파이썬 프로그래밍

(Web Scraping)

01. 웹 스크랩핑 소개

웹 스크래핑(Web Scraping)은 웹 페이지로부터 원하는 정보를 추출하는 기법이다. 어떤 서비스에서 API가 별도로 제공되고 있지 않지만 웹 페이지로 정보가 제공되고 있을 때, 웹 스크래핑 기법을 이용하면 원하는 정보를 획득할 수 있다.

웹 스크래핑은 흔히 웹 크롤링(Web Crawling)이라고도 많이 불린다. 물론 엄밀하게 두 단어는 서로 다른 의미이다. 크롤링은 여러 웹 페이지를 기계적으로 탐색하는 일을 말합니다. 한편 웹 스크래핑은 특정한 하나의 웹 페이지를 탐색하고, 소스코드 작성자가 원하는 정보를 콕 집어 얻어낸다는 점에서 크롤링과 차이가 있다. 그럼에도 크롤링과 스크래핑은 구현방법이 거의 같기 때문에, 실무에서는 구분 없이 불린다.

- [스크랩핑 페이지]-[F12]-[오른쪽마우스]-[Copy]-[Copy XPath] 또는 [Copy full XPath] 값을 얻을 수 있다.

[c:]-[data]-[python]-[scraping] 01_xpath.txt

```
/학교/학년/반/학생[2]   '/학교' 학교 한 단계 아래 강감찬 검색
//*[@학번="1-1-5"]   '//*' 문서전체에서 '[@학번="1-1-5"]' 학번속성이 1-1-5인 홍길동 검색
[네이버]에서 로그인 버튼  //*[@id="account"]/div/a
<학교 이름="나도고등학교">
    <학년 value="1학년">
        <반 value="1반">
            <학생 value="1번" 학번='1-1-1">홍길동</학생>
            <학생 value="2번" 학번='1-1-2">심청이</학생>
            <학생 value="3번" 학번='1-1-3">강감찬</학생>
            <학생 value="4번" 학번='1-1-4">이몽룡</학생>
            <학생 value="5번" 학번='1-1-5">홍길동</학생>ㄴ
        </반>
    </학년>
    <학년 value="2학년">... 강감찬 ...</학년>
</학교>
```

- requests 라이브러리 설치 [VS-Code]-[Extensions] open in browser 설치

```
pip install requests   pip list: 설치된 라이브러리 확인
```

[c:]-[data]-[python]-[scraping] ex01.py ①

```
import requests
res = requests.get("http://naver.com")
print("응답코드: " , res.status_code)  #200이면 정상
if res.status_code == requests.codes.ok:
    print("정상입니다.")
else:
    print("문제가 생겼습니다. [에러코드: {0}]".format(res.status_code)
```

[c:]-[data]-[python]-[scraping] ex01.py ②

```
import requests
res = requests.get("http://naver.com")
res.raise_for_status()  #오류가 있으면 프로그램을 종료한다.
```

[c:]-[data]-[python]-[scraping] ex01.py ③

```
import requests
res = requests.get("http://google.com")
res.raise_for_status()
print(len(res.text))
print(res.text)

with open("mygoogle.html", "w", encoding="utf-8") as f:
    f.write(res.text)
```

02. 정규식 표현

정규표현식(Regular expressions) 은 특정한 규칙을 가진 문자열의 집합을 표현하는 데 사용하는 형식 언어이다. 복잡한 문자열의 검색과 치환을 위해 사용되며, Python 뿐만 아니라 문자열을 처리하는 모든 곳에서 사용된다.

[c:]-[data]-[python]-[scraping] ex02.py ①

```
import re

p = re.compile('ca.e')

# . (ca.e) : 하나의 문자를 의미  예) care, cafe, case (O) | caffe (X)
# ^ (^de) : 문자열의 시작  예) desk, destination (O) | fade (X)
# $ (se$) : 문자열의 끝  예) case, base (O) | face (X)

m = p.match("care")  #match 함수는 주어진 문자열이 처음부터 일치하는지 확인
if m:
    print(m.group())
else:
    print("매칭 되지 않음")

m = p.match("cafe")
if m:
    print(m.group())
else:
    print("매칭 되지 않음")

m = p.match("good care")
if m:
    print(m.group())
else:
    print("매칭 되지 않음")
```

[c:]-[data]-[python]-[scraping] ex02.py 실행결과

```
care
cafe
매칭 되지 않음
```

[c:]-[data]-[python]-[scraping] ex02.py ②

```
import re

p = re.compile("ca.e")

def print_match(m):
    if m:
        print(m.group())
    else:
        print("매칭 되지 않음")

m = p.match("case")
print_match(m)

m = p.match("cafe")
print_match(m)

m = p.match("good care")
print_match(m)

m = p.match("careless")
print_match(m)
```

[c:]-[data]-[python]-[scraping] ex02.py 실행결과

```
case
cafe
매칭 되지 않음
care
```

[c:]-[data]-[python]-[scraping] ex03.py ①

```
import re

p = re.compile("ca.e")

def print_match(m):
    if m:
        print("m.group():", m.group())  #일치하는 문자열 반환
        print("m.string:", m.string)  #입력받은 문자열
        print("m.start():", m.start())  #일치하는 문자열의 시작 index
        print("m.end():", m.end())  #일치하는 문자열의 끝 index
        print("m.span()", m.span())  #일치하는 문자열의 시작과 끝 index
        print()
    else:
        print("매칭 되지 않음")

#search 함수는 주어진 문자열 전체를 검색하여 일치하는 것이 있는지 확인
m = p.search("care")
print_match(m)

m = p.search("good care")
print_match(m)

m = p.search("careless")
print_match(m)
```

