Web Crawling

- 웹 페이지에서 데이터를 수집하는 방법에 대해서 학습

웹크롤링 방법

웹페이지의 종류

- 정적인 페이지 : 웹 브라우저에 화면이 한번 뜨면 이벤트에 의한 화면의 변경이 없는 페이지
- 동적인 페이지 : 웹 브라우저에 화면이 뜨고 이벤트가 발생하면 서버에서 데이터를 가져와 화면을 변경하는 페이지

requests 이용

- 받아오는 문자열에 따라 두가지 방법으로 구분
  - json 문자열로 받아서 파싱하는 방법 : 주로 동적 페이지 크롤링할때 사용
  - html 문자열로 받아서 파싱하는 방법 : 주로 정적 페이지 크롤링할때 사용

selenium 이용

- 브라우저를 직접 열어서 데이터를 받는 방법

크롤링 방법에 따른 속도

- requests json > requests html > selenium

Crawling Naver Stock Data

- 네이버 증권 사이트에서 주가 데이터 수집
- 수집할 데이터 : 일별 kospi, kosdaq 주가, 일별 환율(exchange rate) 데이터
- 데이터 수집 절차
  - 웹서비스 분석 : url
  - 서버에 데이터 요청 : request(url) > response : json(str)
  - 서버에서 받은 데이터 파싱(데이터 형태를 변경) : json(str) > list, dict > DataFrame

In [193…

```
import warnings
warnings.filterwarnings('ignore') # 경고 문구 안뜨게 해주는 설정
import requests
import pandas as pd
```

1. 웹서비스 분석 : url

- pc 웹페이지가 복잡하면 mobile 웹페이지에서 수집

In [194…

```
page_size, page = 30, 1
url = f'https://m.stock.naver.com/api/index/KOSPI/price?pageSize={page_size}&page={page}'
```

```
url
```

Out[194]:
```
'https://m.stock.naver.com/api/index/KOSPI/price?pageSize=30&page=1'
```

2. 서버에 데이터 요청 : request(url) > response : json(str)

- response의 status code가 200이 나오는지 확인
- 403이나 500이 나오면 request가 잘못되거나 web server에서 수집이 안되도록 설정이 된것임
  - header 설정 또는 selenium 사용
- 200이 나오더라도 response 안에 있는 내용을 확인 > 확인하는 방법 : response.text

In [195…
```
response =requests.get(url)
response
```

Out[195]:
```
<Response [200]>
```

In [196…
```
response.text[:500]
```

Out[196]:
```
'[{"localTradedAt":"2024-03-18","closePrice":"2,685.84","compareToPreviousClosePrice":"19.0
0","compareToPreviousPrice":{"code":"2","text":"상승","name":"RISING"},"fluctuationsRatio":"0.
71","openPrice":"2,678.52","highPrice":"2,688.07","lowPrice":"2,665.28"},{"localTradedAt":"202
4-03-15","closePrice":"2,666.84","compareToPreviousClosePrice":"-51.92","compareToPreviousPric
e":{"code":"5","text":"하락","name":"FALLING"},"fluctuationsRatio":"-1.91","openPrice":"2,701.
91","highPrice":"2,705.59","lowPrice":'
```

3. 서버에서 받은 데이터 파싱(데이터 형태를 변경) : json(str) > list, dict > DataFrame

In [197…
```
data = response.json() #리스트로 바꿔줌
type(data)
```

Out[197]:
```
list
```

In [198…
```
cols = ['localTradedAt', 'closePrice']
df = pd.DataFrame(data)[cols]
df.head()
```

Out[198]:

	localTradedAt	closePrice
0	2024-03-18	2,685.84
1	2024-03-15	2,666.84
2	2024-03-14	2,718.76
3	2024-03-13	2,693.57
4	2024-03-12	2,681.81

4. 함수로 만들기

In [199…
```
def stock_price(code='KOSPI', page_size=60, page=1):
    # 1. URL
    url = f'https://m.stock.naver.com/api/index/{code}/price?pageSize={page_size}&page={page}'
```

```
    # 2. request(url)
    response = requests.get(url)
    # 3. parsing json
    data = response.json()
    # 4. 데이터프레임
    cols = ['localTradedAt', 'closePrice']
    df = pd.DataFrame(data)[cols]
    #pd.DataFrame(response.json())[['localTradedAt', 'closePrice']] # 한줄로 줄일 수 있다

    return df
```

In [200…

```
stock_price().head()
```

Out[200]:

	localTradedAt	closePrice
0	2024-03-18	2,685.84
1	2024-03-15	2,666.84
2	2024-03-14	2,718.76
3	2024-03-13	2,693.57
4	2024-03-12	2,681.81

In [201…

```
dfs = []
for page in range(1, 10):
    df = stock_price(page=page)
    dfs.append(df)

df = pd.concat(dfs, ignore_index=True)
df
```

