

티처블머신과 함께 하는 인공지능 교육!

메카넘휠 AI 자율주행로봇 "단비"

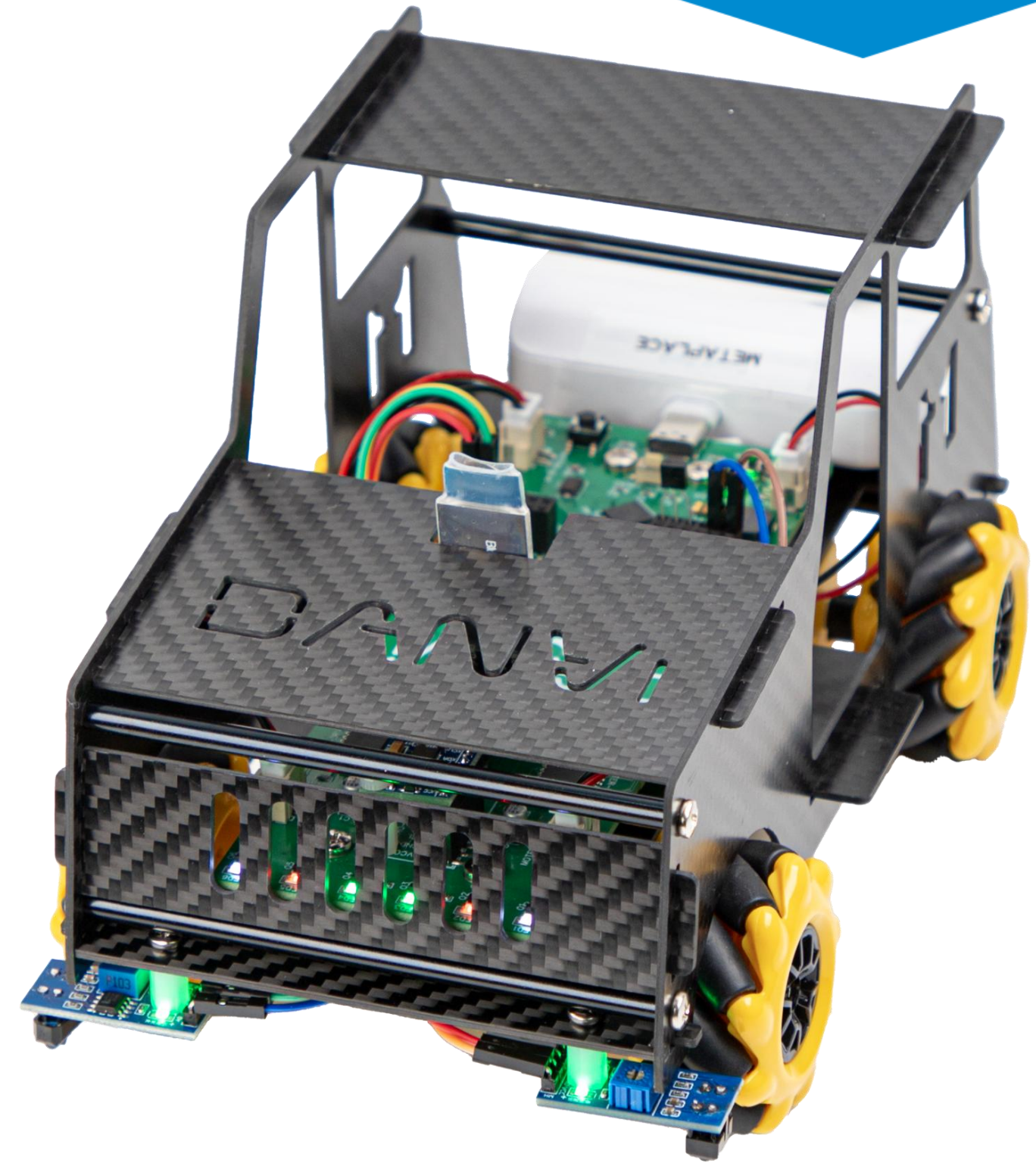
METAPLACE



(주)메타플레이스
강원특별자치도 춘천시 후석로 462번길 7 춘천ICT벤처센터 313호
T.033-252-4787

자유롭게 이동하는 메카넘휠 AI 로봇 단비

Mecanum Wheel Robot AI DanVI



METAPLACE

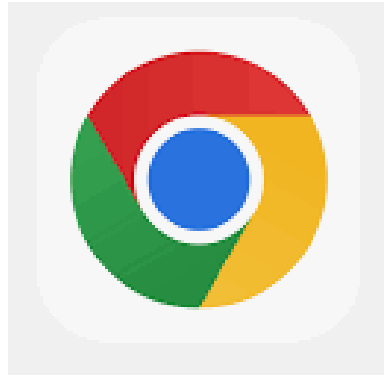
- 링크를 따라가려면 <Ctrl> 키를 누른채 클릭하세요.

교육자료 다운로드



Ctrl+클릭

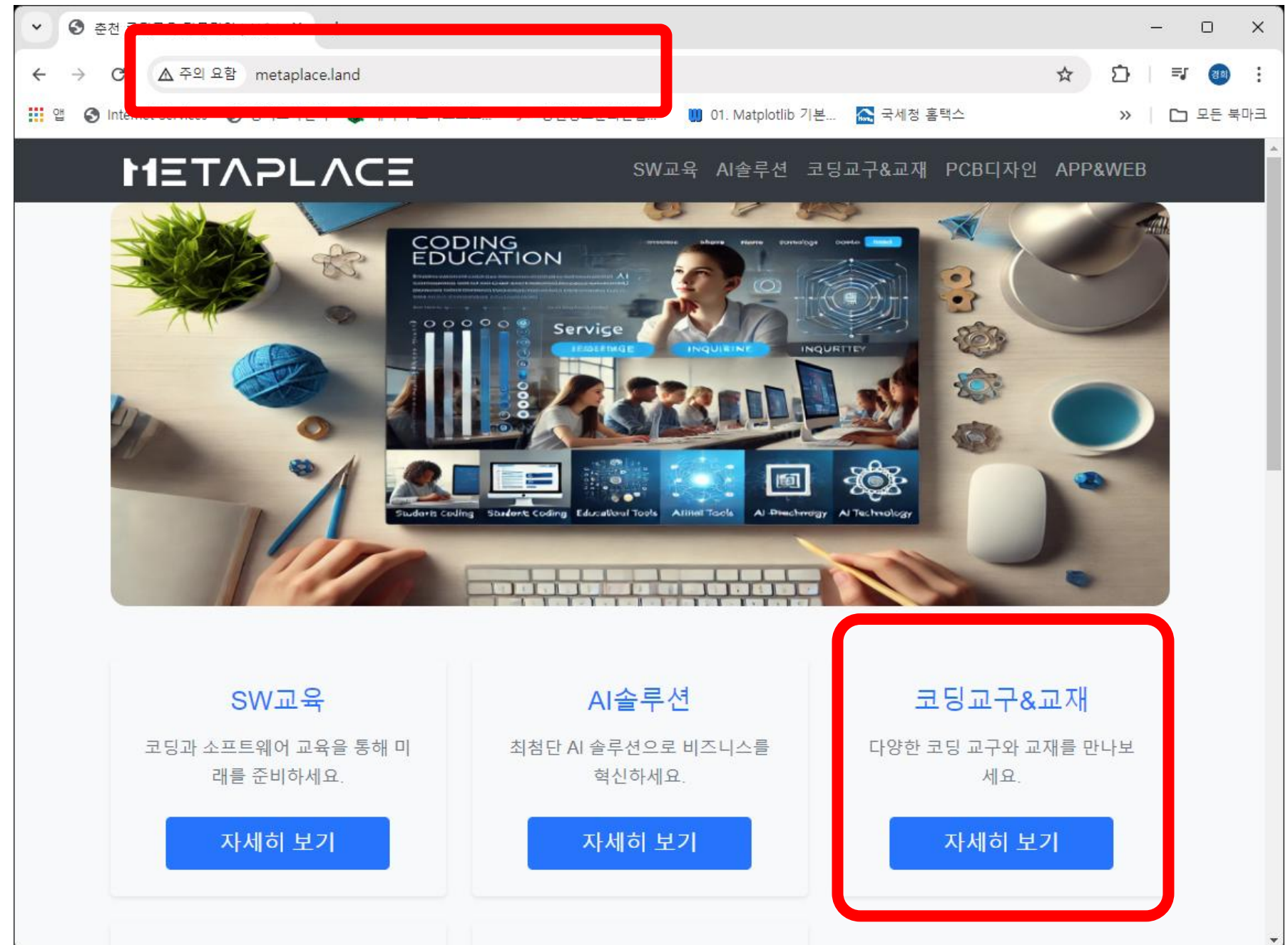
- 메타플레이스를 검색하면 교육자료를 다운로드 할 수 있어요.



① 크롬 브라우저를 선택합니다.

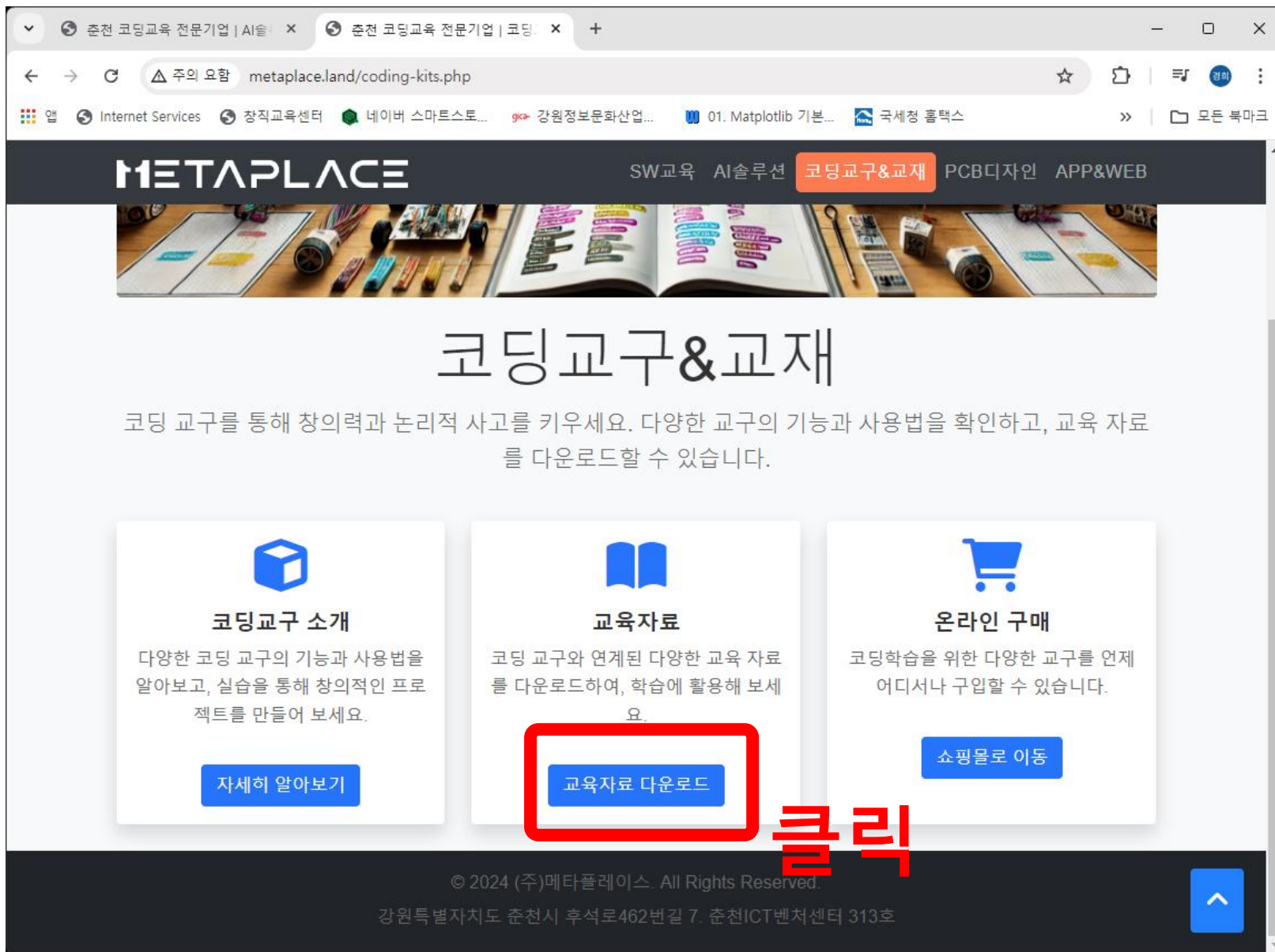
또는

검색창에 : **metaplace.land** 입력합니다.

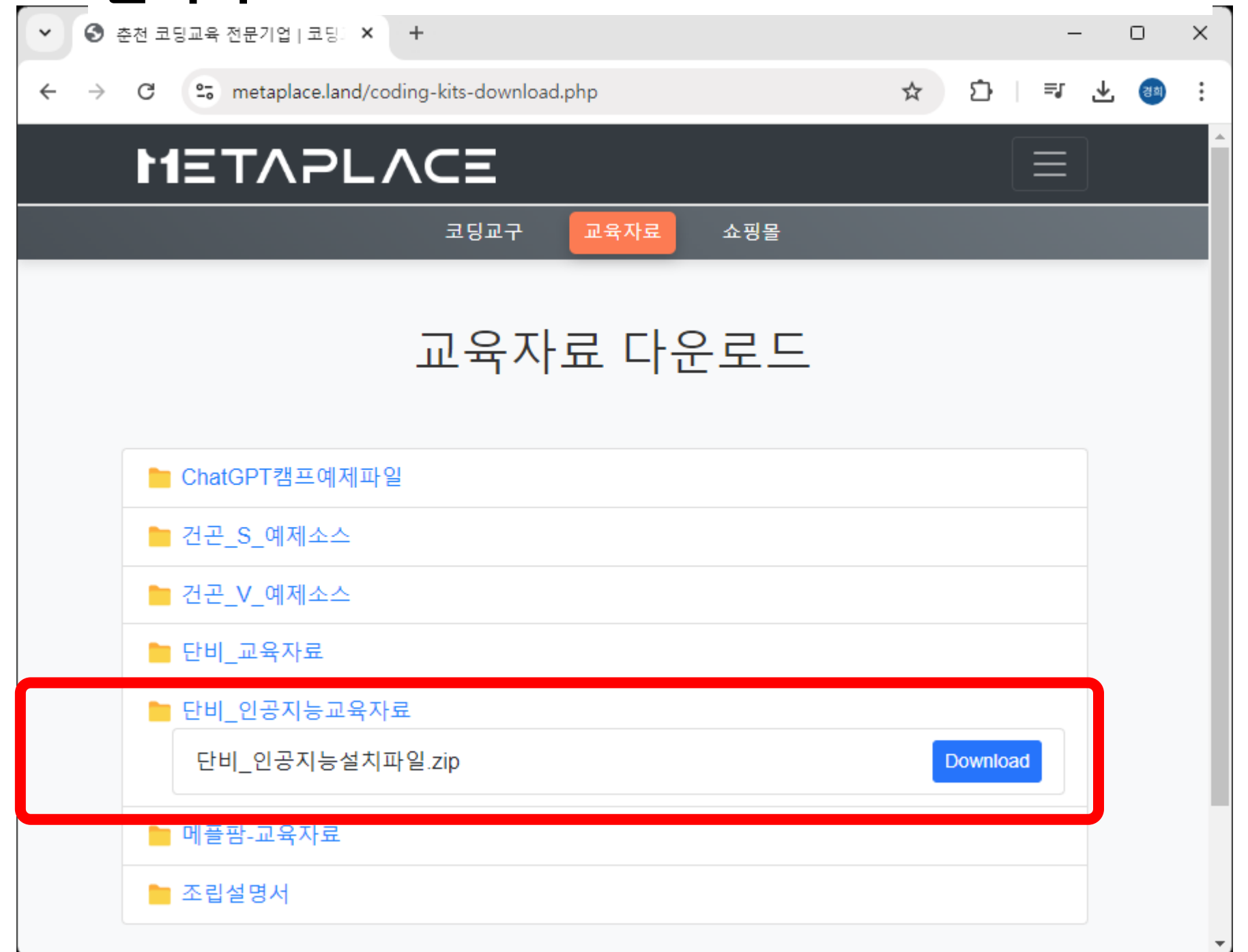


- [코딩교구&교재] - [교육자료] : 필요한 자료를 다운로드 받으세요.

① 교육자료 다운로드를 클릭 합니다.

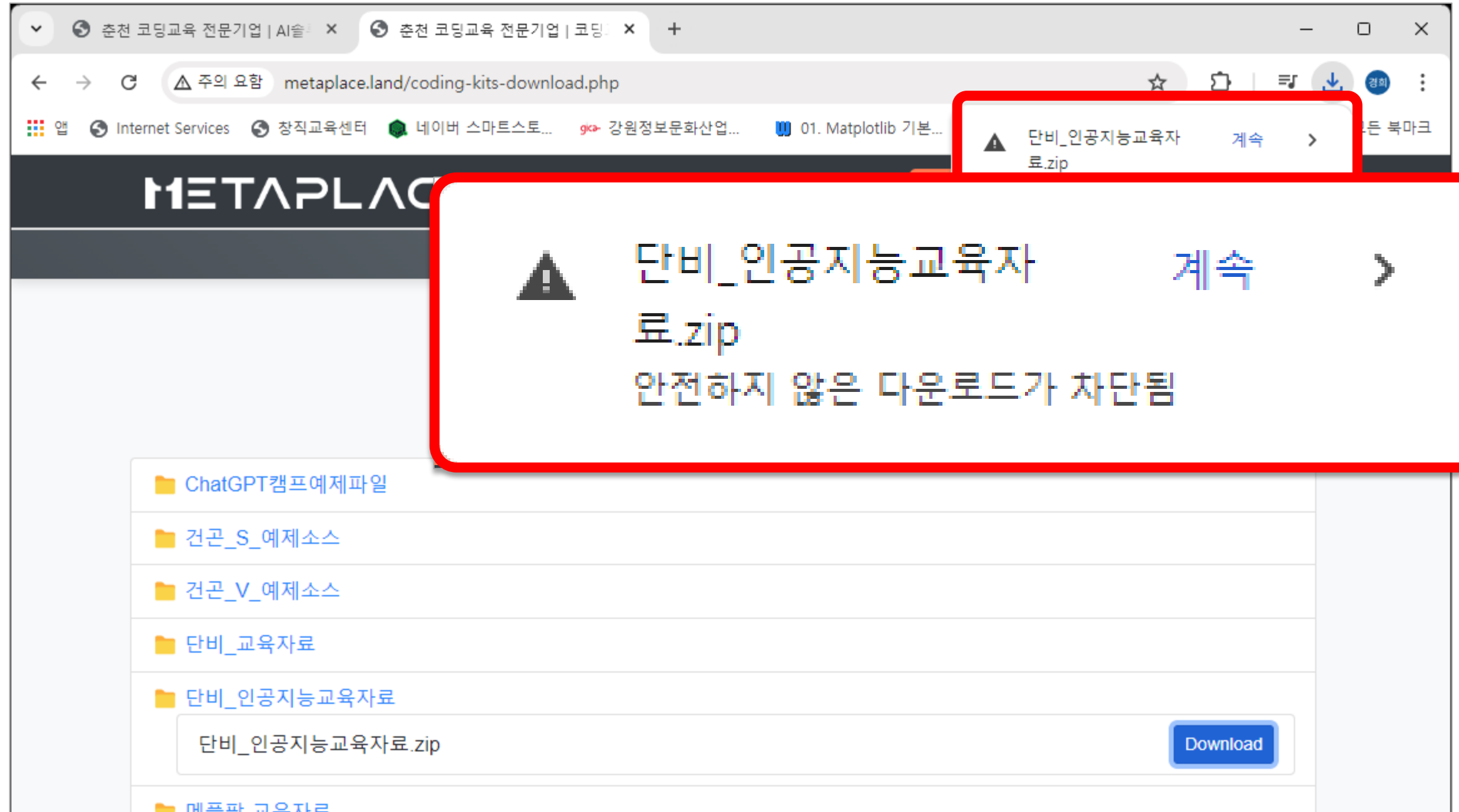


② 필요한 자료를 Download 버튼을 눌러 다운로드 합니다.

