SEEDS SEASON2

Seeds
FIRST STEP
-1 week-

Seeds운영진 **김민성**

INDEX

01 웹 스크래핑과 크롤링 차이

- **웹 크롤링 : 정보를 색인화하고 저장하는데 사용.**
  - **->일반적으로 검색 엔진 및 기타 자동화 도구에 의해 수행된다.**

- **웹 스크래핑: 분석 및 기타 목적을 위해 웹사이트에서 데이터를 추출 하는데 사용.**
  - **->종종 사람 또는 어떤 목적을 위해 특별히 설계된 자동화 도구에 의해 수행된다.**

크롤링	스크래핑
웹에서 페이지 및 링크 다운로드 (웹을 기반으로 작동)	웹을 포함한 다양한 소스에서 데이터 추출 (반드시 웹과 관련된 것은 아님)
동일한 콘텐츠가 여러 페이지에 업로드 된 것을 인식하지 못하므로 중복 제거는 필수적	특정 데이터를 추출하는 것이므로 중복 제거가 반드시 필요한 것은 아님

자료 출처 : Data Scraping vs Data Crawling

02 파이썬 크롤링 라이브러리

BeautifulSoup : Html , XML 파일의 정보를 추출해는 python 라이브러리.

Scrapy: 미들웨어 , 파이프 라인 , js render , proxy , xpath , CLI 등 다양한 기능들과 플러그인들을 사용가능.

Selenium : 웹 자동화 테스트에 사용되는 프레임워크.

03 각 라이브러리 사용시점

- **BeautifulSoup: HTML에서 데이터를 추출해야하는 소규모 또는 중간 규모의 스크래핑 작업에 이상적.**
  - -> 장점: 쉽고 심플하고 빠름
  - -> 단점: js로 렌더링이 필요한 사이트들을 크롤링하기 어렵다. 병렬처리 로직을 별도로 작성하지 않으면 느린편
- **Scrapy : 대규모이고 복잡한 웹 스크래핑 작업에 적합함.**
  - -> 장점: 다양한 플러그인 보유, 훌륭한 커뮤니티와 문서화, 크롤링에 필요한 다양한 기능
  - -> 단점:플러그인들끼리 서로 호환이 안되는 경우가 있을 수 있음
- **Selenium : 웹사이트가 콘텐츠를 비동기적으로 로드하거나 사용자 상호작용에 응답하는 경우 적합함.**
  - -> 장점: 사용자가 보는 웹페이지의 모든 정보를 가져올 수 있음 , js rendering기능 지원, 사용방법이 직관적이고 쉬움.
  - -> 단점: 느리고 메모리를 많이 차지 한다.

04 파이썬 크롤링 라이브러리 차이

	속도	리소스	문서화	커뮤니티	기능	난이도
reqeust,urlib	빠름(조건부)	가벼움	보통	보통	적음	쉬움
selenium	느림	무거움	좋음	좋음	보통	쉬움
scrapy	빠름	가벼움	좋음	좋음	많음	보통

05 bs4 사용법

Vsc

1. VSC 실행후 터미널 창에 -> pip install bs4
   -> pip install BeautifulSoup4

2. from bs4 import BeautifulSoup 모듈 임포트

파이참

1. File -> Settings -> Project Interpreter -> + -> beautifulSoup4 검색 , 선택 -> install Package 클릭

5 bs4 사용법

라이브러리 구분	함수	기능
request	response = requests.get({url})	원하는 웹사이트(url)에 접속 보통 response 변수에 정의함
	response.status_code	응답 코드(Response) 반환
	response.text	HTML 문서 리턴
	response.raise_for_status()	200번대 응답코드가 아니면 예외를 발생시킴
BeautifulSoup4	soup = BeautifulSoup(html, 'html.parser')	응답받은 HTML 파싱 (보통 soup 변수에 정의함)
	soup.prettify()	HTML 구조 파악
	soup.find(name, attrs , recursiv , string **kwargs)	해당 조건에 맞는 하나의 태그 가져옴 (중복이면 가장 첫번째 태그) ->단일노드 반환(1개)
	soup.find_all(name, attrs , recursive , string, limit , **kwargs)	해당 조건에 맞는 모든 태그 가져옴 ->list 반환(복수개)
	soup.find('태그' , attrs= {id (class) : '속성'})	태그의 속성을 찾는 방법
	soup.get_text(), soup.text	텍스트만 추출
	soup.select_one , soup.select	copy selector을 사용하여 찾는 함수

5 bs4 사용법

select() 와 find() 차이

- find와 select는 BeautifulSoup의 메소드로서 데이터 구조를 항해하는 몇 가지 방법이다.
- find() 목적은 원하는 태그를 찾는 것이다. 태그는 이름(name) , 속성 (attribute) , 속성값(value)로 구성된다.