[c:]-[data]-[python]-[scraping] ex03.py 실행결과

```
m.group(): care
m.string: care
m.start(): 0
m.end(): 4
m.span() (0, 4)

m.group(): care
m.string: good care
m.start(): 5
m.end(): 9
m.span() (5, 9)

m.group(): care
m.string: careless
m.start(): 0
m.end(): 4
m.span() (0, 4)
```

[c:]-[data]-[python]-[scraping] ex03.py ②

```
import re

p = re.compile('ca.e')

#일치하는 모든 것을 리스트 형태로 출력
lst = p.findall("good care")
print(lst)

lst = p.findall("good care cafe")
print(lst)
```

[c:]-[data]-[python]-[scraping] ex03.py 실행결과

```
['care']
['care', 'cafe']
```

03. User Agent

일반 사용자가 아닌 크롤러 등이 사이트에 접근하려고 할 때 접근이 차단되도록 설정되어 있다. 따라서 사이트에게 사람인 척 해주어야한다. 사람인 척 하는 방법은 페이지에 접속할 때 User Agent 값을 넘겨주는 것이다. User agent는 접속하는 PC, 브라우저에 따라 달라진다.

```
[c:]-[data]-[python]-[scraping] ex04.py ①
```

```
import requests
res = requests.get("http://nadocoding.tistory.com")
res.raise_for_status()

with open("nadocoding.html", "w", encoding="utf-8") as f:  #구글에서 나도코딩 티스트리검색
    f.write(res.text)
```

```
[c:]-[data]-[python]-[scraping] ex04.py ②
```

```
import requests

url = "http://nadocoding.tistory.com"
headers = { "User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/101.0.4951.54 Safari/537.36" }  #구글에서 'What is user Agent String' 검색 (접속 브라우저마다 다르다)

res = requests.get(url, headers=headers)
res.raise_for_status()

with open("nadocoding.html", "w", encoding="utf-8") as f:
    f.write(res.text)
```

04. BeautifulSoup4

- **스크랩핑에 필요한 라이브러리들을 설치한다.**

```
pip install beautifulSoup4  스크랩핑을 위한 패키지
pip install lxml  html파서
```

1) CVG 무비차트

```
[c:]-[data]-[python]-[scraping] ex05.py ①
```

```
import requests
from bs4 import BeautifulSoup

url = "http://www.cgv.co.kr/movies/?lt=1&ft=0"
res = requests.get(url)
res.raise_for_status()

soup = BeautifulSoup(res.text, "lxml")
print(soup.title)
print(soup.title.get_text())
print(soup.a)  #처음 발견되는 a element를 반환
print(soup.a.attrs)  #a element의 속성들 정보 출력
print(soup.a["href"])  #a element의 href 속성 '값' 정보 출력
```

```
[c:]-[data]-[python]-[scraping] ex05.py ②
```

```
...
chart = soup.find("div", attrs={"class":"sect-movie-chart"})  #div이면서 class이름이 sect-movie-chart 검색
rank1 = chart.find("strong", attrs={"class": "rank"})
print(rank1.getText())
print(rank1.find_next_sibling("a"))  #같은 위치에 a태그 검색
```

[c:]-[data]-[python]-[scraping] ex05.py ③

```
...
title1 = rank1.parent.parent.find("strong", attrs={"class":"title"})
print(title1.getText())
```

- CGV 상용중인 영화목록

[c:]-[data]-[python]-[scraping] ex06.py

```
import requests
from bs4 import BeautifulSoup

url = "http://www.cgv.co.kr/movies/?lt=1&ft=0"
res = requests.get(url)
res.raise_for_status()

soup = BeautifulSoup(res.text, "lxml")

#CVG 전체 영화목록 출력
movies = soup.find_all("strong", attrs={"class":"title"})
for movie in movies:
    print(movie.get_text())
```

- CGV 상용중인 영화 제목과 예매사이트 목록

[c:]-[data]-[python]-[scraping] ex07.py ①

```
import requests
from bs4 import BeautifulSoup

url = "http://www.cgv.co.kr/movies/?lt=1&ft=0"
res = requests.get(url)
res.raise_for_status()

soup = BeautifulSoup(res.text, "lxml")
sect_movie = soup.find("div", attrs={"class":"sect-movie-chart"})
movies = sect_movie.find_all("li")

#CGV 영화제목과 예매사이트 목록 (Ctrl + 클릭)
for movie in movies:
    title = movie.find("strong", attrs={"class":"title"}).get_text()
    link = "http://www.cgv.co.kr" + movie.find("a", attrs={"class":"link-reservation"})["href"]
    print(title, link)
```

- CGV 평균 예매율 출력

[c:]-[data]-[python]-[scraping] ex07.py ②

```
...
total_percent =0;
for movie in movies:
    percent = movie.find("strong", attrs={"class":"percent"})
    percent=(percent.span.get_text())
    value=percent.replace("%","")
    total_percent += float(value)

print("평균 예매율:", total_percent / len(movies))
```

