

크롤링 6번째 세션

NEXT X LIKELION

박지환

| 크롤링이란?

(웹) 크롤링(crawling)

웹 사이트의 데이터를 추출하는 것

- python을 많이 사용합니다.

크롤링이란?



[https://news.naver.com/main/read.nhn?
mode=LSD&mid=shm&sid1=101&oid=055&aid=0000806607](https://news.naver.com/main/read.nhn?mode=LSD&mid=shm&sid1=101&oid=055&aid=0000806607)



NEWS.NAVER.COM

광고 출연료도 선뜻 기부...연이은 나눔 행렬

<앵커> 모두가 감염병으로 어려운 이때, 이웃을 위한 따뜻한 나눔이 이어지...

url만 입력했는데 페북이 알아서
제목, 이미지를 가져와서 preview를 보여준다.

-> 크롤링



NEXT X LIKELION

| 가상환경 구축

Window분들 powershell 사용할게요!

검색 > powershell

| 가상환경 구축

1. session6 폴더 만들기
2. 해당 폴더 들어가서 pipenv shell

| 패키지 설치

requests

- http 요청 패키지
- 파이썬에서 웹 페이지에 요청을 보낼 수 있어요

bs4(beautifulsoup v4)

- Html에서 정보 추출
- Html 파싱 패키지

| 패키지 설치

```
$ pipenv install requests
```

```
$ pipenv install bs4
```

오늘의 목표

스마트 스토어에서 노트 검색 결과 크롤링하기!

쿼리스트링

- 사용자가 입력 데이터를 전달하는 방법중의 하나
- url 주소에 미리 협의된 데이터를 파라미터를 통해 넘기는 것을 말한다.

<https://search.shopping.naver.com/search/all.nhn?변수1=값1&변수2=값2&변수3=값3>

- Url주소 뒤에 ?로 시작
- 변수와 값의 쌍으로 구성(변수 = 값)
- 각각의 쌍은 &로 구분

쿼리스트링

[https://search.shopping.naver.com/search/all.nhn?pagingIndex=29&pagingSize=80
&query=노트](https://search.shopping.naver.com/search/all.nhn?pagingIndex=29&pagingSize=80&query=노트)

- 29번째 페이지
- 한페이지에 80개씩
- 검색어는 노트

느낌 오죠?

for i in range(30) :

<https://search.shopping.naver.com/search/all.nhn?pagingIndex=1&pagingSize=80&query=노트>

<https://search.shopping.naver.com/search/all.nhn?pagingIndex=2&pagingSize=80&query=노트>

<https://search.shopping.naver.com/search/all.nhn?pagingIndex=3&pagingSize=80&query=노트>

<https://search.shopping.naver.com/search/all.nhn?pagingIndex=4&pagingSize=80&query=노트>

<https://search.shopping.naver.com/search/all.nhn?pagingIndex=5&pagingSize=80&query=노트>

⋮

크롤링

먼저 요청을 보내봅시다.

main.py

```
import requests
```

```
from bs4 import BeautifulSoup
```

```
note_html = requests.get('https://search.shopping.naver.com/search/all.nhn?pagingIndex=29&pagingSize=80&query=노트')
```

```
print(note_html)
```

크롤링

흐음.. 어떻게 분석하지?

main.py

```
import requests
```

```
from bs4 import BeautifulSoup
```

```
note_html = requests.get('https://search.shopping.naver.com/search/all.nhn?pagingIndex=29&pagingSize=80&query=노트')
```

```
note_soup = BeautifulSoup(note_html.text, "html.parser")
```

```
print(note_soup)
```

크롤링

상품 목록만 빼오기!

main.py

```
import requests
from bs4 import BeautifulSoup

note_html = requests.get('https://search.shopping.naver.com/search/all.nhn?pagingIndex=29&pagingSize=80&query=노트')

note_soup = BeautifulSoup(note_html.text, "html.parser")
note_list_box = note_soup.find("ul", {"class" : "goods_list"})
note_list = note_list_box.find_all('li', {"class" : "_itemSection"})
print(note_list)
```

