<1쇄 정오표>

- 초판 1쇄(2024년 6월 20일) 출간 후 실습에 사용된 웹 사이트 및 API의 변경이 있어 정오표를 아래와 같이 제공합니다.
- 웹 크롤링의 특성상 변경된 웹 사이트 환경에 따라 내용 수정이 불가피하게 이루어진 점 양해 부탁드립니다.
- 아래 내용은 2쇄에 모두 반영되었습니다.

★ 1쇄 정오표(수정 위치와 범위) ★

- **1.1.2 파이참 설치하기**: 29~39쪽
- **6.2.2 웹 페이지의 동작 원리**: 176~190쪽
- **7.5 이미지 수집하기**: 228~234쪽
- **7.6 수집한 정보 엑셀에 저장하기**: 235~244쪽
- **9.4.2 문서 요약 API 키 발급받기**: 326~337쪽
- **10.1.2 [좋아요] 자동 누르기**: 347~355쪽
- **10.3.2 라인 API 사용하기**: 378~386쪽
- **11.1.1 네이버 메일의 SMTP 설정하기**: 394~402쪽

1.1.2 파이참 설치하기: 29~39쪽

* 파이참 설치 방법이 바뀌었습니다. 아래 내용을 참고해 설치하세요.

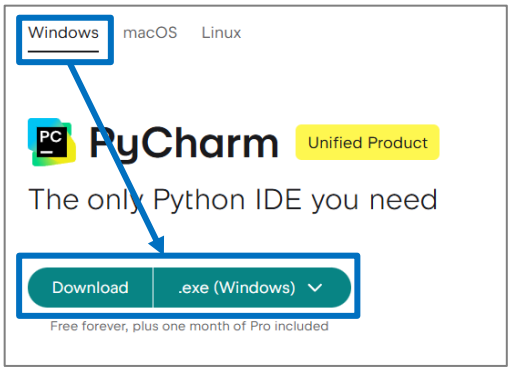
1.1.2 파이참 설치하기

파이참 공식 다운로드 사이트(**https://www.jetbrains.com/pycharm/download**)에 접속합니다.

- 윈도우 사용자는 [Windows] 탭을 선택하고 [Download] 버튼을 클릭합니다.
- 맥OS 사용자는 [macOS] 탭을 선택한 후, 맥OS(m1, m2) 사용자는 **.dmg (Apple Silicon)**을, 맥OS(인텔) 사용자는 **.dmg (Intel)**을 선택하고 [Download] 버튼을 클릭합니다.

그림 1-9 파이참 설치 파일 다운로드

(a) 윈도우



(b) 맥OS

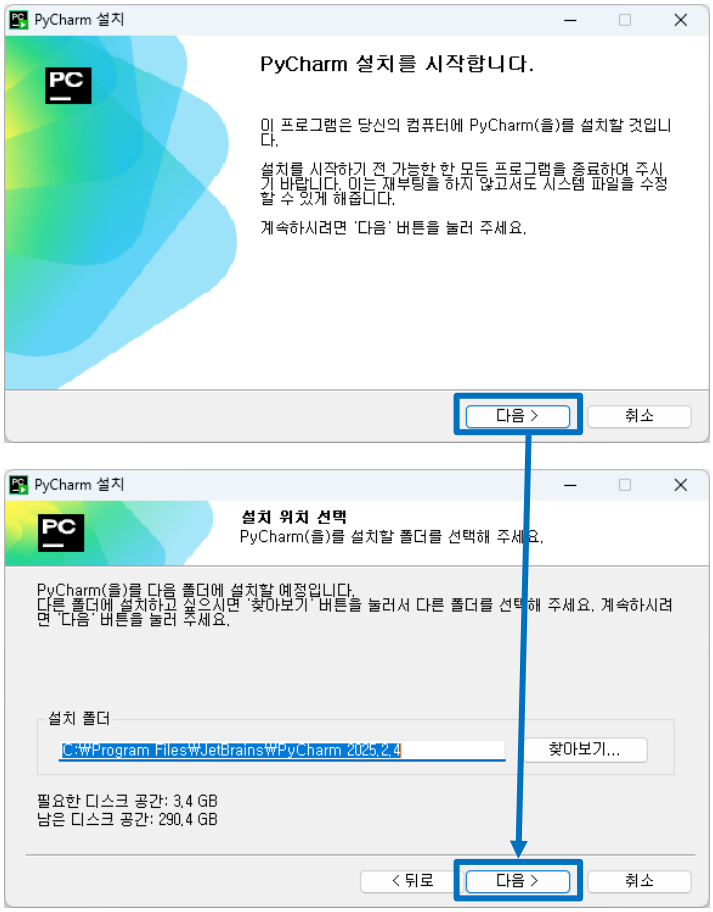
[TIP] 파이참은 예전에 유료 버전인 프로페셔널(Professional)과 무료 버전인 커뮤니티 에디션(Community Edition)의 두 제품이 있었으나, 지금은 하나로 통합됐습니다. 이는 30일간 프로페셔널 기능을 무료로 사용할 수 있고, 이후에는 구독을 해야 하는 구조입니다. 다만, 무료 기능(구 커뮤니티 에디션 수준)은 30일이 지나도 사용할 수 있으며, 이 책의 실습을 진행하는 데는 무료 기능만으로도 충분합니다.

잠시 설치 파일이 다운로드됩니다. 컴퓨터에 파이참을 설치하는 방법은 윈도우와 맥OS로 구분해 설명하겠습니다.

윈도우에 파이참 설치하기

1 내려받은 설치 파일(**pycharm-2025.2.4.exe**)을 더블클릭해 실행합니다. 설치 시작 화면에서 [다음] 버튼을 클릭하고, 설치 위치 선택 화면에서는 기본값을 그대로 둔 채 [다음] 버튼을 클릭합니다.

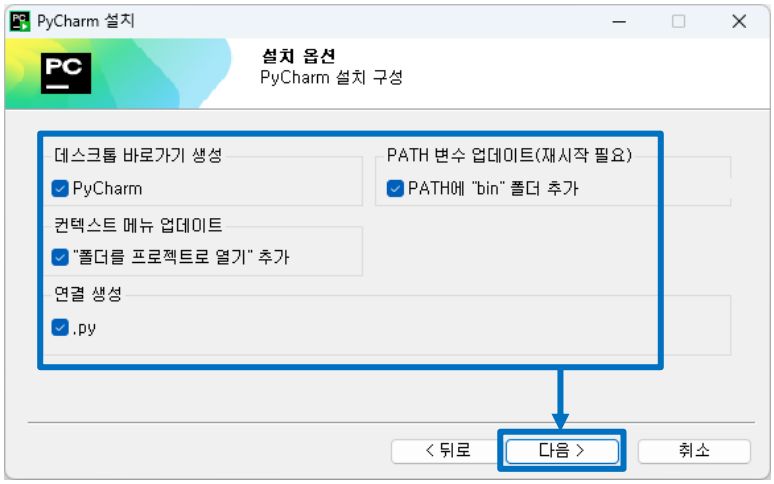
그림 1-10 파이참 설치 시작



[TIP] 파이참의 설치 파일명은 파이참 버전을 나타냅니다. 다운로드 시기에 따라 책과 버전이 다를 수 있으나 실습하는 데에는 문제가 없습니다.

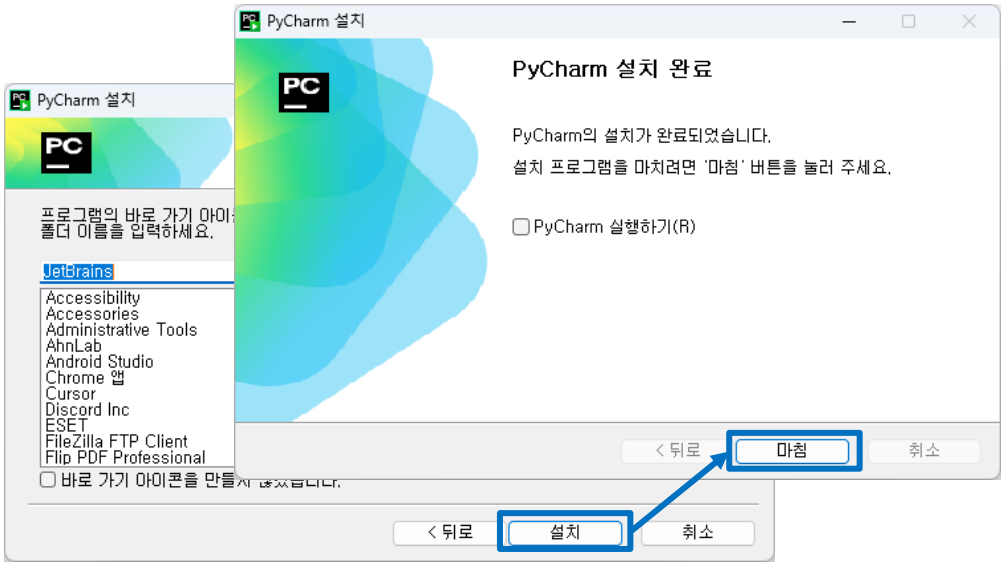
2 설치 옵션 화면이 나타나면 모든 옵션에 체크하고 [다음] 버튼을 클릭합니다.

그림 1-11 파이참 설치 옵션 설정



3 시작 메뉴 폴더 선택 화면에서 [설치] 버튼을 클릭해 설치를 시작합니다. 설치 완료 화면이 나타나면 [마침] 버튼을 클릭합니다.

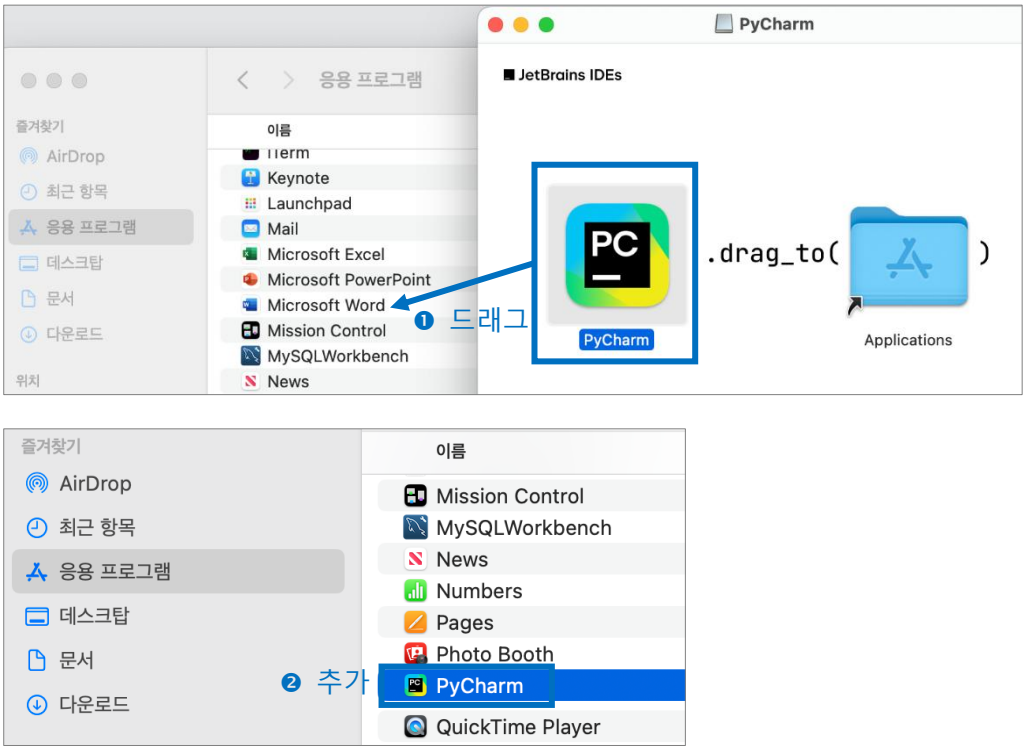
그림 1-12 파이참 설치 완료



맥OS에 파이참 설치하기

내려받은 설치 파일(**pycharm-2025.2.4-aarch64.dmg**)을 더블클릭합니다. PyCharm . drag_to Applications 창이 나타나면 파인더(Finder)를 실행해 [응용 프로그램] 폴더에 들어가 PyCharm 아이콘을 [응용 프로그램] 폴더로 드래그합니다(❶). 그러면 PyCharm 프로그램이 응용 프로그램에 추가되며(❷) 설치가 끝납니다.

그림 1-13 응용 프로그램에 PyCharm 추가

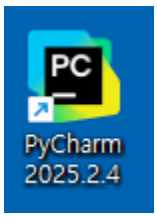


1.1.3 파이참 프로젝트 생성하기

1 파이참을 설치하면 윈도우의 경우 바탕화면에서 파이참 아이콘을 확인할 수 있고, 맥OS의 경우 [응용 프로그램] 폴더에서 파이참 프로그램을 확인할 수 있습니다. 파이참을 더블클릭해 실행합니다.

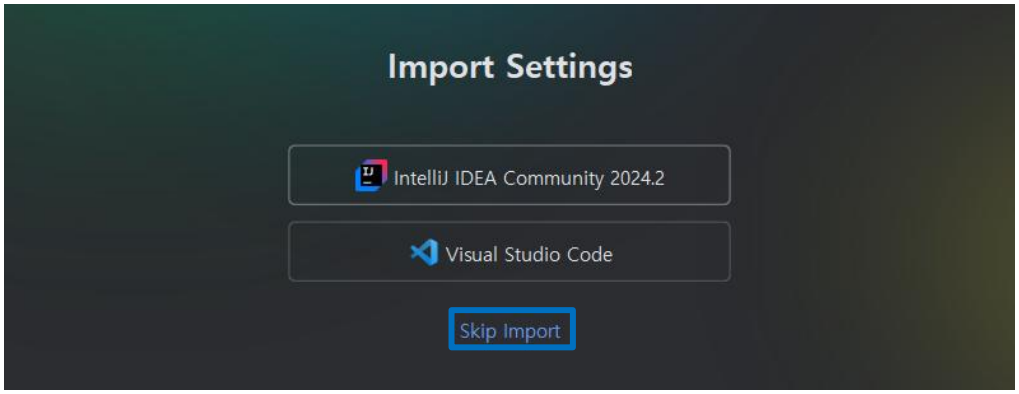
그림 1-14 파이참 프로그램 실행

(a) 윈도우 (b) 맥OS



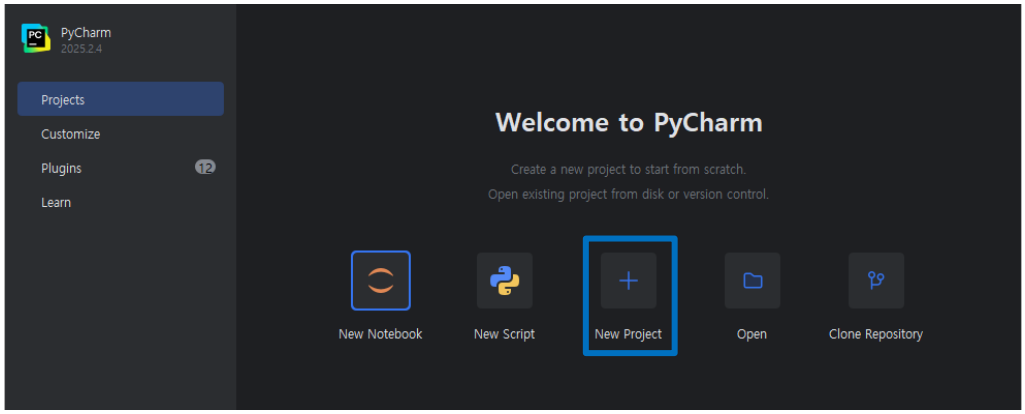
2 언어 및 지역 화면이 뜨면 언어는 **English**를 선택하고, 지역은 설정하지 않은 채 [Next] 버튼을 클릭합니다. 사용자 동의 화면에서 **I confirm that I have read and accept the terms of this User Agreement**에 체크하고 [Continue] 버튼을 클릭합니다. 추가로 기존 설정을 가져올지 묻는 창이 나타나면 **Skip Import**를 클릭합니다.

그림 1-15 파이참 설정 가져오기 선택



3 파이참 시작 화면에서 [New Project]를 클릭합니다. 여기서 프로젝트는 폴더를 가리킵니다. 프로젝트라는 말이 거창해 보이지만 사실은 하나의 프로그램을 만들기 위해 여러 개의 파일을 폴더에 저장한 것입니다. 지금부터 이 폴더를 **프로젝트 폴더**라고 부르겠습니다.

그림 1-16 새 프로젝트 생성

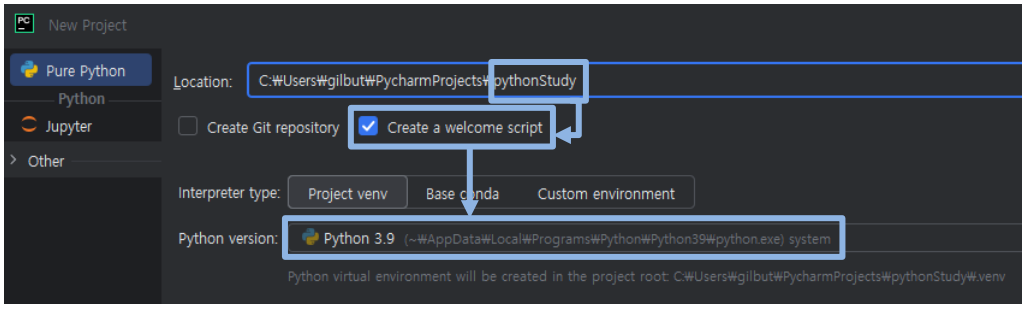


4 [New Project] 창이 나타나면 Location의 프로젝트명을 PythonProject1에서 **pythonStudy**로 수정합니다. 또한 **Create a welcome script**에 체크하고, Python version이 **3.9**로 설정돼 있는지 확인합니다. 경로가 다르면 다음 경로를 참고해 직접 수정하세요. 확인을 모두 마치면 [Create] 버튼을 클릭합니다.