- 터미널에서 python 실행 후 작성가능

2) 지마켓

- '지마켓'에서 '노트북' 검색 후 1페이지에 '상품명', '가격', '평점', '피드백수' 출력

[c:]-[data]-[python]-[scraping] ex08.py ①

```
import requests
from bs4 import BeautifulSoup

url="https://browse.gmarket.co.kr/search?keyword=%EB%85%B8%ED%8A%B8%EB%B6%81&k=42&p=1"
res=requests.get(url)
res.raise_for_status()
soup=BeautifulSoup(res.text, "lxml")

items = soup.find_all("div", attrs={"class":"box__item-container"})
#print(len(items))
for item in items:
    name = item.find("span", attrs={"class":"text__item"})["title"]
    price= item.find("strong", attrs={"class":"text text__value"}).get_text()

    rate = item.find("span", attrs={"class", "image__awards-points"})
    if rate:
        rate=rate["style"]
        index = rate.find(":") + 1
        rate = rate[index:]
    else:
        rate = "평점없음"

    count = item.find("li", attrs={"class", "list-item__feedback-count"})
    if count:
        count=count.find("span", attrs={"class", "text"}).get_text()
        count=count.replace("(", "").replace(")", "")
    else:
        count="피드백없음"
    print(name, price, rate, count)
```

- '지마켓'에서 '노트북' 검색 후 1페이지에서 평점이 90점이상아고 피드백 수가 300이상인 '상품명', '가격', '평점', '피드백수' 출력

[c:]-[data]-[python]-[scraping] ex08.py ②

```
...
for item in items:
    name = item.find("span", attrs={"class":"text__item"})["title"]
    price= item.find("strong", attrs={"class":"text text__value"}).get_text()

    rate = item.find("span", attrs={"class", "image__awards-points"})
    if rate:
        rate=rate["style"]
        index = rate.find(":") + 1
        rate = rate[index:-1]
    else:
        continue

    count = item.find("li", attrs={"class", "list-item__feedback-count"})
    if count:
        count=count.find("span", attrs={"class", "text"}).get_text()
        count=count.replace("(","").replace(")","").replace(",","")
    else:
        continue

    if int(rate) >= 90 and int(count) >= 300:
        print(name, price, rate, count)
```

- '지마켓'에서 '노트북' 검색 후 1~5 페이지의 위 조건을 만족하는 상품정보 출력

[c:]-[data]-[python]-[scraping] ex09.py

```
import requests
from bs4 import BeautifulSoup

for i in range(1, 6):
    url="https://browse.gmarket.co.kr/search?keyword=%EB%85%B8%ED%8A%B8%EB%B6%81&k=42&p={}".format(i)
    res=requests.get(url)
    res.raise_for_status()
    soup=BeautifulSoup(res.text, "lxml")

    print("페이지:", i)
    items = soup.find_all("div", attrs={"class":"box__item-container"})
    for item in items:
        name = item.find("span", attrs={"class":"text__item"})["title"]
        price= item.find("strong", attrs={"class":"text text__value"}).get_text()
        link = item.find("a", attrs={ "class": "link__item" })["href"]

        rate = item.find("span", attrs={"class", "image__awards-points"})
        if rate:
            rate=rate["style"]
            index = rate.find(":") + 1
            rate = rate[index:-1]
        else:
            continue

        count = item.find("li", attrs={"class", "list-item__feedback-count"})
        if count:
            count=count.find("span", attrs={"class", "text"}).get_text()
            count=count.replace("(","").replace(")","").replace(",","")
        else:
            continue

        if int(rate)>=90 and int(count)>=300:
            print(f"제품명: {name}")
            print(f"가격: {price}")
            print(f"평점: {rate}% ({count})개")
            print(f"바로가기: {link}")
            print("-" * 100)
```

[c:]-[data]-[python]-[scraping] 04_bs4_gmarket_pages.py 실행결과

```
페이지: 1
제품명 :  15.6인치 17인치 LG그램 맥북 노트북 파우치 가방 P60
가격: 19,900
평점: 92% (351개)
바로가기: http://item.gmarket.co.kr/Item?goodscode=2102111886
----------------------------------------------------------------------------------------------------
제품명 :  .잘만 ZM-NS2000 노트북거치대 받침대 쿨링패드 쿨러
가격: 39,000
평점: 92% (323개)
바로가기: http://item.gmarket.co.kr/Item?goodscode=695224886
----------------------------------------------------------------------------------------------------
제품명 :  SMO-3550B 노트북 컴퓨터 PC USB 피시 무선 광 마우스
가격: 20,000
평점: 92% (755개)
바로가기: http://item.gmarket.co.kr/Item?goodscode=1100902613
----------------------------------------------------------------------------------------------------
제품명 :  잘만 ZM-NS1000 노트북거치대 받침대 쿨링패드 쿨러
가격: 23,000
평점: 96% (404개)
바로가기: http://item.gmarket.co.kr/Item?goodscode=694896609
----------------------------------------------------------------------------------------------------
...
```

3) 다음 영화 이미지 다운로드

- [다음]-[영화]-[역대 관객 순위]-[2022] 상위 5개이미지 다운로드 검색

[c:]-[data]-[python]-[scraping] ex10.py ①

```
import requests
from bs4 import BeautifulSoup

url="https://search.daum.net/search?w=tot&q=2022%...&DA=MOR&rtmaxcoll=MOR"
res=requests.get(url)
res.raise_for_status()
soup=BeautifulSoup(res.text, "lxml")

images = soup.find_all("img", attrs={ "class":"thumb_img" })

for idx, image in enumerate(images):
    image_url=image["src"]
    if image_url.startswith("//"):
        image_url += "https"

    print(image_url)
    image_res = requests.get(image_url)
    image_res.raise_for_status()

    with open("movie{}.jpg".format(idx+1), "wb") as f:
        f.write(image_res.content)

    if idx >=4:
        break
```

