

Ch20 Web Scraping - \ 추출

웹 스크랩핑(Web Scraping) :\

위주로 추출

| 웹 페이지의 \

에 포함된 데이터들을 데이터 프레임으로 추출

In []:

[실습] 주식 시세 사이트에서 일일 주가 변동 데이터를 추출하여 그래프로 표현하기

> 네이버 주식 시세 관련 사이트에서 (주)NHN 주가 일일시세 변동 데이터 추출하기

> (주)NHN 주식 종목번호 code=181710

URL로 웹 페이지 열어보기

In [5]:

```
## URL로 웹 페이지 열어 보기
import webbrowser as wb

url = 'https://finance.naver.com/item/sise_day.naver?
code=181710&page=1'
wb.open_new(url) #브라우저에 사이트가 열림
```

Out[5]:

True

\

의 text를 데이터 프레임으로 추출

> 첫 \

태그 데이터만 추출

*필요 시 서버에게 Header 정보를 제공

| 서버에게 보내는 Header 정보는 클라이언트가 서버에게 (전

송) 요청과 관련된 추가 정보를 제공함으로서 \ 서버가 클라이언트에게 정보를 전송할 때 Header 정보를 사용하여 전송 하므로서 발생 가능한 문제를 제거함

```
In [1]: ##[NHN 일별 종가] Naver에서 네이버 (NHN: code=181710) 일별 종가 데이터 수집하기 > 내 브라우저 header 정보 미제공
import pandas as pd
import urllib3 #HTTP 클라이언트 구현 모듈

# 데이터를 읽어올 URL 주소 지정
url = 'https://finance.naver.com/item/sise_day.naver?code=181710&page=1' #(주)NHN 주식 종목번호 : 181710

# 웹 페이지에서 표 스크래핑
http = urllib3.PoolManager() #http나 https 연결 관리자 호출
req = http.request('GET', url) #http 연결 관리자로 html 문서 요청
tables = pd.read_html(req.data) #html 문서에서 테이블 태그 (<table> </table>) 부분의 값들을 list 형태(table 개수만큼)로 반환

tables[0] #데이터 프레임
```

ValueError

Traceback (most recent call last)

~\AppData\Local\Temp\ipykernel_11052\248405414.py in <module>

9 http = urllib3.PoolManager()
#http나 https 연결 관리자 호출

10 req = http.request('GET', url)
#http 연결 관리자로 html 문서 요청

---> 11 tables = pd.read_html(req.data)
#html 문서에서 테이블 태그(<table> </table>) 부분의 값들을 list 형태(table 개수 만큼)로 반환

12

13 tables[0] #데이터 프레임

~\anaconda3\lib\site-packages\pandas\io\html.py in read_html(io, match, flavor, header, i

```
ndex_col, skiprows, attrs, parse_dates, thousands, encoding,
decimal, converters, na_values, keep_default_na, displayed_only,
extract_links, dtype_backend, storage_options)
```

```
1243     )
```

```
1244
```

```
-> 1245     return _parse(
1246             flavor=flavor,
1247             io=io,
```

```
~\anaconda3\lib\site-packages
\pandas\io\html.py in _parse(f
lavor, io, match, attrs, encod
ing, displayed_only, extract_l
inks, storage_options, **kwargs
s)
```

```
1006     else:
```

```
1007             assert retained
d is not None # for mypy
```

```
-> 1008             raise retained
```

```
1009
```

```
1010     ret = []
```

```
~\anaconda3\lib\site-packages
\pandas\io\html.py in _parse(f
```

```
lavor, io, match, attrs, encoding, displayed_only, extract_links, storage_options, **kwargs)
    986
    987         try:
--> 988             tables =
p.parse_tables()
    989         except ValueError as caught:
    990             # if `io` is an io-like object, check if
it's seekable

~\anaconda3\lib\site-packages
\pandas\io\html.py in parse_table(self)
    246             list of parsed
(header, body, footer) tuples
from tables.

    247             """
--> 248             tables = self._parse_tables(self._build_doc(),
self.match, self.attrs)
    249             return (self._parse_thead_tbody_tfoot(table)
for table in tables)
```

```

~\anaconda3\lib\site-packages
\pandas\io\html.py in _parse_t
ables(self, document, match,
a
ttrs)
    601         tables = docum
ent.find_all(element_name, att
rs=attrs)
    602         if not tables:
--> 603             raise Valu
eError("No tables found")
    604
    605         result = []

```

ValueError: No tables found

보다 정확한 통신을 위해 서버에게 내 브라우저의 Header 정보를 제공

서버에게 보내는 Header 정보는 클라이언트가 서버에게 (전송)요청과 관련된 추가 정보를 제공함으로서 서버가 클라이언트에게 정보를 전송할 때 Header 정보를 사용하여 전송하므로서 발생 가능한 문제를 제거함

내 브라우저 Header 정보 찾기

> 내 브라우저의 웹 페이지에서 오른쪽 버튼을 클릭하여 '검사'를 선택

개발자 도구의 상단 메뉴에서 "Network" 탭을 선택 웹 페이지를 새로고침하면 네트워크 탭에서 모든 네트워크 요청이 기록됨 "Headers" 탭을 선택하여 'Request Headers'를 클릭 하단의 'User-Agent:' 부분을 복사하여 사용

