

주식 시세 현황 분석



소프트웨어융합대학원
진혜진



1. 프로젝트 개요

2. 웹 기본 지식 이해하기

3. 웹 스크래핑

4. 데이터 수집 및 가공

5. 데이터 시각화



데이터 수집 및 가공

1. 웹 스크래핑 가능 유무 확인하기

■ 웹 스크래핑으로 데이터 수집하기

- 개발자 도구에서 확보한 웹 주소 → 웹 브라우저 주소 창에 입력해 결과를 확인
→ 다른 정보는 없이 일별 시세 정보만 확인이 가능함

일별시세						
날짜	종가	전일비	시가	고가	저가	거래량
2025.04.03	57,600	▼ 1,200	56,900	57,800	56,900	19,508,076
2025.04.02	58,800	— 0	58,800	59,400	58,300	14,402,297
2025.04.01	58,800	▲ 1,000	58,700	59,600	57,900	13,841,706
2025.03.31	57,800	▼ 2,400	59,500	59,700	57,800	17,633,494
2025.03.28	60,200	▼ 1,600	60,700	61,100	60,000	16,282,514
2025.03.27	61,800	▲ 400	60,900	62,000	60,800	20,389,790
2025.03.26	61,400	▲ 1,600	59,800	61,400	59,700	16,431,645
2025.03.25	59,800	▼ 700	60,900	61,100	59,500	17,259,455
2025.03.24	60,500	▼ 1,200	61,200	61,600	60,500	14,088,094
2025.03.21	61,700	▲ 1,500	60,900	61,700	60,400	40,155,612
« 맨앞 1 2 3 4 5 6 7 8 9 10 다음 » 맨뒤 »						

- 웹 브라우저에서 데이터를 확인할 수 있다고 해서 무조건 웹 스크래핑을 할 수 있는 것은 아님
✓ 확보한 데이터 수집용 URL이 파이썬 소스 코드에서 사용할 수 있는 주소인지 확인해야 함

1. 웹 스크래핑 가능 유무 확인하기

■ 웹 브라우저 없이 웹 정보 확인하기



```
# 웹 스크래핑 가능 유무 확인을 위한 모듈 탑재
import requests

# 웹 스크래핑 대상 주소 저장
url = 'https://finance.naver.com/item/sise_day.naver?code=005930&page=2'

# HTTP 요청을 대상 주소로 보내고, HTTP 응답 받아서 저장하기
page = requests.get(url)

# 응답 내용 확인하기
print(page.text)
```

1. 웹 스크래핑 가능 유무 확인하기



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
<title>네이버 :: 세상의 모든 지식, 네이버</title>

<style type="text/css">
.error_content * {margin:0;padding:0;}
.error_content img{border:none;}
.error_content em {font-style:normal;}
.error_content {width:410px; margin:80px auto 0; padding:57px 0 0 0; font-size:12px; font-family:"나눔고딕", "NanumGothic", "돋움";}
.error_content p{margin:0;}
.error_content .error_desc {margin-bottom:21px; overflow:hidden; text-align:center;}
.error_content .error_desc2 {margin-bottom:11px; padding-bottom:7px; color:#888; line-height:18px; border-bottom:1px solid #eee;}
.error_content .error_desc3 {clear:both; color:#888;}
.error_content .error_desc3 a {color:#004790; text-decoration:underline;}
.error_content .error_list_type {clear:both; float:left; width:410px; _width:428px; margin:0 0 18px 0; *margin:0 0 7px 0; padding:0;}
.error_content .error_list_type dt {float:left; width:60px; _width:70px; padding-left:10px; background:url(https://ssl.pstatic.net/static/common/error/090610/txt_desc5.gif) no-repeat top left; height:18px;}
.error_content .error_list_type dd {float:left; width:336px; _width:340px; padding:0 0 0 4px;}
.error_content .error_list_type dd span {color:#339900; letter-spacing:0;}
.error_content .error_list_type dd a{color:#339900;}
.error_content p.btn{margin:29px 0 100px; text-align:center;}
</style>
</head>
```

