

파이썬을 이용한 교보문고 크롤러 제작

크롤러란: 웹 상의 데이터를 자동으로 복사해오게 코딩해 놓은 프로그램.

우리가 만들 크롤러는 교보문고의 상품 데이터 일부를 복사해서 mysql에 저장해두는 파이썬 프로그램. Selenium이라는 크롤링 파이썬 라이브러리를 이용해서 만들 것이며 크롬 브라우저를 이용해서 작동시킬 것입니다.

아래의 내용은 **설치부터 코드 돌려보기까지의 7단계 설명**.

1. 파이썬 설치

2. pycharm 설치(<https://www.jetbrains.com/pycharm/>)

pycharm에서 새 프로젝트 만들기

3. Pycharm으로 파이썬 라이브러리들인 selenium, pymysql, cryptography 설치하기

참고자료 <https://tariat.tistory.com/692>

Selenium은 크롤링하는데 사용하는 라이브러리, pymysql은 mysql의 데이터를 python으로 이용하는데 필요한 라이브러리, cryptography는 pymysql 사용에 필요한 dependency.

4. chromedriver 다운

<https://chromedriver.chromium.org/downloads> 에서 자신이 설치한 크롬버전에 맞게 window용 chromedriver 다운로드하기. 크롬버전 체크는 크롬메뉴 -> 도움말 -> Chrome 정보.

ex) 크롬버전이 80.0.3987.163일 경우 이에 맞게 ChromeDriver도 80.0.3987.163를 win32.zip을 받은 뒤 압축을 풀고 파이썬 프로젝트가 있는 폴더에 놓는다.(.py 파일과 같은 위치)

5. mysql 설치 및 기본 스키마 준비

mysql을 설치한 뒤 mysql workbench를 이용해서 db에 접속한 뒤

```
-----  
create database bookstore;  
use database bookstore;
```

```
CREATE TABLE `book` (  
  `no` int(11) NOT NULL AUTO_INCREMENT,  
  `title` text,
```

```

`thumb` text,
`price` int(11) DEFAULT NULL,
`author` text,
`cat1` text,
`cat2` text,
`cat3` text,
`publisher` text,
PRIMARY KEY (`no`)
);

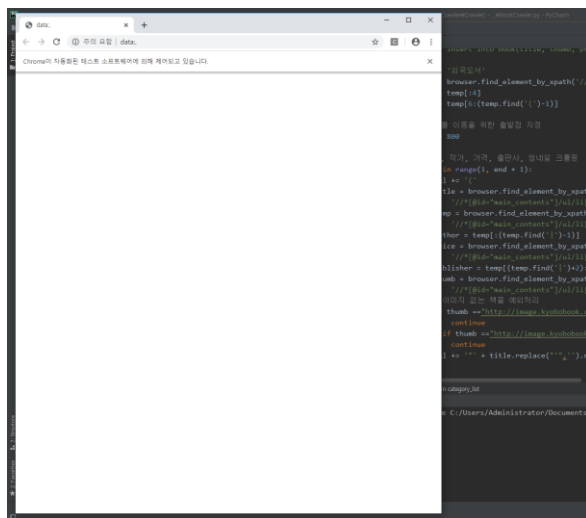
```

위의 명령어 전부를 실행시키고 성공적으로 실행되었는지 확인.

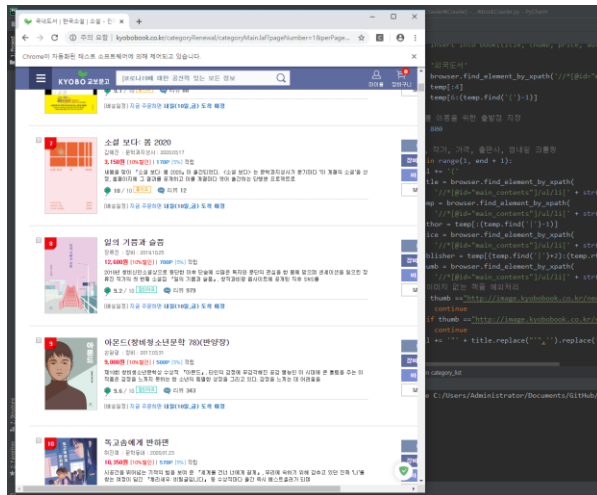
6. mysql과 파이썬 연결

설치된 mysql과 파이썬 코드를 연결.

