Python Crawling

‘내가 개발할 때 100% 만족할 만한 데이터의 형태는 이 세상에 존재하지 않는다’

> 이전까지 진행했던 프로젝트들의 경우 해당 주제에 맞는 데이터들을 공공데이터포털 사이트에서 서칭한후, 해당 데이터를 그대로 DB 에 전달하여 관련 백엔드 로직을 구현하였다.
> 이럴 경우 해당 데이터를 기준으로 기존 DB 테이블 설계를 수정해야 하고, 그에 맞춘 백엔드 로직을 작성하게 되어 주객전도가 된듯한 느낌이 들면서 스트레스를 많이 받았다.
> 이번 프로젝트에는 팀에서 DB 설계를 기준으로 그에 맞는 데이터를 직접 기호에 맞게 수정하여 작업 효율성을 높이는 것을 목적으로 Python Crawling 에 대한 공부를 하였다.

정적 크롤링

> ‘한 페이지 안에서 원하는 정보가 모두 드러날때 데이터를 수집하는 방법’

> 수집할 데이터는 요양 시설에 대한 이름, 전화번호, 주소

▼ 1번째 Code

```
import requests
from bs4 import BeautifulSoup
from openpyxl import Workbook

# 크롤링할 웹 페이지 URL
url = 'http://silver-dot.com/search?sch_city=1&sch_type=&q

# 엑셀 파일 생성
```

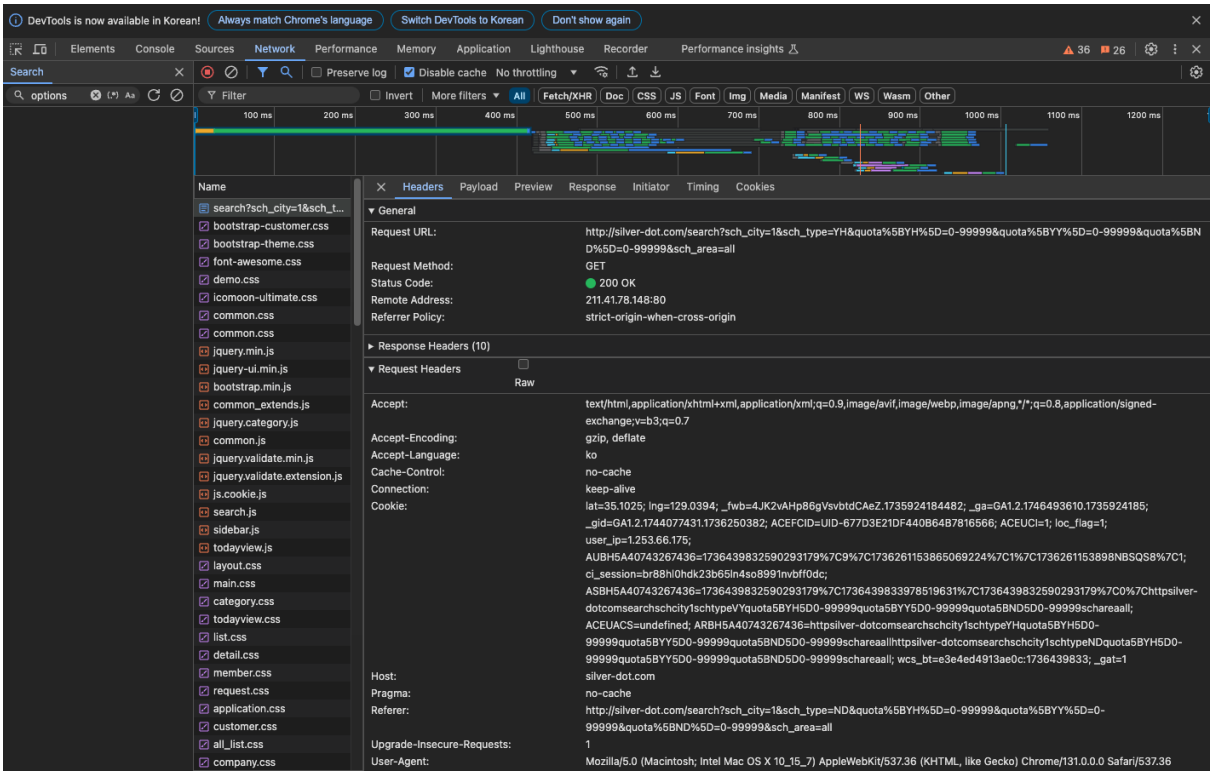
```
wb = Workbook()
ws = wb.active
ws.title = "Crawl Data"

# 엑셀 파일 첫 번째 행에 컬럼명 추가
ws.append(["이름", "전화번호", "주소"])

# 웹 페이지 요청 (GET 방식)
response = requests.get(url)

# 응답 상태 코드가 200(정상)일 경우
if response.status_code == 200:
    print('연결완료')

    # BeautifulSoup으로 HTML 파싱
    soup = BeautifulSoup(response.text, 'html.parser')

    # 'ul' 태그 내에서 'id="search-list"'와 'class="allList c
    ul_tag = soup.find('ul', id='search-list', class_='all

    if ul_tag:
        # 'ul' 태그 내 모든 'li' 태그 찾기
        li_tags = ul_tag.find_all('li')

        # 각 'li' 태그마다 이름, 전화번호, 주소 추출
        for li in li_tags:
            # 'h4' 태그 안의 텍스트 (이름)
            name = li.find('h4', class_='name')
            name_text = name.text.strip() if name else 'N/

            # 'p' 태그 안의 텍스트 (전화번호)
            tel = li.find('p', class_='tel')
            tel_text = tel.text.strip() if tel else 'N/A'

            # 'p' 태그 안의 텍스트 (주소)
            address = li.find('p', class_='address')
            address_text = address.text.strip() if address
```

```
            # 각 항목을 엑셀 파일에 기록
            ws.append([name_text, tel_text, address_text])

            # 결과 출력 (원하는 경우 콘솔 출력도 가능)
            print(f"이름: {name_text}")
            print(f"전화번호: {tel_text}")
            print(f"주소: {address_text}")
            print("-" * 50)  # 구분선

    # 엑셀 파일 저장
    wb.save("crawling_result.xlsx")
    print("엑셀 파일로 저장 완료.")

else:
    print(f"웹 페이지를 가져오는 데 실패했습니다. 상태 코드: {respo
```