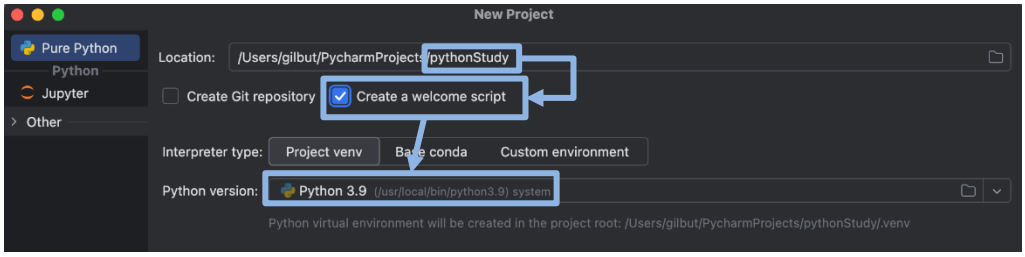
- **윈도우**: C:\Users\사용자명\AppData\Local\Programs\Python\Python39\python.exe
- **맥OS**: /usr/local/bin/python3.9

그림 1-17 프로젝트명과 버전 설정

(a) 윈도우



(b) 맥OS



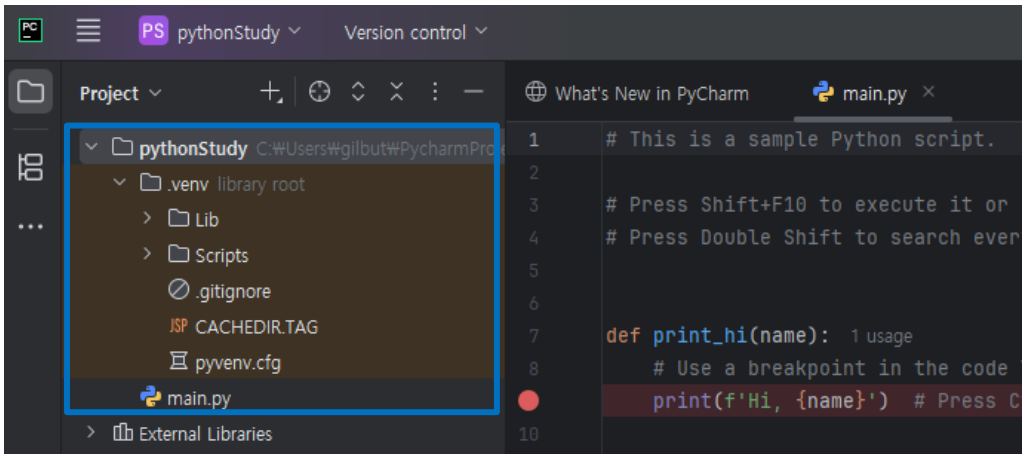
[NOTE] 프로젝트명과 저장 경로 설정 시 주의점

앞에서 프로젝트명을 pythonStudy로 고쳤습니다. 이처럼 프로젝트명은 영어로만 구성해야 하고 띄어쓰기를 포함할 수 없습니다. 띄어쓰기를 하고 싶다면 _(언더바) 기호를 사용하세요.

프로젝트의 저장 경로(Location)는 수정하지 않고 그대로 사용했습니다. 이 경로의 Users 다음에 나오는 사용자명(**그림 1-17**에서는 gilbut)은 윈도우 사용자명이 그대로 반영된 것으로, 이 이름이 한글일 경우 **11.2절 윈도우 작업 스케줄러로 자동 실행하기** 실습을 할 때 제대로 동작하지 않습니다. 따라서 윈도우 사용자명이 한글이라면 영문으로 바꿔야 합니다. 윈도우 사용자명을 변경하는 방법은 익스트림매뉴얼 사이트(**https://extrememanual.net/41523**)를 참고하세요.

5 프로젝트가 성공적으로 만들어지면 다음과 같이 [pythonStudy] 프로젝트 폴더 아래에 **main.py** 파일이 생성된 것을 확인할 수 있습니다.

그림 1-18 프로젝트 생성 확인



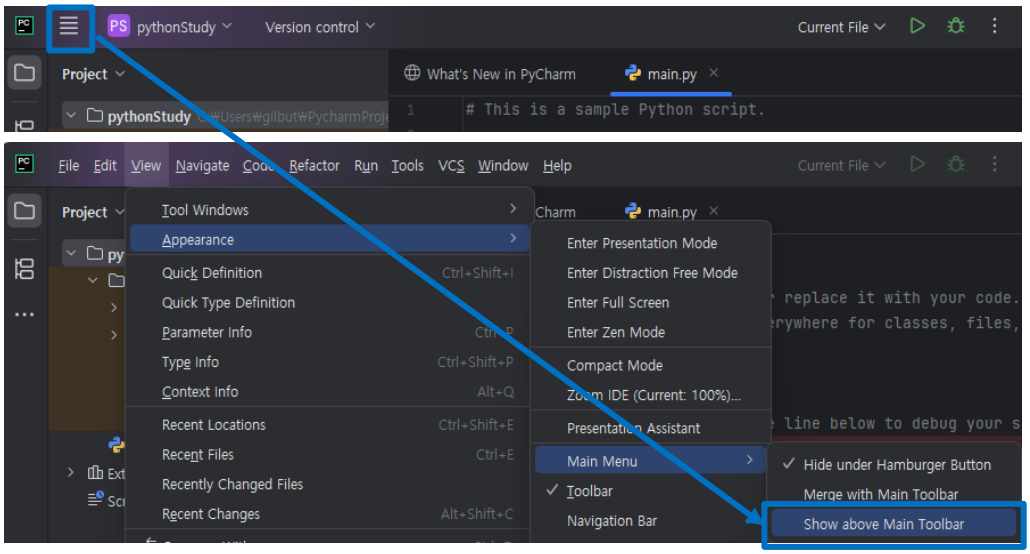
[NOTE] 파이참을 사용하는 이유

파이참과 같이 코딩을 편하게 할 수 있도록 도와주는 도구를 **IDE**(Integrated Development Environment, 통합 개발 환경)라고 합니다. 대표적인 파이썬 IDE로는 파이참, 주피터 노트북(Jupyter Notebook), 스파이더(Spyder), VSCode(Visual Studio Code) 등이 있습니다. 이 중에서 파이참은 UI가 직관적이고, 유용한 단축키를 제공하며, 외장 모듈을 설치하기 쉬워 많은 사람이 사용합니다. 파이참의 다양한 단축키와 유용한 기능은 앞으로 실습하면서 살펴보겠습니다.

1.1.4 파이참 환경 설정하기

1 파이참을 본격적으로 사용하기 전에 몇 가지 환경 설정을 하겠습니다. 윈도우 사용자는 화면 왼쪽 상단의 햄버거 아이콘을 클릭한 후 메뉴에서 **View → Appearance → Main Menu → Show above Main Toolbar**를 선택해 메인 메뉴를 표시합니다. 맥OS 사용자는 메인 메뉴를 표시할 수 없으니 이 부분을 건너뜁니다.

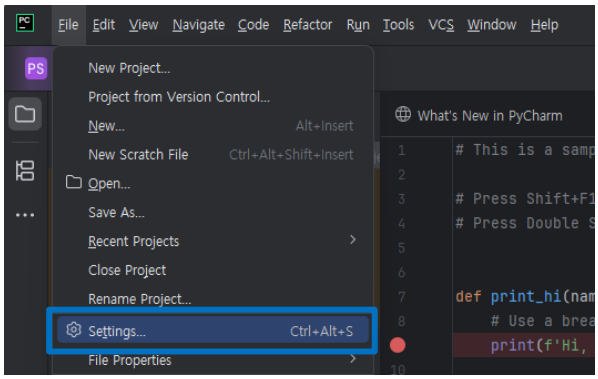
그림 1-19 메인 메뉴 표시



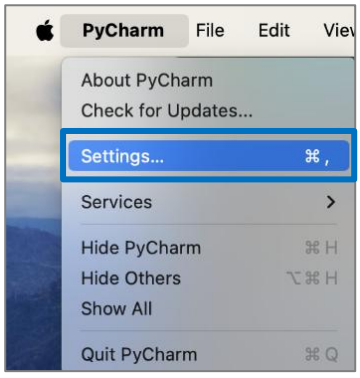
2 파이참의 환경 설정을 하기 위해 Settings 창을 띄웁니다. 윈도우 사용자는 **File → Settings** 메뉴를 선택하고, 맥OS 사용자는 화면 왼쪽 상단에서 애플 로고 옆의 **PyCharm → Settings** 메뉴를 선택합니다.

그림 1-20 Settings 창 띄우기

(a) 윈도우

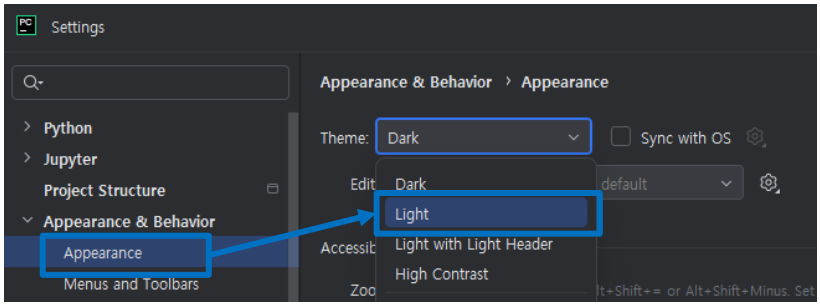


(b) 맥OS



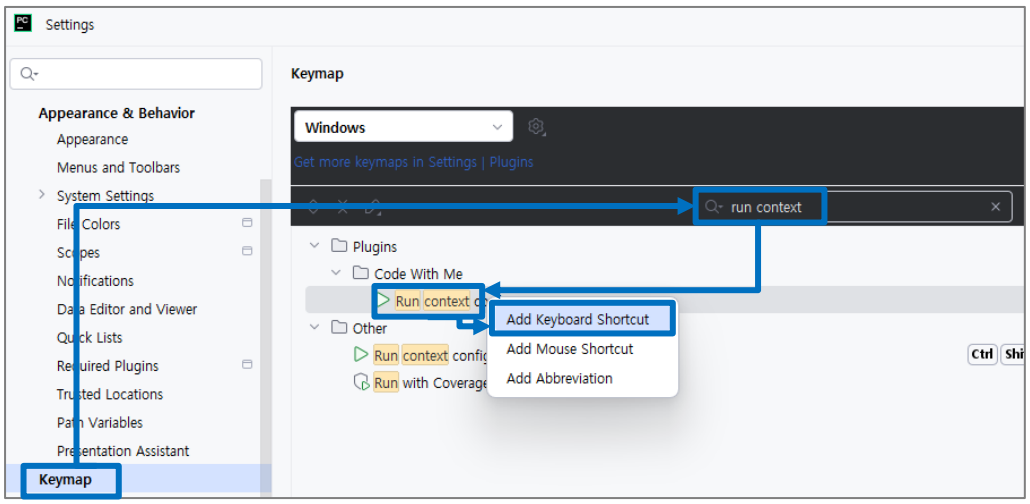
3 Settings 창에서 화면 테마, 실행 단축키, 글자 크기를 변경합니다. 먼저 어두운 화면 테마를 밝은 테마로 바꾸기 위해 왼쪽 **목록에서 Appearance & Behavior > Appearance**를 선택하고 오른쪽의 Theme을 **Light**로 변경합니다. 그러면 화면이 밝은색으로 바뀝니다.

그림 1-21 화면 테마 변경



4 파이참에서 작성한 코드를 실행할 때는 메뉴 또는 단축키를 이용합니다. 그런데 실행에 사용하는 단축키의 경우 기본적으로 복잡하게 설정돼 있습니다. 이를 간단한 키로 바꾸기 위해 왼쪽 목록에서 **Keymap**을 선택하고 오른쪽 검색창에 'run context'를 입력합니다. 아래에 Run context configuration 항목이 뜨면 클릭하고 마우스 오른쪽 버튼을 눌러 [Add Keyboard Shortcut]을 선택합니다.

그림 1-22 실행 단축키 설정 1

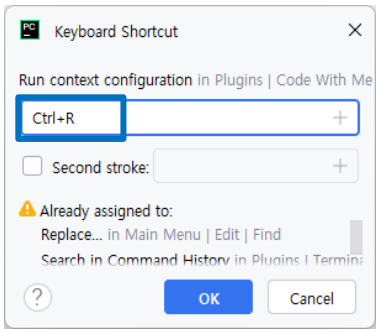


[TIP] 만약 Run context configuration 항목이 여러 개 뜨면 단축키가 이미 할당된 항목이 딱 하나만 있을 것입니다. 그 항목에서 마우스 오른쪽 버튼을 눌러 [Add Keyboard Shortcut]을 선택합니다.

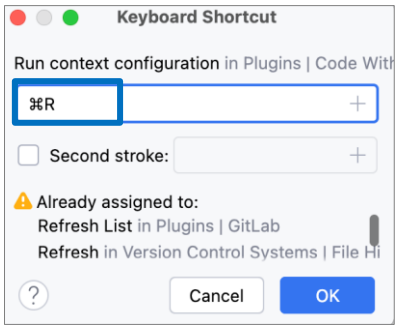
5 실행 단축키는 Run의 앞 글자를 따서 윈도우는 [Ctrl]+[R], 맥OS는 [command]+[R]로 설정하겠습니다. 입력란에서 윈도우 사용자는 [Ctrl] 키를, 맥OS 사용자는 [command] 키를 누른 채 [R] 키를 눌러 단축키를 입력하고 [OK] 버튼을 클릭합니다. 이미 다른 용도로 사용하는 단축키라는 경고창이 뜨면 [Remove] 버튼을 클릭합니다.

그림 1-23 실행 단축키 설정 2

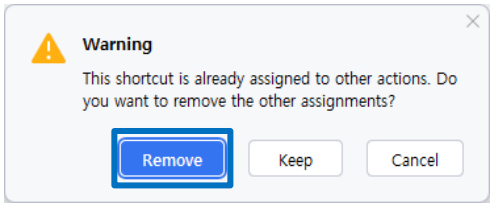
(a) 윈도우



(b) 맥OS

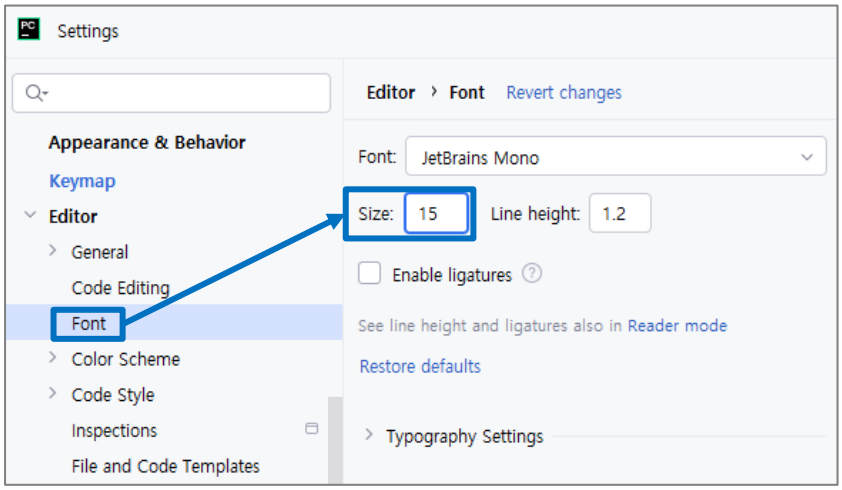


(c) 경고창 확인



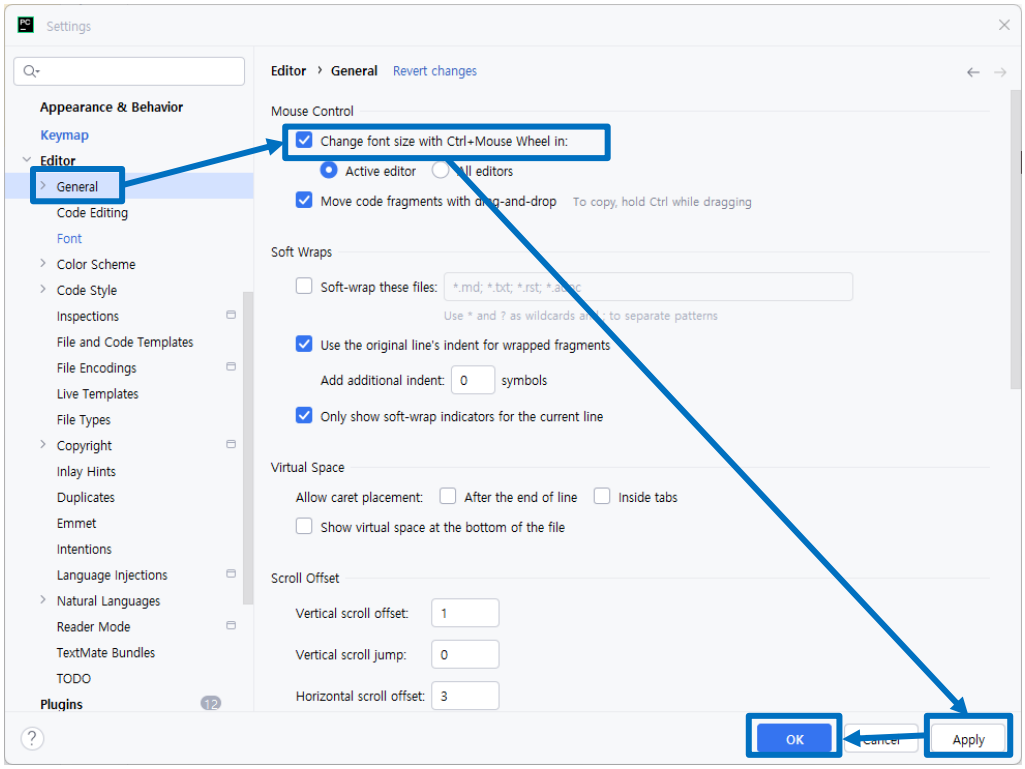
6 파이참을 처음 설치하면 편집창의 글자 크기가 작습니다. 왼쪽 목록에서 **Editor > Font**를 선택하고 오른쪽의 Size를 **15**로 수정해 글자 크기를 키웁니다.

그림 1-24 글자 크기 변경



7 왼쪽 목록에서 **Editor > General**을 선택하고 **Change font size with Ctrl+Mouse Wheel in:**에 체크합니다. 이렇게 설정하면 파이참 화면에서 [Ctrl]+마우스 휠(맥OS는 [command]+마우스 휠)로 글자 크기를 조절할 수 있습니다. 모든 설정이 끝나면 [Apply], [OK] 버튼을 클릭합니다.

그림 1-25 마우스 휠로 글자 크기 조절 설정



6.2.2 웹 페이지의 동작 원리: 176~190쪽

* CGV 웹 사이트가 개편됨에 따라 실습 사이트를 무비차트 사이트로 변경합니다.

