세미프로젝트

기후에 따른 강남의
유동인구 파악 및 간단한
크롤링 예제

오지훈

목차

1. 주제선정
2. 가설 및 결과추론
3. 수집데이터
4. 크롤링 및 json
5. 결과물에 따른 자료 2차수집
6. 결론
7. 참고자료

1. 주제선정

- 주제를 선정한 이유
- 요즘 비가 많이 와서 비와 관련된 주제를 선택.
- 비가 오면 사람들이 밖으로 잘 나오지 않는다는 생각.
- 그렇다면 정확히 비가 오는 날과 비가 오지 않는 날 사람들의 유동인구를 정확히 파악하기 위해 이 주제를 선정

2. 가설 및 결과 추론

- 전국 모든 곳을 검토할 수 없고 모든 기간을 검토할 수 없다
- 유동인구가 굉장히 많은 강남을 선택했고, 기간은 5월부터 6월23일까지
- 약 2달간의 날씨정보와 유동인구를 파악했다.
- 실험 전 내 예상은 비가 오는 날엔 유동인구가 적을 것이라는 생각이다.

3. 수집할 데이터 목록

- 서울시 열린 데이터 광장에서 유동인구
- 기상청 날씨누리에서 5월과 6월 날씨 데이터
- 간단한 크롤링 실험을 위해 비 오는 날 먹기 좋은 음식이라는 주제로 검색하여 웹사이트에서 음식들의 정보를 수집(5곳의 정보를 수집)

4. 크롤링 및 json

◦ 간단한 크롤링 url들 1~5번까지 중복

```python
# 관심사와 관련된 데이터를 수집할 수 있는 웹사이트
url1 = 'https://brunch.co.kr/@gorrajeju/775'
url2 = 'https://menupick.tistory.com/15'
url3 = 'https://welcome8007.com/33'
url4 = 'https://www.maybugs.com/news/articleView.html?idxno=581068'
url5 = 'https://monimoney.tistory.com/entry/%EB%B9%84-%EC%98%A4%EB%8A%94-%EB%82%A0-%EB%A8%B9%EC%9C%BC%EB%A9%B4-%EB%8D%94-%EB%A7%9B%EC%9E%88%EB%
```

Python

```python
userAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36"
```

```python
# 접속객체 생성
session = requests.Session()
# 접속객체에 부가정보(header) 삽입하기
session.headers.update({
    "Referer": "",
    "User-Agent": userAgent
})
```

```python
# 생성한 접속객체를 활용하여 API에 접속
r = session.get(url1)
if r.status_code != 200:
    msg = "[%d Error] %s 에러가 발생함" % (r.status_code, r.reason)
    raise Exception(msg)
```

```python
# 인코딩 형식 지정하여 beautifulsoup객체를 생성
r.encoding = "utf-8"
soup = BeautifulSoup(r.text)
soup
```

4. 크롤링 및 json

◦ 수집 1번

```
# 인코딩 형식 지정하여 beautifulsoup객체를 생성
r.encoding = "utf-8"
soup = BeautifulSoup(r.text)
# soup
```

✓ 0.0s

```
# soup객체로부터 원하는 부분 추출하기
test1 = soup.select('h1 > b')
test1
```

✓ 0.0s

```
[<b>파전 / 수제비</b>, <b>삼겹살</b>, <b>잔치국수</b>, <b>라면 / 곱창&amp;막창</b>]
```

```
h1 = []
h1_name = []
for item in test1:
    h1 += item.text.strip().split()
result1 = "".join(h1).replace("/", "").strip()
print(result1)
h1_name.append(result1[0:2])
h1_name.append(result1[2:5])
h1_name.append(result1[5:8])
h1_name.append(result1[8:12])
h1_name.append(result1[12:14])
h1_name.append(result1[14:])
print(h1_name)
```

✓ 0.0s

결과물

세미프로젝트 > result1.xlsx

	A	B
1		0
2	0	파전
3	1	수제비
4	2	삼겹살
5	3	잔치국수
6	4	라면
7	5	곱창&막창

4. 크롤링 및 json

◦ 수집 2번

```
# soup객체로부터 원하는 부분 추출하기
test4 = soup.select('#articleBody b')
# test4
```

0.0s

```
h4 = []
h4_name = []
for item in test4:
    h4 += item.text.strip().split()
# print(h4)
for i in range(1, 7):
    if i%2 == 0:
        h4_name.append(h4[i-1])
for i in range(15, 28):
    if i%2 == 0:
        h4_name.append(h4[i])
for i in range(11, 15):
    if (i+1)%2 == 0:
        h4_name.append(h4[i])
print(h4_name)
```