1번째 Code 실행 결과(실패)

> 웹 페이지를 가져오는 데 실패했습니다. 상태 코드: 500

> 해당 오류에 대해서 찾아본 결과 크롤링을 할 때 자동 프로그램(ex. 매크로)에 대한 제어 설정이 되어 있는 것을 확인했다. 그래서 이를 해결할 방법은 실제 사용자가 행동하는 것처럼 해당 수집할 대상 Url 요청을 해야 한다. 이는 해당 페이지에서 개발자 도구를 통해 요청 header를 보아야 한다.



> 하지만 해당 요청 header 설정을 통해 crawling 이 정상적으로 실행되는 것이 아니라 특정 header 요소의 조합을 찾아 요청을 보내주어야 성공한다는 것을 확인하였다. 또한 이는 크롤링 하고자 하는 대상 사이트 url 마다 다르다.

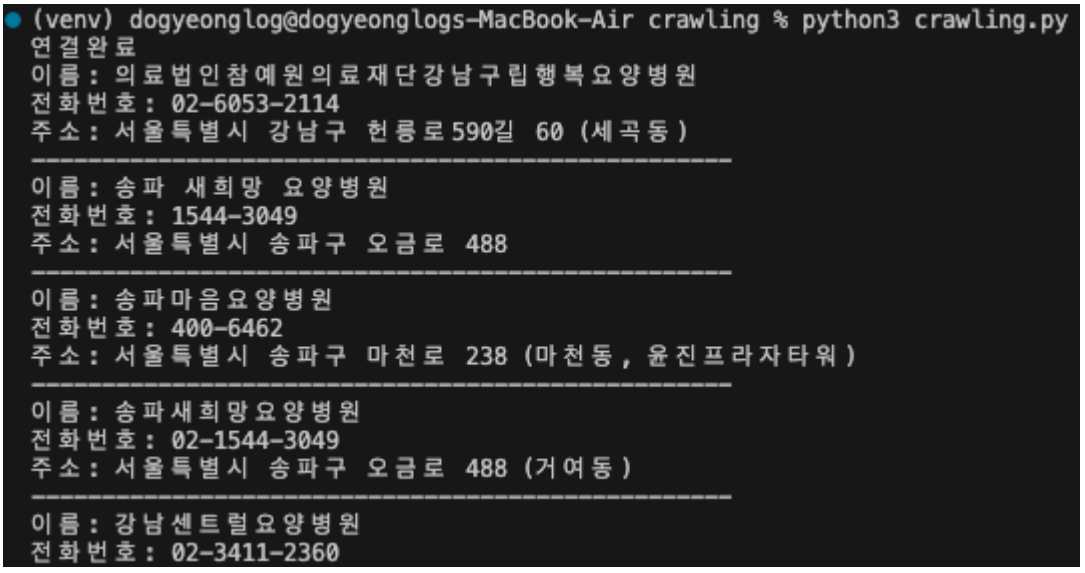
▼ 2번째 Code

```
import requests
from bs4 import BeautifulSoup
from openpyxl import Workbook

# 크롤링할 웹 페이지 URL
url = 'http://silver-dot.com/search?sch_city=1&sch_type=&q

# 요청 헤더 (쿠키 등 추가)
headers = {
    'cookie': 'ACEUCI=1; lat=35.1025; lng=129.0394; ACEFCI
}

# 엑셀 파일 생성
wb = Workbook()
```

```
ws = wb.active
ws.title = "Crawl Data"

# 엑셀 파일 첫 번째 행에 컬럼명 추가
ws.append(["이름", "전화번호", "주소"])

# 웹 페이지 요청 (GET 방식)
response = requests.get(url, headers=headers)

# 응답 상태 코드가 200(정상)일 경우
if response.status_code == 200:
    print('연결완료')

    # BeautifulSoup으로 HTML 파싱
    soup = BeautifulSoup(response.text, 'html.parser')

    # 'ul' 태그 내에서 'id="search-list"'와 'class="allList c
    ul_tag = soup.find('ul', id='search-list', class_='all

    if ul_tag:
        # 'ul' 태그 내 모든 'li' 태그 찾기
        li_tags = ul_tag.find_all('li')

        # 각 'li' 태그마다 이름, 전화번호, 주소 추출
        for li in li_tags:
            # 'h4' 태그 안의 텍스트 (이름)
            name = li.find('h4', class_='name')
            name_text = name.text.strip() if name else 'N/

            # 'p' 태그 안의 텍스트 (전화번호)
            tel = li.find('p', class_='tel')
            tel_text = tel.text.strip() if tel else 'N/A'

            # 'p' 태그 안의 텍스트 (주소)
            address = li.find('p', class_='address')
            address_text = address.text.strip() if address

            # 각 항목을 엑셀 파일에 기록
```

```
            ws.append([name_text, tel_text, address_text])

            # 결과 출력 (원하는 경우 콘솔 출력도 가능)
            print(f"이름: {name_text}")
            print(f"전화번호: {tel_text}")
            print(f"주소: {address_text}")
            print("-" * 50)  # 구분선

    # 엑셀 파일 저장
    wb.save("crawling_result.xlsx")
    print("엑셀 파일로 저장 완료.")

else:
    print(f"웹 페이지를 가져오는 데 실패했습니다. 상태 코드: {respo
```

2번째 Code 실행 결과(성공)



동적 크롤링

> '입력, 클릭, 로그인 등과 같이 페이지 이동이 있어야 보이는 데이터를 수집하는 방법'

> 정적 크롤링에 성공했으나 내가 원하는 데이터 수집의 경우 해당 페이지에 모든 정보가 한번에 뜨는 것이 아니라 더보기 버튼을 클릭해서 없어질때까지 해당 행동을 한 후 그 때 보이는 데이터들을 수집해야 하며, 다른 링크들을 이동하면서 앞의 행동을 반복하여 수집해야 했다. 이를 위해 동적 크롤링을 해야한다.

▼ 1번째 Code

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.action_chains import Action
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions
from openpyxl import Workbook
from bs4 import BeautifulSoup
import time

# Chrome WebDriver 설정
driver = webdriver.Chrome(service=Service(ChromeDriverMana

# 웹 페이지 열기
url = "http://silver-dot.com/search?sch_city=53&sch_type=&
driver.get(url)

# 5초 동안 기다리기
time.sleep(3)

# 페이지 새로 고침
driver.refresh()  # 현재 페이지를 새로 고침

# 새로 고침 후 5초 동안 기다리기 (새로 고침 후 결과를 볼 수 있게 하기
time.sleep(3)

# 엑셀 파일 생성
wb = Workbook()
```

```
ws = wb.active
ws.title = "Crawl Data"
ws.append(["이름", "전화번호", "주소"])