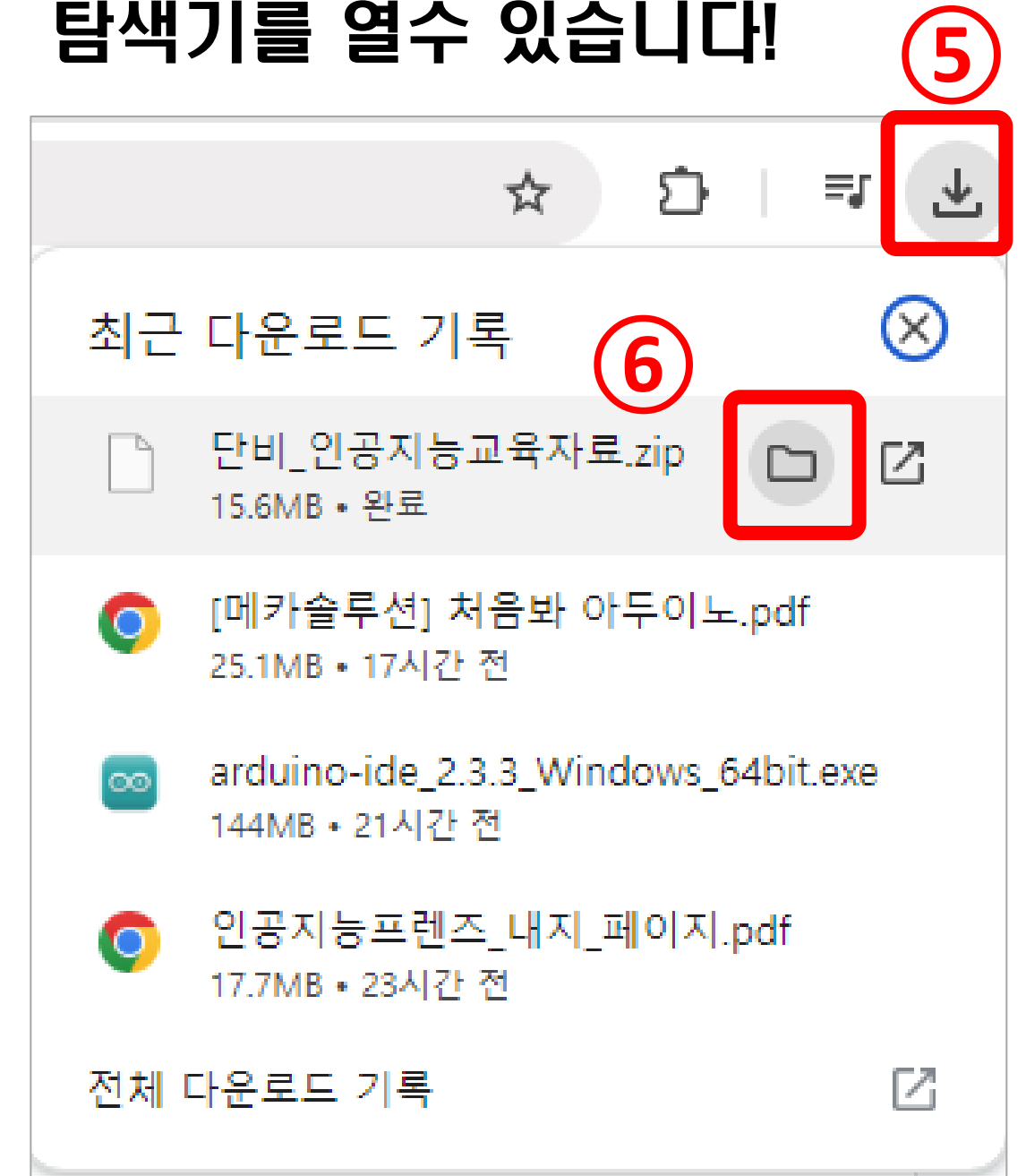


- 다운로드할때 <계속> 을 누르면 다운로드 완료됩니다.

③ 만약, 아래와 같이 “안전하지 않은 다운로드가 차단됨” 이란 메시지가 나오면
⇒ [계속] 을 클릭합니다.



④ 다운로드 버튼을 누르고, 폴더 아이콘을 누르면 편리하게 탐색기를 열수 있습니다!



- 항상 압축을 풀고 사용하셔야 합니다.

① 탐색기가 열리면

②

③ 마우스오른쪽버튼 => 압축풀기

④

압축이 풀려진 폴더를 반드시 사용하세요!

python_pycharm

Step5_Auto_Drive

arduino-ide_2.3.3_Windows_64bit

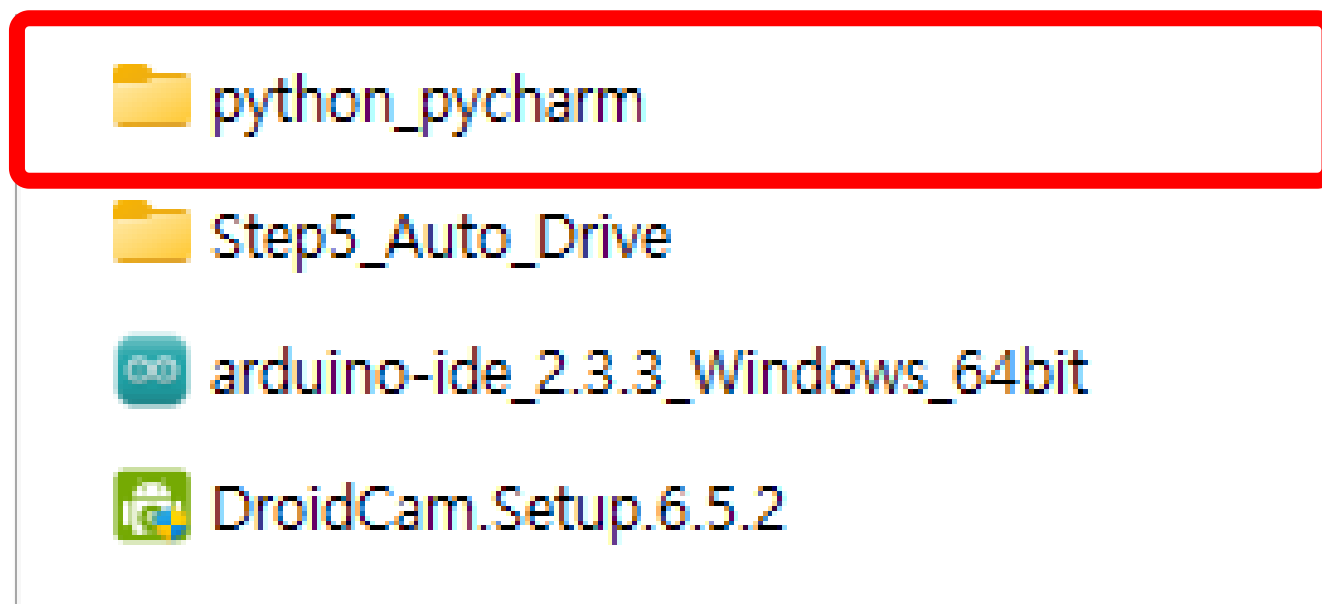
DroidCam.Setup.6.5.2

파이썬 설치를 해보아요.
파이썬은 3.8.10 버전을 사용합니다.

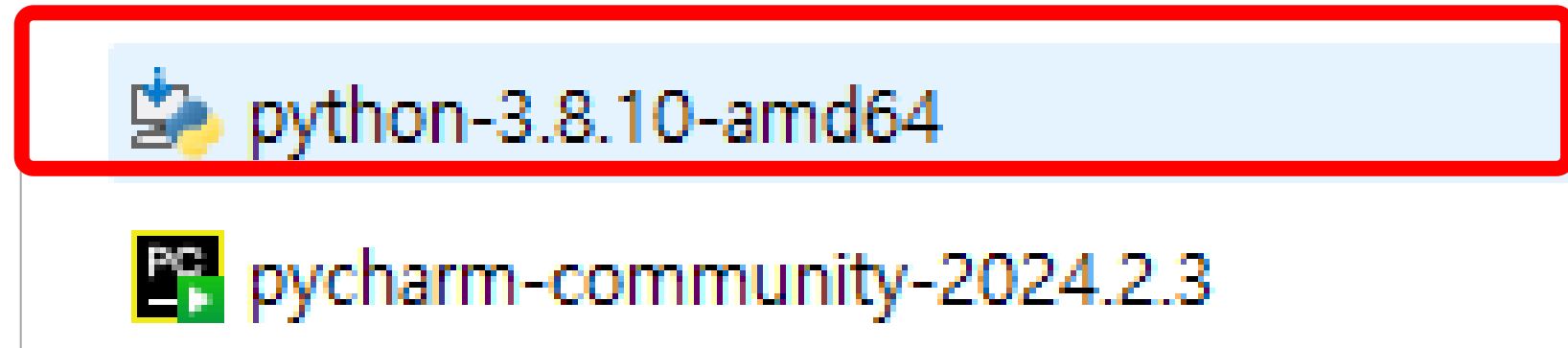


- python_pycharm 폴더로 이동하세요.

① 다운로드 폴더에서 폴더 클릭

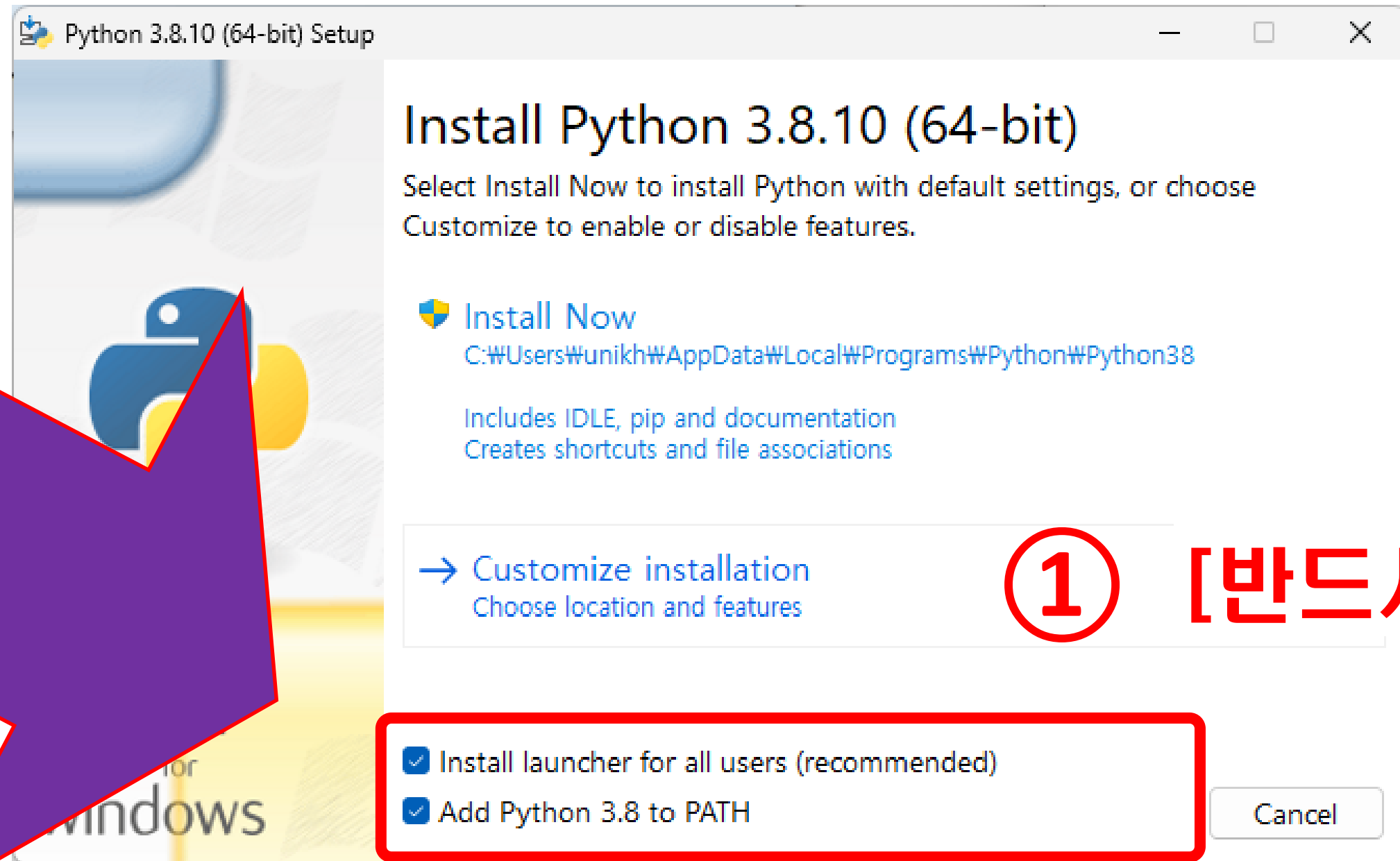


② 파이썬은 **3.8.10 버전으로** 사용합니다.
- 인공지능 학습을 위해 버전을 꼭 지켜주세요.



- 단계별로 설치를 해보겠습니다.

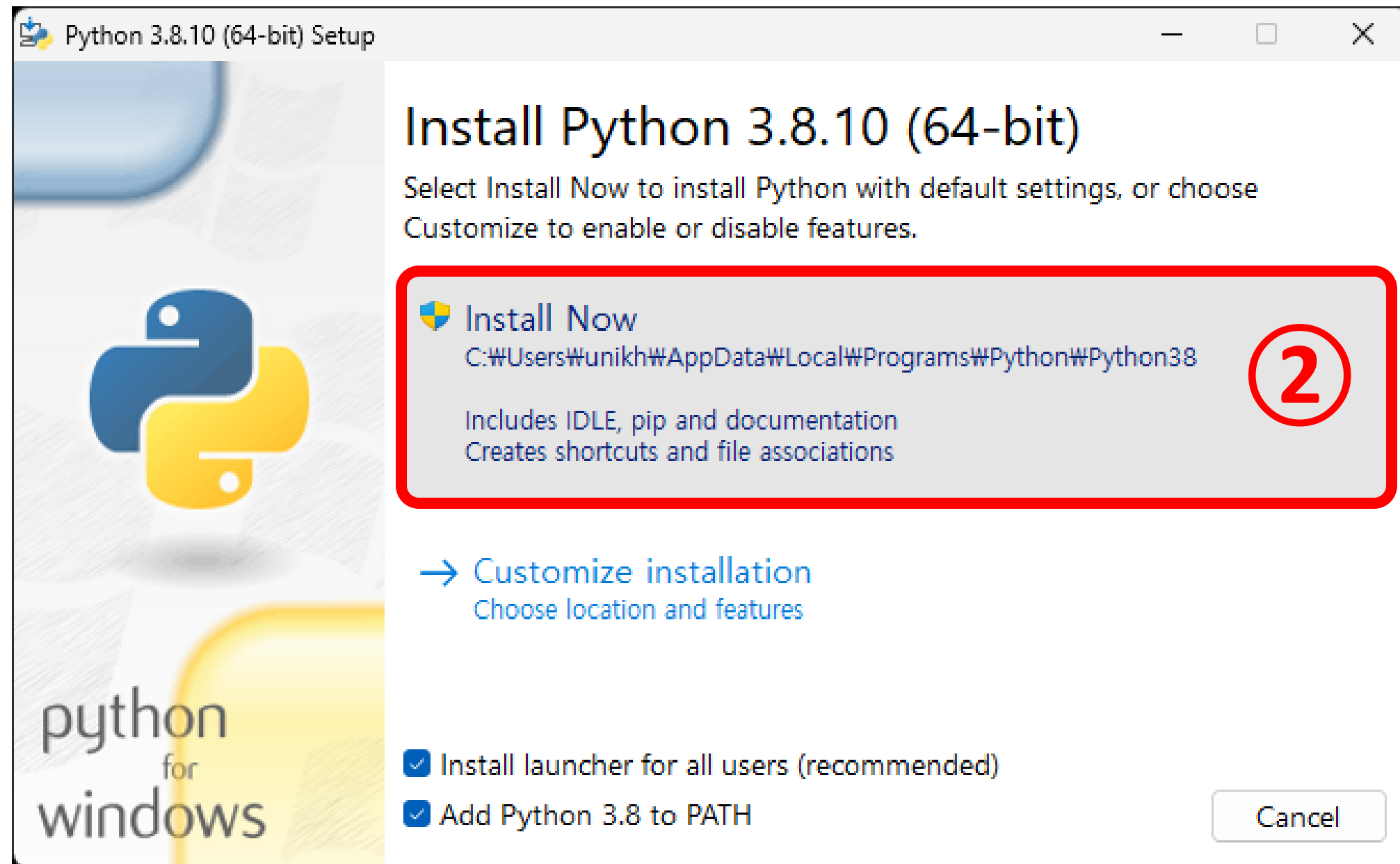
[주의!!!] 단계별로 **옵션을 잘 체크해서 천천히** 따라하세요.
차근차근 체크하지 않으면 파이썬 실행시 어려움이 있어요.



① [반드시 체크!]

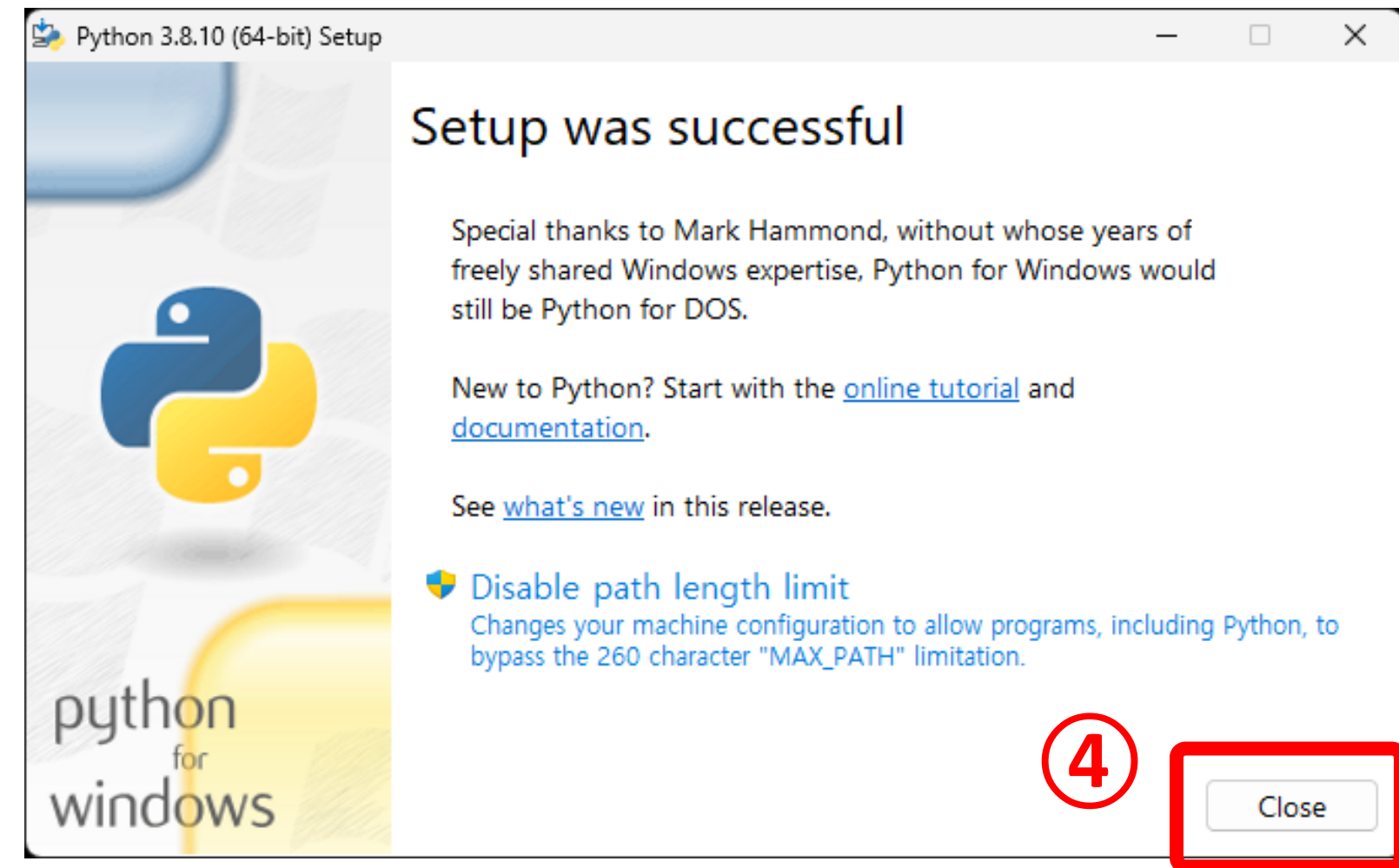
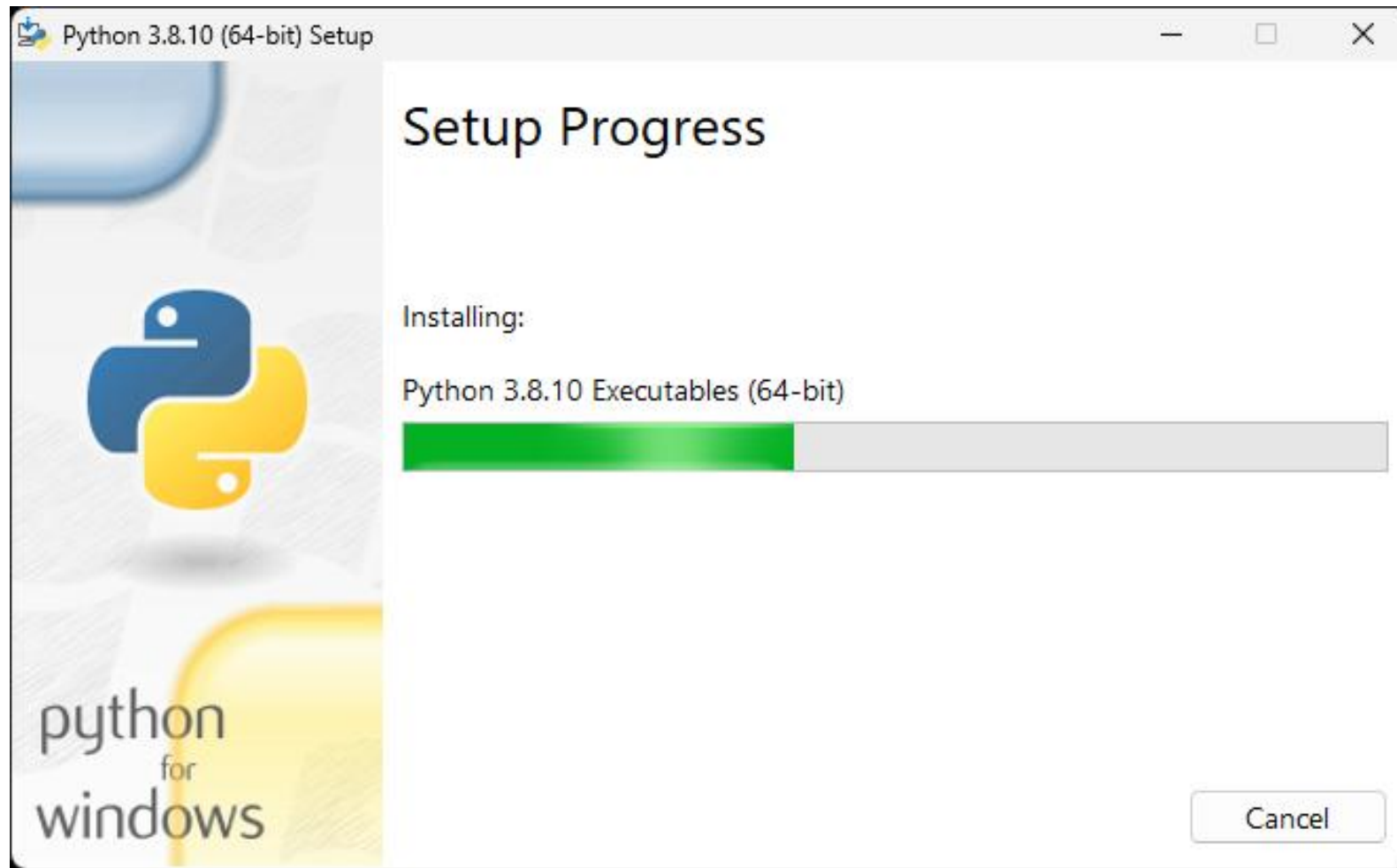
- 단계별로 설치를 해보겠습니다.