0.0s

```
['부침개[전]', '막걸리', '얼큰한', '삼겹살', '소주'
```

```
df = DataFrame(h4_name)
df.to_excel("result4.xlsx")
df
```

0.0s

결과물

세미프로젝트 > result4.xlsx

	A	B
1		0
2	0	부침개[전]
3	1	막걸리
4	2	얼큰한
5	3	삼겹살
6	4	소주
7	5	우동
8	6	치킨
9	7	맥주
10	8	김치나베돈까스
11	9	국밥
12	10	아구

4. 크롤링 및 json

◦ 수집 1~5번 결과물 합쳐서 카운트 하기

```python
# python 3.9
import pandas as pd

# 읽어올 엑셀 파일 지정
filename1 = 'result1.xlsx'
filename2 = 'result2.xlsx'
filename3 = 'result3.xlsx'
filename4 = 'result4.xlsx'
filename5 = 'result5.xlsx'

# 엑셀 파일 읽어 오기
df1 = pd.read_excel(filename1, engine='openpyxl')
df2 = pd.read_excel(filename2, engine='openpyxl')
df3 = pd.read_excel(filename3, engine='openpyxl')
df4 = pd.read_excel(filename4, engine='openpyxl')
df5 = pd.read_excel(filename5, engine='openpyxl')
```

```python
from collections import Counter
lista = []
for i in df1[0]:
    lista.append(i)
for i in df2[0]:
    lista.append(i)
for i in df3[0]:
    lista.append(i)
for i in df4[0]:
    lista.append(i)
for i in df5[0]:
    lista.append(i)
# print(lista)
counter = Counter(lista)
print(counter)
```

```
Counter({'삼겹살': 3, '파전': 2, '수제비': 2, '잔치국수': 2, '김치찌개': 2, '칼국수': 2,
```

4. 크롤링 및 json

◦ Json 코딩

```
import json
import requests
from datetime import datetime, timedelta
from pandas import DataFrame
from collections import deque


with open ("강남.json", 'r', encoding='utf-8') as f:
    data = json.load(f)
date = data['DATA']
mylist = []
for i in date:
    # print(i.keys())
    key_value = list(i.values())
    # print(key_value)
    if int(key_value[4]) > 20230600:
        mylist.insert(0, i)
print(mylist)
df = DataFrame(mylist)
df.to_excel("ingu.xlsx")
df
```

6 월 유동인구

세미프로젝트 > ingu2.xlsx

	A	B	C	D	E	F
1		총인구수	일 최대인원	날짜	일 최소인구	지역
2	0	910,414.58	1,153,866.50	20230601	693,919.15	강남구
3	1	904,247.97	1,141,073.63	20230602	694,864.55	강남구
4	2	766,319.69	880,975.68	20230603	676,577.77	강남구
5	3	707,044.00	775,252.18	20230604	658,429.06	강남구
6	4	825,705.24	1,010,457.06	20230605	654,348.83	강남구
7	5	713,769.33	765,441.00	20230606	666,470.48	강남구
8	6	899,455.73	1,155,505.38	20230607	670,408.54	강남구
9	7	910,864.69	1,161,346.96	20230608	688,267.95	강남구
10	8	904,013.62	1,143,425.32	20230609	689,588.29	강남구
11	9	772,207.15	883,949.17	20230610	686,318.51	강남구
12	10	723,458.83	793,602.97	20230611	667,773.14	강남구
13	11	898,676.90	1,150,830.79	20230612	675,299.33	강남구
14	12	915,445.85	1,168,141.22	20230613	692,603.53	강남구
15	13	919,211.35	1,175,987.86	20230614	695,626.99	강남구
16	14	922,459.91	1,179,711.68	20230615	696,627.85	강남구
17	15	910,937.90	1,155,836.59	20230616	697,860.57	강남구
18	16	780,641.19	907,110.23	20230617	691,909.04	강남구
19	17	727,935.07	809,549.93	20230618	668,329.11	강남구
20	18	900,871.88	1,158,199.16	20230619	677,742.34	강남구
21	19	920,983.69	1,181,729.27	20230620	693,915.03	강남구
22	20	922,347.16	1,180,765.04	20230621	697,720.02	강남구
23	21	934,743.38	1,197,479.07	20230622	702,646.95	강남구
24	22	921,550.26	1,169,908.51	20230623	704,639.01	강남구

4. 크롤링 및 json

◦ Json 코딩

```
with open ("강남.json", 'r', encoding='utf-8') as f:
    data = json.load(f)
date = data['DATA']
mylist = []
for i in date:
    # print(i.keys())
    key_value = list(i.values())
    # print(key_value)
    if 20230500 < int(key_value[4]) < 20230600:
        mylist.insert(0, i)
print(mylist)
df = DataFrame(mylist)
df.to_excel("ingu3.xlsx")
df
```

