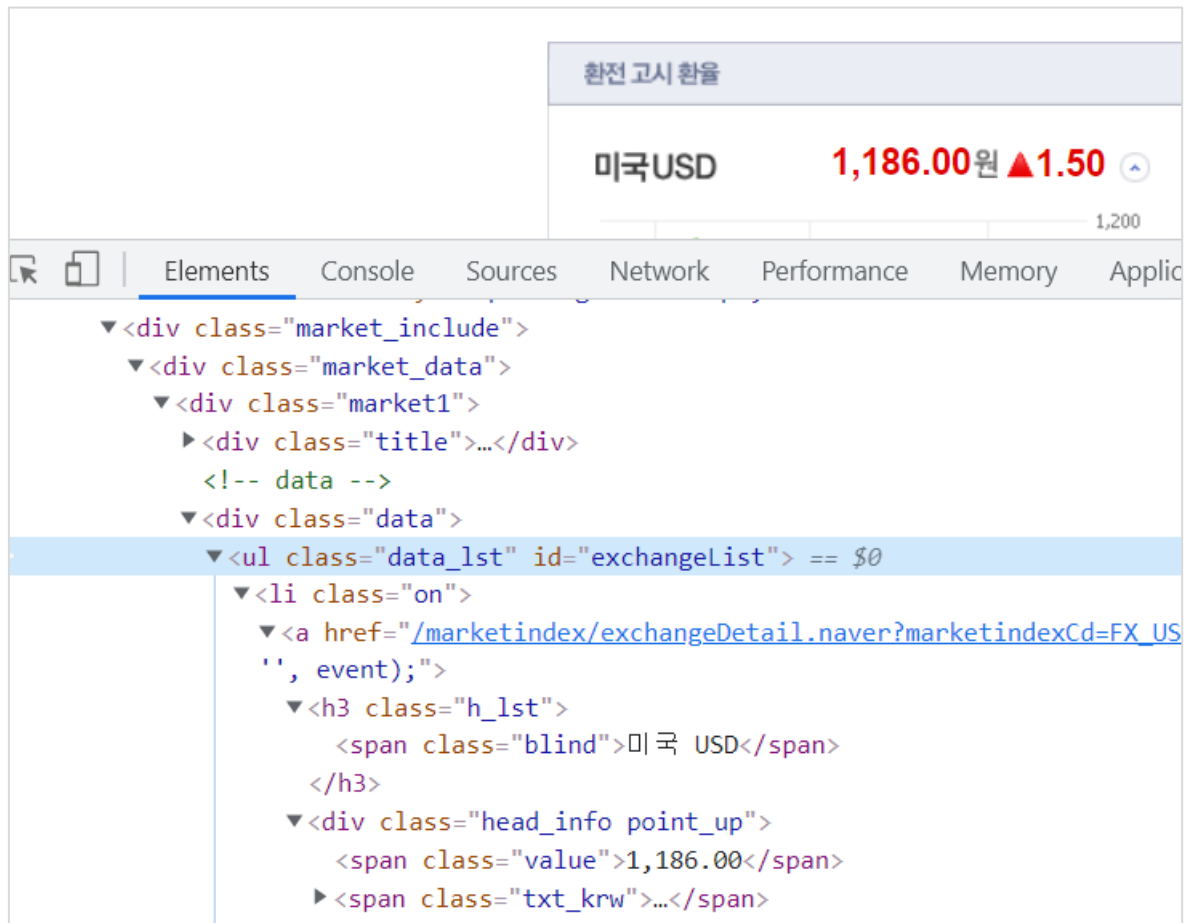
# li를 찾지 않고 직접 'a' 태그로 찾을
all_a = ul.findAll('a')
for a in all_a:
    print(a.text)

# '카페' 메뉴 찾기
print(all_a[1].text)
```

네이버 금융 크롤링하기

● 환율정보 수집하기

네이버 > 증권 > 시장지표 > 환전 고시 환율



The screenshot shows the Naver Finance '환전 고시 환율' (Exchange Rate) page for '미국USD' (US Dollar) at 1,186.00 KRW, up 1.50. A DOM tree overlay is visible, showing the following structure:

```
<div class="market_include">
  <div class="market_data">
    <div class="market1">
      <div class="title">...</div>
      <!-- data -->
      <div class="data">
        <ul class="data_lst" id="exchangeList"> == $0
          <li class="on">
            <a href="/marketindex/exchangeDetail.naver?marketindexCd=FX_US" ...>
              <h3 class="h_lst">
                <span class="blind">미국 USD</span>
              </h3>
              <div class="head_info point_up">
                <span class="value">1,186.00</span>
                <span class="txt_krw">...</span>
              </div>
            </a>
          </li>
        </ul>
      </div>
    </div>
  </div>
</div>
```

네이버 금융 크롤링하기

- 환율정보 수집하기 – find() 사용하여 첫번째 환율 찾기