In [2]: `##[NHN 일별 종가] Naver에서 네이버 (NHN: code=181710) 일별 종가 데이터 수집하기 > 내 브라우저 header 정보 제공`

```
import pandas as pd
```

```

import urllib3 #HTTP 클라이언트 구현 모듈

# 데이터를 읽어올 URL 주소 지정
url = 'https://finance.naver.com/item/sise_day.naver?code=181710&page=1' #(주)NHN 주식 종목번호 : 181710

# HTTP 요청 헤더 정보 설정: 클라이언트가 서버에게 요청과 관련된
# 추가 정보를 제공하는 데 사용
header = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110
Safari/537.3',
    'Referer': url
}

# 웹 페이지에서 표 스크래핑
http = urllib3.PoolManager() #http나 https 연결 관리자 호출
req = http.request('GET', url, headers=header) #http 연결 관리자
#로 html 문서 요청
tables = pd.read_html(req.data) #html 문서에서 테이블 태그
#(<table> </table>) 부분의 값을 List 형태(table 개수만큼)로 반환

tables[0] #데이터 프레임

```

Out[2]:

	날짜	종가	전일비	시가	고가	저가	거래량
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	2023.12.04	23250.0	100.0	23300.0	23850.0	22750.0	31532.0
2	2023.12.01	23150.0	350.0	23350.0	23650.0	23150.0	30090.0
3	2023.11.30	23500.0	500.0	23050.0	23500.0	22900.0	42676.0
4	2023.11.29	23000.0	100.0	23000.0	23600.0	22850.0	93358.0
5	2023.11.28	22900.0	100.0	23100.0	23300.0	22850.0	31122.0
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	2023.11.27	23000.0	100.0	22900.0	23450.0	22850.0	38020.0
10	2023.11.24	22900.0	100.0	22850.0	23250.0	22800.0	32787.0
11	2023.11.23	22800.0	0.0	22800.0	23100.0	22750.0	30115.0
12	2023.11.22	22800.0	0.0	22750.0	23050.0	22500.0	27271.0
13	2023.11.21	22800.0	100.0	22850.0	22950.0	22650.0	27319.0
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN

추출된 데이터 조정 작업

불필요한 결측치 행 제거

```
In [3]: ## tables[0]를 df로 깊은 복사  
df = tables[0].copy()
```

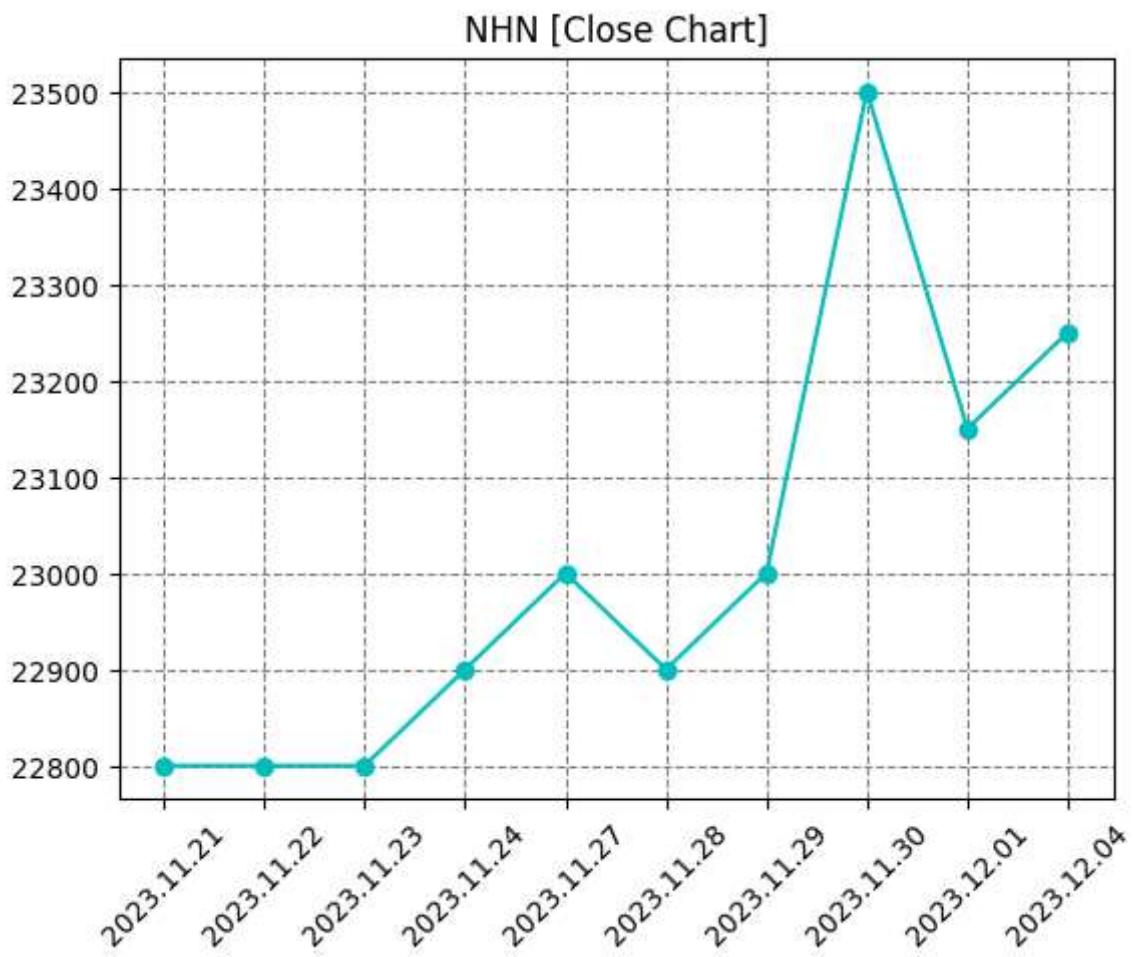
```
In [4]: ## 결측치 제거 후 날짜 오름차순으로 정렬  
df = df.dropna() #결측치(결함값) 포함 행 삭제  
df = df.sort_values(by='날짜')  
df
```