5월 유동인구

세미프로젝트 > ingu3.xlsx

	A	B	C	D	E	F
		총인구수	일 최대인구수	날짜	일 최소인구수	지역
2	0	725,799.93	792,856.33	20230501	661,239.40	강남구
3	1	902,843.03	1,152,347.47	20230502	679,364.33	강남구
4	2	905,251.76	1,152,489.46	20230503	684,474.85	강남구
5	3	897,337.44	1,138,876.47	20230504	685,796.92	강남구
6	4	709,544.36	760,984.46	20230505	656,056.40	강남구
7	5	722,897.15	828,230.25	20230506	638,963.33	강남구
8	6	702,686.73	767,080.38	20230507	642,207.61	강남구
9	7	883,063.06	1,124,255.97	20230508	665,058.38	강남구
10	8	910,942.38	1,162,258.15	20230509	686,942.35	강남구
11	9	909,275.32	1,157,935.73	20230510	691,490.12	강남구
12	10	909,700.35	1,157,998.61	20230511	687,627.86	강남구
13	11	910,136.56	1,151,355.67	20230512	696,332.82	강남구
14	12	787,387.30	903,583.76	20230513	690,140.05	강남구
15	13	732,139.59	800,706.06	20230514	677,945.16	강남구
16	14	898,668.84	1,140,164.82	20230515	679,863.30	강남구
17	15	915,657.20	1,165,449.95	20230516	697,059.79	강남구
18	16	915,801.56	1,158,480.06	20230517	701,573.77	강남구
19	17	919,513.89	1,165,003.78	20230518	704,391.14	강남구
20	18	910,608.38	1,140,441.58	20230519	706,402.77	강남구
21	19	783,215.68	901,387.74	20230520	692,632.57	강남구
22	20	734,148.84	805,935.42	20230521	676,617.22	강남구
23	21	897,379.02	1,139,941.01	20230522	681,187.11	강남구
24	22	913,794.54	1,161,165.89	20230523	694,102.33	강남구
25	23	917,018.33	1,160,694.66	20230524	699,382.64	강남구
26	24	919,598.17	1,161,640.02	20230525	701,125.00	강남구
27	25	904,025.80	1,136,868.97	20230526	702,995.12	강남구
28	26	757,984.12	867,739.05	20230527	663,390.24	강남구
29	27	711,609.31	794,699.87	20230528	648,503.37	강남구
30	28	719,427.56	787,719.57	20230529	654,220.06	강남구
31	29	899,037.05	1,148,120.05	20230530	673,308.08	강남구
32	30	906,301.08	1,146,017.11	20230531	689,143.11	강남구

4. 크롤링 및 json

- 6월 기상청이 발표한 날씨(23)일까지 자료

```
import requests
import json
from bs4 import BeautifulSoup
from pandas import DataFrame
```
✓ 0.5s

```
url = "https://www.weather.go.kr/plus/land/current/past_cal.jsp"
```
✓ 0.0s

```
userAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/5
```
✓ 0.0s

```
# 접속객체 생성
session = requests.Session()
# 접속객체에 부가정보(header) 삽입하기
session.headers.update({
    "Referer": "",
    "User-Agent": userAgent
})
```
✓ 0.0s

```
# 생성한 접속객체를 활용하여 API에 접속
r = session.get(url)
if r.status_code != 200:
    msg = "[%d Error] %s 에러가 발생함" % (r.status_code, r.reason)
    raise Exception(msg)
```
✓ 0.1s

```
r.encoding = 'euc-kr'
soup = BeautifulSoup(r.text)
# soup
```
✓ 0.0s

```
# soup객체로부터 원하는 부분 추출하기
date = soup.select('tr')
# print(date)
```
✓ 0.0s

```
h1 = []
h1_name = []
day = []
info = []
for item in date:
    h1 += item.text.strip().split()
day.append(h1[25:28])
info.append(h1[28:33])
day.append(h1[33:40])
info.append(h1[40:49])
day.append(h1[49:56])
info.append(h1[57:72])
day.append(h1[72:78])
info.append(h1[79:94])
day1 = sum(day, [])
info1 = sum(info, [])
# print(day1)
# print(info1)
```

