# 파이썬 프로그래밍

## (데이터 수집)

# 01. Web Scrapping과 Crawling 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(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.

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

[01.소개] ex01.html [크롬브라우저]-[마우스 오른쪽]-[검사]

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>스크랩핑 홈페이지</title>
</head>
<body>
    <div id="학교명">ICIA 고등학교
        <div>1학년
            <div>1반
                <div id="240101">홍길동</div>
                <div id="240102">심청이</div>
            </div>
            <div>2반
                <div id="240103">강감찬</div>
            </div>
            <div>3반</div>
            <div>3반</div>
        </div>
    </div>
</body>
</html>
```

---

- '홍길동' XPath

```
//*[@id="240101"]
```

- '홍길동' full XPath

```
/html/body/div/div/div[1]/div[1]
```

- requests 소개

Python의 requests 모듈은 HTTP 요청을 보내고 응답을 받는 데 사용되는 라이브러리입니다. requests 모듈은 다양한 HTTP 메서드(GET, POST, PUT, DELETE 등)를 지원하며, 간단하고 직관적인 API를 제공하여 HTTP 클라이언트를 쉽게 구현할 수 있도록 도와준다.

- requests 설치

requests 모듈은 Python 기본 라이브러리가 아니기 때문에 설치가 필요합니다. 다음 명령어를 사용하여 설치할 수 있습니다.

```
pip install requests
```

- requests 설치 확인

```
pip list
```

- **[구글]-[송중기]-[이미지] 웹페이지 출력 후 파일저장**

**[01.소개] ex01.py**

```
import requests

url=''https://www.google.com/search?...&q=송중기&udm=2&...&bih=665&dpr=1.5''
res = requests.get(url)

#<Response [200]> 정상, <Response [403]>인 경우 권한이 없다는 의미로 User-Agent를 지정한다.
print(res)

with open('data/ex01.html', 'w', encoding='utf-8') as file:
    #text 속성을 이용해 UTF-8로 인코딩된 문자열을 받을 수 있다.
    file.write(res.text)
```

**출력 결과**

```
<Response [200]>
```

- **User Agent : 구글에서 'What is User Agent String'로 검색한 후 구한다.**

**일반 사용자가 아닌 크롤러 등이 사이트에 접근하려고 할 때 접근이 차단된 특정 사이트가 있다. 따라서 사이트에게 사람인 척해주어야 한다. 사람인 척하는 방법은 페이지에 접속할 때 User Agent 값을 넘겨주는 것이다. User Agent는 접속하는 PC, 브라우저에 따라 달라진다.**

**[01.소개] ex02.py**

```
import requests

url=''https://www.google.com/search?...&q=송중기&udm=2&...&bih=665&dpr=1.5''
headers = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Chrome/132.0.0.0 Safari/537.36'}
res = requests.get(url, headers=headers)

#User-Agent를 지정하면 403 오류인 경우 200 정상으로 접속한다.
print(res)

with open('data/ex02.html', 'w', encoding='utf-8') as file:
    file.write(res.text)
```

**출력 결과**

```
<Response [200]>
```

- **params 속성 지정**

**[01.소개] ex03.py**

```
import requests

url='https://www.google.com/search?...&udm=2...&bih=665&dpr=1.5'
headers = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Chrome/132.0.0.0 Safari/537.36'}
param = {'q' : '조인성'}
res = requests.get(url, headers=headers, params=param)

print(res)

with open('data/ex03.html', 'w', encoding='utf-8') as file:
    file.write(res.text)
```

**출력 결과**

```
<Response [200]>
```

- **네이버 사이트의 이미지 다운로드**

**[01.소개] ex04.py**

```
import requests
url='https://ssl.pstatic.net/melona/libs/1524/1524564/386e95af7ed41e37832b_20250103143458551.jpg'
res = requests.get(url)

with open('data/naver.jpg', 'wb') as file:
    file.write(res.content) #content 속성을 통해 바이너리 타입으로 데이터를 받을 수 있다.
```

- **request 객체에 접근**

**[01.소개] ex05.py**

```
import requests

url = 'http://google.com'
headers = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Chrome/132.0.0.0 Safari/537.36'}
param = {'q' : 'python'}
res = requests.get(url, headers=headers, params=param)

print(f'1.res.request.path_url:{res.request.path_url}')
print(f'2.res.request.method:{res.request.method}')
print(f'3.res.request.headers:{res.request.headers['User-Agent']}')
print(f'4.res.encoding:{res.encoding}')
```

**출력 결과**

```
1.res.request.path_url:/?q=python&gws_rd=ssl
2.res.request.method:GET
3.res.request.headers:Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/132.0.0.0 Safari/537.36
4.res.encoding:UTF-8
```

- **Response객체의 응답코드(status_code) 속성**

**[01.소개] ex06.py**

```
import requests

url = 'http://google.com'
res = requests.get(url)
print(f'{url}:{res.status_code}')

url = 'http://google.com/user'
res = requests.get(url)
print(f'{url}:{res.status_code}')
```

**출력 결과**

```
http://google.com:200
http://google.com/user:404
```

- **무한정으로 응답을 기다리는 상황을 막기 위해 timeout 파라미터에 값을 넣어준다.**

**[01.소개] ex07.py**

```
import requests
url = 'http://google.com/user'
res = requests.get(url, timeout=0.001) #0.001초 내에 응답이 오지 않으면 예외가 발생한다.
print(res)
```

- raise_for_status() 메서드는 requests.get()으로 데이터를 요청한 후, 응답상태코드를 확인하고, 오류가 발생하면 예외를 발생시킨다.

[01.소개] ex08.py

```
import requests

url = 'http://google.com/user'
res = requests.get(url)
print(res)

#만약 에러가 있다면 멈춰주고 에러를 알려 준다.
res.raise_for_status()

url = 'http://google.com'
res = requests.get(url)
print(res)
```

출력 결과

```
<Response [404]>
Traceback (most recent call last):
  File "c:\data\python\크롤링\01.소개\ex08.py", line 8, in <module>
    res.raise_for_status()
    ~~~~~~~~~~~~~~~~~~~~~^^
  File "C:\Users\hdcho\AppData\Local\Programs\Python\Python313\Lib\site-packages\requests\models.py", line 1024, in raise_for_status
    raise HTTPError(http_error_msg, response=self)
requests.exceptions.HTTPError: 404 Client Error: Not Found for url: http://google.com/user
```

- try ~ except로 HTTPError를 핸들링한다.

[01.소개] ex09.py

```
import requests

try:
    url = 'http://google.com/user'
    res = requests.get(url)
    print(res)
    if res.status_code == requests.codes.ok: #코드 200
        print(f'연결성공입니다. [코드:{res.status_code}]')
    else:
        print(f'연결실패입니다. [코드:{res.status_code}]')

    #만약 에러가 있다면 멈춰주고 에러를 알려 준다
    res.raise_for_status()

    url = 'http://google.com'
    res = requests.get(url)
    print(res)
    if res.status_code == requests.codes.ok: #코드 200
        print(f'연결성공입니다. [코드:{res.status_code}]')
    else:
        print(f'연결실패입니다. [코드:{res.status_code}]')
except requests.HTTPError as err:
    print(f'에러:{err}')
```

출력 결과

```
<Response [404]>
연결실패입니다. [코드:404]
에러:404 Client Error: Not Found for url: http://google.com/user
```

## 02. 정규식 표현

**정규 표현식(Regular expressions) 은 특정한 규칙을 가진 문자열의 집합을 다루는데 사용하는 형식 언어이다. 복잡한 문자열의 검색과 치환을 위해 사용되며, Python 뿐만 아니라 문자열을 처리하는 모든 프로그램 언어에서 사용된다.**

- **. : 하나의 문자를 의미한다. 예) ca.e - care, cafe, case(O) | caffe(X)**
- **^ : 특정 문자열로 시작을 의미한다. 예) ^de - desk, destination(O) | fade(X)**
- **$ : 특정 문자열의 끝남 의미한다.  예) se$ - case, base(O) | seek(X)**

**[02.정규식] ex01.py**

```
import re

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

m = p.match('care')
if m:
    print('match')
else:
    print('no match')

m = p.match('caffe')
if m:
    print('match')
else:
    print('no match')
```

**출력 결과**

```
match
no match
```

**[02.정규식] ex02.py**

```
import re

p = re.compile('^de')

m = p.match('desk')
if m:
    print('match')
else:
    print('no match')

m = p.match('fade')
if m:
    print('match')
else:
    print('no match')

m = p.match('destination')
if m:
    print('match')
else:
    print('no match')
```

**출력 결과**

```
match
no match
match
```

- match() 메서드는 문자열의 처음부터 search() 메서드는 문자열 전체를 검색하여 정규식과 매치되는지 조사한다.

[02.정규식] ex03.py

```
iimport re

p = re.compile('se$')

m = p.match('case')
if m:
    print('match')
else:
    print('no match')

m = p.search('case')
if m:
    print('match')
else:
    print('no match')

m = p.search('seek')
if m:
    print('match')
else:
    print('no match')
```

출력 결과

```
no match
match
no match
```

- findall() 메서드는 일치하는 모든 것을 리스트 타입으로 반환한다.

[02.정규식] ex04.py

```
import re

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

list = p.findall('careless')
print(list)

list = p.findall('good care')
print(list)

list = p.findall('good care cafe')
print(list)

list = p.findall('case care cafe')
print(list)

list = p.findall('caffe')
print(list)
```

출력 결과

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

- 정규식을 체크하기 위한 check_match() 함수를 작성한 후 적용한다.

[02.정규식] ex05.py

```
import re

def check_match(p, word):
    m = p.search(word)
    if m:
        print(f'{word} match')
    else:
        print(f'{word} no match')

print('정규식: ca.e', '-' * 50)
p = re.compile('ca.e')
check_match(p, 'care')
check_match(p, 'caffe')
print('')

print('정규식: ^de', '-' * 50)
p = re.compile('^de')
check_match(p, 'desk')
check_match(p, 'fade')
check_match(p, 'destination')
print('')

print('정규식: se$', '-' * 50)
p = re.compile('se$')
check_match(p, 'case')
check_match(p, 'seek')
check_match(p, 'base')
```

---

출력 결과

```
정규식:ca.e ----------------------------------------------------
care match
caffe no match

정규식:^de ----------------------------------------------------
desk match
fade no match
destination match

정규식: se$ ----------------------------------------------------
case match
seek no match
base match
```

---

- 정규식에서 사용하는 각종 함수를 사용해 본다.

| 메서드 | 목적 |
|---|---|
| group() | 일치하는 문자열만 반환 |
| string | 입력받은 문자열 |
| start() | 일치하는 문자열의 시작 인덱스 |
| end() | 일치하는 문자열의 끝 인덱스 |
| span() | 일치하는 문자열의 시작, 끝 인덱스 |

**[02.정규식] ex06.py**

```
import re

p = re.compile('ca.e')
m = p.search('good care')
if m:
    print('m.group()', m.group())
    print('m.string', m.string)
    print('m.start()', m.start())
    print('m.end()', m.end())
    print('m.span()', m.span())
else:
    print('no match')
```

**출력 결과**

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

**[02.정규식] ex07.py**

```
import re

p = re.compile('^de')
m = p.search('desk')
if m:
    print('m.group()', m.group())
    print('m.string', m.string)
    print('m.start()', m.start())
    print('m.end()', m.end())
    print('m.span()', m.span())
else:
    print('no match')
```

**출력 결과**

```
m.group() de
m.string desk
m.start() 0
m.end() 2
m.span() (0, 2)
```

**[02.정규식] ex08.py**

```
import re

p = re.compile('se$')
m = p.search('case')
if m:
    print('m.group()', m.group())
    print('m.string', m.string)
    print('m.start()', m.start())
    print('m.end()', m.end())
    print('m.span()', m.span())
else:
    print('no match')
```

**출력 결과**

```
m.group() se
m.string case
m.start() 2
m.end() 4
m.span() (2, 4)
```

- **각종 메서드를 사용하는 함수를 정의해 적용한다.**

**[02.정규식] ex09.py**

```
import re

def check_match(p, word):
    print('단어:', word)
    m = p.search(word)
    if m:
        print('m.group():', m.group())
        print('m.string:', m.string)
        print('m.start():', m.start())
        print('m.end()', m.end())
        print('m.span()', m.span())
    else:
        print('no match', word)

print('정규식:ca.e', '-' * 50)
p = re.compile('ca.e')
check_match(p, 'careless')

print('정규식:^de', '-' * 50)
p = re.compile('^de')
check_match(p, 'destination')

print('정규식:se$', '-' * 50)
p = re.compile('se$')
check_match(p, 'base')
check_match(p, 'face')
```

**출력 결과**

```
정규식:ca.e --------------------------------------------------
단어: careless
m.group(): care
m.string: careless
m.start(): 0
m.end() 4
m.span() (0, 4)
정규식:^de --------------------------------------------------
단어: destination
m.group(): de
m.string: destination
m.start(): 0
m.end() 2
m.span() (0, 2)
정규식:se$ --------------------------------------------------
단어: base
m.group(): se
m.string: base
m.start(): 2
m.end() 4
m.span() (2, 4)
단어: face
no match face
```

# 03. 스크랩핑(Scraping)

**인터넷에서 데이터를 수집하는 기술에는 스크랩핑(scraping)과 크롤링(crawing)이 있다. 스크랩핑은 정적 웹페이지에서 특정 데이터를 추출하는 것이고 크롤링은 크롤러 또는 스파이더라는 프로그램을 사용하여 동적 웹페이지에서 데이터를 추출하는 것이다.**

- **스크랩핑에 필요한 패키지(beautifulSoup4)와 HTML 구조를 분석하는 파싱(parsing) 패키지(lxml)를 설치한다.**

```
pip install beautifulSoup4
pip install lxml
```

## • CGV 무비차트에서 스크랩핑

- **CGV 무비차트에서 처음 발견되는 a 태그의 속성들 정보 출력**

**[03.스크랩핑] ex01.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')

print('1:', soup.title) #title 태그를 출력
print('2:', soup.title.get_text()) #title 태그의 text값을 출력
print('3:', soup.a) #처음 발견되는 a element를 반환
print('4:', soup.a.attrs) #a element의 속성들 정보 출력
print('5:', soup.a['href']) #a element의 href 속성 '값' 정보 출력
```

**출력 결과**

```
1: <title id="ctl00_headerTitle">무비차트 &lt; 무비차트 | 깊이 빠져 보다, CGV</title>
2: 무비차트 < 무비차트 | 깊이 빠져 보다, CGV
3: <a href="#contents" id="skipHeader">메인 컨텐츠 바로가기</a>
4: {'href': '#contents', 'id': 'skipHeader'}
5: #contents
```

- **CGV 무비차트의 1위 영화의 형제, 부모 태그**

**[03.스크랩핑] ex02.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')

chart = soup.find('div', attrs={'class':'sect-movie-chart'})
rank1 = chart.find('strong', attrs={'class':'rank'})
print('1:', rank1.getText())

rank1_sibling = rank1.find_next_sibling('a')
print('2:', 'http://cgv.co.kr' + rank1_sibling['href'])

rank1_grand=rank1.parent.parent
title1 = rank1_grand.find('strong', attrs={'class':'title'} )
print('3:', title1.getText())
```

**출력 결과**

```
1: No.1
2: http://cgv.co.kr/movies/detail-view/?midx=89375
3: 말할 수 없는 비밀
```

- **CGV 상용 중인 영화제목 목록**

**[03.스크랩핑] ex03.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:
    #movie.getText() 동일
    print(movie.get_text())
```

**출력 결과**

```
말할 수 없는 비밀
히트맨2
검은 수녀들
브로큰
...
```

- **CGV 상용 중인 영화제목과 예매사이트 목록**

**[03.스크랩핑] ex04.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')

sect_movie = soup.find('div', attrs={'class':'sect-movie-chart'})
movies = sect_movie.find_all('li')

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']
    #link 출력 결과에서 Ctrl + 클릭하면 예매사이트로 접속한다.
    print(title, link)
```

**출력 결과**

```
말할 수 없는 비밀 http://www.cgv.co.kr/ticket/?MOVIE_CD=20039857&MOVIE_CD_GROUP=20039857
히트맨2 http://www.cgv.co.kr/ticket/?MOVIE_CD=20040036&MOVIE_CD_GROUP=20039813
검은 수녀들 http://www.cgv.co.kr/ticket/?MOVIE_CD=20040005&MOVIE_CD_GROUP=20040004
브로큰 http://www.cgv.co.kr/ticket/?MOVIE_CD=20039948&MOVIE_CD_GROUP=20039948
...
```

- **CGV 평균 예매율 출력 (터미널에서 python 실행 후 작성도 가능하다.)**

**[03.스크랩핑] ex05.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')

sect_movie = soup.find('div', attrs={'class':'sect-movie-chart'})
movies = sect_movie.find_all('li')

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(f'평균 예매율:{total_percent / len(movies)}')
```

## - 네이버 증권 스크랩핑

- **[네이버]-[네이버 증권]-[Top 종목]-[거래상위] 목록을 출력**

**[03.스크랩핑] ex06.py**

```
import requests
from bs4 import BeautifulSoup

url = 'https://finance.naver.com/'
res = requests.get(url)
res.raise_for_status()
soup = BeautifulSoup(res.text, 'lxml')

items = soup.find('tbody', attrs={'id':'_topItems1'})

rank1 = items.find('tr')
print(rank1.a.get_text())

# next_sibling 다음 요소로 넘기는 메서드는 개행이 존재하기 때문에 두 번 넘겨야한다.
rank2 = rank1.next_sibling.next_sibling
print(rank2.a.get_text())

#find_next_sibling() 함수를 이용하면 개행 문자가 있는지 확인할 필요가 없다.
rank3 = rank2.find_next_sibling('tr')
print(rank3.a.get_text())

rank2 = rank3.find_previous_sibling('tr')
print(rank2.a.get_text())

#parent 메서드는 rank1의 부모 관련 요소를 가져온다.
parent = rank1.parent
rank1 = parent.find('tr')
print(rank1.a.get_text())

#find_next_siblings() 함수는 rank1의 형제 요소들을 가져온다.
siblings = rank1.find_next_siblings('tr')
print(len(siblings))
rank2 = siblings[0]
print(rank2.find('a').get_text())
```

- **[네이버]-[네이버 증권]-[인기검색종목] 목록을 출력**

**[03.스크랩핑] ex07.py**

```
import requests
from bs4 import BeautifulSoup

url = 'https://finance.naver.com/'
res = requests.get(url)
res.raise_for_status()
soup = BeautifulSoup(res.text, 'lxml')

popular = soup.find('div', attrs={'class':'aside_area aside_popular'})
items = popular.find_all('a')
for item in items:
    print(item.get_text())
```