Out[4]:

	날짜	종가	전일비	시가	고가	저가	거래량
13	2023.11.21	22800.0	100.0	22850.0	22950.0	22650.0	27319.0
12	2023.11.22	22800.0	0.0	22750.0	23050.0	22500.0	27271.0
11	2023.11.23	22800.0	0.0	22800.0	23100.0	22750.0	30115.0
10	2023.11.24	22900.0	100.0	22850.0	23250.0	22800.0	32787.0
9	2023.11.27	23000.0	100.0	22900.0	23450.0	22850.0	38020.0
5	2023.11.28	22900.0	100.0	23100.0	23300.0	22850.0	31122.0
4	2023.11.29	23000.0	100.0	23000.0	23600.0	22850.0	93358.0
3	2023.11.30	23500.0	500.0	23050.0	23500.0	22900.0	42676.0
2	2023.12.01	23150.0	350.0	23350.0	23650.0	23150.0	30090.0
1	2023.12.04	23250.0	100.0	23300.0	23850.0	22750.0	31532.0

일일 주가변동 선 그래프 그리기

```
In [5]: ## 데이터프레임으로 주가변동 차트 그리기  
import matplotlib.pyplot as plt  
  
plt.title('NHN [Close Chart]')  
plt.xticks(rotation=45) #x-레이블 회전  
plt.plot(df['날짜'], df['종가'], 'co-')  
plt.grid(color='gray', linestyle='--')  
plt.show()
```



In []:

[Upgrade] NHN 일별 종가 데이터 추출 (전체 페이지 데이터 추출하기)

> 모든 페이지의 데이터를 추출하여 분석하기

HTML 추출하기

전체 페이지를 구하기 위해 \ 전체 피이지는 전체 \

을 반복문으로 추출할 때 사용

```
In [6]: ## Naver에서 (주)NHN 일별시세 추가 페이지 html 가져오기
from bs4 import BeautifulSoup
from urllib.request import urlopen, Request

url = 'https://finance.naver.com/item/sise_day.naver?code=181710&page=1'

# HTTP 요청 헤더 정보 설정
header = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3',
    'Referer': 'https://finance.naver.com/item/sise_day.naver?'
}
```

```

# 웹 페이지 요청 및 응답 데이터 읽기
req = Request(url, headers=header) #http 연결 요청
html = urlopen(req).read() #http 연결 정보로 html 문서 요청

# 파싱
soup = BeautifulSoup(html, 'lxml') #Parsing

```

맨 뒤 페이지 번호 찾기

```

In [7]: ## html에서 맨 뒤 페이지 구하기
pgrr = soup.find('td', {'class':'pgRR'}) # {'속성': '속성값'} 검색
s = str(pgrr.a['href']).split('=') #문자열에서 '=' 기준으로 분리하여 리스트로 반환
last_pg = s[-1] #리스트에서 맨 뒤 값
last_pg

Out[7]: '253'

In [8]: pgrr.a['href']

Out[8]: '/item/sise_day.naver?code=181710&page=253'

```

첫 페이지에서 \

표 스크래핑

```

In [9]: ## 첫 페이지에서 표 스크래핑
http = urllib3.PoolManager() #http나 https 연결 관리자 호출
req = http.request('GET', url, headers=header) #http 연결 관리자로 html 문서 요청
tables = pd.read_html(req.data) #html 문서에서 테이블 태그(<table> </table>) 부분의 값들을 list 형태(table 개수만큼)로 반환
df = tables[0].copy() #데이터 프레임 df

```

Out[9]:

	날짜	종가	전일비	시가	고가	저가	거래량
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	2023.12.04	23200.0	50.0	23300.0	23850.0	22750.0	31534.0
2	2023.12.01	23150.0	350.0	23350.0	23650.0	23150.0	30090.0
3	2023.11.30	23500.0	500.0	23050.0	23500.0	22900.0	42676.0
4	2023.11.29	23000.0	100.0	23000.0	23600.0	22850.0	93358.0
5	2023.11.28	22900.0	100.0	23100.0	23300.0	22850.0	31122.0
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN

	날짜	종가	전일비	시가	고가	저가	거래량
9	2023.11.27	23000.0	100.0	22900.0	23450.0	22850.0	38020.0
10	2023.11.24	22900.0	100.0	22850.0	23250.0	22800.0	32787.0
11	2023.11.23	22800.0	0.0	22800.0	23100.0	22750.0	30115.0
12	2023.11.22	22800.0	0.0	22750.0	23050.0	22500.0	27271.0
13	2023.11.21	22800.0	100.0	22850.0	22950.0	22650.0	27319.0
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN

각 페이지에서 데이터를 추출하여 데이터 프레임에 concat()로 행 추가하기

```
In [15]: ## Naver에서 (주)NHN 일별 종가 추이 데이터프레임으로 구성하기
import urllib3 #HTTP 클라이언트 구현 모듈
import pandas as pd

df = pd.DataFrame()
url = 'https://finance.naver.com/item/sise_day.naver?code=181710'

for pg in range(1, int(last_pg)+1): # page 1에서 맨 끝 페이지까지 페이지 수 pg를 구
    pg_url = '{}&page={}'.format(url, pg) # '&page=' 문자열 구성
    http = urllib3.PoolManager() #http 연결 관리자 호출
    req = http.request('GET', pg_url, headers=header) #http 연결 관리자로 html 문서
    tables = pd.read_html(req.data) #html 문서에서 테이블 태그(<table> </table>) 데
    df = pd.concat([tables[0], df]) #데이터프레임에 추가
df
```

Out[15]:

	날짜	종가	전일비	시가	고가	저가	거래량
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	2013.09.03	110000.0	4000.0	114500.0	115000.0	110000.0	607923.0
2	2013.09.02	114000.0	5500.0	110000.0	114500.0	108000.0	1704882.0
3	2013.08.30	108500.0	19000.0	108500.0	111500.0	108500.0	5126731.0
4	2013.08.29	127500.0	22000.0	149500.0	149500.0	127500.0	33359.0
...
10	2023.11.24	22900.0	100.0	22850.0	23250.0	22800.0	32787.0
11	2023.11.23	22800.0	0.0	22800.0	23100.0	22750.0	30115.0
12	2023.11.22	22800.0	0.0	22750.0	23050.0	22500.0	27271.0
13	2023.11.21	22800.0	100.0	22850.0	22950.0	22650.0	27319.0
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN

3795 rows × 7 columns

데이터 프레임 후반 조정 작업

```
In [16]: ## 결측치 확인
df.isna().sum()
```

```
Out[16]: 날짜      1271  
종가      1271  
전일비    1271  
시가      1271  
고가      1271  
저가      1271  
거래량    1271  
dtype: int64
```

```
In [17]: ## 데이터프레임 결측치(결함값) 포함 행 삭제  
df = df.dropna() #결측치(결함값) 포함 행 삭제  
df
```

```
Out[17]:
```

	날짜	종가	전일비	시가	고가	저가	거래량
1	2013.09.03	110000.0	4000.0	114500.0	115000.0	110000.0	607923.0
2	2013.09.02	114000.0	5500.0	110000.0	114500.0	108000.0	1704882.0
3	2013.08.30	108500.0	19000.0	108500.0	111500.0	108500.0	5126731.0
4	2013.08.29	127500.0	22000.0	149500.0	149500.0	127500.0	33359.0
1	2013.09.17	109500.0	1500.0	111000.0	112500.0	109000.0	276014.0
...
9	2023.11.27	23000.0	100.0	22900.0	23450.0	22850.0	38020.0
10	2023.11.24	22900.0	100.0	22850.0	23250.0	22800.0	32787.0
11	2023.11.23	22800.0	0.0	22800.0	23100.0	22750.0	30115.0
12	2023.11.22	22800.0	0.0	22750.0	23050.0	22500.0	27271.0
13	2023.11.21	22800.0	100.0	22850.0	22950.0	22650.0	27319.0

2524 rows × 7 columns

```
In [20]: ## 데이터프레임 30개 데이터 행만 선택 : 날짜 오름차순으로 정렬하여 재구성  
df_new = df.iloc[0:30] #30개 행만 선택 사용  
df_new = df_new.sort_values(by='날짜')  
df_new
```

Out[20]:

	날짜	종가	전일비	시가	고가	저가	거래량
4	2013.08.29	127500.0	22000.0	149500.0	149500.0	127500.0	33359.0
3	2013.08.30	108500.0	19000.0	108500.0	111500.0	108500.0	5126731.0
2	2013.09.02	114000.0	5500.0	110000.0	114500.0	108000.0	1704882.0
1	2013.09.03	110000.0	4000.0	114500.0	115000.0	110000.0	607923.0
13	2013.09.04	109000.0	1000.0	109000.0	112500.0	108500.0	324499.0
12	2013.09.05	109000.0	0.0	109500.0	110500.0	107000.0	552482.0
11	2013.09.06	104000.0	5000.0	109000.0	110500.0	103000.0	538761.0
10	2013.09.09	107000.0	3000.0	104500.0	108000.0	102500.0	460874.0
9	2013.09.10	114500.0	7500.0	108500.0	116000.0	107500.0	813503.0
5	2013.09.11	113500.0	1000.0	114000.0	115000.0	112000.0	384258.0
4	2013.09.12	116000.0	2500.0	116000.0	118500.0	114000.0	731557.0
3	2013.09.13	115000.0	1000.0	114500.0	116500.0	113000.0	249092.0
2	2013.09.16	111000.0	4000.0	116000.0	116500.0	110500.0	299952.0
1	2013.09.17	109500.0	1500.0	111000.0	112500.0	109000.0	276014.0
13	2013.09.23	108500.0	1000.0	110000.0	114000.0	108500.0	437542.0
12	2013.09.24	111500.0	3000.0	109000.0	112000.0	108500.0	287737.0
11	2013.09.25	115500.0	4000.0	112000.0	116500.0	111000.0	496101.0
10	2013.09.26	122000.0	6500.0	116500.0	126500.0	115500.0	967404.0
9	2013.09.27	121500.0	500.0	122000.0	122500.0	119000.0	324156.0
5	2013.09.30	118000.0	3500.0	120500.0	122500.0	118000.0	290821.0
4	2013.10.01	117500.0	500.0	118500.0	121000.0	115500.0	268328.0
3	2013.10.02	124000.0	6500.0	118000.0	124500.0	118000.0	457264.0
2	2013.10.04	120000.0	4000.0	123500.0	125500.0	118500.0	418153.0
1	2013.10.07	114000.0	6000.0	119500.0	120500.0	114000.0	364640.0
9	2013.10.15	110500.0	2000.0	109500.0	112000.0	109000.0	316911.0
5	2013.10.16	101000.0	9500.0	109500.0	111500.0	99500.0	1409489.0
4	2013.10.17	100000.0	1000.0	102000.0	103500.0	100000.0	417219.0
3	2013.10.18	102500.0	2500.0	100500.0	104000.0	100500.0	365678.0
2	2013.10.21	106500.0	4000.0	103500.0	108000.0	102500.0	328227.0
1	2013.10.22	110000.0	3500.0	107000.0	111000.0	104500.0	438671.0

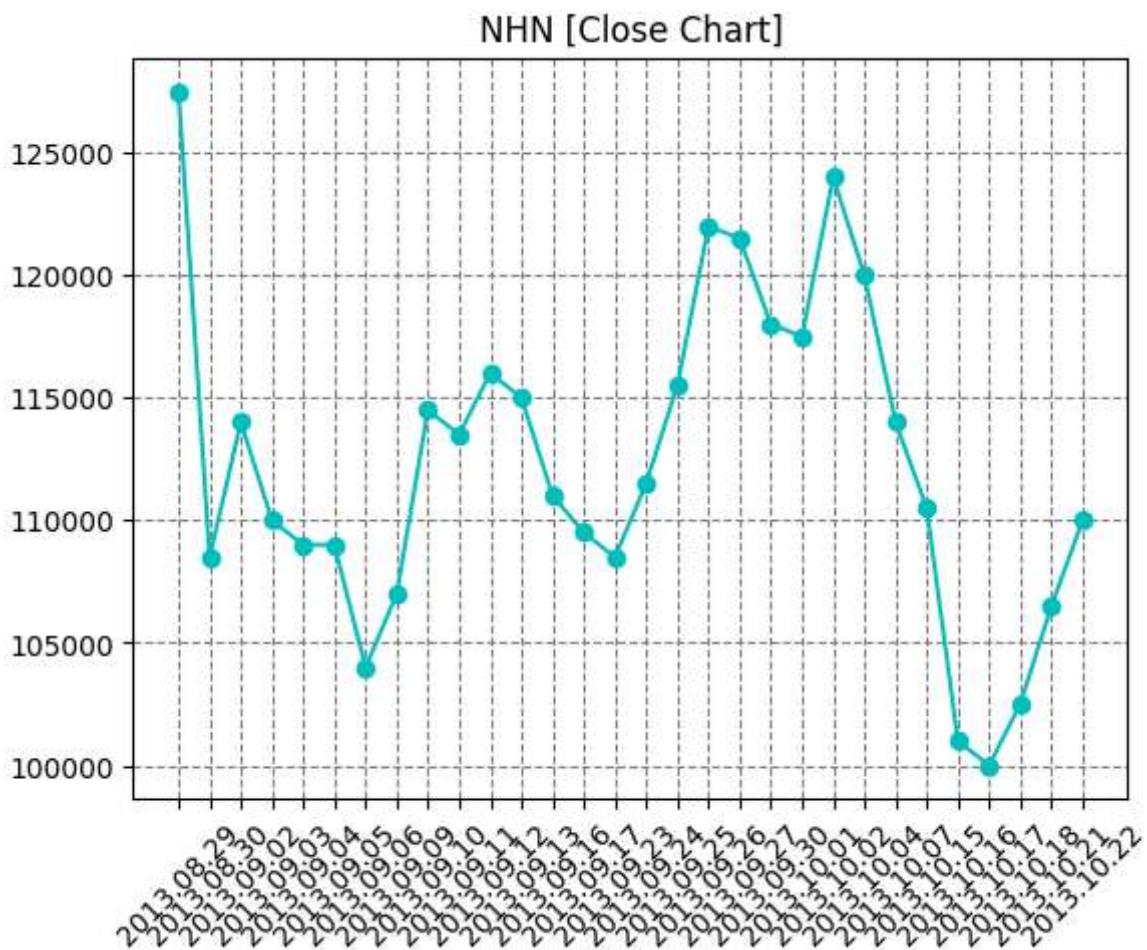
일일 주가 변동 선 그래프 그리기

In [21]:

```
## 데이터프레임으로 차트 그래프 그리기
import matplotlib.pyplot as plt

plt.title('NHN [Close Chart]')
plt.xticks(rotation=45) #x-레이블 회전
plt.plot(df_new['날짜'], df_new['종가'], 'co-')
```

```
plt.grid(color='gray', linestyle='--')
plt.show()
```



```

url = 'https://finance.naver.com/item/sise_day.naver?code=181710'

for pg in range(1, int(last_pg)+1): # page 1에서 맨 끝 페이지까지 페이지 수 pg를 구
    pg_url = '{}&page={}'.format(url, pg) # '&page=' 문자열 구성
    http = urllib3.PoolManager() #http 연결 관리자 호출
    req = http.request('GET', pg_url, headers=header) #http 연결 관리자로 html 문서
    tables = pd.read_html(req.data) #html 문서에서 테이블 태그(<table> </table>) 데
    df = pd.concat([tables[0], df]) #데이터프레임에 추가
df