[주의!!!] 단계별로 **옵션을 잘 체크해서 천천히** 따라하세요.
차근차근 체크하지 않으면 파이썬 실행시 어려움이 있어요.



- 단계별로 설치를 해보겠습니다.

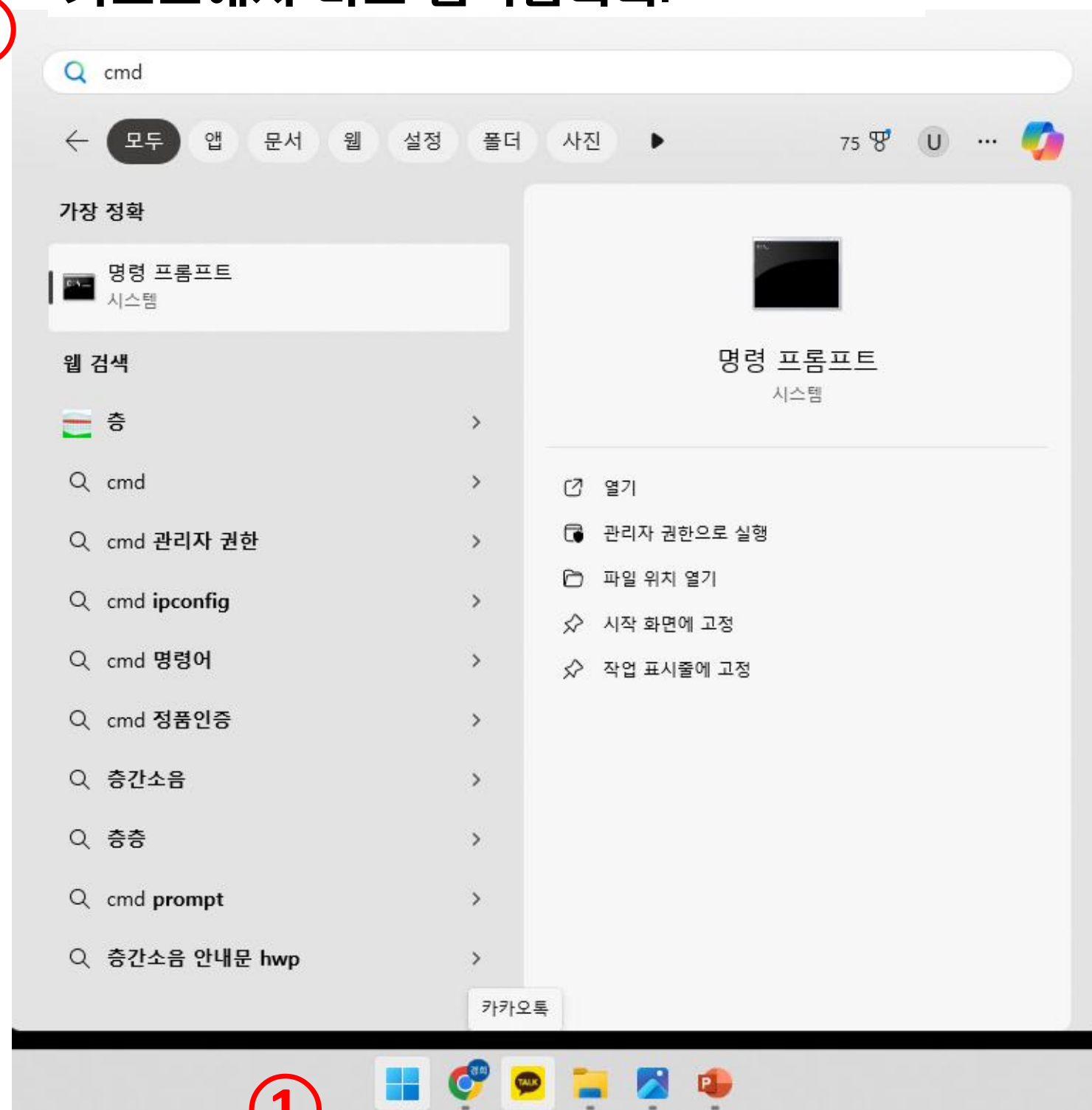
③ 설치 진행을 기다립니다.



설치가 완료되면,
클릭합니다.

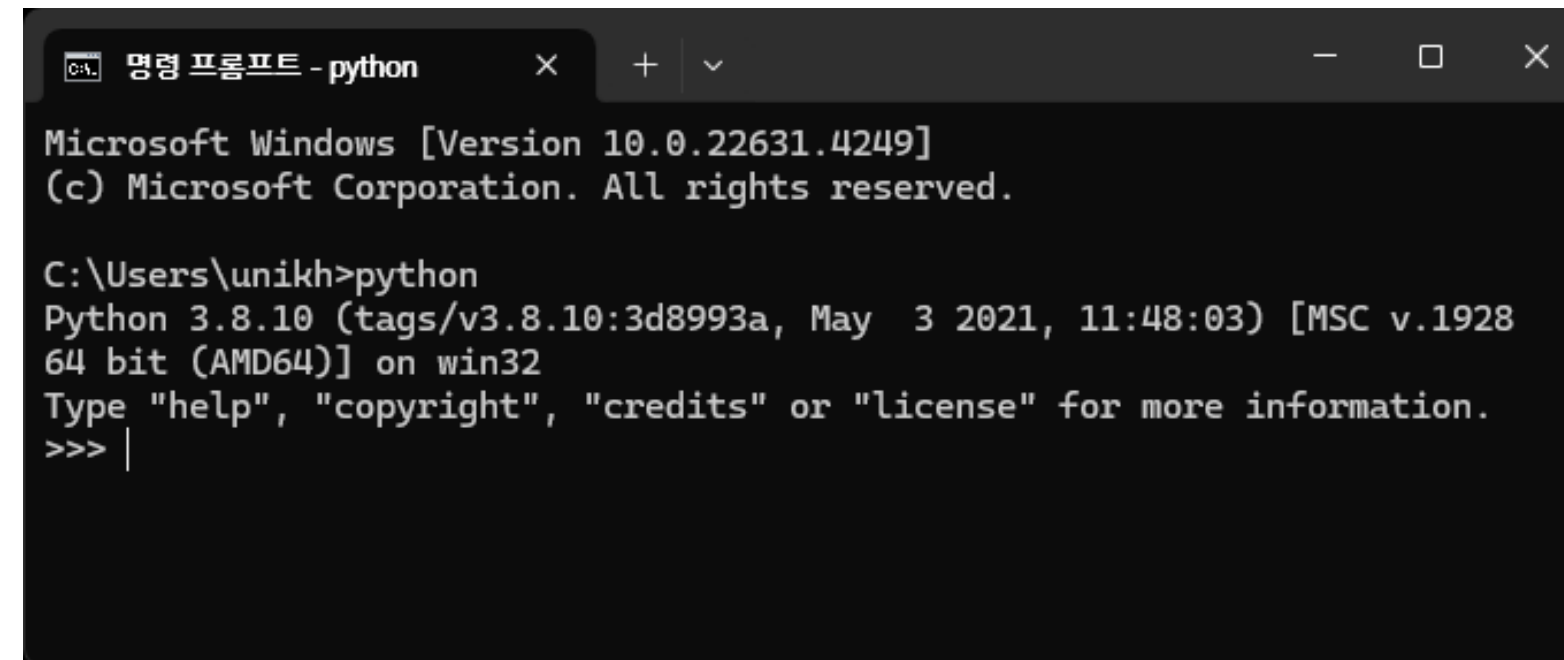
- 파이썬을 실행해 볼게요.

② 키보드에서 바로 입력합니다.

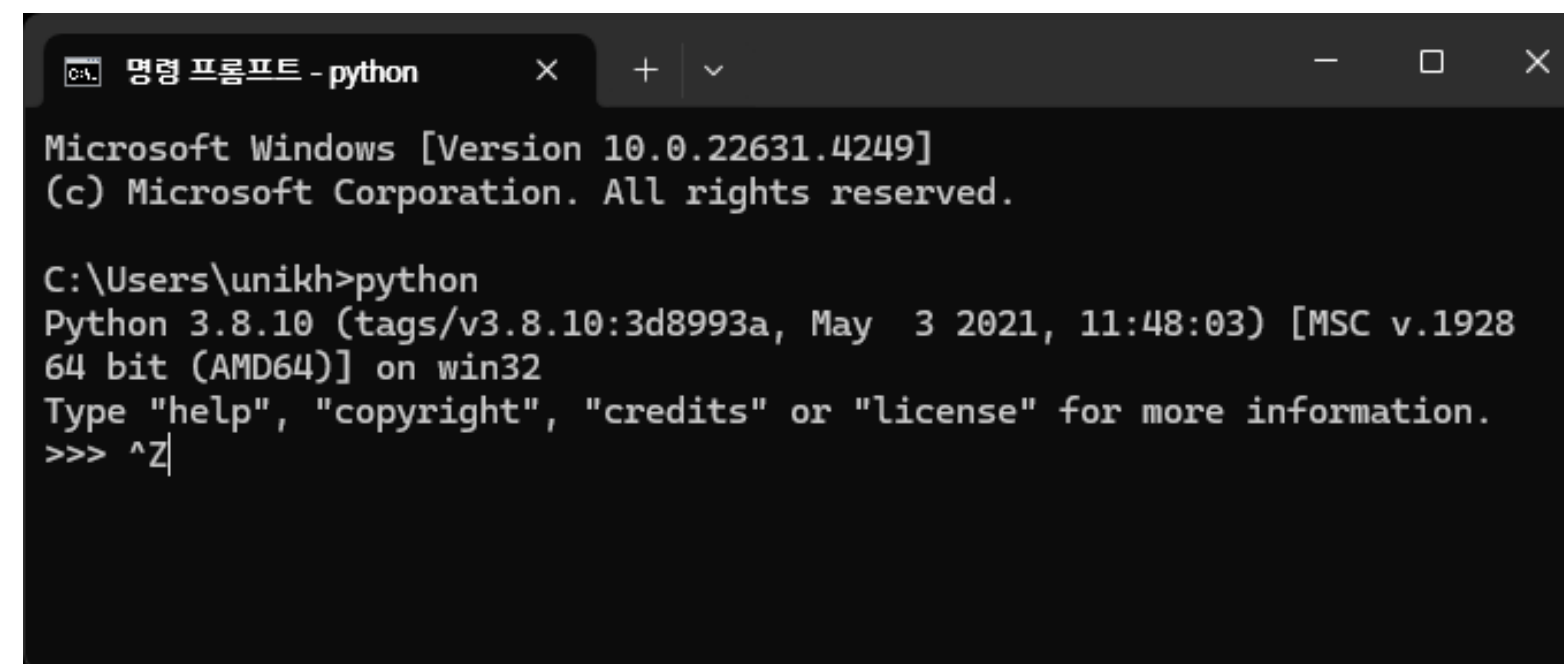


① 위도우창을 클릭후, "cmd"라고 입력하면 됩니다.

③ python (엔터) 입력해봅니다.



④ 파이썬 3.8.10 설치되었습니다.