# 더보기 버튼 클릭 반복
while True:
    try:
        # "더보기" 버튼을 WebDriverWait으로 동적으로 기다리기
        more_button = WebDriverWait(driver, 3).until(
            EC.element_to_be_clickable((By.ID, 'list-more-
        )
        more_button.click()

        # 버튼 클릭 후 페이지 로딩 대기
        time.sleep(2)  # 버튼 클릭 후 대기 시간 (필요시 조정)
    except Exception as e:
        print("더 이상 '더보기' 버튼이 없습니다.")
        break  # 더 이상 버튼이 없으면 반복 종료

# 페이지에서 데이터 추출
soup = BeautifulSoup(driver.page_source, 'html.parser')

# 'ul' 태그 내에서 'id="search-list"'와 'class="allList clear
ul_tag = soup.find('ul', id='search-list', class_='allList

if ul_tag:
    # 'ul' 태그 내 모든 'li' 태그 찾기
    li_tags = ul_tag.find_all('li')

    # 각 'li' 태그마다 이름, 전화번호, 주소 추출
    for li in li_tags:
        # 'h4' 태그 안의 텍스트 (이름)
        name = li.find('h4', class_='name')
        name_text = name.text.strip() if name else 'N/A'

        # 'p' 태그 안의 텍스트 (전화번호)
        tel = li.find('p', class_='tel')
        tel_text = tel.text.strip() if tel else 'N/A'  # 없
```

```
            # 'p' 태그 안의 텍스트 (주소)
            address = li.find('p', class_='address')
            address_text = address.text.strip() if address els

            # 각 항목을 엑셀 파일에 기록
            ws1.append([name_text, tel_text, address_text])

            # 결과 출력 (원하는 경우 콘솔 출력도 가능)
            print(f"이름: {name_text}")
            print(f"전화번호: {tel_text}")
            print(f"주소: {address_text}")
            print("-" * 50)  # 구분선