상품 제목 빼오기!

```
▼<li class="_itemSection" data-nv-mid=
"81968727508" data-tr="s1s1" data-mall-pid=
"4424204955" data-mall-seq="224531" data-expose-
check="true" data-expose-area="lst*N" data-
expose-id="81968727508" data-expose-rank="2017"
data-is-shop-n="true" data-is-adult="false">
  ▶<div class="img_area">...</div>
  ▼<div class="info"> == $0
    ▼<div class="tit">
      <a href="https://cr2.shopping.naver.com/
      adcr.nhn?x=wB2ez8h8YR0EusmAjN%2B11P%2F%2...
      ksr21nRwdocfBoLYilkT18ogz5VsGBwAyjl4%3D&nv
      _mid=81968727508&cat_id=50003556" class=
      "link" target="_blank" title="미니수첩(플
      레잉코기-파랑)">미니수첩(플레잉코기-파랑)
      </a>
      <!-- N=a:lst*N.title,i:81968727508,r:2017
      -->
    </div>
```

크롤링

첫번째 상품 제목 빼오기!

main.py

```
import requests
```

```
from bs4 import BeautifulSoup
```

```
note_html = requests.get('https://search.shopping.naver.com/search/all.nhn?pagingIndex=29&pagingSize=80&query=노트')
```

```
note_soup = BeautifulSoup(note_html.text, "html.parser")
```

```
note_list_box = note_soup.find("ul", {"class" : "goods_list"})
```

```
note_list = note_list_box.find_all('li', {"class" : "_itemSection"})
```

```
title = note_list[0].find("div", {"class": "tit"}).find("a").string
```

```
print(title)
```


상품 가격 빼오기!

```
▼<li class="_itemSection" data-nv-mid="81968727508" data-tr="s1s1" data-mall-pid="4424204955" data-mall-seq="224531" data-expose-check="true" data-expose-area="1st*N" data-expose-id="81968727508" data-expose-rank="2017" data-is-shop-n="true" data-is-adult="false">
  ▶<div class="img_area">...</div>
  ▼<div class="info">
    ▶<div class="tit">...</div>
    ▼<span class="price">
      ▼<em> == $0
        <span class="num _price_reload" data-reload-date="2020.04.11.">2,000</span>
        "원"
      </em>
    </span>
```

크롤링

첫번째 상품 가격 빼오기!

```
import requests
from bs4 import BeautifulSoup

note_html = requests.get('https://search.shopping.naver.com/search/all.nhn
?pagingIndex=29&pagingSize=80&query=노트')
note_soup = BeautifulSoup(note_html.text, "html.parser")
note_list_box = note_soup.find("ul", {"class" : "goods_list"})
note_list = note_list_box.find_all('li', {"class" : "_itemSection"})

price = note_list[0].find("span", {"class": "price"}).text
print(price)
```

크롤링

첫번째 상품 이미지 주소 빼오기!

```
import requests
from bs4 import BeautifulSoup

note_html = requests.get('https://search.shopping.naver.com/search/all.nhn?pagingIndex=29&pagingSize=80&query=노트')
note_soup = BeautifulSoup(note_html.text, "html.parser")
note_list_box = note_soup.find("ul", {"class" : "goods_list"})
note_list = note_list_box.find_all('li', {"class" : "_itemSection"})
img_src = note_list[0].find('img', {"class" : "_productLazyImg"})['data-original']
print(img_src)
```