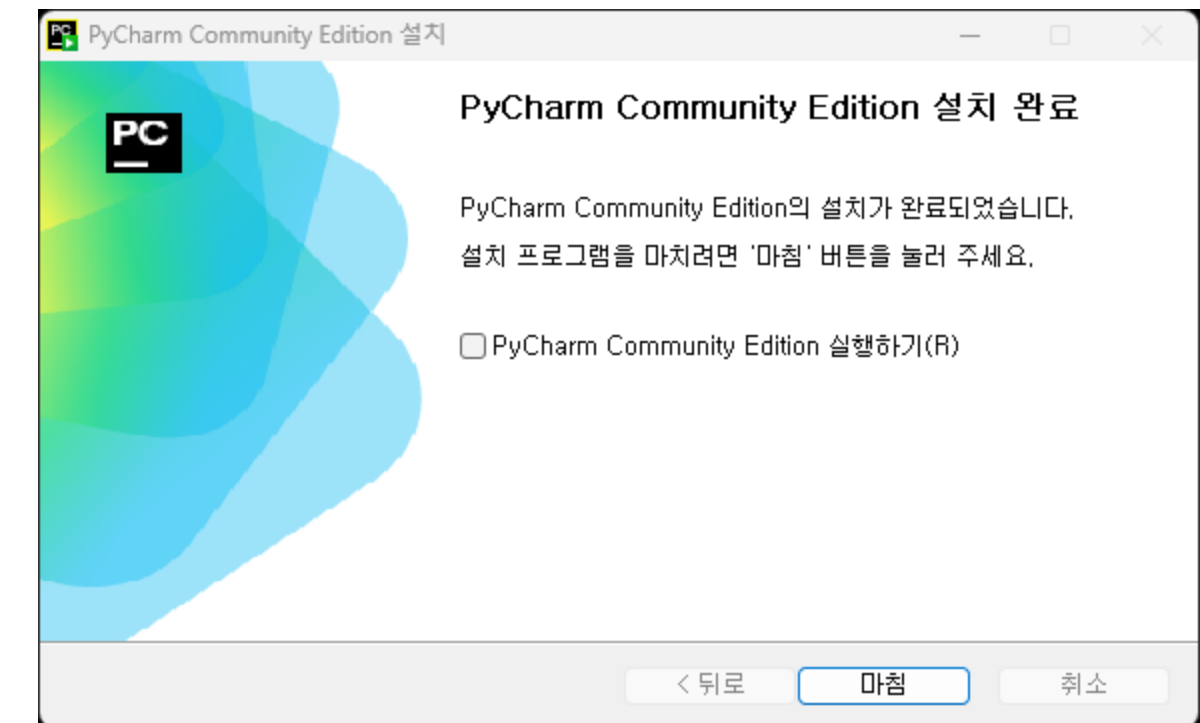
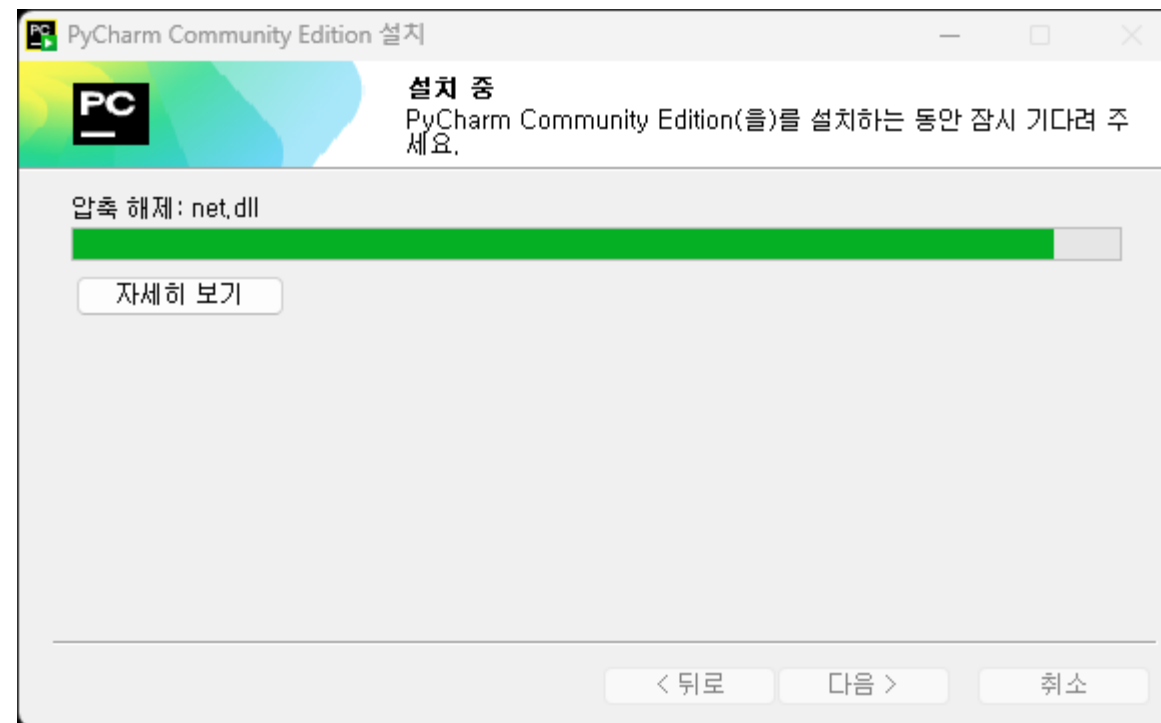
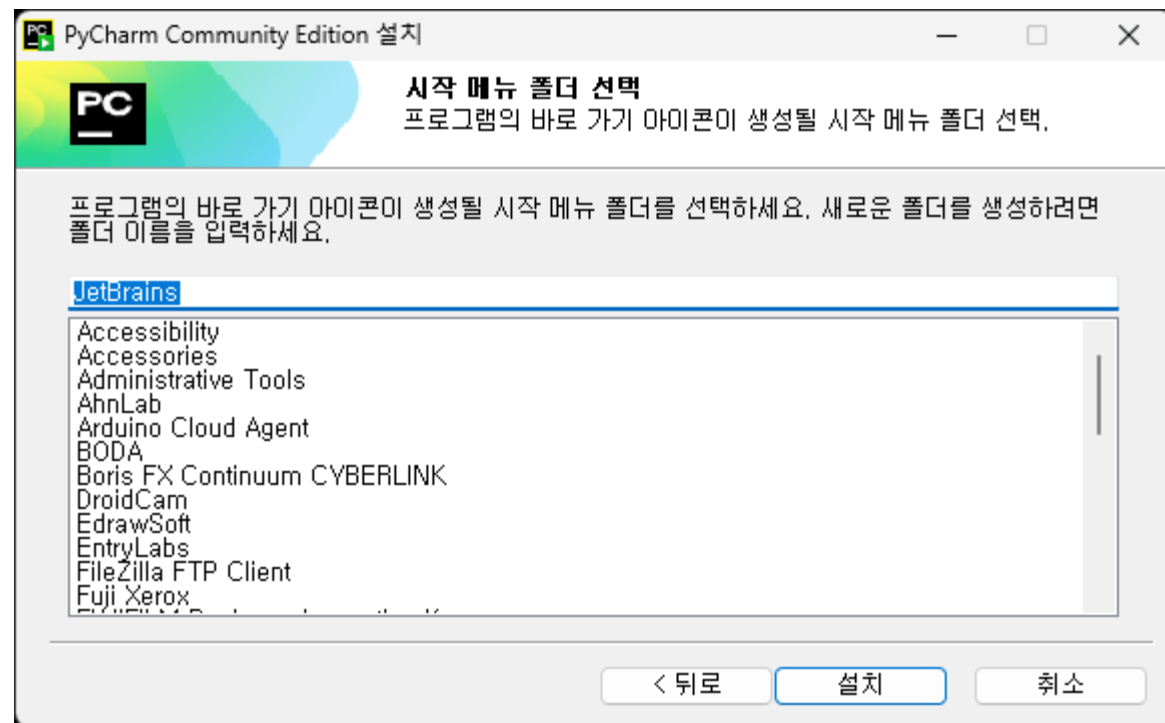
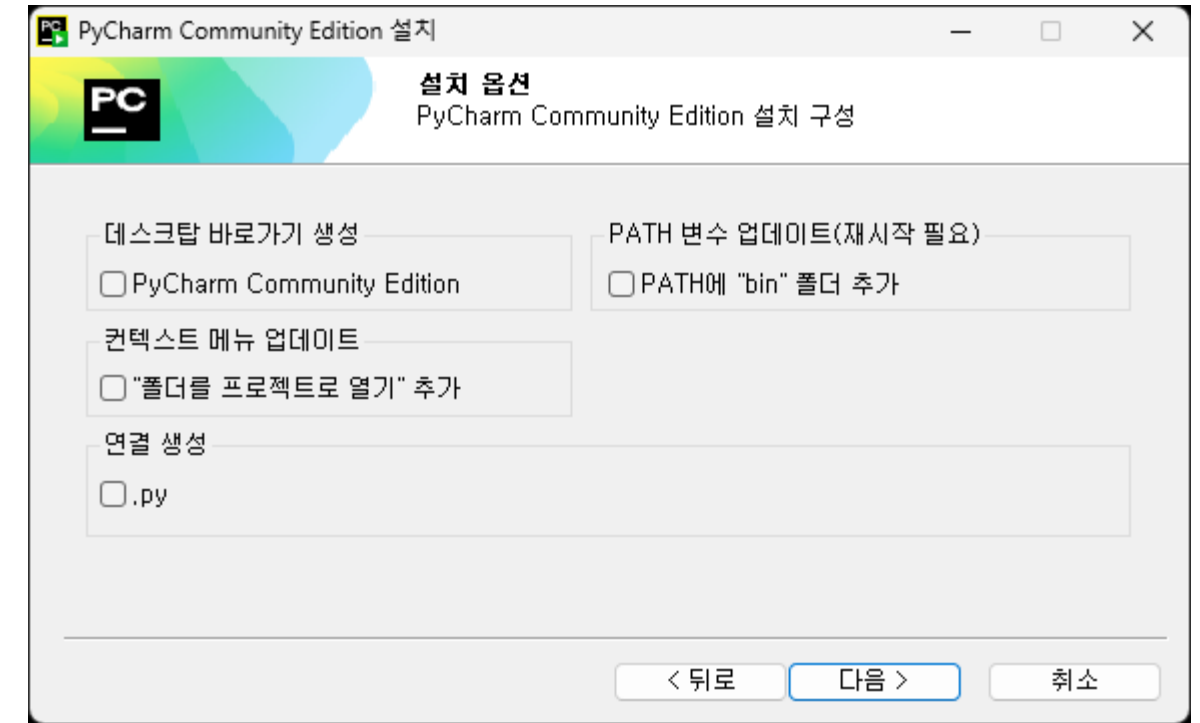
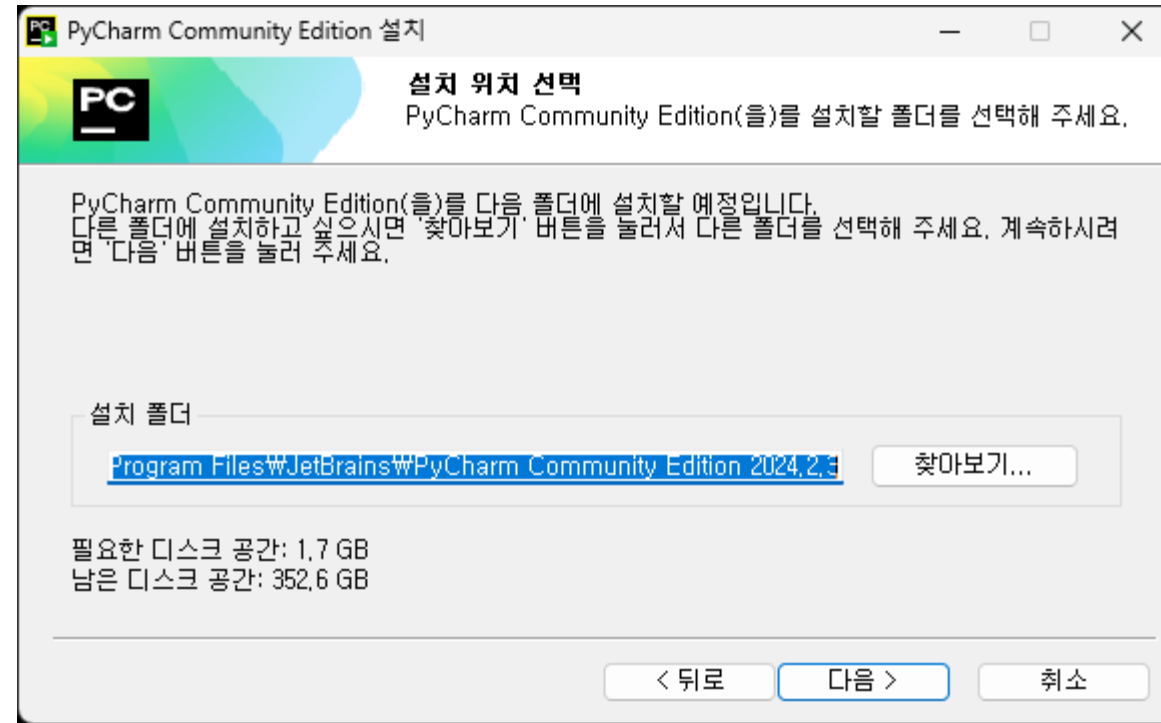
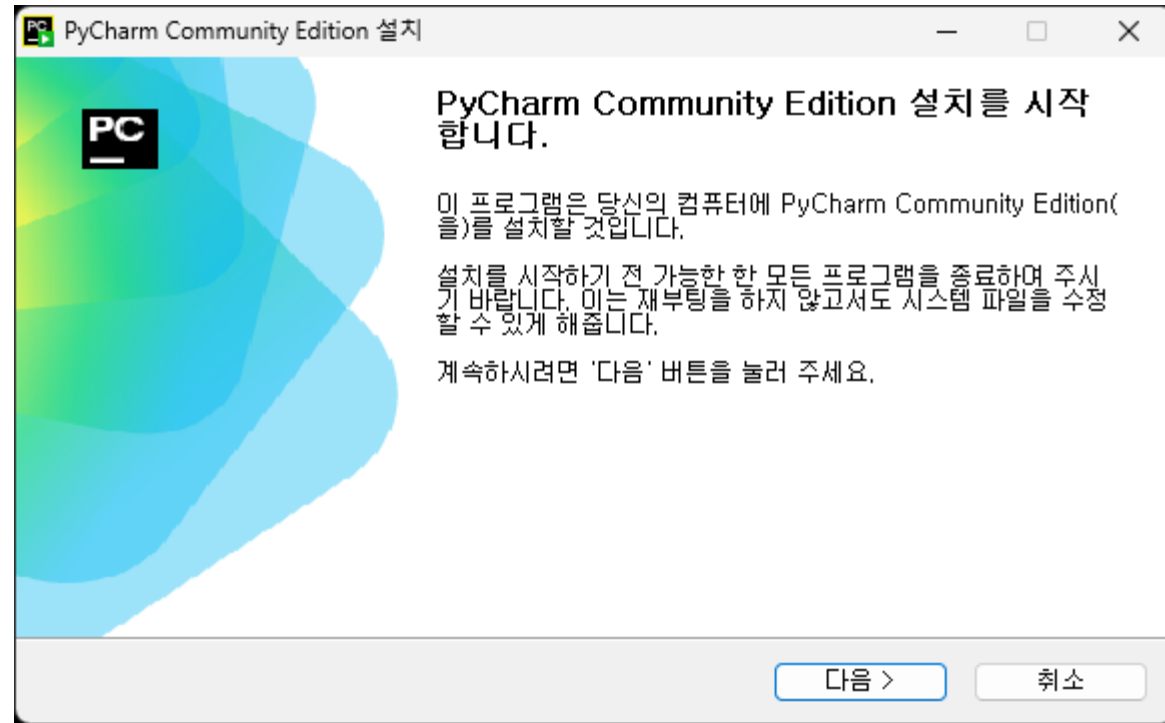


⑤ Ctrl + Z 을 누르면 종료

파이참 설치를 해보아요.
파이참은 최신버전을 사용하셔도 됩니다.

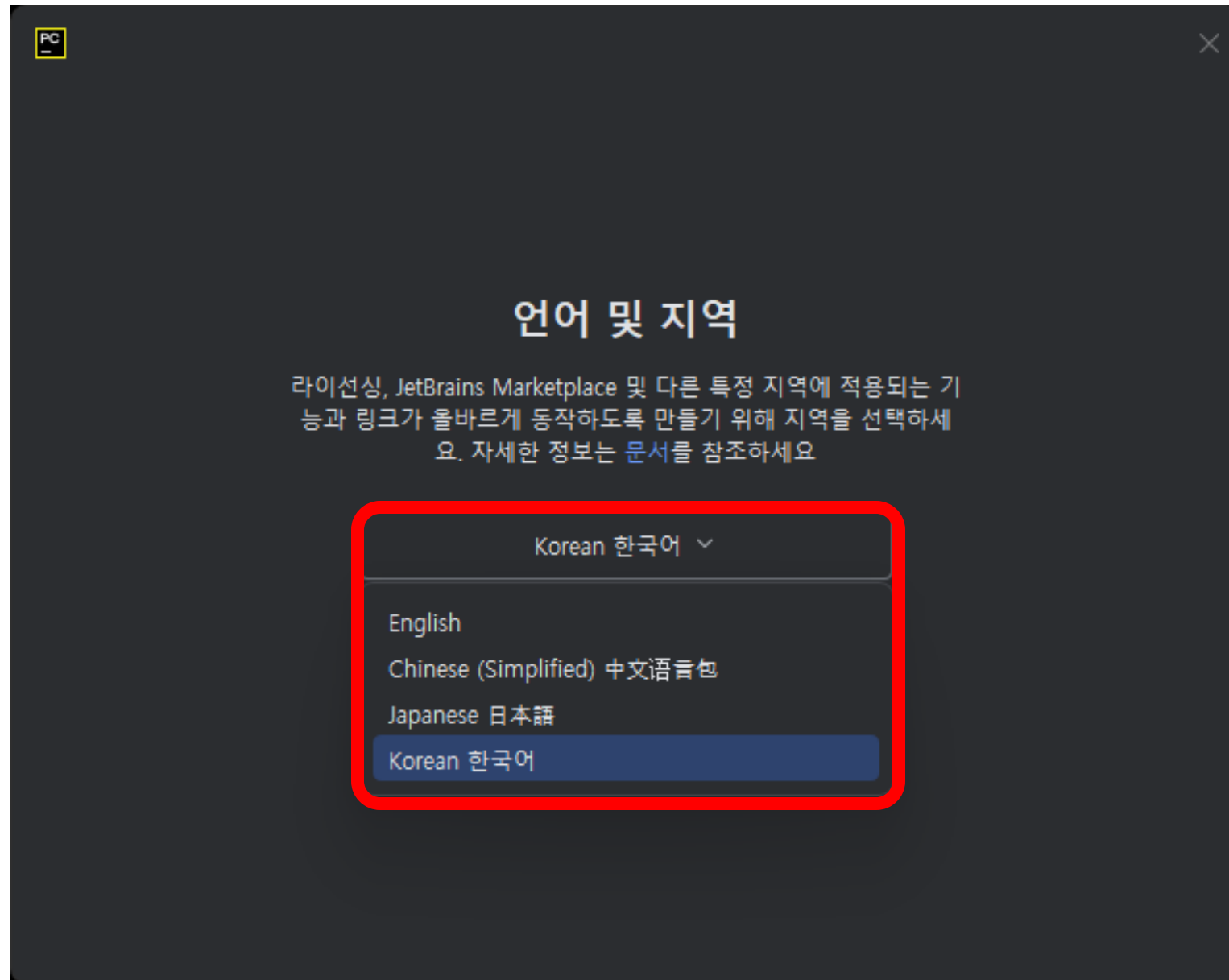


- 단계별로 설치를 해보겠습니다.

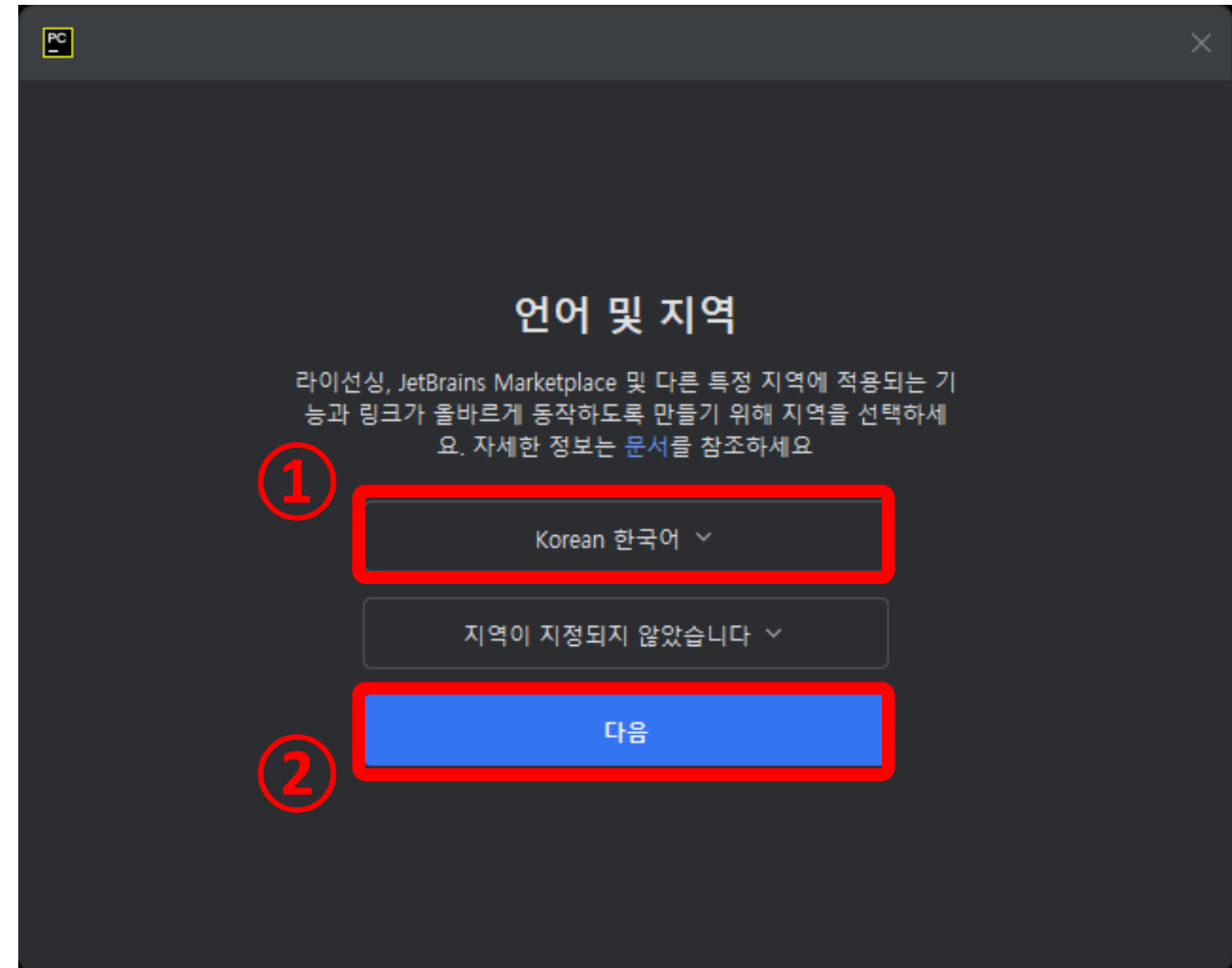


- 단계별로 설치를 해보겠습니다.

- 영어버전/한국어버전을 사용가능

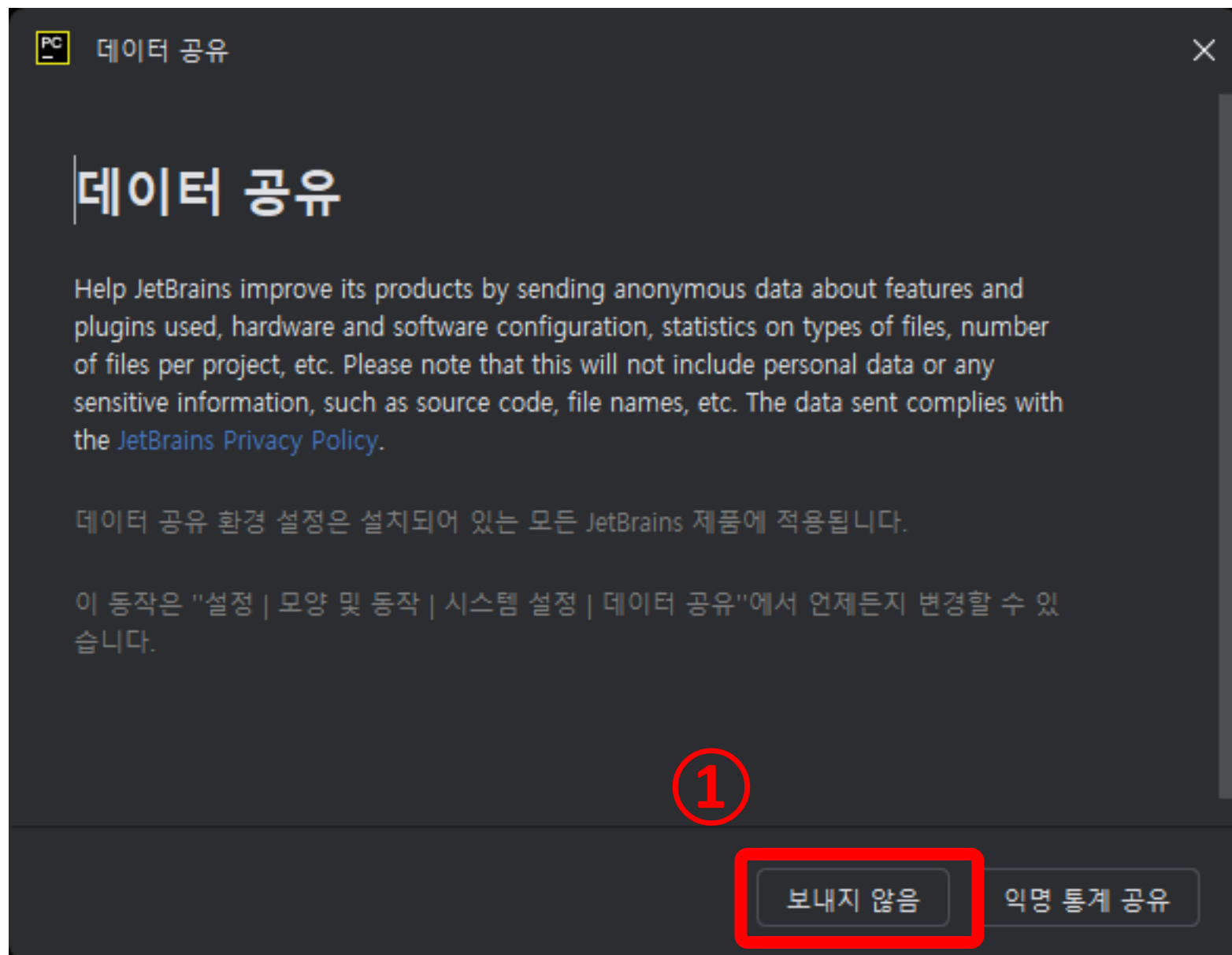


- 한국어를 선택한 상태에서 설명합니다!

