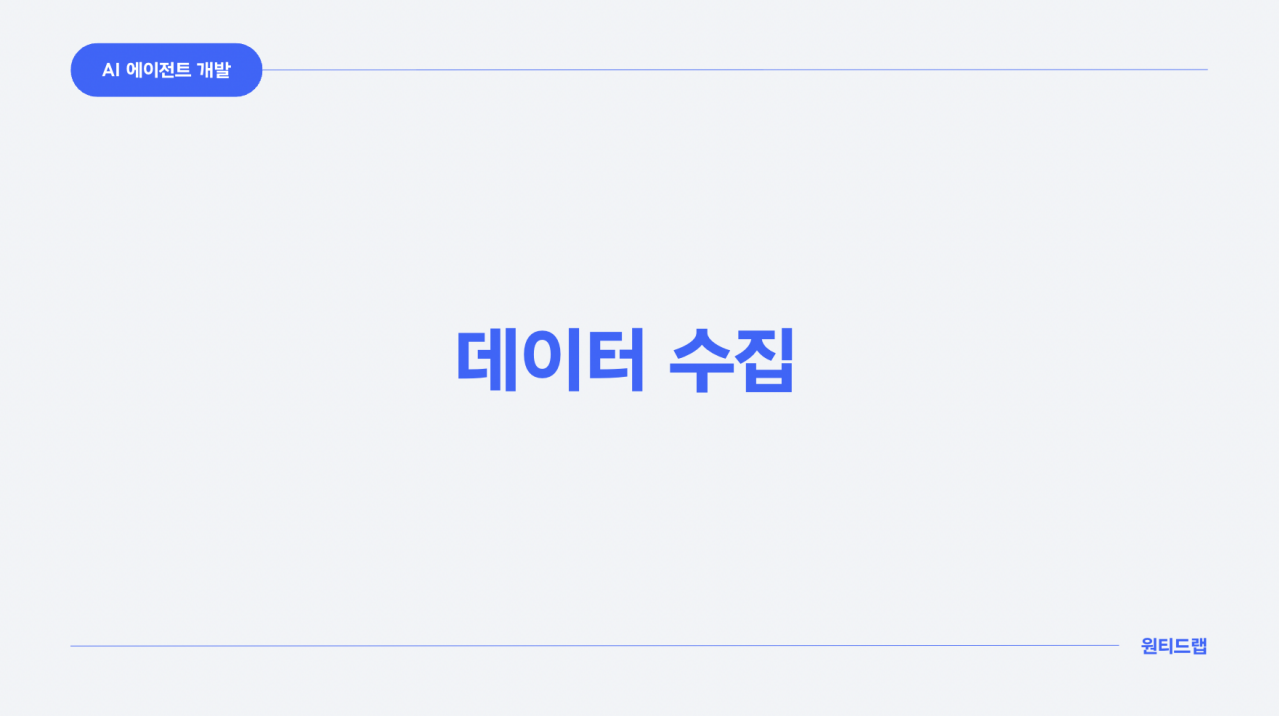
08 데이터 수집



⚠ 크롤링 시 유의할 점

- **서비스 이용 약관 확인**
  불법 크롤링은 법적 문제가 될 수 있다.
- `robots.txt` **확인**
  `https://www.example.com/robots.txt` 에 접속하면 크롤링이 가능한 범위를 알 수 있다.
- **IP 차단 방지**
  User-Agent 설정 및 `time.sleep()` 을 이용하여 짧은 시간 과도한 요청은 피해야 한다.

User-Agent란?

- 웹 요청(Request)을 보낼 때, "누가 이 요청을 보냈는지 설명하는 문자열"
- 브라우저 정보, 운영체제, 기기 정보 등을 포함
- 웹 서버는 이 정보를 기반으로 모바일/PC 버전 구분, 봇 여부 판단, 보안 및 차단 처리 등을 수행한다.
- 확인하는 방법
  크롬 기준: 웹페이지에서 F12 → 개발자 도구 → Network 탭 → 요청 1개 클릭 → Headers 탭에서 확인 가능

1. API를 이용한 데이터 수집

1) API(Application Programming Interface)

- 서버 간 데이터를 주고받을 수 있도록 정해놓은 규칙.
- JSON, XML 등의 구조화된 형식으로 응답한다.
- 요청 파라미터로 원하는 정보만 요청 가능하다.
- 대부분 API Key가 필요하다.