---

- **[네이버]-[네이버 증권]-[해외증시] 목록을 출력**

**[03.스크랩핑] ex08.py**

```
import requests
from bs4 import BeautifulSoup

url = 'https://finance.naver.com'
res = requests.get(url)
res.raise_for_status()
soup = BeautifulSoup(res.text, 'lxml')

overseas = soup.find('div', attrs={'class':'aside_area aside_stock'})
items = overseas.find_all('a')

for item in items:
    title = item.get_text()
    link = url + item['href']
    print('-' * 100)
    print(title)
    print(link)
```

---

- **[네이버]-[네이버 증권]-[주요뉴스] 목록을 출력**

**[03.스크랩핑] ex09.py**

```
import requests
from bs4 import BeautifulSoup

url = 'https://finance.naver.com'
res = requests.get(url)
res.raise_for_status()
soup = BeautifulSoup(res.text, 'lxml')

news = soup.find('div', attrs={'class':'news_area _replaceNewsLink'})
items = news.find_all('a')

for item in items:
    title = item.get_text()
    link = url + item['href']
    print('-' * 100)
    print(title)
    print(link)
```

---

- **[구글]-[송중기]-[이미지] 검색 후 제목 출력**

[03.스크랩핑] ex10.py

```
import requests
from bs4 import BeautifulSoup

url = 'https://www.google.com/search?sca_esv=1eda6e0add178775&q=송중기&udm=2&...&dpr=0.8'
headers = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Chrome/131.0.0.0 Safari/537.36',
           'Accept-Language':'ko-KR,ko;q=0.8,en-US;q=0.5,en;q=0.3'}

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

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

with open('data/song.html', 'w', encoding='utf-8') as f:
    f.write(soup.prettify())

images = soup.find_all('a', attrs={"class":"EZAeBe"})
print(len(images))

for idx, image in enumerate(images):
    title = image.find("div", attrs={"class":"tol8Rb OSrXXb"})
    print('-' * 100)
    print(idx+1, title.get_text())
    print(image['href'])
```

---

## • 웹 스크래핑 데이터 CSV 파일저장

- **[네이버]-[증권]-[Top종목]-[거래상위]-[더보기]-[코스닥] CSV 파일저장**

[03.스크랩핑] ex11.py

```
import csv
import requests
from bs4 import BeautifulSoup
import re

filename = "data/코스닥거래상위1-100.csv"
f = open(filename, "w", encoding="utf-8-sig", newline="")
writer = csv.writer(f)

url = "https://finance.naver.com/sise/sise_quant.naver?sosok=1"
res = requests.get(url)
res.raise_for_status()

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

rows = soup.find("table", attrs={"class":"type_2"}).find_all("tr")

for row in rows:
    columns = row.find_all("td")
    #<tr><td colspan="10"></td></tr>인 경우 skip
    if len(columns) <= 1:
        continue
    #re.sub(pattern, replace, text) : text 중 pattern에 해당하는 부분을 replace로 대체한다.
    data = [re.sub("\t|\n|상승|하락|보합", "", column.get_text()) for column in columns]
    writer.writerow(data)
```

---

- **[네이버]-[증권]-[시가총액 순위]-[더보기]-[코스피] CSV 파일저장**

[03.스크랩핑] ex12.py

```
import csv
import requests
from bs4 import BeautifulSoup
import re

filename = "data/시가총액1-200.csv"
f = open(filename, "w", encoding="utf-8-sig", newline="")
writer = csv.writer(f)
title = "N	종목명	현재가	전일비	등락률	액면가	시가총액	상장주식수	외국인비율	거래량	PER	ROE".split("\t")
writer.writerow(title)

for page in range(1, 5):
    url=f"https://finance.naver.com/sise/sise_market_sum.naver?&page={page}"
    res = requests.get(url)
    res.raise_for_status()

    soup = BeautifulSoup(res.text, "lxml")
    rows = soup.find("table", attrs={"class":"type_2"}).find("tbody").find_all("tr")
    for row in rows:
        columns = row.find_all("td")
        if len(columns) <= 1:
            continue
        data=[re.sub("\t|\n|상승|하락|보합","", column.get_text()) for column in columns]
        writer.writerow(data)
```

---

- **구글에서 '할리스' 검색 후 [할리스 커피]-[Store]-[매장검색] CSV 파일저장**

[03.스크랩핑] ex13.py

```
import requests
from bs4 import BeautifulSoup
import csv

f = open('data/할리스.csv', "w", encoding="utf-8-sig", newline="") #결과를 저장할 파일을 생성한다.
writer = csv.writer(f)

seq = 0
for page in range(1, 11): #1페이지 ~ 10페이지까지 반복
    url=f'https://www.hollys.co.kr/store/korea/korStore2.do?pageNo={page}&sido=&gugun=&store='
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, "lxml")

    tbody = soup.find('tbody')
    es = tbody.find_all('tr')
    print(f'========== {page} ==========')
    for index, e in enumerate(es):
        seq += 1
        store = e.find_all('td') #매장정보
        store_name = store[1].get_text().strip() #매장이름
        store_city = store[0].get_text().strip() #매장시도
        store_address = store[3].get_text().strip().replace(' .', '') #매장주소
        store_phone = store[5].get_text().strip() #매장전화
        data = [seq, store_name, store_city, store_address, store_phone] #매장정보를 list 자료형에 저장
        writer.writerow(data)
        print(seq, store_name, store_city, store_address, store_phone)
```

---

- **네트워크 이미지 다운로드**

- **CGV 무비차트 이미지 다운로드**

[03.스크랩핑] ex14.py

```
import requests
from bs4 import BeautifulSoup
import os

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

chart = soup.find("div", attrs={"class":"sect-movie-chart"})
movies = chart.find_all("div", attrs={"class":"box-image"})

if not os.path.exists("movies"):
    os.mkdir("movies")

for index, item in enumerate(movies):
    image = item.find('img')["src"]
    res_image = requests.get(image)
    res_image.raise_for_status()

    with open("movies/movie{}.jpg".format(index + 1), "wb") as f:
        f.write(res_image.content)

    print(f'{index}:{image}')
```

---

- **[할리스 커피]-[Menu]-[COFFEE] 메뉴 이미지 다운로드**

[03.스크랩핑] ex15.py

```
import requests
from bs4 import BeautifulSoup
import os

url='https://www.hollys.co.kr/menu/espresso.do'
res = requests.get(url)
res.raise_for_status()
soup = BeautifulSoup(res.text, "lxml")

ul = soup.find('ul', attrs={'class':'menu_list01 mar_t_40'})
es = ul.find_all('img')

if not os.path.exists("hollys"):
    os.mkdir("hollys")

for index, e in enumerate(es):
    url = 'http:' + e['src']
    res_image = requests.get(url)
    res_image.raise_for_status()

    with open("hollys/menu{}.jpg".format(index + 1), "wb") as f:
        f.write(res_image.content)

    print(f'{index}:{url}')
```

---

- **스크랩핑 프로젝트**

- **[네이버 검색]-[서울 날씨] 검색 후 날씨 정보 아래와 같이 출력**

**출력 결과**

```
[오늘의 날씨]
------------------------------------------------
어제보다 4.5° 낮아요 비
현재 23.8° (최저 24° /초고 27° )
오전 강수확률 90% / 오후 강수확률 90%

미세먼지 좋음
초미세먼지 좋음
------------------------------------------------
```

**[03.스크랩핑] ex16.py**

```
import requests
from bs4 import BeautifulSoup

def scrap_weather():
    url ='https://search.naver.com/search.naver?where=nexearch&sm=top_hty&fbm=0&ie=utf8&query=오늘의 날씨'
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, 'lxml')

    #어제보다 1.1° 낮아요 비
    cast = soup.find('p', attrs={'class':'summary'}).get_text()

    #현재온도
    temp = soup.find('div', attrs={'class':'temperature_text'})
    curr_temp = temp.strong.contents[1]
    celsius = temp.strong.contents[2].get_text()
    curr_temp = curr_temp + celsius

    #최저온도 / 최고온도
    lowest = soup.find('span', attrs={'class':'lowest'}).contents[1]
    highest = soup.find('span', attrs={'class':'highest'}).contents[1]

    #오전 / 오후 강수확률
    rainfall = soup.find_all('span', attrs={'class':'weather_inner'})
    moring = rainfall[0].find('span', attrs={'class':'rainfall'}).get_text()
    afternoon = rainfall[1].find('span', attrs={'class':'rainfall'}).get_text()

    #미세먼지 / 초미세먼지
    chart = soup.find('ul', attrs={'class':'today_chart_list'})
    level = chart.find_all('li', attrs={'class':'item_today level1'})
    level0 = level[0].find('span', attrs={'class':'txt'}).get_text()
    level1 = level[1].find('span', attrs={'class':'txt'}).get_text()

    print('[오늘의 날씨]')
    print('-' * 50)
    print(cast)
    print(f'현재 {curr_temp} (최저 {lowest} /초고 {highest})')
    print(f'오전 강수확률 {moring} / 오후 강수확률 {afternoon}\n')
    print(f'미세먼지 {level0}')
    print(f'초미세먼지 {level1}')
    print('-' * 50)

if __name__ == '__main__':
    scrap_weather()
```

- **[네이버 뉴스] 'IT/과학', '경제' 클릭 후 뉴스 정보 아래와 같이 3까지만 출력**

**1) 클래스명이 className1이거나 className2 검색 attrs={"class":["className1", "className2"]}**
**2) 클래스명이 className이고 아이디가 idName 검색 attrs={"class":"className", "id":"idName"}**
**3) 클래스명이 className이고 텍스트 내용이 미세먼지이거나 초미세먼지 검색 attrs={"class":"className"}, text=["미세먼지", "초미세먼지"]**

**출력 결과**

```
[IT/과학] 뉴스
1. 삼성SDS 지난해 영업익 12.7% 늘어…클라우드 사업 호조
  (링크: https://n.news.naver.com/mnews/article/009/0005433842)
2. 삼성 노태문 "갤럭시 S25로 모바일 AI 혁신 현실화"
  (링크: https://n.news.naver.com/mnews/article/374/0000422198)
3. 레전드 오브 이미르, 블록체인 기술로 '투명성·재미' 잡은 대작
  (링크: https://n.news.naver.com/mnews/article/421/0008039996)
[경제] 뉴스
1. 내수 부진에 탄핵정국까지…작년 한국 경제 성장률 2%, 4분기 0.1%에 그쳐
  (링크: https://n.news.naver.com/mnews/article/087/0001094510)
2. PF 사업장 거래 활성화한다…당국, 정보공개 플랫폼 구축
  (링크: https://n.news.naver.com/mnews/article/018/0005930812)
3. 최 대행 "APEC, 절호의 기회…정치·경제 회복력 보여야"
  (링크: https://n.news.naver.com/mnews/article/055/0001225912)
[생활/문화] 뉴스
1. “中서 유입” vs “한중 합작” … 미세먼지 원인 갑론을박 [미드나잇 이슈]
  (링크: https://n.news.naver.com/mnews/article/022/0004005165)
2. 스텔란티스코리아, 지프·푸조 통합 전시장 확대
  (링크: https://n.news.naver.com/mnews/article/014/0005299379)
3. 뉴진스, 새 활동명 공모..."절대 돌아갈 생각 없어"
  (링크: https://n.news.naver.com/mnews/article/052/0002144705)
```

**[03.스크랩핑] ex17.py**

```
import requests
from bs4 import BeautifulSoup

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

def scrap_news(url):
    soup = create_soup(url)
    news_list = soup.find('ul', attrs={'class':'sa_list'}).find_all('li', limit=3)
    for index, news in enumerate(news_list):
        title = news.find('strong', attrs={'class':'sa_text_strong'}).get_text()
        link = news.find('a')['href']
        print(f'{index+1}. {title}')
        print(f'  (링크: {link})')

if __name__ == '__main__':
    print('[IT/과학] 뉴스')
    url ='https://news.naver.com/section/105'
    scrap_news(url)
    print()
    print('[경제] 뉴스')
    url = 'https://news.naver.com/section/101'
    scrap_news(url)
    print()
    print('[경제] 뉴스')
    url = 'https://news.naver.com/section/103
    scrap_news(url)
```

- **해커스의 오늘의 영어 회화 정보 출력**

**네이버에서 '해커스'단어 검색 후 [해커스토익]-[기초영어/회화]-[매일영어회화학습]**

**출력 결과**

```
[오늘의 영어 회화]
(영어지문)
Rob: Hello, I am Rob Ketterman. I am here to interview for a sales position.
Danielle: Oh yes. Did I spell your name correctly on this form?
Rob: It's k-e-t-t-e-r-m-a-n.
Danielle: Oh, let me just correct that.

(한글지문)
Rob: 안녕하세요. Rob Ketterman입니다. 영업직 인터뷰에 응시하러 왔습니다.
Danielle: 아 네. 제가 이 양식에 당신의 이름을 정확하게 기재했나요?
Rob: k-e-t-t-e-r-m-a-n 입니다.
Danielle: 아, 그럼 좀 고쳐야겠네요.
```

**[03.스크랩핑] ex18.py**

```
import requests
from bs4 import BeautifulSoup
import re

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

def scrap_english():
    print('[오늘의 영어 회화]')
    url = 'https://www.hackers.co.kr/?c=s_eng/eng_contents/I_others_english&keywd=...&logger_kw=...'
    soup = create_soup(url)

    sentences = soup.find_all('div', attrs={'id':re.compile('^conv_kor_t')})

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

    print('(한글지문)')
    for sentence in sentences[:len(sentences)//2]:
        print(sentence.get_text().strip())

if __name__ == '__main__':
    scrap_english()
```

**[03.스크랩핑] ex19.py**

```
import ex16, ex17, ex18

if __name__ == '__main__':
    ex16.scrap_weather()

    print('[IT/과학] 뉴스')
    url ='https://news.naver.com/section/105'
    ex17.scrap_news(url)

    ex18.scrap_english()
```

# 04. 크롤링(Crawling)

**selenium은 웹 사이트 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.**

- **Selenium 패키지를 설치한다.**

```
pip install selenium
```

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

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

## • Selenium 기본

- **selenium의 webdriver를 이용한 크롤링**

[04.크롤링] ex01.py

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

browser = webdriver.Chrome()
browser.get("http://naver.com")

#로그인 버튼의 클래스 명
e = browser.find_element(By.CLASS_NAME, "MyView-module__link_login___HpHMW")
e.click()

browser.back()
browser.forward()
browser.refresh()
browser.back()

#검색어 입력 상자에 '나도코딩'을 입력 후 엔터
e = browser.find_element(By.ID, "query")
e.send_keys("나도코딩")
e.send_keys(Keys.ENTER)

#태그명이 'a'인 모든 태그 검색
es = browser.find_elements(By.TAG_NAME, "a")
for idx, e in enumerate(es):
    link = e.get_attribute("href")
    print(idx+1, link)

#다음 페이지로 이동
browser.get("http://daum.net")

#검색어 상자를 찾아 '나도코딩' 입력
e = browser.find_element(By.NAME, "q")
e.send_keys("나도코딩")

#검색 버튼을 찾아 클릭
e = browser.find_element(By.XPATH, '//*[@id="daumSearch"]/fieldset/div/div/button[3]')
e.click()

time.sleep(10)
```

- **Selenium 심화**

- **네이버 사이트 자동 로그인 작업**

[04.크롤링] ex02.py

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

browser = webdriver.Chrome()

#1. 네이버 이동
browser.get("http://naver.com")

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

#3. 아이디, 비밀번호 입력
browser.find_element(By.ID, "id").send_keys("naver_id")
browser.find_element(By.ID, "pw").send_keys("naver_pw!")

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

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

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

#7. 브라우저 종료
browser.close() #현재 탭 종료
browser.quit() #브라우저 종료
time.sleep(10)
```

- **브라우저 화면을 항상 띄우고 화면을 최대화한 후 화면 캡쳐해 저장**

[04.크롤링] ex03.py

```
from selenium import webdriver

options = webdriver.ChromeOptions()
options.add_experimental_option("detach", True)  #브라우저가 안 꺼지는 옵션
browser = webdriver.Chrome(options=options)

#브라우저 창 최대화
browser.maximize_window()

#네이버 항공권 페이지 이동
url = "https://flight.naver.com/"
browser.get(url)

browser.get_screenshot_as_file("data/fight.png")
print("프로그램종료")
browser.quit()
```

- 크롬 브라우저를 띄우지 않고 Background에서 빠르게 Scrapping이 가능하다. (브라우저를 띄우면 메모리도 많이 필요하다)

[04.크롤링] ex04.py

```
from selenium import webdriver

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

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

browser.get_screenshot_as_file("data/google.png")
print("프로그램종료")
```

---

- 'HeadlessChrome'이고 'user agent'가 미 설정 된 경우 'user agent' 값이 날아가서 브라우저의 접속을 막을 수 있다.

[04.크롤링] ex05.py

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

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

#구글에서 user agent string 검색 후 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()
```

---

출력 결과

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/131.0.0.0 Safari/537.36
```

---

- 옵션으로 'user agent'를 추가로 설정하면 'user agent'가 정상적 적용된다.

