Crawling Python Programming

1일차 : Crawling

1. Crawling
2. Requests
3. Beautiful Soup

2일차 : Selenium

1. Exam – Beautiful Soup
2. Selenium
3. Selenium : find_element

3일차 : 심화

1. 로그인 자동화
2. 파싱 작업 심화
3. 이미지 가져오기

4일차 : 심화

1. 본문 내용 가져오기
2. Iframe 가져오기
3. Requests로 서버 요청하기

1일차 수업 내용

- Crawling 이론
- HTML 계층구조와 BeautifulSoup
- Request - 웹사이트에서 HTML 가져오기
- Request - URL을 이용한 데이터 파싱
- Request – BeautifulSoup와 Requests 사용방법 연습하기
- Request - URL을 이용한 데이터 파싱 심화
- 이미지 저장하기

! Crawling

Crawling

크롤링(Crawling)이란 Web에서 돌아다니면서 원하는 정보를 수집하는 행위

정적 크롤링

- 특별한 절차 없이 URL을 통해 데이터 수집
- 속도가 빠르다
- 수집 대상에 한계가 존재한다
- 사용가능 라이브러리 (외장 모듈)
  requests

동적 크롤링

- 특별한 절차가 필요함
- 속도가 느리다
- 수집 대상에 한계가 거의 존재하지 않는다
- 사용가능 라이브러리 (외장 모듈)
  selenium

작업 절차

정적 : requests

동적 : selenium

BetutifulSoup

Pandas

BeautifulSoup 기본 예제 및 계층구조

```
from bs4 import BeautifulSoup
```

BeautifulSoup 모듈 연결

```
html = """
<body>
<div>
    <div class='snack'>
        <p id='first'> 양파링</p>
        <p id='second'> 새우깡</p>
        <p> 초코파이</p>
        <p class='sweet'> 맛동산 </p>

    <div class='icecream'>
        <p> 빵빠레</p>
        <p id='third'> 죠스바</p>
        <p> 꼬깔콘</p>
        <p class='sweet'> 쭈쭈바</p>
    </div>
</div>
</body>
"""
```

Web html 소스 코드

※ 원하는 페이지
크롤링 모듈 사용
- Requests 등
- Selenium 등

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

BeautifulSoup에 코드 연결

```
# 양파링

path = "div > div.snack > p#first"

soup.select(path)[0].text
```

탐색
예시) 리스트 형식으로 가져온다
>>> soup.select()

```
[<p id="first"> 양파링</p>]
```

```
# 죠스바
path = "div > div.icecream > p#third"

soup.select(path)[0].text
```

```
# 맛동산, 쭈꾸바

path = "div > p.sweet"

for i in soup.select(path):
    print(i.text)
```

```
# snack의 모든 이름을 출력

path = "div > div.snack > p"

for i in soup.select(path):
    print(i.text)
```

```
for i in soup.select(path):
    print(i)

for i in soup.select(path):
    print(i.text)
```

```
<p class="sweet"> 맛동산 </p>
<p class="sweet"> 쭈쭈바</p>
 맛동산
 쭈쭈바
```

웹사이트에서 원하는 html 파일 가져오기

```
# nate에서 검색 후 뉴스페이지에서
# 제목 : ...
# 서머리 : .....

import requests
from bs4 import BeautifulSoup

select = input("제목을 작성하시오 : ")

#웹사이트 연결하기
url = f"https://search.daum.net/nate?w=news&nil_search=btn&DA=NTB&enc=utf8&cluster=y&cluster_page=1&q={select}"
res = requests.get(url)

# BeautifulSoup에 html 파싱하기
soup = BeautifulSoup(res.text, 'html.parser')

path = "div.c-item-content > div.item-bundle-mid > div"
#dnsColl > div:nth-child(1) > ul > li:nth-child(1) > div.c-item-content > div.item-bundle-mid


main_path = "div.c-item-content > div.item-bundle-mid"
title_path = "div.c-item-content > div.item-bundle-mid > div.item-title"
contents_path = "div.c-item-content > div.item-bundle-mid > div.item-contents"

for item in soup.select(main_path):
    title = item.select(title_path)
    summary = item.select(contents_path)

    print(title[0].text)
    print(summary[0].text)
    print()
```

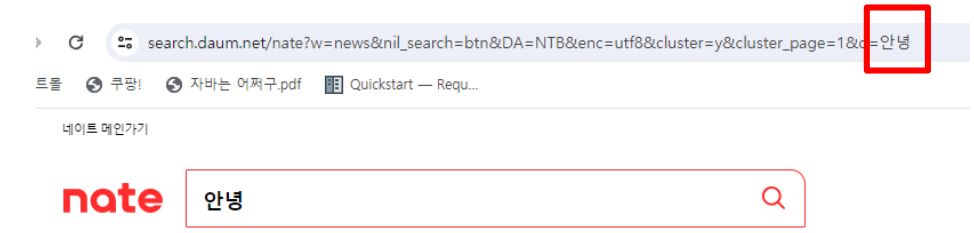
url : 웹사이트 고유 주소 (도메인)

request.get(url) 함수를 통해 웹 사이트의 소스 코드를 가져온다.

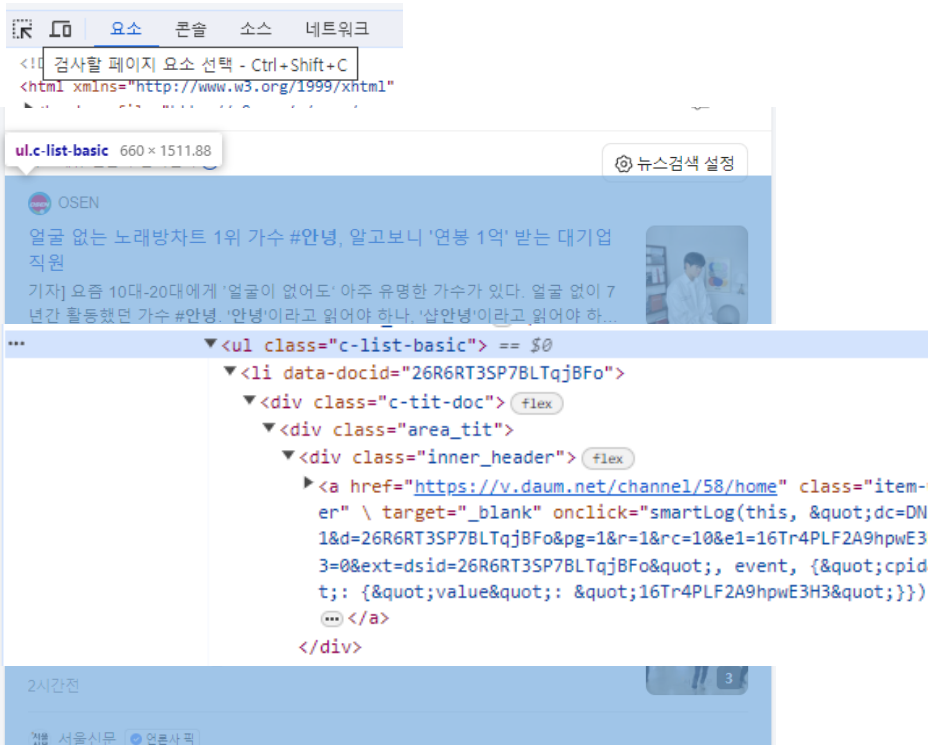
```
제목 :    김의성 '반갑습니다'[★포토]
서머리 :    [스타뉴스 | 김창현 기자] 김의성 '반갑습니다'[★포토]

제목 :    강기영,'반갑습니다' [사진]
서머리 :    [OSEN=김성락 기자] 27일 오후 서울 광진구 롯데시네마 건

제목 :    홍사빈 '반갑습니다'
서머리 :    제44회 청룡영화상 레드카펫 행사가 24일 오후 서울 여의

제목 :    최경환, 경산서 총선 본격 몸풀기 "시민여러분 반갑습니다"
서머리 :    최경환 전 경제부총리가 22일 자신의 옛 지역구인 경북 경
```

Requests - URL 접근 데이터 파싱

```
import requests
from bs4 import BeautifulSoup

mainList = []
mainDic = {}

select = "안녕" #input("제목을 작성하시오 : ")

url = f"https://search.daum.net/nate?w=news&nil_search=btn&DA=NTB&enc=utf8&cluster=y&cluster_page=1&q={select}"

res = requests.get(url)
soup = BeautifulSoup(res.text, 'html.parser')

main_path = "#dnsColl > div > ul > li"
company_path = "div.c-tit-doc > div.area_tit > div > a > strong > span"
#company_path = "div.c-tit-doc > div.area_tit > div > a"
title_path = "div.item-title"

contents_path = "div.item-contents > p"
content_href_path = "div.item-title > strong > a"
today_path = "div.c-item-content > div.item-bundle-mid > div.item-contents > span"
#dnsColl > div:nth-child(1) > ul > li:nth-child(1) > div.c-item-content > div.item-bundle-mid > div.item-title > strong > a
soup.select(main_path)

for i, item in enumerate(soup.select(main_path)):
    company = item.select(company_path)
    title = item.select(title_path)
    summary = item.select(contents_path)
    today = item.select(today_path)
    content_href = item.select(content_href_path)

    mainDic[i] = {"title" : title[0].text, "summary" : summary[0].text, "company" : company[0].text, "today" :
today[0].text, "href" : content_href[0]['href']}
    mainList.append({"title" : title[0].text, "summary" : summary[0].text, "company" : company[0].text, "today" :
today[0].text, "href" : content_href[0]['href']})

for key,value in mainDic.items():
    print(key + 1, "번")
    for key, item_value in value.items():
        print( key , " : ", item_value)

    print()
```



- URL에서 검색창이 뜨는 곳 탐색
- F12 및 개발자 도구를 열어 각각의 리스트를 찾는다.
- 왼쪽 위에 검사할 페이지 요소를 선택 코드를 찾는다.
- 계층구조 경로를 찾아 데이터를 파싱한다.