6.2.2 웹 페이지의 동작 원리

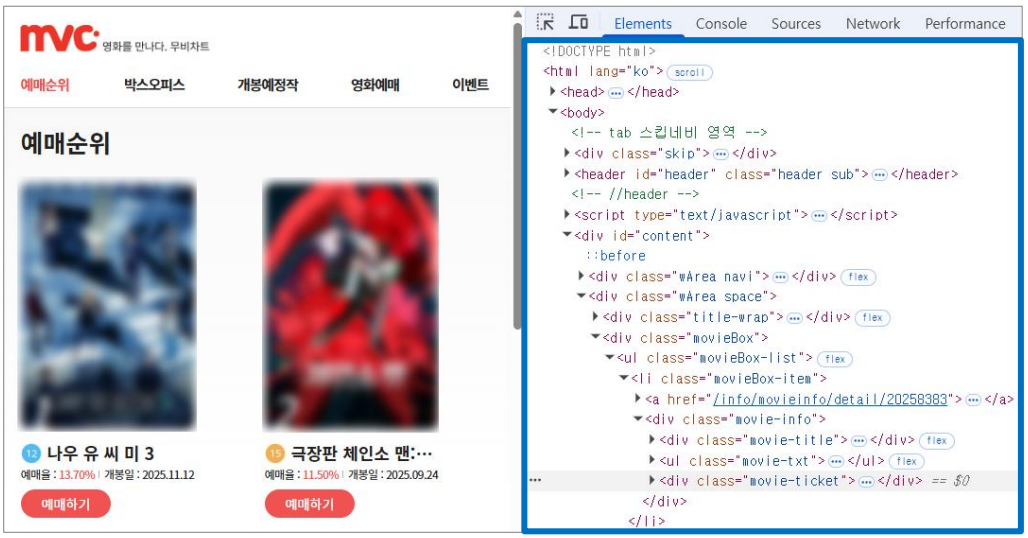
본격적으로 크롤링 프로그램을 만들기에 앞서 웹 페이지가 어떤 원리로 동작하는지 알아봅시다. 크롬 브라우저의 주소창에 **www.moviechart.co.kr**을 입력해 무비 차트 웹 사이트에 접속합니다. 상단에 있는 [예매순위]를 클릭한 후 페이지가 바뀌면 화면 빈 곳에서 마우스 오른쪽 버튼을 눌러 [검사]를 선택합니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-11 무비 차트 페이지에서 [검사] 선택**

그러면 웹 브라우저 오른쪽에 **개발자 도구**(DevTools)라는 화면이 뜹니다. 화면의 [Elements] 탭에 알 수 없는 코드가 보이는데, 이것이 바로 **HTML**(HyperText Markup Language, 하이퍼텍스트 마크업 언어) 코드입니다. HTML은 웹 페이지를 만들 때 사용하는 언어로, 파이썬과는 완전히 다른 언어이며, 지금 보고 있는 코드는 무비 차트 웹 개발자가 작성한 HTML 코드입니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-12 개발자 도구에서 본 무비 차트의 HTML 코**

드

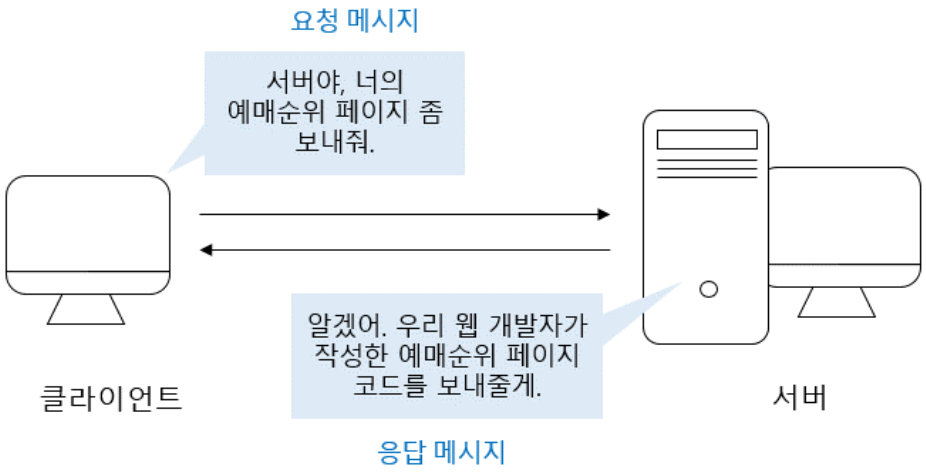


그렇다면 무비 차트 웹 사이트는 어떻게 동작하기에 [예매순위] 메뉴를 클릭하자마자 예매순위 페이지를 보여주는 것일까요? 이는 **클라이언트**(client)와 **서버**(server)의 동작 원리를 통해 이해할 수 있습니다.

- **클라이언트**: 사용자가 서버에 접속할 때 사용하는 컴퓨터 혹은 프로그램을 말합니다.
- **서버**: 클라이언트의 요청을 받아 응답하는 컴퓨터 혹은 프로그램을 말합니다. 무비 차트 서버는 수많은 클라이언트로부터 요청을 받아 처리한 뒤 그 결과를 반환합니다.

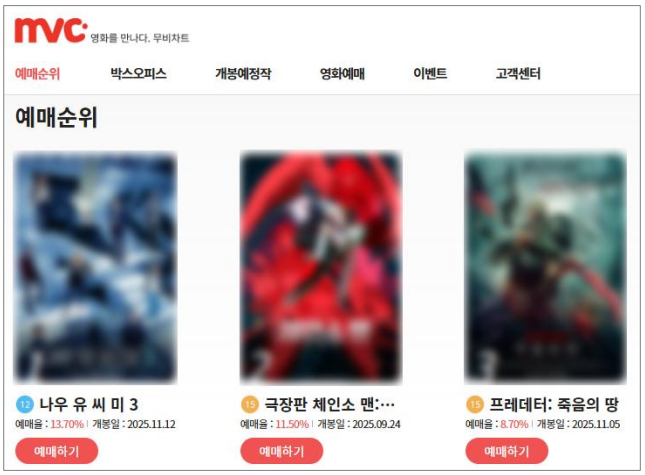
서버와 클라이언트는 요청 메시지와 응답 메시지를 주고받으며 통신합니다. 클라이언트에서 [예매순위] 메뉴를 클릭하면 예매순위 페이지를 보여달라는 요청 메시지가 서버에 전송됩니다. 요청 메시지를 받은 서버는 이를 해석한 후 예매순위 페이지의 HTML 코드를 응답 메시지에 실어 클라이언트에 보냅니다. 요청 메시지와 응답 메시지는 매우 빠른 속도로 전송되기 때문에 이 과정이 순식간에 일어납니다.

그림 6-13 클라이언트와 서버가 주고받는 메시지



그런데 신기한 점이 있습니다. 서버는 응답으로 예매순위 페이지의 HTML 코드를 보내는데 클라이언트의 화면에는 HTML 코드 대신 보기 좋게 시각화된 웹 페이지가 나타납니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-14 응답으로 받은 예매순위 페이지**



이는 크롬, 엣지, 사파리 등의 웹 브라우저가 서버로부터 전달받은 HTML 코드를 시각화해 보여주기 때문입니다. 이렇게 HTML 코드를 읽어 실제 웹 사이트의 모습으로 보여주는 것을 **렌더링**(rendering)이라고 합니다.

여기서 중요한 점은 클라이언트에서 서버로 요청 메시지를 보내면 서버는 HTML 코드를 응답 메시지로 보낸다는 것입니다. 여기까지 이해했다면 크롤링 원리의 절반을 마스터했다고 할 수 있습니다.

6.2.3 무비 차트 수집 프로그램 만들기

모든 웹 페이지는 HTML로 작성되고 클라이언트와 서버의 요청과 응답으로 동작합니다. 이제 본격적으로 무비 차트 수집 프로그램을 만들기 위해 먼저 [pythonStudy] 폴더에 **ch06-무비차트수집.py** 파일을 생성합니다. 그리고 서버로부터 HTML 코드 받아오기, 받아온 HTML 코드를 보기 좋게 정리하기, 1위 영화 제목 출력하기, 전체 영화 제목 출력하기 순으로 프로그램을 작성하겠습니다.

서버로부터 HTML 코드 받아오기

서버에 예매순위 페이지를 보여달라고 요청하고, 서버로부터 예매순위 페이지의 HTML 코드를 응답으로 받습니다.

❶ 실습에 필요한 `requests`, `bs4` 모듈을 불러옵니다. `bs4` 모듈의 경우 모듈 전체를 불러오지 않고 모듈 내 `BeautifulSoup`만 불러와도 되므로 `from ~ import` 문을 사용해 불러옵니다.

❷ `requests.get()` 명령으로 요청 메시지를 보내고 응답 메시지를 받아 `code` 변수에 저장합니다. `requests.get()`은 괄호 안의 서버 주소로 요청 메시지를 보내라는 뜻으로, 예매순위 페이지의 URL을 복사해 괄호 안에 붙여넣고 URL을 `" "` 또는 `' '`로 감쌉니다.

❸ 응답 메시지는 예매순위 페이지의 HTML 코드를 담고 있는 응답 본문, 응답에 필요한 부가 정보를 담고 있는 응답 헤더로 구성됩니다. 여기서 응답 본문만 추출하려면 `code` 뒤에 `.text`를 붙여 `code.text`라고 작성해야 합니다. `print()` 문으로 `code.text`를 출력합니다.

그림 6-15 예매순위 페이지의 URL 복사해 붙여넣기

<코드> ch06-무비차트수집.py

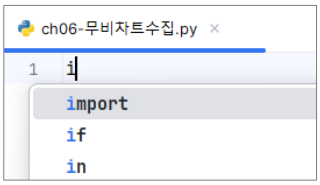
```
import requests ---------------------- ❶ 모듈 불러오기
from bs4 import BeautifulSoup
# 서버로부터 HTML 코드 받아오기      ┌ ❷ 예매순위 페이지 요청 및 응답받기
code = requests.get("https://www.moviechart.co.kr/rank/realtime/index/image")
print(code.text) --------------------- ❸ 응답 본문 출력
```

</코드>

[NOTE] 자동 완성 기능

혹시 앞의 짧은 코드를 입력할 때 오타가 나지 않았나요? 만약 오타가 있다면 몇 시간을 헤맬 수도 있으니 코드를 입력할 때는 자동 완성 기능을 이용하는 것이 좋습니다. 다음 그림처럼 `import`의 `i`만 입력해도 자동 완성 목록이 나타나며, 여기서 원하는 단어를 선택하면 오타 없이 입력할 수 있습니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-16 파이참의 자동 완성 기능**



코드를 실행하면 예매순위 페이지의 HTML 코드가 출력됩니다.

<실행결과>

```
<!doctype html>
<html lang="ko">
<head>
    <meta charset="utf-8">
(중략)
</footer>
<!-- //footer -->
</body>
</html>
```

</실행결과>

예매순위 페이지의 HTML 코드를 받아왔으니 이제 영화 제목을 수집해보겠습니다. 방법은 간단합니다. 예를 들어 '스타워즈'라는 영화 제목을 찾고 싶다면 컴퓨터에 예매순위 페이지의 HTML 코드를 주며 "이것을 다 뒤져서 '스타워즈'라는 글자를 찾아"라고 명령하면 됩니다. 그러면 컴퓨터는 HTML 코드를 한 글자 한 글자 다 뒤져서 '스타워즈'라는 영화 제목을 찾아냅니다. 달리 비유하자면 이는 교수님이 대학원생에게 두꺼운 전공 서적을 주며 "여기서 수학 공식을 찾아 정리하세요"라고 지시하는 것과 같습니다.

그런데 문제가 있습니다. 명령이 불친절할수록 컴퓨터가 해당 단어를 찾는 데 시간이 많이 걸립니다. 따라서 단어를 찾는 시간을 줄이기 위해 HTML 코드를 정리하겠습니다.

HTML 코드 정리하기

ch06-무비차트수집.py 파일에 다음 코드를 추가합니다.

❶ 코드 마지막 줄의 `print(code.text)` 문은 더 이상 필요 없으니 주석 처리합니다.

❷ HTML 코드를 정리하기 위해 파일의 맨 앞에서 불러온 `BeautifulSoup()` 명령을 사용합니다. 괄호 안의 첫 번째 자리에는 서버로부터 받은 HTML 코드(`code.text`)를 넣고, 두 번째 자리에는 `"html.parser"`를 넣습니다. `"html.parser"`는 HTML 코드를 정리하는 방법 중 하나입니다. 이렇게 정리한 HTML 코드를 `soup` 변수에 저장합니다.

❸ 결과를 확인하기 위해 `soup` 변수를 출력합니다.

<코드> ch06-무비차트수집.py

```
code = requests.get("https://www.moviechart.co.kr/rank/realtime/index/image")

# print(code.text) ------------------------------ ❶ 주석 처리

# HTML 코드 정리하기

soup = BeautifulSoup(code.text, "html.parser") --- ❷ HTML 코드 정리

print(soup) ----------------------------------------- ❸ 출력
```

</코드>

[TIP] HTML 코드를 정리하는 방법에는 "html.parser" 말고도 "xml", "lxml", "html5lib" 등이 있습니다. 각각은 HTML 코드를 정리하는 방식과 속도가 조금씩 다른데 어떤 것을 사용해도 상관없으며, 여기서는 가장 대중적인 "html.parser"를 사용했습니다.

코드를 실행하면 앞에서 확인했던 것과 마찬가지로 긴 HTML 코드가 출력됩니다. `BeautifulSoup()` 명령을 이용하면 코드가 정리된다고 했는데 도대체 어디가 정리된 것인지 파악하기 어렵습니다.

<실행결과>

```
<!doctype html>
<html lang="ko">
<head>
<meta charset="utf-8">
(중략)
</footer>
<!-- //footer -->
</body>
</html>
```

</실행결과>

크롤링은 교수님이 대학원생에게 전공 서적을 주면서 수학 공식을 찾아 정리하라고 지시하는 것과 같다고 했습니다. 그런데 교수님이 페이지를 알려주고, 목차를 정리해주고, 형광펜으로 표시도 해준다면 대학원생의 작업에 큰 도움이 될 것입니다.

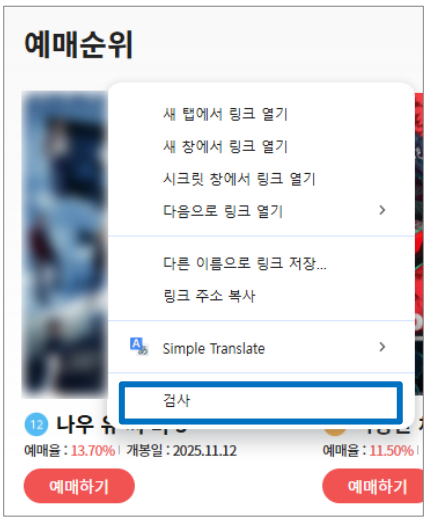
`BeautifulSoup()`은 바로 그러한 일을 합니다. 우리 눈에는 보이지 않지만, 컴퓨터가 좀 더 쉽게 데이터를 찾을 수 있도록 HTML 코드를 정리하는 것입니다.

> [NOTE] BeautifulSoup()라는 명칭의 유래
>
> 개발자들은 복잡하게 작성된 HTML 코드를 tag soup(태그 수프)라고 불렀습니다. beautiful soup(뷰티풀 수프)는 이렇게 복잡한 tag soup 코드를 예쁘게 만들어준다는 의미에서 붙은 명칭입니다.

1위 영화 제목 출력하기

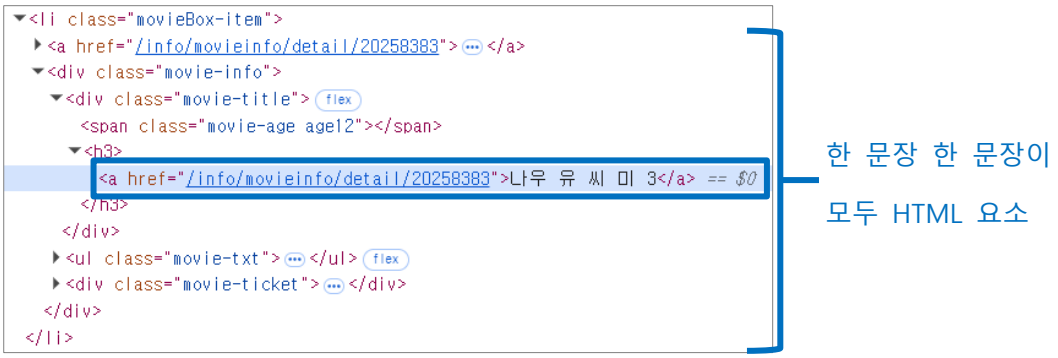
HTML 코드를 보기 좋게 정리했으니 1위 영화 제목을 찾아 출력하겠습니다. 1위 영화 제목을 출력하려면 1위 영화 제목의 **요소**(element)를 찾아야 합니다. 요소가 무엇인지 알기 위해 예매순위 페이지에서 1위 영화 제목에 마우스 커서를 놓고 오른쪽 버튼을 눌러 [검사] 메뉴를 선택합니다(영화 이미지가 아니라 영화 제목에서 마우스 오른쪽 버튼을 눌러야 합니다).

그림 6-17 1위 영화 제목에서 [검사] 메뉴 선택



개발자 도구가 뜨고 HTML 코드가 나타납니다. 여기서 요소란 HTML 코드를 구성하는 각각의 문장을 말합니다. 모든 HTML 코드는 요소로 구성되며, 요소는 계층 구조로 이뤄져 있습니다. 현재는 1위 영화 제목 요소가 선택됐습니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-18 HTML 요소 확인**



1위 영화 제목이 있는 요소는 a로 시작됩니다. 그러므로 "내가 원하는 요소는 맨 앞에 a가 쓰여 있으니 맨 앞에 a가 쓰여 있는 요소를 찾아"라고 컴퓨터에 명령하면 됩니다. 그러면 컴퓨터 입장에서는 무작정 찾는 것보다 일이 수월합니다. 대학원생 입장에서는 두꺼운 전공 서적을 주며 "여기서 수학 공식을 찾아 정리하세요"라고 지시하는 교수님보다 "필요한 수학 공식이 있는 페이지를 알려줄 테니 해당 페이지의 수학 공식을 정리하세요"라고 지시하는 교수님이 더 좋을 것입니다.

다음과 같이 1위 영화 제목 요소를 가져옵니다.

❶ HTML 코드 전체를 출력할 필요가 없기 때문에 print(soup) 문을 주석 처리합니다.

❷ soup.select_one()은 soup 변수에 저장된 코드 중에서 원하는 요소를 찾으라는 명령입니다. 괄호 안에 원하는 요소, 즉 "a"를 넣으면 컴퓨터는 soup 변수에 저장된 코드 중에서 맨 앞에 a가 쓰여 있는 요소를 찾습니다. 이렇게 찾은 요소를 title 변수에 저장합니다.

❸ 요소를 잘 찾았는지 확인하기 위해 title 변수를 출력합니다.

<코드> ch06-무비차트수집.py

```
# HTML 코드 정리하기
soup = BeautifulSoup(code.text, "html.parser")
# print(soup) ---------------------- ❶ 주석 처리
# 원하는 요소 가져오기
title = soup.select_one("a") ------ ❷ a 가져오기
print(title) ---------------------- ❸ 출력
```

</코드>

코드를 실행한 결과는 다음과 같습니다.

<실행결과>

```
<a href="#content">본문 바로가기</a>
```

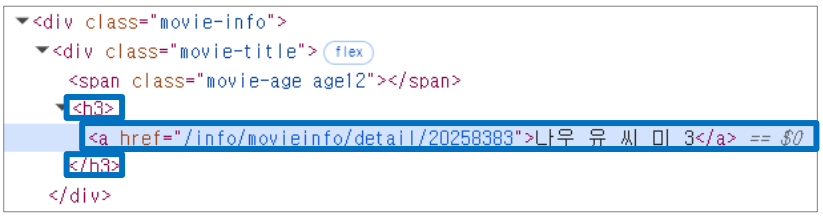
</실행결과>

컴퓨터가 a 요소를 하나 찾아냈습니다. 그런데 1위 영화 제목 요소가 아니라 엉뚱한 요소네요. 이 문제를 해결하기 전에 잠시 다음과 같은 경우를 생각해봅시다.