Out[201]:

	localTradedAt	closePrice
0	2024-03-18	2,685.84
1	2024-03-15	2,666.84
2	2024-03-14	2,718.76
3	2024-03-13	2,693.57
4	2024-03-12	2,681.81
…	…	…
535	2022-01-12	2,972.48
536	2022-01-11	2,927.38
537	2022-01-10	2,926.72
538	2022-01-07	2,954.89
539	2022-01-06	2,920.53

540 rows × 2 columns

In [202…

```
df = stock_price(code='KOSDAQ')
df
```

Out[202]:

	localTradedAt	closePrice
0	2024-03-18	894.48
1	2024-03-15	880.46
2	2024-03-14	887.52
3	2024-03-13	889.93
4	2024-03-12	889.71
5	2024-03-11	875.93
6	2024-03-08	873.18
7	2024-03-07	863.37
8	2024-03-06	870.67
9	2024-03-05	866.37
10	2024-03-04	872.97
11	2024-02-29	862.96
12	2024-02-28	863.39
13	2024-02-27	853.75
14	2024-02-26	867.40
15	2024-02-23	868.57
16	2024-02-22	870.11
17	2024-02-21	864.07
18	2024-02-20	866.17
19	2024-02-19	858.47
20	2024-02-16	857.60
21	2024-02-15	859.21
22	2024-02-14	853.30
23	2024-02-13	845.15
24	2024-02-08	826.58
25	2024-02-07	811.92
26	2024-02-06	807.03
27	2024-02-05	807.99
28	2024-02-02	814.77
29	2024-02-01	798.73
30	2024-01-31	799.24
31	2024-01-30	818.86
32	2024-01-29	819.14
33	2024-01-26	837.24

	localTradedAt	closePrice
34	2024-01-25	823.74
35	2024-01-24	836.21
36	2024-01-23	840.11
37	2024-01-22	839.69
38	2024-01-19	842.67
39	2024-01-18	840.33
40	2024-01-17	833.05
41	2024-01-16	854.83
42	2024-01-15	859.71
43	2024-01-12	868.08
44	2024-01-11	882.53
45	2024-01-10	875.46
46	2024-01-09	884.64
47	2024-01-08	879.34
48	2024-01-05	878.33
49	2024-01-04	866.25
50	2024-01-03	871.57
51	2024-01-02	878.93
52	2023-12-28	866.57
53	2023-12-27	859.79
54	2023-12-26	848.34
55	2023-12-22	854.62
56	2023-12-21	859.44
57	2023-12-20	862.98
58	2023-12-19	858.30
59	2023-12-18	850.96

5. 원달러 환율 데이터 수집 : 실습

In [203…

```
# 1. URL
code, page, page_size= 'FX_USDKRW', 1, 30
url = f'https://m.stock.naver.com/front-api/v1/marketIndex/prices\
?category=exchange&reutersCode={code}&page={page}&pageSize={page_size}'
# 2. request > response
response = requests.get(url)
# 3. json(str) > DataFrame
data = response.json()['result']
```

```
usd_df = pd.DataFrame(data)[['localTradedAt', 'closePrice']]
usd_df.tail(2)
```

Out[203]:

	localTradedAt	closePrice
28	2024-02-02	1,338.50
29	2024-02-01	1,332.00

In [204…

```
print(url)
response = requests.get(url)
response.text[:300]
```

Out[204]:

```
https://m.stock.naver.com/front-api/v1/marketIndex/prices?category=exchange&reutersCode=FX_USD
KRW&page=1&pageSize=30
'{"isSuccess":true,"detailCode":"","message":"","result":[{"localTradedAt":"2024-03-18","close
Price":"1,334.00","fluctuations":"2.00","fluctuationsRatio":"0.15","fluctuationsType":{"cod
e":"2","text":"상승","name":"RISING"},"cashBuyValue":"1,357.34","cashSellValue":"1,310.66","se
ndValue":"1,347.0","rece'
```

In [205…

```
response.json()['result'][:1]
```

Out[205]:

```
[{'localTradedAt': '2024-03-18',
  'closePrice': '1,334.00',
  'fluctuations': '2.00',
  'fluctuationsRatio': '0.15',
  'fluctuationsType': {'code': '2', 'text': '상승', 'name': 'RISING'},
  'cashBuyValue': '1,357.34',
  'cashSellValue': '1,310.66',
  'sendValue': '1,347.0',
  'receiveValue': '1,321.0'}]
```

6. 시각화

In [206…

```
%matplotlib inline
%config InlineBackend.figure_formats = {'png', 'retina'}
```

In [207…

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [208…

```
kospi_df = stock_price(page_size=30)
kosdaq_df = stock_price(code='KOSDAQ', page_size=30)
```