Requests – 토마토 홈페이지 파싱하기

토마토에 있는 영화 순위를 파싱한다.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
#select = "안녕" #input("제목을 작성하시오 : ")

url =
f"https://www.rottentomatoes.com/browse/movies_at_home/sort:popular"

res = requests.get(url)
soup = BeautifulSoup(res.text, 'html.parser')

list_path = "#main-page-content > div.discovery > div.discovery-grids-
container > div > div > div"
title_path = "tile-dynamic > a > span.p--small"
tomato_path = "div > div > span.percentage"
popcon_path = "div > div > span.percentage"
streaming_day_path = "tile-dynamic > a > span.smaller"

list_path = "div.flex-container"

list_up = soup.select(list_path)
```

```
movie_list = []

for item in list_up:

    # strip 여백 없애주는
    title = item.select('.p--small')[0].text.strip()
    streaming_day = item.select('.smaller')[0].text.strip()
    audiencescore = item.select('a')[0].select('score-pairs-
deprecated')[0]['audiencescore']
    criticsscore = item.select('a')[0].select('score-pairs-
deprecated')[0]['criticsscore']

    movie_list.append({"title" : title, "audiencescore" : audiencescore ,
"criticsscore" : criticsscore , "streaming_day" : streaming_day})

news_df = pd.DataFrame(movie_list)
news_df
```

Requests – 토마토 홈페이지 파싱하기

토마토에 있는 영화 순위를 파싱한다.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
#select = "안녕" #input("제목을 작성하시오 : ")

url =
f"https://www.rottentomatoes.com/browse/movies_at_home/sort:popular"

res = requests.get(url)
soup = BeautifulSoup(res.text, 'html.parser')

list_path = "#main-page-content > div.discovery > div.discovery-grids-
container > div > div > div"
title_path = "tile-dynamic > a > span.p--small"
tomato_path = "div > div > span.percentage"
popcon_path = "div > div > span.percentage"
streaming_day_path = "tile-dynamic > a > span.smaller"

list_path = "div.flex-container"

list_up = soup.select(list_path)
```

```
movie_list = []

for item in list_up:

    # strip 여백 없애주는
    title = item.select('.p--small')[0].text.strip()
    streaming_day = item.select('.smaller')[0].text.strip()
    audiencescore = item.select('a')[0].select('score-pairs-
deprecated')[0]['audiencescore']
    criticsscore = item.select('a')[0].select('score-pairs-
deprecated')[0]['criticsscore']

    movie_list.append({"title" : title, "audiencescore" : audiencescore ,
"criticsscore" : criticsscore , "streaming_day" : streaming_day})

news_df = pd.DataFrame(movie_list)
news_df
```

판다를 쓰면 다음과 같이 리스트에 데이터가 쌓인다.

	title	audiencescore	criticsscore	streaming_day
0	Oppenheimer	91	93	Streaming Nov 21, 2023
1	Leo	94	81	Streaming Nov 21, 2023
2	The Holdovers	90	96	Streaming Nov 28, 2023
3	The Killer	59	85	Streaming Nov 10, 2023
4	Stamped From the Beginning	69	100	Streaming Nov 20, 2023
5	Five Nights at Freddy's	87	30	Streaming Oct 27, 2023
6	Good Burger 2	73	58	Streaming Nov 22, 2023
7	Mister Organ	92	95	Streaming Nov 21, 2023
8	The Creator	76	66	Streaming Nov 14, 2023

! Requests – 토마토 홈페이지 파싱하기 - 모범답안

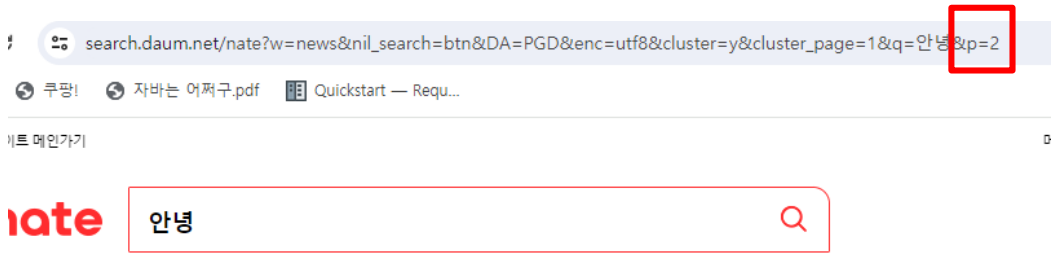
토마토에 있는 영화 순위를 파싱한다.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
#select = "안녕" #input("제목을 작성하시오 : ")

url = f"https://www.rottentomatoes.com/browse/movies_at_home/sort:popular"

res = requests.get(url)
soup = BeautifulSoup(res.text, 'html.parser')

list_path = "div.flex-container"

list_up = soup.select(list_path)

movie_list = []

for item in list_up:
    title = item.select('.p--small')[0].text.strip()
    streaming_day = item.select('.smaller')[0].text.strip()
    audiencescore = item.select('a')[0].select('score-pairs-deprecated')[0]['audiencescore']
    criticsscore = item.select('a')[0].select('score-pairs-deprecated')[0]['criticsscore']

    movie_list.append({"title" : title, "audiencescore" : audiencescore , "criticsscore" : criticsscore , "streaming_day" : streaming_day})

movies_df = pd.DataFrame(movie_list)

movies_df

movies_df.to_csv("movies.csv")
```

Requests – 네이트 검색 결과 10페이지 csv로 가져오기

토마토에 있는 영화 순위를 파싱한다.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

select = "안녕"

resultList = []

for i in range(1,11):

    url =
f"https://search.daum.net/nate?w=news&nil_search=btn&DA=NTB&enc=utf8&cluster=y&cluster_page=1&q={select
}&p={i}"
    res = requests.get(url)

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

    list_path = "#dnsColl > div:nth-child(1) > ul > li"

    company_path = "div.c-tit-doc > div.area_tit > div > a > strong > span"
    title_path = "div.item-title"

    contents_path = "div.item-contents > p"
    content_href_path = "div.item-title > strong > a"
    today_path = "div.c-item-content > div.item-bundle-mid > div.item-contents > span"

    soup.select(main_path)

    for i, item in enumerate(soup.select(main_path)):
        company = item.select(company_path)
        title = item.select(title_path)
        summary = item.select(contents_path)
        today = item.select(today_path)
        content_href = item.select(content_href_path)

        resultList.append({"title" : title[0].text, "summary" : summary[0].text, "company" :
company[0].text, "today" : today[0].text, "href" : content_href[0]['href']})
```



페이지를 넘길 경우 위와 같이 URL이 추가된다.

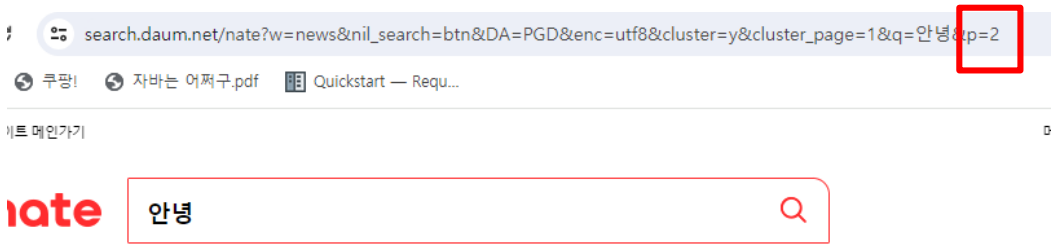
```
for item in resultList:
    print(item)

result_df = pd.DataFrame(resultList)

result_df.to_csv("NateResult.csv")
```

리스트에 데이터를 담아서 CSV에 데이터를 담는다.

Requests – 네이트 검색 결과 10페이지 csv로 가져오기

```
# 네이트 검색 결과 뉴스페이지에서 검색어 결과 10 페이지까지 크롤링 해서 csv file로 저장하시오
import requests
from bs4 import BeautifulSoup
import pandas as pd

select = "안녕"

resultList = []

for i in range(1,11):

    url =
f"https://search.daum.net/nate?w=news&nil_search=btn&DA=NTB&enc=utf8&cluster=y&cluster_page=1&q={select}&p={i}"
    res = requests.get(url)

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

    main_path = '#dnsColl > div > ul > li'

    list_path = "#dnsColl > div:nth-child(1) > ul > li"

    company_path = "div.c-tit-doc > div.area_tit > div > a > strong > span"
    title_path = "div.item-title"

    contents_path = "div.item-contents > p"
    content_href_path = "div.item-title > strong > a"
    today_path = "div.c-item-content > div.item-bundle-mid > div.item-contents > span"

    soup.select(main_path)

    for i, item in enumerate(soup.select(main_path)):
        company = item.select(company_path)
        title = item.select(title_path)
        summary = item.select(contents_path)
        today = item.select(today_path)
        content_href = item.select(content_href_path)

        resultList.append({"title" : title[0].text, "summary" : summary[0].text, "company" : company[0].text,
"today" : today[0].text, "href" : content_href[0]['href']})

for item in resultList:
    print(item)

result_df = pd.DataFrame(resultList)

result_df.to_csv("NateResult.csv")

result_df
```

페이지별로 반복해서 내용을 파싱한다.



