14. 웹 데이터 수집 #src: 8_Python/ch14_웹데이터수집 참고

1) Beautiful Soup과 parser

(1) BeautifulSoup; 스크린 스크래핑 프로젝트를 위해 설계된 파이썬 라이브러리

(2) parser; BeautifulSoup(markup, "html.parser") #파이썬 2.7.3, 3.2.2 이상 버전에서 호환됨

(3) Selector API; 가장 일반적으로 사용되는 CSS선택자를 지원: select(), select_one(), find() 등

① soup.select("css 선택자"); css 선택자에 해당하는 모든 요소를 반환

② soup.select_one("css 선택자"); css 선택자에 해당하는 첫번째 태그 요소만 반환

※요소 안의 텍스트는 text(or string), 태그 이름은 name, 태그 속성들은 attrs를 이용해서 조회

(4) CSS(Cascading Style Sheet) 선택자; HTML문서의 태그이름, class 속성, id 속성 등을 이용

① 내포(후손) 선택자( ): "div b" ② 자식 선택자(>): "div > b"

③ 클래스 선택자(.): ".class" ④ 아이디 선택자(#): "#id"

⑤ 속성 선택자([=]): "[name=value]"

(5) 상태코드; HTTP응답 메시지의 상태라인에는 상태코드와 상태 메시지를 포함하고 있음

① 200번 영역: 성공 (200: 성공 / 201: POST 요청 처리 / 204: 전송할 데이터 없음)

② 400번 영역: 오류 (401: 사용자 인증 / 403: 접근권한無 / 404: URL없음 / 406: Not acceptable)

#100번 영역: 정보 전송 #300번 영역: 리다이렉션 #500번 영역: 서버측 오류

2) requests를 이용한 웹 데이터 수집(정적 웹크롤링)

(1) requests 모듈; ① 요청 → ② 처리 → ③ 응답; HTML, XML등으로 반환 → ④ 렌더링

(2) GET 요청; HTTP 요청(Request)방식 중 하나

response = requests.get('URL'); 지정된 리소스에서 데이터 가져옴 #성공시<Response [200]>

response.content; 응답 내용을 바이트 단위로 볼 때 사용 #response.encoding = 'utf-8'

※ 크롤링 막혀있을 때: html = requests.get(url, headers={'User-agent':'Mozilla/5.0'}) 시도

3) Selenium을 이용한 웹데이터 수집(동적 웹크롤링)

(1) Selenium; 브라우저의 동작을 자동화 해주는 프로그램

- 셀레니움 파이썬 파인딩은 셀레니움 WebDriver를 사용하여 인터넷 익스플로러, Chrome, Firefox 등 브라우저에 접근하고 브라우저의 동작을 제어할 수 있는 API를 제공

```
from selenium import webdriver
driver = webdriver.Chrome("chromedriver가 설치된 절대 경로") #웹드라이버 실행
driver.get("URL") #사이트 접속
#입력양식 채우기 및 이벤트 처리 ex) element.send_keys("입력할 내용")
driver.close() #브라우저 종료
```