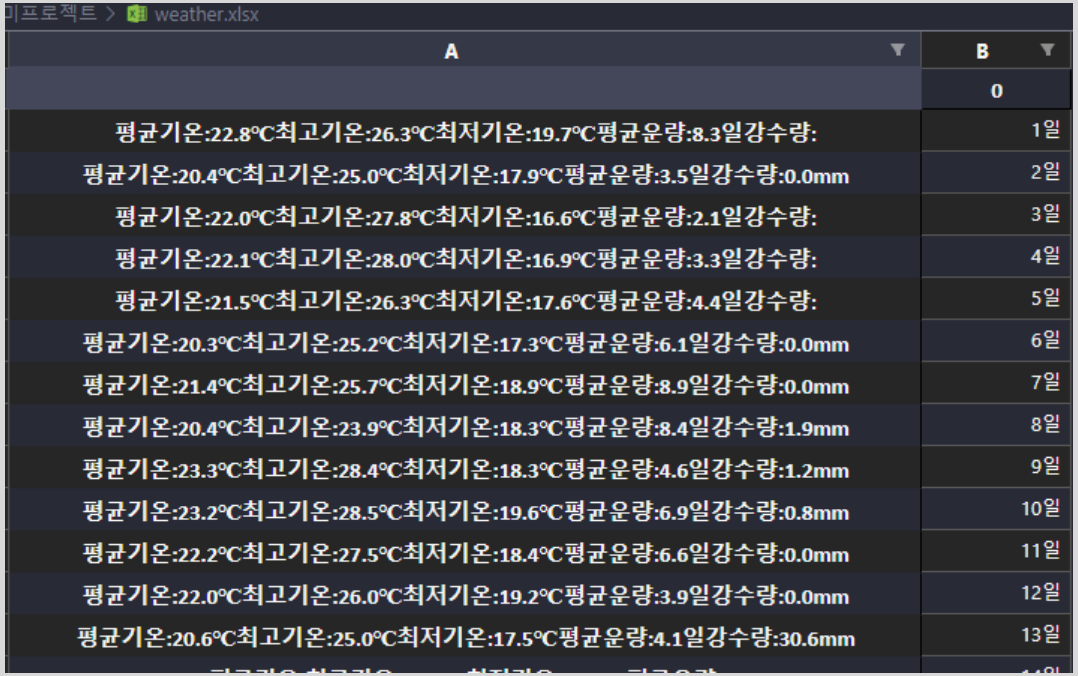
```
for i in info1:
    if i == "-":
        info1.remove("-")
# print(info1)
for i in range(0,5):
    day1.append(i)
print(len(day1))
print(len(info1))
```

```
28
28
```

```
df = DataFrame(day1, info1)
df.to_excel("weather.xlsx")
df
```

4. 크롤링 및 json

6 월 기상청 자료 결과



A	B
	0
평균기온:22.8℃최고기온:26.3℃최저기온:19.7℃평균운량:8.3일강수량:	1일
평균기온:20.4℃최고기온:25.0℃최저기온:17.9℃평균운량:3.5일강수량:0.0mm	2일
평균기온:22.0℃최고기온:27.8℃최저기온:16.6℃평균운량:2.1일강수량:	3일
평균기온:22.1℃최고기온:28.0℃최저기온:16.9℃평균운량:3.3일강수량:	4일
평균기온:21.5℃최고기온:26.3℃최저기온:17.6℃평균운량:4.4일강수량:	5일
평균기온:20.3℃최고기온:25.2℃최저기온:17.3℃평균운량:6.1일강수량:0.0mm	6일
평균기온:21.4℃최고기온:25.7℃최저기온:18.9℃평균운량:8.9일강수량:0.0mm	7일
평균기온:20.4℃최고기온:23.9℃최저기온:18.3℃평균운량:8.4일강수량:1.9mm	8일
평균기온:23.3℃최고기온:28.4℃최저기온:18.3℃평균운량:4.6일강수량:1.2mm	9일
평균기온:23.2℃최고기온:28.5℃최저기온:19.6℃평균운량:6.9일강수량:0.8mm	10일
평균기온:22.2℃최고기온:27.5℃최저기온:18.4℃평균운량:6.6일강수량:0.0mm	11일
평균기온:22.0℃최고기온:26.0℃최저기온:19.2℃평균운량:3.9일강수량:0.0mm	12일
평균기온:20.6℃최고기온:25.0℃최저기온:17.5℃평균운량:4.1일강수량:30.6mm	13일

4. 크롤링 및 json

◦ 5월 기상청 날씨

```
r.encoding = 'euc-kr'
soup = BeautifulSoup(r.text)
# soup
```
✓ 0.0s

```
# soup객체로부터 원하는 부분 추출하기
date = soup.select('tr')
# print(date)
```
✓ 0.0s

```
h2 = []
h2_name = []
day = []
info = []
for item in date:
    h2 += item.text.strip().split()
day = [s for s in h2[25:] if "일" in s]
for i in h2[30:]:
    if '비' in i or "소나기" in i:
        info.append(i)
# print(info)
# print(len(day))
```
✓ 0.0s