페이지를 넘길 경우 위와 같이 URL이 추가된다.

```
for item in resultList:
    print(item)

result_df = pd.DataFrame(resultList)

result_df.to_csv("NateResult.csv")
```

리스트에 데이터를 담아서 CSV에 데이터를 담는다.

```
,title,summary,company,today,href
0,"  얼굴 없는 노래방차트 1위 가수 #안녕, 알고보니 '연봉 1억' 받는 대기업 직원  "," 기자] 요즘 10대-20대에
1,"  ""바가지요금 안녕""… 외국인 관광객용 택시 호출 앱 출시  ", 외국인 관광객 전용 택시 호출 앱 '타바'. 서
2,  "웃으며 춤추듯 운동하니 관절통증도 우울증도 안녕"  ," 1일 경북 구미 새마을운동테마공원 부속동에서 열린
3,  [단독] 우울증에 신음하는 농어촌… 진료받을 정신과가 없어요 [대한민국 정신건강리포트-당신의 마음은 안녕하
```

Requests – 이미지 저장하기

```python
import requests
from bs4 import BeautifulSoup
import pandas as pd

url = f"https://xkcd.com/2672/"

res = requests.get(url)
soup = BeautifulSoup(res.text, 'html.parser')

path = "#comic > a > img"
# 경로 가져오기
tag = soup.select(path)
img_url = 'https:' + tag[0]['src']

# 이진 파일 내용 가져오기
res = requests.get(img_url)

# 파일 입출력 가져오기
#f = open('catoon.jpg','wb')
#f.write(res.content)
#f.close()

# 파일 입출력 가져오기 [안전]
f = open('catoon.jpg','wb')
# 100 바이트씩 읽어서 저장
for chunk in res.iter_content(100):
    f.write(chunk)
f.close()
```

간단한 이미지 가져오기

URL에 맞게 이미지를 파싱한다.

이미지가 가지고 있는 링크 주소를 탐색한뒤 웹 주소로 완성시킨다.

파일 입출력 (이진)을 통해 이미지를 저장한다.

2일차 수업 내용

- Exam
- Request 와 Selenium
- Selenium - 'https://korean.visitkorea.or.kr' 접속하고, 특정 키워드로 접속하기
- Selenium - 1 ~ 10 페이지까지 한 페이지씩 내용 추출하기
- Selenium - imdb에서 영화정보를 크롤링 하여 출력하기
- Selenium - 다나와 사이트 로그인 및 관심상품 출력
- Selenium – 아마존의 best seller 상품 출력

! Exam 01 - 리스트에 데이터 정리해서 넣기

```
html2 = '''
<html>
    <head>
        <h1> 사야할 과일
    </head>
    <body>
        <h1> 시장가서 사야할 과일 목록
            <div> <p id='fruits1' class='name1' title='바나나'> 바나나
                  <span class = 'price'> 3000원 </span>
                  <span class = 'count'> 10개 </span>
                  <span class = 'store'> 바나나가게 </span>
                  <a href = 'https://www.banana.com'> banana.com </a>
                  </p>
            </div>
            <div> <p id='fruits2' class='name2' title='체리'> 체리
                <span class = 'price'> 100원 </span>
                <span class = 'count'> 50개 </span>
                <span class = 'store'> 체리가게</span>
                <a href = 'https://www.cherry.com'> cherry.com </a>
                </p>
            </div>
            <div> <p id='fruits3' class='name3' title='오렌지'> 오렌지
                <span class = 'price'> 500원 </span>
                <span class = 'count'> 20개 </span>
                <span class = 'store'> 오렌지가게</span>
                <a href = 'https://www.orange.com'> orange.com </a>
                </p>
            <div>
    </body>
</html>
'''
```

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html2, 'html.parser')
```

```
# 7번 문제 예시.

path = "div > p"

fluitList = []
for item in soup.select(path):
    #name = item["title"]
    name = str(item.contents[0]).strip()
    price = item.select("span.price")[0].text.strip()
    count = item.select("span.count")[0].text.strip()
    store = item.select("span.store")[0].text.strip()

    fluitList.append({"제품" : name , "재고" : count , "가격" :
price})

for item in fluitList:
    print(item)

# 1. 가게 이름을 모두 크롤링해서 순차적으로 출력하시오
# 2, 가게의 url주소를 모두 크롤링해서 순차적으로 출력하시
# 3. '체리가게'의 제품가격을 출력하시오.
# 4. '바나나가게'의 제품 제고는 몇 개인가?
# 5. '오렌지' 가격은 얼만인가?
# 6. 모든 과일의 총 재고는 몇 개인가?
# 7. 아래와 같이 출력하시오.
# 제품: 바나나, 재고: 10개, 가격: 3000원
# 제품: 체리, 재고: 50개, 가격: 100원
# 제품: 오랜지, 재고: 20개, 가격: 500원
```

! Exam 01 - 정답

```
html2 = '''
<html>
    <head>
        <h1> 사야할 과일
    </head>
    <body>
        <h1> 시장가서 사야할 과일 목록
            <div> <p id='fruits1' class='name1' title='바나나'> 바나나
                  <span class = 'price'> 3000원 </span>
                  <span class = 'count'> 10개 </span>
                  <span class = 'store'> 바나나가게 </span>
                  <a href = 'https://www.banana.com'> banana.com </a>
                  </p>
            </div>
            <div> <p id='fruits2' class='name2' title='체리'> 체리
                <span class = 'price'> 100원 </span>
                <span class = 'count'> 50개 </span>
                <span class = 'store'> 체리가게</span>
                <a href = 'https://www.cherry.com'> cherry.com </a>
                </p>
            </div>
            <div> <p id='fruits3' class='name3' title='오렌지'> 오렌지
                <span class = 'price'> 500원 </span>
                <span class = 'count'> 20개 </span>
                <span class = 'store'> 오렌지가게</span>
                <a href = 'https://www.orange.com'> orange.com </a>
                </p>
            <div>
    </body>
</html>
'''
```

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html2, 'html.parser')
```

```
# 1. 가게 이름을 모두 크롤링해서 순차적으로 출력하시오
path = 'div > p > span.store'
for t in soup.select(path):
  print(t.text)

# 2, 가게의 url주소를 모두 크롤링해서 순차적으로 출력하시오
path = "div > p > a"
for tag in soup.select(path):
  print(tag["href"])

# 3. '체리가게'의 제품가격을 출력하시오.
path = "div > p#fruits2 > span.price"
soup.select(path)[0].text


# 4. '바나나가게'의 제품 제고는 몇 개인가?
path = "div > p#fruits1 > span.count"
soup.select(path)[0].text


# 5. '오렌지' 가격은 얼만인가?
path = "div > p#fruits3 > span.price"
soup.select(path)[0].text
```

Exam 01 - 정답

```
html2 = '''
<html>
    <head>
        <h1> 사야할 과일
    </head>
    <body>
        <h1> 시장가서 사야할 과일 목록
            <div> <p id='fruits1' class='name1' title='바나나'> 바나나
                  <span class = 'price'> 3000원 </span>
                  <span class = 'count'> 10개 </span>
                  <span class = 'store'> 바나나가게 </span>
                  <a href = 'https://www.banana.com'> banana.com </a>
                  </p>
            </div>
            <div> <p id='fruits2' class='name2' title='체리'> 체리
                <span class = 'price'> 100원 </span>
                <span class = 'count'> 50개 </span>
                <span class = 'store'> 체리가게</span>
                <a href = 'https://www.cherry.com'> cherry.com </a>
                </p>
            </div>
            <div> <p id='fruits3' class='name3' title='오렌지'> 오렌지
                <span class = 'price'> 500원 </span>
                <span class = 'count'> 20개 </span>
                <span class = 'store'> 오렌지가게</span>
                <a href = 'https://www.orange.com'> orange.com </a>
                </p>
            <div>
    </body>
</html>
'''
```

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html2, 'html.parser')
```

```
# 6. 모든 과일의 총 재고는 몇 개인가?
path = "span.count"
suma = 0
for tag in soup.select(path):
  # print(int(tag.text.strip()[:-1]))
  suma += int(tag.text.strip()[:-1])
suma


# 7. 아래와 같이 출력하시오.

# """
# 제품: 바나나, 재고: 10개, 가격: 3000원
# 제품: 체리, 재고: 50개, 가격: 100원
# 제품: 오랜지, 재고: 20개, 가격: 500원
# """
for p in soup.select('p'):
  price = p.select('span.price')[0].text
  stock = p.select('span.count')[0].text
  name = p.select('span.store')[0].text.strip()[:-2]

  text = "제품 :{},   재고 :{},   가격 :{}".format(name, stock,
price)
  print(text)
```

! Exam 02 - 개인 풀이