[c:]-[data]-[python]-[scraping] 04_bs4.py

```
https://search1.kakaocdn.net/thumb/R232x328.q85/?fname=http%3A%2F%2Ft1.daumcdn.net%2Fmovie%2F4e00e81f2b6f4d2eb65b3387240cc3
https://search1.kakaocdn.net/thumb/R232x328.q85/?fname=http%3A%2F%2Ft1.daumcdn.net%2Fmovie%2F5574fb2c20c844629aa9ad1d6043ee
https://search1.kakaocdn.net/thumb/R232x328.q85/?fname=http%3A%2F%2Ft1.daumcdn.net%2Fmovie%2F5afd212b68e34e61a964d969dd898e
https://search1.kakaocdn.net/thumb/R232x328.q85/?fname=http%3A%2F%2Ft1.daumcdn.net%2Fmovie%2F3673a8a0c5ff4f5c8c25cc959fd6985b
https://search1.kakaocdn.net/thumb/R232x328.q85/?fname=http%3A%2F%2Ft1.daumcdn.net%2Fmovie%2Fcab3b02a7b274bd6838b80a5e481fe
```

- 2018~2022년 관객순위 5위 이미지 다운로드

[c:]-[data]-[python]-[scraping] ex10.py ②

```
import requests
from bs4 import BeautifulSoup

for year in range(2018, 2023):
    url="https://search.daum.net/search?w=tot&q={}...&DA=MOR&rtmaxcoll=MOR".format(year)
    res=requests.get(url)
    res.raise_for_status()
    soup=BeautifulSoup(res.text, "lxml")

    images = soup.find_all("img", attrs={ "class":"thumb_img" })
    for idx, image in enumerate(images):
        ...
        with open("movie_{}_{}.jpg".format(year, idx+1), "wb") as f:
            f.write(image_res.content)

        if idx >=4:
            break
```

4) 웹 스크래핑 데이터 CSV 파일 저장

- [네이버]-[코스피 시가총액 순위]-[시가총액 상위종목 더보기]

[c:]-[data]-[python]-[scraping] ex11.py ①

```
import requests
from bs4 import BeautifulSoup

url="https://finance.naver.com/sise/sise_market_sum.nhn?sosok=0&page="

for page in range(1, 5):
    res = requests.get(url + str(page))
    res.raise_for_status()
    soup=BeautifulSoup(res.text, "lxml")

    data_rows=soup.find("table", attrs={"class", "type_2"}).find("tbody").find_all("tr")
    for row in data_rows:
        columns = row.find_all("td")
        #의미 없는 데이터 skip
        if len(columns) <=1 :
            continue

        data=[column.get_text().strip() for column in columns]
        print(data)
```

[c:]-[data]-[python]-[scraping] ex11.py ②

```
import csv
import requests
from bs4 import BeautifulSoup

url="https://finance.naver.com/sise/sise_market_sum.nhn?sosok=0&page="

filename = "시가총액1-200.csv"
f = open(filename, "w", encoding="utf-8-sig", newline="")  #자동 줄 바꿈 제거를 위해 newline="" 옵션을 준다.
writer =csv.writer(f)

title="N	종목명	현재가	전일비	등락률	액면가	시가총액	상장주식수	외국인비율거래량	PERROE	토론실".split("\t")
#print(type(title))

writer.writerow(title)
for page in range(1, 5):
    res = requests.get(url + str(page))
    res.raise_for_status()
    soup=BeautifulSoup(res.text, "lxml")

    data_rows=soup.find("table", attrs={"class", "type_2"}).find("tbody").find_all("tr")
    for row in data_rows:
        columns = row.find_all("td")
        #의미 없는 데이터 skip
        if len(columns) <=1 :
            continue

        data=[column.get_text().strip() for column in columns]
        #print(data)
        writer.writerow(data)
```

05. Selenium

selenium은 웹사이트 테스트를 위한 도구로 브라우저 동작을 자동화할 수 있다. selenium을 이용하는 웹크롤링 방식은 바로 이점을 적극적으로 활용하는 것이다. 프로그래밍으로 브라우저 동작을 제어해서 마치 사람이 이용하는 것 같이 웹페이지를 요청하고 응답을 받아올 수 있다.

1) Selenium 기본

- Selenium 라이브러리를 설치한다.

```
pip install selenium
```

- 아래 사이트로 이동하여 현재 크롬버전과 같은 프로그램을 다운로드 받은 후 압축을 해제하여 프로젝트 폴더에 저장한다.

```
https://chromedriver.chromium.org/downloads
```

- 네이버 사이트로 이동한 후 로그인 페이지로 이동한다.

[c:]-[data]-[python]-[scraping] ex12.py ①

```
from selenium import webdriver
from selenium.webdriver.common.by import By
import time

url = "https://naver.com"
browser = webdriver.Chrome()  #selenium 최신버전에서는 드라이버를 다운로드 받을 필요 없다.
browser.get(url)
time.sleep(1)

e = browser.find_element(By.CLASS_NAME, "MyView-module__link_login___HpHMW")
e.click()
browser.back()
browser.forward()
browser.refresh()
browser.back()

time.sleep(30)
```

- 네이버 검색창에서 '나도코딩' 검색후 'a' 태그의 링크값들을 출력한다.