5월 기상철 날씨 결과

세미프로젝트 > 5월 (1).xlsx

	A	B	C
1		0	
2	0	1일	0
3	1	2일	0
4	2	3일	0
5	3	4일	0
6	4	5일	비박무
7	5	6일	비박무
8	6	7일	0
9	7	8일	0
10	8	9일	0
11	9	10일	0
12	10	11일	0
13	11	12일	0
14	12	13일	비박무
15	13	14일	0
16	14	15일	0
17	15	16일	0
18	16	17일	0
19	17	18일	비박무
20	18	19일	0
21	19	20일	0
22	20	21일	비안개비박무황
23	21	22일	소나기황사
24	22	23일	황사
25	23	24일	0

4. 크롤링 및 json

5월 자료 합

	A	B	C	D	E	F	G	H
1		0		총인구수	일 최대인구수	날짜	일 최소인구수	지역
2	0	1일	0	725,799.93	792,856.33	20,230,501	661,239.40	강남구
3	1	2일	0	902,843.03	1,152,347.47	20,230,502	679,364.33	강남구
4	2	3일	0	905,251.76	1,152,489.46	20,230,503	684,474.85	강남구
5	3	4일	0	897,337.44	1,138,876.47	20,230,504	685,796.92	강남구
6	4	5일	비박무	709,544.36	760,984.46	20,230,505	656,056.40	강남구
7	5	6일	비박무	722,897.15	828,230.25	20,230,506	638,963.33	강남구
8	6	7일	0	702,686.73	767,080.38	20,230,507	642,207.61	강남구
9	7	8일	0	883,063.06	1,124,255.97	20,230,508	665,058.38	강남구
10	8	9일	0	910,942.38	1,162,258.15	20,230,509	686,942.35	강남구
11	9	10일	0	909,275.32	1,157,935.73	20,230,510	691,490.12	강남구
12	10	11일	0	909,700.35	1,157,998.61	20,230,511	687,627.86	강남구
13	11	12일	0	910,136.56	1,151,355.67	20,230,512	696,332.82	강남구
14	12	13일	비박무	787,387.30	903,583.76	20,230,513	690,140.05	강남구
15	13	14일	0	732,139.59	800,706.06	20,230,514	677,945.16	강남구
16	14	15일	0	898,668.84	1,140,164.82	20,230,515	679,863.30	강남구
17	15	16일	0	915,657.20	1,165,449.95	20,230,516	697,059.79	강남구
18	16	17일	0	915,801.56	1,158,480.06	20,230,517	701,573.77	강남구
19	17	18일	비박무	919,513.89	1,165,003.78	20,230,518	704,391.14	강남구
20	18	19일	0	910,608.38	1,140,441.58	20,230,519	706,402.77	강남구
21	19	20일	0	783,215.68	901,387.74	20,230,520	692,632.57	강남구
22	20	21일	비안개비박무황	734,148.84	805,935.42	20,230,521	676,617.22	강남구
23	21	22일	소나기황사	897,379.02	1,139,941.01	20,230,522	681,187.11	강남구
24	22	23일	황사	913,794.54	1,161,165.89	20,230,523	694,102.33	강남구
25	23	24일	0	917,018.33	1,160,694.66	20,230,524	699,382.64	강남구
26	24	25일	비	919,598.17	1,161,640.02	20,230,525	701,125.00	강남구

6월 자료 합본

세미프로젝트 > 6월합친것.xlsx

	A	B	C	D	E	F	G
1		날짜	총인구수	일 최대인원	일 최소인구	강수량	지역
2	0	20230601	910,414.58	1,153,866.50	693,919.15	0	강남구
3	1	20230602	904,247.97	1,141,073.63	694,864.55	0	강남구
4	2	20230603	766,319.69	880,975.68	676,577.77	0	강남구
5	3	20230604	707,044.00	775,252.18	658,429.06	0	강남구
6	4	20230605	825,705.24	1,010,457.06	654,348.83	0	강남구
7	5	20230606	713,769.33	765,441.00	666,470.48	0	강남구
8	6	20230607	899,455.73	1,155,505.38	670,408.54	0	강남구
9	7	20230608	910,864.69	1,161,346.96	688,267.95	1.90	강남구
10	8	20230609	904,013.62	1,143,425.32	689,588.29	1.20	강남구
11	9	20230610	772,207.15	883,949.17	686,318.51	0.80	강남구
12	10	20230611	723,458.83	793,602.97	667,773.14	0	강남구
13	11	20230612	898,676.90	1,150,830.79	675,299.33	0	강남구
14	12	20230613	915,445.85	1,168,141.22	692,603.53	30.60	강남구
15	13	20230614	919,211.35	1,175,987.86	695,626.99	0	강남구
16	14	20230615	922,459.91	1,179,711.68	696,627.85	0	강남구
17	15	20230616	910,937.90	1,155,836.59	697,860.57	0	강남구
18	16	20230617	780,641.19	907,110.23	691,909.04	0	강남구
19	17	20230618	727,935.07	809,549.93	668,329.11	0	강남구
20	18	20230619	900,871.88	1,158,199.16	677,742.34	6.40	강남구
21	19	20230620	920,983.69	1,181,729.27	693,915.03	24.90	강남구
22	20	20230621	922,347.16	1,180,765.04	697,720.02	0	강남구
23	21	20230622	934,743.38	1,197,479.07	702,646.95	0	강남구

