실전 웹 크롤링과 자동화 (For 신호용)

1권: 실전 크롤링 입문 - 원티드(Wanted) 정복하기

Chapter 1-4: 정보 추출의 기술 - `Beautiful Soup`과 `CSS Selector`

Prologue: 옥석(玉石)을 가려내는 눈

이전 챕터까지의 노력으로, `Selenium`은 우리에게 채용 공고가 모두 로드된, 정보로 가득 찬 HTML 페이지 전체를 가져다주었습니다. 하지만 이 HTML은 수많은 태그(`<div>`, `<a>`, `<span>` 등)와 속성들이 뒤섞여 있는, 아직 정제되지 않은 원석(原石)과 같습니다.

이번 챕터에서는 **`Beautiful Soup`** 이라는 강력한 도구를 사용하여, 이 복잡한 HTML 원석을 우리가 다루기 쉬운 형태로 가공하고, **`CSS Selector`** 라는 날카로운 도구로 우리가 원하는 정보(회사명, 직무, 기술 스택 등)만을 정확하게 도려내는 방법을 배웁니다.

---

1. 학습 목표 (Objectives)

- `Beautiful Soup`의 역할과 기본 사용법을 이해한다.
- `CSS Selector`를 활용하여 복잡한 HTML 구조 속에서 원하는 요소를 정밀하게 선택할 수 있다.
- 선택한 요소에서 텍스트(Text) 정보와 속성(Attribute) 값을 추출하는 코드를 작성할 수 있다.

---

2. 핵심 개념 (Core Concepts)

2.1 `Beautiful Soup`: HTML을 위한 요리사

`Beautiful Soup`는 파이썬으로 HTML과 XML 파일을 파싱(parsing)하기 위한 라이브러리입니다. '파싱'이란, 복잡하게 얽힌 코드 뭉치를 의미 있는 계층 구조(트리 구조)로 변환하여 우리가 쉽게 탐색하고 조작할 수 있도록 만드는 과정을 의미합니다.

> **[Developer's Tip]** `Selenium`이 가져온 날것의 HTML 텍스트(`driver.page_source`)를 `Beautiful Soup` 객체로 변환하면, 우리는 `.find()`, `.find_all()`, `.select()` 와 같은 편리한 메서드를 사용하여 HTML을 마치 파이썬 딕셔너리처럼 쉽게 다룰 수 있게 됩니다.

2.2 `CSS Selector`: 정보의 주소를 찾는 내비게이션

`CSS Selector`는 원래 웹페이지의 특정 요소에 스타일(색상, 크기 등)을 적용하기 위해 사용하는 문법입니다. 하지만 우리는 이 강력한 '선택' 기능을 크롤링에 활용합니다.

[반드시 알아야 할 기본 문법]

- **`.class_name`**: 특정 클래스 이름을 가진 요소를 선택 (예: `.job-title`)

- `#id_name`: 특정 ID를 가진 요소를 선택 (예: `#job-list`)
- `tag_name`: 특정 태그를 가진 요소를 선택 (예: `div`, `a`, `span`)
- `parent > child`: 부모 바로 아래의 자식 요소를 선택 (예: `div.job-card > a.job-title`)
- `ancestor descendant`: 조상 아래의 모든 후손 요소를 선택 (공백으로 구분) (예: `#job-list .job-title`)
- `[attribute='value']`: 특정 속성과 값을 가진 요소를 선택 (예: `a[href^='/wd/']` -> `href`가 `/wd/`로 시작하는 `a` 태그)

이 문법들을 조합하면, 우리는 페이지의 그 어떤 요소라도 '저격'할 수 있게 됩니다.

---

3. 기초 실습 (Basic Practice)

이제 `main.py`에 `Beautiful Soup`을 추가하여, 스크롤이 완료된 페이지에서 채용 공고 목록을 가져오는 실습을 하겠습니다.

[환경 설정] 먼저, `Beautiful Soup`과 HTML 파서인 `lxml`을 설치합니다.

````
pip install beautifulsoup4 lxml```

**[코드 수정]**
`main.py`의 뒷부분을 아래와 같이 수정합니다.

```python
# main.py
... (이전 스크롤 코드와 동일) ...

print("페이지 스크롤이 완료되었습니다.")
# time.sleep(5) -> 이제 불필요하므로 주석 처리하거나 삭제합니다.

# 6. HTML 파싱하여 정보 추출하기
# beautifulsoup4 라이브러리를 임포트합니다.
from bs4 import BeautifulSoup

# driver.page_source는 현재 Selenium이 보고 있는 페이지의 전체 HTML 소스를 가져옵니다.
html = driver.page_source
# BeautifulSoup 객체를 생성합니다. 첫 번째 인자는 HTML 소스, 두 번째 인자는 사용할 파서입니다.
# 'lxml'은 매우 빠른 속도를 자랑하는 인기 있는 파서입니다.
soup = BeautifulSoup(html, "lxml")

# 이제 'soup' 객체를 사용하여 정보를 찾습니다.
# 원티드 채용 공고 하나하나를 감싸고 있는 카드 형태의 요소를 찾아야 합니다.
# 개발자 도구로 확인해보면, 각 공고는 data-cy="job-card" 속성을 가진 div 요소입니다.
# CSS Selector로 이 요소들을 모두(select) 찾습니다.
job_cards = soup.select("div[data-cy='job-card']")

# 찾은 공고의 개수를 출력해봅니다.
print(f"총 {len(job_cards)}개의 채용 공고를 찾았습니다.")

# 7. 브라우저 닫기
driver.quit()
````
````

```
print("Selenium 실행이 완료되었습니다.")
```

- **주의:** `from bs4 import BeautifulSoup`은 파이썬 파일의 맨 위로 옮겨서 다른 `import` 문들과 함께 관리하는 것이 좋습니다.

[Mission 4] 첫 데이터 추출

신호용 님의 네 번째 미션입니다.

> 1. `beautifulsoup4`와 `lxml`을 설치하십시오.
> 2. `main.py`를 위 코드와 같이 수정하고, 터미널에서 `python main.py`를 실행하십시오.
> 3. **예상 결과:** 이전과 동일하게 브라우저가 자동으로 동작하여 스크롤까지 완료한 뒤, 터미널에 **"총 N개의 채용 공고를 찾았습니다."** 와 같이, 현재 페이지에 로드된 채용 공고의 총개수가 정확하게 출력되는지 확인하십시오.