    # 엑셀 파일 저장
    wb.save("crawling_result.xlsx")
    print("엑셀 파일로 저장 완료.")

    # 브라우저 창을 닫지 않고 유지하기 위해
    input("브라우저 창을 유지하려면 Enter를 눌러주세요...")  # 이 코드가

    # 브라우저 종료
    driver.quit()
```

1번째 Code 실행 결과(성공) & 개선된 1번째 Code 실행 결과(실패)

> 링크 이동 관련한 로직을 빼먹었고, 이를 다시 고려한 코드를 작성하였으나 코드가 유실되어 해당 링크 이동관련을 개선한 코드는 첨부를 하지 못했다. 1번째 Code 는 해당 데이터를 잘 받아왔으나, 개선된 1번째 Code 실행 결과는 도중에 브라우저가 멈추는 문제가 발생하였다.
> 해당 문제의 원인을 찾아본 결과, 더 보기 버튼을 모두 클릭하고 해당 데이터를 수집하는 과정에서 다음 행동을 지시하는 로직이 실행되어 종료된 것이었다. 또한 집에서 와이파이를 연결하여 인터넷을 사용중인데 인터넷 연결이 좋지 못해 url 이동 시 네트워크 문제도 원인이 되었다.

> 이를 해결하기 위해 긴 크롤링 시간을 줄이는 방법에 대해 고민하였고 (네트워크 환경), 그러기 위해서는 url 이동하는 범위를 적절히 조절하면서 실행을 1번으로 끝내는 것이 아닌 여러 번 나누어서 실행하였다.

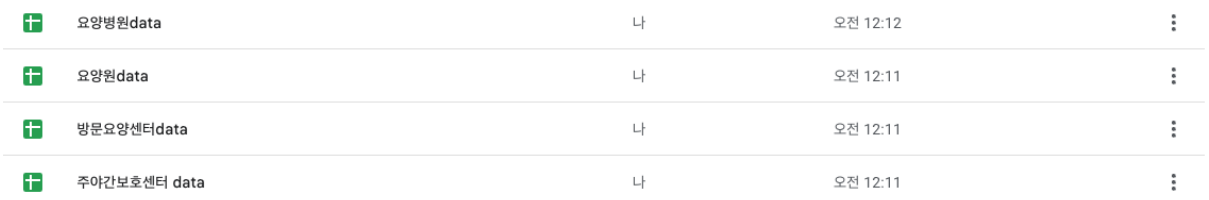
▼ 2번째 Code(url 주소가 다른 4개 파일이 존재)

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions
from openpyxl import Workbook
from bs4 import BeautifulSoup
import time

# Chrome WebDriver 설정
driver = webdriver.Chrome(service=Service(ChromeDriverMana

# 엑셀 파일 생성
wb = Workbook()
ws = wb.active
ws.title = "Crawl Data"
ws.append(["이름", "전화번호", "주소"])

# 여러 URL을 배열에 담아놓기 (예시: 서울, 부산, 경기)
urls = [
    'http://silver-dot.com/search?sch_city=1&sch_type=ND&q
    'http://silver-dot.com/search?sch_city=84&sch_type=ND&
    'http://silver-dot.com/search?sch_city=53&sch_type=ND&
    'http://silver-dot.com/search?sch_city=116&sch_type=ND
    'http://silver-dot.com/search?sch_city=27&sch_type=ND&
    'http://silver-dot.com/search?sch_city=225&sch_type=ND
    'http://silver-dot.com/search?sch_city=44&sch_type=ND&
    'http://silver-dot.com/search?sch_city=201&sch_type=ND
    'http://silver-dot.com/search?sch_city=70&sch_type=ND&
    'http://silver-dot.com/search?sch_city=147&sch_type=ND
    'http://silver-dot.com/search?sch_city=135&sch_type=ND
```

```
    'http://silver-dot.com/search?sch_city=64&sch_type=ND&
    'http://silver-dot.com/search?sch_city=178&sch_type=ND
    'http://silver-dot.com/search?sch_city=76&sch_type=ND&
    'http://silver-dot.com/search?sch_city=163&sch_type=ND