```

Out[22]:

	날짜	종가	전일비	시가	고가	저가	거래량
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	2013.09.03	110000.0	4000.0	114500.0	115000.0	110000.0	607923.0
2	2013.09.02	114000.0	5500.0	110000.0	114500.0	108000.0	1704882.0
3	2013.08.30	108500.0	19000.0	108500.0	111500.0	108500.0	5126731.0
4	2013.08.29	127500.0	22000.0	149500.0	149500.0	127500.0	33359.0
...
10	2023.11.24	22900.0	100.0	22850.0	23250.0	22800.0	32787.0
11	2023.11.23	22800.0	0.0	22800.0	23100.0	22750.0	30115.0
12	2023.11.22	22800.0	0.0	22750.0	23050.0	22500.0	27271.0
13	2023.11.21	22800.0	100.0	22850.0	22950.0	22650.0	27319.0
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN

3795 rows × 7 columns

In [23]:

```

## 데이터프레임 결측치(결함값) 포함 행 삭제, 재구성
df = df.dropna() #결측치(결함값) 포함 행 삭제
df = df.iloc[0:30] #30개 행만 선택 사용
df = df.rename(columns={'날짜':'Date', '시가':'Open', '고가':'High', '저가':'Low', '거래량':'Volume'})
df = df.sort_values(by='Date')
df

```

Out[23]:

	Date	Close	전일비	Open	High	Low	Volumn
4	2013.08.29	127500.0	22000.0	149500.0	149500.0	127500.0	33359.0
3	2013.08.30	108500.0	19000.0	108500.0	111500.0	108500.0	5126731.0
2	2013.09.02	114000.0	5500.0	110000.0	114500.0	108000.0	1704882.0
1	2013.09.03	110000.0	4000.0	114500.0	115000.0	110000.0	607923.0
13	2013.09.04	109000.0	1000.0	109000.0	112500.0	108500.0	324499.0
12	2013.09.05	109000.0	0.0	109500.0	110500.0	107000.0	552482.0
11	2013.09.06	104000.0	5000.0	109000.0	110500.0	103000.0	538761.0
10	2013.09.09	107000.0	3000.0	104500.0	108000.0	102500.0	460874.0
9	2013.09.10	114500.0	7500.0	108500.0	116000.0	107500.0	813503.0
5	2013.09.11	113500.0	1000.0	114000.0	115000.0	112000.0	384258.0
4	2013.09.12	116000.0	2500.0	116000.0	118500.0	114000.0	731557.0
3	2013.09.13	115000.0	1000.0	114500.0	116500.0	113000.0	249092.0
2	2013.09.16	111000.0	4000.0	116000.0	116500.0	110500.0	299952.0
1	2013.09.17	109500.0	1500.0	111000.0	112500.0	109000.0	276014.0
13	2013.09.23	108500.0	1000.0	110000.0	114000.0	108500.0	437542.0
12	2013.09.24	111500.0	3000.0	109000.0	112000.0	108500.0	287737.0
11	2013.09.25	115500.0	4000.0	112000.0	116500.0	111000.0	496101.0
10	2013.09.26	122000.0	6500.0	116500.0	126500.0	115500.0	967404.0
9	2013.09.27	121500.0	500.0	122000.0	122500.0	119000.0	324156.0
5	2013.09.30	118000.0	3500.0	120500.0	122500.0	118000.0	290821.0
4	2013.10.01	117500.0	500.0	118500.0	121000.0	115500.0	268328.0
3	2013.10.02	124000.0	6500.0	118000.0	124500.0	118000.0	457264.0
2	2013.10.04	120000.0	4000.0	123500.0	125500.0	118500.0	418153.0
1	2013.10.07	114000.0	6000.0	119500.0	120500.0	114000.0	364640.0
9	2013.10.15	110500.0	2000.0	109500.0	112000.0	109000.0	316911.0
5	2013.10.16	101000.0	9500.0	109500.0	111500.0	99500.0	1409489.0
4	2013.10.17	100000.0	1000.0	102000.0	103500.0	100000.0	417219.0
3	2013.10.18	102500.0	2500.0	100500.0	104000.0	100500.0	365678.0
2	2013.10.21	106500.0	4000.0	103500.0	108000.0	102500.0	328227.0
1	2013.10.22	110000.0	3500.0	107000.0	111000.0	104500.0	438671.0