[04.크롤링] ex05.py

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

options = webdriver.ChromeOptions()
options.add_argument('headless')
options.add_argument("window-size=1920x1080")
options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Chrome/131.0.0.0 Safari/537.36')
browser = webdriver.Chrome(options=options)
...
```

---

출력 결과

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36
```

---

- **쿠팡 쇼핑몰 크롤링**

- **[구팡]-[검색어]-[노트북] 1페이지 검색 목록 출력**

**[04.크롤링] ex06.py**

```
from selenium import webdriver
import re
from bs4 import BeautifulSoup

options = webdriver.ChromeOptions()
options.add_argument('headless')
options.add_argument("window-size=1920x1080")
options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Chrome/131.0.0.0 Safari/537.36')
browser = webdriver.Chrome(options=options)

url = "https://www.coupang.com/np/search?component=&q=노트북&channel=user"
browser.get(url)

soup = BeautifulSoup(browser.page_source, 'lxml')
items = soup.find_all("li", attrs={"class":re.compile("^search-product")})

for index, item in enumerate(items):
    name=item.find("div", attrs={"class": "name"}).get_text()  #상품이름
    price=item.find("strong", attrs={"class", "price-value"}).get_text()  #상품가격

    rate = item.find("em", attrs={"class": "rating"})  #상품평점
    if rate:
        rate = rate.get_text()
    else:
        rate = "평점없음"

    rate_cnt = item.find("span", attrs={"class": "rating-total-count"})  #상품평
    if rate_cnt:
        rate_cnt = rate_cnt.get_text()
    else:
        rate_cnt = "상품평없음"

    print(f'{index+1}.{name.strip()}')
    print(f'상품가격:{price}원')
    print(f'상품평점:{rate}')
    print(f'상품평:{rate_cnt[1:-1]}개')
    print("-" * 100)
```

**[04.크롤링] ex06.py 결과**

```
1.[풀박스패키지] LG그램15 코어i5(10세대/ 램 16G/ SSD 512G/ 윈도우11프로 초경량 1.09kg(전시몰닷컴), 15ZB995, WIN11 Pro, 16GB, 512GB
상품가격:784,000원
상품평점:5.0
상품평:349개
----------------------------------------------------------------------------------------------------
2.삼성 LG HP 노트북 i5 가정 업무 게임 포토샵 주식용 Win10/11 무상1년 사은품, 실버, LG 4세대i5 15N540, 500GB, 8GB, WIN10 Pro
상품가격:289,000원
상품평점:4.5
상품평:194개
----------------------------------------------------------------------------------------------------
...
56.베이직스 2023 베이직북14 프로 코어i5 인텔 10세대, 화이트, 512GB, 16GB, WIN11 Home, BP1423FW
상품가격:690,000원
상품평점:4.5
상품평:260개
----------------------------------------------------------------------------------------------------
```

- **삼성 제품 제외 평점 4.5점 이상, 리뷰 100개 이상인 제품만 출력**

**[04.크롤링] ex07.py**

```
from selenium import webdriver
import re
from bs4 import BeautifulSoup

options = webdriver.ChromeOptions()
options.add_argument('headless')
options.add_argument("window-size=1920x1080")
options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Chrome/131.0.0.0 Safari/537.36')
browser = webdriver.Chrome(options=options)

url = "https://www.coupang.com/np/search?component=&q=노트북&channel=user"
browser.get(url)

soup = BeautifulSoup(browser.page_source, 'lxml')
items = soup.find_all("li", attrs={"class":re.compile("^search-product")})

cnt = 0
for index, item in enumerate(items):
    name=item.find("div", attrs={"class":"name"}).get_text()

    if '삼성' in name:  #애플 제품 제외
        print("<삼성 상품 제외 합니다>")
        print("-" * 100)
        continue
    price=item.find("strong", attrs={"class", "price-value"}).get_text()

    rate = item.find("em", attrs={"class":"rating"})  #상품평점
    if rate:
        rate = rate.get_text()
    else:
        print("<평점 없는 상품 제외 합니다>")
        print("-" * 100)
        continue

    rate_cnt = item.find("span", attrs={"class":"rating-total-count"})  #평점수
    if rate_cnt:
        rate_cnt = rate_cnt.get_text()
        rate_cnt = rate_cnt[1:-1]
    else:
        print("<평점 수 없는 상품 제외 합니다>")
        print("-" * 100)
        continue

    #평점 4.5 이상, 리뷰 100개 이상만 조회
    if float(rate) >= 4.5 and int(rate_cnt) >= 100:
        cnt += 1
        print(index+1, name.strip(), price, rate, rate_cnt)
        print("-" * 100)
        link = item.find('a', attrs={"class":"search-product-link"})['href']
        index += 1
        print(f'상품이름:{name}')
        print(f'상품가격:{price}원')
        print(f'상품평점:{rate}점({rate_cnt}개)')
        print("바로가기:{}".format("https://www.coupang.com" + link))
        print("-" * 100)
print(f'조건을 만족하는 상품수:{cnt}개')
```

- **쿠팡에서 노트북 검색 후 1페이지의 Thumbnail 이미지 다운로드**

[04.크롤링] ex08.py

```
from selenium import webdriver
import re
from bs4 import BeautifulSoup
import os
import requests

options = webdriver.ChromeOptions()
options.add_argument('headless')
options.add_argument("window-size=1920x1080")
options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.0.0 Safari/537.36')
browser = webdriver.Chrome(options=options)

url = "https://www.coupang.com/np/search?component=&q=노트북&channel=user"
browser.get(url)

soup = BeautifulSoup(browser.page_source, 'lxml')
items = soup.find_all("li", attrs={"class":re.compile("^search-product")})

if not os.path.exists("product"):
    os.mkdir("product")

for index, item in enumerate(items):
    image=item.find("img", attrs={"class":"search-product-wrap-img"})["src"]
    if 'thumbnail' in image:
        image = "https:" + image
        res_image = requests.get(image)  #이미지를 요청
        res_image.raise_for_status()  #응답 상태 코드를 확인하고, 오류가 발생하면 예외를 발생

        with open("product/product{}.jpg".format(index+1), "wb") as f:  #이미지를 파일로 저장
            f.write(res_image.content)
print(f'이미지 다운로드 완료')
```

- **쿠팡 페이지 접속 후 스크롤을 페이지 가장 아래로 이동**

[04.크롤링] ex09.py

```
from selenium import webdriver
import time

options = webdriver.ChromeOptions()
options.add_experimental_option("detach", True)
browser = webdriver.Chrome(options=options)
browser.maximize_window()
url = "https://www.coupang.com/np/search?component=&q=노트북&channel=user"
browser.get(url)

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

while True:
    browser.execute_script("window.scrollTo(0, document.body.scrollHeight)")
    time.sleep(interval)

    curr_height = browser.execute_script("return document.body.scrollHeight")  #이전 스크롤높이와 현재 스크롤높이가 같으면 break
    if prev_height == curr_height:
        break
    else:
        prev_height = curr_height
print("스크롤 완료")
```

- **Selenium 활용**

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

[04.크롤링] ex10.py

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

#브라우저 창 항상 유지
options = webdriver.ChromeOptions()
options.add_experimental_option("detach", True)
browser = webdriver.Chrome(options=options)

#브라우저 창 최대화
browser.maximize_window()

#1.네이버 항공권 페이지 이동
url = "https://flight.naver.com/"
browser.get(url)

#잠시 기다리는 시간 지정(초)
wait = 2

#2.가는 날 선택
e = browser.find_element(By.XPATH, '//button[text()="가는 날"]')
e.click()
time.sleep(wait)

#3.이번달 25일 선택
es = browser.find_elements(By.XPATH, '//b[text()="25"]')
es[0].click()
time.sleep(wait)

#4.이번달 26일 선택
es = browser.find_elements(By.XPATH, '//b[text()="26"]')
es[0].click()
time.sleep(wait)

#5.도착지 선택
e = browser.find_element(By.XPATH, '//b[text()="도착"]')
e.click()
time.sleep(wait)

#6.제주 선택
e = browser.find_element(By.XPATH, '//button[contains(text(), "제주")]')
e.click()

try:
    #첫 번째 결과를 출력할 때까지 최대 10초 기다린다.
    first = '//*[@id="__next"]/div/main/section/div/div/div/div/a[1]/div/button[1]'
    WebDriverWait(browser, 10).until(EC.presence_of_all_elements_located((By.XPATH, first)))
    #첫 번째 결과 출력
    e = browser.find_element(By.XPATH, first)
    print(e.text)
finally:
    browser.quit()
```

**[04.크롤링] ex11.py**

```
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

#브라우저 창 항상 유지
options = webdriver.ChromeOptions()
options.add_experimental_option("detach", True)
browser = webdriver.Chrome(options=options)

#브라우저 창 최대화
browser.maximize_window()

#1.네이버 항공권 페이지 이동
url = "https://flight.naver.com/"
browser.get(url)

#xpath를 찾을 때 까지 기다리는 함수
def wait_until(xpath):
    WebDriverWait(browser, 10).until(EC.presence_of_all_elements_located((By.XPATH, xpath)))

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

#3.이번달 25일 선택
xpath = '//b[text()="25"]'
wait_until(xpath)
es = browser.find_elements(By.XPATH, xpath)
es[0].click()

#4.이번달 26일 선택
xpath = '//b[text()="26"]'
wait_until(xpath)
es = browser.find_elements(By.XPATH, xpath)
es[0].click()

#5.도착지 선택
xpath = '//b[text()="도착"]'
wait_until(xpath)
e = browser.find_element(By.XPATH, xpath)
e.click()

#6.제주 선택
xpath = '//button[contains(text(), "제주")]'
wait_until(xpath)
e = browser.find_element(By.XPATH, xpath)
e.click()

try:
    #첫 번째 결과를 출력할 때까지 최대 10초 기다린다.
    first = '//*[@id="__next"]/div/main/section/div/div/div/div/a[1]/div/button[1]'
    wait_until(first)
    #첫 번째 결과 출력
    e = browser.find_element(By.XPATH, first)
    print(e.text)
finally:
    browser.quit()
```

- **크롤링 프로젝트**

- **[지마켓]-[검색어 입력]-[검색] 결과출력**

[04.크롤링] ex12.py

```
from selenium import webdriver
import time

options = webdriver.ChromeOptions()
options.add_argument('headless')
options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Chrome/131.0.0.0 Safari/537.36')
browser = webdriver.Chrome(options=options)
browser.maximize_window()
url='https://www.gmarket.co.kr/n/search?keyword=향수&k=53&p=1'
browser.get(url)

#스크롤을 페이지의 가장 아래로 이동
prev_height = browser.execute_script('return document.body.scrollHeight')
while True:
    browser.execute_script('window.scrollTo(0, document.body.scrollHeight)')
    time.sleep(2)
    curr_height = browser.execute_script('return document.body.scrollHeight')
    if prev_height == curr_height: break
    prev_height = curr_height

from bs4 import BeautifulSoup
import re
soup = BeautifulSoup(browser.page_source, 'lxml')
es = soup.find_all('div', attrs={'class':re.compile('^box__item-container')})

for index, e in enumerate(es):
    title = e.find('span', attrs={'class':'text__item'}) #상품 이름
    if title:
        title = title.get_text()
    else:
        title = ''

    image = e.find('img') #상품 이미지 주소
    if image:
        image = 'https:' + image['src']

    price = e.find('strong', attrs={'class':'text text__value'}) #상품 가격
    if price:
        price = price.get_text()
    else:
        price = ''

    pay_count = e.find('li', attrs={'class':re.compile('list-item__pay-count$')}) #구매건수
    if pay_count:
        pay_count = re.sub('구매|건', '', pay_count.get_text()).strip()
    else:
        pay_count = '0'
    #상품정보 출력
    print(f'{index+1}.{title}')
    print(f'이미지주소:{image}')
    print(f'상품가격:{price}')
    print(f'구매건수:{pay_count}')
    print('-' * 100)
browser.quit()
```

- **[당근마켓]-[검색어 입력]-[검색] 결과출력**

[04.크롤링] ex13.py

```
from selenium import webdriver
from bs4 import BeautifulSoup
import time
import re
import json

def create_soup(query):
    options = webdriver.ChromeOptions()
    options.add_argument('headless')
    options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Chrome/131.0.0.0 Safari/537.36')
    browser = webdriver.Chrome(options=options)
    browser.maximize_window()

    url=f'https://www.daangn.com/kr/buy-sell/?in=서구&search={query}'
    browser.get(url)

    #스크롤을 이동하여 페이지 마지막으로 이동
    prev_height = browser.execute_script('return document.body.scrollHeight')
    while True:
        browser.execute_script('window.scrollTo(0, document.body.scrollHeight)')
        time.sleep(1)
        curr_height = browser.execute_script('return document.body.scrollHeight')
        if prev_height == curr_height:
            print('스크롤완료!')
            break
        prev_height = curr_height

    soup = BeautifulSoup(browser.page_source, 'lxml')
    browser.quit()
    return soup

def search(query):
    soup=create_soup(query)
    es = soup.find_all('a', attrs={'class':re.compile('^_13tpfox6')})
    items = []
    for index, e in enumerate(es):
        title = e.find('h2', attrs={'class':re.compile('^_1b153uwh')}).get_text()
        address= e.find('div', attrs={'class':re.compile('^_1b153uwk')}).get_text().strip()
        price= e.find('div', attrs={'class':re.compile('^_1b153uwi')}).get_text().strip()
        image = e.find('img')['src']
        #스크랩핑 데이터를 딕션너리 자료형으로 변환
        data = {'no':index+1, 'title':title, 'address':address, 'price':price, 'image':image}
        items.append(data)
    return items

#당근마켓에서 '노트북'을 검색 후 결과를 저장한다.
json_data = search('노트북')

#딕션너리 자료형을 문자열로 변환
#ensure_ascii=False : 저장할 때 아스키 형태로 변환하지 않겠다는 의미.
json_str = json.dumps(json_data, indent=4, ensure_ascii=False)
print(json_str)

#JSON 자료형으로 파일에 저장
with open('data/당근.json', 'w', encoding='utf-8') as file:
    json.dump(json_data, file, ensure_ascii=False)
```

- **[기상청]-[전국] 전국 날씨출력**

**[04.크롤링] ex14.py**

```
from selenium import webdriver
from bs4 import BeautifulSoup
import re

options = webdriver.ChromeOptions()
options.add_argument('headless')
options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.0.0 Safari/537.36')

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

#1. [기상청]-[전국]-[현재] 페이지 이동
url='https://www.weather.go.kr/w/index.do'
browser.get(url)

#2. 스크랩핑
soup = BeautifulSoup(browser.page_source, 'lxml')

local = soup.find('div', attrs={'id':'weather2'})
es = local.find_all('dl', attrs={'class':re.compile('^po')})

for index, e in enumerate(es):
    title = e.find('dt').get_text()
    temp = e.find('span').get_text()
    weather = e.find('i').get_text()
    print(title, temp, weather)
```

---

**출력 결과**

```
백령도 5.4 맑음
인천 3.1 맑음
서울 6.4 맑음
파주 3.5 맑음
춘천 2.1 맑음
속초 5.9 구름많음
강릉 6.5 구름조금
울릉도독도 5.4 눈
수원 6.7 맑음
청주 4.4 맑음
안동 5.2 맑음
대전 6.7 맑음
홍성 5.1 맑음
전주 5.9 맑음
대구 7.2 맑음
울산 10.9 구름조금
포항 10.0 구름많음
부산 13.6 구름조금
창원 10.0 맑음
광주 9.4 맑음
목포 7.0 맑음
여수 8.5 맑음
흑산도 7.5 구름조금
제주 13.2 구름많음
```

---

- **[기상청]-[전국]-[어제] 어제 전국 날씨출력**

**[04.크롤링] ex15.py**

```
from selenium import webdriver
from bs4 import BeautifulSoup
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import re

options = webdriver.ChromeOptions()
options.add_argument('headless')
options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.0.0 Safari/537.36')

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

#1.기상청 홈페이지 이동
url = "https://www.weather.go.kr"
browser.get(url)

def wait_until(xpath):
    WebDriverWait(browser, 10).until(EC.presence_of_all_elements_located((By.XPATH, xpath)))

#2.전국 클릭
xpath = '//*[@id="content-body"]/div[4]/h2/a'
wait_until(xpath)
e = browser.find_element(By.XPATH, xpath)
e.click()

#3.어제 클릭
xpath = '//*[@id="content-body"]/div[4]/div/div/div[1]/ul/li[1]/a'
wait_until(xpath)
e = browser.find_element(By.XPATH, xpath)
e.click()

#4.전국 날씨정보 영역 찾기
xpath = '//*[@id="minmax"]'
wait_until(xpath)

#5.스크랩핑
soup = BeautifulSoup(browser.page_source, 'lxml')
local = soup.find('div', attrs={'id':'minmax'})
es = local.find_all('dl', attrs={'class':re.compile('^po')})
for index, e in enumerate(es):
    title = e.find('dt').get_text()
    red = e.find('span', attrs={'class', 'red'}).get_text()
    blue = e.find('span', attrs={'class', 'blue'}).get_text()
    print(f'{title}:최저({blue})/최고({red})')
```

**출력 결과**

```
백령도:최저(0.9)/최고(4.6)
인천:최저(-2.2)/최고(5.6)
서울:최저(-2.4)/최고(7.7)
파주:최저(-8.9)/최고(5.7)
...
흑산도:최저(4.5)/최고(8.1)
제주:최저(5.2)/최고(14.4)
```

- **[기상청]-[전국]-[예보] 요일별 오전, 오후 전국 예보 날씨출력**

**[04.크롤링] ex16.py**

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

options = webdriver.ChromeOptions()
options.add_argument('headless')
options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.0.0 Safari/537.36')
browser = webdriver.Chrome(options=options)
browser.maximize_window()

#1.기상청 홈페이지 이동
url = "https://www.weather.go.kr"
browser.get(url)

#2.전국 클릭
xpath = '//*[@id="content-body"]/div[4]/h2/a'
e = browser.find_element(By.XPATH, xpath)
e.click()
time.sleep(1)

#3.예보 클릭
xpath = '//*[@id="content-body"]/div[4]/div/div/div[1]/ul/li[3]'
e = browser.find_element(By.XPATH, xpath)
e.click()
time.sleep(1)

#4.요일을 반복하여 선택
for i in range(1, 8):
    xpath = f'//*[@id="local-weather"]/div/div[{i}]/h3/a'
    e = browser.find_element(By.XPATH, xpath)
    print(f'---------- {e.text.replace('\n', '')} ----------')
    e.click()
    time.sleep(1)

    #5.스크랩핑
    soup = BeautifulSoup(browser.page_source, 'lxml')
    local = soup.find('div', attrs={'id':'weather'})
    es = local.find_all('dl', attrs={'class':re.compile('^po')})
    for index, e in enumerate(es):
        title = e.find('dt').get_text()
        temp = e.find('span').get_text()
        weather = e.find('i').get_text()
        print(f'{title}:{temp}도({weather})')
```

**출력 결과**

```
---------- 24(금)오후 ----------
인천:7도(맑음)
서울:9도(맑음)
...
---------- 27(월)오후 ----------
인천:7도(맑음)
서울:9도(맑음)
...
```

- **네이버 부동산 사이트에서 '청라자이' 검색출력**

**[04.크롤링] ex17.py**

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

def browser_execute():
    options = webdriver.ChromeOptions()
    options.add_experimental_option("detach", True)

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

    e = browser.find_element(By.ID, "queryInputHeader")
    e.send_keys("루원시티")
    e.send_keys(Keys.ENTER)

    #크롤링 후 스크랩핑을 한다.
    from bs4 import BeautifulSoup
    soup = BeautifulSoup(browser.page_source, "lxml")

    es = soup.find_all("a", attrs={"class":"item_link"})

    for index, e in enumerate(es):
        print(f'========== 정보{index+1} ==========')
        title = e.find("div", attrs={"class":"title"})
        address = e.find("div", attrs={"class":"address"})
        type = e.find("strong", attrs={"class":"type"})
        info = e.find("div", attrs={"class":"info_area"})
        specs = info.find_all("span", attrs={"class":"spec"})

        #정보 출력
        print(title.get_text())
        print(address.get_text())
        print(type.get_text())
        for spec in specs:
            print(spec.get_text() + "|", end="")
        print('')
    browser.quit()