```
<!-- ERROR -->
<body>
<div class="error_content">
  <p class="error_desc">
  <p class="error_desc2">방문하시려는 페이지의 주소가 잘못 입력되었거나,
    페이지의 주소가 변경 혹은 삭제되어 요청하신 페이지를 찾을 수 없습니다.<br>
    입력하신 주소가 정확한지 다시 한번 확인해 주시기 바랍니다.
  </p>
  <p class="error_desc3">관련 문의사항은 <a href="https://help.naver.com/" target="_blank">고객센터</a>에 알려주시면 친절히 안내해드리겠습니다.
  <p class="btn">
    <a href="javascript:history.back()">
    <a href="https://finance.naver.com">
  </p>
</div>
</body>
</html>
```

1. 웹 스크래핑 가능 유무 확인하기

■ In [1]: 코드 실행 결과

```
----- ( 중략 ) -----
</head>
<!-- ERROR -->
<body>
<div class="error_content">
  <p class="error_desc"></p>
  <p class="error_desc2">방문하시려는 페이지의 주소가 잘못 입력되었거나,<br>
    페이지의 주소가 변경 혹은 삭제되어 요청하신 페이지를 찾을 수 없습니다.<br>
    입력하신 주소가 정확한지 다시 한번 확인해 주시기 바랍니다.
  </p>
  <p class="error_desc3">관련 문의사항은 <a href="https://help.naver.com/" target="_blank">고객센터</a>에 알려주시면 친절히 안내해드리겠습니다. 감사합니다.</p>
  <p class="btn">
    <a href="javascript:history.back()"></a>
    <a href="https://finance.naver.com"></a>
  </p>
</div>
</body>
</html>
```

- 웹 브라우저 확인 : 아무 문제가 없었던 URL
- 파이썬 소스 코드 : 에러 메시지와 함께 원하는 HTML 페이지가 보이지 않음
 - ✓네이버 주식 웹 사이트에서 웹 브라우저가 아닌 방식으로 HTTP 요청을 보내면 HTTP 응답을 전송하지 못하게 막아 놓았다는 의미임

1. 웹 스크래핑 가능 유무 확인하기

■ 요청 헤더에 값 설정 후 웹 정보 확인하기

- 문제
 - 소스 코드에 웹 브라우저에서 요청할 때 설정한 값을 입력하지 않았기 때문에 발생함
 - '웹 브라우저 에서 보낸 요청이다'라는 의미의 user-agent 값을 요청 헤더에 설정하기
- 소스 코드 실행
- requests.get() 메소드에 headers 인수를 추가→ user-agent를 설정
 - 앞에서 실행한 결과와는 다른 HTML 페이지 내용을 확인할 수 있음

1. 웹 스크래핑 가능 유무 확인하기

▶ # 헤더 정보 설정하기

```
my_headers = {'user-agent': 'Mozilla/5.0'}
```

요청 헤더 설정하여 HTTP 응답 받아오기

```
page = requests.get(url, headers=my_headers)
```

응답 내용 확인하기

```
print(page.text)
```



```
<html lang="ko">
```

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
```

```
<title>네이버페이 증권</title>
```

```
<link rel="stylesheet" type="text/css" href="https://ssl.pstatic.net/imgstock/static.pc/20240411105708/css/news
```

```
<link rel="stylesheet" type="text/css" href="https://ssl.pstatic.net/imgstock/static.pc/20240411105708/css/comm
```

```
<link rel="stylesheet" type="text/css" href="https://ssl.pstatic.net/imgstock/static.pc/20240411105708/css/layo
```

```
<link rel="stylesheet" type="text/css" href="https://ssl.pstatic.net/imgstock/static.pc/20240411105708/css/main
```

```
<link rel="stylesheet" type="text/css" href="https://ssl.pstatic.net/imgstock/static.pc/20240411105708/css/news
```

```
<link rel="stylesheet" type="text/css" href="https://ssl.pstatic.net/imgstock/static.pc/20240411105708/css/news
```

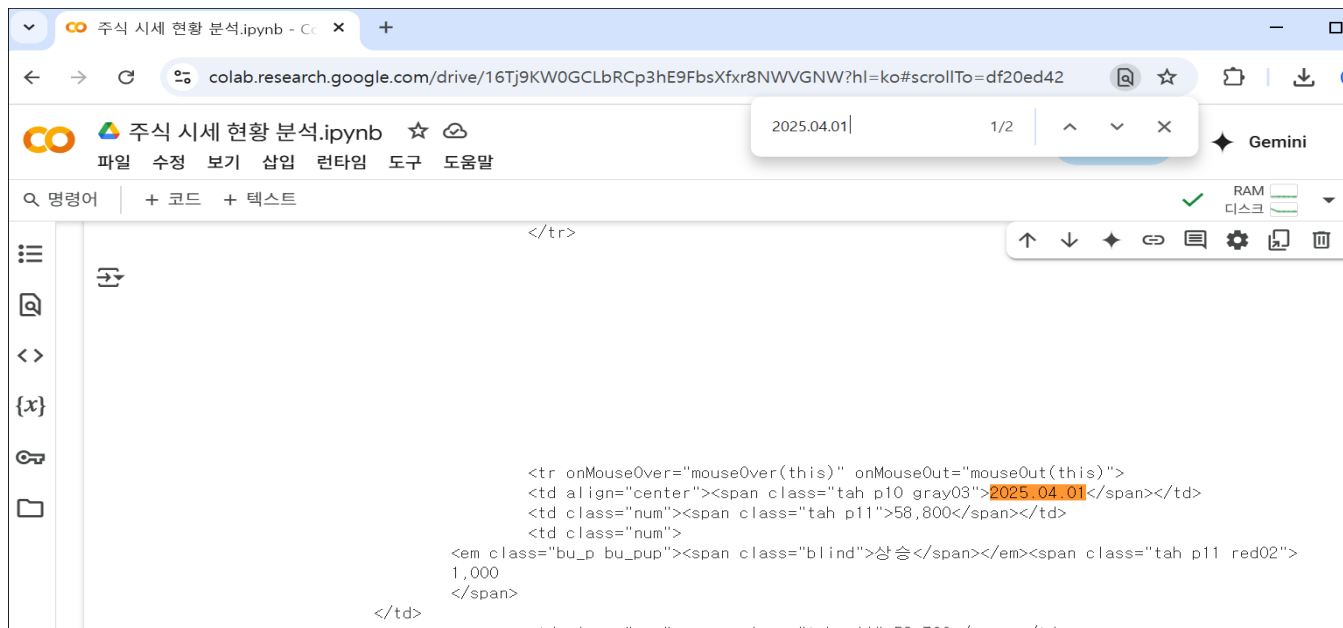
```
<link rel="stylesheet" type="text/css" href="https://ssl.pstatic.net/imgstock/static.pc/20240411105708/css/worl
```

```
</head>
```

1. 웹 스크래핑 가능 유무 확인하기

■ 웹 브라우저에서 확인했던 데이터 검색하기

- <Ctrl> + <F> → 검색 창 → 첫 행의 날짜인 2025.04.01.로 검색



- ✓ 웹 스크래핑 대상을 선정해 파이썬 소스 코드만으로 웹 브라우저 없이 HTTP 요청을 서버에 전송 → HTTP 응답을 받아 HTML 페이지 형식으로 내용을 확인할 수 있게 됨

1. 웹 스크래핑 가능 유무 확인하기

※ CHECK! 웹 스크래핑이 가능한지 반드시 확인하세요

- 웹 페이지에서 데이터가 보이니 소스 코드로 데이터 수집이 가능한지 확인하지 않고 바로 웹 스크래핑을 수행 하는 경우가 많습니다.
 - 웹 페이지에서 보이는 페이지 내용과 소스 코드를 실행해 얻은 페이지 내용이 다른 경우, 작성한 소스 코드가 의도한 대로 동작하지 않아 낭패를 보는 경우가 빈번합니다.
- 원하는 정보를 가져오지 못하는 이유가 소스 코드 실수 때문이라고 오해하고 문제를 찾는 데 많은 시간을 소모할 수도 있습니다.
 - 따라서, 웹 스크래핑을 본격적으로 수행하기 전에 반드시 소스 코드로 데이터를 제대로 가져올 수 있는지 확인하기 바랍니다.

2. 웹 데이터 수집하기

■ HTML 테이블 데이터 가져오기

- 웹 브라우저로 수집 대상 데이터 확인
- 웹 브라우저의 개발자 도구를 열어 <table> 태그를 확인
 - 현재 수집할 값이 있는 표가 보임 → 이러한 표를 그리려면 <table> 태그를 사용함

table.type2 680 x 294.67

날짜	증가	전일비	시가	고가	저가	거래량
2025.04.03	57,600	▼ 1,200	56,900	57,800	56,900	19,508,076
2025.04.02	58,800	— 0	58,800	59,400	58,300	14,402,297
2025.04.01	58,800	▲ 1,000	58,700	59,600	57,900	13,841,706
2025.03.31	57,800	▼ 2,400	59,500	59,700	57,800	17,633,494
2025.03.28	60,200	▼ 1,600	60,700	61,100	60,000	16,282,514
2025.03.27	61,800	▲ 400	60,900	62,000	60,800	20,389,790
2025.03.26	61,400	▲ 1,600	59,800	61,400	59,700	16,431,645
2025.03.25	59,800	▼ 700	60,900	61,100	59,500	17,259,455
2025.03.24	60,500	▼ 1,200	61,200	61,600	60,500	14,088,094
2025.03.21	61,700	▲ 1,500	60,900	61,700	60,400	40,155,612

인기검색종목

코스피

1 삼성전자

2 현대로템

3 한화에어로

4 SK하이닉

5 삼성중공

6 두산에너

7 한화오션

8 대한제당

9 한미반도

10 한화솔류

시세정보 바로

```
</strong>
</h4>
...
<table cellspacing="0" class="type2">...</table> == $0
<!-- 페이지 네비게이션 시작-->
<table summary="페이지 네비게이션 리스트" class="Nnavi" align="center">
<!-- 페이지 네비게이션 끝-->
<script type="text/javascript" src="https://ssl.gstatic.net/imgstock/sta
min.js.1.5.3.euckr.js"></script>
<script type="text/javascript" src="https://ssl.gstatic.net/imgstock/sta
s"></script>
<script type="text/javascript" src="https://ssl.gstatic.net/imgstock/sta
s"></script>
<script type="text/javascript">...</script>
</body>
</html>
</iframe>
<!--// sub start -->
</div>
```

2. 웹 데이터 수집하기

■ HTML 테이블 데이터 가져오기

- 판다스의 `read_html()` 메서드 사용

→ 인수 : HTTP 응답 내용을 확인할 때 사용 했던 HTML 페이지의 텍스트(`page.text`)를 넣으면 됨

- 다음 소스 코드를 실행

→ `read_html()` 메서드가 반환하는 객체(`pages`)의 데이터 타입을 판다스 `type()` 함수로 확인
→ `read_html()` 메서드는 HTML 테이블을 읽어 데이터프레임 객체로 구성된 리스트를 반환함

```
# 판다스 라이브러리 탑재
import pandas as pd

# HTML 페이지에서 테이블 추출
pages = pd.read_html(page.text)

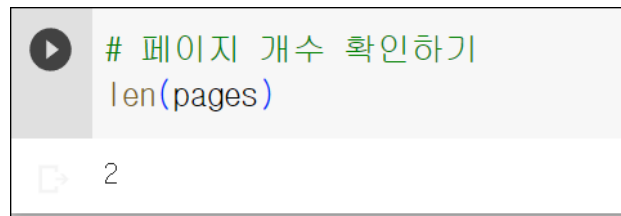
# 추출한 데이터 타입 확인하기
type(pages)
```

list

2. 웹 데이터 수집하기

■ 필요한 테이블 데이터 고르기

- 파이썬 len() 함수 → 리스트의 데이터프레임 객체 개수 확인
- 현재 2개가 있는 것을 알 수 있음



```
# 페이지 개수 확인하기
len(pages)
```

2

- 웹 페이지의 데이터 테이블 개수 : 1개
- 함수로 확인한 데이터 테이블 개수 : 2개
- ✓ 개수가 차이나는 이유? 항목 2개의 값을 차례로 살펴보기

2. 웹 데이터 수집하기

■ 두 개의 테이블 데이터 확인하기

# 첫번째 테이블 내용 확인하기 pages[0]							
	날짜	종가	전일비	시가	고가	저가	거래량
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	2025.04.03	57600.0	하락 1,200	56900.0	57800.0	56900.0	19508076.0
2	2025.04.02	58800.0	보합0	58800.0	59400.0	58300.0	14402297.0
3	2025.04.01	58800.0	상승 1,000	58700.0	59600.0	57900.0	13841706.0
4	2025.03.31	57800.0	하락 2,400	59500.0	59700.0	57800.0	17633494.0
5	2025.03.28	60200.0	하락 1,600	60700.0	61100.0	60000.0	16282514.0
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	2025.03.27	61800.0	상승 400	60900.0	62000.0	60800.0	20389790.0
10	2025.03.26	61400.0	상승 1,600	59800.0	61400.0	59700.0	16431645.0
11	2025.03.25	59800.0	하락 700	60900.0	61100.0	59500.0	17259455.0
12	2025.03.24	60500.0	하락 1,200	61200.0	61600.0	60500.0	14088094.0
13	2025.03.21	61700.0	상승 1,500	60900.0	61700.0	60400.0	40155612.0
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN

2. 웹 데이터 수집하기

■ 두 개의 테이블 데이터 확인하기

두번째 테이블 내용 확인하기

pages[1]

0

맨앞

1

2

3

4

5

6

7

8

9

10

다음

맨뒤

- 첫 번째 항목을 출력한 결과 : 우리가 찾던 데이터
- 두 번째 항목을 출력한 결과 : 테이블 밑에 있는 페이지 바
 - 눈에 보이는 테이블 이외에도 HTML에 `<table>` 태그가 쓰인 곳이 여러 곳일 수 있음
 - 어떤 항목의 데이터프레임이 원하는 데이터인지 반드시 확인해야 함

3. 웹 페이지 URL 분석하기

■ 페이징(Paging) 처리

- 데이터를 여러 페이지로 나누어 출력하도록 설정하는 것을 의미
- 너무 많은 데이터를 한 번에 보여줄 때 발생하는 일
 - 페이지 로딩 시간이 길어질 수 있으며, 자원이 부족해 오류가 발생할 수도 있음
 - 주식 일별 시세와 같이 오랫동안 축적한 데이터는 한 페이지에 제한된 개수의 데이터만 보여 주는 것이 일반적임

3. 웹 페이지 URL 분석하기

■ URL 주소 분석하기

- 여러 페이지에 담긴 데이터를 가져오기 위한 방법 : URL 확인하기

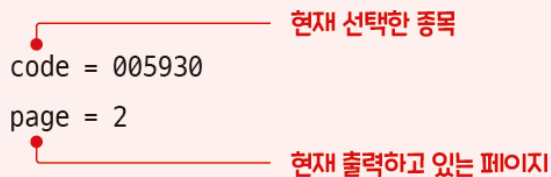
`https://finance.naver.com/item/sise_day.naver?code=005930&page=2`

- URL에는 정적인 주소 외에 동적으로 변경할 수 있는 정보를 메서드 인수와 같이 추가하는 방법이 담겨 있음
- 이 방법을 **쿼리 문자열**(Query String) 혹은 **URL 매개변수**(파라미터Parameter)라고 부름
 - 쿼리 문자열 : URL 끝에서 물음표 기호(?)로 시작
 - 매개변수 이름과 값을 **이름=값** 형식으로 표기함
 - 매개변수가 여러 개인 경우, 앰퍼샌드 기호(&)를 사용해 구분

3. 웹 페이지 URL 분석하기

■ URL 주소 분석하기

- 2개의 매개변수를 보내고 있는 URL : `code=005930&page=2`



code = 005930
page = 2

현재 선택한 종목
현재 출력하고 있는 페이지

- 코드(code) : 현재 선택한 종목을 나타내는 정보
 - 페이지(page) : 현재 출력하고 있는 페이지를 뜻함
- 여러 페이지 정보를 가져오려면 매개변수 페이지 숫자를 증가하면서 데이터를 축적하면 됨

4. 페이징 처리가 되어 있는 데이터 수집하기

■ 페이지 번호 증가용 반복문 작성

- 페이지 번호를 제외한 나머지 부분을 저장한 새로운 변수 : new_url 생성
- 반복문을 순회하면서 추출한 테이블 정보를 축적할 데이터 프레임 : all_tables 생성



```
# 페이지 번호를 제외한 주소 저장
```

```
new_url = 'https://finance.naver.com/item/sise_day.naver?code=005930&page='
```

```
# 데이터를 축적할 데이터프레임 생성
```

```
all_tables = pd.DataFrame()
```

4. 페이징 처리가 되어 있는 데이터 수집하기

■ 페이지 번호 증가용 for문 작성

```
# 페이지 번호 증가용 반복문
for page_number in range(1, 2):

    # 페이지 번호 추가한 주소 완성
    full_url = new_url + str(page_number)

    # 주소 확인하기
    print(f'{page_number} 번째 페이지 읽어오기({full_url})')

    # HTTP 요청 전송 후 응답 받아오기
    page = requests.get(full_url, headers=my_headers)

    # 테이블 추출
    table = pd.read_html(page.text)[0]

    # 수행할 내용 확인
    print(f'전체 {len(all_tables.index)} 줄 에 {len(table.index)} 줄 추가')

    # 데이터 축적용 데이터프레임에 테이블 추가
    all_tables = pd.concat([all_tables, table])
```

➡ 1 번째 페이지 읽어오기(https://finance.naver.com/item/sise_day.naver?code=005930&page=1)
전체 0 줄 에 15 줄 추가

4. 페이징 처리가 되어 있는 데이터 수집하기

■ 결손치 제거

- for 문을 1회 수행한 결과 : 다음 소스 코드를 실행
- NaN : HTML <table> 태그 안에 존재하는 행이지만 데이터는 포함하고 있지 않음

▶ # 전체 테이블 결과 확인
all_tables

	날짜	종가	전일비	시가	고가	저가	거래량
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	2025.04.17	55100.0	상승 400	54700.0	55300.0	54500.0	10862251.0
2	2025.04.16	54700.0	하락 1,900	56000.0	56200.0	54500.0	14437025.0
3	2025.04.15	56600.0	상승 400	56300.0	57100.0	56200.0	8998640.0
4	2025.04.14	56200.0	상승 1,000	56300.0	56700.0	55800.0	12852613.0
5	2025.04.11	55200.0	하락 1,200	55600.0	55700.0	54800.0	13930480.0
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	2025.04.10	56400.0	상승 3,400	56600.0	56700.0	55000.0	22948172.0
10	2025.04.09	53000.0	하락 500	53300.0	54300.0	52900.0	20576960.0
11	2025.04.08	53500.0	상승 300	55000.0	55300.0	53300.0	25532845.0
12	2025.04.07	53200.0	하락 2,900	53300.0	54100.0	53100.0	31998883.0
13	2025.04.04	56100.0	하락 1,500	56200.0	58200.0	55700.0	23527139.0
14	NaN	NaN	NaN	NaN	NaN	NaN	NaN

4. 페이징 처리가 되어 있는 데이터 수집하기

■ 결손치 제거

- 다음 소스 코드 실행
→ 결손치가 제거 된 값 확인



결손치 제거

```
all_tables.dropna(inplace=True)
```

데이터 확인하기

```
all_tables
```



	날짜	종가	전일비	시가	고가	저가	거래량
1	2025.04.17	55100.0	상승 400	54700.0	55300.0	54500.0	10862251.0
2	2025.04.16	54700.0	하락 1,900	56000.0	56200.0	54500.0	14437025.0
3	2025.04.15	56600.0	상승 400	56300.0	57100.0	56200.0	8998640.0
4	2025.04.14	56200.0	상승 1,000	56300.0	56700.0	55800.0	12852613.0
5	2025.04.11	55200.0	하락 1,200	55600.0	55700.0	54800.0	13930480.0
9	2025.04.10	56400.0	상승 3,400	56600.0	56700.0	55000.0	22948172.0
10	2025.04.09	53000.0	하락 500	53300.0	54300.0	52900.0	20576960.0
11	2025.04.08	53500.0	상승 300	55000.0	55300.0	53300.0	25532845.0
12	2025.04.07	53200.0	하락 2,900	53300.0	54100.0	53100.0	31998883.0
13	2025.04.04	56100.0	하락 1,500	56200.0	58200.0	55700.0	23527139.0

4. 페이징 처리가 되어 있는 데이터 수집하기

■ 10페이지 데이터 수집하기

- 10페이지 데이터 : 100일 데이터 수집
- for 문의 range() 인수가 1부터 10까지 순회하도록 다음과 같이 수정하고 실행함

```
▶ # 데이터를 축적할 데이터프레임 생성
all_tables = pd.DataFrame()

# 페이지 번호 증가용 반복문
for page_number in range(1, 11):

    # 페이지 번호 추가한 주소 완성
    full_url = new_url + str(page_number)

    # 주소 확인하기
    print(f'{page_number} 번째 페이지 읽어오기({full_url})')

    # HTTP 요청 전송 후 응답 받아오기
    page = requests.get(full_url, headers=my_headers)

    # 테이블 추출
    table = pd.read_html(page.text)[0]

    # 수행할 내용 확인
    print(f'전체 {len(all_tables.index)} 줄에 {len(table.index)} 줄 추가')

    # 데이터 축적용 데이터프레임에 테이블 추가
    all_tables = pd.concat([all_tables, table])
```


4. 페이징 처리가 되어 있는 데이터 수집하기

■ 10페이지 데이터 수집하기

☞ 1 번째 페이지 읽어오기(https://finance.naver.com/item/sise_day.naver?code=005930&page=1)
전체 0 줄에 15 줄 추가
2 번째 페이지 읽어오기(https://finance.naver.com/item/sise_day.naver?code=005930&page=2)
전체 15 줄에 15 줄 추가
3 번째 페이지 읽어오기(https://finance.naver.com/item/sise_day.naver?code=005930&page=3)
전체 30 줄에 15 줄 추가
4 번째 페이지 읽어오기(https://finance.naver.com/item/sise_day.naver?code=005930&page=4)
전체 45 줄에 15 줄 추가
5 번째 페이지 읽어오기(https://finance.naver.com/item/sise_day.naver?code=005930&page=5)
전체 60 줄에 15 줄 추가
6 번째 페이지 읽어오기(https://finance.naver.com/item/sise_day.naver?code=005930&page=6)
전체 75 줄에 15 줄 추가
7 번째 페이지 읽어오기(https://finance.naver.com/item/sise_day.naver?code=005930&page=7)
전체 90 줄에 15 줄 추가
8 번째 페이지 읽어오기(https://finance.naver.com/item/sise_day.naver?code=005930&page=8)
전체 105 줄에 15 줄 추가
9 번째 페이지 읽어오기(https://finance.naver.com/item/sise_day.naver?code=005930&page=9)
전체 120 줄에 15 줄 추가
10 번째 페이지 읽어오기(https://finance.naver.com/item/sise_day.naver?code=005930&page=10)
전체 135 줄에 15 줄 추가

4. 페이징 처리가 되어 있는 데이터 수집하기

■ 결손치를 제거한 전체 데이터 확인

- 7개의 정보가 있는 100일치 데이터가 수집됨

```
# 결손치 제거
all_tables.dropna(inplace=True)

# 데이터 확인하기
all_tables
```

	날짜	종가	전일비	시가	고가	저가	거래량
1	2025.04.17	55100.0	상승 400	54700.0	55300.0	54500.0	10862598.0
2	2025.04.16	54700.0	하락 1,900	56000.0	56200.0	54500.0	14437025.0
3	2025.04.15	56600.0	상승 400	56300.0	57100.0	56200.0	8998640.0
4	2025.04.14	56200.0	상승 1,000	56300.0	56700.0	55800.0	12852613.0
5	2025.04.11	55200.0	하락 1,200	55600.0	55700.0	54800.0	13930480.0
...
9	2024.11.25	57900.0	상승 1,900	57400.0	57900.0	56700.0	36237325.0
10	2024.11.22	56000.0	하락 400	56000.0	56700.0	55900.0	15281543.0
11	2024.11.21	56400.0	상승 1,100	54900.0	56900.0	54700.0	19096850.0
12	2024.11.20	55300.0	하락 1,000	56100.0	56500.0	54800.0	20864667.0
13	2024.11.19	56300.0	하락 400	56500.0	57500.0	55900.0	31539632.0

100 rows × 7 columns