7. 코드의 128번째줄의 password를 본인 mysql 비밀번호로 바꾸고 코드를 run한다.



코드실행 직후의 화면. 빈 화면의 크롬 브라우저가 자동으로 실행됨.



조금 지나면 교보문고 페이지가 로딩되고 조금씩 화면이 내려가며 도서 데이터를 복사하며, Mysql에서 select문을 통해 데이터베이스에 계속 추가됨을 확인할 수 있다.

여기부터는 파이썬 크롤러 코드에 대한 설명

```

9      # 국내도서 크롤링
10     def domestic(browser, category_list):...
63
64     # 해외도서 크롤링용
65     def foreign(browser, category_list):...
126
127     # pymysql
128     connect = pymysql.connect(host='localhost', user='root', password='4885', db='bookstore', charset='utf8')
129     cursor = connect.cursor(pymysql.cursors.DictCursor)
130
131     # 국내 도서의 카테고리들 (소설, 경제/경영, 정치/사회, 역사/문화)의 하위 카테고리들
132     category_list = list(range(101, 112, 2))
133     category_list.extend(list(range(1301, 1316, 2)))
134     category_list.extend(list(range(1701, 1716, 2)))
135     category_list.extend(list(range(1901, 1718, 2)))
136
137     # 크롬드라이버 위치설정
138     path = "./chromedriver.exe"
139     browser = webdriver.Chrome(path)
140
141     # 국내도서 크롤링용 주소, 마지막 linkClass에 무슨 값을 넣어주느냐에 따라 카테고리가 다름
142     addr = 'http://www.kyobobook.co.kr/categoryRenewal/categoryMain.laf?pageNumber=1&perPage=300&mallGb=KOR&linkClass='
143
144     # 국내도서 크롤링 시작
145     domestic(browser, category_list)
146
147     # 국내도서 끝, 해외도서 시작
148
149     # 해외도서 카테고리
150     category_list = list(range(1, 16, 2))
  
```

코드는 각각 국내도서와 해외도서를 크롤링하는 함수 **domestic**, **foreign**의 선언부와 127~160번째 줄까지의 실제로 코드가 실행되는 부분으로 구성됩니다.

실제로 코드가 실행되는 127번부터 설명하겠습니다.

128번줄은 python코드로 mysql에 접근하게 해주는 pymysql을 실행하는 함수입니다. 위에 설치 매뉴얼에서 말씀드린 대로 password부분을 본인 mysql 비밀번호에 맞게 바꾸셔야 합니다.

129번줄 코드는 pymysql로 mysql에 접근해서 실제로 무슨 동작을 시킬지를 나타내는 cursor를 만듭니다.

132번줄에서 135번줄 코드는 교보문고의 하위 카테고리들의 번호를 나타내기 위해 만든 리스트입니다.

우리가 만들려는 크롤러는 **교보문고의 각 3차 카테고리(ex, 국내도서->소설->한국소설)의 베스트 셀러들을 복사**하는 크롤러인데, 크롤러가 동작하려면 어느 url들을 방문해서 크롤링해야 하는지 미리 지정해줘야 해서 수십개나 되는 카테고리들의 url들을 전부 적기는 비효율적일 테니 각 카테고리의 url을 비교해서 패턴을 찾아냅니다.

<http://www.kyobobook.co.kr/categoryRenewal/categoryMain.laf?linkClass=0101&mallGb=KOR&orderClick=JAR>

을 들어가 보시면 국내도서->소설->한국소설 의 베스트셀러들을 보여줍니다.