In [209…

```
usd_df.tail(2)
```

Out[209]:

	localTradedAt	closePrice
28	2024-02-02	1,338.50
29	2024-02-01	1,332.00

In [210…

```
kospi_df.tail(2)
```

Out[210]:

	localTradedAt	closePrice
28	2024-02-02	2,615.31
29	2024-02-01	2,542.46

In [211…
```
kosdaq_df.tail(2)
```

Out[211]:

	localTradedAt	closePrice
28	2024-02-02	814.77
29	2024-02-01	798.73

In [212…
```
# 데이터 전처리 1 : 데이터 타입 변경
print(kospi_df.dtypes)
kospi_df["kospi"] = kospi_df["closePrice"].apply(lambda data: float(data.replace(",", "")))
kospi_df = kospi_df.drop(columns=["closePrice"])
print(kospi_df.dtypes)
```

```
localTradedAt    object
closePrice       object
dtype: object
localTradedAt     object
kospi            float64
dtype: object
```

In [213…
```
kosdaq_df["kosdaq"] = kosdaq_df["closePrice"].apply(lambda data: float(data.replace(",", "")))
usd_df["usd"] = usd_df["closePrice"].apply(lambda data: float(data.replace(",", "")))
```

In [214…
```
kosdaq_df = kosdaq_df.drop(columns=["closePrice"])
usd_df = usd_df.drop(columns=["closePrice"])
```

In [215…
```
merge_df_1 = pd.merge(kospi_df, kosdaq_df, on="localTradedAt")
merge_df_2 = pd.merge(merge_df_1, usd_df, on="localTradedAt")
merge_df = merge_df_2.copy()
merge_df.tail(2)
```

Out[215]:

	localTradedAt	kospi	kosdaq	usd
28	2024-02-02	2615.31	814.77	1338.5
29	2024-02-01	2542.46	798.73	1332.0

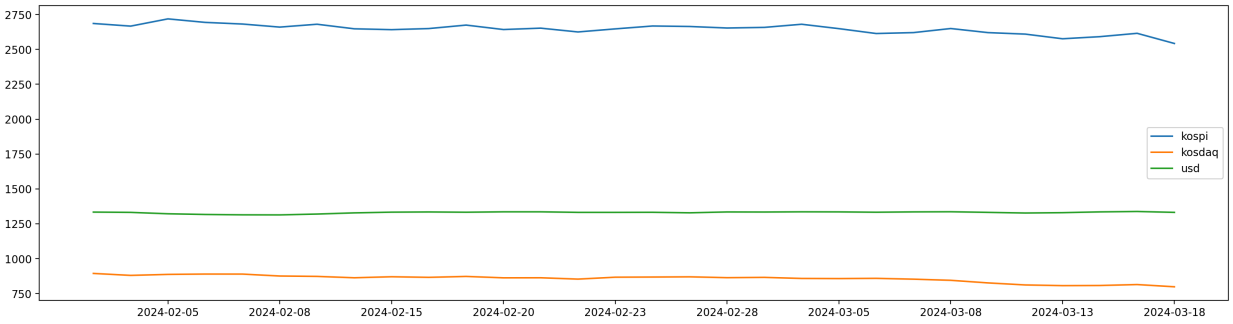
In [216…
```
# 시각화
plt.figure(figsize=(20, 5))

# plt.plot(merge_df["localTradedAt"], merge_df["kospi"], label="kospi")
# plt.plot(merge_df["localTradedAt"], merge_df["kosdaq"], label="kosdaq")
# plt.plot(merge_df["localTradedAt"], merge_df["usd"], label="usd")

columns = merge_df.columns[1:]
for column in columns:
    plt.plot(merge_df["localTradedAt"][::-1], merge_df[column], label=column)

xticks_count = 11
plt.xticks(merge_df["localTradedAt"][::int(len(merge_df) // xticks_count) + 1])
```

```
plt.legend(loc=0)
plt.show()
```



In [168…
```
# lambda 함수
# 일회용함수 : 간단한 라라미터와 리턴코드로 되어 있는 함수를 대체
# 함수, 변수 2개 선언 -> 메로리 2칸 사용
def plus(n1, n2):
    return n1 + n2

def calc(func, n1, n2):
    return func(n1, n2) # 콜백 함수

calc(plus, 1, 2)
```

Out[168]: 3

In [170…
```
plus_lambda = lambda n1, n2: n1+n2
plus_lambda(1, 2)
```

Out[170]: 3

In [171…
```
def calc(func, n1, n2):
    return func(n1, n2) # 콜백 함수

calc(lambda n1, n2: n1+n2, 1, 2)
```

Out[171]: 3

7. 데이터 스케일링

- min max scaling
- $z = \frac{x_i - min(x)}{max(x) - min(x)} (0 \leqq z \leqq 1)$
- latex syntax : `https://jjycjnmath.tistory.com/117`