2) URL의 구조

https://www.example.com/search?keyword=apple&sort=related&page=2

- 도메인 주소: `https://www.example.com`
- 경로(엔드포인트): `/search`
- **쿼리 스트링(Query String)**: 데이터를 가져오는 조건
  - `keyword=apple`
  - `sort=related`
  - `page=2`

3) Requests

```
uv add requests
```

```
import requests

url = "https://www.example.com/search"
params = {"keyword": "apple", "sort": "related", "page": 2}

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

print(data)
```

ⓘ 직접 쿼리 스트링을 만들 경우 한글 인코딩에 주의해야 한다.

```
import requests
from urllib.parse import quote

keyword = "사과"
enc_keyword = quote(keyword)

url = f"https://www.example.com/serach?keyword={enc_keyword}&sort=related&page=2"
```

2. 정적 크롤링: BeautifulSoup

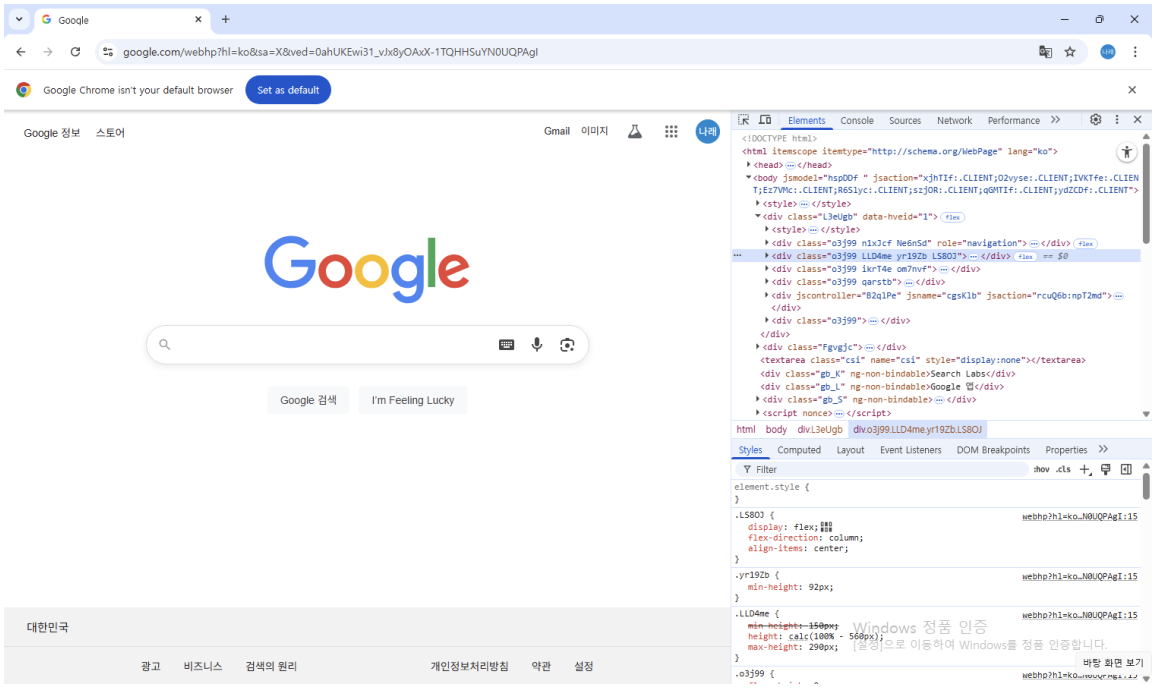
- HTML/XML 파싱에 특화된 라이브러리
- 속도가 빠르고 가벼우며, 단순한 구조의 페이지에서 효율적이다.
- 자바스크립트 실행이 불가능하다.

1) 주요 함수

함수	설명
`BeautifulSoup(html, 'html.parser')`	HTML 파싱 객체 생성
`.find(tag, attrs)`	조건에 맞는 첫 번째 태그 찾기
`.find_all(tag, attrs)`	조건에 맞는 모든 태그 리스트
`.text` / `.get_text()`	태그 안의 텍스트 추출
`.select(css_selector)`	CSS 선택자 기반 추출

2) 예시 코드

제어하고 싶은 곳에 마우스 오른쪽 버튼 → 검사(Inspect) 클릭




```
uv add beautifulsoup4
```

```
from bs4 import BeautifulSoup
import requests

url = "https://example.com"
```

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

titles = soup.find_all("h2")
for t in titles:
    print(t.get_text())
```

3. 동적 크롤링: Selenium

- 브라우저를 실제로 띄워 자바스크립트 실행이 가능하다.
- 동적으로 로딩되는 웹사이트에 적합하다.
- 코딩으로 마우스/키보드 조작이 가능하다.
- 속도가 느리고 리소스를 많이 사용한다.

1) 주요 함수

구문	설명
`webdriver.Chrome()`	크롬 브라우저 실행
`driver.get(url)`	특정 URL 접속
`find_element(By.XPATH, ...)`	요소 탐색
`element.click()`	클릭
`element.send_keys(...)`	키 입력
`driver.page_source`	전체 HTML 추출
`driver.quit()`	브라우저 종료

2) 예시 코드

```
uv add selenium
```

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

driver = webdriver.Chrome()
driver.get("https://example.com")

# 버튼 클릭
button = driver.find_element(By.CSS_SELECTOR, '#gb > div.gb_M.gb_0.gb_Pf.gb_Wf > div:nth-child(1) > a')
button.click()

time.sleep(2)

html = driver.page_source
driver.quit()
```