```
html = """
<html>
<head>
    <title>The Dormouse's story</title>
</head>
<body>
    <h1>this is h1 area</h1>
    <h2>this is h2 area</h2>
    <p class="title">
        <b>The Dormouse's story</b>
    </p>
    <p class="story">
        Once upon a time there were three little sisters
        <a href="http://example.com/elsie" class="sister"
id="link1">Elsie</a>
        <a href="http://example.com/lacie" class="sister"
id="link2">Lacie</a>
        <a data-io="link3" href="http://example.com/tillie"
class="brother" id="link3">Tillie</a>
        <b>
            test
            <a href="http://example.com/test" class="brother"
id="link4">TEST</a>
        </b>

    </p>
    <p class="story">story...</p>
</body>
</html>
"""
```

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'html.parser')
```

```
# 'The Dormouse's story'를 출력하시오..
print(soup.head.text)

# 'this is h2 area'를 출력
print(soup.body.h2.text)

# 'Tillie'를 출력하시오..
path = "p.story > a"
print(soup.select(path)[2].text)

# 'Elsie'를 출력하시오.
print(soup.select(path)[0].text)

# "http://example.com/lacie"를 출력하시오
print(soup.select(path)[1]['href'])

# 'test'를 출력하시오
print(soup.select("p.story > b")[0].contents[0].strip())

# 'Test'를 출력하시오. 단, id
print(soup.select("p > b > a#link4")[0].text)

# 'story...'를 출력하시오
p_story_list = soup.select("p.story")
print(p_story_list[len(p_story_list) - 1].text)
```

! Exam 02 - 정답

```
html = """
<html>
<head>
    <title>The Dormouse's story</title>
</head>
<body>
    <h1>this is h1 area</h1>
    <h2>this is h2 area</h2>
    <p class="title">
        <b>The Dormouse's story</b>
    </p>
    <p class="story">
        Once upon a time there were three little sisters
        <a href="http://example.com/elsie" class="sister"
id="link1">Elsie</a>
        <a href="http://example.com/lacie" class="sister"
id="link2">Lacie</a>
        <a data-io="link3" href="http://example.com/tillie"
class="brother" id="link3">Tillie</a>
        <b>
            test
            <a href="http://example.com/test" class="brother"
id="link4">TEST</a>
        </b>

    </p>
    <p class="story">story...</p>
</body>
</html>
"""
```

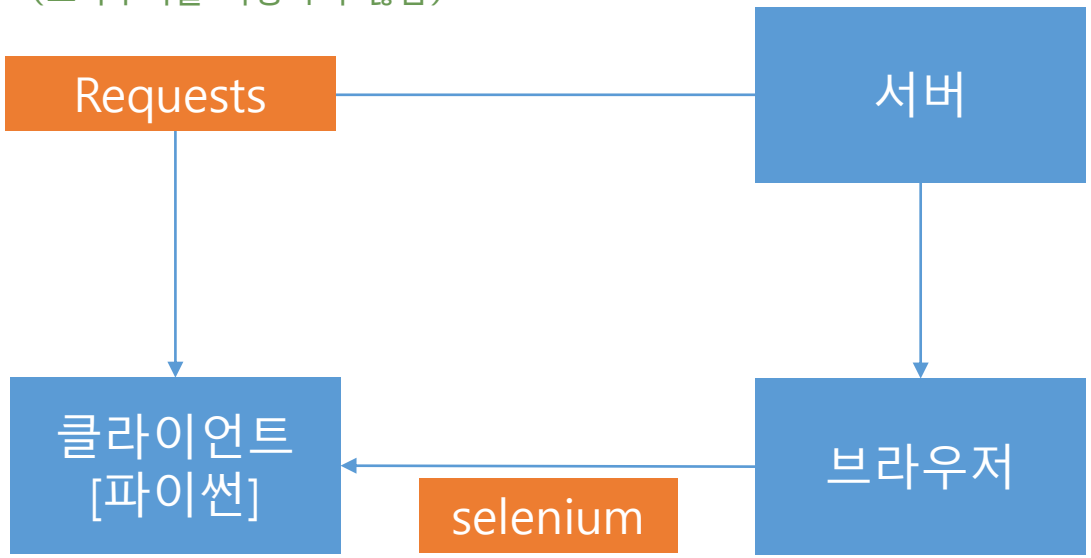
```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html, 'html.parser')
```

```
# 'The Dormouse's story'를 출력하시오.
path = 'p.title > b'
soup.select(path)[0].text

# 'this is h2 area'를 출력
soup.select('h2')[0].text

# 'Tillie'를 출력하시오..
path = "a#link3"
soup.select(path)[0].text

# 'Elsie'를 출력하시오.
path = "a#link1"
soup.select(path)[0].text

# "http://example.com/lacie"를 출력하시오
path = "a#link3"
soup.select(path)[0]['href']
# 'test'를 출력하시오
path = "p.story > b"
soup.select(path)[0].text.strip().split()[0]
# 'Test'를 출력하시오. 단, id나 class속성을 사용하지 마시오..
path = "p > b > a"
soup.select(path)[0].text
# 'story...'를 출력하시오
path = "p.story"
soup.select(path)[1].text
```

! Request 과 selenium

```
import requests
from bs4 import BeautifulSoup

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

url = "https://korean.visitkorea.or.kr"

res = requests.get(url)

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

# path 찾는 방법
# 사이트 > 우상단 단추 > 도구더보기 > 개발자도구 > 좌상단 : select an element
# > 원하는 콘텐츠 클릭 > script highlight > 우클릭 > copy select
path = '#search_result > ul > li > div.cont > div.tit > a'

soup.select(path) # requests가 안됨
```

```
# requests(url) - 프로그램적으로 서버에 요청
(브라우저를 사용하지 않음)
```




```
# selenium : page_source - 브라우저를 통해서 요청
```

requests

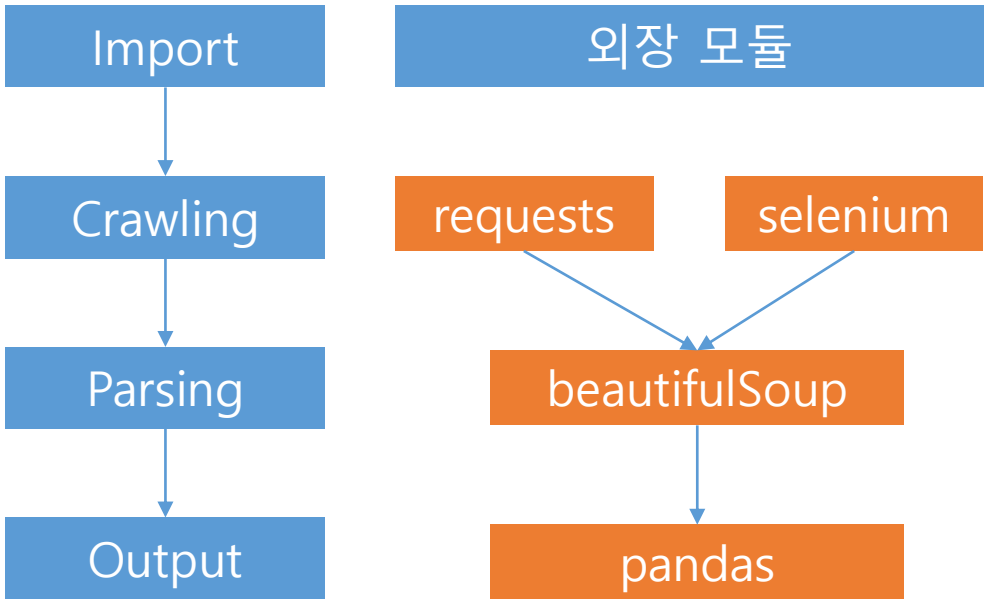
Requests는 보안 및 기타 상의 이유로 사이트를 막아놓는 경우가 있다. 따라서, 사람이 직접 사용하는 듯한 모션을 취함으로써 웹 페이지 코드를 가져올 수 있도록 한다. [브라우저(크롬)으로 접속, 브라우저에서 내용을 가져온다.

! selenium

```python
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.common.by import By
import time

url = "https://korean.visitkorea.or.kr"

# 실시간 사람이 하는 듯한 작업
driver = webdriver.Chrome()
driver.get(url)

sk = input('검색할 키워드를 입력하세요 : ')

### 사용자의 동작을 정의하는 내용
# 페이지 로딩이후 element를 찾기 때문에
# time.sleep(1)을 통해 로딩이 완료되기 까지 대기 시간을 정의할 수 있다.
time.sleep(1)

# id를 통해 잡기
elem = driver.find_element('id','inp_search_mo')
elem.send_keys(sk)

# 버튼을 찾아서 클릭한다.
btn_elem = driver.find_element(By.CSS_SELECTOR, '#placeHolder > a')
btn_elem.click()

# 버튼을 찾아서 클릭한다
elem = driver.find_element(By.CSS_SELECTOR, '#s_attraction > div.more_view > a')
elem.click()
```

```python
# html 코드 가져오기
html = driver.page_source

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

path = '#search_result > ul > li > div.cont > div.tit > a'

for item in soup.select(path):
    print(item.text)
```