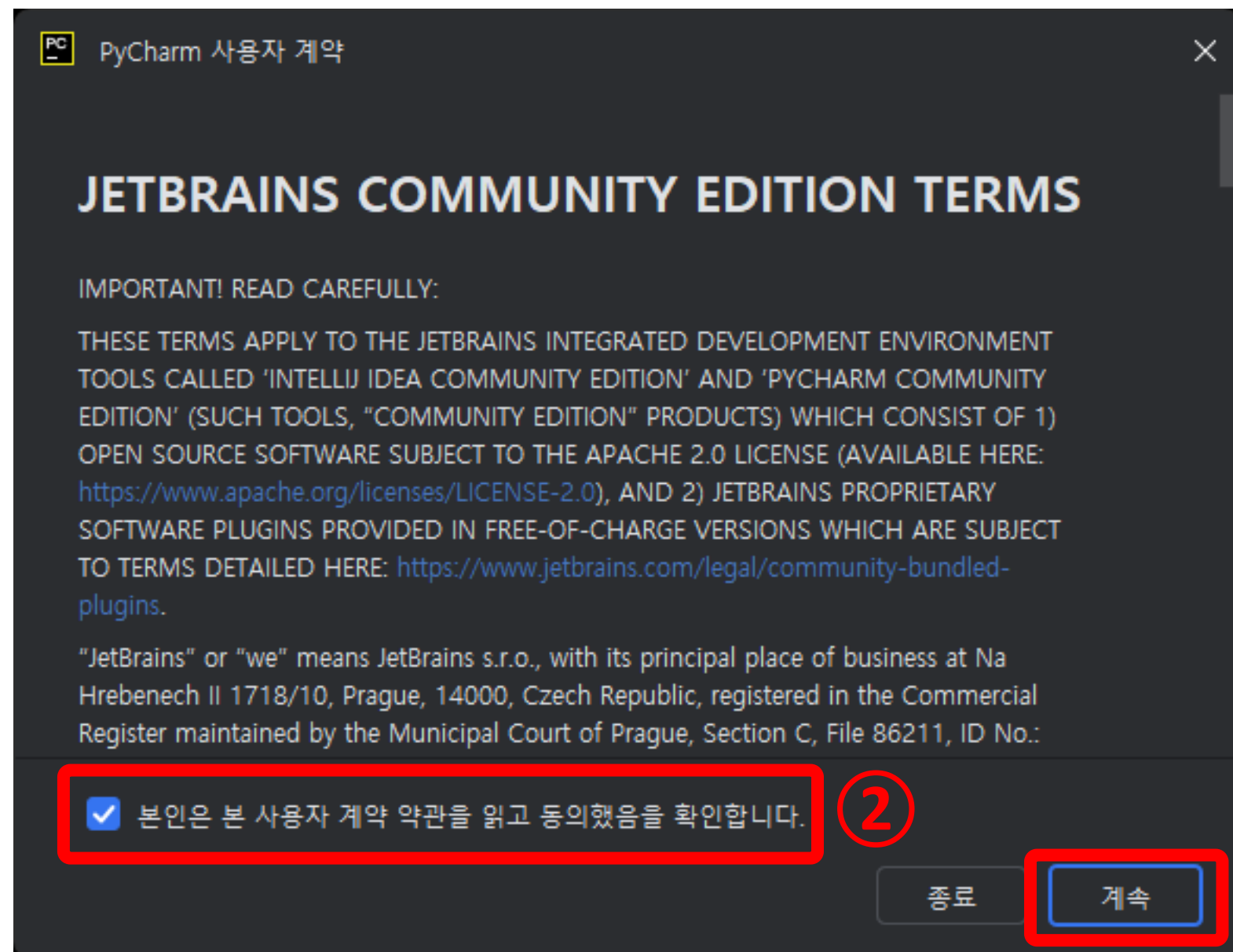


- 단계별로 설치를 해보겠습니다.

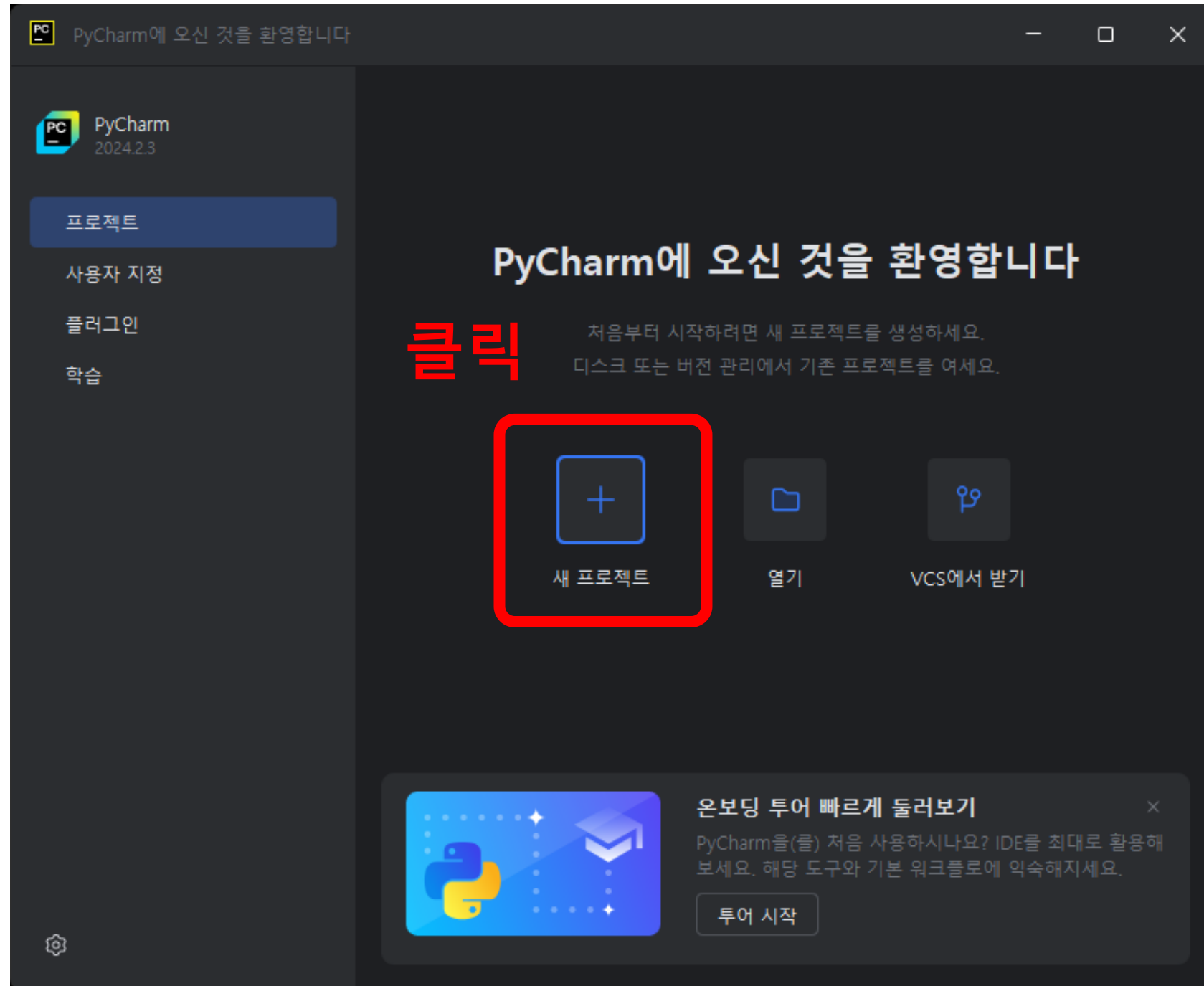
- 데이터 공유는 “보내지않음”



- 계약약관을 체크하고, <계속>합니다.

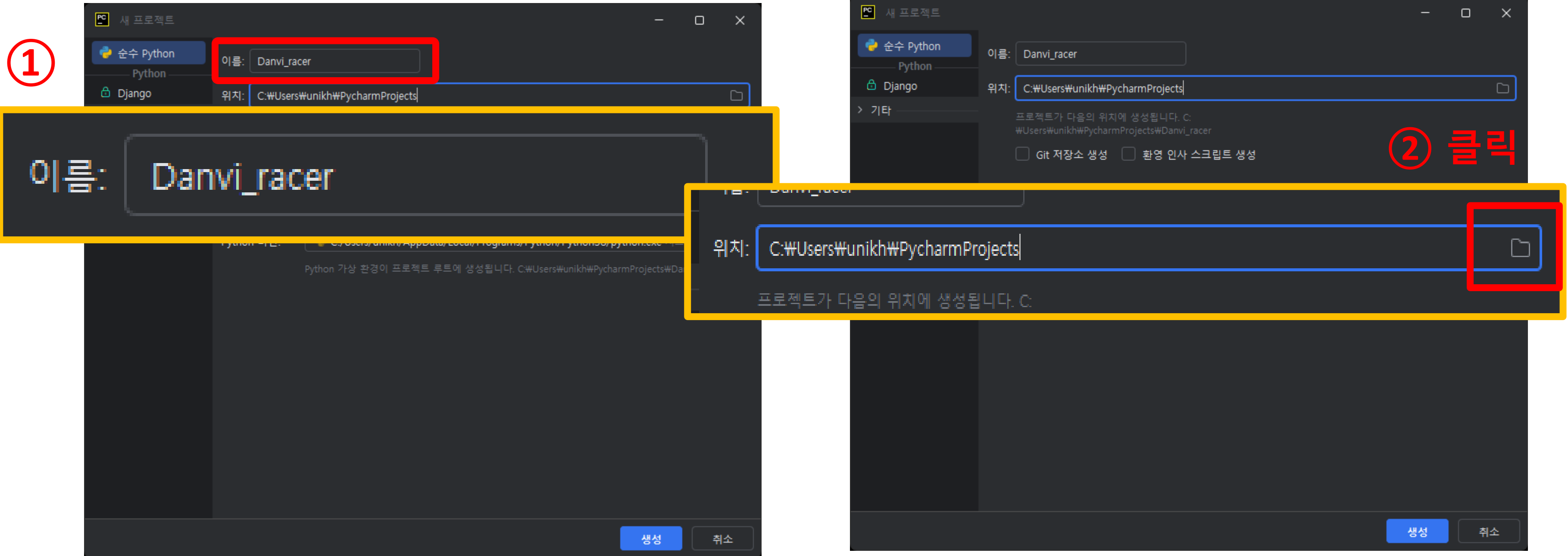


- 설치완료 후 <새 프로젝트>를 생성합니다.



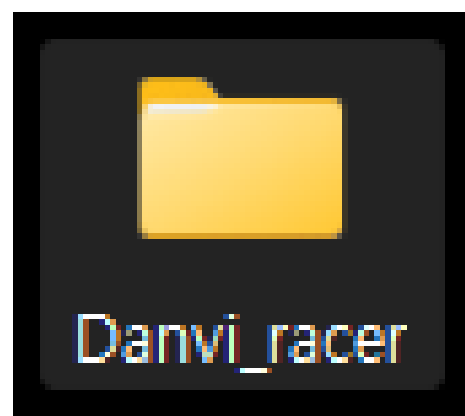
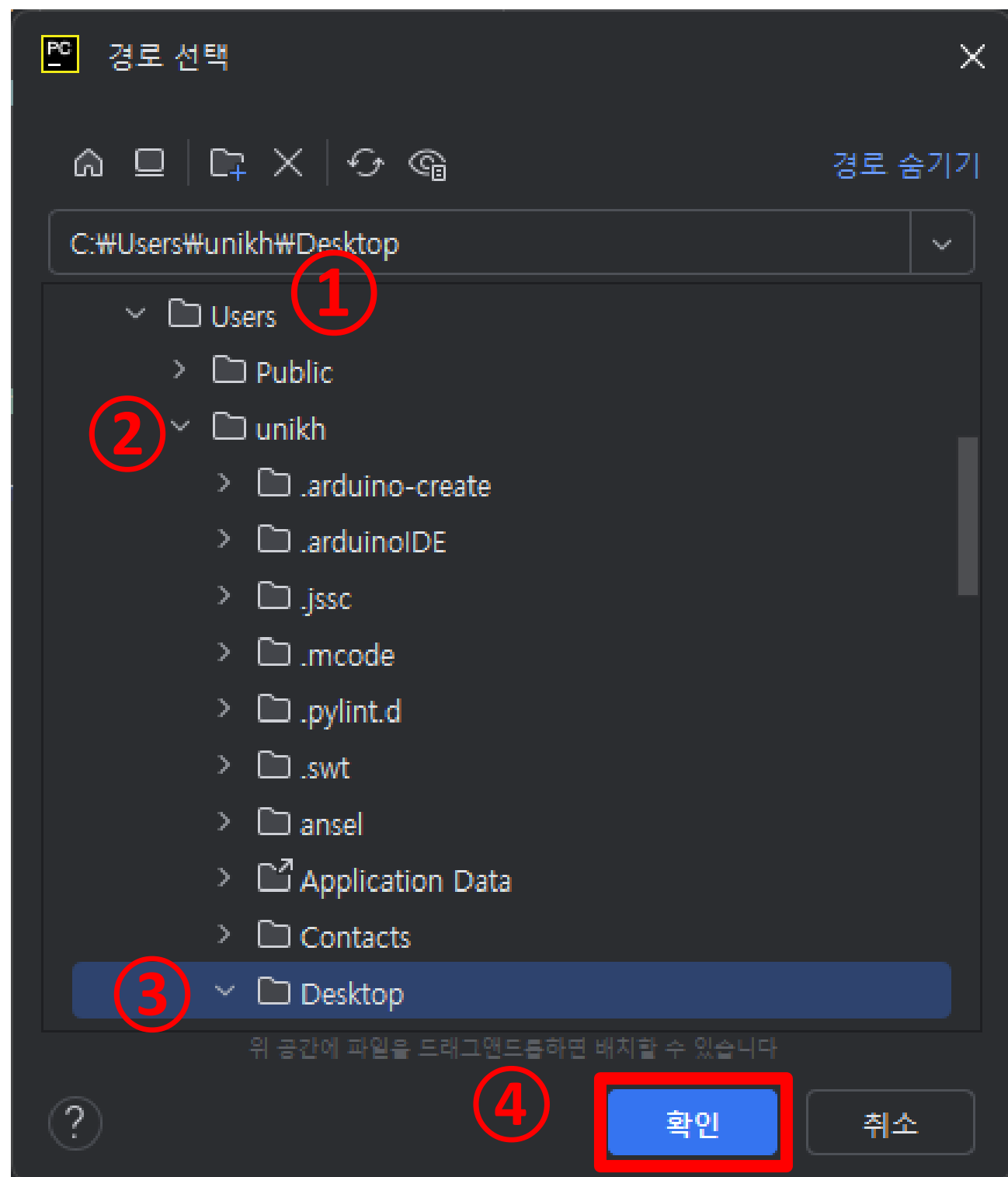
- 새로운 프로젝트를 생성합니다.

- 프로젝트 이름과 위치를 정확히 선택합니다.

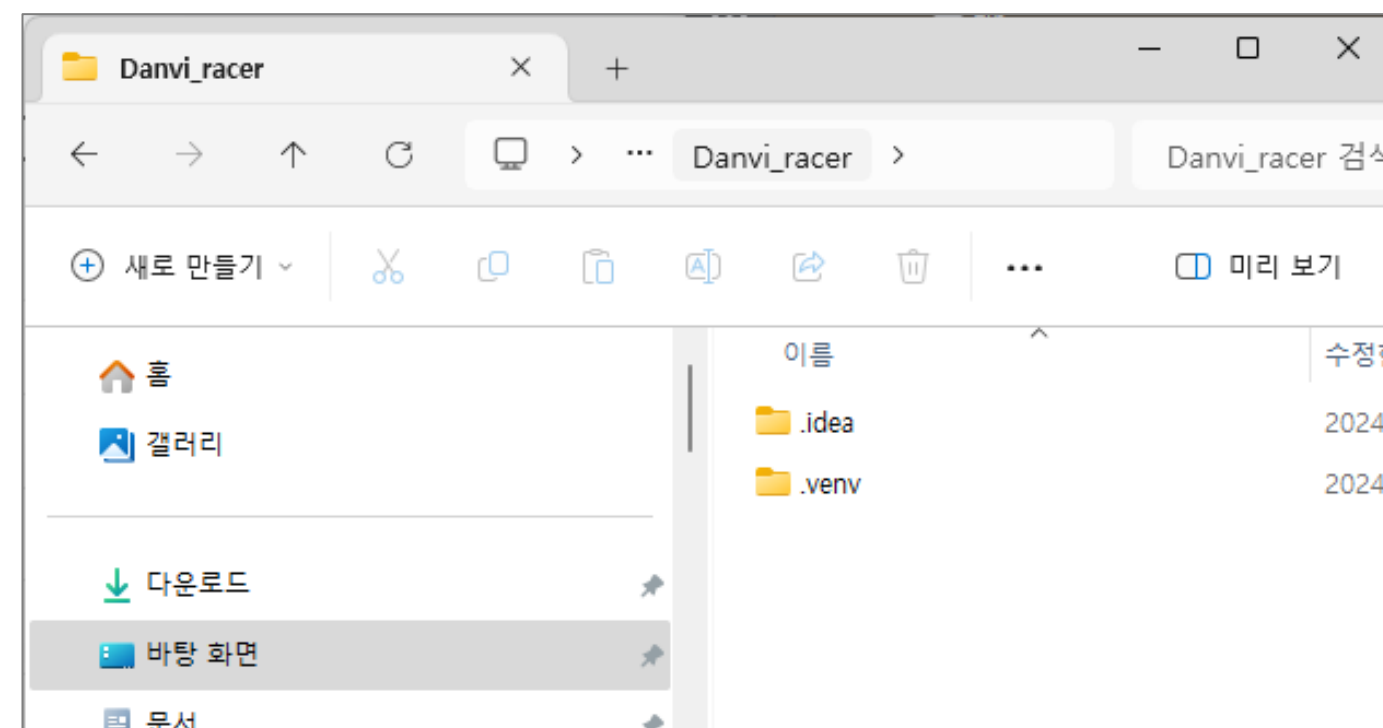


- 새로운 프로젝트를 생성합니다.

- 프로젝트 위치는 <바탕화면>에 생성합니다.

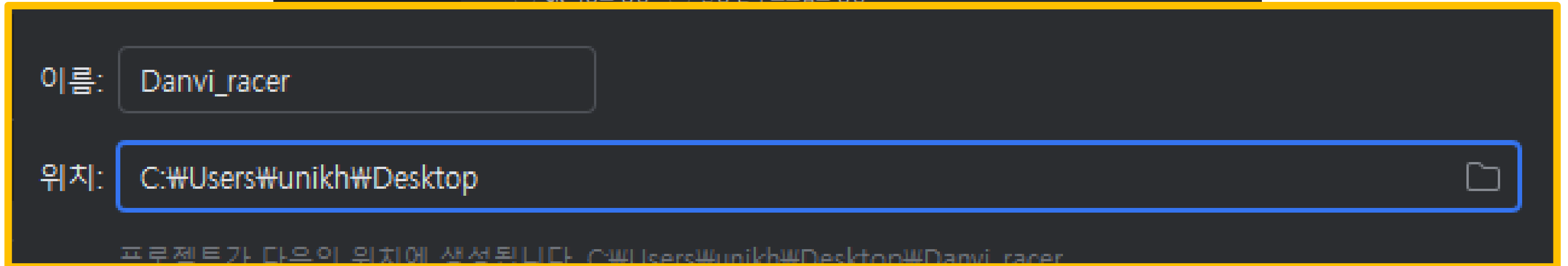
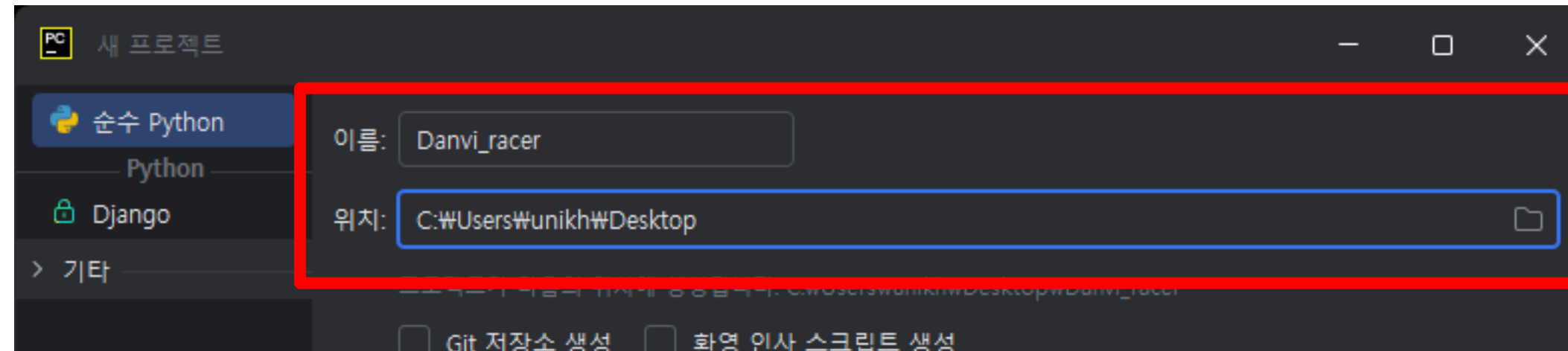


⑤ 바탕화면에 폴더 생성확인!