    'http://silver-dot.com/search?sch_city=244&sch_type=ND
]

# 각 URL을 반복하여 처리
for url in urls:
    # 해당 URL로 접속
    driver.get(url)

    # 페이지 로딩 대기 (동적 콘텐츠 로드 대기)
    WebDriverWait(driver, 20).until(EC.presence_of_element

    # 페이지 새로 고침
    driver.refresh()
    WebDriverWait(driver, 20).until(EC.presence_of_element

    # 더보기 버튼 클릭 반복
    while True:
        try:
            # "더보기" 버튼을 WebDriverWait으로 동적으로 기다리기
            more_button = WebDriverWait(driver, 5).until(
                EC.element_to_be_clickable((By.ID, 'list-m
            )
            more_button.click()

            # 버튼 클릭 후 페이지 로딩 대기
            time.sleep(5)  # '더보기' 클릭 후 대기시간을 길게 설
            WebDriverWait(driver, 10).until(
                EC.presence_of_element_located((By.ID, 'li
            )

        except Exception as e:
            print(f"{url} - 더 이상 '더보기' 버튼이 없습니다.")
            break  # 더 이상 버튼이 없으면 반복 종료
```

```
    # 페이지에서 데이터 추출
    soup = BeautifulSoup(driver.page_source, 'html.parser'

    # 'ul' 태그 내에서 'id="search-list"'와 'class="allList c
    ul_tag = soup.find('ul', id='search-list', class_='all

    if ul_tag:
        # 'ul' 태그 내 모든 'li' 태그 찾기
        li_tags = ul_tag.find_all('li')

        # 각 'li' 태그마다 이름, 전화번호, 주소 추출
        for li in li_tags:
            # 'h4' 태그 안의 텍스트 (이름)
            name = li.find('h4', class_='name')
            name_text = name.text.strip() if name else 'N/

            # 'p' 태그 안의 텍스트 (전화번호)
            tel = li.find('p', class_='tel')
            tel_text = tel.text.strip() if tel else 'N/A'

            # 'p' 태그 안의 텍스트 (주소)
            address = li.find('p', class_='address')
            address_text = address.text.strip() if address

            # 각 항목을 엑셀 파일에 기록
            ws.append([name_text, tel_text, address_text])

# 엑셀 파일 저장
wb.save("crawling_result.xlsx")
print("엑셀 파일로 저장 완료.")

# 브라우저 창을 닫지 않고 유지하기 위해
input("브라우저 창을 유지하려면 Enter를 눌러주세요...")

# 브라우저 종료
driver.quit()
```

2번째 Code 실행 결과(성공)



> 데이터 크롤링 시 버튼 클릭을 최소화 및 url 이동 범위를 직접 지정하여 4개의 데이터를 확보하였다. 이후 이 4개의 데이터를 하나의 파일로 합칠 예정이다.