고려대에 다니는 A 군이 고연전에 가기 위해 동생에게 "고연전에 가야 하니까 내 방에 있는 티셔츠 좀 가져다 줘"라고 시켰습니다. A 군은 당연히 고려대의 상징인 빨간색 티셔츠를 가져올 것이라 예상했는데, 동생은 파란색 티셔츠를 가져왔습니다. 당황한 A 군은 동생에게 "내가 원하는 것은 칼라가 달려 있고, 고려대라고 쓰여 있고, 호랑이가 그려진 빨간색 티셔츠야"라고 말했습니다. 이 말을 들은 동생은 방에 가서 고려대 티셔츠를 가져왔습니다.

컴퓨터에 명령을 내릴 때도 이와 마찬가지입니다. 필요한 요소의 생김새를 자세히 묘사해줘야 그 요소를 정확히 가져옵니다. 개발자 도구에서 본 1위 영화 제목 요소를 다시 살펴봅시다.

그림 6-19 1위 영화 제목 요소 확인



a 요소가 h3라는 다른 요소 안에 들어 있습니다. 따라서 컴퓨터에 다음과 같이 명령하면 됩니다. "내가 원하는 요소는 a이긴 한데, h3 안에 들어 있는 a야. 이러한 요소를 찾아와." 이 명령을 코드로 옮겨보겠습니다.

select_one()에 넣은 인자를 "a"에서 "h3 > a"라고 수정합니다. 이렇게 기호와 글자를 이용해 자신이 원하는 요소를 컴퓨터에 알려주는 것을 **CSS 선택자**(Cascading Style Sheets Selector)라고 합니다. CSS 선택자에 대해서는 **6.3절 HTML 요소와 CSS 선택자**에서 자세히 알아보겠습니다. 코드를 수정하고 실행하면 원하는 결과가 출력됩니다(영화 제목은 실습하는 시점에 따라 다릅니다).

<코드> ch06-무비차트수집.py

```
# 원하는 요소 가져오기
title = soup.select_one("h3 > a")
print(title)
```

</코드>

<실행결과>

```
<a href="/info/movieinfo/detail/20258383">나우 유 씨 미 3</a>
```

</실행결과>

이어서 a 요소 전체(나우 유 씨 미 3가 아니라 요소 안에 있는 영화 제목(나우 유 씨 미 3)만 출력해보겠습니다. 마지막 print() 문의 title 변수 뒤에 .text를 붙이면 되는데, .text는 해당 **요소의 내용**(화면에 보이는 텍스트, 여기서는 영화 제목)을 문자열로 추출하는 명령입니다. 이렇게 코드를 수정하고 실행하면 영화 제목이 출력됩니다.

<코드> ch06-무비차트수집.py

```
# 원하는 요소 가져오기
title = soup.select_one("h3 > a")
print(title.text)
```

</코드>

<실행결과>

```
나우 유 씨 미 3
```

</실행결과>

주석을 제외하면 6줄밖에 안 되는 코드로 실시간 1위 영화 제목을 출력하는 프로그램을 완성했습니다.

전체 영화 제목 출력하기

예매순위 페이지에 있는 전체 영화 제목을 출력하려면 1위뿐만 아니라 나머지 순위의 영화 제목들이 어떤 요소에 들어 있는지 확인해야 합니다. 2위 영화 제목에서 마우스 오른쪽 버튼을 눌러 [검사] 메뉴를 선택하고 해당 HTML 요소를 확인해봅시다. 그 아래에 코드 모양이 똑같은 <li> 요소의 오른쪽 삼각형(▶)을 클릭해 세부 내용을 펼쳐보면 3위 영화 제목도 확인할 수 있습니다.

[TIP] 개발자 도구의 코드에서 오른쪽 삼각형(▶)은 이하에 코드가 숨겨져 있다는 뜻으로, 이것을 클릭하면 코드를 펼칠 수 있습니다. 또한 코드 내의 점 3개(…)는 코드가 길어서 숨겨져 있다는 표시로, 이것을 클릭하면 해당 코드 전체를 볼 수 있습니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-20 2위 영화 제목 확인 후 3위 영화 제목 확인**

다행히도 무비 차트 웹 개발자가 모든 영화 제목 요소의 생김새를 똑같이 만들어놓았습니다. 그렇다면 컴퓨터에 "내가 원하는 요소의 생김새가 모두 `"h3 > a"`이니 그것을 전부 가져와"라고 명령하면 됩니다.

❶ 요소 하나만 가져와 출력했던 기존 코드를 주석 처리합니다.

❷ 찾는 요소가 여러 개 있을 때 `select_one()`은 HTML 문서의 위에서 아래로 탐색해 맨 처음에 찾은 요소만 가져오고, `select()`는 모든 요소를 가져옵니다. 지금은 모든 영화 제목을 가져와야 하므로 `select_one()` 대신 `select()`를 사용해 `"h3 > a"` 요소를 가져온 후 `title` 변수에 저장합니다.

❸ `title` 변수를 출력합니다. 이때 `.text`를 붙이지 않는데, 그 이유는 실행 결과를 확인한 뒤 설명하겠습니다. 지금은 `title`만 출력하세요.

<코드> ch06-무비차트수집.py

```
# title = soup.select_one("h3 > a") --- ❶ 주석 처리
# print(title.text) ------------------- ❶ 주석 처리
# 여러 요소 한 번에 가져오기
title = soup.select("h3 > a") ---------- ❷ 모든 h3 > a 가져오기
print(title) --------------------------- ❸ 출력
```

</코드>

<실행결과>

```
[<a href="/info/movieinfo/detail/20258383">나우 유 씨 미 3</a>, <a href="/info/movieinfo/detail/20256757">극장판 체인소 맨: 레제편</a>, (중략) <a href="/info/movieinfo/detail/20257795">베이비걸</a>]
```

</실행결과>

실행 결과의 양 끝이 []로 둘러싸여 있습니다. 이는 title 변수의 자료형이 리스트라는 뜻으로, 이 리스트 안에 모든 영화 제목 요소가 들어 있습니다. 앞에서 title 변수에 .text를 쓸 수 없다고 한 이유가 바로 이것입니다. 요소 하나만 가져올 때는 해당 요소의 내용을 .text로 추출할 수 있지만, 여러 요소를 가져와 리스트 자료형으로 저장한 title 변수에서는 .text로 추출할 수 없습니다.

이 문제를 해결하려면 반복문인 for 문을 사용해야 합니다. 리스트 자료형이 나오면 for 문이 바로 생각날 정도로 for 문은 리스트 자료형과 궁합이 잘 맞습니다 따라서 다음과 같이 코드를 수정합니다.

❶ 마지막 줄의 print(title) 문을 지우고 for i in title: 문을 추가합니다. for 문의 변수 i는 리스트 자료형인 title에서 원소를 하나씩 꺼내 저장합니다. 이때 i는 원소를 하나만 가져와 저장했으니 .text를 붙일 수 있습니다.

❷ i.text를 출력합니다.

<코드> ch06-무비차트수집.py

```
# 여러 요소 한 번에 가져오기
title = soup.select("h3 > a")
for i in title: ------ ❶ 기존 print() 문 삭제, for 문 작성
    print(i.text) ---- ❷ 출력
```

</코드>

코드를 실행하면 모든 순위의 영화 제목이 출력됩니다.

<실행결과>

```
나우 유 씨 미 3
(중략)
베이비걸
```

</실행결과>

영화 제목 앞에 '1위 : , 2위 : , ...'와 같은 문자열을 붙이고 싶다면 num 변수를 선언하고 1로 초기화한 후, for 문을 한 번 반복할 때마다 1씩 증가시킵니다. 그리고 문자열 포매팅 문장을 만들어 순위를 출력하면 됩니다. 문자열 포매팅이 기억나지 않으면 **3.2.7절 문자열 포매팅**을 참고하세요.

<코드> ch06-무비차트수집.py

```
# 여러 요소 한 번에 가져오기
title = soup.select("h3 > a")
num = 1
for i in title:
    print(f"{num}위 : {i.text}")
    num += 1
```

</코드>

<실행결과>

```
1위 : 나우 유 씨 미 3
(중략)
20위 : 베이비걸
```

</실행결과>

[NOTE] 순위를 표시하는 다양한 방법

순위를 표시하는 방법은 앞서 소개한 방법 외에 세 가지가 더 있습니다. 필자는 세 번째가 가장 세련되다고 생각하지만 이는 개인의 취향일 뿐 어떤 것을 사용해도 무방합니다.

<코드> 방법 1

```
title = soup.select("h3 > a")
for i in title:
    print(f"{title.index(i) + 1}위 : {i.text}")
```

</코드>

<코드> 방법 2

```
title = soup.select("h3 > a")
for i in range(len(title)):
    print(f"{i + 1}위 : {title[i].text}")
```

</코드>

<코드> 방법 3

```
title = soup.select("h3 > a")
for idx, i in enumerate(title):
    print(f"{idx + 1}위 : {i.text}")
```

</코드>

이로써 무비 차트 데이터를 실시간으로 수집하는 크롤링 프로그램을 완성했습니다.

<1분 퀴즈>

1 다음 코드에서 음영으로 표시된 곳에는 한 줄마다 잘못된 부분이 있습니다. 해당 부분을 찾아 바르게 고치세요.

<코드>

```
import requests
from bs4 import BeautifulSoup
code = requests.get(https://www.moviechart.co.kr/rank/realtime/index/image)
soup = BeautifulSoup(code, "html.parser")
title = soup.select("h3 >> a")
for i in title:
    print(title.text)
```

</코드>

<정답>

```
import requests
from bs4 import BeautifulSoup
code = requests.get("https://www.moviechart.co.kr/rank/realtime/index/image") # "로 감싸기
soup = BeautifulSoup(code.text, "html.parser") # .text 붙이기
title = soup.select("h3 > a") # >>를 >로 수정
for i in title:
    print(i.text) # title을 i로 수정
```

</정답>

7.5 이미지 수집하기: 228~234쪽

* CGV 웹 사이트가 개편됨에 따라 실습 사이트를 무비차트 사이트로 변경합니다.

7.5 이미지 수집하기

지금까지 웹 페이지에 있는 텍스트만 수집했는데, 좀 더 나아가 이미지도 수집해봅시다. 무비차트 사이트의 예매순위 페이지에서 영화 제목을 수집했었죠? 이번에는 영화 제목과 함께 포스터 이미지를 수집하겠습니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-26 영화 제목과 포스터 이미지**

지금부터는 CSS 선택자 지정 과정에 대한 설명을 생략하겠습니다. 앞에서 충분히 연습했으니 CSS 선택자를 스스로 찾을 수 있을 것입니다. 만약 스스로 하기 어렵다면 책에서 사용한 CSS 선택자를 그대로 따라 써보고, 왜 그러한 선택자를 사용했는지 이해하기 바랍니다.

7.5.1 이미지 URL 가져오기

[pythonStudy] 폴더에 **ch07-포스터이미지수집.py** 파일을 만듭니다. 그리고 무비차트 페이지(**https://www.moviechart.co.kr/rank/realtime/index/image**)에 접속해 개발자 도구를 엽니다.

이미지를 수집하려면 앞에서 배운 크롤링 방법과 마찬가지로 이미지 요소를 확인한 뒤 해당 요소를 가져오면 됩니다. 영화 제목과 포스터 이미지 요소의 선택자를 무엇으로 하면 좋을지 개발자 도구에서 확인합니다.

- **영화 제목 선택자**: "h3 > a"
- **포스터 이미지 선택자**: "li.movieBox-item > a > img"

선택자를 확인했다면 크롤링 코드를 작성합니다.

❶ 기본 크롤링 코드를 작성합니다.

❷ 모든 영화 제목 요소를 찾아 title_elements 변수에 저장합니다.

❸ 모든 포스터 이미지 요소를 찾아 image_elements 변수에 저장합니다.

❹ for 문으로 title_elements와 image_elements 리스트에 저장된 요소를 꺼내 각각 title, image 변수에 저장합니다.

❺ 영화 제목과 포스터 이미지 요소를 출력합니다.

❻ for 문을 한 번 반복할 때마다 구분선을 넣습니다.

<코드> ch07-포스터이미지수집.py

```
import requests --- ❶ 기본 크롤링 코드 작성
from bs4 import BeautifulSoup
code = requests.get("https://www.moviechart.co.kr/rank/realtime/index/image")
soup = BeautifulSoup(code.text, "html.parser")
title_elements = soup.select("h3 > a") --- ❷ 영화 제목 가져오기
image_elements = soup.select("li.movieBox-item > a > img") -- ❸ 포스터 이미지 가져오기
for title, image in zip(title_elements, image_elements): ---- ❹ 반복문 작성
    print(f"제목 : {title.text}") ---------- ❺ 영화 제목과 포스터 이미지 요소 출력
    print(image)
    print("----------------------------") -- ❻ 구분선 출력
```

</코드>

코드를 실행하면 영화 제목과 포스터 이미지 요소가 출력됩니다.

<실행결과>

```
제목 : 나우 유 씨 미 3
<img alt="나우 유 씨 미 3" src="/thumb?width=178&amp;height=267&amp;m_code=20258383&amp;source=https://admin.moviechart.co.kr/assets/upload/movie/251106024456_4838.jpg"/>
----------------------------
(중략)
----------------------------
```

</실행결과>

실행 결과에서 `<img>` 요소의 `src` 속성값을 보면 URL이 들어 있습니다. 이 URL을 클릭하면 영화 포스터가 있는 웹 페이지로 이동합니다. 이는 영화 포스터 이미지를 갖고 있는 서버의 주소가 `src` 속성값에 포함된 URL이라는 뜻입니다. 그렇다면 `.attrs`를 이용해 해당 URL을 가져올 수 있습니다. 코드를 수정하고 실행해보면 영화 포스터 이미지의 URL이 출력됩니다.

<코드> ch07-포스터이미지수집.py

```
for title, image in zip(title_elements, image_elements):
    print(f"제목 : {title.text}")
    print(image.attrs["src"])
    print("----------------------------")
```

</코드>

<실행결과>

```
제목 : 나우 유 씨 미 3
/thumb?width=178&height=267&m_code=20258383&source=https://admin.moviechart.co.kr/assets/upload/movie/251106024456_4838.jpg
----------------------------
(중략)
----------------------------
```

</실행결과>

이 때 출력되는 URL 주소가 미완성되어 있는 모습입니다. 보통 URL 주소는 https://www로 시작하니까요. 이렇게 출력되는 URL 주소가 미완성되어 있을 경우에는 앞에 홈페이지 메인 주소를 이어 붙여줍니다.

<코드> ch07-포스터이미지수집.py

```
for title, image in zip(title_elements, image_elements):
    print(f"제목 : {title.text}")
    print("https://www.moviechart.co.kr" + image.attrs["src"])
    print("----------------------------")
```

</코드>

<실행결과>

```
제목 : 나우 유 씨 미 3
https://www.moviechart.co.kr/thumb?width=178&height=267&m_code=20258383&source=https://admin.moviechart.co.kr/assets/upload/movie/251106024456_4838.jpg
----------------------------
(중략)
----------------------------
```

</실행결과>

7.5.2 이미지 내려받기

영화 포스터 이미지가 저장된 URL을 알았으니 이 이미지를 자신의 컴퓨터에 내려받겠습니다. 이미지를 내려받아 로컬 파일로 저장하려면 `urllib.request` 모듈이 필요하고, 이미지를 저장할 폴더를 만들려면 `os` 모듈이 필요합니다. `urllib.request` 모듈에는 `req`라는 별명을 붙이고, `os` 모듈은 그대로 불러옵니다.

<코드> ch07-포스터이미지수집.py

```
import requests
from bs4 import BeautifulSoup
import urllib.request as req
import os
```

</코드>

> [NOTE] 서브모듈
>
> urllib.request처럼 '모듈명.모듈명'으로 작성하는 모듈을 서브모듈이라고 합니다. 서브모듈은 모든 기능을 하나의 모듈에 넣어 구현하기 복잡할 때 원래 모듈(urllib) 내에 서브모듈(request)을 만들어 세부 기능을 분리해놓은 것입니다. 실제로 urllib 모듈 안에는 request라는 서브모듈뿐만 아니라 parse, cookie 등의 서브모듈이 있습니다.

for 문을 반복하기 전에 이미지를 내려받아 저장할 폴더를 생성합니다.

❶ 새 폴더의 경로를 큰따옴표로 감싸 문자열로 만들고 folder_name 변수에 저장합니다. 경로를 작성할 때 마침표를 찍으면 현재 프로젝트 폴더를 의미합니다. 따라서 ./무비차트 포스터는 현재 프로젝트 폴더 내에 '무비차트 포스터'라는 폴더를 만들라는 의미입니다.

❷ os 모듈의 mkdir() 명령을 사용하면 폴더를 만들 수 있습니다. 그런데 해당 폴더가 이미 존재하면 오류가 발생합니다. 따라서 if 문을 사용해 폴더 존재 여부를 확인하고, 폴더가 존재하지 않을 때 mkdir() 명령을 실행해야 합니다. 어떤 폴더가 존재하는지 확인하는 명령은 os.path 모듈의 exists()입니다.

<형식>

```
os.path.exists(폴더경로) # 폴더가 존재하면 True, 존재하지 않으면 False
```

</형식>

여기서는 폴더가 존재하지 않으니 os.path.exists()가 False를 반환하며, 그 앞에 not 연산자를 붙이면 True가 됩니다. 이러면 if 문을 충족해 내부 코드를 실행합니다. 폴더가 존재하는지 확인하고 폴더를 생성하는 if 문은 폴더를 만들 때 자주 사용하니 꼭 기억해두세요.

<코드> ch07-포스터이미지수집.py

```
image_elements = soup.select("li.movieBox-item > a > img")
# 이미지 내려받을 폴더 생성
folder_name = "./무비차트 포스터" --- ❶ 폴더 경로를 변수에 저장
if not os.path.exists(folder_name): - ❷ 폴더 생성
    os.mkdir(folder_name)
for title, image in zip(title_elements, image_elements):
    print(f"제목 : {title.text}")
```

</코드>

새로 만든 폴더에 포스터 이미지를 내려받습니다.

❶ 이미지 파일명을 영화 순위대로 짓기 위해 num 변수를 만들고 1로 초기화합니다.

❷ for 문에서 이미지 URL을 출력하는 대신 img_url 변수에 저장하는 코드로 수정합니다.

❸ req 모듈의 urlretrieve() 명령은 주어진 URL에서 데이터(이 경우에는 이미지)를 내려받아 로컬 파일로 저장합니다. 괄호 안의 첫 번째 자리에는 이미지 URL을 넣고, 두 번째 자리에는 이미지를 저장할 폴더 경로와 파일명을 넣습니다.

❹ 다음 이미지를 저장하기 위해 num 변수를 1 증가시킵니다.