<http://www.kyobobook.co.kr/categoryRenewal/categoryMain.laf?linkClass=0103&mallGb=KOR&orderClick=JAR>

또한 위의 링크를 누르시면 국내도서->소설->영미소설 의 베스트 셀러를 보여주는데 이 두 주소의 차이점은 주소 중간의 linkClass=의 숫자가 0101과 0103이라는 점인데 이런 식으로 url을 비교해보면 각 카테고리들의 url 패턴을 파악할 수 있습니다. 교보문고 한국도서의 경우는 linkClass가 몇 인지에 따라 카테고리가 다르고 mallGb가 무엇인지에 따라 어느 언어로 쓰여진 도서인지가 나옵니다.

이제 패턴을 알았으니 각 카테고리별 url을 만들 수 있습니다.

132번줄은 category_list라는 리스트를 만들어 국내도서->소설 카테고리의 하위 카테고리들의 카테고리 번호들을 저장할 건데, 위의 두 링크에서 보셨다시피 각 카테고리별로 +2씩 해주면 다음 번호이기에 101에서 111번까지 2씩 건너뛴 리스트 저장합니다.

133~135번줄 코드는 132번에서 만든 category_list에 소설이외의 다른 카테고리들의 카테고리 번호들을 추가해줍니다.

139번줄 코드는 크롬드라이버를 통해서 selenium 라이브러리를 이용해 크롤링을 하는데 사용할 실제 크롬화면인 browser 변수를 초기화하는 함수입니다. path에는 chromedriver의 위치가 들어갑니다.

142번의 addr은 132~135번에서 만들어 둔 카테고리들을 나타내는 linkClass 부분만 비워둔 교보 문고 url입니다. Category_list와 합쳐서 실제 각 카테고리들의 url을 나타내는데 사용할 겁니다.

여기서 간단한 팁

1. url에 있는 값들(앞에서 봤던 linkClass나 mallGb 같은)의 위치는 중요하지 않습니다. 그렇기에 linkClass부분을 제일 뒤로 두어 category_list와 합치기 편하게 합니다.
2. 교보문고의 경우 한 화면에 몇 개의 상품을 볼지를 정할 수 있는 버튼이 있는데

The screenshot shows the KYOBO 교보문고 website. At the top, there's a navigation bar with links like '로그인', '회원가입', '출석체크', '고객센터', '주문배송'. Below the navigation bar, there's a search bar and a 'KYOBO 교보문고' logo. The main content area displays a list of books. A red circle highlights the '20개씩 보기' (View 20 items) dropdown menu, which is used to change the number of items displayed per page. The page shows three book items with their titles, prices, and ratings.

이를 50개로 변경하면 URL에 **perPage=50**이라는 값이 추가됩니다. 즉, 저 값이 한 페이지당 몇 개를 보여줄지 결정하는 건데 **최대한 크롤러의 url 이동을 줄이기 위해** perPage 값을 최대한 늘립니다(베스트셀러는 300개만 나오기에 300으로 변경). 이렇게 변경해둔 url이 142번줄의 add 입니다.

이제 준비가 완료되었으니 145번줄에서 국내도서를 크롤링하는 domestic함수를 실행시킵니다.

domestic함수는 국내도서들의 3차 하위 카테고리들을 넘어서며 각 카테고리별 베스트셀러 도서들의 정보를 가져옵니다.

11번줄의 코드는 카테고리 번호들을 담은 category_list의 각 원소들과 addr를 합쳐서 각 카테고리별로 크롤러가 이동하게 하는 for문입니다. 앞에서 linkClass는 모두 4자리수였는데 0101같은 일반적인 수로는 표현할 수 없는 값이 있기에 자릿수를 맞춰 주기 위해 12~15번줄 코드를 추가했습니다. 1000мл의 수는 0을 앞에 추가해서 addr와 합치고, browser.get() 함수를 사용해서 해당 카테고리 사이트로 크롤러를 이동합니다.