크롤링

눈썰미

https://shopping-phinf.pstatic.net/main_8196872/81968727508.1.jpg?type=f140

크기

크롤링

첫번째 상품 데이터 만들기

```
import requests
from bs4 import BeautifulSoup
note_html = requests.get('https://search.shopping.naver.com/search/all.nhn
?pagingIndex=29&pagingSize=80&query=노트')
note_soup = BeautifulSoup(note_html.text, "html.parser")
note_list_box = note_soup.find("ul", {"class" : "goods_list"})
note_list = note_list_box.find_all('li', {"class" : "_itemSection"})

title = note_list[0].find("div", {"class": "tit"}).find("a").string
price = note_list[0].find("span", {"class": "price"}).text

note = {
    'title' : title,
    'price' : price,
}

print(note)
```

공백 없애기

```
'\n2,000원\n'
```

```
price = note.find("span", {"class": "price"}).text.strip()
```

크롤링

리스트의 모든 상품 데이터 만들기

```
import requests
from bs4 import BeautifulSoup

note_html = requests.get(f'https://search.shopping.naver.com/search/all.nhn?pagingIndex=29&pagingSize=80&query=노트')
note_soup = BeautifulSoup(note_html.text, "html.parser")
note_list_box = note_soup.find("ul", {"class" : "goods_list"})
note_list = note_list_box.find_all('li', {"class" : "_itemSection"})
result = []

for note in note_list:

    title = note.find("div", {"class": "tit"}).find("a").string
    price = note.find("span", {"class": "price"}).text.strip()
    note_info = {
        'title' : title,
        'price' : price,
    }
    result.append(note_info)
print(result)
```

크롤링

리스트의 모든 상품 데이터 만들기

코드가 길어졌네요 ㅜㅜ

파일을 나누어 볼까요?

main.py

note.py

크롤링

note.py

```
def extract_info(note_list):  
  
    result = []  
  
    for note in note_list:  
  
        title = note.find("div",{"class":"tit"}).find("a").string  
        price = note.find("span",{"class":"price"}).text.strip()  
        note_info = {  
            'title' : title,  
            'price' : price,  
        }  
        result.append(note_info)  
  
    return result
```

크롤링

main.py

```
import requests
from bs4 import BeautifulSoup
from note import extract_info

note_html = requests.get('https://search.shopping.naver.com/search/all.nhn?pagingIndex=29&pagingSize=80&query=노트')
note_soup = BeautifulSoup(note_html.text, "html.parser")

note_list_box = note_soup.find("ul", {"class" : "goods_list"})
note_list = note_list_box.find_all('li', {"class" : "_itemSection"})

print(extract_info(note_list))
```

크롤링

실습

이미지 주소 데이터에 추가!

크롤링

정답

note.py

```
def extract_info(note_list):  
    result = []  
    for note in note_list:  
  
        title = note.find("div", {"class": "tit"}).find("a").string  
        price = note.find("span", {"class": "price"}).text.strip()  
        img_src = note.find('img', {"class": "_productLazyImg"})['data-original']  
        note_info = {  
            'title' : title,  
            'price' : price,  
            'img_src' : img_src,  
        }  
        result.append(note_info)  
    return result
```

크롤링

정리

<https://search.shopping.naver.com/search/all.nhn?pagingIndex=29&pagingSize=80&query=노트>

note.py => 한 페이지 당 모든 상품 목록 크롤링 완료

1~30페이지의 모든 상품 목록을 크롤링 해야된다!

크롤링

쿼리스트링

느낌 오죠?

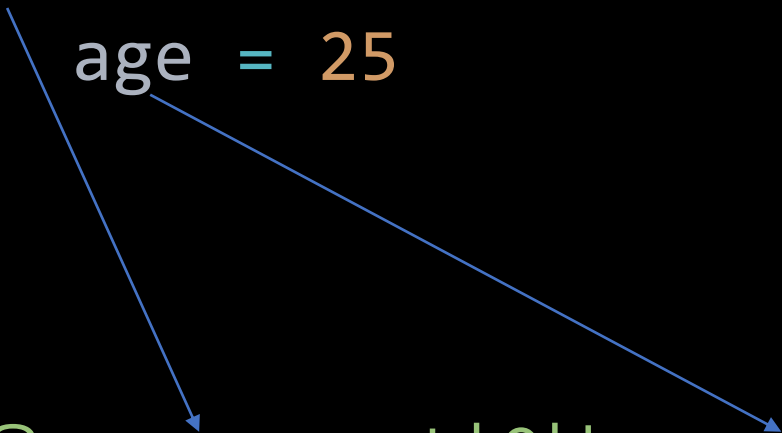
for i in range(30) :