⑥ Danvi_racer폴더에 두개의 폴더확인

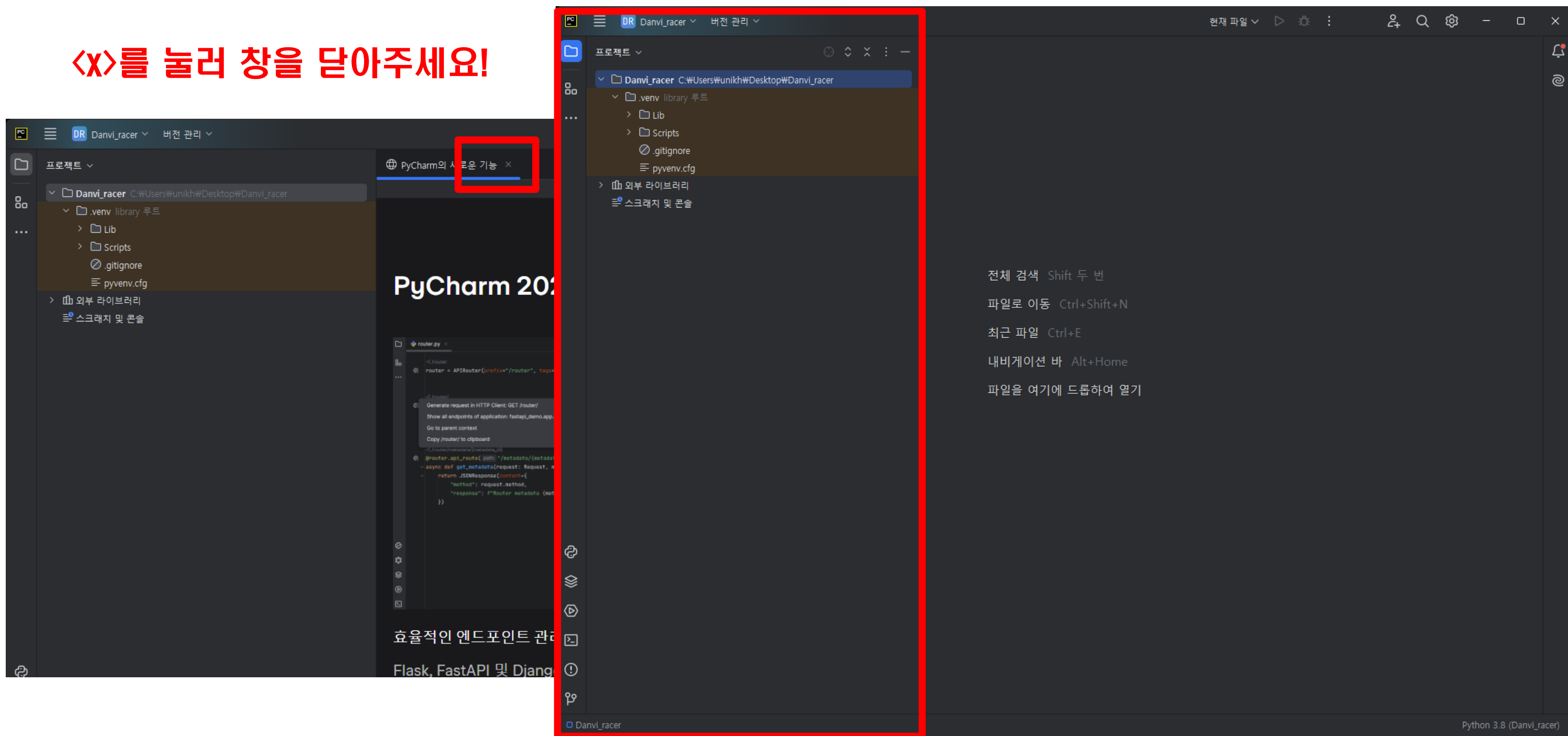
- 프로젝트 이름: Danvi_racer/ 위치를 확인해 주세요!



- 이름과 위치를 다시한번 확인 하세요!

- 단비 자율주행 프로젝트를 시작합니다!

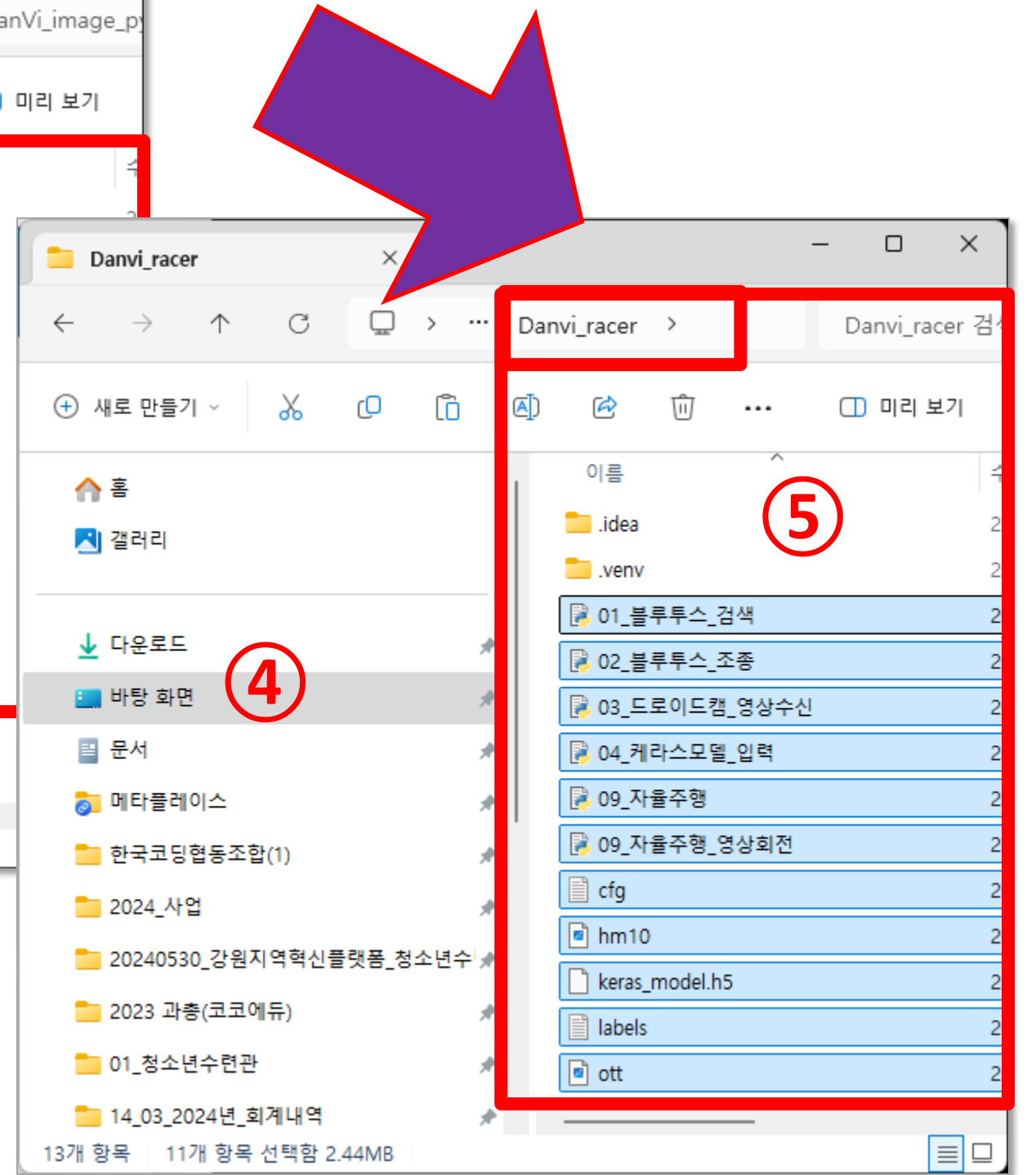
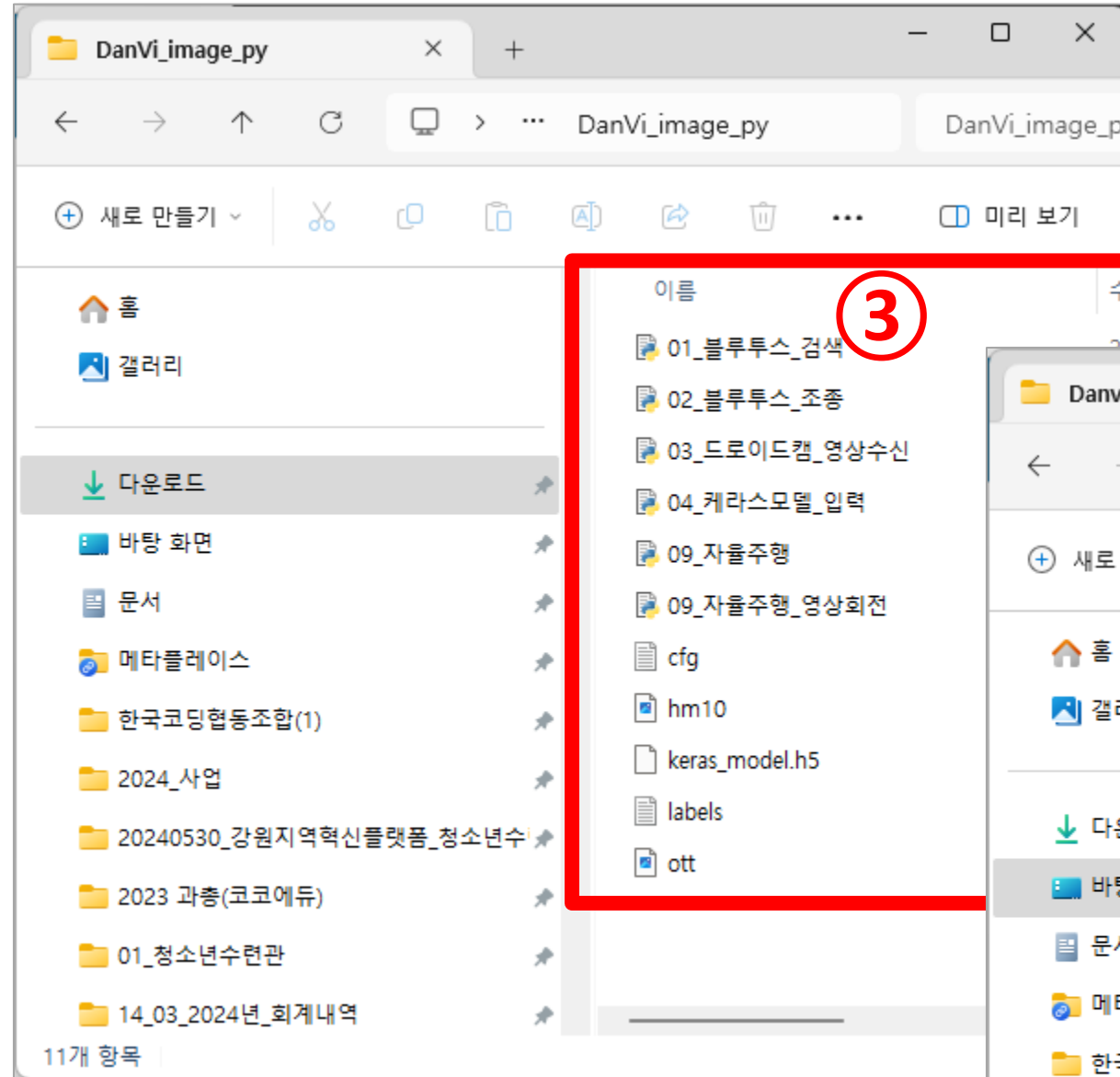
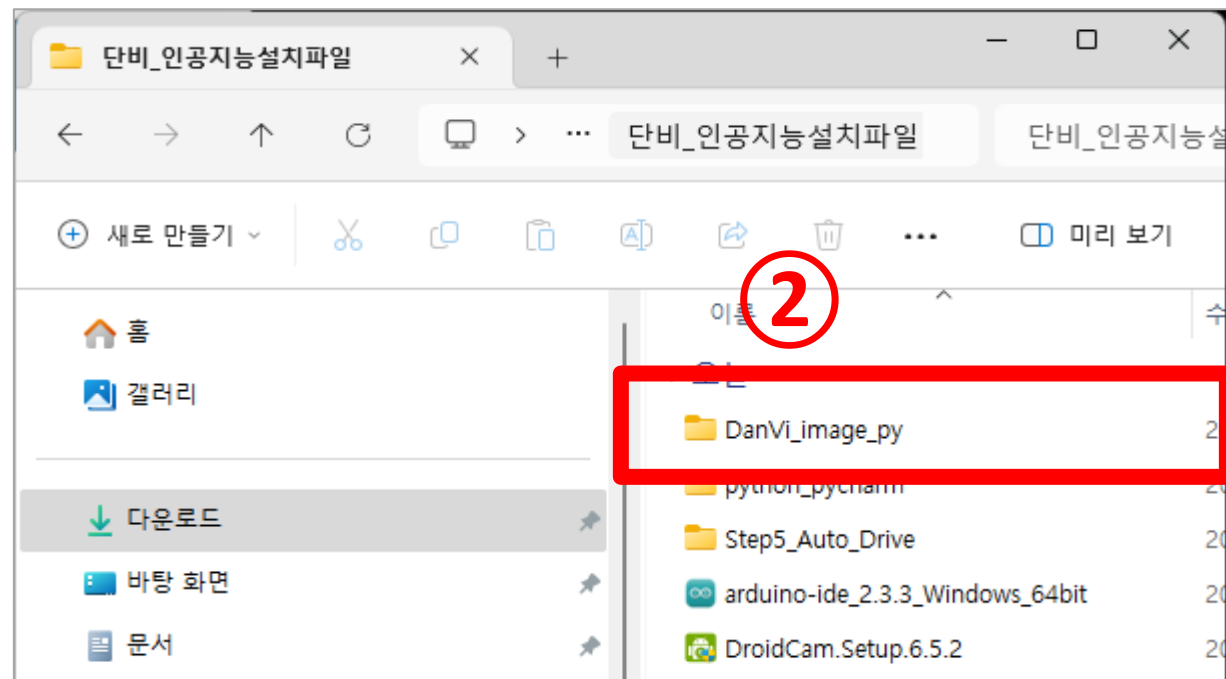
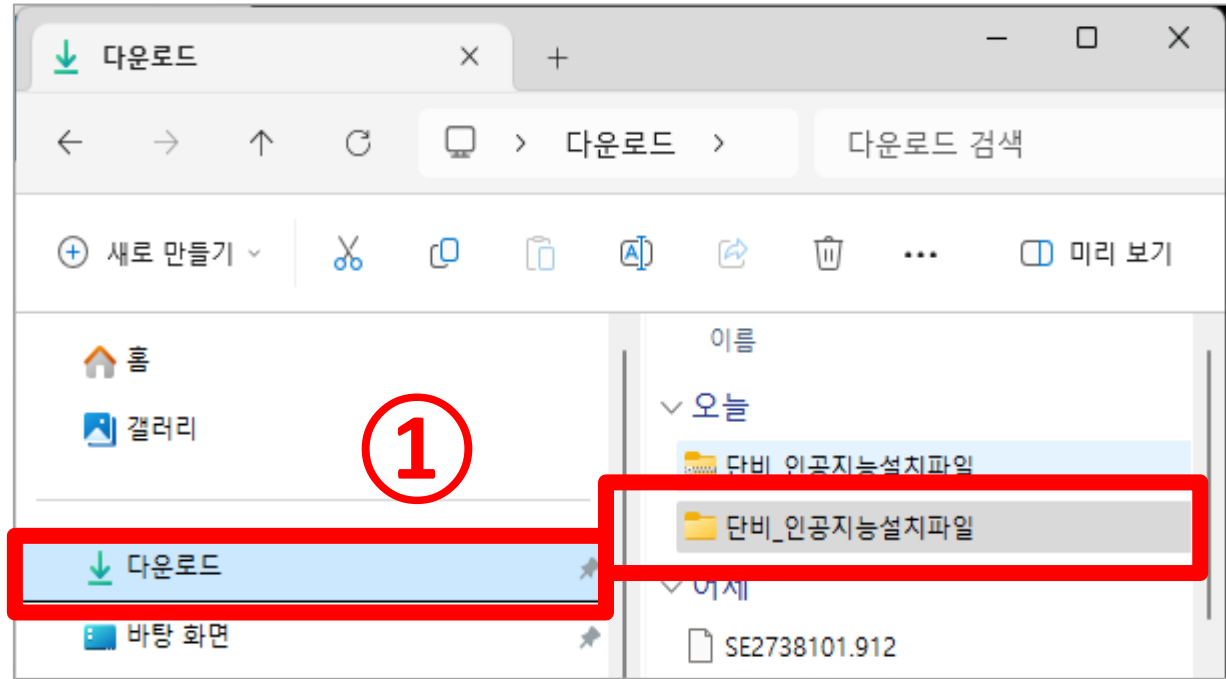
<X>를 눌러 창을 닫아주세요!



Teachable Machine

티처블 머신에서 학습모델
"내보내기"

- Danvi_racer 폴더에 파이썬 예제 파일 복사합니다.

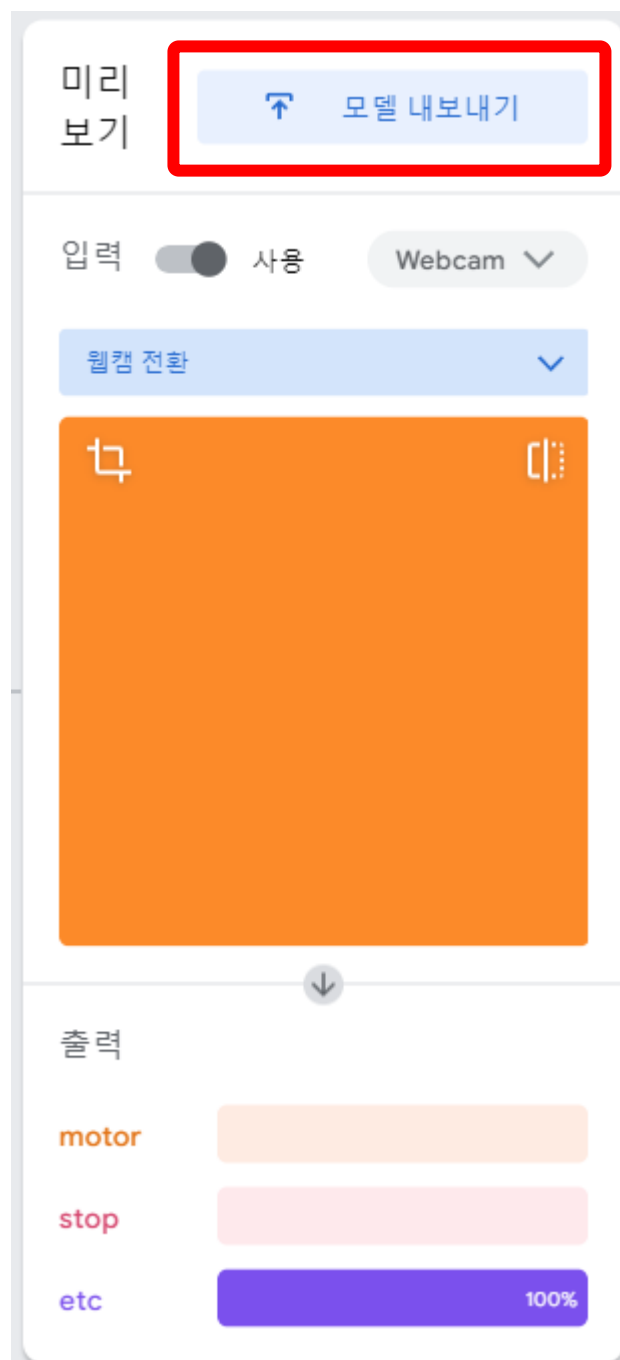


[Danvi_image_py] 폴더의 모든 파일을

[바탕화면]-[Danvi_racer] 폴더로 복사!!

