- HTML(HyperText Markup Language)
  - 확장자는 html, htm이며 인터넷 익스플로러와 같은 웹 브라우저에서 읽을 수 있는 웹 문서를 만들기 위한 언어이다.
  - HTML의 특성
    - 하이퍼텍스트(Hypertext) : 참조(하이퍼링크)를 통해 독자가 한 문서에서 다른 문서로 즉시 접근할 수 있는 텍스트.
    - 마크업 언어(Markup Language) : 문서의 내용 뿐만 아니라, 태그 등을 이용하여 글자 크기, 모양 등 문서의 출력 형태까지 명기하는 언어의 한 가지이다.
    - HTML에서는 태그(tag)라고 하는 심볼을 사용하여 문서의 구조, 출력 형태 등을 조정한다.
    - 태그는 < >를 사용하여 표시하며, <태그명>으로 시작하고 </태그명>으로 끝맺는데(*), 이 사이에 내용을 넣는다.
    - 태그에는 태그의 성격을 구체화 하는 속성(attribute)를 부여할 수 있다.

(*) 종료태그라고 하며, 일부 태그는 종료태그가 없을 수도 있다.

✦HTML 문서의 예

```
<!doctype html>
<head><title>An Example</title></head>
<body>
  <h3>The second heading(small font).</h3>
  <p>A paragraph(number list):
    <b style="color:red;">123</b>, <b style="color:red;">356</b>,
    <b>641</b>, <b>387</b> </p>
  <p>Hyperlink: <a href="http://www.google.com">google</a></p>
  <p>Link with italic: <a href="http://www.naver.com"><i>naver</i></a></p>
</body>
```

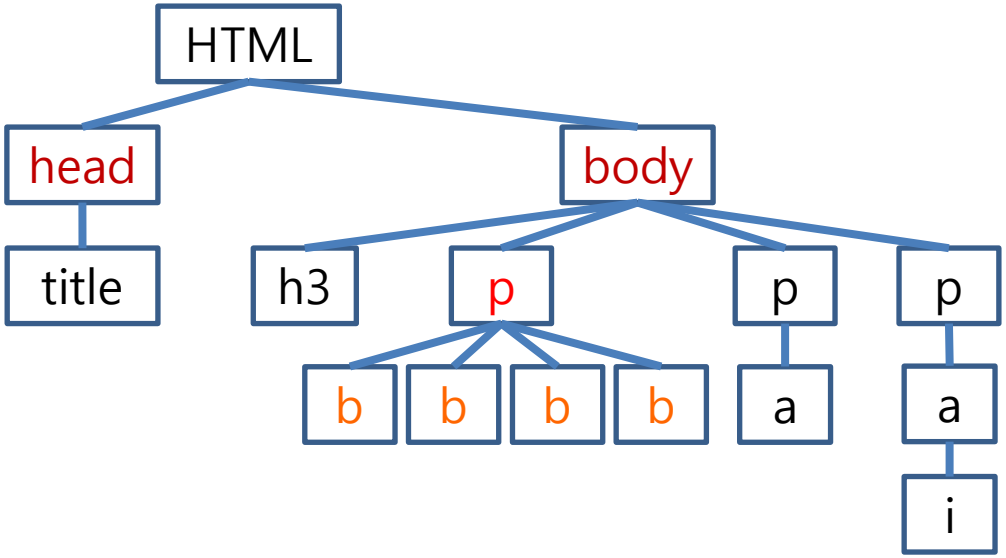
head : 브라우저 정보
title : 툴바의 제목
a (anchor) : 웹문서(or 문서내 다른 곳)를 연결
style="color:red;" :
속성 속성값 (*)

위 문서가 저장된 파일을 웹브라우저에서 읽으면 다음과 같이 보인다.

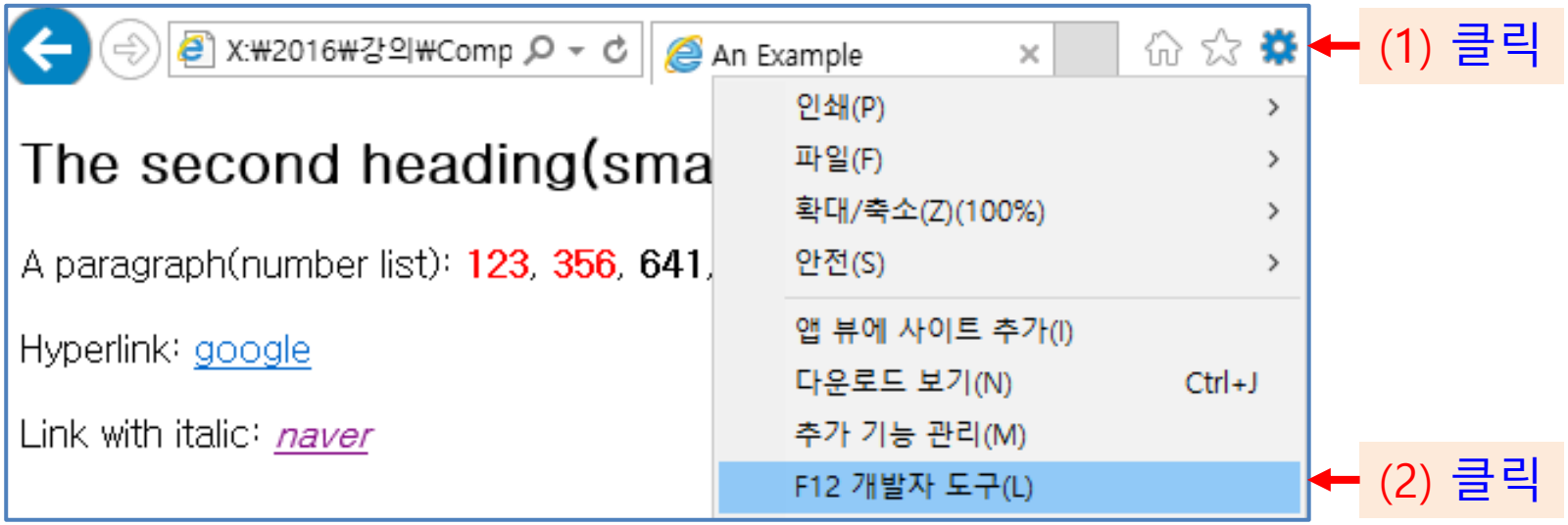
(*) 본 강의에서 태그와 속성의 의미를 알 필요는 없다

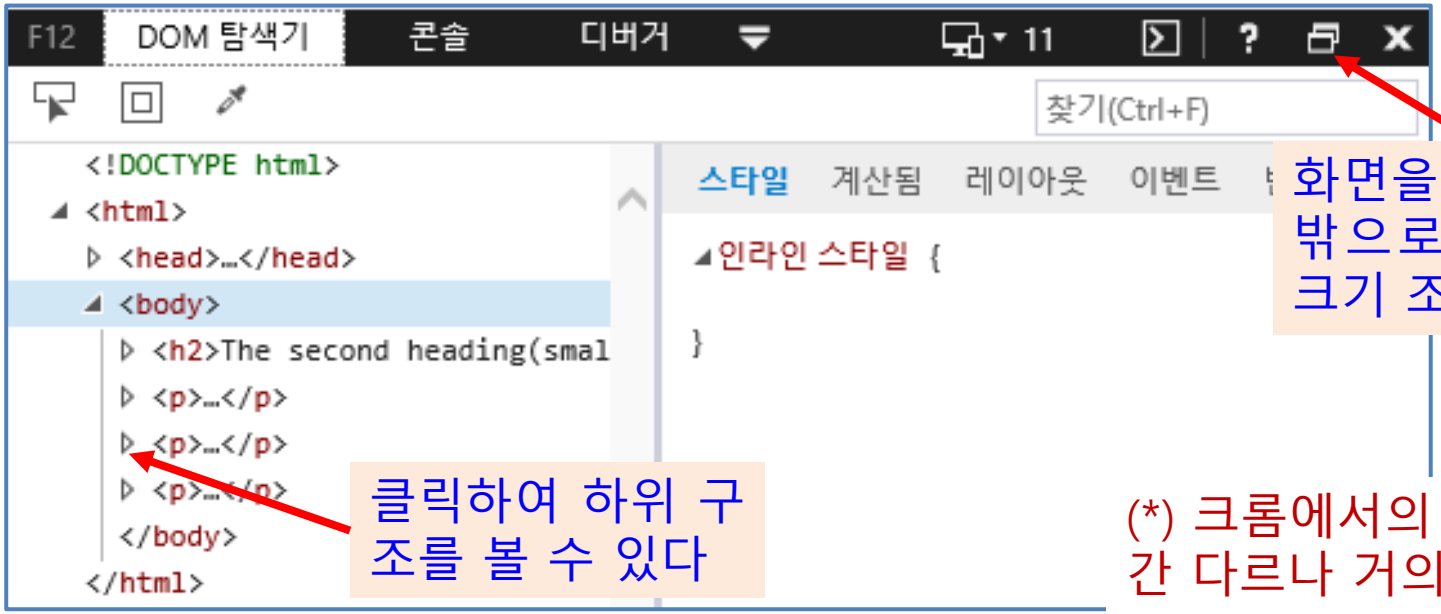
✦HTML 문서는 태그를 노드로 하는 트리 구조를 갖는다.

```
<!doctype html>
<head><title>An Example</title></head>
<body>
  <h3>The second heading(small font).</h3>
  <p>A paragraph(number list):
    <b style="color:red;">123</b>, <b style="color:red;">356</b>,
    <b>641</b>, <b>387</b> </p>
  <p>Hyperlink: <a href="http://www.google.com">google</a></p>
  <p>Link with italic: <a href="http://www.naver.com"><i>naver</i></a></p>
</body>
```



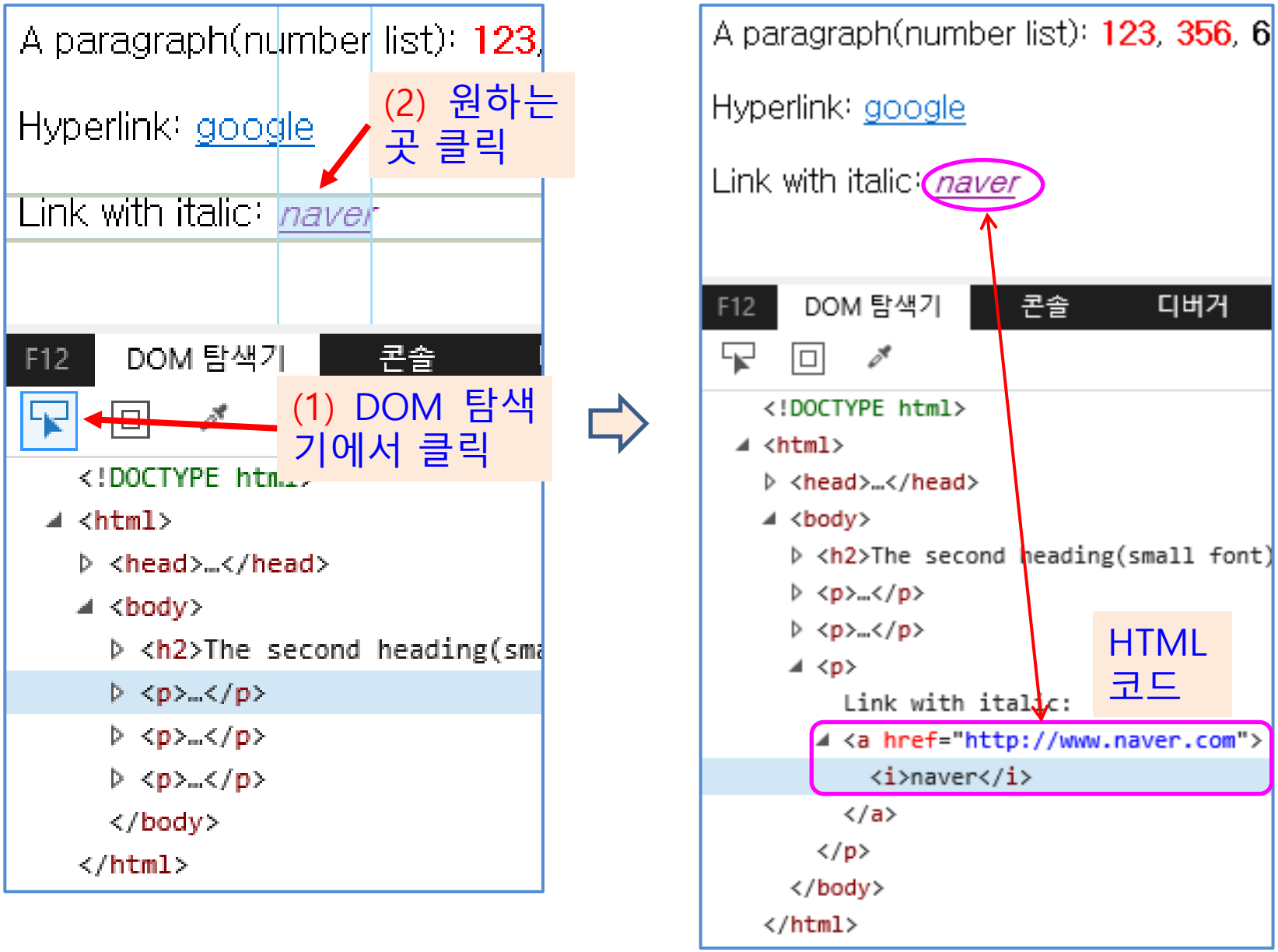
✦웹브라우저에서 HTML 문서 보기(*)





(*) 크롬에서의 방법은 약간 다르나 거의 유사하다

✦웹브라우저에서 HTML 문서내 원하는 부분의 코드 보기



웹 데이터 추출 (Web Scraping)

- 웹 문서에서 원하는 데이터를 정제해서 추출할 수 있다.
- 이를 위해서는 인터넷에서 문서를 읽어오고 이로부터 필요한 데이터를 추출하는 방법이 필요하다.
- 본 강의에서는 이를 위한 가장 기본적인 방법을 소개한다.

◆ 외부 라이브러리 (패키지) 설치

✦PIP를 통한 패키지 beautifulsoup4와 matplotlib 설치

◆PIP(Pip Installs Packages): 파이썬 패키지 관리 프로그램으로 명령프롬프트에서 다음과 같이 사용할 수 있다(*).

```
C:\>pip list                    # 현재 파이썬에 설치된 패키지 확인
C:\>pip install package_name
                  # 패키지 설치 또는 삭제(install 대신 uninstall)
C:\>pip show package_name   # 설치된 패키지 세부내용 확인
C:\>pip search package_name   # 인터넷에서 패키지 검색
C:\>python -m pip install --upgrade pip
                  # pip 프로그램이 구형일 경우 신형으로 업그레이드
```

◆pip list를 통하여 beautifulsoup4와 matplotlib가 설치되어 있는지 확인한다(미 설치이면 다음 명령을 실행한다).

```
C:\>pip install beautifulsoup4
C:\>pip install matplotlib
```

(*) 갈색 글은 입력하지 않는다.

◆ 인터넷에서 웹 문서를 읽는 방법

✦ 모듈 urllib.request의 함수 urlopen( )을 사용한다(1).

- ◆ 인수는 url(2) 즉, 인터넷 주소이다(문자열).
- ◆ HTTPResponse라는 object를 반환한다.
- ◆ 이 object는 모듈 BeautifulSoup의 메소드 prettify( )를 통하여 HTML 문서로 출력할 수 있다(다음 쪽에서 설명).

✦ Example

```
from urllib.request import *
wp = urlopen('http://mail.sogang.ac.kr')
print(type(wp)) #<class 'http.client.HTTPResponse'>
```

(1) 서로 연관된 모듈을 모아둔 것을 패키지(package)라고 한다. urllib는 패키지이고 request는 urllib에 포함된 모듈이다.

(2) url(uniform resource locator) : 네트워크 상에서 자원이 어디 있는지 알려주는 규약. 웹 사이트 주소가 이에 속한다.

◆ 데이터 추출

- ✦모듈 Beautiful Soup를 사용한다(외부 모듈로 설치 필요).
- ✦Beautiful Soup는 HTML 코드를 입력 받아, 구문 분석(*)을 통하여 데이터 추출에 용이한 구조로 변환한다.
- ✦Example

```
from urllib.request import *
from bs4 import *  # import Beautiful Soup

wp = urlopen('http://mail.sogang.ac.kr')

soup = BeautifulSoup(wp, 'html.parser') # parsing
print(type(soup))  # <class 'bs4.BeautifulSoup'>
print(soup.prettify()) # HTML 코드 출력
```

이런 형태로 출력된다

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "h
<html>
 <head>
  <title>
  </title>
  <meta content="IE=edge" http-equiv="X-UA-Compatible"> ...
```

(*) 파싱(parsing)이라고 하며, 일반적으로 트리 구조(parse tree)를 만든다.

◆ Method .find_all( )을 통한 자료 추출

✦ 다음과 같은 HTML 문서를 읽었다고 하자

```
<p>A paragraph(number list):
 <b style="color:red;">123</b>, <b style="color:red;">356</b>,
 <b>641</b>, <b>387</b> </p>
<p>Hyperlink: <a href="http://www.google.com">google</a></p>
```

```
# soup : BeautifulSoup()의 반환 값이라고 가정
bList = soup.find_all('b') # 태그가 b인 원소들의 리스트
for b in bList :
  print(b.get_text()) # 123 356 641 387 (문자열 출력)
        # Method .get_text()는 bList의 원소에서 화면에 보이는
        # 것만을 추출하여 문자열로 반환.
```

✦ bList의 실제 내용

```
bList = [<b style="color:red;">123</b>,
 <b style="color:red;">356</b>, <b>641</b>, <b>387</b>]
```

```
<p>A paragraph(number list):
  <b style="color:red;">123</b>, <b style="color:red;">356</b>,
  <b>641</b>, <b>387</b> </p>
<p>Hyperlink: <a href="http://www.google.com">google</a></p>
```

✦ 태그가 b이고 속성이 style="color:red;"인 정수 356은 다음과 같이 태그와 속성 및 속성 값을 포함시켜 얻을 수 있다.

```
bList = soup.find_all('b', {'style':'color:red;'} )

print(bList[1].get_text()) # 356 (bList의 두번째)

  # bList = [<b style="color:red;">123</b>,
             <b style="color:red;">356</b>]
```

✦ Methods .find_all( )과 .get_text( )는 아래와 같은 유형의 Beautiful Soup object들에 적용할 수 있다.

```
BeautifulSoup()의 반환 유형: bs4.BeautifulSoup
find_all()의 반환 유형: bs4.element.ResultSet
```

```
<p>A paragraph(number list):
  <b style="color:red;">123</b>, <b style="color:red;">356</b>,
  <b>641</b>, <b>387</b> </p>
<p>Hyperlink: <a href="http://www.google.com">google</a></p>
```

✦ 두 단계 이상을 거쳐 태그가 b인 정수를 추출할 수도 있다(1).

```
pList = soup.find_all('p') # 태그가 p인 요소들의 리스트

# pList[0]에 태그가 b인 요소들이 있다.
bList = pList[0].find_all('b') # 태그가 b인 요소 리스트
for b in bList :
  print(b.get_text()) # 123 356 641 387 차례로 출력
```

✦ 태그, 속성 (및 속성값) 등을 조합하여 원하는 자료를 다양하게 추출할 수 있다(2).

(1) 사실 이 예에서는 불필요하다.
(2) 여기서는 종가 추출에 꼭 필요한 것만 보였다.

```
<p>A paragraph(number list):
  <b style="color:red;">123</b>, <b style="color:red;">356</b>,
  <b>641</b>, <b>387</b> </p>
<p>Hyperlink: <a href="http://www.google.com">google</a></p>
```

Method find( )

- find_all( )은 인수가 일치하는 모든 항목을 찾으나, find( )는 인수가 일치하는 첫 번째 항목만을 찾아 반환한다.
- find( ) 를 사용하면 속성 값도 얻을 수 있다.

```
# 태그가 b이고 style 속성이 없는 첫 번째 정수
b = soup.find('b', {'style':''})
print(b.get_text())  # 641

# 태그가 a인 첫 번째 요소의 속성 href의 값
url = soup.find('a')['href']
print(url) # http://www.google.com
```

일별 주식 종가 추출

◆ 일별 주식 종가

✦ 네이버, 삼성전자 주식 페이지에서 투자자별 매매동향을 선택하면(*), 아래 보인 것과 같이 일별 주식 종가가 보인다.

외국인 · 기관 순매매 거래량

날짜	종가	전일비	등락률	거래량	기관	외국인		
					순매매량	순매매량	보유주수	보유율
2016.11.18	1,586,000	▲ 18,000	+1.15%	186,856	-23,059	+34,910	71,245,289	50.64%
2016.11.17	1,568,000	▲ 10,000	+0.64%	157,411	-23,216	+23,213	71,210,379	50.62%
2016.11.16	1,558,000	▲ 19,000	+1.23%	266,164	-18,200	-1,172	71,177,470	50.60%
2016.11.15	1,539,000	▼ 14,000	-0.90%	273,314	+17,247	-73,398	71,177,054	50.60%

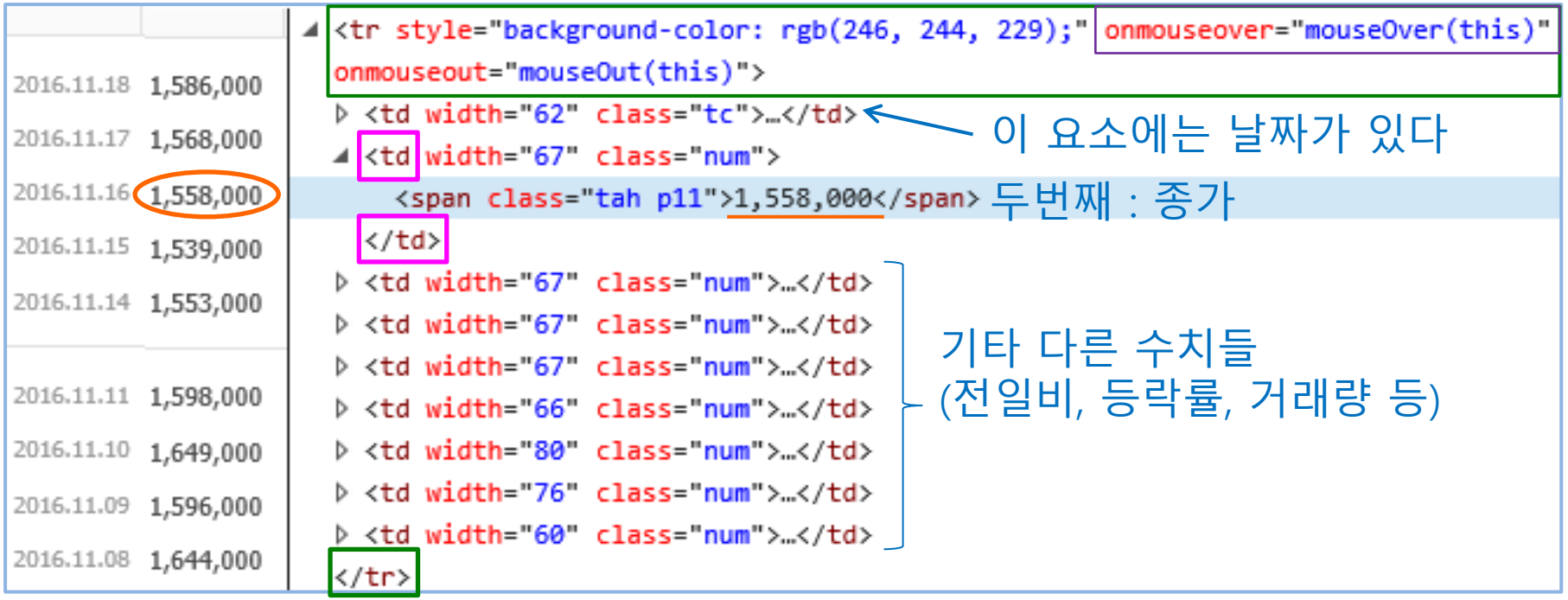
✦ 이로부터 원하는 날짜 만큼 과거 일별 주식 종가를 추출해 보자.

(*) http://finance.naver.com/item/frgn.nhn?code=005930&page=1

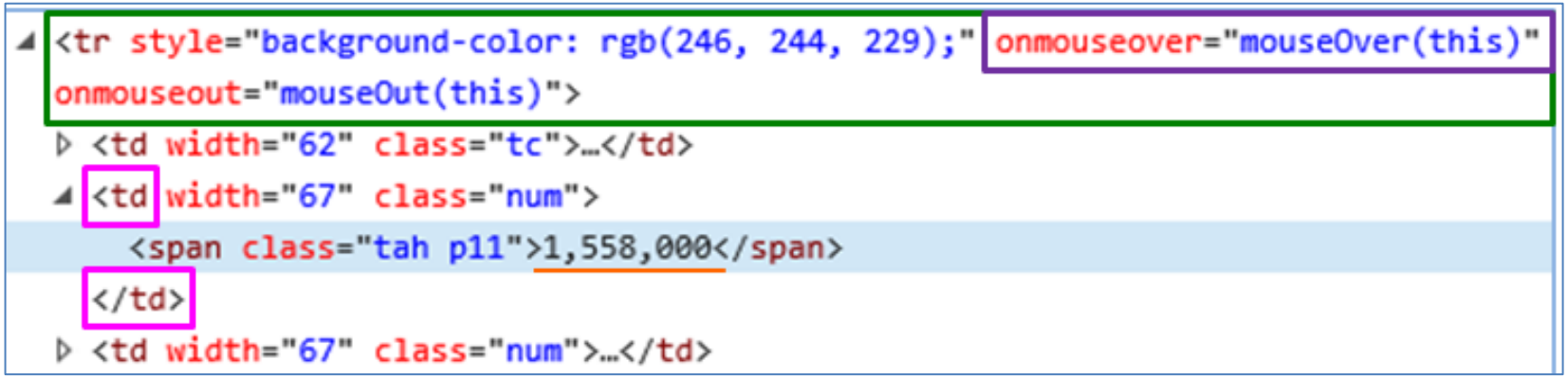
◆ 패턴 인식

✦먼저 추출한 일별 종가 주변의 HTML 코드를 살펴보자.

✦DOM 탐색기로 코드를 보면 아래와 같은 패턴을 볼 수 있다.



1. 주식 종가는 태그가 **tr**인 요소에 속해있다.
2. 태그 **tr**과 속성 **onmouseover**를 사용하여 요소들을 추출하자.
3. 추출한 리스트의 원소들은 태그가 **td**인 요소들이며 종가는 리스트의 두 번째 원소에 있다.



✦웹 문서 페이지 하나에 TR object가 일별로 20 개 존재한다.

✦따라서, 웹 문서에서 일별 종가 추출은 다음 과정에 의한다.

1. 태그가 **tr**, 속성이 **onmouseover="mouseOver(this)"**인 요소들을 모두 추출한 리스트(이를 **trList**라고 하자)
2. **trList**의 각 원소에서 태그가 **td**인 요소들을 추출한 리스트(**tdList**라고 하자)
3. **tdList**의 두 번째 원소에서 종가 추출(즉, **tdList[1]**)

◆ 알고리즘 설계

✦ 웹 문서 읽기

- ◆ DN을 주어진 추출하고자 하는 일별 종가의 수라고 하자.
- ◆ 네이버 해당 웹 문서 한 쪽에는 20개의 일별 종가가 있다.
- ◆ 따라서, 읽어야 할 웹 문서의 쪽 수 PN을 계산하여야 한다.
- ◆ PN 값은 부등식 $DN \le PN \times 20 < DN + 20$이 성립하도록 결정하고, 편의상 PN × 20개를 추출하는 것으로 한다(*).
- ◆ 읽어야 할 웹 문서 쪽수가 예를 들어 3이라면, 다음과 같은 세 주소에서 각 쪽을 읽어 추출하여야 한다(삼성의 경우).

```
http://finance.naver.com/item/frgn.nhn?code=005930&page=1
http://finance.naver.com/item/frgn.nhn?code=005930&page=2
http://finance.naver.com/item/frgn.nhn?code=005930&page=3
```

쪽 수를 1부터 차례로 바꿔야 한다

(*) 편의상, 추출할 종가를 PN 개의 페이지에서 PN × 20 개를 추출한다.

✦ 알고리즘 전체 개요

- ◆ 웹 문서를 PN 쪽 읽어야 하므로, 반복문을 사용한다.
- ◆ webUrl을 쪽 번호를 제외한 웹 주소라고 하자. 예를들어

```
webUrl="http://finance.naver.com/item/frgn.nhn?code=005930&page="
```

- ◆ 알고리즘 개요 (입력: webUrl, DN)

```
1. pList = list()                    # pList = 종가를 저장하는 리스트
2. DN을 입력 받아 PN을 구한다        # DN ≤ PN × 20 < DN + 20이도록
3. for page = 1, 2, ..., PN에 대해서
4.    wPage = urlopen(웹 주소) # webUrl에 쪽수를 덧붙여서
5.    soup = BeautifulSoup(wPage, 'html.parser') # 문서 파싱
6.    soup에서 trList를 구한다  # 태그가 tr인 요소 리스트(*)
7.    for trList의 원소 tr에 대해서
8.       tr에서 tdList를 구한다  # 태그가 td인 요소 리스트
9.       tdList[1]에서 종가를 추출하여 pList에 추가한다.
10.pList의 값들을 과거 종가가 먼저 오도록 순서를 바꾼다 # 그래픽용
11.return pList
```

(*) 태그가 tr이고 속성이 onmouseover= "mouseOver(this)" 인 요소

```
2. DN을 입력 받아 PN을 구한다    # DN ≤ PN × 20 < DN + 20이도록
3. for page = 1, 2, ..., PN에 대해서
4.    wPage = urlopen(웹 주소) # webUrl에 쪽수를 덧붙여서
5.    soup = BeautifulSoup(wPage, 'html.parser') # 문서 파싱
6.    soup에서 trList를 구한다  # 태그가 tr인 요소 리스트
7.    for trList의 원소 tr에 대해서
8.       tr에서 tdList를 구한다  # 태그가 td인 요소 리스트
9.       tdList[1]에서 종가를 추출하여 pList에 추가한다.
```

◆Line 2 :

```
PN = DN // 20       # 페이지당 20일 분의 종가
if DN > PN * 20 :   # 추출 횟수를 20의 배수로
   PN += 1          # 조정
```

◆Line 4 : webUrl 문자열 뒤에 page를 문자열로 바꾼 것을 덧붙인다.

◆Line 6 : find_all()을 사용하여 soup에서 태그가 tr이고 속성이 onmouseover="mouseOver(this)"인 요소들의 리스트 추출.

◆Line 8 : 이 역시 find_all()을 사용하여 tr에서 태그가 td인 요소들의 리스트 추출.

```
7.    for trList의 원소 tr에 대해서
8.       tr에서 tdList를 구한다   # 태그가 td인 요소 리스트
9.       tdList[1]에서 종가를 추출하여 pList에 추가한다.
10.pList의 값들을 과거 종가가 먼저 오도록 순서를 바꾼다 # 그래픽용
11.return pList
```

◆Line 9 : 이는 다음과 같은 과정이 필요하다.

(a) .get_text()을 사용하여 종가 문자열을 얻는데, 이 문자열을 price라고 하자.

(b) price에는 콤마가 포함되어 있으므로, .replace()로 콤마를 제거한 후 다시 price에 저장한다.

(c) price를 int()를 사용하여 정수로 바꾼다.

(d) .append()를 사용하여 price를 pList에 추가한다.

◆Line 10

리스트 method .reverse()를 사용하여 리스트 원소를 거꾸로 배치한다.

큐(Queue) ADT

(*) https://habibadanvalbarry.wordpress.com/tag/envy/

- 매표소, 식당 등 많은 장소에서는 먼저 온 사람이 먼저 서비스를 받는데, 이를 추상화한 자료구조를 큐라고 한다.
- 큐에서는 가장 먼저 추가된 원소가 먼저 삭제된다 (FIFO (First In First Out)라고도 한다).
- 큐 자료구조에 필요한 주된 연산은 다음과 같다.
  - enqueue(a) : 큐에 원소 a를 추가.
  - dequeue( ) : 가장 오래 전에 추가한 원소를 삭제하여 반환.
  - front( ) : 가장 오래 전에 추가한 원소를 삭제 않고, 반환.
  - isEmpty( ) : 빈 큐인지 조사.

✦ 큐 ADT의 구현

- ◆ 리스트를 사용하여 구현 가능하지만 비효율적이다(*).
- ◆ enqueue(a) : 리스트 메소드 append(a)를 사용한다.
- ◆ dequeue( ) : 리스트 메소드 pop(0)를 사용한다.
- ◆ front( ) : 리스트의 첫 번째 원소이다.
- ◆ isEmpty( ) : len( ) 함수 사용(저장된 원소의 개수도).
- ◆ Example

```
Q = [1,2,3]
Q.append(4)        # enqueue(4)와 동일
print(Q)           # [1, 2, 3, 4]
a = Q.pop(0)       # dequeue()와 동일
print(a, Q)        # 1 [2, 3, 4]
b = Q[0]           # front()와 동일
print(b, Q)        # 2 [2, 3, 4]
```

(*) **deque**를 사용하는 것이 효율적이니 차후 필요시 deque를 사용하자.
https://docs.python.org/3/library/collections.html#collections.deque

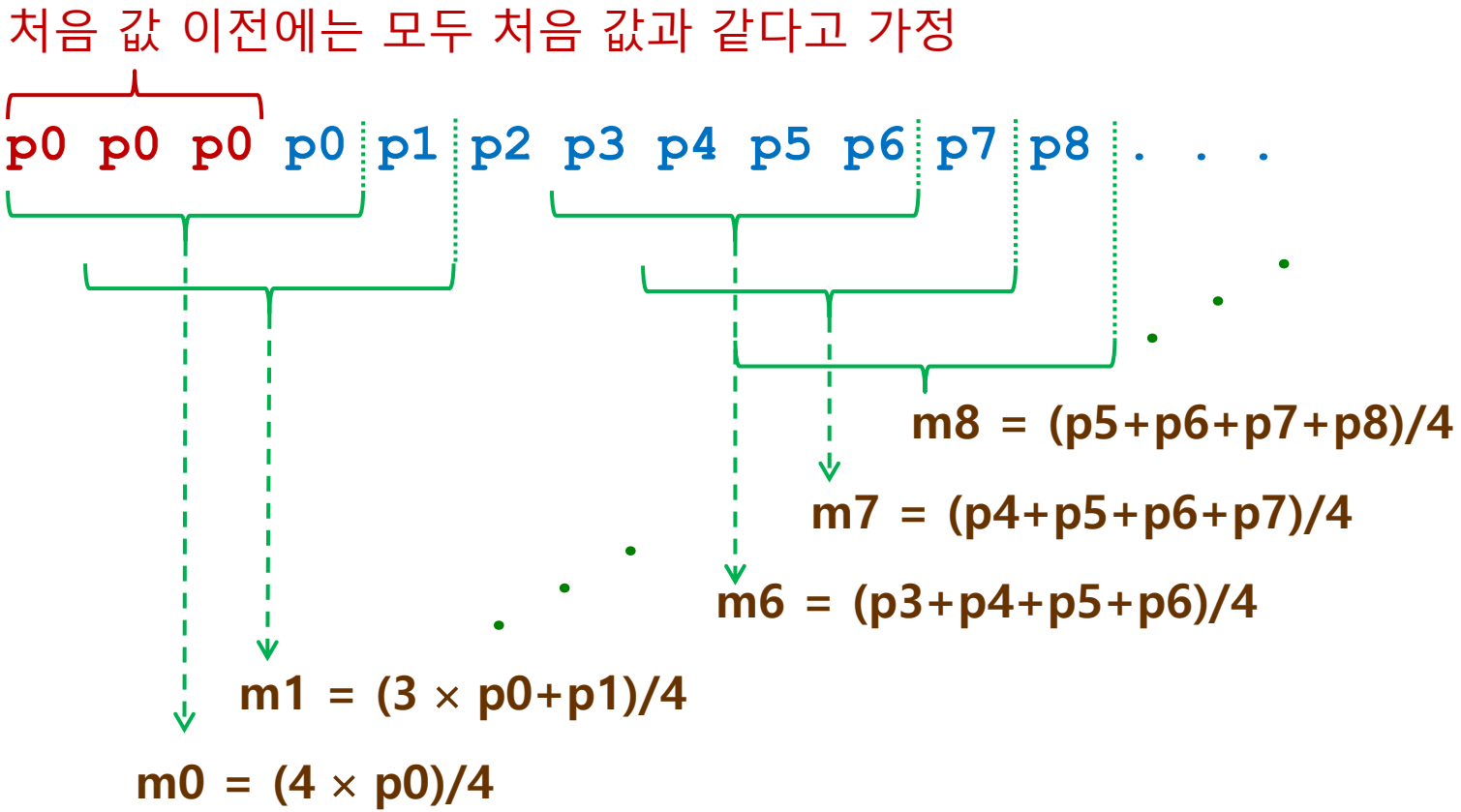
✦스택을 사용한 문자열 복원 함수

```
def decodeString_stack(string) :
    """ Recover a string from a keylog string
        입력: string(문자열), 출력: 복원된 문자열 반환
        deCoded[ ] : 복원된 문자열 리스트(커서 좌측)
        temp[ ] : 복원된 문자열 리스트(커서 우측)
    """
    deCoded = []; temp = []   # 두 개의 스택 마련

    for ch in string :    # string의 각 문자에 대해
        pass              #   20, 21 쪽의 경우에 따른 처리

    temp의 문자들을 deCoded에 추가(*)   # 두 리스트를 합친다

    return ''.join(deCoded) # .join()을 사용하여 문자열로 변환
```

(*) temp가 빈 리스트가 될 때까지 temp의 마지막 원소를 제거하여 deCoded의 마지막에 추가한다.

큐 응용 예: 이동 평균

- 이동 평균(Moving Average)
  - 현재 값에 일정수의 이전 값을 더하여 평균을 구한 값.
  - 패턴 인식 (4일 이동 평균의 경우)

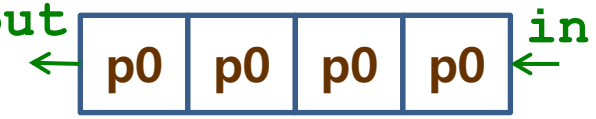


- 이동 평균 계산에 큐(queue)를 이용하면 계산이 편리하다.
- Example (4일 이동 평균의 경우)
  - 초기 설정 및 첫 번째 이동 평균 계산

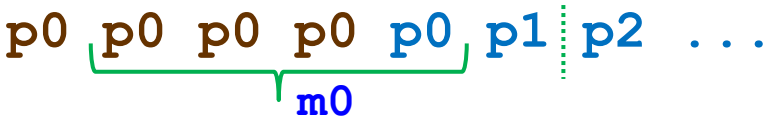
변수 mSum에 초기 값으로 p0 * 4를 저장.
큐 Q에는 p0를 네 개 저장.

mSum = p0+p0+p0+p0

Q

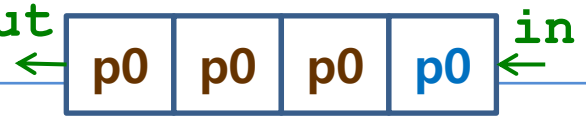


이제 m0를 계산하자.



1. Q에서 하나를 제거하여(p0) mSum에서 이를 뺀다.
2. 리스트의 현재 값(p0)을 mSum에 더한 후 평균을 구한다.
3. 리스트의 현재 값(p0)을 Q에 넣는다.

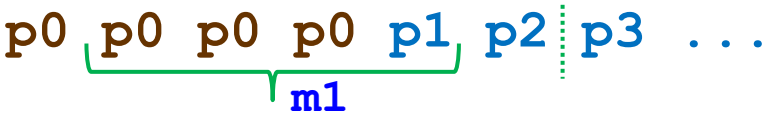
```
mSum = mSum - Q.pop(0)   # mSum = p0+p0+p0
mSum = mSum + p0         # mSum = p0+p0+p0+p0
m0 = mSum/4              # m0를 이동 평균 값 리스트에 추가
Q에 p0를 추가             # Q
```

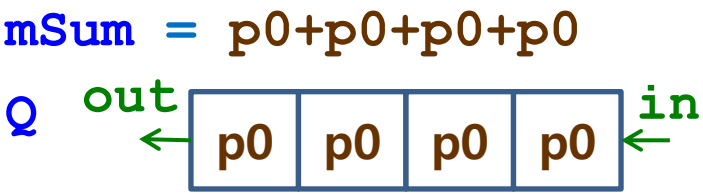


◆두 번째 이동 평균 계산

변수 mSum에는 현재 값 이전 네 개의 값의 합이 저장되어 있다.
큐 Q에는 현재 값 이전 네 개의 값이 저장되어 있다.

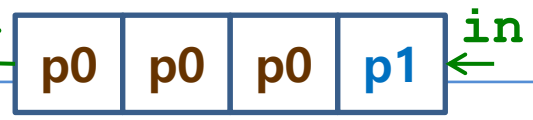
이제 m1을 계산하자.



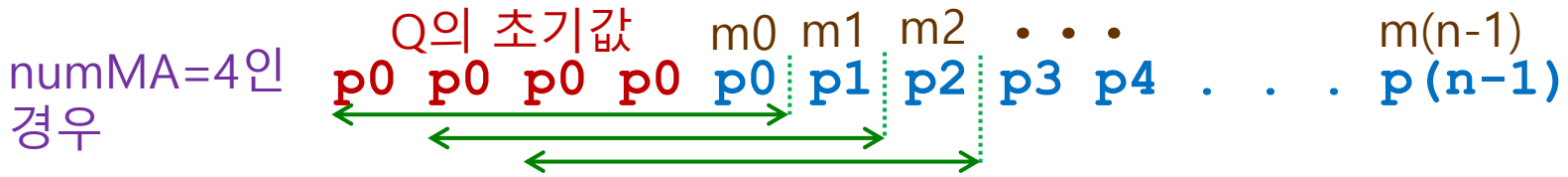


1. Q에서 하나를 제거하여(p0) mSum에서 이를 뺀다.
2. 리스트의 현재 값(p1)을 mSum에 더한 후 평균을 구한다.
3. 리스트의 현재 값(p1)을 Q에 넣는다.

```
mSum = mSum - Q.pop(0)   # mSum = p0+p0+p0
mSum = mSum + p1         # mSum = p0+p0+p0+p1
m1 = mSum/4              # m1을 이동 평균 값 리스트에 추가
Q에 p1을 추가             # Q
```



m2, m3, ... 등의 계산도 위 과정을 반복하면 된다.



✦이동 평균 계산 알고리즘

```
# 입력: pList(정수 리스트), numMA(이동평균을 위한 개수)
1.  mList = list()  # 이동 평균 값 저장을 위한 빈 리스트
2.  Q = list()      # 빈 큐
3.  mSum = pList[0] * numMA # 초기 값 설정
4.  Q에 pList[0]을 numMA 개 추가
5.  for pList의 정수 M에 대해
6.     mSum = mSum - Q.pop(0) # Q에서 제거한 값을 mSum에서 뺀다
7.     mSum = mSum + M        # mSum에 현재 값을 더한다
8.     mList에 (mSum/numMA)를 추가  # 이동평균계산 및 추가
9.     Q에 M을 추가
10. return mList
```

matplotlib (http://matplotlib.org/)

- Matplotlib는 각종 데이터를 차트로 보이기에 적합한 패키지인데, 터틀 그래픽보다 많은 강력한 기능을 갖추고 있다.
- Plotting을 위한 기본적인 함수는 다음과 같다(1).

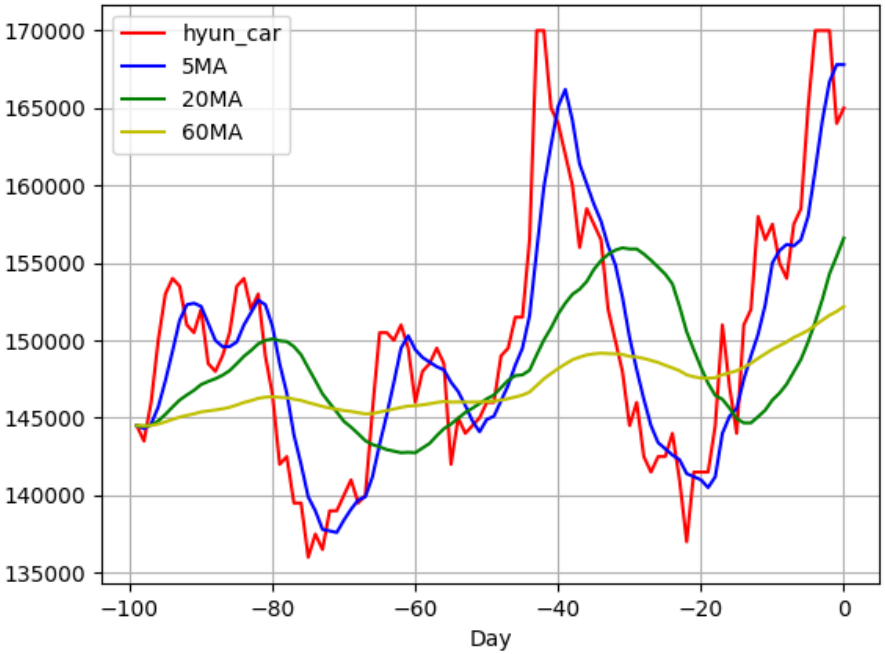
function	Description(인수 x(축), y(값) : 리스트)
plot(y [, 'r/g/b', label='xxx'])	y 리스트의 값들로 그래프를 그린다. x축은 0,1,2, …, N-1로 자동 설정. 'r'은 빨강, 'g'는 녹색, 'b'는 파랑 등 color(2) . 'xxx'는 그래프 이름.
plot(x, y [, 'r/g/b', label='xxx'])	y축 좌표 값이 주어질 경우(나머지는 상동)
xlabel('xxx'), ylabel('xxx')	x축과 y축의 레이블을 'xxx'로 설정
axis([xmin, xmax, ymin, ymax])	x축과 y축의 최소값, 최대값을 설정
grid (True/False)	그리드 선의 유무 결정
legend(loc = 'upper left')	범례를 지정된 위치에 추가한다.
show( )	그래프를 화면에 보여준다.

(2) [ ] 안의 내용은 옵션(즉, 없어도 된다)
(1) 칼라는 http://matplotlib.org/api/colors_api.html 참조

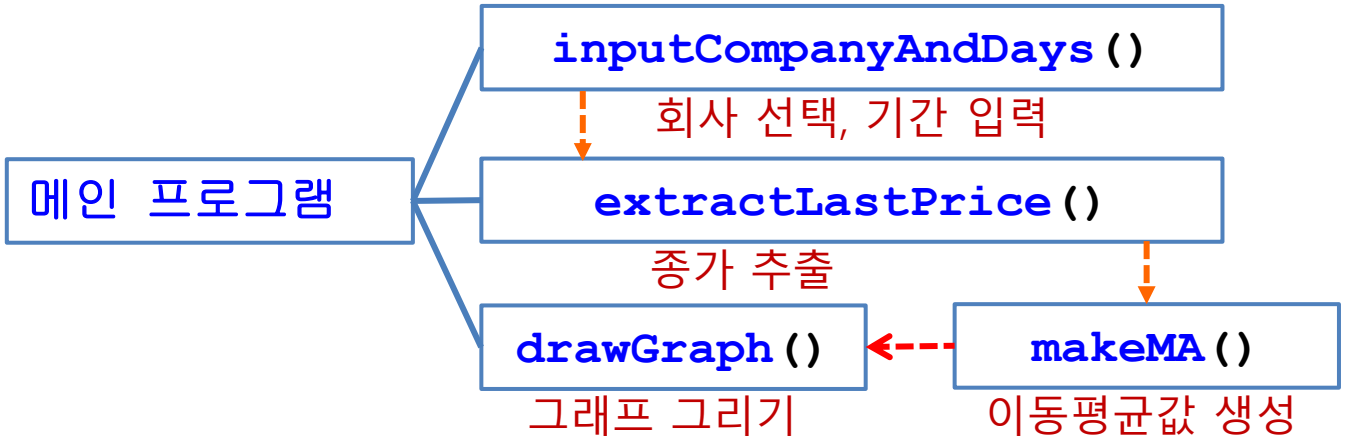
✦ 프로젝트3 설명

- 세개의 회사 중 하나를 선택하고 구하고자 하는 종가의 개수를 입력하면,
- 5일, 20일, 60일 이동 평균을 구하여 이를 도표로 보인다.
- 실행 예:

```
1:samsung, 2:lge, 3:hyun_car
회사를 고르세요 : 3
종가 추출 기간을 입력하세요(20의 배수가 되도록 상향 조정합니다) : 100
```



- 이 프로그램의 함수 구성은 다음과 같다.



- 함수는 앞에서 배운 부분을 참고하여 만든다.
- **extractLastPrice() - 18page**
- **makeMA() - 27page**
- **drawGraph() - 28page**