18~24번줄 코드는 크롤러가 url로 이동한 뒤에 웹사이트가 로딩되는데까지 시간이 걸리기에 크롤링하려고 하는 부분이 로딩이 되었는지를 확인하며 기다리는 코드입니다. webDriverWait의 두번째 매개변수는 최대한 몇 초까지 기다릴지를 나타내며 10초정도면 다 로딩되기에 최대 10초를 기다리게 설정해 두었습니다.

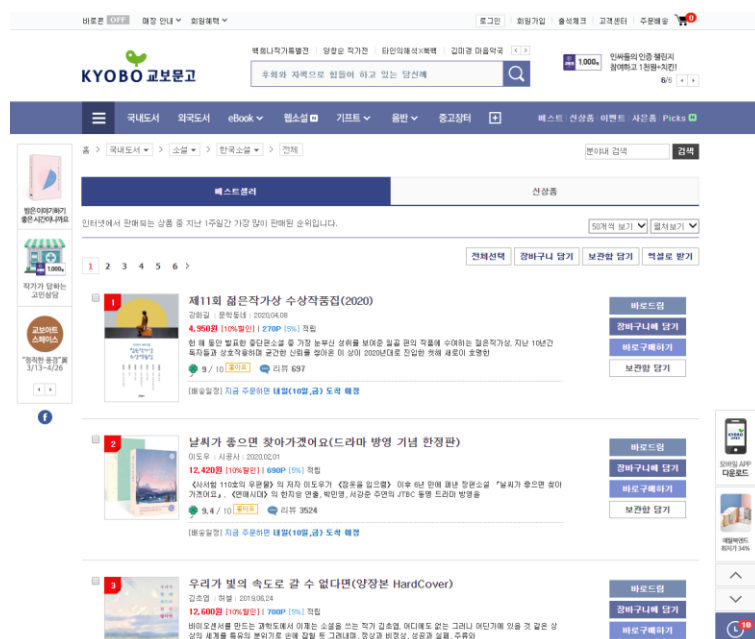
다음은 27번줄의 browser.find_elements_by_xpath 함수를 설명하겠습니다.

browser.find_elements_by_xpath함수는 현재 떠있는 브라우저 창의 요소들을 xpath라는 것을 이용해 찾아내는 함수입니다.

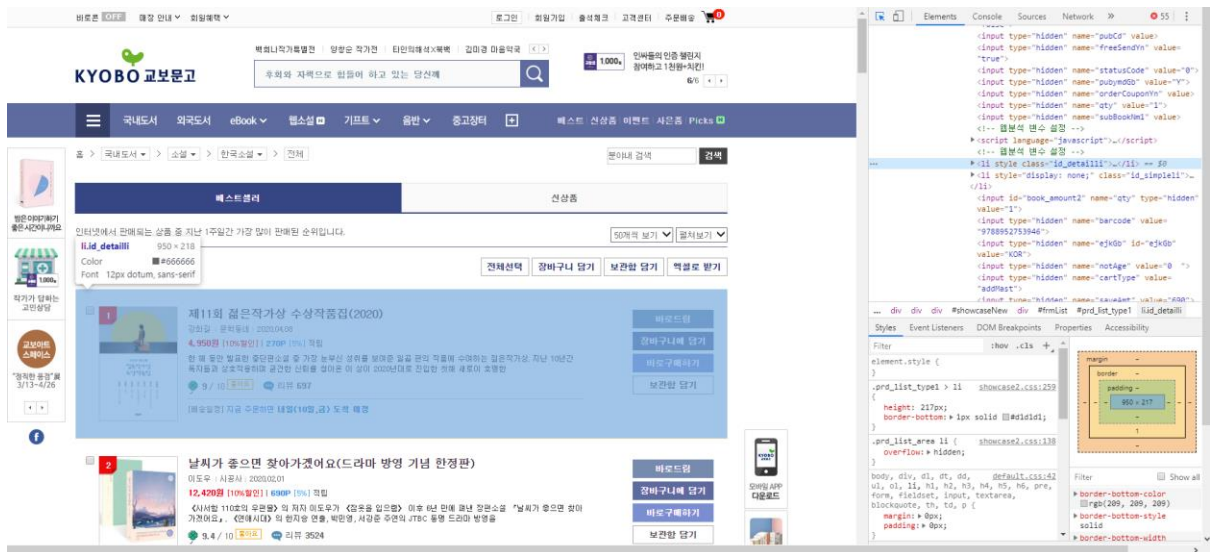
예시를 보여드리자면

http://www.kyobobook.co.kr/categoryRenewal/categoryMain.laf?pageNumber=1&perPage=50&mainGlb=KOR&linkClass=0101&ejkGb=&sortColumn=near_date