<코드> ch07-포스터이미지수집.py

```
if not os.path.exists(folder_name):
    os.mkdir(folder_name)
num = 1 --- ❶ num 변수 생성
for title, image in zip(title_elements, image_elements):
    print(f"제목 : {title.text}")
    img_url = "https://www.moviechart.co.kr" + image.attrs["src"] --- ❷ 이미지 URL 저장
    req.urlretrieve(img_url, f"{folder_name}/{num}.jpg") ------------ ❸ 이미지 저장
    num += 1 --- ❹ 이미지 파일명 1 증가
    print("----------------------------")
```

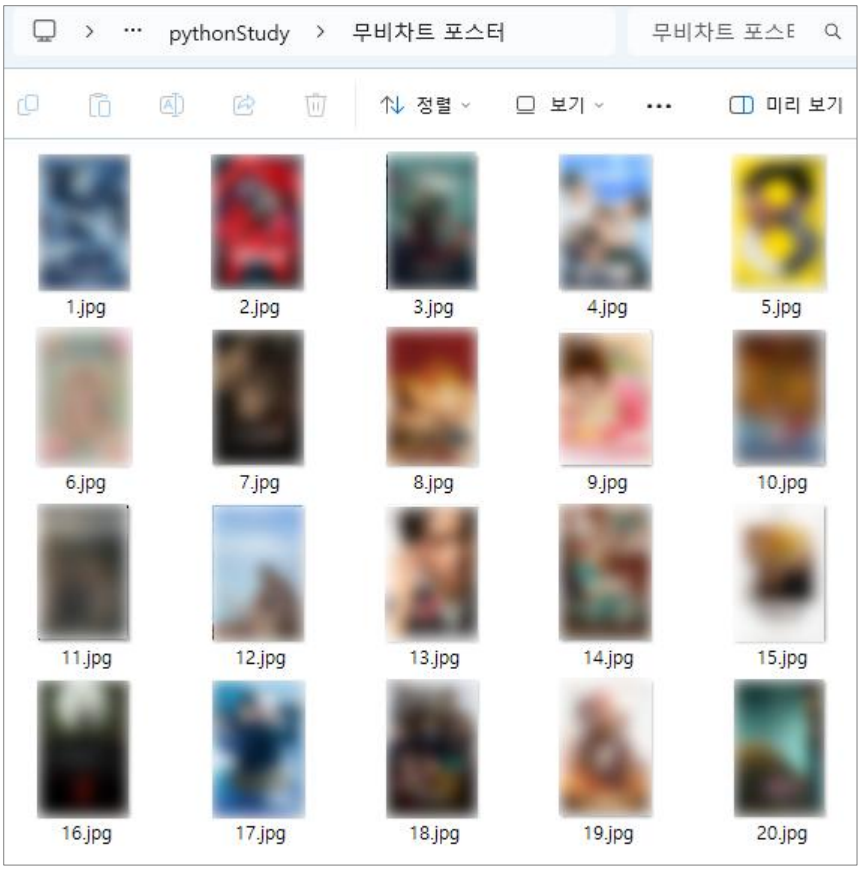
</코드>

코드를 실행하고 프로젝트 폴더를 확인합니다.

- **윈도우**: C:\Users\사용자계정명\PycharmProjects\pythonStudy
- **맥OS**: /Users/사용자계정명/PycharmProjects/pythonStudy

[무비차트 포스터] 폴더가 생성됐고, 여기에 영화 포스터 이미지가 저장돼 있습니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-27 영화 포스터 이미지 수집 결과**



필자의 지인 중 한 아이돌 삼촌팬은 덕질을 하기 위해 이 코드를 활용하고, 한 디자이너는 레퍼런스 자료를 수집하기 위해 이 코드를 활용합니다. 여러분도 이제 `urlretrieve()` 명령을 사용해 다양한 사이트에서 이미지를 수집할 수 있겠죠?

7.6 수집한 정보 엑셀에 저장하기: 235~244쪽

* CGV 웹 사이트가 개편됨에 따라 실습 사이트를 무비차트 사이트로 변경합니다.

7.6 수집한 정보 엑셀에 저장하기

지금까지는 크롤링한 결과를 실행창에 출력했는데, 이번에는 엑셀 파일에 저장해봅시다. 이를 위해 6장에서 `openpyxl`과 `pillow` 모듈을 설치했습니다. 만약 모듈이 설치돼 있지 않다면 **6.1.1절 크롤링 관련 모듈 설치하기**를 참고해 설치하세요.

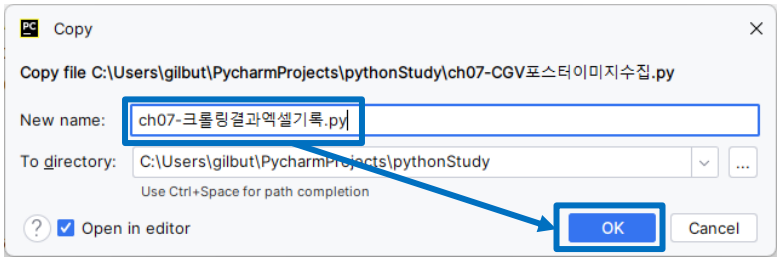
7.6.1 추가 정보 수집하기

앞에서 실습했던 **ch07-포스터이미지수집.py** 코드를 업그레이드해 영화 제목, 포스터 이미지뿐만 아니라 예매율, 개봉일 정보도 수집하겠습니다.

그림 7-28 예매율과 개봉일

프로젝트 탐색기의 **ch07-포스터이미지수집.py**에서 마우스 오른쪽 버튼을 눌러 [Copy]를 선택합니다. 그리고 이 상태에서 다시 마우스 오른쪽 버튼을 눌러 [Paste]를 선택합니다. [Copy] 창이 뜨면 파일 이름을 **ch07-크롤링결과엑셀기록.py**로 수정하고 [OK] 버튼을 클릭합니다.

그림 7-29 파일 복사 후 파일명 수정



복사한 코드에서 다음 코드를 추가합니다.

❶ 예매율 요소와 개봉일 요소의 CSS 선택자를 확인하고 각각 `percent_elements`, `date_elements` 변수에 저장합니다.

- **예매율 선택자**: `"li.ticketing > span"`
- **개봉일 선택자**: `"li.movie-launch"`

❷ `for` 문에 예매율, 개봉일 정보를 가져옵니다.

❸ 예매율과 개봉일을 출력합니다. 이때 주의할 점이 있습니다. 개봉일 요소의 내용을 그대로 `date.text`로 추출하면 개봉일 앞에 있는 불필요한 글자(개봉일 :)도 같이 출력됩니다. 따라서 이를 다듬기 위해 문자열 슬라이싱을 이용해 날짜 부분만 잘라냅니다.

그림 7-30 개봉일 앞뒤의 공백과 불필요한 글자가 있는 상태

<코드> 07장-크롤링결과엑셀기록.py

```
import requests
(중략)
image_elements = soup.select("li.movieBox-item > a > img")
percent_elements = soup.select("li.ticketing > span") -- ❶ 요소 가져오기
date_elements = soup.select("li.movie-launch")
(중략)
for title, image, percent, date in zip(title_elements, image_elements, percent_elements, date_elements): ------------------------ ❷ 예매율, 개봉일 추가
    print(f"제목 : {title.text}")
    print(f"예매율 : {percent.text}") -- ❸ 예매율, 개봉일 출력
    print(f"개봉일 : {date.text.strip()[6:]}")
```

</코드>

코드를 실행하면 예매율과 개봉일까지 잘 출력됩니다.

<실행결과>

```
제목 : 나우 유 씨 미 3
예매율 : 13.70%
개봉일 : 2025.11.12
----------------------------
(중략)
----------------------------
```

</실행결과>

[NOTE] 문자열 슬라이싱

3.3.2절 리스트 출력 방법에서 설명했듯이 슬라이싱은 리스트에서 여러 개의 원소를 구간별로 꺼낼 때 사용하는 문법입니다. 3.3.2절에서는 슬라이싱을 리스트에서 한다고 했는데 문자열에서도 가능합니다. 문자열의 첫 글자에 인덱스 0번이 매겨지므로 이 인덱스를 이용하면 한 글자만 가져오는 인덱싱(예: `[0]`)과 여러 글자를 가져오는 슬라이싱(예: `[:10]`)을 할 수 있습니다.

7.6.2 엑셀에 기록하기

텍스트 기록하기

크롤링 결과 중 텍스트에 해당하는 제목, 예매율, 개봉일을 엑셀 파일에 기록하겠습니다. 엑셀 파일에 기록하는 것은 복잡한 일이 아니라, 수집한 정보를 기록하는 다음 문장만 알고 있으면 됩니다.

<형식>

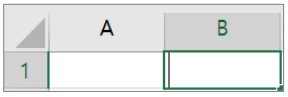
```
sheet.cell(row=행 번호, column=열 번호).value = "넣고 싶은 값"
```

</형식>

물론 이 문장 앞에 엑셀 파일을 생성하고 시트를 불러오는 코드 등을 추가로 작성해야 합니다. 하지만 그러한 것은 나중에 살펴보고 우선 이 문장을 이해해봅시다.

여기서 sheet는 엑셀 시트를 통째로 저장하는 변수입니다. 이 변수에 cell() 명령을 쓰면 엑셀의 셀 하나를 클릭한 것과 같은 상태가 됩니다. 예를 들어 sheet.cell(row=1, column=2)의 경우 1번 행과 2번 열에 해당하는 셀을 클릭한 상태가 됩니다. 이 상태에서 sheet.cell(row=1, column=2).value와 같이 뒤에 .value를 붙이면 해당 셀을 더블클릭해 그 셀에 키보드 커서를 놓은 것과 같은 상태, 즉 키보드로 뭔가를 입력할 수 있는 상태가 됩니다. 따라서 뒤에 "넣고 싶은 값"을 =(할당 연산자)로 넣으면 해당 셀에 값이 들어갑니다.

그림 7-31 sheet.cell(row=1, column=2).value 실행 시의 상태



이를 코드로 작성해보겠습니다. 실행창에 출력했던 값(title.text, percent.text, date.text.strip()[6:])을 sheet.cell().value = 문장의 오른쪽에 작성합니다. 이때 해당 값을 엑셀 시트의 몇 번째 행, 몇 번째 열에 기록할지를 cell() 괄호 안에 행 번호와 열 번호로 써야 하는데, 행 번호의 경우 num 변수가 1부터 시작해 for 문을 돌면서 1씩 증가하므로 이를 활용합니다. 또한 영화 제목은 2번 열에, 예매율은 3번 열에, 개봉일은 4번 열에 기록하기로 하고 열 번호를 직접 입력합니다. 이렇게 하면 수집한 영화 제목, 예매율, 개봉일을 엑셀 시트에 넣을 수 있습니다.

<코드> ch07-크롤링결과엑셀기록.py

```
for title, image, percent, date in zip(title_elements, image_elements, percent_elements, date_elements):
    (중략)
    req.urlretrieve(img_url, f"{folder_name}/{num}.jpg")
    # 엑셀 시트에 영화 제목, 예매율, 개봉일 기록
    sheet.cell(row=num, column=2).value = title.text
    sheet.cell(row=num, column=3).value = percent.text
    sheet.cell(row=num, column=4).value = date.text.strip()[6:]
    num += 1
```

</코드>

수집한 정보를 엑셀 파일에 기록하는 주요 코드를 작성했으니 엑셀 파일을 만들고 저장하는 나머지 코드를 작성합니다.

❶ 엑셀 파일 생성에 필요한 `openpyxl` 모듈을 불러옵니다.

❷ `openpyxl` 모듈의 `Workbook()` 명령으로 엑셀 파일을 만들어 `book` 변수에 저장합니다.

❸ `book` 변수의 엑셀 파일에서 현재 활성화된 시트를 `book.active`로 불러옵니다. `active`는 현재 활성화된 엑셀 시트를 반환하는 명령입니다.

❹ 모든 작업을 마치고 `for` 문을 빠져나와 `book.save()`로 엑셀 파일을 저장합니다. `save()`는 `book` 변수의 엑셀 파일을 저장하는 명령으로, 괄호 안에 파일의 저장 경로를 넣습니다. 여기서는 현재 프로젝트 폴더(`.`)에 **무비차트.xlsx**라는 파일명으로 저장했습니다.

<코드> ch07-크롤링결과엑셀기록.py

```
import os

import openpyxl -------------- ❶ 모듈 불러오기

# 엑셀 파일 생성

book = openpyxl.Workbook() --- ❷ 엑셀 파일 생성

sheet = book.active ---------- ❸ 현재 시트 불러오기

(중략)

for title, image, percent, date in zip(title_elements, image_elements, percent_elements, date_elements):

    (중략)

    print("----------------------------")

book.save("./무비차트.xlsx") --- ❸ 파일 저장
```

</코드>

코드를 실행하면 크롤링 결과가 엑셀에 저장됩니다.

그림 7-32 영화 제목, 예매율, 개봉일 저장

	A	B	C	D
1		나우 유 씨	13.70%	2025.11.12
2		극장판 체	11.50%	2025.09.24
3		프레데터:	8.70%	2025.11.05
4		퍼스트 라	7.30%	2025.10.29
5		8번 출구	2.60%	2025.10.22
6		세계의 주	2.10%	2025.10.22
7		구원자	2.00%	2025.11.05
8		부고니아	1.90%	2025.11.05
9		극장판 똘	1.80%	2025.11.07
10		극장판 귀	1.70%	2025.08.22
11		난징사진관	1.30%	2025.11.05
12		엄마를 버	1.20%	2025.11.05
13		국보	0.80%	2025.11.19
14		달팽이 농	0.50%	2025.11.12
15		럭키 데이	0.50%	2025.11.12
16		뱀파이어	0.40%	2025.11.12
17		극장판 주	0.40%	2025.10.16
18		어쩔수가없	0.30%	2025.09.24
19		원 배틀 애	0.30%	2025.10.01
20		베이비걸	0.30%	2025.10.29

영화 포스터 이미지 기록하기

끝으로 크롤링한 영화 포스터 이미지를 엑셀에 저장해보겠습니다. 크게 세 단계로 나눠 진행합니다. 첫째, 앞에서 내려받은 영화 포스터 이미지의 크기가 크기 때문에 작게 줄입니다. 둘째, 크기를 조절한 이미지를 엑셀 시트에 첨부합니다. 셋째, 이미지가 셀에 저장됐을 때 다 보일 수 있도록 셀의 행 높이와 열 너비를 조절합니다.

세 단계를 코딩하기 전에 필요한 모듈을 불러옵니다.

❶ openpyxl.drawing.image 모듈은 이미지를 삽입하고 조작하는 데 사용하는데, 그중 Image만 불러옵니다.

❷ PIL 모듈(pillow 모듈)은 이미지를 처리하기 위해 사용합니다. 모듈 내 Image를 불러와야 하는데, 앞서 불러온 것과 이름이 같으니 PILImage로 별명을 붙여 불러옵니다.

<코드> ch07-크롤링결과엑셀기록.py

```
import openpyxl
from openpyxl.drawing.image import Image ---- ❶
from PIL import Image as PILImage ---------- ❷
```

</코드>

[TIP] pillow 모듈은 설치할 때는 pillow지만 코드에서 불러와 사용할 때는 PIL이라고 씁니다. 설치할 때와 코드에서 사용할 때의 모듈 이름이 다릅니다.

첫 번째 단계로 이미지 크기를 조정합니다.

❶ for 문 안에서 PILImage.open() 명령으로 영화 포스터 이미지 파일을 엽니다. 괄호 안에는 이미지 파일의 파일 경로를 넣습니다. 그리고 열린 이미지 파일을 img 변수에 저장합니다.

❷ img 변수에 저장된 영화 포스터 이미지의 너비와 높이를 img.resize() 명령으로 조정합니다. 첫 번째 인자로 너비를, 두 번째 인자로 높이를 기입하면 되는데, 여기서는 1/2 크기로 줄이고 img_resized 변수에 저장합니다.

❸ img_resized 변수에 저장된 파일을 img_resized.save() 명령으로 새로 저장합니다. 괄호 안에는 새로 저장할 파일의 경로를 넣습니다.

❹ ❶번에서 열었던 이미지 파일을 img.close() 명령으로 닫습니다.

<코드> ch07-크롤링결과엑셀기록.py

```
for title, image, percent, date in zip(title_elements, image_elements, percent_elements, date_elements):
    (중략)
    req.urlretrieve(img_url, f"{folder_name}/{num}.jpg")
    # 이미지 크기 조절
    img = PILImage.open(f"{folder_name}/{num}.jpg") ┐ ❶~❷ 이미지 가져와 조정
    img_resized = img.resize((int(img.width * 0.5), int(img.height * 0.5)))
    img_resized.save(f"{folder_name}/{num}_resized.jpg") --- ❸ 새로 저장
    img.close() --- ❹ 이미지 닫기
    # 엑셀 시트에 영화 제목, 예매율, 개봉일 기록
```

</코드>

두 번째 단계에서는 크기를 조정해 새로 저장한 이미지를 엑셀 파일에 첨부합니다. 이때 `sheet.add_image()` 명령을 사용합니다.

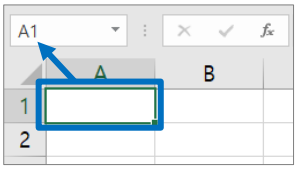
<형식>

```
sheet.add_image('Image(파일경로)', 셀이름)
```

<형식>

`sheet.add_image()` 명령의 괄호 안 첫 번째 자리에는 `Image(파일경로)` 명령으로 가져올 이미지를 넣고, 두 번째 자리에는 이미지가 첨부될 셀 이름을 넣습니다. 여기서는 A 열에 이미지를 첨부하므로 이미지가 A1, A2, A3, … 셀 순으로 들어갑니다. 마침 `num` 변수가 행 번호를 갖고 있으니 문자열 포매팅을 활용해 `f"A{num}"`으로 작성합니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-33 셀 이름 확인 방법**



<코드> ch07-크롤링결과엑셀기록.py

```
# 이미지 크기 조절
(중략)
img.close()
# 엑셀에 이미지 첨부
sheet.add_image(Image(f"{folder_name}/{num}_resized.jpg"), f"A{num}")
```

</코드>

마지막으로 for 문을 반복하면서 각 행의 높이를 조절하고, for 문을 모두 반복했을 때는 열 너비를 조절합니다. 행 높이를 조절하는 데에는 sheet.row_dimensions[행번호].height 명령을 사용하고, 열 너비를 조절하는 데에는 sheet.column_dimensions[열이름].width 명령을 사용합니다.