#함수 실행
browser_execute()
```

**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
...
```

- **[커피빈]-[STORE&서비스]-[매장찾기] Store ID 31번 매장정보출력**

[04.크롤링] ex18.py

```
from selenium import webdriver
from bs4 import BeautifulSoup
import time

options = webdriver.ChromeOptions()
options.add_argument('headless')
options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Chrome/131.0.0.0 Safari/537.36')
browser = webdriver.Chrome(options=options)
browser.maximize_window()

#1.기상청 홈페이지 이동
url = 'https://www.coffeebeankorea.com/store/store.asp'
browser.get(url)

#2.storePop2() 함수 실행
browser.execute_script('storePop2(31)')
time.sleep(1)

#3.스크랩핑
soup = BeautifulSoup(browser.page_source, 'html.parser')

store_name_h2 = soup.select('div.store_txt > h2')
store_name = store_name_h2[0].string

store_info = soup.select('div.store_txt > table.store_table > tbody > tr > td')
store_address = list(store_info[2])[0]

store_phone = store_info[3].string
print(f'{store_name} | {store_address} | {store_phone}')
```

---

- **[커피빈]-[매장찾기] 페이지에서 Store ID를 구한다.**

[04.크롤링] ex19.py

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

options = webdriver.ChromeOptions()
options.add_argument('headless')
options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Chrome/131.0.0.0 Safari/537.36')
browser = webdriver.Chrome(options=options)
browser.maximize_window()

#1.기상청 홈페이지 이동
url = 'https://www.coffeebeankorea.com/store/store.asp'
browser.get(url)

#2. Store ID 구하기
store = browser.find_element(By.ID, 'storeListUL')
es = store.find_elements(By.TAG_NAME, 'li')

store_id = []
for e in es:
    id = e.get_attribute('data-no')
    store_id.append(id)
print(len(store_id))
```

---

- **커피빈의 전국 매장 정보를 크롤링하여 파일에 저장한다.**

[04.크롤링] ex20.py

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

options = webdriver.ChromeOptions()
options.add_argument('headless')
options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.0.0 Safari/537.36')
browser = webdriver.Chrome(options=options)
browser.maximize_window()

#커피빈 전국 매장의 Store ID를 구하는 함수
def store_id(url):
    #1.커피빈 매장찾기 페이지로 이동
    browser.get(url)
    #2. 커피매장 정보영역 찾기
    store = browser.find_element(By.ID, 'storeListUL')
    es = store.find_elements(By.TAG_NAME, 'li')
    #3. Store ID(data-no)를 구해 Return한다.
    store_id = []
    for e in es:
        id = e.get_attribute('data-no')
        store_id.append(id)
    return store_id

#전국 Store ID를 받아 매장 정보를 구하는 함수
def store_info(store_id):
    f = open('data/커피빈.csv', "w", encoding="utf-8-sig", newline="")
    writer = csv.writer(f)
    #Store ID 수만큼 반복
    for index, id in enumerate(store_id):
        #1.커피빈 매장찾기 페이지로 이동
        browser.get(url)
        browser.execute_script(f'storePop2({id})')
        time.sleep(1)
        #2.스크랩핑
        soup = BeautifulSoup(browser.page_source, 'html.parser')
        #매장이름
        store_name_h2 = soup.select('div.store_txt > h2')
        store_name = store_name_h2[0].string
        #매장정보
        store_info = soup.select('div.store_txt > table.store_table > tbody > tr > td')
        #매장이름
        store_address = list(store_info[2])[0]
        #매장전화
        store_phone = store_info[3].string
        #매장정보 출력, 파일에 저장
        print(f'{index+1}.{store_name} | {store_address} | {store_phone}')
        data = [index+1, store_name, store_address, store_phone]
        writer.writerow(data)

if __name__ == '__main__':  #해당 파이썬 파일에서 직접 실행했을 때만 아래 코드를 실행
    url = 'https://www.coffeebeankorea.com/store/store.asp'
    store_id = store_id(url)
    store_info(store_id)
```

- **[한빛미디어]-[검색]-[검색어입력] 검색결과 출력과 이미지 다운로드**

**[04.크롤링] ex21.py**

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

options = webdriver.ChromeOptions()
options.add_argument('headless')
options.add_argument('user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.0.0 Safari/537.36')
browser = webdriver.Chrome(options=options)
browser.maximize_window()

url = 'https://www.hanbit.co.kr/'
browser.get(url)

search = browser.find_element(By.CSS_SELECTOR, '#top_search > a')
search.click()
time.sleep(2)

query = '자바'
text = browser.find_element(By.CSS_SELECTOR, '#keyword_str')
text.send_keys(query)
text.send_keys(Keys.ENTER)
time.sleep(2)

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

div = soup.find('div', attrs={'class':'ser_list_wrap'})
es = div.find_all('li', attrs={'class':'ser_bg'})

if not os.path.exists('books'): #'books' 폴더가 없으면 새로 생성
    os.mkdir('books')

for index, e in enumerate(es):
    title = e.find('strong', attrs={'class':'ser_text_title'})
    title = title.get_text()
    authors = e.find('span', attrs={'class':'ser_text_sub2'})
    authors = authors.get_text().split('/')[0].strip()
    price = e.find('span', attrs={'class':'ser_text_point'})
    image = e.find('img')['src']
    image = 'https://www.hanbit.co.kr/' + image

    #이미지를 로컬에 저장
    res_image = requests.get(image)
    res_image.raise_for_status()

    with open(f'books/image{index}.jpg', 'wb') as file:
        file.write(res_image.content)

    price = price.get_text()
    print(f'제목:{index}.{title}')
    print(f'저자:{authors}')
    print(f'가격:{price}')
    print('-' * 100)
```

## • 05. 오픈 API

웹에서는 HTTP 통신을 사용하며 요청과 응답의 구조로 서비스가 이루어진다. 사용자가 자신에게 필요한 데이터가 있는 서버(웹사이트)의 url에 접속하여 수집할 데이터에 대한 요청(request)을 보내면 서버가 그에 대한 응답(response)으로 데이터를 JSON 또는 XML 형식으로 사용자에게 보낸다. 이때 사용하는 것이 웹 API이다. 웹 API(Application Programming Interface)는 지도, 검색, 주가, 환율 등 다양한 정보가 담긴 웹사이트의 기능을 외부에서 쉽게 사용할 수 있도록 사용 절차와 규약을 정의한 것이다.

### • 네이버 검색 API

네이버 개발자 센터 : http://developers.naver.com

- 네이버 개발자 센터의 쇼핑 검색 API를 이용하여 특정단어의 검색결과를 추출한다.

[05.API] ex01.py

```
import requests
import datetime
import json
import csv

def getNaver(url, query):
    headers = {
        'X-Naver-Client-Id':'xxxxxxxxxx',
        'X-Naver-Client-Secret':'xxxxxxxxxx'
    }
    params = {
        'query':query,
        'display':100
    }
    try:
        response = requests.get(url, headers=headers, params=params)
        if response.status_code == 200: #요청에 대한 응답이 성고한 경우
            data_json = response.json() #응답(response)결과를 JSON 형식으로 파싱
            data_csv = [['상품명', '가격', '브랜드', '메이커']]  #csv파일의 title
            for item in data_json['items']: #JSON 데이터 중 'items' 배열 형식 데이터를 반복
                title = item['title']
                price = item['lprice']
                brand = item['brand']
                maker = item['maker']
                print(title, price, brand, maker)
                print('-' * 100)
                data_csv.append([title, price, brand, maker])
            return data_json['items'], data_csv #JSON, csv 형식의 데이터를 반환
    except Exception as e:
        print(e)
        print(f'[{datetime.datetime.now()}] Error for URL : {url}')
        return None

if __name__ == '__main__':  #해당 파이썬 파일에서 직접 실행했을 때만 아래 코드를 실행
    query = '노트북'
    url = 'https://openapi.naver.com/v1/search/shop.json'
    data = getNaver(url, query)

    #검색결과를 json 파일로 저장
    with open(f'data/{query}.json', 'w', encoding='utf-8') as json_file:
        json.dump(data[0], json_file, ensure_ascii=False, indent=4)

    #검색결과를 csv 파일로 저장
    with open(f'data/{query}.csv', "w", encoding="utf-8-sig", newline="") as csv_file:
        writer = csv.writer(csv_file)
        writer.writerows(data[1])
```

- **네이버 쇼핑 검색 API를 이용해 검색 후 상품 이미지를 다운로드한다.**

[05.API] ex02.py

```
import requests
import datetime
import os

def getNaver(url, query):
    headers = {
        'X-Naver-Client-Id':'xxxxxxxxxx',
        'X-Naver-Client-Secret':'xxxxxxxxxx'
    }
    params = {
        'query':query,
        'display':100
    }
    try:
        response = requests.get(url, headers=headers, params=params)
        if response.status_code == 200:
            data_json = response.json()
            images = []
            for item in data_json['items']:
                image = item['image']
                print(image)
                images.append(image)
            return images
    except Exception as e:
        print(e)
        print(f'[{datetime.datetime.now()}] Error for URL : {url}')
        return None

if __name__ == '__main__':  #해당 파이썬 파일에서 직접 실행했을 때만 아래 코드를 실행
    query = '노트북'
    url = 'https://openapi.naver.com/v1/search/shop.json'
    images = getNaver(url, query)
    try:
        #'shop'폴더가 없으면 새로 생성
        if not os.path.exists('shop'):
            os.mkdir('shop')
        #이미지 URL을 이용하여 이미지 다운로드
        for index, url in enumerate(images):
            res_image = requests.get(url)
            res_image.raise_for_status()
            #이미지를 로컬에 저장
            with open(f'shop/image{index}.jpg', 'wb') as file:
                file.write(res_image.content)
    except Exception as e:
        print('error', e)
```

출력 결과

```
https://shopping-phinf.pstatic.net/main_8752866/87528666743.7.jpg
https://shopping-phinf.pstatic.net/main_8795824/87958243440.2.jpg
https://shopping-phinf.pstatic.net/main_5264831/52648316006.20250124141424.jpg
https://shopping-phinf.pstatic.net/main_5264816/52648169020.20250124133117.jpg
https://shopping-phinf.pstatic.net/main_4088231/40882313631.20230707160605.jpg
https://shopping-phinf.pstatic.net/main_3974611/39746112618.20230502165309.jpg
https://shopping-phinf.pstatic.net/main_4619294/46192941618.20240305111031.jpg
https://shopping-phinf.pstatic.net/main_5242457/52424575618.20250110164726.jpg
https://shopping-phinf.pstatic.net/main_4663306/46633068618.20240325185204.jpg
```

- **카카오 검색 API**

**카카오 앱 개발 플랫폼 서비스 : http://developers.kakao.com**

- **카카오 개발자 센터의 도서 검색 API를 이용하여 특정 단어의 검색결과를 추출한다.**

**[05.API] ex03.py**

```
import requests
import json
import datetime

def getKakao(url, query, size):
    headers = { 'authorization':'KakaoAK xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx' }
    try:
        page = 1
        json_list = []
        while True:
            query_url = f'{url}?query={query}&size={size}&page={page}'
            response = requests.request(method='get', url=query_url, headers=headers)
            #응답(response)결과를 JSON 형식으로 파싱한다.
            json_data = response.json()
            #is_end가 True이면 마지막 페이지지
            is_end = json_data['meta']['is_end']
            #is_end가 True이면 반복문을 빠져나간다.
            if bool(is_end):
                break
            else:
                documents = json_data['documents']
                for index, item in enumerate(documents):
                    title = item['title']
                    authors = item['authors']
                    price = item['price']
                    seq = (page-1) * size + index + 1
                    print(seq, title, authors, price)
                    json_list.append(item)
            page += 1
        return json_list
    except Exception as e:
        print(e)
        print(f'[{datetime.datetime.now()}] Error for URL : {url}')
        return None

if __name__ == '__main__':  #해당 파이썬 파일에서 직접 실행했을 때만 아래 코드를 실행
    query = '홍길동'
    size = 10
    url = 'https://dapi.kakao.com/v3/search/book'

    json_data = getKakao(url, query, size) #카카오 검색 함수를 호출한다.
    with open(f'data/도서.json', 'w', encoding='utf-8') as json_file: #검색결과를 파일에 저장
        json.dump(json_data, json_file, ensure_ascii=False, indent=4)
```

**출력 결과**

```
1 Do it! 점프 투 파이썬 ['박응용'] 22000
2 코딩 자율학습 나도코딩의 파이썬 입문 ['나도코딩'] 24000
3 혼자 공부하는 파이썬 ['윤인성'] 22000
4 챗GPT로 만드는 주식 & 암호화폐 자동매매 시스템 ['설근민'] 20000
5 Do it! 점프 투 파이썬 ['박응용'] 18800
6 파이썬 데이터 분석가 되기 + 챗GPT ['셀레나'] 26000
...
```

- **카카오 지도 API**

**구글에서 식신로드, 역대 서울지역 '만점 식당' 20선 검색 후 주소 수집 및 지도 표시**

- **주소, 업체명, 전화번호 수집**

**[05.API] ex04.py**

```
import requests
from bs4 import BeautifulSoup
import re
import pprint #dictionary 형을 보기 좋게 출력한다.

#주소, 업체명, 전화번호 수집
def getInfo():
    url = 'https://www.wikitree.co.kr/articles/217101'
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'lxml')
    #업체명을 저장할 리스트
    name_list = []
    for e in soup.select('strong'):
        if '회' in e.text: #e요소에 '회'가 있으면
            name = e.text.split('회 ')[-1] #'회 '를 기준으로 나누어 배열에 저장 후 마지막 요소를 수집한다.
            name_list.append(name)

    #모든 p 태그들을 수집한다.
    p =soup.select('p')
    #전화번호를 저장할 list
    tel_list = []
   #주소를 저장할 list
    add_list = []

    for i in range(len(p)):
        #정규식 {2,} 숫자 2자리이상, {3,} 숫자 3자리이상, {4} 숫자 4자리
        if re.match('[0-9]{2,}-[0-9]{3,}-[0-9]{4}', p[i].text):
            tel = p[i].text
            tel_list.append(tel)
            #주소를 수집하여 공백을 제거한다.
            add = p[i+1].text.strip()
            add_list.append(add)

    #zip 두개의 list를 묶어준다.
    add_tel = list(zip(add_list, tel_list))
    name_add_tel = list(zip(name_list, add_tel))
    #dictionary로 변경한다.
    info = dict(name_add_tel)
    return info, name_list, tel_list, add_list

if __name__ == '__main__':
    info = getInfo()
    pprint.pprint(info)
```

**출력 결과**

```
{'까사디노아': ('서울 마포구 연남동 257-8', '02-3142-1108'),
 '다성일식': ('서울시 서대문구 창천동 72-36', '02-338-8951'),
 '대보명가': ('서울시 강북구 수유동 563-14', '02-907-6998'),
 '돕감자탕': ('서울시 광진구 화양동 9-22', '02-499-2838'),
...
 '충주집': ('서울시 동대문구 제기동 138-2', '02-923-1068'),
 '해뜨는집': ('서울시 성북구 동소문동 1가 62', '02-764-6354')}
```

- **주소와 Kakao API를 활용 위, 경도 수집 [카카오개발자센터]-[문서]-[로컬]-[REST API]-[주소를 좌표로 변환하기]**

**[05.API] ex05.py**

```
from ex04 import getInfo
import requests
import time
import pprint

file = open('data/data_kakao_key.txt')
key = file.readlines()
apiKey = key[0].strip()

def getLatLng():
    info, name_list, tel_list, add_list = getInfo()

    lat_list = [] #위도 리스트
    lng_list = [] #경도도 리스트
    for add in add_list:
        url = 'https://dapi.kakao.com/v2/local/search/address.json'
        params = {
            'query':add
        }
        headers = {
            'Authorization': f'KakaoAK {apiKey}'
        }

        response = requests.get(url, params=params, headers=headers)
        #응답결과를 json 타입으로 변경한다.
        json_data = response.json()

        #documents에 첫 번째 아이템
        address = json_data['documents'][0]['address']
        lng_list.append(address['x']) #경도(longitude)
        lat_list.append(address['y']) #위도(latitude)
        time.sleep(0.5) #0.5초 기다린다.

    lat_lng = list(zip(lat_list, lng_list))
    name_lat_lng = list(zip(name_list, lat_lng))
    info = dict(name_lat_lng)
    return name_list, tel_list, add_list, lat_list, lng_list

if __name__=='__main__':
    info = getLatLng()
    pprint.pprint(info)
```

**실행 결과**

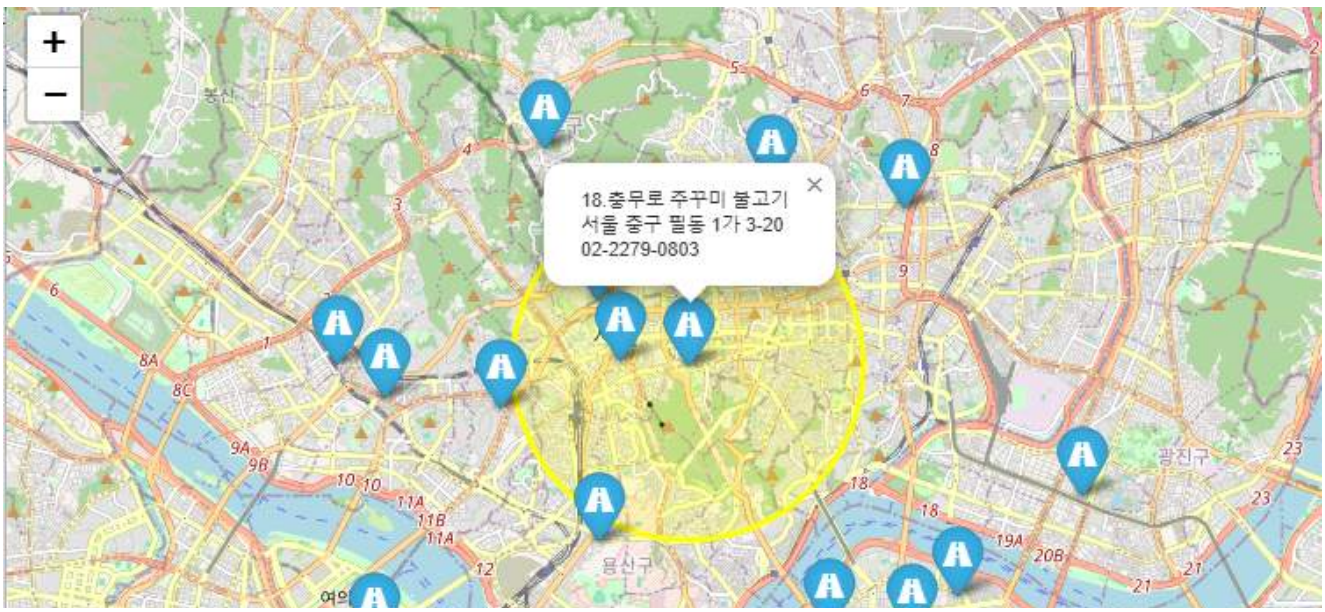
```
{'까사디노아': ('126.923556958229', '37.5620838593197'),
 '다성일식': ('126.932965561024', '37.5568947886339'),
 '대보명가': ('127.007127542151', '37.6456327086469'),
 '돕감자탕': ('127.069272933219', '37.5414828193382'),
 '라틀리에 모니크': ('127.044857053316', '37.5261747199815'),
 '립스테이크': ('126.964055305288', '37.5955770447834'),
...
 '일품헌': ('127.039180053795', '37.4844942544155'),
 '줄리에뜨': ('127.00090838481', '37.4947067319368'),
 '창고43': ('126.930731521429', '37.5189272446284'),
 '충무로 주꾸미 불고기': ('126.992201358274', '37.5617589423154'),
 '충주집': ('127.034542600462', '37.5862034507119'),
 '해뜨는집': ('127.00839172712', '37.5900669779109')}
```

- **식신로드, 역대 서울지역 '만점 식당' 20선 지도 출력**

**[05.API] ex06.py**