기본적인 순서도



Selenium - 'https://korean.visitkorea.or.kr' 접속하고, 특정 키워드로 접속하기

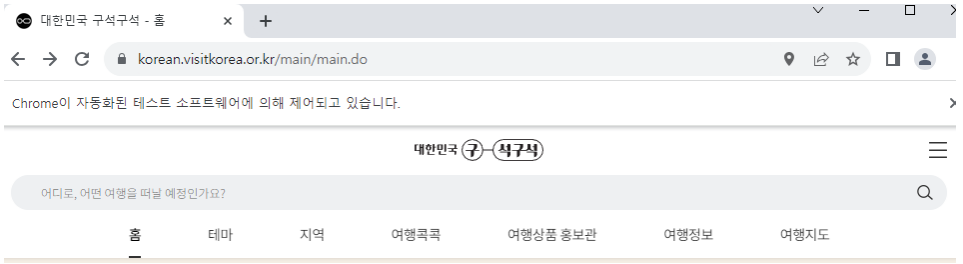
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time
```

```
url = "https://korean.visitkorea.or.kr"

# 크롬드라이버 열기
driver = webdriver.Chrome()
# url 열기
driver.get(url)
# 윈도우사이즈 세팅하기
driver.set_window_size(1000,1500)
```

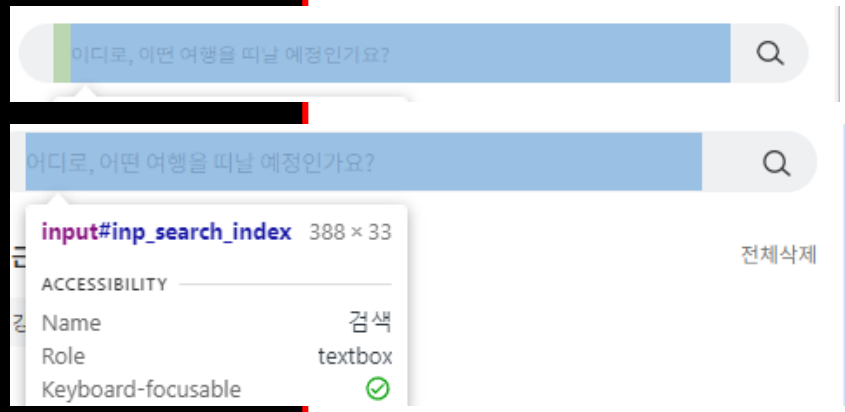
```
select = "강릉"

# 대기 시간
time.sleep(1)

################################
# id를 통해 입력창 잡기
elem = driver.find_element(By.ID,'inp_search_mo')
elem.click()

elem = driver.find_element(By.ID,'inp_search_index')
elem.send_keys(sk)

# 버튼을 찾아서 클릭한다.
btn_elem = driver.find_element(By.CSS_SELECTOR, 'body > div.search_index >
div > div > div > div.form > button:nth-child(3)')
btn_elem.click() # 검사결과페이지로 이동한다
```

1. Import 세팅

2. url 입력 및 연결 : 화면 실행



3. 명령에 따라 자동화 제어 실행



Selenium - 'https://korean.visitkorea.or.kr' 접속하고, 특정 키워드로 접속하기


```
# 크롬드라이버 열기
driver = webdriver.Chrome()
# url 열기
driver.get(url)
# 윈도우사이즈 세팅하기
driver.set_window_size(1000,1500)

select = "강릉"

# 대기 시간
time.sleep(1)

################################
# id를 통해 입력창 잡기
elem = driver.find_element(By.ID,'inp_search_mo')
elem.click()

elem = driver.find_element(By.ID,'inp_search_index')
elem.send_keys(sk)

# 버튼을 찾아서 클릭한다.
btn_elem = driver.find_element(By.CSS_SELECTOR, 'body > div.search_index > div >
div > div > div.form > button:nth-child(3)')
btn_elem.click() # 검사결과페이지로 이동한다
```

Selenium 과 driver 함수를 통해 원하는 페이지로 접근한 뒤 파싱한다.

```
# 원하는 페이지의 소스 코드르 가져온다
html = driver.page_source

# 파싱을 시작한다.
soup = BeautifulSoup(html, "html.parser")
```

Selenium - 'https://korean.visitkorea.or.kr' – find_element 활용하기 - 파싱

Element를 찾아서 명령을 내린다

```
elem = driver.find_element(By.ID,'inp_search_mo')
elem.click()
```

받아올 수 있는 방법

By.CLASS_NAME
By.ID
By.NAME
By.LINK_TEXT
By.PARTIAL_LINK_TEXT
By.TAG_NAME
By.CSS_SELECTOR
By.XPATH

```
driver.find_element(By.LINK_TEXT, f"{page}").click()
```

Link='1' 태그가 있는 내용을 클릭한다.

10 페이지씩 넘기는 코드

```
count = 1
while count <= 10:
    count += 1
    # paging => 페이지 번호를 클릭한다..
    if count == 6:  # 6 => 다음을 클릭
        page= '다음'
    elif count == 11:  # 11 => 반복을 마침
        break
    else:
        page = count

    print(page)

    driver.find_element(By.LINK_TEXT, f"{page}").click()

    time.sleep(1.0)
```

Requests와 같이 BeautifulSoup에 HTML 요소를 넣어, 내용을 가져온다.

```
# 웹 스크립트에서 원하는 콘텐츠를 추출
html = driver.page_source
soup = BeautifulSoup(html, 'html.parser')
```

Selenium – 1 ~ 10 페이지까지 한 페이지씩 내용 추출하기

```
from bs4 import BeautifulSoup

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

url = "https://korean.visitkorea.or.kr"

# 크롬드라이버 열기
driver = webdriver.Chrome()
# url 열기
driver.get(url)
# 윈도우사이즈 세팅하기
driver.set_window_size(1000,1500)

select = "강릉"

# 대기 시간
time.sleep(1)

################################
# id를 통해 입력창 잡기
elem = driver.find_element(By.ID,'inp_search_mo')
elem.send_keys(select)

# 버튼을 찾아서 클릭한다.
btn_elem = driver.find_element(By.CSS_SELECTOR, '#placeHolder > a')
btn_elem.click() # 검사결과페이지로 이동한다

# 대기 시간
time.sleep(1)

# 더보기 버튼을 찾아서 클릭한다
elem = driver.find_element(By.CSS_SELECTOR, '#s_recommend > div.more_view > a')
elem.click()
```

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

li_path = '#search_result > ul > li'
li_tag = soup.select(li_path)

count = 1
while count <= 10:
    # 분석된 데이터 출력하기
    for item in li_tag:
        try:
            title = item.select('div.cont > div.tit > a')[0].text
            try:
                region = item.select('div.cont > span')[0].text
            except:
                region = ''

            tags = [t.text for t in item.select('div.cont > div.tag > span')]
        except:
            pass

        print(title, region, tags)

    count += 1
    # paging => 페이지 번호를 클릭한다..
    if count == 6:  # 6 => 다음을 클릭
        page= '다음'
    elif count == 11:  # 11 => 반복을 마침
        break
    else:
        page = count

    print(page)

    driver.find_element(By.LINK_TEXT, f"{page}").click()

    time.sleep(1.0)
```

Selenium – imdb에서 영화정보를 크롤링 하여 출력하는 코드 [리스트에 담기]

Imdb에서 영화정보를 출력하는 코드

```
from bs4 import BeautifulSoup

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

url = "https://m.imdb.com/chart/top/?ref_=nv_mv_250"

# 크롬드라이버 열기
driver = webdriver.Chrome()
# url 열기
driver.get(url)

time.sleep(1)

################################
html = driver.page_source

# 웹 스크립트에서 원하는 콘텐츠를 추출
html = driver.page_source
soup = BeautifulSoup(html, 'html.parser')
# 원하는 콘텐츠가 있는 경로
path = "#__next > main > div > div.ipc-page-content-
container.ipc-page-content-container--center > section > div >
div.ipc-page-grid.ipc-page-grid--bias-left > div > ul > li"
```

```
movie_list = []

for item in soup.select(path):
    # name 가져오기
    name = item.select("div.ipc-title.ipc-title--base.ipc-title--
title.ipc-title-link-no-icon.ipc-title--on-textPrimary.sc-479faa3c-
9.dkLVoC.cli-title > a > h3")

    # span 내용 가져오기
    newList = item.select("span")

    # 각 내용에 맞는 리스트 값 가져오기
    movie_ranking = name[0].text.split()[0]
    movie_name = name[0].text.split()[1]
    movie_year = newList[1].text
    movie_rate = newList[4].text[:3]

    # 추가하기
    movie_list.append({"ranking":movie_ranking, "name" : movie_name,
"year" : movie_year, "rate":movie_rate})

for item in movie_list:
    print(item)
```

```
{'ranking': '1.', 'name': '쇼생크', 'year': '1994', 'rate': '9.3'}
{'ranking': '2.', 'name': 'Daeboo', 'year': '1972', 'rate': '9.2'}
{'ranking': '3.', 'name': '다크', 'year': '2008', 'rate': '9.0'}
{'ranking': '4.', 'name': 'The', 'year': '1974', 'rate': '9.0'}
```