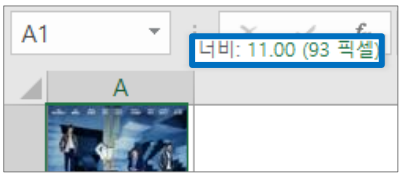
<코드> ch07-크롤링결과엑셀기록.py

```
for title, image, percent, date in zip(title_elements, image_elements, percent_elements, date_elements):
    (중략)
    sheet.cell(row=num, column=4).value = date.text.strip()[6:]
    # 행 높이 조절
    sheet.row_dimensions[num].height = 98
    num += 1
    print("----------------------------")
# 열 너비 조절
sheet.column_dimensions["A"].width = 11
sheet.column_dimensions["B"].width = 35
sheet.column_dimensions["C"].width = 15
sheet.column_dimensions["D"].width = 21
book.save("./무비차트.xlsx") # 파일 저장
```

</코드>

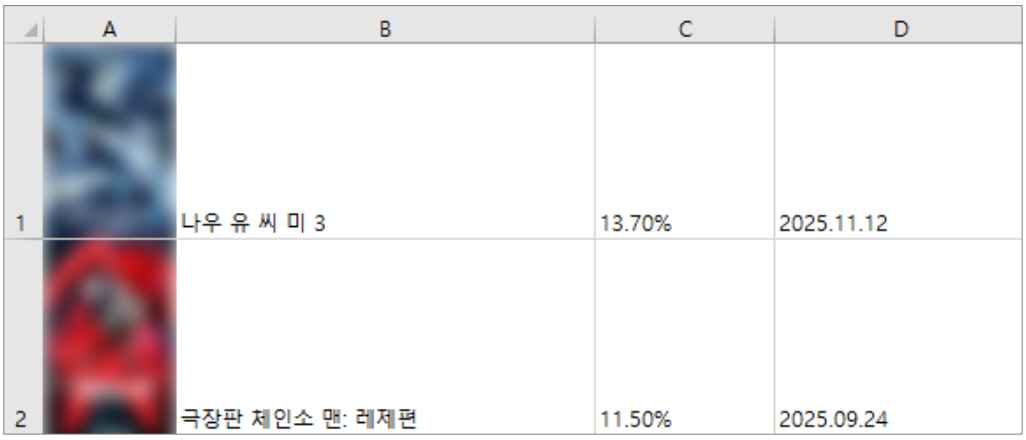
적절한 높이와 너비는 임의의 숫자를 넣고 실행한 후 엑셀에 첨부된 이미지를 직접 확인하면서 찾아야 합니다. 엑셀에서 마우스로 열 너비를 조절하면 다음 그림과 같이 열 너비에 대한 정보가 뜨는데, 93 픽셀 말고 11.00이라는 숫자를 파이썬 코드에 기입하면 됩니다. 행 높이도 같은 방법으로 조절합니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-34 엑셀에서 열 너비 확인**



열 너비와 행 높이를 조절한 후 최종적으로 실행하면 엑셀 파일에 적절한 크기의 영화 포스터 이미지가 들어갑니다(엑셀 파일이 열려 있다면 파일을 닫고 실행합니다).

그림 오류! 지정한 스타일은 사용되지 않습니다.**-35 결과가 기록된 엑셀 파일**



여기서 더 업그레이드하려면 셀에 기록된 글자의 폰트를 키우고, 셀 색깔을 바꾸고, 가운데 정렬을 하는 등 다양한 서식을 변경할 수 있습니다. 엑셀 서식을 변경하는 방법은 **12장 엑셀 자동화 기본기 익히기**에서 자세히 설명하겠습니다.

[TIP] 세 단계의 코드가 복잡해 보이지만 부담 갖지 않아도 됩니다. 처음 코딩을 배우는 입문자는 '이 많은 코드를 언제 다 외워서 활용하지?'라고 걱정하는데, 절대 외울 필요 없으니 안심하세요. 아무리 실력이 좋은 개발자라도 모든 명령의 사용법을 외우지는 않습니다. 지금은 코드가 어떤 기능을 하는지 정도만 이해하면 됩니다. 나중에 이러한 기능이 필요한 순간이 찾아오면 해당 코드를 복사, 붙여넣기해 활용하면 됩니다. 코드 복사, 붙여넣기가 왠지 프로답지 못하고 실력이 부족해 보이는 것 같지만, 실제 개발자들이 가장 많이 사용하는 키는 복사, 붙여넣기 단축키입니다.

9.4.2 문서 요약 API 키 발급받기: 326~337쪽

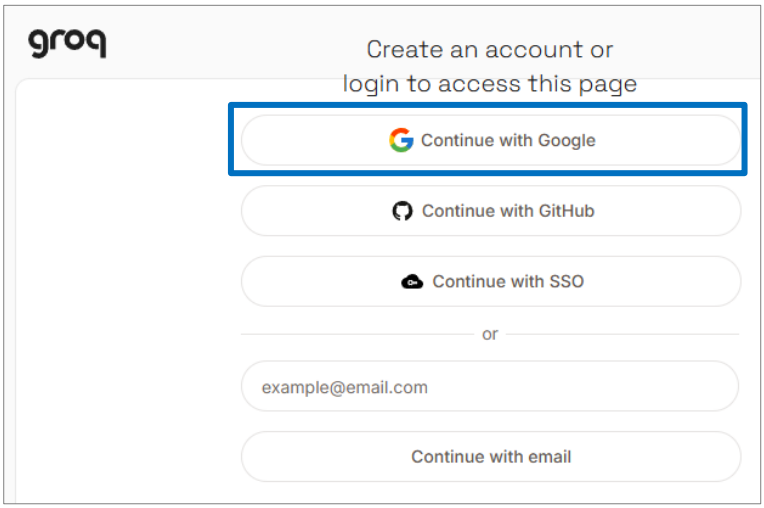
** 네이버 문서 요약 API가 서비스를 종료함에 따라 Groq API로 변경합니다.

9.4.2 문서 요약 API 키 발급받기

수집한 기사 본문을 요약하기 위해 Groq의 LLM(Large Language Model, 대형 언어 모델) API를 사용하겠습니다. LLM은 사람처럼 자연스러운 문장을 생성하는 인공지능 모델로, Groq의 API는 빠르고 성능 좋은 최신 LLM을 무료로 제공해 문서 요약 작업에 유용하게 사용할 수 있습니다.

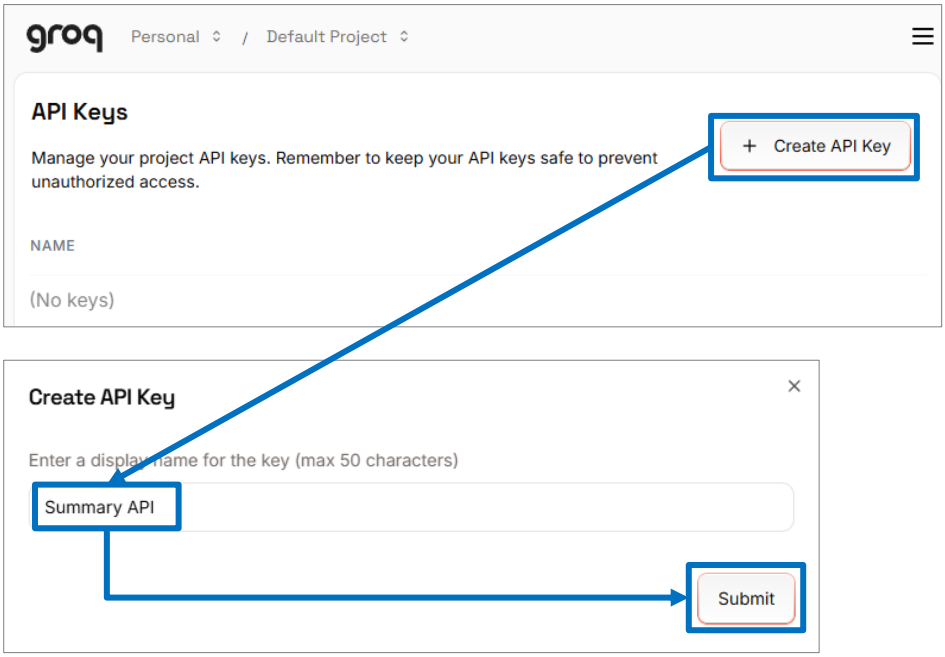
Groq API 사이트(**https://console.groq.com/keys**)에 접속해 로그인합니다. 구글 계정이 있다면 [Continue with Google]을 클릭해 편리하게 로그인할 수 있습니다.

그림 9-32 로그인 화면



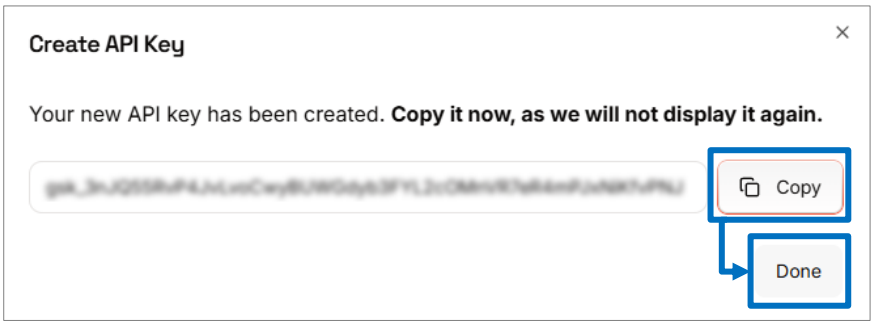
로그인이 완료되면 [Create API Key] 버튼을 클릭해 API Key를 발급받습니다. API 이름은 자유롭게 지어도 됩니다. 여기서는 **Summary API**라고 짓고 [Submit] 버튼을 클릭합니다.

그림 9-33 API Key 발급



API Key가 생성되면 [Copy] 버튼을 클릭해 복사한 후 메모장에 붙여넣고 [Done] 버튼을 클릭합니다.

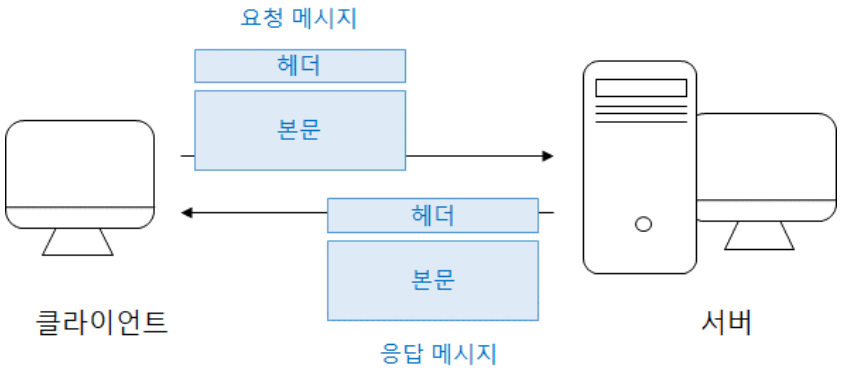
그림 9-34 API Key 복사



9.4.3 뉴스 기사 요약하기

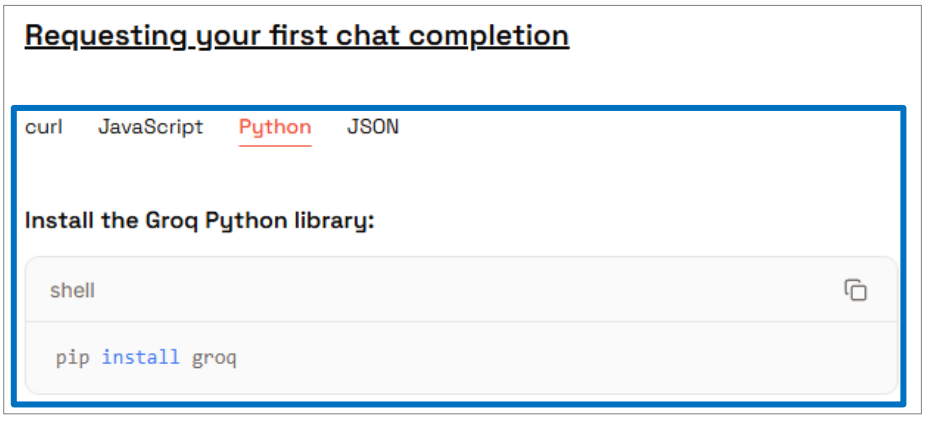
발급받은 API를 사용해 뉴스 기사를 요약해봅시다. 그 전에 이해를 돕기 위해 **6.2.2절 웹 페이지의 동작 원리**에서 배웠던 내용을 떠올려보겠습니다. 클라이언트와 서버는 요청 메시지와 응답 메시지를 주고받으며 통신한다고 설명했는데, API 호출도 이와 마찬가지입니다. 클라이언트에서 API 요청 메시지를 보내면 서버가 이를 받아 처리한 후 응답 메시지를 반환합니다. 요청 메시지와 응답 메시지는 각각 헤더와 본문으로 구성되며, 헤더에는 요청과 관련된 각종 정보가 담기고 본문에는 실제 전송되는 데이터가 담깁니다.

그림 9-35 API 호출 시 주고받는 메시지



다시 실습으로 돌아와, 문서 요약 API의 설명서를 살펴보기 위해 Groq API 설명서(**https://console.groq.com/docs/quickstart**)에 접속합니다. 설명서를 보면 Groq API를 사용하려면 `groq`이라는 파이썬 모듈을 설치해야 된다고 합니다.

그림 9-36 Groq API 설명서



실행창의 Python Packages에서 groq(0.23.0 버전) 모듈을 설치합니다.

그림 9-37 groq 모듈 설치

Groq API 설명서에서 스크롤을 내려보면 **Performing a Chat Completion:** 코드가 있습니다. 이는 Groq API 호출 및 응답 코드로, 다음 그림에 표시한 것처럼 ❶번은 발급받은 API Key를 넣어주는 부분이고, ❷번은 API 서버에 보낼 요청 메시지를 작성하는 부분, ❸번은 API 서버에서 받은 응답 메시지를 출력하는 부분입니다.

그림 9-38 Groq API 호출 및 응답 코드

Performing a Chat Completion:

Python

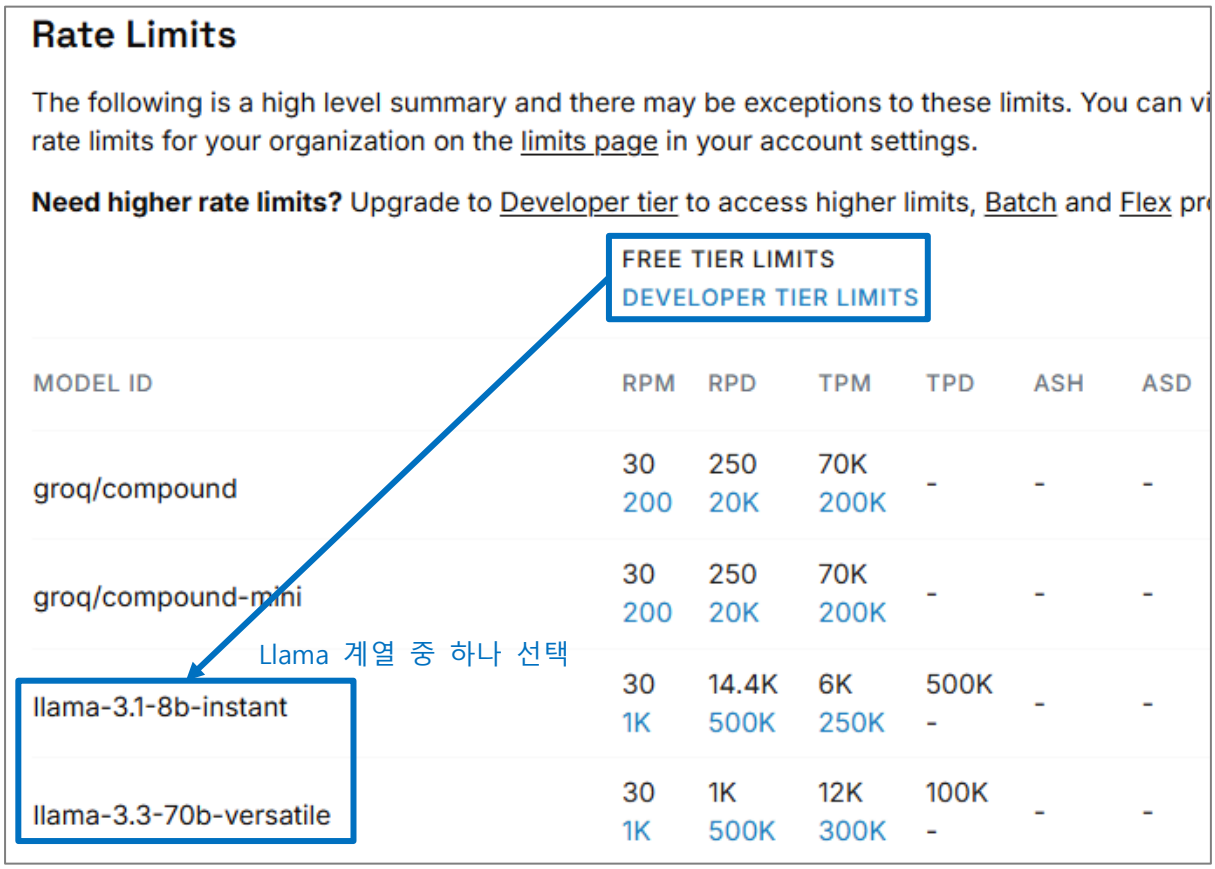
```python
import os

from groq import Groq

client = Groq(
    api_key=os.environ.get("GROQ_API_KEY"),
)

chat_completion = client.chat.completions.create(
    messages=[
        {
            "role": "user",
            "content": "Explain the importance of fast language models",
        }
    ],
    model="llama-3.3-70b-versatile",
)

print(chat_completion.choices[0].message.content)
```

❶ ❷ ❸

여기서 ❷번의 `"content"`와 `model`에 들어가는 값을 주목해주세요.

- `"content"`: LLM 모델에 요청할 문장을 입력합니다. 챗GPT에게 궁금한 것을 물어보거나 어떤 작업을 명령할 때 작성하는 프롬프트와 같습니다. 지금은 수집한 뉴스 기사를 요약해야 하므로 **"이 뉴스 기사를 요약해줘."**라고 작성하면 됩니다.
- `model`: 사용할 LLM 모델을 설정합니다. Groq에서는 DeepSeek, Llama, Qwen 등의 LLM 모델을 지원하며, 지원하는 모델 리스트는 Rate Limits(요금 제한) 페이지(**https://console.groq.com/docs/rate-limits**)에서 확인할 수 있습니다. 본 실습에서는 무료인 Llama 계열의 **llama-3.3-70b-versatile** 모델을 사용하겠습니다(이후 모델은 계속해서 변경이 있을 수 있으므로 FREE TIER의 모델 리스트에서 Llama 계열을 골라 사용하세요).

그림 9-39 LLM 모델 선택



뉴스 기사를 요약하기 위한 Groq API 호출 및 응답 코드를 작성하면 다음과 같습니다.

<코드>

```
from groq import Groq


client = Groq(
    api_key="발급받은_API_Key_입력"
)


chat_completion = client.chat.completions.create(
    messages=[
        {
            "role": "user",
            "content": f"다음 뉴스 본문을 한국어로 3줄로 간략히 요약해줘. 각
                       요약 항목은 줄바꿈해. {뉴스_본문}" --- 프롬프트 입력
         }
    ],
    model="llama-3.3-70b-versatile", --- 사용할 LLM 모델 설정
)


print(chat_completion.choices[0].message.content) --- 응답 메시지 출력
```

</코드>

ch09-뉴스기사요약 파일로 돌아와 Groq API 호출 및 응답 코드를 추가합니다.

❶ groq 모듈을 추가합니다.

❷ 발급받은 API Key를 저장하고, 서버에 보낼 요청 메시지를 작성합니다. 요청 메시지의 "content"에는 뉴스 본문을 3줄로 요약해달라는 문장과 함께 수집한 뉴스 본문인 news_content를 문자열 포매팅으로 문자열에 넣어줍니다.