```
from ex05 import getLatLng
import folium #folium은 python에서 사용할 수 있는 지도 시각화 라이브러리

#지도출력
def getMap():
    name_list, tel_list, add_list,  lat_list, lng_list = getLatLng()

    #남산타워 기준
    lat = 37.5511225714939
    lng = 126.987867837993
    map = folium.Map((lat, lng), zoom_start=12, width=750, height=500)
    for i in range(len(name_list)):
        location = (lat_list[i], lng_list[i])
        text = f'{i}.{name_list[i]}<br>{add_list[i]}<br>{tel_list[i]}<br>'
        popup = folium.Popup(text, max_width=200)
        folium.Marker(
            location,
            popup,
            icon=folium.Icon(color='blue', icon='glyphicon-road') #https://getbootstrap.com/docs/3.3/components/
        ).add_to(map)

    #지정한 위경도 위치에 원을 지도 해상도에 맞춰서 그려줌(원 그기가 변경됨됨)
    folium.CircleMarker(
        [lat_list[18], lng_list[18]],
        radius=100, #반경 100미터
        color='yellow',
        fill_color='yellow',
        popup='Circle',
        tooltip='tooltip'
    ).add_to(map)
    #지도를 HTML로 저장
    map.save('data/식신로드.html')

if __name__ == '__main__':
    getMap()
```

**실행 결과**



- **인천 지역 노랑통닭 가맹점 데이터 수집**

[05.API] ex07.py ①

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import Select
import time
import csv

url='https://www.norangtongdak.co.kr/store/store.html'
browser = webdriver.Chrome()
browser.get(url)
browser.implicitly_wait(3) #3초 안에 웹페이지를 load하면 바로 넘어가거나 10초를 기다림
#지역 선택
sido = browser.find_element(By.ID, 'sido')
sido = Select(sido)
#구군 선택
sigugun = browser.find_element(By.ID, 'sigugun')
sigugun = Select(sigugun)
btn = browser.find_element(By.XPATH, '//*[@id="wrapper"]/div[5]/div[1]/div/div/form/div[2]/button')

seq = 1
name_list = []
add_list = []
tel_list = []
info_list = []
city = '인천'

for i in range(len(sido.options)):
    #print(sido.options[i].text)
    if sido.options[i].text == city:
        #시도 선택박스에 위치한 순서 값으로 선택한다.
        sido.select_by_index(i)
        for j in range(len(sigugun.options)):
            #print(sigugun.options[j].text)
            if j != 0:
                #구군 선택박스에 위치한 순서 값으로 선택한다.
                sigugun.select_by_index(j)
                btn.click()
                time.sleep(1)
                info = browser.find_element(By.XPATH, '//*[@id="wrapper"]/div[5]/div[1]/div/div/div')
                names = info.find_elements(By.CLASS_NAME, 'st2')
                tels = info.find_elements(By.CLASS_NAME, 'st3')
                adds = info.find_elements(By.CLASS_NAME, 'st5')
                for i in range(len(names)):
                    name= names[i].text
                    tel = tels[i].text
                    add = adds[i].text
                    name_list.append(name)
                    tel_list.append(tel)
                    add_list.append(add)
                    #print(seq, name, tel, add)
                    info_list.append([seq, name, tel, add])
                    seq += 1
        break

#검색결과를 파일에 저장
with open(f'data/노랑통닭{city}.csv', "w", encoding="utf-8-sig", newline="") as csv_file:
    writer = csv.writer(csv_file)
    writer.writerows(info_list)
```

**실행 결과**

```
1 영종운남점 032-751-0905 인천 중구 운남동로3번길 21 (운남동)
2 동인천점 032-773-0333 인천 중구 우현로67번길 32 (인현동) 1층
3 영종하늘도시점 032-747-1767 인천 중구 중산동 1885-8 썬앤문타워 1층 101호
4 인천 운서역점 032-752-2468 인천 중구 신도시남로142번길 6 (운서동, 메가스타 영종)
Created TensorFlow Lite XNNPACK delegate for CPU.
5 학익점 032-710-2920 인천 미추홀구 학익동 231-19 이당빌딩 101호
6 학익점 032-710-2920 인천 미추홀구 매소홀로 409 (학익동, 이당빌딩) 101호
7 SSG랜더스필드점  인천 미추홀구 매소홀로 618 (문학동, 문학경기장)
8 도화점 032-872-7373 인천 미추홀구 숙골로95번길 19 (도화동)
9 인하대점 032-875-8253 인천 미추홀구 경인남길30번길 65 (용현동, 대동빌딩)
10 송도6.8공구점 032-833-7572 인천 연수구 센트럴로 415 (송도동, 힐스테이트 송도 더테라스) 1층 117호
11 인천 옥련점 032-834-9283 인천 연수구 청량로 181 (옥련동, 청원타운) 청원타운 103호
12 송도2호점 032-811-2292 인천 연수구 송도과학로27번길 55 (송도동, 롯데캐슬 캠퍼스타운)
13 연수점 032-821-7788 인천 연수구 샘말로43번길 3 (연수동)
14 송도점 032-832-2292 인천 연수구 송도동 3-6 송도아크리아1 118호
15 간석점 032-466-0703 인천 남동구 석산로 166 (간석동, 간석래미안자이아파트 상가2동)
16 구월점 032-434-9282 인천 남동구 인하로507번길 22 (구월동)
...
```

- **주소와 Kakao API를 활용 위, 경도 수집**

**[05.API] ex07.py ②**

```
...
#주소와 Kakao API를 활용 위, 경도 수집
import requests
import pprint

file = open('data/data_kakao_key.txt')
key = file.readlines()
apiKey = key[0].strip()

lat_list = []
lng_list = []

for add in add_list:
    url = 'https://dapi.kakao.com/v2/local/search/address.json'
    params = {
        'query':add
    }
    headers = {
        'Authorization':f'KakaoAK {apiKey}'
    }
    response = requests.get(url, params=params, headers=headers)
    json_data = response.json()

    address = json_data['documents'][0]['address']
    #경도(longitude)
    lng_list.append(address['x'])
    #위도(latitude)
    lat_list.append(address['y'])

    time.sleep(0.5)

lat_lng = list(zip(lat_list, lng_list))
name_lat_lng = list(zip(name_list, lat_lng))
info = dict(name_lat_lng)
pprint.pprint(info)
```

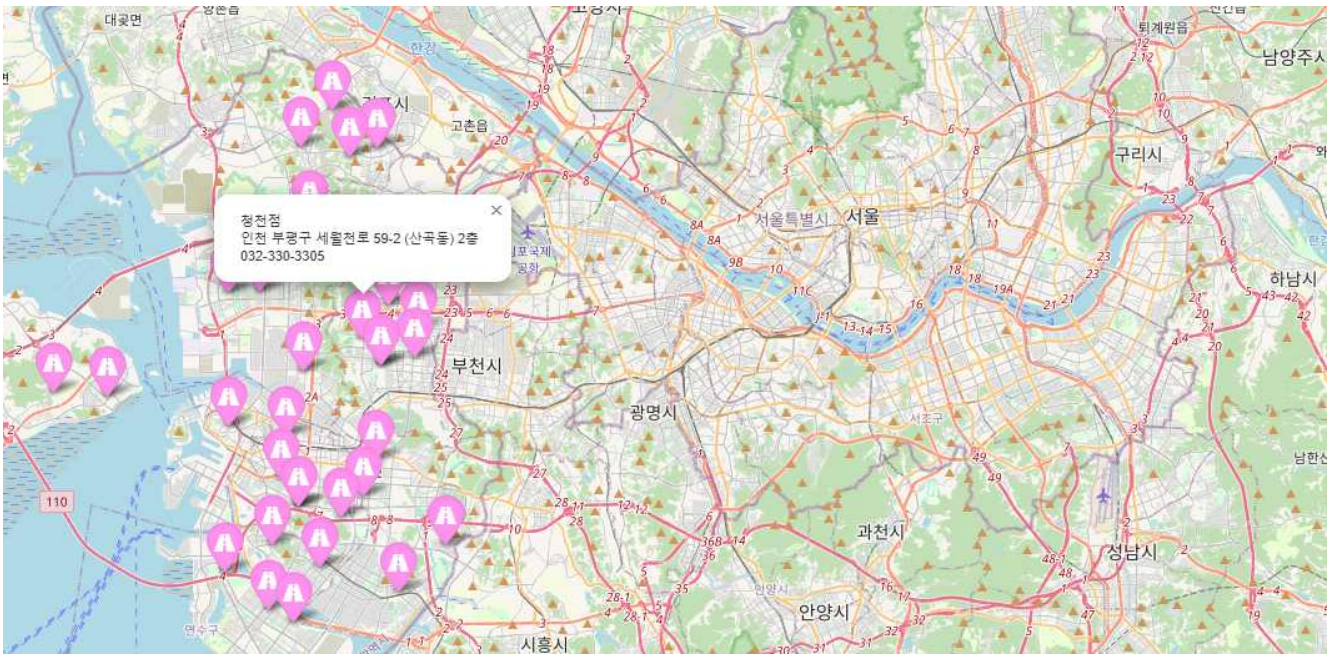
**실행 결과**

```
{'SSG랜더스필드점': ('126.690759830613', '37.4350819826381'),
 '간석점': ('126.710338072269', '37.4603812456991'),
 '강화점': ('126.487094761269', '37.7430391038359'),
 '경인교대점': ('126.724030001719', '37.5376391167284'),
 '구월점': ('126.703211011072', '37.4442444883352'),
 '도화점': ('126.660751264639', '37.47025894914'),
 '동인천점': ('126.629279451782', '37.4746225770893'),
 '마전점': ('126.668756734965', '37.5967793086726'),
 '백령점': ('124.717227333045', '37.9713860399441'),
...
 '청천점': ('126.702679287312', '37.5130236235186'),
 '학익점': ('126.667531686199', '37.4402016404617')}
```

**[05.API] ex07.py ③**

```
...
#지도 출력
import folium
map = folium.Map(('37.5511225714939', '126.987867837993'), zoom_start=11) #남산타워 기준

for i in range(len(name_list)):
    location = (lat_list[i], lng_list[i])
    text = f'{name_list[i]}<br>{add_list[i]}<br>{tel_list[i]}<br>'
    popup = folium.Popup(text, max_width=200)

    #마커 생성
    folium.Marker(
        location,
        popup,
        icon=folium.Icon(color='pink', icon='glyphicon-road')
    ).add_to(map)

#지도를 HTML로 저장
map.save('data/map_노란통닭(인천).html')
```

**실행 결과**



- **공공데이터 API**

**공공데이터란 공공기관에서 업무를 수행하여 만들었거나 관리하고 있는 다양한 형태의 데이터를 일컫는다. 우리나라에서는 정부3.0의 핵심 정책 중 하나로 공공데이터 정책을 추진하였다. 이는 보유한 공공데이터를 적극 개방하여 국민과 공유하며 소통과 협력을 확대하기 위함이다.**

- **공공데이터 포털의 API를 이용해 출입국관광통계서비스 데이터를 크롤링해 보자. [개별계정 상세보기]-[기본정보]-[상세설명]**

**[data] data_go_key.txt**

```
인증키 Encoding xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
인증키 Decoding xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

**[05.API] ex08.py**

```
import requests
import json

def getKey():
    file = open('data/data_go_key.txt')
    key = file.readlines()
    encKey = key[0].strip()
    decKey = key[1].strip()
    serviceKey = encKey
    return serviceKey

def getRequests(ym, natCd, edCd):
    try:
        serviceKey = getKey()
        service_url = 'http://openapi.tour.go.kr/openapi/service/EdrcntTourismStatsService/getEdrcntTourismStatsList'
        params = f'?serviceKey={serviceKey}&YM={ym}&NAT_CD={natCd}&ED_CE={edCd}&_type=json'
        url = service_url + params
        res = requests.get(url)
        if res.status_code == 200:
            jsonResult = json.loads(res.text)
            return jsonResult
    except Exception as err:
        print('에러:' + err)
        return None

if __name__ == '__main__':
    #년:yyyy 월:mm
    ym = 202401

    #중국:112 / 일본:130 / 미국:275
    natCd = 112

    #E:방한 관광객 D:해외 출국
    edCd = 'E'

    jsonResult = getRequests(ym, natCd, edCd)
    print(jsonResult)
```

**2024년 01월 출력 결과**

```
{'response': {'header': {'resultCode': '0000', 'resultMsg': 'OK'}, 'body': {'items': {'item': {'ed': '방한외래관광객', 'edCd': 'E', 'natCd': 112, 'natKorNm': '중국', 'num': 280035, 'rnum': 1, 'ym': 202401}}, 'numOfRows': 10, 'pageNo': 1, 'totalCount': 1}}}
```

**2025년 01월 출력 결과**

```
{'response': {'header': {'resultCode': '0000', 'resultMsg': 'OK'}, 'body': {'items': '', 'numOfRows': 10, 'pageNo': 1, 'totalCount': 0}}}
```

- **2024년 01월 중국의 출입국관광통계서비스 데이터를 크롤링해 보자.**

**[05.API] ex09.py**

```
import requests
import json

def getKey():
    file = open('data/data_go_key.txt')
    key = file.readlines()
    encKey = key[0].strip()
    decKey = key[1].strip()
    serviceKey = encKey
    return serviceKey

def getRequests(ym, natCd, edCd):
    try:
        serviceKey = getKey()
        service_url = 'http://openapi.tour.go.kr/openapi/service/EdrcntTourismStatsService/getEdrcntTourismStatsList'
        params = f'?serviceKey={serviceKey}&YM={ym}&NAT_CD={natCd}&ED_CE={edCd}&_type=json'
        url = service_url + params
        res = requests.get(url)
        if res.status_code == 200:
            jsonResult = json.loads(res.text)
            return jsonResult
    except Exception as err:
        print('에러:' + err)
        return None

def getTourism(ym, natCd, edCd):
    jsonResult = getRequests(ym, natCd, edCd)
    if jsonResult:
        items = jsonResult['response']['body']['items']
        if items == '':
            return None
        else:
            item = items['item']
            natName = item['natKorNm'].replace(' ','')
            num = item['num']
            ym = item['ym']
            natCd = item['natCd']
            result = [ym, natCd, natName, num]
            return result

if __name__ == '__main__':
    #년:yyyy 월:mm
    ym = 202401

    #중국:112 / 일본:130 / 미국:275
    natCd = 112

    #E:방한 관광객 D:해외 출국
    edCd = 'E'

    jsonResult = getTourism(ym, natCd, edCd)
    print(jsonResult)
```

**실행 결과**

```
[202401, 112, '중국', 280035]
```

- **2023년 1월~12월 중국의 출입국관광통계서비스 데이터를 크롤링 후 CSV 파일에 저장한다.**

**[05.API] ex10.py**

```
import requests
import json
import csv

def getKey():
    file = open('data/data_go_key.txt')
    key = file.readlines()
    encKey = key[0].strip()
    serviceKey = encKey
    return serviceKey

def getRequests(ym, natCd, edCd):
    try:
        serviceKey = getKey()
        service_url = 'http://openapi.tour.go.kr/openapi/service/EdrcntTourismStatsService/getEdrcntTourismStatsList'
        params = f'?serviceKey={serviceKey}&YM={ym}&NAT_CD={natCd}&ED_CE={edCd}&_type=json'
        url = service_url + params
        res = requests.get(url)
        if res.status_code == 200:
            jsonResult = json.loads(res.text)
            return jsonResult
    except Exception as err:
        print('에러:' + err)
        return None

def getTourism(ym, natCd, edCd):
    jsonResult = getRequests(ym, natCd, edCd)
    if jsonResult:
        items = jsonResult['response']['body']['items']
        if items == '':
            return None
        else:
            item = items['item']
            natName = item['natKorNm'].replace(' ','')
            num = item['num']
            ym = item['ym']
            natCd = item['natCd']
            result = [ym, natCd, natName, num]
            return result

if __name__ == '__main__':
    year = 2023
    natCd = 112
    edCd = 'E'

    #특정년도 1월~12월 데이터를 배열에 저장
    results = []
    results.append(['입국년월', '국가코드', '입국자국가', '입국자수'])
    for month in range(1, 13):
        ym = f'{year}{month:02}'
        result = getTourism(ym, natCd, edCd)
        results.append(result)

    #검색결과를 csv 파일로 저장
    with open(f'data/{year}년.csv', "w", encoding="utf-8-sig", newline="") as csv_file:
        writer = csv.writer(csv_file)
        writer.writerows(results)
```

- **세종특별자치시_세종특별자치시 과속방지턱 조회 서비스 [개별계정 상세보기]-[기본정보]-[상세설명]**

[05.API] ex11.py

```
import requests
import pprint
import json

file = open('data/data_go_key.txt')
key = file.readlines()
enKey = key[0].strip()
deKey = key[1].strip()

url = 'http://apis.data.go.kr/5690000/sjSpeedBump/sj_00000170'
params ={'serviceKey' : deKey, 'pageIndex' : '1', 'pageUnit' : '20', 'dataTy' : 'json', 'searchCondition' : 'ovrspd_Prvn_Stlese',
'searchKeyword' : '1' }

response = requests.get(url, params=params)
json_data = json.loads(response.content)
items = json_data['body']['items']
#pprint.pprint(items)

import folium

map = folium.Map(('36.5007', '127.2579'), zoom_start=15) #세종 어진동 기준
for item in items:
    add = item['addr']
    tel = item['telno']
    lat = item['la']
    lng = item['lo']
    print(add, lat, lng)
    location = (lat, lng)
    text = f'{add}'
    popup = folium.Popup(text, max_width=200)