5. 결과물

- 기상청 자료와 유동인구 분포를 봤을 때 내 생각보다 별로 관련이 크지 않아 당황스러웠고
- 원래 가설대로라면 비올 때 사람들이 적게 움직이고, 그들에게 추천해줄 음식이 있다면
- 그 음식을 추려서 가격정보와 위치 정보 등을 가져오려고 했으나
- 기후와 유동인구 사이에 큰 특이점을 발견하지 못하였기때문에
- 왜? 왜 큰 관련이 없는가에 대하여 크롤링을 해봐야겠다는 결론이 나왔다.

5. 결과물에 따른 다른 2차 자료 수집

```
import requests
from bs4 import BeautifulSoup
from pandas import DataFrame
```
✓ 0.0s

```
userAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
```
✓ 0.0s

```
Clist = [0, 10, 20]
```
✓ 0.0s

```
# 접속객체 생성
session = requests.Session()
# 접속객체에 부가정보(header) 삽입하기
session.headers.update({
    "Referer": "",
    "User-Agent": userAgent
})
```
✓ 0.0s

논문파일 페이지 별로 크롤링

```
# 수집한 정보를 저장할 빈 리스트
artricle = []
for c in Clist:
    url = "https://scholar.google.co.kr/scholar?start={}&q=%EA%B8%B0%EC%83%81+%EB%85%BC%EB%AC%B8&hl=ko&as_sdt=0,5&as_vis=1".format (c)

    r = session.get(url)
    # 접속에 실패한 경우
    if r.status_code != 200:
    # 에러코드와 에러메시지 출력
        msg = "[%d Error] %s 에러가 발생함" % (r.status_code, r.reason)
    # 에러를 강제로 생성시킴
        raise Exception(msg)
    r.encoding = "utf-8"
    #print(r.text)
    soup = BeautifulSoup(r.text)
    # soup

    college_list = soup.select("h3 > a")
    print(college_list) # 페이지 별로 artcle 수집
```

5. 결과물에 따른 다른 자료 수집

- 결과가 어제까진 잘 나왔으나 오늘은 로봇이 아닙니다 화면이 뜨면서 빈 리스트가 출력
- 구글링 해보니 안 배웠던 부분들이 있어서 다시 할 수 있는 네이버 뉴스 기사를 수집하기로 결정
- 어제 나온 결과물 ->