? <u>pagingIndex=1</u> &pagingSize=80&query=노트	→	note_list	→	extract_info(<i>note_list</i>)
				+
? <u>pagingIndex=2</u> &pagingSize=80&query=노트	→	note_list	→	extract_info(<i>note_list</i>)
				+
? <u>pagingIndex=3</u> &pagingSize=80&query=노트	→	note_list	→	extract_info(<i>note_list</i>)
				+
? <u>pagingIndex=4</u> &pagingSize=80&query=노트	→	note_list	→	extract_info(<i>note_list</i>)
				+
? <u>pagingIndex=5</u> &pagingSize=80&query=노트	→	note_list	→	extract_info(<i>note_list</i>)

f-string

파이썬 문자열에 변수 넣는 방식

```
name = '박지환'  
age = 25  
  
print(f'이름은 {name}, 나이는 {age}')
```



크롤링

실습

30페이지 전체 상품 목록 print하기

- `extract_info(note_list)` 사용
 - f-string 사용

크롤링

힌트

- pagingIndex=29 -> f-string 이용해 변수 넣기
 - for i in range(30):

크롤링

(반)정답

main.py

```
final_result = []

for i in range(30):
    note_html = requests.get(f'https://search.shopping.naver.com/search/all.nhn?pagingIndex={i+1}&pagingSize=80&query=노트')
    note_soup = BeautifulSoup(note_html.text, "html.parser")
    note_list_box = note_soup.find("ul", {"class" : "goods_list"})
    note_list = note_list_box.find_all('li', {"class" : "_itemSection"})

    final_result = final_result + extract_info(note_list)

print(final_result)
```

오류 해결

```
{'title': '[커플문답] 커플백문백답 100문100답 사랑문답 연애 질문 책', 'price': '광고₩₩₩₩₩xa0₩₩₩₩₩광고 노출기준은 검색어에  
대한 연관성과광고주의 입찰가입니다.₩₩₩₩₩안내 레이어 달기₩₩₩₩₩9,900원', 'img_src': 'https://shopping-phinf.pstatic.net/main_8  
240578/82405783975.jpg?type=f140'}
```

오잉

price에 문제가 생겼네요?

- note.py의 extract_info 함수에서!
 - 고고우

문제 해결 ㅎㅎ

```
def extract_info(note_list):  
    result = []  
    for note in note_list:  
  
        title = note.find("div", {"class": "tit"}).find("a").string  
        price = note.find("span", {"class": "price"}).find('em').text.strip()  
        img_src = note.find('img', {"class": "_productLazyImg"})['data-original']  
        note_info = {  
            'title' : title,  
            'price' : price,  
            'img_src' : img_src,  
        }  
        result.append(note_info)  
    return result
```

크롤링

자 이제 30페이지까지 모든 상품 크롤링 완료했는데
머하지..?



크롤링

코딩 시작한지 한달..
데이터베이스가 머쥬?
아니 너무 어려운거 아닌가요

➤ ㄱ ㄷ ㄱ ㄷ. 저도 데이터베이스 몰라요

일단 해보쥬 ㅎㅎ

CSV

Comma Separated Values

- 필드를 쉼표(,)로 구분한 텍스트 데이터 및 텍스트 파일
 - 구글 스프레드 시트, 엑셀 등 호환 가능
- 파이썬에는 csv 다루는 기능이 내장 되어있다.