    #마커 생성
    folium.Marker(
        location,
        popup,
        icon=folium.Icon(color='red', icon='glyphicon-road')
    ).add_to(map)

#지도를 HTML로 저장
map.save('data/세종시과속방지턱.html')
```

실행 결과

- **한국환경공단_에어코리아_미세먼지 경보 발령 현황**

[05.API] ex12.py

```
import requests
import json
import math
import csv

file = open('data/data_go_key.txt')
key = file.readlines()
enKey = key[0].strip()
deKey = key[1].strip()

#API 접속후 응답데이터 불러오기
def getRequests(year, page):
    url = 'http://apis.data.go.kr/B552584/UlfptcaAlarmInqireSvc/getUlfptcaAlarmInfo'
    params ={'serviceKey':deKey, 'returnType':'json', 'numOfRows':'100', 'pageNo':page, 'year':year, 'itemCode':'PM10'}

    response = requests.get(url, params=params)
    json_data = json.loads(response.content)
    return json_data

#json 데이터 중 items 데이터 구하기
def getItems(json_data):
    items = json_data['response']['body']['items']
    for index, item in enumerate(items):
        issueDate = item['issueDate'] #발령일
        issueTime = item['issueTime'] #발령시간
        districtName = item['districtName'] #발령지역
        moveName = item['moveName'] #발령권역
        issueVal = item['issueVal'] #발령시 미세먼지 농도(단위: ug/m3)
        clearDate = item['clearDate'] #해제 날짜
        clearTime = item['clearTime'] #해제 시간

        seq = (page-1) * 100 + index + 1
        data = [seq, issueDate, issueTime, districtName, moveName, issueVal, clearDate, clearTime]
        item_list.append(data)
        print(data)

if __name__=='__main__':
    year = input('미세먼지 경보 발령 현황:')
    page=1
    item_list = [['일련번호','발령일','발령시간','발령지역','발령권역','농도','해제날짜','해제시간']]

    json_data = getRequests(year, page)
    totalCount = json_data['response']['body']['totalCount']
    if totalCount > 0:
        getItems(json_data)

    #마지막 페이지구하기기
    lastPage = math.ceil(totalCount/100)

    for page in range(2, lastPage+1):
        json_data = getRequests(year, page)
        getItems(json_data)

    #검색결과를 csv 파일로 저장
    with open(f'data/{year}년 미세먼지 경보 발령 현황.csv', "w", encoding="utf-8-sig", newline="") as csv_file:
        writer = csv.writer(csv_file)
        writer.writerows(item_list)
```

- **국토교통부_(TAGO)_고속버스정보**

[05.API] ex13.py

```
import requests
import json
import pprint
from datetime import datetime
import math

file = open('data/data_go_key.txt')
key = file.readlines()
enKey = key[0].strip()
deKey = key[1].strip()

def getReauest(page, depId, arrId, year_month):
    url = 'http://apis.data.go.kr/1613000/ExpBusInfoService/getStrtpntAlocFndExpbusInfo'
    params ={'serviceKey' : deKey, 'pageNo' : page, 'numOfRows' : '10', '_type' : 'json',
             'depTerminalId' : depId, 'arrTerminalId' : arrId, 'depPlandTime' : year_month, 'busGradeId' : '1' }

    response = requests.get(url, params=params)
    json_data = json.loads(response.content)
    #pprint.pprint(json_data)
    return json_data

def getItems(json_data):
    items = json_data['response']['body']['items']['item']
    for index, item in enumerate(items):
        routeId = item['routeId'] #노선ID
        gradeNm = item['gradeNm'] #버스등급
        datetime_format ='%Y%m%d%H%M'
        depPlandTime = datetime.strptime(str(item['depPlandTime']), datetime_format) #출발시간
        depfmt = depPlandTime.strftime('%Y-%m-%d %H:%M')
        arrPlandTime = datetime.strptime(str(item['arrPlandTime']), datetime_format) #도착시간
        arrfmt = arrPlandTime.strftime('%Y-%m-%d %H:%M')
        depPlaceNm = item['depPlaceNm'] #출발지
        arrPlaceNm = item['arrPlaceNm'] #도착지
        charge = item['charge'] #운임

        seq = (page-1) * 10 + index + 1
        data = [seq, routeId, gradeNm, depfmt, arrfmt, depPlaceNm, arrPlaceNm, charge]
        item_list.append(data)
        print(data)

if __name__=='__main__':
    page=1
    depId = 'NAEK010' #출발터미널 ID
    arrId = 'NAEK300' #도착터미널 ID
    year_month='20250130' #출발년월일

    item_list = []
    json_data = getReauest(page, depId, arrId, year_month)
    totalCount = json_data['response']['body']['totalCount']
    if totalCount > 0:
        getItems(json_data)

    lastPage = math.ceil(totalCount/10)
    for page in range(2, lastPage+1):
        json_data = getReauest(page, depId, arrId, year_month)
        getItems(json_data)
```

- **국토교통부_(TAGO)_고속버스정보 터미널 검색: [개발계정 상세보기]-[고속버스터미널 목록 조회]-[확인]**

**[05.API] ex14.py**

```
import requests
import json

file = open('data/data_go_key.txt')
key = file.readlines()
enKey = key[0].strip()
deKey = key[1].strip()

def getReauest(searchName):
    #오류코드가 443인 경우 https에서 s를 빼고 http로 호출
    url = 'http://apis.data.go.kr/1613000/ExpBusInfoService/getExpBusTrminlList'
    params ={'serviceKey' : deKey, 'pageNo' : 1, 'numOfRows' : '10', '_type' : 'json', 'terminalNm':searchName}

    response = requests.get(url, params=params)
    json_data = json.loads(response.content)
    #print(json_data)
    return json_data

if __name__=='__main__':
    #검색명이 'Q'일 때까지 반복한다.
    while True:
        searchName = input('검색할 터미널명 (종료:q):')
        if searchName == 'Q' or searchName == 'q':
            print('검색을 종료합니다.')
            break

        json_data = getReauest(searchName)

        totalCount = json_data['response']['body']['totalCount']
        if totalCount == 0: #검색 결과가 없는 경우
            print('터미널이 존재하지 않습니다.')
        elif totalCount == 1: #검색수가 1인 경우
            item = json_data['response']['body']['items']['item']
            id = item['terminalId']
            name = item['terminalNm']
            print(id, name)
        else: #검색수가 여러 개인 경우
            items = json_data['response']['body']['items']['item']
            for item in items:
                id = item['terminalId']
                name = item['terminalNm']
                print(id, name)
```

**실행 결과**

```
검색할 터미널명 (종료:q):부천
NAEK101 부천
검색할 터미널명 (종료:q):서울
NAEK010 서울경부
NAEK020 센트럴시티(서울)
NAEK021 센트럴시티(서울)
NAEK030 동서울(030)
NAEK032 동서울
검색할 터미널명 (종료:q):청라
터미널이 존재하지 않습니다.
검색할 터미널명 (종료:q):q
검색을 종료합니다
```

- **국토교통부_아파트 전월세 실거래가 자료**

[05.API] ex15.py

```
import requests
from bs4 import BeautifulSoup
import math
import csv

file = open('data/data_go_key.txt')
key = file.readlines()
enKey = key[0].strip()
deKey = key[1].strip()

#특정 버스가 정차하는 정류장 목록을 만드는 함수
def getRequests(gu_code, year_month, page):
    url = f'http://apis.data.go.kr/1613000/RTMSDataSvcAptRent/getRTMSDataSvcAptRent'
    params = {'serviceKey':deKey, 'LAWD_CD':gu_code, 'DEAL_YMD':{year_month},'pageNo':page, 'numOfRows':10}

    response = requests.get(url, params=params)

    soup = BeautifulSoup(response.text, 'lxml-xml')
    #print(soup.prettify())
    return soup

def getItems(soup):
    items = soup.find_all('item')
    for index, item in enumerate(items):
        name = item.find('aptNm').text #아파트명
        floor = item.find('floor').text #층
        excluUseAr = item.find('excluUseAr').text #전용면적
        umdNm = item.find('umdNm').text #법정동
        jibun = item.find('jibun').text #지번
        deposit = item.find('deposit').text #보증금액(만원)
        monthlyRent = item.find('monthlyRent').text #월세금액(만원)
        seq = (page-1) * 10 + index + 1
        data = [seq, name, floor, excluUseAr, umdNm, jibun, deposit, monthlyRent]
        item_list.append(data)
        print(data)

if __name__ == '__main__':
    #https://www.code.go.kr/stdcode/regCodeL.do / gu_code = '11545' 구단위 입력 ex)11545:서울 금천구
    gu_code = '28260' #인천 서구
    year_month = '202501'
    page = 1

    soup = getRequests(gu_code, year_month, page)
    totalCount = int(soup.find('totalCount').text)
    lastPage = math.ceil(totalCount/10)
    item_list = []
    if totalCount > 0:
        getItems(soup)

    for page in range(2, lastPage+1):
        soup = getRequests(gu_code, year_month, page)
        getItems(soup)

    #검색결과를 csv 파일로 저장
    with open(f'data/{year_month}_{gu_code}아파트 전월세.csv', "w", encoding="utf-8-sig", newline="") as csv_file:
        writer = csv.writer(csv_file)
        writer.writerows(item_list)
```

- **국토교통부_아파트 전월세 실거래가 자료 (여러 개월을 10개씩 출력)**

**[05.API] ex16.py**

```
import requests
from bs4 import BeautifulSoup
import math
import csv

file = open('data/data_go_key.txt')
key = file.readlines()
enKey = key[0].strip()
deKey = key[1].strip()

def getRequests(gu_code, year_month, page):
    url = f'http://apis.data.go.kr/1613000/RTMSDataSvcAptRent/getRTMSDataSvcAptRent'
    params = {'serviceKey':deKey, 'LAWD_CD':gu_code, 'DEAL_YMD':{year_month},'pageNo':page,'numOfRows':10}
    response = requests.get(url, params=params)

    soup = BeautifulSoup(response.text, 'lxml-xml')
    #print(soup.prettify())
    return soup

if __name__=='__main__':
    gu_code = '28260'
    year_month = ['202410', '202411', '202412', '202501']

    row_list = []
    is_head = False
    for i in range(len(year_month)):
        page = 1
        soup = getRequests(gu_code, year_month[i], page)
        rows = soup.find_all('item')

        if is_head == False:
            columns = rows[0].find_all()
            name_list = [columns[k].name for k in range(len(columns))]
            row_list.append(name_list)
            is_head= True

        for j in range(len(rows)):
            columns = rows[j].find_all()
            col_list =[columns[k].text for k in range(len(columns))]
            row_list.append(col_list)

    #검색결과를 csv 파일로 저장
    with open(f'data/{gu_code}구 아파트 전월세.csv', "w", encoding="utf-8-sig", newline="") as csv_file:
        writer = csv.writer(csv_file)
        writer.writerows(row_list)
    print(row_list)
```

**실행 결과**

| A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|
| aptNm | buildYear | contractTerm | contractType | dealDay | dealMonth | dealYear | deposit | excluUseAr | floor | jibun | monthlyRent |
| 검단한신더휴어반파크 | 2023 | | | 18 | 10 | 2024 | 18,400 | 84.9744 | 15 | 1274 | 33 |
| 검단신도시모아미래도엘리트파크 | 2022 | | | 31 | 10 | 2024 | 36,000 | 84.6561 | 11 | 1318-8 | 0 |
| 청라호수공원한신더휴 | 2020 | | | 22 | 10 | 2024 | 30,000 | 84.9613 | 12 | 103-15 | 0 |
| 검단한신더휴어반파크 | 2023 | | | 4 | 10 | 2024 | 4,600 | 74.7155 | 26 | 1274 | 51 |
| 루원제일풍경채 | 2018 | | | 19 | 10 | 2024 | 10,000 | 74.7683 | 15 | 603-1 | 100 |
| 유호D단지(D1,D2동) | 1992 | | | 22 | 10 | 2024 | 1,000 | 59.64 | 4 | 995-3 | 45 |
| 청라29블럭호반베르디움 | 2012 | | | 17 | 10 | 2024 | 27,300 | 84.9382 | 3 | 173-1 | 0 |

- **국토교통부_아파트 전월세 실거래가 자료 (여러 개월의 마지막 페이지까지 반복 출력)**

[05.API] ex17.py

```
import requests
from bs4 import BeautifulSoup
import math
import csv

file = open('data/data_go_key.txt')
key = file.readlines()
enKey = key[0].strip()
deKey = key[1].strip()

def getRequests(gu_code, year_month, page):
    url = f'http://apis.data.go.kr/1613000/RTMSDataSvcAptRent/getRTMSDataSvcAptRent'
    params = {'serviceKey':deKey, 'LAWD_CD':gu_code, 'DEAL_YMD':{year_month},'pageNo':page,'numOfRows':10}
    response = requests.get(url, params=params)
    soup = BeautifulSoup(response.text, 'lxml-xml')
    #print(soup.prettify())
    return soup

if __name__=='__main__':
    gu_code = '28260'
    year_month = ['202410', '202411', '202412', '202501']
    row_list = []
    is_head = False
    for i in range(len(year_month)):
        page = 1
        print(page, '-' * 100)
        soup = getRequests(gu_code, year_month[i], page)
        rows = soup.find_all('item')

        if is_head == False:
            columns = rows[0].find_all()
            name_list = [columns[k].name for k in range(len(columns))]
            row_list.append(name_list)
            is_head= True

        for j in range(len(rows)):
            columns = rows[j].find_all()
            col_list =[columns[k].text for k in range(len(columns))]
            row_list.append(col_list)
            #print(col_list)
        totalCount = int(soup.find('totalCount').text)
        lastPage = math.ceil(totalCount/10)

        for page in range(2, lastPage+1): #마지막 페이지까지 반복
            print(page, '-' * 100)
            soup = getRequests(gu_code, year_month[i], page)
            rows = soup.find_all('item')

            for j in range(len(rows)):
                columns = rows[j].find_all()
                col_list =[columns[k].text for k in range(len(columns))]
                row_list.append(col_list)
                print(col_list)

    with open(f'data/{gu_code}구 아파트 전월세.csv', "w", encoding="utf-8-sig", newline="") as csv_file: #검색결과를 csv 파일로 저장
        writer = csv.writer(csv_file)
        writer.writerows(row_list)
```

- **인천 버스 번호를 입력 받아서 그 버스노선의 버스 정차정거장 목록, 첫차시간, 막차시간, 배차시간을 출력하라.**

[05.API] ex18.py

```
import requests
from bs4 import BeautifulSoup
from datetime import datetime
file = open('data/data_go_key.txt')
key = file.readlines()
enKey = key[0].strip()
deKey = key[1].strip()

#[인천광역시_버스노선 조회]-[노선버스 목록조회]
def getBusInfo(busNo): #버스번호를 인자로 버스정보(버스ID, 첫차시간, 막차시간, 배차간격) 반환 함수
    url = 'http://apis.data.go.kr/6280000/busRouteService/getBusRouteNo'
    params ={'serviceKey' : deKey,'pageNo':1, 'numOfRows':10, 'routeNo' : busNo }
    response = requests.get(url, params=params)
    soup = BeautifulSoup(response.content, 'lxml-xml')
    items = soup.select('itemList')
    busInfo = ''
    for item in items:
        if busNo == item.find('ROUTENO').text: #입력한 버스번호와 일치하는 경우
            busId = item.find('ROUTEID').text #버스ID
            firstTime = datetime.strptime(item.find('FBUS_DEPHMS').text, '%H%M') #첫차시간
            first = datetime.strftime(firstTime, '%H시%M분')
            lastTime =datetime.strptime(item.find('LBUS_DEPHMS').text, '%H%M') #막차시간
            last = datetime.strftime(lastTime, '%H시%M분')
            gap = item.find('MAX_ALLOCGAP').text #배차간격
            busInfo ={'busId':busId, 'first':first, 'last':last, 'gap':gap}
            break
    return busInfo

#[인천광역시_버스노선 조회]
def getBusRoute(busId): #버스ID를 인자로 정류장이름 목록을 반환하는 함수
    url = 'http://apis.data.go.kr/6280000/busRouteService/getBusRouteSectionList'
    params ={'serviceKey' : deKey, 'pageNo' : '1', 'numOfRows' : '200', 'routeId' : busId }
    response = requests.get(url, params=params)
    soup = BeautifulSoup(response.content, 'lxml-xml')
    items = soup.select('itemList')
    stop_list = []
    for item in items:
        stopName = item.find('BSTOPNM').text #정거장 이름
        stop_list.append(stopName)
    return stop_list

if __name__=='__main__':
    busNo = input('조회할 버스번호를 입력 하세요 :')
    busInfo = getBusInfo(busNo)
    if busInfo != '':
        print(f'첫차시간:{busInfo['first']} / 막차시간:{busInfo['last']} / 배차간격: {busInfo['gap']}분')
        stop_list = getBusRoute(busInfo['busId'])
        print(f'정차정거장:{stop_list[0:5]}~{stop_list[-5:]}')
        print('정거장수:', len(stop_list))
    else: print(f'{busNo}번 버스가 존재하지 않습니다.')
```

실행 결과

```
조회할 버스번호를 입력 하세요 :12
첫차시간:04시40분 / 막차시간:22시20분 / 배차간격: 8분
정차정거장:['항동7가버스차고지(삼거리)', '연안부두바다쉼터', '연안여객터미널(제주행)']~['연안부두로삼거리', '방파제입구', '항동7가버스차고지(삼거리)']
정거장수: 152
```

- **정거장을 입력받아서 이 정거장에 정차하는 버스목록 출력**

[05.API] ex19.py

```
import requests
from bs4 import BeautifulSoup

file = open('data/data_go_key.txt')
key = file.readlines()
enKey = key[0].strip()
deKey = key[1].strip()

#[인천광역시_정류소 조회]
def getStopName(stopName): #정류장명을 인자로 사용, 그 정류장명를 포함한 모든 정류장 목록을 반환하는 함수
    url = 'http://apis.data.go.kr/6280000/busStationService/getBusStationNmList'
    params ={'serviceKey' : deKey, 'pageNo' : '1', 'numOfRows' : '10', 'bstopNm' : stopName }
    response = requests.get(url, params=params)
    response.encoding='UTF-8'
    soup = BeautifulSoup(response.text, 'lxml-xml')
    items = soup.find_all('itemList')
    stop_list = []
    for item in items:
        st_id = item.find('BSTOPID').text #정류장ID
        st_name = item.find('BSTOPNM').text #정류장명
        bus_list = getBusNo(st_id)
        stop_list.append({'st_id':st_id, 'st_name':st_name, 'bus_list':bus_list})
    return stop_list