로 들어가시면

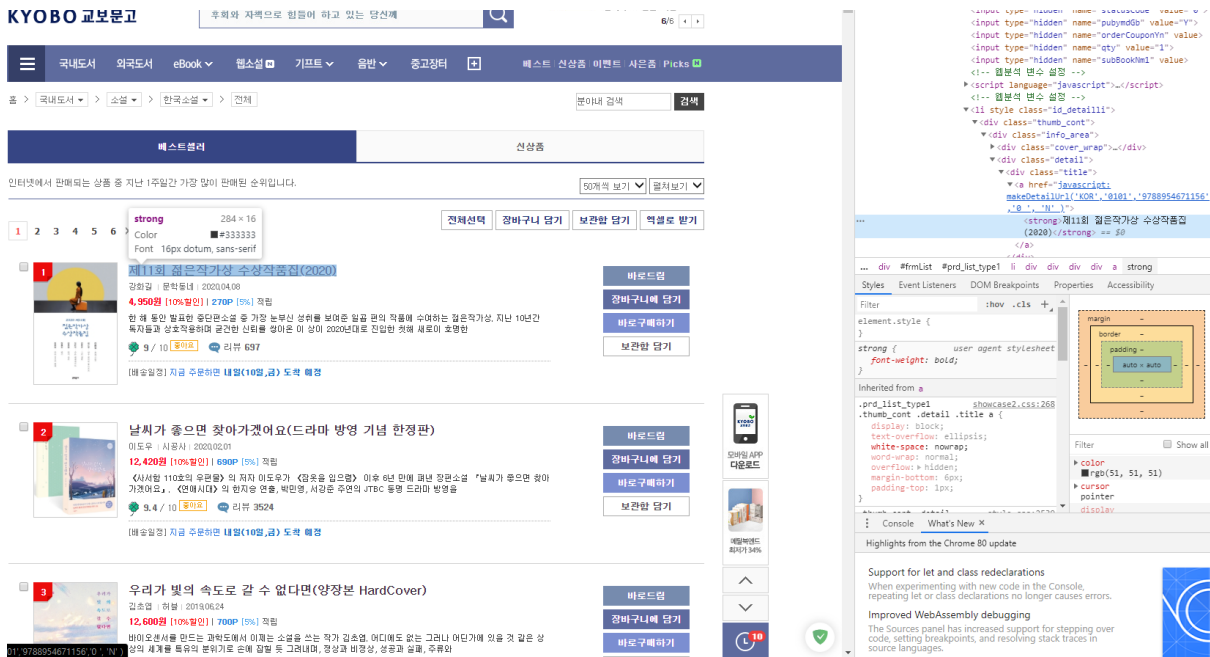


다음과 같은 창이 뜹니다. 여기서 우리가 복사하고 싶은 정보들의 위치를 xpath로 찾는건데 이를 위해 ctrl + shift + c를 누르시면

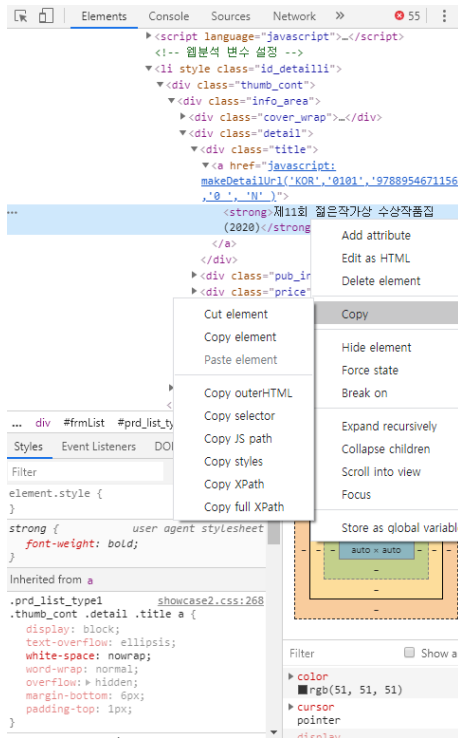


크롬창이 위처럼 변하고 마우스 커서가 있는 곳에 있는 웹 요소의 정보를 알 수 있게 됩니다.

그럼 첫 예시로 제일 위에 있는 책의 제목을 눌러보면



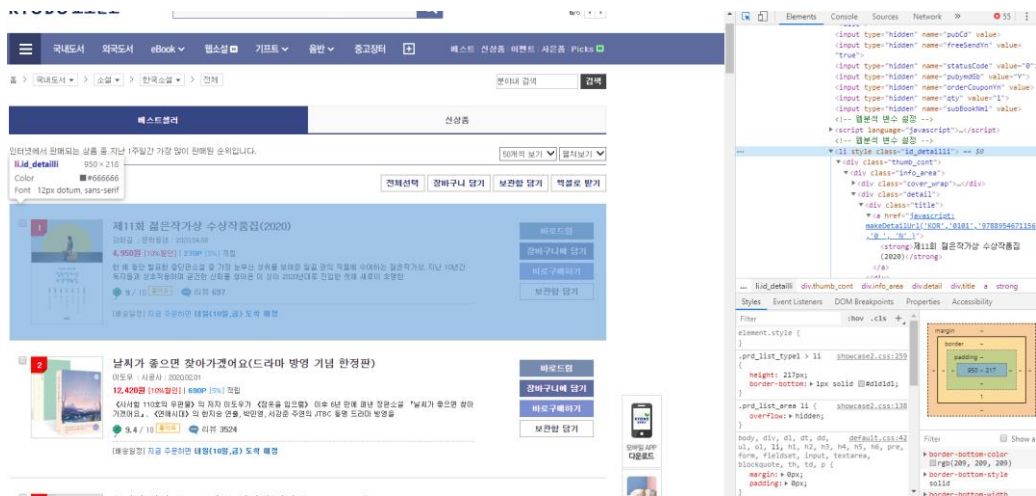
위의 사진처럼 되는데 좌측에 나오는 실제 브라우저 화면에 해당 제목의 영역이 뜨고, 우측에는 html에서 어느 부분이 해당 부분인지가 나옵니다. 그럼 하이라이트 된 부분을 오른쪽 클릭하면



이런 창이 뜨는데 copy를 누르시면 하단에서 두번째에 Copy XPath가 있습니다. 이를 통해 우리가 원하는 요소의 xpath를 알 수 있고, 위에서 나온 find_elements_by_xpath함수의 매개변수로 xPath를 넣어 우리가 원하는 요소의 데이터를 가져올 수 있습니다.

위의 설명에서는 혼동을 줄이기 위해 find_elements_by_xpath 함수로 설명을 했는데, find_element_s_by_xpath 함수는 보시다시피 elementS, 즉, 복수로 되어있어서 한가지 요소를 찾아내는 함수가 아니고 해당 위치에 있는 요소들 집합을 찾아내는 함수이고, ex) ul밑의 li들의 집합. 한가지 요소만 보려면 **find_element_by_xpath**를 사용하셔야 합니다. **s가 있고 없고의 차이**입니다.