[c:]-[data]-[python]-[scraping] ex12.py ②

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time

...
e = browser.find_element(By.ID, "query")
e.send_keys("나도코딩")
e.send_keys(Keys.ENTER)

es = browser.find_elements(By.TAG_NAME, "a")
for e in es:
    href=e.get_attribute("href")
    print(href)

time.sleep(30)
```

- **다음 검색창에서 '나도코딩' 검색한 후 검색버튼을 클릭한다.**

[c:]-[data]-[python]-[scraping] 05_selenium.py ③

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys

browser = webdriver.Chrome()
url = "https://www.daum.net/"

e = browser.find_element(By.ID, "q")
e.send_keys("나도코딩")
#e.send_keys(Keys.ENTER)
#마우스오른쪽 -> copy -> Copy XPath
e = browser.find_element(By.XPATH, "//*[@id='daumSearch']/fieldset/div/div/button[3]")
e.click()
time.sleep(30)
```

2) Selenium 심화

- **네이버 로그인**

[c:]-[data]-[python]-[scraping] ex13.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time

url = "https://www.naver.com/"
browser = webdriver.Chrome()

#네이버이동
browser.get(url)
time.sleep(3)

#2. 로그인 버튼 클릭
elem = browser.find_element(By.CLASS_NAME,"MyView-module__link_login___HpHMW")
elem.click()

#3. id, pw 입력
browser.find_element(By.ID, "id").send_keys("my_id")
browser.find_element(By.ID, "pw").send_keys("my_password")

#4. 로그인 버튼 클릭
browser.find_element(By.ID, "log.login").click()
time.sleep(3)

#5. id를 새로 입력
browser.find_element(By.ID, "id").clear()
browser.find_element(By.ID, "id").send_keys("my_new_id")

#6. html 정보 출력
print(browser.page_source)

#7. 브라우저 종료
#browser.close()  #현재 탭만 종료
browser.quit()  #전체 브라우저 종료
time.sleep(60)
```

3) Selenium 활용

- **[네이버]-[네이버 항공권]에서 이번달 25일, 26일 제주도 항공권 검색**

[c:]-[data]-[python]-[scraping] ex14.py ①

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

url = "https://m-flight.naver.com/"
browser = webdriver.Chrome()
browser.maximize_window()  #창 최대화
browser.get(url)
time.sleep(2)

#가는 날 선택
begin_date=browser.find_element(By.XPATH, '//button[text()="가는 날"]')
begin_date.click()

#가는 날 날짜선택 (이번 달 27일)
day27 = browser.find_elements(By.XPATH, '//b[text()="27"]')
day27[0].click()

#오는 날 날짜선택 (이번 달 28일)
day28 = browser.find_elements(By.XPATH, '//b[text()="28"]')
day28[0].click()

#도착지선택
arrival =browser.find_element(By.XPATH, '//b[text()="도착"]')
arrival.click()
time.sleep(2)

#국내선택
domestic = browser.find_element(By.XPATH, '//button[text()="국내"]')
domestic.click()

#제주선택
jeju = browser.find_element(By.XPATH, '//i[contains(text(), "제주국제공항")]')
jeju.click()

#항공권검색
search = browser.find_element(By.XPATH, '//span[contains(text(), "항공권 검색")]')
search.click()

#엘리먼트값을 가져 올 때까지 최대 30초 기다린다.
first='//div[@class="domestic_Flight__sK0eA result"]'
elem = WebDriverWait(browser, 30).until(EC.presence_of_element_located((By.XPATH, first)))
print(elem.text)

browser.quit()
```

[c:]-[data]-[python]-[scraping] ex14.py ②

```
...
def wait_until(xpath_str):
    WebDriverWait(browser, 30).until(EC.presence_of_element_located((By.XPATH, xpath_str)))
...
#가는 날 선택
wait_until('//button[text()="가는 날"]')
begin_date=browser.find_element(By.XPATH, '//button[text()="가는 날"]')
begin_date.click()
...
```

- [구글]-[송중기]-[이미지] 검색 후 제목 출력

[c:]-[data]-[python]-[scraping] ex15.py

```
import requests
from bs4 import BeautifulSoup

url = "https://www.google.com/search?q=%EC%86%A1%EC%A4%91%EA%B8%B0&hl=ko&sxsrf=A..."
headers = { "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Safari/537.36", "Accept-Language" : "ko-KR, ko"}
res = requests.get(url, headers=headers)
res.raise_for_status()
soup = BeautifulSoup(res.text, "lxml")

images = soup.find_all("div", attrs={"class": "isv-r PNCib MSM1fd BUooTd"})
print(len(images))

for idx, image in enumerate(images):
    title = image.find("div", attrs={"class":"zbRPDe M2qv4b P4HtKe"}).get_text()
    print(idx+1 , title)
```