#[인천광역시_정류소 조회]-[정류소경유노선 목록 조회]
def getBusNo(stopId): #정류소ID를 인자로 그 정류소를 지나가는 버스노선 목록
    url = 'http://apis.data.go.kr/6280000/busStationService/getBusStationViaRouteList'
    params ={'serviceKey' : deKey, 'pageNo' : '1', 'numOfRows' : '10', 'bstopId' : stopId }
    response = requests.get(url, params=params)
    soup = BeautifulSoup(response.content, 'lxml-xml')
    items = soup.select('itemList')
    bus_list = []
    for item in items:
        busNo = item.find('ROUTENO').text #버스번호
        bus_list.append(busNo)
    return bus_list

if __name__=='__main__':
    stopName = input('정류장명에 포함된 문자열:')
    stop_list = getStopName(stopName)
    for stop in stop_list:
        print(f'정류장ID:{stop['st_id']}, 정류장명:{stop['st_name']}, 경유버스목록:{stop['bus_list']}')
```

실행 결과

```
정류장명에 포함된 문자열:루원시티
정류장ID:168001051, 정류장명:가정(루원시티)역, 경유버스목록:['M6458', '7700', '701', '702']
정류장ID:168001075, 정류장명:가정(루원시티)역, 경유버스목록:['1', '12', '13', '42', '80', '103', '202', '584', '591', '1000']
정류장ID:168001076, 정류장명:루원시티사거리, 경유버스목록:['13', '800', '71']
정류장ID:168001184, 정류장명:루원시티어울림아파트, 경유버스목록:['1', '591', '43-1']
정류장ID:168001309, 정류장명:루원시티프라디움105동, 경유버스목록:['1', '202', '591', '3-2', '594', '47', '71']
정류장ID:168000980, 정류장명:포레나루원시티아파트, 경유버스목록:['596']
정류장ID:168000981, 정류장명:포레나루원시티아파트, 경유버스목록:['596']
정류장ID:168001072, 정류장명:루원시티2차리더스뷰아파트, 경유버스목록:['12', '14', '80', '103', '202', '584', '2-1', '594', '86', '596']
정류장ID:168001050, 정류장명:가정(루원시티)역, 경유버스목록:['M6458', '7700', '701', '702']
정류장ID:168001078, 정류장명:가정(루원시티)역, 경유버스목록:['1', '12', '13', '42', '80', '103', '202', '584', '591', '1000']
```

# 07. Scrapping 결과 웹페이지로 서비스

## • 웹페이지 Layout 작성하기

### 1) flask 라이브러리 설치 웹서버 실행

```
pip install flask
python app.py
```

### 2) 웹페이지 Layout 작성

[data]-[python]-[web]-[ex02]-[templates] header.html

```
<div>
    <img src="/static/images/header01.png" width="100%">
</div>
```

[data]-[python]-[web]-[ex02]-[templates] footer.html

```
<div class="my-5" style="margin-bottom: 300px;">
    <hr>
    <h4 class="text-center">Copyright 2023 홍길동 All rights reserved.</h4>
</div>
```

[data]-[python]-[web]-[ex02]-[templates] menu.html

```
<div class="mt-5">
    <a href="/" class="me-3">Home</a>
    <a href="/movie/chart" class="me-3">뮤비챠트</a>
    <hr>
</div>
```

[data]-[python]-[web]-[ex02]-[templates] menu.html

```
<div class="my-5">
    <h1 class="text-center">홈페이지</h1>
</div>
```

[data]-[python]-[web]-[ex02]-[templates] index.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'%}
        {%include 'menu.html'%}
        {%include pageName%}
        {%include 'footer.html'%}
    </div>
</body>
</html>
```

[data]-[python]-[web]-[ex02]-[static]-[css] style.css

```
@import url('https://fonts.googleapis.com/css2?family=Sunflower:wght@300&display=swap');

* { font-family: "Sunflower", sans-serif; font-weight: 300;  font-style: normal; }

.ellipsis { overflow: hidden; text-overflow: ellipsis; display: -webkit-box; -webkit-line-clamp: 1; -webkit-box-orient: vertical; }
```

### 3) 웹서버 설정 파일 작성

[data]-[python]-[web]-[ex02] app.py

```
from flask import Flask, render_template

app = Flask(__name__, template_folder='templates')

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

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

## • CGV 무비차트 목록 프로그램

### 1) 무비차트 출력 페이지 작성

[data]-[python]-[web]-[ex02]-[templates]-[movie] chart.html

```
<div class="my-5">
    <h1 class="text-center mb-5">{{title}}</h1>
</div>
```

### 2) 무비차트 출력 라우터 작성

[data]-[python]-[web]-[ex02]-[routes] movie.py

```
from flask import Blueprint, render_template

bp = Blueprint('bbs', __name__, url_prefix='/movie')

@bp.route('/chart')
def chart():
    return render_template('index.html', title='무비차트', pageName='movie/chart.html')
```

### 3) movie.py 라우터 app.py에 등록

[data]-[python]-[web]-[ex02] app.py

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

app = Flask(__name__, template_folder='templates')
app.register_blueprint(movie.bp)

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

4) movie.py 라우터에 무비차트 Scraping 함수를 작성한다.

[data]-[python]-[web]-[ex02]-[routes] movie.py

```
from flask import Blueprint, render_template
from bs4 import BeautifulSoup
import requests

bp = Blueprint('bbs', __name__, url_prefix='/movie')
@bp.route('/chart.json')
def scrap_chart():
    url = 'http://www.cgv.co.kr/movies/?lt=1&ft=0'
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, 'lxml')
    es = soup.find('div', attrs={'class':'sect-movie-chart'}).find_all('li', limit=12)
    items = []
    for e in es:
        title = e.find('strong', attrs={'class':'title'}).get_text()
        image = e.find('img')['src']
        percent = e.find('strong', attrs={'class':'percent'}).get_text()
        date = e.find('span', attrs={'txt-info'}).get_text()
        reservation = 'http://cgv.co.kr' + e.find('a', attrs={'class':'link-reservation'})['href']
        item = {'title': title, 'image':image, 'percent':percent, 'date':date, 'reservation':reservation}
        items.append(item)
    return items
```

5) CGV 무비차트 Scraping 결과를 chart.html 페이지에 출력한다.

[data]-[python]-[web]-[ex02]-[templates]-[movie] chart.html

```
<div class="my-5">
    <h1 class="text-center mb-5">{{title}}</h1>
    <div id="div_chart" class="row"></div>
    {%raw%}
        <script id="temp_chart" type="x-handlebars-template">
            {{#each .}}
                <div class="col-6 col-md-3 col-lg-2 mb-3">
                    <div class="card">
                        <div class="card-body" style="font-size:12px;">
                            <img src="{{image}}" width="100%" class="mb-2">
                            <div class="ellipsis">{{title}}</div>
                            <div>{{percent}}</div>
                            <div>{{date}}</div>
                            <div><a href="{{reservation}}" class="btn btn-danger btn-sm px-5">예매하기</a></div>
                        </div>
                    </div>
                </div>
            {{/each}}
        </script>
    {%endraw%}
</div>
<script>
        $.ajax({
            type:'get',
            url:'/movie/chart.json',
            data:'json',
            success:function(data){
                const temp=Handlebars.compile($("#temp_chart").html());
                $("#div_chart").html(temp(data));
            }
        });
</script>
```

- **Scraping 결과 JSON 파일로 저장과 읽기**

**[data]-[python]-[web]-[ex02]-[routes] movie.py**

```
from flask import Blueprint, render_template
from bs4 import BeautifulSoup
import requests
import json

bp = Blueprint('bbs', __name__, url_prefix='/movie')

@bp.route('/chart.json')
def scrap_chart():
    url = 'http://www.cgv.co.kr/movies/?lt=1&ft=0'
    res = requests.get(url)
    res.raise_for_status()
    soup = BeautifulSoup(res.text, 'lxml')
    e = soup.find('div', attrs={'class':'sect-movie-chart'})
    es = e.find_all('li', limit=12)
    items = []
    for e in es:
        ...
    with open('static/movie.json', 'w', encoding='utf8') as file:
        json.dump(items, file, indent='\t', ensure_ascii=False)

    with open('static/movie.json', 'r', encoding='utf8') as file:
        json_data = json.load(file)
        print(type(json_data))
...
```

- **로컬 JSON 파일 html에서 읽기**

**[data]-[python]-[web]-[ex02]-[static] movie.json**

```
data = [
    {
        "title": "데드풀과 울버린",
        "image": "https://img.cgv.co.kr/Movie/Thumbnail/Poster/000088/88228/88228_320.jpg",
        "percent": "예매율19.2%",
        "date": "\n\r\n                                2024.07.24 \r\n                                개봉\nD-6\n\n",
        "reservation": "http://cgv.co.kr/ticket/?MOVIE_CD=20037162&MOVIE_CD_GROUP=20036434"
    },
    {
        "title": "명탐정 코난-100만 달러의 펜타그램",
        "image": "https://img.cgv.co.kr/Movie/Thumbnail/Poster/000088/88390/88390_320.jpg",
        "percent": "예매율14.2%",
        "date": "\n\r\n                                2024.07.17 \r\n                                개봉\n\n",
        "reservation": "http://cgv.co.kr/ticket/?MOVIE_CD=20036928&MOVIE_CD_GROUP=20036928"
    },
    ...
]
```

**[data]-[python]-[web]-[ex02]-[templates]-[movie] chart.html**

```
<script src="/static/movie.json"></script>
<script>
    const temp=Handlebars.compile($("#temp_chart").html());
    $("#div_chart").html(temp(data));
</script>
```

- **증권 현황 프로그램**

**1) 메뉴 페이지에 증권 현황 메뉴를 추가하고 증권 현황 출력 페이지를 작성한다.**

[data]-[python]-[web]-[ex02]-[templates] menu.html

```
<div class="mt-5">
    <a href="/" class="me-3">Home</a>
    <a href="/movie/chart" class="me-3">뮤비챠트</a>
    <a href="/finance/list" class="me-3">주식현황</a>
    <hr>
</div>
```

[data]-[python]-[web]-[ex02]-[templates]-[finance] list.html

```
<div class="my-5">
    <h1 class="text-center mb-5">주식현황</h1>
    <div class="row px-5">
        <div class="col-md-6">
            <h5 class="text-center">TOP종목</h5>
            <div id="div_top"></div>
        </div>
        <div class="col-md-6">
            <h5 class="text-center">인기검색종목</h5>
            <div id="div_popular"></div>
        </div>
    </div>
</div>
```

**2) 증권 현황 출력을 위한 라우터를 작성하고 app.py에 등록한다.**

[data]-[python]-[web]-[ex02]-[routes] finance.py

```
from flask import Blueprint, render_template

bp = Blueprint('finance', __name__, url_prefix='/finance')

@bp.route('/list')
def list():
    return render_template('index.html', title='주식현황', pageName='finance/list.html')
```

[data]-[python]-[web]-[ex02] app.py

```
from flask import Flask, render_template
from routes import movie, finance

app = Flask(__name__, template_folder='templates')
app.register_blueprint(movie.bp)
app.register_blueprint(finance.bp)

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

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

### 3) finance.py 라우터에 TOP종목, 인기검색종목 Scrapping 함수를 작성한다.

[data]-[python]-[web]-[ex02]-[routes] finance.py

```
from flask import Blueprint, render_template
import requests
from bs4 import BeautifulSoup

bp = Blueprint('finance', __name__, url_prefix='/finance')

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

def scrap_top():
    url = 'https://finance.naver.com/'
    soup = create_soup(url)

    items = []
    es = soup.find('tbody', attrs={'id':'_topItems1'}).find_all('tr', limit=5)
    for e in es:
        title = e.find('a').get_text()
        rows = e.find_all('td')
        price = rows[0].get_text()
        up_down = rows[1].get_text()
        rate = rows[2].get_text().strip()
        item = {'title':title,'price':price,'up_down':up_down,'rate':rate}
        items.append(item)
    return items

def scrap_popular():
    url = 'https://finance.naver.com/'
    soup = create_soup(url)

    items = []
    e= soup.find('div', attrs={'class':'aside_area aside_popular'}).find('tbody')
    es = e.find_all('tr')
    for e in es:
        title = e.find('a').get_text()
        price = e.find_all('td')[0].get_text()
        up_down = e.find_all('td')[1].find('em').get_text().strip()
        up_down_price = e.find_all('td')[1].find('span', attrs={'class':'tah'}).get_text().strip()
        up_down = up_down + up_down_price
        item = {'title':title, 'price':price, 'up_down':up_down}
        items.append(item)
    return items

@bp.route('/list.json')
def listJson():
    top = scrap_top()
    popular = scrap_popular()
    data = {'top':top, 'popular': popular}
    return data

@bp.route('/list')
def list():
    return render_template('index.html', title='주식현황', pageName='finance/list.html')
```

4) TOP종목, 인기검색종목 출력 html 페이지 출력 프로그램을 작성한다.

[data]-[python]-[web]-[ex02]-[templates]-[finance] list.html

```
<div>
    ...
    {%raw%}
    <script id="temp_top" type="x-handlebars-template">
        <table class="table table-striped">
            <tr>
                <th>종목명</th>
                <th class="text-end">현재가</th>
                <th class="text-end">전일비</th>
                <th class="text-end">등락률</th>
            </tr>
            {{#each top}}
            <tr class="text-end">
                <td class="text-start">{{title}}</td>
                <td style="{{fnc_up_down_style up_down}}">{{price}}</td>
                <td style="{{fnc_up_down_style up_down}}">{{fnc_up_down up_down}}</td>
                <td style="{{fnc_up_down_style up_down}}">{{rate}}</td>
            </tr>
            {{/each}}
        </table>
    </script>
    <script id="temp_popular" type="x-handlebars-template">
        <table class="table table-striped">
            <tr>
                <th>종목명</th>
                <th class="text-end">현재가</th>
                <th class="text-end">전일비</th>
            </tr>
            {{#each popular}}
            <tr class="text-end">
                <td class="text-start">{{title}}</td>
                <td style={{fnc_up_down_style up_down}}>{{price}}</td>
                <td style={{fnc_up_down_style up_down}}>{{fnc_up_down up_down}}</td>
            </tr>
            {{/each}}
        </table>
    </script>
    <script>
        Handlebars.registerHelper("fnc_up_down", function(up_down){
            if(up_down.substr(0,2) == "상승")return "▲" + up_down.substr(2);
            elsereturn "▼" + up_down.substr(2);
        });
        Handlebars.registerHelper("fnc_up_down_style", function(up_down){
            if(up_down.substr(0,2) == "상승")return "color:red;";
            else return "color:blue;"
        });
    </script>
    {%endraw%}
</div>
<script>
    $.ajax({
        type:"get",
        url:"/finance/list.json",
        dataType:"json",
        success:function(data){
            const temp=Handlebars.compile($("#temp_top").html());
            $("#div_top").html(temp(data));
            const temp1=Handlebars.compile($("#temp_popular").html());
            $("#div_popular").html(temp1(data));
        }
    });
</script>
```

- **상품검색 프로그램**

**1) 메뉴 페이지에 상품검색 메뉴를 추가하고 상품검색 출력 페이지를 작성한다.**

[data]-[python]-[web]-[ex02]-[templates] menu.html

```
<div class="mt-5">
    <a href="/" class="me-3">Home</a>
    <a href="/movie/chart" class="me-3">뮤비챠트</a>
    <a href="/finance/list" class="me-3">증권현황</a>
    <a href="/shop/search" class="me-3">상품검색</a>
    <hr>
</div>
```

[data]-[python]-[web]-[ex02]-[templates]-[shop] search.html

```
<div class="my-5">
    <h1 class="text-center mb-5">상품검색</h1>
    <div class="row">
        <div class="col-md-5 col-lg-4">
            <form name="frm">
                <div class="input-group">
                    <input name="query" value="노트북" class="form-control" placeholder="검색어">
                    <button class="btn btn-primary">검색</button>
                </div>
            </form>
        </div>
    </div>
</div>
```

**2) 상품검색 출력을 위한 라우터를 작성하고 app.py에 등록한다.**

[data]-[python]-[web]-[ex02]-[routes] shop.py

```
from flask import Blueprint, render_template
import requests
from bs4 import BeautifulSoup

bp = Blueprint('shop', __name__, url_prefix='/shop')

@bp.route('search')
def search():
    return render_template('index.html', title='상품검색', pageName='shop/search.html')
```

[data]-[python]-[web]-[ex02] app.py

```
from flask import Flask, render_template
from routes import movie, finance, shop

app = Flask(__name__, template_folder='templates')
app.register_blueprint(movie.bp)
app.register_blueprint(finance.bp)
app.register_blueprint(shop.bp)

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

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

### 3) shop.py 라우터에 상품검색 Scrapping 함수를 작성한다.

[data]-[python]-[web]-[ex02]-[routes] shop.py

```
from flask import Blueprint, render_template, request
import requests
from bs4 import BeautifulSoup
bp = Blueprint('shop', __name__, url_prefix='/shop')

def scrap_shop(query):
    url = 'https://www.coupang.com/np/search?q={}&channel=recent...&backgroundColor='.format(query)
    headers={'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) ... Chrome/126.0.0.0 Safari/537.36',
             "Accept-Language":"ko-KR,ko;q=0.8,en-US;q=0.5,en;q=0.3"}
    res = requests.get(url, headers = headers)
    res.raise_for_status()

    soup = BeautifulSoup(res.text, 'lxml')
    es = soup.find_all('li', attrs={'class':'search-product'})
    items = []
    index = 0
    for e in es:
        id = e.attrs.get('id')
        name = e.find('div', attrs={'class':'name'})
        if name:
            name = name.get_text().strip()
        else:
            continue
        price = e.find('strong', attrs={'class':'price-value'})
        if price:
            price= price.get_text().strip()
        else:
            continue
        image = e.find('img', attrs={'class':'search-product-wrap-img'})
        if image:
            if 'blank' in image['src']:
                continue
            else:
                image = 'https:' + image['src']
        else:
            continue
        rate = e.find('em', attrs={'class', 'rating'})
        if rate:
            rate = rate.get_text()
        else:
            continue
        review_cnt = e.find('span', attrs={'class':'rating-total-count'})
        if review_cnt:
            review_cnt = review_cnt.get_text()[1:-1]
        else:
            continue
        index += 1
        item = {'index':index,'sid':id,'title':name,'price':price,'image':image,'rate':rate, 'review_cnt':review_cnt}
        items.append(item)
    return items

@bp.route('/search.json')
def searchJSON():
    args = request.args
    query = args['query']
    items = scrap_shop(query)
    return items

@bp.route('search')
def search():
    return render_template('index.html', title='상품검색', pageName='shop/search.html')
```

4) 상품검색을 위한 html 페이지를 작성한다.

[data]-[python]-[web]-[ex02]-[templates]-[shop] search.html

```
<div class="my-5">
    <h1 class="text-center mb-5">상품검색</h1>
    <div class="row">
        ...
        <hr>
    </div>
    <div id="div_search"></div>
    {%raw%}
    <script id="temp_search" type="x-handlebars-template">
        <table class="table">
            {{#each .}}
            <tr style="font-size:12px;">
                <td>{{index}}</td>
                <td><img src="{{image}}" width="50px"></td>
                <td with="80%">
                    <div class="ellipsis">{{title}}</div>
                    <div>별점:{{rate}}</div>
                    <div>리뷰수:{{review_cnt}}</div>
                </td>
                <td width="100">
                    <button class="btn btn-sm btn-outline-primary">등록</button>
                </td>
            </tr>
            {{/each}}
        </table>
    </script>
    {%endraw%}
</div>
<script>
    let query=$(frm.query).val();
    getList();

    $(frm).on("submit", function(e){
        e.preventDefault();
        query=$(frm.query).val();
        if(query == "") {
            alert("검색어를 입력하세요.");
            return;
        }
        getList();
    });

    function getList(){
        $("#loading").show();
        $.ajax({
            type:"get",
            url:"/shop/search.json",
            data:{query},
            success:function(data){
                const temp=Handlebars.compile($("#temp_search").html());
                $("#div_search").html(temp(data));
                $("#loading").hide();
            }
        });
    }
</script>
```

5) 로딩바 출력을 위해 index.html 페이지를 수정한다. https://loading.io/

[data]-[python]-[web]-[ex02]-[templates] index.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>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/twbs-pagination/1.4.2/jquery.twbsPagination.min.js"></script>
    <title>{{title}}</title>
    <style>
        #loading {
            position: absolute;
            display: block;
            left:50%;
            top:50%;
            display: none;
        }
</style>
</head>
<body>
    <div class="container">
        {%include 'header.html'%}
        {%include 'menu.html'%}
        <img src="/static/images/loading.gif" id="loading">
        {%include pageName%}
        {%include 'footer.html'%}
    </div>
</body>
</html>
```

6) 목록 페이지에 페이지네이션 기능을 추가한다.