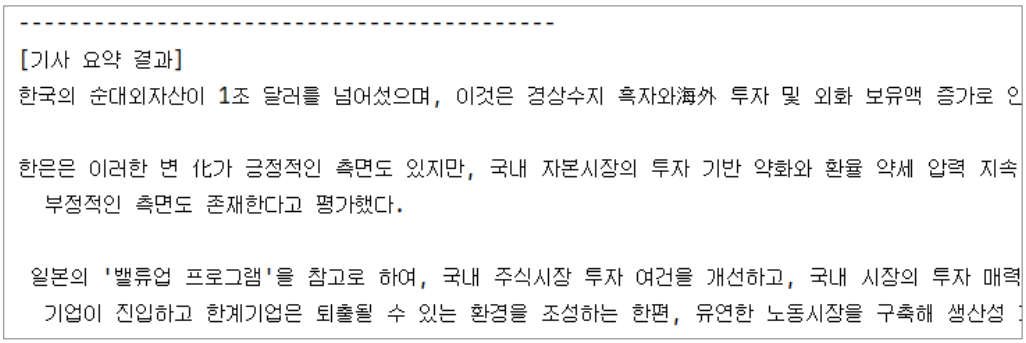
❸ 구분선을 출력한 다음 LLM 모델의 응답 메시지를 출력합니다.

<코드> ch09-뉴스기사요약.py

```
from newspaper import Article
from groq import Groq ---------- ❶
(중략)
news_content = article.text
print(news_content)


# Groq API 호출 및 응답받기 --- ❷
client = Groq(
    api_key="발급받은_API_Key_입력"
)
chat_completion = client.chat.completions.create(
    messages=[
        {
            "role": "user",
            "content": f"다음 뉴스 본문을 한국어로 3줄로 간략히 요약해줘. 각
                       요약 항목은 줄바꿈해. {news_content}"
         }
    ],
    model="llama-3.3-70b-versatile",
)


# 기사 요약문 출력 ------------- ❸
print("----------------------------------------")
print("[기사 요약 결과]")
print(chat_completion.choices[0].message.content)
```

코드 실행 결과 3줄로 요약된 결과가 잘 출력됩니다.

그림 9-40 문서 요약 결과



9.4.4 요약 결과를 음성으로 변환하기

끝으로 요약한 문서를 음성으로 변환해봅시다. 이 작업을 하려면 gtts와 playsound 모듈이 필요하므로 코드 상단에 두 모듈을 불러오는 문장을 추가합니다.

❶ gtts(Google Text-to-Speech)는 구글의 텍스트를 음성으로 변환하는 모듈로, 별도의 API 키를 발급받지 않고 무료로 사용할 수 있습니다. from 다음의 gtts에 빨간색 밑줄이 생기므로 실행창의 Python Packages로 가서 gtts(버전 2.5.1) 모듈을 설치합니다.

❷ playsound는 오디오 파일을 재생하는 모듈입니다. from 다음의 playsound에 빨간색 밑줄이 생기므로 실행창의 Python Packages로 가서 playsound(버전 1.2.2) 모듈을 설치합니다.

<코드> ch09-뉴스기사요약.py

```
from groq import Groq
from gtts import gTTS  ------------ ❶ gtts 모듈 설치
from playsound import playsound --- ❷ playsound 모듈 설치
```

</코드>

[TIP] gtts 모듈로 텍스트 음성 변환을 너무 자주 요청하거나 너무 긴 텍스트를 음성으로 변환하려고 하면 거절당할 수도 있습니다. 이때는 불가피하게 유료 버전을 사용해야 하는데, '구글 클라우드 tts api', '네이버 tts api', '카카오 tts api'를 검색하면 유료 tts API를 찾을 수 있습니다.

코드의 맨 아래에 다음 문장을 추가합니다.

❶ gTTS()는 text에 음성으로 변환할 텍스트를 받고 lang에 변환할 언어(ko, 한국어)를 지정해 텍스트를 음성으로 변환하는 명령입니다. 변환한 음성은 comment_to_voice에 저장합니다.

❷ 변환한 음성을 comment_to_voice.save() 명령을 활용해 "news.mp3" 파일로 저장합니다.

❸ playsound() 명령으로 "news.mp3" 파일을 재생합니다.

<코드> ch09-뉴스기사요약.py

```
print(chat_completion.choices[0].message.content)

# 텍스트-음성 변환                    ┌ ❶ 요약 문서를 음성으로 변환
comment_to_voice = gTTS(text=chat_completion.choices[0].message.content, lang="ko")
comment_to_voice.save("news.mp3") -- ❷ mp3 파일로 저장
playsound("news.mp3") -------------- ❸ mp3 파일 재생
```

</코드>

코드를 실행하면 요약된 기사가 음성으로 재생됩니다. 이렇게 텍스트를 음성으로 변환하는 TTS API를 활용하면 매일 『뉴욕타임스』의 기사를 수집하고 음성으로 변환해 이를 영어 듣기 학습에 사용할 수 있습니다. 또한 매일 아침 컴퓨터를 켰을 때 그날 수신된 이메일을 읽는 프로그램을 만들 수도 있습니다. 반대로 음성을 텍스트로 변환하는 음성 인식 API를 활용하면 회의 중 녹음한 음성 파일을 텍스트로 변환하고 요약해 회의록을 작성하는 프로그램을 만들 수 있습니다. 이처럼 텍스트-음성 변환 또는 음성-텍스트 변환 API는 일상과 업무에 유용하게 사용할 수 있는 도구입니다.

<1분 퀴즈>

3 앞에서 실습한 코드에 대한 설명 중 옳지 <u>않은</u> 것을 고르세요.

① newspaper는 뉴스 기사를 크롤링하고 기사 본문 내용을 가공하는 모듈이다.

② Groq API는 DeepSeek, Llama, Qwen 등 다양한 LLM 모델을 무료로 제공한다.

③ Groq API를 사용하면 문서 요약 작업을 할 수 있다.

④ 구글의 gtts 모듈은 별도의 API 키를 발급받아야 사용할 수 있다.

<정답>④</정답>

10.1.2 [좋아요] 자동 누르기: 347~355쪽

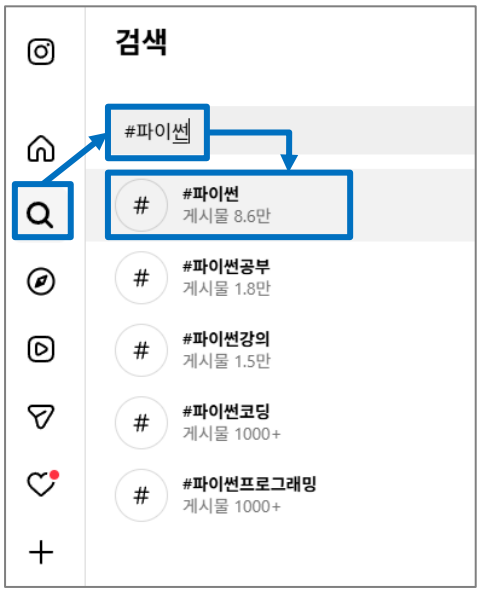
* 인스타그램 [좋아요] 요소의 CSS 선택자가 바뀌어 설명을 수정합니다.

10.1.2 [좋아요] 자동 누르기

로그인 완료 화면에서 해시태그를 검색하는 동작을 명령하는 코드를 작성하겠습니다. 해시태그를 검색하는 과정은 ❶ 왼쪽의 돋보기 아이콘 클릭, ❷ # 다음에 원하는 키워드 입력, ❸ 검색 결과에 뜬 해시태그 클릭 순으로 진행됩니다. 이는 파이썬 코드로 클릭, 키보드 입력과 같은 명령을 작성해 구현할 수 있지만 여기서는 좀 더 쉬운 방법을 소개하겠습니다.

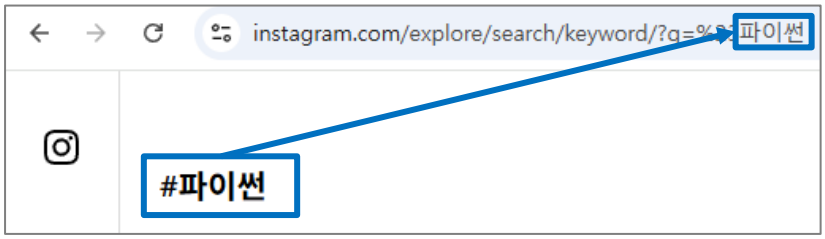
로그인 완료 화면에서 왼쪽에 있는 돋보기 아이콘을 누르고 검색창에 '#파이썬'을 입력한 후 목록에서 [#파이썬]을 선택합니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-4 '파이썬' 검색**



검색 결과 페이지로 이동하면 다음과 같이 URL에 해시태그 키워드(파이썬)가 들어 있습니다. 이는 **https://www.instagram.com/explore/search/keyword/?q=%23** 뒤에 원하는 키워드만 넣으면 해당 키워드를 검색한 페이지로 바로 이동할 수 있다는 뜻입니다. 예를 들어 '서울맛집'이라는 키워드를 넣은 **https://www.instagram.com/explore/search/keyword/?q=%23서울맛집**의 경우 '서울맛집'을 검색한 페이지로 이동합니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-5 해시태그 검색 결과 페이지의 URL**



해시태그 검색 결과 페이지의 URL을 알았으니 다음과 같이 해시태그를 입력받아 해당 키워드의 검색 페이지에 접속하는 코드를 추가합니다. 로그인한 후 일정 시간 대기할 수 있도록 time.sleep() 문도 잊지 않고 작성합니다.

<코드> ch10-인스타그램좋아요.py

```
# 크롬 브라우저 열어 인스타그램 로그인 페이지 접속
keyword = input("해시태그 입력 >> ")
(중략)
# 로그인하기
(중략)
button.click()
time.sleep(7)
# 해시태그 검색 페이지 접속
browser.get(f"https://www.instagram.com/explore/search/keyword/?q=%23{keyword}")
time.sleep(6)
```

</코드>

코드를 실행하면 해시태그를 입력하라는 입력 상자가 나타납니다. 인스타그램에서는 해시태그에 띄어쓰기가 포함되면 검색이 되지 않으니 띄어쓰기를 하지 않도록 주의하며 '파이썬'을 입력합니다.

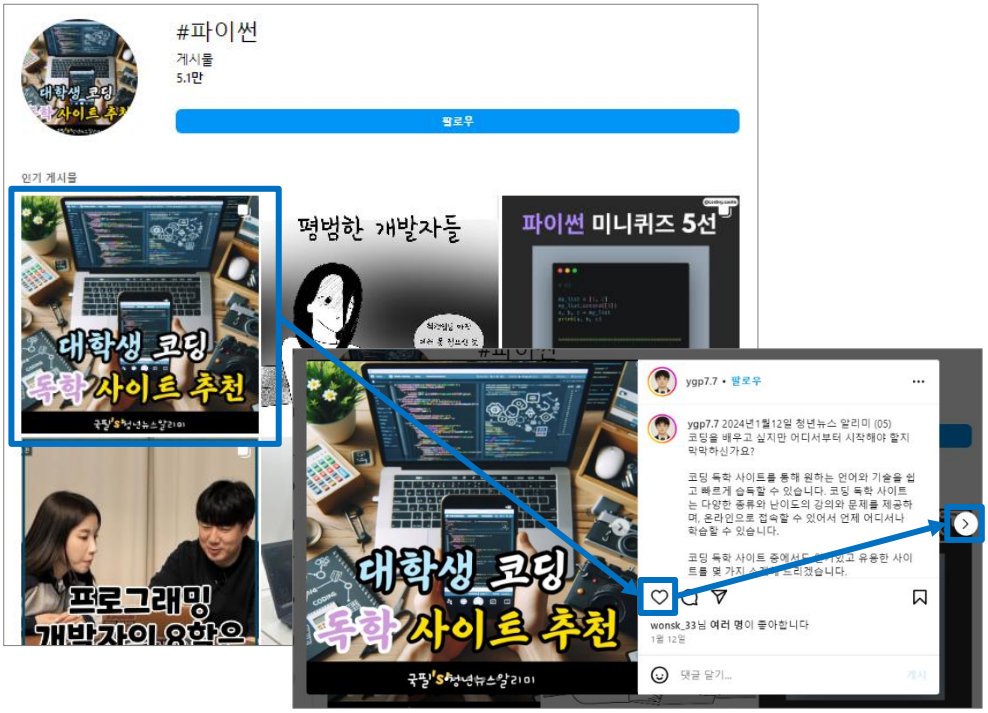
<실행결과>

```
해시태그 입력 >> 파이썬
```

</실행결과>

크롬 창이 뜨면서 인스타그램의 '파이썬' 검색 페이지로 넘어갑니다. 여기서 첫 번째 게시물을 클릭하고 [좋아요]를 누른 후 >(화살표)를 클릭해 다음 게시물로 넘어가는 과정을 모든 검색 게시물에 대해 수행해보겠습니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-6 [좋아요] 자동 누르기 순서**



이 작업에는 무한 반복문을 사용해야 합니다. 또한 첫 번째 게시물 클릭하기와 [좋아요] 누르고 다음 게시물로 넘어가기로 나눠 구현하겠습니다.

그런데 코드를 작성하기에 앞서 주의할 점이 있습니다. 인스타그램의 CSS 선택자는 주기적으로 변합니다. 필자가 확인해본 바로는 5~6개월에 한 번씩 변하기 때문에 실습 시점의 CSS 선택자가 이 책의 CSS 선택자와 다를 수 있습니다. 따라서 실습할 때는 이 책의 CSS 선택자를 그대로 사용하지 말고 해당 요소의 CSS 선택자를 꼭 확인한 후 작성하세요. 인스타그램은 HTML 코드가 복잡해 CSS 선택자를 작성하기가 어려운데, 이러한 과정을 통해 크롤링 실력을 키울 수 있습니다.

첫 번째 게시물 클릭하기

검색 결과 페이지에서 [Ctrl]+[Shift]+[C] 키를 눌러 개발자 도구를 열고 첫 번째 게시물의 HTML 요소를 확인합니다. CSS 선택자를 `"div._aagu"`로 지정하면 될 것 같은데, 실제로 [Ctrl]+[F] 키를 눌러 'div._aagu'를 검색해보면 28개의 게시물 요소가 검색되므로 제대로 찾았다는 것을 알 수 있

습니다.

그림 10-7 첫 번째 게시물의 CSS 선택자

find_element() 명령으로 첫 번째 게시물을 가져와 클릭합니다.

<코드> ch10-인스타그램좋아요.py

```
# 해시태그 검색 페이지 접속
browser.get(f"https://www.instagram.com/explore/search/keyword/?q=%23{keyword}")
time.sleep(6)
# 첫 번째 게시물 클릭
first_photo = browser.find_element(By.CSS_SELECTOR, "div._aagu")
first_photo.click()
time.sleep(5)
```

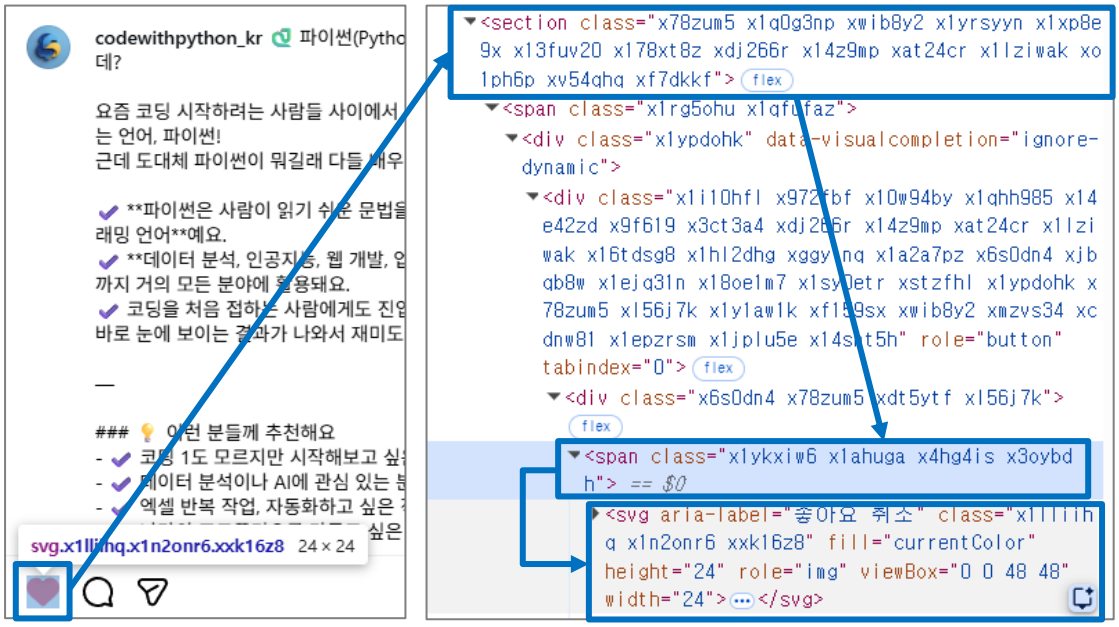
</코드>

[좋아요] 누르고 다음 게시물로 넘어가기

첫 번째 게시물의 [좋아요]에 해당하는 HTML 요소를 확인해봅시다. [좋아요]는 게시물에도 있고 댓글에도 있기 때문에 게시물의 [좋아요]를 정확히 지정하기 위해 조부모인 <section> 요소를 활용합니다. 다시 말해 <section> 요소의 후손인 <svg> 요소로 게시물의 [좋아요] 선택자를 작성합니다.

- **게시물의 [좋아요] 선택자**:
  "section.x78zum5.x1q0g3np.xwib8y2.x1yrsyyn.x1xp8e9x.x13fuv20.x178xt8z.xdj266r.x14z9mp.xat24cr.x1lziwak.xo1ph6p.xv54qhq.xf7dkkf span > svg"

그림 10-8 [좋아요] CSS 선택자

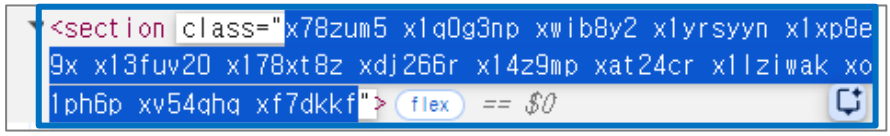


[TIP] 앞서 언급했듯이 인스타그램은 HTML 요소의 class 속성값이 주기적으로 바뀌기 때문에 **그림 10-8**의 요소와 여러분이 실제로 확인한 요소가 다를 수 있습니다. 이 점을 염두에 두고 CSS 선택자를 작성할 때는 항상 해당 요소를 직접 찾아 CSS 선택자로 사용하세요.

<NOTE> 오타 없이 CSS 선택자 작성하는 법

<section> 요소의 class 속성값이 너무 길어 직접 타이핑하면 오타가 발생하기 쉬운데, 해당 속성값을 복사해 사용하는 방법이 있습니다. 개발자 도구에서 해당 class 속성값을 더블클릭하면 자동으로 선택되며, 그 상태에서 [Ctrl]+[C] 키를 누르면 속성값이 복사됩니다. 속성값이 길다 싶으면 꼭 이 방법을 사용하세요.

그림 10-9 HTML 코드 복사



더블클릭 후 [Ctrl]+[C]

다음 게시물로 넘어가는 >(화살표)의 HTML 요소도 확인해봅시다. 마찬가지로 조부모인 요소를 활용해 요소의 자손인 <button> 요소의 후손인 <svg> 요소로 작성합니다.