[c:]-[data]-[python]-[scraping] ex16.py ①

```
from selenium import webdriver
import time

browser = webdriver.Chrome()
browser.maximize_window()
url = "https://www.google.com/search?sca_esv=559279238&sxsrf=AB5stBiMr67JbwP_QILyccNbTeDnJ_iTIQ:169275647951..."
browser.get(url)

#browser.execute_script("window.scrollTo(0, 1080)")  #지정한 위치로 스크롤 내리기
#browser.execute_script("window.scrollTo(0, document.body.scrollHeight)")  #화면 가장 아래로 스크롤 내리기

interval = 2
prev_height = browser.execute_script("return document.body.scrollHeight")

#페이지의 끝일 때 까지 반복 수행
while True:
    browser.execute_script("window.scrollTo(0, document.body.scrollHeight)")
    time.sleep(interval)  #페이지 로딩 대기

    curr_height = browser.execute_script("return document.body.scrollHeight")
    if curr_height == prev_height:
        break

    prev_height = curr_height

print("스크롤 완료")

import requests
from bs4 import BeautifulSoup
soup = BeautifulSoup(browser.page_source, "lxml")

images = soup.find_all("div", attrs={"class":"isv-r PNCib MSM1fd BUooTd"})
print(len(images))

for idx, image in enumerate(images):
    title = image.find("div", attrs={"class":"zbRPDe M2qv4b P4HtKe"}).get_text()
    print(idx+1 , title)

time.sleep(60*60)
```

---

4) Headless 크롬

크롬 브라우저를 띄우지 않고 빠르게 스크랩핑 작업이 가능하다. (브라우저를 띄운 경우 메모리도 많이 필요하다)

[c:]-[data]-[python]-[scraping] ex16.py ②

```
...

options = webdriver.ChromeOptions()
options.headless = True
options.add_argument("window-size=1920x1080")
browser = webdriver.Chrome(options=options)
...

print("스크롤 완료")
browser.get_screenshot_as_file("google_images.png")
...
```

HeadlessChrome이고 user agent값이 설정되지 않은 경우 User agent값이 날아가서 브라우저의 접속을 막을 수 있다.

[c:]-[data]-[python]-[scraping] ex17.py ①

```
from selenium import webdriver
from selenium.webdriver.common.by import By

options = webdriver.ChromeOptions()
options.headless = True
options.add_argument("window-size=1920x1080")

browser = webdriver.Chrome(options=options)
browser.maximize_window()

#구글에서 user agent string 검색후 what is my user agent 클릭
url="https://www.whatismybrowser.com/detect/what-is-my-user-agent/"
browser.get(url)

detected_value = browser.find_element(By.ID, "detected_value")
print(detected_value.text)

browser.quit()
```

[c:]-[data]-[python]-[scraping] ex17.py 실행결과

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/102.0.5005.62
Safari/537.36
```

user agent값이 설정된 경우에는 user agent값이 정상적 적용되어 출력된다.

[c:]-[data]-[python]-[scraping] ex17.py ②

```
from selenium import webdriver
from selenium.webdriver.common.by import By

options = webdriver.ChromeOptions()
options.headless = True
options.add_argument("window-size=1920x1080")
options.add_argument("user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64)... Chrome/102.0.5005.62 Safari/537.36")
...
```

[c:]-[data]-[python]-[scraping] 05_headless_usergent.py 실행결과

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.62 Safari/537.36
```

06. 프로젝트

무분별한 웹 크롤링이나 웹 스크래핑은 대상 서버에 부하가 걸려 계정/IP가 차단된다. 또한 이미지나 텍스트 등 데이터 무단 활용 시 저작권 등 침해요소가 있어 법적 제재를 받을 수 있다. robots.txt 파일은 법적 효력은 없지만 대상 사이트에서 크롤링을 금지하는 권고 사항이다.

- **네이버 부동산 사이트에서 '청라자이' 검색 결과를 출력**

출력 결과

```
========== 정보 1 ==========
0.청라자이
인천시 서구 청라동
아파트884세대총19동2010.06.123.5m² ~278.12m²
========== 정보 2 ==========
1.청라힐스자이
대구시 중구 남산동
아파트분양권947세대총13동2023.01.77.33m² ~129.68m²
========== 정보 3 ==========
2.청라파크자이더테라스(2블럭)
인천시 서구 청라동
아파트376세대총18동2016.03.102.15m² ~112.64m²
========== 정보 4 ==========
3.청라파크자이더테라스(1블럭)
인천시 서구 청라동
아파트270세대총17동2016.03.102.32m² ~112.83m²
```

[c:]-[data]-[python]-[scraping] ex18.py

```
from selenium import webdriver
from seleninm.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time

def browser_execute():
    browser = webdriver.Chrome()
    browser.maximize_window()

    url = "https://land.naver.com/"
    browser.get(url)

    e = browser.find_element(By.ID, "queryInputHeader")
    e.send_keys("청라자이")
    e.send_keys(Keys.ENTER)
    time.sleep(2)

    from bs4 import BeautifulSoup
    soup = BeautifulSoup(browser.page_source, "lxml")

    #with open("realty.html", "w", encoding="utf8") as f:
    #    f.write(soup.prettify())

    es = soup.find_all("div", attrs={"class":"item"})
    print("검색결과:{0}건".format(len(es)))

    for index, e in enumerate(es, start=1):
        title = e.find("div", attrs={"class":"title"}).get_text()
        address = e.find("div", attrs={"class":"address"}).get_text()
        info = e.find("div", attrs={"class":"info_area"}).get_text()
        print("========== 정보 {} ==========".format(index))
        print(f"{ index }.{ title }")
        print(address)
        print(info)

    #브라우저가 닫히지 않는다.
    while True:
        pass

browser_execute()
```