Selenium – 다나와 사이트 로그인 및 관심상품 출력

다나와 사이트 로그인 및 관심상품 출력

```
# 수동 다나와 사이트에 회원가입 > 로그인 > 관심상품등록
# selenium : 로그인 > 관심상품 목록 출력...

from bs4 import BeautifulSoup

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

url = "https://www.danawa.com/"

# 크롬드라이버 열기
driver = webdriver.Chrome()
# url 열기
driver.get(url)

time.sleep(1)

id = " 아이디@주소 "
Pw = " 비밀번호"

### 로그인 화면 접속
elem = driver.find_element(By.CSS_SELECTOR,"#danawa_header > div > 
div > div.main-header__banner > div.main-header__user > div:nth-
child(5) > a")
elem.click()

time.sleep(1)
```

```
### 로그인 단계
# id 입력
elem = driver.find_element(By.CSS_SELECTOR,"#danawa-member-login-input-id")
elem.send_keys(id)

# pw 입력
elem = driver.find_element(By.CSS_SELECTOR,"#danawa-member-login-input-pwd")
elem.send_keys(pw)

# 로그인 버튼 실행
elem = driver.find_element(By.CSS_SELECTOR,"#danawa-member-login-loginButton")
elem.click()

time.sleep(1)

### 관심 목록 들어가기
elem = driver.find_element(By.CSS_SELECTOR,"#danawa_header > div > div > div.main-
header__banner > div.main-header__user > div:nth-child(4) > a")
elem.click()

time.sleep(1)
```

```
# 파싱
html = driver.page_source
soup = BeautifulSoup(html, "html.parser")

# 관심 목록 상품 목록
path = "#wishProductListArea > table > tbody > tr"

# 값 가져와 출력하기
for item in soup.select(path):
    print(item.select('div.tit')[0].text.strip())
    print(item.select('em.num')[0].text.strip())
    print(item.select('div.sub_info.clear > dl > dd')[0].text.strip())
    print()
```

Selenium – 아마존의 best seller 상품 출력

아마존의 best seller 상품 출력 코드

```
# 아마존의 best seller 상품 출력
# url = 'https://www.amazon.com/Best-Sellers-
Appliances/zgbs/appliances/ref=zg_bs_nav_0'

from bs4 import BeautifulSoup

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

url = "https://www.amazon.com/Best-Sellers-
Appliances/zgbs/appliances/ref=zg_bs_nav_0"

# 크롬드라이버 열기
driver = webdriver.Chrome()
# url 열기
driver.get(url)
```

```
{'name': 'Silonn Ice Maker Countertop, 9 Cubes Ready in 6 Mins, 26lbs in 24Hrs, Self-Cleaning Ice Machine with Ice Scoop and Baske
{'name': 'GE Profile Opal 1.0 Nugget Ice Maker¦ Countertop Pebble Ice Maker ¦ Portable Ice Machine Makes up to 34 lbs. of Ice Per
{'name': '', 'score': '', 'people': '', 'price': ''}
{'name': '', 'score': '', 'people': '', 'price': ''}
{'name': '', 'score': '', 'people': '', 'price': ''}
{'name': '', 'score': '', 'people': '', 'price': ''}
{'name': 'Nugget Countertop Ice Maker with Soft Chewable Pellet Ice, Pebble Portable Ice Machine, 34lbs Per Day, Self-Cleaning, So
{'name': 'ecozy Portable Countertop Ice Maker - 9 Ice Cubes in 6 Minutes, 26 lbs Daily Output, Self-Cleaning with Ice Bags, Scoop,
{'name': 'Upstreman 3.2 Cu.Ft Mini Fridge with Freezer, Single Door, Adjustable Thermostat, Refrigerator for Dorm, Office, Bedroom
{'name': 'EUHOMY Countertop Ice Maker Machine with Handle, 26lbs in 24Hrs, 9 Ice Cubes Ready in 6 Mins, Auto-Cleaning Portable Ice
{'name': 'Silonn Ice Maker Countertop, Portable Ice Machine with Carry Handle, Self-Cleaning Ice Makers with Basket and Scoop, 9 C
{'name': 'Oraimo Nugget Ice Maker, Ice Makers Countertop, 26 Lbs/Day Tooth-Friendly Chewable Ice with Self-Cleaning & Auto Water R
{'name': 'Upstreman 1.7 Cu.ft Mini Fridge with Freezer, Adjustable Thermostat, Energy Saving, Low Noise, Single Door Compact Refri
{'name': 'BANGSON Compact Fridge with Freezer, 3.2 CU.FT. Small refrigerator with Freezer, 5 Adjustable Temperatures, 38 dB Low No
{'name': 'Countertop Ice\xa0Maker 6 Mins 9 Bullet Ice, 26.5lbs/24Hrs, Portable Ice Maker Machine with Self-Cleaning, Bags, Ice Sco
```

```
html = driver.page_source
soup = BeautifulSoup(html, "html.parser")

# 그리드 가져오기
path = "div.p13n-desktop-grid > div > div > div"

# 리스트에 값넣기
product_list = []

for item in soup.select(path):
    name = ""
    score = ""
    people = ""
    price = ""
    try:
        # 각 분야별 위치 찾아서 적용하기
        name = item.select('a.a-link-normal > span > div')[0].text.strip()
        score = item.select('a.a-link-normal > i > span')[0].text.strip()
        people = item.select('a.a-link-normal > span.a-size-small')[0].text.strip()
        price = item.select('a.a-link-normal > div > span > span')[0].text.strip()
    except:

        pass

    # 리스트에 상품 추가
    product_list.append({"name" : name, "score" : score , "people" : people,
"price" : price})

# 출력
for item in product_list:
    print(item)
```

3일차 수업 내용

- Exam
- Exam – 다나와 관심목록
- Exam – 아마존 베스트 셀러
- Selenium – 교보문고에서 베스트 셀러 가져오기
- Selenium – 구글에서 이미지 검색 이후 가져오기

! Exam

문제, 각각의 칼럼 및 내용을 탐색하시오

```
from bs4 import BeautifulSoup

html = """
    <body>
      <table border=1>
         <tr>
            <td> 항목 </td>
            <td> 2013 </td>
            <td> 2014 </td>
         </tr>
         <tr>
            <td> 매출액 </td>
            <td> 100 </td>
            <td> 200 </td>
         </tr>
      </table>
    </body>
"""
```

```
soup = BeautifulSoup(html,'html.parser')
test = path = 'table > tr'
result_list = []

for item in soup.select(path):
    result_list.append([x.text.strip() for x in item.select('td')])

for item in result_list:
    print(item)
```

문제 및 답안

```
# 2번째 칼럼을 출력
path = 'tr > td:nth-child(3)'
tds = soup.select(path)
[td.text.strip() for td in tds]
```

```
# 1번째 row를 출력
path = 'tr:nth-child(1) > td'
soup.select(path)
```

```
# 2014년도 매출액 출력
path = 'tr:nth-child(1) > td'
soup.select(path)
```

pandas 예시

```
# pandas
import pandas as pd
pd_table = pd.read_html(html)

print(pd_table)

print('------------------')
# 2번째 칼럼을 출력
for i in range(len(pd_table[0])):
    print(pd_table[0][i][1])

print('------------------')
# 1번째 row를 출력
print(pd_table[0][1])

print('------------------')
# 2014년도 매출액 출력
print(pd_table[0][2][1])
print('------------------')
```

! Exam – 다나와 관심목록

```
# 다나와 관심상품
from bs4 import BeautifulSoup

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

import my_id

url = "https://www.danawa.com/"

# 크롬드라이버 열기
driver = webdriver.Chrome()
# url 열기
driver.get(url)
time.sleep(1)

### 로그인 화면 접속
elem = driver.find_element(By.CSS_SELECTOR, '#danawa_header > div > div > div.main-
header__banner > div.main-header__user > div:nth-child(5) > a > span')
elem.click()
time.sleep(1)

### 로그인 단계
# id 입력
elem = driver.find_element(By.ID,"danawa-member-login-input-id")
elem.send_keys(my_id.private_id['id'])

# pw 입력
elem = driver.find_element(By.ID,"danawa-member-login-input-pwd")
elem.send_keys(my_id.private_id['pw'])

# 로그인 버튼 실행
elem = driver.find_element(By.ID,"danawa-member-login-loginButton")
elem.click()
time.sleep(1)

### 관심 목록 들어가기
elem = driver.find_element(By.CLASS_NAME,'btn_user.btn_user--wish')
elem.click()
time.sleep(1)
```

다나와 로그인과 관심목록 파악하기

1. URL 을 불러온다.
2. 다나와 로그인으로 접속한다
3. ID와 패스워드를 작성한다. [my_id.py 파일 import]
4. 로그인 이후 관심목록으로 접속한다
5. 데이터를 파싱한다.

```
# my_id.py 파일
private_id = {"id" : '아이디', 'pw' :'비밀번호'}
```

```
# 파싱
html = driver.page_source
soup = BeautifulSoup(html, "html.parser")
#
trs = soup.select('#wishProductListArea > table > tbody > tr')

title = trs[0].select('#wishProductListArea > table > tbody > tr:nth-child(1) > td.info > div.tit >
a')[0].text
desc = trs[0].select('#wishProductListArea > table > tbody > tr:nth-child(1) > td.info > dl >
dd')[0].text

for tr in trs:
    title = tr.select('td.info > div.tit > a')[0].text
    desc = tr.select('td.info > dl > dd')[0].text
    price = tr.select('td.lowest > div.cost > span > em')[0].text

    print(title)
    print(desc)
    print(price)
    print()
```

! Exam – 아마존 베스트 셀러

```python
from bs4 import BeautifulSoup

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

url = "https://www.amazon.com/Best-Sellers/zgbs"

# 크롬드라이버 열기
driver = webdriver.Chrome()
# url 열기
driver.get(url)

# Appliances화면으로 넘어가기
elem = driver.find_element(By.LINK_TEXT,'Appliances')
elem.click()

time.sleep(1)
```

```python
# 스크롤을 끝까지 내리는 코드
SCROLL_PAUSE_TIME = 2

last_height = driver.execute_script("return document.body.scrollHeight")

while True:
    driver.execute_script('window.scrollTo(0, document.body.scrollHeight);')
    time.sleep(SCROLL_PAUSE_TIME)

    new_height = driver.execute_script('return document.body.scrollHeight')
    if new_height == last_height:
        try:
            time.sleep(SCROLL_PAUSE_TIME)
            driver.find_element(By.CSS_SELECTOR, "input.mye4qd").click()
        except:
            break
    last_height = new_height

##
```