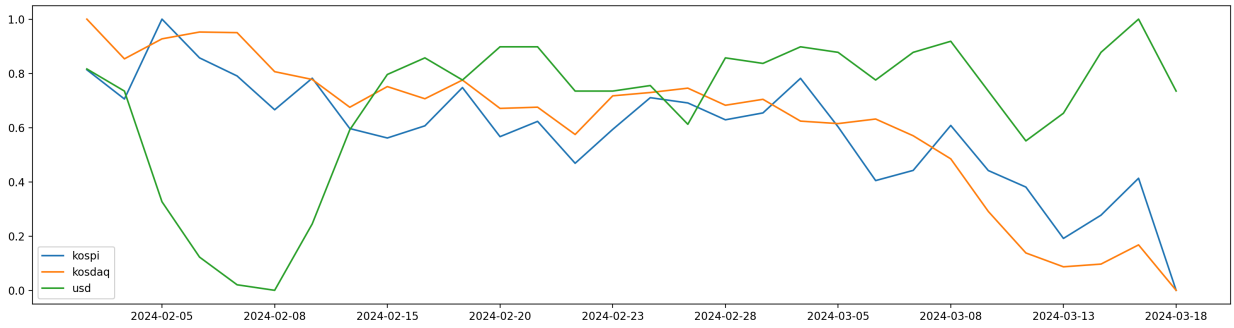
In [217…
```
from sklearn.preprocessing import minmax_scale
```

In [219…
```
# 시각화
plt.figure(figsize=(20, 5))

columns = merge_df.columns[1:]
for column in columns:
    plt.plot(merge_df["localTradedAt"][::-1], minmax_scale(merge_df[column]), label=column)

xticks_count = 11
```

```
plt.xticks(merge_df["localTradedAt"][::int(len(merge_df) // xticks_count) + 1])
plt.legend(loc=0)
plt.show()
```



8. 상관관계 분석

- 피어슨 상관계수(Pearson Correlation Coefficient)
- 두 데이터 집합의 상관도를 분석할때 사용되는 지표
- 상관계수의 해석
  - -1에 가까울수록 서로 반대방향으로 움직임
  - 1에 가까울수록 서로 같은방향으로 움직임
  - 0에 가까울수록 두 데이터는 관계가 없음

In [220…

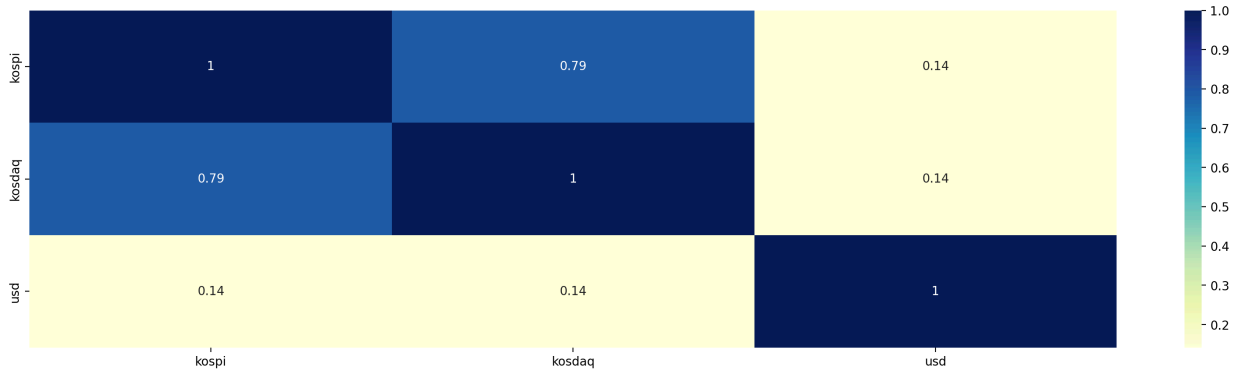
```
# 해석 1 : kospi, kosdaq은 아주 강한 양의 상관관계를 갖는다. (데이터가 같은 방향으로 움직임)
# 해석 2 : kospi와 usd를 강한 음의 상관관계를 갖는다. (데이터가 반대 방향으로 움직임)
corr_df = merge_df[merge_df.columns[1:]].corr()
corr_df
```

Out[220]:

	kospi	kosdaq	usd
kospi	1.000000	0.890528	-0.376217
kosdaq	0.890528	1.000000	-0.378020
usd	-0.376217	-0.378020	1.000000

In [221…

```
# 결정계수 : r-squared
# 1과 가까울수록 강한 관계, 0과 가까울수록 약한 관계
plt.figure(figsize=(20, 5))
sns.heatmap(corr_df**2, cmap="YlGnBu", annot=True)
plt.show()
```



In [ ]: