# Web Scraping – 고급

## 우리가 추출하고자 하는 정보가 원본 소스 코드에 들어있지 않는 경우

기본편에서 설명한 것 처럼 특정 웹페이지에서 우리가 원하는 정보를 파이썬을 이용해서 수집하는 경우에는 일반적으로 다음과 같은 과정을 거칩니다.

1. 원하는 정보를 담고 있는 웹 페이지의 소스 코드 다운로드 하기
2. 다운로드 받은 소스 코드에서 우리가 원하는 정보를 담고 있는 태그 찾기
3. 해당 태그에서 우리가 원하는 정보 추출하기

보통의 경우에는①의 목적으로는 requests라는 모듈을 그리고②, ③을 위해서는 BeatifulSoup을 사용합니다. 하지만, 이러한 방법으로 웹에 있는 모든 데이터를 다운로드 받을 수 있는 것은 아닙니다. 우리는 원하는 정보가 브라우저 상에서는 보이더라도 requests를 통해서 다운로드 받는 소스 코드에는 그러한 내용이 저장되어 있지 않는 경우가 있습니다. 이러한 경우에는 requests만을 사용해서는 우리가 원하는 정보를 Python을 이용해서 추출할 수가 없습니다. requests.get()을 통해서 얻는 소스 코드에는 그러한 정보가 들어있지 않기 때문입니다.

Tips!

requests를 통해서 다운로드 받아지는 소스 코드를 보통 원본 소스 코드라고 표현하는데, 그 이유는 requests를 통해서 다운로드 받는 소스 코드는 브라우저가 해당 페이지를 화면에 표시하기 위해서 서버로부터 받는 최초의 소스 코드라고 생가할 수 있기 때문입니다. 브라우저는 웹페이지를 사용자에게 보여주기 위해서 서버와 통신을 한번만 하는 것이 아니라, 원본 소스 코드를 다운로드 받은 이후에도 지속적으로 통신을 하게 되고, 그러한 통신을 통해 추가적인 소스 코드 또는 데이터를 다운로드 받습니다..

어떠한 웹페이지에 대해서 requests를 이용해서 다운로드 받는 (원본) 소스 코드는 해당 웹페이지의 소스 코드를 웹 브라우저상에서 보는 것과 동일합니다. 즉, 웹 브라우저 상에서 해당 페이지의 소스 코드를 확인했는데 그 안에 우리가 원하는 정보가 저장되어 있지 않다면 이는 requests만을 사용해서는 해당 정보를 파이썬에서 추출할 수 없다는 것을 의미합니다.

이러한 원본 소스 코드에 우리가 원하는 정보가 들어있지 않는 경우에는 크게 아래 두 가지 경우가 해당합니다.

1. 우리가 원하는 정보가 원본 소스 코드안에 포함 되어서 서버에서 저장되는 것이 아니라, 별도의 파일을 통해서 전달되는 경우 (별도의 파일은 고유의 url 주소를 갖고, 이러한 url 주소는 보통 원본 소스 코드 안에 저장되어 있습니다.)
2. 우리가 원하는 정보가 원본 소스 코드가 서버로부터 다운로드 받아진 이후 추가적인 서버와의 통신을 통해서 받아지는 경우

위와 같은 경우에는 requests만을 통해서는 우리가 원하는 정보를 추출할 수 없는 것입니다. 이를 위해서 사용하는 추가적인 방법이 브라우저에서 제공되는 네트워크 검사 (network inspection) 기능을 사용하거나 아니면 selenium 을 사용하는 것입니다. 네트워크 검사 기능은 보통 1의 경우에 사용되고 selenium은 2 경우에 사용됩니다. 네트워크 검사 방법은 우리가 원하는 정보를 저장하고 있는 파일을 찾는 것이 쉽지 않고, 해당 방법을 사용해서도 수집이 안되는 정보가 있기 때문에 여기서는 더 다양한 경우에 사용될 수 있는 selenium을 이용하는 방법에 대해서만 설명하도록 하겠습니다.

### Selenium 사용하기[[1]](#footnote-1)

우리가 추출하고자 하는 정보가 requests.get()을 통해서 다운로드 받는 원본 소스 코드에 포함되어 있지 않을 때 사용할 수 있는 방법이 selenium을 사용하는 것입니다. requests.get()을 통해서 다운로드 받는 원본 소스 코드에 우리가 원하는 정보가 들어있지 않은 가장 큰 이유 중 하나는 해당 정보가 브라우저가 해당 웹페이지를 디스플레이하기 위해서 최초에 다운로드 받는 소스 코드가 아니라 이후에 실시간으로 통신을 하면서 추가적으로 다운로드 받는 데이터 (또는 소스 코드)에 포함되어 있게 때문에 그렇습니다. 즉, 브라우저는 하나의 웹페이지를 디스플레이하기 위해서 서버와 한번만 통신하는 것이 아니라 지속적으로 실시간으로 통신을 하면서 필요한 내용을 추가적으로 다운로드 받습니다. 하지만, 원본 소스 코드 (즉, 브라우저가 처음 서버로부터 다운로드 받는 소스 코드)에는 이러한 실시간으로 추가적으로 다운로드 받는 내용이 포함되어 있지 않습니다. 따라서 우리가 최종적으로 추출하고자 하는 데이터가 이렇게 실시간으로 서버로부터 다운로드 받아지는 데이터라면, requests.get()을 가지고는 추출할 수가 없습니다.

이러한 경우에 사용할 수 있는 방법이 selenium을 사용하는 것입니다. Selenium은 원래 컴퓨터 프로그래밍 언어를 통해서 특정한 웹브라우저를 통제하고자 할 때 사용할 수 있는 프로그램입니다. 우리는 이러한 selenium을 이용해서 특정한 웹페이지의 소스 코드를 서버로부터 실시간으로 다운로드 받을 수 있습니다. 이렇게 실시간으로 다운로드 받는 소스 코드에는 우리가 브라우저의 웹페이지 상에서 볼 수 있는 내용이 담겨 있습니다. 즉, 우리가 추출하고자 하는 정보를 포함하고 있는 것입니다.

**Selenium이 작동하는 방식 이해하기**

사용자가 특정한 웹페이지를 방문하기 위해서는 일련의 행동들을 합니다. 예를 들어, 브라우저를 실행시키고 접속하고자 하는 웹페이지의 URL을 주소창에 입력하여 해당 페이지로 이동한 다음에, 해당 페이지에서 특정한 링크나 아이콘을 클릭해서 원하는 정보를 담고 있는 페이지로 이동할 수가 있습니다. 그리고 최종 페이지에서 사용자가 원하는 정보를 브라우저를 통해서 볼 수가 있는 것입니다.

Selenium은 이러한 사용자의 행동을 흉내내어 특정한 웹페이지를 브라우저상에 표현하는데 필요한 데이터를 서버로부터 실시간으로 다운로드 받습니다. Selenium이 실시간으로 다운로드 받는 이러한 데이터에는 우리가 추출하고자 하는 정보가 포함되어 있습니다.

**Selenium 설치**

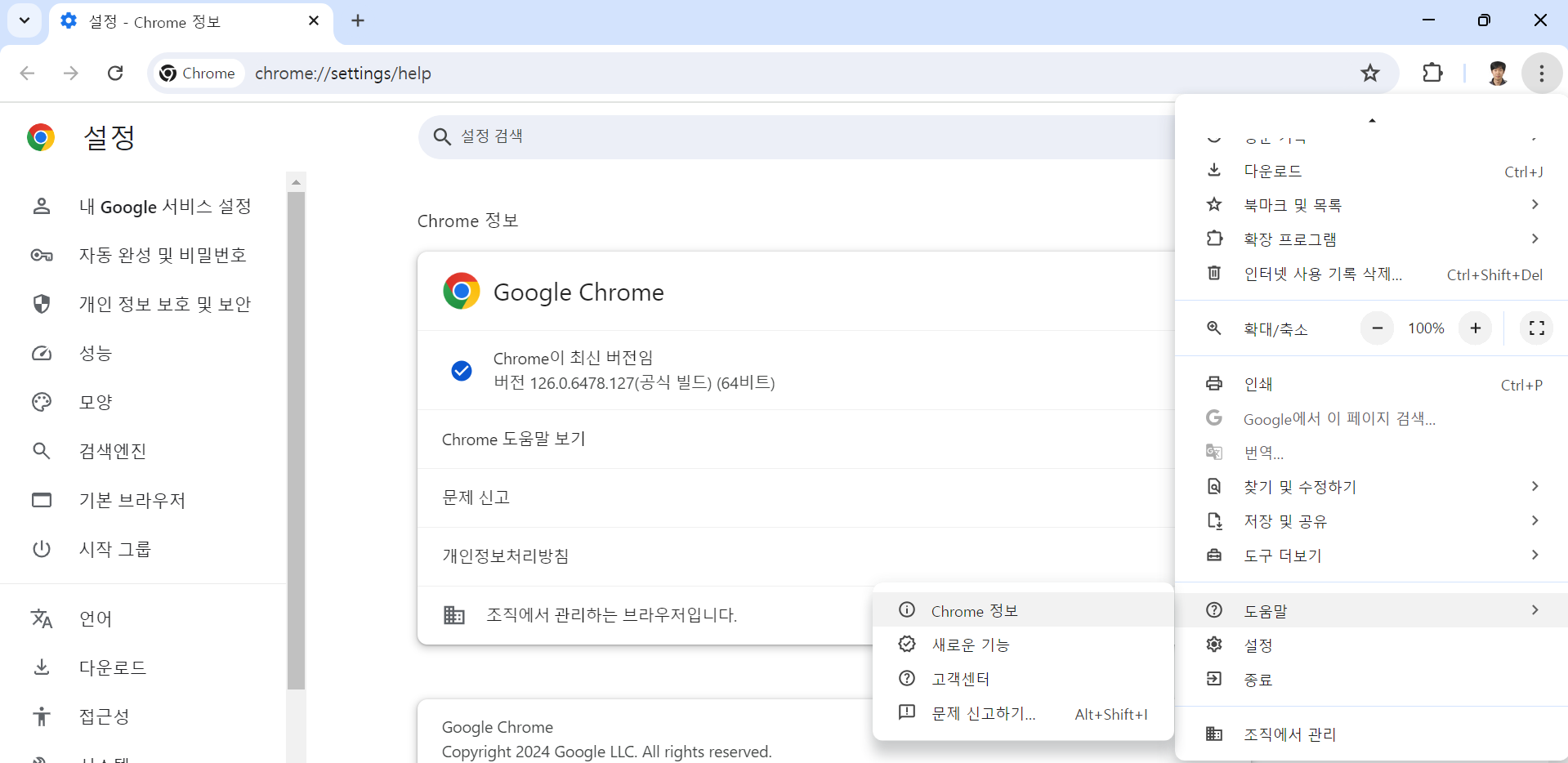
Selenium을 파이썬에서 사용하기 위해서는 selenium이라는 모듈을 먼저 설치해야 합니다. Anaconda를 통해서는 기본적으로 설치가 되지 않기 때문에 selenium은 사용자가 추가적으로 설치를 해줘야 합니다. 이는 아래와 같은 명령 프롬프트 창에서 다음과 같은 명령어를 실행시켜 할 수가 있습니다.

pip install selenium

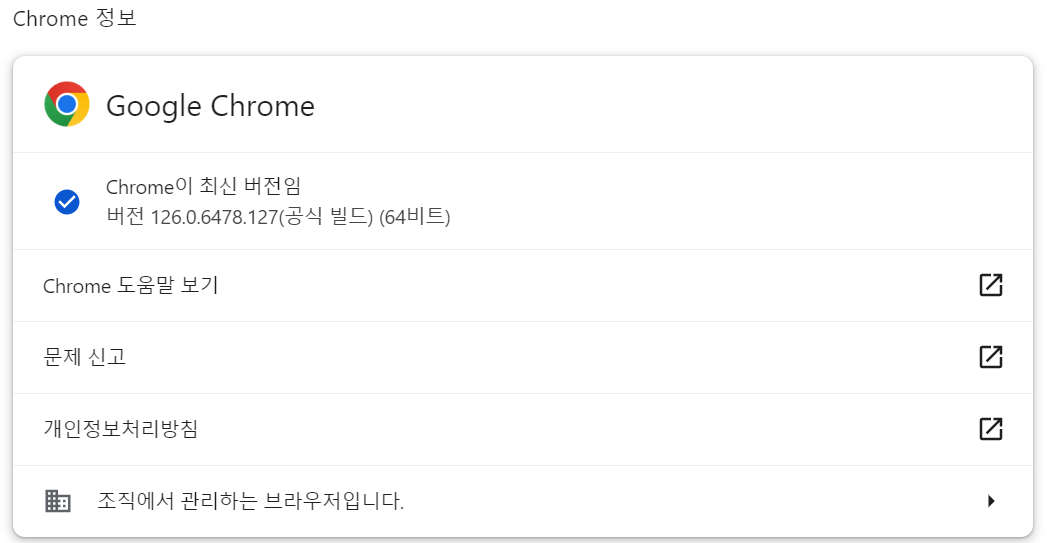
Selenium을 설치한 이후에는 해당 모듈을 Python에서 import해서 사용하면 됩니다. 아래와 같이 import를 할 수가 있습니다.

from selenium import webdriver

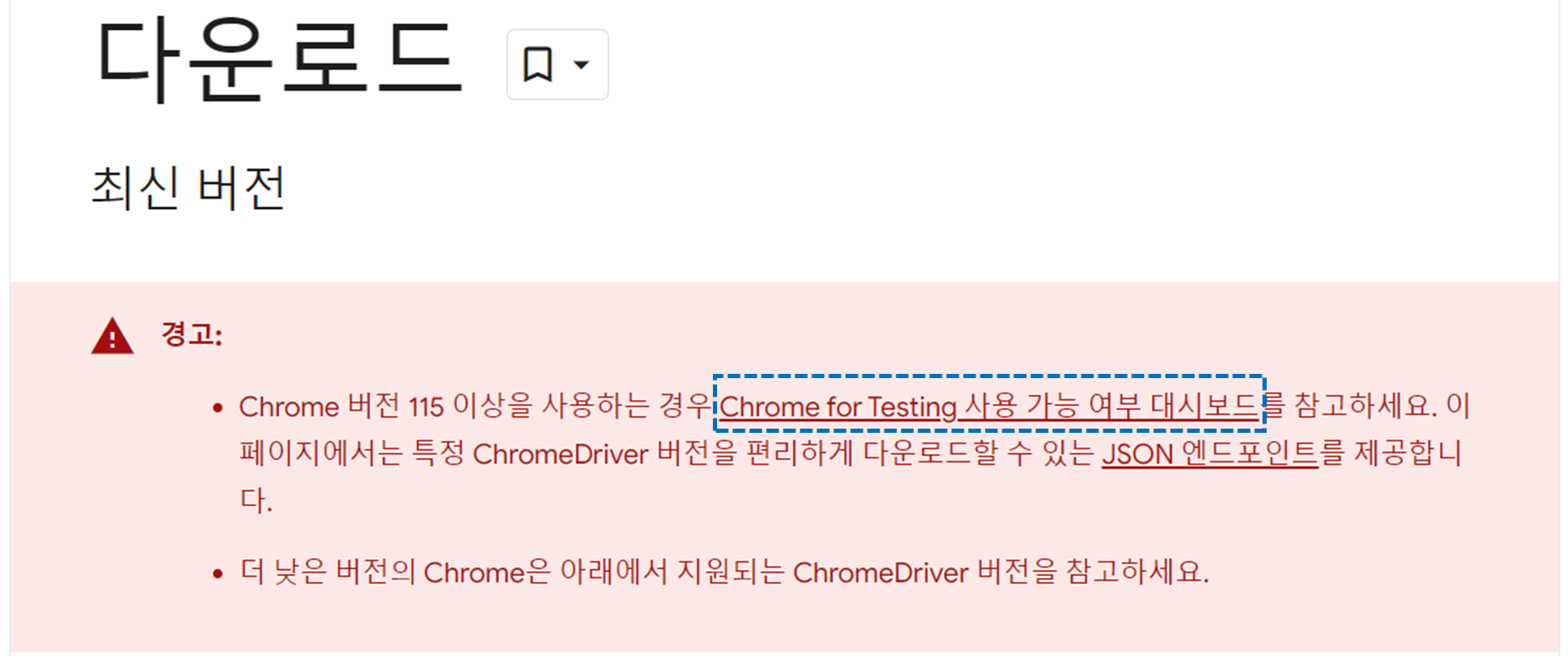
Selenium 모듈에 포함되어 있는 모든 하위 모듈을 사용하는 것이 아니라 기본적으로는 webdriver라는 모듈만을 사용합니다. 즉, webdriver 모듈을 사용해서 특정한 웹 브라우저를 통제할 수가 있습니다. webdriver를 이용해서 특정한 브라우저를 통제하기 위해서는 selenium을 이용해서 통제하고자하는 브라우저의 드라이버를 먼저 다운로드 받아야 합니다. 여기서는 selenium을 이용해서 크롬 브라우저를 통제하는 방법에 대해서 설명하도록 하겠습니다. 즉, selenium의 webdriver 모듈을 통해서 Chrome 브라우저를 통제하기 위해서는 먼저 Chrome 브라우저의 드라이버를 다운로드 받아야 합니다. 이를 위해서 <https://chromedriver.chromium.org/downloads> 사이트로 이동합니다. 여기서 여러분이 사용하는 크롬 브라우저의 버전에 맞는 버전의 드라이버를 다운로드 받으면 됩니다. 크롬 브라우저의 버전은 아래와 같이 확인합니다.



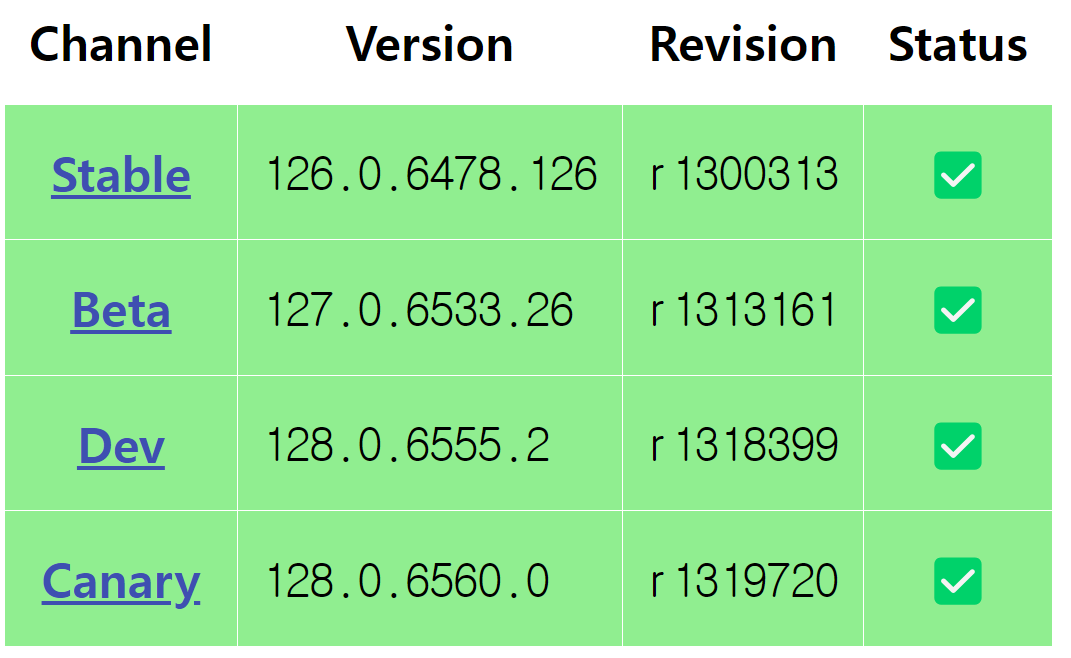
위의 Chrome 정보를 클릭하면 아래와 같이 버전 정보가 나옵니다.



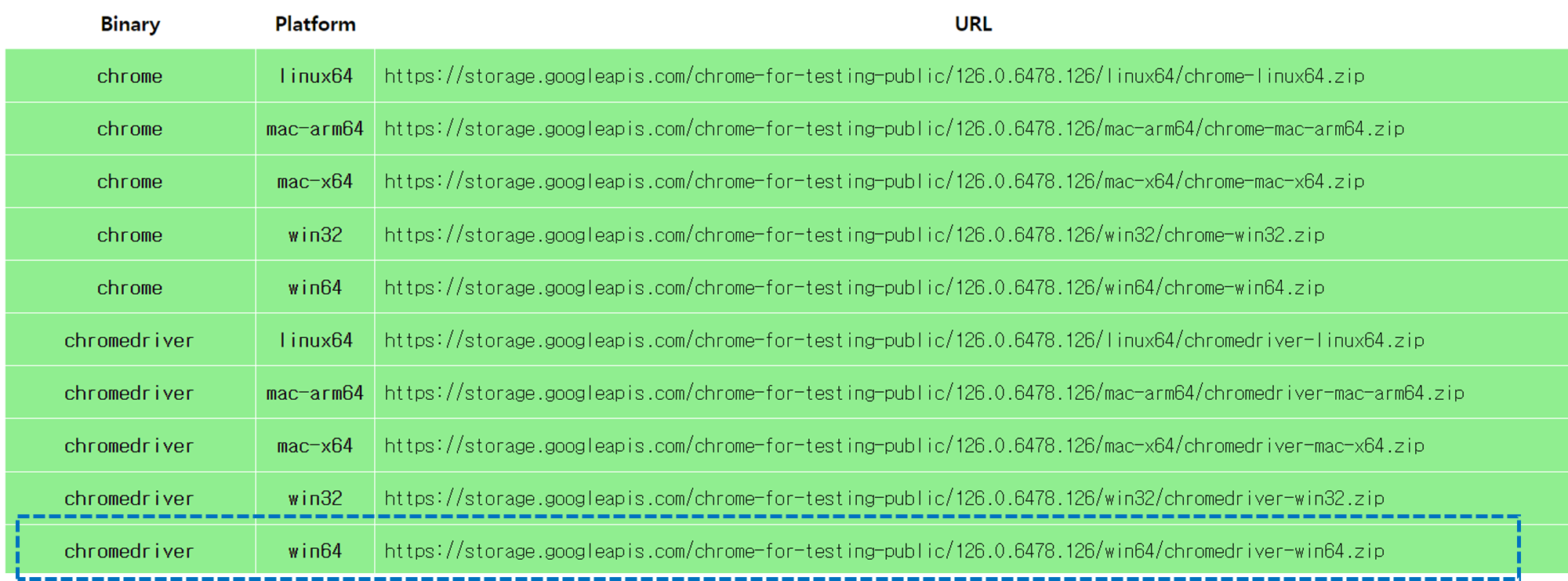
크롬 드라이버 다운로드 페이지에서 위의 버전에 맞는 버전의 드라이버를 다운로드 받습니다. 126 버전을 사용하고 있기 때문에, 126에 해당하는 크롬 드라이버 파일을 다운로드 받아야 합니다 (여러분이 사용하는 버전은 다를 수 있습니다). <https://chromedriver.chromium.org/downloads>에 접속하게 되면 아래 화면이 나옵니다. 여러분이 사용하는 크롬 브라우저의 버전이 115 이상인 경우에는 ‘Chrome for Testing 사용 가능 여부 대시보드’ 링크를 클릭해서 드라이버 파일을 다운로드 받습니다.



해당 링크를 클릭하면 아래와 같은 화면이 나오는데 여기에서 여러분의 크롬 드라이버 버전에 맞는 링크를 클릭합니다. 여기서는 ‘Stable’ 링크를 클릭합니다.



‘Stable’ 링크를 클릭하면 아래의 페이지가 나오는데 여기서 여러분 컴퓨터 운영체제에 맞는 zip 파일을 다운로드 받습니다. 윈도우의 경우는 win64에 해당하는 파일을 다운로드 받습니다.



zip파일의 압축을 풀어서 .exe파일을 특정한 폴더에 저장합니다. 이 때 폴더의 이름을 기억해야 합니다. 해당 드라이버를 가지고 Chrome 브라우저를 통제할 것이라고 파이썬에서 명시를 해줘야지만 우리가 selenium을 통해서 Chrome 브라우저를 통제할 수가 있습니다. 이는 아래와 같이 할 수가 있습니다. 파이썬 파일은 selenium\_example.ipynb을 참고하세요.

from selenium import webdriver # selenium의 webdriver 모듈을 임포트 합니다.

from selenium.webdriver.chrome.service import Service

# chromedriver.exe 파일의 경로를 입력합니다.

s = Service(r'C:\Users\Sang\chromedriver-win64\chromedriver.exe')

# chromedriver.exe가 저장되어 있는 폴더의 전체 경로입니다. 여러분이 해당 파일을 다른 경로에

# 저장했다면 경로명은 다르게 됩니다.

driver = webdriver.Chrome(service=s)

이렇게 하면 새로운 Chrome 브라우저가 실행이 되고, driver라는 변수를 통해서 해당 Chrome 브라우저를 통제할 수가 있습니다. 즉, driver 변수를 통해서 브라우저상에서 사람이 하는 행동을 흉내낼 수 있는 것입니다.

브라우저를 위와 같이 실행한 이후에 특정한 웹페이지로 이동하기 위해서는 selenium에서 제공되는 get()함수를 사용합니다. requests의 get()함수와 비슷하게 접속하고자 하는 웹페이지의 URL 주소를 selenium get()함수의 인자로 입력합니다.

예를 들어서 우리가 최종적으로 접속하고자 하는 웹 페이지가 예스24에서 제공되는 베스트셀러 중 첫번째 책에 대한 페이지라고 가정하겠습니다 (아래 그림 참고).



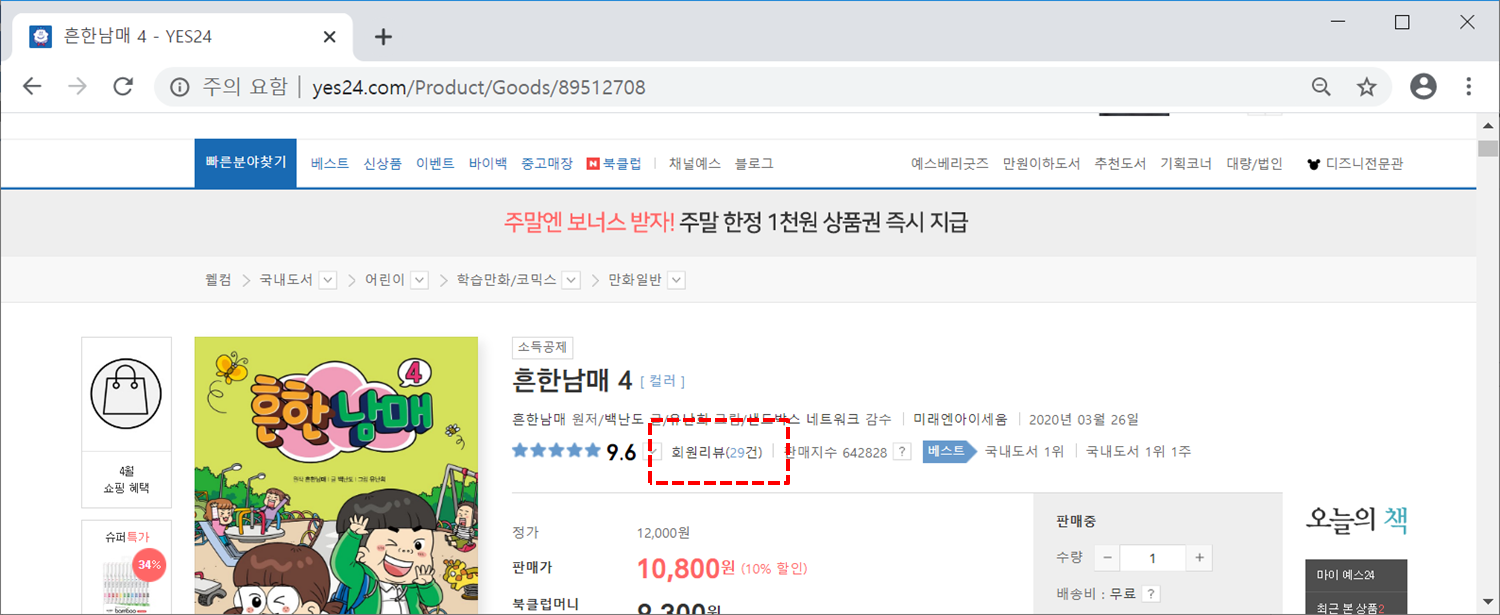
위 책에 대한 페이지는 아래와 같습니다. 여기에서 우리가 추출하고자 하는 정보가 책에 대한 리뷰 정보라고 하겠습니다 (아래 그림 참고).



해당 리뷰 정보를 selenium을 사용해서 추출해 보겠습니다. selenium을 사용하기 전에 여러분들이 먼저 생각해봐야 하는 것이 무엇이냐하면, 해당 페이지로 어떻게 이동할 수 있느냐 하는 것입니다. 해당 페이지의 주소를 알면 주소 정보를 갖고 바로 이동할 수 있습니다. 하지만, 해당 페이지의 주소 정보를 모르는 경우에는 어떻게 해야 할까요? 여러분은 아마도 다음과 같은 과정들을 통해서 해당 페이지로 이동할 것입니다.

1. 브라우저 상에서 [www.yes24.com](http://www.yes24.com) 을 입력해서 예스24의 메인 페이지로 이동하기
2. 메인 페이지에서 '베스트' 를 클릭하여 베스트셀러 정보가 나오는 페이지로 이동하기
3. 베스트셀러 페이지에서 우리가 원하는 책인 첫번째 책에 대한 링크를 클릭하여 해당 페이지로 이동하기

해당 페이지로 이동한 다음 책에 대한 리뷰를 보기 위해서는 아래와 같이 '회원리뷰' 링크를 클릭합니다.



Selenium을 이용해서 여러분이 원하는 책에 대한 리뷰 정보를 가져오기 위해서는 위의 과정들을 selenium을 이용해서 수행해야 합니다.

**Selenium을 이용해서 리뷰 정보 가져오기**

① 일단 아래 코드를 사용해서 크롬 브라우저를 실행합니다.

from selenium import webdriver # selenium의 webdriver 모듈을 임포트 합니다.

from selenium.webdriver.chrome.service import Service

# chromedriver.exe 파일의 경로를 입력합니다.

s = Service(r'C:\Users\Sang\chromedriver-win64\chromedriver.exe')

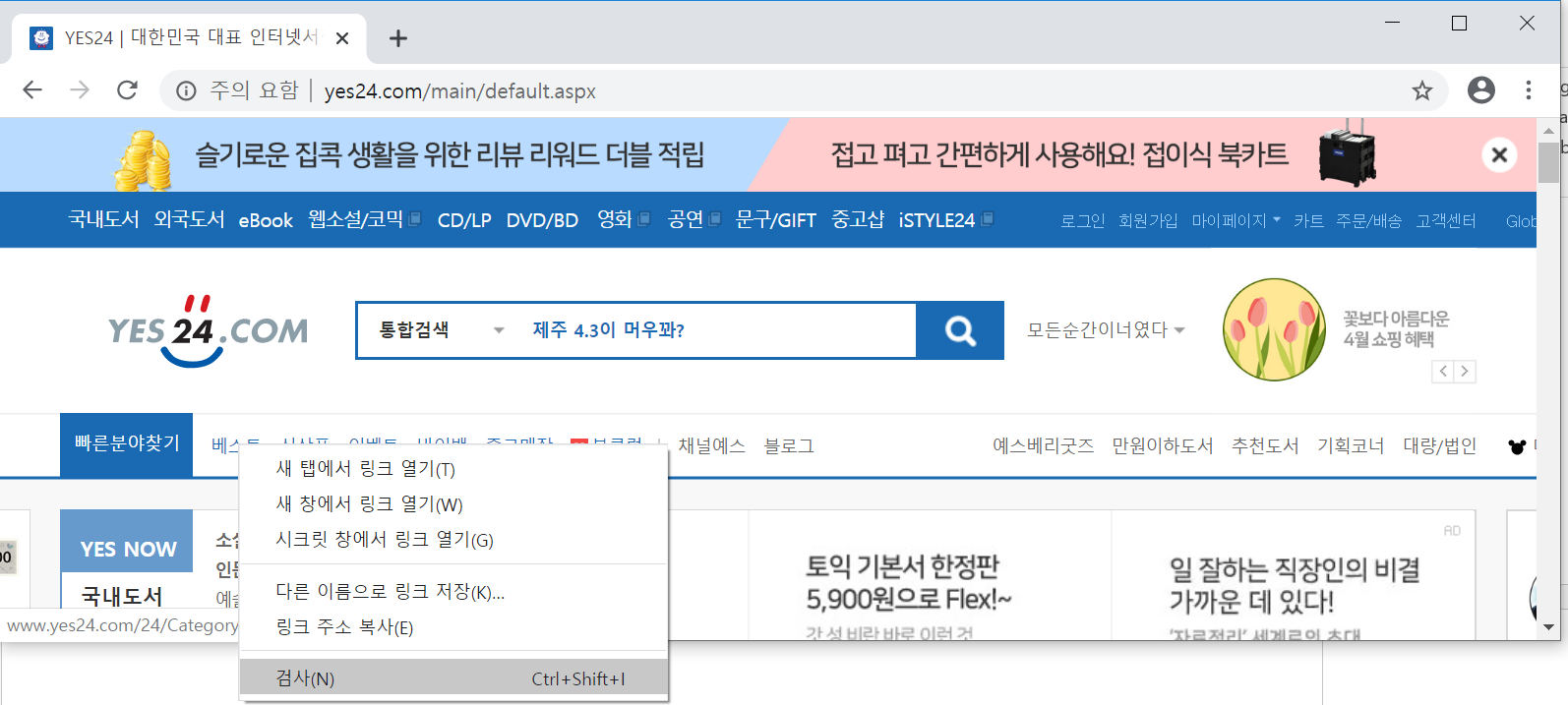
driver = webdriver.Chrome(service=s)

② 다음 코드를 이용해서 예스24 메인 페이지로 이동합니다.

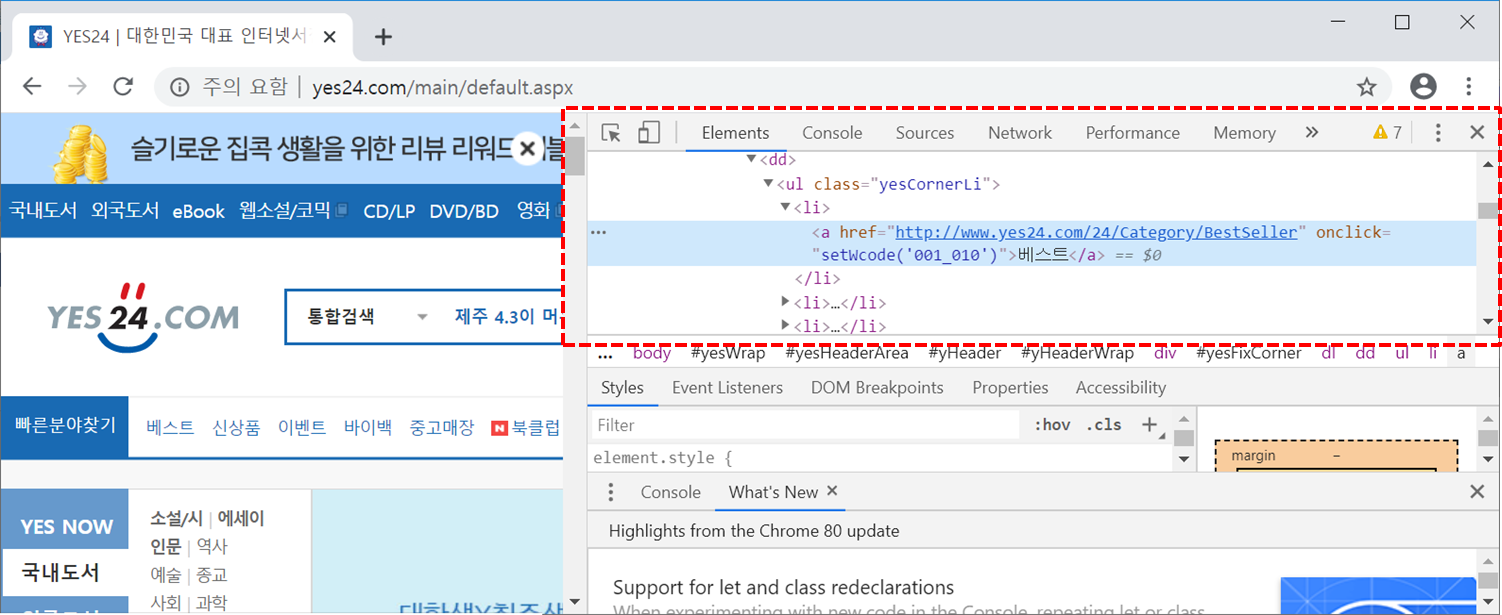
url = 'http://www.yes24.com'  
driver.get(url)

③ 메인 페이지에서 '베스트' 버튼 클릭하기

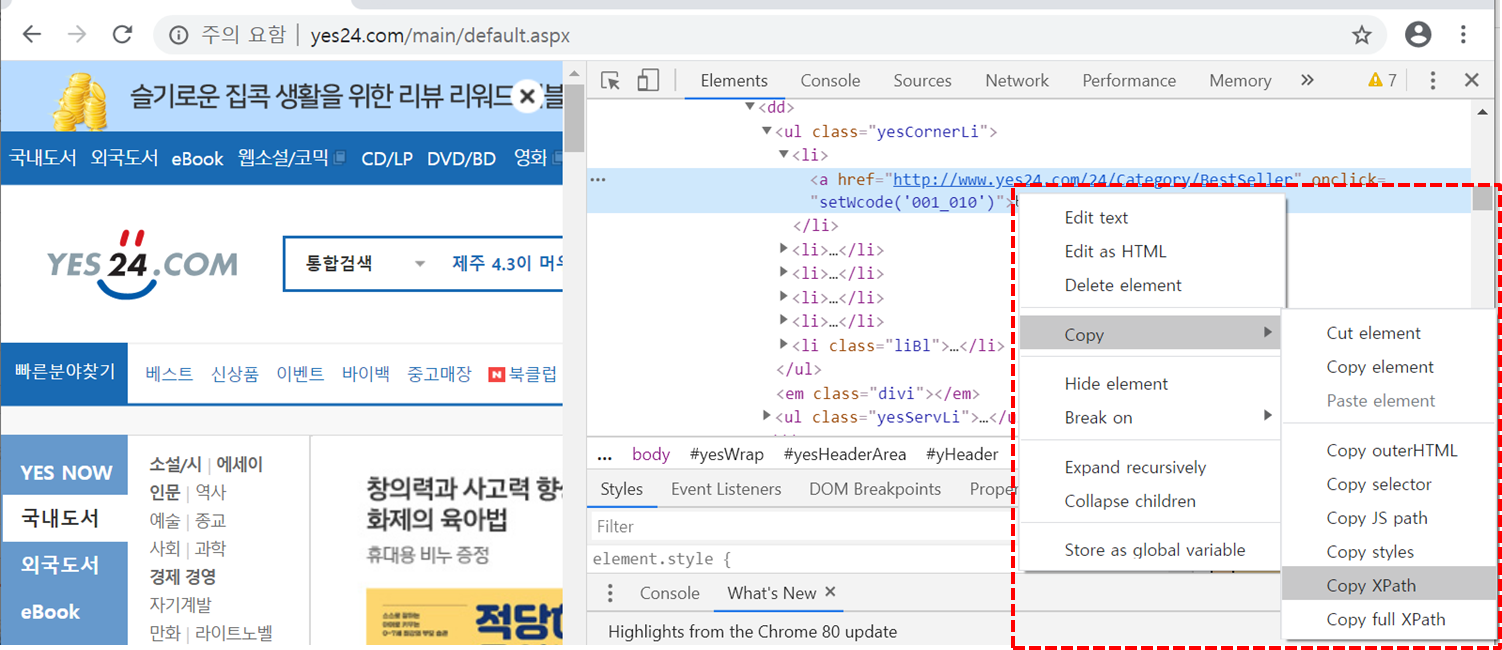
Selenium을 통해서 특정 링크나 버튼을 클릭하기 위해서는 먼저 해당 링크나 버튼을 찾아야 합니다. 웹페이지를 구성하는 링크, 버튼 등을 우리는 원소(element)라고 부릅니다. 그리고 각 element는 각자 고유한 정보를 가지고 있습니다. 주소 정보, ID 정보 등이 포함됩니다. 여기서는 각 element가 갖는 고유의 주소 정보를 사용해서 해당 element에 찾고, 접근해 보겠습니다. 이러한 고유한 주소를 xpath라고 합니다. 특정 링크나 버튼의 xpath 주소를 찾기 위해서는 크롬 브라우저를 사용합니다. 이를 위해서 먼저 아래 그림과 같이 우리가 클릭하고자 하는 링크 (혹은 아이콘 또는 버튼)를 마우스 오른쪽 버튼을 사용하여 클릭합니다.