- [네이버 검색]-[서울 날씨] 검색 후 날씨정보 아래와 같이 출력

출력 결과

```
[오늘의 날씨]
------------------------------------------------
어제보다 1.1° 낮아요 비
현재: 24.3° , 체감: 27.3° , 습도: 8.0mm
오전 강수확률: 80%
오후 강수확률: 70%
미세먼지: 좋음
초미세먼지: 좋음
------------------------------------------------
```

[c:]-[data]-[python]-[scraping] ex19.py

```
import requests
from bs4 import BeautifulSoup

def create_suop(url):
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, "lxml")
    return soup

def scrape_weather():
    print("[오늘의 날씨]")
    url = "https://search.naver.com/search.naver?where=nexearch&sm=top_hty&fbm=0&ie=utf8&query=오늘의 날씨"
    soup = create_suop(url)

    summary = soup.find("p", attrs={ "class":"summary" }).get_text()

    #현재 온도
    temp_text = soup.find("div", attrs={ "class":"temperature_text" })
    cur_temp = temp_text.strong.contents[1]
    cur_temp +=temp_text.find("span", attrs={"class":"celsius"}).get_text()

    #체감온도, 습도
    sum_list = soup.find("dl", attrs={"class":"summary_list"}).find_all("dd", attrs={"class":"desc"})

    #오전, 오후 강수량
    cell_weather = soup.find("div", attrs={"class":"cell_weather"}).find_all("span", attrs={"class":"rainfall"})

    #클래스명이 className1이거나 className2 검색 attrs={"class":["className1", "className2"]}
    #클래스명이 className이고 아이디가 idName 검색 attrs={"class":"className", "id":"idName"}
    #클래스명이 className이고 텍스트내용이 미세먼지이거나 초미세먼지 검색 attrs={"class":"className"}, text=["미세먼지", "초미세먼지"]

    #미세먼지
    item_today= soup.find("li", attrs={"class":"item_today level2"}).find_all("span", attrs={"class":"txt"})
    print("-" * 50)
    print(summary)
    print("현재: {}, 체감: {}, 습도: {}".format(cur_temp, sum_list[0].get_text(), sum_list[1].get_text()))
    print("오전 강수확률: {}".format(cell_weather[0].get_text()))
    print("오후 강수확률: {}".format(cell_weather[1].get_text()))
    print("미세먼지: {}".format(item_today[0].get_text()))
    print("초미세먼지: {}".format(item_today[0].get_text()))
    print("-" * 50)

if __name__ == "__main__":
    scrape_weather()
```

- **[네이버 뉴스]-[IT/과학] 검색 후 뉴스 정보 아래와 같이 3까지만 출력**

출력 결과

```
[IT/과학 헤드라인 뉴스]
----------------------------------------------------------------------------------------------------------------------------------------
1. "반도체 인재" 양성 급한데 4대 과기원 반도체학과 확대
  (https://...)
2. 7월부터 "메타버스 근무제" 시행 카카오
  (https://...)
3. AI로 만성질환관리 돕는다. SKT 국민건강보험공단과 ...
  (https://...)
----------------------------------------------------------------------------------------------------------------------------------------
```

[c:]-[data]-[python]-[scraping] ex19.py

```
import requests
from bs4 import BeautifulSoup

def create_suop(url):
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, "lxml")
    return soup

def scrape_weather():
    print("[오늘의 날씨]")
    url = "https://search.naver.com/search.naver?where=nexearch&sm=top_hty&fbm=0&ie=utf8&query=오늘의 날씨"
    soup = create_suop(url)
    summary = soup.find("p", attrs={ "class":"summary" }).get_text()

    #현재 온도
    temp_text = soup.find("div", attrs={ "class":"temperature_text" })
    cur_temp = temp_text.strong.contents[1]
    cur_temp +=temp_text.find("span", attrs={"class":"celsius"}).get_text()

    #체감온도, 습도
    sum_list = soup.find("dl", attrs={"class":"summary_list"}).find_all("dd", attrs={"class":"desc"})

    #오전, 오후 강수량
    cell_weather = soup.find("div", attrs={"class":"cell_weather"}).find_all("span", attrs={"class":"rainfall"})
    ...

def scrcap_headline_news():
    print("[IT/과학 헤드라인 뉴스]")
    url="https://news.naver.com/main/main.naver?mode=LSD&mid=shm&sid1=105"
    soup = create_suop(url)

    news_list=soup.find("ul", attrs={"class":"sh_list"}).find_all("li", limit=3)
    #print(len(news_list))
    for index, news in enumerate(news_list):
        item=news.find("div", attrs={"class":"sh_text"})
        title=item.a.get_text()
        link=item.a["href"]
        print("{}. {}".format(index+1, title))
        print("(링크: {})".format(link))
    print()

if __name__ == "__main__":
    #scrape_weather()
    scrcap_headline_news()
```

- **해커스의 오늘의 영어회화 정보 출력**

출력 결과

```
[오늘의 영어 회화]
(영어지문)
Mrs. Flores : And just what is your explanation?
Rob : Actually, I've completed the research but I still need time to write up the report.
Mrs. Flores : And how long do you expect that to take?
Rob : Just another few days.
(한글지문)
Mrs. Flores : 이유가 무엇인가요?
Rob : 사실 연구를 끝냈습니다만, 보고서를 작성하기 위한 시간이 더 필요해요.
Mrs. Flores : 얼마나 오랫동안 걸릴 거 같아요?
Rob : 단 며칠이요.
```