아마존 베스트 셀러, 모범답안

1. URL 을 불러온다.
2. Appliances 화면으로 접속한다
3. 스크롤을 맨 아래까지 내린다
4. 원하는 페이지에서 코드를 파싱한다.
5. 파싱하고자 하는 데이터를 찾는다.
6. 출력한다.

```python
html = driver.page_source
soup = BeautifulSoup(html, 'html.parser')
divs = soup.select('#gridItemRoot')

title = divs[0].select('#B08ZYJ8CRX > a:nth-child(2) > span > div')[0].text
rate = divs[0].select('#B08ZYJ8CRX > div:nth-child(3) > div > a > i > span')[0].text
price = divs[0].select('#B08ZYJ8CRX > div:nth-child(4) > a > span > span > span')[0].text

print(title, rate, price)

for div in divs:
    try:
        title = div.select('._cDEzb_p13n-sc-css-line-clamp-3_g3dy1')[0].text
        rate = div.select('.a-size-small')[0].text
        price = div.select('span.p13n-sc-price')[0].text

        print(title, rate, price)
    except:
        pass
```

Selenium2 – 교보문고에서 베스트 셀러 가져오기

교보문고 베스트 셀러 가져오기

```
# 교보문고
# 일간 베스트셀러 목록 100개 출력하기
# => 제목, 저자, 출판사, 출판일, 가격, 서머리
# csv file로 저장하기

from bs4 import BeautifulSoup

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

url = 'https://www.kyobobook.co.kr/'

# 크롬드라이버 열기
driver = webdriver.Chrome()
# url 열기
driver.get(url)

time.sleep(1)

# 베스트로 페이지 전환
elem = driver.find_element(By.LINK_TEXT,'베스트')
elem.click()
```

1. 반복문을 사용하여, 크롬 브라우저를 제어 내용을 파싱한다.
2. 반복하면서 리스트에 데이터를 넣는다
3. 이에 대한 내용을 CSV로 저장한다.

```
while True:
    time.sleep(2)
    html = driver.page_source
    soup = BeautifulSoup(html, "html.parser")

    for item in soup.select('.prod_list > li.prod_item'):
        try:
            name = item.select('a.prod_info')[0].text.strip() #제목
            author =
item.select('span.prod_author')[0].text.strip().split('·')[0].strip() #저자
            company =
item.select('span.prod_author')[0].text.strip().split('·')[1].strip() #출판사
            date = item.select('span.date')[0].text.split('·')[1].strip() #출판일
            price = item.select('span.val')[0].text.strip() #가격
            summary = item.select('p.prod_introduction')[0].text.strip() #서머리
            best_seller_list.append({"name" : name, "author" : author, "company" :
company, "date" : date, "price" : price, "summary" : summary})
        except:
            # summary가 없을 경우 빈칸으로 추가
            best_seller_list.append({"name" : name, "author" : author, "company" :
company, "date" : date, "price" : price, "summary" : ""})

    count += 1

    if len(best_seller_list) > 99:
        break
    if count > 7:
        break

    elem = driver.find_element(By.LINK_TEXT,f'{count}')
    elem.click()
```

```
import pandas as pd
news_df = pd.DataFrame(best_seller_list)
news_df.to_csv('best_seller.csv')
```

! Selenium2 - 구글에서 이미지 검색 이후 가져오기

```
import requests
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time
from urllib.request import urlretrieve

url = 'https://www.google.com/'

driver = webdriver.Chrome()
driver.get(url)
time.sleep(1)

elem = driver.find_element(By.CSS_SELECTOR, '#APjFqb')
sk = "할로"
elem.send_keys(sk)
elem.send_keys(Keys.RETURN)
time.sleep(1)
elem = driver.find_element(By.LINK_TEXT,'이미지')
elem.click()
time.sleep(1)
```

```
SCROLL_PAUSE_TIME = 2
last_height = driver.execute_script("return document.body.scrollHeight")

while True:
    driver.execute_script('window.scrollTo(0, document.body.scrollHeight);')
    time.sleep(SCROLL_PAUSE_TIME)
    new_height = driver.execute_script('return document.body.scrollHeight')
    if new_height == last_height:
        try:
            time.sleep(SCROLL_PAUSE_TIME)
            driver.find_element(By.CSS_SELECTOR, "input.mye4qd").click()
        except:
            break
    last_height = new_height
    time.sleep(1)
    try:
        elem = driver.find_element(By.CSS_SELECTOR, '#islmp > div > div > div > div >
div.C5Hr4 > div.K414Oe > div.FAGjZe > input')
        elem.click()
    except:
        pass
```

구글에서 이미지 검색 이후 이미지 가져오기

1. 이미지를 검색한다
2. 스크롤을 맨 아래로 내린다
3. 데이터를 파싱한다.
4. 저장한다.

```
from urllib.request import urlretrieve
import sys
import os

elems = driver.find_elements(By.CSS_SELECTOR,'#islrg > div.islrc > div > a.FRuiCf.islib.nfEiy >
div.fR600b.islir > img')
len(elems)

current_path = globals()['_dh'][0]
folderName = 'Test_Image'

for i, item in enumerate(elems):
    try:
        item.click()
        time.sleep(1)
        elem = driver.find_element(By.XPATH,'//*[@id="Sva75c"]/div[2]/div[2]/div[2]/div[2]/c-
wiz/div/div/div/div/div[3]/div[1]/a/img[1]')

        img_url = elem.get_attribute('src')
        print(img_url)

        image_path = os.path.join(current_path,folderName)

        if os.path.exists(folderName) == True: # 파일이랑 폴더 경로
            os.mkdir(folderName) # 폴더 생성하기

        urlretrieve(img_url, os.path.join(image_path,'go_img{}.jpg'.format(i)))
    except:
        print("error")
        print(item)
        pass
```

4일차 수업 내용

- 검색 본문 내용 가져오기 1
- 검색 본문 내용 가져오기 2
- Iframe 네이버 증권 시세 가져오기
- Iframe 중고나라 데이터 가져오기
- Requests 특정 헤더 가져오기

! 검색 본문 내용 가져오기 1

검색어를 입력 받은 후 Naver 뉴스를 크롤링하고 다음과 같이 출력하시오.

```
# ---------------
# title : xxx
# press : xxx
# date : xxx
# summary : xxx
```

답안

```
# 모듈 불러오기
import requests
from bs4 import BeautifulSoup

# 초기 검색 내용 및 변수세팅
msg = "스마일"
new_list = []
site_num = 2

# url 불러오기
url =
"https://search.naver.com/search.naver?where=news&sm=tab_pge&query={}&s
ort=0&photo=0&field=0&pd=0&ds=&de=&cluster_rank=27&mynews=0&office_type
=0&office_section_code=0&news_office_checked=&office_category=0&service
_area=0&nso=so:r,p:all,a:all&start={}".format(msg, (site_num*10 + 1))

# html 불러오기
res = requests.get(url)
```

URL의 구성요소

주소/경로/경로/해당파일

해당파일 이름 [파일명_번호] : 번호를 포멧팅하여 연결

```
# 불러온 내용 soup에 넣기
soup = BeautifulSoup(res.text, 'html.parser')

# 파싱
for item in soup.select('ul.list_news > li'):
    title = item.select('a.news_tit')[0].text.strip()
    href = item.select('a.news_tit')[0]['href']
    press = item.select('div.info_group > a')[0].text.strip()
    summary = item.select('div.news_dsc > div > a')[0].text.strip()
    date = item.select('div.info_group > span')[0].text.strip()

    print("Title : ", title)
    print("Press : ", press)
    print("Date : ", date)
    print("Summary : ", summary)
    print("Href : ", href)
    print()
```

```
Title :  '올해는 더 뜨겁다' 스마일게이트, CFS2023 그랜드 파이널 中서 열린다
Press :  메트로신문
Date :  1일 전
Summary :  스마일게이트, CFS 2023 그랜드 파이널 중국 청두서 본격 개막/스마일게이트 스마일게이트 엔터테인먼트는 중국 청두에서 전세계 최고 권위의 '크로스파이어' e스포츠 대회인 'CF
Href :  http://www.metroseoul.co.kr/article/20231130500011
```

검색 본문 내용 가져오기 1

각 리스트 URL과 Content 뽑아내기

```
# title : xxx
# press : xxx
# date : xxx
# summary : xxx
# content url : xxx
# content : xxx
```

답안

```
print("Title : ", title)
print("Press : ", press)
print("Date : ", date)
print("Summary : ", summary)
print("URL : ", href)

res = requests.get(href)
content_soup = BeautifulSoup(res.text,'html.parser')
```

양식이 다를 경우 문제가 생길 수 있다.