다시 27번줄 코드 설명으로 돌아가서, 27번줄 코드는 해당 페이지에 몇 개의 책이 있는지를 알아내는 코드입니다.



ctrl + shift + c를 눌러서 제일 첫 책의 li태그부분의 xpath를 복사하면

//*[@id="prd_list_type1"]/li[1] 이 나오고 그 다음 책의 li태그 부분을 복사하면

//*[@id="prd_list_type1"]/li[3] 이 나오는데 여기서 패턴을 찾을 수 있습니다.

n번째 책은 1+2(n-1)번째 li에 있습니다. 이 처리는 후에 할 것이고 지금은 li의 개수를 find_element_by_xpath에 //*[@id="prd_list_type1"]/li를 넣은 뒤 len 함수로 구하여 end 변수에 대입합니다.

30번줄은 해당 페이지에서 복사한 데이터들을 저장하는 sql의 앞 부분을 미리 만들어 둔 것입니다. 제목, 썸네일, 가격, 저자, 출판사, 1차카테고리, 2차카테고리, 3차카테고리를 넣을 것입니다.

32~34번줄은 카테고리들을 미리 설정해 두는 것인데 1차는 국내도서로 고정되어있고, 2차와 3차가 바뀌기에 매번 url을 이동할 때마다 크롤링을 해오게 합니다.



xpath를 복사해서 find_element_by_xpath를 사용해서 해당 요소를 복사해오는데 가장 마지막

에 **.text**부분을 꼭 넣어줘야 실제 데이터를 텍스트로 가져옵니다.

40번줄부터가 실질적인 크롤링 부분인데 27번째줄에서 구한 end(페이지에 몇 개의 책이 있는지 구하기 위한 변수)를 이용해 **1부터 end까지 2씩 증가**시키며 반복문을 실행합니다. 이를 통해 **각 li, 즉, 각 책 정보들에 접근**할 수 있습니다.

41번줄은 sql문에 추가될 데이터를 위해 괄호를 열어서 추가합니다.

42~52번은 **제목, 작가, 가격, 출판사, 썸네일**을 크롤링하는 코드인데 각 책들이 **li번호외에는 구조가 똑같기에 li번호만 바꾸며 데이터를 복사**하는 코드입니다.

46번줄의 price[-1]은 가격에 ~~원이 들어가 있어서 원을 빼기 위한 코드입니다.

50번줄의 temp는 썸네일을 추출하는 코드인데 .get_attribute를 통해 **해당 요소의 특정 속성 값**을 알 수 있습니다. **ex)** img 태그의 src나 a 태그의 link 속성.

52번줄은 50번에서 가져온 썸네일 주소가 굉장히 작은 이미지여서 확대 이미지로 변경하기 위한 작업입니다.

53번줄은 그동안 크롤링한 책의정보를 DB에 집어넣기 위해 sql문형태로 추가시키는 작업입니다.

각 부분에 .replace가 있는 이유는 책의 정보에 '이나 "이 들어있으면 sql문이 망가져서 이를 방지하기 위해 미리 변경해서 sql문을 작성하기위해서 입니다.

57번줄은 37번줄에서 정해 놓은 500에 218을 추가하는 코드인데 대략 500px이 첫 책이 시작하는 위치이고 각 책들이 218px씩 차지해서 현재 크롤링하는 위치를 보여주기 위해서 **브라우저의 스크롤을 이동시키는 작업**을 위해 위치를 잡아주는 코드입니다.

58번줄에서는 window.scrollTo 자바스크립트를 동작시키는 execute_script함수를 사용해서 하나의 도서정보 복사를 완료할 때마다 한 칸씩 브라우저도 내려가게 하는 코드입니다.

60번줄 코드는 for문 밖이기에 **한 페이지의 모든 책의 정보 복사를 완료했을 때 작동**되는 코드이며 이는 한 카테고리를 다 복사한 것이기에 이를 표시하기위한 print문입니다.

61번줄 코드는 54번줄에서 볼 수 있듯이 **sql문이 실패로 끝나서 실패를 빼고 sql문을 실행**해주는 위한 코드입니다.

62번은 DB변동사항 **commit**을 위한 코드입니다.

이걸로 국내도서를 크롤링하는 domestic함수의 설명이 끝났습니다. 해외도서를 크롤링하는 foreign함수도 거의 동일해서 foreign함수는 domestic과 다른 부분만 설명하겠습니다.