[<a data-clk="hl=ko&sa=T&ct=res&cd=0&d=5124573006012166822&ei=x_abZMnVGMuKywTg9YW4CA" data-clk-at
jHkcJ" href="https://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE09803423" id="pjLpcP8jHkcJ">기후변화가 기상</
물과 홍수에 미치는 영향, <a data-clk="hl=ko&sa=T&ct=res&cd=1&d=17429875264829021014&ei=x_abZM
YW4CA" data-clk-atid="YluAtzRb4_EJ" href="https://papersearch.net/thesis/article.asp?key=74488" id="YluAtzRb4_EJ">한국
의 기상인자가 송이 발생에 미치는 영향과 그 극복방안, <a data-clk="hl=ko&sa=T&ct=res&cd=2&d=
284372197&ei=x_abZMnVGMuKywTg9YW4CA" data-clk-atid="5XyaePL2BogJ" href="https://www.dbpia.co.kr/Journal/articleDet
=NODE10419169" id="5XyaePL2BogJ">기상특성을 이용한 전국 산불발생확률모형 개발, <a data-clk="hl=ko&sa=T
&cd=3&d=16638574743183459196&ei=x_abZMnVGMuKywTg9YW4CA" data-clk-atid="flM88JlX60YJ" href="https://papers
hesis/article.asp?key=1578350" id="flM88JlX60YJ">연구논문/연안해역의 기상 파랑관측망 설계 및 해석기술의
파랑관측자료의 해석방법, <a data-clk="hl=ko&sa=T&ct=res&cd=4&d=9388420938626650768amp;ei=x_abZMnV
4CA" data-clk-atid="dDveAOlvBwOJ" href="https://papersearch.net/thesis/article.asp?key=2897016" id="dDveAOlvBwOJ">
(論文): 기상 전용 항공기 도입 및 운영방안 연구, <a data-clk="hl=ko&sa=T&ct=res&cd=5&d=5942
2287&ei=x_abZMnVGMuKywTg9YW4CA" data-clk-atid="HyEotuUyeFIJ" href="https://kmbase.medric.or.kr/KMID/I0245200601500
="HyEotuUyeFIJ">대구지역의 기상조건에 따른 도시열섬강도의 계절별 변화특성, <a data-clk="hl=ko&sa=T&amp
p;ct=res&cd=6&d=21818423654668919178amp;ei=x_abZMnVGMuKywTg9YW4CA" data-clk-atid="jVaNTRJ2Rx4J" href="https:/
chgate.net/profile/Hyunsu-Kim-10/publication/305703718_Weather-sensitive_Diseases_and_Their_Correlations_with_Meteoro
tors_Results_from_Academic_Papers/links/579affba08ae024e100e75c7/Weather-sensitive-Diseases-and-Their-Correlations-wi
ogical-Factors-Results-from-Academic-Papers.pdf" id="jVaNTRJ2Rx4J">학술논문 분석을 통한 기상민감질환 선
상인자와의 관련성 고찰, <a data-clk="hl=ko&sa=T&ct=res&cd=7&d=41679896533536519748amp;ei=x_abZ
g9YW4CA" data-clk-atid="Bt-lu4GslzkJ" href="https://papersearch.net/thesis/article.asp?key=3704962" id="Bt-lu4GslzkJ">
>기상학회지 수록 논문에 기반한 산림기상 연구 추세와 전망, <a data-clk="hl=ko&sa=T&ct=res
amp;d=4767214941604784058amp;ei=x_abZMnVGMuKywTg9YW4CA" data-clk-atid="xZA6VKynnQYJ" href="https://papersearch.net/go
ulltext.asp?file_name=23107648.pdf" id="xZA6VKynnQYJ">학술논문 분석과 설문조사를 통한 기상민감질환 선정
인자와의 상관성 평가, <a data-clk="hl=ko&sa=T&ct=res&cd=9&d=12476919916987030401&ei=x_abZM
9YW4CA" data-clk-atid="gRsCF4_sJqOJ" href="https://papersearch.net/thesis/article.asp?key=69837" id="gRsCF4_sJqOJ">
섬바디의 생태학적 연구; 제 3 보 계절생산성과 기상요인과의 관계]
[<a data-clk="hl=ko&sa=T&ct=res&cd=10&d=24610363972067625068amp;ei=x_abZKuTM7qI6rQPj-mt2AE" data-clk-
pBbJyIJ" href="https://papersearch.net/thesis/article.asp?key=3996325" id="CvTQapBbJyIJ">데이터마이닝을 활용한 북한 <
기후 연구 동향 분석: 1982~ 2022 년 북한 학술지 논문을 중심으로, <a data-clk="hl=ko&sa=T&ct=res&amp
d=8972040133600290431&ei=x_abZKuTM7qI6rQPj-mt2AE" data-clk-atid="fybLMfcWg3wJ" href="https://www.dbpia.co.kr/Jour
Detail?nodeId=NODE00939861" id="fybLMfcWg3wJ">우리학회에서 발행하는 학술지에 대한 논문 투고에 대하여 다음과 같
업무에 참조하시기 바랍니다.-한국기상학회지• 제출논문: 3 부• 심사료 4 0, 0 0 0 원 (우리 …, <a data-
amp;sa=T&ct=res&cd=12&d=13426451345790042533&ei=x_abZKuTM7qI6rQPj-mt2AE" data-clk-atid="pYHhO1xWYLoJ"
s://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE09750703" id="pYHhO1xWYLoJ">2003 년 여름 강우패턴과 기상변
data-clk="hl=ko&sa=T&ct=res&cd=13&d=8981893593817250010&ei=x_abZKuTM7qI6rQPj-mt2AE" data-clk-atid=
nwJ" href="https://papersearch.net/thesis/article.asp?key=80043" id="2njYralYpnwJ">연구논문; 1992-93 년 하계 등
온과 염분의 분포, <a data-clk="hl=ko&sa=T&ct=res&cd=14&d=6405362570659084961&ei=x_abZKuTM7qI6r

5. 결과물에 따른 다른 자료 수집

◦ 네이버 뉴스 기사 수집

```
from bs4 import BeautifulSoup
import requests
import time
from pandas import DataFrame
```