[사전 준비] pip install mplfinance로 라이브러리 설치 필요

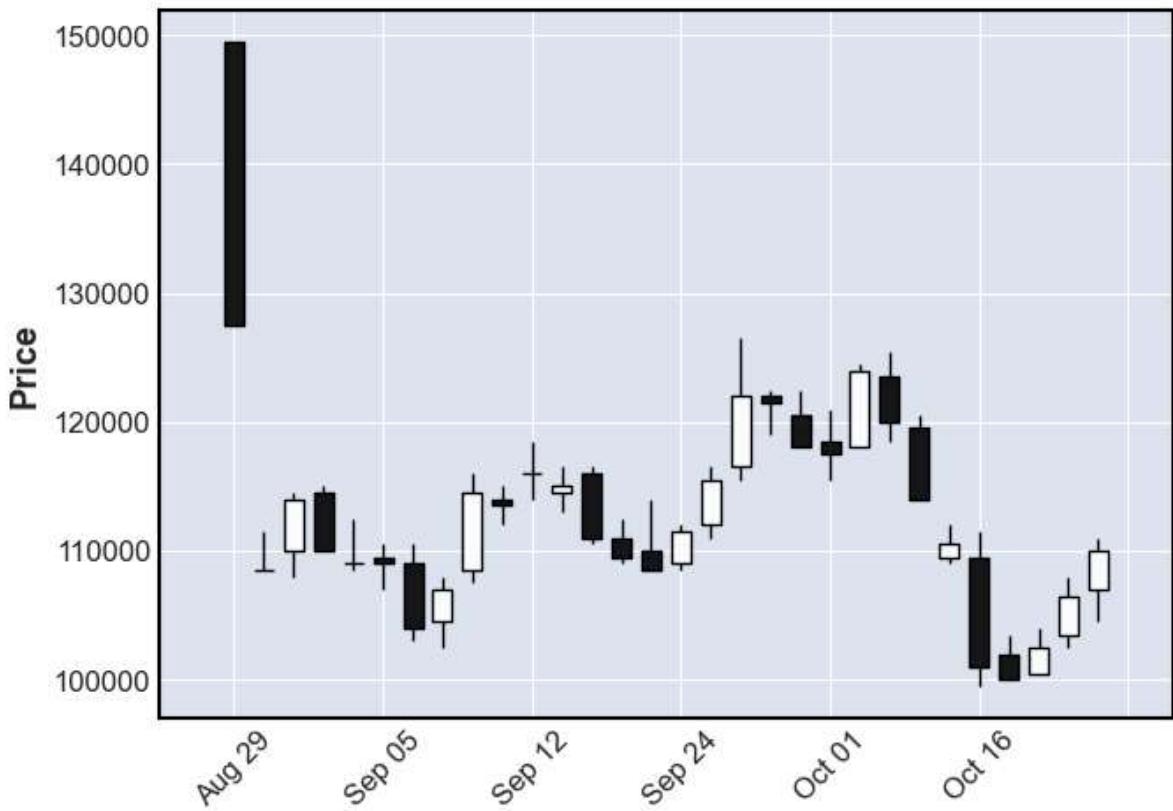
In [24]:

```
## 데이터프레임으로 차트 그래프 그리기 : type='candle'
import mplfinance as mpf #pip install mplfinance로 설치 필요

df.index = pd.to_datetime(df.Date)
df_new = df[['Date', 'Open', 'High', 'Low', 'Close', 'Volumn']]
```

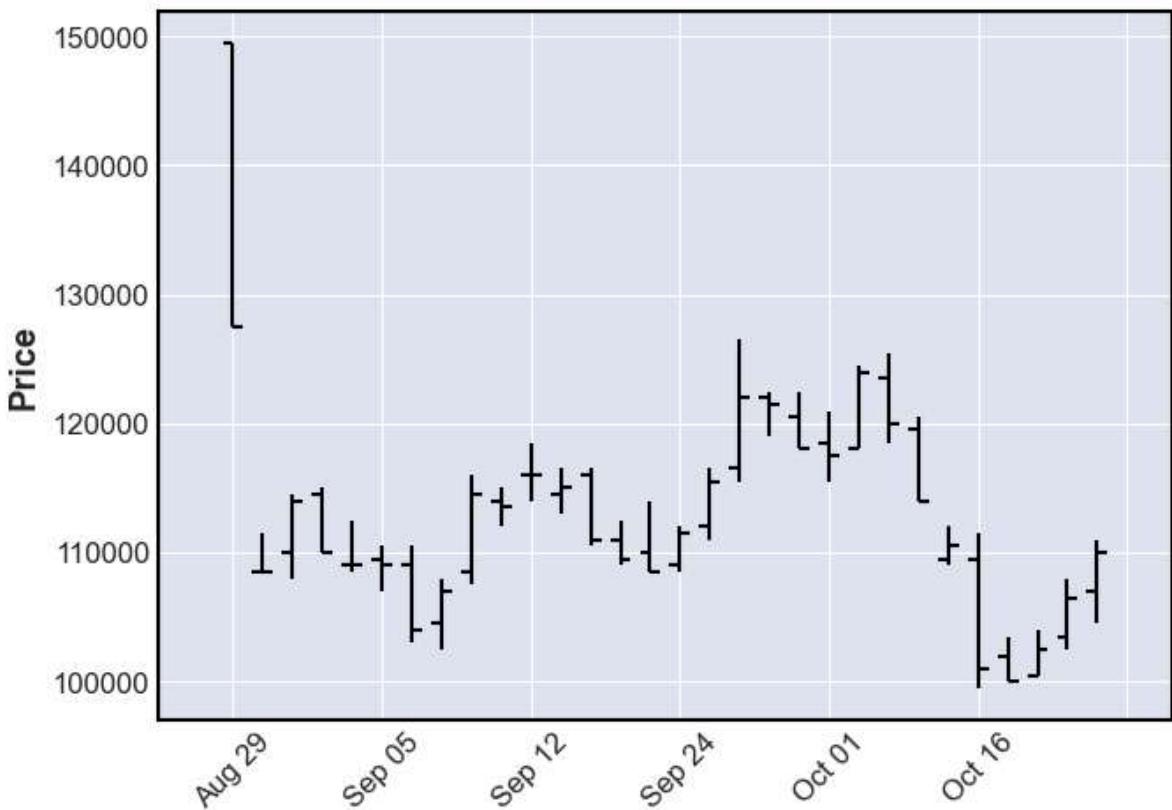
```
titles = 'NHN [Candle Chart]'  
mpf.plot(df_new, title=titles, type='candle')
```