일단 다시 실제 실행되는 코드부분으로 돌아와서

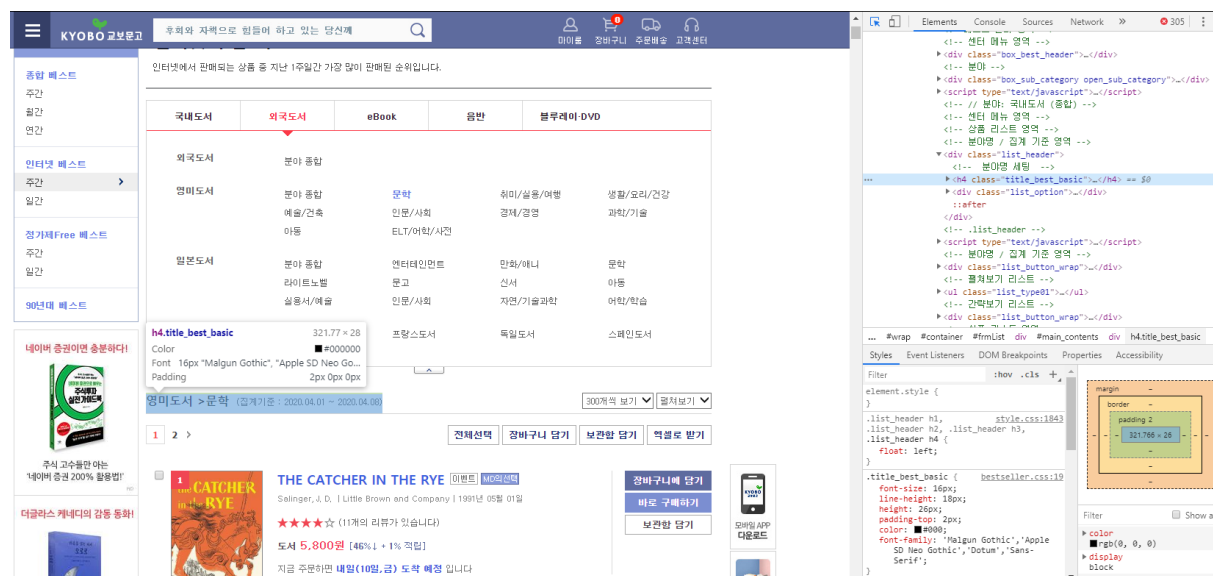
150~151번줄 코드는 해외 카테고리를 위한 리스트입니다. 국내도서와 다르게 두 자리 수입니다.

154,155번 줄을 보시면 해외도서 url은 **mallGb**와 **linkClass**의 두개의 값을 바꿔야해서 url을 두부분으로 나눠서 후에 각각의 값을 추가해 사용할 겁니다.

158번줄로 foreign함수를 실행합니다.

70번줄은 15번까지는 영미도서, 그 이후는 일본도서로 되어있어서 2차 카테고리 구분을 위해 작성한 if문입니다.

91번줄은 2차 카테고리 3차 카테고리를 위해 임시로 크롤링한 코드입니다.



보시는 것처럼 h4 태그 하나에 2차 카테고리, 3차 카테고리, 필요 없는 정보가 다같이 몰려있습니다. 이런 경우에는 패턴을 찾아야 하는데 일단 2차 카테고리 사용 할 영미도서, 일본도서는 모두 4글자이기에 앞의 4자리만 떼서 2차 카테고리 사용하고, 3차 카테고리는 앞의 4자리를 제외하고, 뒤의 날짜가 시작하는 괄호의 시작 전까지로 하면 되기에 temp[6:(temp.find('(')-1)]를 사용해서 분리합니다.

120번 줄은 해외도서는 각 li들의 크기가 일정치가 않아서 대략적인 평균값인 291로 하고 크롤링 할 때마다 스크롤하게 하기위한 코드입니다.

이렇게 foreign함수 또한 끝나고 마지막으로 160번줄 코드로 DB연결을 끊고 프로그램이 종료됩니다.