- **> 선택자**: "div._aagg._aaqh > button._abl- svg"

그림 10-10 > CSS 선택자

[좋아요]와 >의 선택자를 확인했으니 무한 반복문을 돌며 두 요소를 찾아 변수에 저장합니다.

<코드> ch10-인스타그램좋아요.py

```
# 첫 번째 게시물 클릭
(중략)
time.sleep(5)
# 좋아요 자동 누르기 시작
while True:
    like = browser.find_element(By.CSS_SELECTOR, "section.x78zum5.x1q0g3np.
            xwib8y2.x1yrsyyn.x1xp8e9x.x13fuv20.x178xt8z.xdj266r.x14z9mp.xat24cr.
            x1lziwak.xo1ph6p.xv54qhq.xf7dkkf span > svg")
    next = browser.find_element(By.CSS_SELECTOR, "div._aaqg._aaqh > button._abl- svg")
```

</코드>

이제 [좋아요]를 누른 후 >(화살표)를 누르면 되는데 문제가 있습니다. 이미 [좋아요]를 누른 게시물의 [좋아요]를 다시 누르면 [좋아요]가 취소되는 문제가 발생합니다. 이 문제를 해결해보겠습니다.

10.1.3 [좋아요] 취소 건너뛰기

[좋아요]가 취소되는 것을 막으려면 두 가지로 나눠 생각해야 합니다. 게시물의 [좋아요]가 눌려 있지 않으면 [좋아요]를 누른 후 다음 게시물로 넘어가고, [좋아요]가 눌려 있으면 바로 다음 게시물로 넘어가야 합니다. 이 두 가지 경우를 주석으로 구분하고 코드를 작성합니다.

<코드> ch10-인스타그램좋아요.py

```
# 좋아요 자동 누르기 시작
while True:
    like = browser.find_element(By.CSS_SELECTOR, "section.x78zum5.x1q0g3np.
            xwib8y2.x1yrsyyn.x1xp8e9x.x13fuv20.x178xt8z.xdj266r.x14z9mp.xat24cr.
            x1lziwak.xo1ph6p.xv54qhq.xf7dkkf span > svg")
    next = browser.find_element(By.CSS_SELECTOR, "div._aaqg._aaqh > button._abl- svg")
    # 좋아요가 눌려 있지 않다면
        browser.execute_script("document.querySelector('svg[aria-label=\"좋아요\"]').
                        dispatchEvent(new MouseEvent('click', {bubbles: true}));")
        time.sleep(2)
        next.click()
        time.sleep(2)
    # 좋아요가 눌려 있다면
        next.click()
        time.sleep(2)
```

</코드>

앞의 코드에서는 [좋아요]를 누르기 위해 like.click()이 아닌 browser.execute_script() 명령을 사용했습니다. 현재 인스타그램 사이트에서 click()을 사용하지 못하게 막았기 때문입니다(아마도 매크로 봇을 차단하기 위한 조치로 보입니다). 이처럼 selenium 모듈의 명령(여기서는 click())이 어떠한 이유로 동작하지 않을 때는 execute_script() 명령을 사용하면 대부분 해결할 수 있습니다. 이는 자바스크립트로 작성한 코드를 실행하는 명령으로, execute_script()의 괄호 안 코드가 바로 [좋아요]를 누르는 명령입니다.

이제 문제는 '[좋아요]가 눌려 있지 않다면'을 어떻게 코드로 표현하느냐입니다. 힌트를 찾기 위해 [좋아요] 요소를 다시 한번 살펴봅시다. [좋아요]가 눌려 있지 않을 때와 눌려 있을 때의 요소는 다음과 같습니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-11 [좋아요] 요소의 HTML 코드**

(a) [좋아요]가 눌려 있지 않을 때

```
▶<svg aria-label="좋아요" class="x1lliihq x1n2onr6 xyb1xck"
fill="currentColor" height="24" role="img" viewBox="0 0 24
24" width="24">…</svg>
```

(b) [좋아요]가 눌려 있을 때

```
▶<svg aria-label="좋아요 취소" class="x1lliihq x1n2onr6 xxk1
6z8" fill="currentColor" height="24" role="img" viewBox="0
0 48 48" width="24">…</svg>
```

차이점을 발견했나요? [좋아요]가 눌려 있느냐 아니냐에 따라 aria-label 속성값이 다릅니다. aria-label 속성값이 "좋아요"이면 눌려 있지 않은 것이고, "좋아요 취소"이면 눌려 있는 것입니다. 따라서 코드의 주석을 다음과 같이 바꿀 수 있습니다.

<코드> ch10-인스타그램좋아요.py

```
# like 요소의 aria-label 속성값이 "좋아요"이면
    browser.execute_script("document.querySelector('svg[aria-label=\"좋아요\"]').
                    dispatchEvent(new MouseEvent('click', bubbles: true}));")
    time.sleep(2)
    next.click()
    time.sleep(2)
# like 요소의 aria-label 속성값이 "좋아요 취소"이면
    next.click()
    time.sleep(2)
```

</코드>

다음은 이를 if 문으로 작성한 것입니다.

<코드> ch10-인스타그램좋아요.py

```
# like 요소의 aria-label 속성값이 "좋아요"이면
if like.get_attribute("aria-label") == "좋아요":
    browser.execute_script("document.querySelector('svg[aria-label=\"좋아요\"]').
                    dispatchEvent(new MouseEvent('click', bubbles: true}));")
    time.sleep(2)
    next.click()
    time.sleep(2)
# like 요소의 aria-label 속성값이 "좋아요 취소"이면
else:
    next.click()
    time.sleep(2)
```

</코드>

코드를 실행하면 자동으로 [좋아요]를 누르되, 이미 [좋아요]가 눌려 있는 게시물은 건너뛰고 다음 게시물로 넘어갑니다. 크롬 창을 닫아 무한 반복되는 프로그램을 중지합니다. 이어서 무한 반복문 실행과 관련해 인스타그램의 매크로 정책에 대해 알아보겠습니다.

10.3.2 라인 API 사용하기: 378~386쪽

** 라인 API가 서비스를 종료함에 따라 ntfy API로 변경했습니다.

10.3.2 ntfy API 사용하기

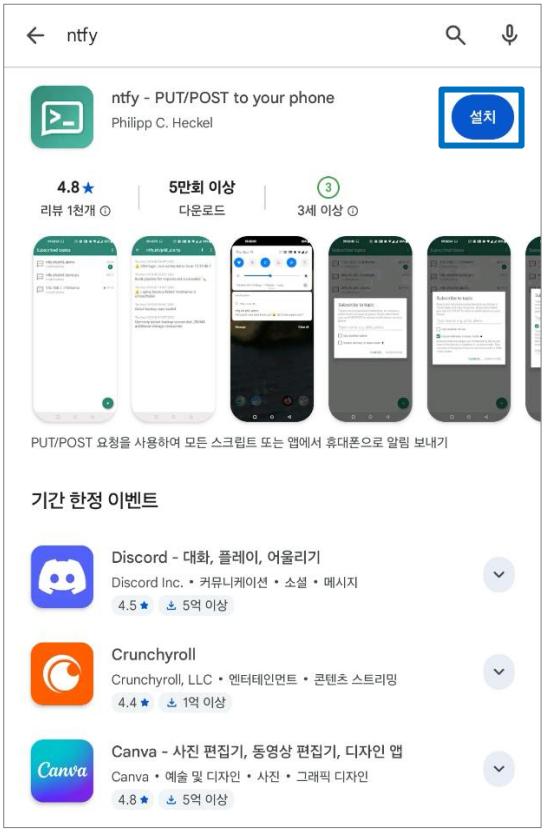
중고나라에 관심 물건이 올라왔을 때 스마트폰으로 알려주는 방법에는 문자 메시지를 이용하는 방법과 앱의 알림 서비스를 이용하는 방법이 있습니다. 문자 메시지의 경우 구글에서 '문자 메시지 API'를 검색하면 관련 API를 쉽게 찾을 수 있습니다. 그러나 건당 10원 이하의 비용이 들고 긴 문자의 경우 30원이므로 여기서는 앱의 알림 서비스를 이용하겠습니다.

앱을 통한 알림 서비스로는 카카오톡이 가장 유명합니다. 카카오에서 제공하는 '나에게 보내기'와 '카카오 채널' 기능을 이용하면 알림 서비스를 구현할 수 있습니다. 그러나 둘 다 한계가 있습니다. '나에게 보내기'의 경우 메시지는 잘 보내지만 스마트폰의 알림이 울리지 않고, '카카오 채널'의 경우 스마트폰의 알림은 울리지만 해당 API를 비즈니스 목적으로만 사용할 수 있기 때문에 API 신청 시 사업자 등록 번호가 필요합니다.

이래저래 카카오톡은 제한이 있으니 이 실습에서는 ntfy라는 무료 알림 서비스를 사용하겠습니다. ntfy는 설치와 설정이 간단하며, 별도의 회원가입이나 API 키 없이도 바로 사용할 수 있습니다.

먼저 자신의 스마트폰에 ntfy 앱을 설치합니다. 안드로이드의 경우 구글 플레이스토어(Play Store), 아이폰의 경우 애플 앱스토어(App Store)에서 'ntfy'를 검색해 설치할 수 있습니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-32 ntfy 앱 설치**



ntfy 앱을 사용하려면 앱에서 간단한 설정을 해야 합니다. ntfy 앱을 실행해 알림을 **허용**하고, 홈 화면에서 + 아이콘을 터치해 구독할 주제를 등록합니다. 여기서 주제란 알림 태그와 같습니다. 예를 들어 주제를 'test'로 설정하면, 누군가 'test'라는 태그로 알림을 보냈을 때 내 스마트폰에 알림이 울리게 됩니다. 주제는 최대한 본인만 알 수 있는 이름으로 짓습니다. 너무 단순하게 지으면 모르는 사람이 나에게 알림을 보내게 될 수 있기 때문입니다. 다음 그림에서는 주제를 **pythonstudy_1234**라고 지었으며, 여러분은 책과 다른 이름(예: pythonstudy_본인생일 등)으로 지으세요.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-33 알림 허용 및 구독할 주제 등록**

PC 화면으로 돌아와 [pythonStudy] 폴더에 **ch10-알림봇** 파일을 만들고, 스마트폰 알림 API를 호출하는 코드를 작성합니다.

❶ 앱에서 등록했던 주제(알림 태그)를 `tag` 변수에 문자열로 저장합니다.

❷ 알림의 제목을 `headers`의 `"Title"`에 작성합니다.

❸ 알림의 내용을 `message` 변수에 문자열로 저장합니다.

❹ 제목에 한글이 들어 있을 경우 한글이 깨지지 않도록 설정합니다.

❺ `requests.post()` 명령으로 해당 토큰을 전달하고 알림 API를 호출합니다. 이 명령은 ntfy API

설명서(**https://docs.ntfy.sh/publish/#__tabbed_1_7**)에서 제공한 내용을 참조해 작성한 것으로, 첫 번째 인자로 API 호출 URL("https://ntfy.sh/{tag}")을, 두 번째 인자로 요청 본문(data=message.encode("utf-8"))을, 세 번째 인자로 요청 헤더(headers=headers)를 전달합니다. requests 모듈에 빨간색 밑줄이 생기면 [Alt]+[Enter] 키를 눌러 requests 모듈을 추가합니다.

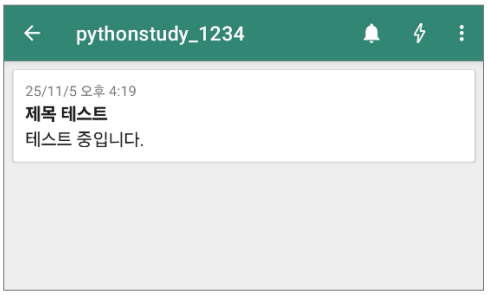
<코드> ch10-알림봇.py

```
import requests ---------------------- ❺ 모듈 자동 추가

tag = "pythonstudy_1234" --------------- ❶ 주제(알림 태그) 등록
headers = {"Title" : "제목 테스트"} --- ❷ 알림 제목 저장
message = "테스트 중입니다." ---------- ❸ 알림 내용 저장

headers["Title"] = headers["Title"].encode("utf-8") --- ❹ 한글 깨짐 방지
requests.post(f"https://ntfy.sh/{tag}", data=message.encode("utf-8"), headers=headers)
                                                    └ ❺ 알림 API 호출
```

</코드>

코드를 실행하면 스마트폰으로 알림 메시지가 옵니다.

그림 오류! 지정한 스타일은 사용되지 않습니다.**-34 알림 메시지**



data 변수에 다음 내용을 추가하면 메시지와 함께 이미지를 보낼 수 있습니다.

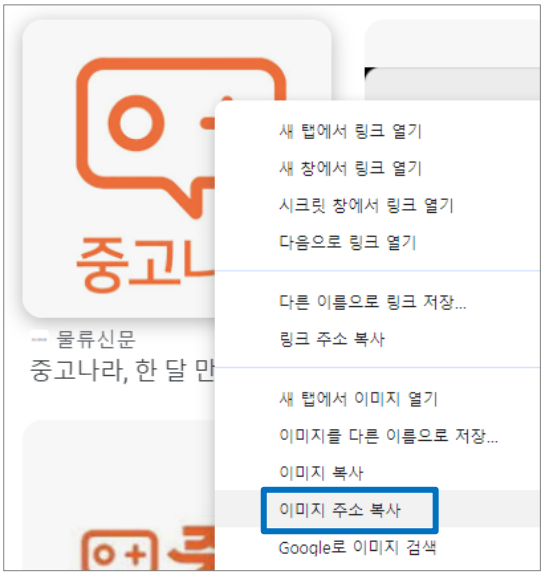
<코드> ch10-알림봇.py

```
import requests


tag = "pythonstudy_1234"
headers = {"Title" : "제목 테스트"}
message = "테스트 중입니다."


headers["Title"] = headers["Title"].encode("utf-8")
headers["Attach"] = "https://platum.kr/wp-content/uploads/2020/08/
                     58809805_2597174657088183_7071241453283835904_o.jpg"
requests.post(f"https://ntfy.sh/{tag}", data=message.encode("utf-8"), headers=headers)
```

</코드>

여기서 headers["Attach"]에는 이미지가 저장된 웹 사이트의 URL을 넣습니다. 가령 어떤 웹 페이지에 있는 이미지를 넣고 싶은 경우, 해당 이미지에서 마우스 오른쪽 버튼을 눌러 [이미지 주소 복사]를 선택하면 이미지가 저장돼 있는 URL이 복사되는데, 이 URL을 넣으면 됩니다. 앞의 코드에서는 platum.kr 사이트에 있는 이미지의 URL을 넣었습니다.

그림 10-35 이미지 주소 복사



코드를 실행하면 스마트폰으로 알림 메시지와 함께 이미지가 전송되며, 수신된 이미지를 터치하면 더 크게 볼 수 있습니다(다만 아이폰은 ntfy 앱 내부에 이미지를 표시하는 기능이 없어 이미지를 볼 수 없습니다).

그림 오류! 지정한 스타일은 사용되지 않습니다.**-36 알림 메시지와 이미지 확인**

이 코드를 앞에서 만든 **ch10-중고나라관심물건알림** 파일에 적용해보겠습니다. 이미지를 같이 보낼 필요는 없으니 이미지 보내는 부분을 없애고 알림 메시지와 중고나라 게시판의 URL(https://cafe.daum.net/talingpython/rRa6)을 같이 보냅니다.

그림 10-37 중고나라 게시판의 URL

ch10-중고나라관심물건알림 파일에서 다음 코드를 추가합니다. `requests` 모듈에 빨간색 밑줄이 생기면 [Alt]+[Enter] 키를 눌러 `requests` 모듈을 추가합니다.

<코드> ch10-중고나라관심물건알림.py

```
import os
import requests --- 모듈 자동 추가
(중략)
while True:
    (중략)
    if new_one >= 1:
        print(f"[알림] 청소기 관련 글이 {new_one}개 올라왔습니다.")
        tag = "pythonstudy_1234"
        headers = {"Title": "중고나라 관심 물건 알림"}
        message = f"[알림] 청소기 관련 글이 {new_one}개 올라왔습니다.
                    https://cafe.daum.net/talingpython/rRa6"
        headers["Title"] = headers["Title"].encode("utf-8")
        requests.post(f"https://ntfy.sh/{tag}", data=message.encode("utf-8"), headers=headers)
    browser.refresh() # 새로 고침
    time.sleep(10)
```

</코드>

중고나라 게시판에 청소기 관련 글을 올리고 코드를 실행하면 스마트폰으로 다음과 같은 메시지를 받게 되고, 메시지의 URL을 누르면 바로 중고나라 게시판으로 이동합니다(다만 아이폰은 본문에 있는 URL을 자동으로 연결하지 않습니다).

그림 오류! 지정한 스타일은 사용되지 않습니다.**-38 중고나라 관심 물건 알림 메시지**

이 코드를 응용하면 실제 중고나라 사이트에 올라온 물건의 사진도 메시지로 받을 수 있습니다. 또한 다른 사이트에 올라오는 관심 있는 뉴스, 사업 공고, 각종 공연, 비행기 특가, 기차 예매 등의 정보를 스마트폰 알림으로 받을 수 있습니다.

11.1.1 네이버 메일의 SMTP 설정하기: 394~402쪽

** 네이버는 2025년 10월부터 SMTP 서버 사용 시 별도의 비밀번호를 사용하도록 정책을 바꿨습니다. 이에 따라 11.1.1절을 실습한 후 아래 순서대로 보안 설정된 비밀번호를 생성하세요. 그리고 11.1.2절부터 예제 코드에 새로 생성한 비밀번호를 입력해 실습합니다.

<보안 설정 비밀번호 발급 방법>

❶ **https://nid.naver.com**에 접속합니다.

❷ **보안설정 > 2단계 인증 > 네이버 로그인 비밀번호 재확인 > 인증 알림을 받을 수 있는 기기 설정 > 인증 알림 기기에서 2단계 인증 요청 허락**을 거쳐 2단계 인증 설정을 완료합니다.

❸ 다시 **https://nid.naver.com**에 접속한 후 **보안설정 > 2단계 인증 > 네이버 로그인 비밀번호 재확인**을 합니다.

❹ 애플리케이션 비밀번호 생성에서 '종류선택'의 **아웃룩**을 선택하고, '비밀번호 생성'의 [생성하기] 버튼을 클릭합니다.

❺ '비밀번호 확인'에 생성된 비밀번호를 복사하여 메모장에 붙여넣습니다.

보안 설정을 통해 비밀번호 새로 생성

<예제 코드에 보안 설정 비밀번호 입력>

<코드> ch11-이메일발송.py

```
import smtplib --- ❶ 모듈 자동 추가
import openpyxl                          ┌ ❶ 네이버 SMTP 서버 연결
naver_server = smtplib.SMTP_SSL("smtp.naver.com", 465)
naver_id = "여기에 네이버 아이디 입력" -- ❷ 네이버 아이디, 비밀번호 저장
naver_pw = "여기에 네이버 보안설정 비밀번호 입력"
my_mail = f"{naver_id}@naver.com" ------- ❸ 발송자의 이메일 주소 저장
naver_server.login(naver_id, naver_pw) -- ❹ 네이버 SMTP 서버에 로그인
book = openpyxl.load_workbook("./주문내역.xlsx")
```

</코드>