```
import requests
from bs4 import BeautifulSoup
```

👉 실행 결과

미국 USD 1,186.00

```
# 네이버 증권 > 시장지표 > 환전 고시 환율
resp = requests.get("https://finance.naver.com/marketindex")
soup = BeautifulSoup(resp.text, "html.parser")
ul = soup.find('ul', attrs={'class': 'data_lst'}) #첫번째 ul
#print(ul)
li = ul.find('li') #첫번째 li
#print(li)
exchange = li.find('span', attrs={'class': 'blind'}) # 환율 종류
#print(exchange)
value = li.find('span', attrs={'class': 'value'}) # 환율 지수
print(exchange.text, value.text)
```

네이버 금융 크롤링하기

- 환율정보 수집하기 – findall() 사용하기

```
resp = requests.get("https://finance.naver.com/marketindex")
soup = BeautifulSoup(resp.text, "html.parser")
ul = soup.find('ul', {'class': 'data_lst'})
lis = ul.findAll('li') #모든 li 찾기
#print(lis)
for li in lis:
    exchange = li.find('span', attrs={'class': 'blind'}) # 환율 종류
    value = li.find('span', attrs={'class': 'value'}) # 환율 지수
    #print(exchange.string, ': ', value.string)
    print(exchange.string.split(' ')[-1], ': ', value.string)
    #맨 마지막 문자열 추출
```

☞ 실행 결과

USD : 1,168.00
JPY(100엔) : 1,069.06
EUR : 1,381.57
CNY : 181.72

네이버 금융 크롤링하기

- 환율정보 수집하기

select(태그요소.선택자이름) – 전체 검색(리스트로 반환)

select_one(태그요소.선택자이름) – 1개 검색

```
response = requests.get("https://finance.naver.com/marketindex")
soup = BeautifulSoup(response.text, "html.parser")
lis = soup.select("div.market1 ul li") #market1 전체 검색
#lis = soup.select("ul.data_lst li") #market2, 3까지 모두 찾을
#print(lis)

for li in lis:
    exchange = li.select_one("span.blind") #환율 종류
    #print(exchange)
    value = li.select_one("span.value") #환율 지수
    #print(exchange.string, value.string)
    print(exchange.string.split(' ')[-1], ': ', value.string)
```

👉 실행 결과

```
USD : 1,168.00
JPY(100엔) : 1,069.06
EUR : 1,381.57
CNY : 181.72
```

네이버 금융 크롤링하기

◆ 삼성전자 주식 가격 가져오기

인기 검색 종목			더보기
1. 카카오게임즈	71,600	▲ 14,100	
2. 삼성전자	80,000	▼ 100	
3. 진원생명과학	49,850	▲ 3,200	
4. 카카오	159,500	▼ 500	
5. 이연제약	58,300	▲ 9,650	

네이버 > 금융 홈 > 우측

finance.naver.com/item/main.nhn?code=005930

NAVER 금융 종목명·지
금융 홈 **국내증시** 해외
삼성전자 005930 코스피
80,000
전일대비 ▼100 | -0.12%
선차트 1일 | 1주일 | 3개월

네이버 금융 크롤링하기

◆ 단일 주식 종목 찾아 오기 - 함수이용

```
▼<div class="today">
  ▼<p class="no_today">
    ▼<em class="no_up">
      <span class="blind">77,800</span>
      <span class="no7">7</span>
      <span class="no7">7</span>
      <span class="shim">,</span>
      <span class="no8">8</span>
      <span class="no0">0</span>
```

```
def getcontent():
    url = "https://finance.naver.com/item/main.naver?code=005930"
    response = requests.get(url)
    content = BeautifulSoup(response.text, "html.parser")
    return content

content = getcontent()
#no_today = soup.find('p', attrs={'class':'no_today'})
no_today = content.select_one('p.no_today')
print(no_today)
# price = no_today.find('span', attrs={'class':'blind'})
price = no_today.select_one('span.blind')
print(price)
print("삼성전자 주가 : {}원".format(price.text))
```


네이버 금융 크롤링하기

◆ 단일 주식 종목 찾아 오기 - 함수이용

```
<p class="no_today">  
<em class="no_down">  
<span class="blind">80,100</span>  
<span class="no8">8</span><span class="no0">0</span>  
</em>  
</p>  
삼성전자 주가 : 80,100원
```

거래중일때 웹에서는 보이지 않음

네이버 금융 크롤링하기

◆ 주식정보 찾기 – 여러 종목 가격 가져오기

```
def getcontent(item_code):  
    url = "https://finance.naver.com/item/main.nhn?code=" + item_code  
    html = request.urlopen(url)  
    content = BeautifulSoup(html, 'html.parser')  
    return content  
  
def get_price(item_code):  
    content = getcontent(item_code)  
    # no_today = content.find('p', {'class': 'no_today'})  
    no_today = content.select_one("p.no_today")  
    # price = no_today.find('span', {'class': 'blind'})  
    price = no_today.select_one("span.blind")  
    return price
```

네이버 금융 크롤링하기

◆ 주식정보 찾기 – 여러 종목 가격 가져오기

```
삼성전자 = get_price("005930")  
네이버 = get_price("035420")  
카카오 = get_price("035720")  
print("삼성전자 주가 : {}원".format(삼성전자.text))  
print("네이버 주가 : {}원".format(네이버.text))  
print("카카오 주가 : {}원".format(카카오.text))
```

```
삼성전자 주가 : 80,000원  
네이버 주가 : 414,000원  
카카오 주가 : 159,500원
```