- 텐서플로우 : 케라스 모델 다운로드하기

① 모델 내보내기

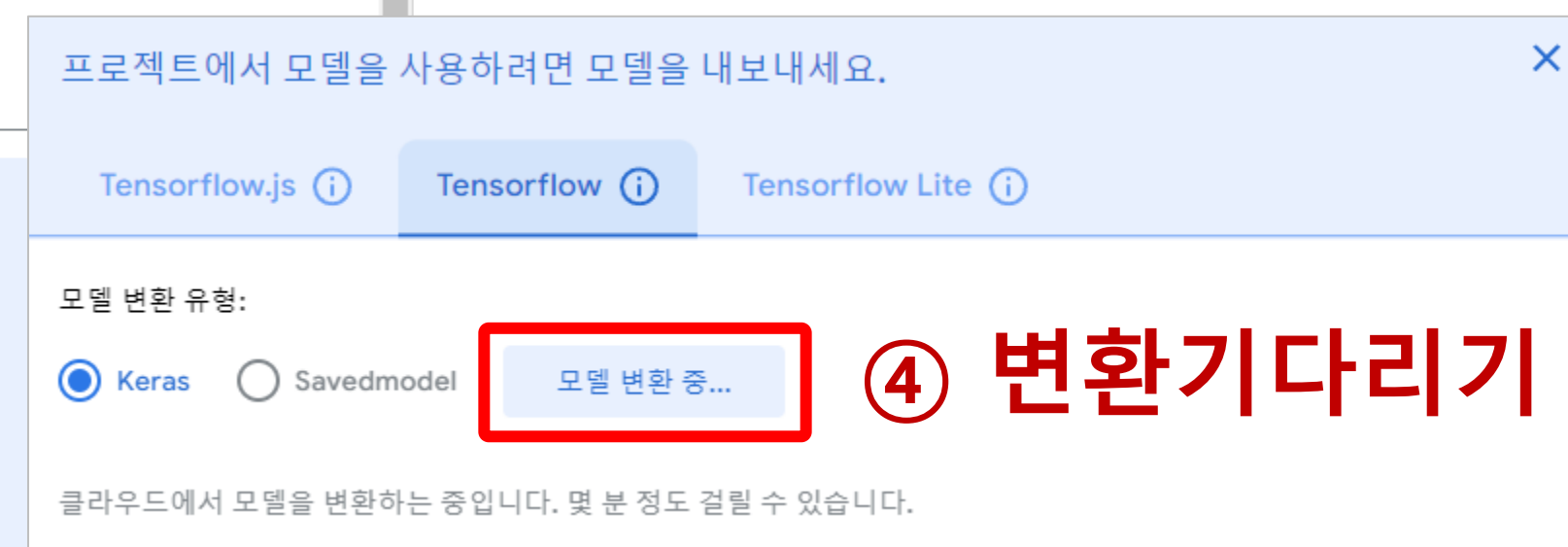


② 두번째탭 선택



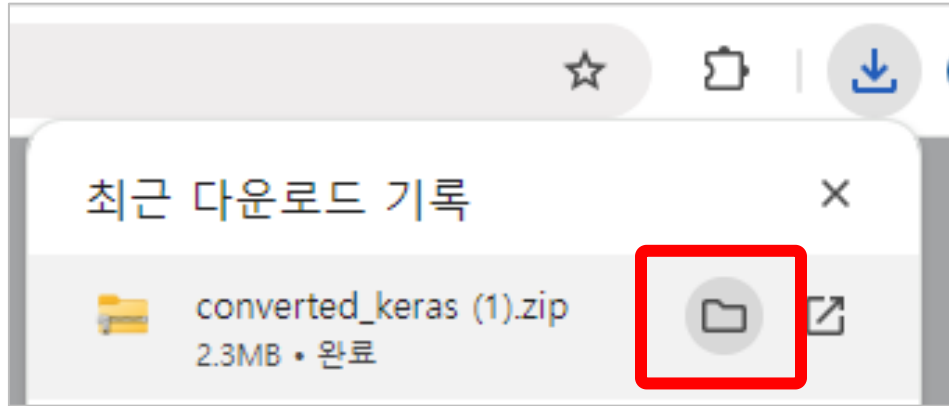
③

④ 변환기다리기

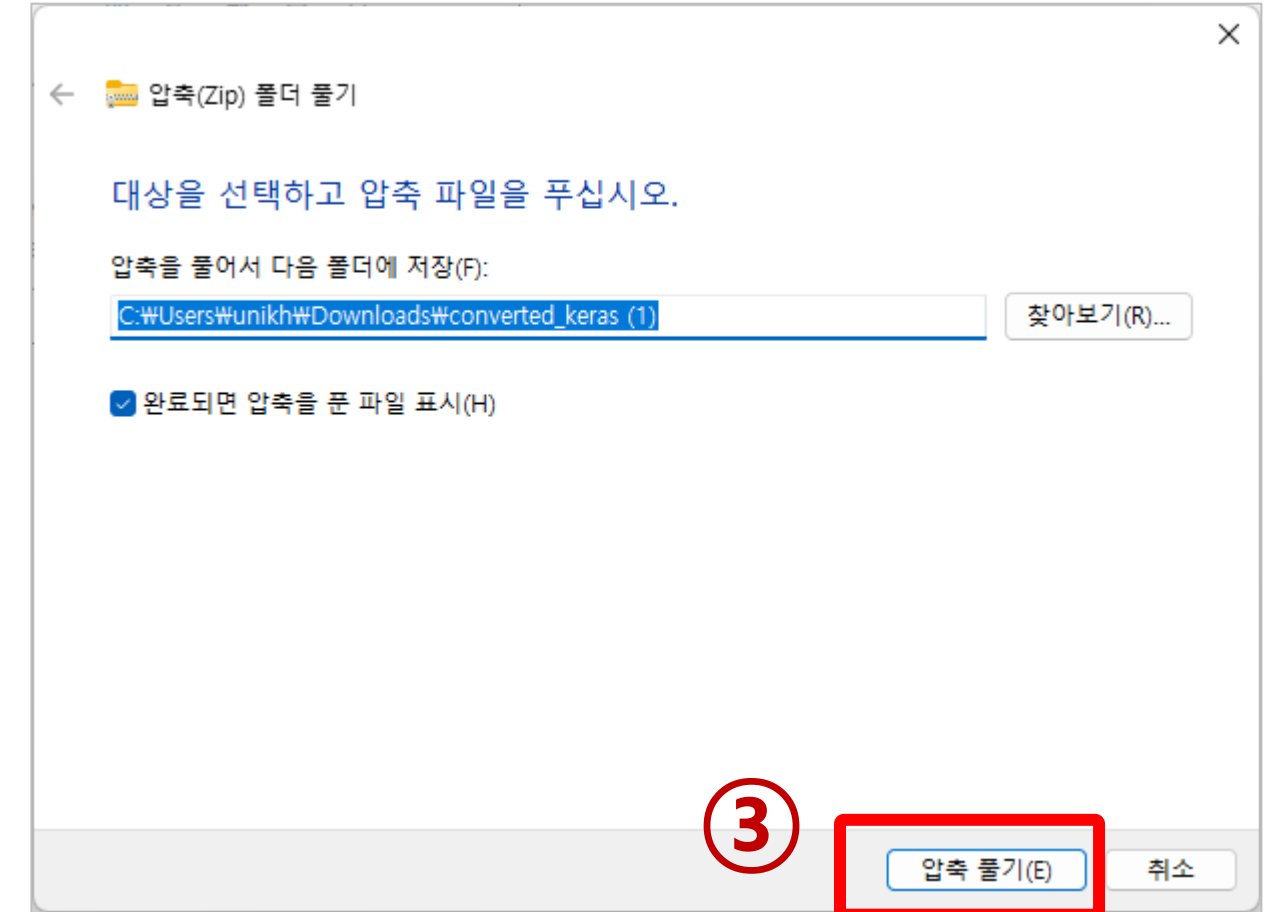
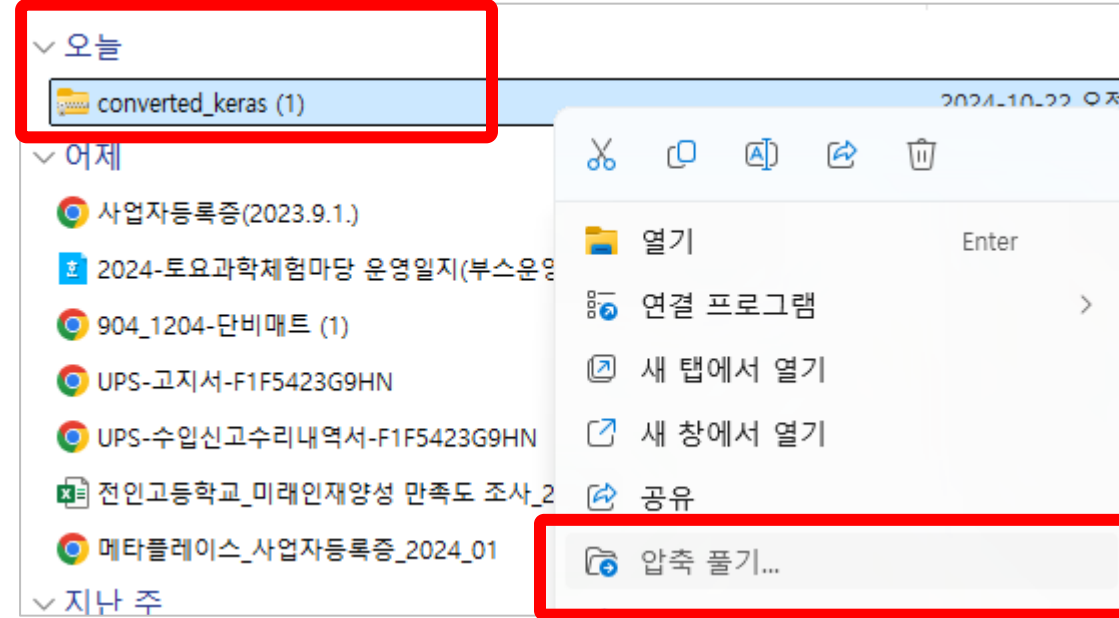


6- 텐서플로우 : 케라스 모델 다운로드하기

① 윈도우 탐색기에서 폴더열기



② 압축풀기

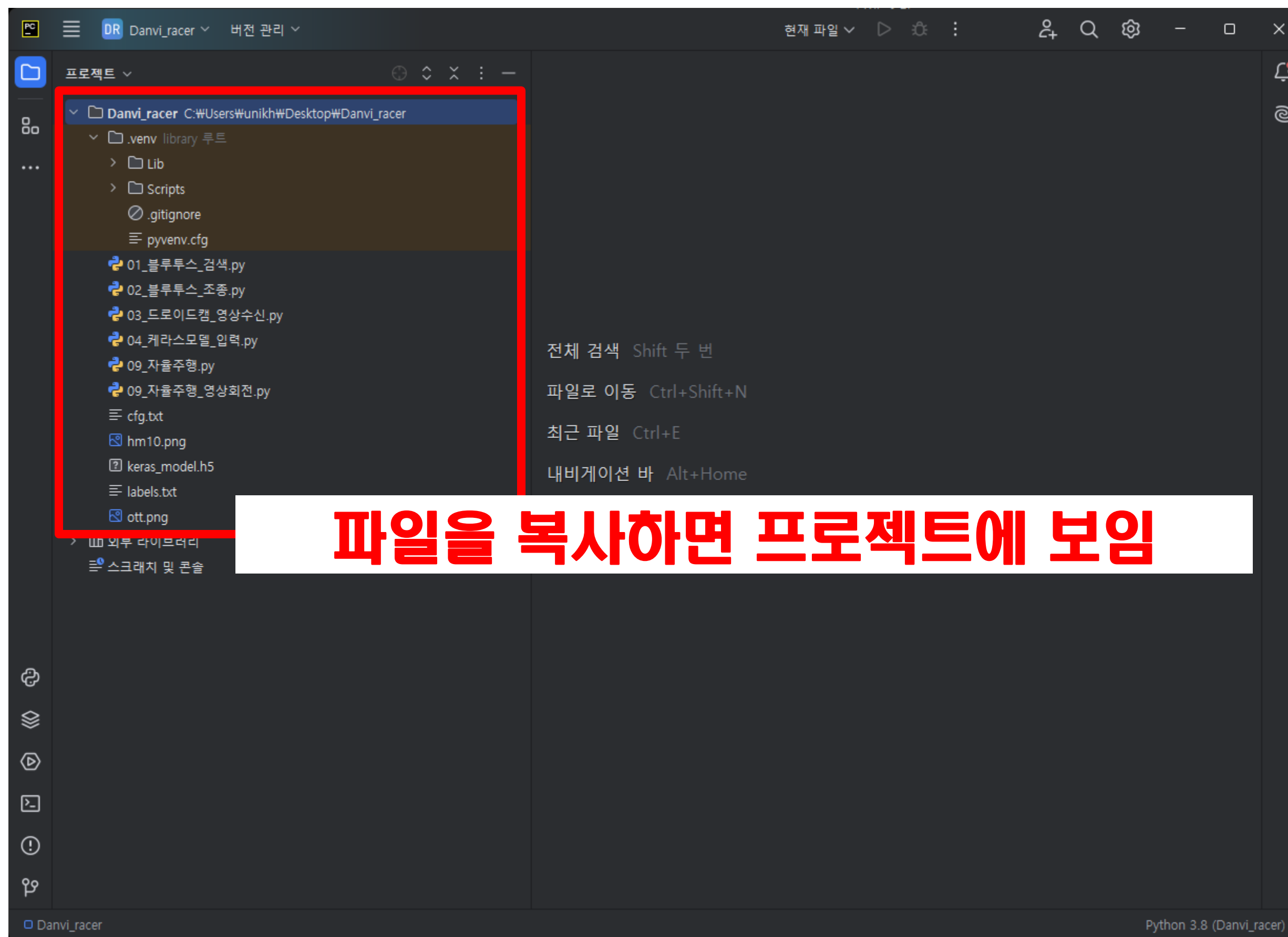


④ 두 개의 파일을 파이썬 프로젝트 폴더에 복사하기

이름	수정한 날짜	유형	크기
keras_model.h5	2024-10-22 오전 11:58	H5 파일	2,396KB
labels	2024-10-22 오전 11:58	텍스트 문서	1KB

[주의] 앞에서 예제파일 폴더에 내가 만든 모델을 덮어쓰기 해주세요!!!!

- 자율주행을 위한 준비가 되었습니다!

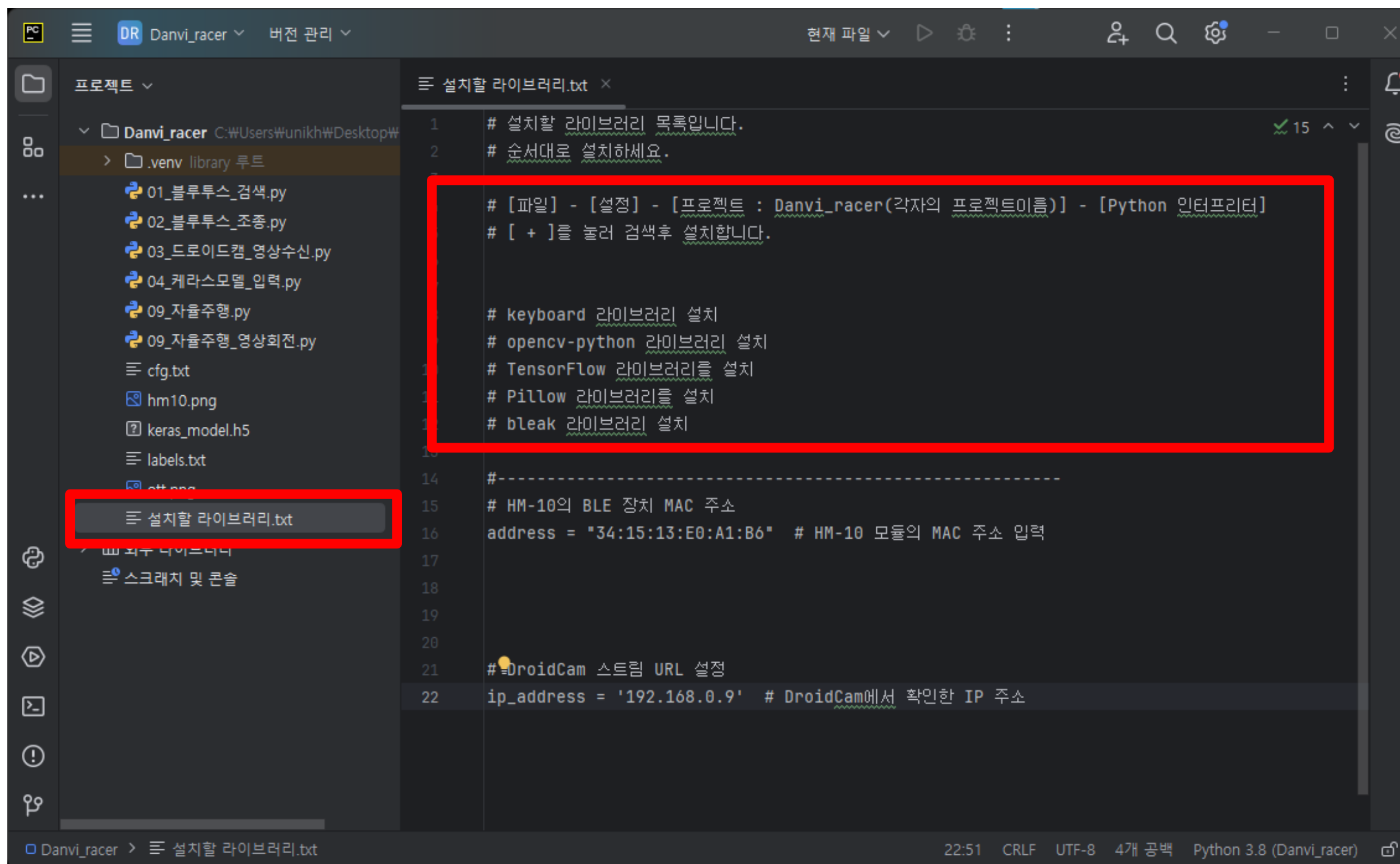




메카넬 AI 로봇의 자율주행을 위한 라이브러리 설치

- 라이브러리 설치하기 : 설치할 라이브러리.txt 파일 참조

- 인공지능 단비로봇의 영상을 인식하기 위해서 다음 라이브러리를 모두 순서대로 설치합니다.



```
1 # 설치할 라이브러리 목록입니다.
2 # 순서대로 설치하세요.

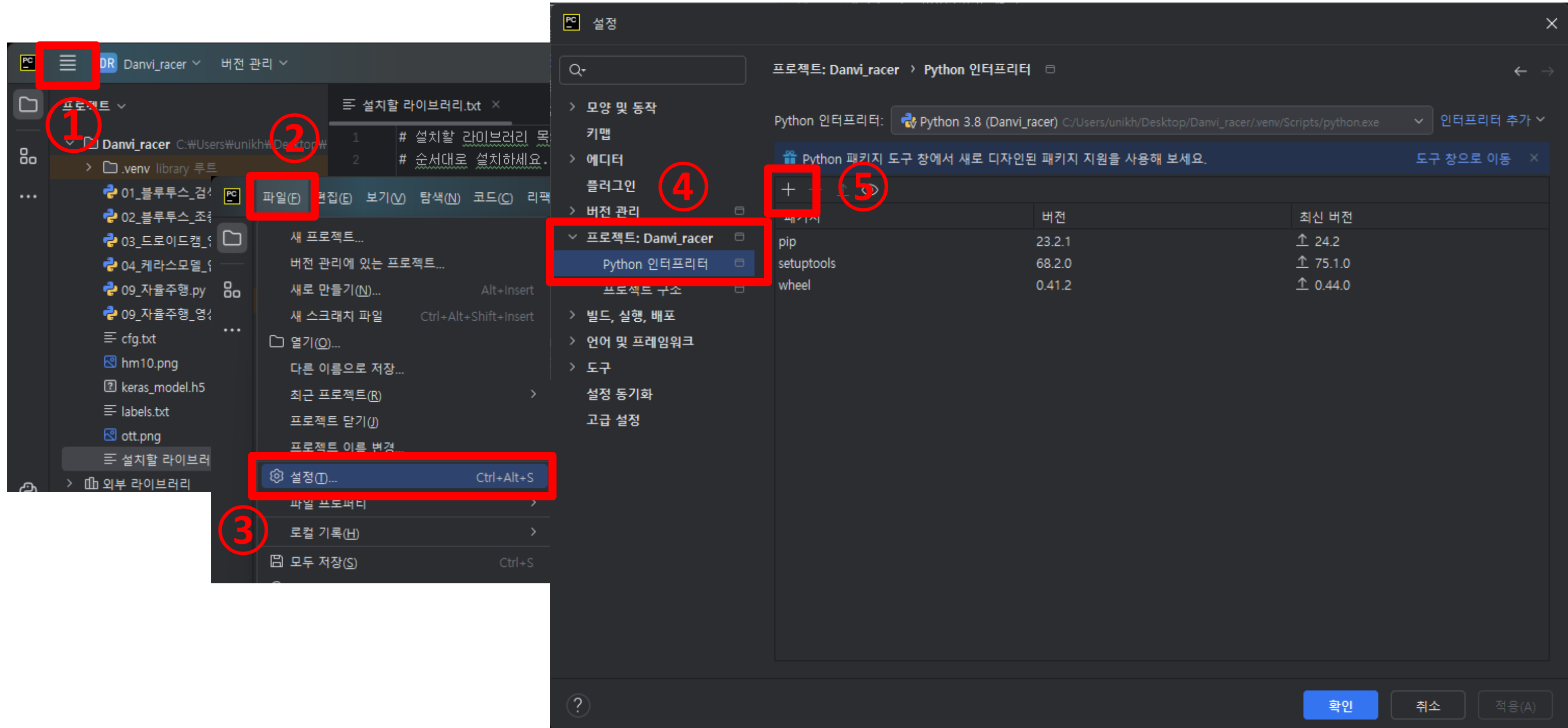
# [파일] - [설정] - [프로젝트 : Danvi_racer(각자의 프로젝트이름)] - [Python 인터프리터]
# [ + ]를 눌러 검색후 설치합니다.

# keyboard 라이브러리 설치
# opencv-python 라이브러리 설치
# TensorFlow 라이브러리를 설치
# Pillow 라이브러리를 설치
# bleak 라이브러리 설치

#-----
15 # HM-10의 BLE 장치 MAC 주소
16 address = "34:15:13:E0:A1:B6" # HM-10 모듈의 MAC 주소 입력
17
18
19
20
21 # DroidCam 스트림 URL 설정
22 ip_address = '192.168.0.9' # DroidCam에서 확인한 IP 주소
```