나오는 메뉴 중에서 '검사(N)'을 클릭합니다. 그러면 아래와 같은 화면을 볼 수 있습니다. 해당 링크는 결국 하나의 태그를 의미합니다. 우리가 해야 하는 것은 이 태그의 xpath 정보를 확인하는 것입니다. 이를 위해서 해당 태그 (아래 그림에서 하이라이트된 태그)를 다시 마우스 오른쪽 버튼을 이용하여 클릭합니다.



마우스 오른쪽 버튼을 클리하여 나오는 메뉴 중에서 'Copy'를 선택하고, 다시 하위메뉴 중에서 'Copy XPath'를 클릭합니다. 그러면 해당 링크의 XPath 정보가 복사됩니다.



해당 정보를 selenium에서 제공되는 find\_element\_by\_xpath() 함수의 인자로 아래와 같이 전달합니다.[[2]](#footnote-2) find\_element\_by\_xpath() 함수는 element (여기서는 '베스트' 링크)를 xpath를 이용해서 찾을 때 사용합니다.

1. element = driver.find\_element(by=By.XPATH, value='//\*[@id="yesFixCorner"]/dl/dd/ul[1]/li[1]/a')

우리가 클릭하고자 하는 element를 find\_element() 함수를 이용해서 찾은 다음에는 클릭을 해야 합니다. 이는 아래 코드를 이용해서 수행합니다.

element.click()

그러면 해당 링크와 연결된 페이지로 이동합니다. 즉, 베스트셀러 페이지로 이동하게 되는 것입니다.

그 다음에는 앞에서 했던 과정을 반복합니다. 즉, 클릭하고자 하는 첫번째 책에 대한 링크를 find\_element() 함수를 이용해서 찾고, click() 함수를 이용해서 클릭합니다. 해당 링크의 XPath를 알기 위해서는 다시 해당 링크를 마우스 오른쪽 버튼을 클릭한 후 Copy > Copy XPath 메뉴를 선택합니다. 아래와 같이 코딩합니다.

element = driver.find\_element(by=By.XPATH, value='//\*[@id="bestList"]/ol/li[1]/p[1]/a')

element.click()

이제 마지막으로 우리가 원하는 정보를 보기위해서 '회원리뷰' 링크를 클릭합니다. 과정의 앞에서 설명한 것과 동일합니다. 아래와 같이 코딩할 수 있습니다.

# "회원리뷰" 링크 찾기

element = driver.find\_element(by=By.XPATH, value='//\*[@id="yDetailTopWrap"]/div[2]/div[1]/span[3]/span[2]/a')

element.click()

이러한 과정을 거쳐서 selenium을 통해 우리가 원하는 정보를 담고 있는 페이지로 이동한 후에 우리가 해야하는 것은 해당 페이지의 소스 코드를 실시간으로 서버로부터 다운로드 받는 것입니다. 이는 다음 코드를 사용합니다.

html = driver.page\_source

이렇게 하면 html안에는 우리가 원하는 정보를 포함하고 있는 소스 코드가 저장되어 있습니다. 여기까지가 selenium의 임무입니다. selenium을 다 사용하고 난 후에는 해당 브라우저를 종료해야 합니다. 이를 위해서는 아래 코드를 사용합니다.

driver.close()

소스 코드를 다운로드 다음에는 최종적으로 추출하고자 하는 내용 (여기서는 회원리뷰 정보)을 BeautifulSoup을 통해서 추출하면 됩니다. 즉, selenium을 이용해서 다운로드 받은 소스 코드에 저장되어 있는 태그들 중에서 우리가 원하는 정보를 저장하고 있는 태그를 BeautifulSoup을 이용해서 찾고, 그 내용을 추출하면 되는 것입니다.

1. 관련 정보는 <https://selenium-python.readthedocs.io/> 에서 확인할 수 있습니다. [↑](#footnote-ref-1)
2. element를 찾을 때 사용할 수 있는 함수의 목록은 https://selenium-python.readthedocs.io/navigating.html 에서 확인할 수 있습니다. [↑](#footnote-ref-2)