✓ 0.0s

```
page_num = [0, 1, 11, 21]
search = input("검색할 내용을 입력하세요:")
#url 생성
urlEL = "https://search.naver.com/search.naver?sm=tab_hty.top&where=news&query=" + search + "%start=" + str(page_num[0])
print("생성url: ",urlEL)
```

✓ 6.4s

```
생성url:  https://search.naver.com/search.naver?sm=tab_hty.top&where=news&query=날씨%start=0
```

+ Code + Markdown

```
userAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36"
```

✓ 0.0s

5. 결과물에 따른 다른 자료 수집

◦ 네이버 뉴스 기사 수집

```
# 접속객체 생성
session = requests.Session()
# 접속객체에 부가정보(header) 삽입하기
session.headers.update({
    "Referer": "",
    "User-Agent": userAgent
})
news = []
for i in range(len(page_num)):
    url = "https://search.naver.com/search.naver?sm=tab_hty.top&where=news&query=" + search + "%start=" + str(page_num[i])

    r = session.get(url)
    if r.status_code != 200:
        msg = "[%d Error] %s 에러가 발생함" % (r.status_code, r.reason)
        raise Exception(msg)
    r.encoding = "utf-8"
    soup = BeautifulSoup(r.text)

    articles = soup.select("div.group_news > ul.list_news > li div.news_area > a")

    news_title = []
    content = []
    agency = []
    for i in articles:
        news_title.append(i.attrs['title'])

        content.append(i.attrs['href'])
# print(news_title)
# print(content)
df = DataFrame(content, news_title)
df.to_excel("네이버기사.xlsx")
df
```

5. 결과물에 따른 다른 자료 수집

◦ 네이버 뉴스 기사 수집 결과물

세미프로젝트 > 네이버기사.xlsx

	A	B
1		0
2	중랑구, 침수피해 예방에 총력	http://www.breaknews.com/972117
3	LS그룹 국가 재난상황 피해복구 지원, 미래세대 꿈	https://sports.donga.com/article/all/20230502/119102589/2
4	사회공헌 앞장서 온 LS그룹, "따뜻한 나눔 활동, 기	http://www.sporbiz.co.kr/news/articleView.html?idxno=650019
5	LS그룹, 국가 재난상황 피해복구 지원...'미래세대 권	http://www.sentv.co.kr/news/view/654527
6	LS그룹 계열사 국가 재난상황 지원 "미래세대 꿈의	http://www.fntimes.com/html/view.php?ud=202303300122552
7	LS는 사회공헌의 마술사...미래세대의 발판 세우다	https://www.nbntv.kr/news/articleView.html?idxno=69222
8	"미래세대 꿈의 발판 제공"...LS그룹, 사회적 책임 실	http://www.inews24.com/view/1579538
9	보은군, 행안부 2023 중초지구 풍수해생활권 종합	http://www.breaknews.com/925896
10	[유통가 레이더] 배민, 쇼핑라이브에서 '소상공인의	http://www.greened.kr/news/articleView.html?idxno=297302
11	[지구촌 한줄뉴스](종합)펠로시 대만 방문·미국 우	http://www.newscj.com/article/20220802580109

6. 결론

- 날씨와 유동인구간의 상관관계를 밝히고 그에 따라 비올 때 생각나는 음식의 대한 자료를 얻으려고 했으나 날씨와 유동인구간의 상관관계가 그리 높지 않았음
- 그렇다면 왜 높지 않은가에 대하여 알기 위해 구글 논문을 크롤링 하였으나 셀레니움과 구글드라이버 설치를 통해서 열어봐도 로봇이 아닙니다 화면으로 막힘
- 그래서 새롭게 날씨와 관련된 기사를 통해 유동인구와 관련된 자료를 수집을 시도함
- 아쉬웠던 점은 논문크롤링이 어제까지 됐을 때 결과물을 어느 정도 출력해놓을 껄 했던 아쉬움과 시간이 없어서 네이버 기사에 링크를 타고 들어가 안에 있는 내용물들에 대한 수집을 못했다는 점이 아쉬웠다.

7. 참고자료

참고한 자료나 홈페이지 url

음식과 관련된 웹사이트 참고자료

```
url1 = 'https://brunch.co.kr/@gorrajeju/775'
url2 = 'https://menupick.tistory.com/15'
url3 = 'https://welcome8007.com/33'
url4 = 'https://www.maybugs.com/news/articleView.html?idxno=581068'
url5 = 'https://www.daily.co.kr/life312718976#lifeback'
```

Json 강남 유동인구 파일 얻은 곳 : https://data.seoul.go.kr/

기상청 날씨 정보 : https://www.weather.go.kr/w/index.do

구글 논문자료 : https://scholar.google.co.kr/

네이버 기사 : https://search.naver.com/search.naversm=tab_hty.top&where=news&query=