크롤링

csv 파일을 만들어 볼까요?

```
file = open("notes.csv", mode="w", newline='')
```

쓰기모드

크롤링

Csv 파일에 항목 넣기

```
import csv

file = open("notes.csv", mode="w", newline='')
writer = csv.writer(file)
writer.writerow(["title", "price", "img_src"])
```

크롤링

Csv 파일에 내용 넣기

```
for note in final_result:  
    row = []  
    row.append(note['title'])  
    row.append(note['price'])  
    row.append(note['img_src'])  
    writer.writerow(row)
```

크롤링

전체 코드

```
import requests
from bs4 import BeautifulSoup
from note import extract_info
import csv

file = open("notes.csv", mode="w", newline='')
writer = csv.writer(file)
writer.writerow(["title", "price", "img_src"])

final_result = []

for i in range(30):
    note_html = requests.get(f'https://search.shopping.naver.com/search/all.nhn?pagingIndex={i+1}&pageSize=80&query=노트')
    note_soup = BeautifulSoup(note_html.text, "html.parser")
    note_list_box = note_soup.find("ul", {"class" : "goods_list"})
    note_list = note_list_box.find_all('li', {"class" : "_itemSection"})

    final_result = final_result + extract_info(note_list)

for result in final_result:

    row = []
    row.append(result['title'])
    row.append(result['price'])
    row.append(result['img_src'])

    writer.writerow(row)
```

크롤링

실습

- 각 페이지 링크까지 크롤링해서 csv 파일에 저장

크롤링

힌트

없습니다.

돌아가세요.

```
def extract_info(note_list):
    result = []

    for note in note_list:
        title = note.find("div", {"class" : "tit"}).find("a").string
        price = note.find("span", {"class" : "price"}).find('em').text.strip()
        img_src = note.find("img", {"class" : "_productLazyImg"})['data-original']
        link = note.find("div", {"class" : "tit"}).find("a")['href']
        note_info = {
            'title' : title,
            'price' : price,
            'img_src' : img_src,
            'link' : link,
        }

        result.append(note_info)

    return result
```

크롤링

과제 1

네이버 책 사이트 크롤링

https://book.naver.com/category/index.nhn?cate_code=100&tab=new_book&list_type=list&sort_type=publishday

과제 1

- 1~8페이지 크롤링
 - 오류 발생시 가능한 페이지까지 크롤링
- Ex) 6페이지에서 오류 -> 5페이지까지 크롤링

- 제목
- 이미지 주소
- 상세 페이지 링크
- 저자
- 출판사
- 가격(optional)

크롤링

힌트

가격

- 예외 처리 필요

Ex)

```
price_box = book.find('em', {'class' : 'price'})
if price_box != None:
    price = price_box.string
else:
    price = '없음'
```

과제 1

> naver_books.csv로 저장

Github repo에 코드, csv 파일 업로드 후

Issue 댓글에 repo 주소 달기

크롤링

과제 2

선별진료소 크롤링

https://www.mohw.go.kr/react/popup_200128_3.html

드라이브에 있는
corona_hospital.py 사용

과제 2

- 시도
- 시군구
- 선별진료소(이름)
- 전화번호

과제 2

> corona_hospital.csv로 저장

Github repo에 코드, csv 파일 업로드 후

Issue 댓글에 repo 주소 달기

크롤링

힌트

모든 태그 찾을 때
-> find_all

안에 있는 모든 텍스트를 배열 형식으로
-> contents

크롤링

심화 과제(optional)

yes 24 사이트 크롤링

https://book.naver.com/category/index.nhn?cate_code=100&tab=new_book&list_type=list&sort_type=publishday

심화 과제

- 1~20페이지 크롤링
 - 오류 발생시 가능한 페이지까지 크롤링
- Ex) 5페이지에서 오류 -> 4페이지까지 크롤링

- 제목
- 이미지 주소
- 저자
- 출판사
- 가격
- 요약글

크롤링

공지

과제 마감 : 4월 27일까지

다음 세션: 5월 11일