ex)

```
tag =   "<p class='youngone' id='junu'> Hello World! </p>"
soup = BeautifulSoup(tag) # 태그 이름만 특정
soup.find('p') # 태그 속성만 특정
soup.find(class_='youngone')
soup.find(attrs = {'class':'youngone'})# 태그 이름과 속성 모두 특정
soup.find('p', class_='youngone') #결과: <p class="youngone" id="junu"> Hello World! </p>
```

5 bs4 사용법

ex 2)

```
tag = "<p class='youngone' id='junu'> Hello World! </p>"
soup = BeautifulSoup(tag)
object_tag = soup.find('p')

#태그의 이름
object_tag.name
#결과: 'p'

#태그에 담긴 텍스트
object_tag.text
#결과: ' Hello World! '

#태그의 속성과 속성값
object_tag.attrs
#결과: {'class': ['youngone'], 'id': 'junu'}
```

5 bs4 사용법

select() 와 find() 차이

- select()는 CSS selector로 tag 객체를 찾아 반환한다. 이는 CSS에서 HTML을 태깅하는 방법을 활용한 메소드다. 가장 첫 번째 결과를 반환하는 select_one()은 find()에, 전체 결과를 리스트로 반환하는 select()는 find_all()에 대응한다.
- find처럼 태그 이름, 속성, 속성값을 특정하는 방식은 같다. 하지만 CSS는 이 외에도 다향한 선택자(selector)를 갖기 때문에 여러 요소를 조합하여 태그를 특정하기가 쉽다.

5 bs4 사용법

select() 와 find() 차이

- 예를 들어 특정 경로의 태그를 객체로 반환하고 싶을 때, find의 경우 반복적으로 코드를 작성해야 한다. select는 직접 하위 경로를 지정할 수 있기 때문에 간편하다.
- 결론적으로 select를 사용하는게 더 다양하고 직관적인 방법으로 태그를 쫓을 수 있기 때문에 select() 사용을 권장하고 있다. 또한 select()가 수행시간도 더 빠르며 메모리 소모량도 더 적다.

ex)

```
#find
soup.find('div').find('p')
#select
soup.select_one('div > p')
```

5 bs4 사용법

ex) 네이버 지식인 크롤링 예제

```
import requests
from bs4 import BeautifulSoup

url = 'https://kin.naver.com/search/list.nhn?query=%ED%8C%8C%EC%9D%B4%EC%8D%AC'
response = requests.get(url)
if response.status_code == 200:  #html 상태가 200일때, html을 soup객체로 반환
    html = response.text
    soup = BeautifulSoup(html, 'html.parser')
    print(soup)
else :
    print(response.status_code)
```

5 bs4 사용법

ex) 네이버 지식인 크롤링 예제

```
import requests
from bs4 import BeautifulSoup

url = 'https://kin.naver.com/search/list.nhn?query=%ED%8C%8C%EC%9D%B4%EC%8D%AC'
response = requests.get(url)
if response.status_code == 200:  #html 상태가 200일때, html을 soup객체로 반환
    html = response.text
    soup = BeautifulSoup(html, 'html.parser')
    title = soup.select_one('#s_content > div.section > ul > li:nth-child(1) > dl > dt > a')
      #copy selector를 사용하여 html 태그의 요소를 찾음
    print(title)
    print(title.get_text()) #get_text()함수로 텍스트만 뽑아올수 있음.
else :
    print(response.status_code)
```

5 bs4 사용법

ex) 네이버 지식인 크롤링 예제 2

```
import requests
from bs4 import BeautifulSoup

url = 'https://kin.naver.com/search/list.nhn?query=%ED%8C%8C%EC%9D%B4%EC%8D%AC'
response = requests.get(url)
if response.status_code == 200:
    html = response.text
    soup = BeautifulSoup(html, 'html.parser')
    ul = soup. select_one('ul.basic1') # ul 태그 내용 출력
    titles= ul.select('li > dl > dt > a')  # select는 찾은 모든 html을 리스트 형태로 반환
    for title in titles:
        print(title.get_text())
else :
    print(response.status_code)
```

5 bs4 사용법

ex)네이버 지식인 크롤링 예제 2

```
import requests
from bs4 import BeautifulSoup

url = 'https://kin.naver.com/search/list.nhn?query=%ED%8C%8C%EC%9D%B4%EC%8D%AC'
response = requests.get(url)
if response.status_code == 200:
    html = response.text
    soup = BeautifulSoup(html, 'html.parser')
    ul = soup. select_one('ul.basic1') # select_one 함수는 문서의 처음부터 시작하여 조건에 맞는
                                         하나를 찾게 함.
    titles= ul.select('li > dl > dt > a')  # li > dl > dt > a  : html페이지에 있는 태그 순서대로 정보를
                                         찾아오겠다는 뜻
    for title in titles:
        print(title.get_text())
else :
    print(response.status_code)
```

과제

본인이 원하는 내용의 정보나 데이터를 추출해오세요.