[c:]-[data]-[python]-[scraping] ex19.py

```
...
def scrap_english():
    from selenium import webdriver
    from selenium.webdriver.common.by import By
    import time

    options = webdriver.ChromeOptions()
    options.headless = True
    options.add_argument("window-size=1920x1080")
    browser = webdriver.Chrome(options=options)

    url="https://www.hackers.co.kr/?c=s_eng/eng_contents/I_others_english&keywd=freelec_..."  #[해커스영어]-[무료강의]-[영어회화강의]
    browser.get(url)
    browser.maximize_window()
    time.sleep(3)

    #반복수행
    interval = 2
    prev_height = browser.execute_script("return document.body.scrollHeight")
    while True:
        browser.execute_script("window.scrollTo(0, document.body.scrollHeight)")
        #페이지 로딩 대기
        time.sleep(interval)

        curr_height = browser.execute_script("return document.body.scrollHeight")
        if curr_height == prev_height:
            break
        prev_height = curr_height

    # with open("english.html", "w", encoding="utf-8") as f:
    #     f.write(soup.prettify())
    soup=BeautifulSoup(browser.page_source, "lxml")
    sentences=soup.find_all("span", attrs={"class":"conv_sub"})

    print("[오늘의 영어회화]")
    print("(영어지문)")
    for sentence in sentences[len(sentences)//2:]:  #8문장이 있다고 가정할 때, index 기준 4~7까지
        print(sentence.get_text().strip())

    print("(한글지문)")
    for sentence in sentences[:len(sentences)//2]:  #8문장이 있다고 가정할 때, index 기준 0~3까지
        print(sentence.get_text().strip())

if __name__ == "__main__":
    #scrape_weather()
    #scrcap_headline_news()
    scrap_english()
```

07. 웹페이지

- **flask 라이브러리 설치**

```
pip install flask
```

- **웹서버 실행 (flask run)**

```
python app.py
```

app.py

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html', title='홈페이지')

if __name__ == '__main__':
    app.run(port=5000, debug=True)
```

[templates] header.html

```
<div class="my-5">
    <h1 class="text-center mb-5">스크랩핑/크롤링</h1>
</div>
```

[templates] footer.html

```
<div class="my-5">
    <h4 class="text-center">Copyright 2023 홍길동 All rights reserved.</h4>
</div>
```

[templates] base.html

```
<html>
<head>
    <link rel="stylesheet" href="/static/css/style.css"/>
    <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/handlebars.js/3.0.1/handlebars.js"></script>
    <title>{{title}}</title>
</head>
<body>
    <div class="container">
        {%include 'header.html'%}
        {%block main_area%}
        {%endblock%}
        {%include 'footer.html'%}
    </div>
</body>
</html>
```

[templates] index.html

```
{%extends 'base.html'%}
{%block main_area%}
    <h1>홈페이지</h1>
{%endblock%}
```

- movieRoute 등록

app.py

```
from flask import Flask, render_template
from movieRoute import movie

app = Flask(__name__)
app.register_blueprint(movie, url_prefix='/movie')

@app.route('/')
def index():
    return render_template('index.html', title='홈페이지')

if __name__ == '__main__':
    app.run(port=5000, debug=True)
```

movieRout.py

```
from flask import Blueprint, render_template
movie=Blueprint('movie', __name__)

import requests
from bs4 import BeautifulSoup

url = "http://www.cgv.co.kr/movies/?lt=1&ft=0"
res = requests.get(url)
res.raise_for_status()
soup = BeautifulSoup(res.text, 'lxml')
movies = soup.find_all('strong', attrs={'class':'title'})

@movie.route('/list.json')
def list_json():
    json_movies=[]
    for movie in movies:
        json_movies.append({'title':movie.getText()})
    return json_movies

@movie.route('/list')
def list():
    return render_template('movie.html')
```

[templates] movie.html

```
{%extends 'base.html'%}
{%block main_area%}
    <h1>뮤비차트</h1>
{%endblock%}
```

- **스크랩핑 결과 JSON파일로 저장**

jsonWrite.py

```
import json
import requests
from bs4 import BeautifulSoup

url = "http://www.cgv.co.kr/movies/?lt=1&ft=0"
res = requests.get(url)
res.raise_for_status()

soup = BeautifulSoup(res.text, 'lxml')
movies = soup.find_all('strong', attrs={'class':'title'})

print_movies=[]
for movie in movies:
    print_movies.append({'title':movie.getText()})
    print(movie.getText())

with open('./movie.json', 'w', encoding='utf-8') as file:
    json.dump(print_movies, file, indent="\t", ensure_ascii=False)
```

- **JSON 파일 읽기**

jsonRead.py

```
import json
with open('./movie.json', 'r',  encoding='UTF8') as file:
    json_data = json.load(file)
    print(type(json_data))

    for item in json_data:
        print(item['title'])
```

- **로컬 JSON파일 html에서 읽기**

test.json

```
testData = [
    {
      "name": "홍길동",
      "age": 14
    },
    {
      "name": "최진우",
      "age": 31
    }
  ]
```

---

test.html

```
<script src="test.json" type="text/javascript"></script>
<script>
   let testData = JSON.parse(JSON.stringify(TestFile));
   console.log(testData);
</script>
```
