```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from bs4 import BeautifulSoup
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
import time
from datetime import datetime, timedelta, date
import json
import requests
```

필요한 라이브러리를 설치합니다.

```python
class HanKyungScraper:
    def __init__(self, start_date: str, end_date: str, output: str = ""):
        self.start_date = start_date
        self.end_date = end_date
        self.output = output
        self.driver = self._init_driver()
        self.results: list[dict[str, str]] = []

    def _init_driver(self) -> webdriver.Chrome:
        chrome_options = Options()
        chrome_options.add_experimental_option("detach", True)
        chrome_options.add_experimental_option("excludeSwitches", ["enable-logging"])
        service = Service()
        driver = webdriver.Chrome(service=service, options=chrome_options)
        return driver
```

생성 시 변수와 드라이버 설정을 해줍니다.

```
def _load_more_pages(self, previous_date_str: str) -> None:
    while True:
        try:
            more_btn = self.driver.find_elements(By.CSS_SELECTOR, ".btn-more")
            for btn in more_btn:
                if btn.text == "더보기":
                    btn.click()
                    time.sleep(2)

            date_elements = self.driver.find_elements(By.CSS_SELECTOR, ".txt-date")
            for date_element in date_elements:
                if date_element.text != "":
                    formatted_date = datetime.strptime(date_element.text, "%Y.%m.%d").strftime("
                    if formatted_date == previous_date_str:
                        return

        except Exception as e:
            print(f"에러가 발생했습니다: {str(e)}\n계속 진행합니다.")
```

더보기 버튼을 누르는 함수를 만듭니다. 일단 start_date 까지 모든 뉴스를 로드하는걸 목표로 합니다. whlie 문 중에 오류가 생겨도 계속 진행하는 구문 처리도 했습니다.

```
def _scrape_articles(self) -> list[dict[str, str]]:
    elements = self.driver.find_elements(By.CSS_SELECTOR, ".news-tit a")
    new_articles: list[dict[str, str]] = []
    titles_set: set[str] = set(article['title'] for article in self.results)

    if elements:
        for element in elements:
            if element.text != "" and element.text not in titles_set:
                href = element.get_attribute("href") or ""  # None일 경우 빈 문자열로 처리
                title = element.text
                new_articles.append({
                    "title": title,
                    "href": href
                })
                titles_set.add(title)
    return new_articles
```

화면에 그려진 뉴스들을 긁어 옵니다. .news-tit 밑에 a 태그에 뉴스 제목과 링크를 수집할 수 있습니다. 클라이언트사이드 렌더링이 있는지, 다른 영역의 기사들도 스크랩될거 같아서 text 가 있는 제목만 가져와서 리스트에 저장합니다.

```python
def _process_article(self, article: dict[str, str], previous_date_str: str, end_date_obj: date) -> bool:
    try:
        response = requests.get(article["href"])
        response.raise_for_status()
        html = response.text
        soup = BeautifulSoup(html, 'html.parser')

        date_list = soup.select(".txt-date")
        if len(date_list) < 2:
            raise ValueError("날짜 정보가 충분하지 않습니다.\n계속 진행합니다.")

        date = date_list[0].text
        formatted_date = datetime.strptime(date, "%Y.%m.%d %H:%M").strftime("%Y%m%d")
        formatted_date2 = datetime.strptime(date, "%Y.%m.%d %H:%M").date()

        if formatted_date2 > end_date_obj:
            print(f"{formatted_date2}가 {end_date_obj}를 초과합니다.")
            return True

        date_edit = date_list[1].text if len(date_list) > 1 else ""

        content = soup.select("#articletxt")
        if not content:
            raise ValueError("기사를 찾을 수 없습니다.\n계속 진행합니다.")

        article_text = content[0].get_text(separator="\n", strip=True)
        href = article["href"]
        title = article["title"]

        if previous_date_str == formatted_date:
            return False

        self.results.append({
            "date": date,      You, 16분 전 • Uncommitted changes
            "date_edit": date_edit,
            "href": href,
            "title": title,
            "content": article_text
        })
        print(f"{date} 기사까지 스크랩 완료!")

        return True

    except Exception as e:
        print(f"에러가 발생했습니다. URL: {article['href']}, 에러: {str(e)}\n계속 진행합니다.")
        return True
```

수집 해야할 기사의 제목과 링크를 러프하게 수집했으므로 수집된 링크 모두 셀레니움으로 들어가면 너무 리소스 낭비니까 뷰티풀 수프로 데이터를 받아옵니다. date 를 바탕으로 입력 날짜를 초과하진 않는지, 시작날짜 이전이진 않은지에 따라서 크롤링 여부를 결정합니다. 에러가 발생해도 데이터를 수집할 수 있도록 try/ except 구문을 넣어줍니다.
이 메서드가 잘 돌아가면 results 리스트에 딕셔너리 형태로 데이터가 잘 수집되었을겁니다. 중간에 멈춘다고 하더라도 중간까지는 잘 수집되어있을겁니다. 이제 json으로 바꾸기만 하면 됩니다.

```python
def scrape(self) -> None:
    self.driver.get("https://www.hankyung.com/all-news")
    self.driver.implicitly_wait(5)
    self.driver.maximize_window()

    elements = self.driver.find_elements(By.CSS_SELECTOR, ".nav-link")
    economi = elements[14]
    economi.click()

    time.sleep(5)  # 페이지 로드 대기

    start_date_obj = datetime.strptime(self.start_date, "%Y%m%d")
    end_date_obj = datetime.strptime(self.end_date, "%Y%m%d").date()
    previous_date_obj = start_date_obj - timedelta(days=1)
    previous_date_str = previous_date_obj.strftime("%Y%m%d")

    try:
        self._load_more_pages(previous_date_str)
        new_articles = self._scrape_articles()

        for article in new_articles:
            if not self._process_article(article, previous_date_str, end_date_obj):
                break

    except Exception as e:
        print(f"에러가 발생했습니다: {str(e)}\n계속 진행합니다.")
    finally:
        self._save_results()
```

크롤링할 페이지로 가서 경제탭을 눌러줍니다.
더보기를 누르는 메서드를 시행합니다. 새롭게 수집된 기사들에 한해서 기사를 리스트에 넣는 작업을 수행합니다. 에러가 나도 진행합니다.
정상 작동하든 에러가 나든 마지막에 저장하는 메서드를 한번 실행해줍니다.
지금까지 만든 메서드는 이 스크랩 메서드를 위해 만든거였습니다.

```python
def _save_results(self) -> None:
    output_path = self.output if self.output else "result.json"
    with open(output_path, "w", encoding="utf-8") as f:
        json.dump(self.results, f, ensure_ascii=False, indent=4)
    print("기사 데이터를 JSON 파일로 저장했습니다.")

def __del__(self):
    try:
        self.driver.quit()
    except Exception as e:
        print(f"에러가 발생했습니다: {str(e)}")      You, 24분 전 • Uncommitted chan
```

저장과 드라이버 끄는 메서드입니다.

이제 메인에서 args 받고 크롤링 하면 됩니다.