[data]-[python]-[web]-[ex02]-[templates]-[shop] search.html

```
<div class="my-5">
    <h1 class="text-center mb-5">상품검색</h1>
    <div class="row">
        <div class="col-md-5 col-lg-4">
            <form name="frm">
                <div class="input-group">
                    <input name="query" value="노트북" class="form-control" placeholder="검색어">
                    <button class="btn btn-primary">검색</button>
                </div>
            </form>
        </div>
        <hr>
    </div>
    <div id="div_search"></div>
    <div id="pagination" class="pagination justify-content-center mt-5" style="display:none;"></div>
     ...
</div>
```

[data]-[python]-[web]-[ex02]-[templates]-[shop] search.html

```
<script>
    let page=1;
    let size=5;
    let totalPages=1;
    let result;
    let query=$(frm.query).val();
    getList();

    $(frm).on("submit", function(e){
        ...
        page=1;
        getList();
    });

    function getList(){
        $("#loading").show();
        $.ajax({
            ...
            success:function(data){
                result = data;
                getPagination();
                $("#loading").hide();
            }
        });
    }

    function getPagination() {
        let total = result.length;
        let curTotalPages = total == 0 ? 1 : Math.ceil(total/size);
        if(total > size) $("#pagination").show();
        else $("#pagination").hide();
        if(totalPages != curTotalPages) {
            totalPages = curTotalPages;
            $("#pagination").twbsPagination("changeTotalPages", totalPages, page);
        }
        let start = ((page - 1) * size) + 1;
        let end = (page * size);
        let items = [];
        result.forEach(item => {
          let idx = parseInt(item.index);
          if(idx >= start && idx <= end) items.push(item);
        });
        const temp=Handlebars.compile($("#temp_search").html());
        $("#div_search").html(temp(items));
    }

    $('#pagination').twbsPagination({
        totalPages:totalPages,
        visiblePages: 5,
        startPage : 1,
        initiateStartPageClick: false,
        first:'<<',
        prev :'<',
        next :'>',
        last :'>>',
        onPageClick: function (event, clickedPage) {
            if(page != clickedPage) {
                page=clickedPage;
                getPagination();
            }
        }
    });
</script>
```

- **상품등록 프로그램**

**1) 상품정보 등록을 위한 테이블을 생성한다.**

```
create table shop(
    sid char(10) not null primary key,
    title varchar(500) not null,
    price int default 0,
    image varchar(500),
    rate float default 0,
    review_cnt int default 0,
    reg_date datetime default now()
);
```

**2) Database 접속 프로그램을 작성한다.**

[data]-[python]-[web]-[ex02]-[dao] config.json

```
{
    "host": "localhost",
    "user": "web",
    "password": "pass",
    "db":"webdb"
}
```

---

[data]-[python]-[web]-[ex02]-[dao] db.py

```
import pymysql
import json

file =open('./dao/config.json', 'r', encoding='utf-8')
config = json.loads(file.read())

connection = pymysql.connect(
    host=config['host'],
    user=config['user'],
    password=config['password'],
    db=config['db'],
    charset='utf8',
    cursorclass=pymysql.cursors.DictCursor)
```

---

**2) dao shop.py에 상품등록을 위한 함수를 작성한다.**

[data]-[python]-[web]-[ex02]-[dao] shop.py

```
from dao import db

def insert(shop):
    try:
        with db.connection.cursor() as cursor:
            sql = "insert into shop(sid, title, price, image, rate, review_cnt) values(%s,%s,%s,%s,%s,%s)"
            price = int(shop.get('price').replace(',', ''))
            cursor.execute(sql, (shop.get('sid'),shop.get('title'),price,shop.get('image'),shop.get('rate'),shop.get('review_cnt')))
            db.connection.commit()
            return 'success'
    except Exception as err:
        print('등록오류:', err)
        return 'fail'
    finally:
        cursor.close()
```

---

3) 라우터 shop.py에 상품등록을 위한 함수를 작성한다.

[data]-[python]-[web]-[ex02]-[routes] shop.py

```
from flask import Blueprint, render_template, request
import requests
from bs4 import BeautifulSoup
import json
from dao import shop as shopDAO
...
@bp.route('insert', methods=['POST'])
def insertPost():
    req = json.loads(request.get_data())
    result = shopDAO.insert(req)
    return result
```

4) search.html 파일에 상품등록을 위한 프로그램을 추가한다.

[data]-[python]-[web]-[ex02]-[templates]-[shop] search.html

```
{%raw%}
<script id="temp_search" type="x-handlebars-template">
    <table class="table">
        {{#each .}}
        <tr style="font-size:12px;">
            <td>{{index}}</td>
            <td><img src="{{image}}" width="50px"></td>
            <td with="80%">
                <div class="ellipsis">{{title}}</div>
                <div>ID:{{sid}}</div>
                <div>별점:{{rate}}</div>
                <div>리뷰수:{{review_cnt}}</div>
            </td>
            <td width="100">
                <button class="btn btn-sm btn-outline-primary insert" shop="{{fnc_shop @this}}">등록</button>
            </td>
        </tr>
        {{/each}}
    </table>
</script>
<script>
    Handlebars.registerHelper("fnc_shop", function(shop){
        return JSON.stringify(shop);
    });
</script>
{%endraw%}
<script>
    ...
    $("#div_search").on("click", ".insert", function(){  //등록 버튼을 클릭한 경우
        let data = JSON.parse($(this).attr("shop"));
        $.ajax({
            type:"post",
            url:"/shop/insert",
            data:JSON.stringify(data),
            success:function(data){
                if(data == 'success'){
                    alert("새로운 상품이 등록되었습니다.");
                }else{
                    alert("이미 등록된 상품입니다.");
                }
            }
        });
    });
</script>
```

- **상품목록 프로그램**

**1) 상품목록 페이지를 작성하고 라우터에 등록 후 메뉴에 추가한다.**

[data]-[python]-[web]-[ex02]-[templates]-[shop] list.html

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

[data]-[python]-[web]-[ex02]-[routes] shop.py

```
...
@bp.route('/list')
def list():
    return render_template('index.html', title='상품목록', pageName='shop/list.html')
```

[data]-[python]-[web]-[ex02]-[templates] menu.html

```
<div class="mt-5">
    <a href="/" class="me-3">Home</a>
    <a href="/movie/chart" class="me-3">뮤비챠트</a>
    <a href="/finance/list" class="me-3">증권현황</a>
    <a href="/shop/search" class="me-3">상품검색</a>
    <a href="/shop/list" class="me-3">상품목록</a>
    <hr>
</div>
```

**2) 상품목록 출력을 위해 dao와 router에 list 함수를 작성한다.**

[data]-[python]-[web]-[ex02]-[doa] shop.py

```
from dao import db

def insert(shop):
  try:
    with db.connection.cursor() as cursor:
      sql = "insert into shop(sid, title, price, image, rate, review_cnt) values(%s,%s,%s,%s,%s,%s)"
      price = int(shop.get('price').replace(',',''))
      cursor.execute(sql, (shop.get('sid'), shop.get('title'), price, shop.get('image'), shop.get('rate'), shop.get('review_cnt')))
      db.connection.commit()
      return 'success'
  except Exception as err:
    print('등록오류:', err)
    return 'fail'
  finally:
    cursor.close()

def list():
    try:
        with db.connection.cursor() as cursor:
            sql = "select *, format(price,0) fmtprice, date_format(reg_date, '%Y-%m-%d %T') fmtdate \
                  from shop \
                  order by reg_date desc"
            cursor.execute(sql)
            rows = cursor.fetchall()
            return rows
    except Exception as err:
        print('목록오류:', err)
        return 'fail'
    finally:
        cursor.close()
```

[data]-[python]-[web]-[ex02]-[routes] shop.py

```
from flask import Blueprint, render_template, request
import requests
from bs4 import BeautifulSoup
import json
from dao import shop as shopDAO
...
@bp.route('/list')
def list():
  return render_template('index.html', title='상품목록', pageName='shop/list.html')

@bp.route('/list.json')
def listJSON():
    rows = shopDAO.list()
    return rows
```

3) list.html 페이지에 상품목록 출력 프로그램을 추가한다.

[data]-[python]-[web]-[ex02]-[templates]-[shop] list.html

```
<div class="my-5">
    <h1 class="text-center mb-5">상품목록</h1>
    <div id="div_list"></div>
    {%raw%}
    <script id="temp_list" type="x-handlebars-template">
        <table class="table" style="font-size:12px;">
            {{#each .}}
            <tr>
                <td><img src="{{image}}" width="50px"></td>
                <td width="70%">
                    <div class="ellipsis">{{title}}</div>
                    <div>ID:{{sid}}</div>
                    <div>가격:{{fmtprice}}</div>
                    <div>등록일:{{fmtdate}} ({{rate}}/{{review_cnt}})</div>
                </td>
                <td>
                    <button class="btn btn-sm btn-outline-danger">삭제</button>
                </td>
            </tr>
            {{/each}}
        </table>
    </script>
    {%endraw%}
</div>
<script>
    getList();
    function getList(){
        $.ajax({
            type:"get",
            url:"/shop/list.json",
            success:function(data){
                const temp=Handlebars.compile($("#temp_list").html());
                $("#div_list").html(temp(data));
            }
        });
    }
</script>
```

- **상품목록 페이징 프로그램**

**1) 페이징 처리를 위해 dao의 list 함수를 수정한다.**

[data]-[python]-[web]-[ex02]-[doa] shop.py

```
def list(args):
    page = int(args['page'])
    size = int(args.get('size'))
    start = (page-1) * size
    try:
        with db.connection.cursor() as cursor:
            sql = "select *,format(price,0) fmtprice,date_format(reg_date, '%%Y-%%m-%%d %%T') fmtdate \
                  from shop \
                  order by reg_date desc limit %s, %s"
            cursor.execute(sql, (start, size))
            rows = cursor.fetchall()
            return rows
    except Exception as err:
        print('목록오류:', err)
        return 'fail'
    finally:
        cursor.close()

def total():
    try:
        with db.connection.cursor() as cursor:
            sql = "select count(*) total from shop"
            cursor.execute(sql)
            return cursor.fetchone()
    except Exception as err:
        print('데이터갯수오류', err)
    finally:
        cursor.close()
```

**2) 페이징 처리를 위해 router의 listJSON() 함수를 수정한다.**

[data]-[python]-[web]-[ex02]-[routes] shop.py

```
from flask import Blueprint, render_template, request
import requests
from bs4 import BeautifulSoup
import json
from dao import shop as shopDAO

@bp.route('/list')
def list():
    return render_template('index.html', title='상품목록', pageName='shop/list.html')

@bp.route('/list.json')
def listJSON():
    args = request.args
    row = shopDAO.total()
    rows = shopDAO.list(args)
    data = {'total': row.get('total'), 'list': rows}
    return data
```

### 3) list.html 페이지에 페이징 기능을 추가한다.

[data]-[python]-[web]-[ex02]-[templates]-[shop] list.html

```
<div class="my-5">
    <h1 class="text-center mb-5">상품목록</h1>
    <div>
        상품수:<span id="total"></span>개<hr>
    </div>
    <div id="div_list"></div>
    <div id="pagination" class="pagination justify-content-center mt-5" style="display:none;"></div>
    {%raw%}
    <script id="temp_list" type="x-handlebars-template">
        <table class="table" style="font-size:12px;">
            {{#each list}}
                ...
            {{/each}}
        </table>
    </script>
    {%endraw%}
</div>
<script>
    let page=1;
    let size=5;
    let totalPages=1;
    let total=0;
    getList();
    function getList(){
        $.ajax({
            type:"get",
            url:"/shop/list.json",
            data:{page, size},
            success:function(data){
                const temp=Handlebars.compile($("#temp_list").html());
                $("#div_list").html(temp(data));
                //상품 수가 size보다 크면 Pagination이 보이고 작으면 Pagination을 숨긴다.
                total=data.total;
                $("#total").html(total);
                if(total > size) $("#pagination").show()
                else $("#pagination").show();
                //현재 페이지 수가 이전 페이지 수와 다르면 Pagination의 페이지 수를 변경한다.
                const curTotalPages = total === 0 ? 1 : Math.ceil(total/size)
                if(totalPages != curTotalPages){
                    totalPages = curTotalPages;
                    $("#pagination").twbsPagination("changeTotalPages", totalPages, page)
                }
            }
        });
    }

    //Pagination 출력 함수
    $('#pagination').twbsPagination({
        totalPages:totalPages,
        visiblePages: 5,
        startPage : 1,
        initiateStartPageClick: false,
        first:'<<',  prev :'<',  next :'>',  last :'>>',
        onPageClick: function (event, clickedPage) {
            //현재 페이지와 클릭한 페이지가 다른 경우에만 getList() 함수를 실행한다.
            if(page != clickedPage) {
                page=clickedPage;
                getList();
            }
        }
    });
</script>
```

- **상품삭제 프로그램**

**1) 상품삭제를 위해 dao의 delete 함수를 수정한다.**

[data]-[python]-[web]-[ex02]-[doa] shop.py

```
...
def delete(sid):
    try:
        with db.connection.cursor() as cursor:
            sql = "delete from shop where sid=%s"
            cursor.execute(sql, sid)
            db.connection.commit()
            return 'success'
    except Exception as err:
        print('삭제오류', err)
        return 'fail'
    finally:
        cursor.close()
```

**2) 상품삭제를 위해 router의 delete() 함수를 작성한다.**

[data]-[python]-[web]-[ex02]-[routes] shop.py

```
...
@bp.route('/delete/<int:sid>', methods=['POST'])
def delete(sid):
    result = shopDAO.delete(sid)
    return result
```

**3) list.html 페이지에 상품삭제 프로그램을 추가한다.**

[data]-[python]-[web]-[ex02]-[templates]-[shop] list.html

```
{{#each list}}
    <tr>
        ...
        <td>
            <button class="btn btn-sm btn-outline-danger delete" sid="{{sid}}">삭제</button>
        </td>
    </tr>
{{/each}}
<script>
    ...
    //삭제 버튼을 클릭한 경우
    $("#div_list").on("click", ".delete", function(){
        const sid = $(this).attr("sid");
        if(!confirm(`${sid}번 상품을 삭제하실래요?`)) return;
        $.ajax({
            type:"post",
            url:`/shop/delete/${sid}`,
            success:function(data){
                if(data == 'success'){
                    if((page-1)*size == total-1) page--;
                    getList();
                }
            }
        });
    });
</script>
```

- **선택한 이미지 저장 프로그램**

**1) router에 이미지 저장하는 함수를 작성한다.**

[data]-[python]-[web]-[ex02]-[routes] shop.py

```
...
@bp.route('/image/save')
def download():
    try:
        args = request.args
        url = args['url']
        sid = args['sid']
        res_image = requests.get(url)
        res_image.raise_for_status()
        with open(f'static/images/shop/{sid}.jpg', 'wb') as file:
            file.write(res_image.content)
            return 'success'
    except Exception as err:
        print('이미지저장오류:', err)
        return 'fail'
```

---

**2) list.html에 이미지 저장 함수를 추가한다.**

[data]-[python]-[web]-[ex02]-[templates]-[shop] list.html

```
<script id="temp_list" type="x-handlebars-template">
    <table class="table" style="font-size:12px;">
        {{#each list}}
        <tr>
            <td><img src="{{image}}" width="50px" sid="{{sid}}"></td>
            <td width="70%">
                <div class="ellipsis">{{title}}</div>
                <div>ID:{{sid}}</div>
                <div>가격:{{fmtprice}}</div>
                <div>등록일:{{fmtdate}} ({{rate}}/{{review_cnt}})</div>
            </td>
            <td>
                <button class="btn btn-sm btn-outline-danger delete" sid="{{sid}}">삭제</button>
            </td>
        </tr>
        {{/each}}
    </table>
</script>
<script>
    ...
    $("#div_list").on("click", "img", function(){  //이미지를 클릭한 경우
        const url = $(this).attr("src");
        const sid = $(this).attr("sid");
        if(!confirm("선택한 이미지를 다운로드하시겠습니까?")) return;
        $.ajax({
            type:"get",
            url:`/shop/image/save?url=${url}&sid=${sid}`,
            success:function(data){
                if(data == 'success'){
                    alert("이미지 저장이 완료되었습니다.");
                }
             }
        });
    });
</script>
```

---