이에 새로운 모듈을 추가한다

Newspaper3k : 뉴스 페이지의 양식을 분석하여 파싱해 주는 모듈

!pip install newspaper3k

```
# 모듈 불러오기
import requests
from bs4 import BeautifulSoup
from newspaper import Article
```

```
try:
        title = item.select('a.news_tit')[0].text.strip()
        href = item.select('a.news_tit')[0]['href']
        press = item.select('div.info_group > a')[0].text.strip()
        summary = item.select('div.news_dsc > div > a')[0].text.strip()
        date = item.select('div.info_group > span')[0].text.strip()

        a = Article(href,language='ko')
        a.download()
        a.parse()

        content_title = a.title
        content_summary = a.text

        new_list.append({"Title" : title, "Press" : press, "Date" : date,
"Summary" : summary, "URL" : href})
        content_list.append({"content_title" : content_title , "content_summary" :
content_summary })
    except:
        pass
```

```
# csv로 파일 추출하기
import pandas as pd

new_pd = pd.DataFrame(new_list)
content_pd = pd.DataFrame(content_list)

new_pd.to_csv("new_second.csv")
content_pd.to_csv("new_second_content.csv")
```

! 검색 본문 내용 가져오기 2

검색어를 입력 받은 후 Nate 뉴스를 크롤링하고 다음과 같이 출력하시오.

```
# nate 검색 후 뉴스결과에서 제목, 언론사, 날짜, 서머리, 본문까지를
csv file로 저장하시오
import requests
from bs4 import BeautifulSoup
from newspaper import Article

# 초기 검색 내용 및 변수 세팅
msg = "검색내용"
site_num = 1
news_list = []

# Nate URL 불러오기
url = "https://news.nate.com/search?q={}&page={}".format(msg,site_num)

# html 불러오기
res = requests.get(url)
```

1. Nate에서 검색 관련 URL 가져오기
2. Requests로 URL을 통해 HTML 내용 가져오기
3. BeautifulSoup 에 내용담고 Parsing하기
4. 웹페이지를 분석한 내용 가져오기
5. 웹페이지를 분석한 newspaper3k 함수를 이용하여 원하는 내용 가져오기
6. 딕셔너리에 내용 담기
7. Pandas를 이용하여 엑셀로 저장하기

```
# 불러온 내용 soup에 넣기
soup = BeautifulSoup(res.text, 'html.parser')

for item in soup.select('.search-result > ul > li'):
    try:
        href = item.select('a')[0]['href']

        a = Article('https:' + href, language='ko')
        a.download()
        a.parse()

        d = dict()
        d['title'] = a.title
        d['press'] = item.select('span.time')[0].text.strip().split()[0]
        d['day'] = item.select('span.time')[0].text.strip().split()[1]
        d['summary'] = item.select('span.txt')[0].text.strip()
        d['content'] = a.text

        news_list.append(d)
    except:
        pass
```

```
import pandas as pd

new_pd = pd.DataFrame(news_list)
new_pd.to_csv("nate_news.csv")
```

! Iframe 네이버 증권 시세 가져오기

네이버 증권에서 일별시세 가져오기

```
# 네이버 증권
# 일별시세
from bs4 import BeautifulSoup
import pandas as pd

# selenium 임포트
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome()
driver.get(url)

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

```
url = 'https://finance.naver.com/item/sise.naver?code=005930'
sub_url = '/item/sise_day.naver?code=005930'
main_url = 'https://finance.naver.com'

full_url = main_url+ sub_url

#res = requests.get(full_url)

driver = webdriver.Chrome()
driver.get(full_url)

html = driver.page_source

data = pd.read_html(html)[0]
data.dropna(inplace=True)
data
```

```
# IFrame 다른 웹에 있는 웹페이즈 내용을 가져와서 보여주는 기능
# iframe에 있는 콘텐츠는 현 웹페이지 서버에서 제하는 콘텐츠가 아니라
# 다른 웹의 화면만 가져와서 보여주는 기능
# iframe의 콘텐츠는 다른 웹 서버에서 제공하는 것
```

```
<iframe name="day" src="/item/sise_time.naver?code=005930&thistime=20231204134133" width="100%" height="360" marginheight="0" bottommargin="0" topmargin="0" scrolling="no" frameborder="0" title="주요 시세" style="margin-top:20px;clear:both">…</iframe>
<iframe name="day" src="/item/sise_day.naver?code=005930" width="100%" height="360" marginheight="0" bottommargin="0" topmargin="0" scrolling="no" frameborder="0" title="일별 시세"> == $0
  #document
    <html lang="ko">
```

1. 해상 페이지에서 Iframe으로 구성된 내용을 가져온다.
2. 그에 맞는 url을 찾아 sub_url에 작성 full_url을 작성한다.
3. Iframe이 표시하고 있는 url 페이지에서 값을 가져와 파싱한다.

Iframe 중고나라 데이터 가져오기

네이버 중고나라에서 데이터 가져오기

```
from bs4 import BeautifulSoup
import pandas as pd

# selenium 임포트
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys

main_url = 'https://cafe.naver.com/joonggonara.cafe'

driver = webdriver.Chrome()
driver.get(main_url)

select = "자전거"

elem = driver.find_element(By.CSS_SELECTOR,'#topLayerQueryInput')
elem.send_keys(select)
elem.send_keys(Keys.RETURN)

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

frame = soup.find('iframe', id = 'cafe_main')
frameaddr = main_url + frame['src']

driver.get(frameaddr)
```

```
driver.switch_to.frame('cafe_main')

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

for item in soup.select('div.article-board > table > tbody > tr'):
    try:
        print(item.select('td.td_article > div.board-number > div')[0].text.strip())
        print(item.select('div.board-list > div > a')[0].text.strip())
        print(item.select('td.td_date')[0].text.strip())
        print(item.select('td.td_view')[0].text.strip())
    except:
        pass
```

```
driver.switch_to.frame('cafe_main')
```

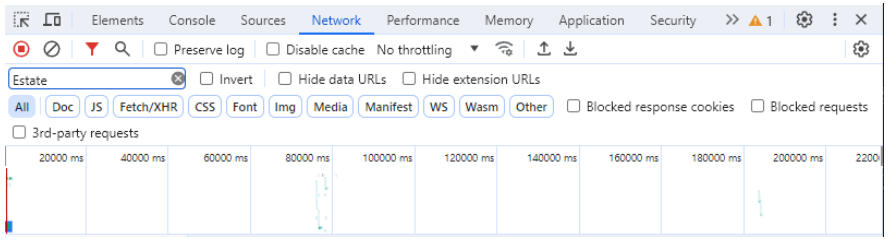
Iframe이 가지고 있는 ID로 iframe url에 접근할 수 있다.

이를 통해 현재 driver가 적용중인 페이지를 바꿀 수 있다.

```
driver.switch_to.default_content() # 다시 main frame으로 돌아가기
```

! Requests 특정 헤더 가져오기

Requests 네트워크에서 특정 헤더를 통해 특정 데이터 가져오기



▼ Request Headers ☐ Raw

Accept:	*/*
Accept-Encoding:	gzip, deflate, br
Accept-Language:	ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Authorization:	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IlJFQUxFU1RBVEUiLCJpYXQiOjE3MDE2NTYzMDYsImV4cCI6MTcwMTY2NzEwNn0.KIIujSnvI_skPG8FaWV8rLTPP31JIwc2iGIIHwauXfM
Connection:	keep-alive
Cookie:	NNB=4QFALYOUGJWWK; page_uid=iT02GIqVOsossaCs6GRsssssD0-096402; _naver_usersession_=Z8GjYvfJQWlpJg5yRbNhAQ==; nx_ssl=2; nhn.realestate.article.rlet_type_cd=A01; nhn.realestate.article.trade_type_cd=""; nhn.realestate.article.ipaddress_city=1100000000; landHomeFlashUseYn=Y; realestate.beta.lastclick.cortar=1174000000; REALESTATE=Mon%20Dec%2004%202023%2011%3A18%3A26%20GMT%2B0900%20(KST); wcs_bt=4f99b5681ce60:1701656302
Host:	new.land.naver.com
Referer:	https://new.land.naver.com/complexes?ms=37.530126,127.123771,15&a=APT:PRE&e=RETAIL
Sec-Ch-Ua:	"Google Chrome";v="119", "Chromium";v="119", "Not?A_Brand";v="24"
Sec-Ch-Ua-Mobile:	?0
Sec-Ch-Ua-Platform:	"Windows"
Sec-Fetch-Dest:	empty
Sec-Fetch-Mode:	cors
Sec-Fetch-Site:	same-origin
User-Agent:	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36

Requests 사용시 서버에 직접 요청을 하여 데이터를 파싱할 수 있다.
개발자 도구의 네트워크 탭에서 이를 확인할 수 있으며 아래와 같이 서버에 데이터를 요청한다.

```
import requests
from bs4 import BeautifulSoup
url =
'https://new.land.naver.com/complexes?ms=37.5301685,127.1233955,16&a=APT:PRE&e=RETAIL'

headers = {
    'Referer' :
'https://new.land.naver.com/complexes?ms=37.530126,127.123771,15&a=APT:PRE&e=RETAIL',
    'User_Agent' : "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/119.0.0.0 Safari/537.36"
}

res = requests.get(url, headers = headers)
res.text
soup = BeautifulSoup(res.text)
print(res.text)
```

! 부록

Requests 공식 문서

https://requests.readthedocs.io/en/latest/user/quickstart/

BeautifulSoup 공식 문서

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Selenium 공식 문서

https://selenium-python.readthedocs.io/

newspaper3k 공식 문서

https://newspaper.readthedocs.io/en/latest/