NHN [Candle Chart]



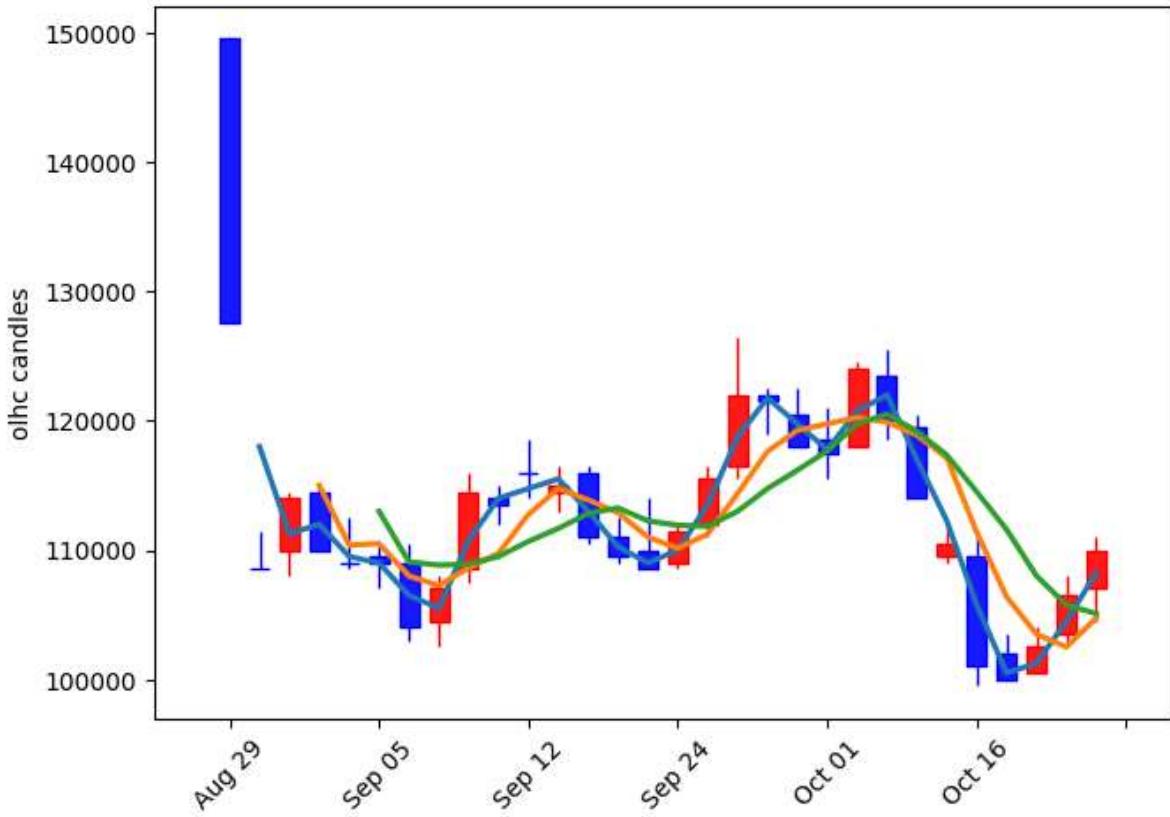
```
In [25]: ## 데이터프레임으로 차트 그래프 그리기 : type='ohlc'  
import mplfinance as mpf #pip install mplfinance 로 설치 필요  
  
titles = 'NHN [OHLC Candle Chart]'  
mpf.plot(df_new, title=titles, type='ohlc')
```

NHN [OLHC Candle Chart]



```
In [26]: ## 데이터프레임으로 차트 그래프 그리기 : type='candle' + 'OLHC'
import mplfinance as mpf #pip install mplfinance 로 설치 필요
titles = 'NHN [OLHC Candle Chart]'
mc = mpf.make_marketcolors(up='red', down='blue', inherit=True) #Market 색상 객체
s = mpf.make_mpf_style(marketcards=mc) #Market 스타일 객체 생성
kwargs = dict(title=titles, type='candle', mav=(2, 4, 6), ylabel='olhc candles', style=s)
mpf.plot(df_new, **kwargs)
```

NHN [OLHC Candle Chart]



In []: