0.Web Scraping

[Web scraping & web crolling]

web scraping = 웹 페이지에서 원하는 부분만 scraping

web crolling = 웹 페이지의 허용범위 내, link들을 따라가며 모든 내용들을 가져오기

[web]

HTML + CSS + JS

집 = WEB

- outline (structure) : HTML - 뼈대 - extension "open in browser"
- Interior(allignment) : CSS - 예쁘게
- Actual Implement : JAVA SCRIPT - 살아있게

*현재 시점의 web page에 대한 정보가 다르기 때문에 ,강의 내용 그대로 따라가기 보다는 현재 시점을 고려하면서 공부할 것.

##

[웹 스크래핑 변경사항]

1. "티스토리"는 UserAgent 를 변경하지 않아도 정상적으로 html 을 받아옵니다.
2. "네이버"는 로그인 시도 시 자동입력방지 문자 입력 페이지가 뜹니다. 우회방법으로 자바스크립트를 이용하는 방법이 소개된 링크를 참고해주세요.

https://jaeseokim.github.io/Python/python-Selenium을-이용한-웹-크롤링-Naver-login-후-구독-Feed-크롤링/

저장이 완료되었습니다.

항목이 웹에서 접근했을 때와는 조금 다르게 가져오는듯 합
정상, 20%는 페이지에 존재하지 않는 값을 가져옵니다.
(어쩌면 다음 페이지에 나오는 내용일 수도 있습니다) 또한 80% 의 항목도 웹 페이지와는 달리 순서가 조금 뒤죽박죽 섞인듯 보입니다. requests 만 써서 가져왔을 때 쿠팡에서 반환해주는 값에 차이가 있는듯한데, selenium 을 통한 결과를 비교해볼 필요가 있어 보이네요. 수업시간에 결과 내용에 대해 전수 검사를 해볼 생각을 미처 해보지 못하여 내용에 오류가 있었던 점, 진심으로 사과 드립니다.

4. "프로젝트" 강의 내용 중 네이버 뉴스를 가져올 때 500 Server Error 가 나고 있습니다. 이 때는 requests 에 headers 로 여러분 PC 의 user-agent 를 넣어주시면 됩니다.

(예시)

```
def create_soup(url):
    headers = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, l

    res = requests.get(url, headers=headers)
    res.raise_for_status()
```

```
    soup = BeautifulSoup(res.text, "lxml")
    return soup
```

1.HTML

Hyper Text Markup Language

```
<html/> #element   - 한문장으로 쓰기


<html></html> #여는 tag + # 닫는 tag



 <html>

    <head></head> head #: 홈페이지 제목, 선행 작업들

    <body></body> body #: 웹 페이지의 본문

</html>


<html>
    <head>
        <meta charset = "utf-8">
        <title>나도코딩 홈페이지</title>
    </head>
    <body>
        <input type = "text" placeholder = "아이디를 입력하세요">
        <!-- type= value= attribute 라고 하며, element의 세부 속성을 의미-->
        <input type = "password">
        <input type = "button" value = "로그인">
        <a href = "http://google.com">구글로 이동하기</a> <!--링크 타고 페이지 넘어가기-->

    </body>


</html>

<!--w3schools.com 에 들어서 HTML에 대해 배우기-->
```

저장이 완료되었습니다.

2.xpath

Xpath : HTML 속에 필요한 성분 element tag에 들어있는 내용을 가져오기 위해서, 해당 element의 경로

*비슷한 element의 비슷한 속성, 값이 있을 때, 경로를 명확하게 지정해주기 위해 사용

```
parent element / child element / child element / chid element

parent : 상위 element

child : 하위 element

sibling : 같은 계층에서 동등한 관계의 element
```

---------example]

```
/학교 /학년/반/학생[2]
//*[@학번 = "1-1-5"] : 엄청나게 긴 xpath를 한 특정한 element의 unique한 값을 이용.

//*[@id="u_skip"]  : unique 한 값 value를 이용해서 가져온다. (copy xpath)

/html/body/div/span/a...
* 각층의 element의 이름을 특정지으면서 가져온다.(copy full xpath)

#########################
          / : 현재 위치에서 한단계 아래로 ....
          //: 현재 위치의 하위 element에 대해서 모두 검색 ( * : 이름 상관 없이 모두)

########################


<학교 이름 = "나도고등학교">
    <학년 value = "1학년">
        <반 value = "1반">
            <학생 value = "1번" 학번 = "1-1-1"> 이지은 </학생>
                                  "1-1-2"> 유재석 </학생>
                                  "1-1-3"> 조세호 </학생>


        </반>
        <반 value = "2반" /반>


    </학년>

    <학년 value = "2학년"/> ... 3반 유재석<...>
    <학년 value = "3학년"/>
</학교>
```

저장이 완료되었습니다.

3.requests

```
# pip install requests  : module 설치
```

```
# Requests library: web scrapping  = web page의 문서 정보 가져오기

import requests
res = requests.get("http://goolge.com")

# 실제로 가져올수 있는지 여부를 확인
# print("응답 코드 : ", res.status_code) # 200이면 정상

# res1 = requests.get("http://nadocoding.tistory.com")

# print("응답코드 : ", res1.status_code) ## 403 이면 비정상 (권한이 없음.)


## 정상여부 확인 작업
# if res.status_code == requests.codes.ok :
#     print("정상입니다.")

# else:
#     print("문제가 생겼습니다. [에러코드", res.status_code,"]")

## 정상 여부 확인 작업(2)
res.raise_for_status()  ## 비정상(403)이면 error출력, 프로그램 종료
# print("웹 스크래핑을 진행합니다.")



# ##따라서 : 기본 문장들....
# import requests
# res = requests.get("http://naver.com")
# res.raise_for_status()

print(len(res.text)) # html의 문자 개수
print(res.text)

#html내용 파일로 저장 .html형식으로
                                                    g = "utf8") as f:
```

저장이 완료되었습니다.

4.regular expression

regular expression : 해당 내용에 대한 정해진 형식(specification)

ex) 주민번호 : 6자리 - 7자리 / 이메일 : email@naver.com

교통사고 목격 , 뺑소니, 차번호판 ca?e

cafe, case, cave, care.....

caae, cabe, cace,cade.....

1. p= re.compile("원하는 정규식,형태")
2. m = p.match("비교할 문자열") : 처음부터 일치하는지 일치 확인
3. m = p.search("비교할 문자열") : 문자열 중에 일치하는게 있는지 확인

4. lst = p.findall("비교할 문자열") : 일치하는 모든 것을 "리스트"형태로 반환

원하는 형태 : 정규식

(ca.e) . : 하나의 문자를 의미 > care, cafe, case (o) | caffe (x)

(^de) ^ : 문자열의 시작 > desk, destination(o) | fade(x)

(se$) : 문자열의 끝 > case , base | face(x)

정규식 공부 : w3schools.com > learn python > RegEX

or python re(docs.python.org - 공식 홈페이지)

```
# regular expression : 해당 내용에 대한 정해진 형식(specification)

#ex) 주민번호 : 6자리 - 7자리  / 이메일 : email@naver.com

import re

# 교통사고 목격 , 뺑소니, 차번호판
#ca?e
#cafe, case, cave, care.........
#caae, cabe, cace,cade.......
#regular expression

p = re.compile("ca.e") # pattern 지정
#(ca.e) . : 하나의 문자를 의미 > care, cafe, case (o) | caffe (x)
#(^de) ^ : 문자열의 시작 > desk, destination(o) | fade(x)
# (se$) $ : 문자열의 끝 > case , base | face(x)


def print_match(m):
    if m :
        print("m.group() : ", m.group())

                                  일치한 문자열만 ex) ca.e - care 매칭 가능 : care값 저장
                                  earch("good care")에서 매칭된 내용 없음) error 발생

        print("m.string : ",m.string) # 일치한다면, 입력받은 문자열 전체( string = 함수x, 변수임.)
        print("m.start(): ", m.start()) # 일치하는 입력받은 문자열의 시작 index
        print("m.end() : ", m.end()) # 일치하는 입력받은 문자열의 끝 index
        print("m.span():", m.span()) # 일치하는 입력받은 문자열의 시작  /  끝 index

    else:
        print("매칭되지 않았습니다.") # 여부에 따라 error되지 않고, 계속 진행.



# m = p.match("good care") # 받은 input값과 p가 matching하는지 여부 정보(type: re.Match)
# # 경우들
# #"good care" : 앞 부분에 추가적으로 받기 때문에 match x
# #"careless" : ca.e(less) match 가능 / match : 주어진 문자열의 처음부터 일치하는지 확인.


# print_match(m)
```

저장이 완료되었습니다.

```
# m = p.search("good care") # search : 주어진 문자열 중에 일치하는게 있는지 확인

# print_match(m) # output = care

#[경우] : good care , careless
#group() 결과
#"good care" : ca.e 과 일치하는 단어가 있으므로 care 출력
#"careless" : 역시 ca.e(기준)과 매칭되는 care가 문자열 중에 있으므로 care grouping

#m.string 결과
#"good care" : good care

#m.start()  : 5
#m.end(): 9
#m.span() : (5,9)



lst = p.findall("careless cafe")  # findall : 일치하는 모든 것을 리스트 형태로 반환 ( 즉 여러개가 일
print(lst)#['care', 'cafe']



##########################################

#1. p= re.compile("원하는 정규식,형태")
#2. m = p.match("비교할 문자열") : 처음부터 일치하는지 일치 확인
#3. m = p.search("비교할 문자열") : 문자열 중에 일치하는게 있는지 확인
#4. lst = p.findall("비교할 문자열") : 일치하는 모든 것을 "리스트"형태로 반환

# 원하는 형태 : 정규식
##(ca.e) . : 하나의 문자를 의미 > care, cafe, case (o) | caffe (x)
#(^de) ^ : 문자열의 시작 > desk, destination(o) | fade(x)
# (se$) $ : 문자열의 끝 > case , base | face(x)
```

저장이 완료되었습니다.

ython > RegEX or python re(docs.python.org - 공식 홈페이지)

5.user agent

User Agent

해당 웹사이트를 접속하는 사용자(사람, mobile)에 따라 맞는 웹 사이트를 제공한다.

but 프로그램이 접속하는 경우, 웹사이트 자체가 차단할 수 있음.

이때, User agent를 통해서 처리 할 수 있음.

[웹 스크래핑 변경사항]

1. "티스토리"는 UserAgent 를 변경하지 않아도 정상적으로 html 을 받아옵니다.

My user agent (현 시점 나의 user agent)

: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93 Safari/537.36

```
import requests

url = "http://nadocoding.tistory.com"
headers ={"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G

res = requests.get(url, headers = headers) ## requests를 할 때, 접속하는 해당 user agent값을 넘겨줌
res.raise_for_status()

with open("nadocoding.html", "w", encoding = "utf8") as f:
    f.write(res.text)
```