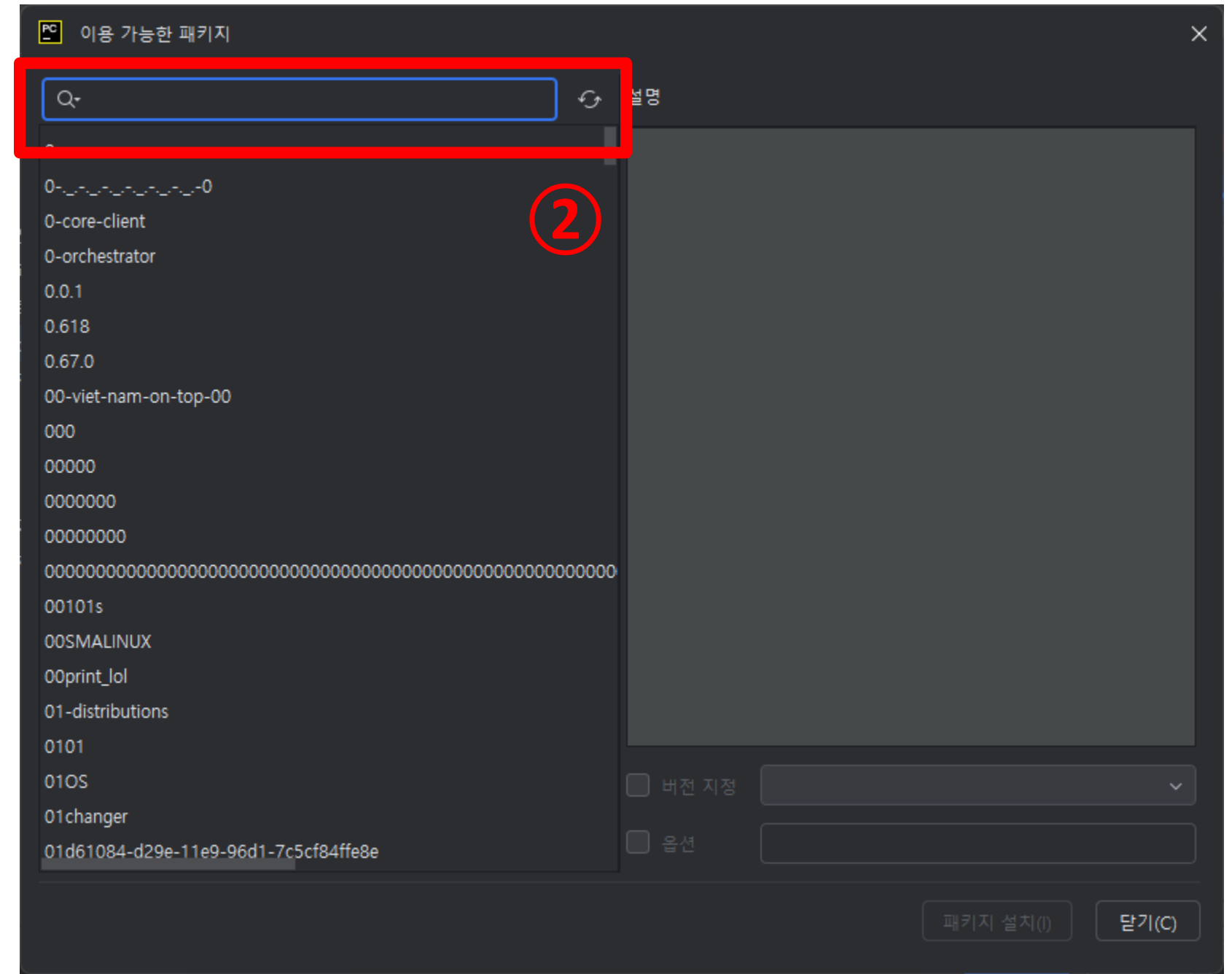
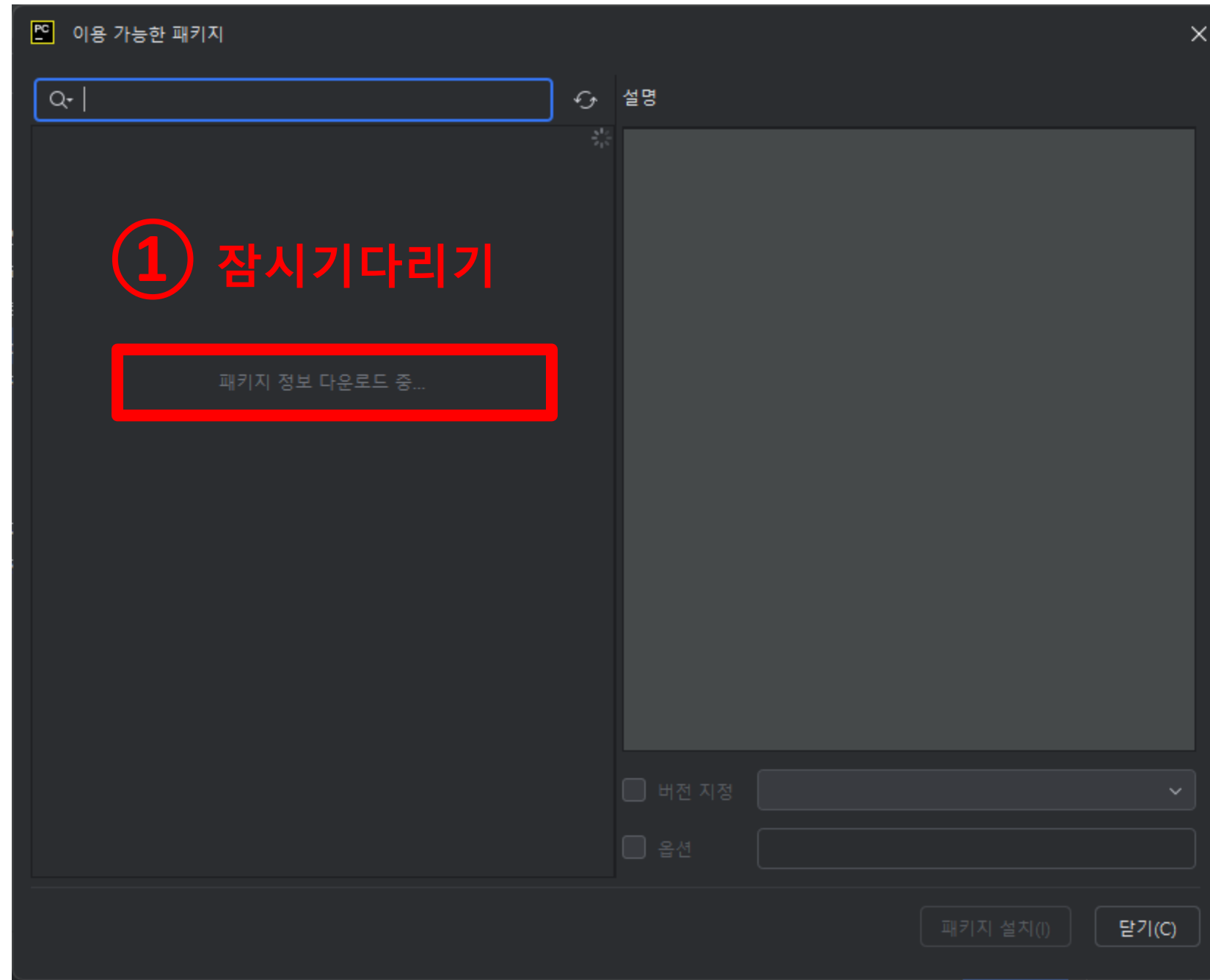
- 설정에서 메뉴를 선택합니다.

- ≡ 를 누르면 [파일]메뉴가 나타나고, [설정] – [Python 인터프리터]메뉴에서 [+]를 눌러 패키지 설치 창을 확인합니다.



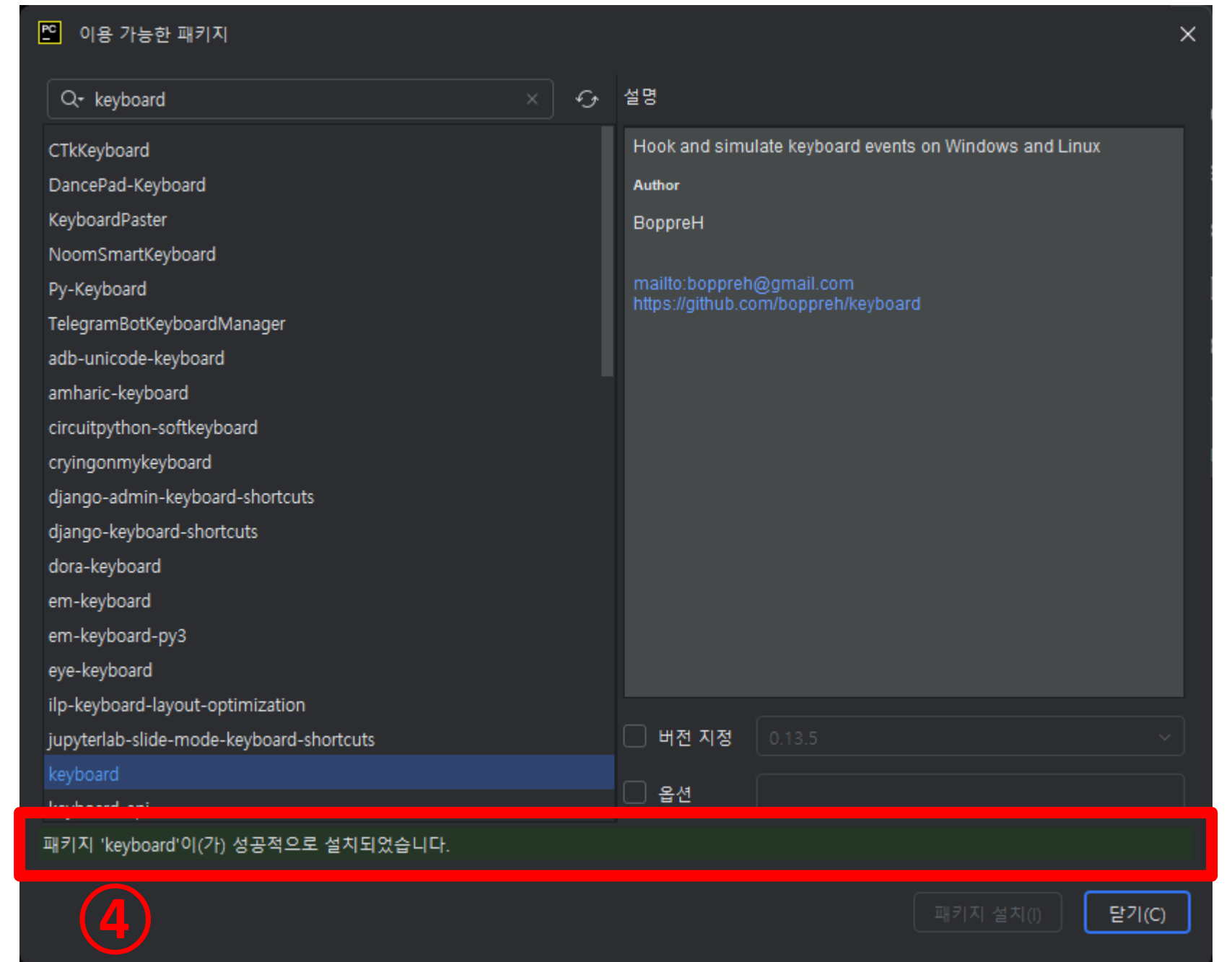
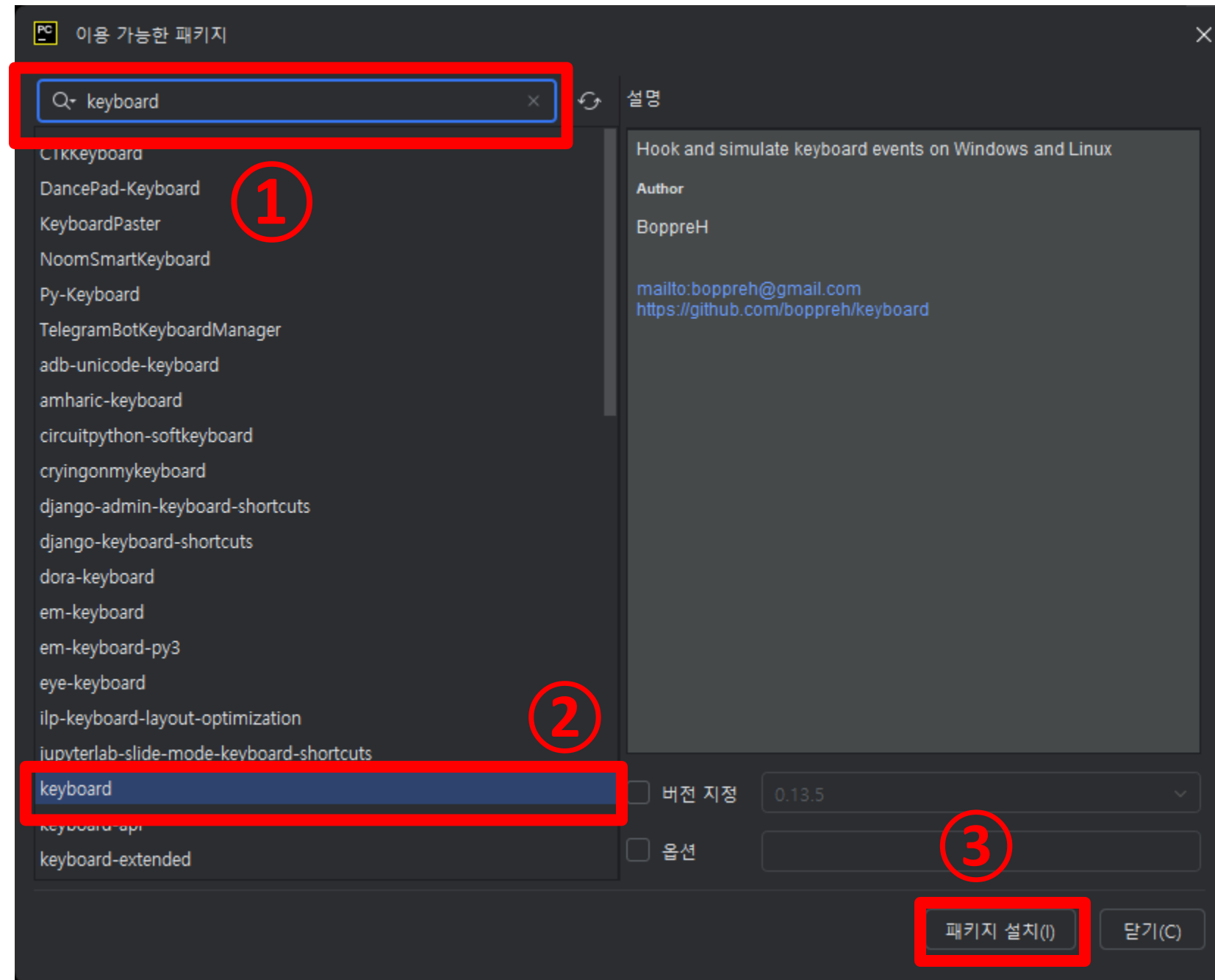
- 이용가능한 패키지 목록확인하기

- ①번과 같이 나타나면 잠시 기다립니다. ②번과 같이 목록이 나타나면 검색하여 패키지 설치를 시작합니다.



- Keyboard를 검색창에 입력하고 패키지를 설치한다.

- 라이브러리 이름과 같은지 확인한 후, [패키지 설치]버튼을 눌러 성공적으로 설치되었다는 메시지를 확인합니다.



- 다음순서로 패키지를 설치합니다.

[파일] - [설정] - [프로젝트 : Danvi_racer(각자의 프로젝트이름)] - [Python 인터프리터]

[+]를 눌러 검색후 설치합니다.

설치할 라이브러리를 순서대로 설치하세요. 특히 bleak를 마지막에 설치하는것이 좋습니다.

opencv-python

tensorflow

pillow

keyboard

bleak

메카넘휠 AI 로봇의 자율주행을 위한 나만의 블루투스 이름 바꾸기



: 여러 블루투스가 있으면 나만의 이름으로 알 수 있는 블루투스로 이름을 변경해야 합니다.



HM-10 블루투스



블루투스에 이름이 붙어있으면 이 과정은 건너뛰기!

나만의 로봇을 제어하려면 블루투스 이름을 변경하기

```
DANVI_06_Bluetooth_Naming | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Micro
DANVI_06_Bluetooth_Naming.ino
1 void setup()
2 {
3   //시리얼 모니터 연결
4   Serial.begin(9600);
5   //블루투스시리얼[하드웨어] 초기화(9600)bps
6   Serial1.begin(9600);
7 }
8
9 void loop()
10 {
11 // 블루투스시리얼[하드웨어] 명령 수신
12 if (Serial1.available())
13 // Serial1[하드웨어] 값이 있으면
14 {
15   Serial.write(Serial1.read());
16   //블루투스측 내용을 시리얼모니터에 출력
17 }
18 // 시리얼통신 명령 수신
19 if (Serial.available())
20 {
21   Serial1.write(Serial.read());
22   // 시리얼모니터 내용을 블루투스측에 출력
23 }
24 }
25
26 /* 블루투스 이름 변경방법
27 "No line ending [9600 baud]"
28 AT (OK)
29 AT+NAME이름 (OK+Set:이름)
30 */
```

- 아두이노 Micro 보드에서의 블루투스 명령어 쓰는 방법입니다.

- *Serial1.begin(9600);*

- 시리얼 통신 속도를 9600으로 설정

- *Serial1.read();*

- 시리얼 포트에서 데이터 읽음

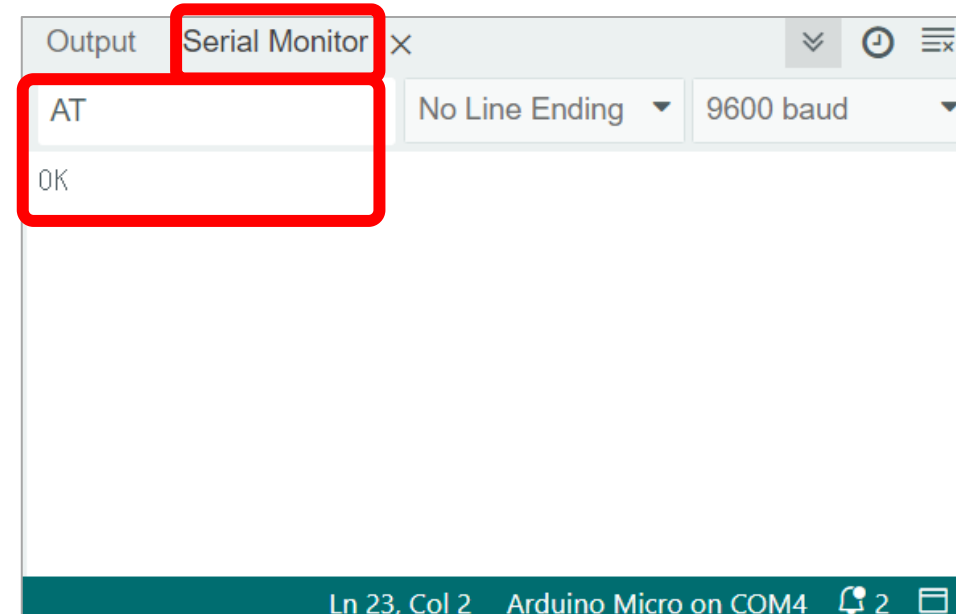
- *Serial1.write();*

- 시리얼 포트에 데이터를 씀

- 시리얼모니터에서

- OK (엔터)

- OK+NAME이름을(엔터)





메카넘휠 AI 로봇의 자율주행을 위한 인공지능 학습을 시작할게요!

STEP 1. 영상연결하기

파일명 : 01_드roid캠_영상수신.py

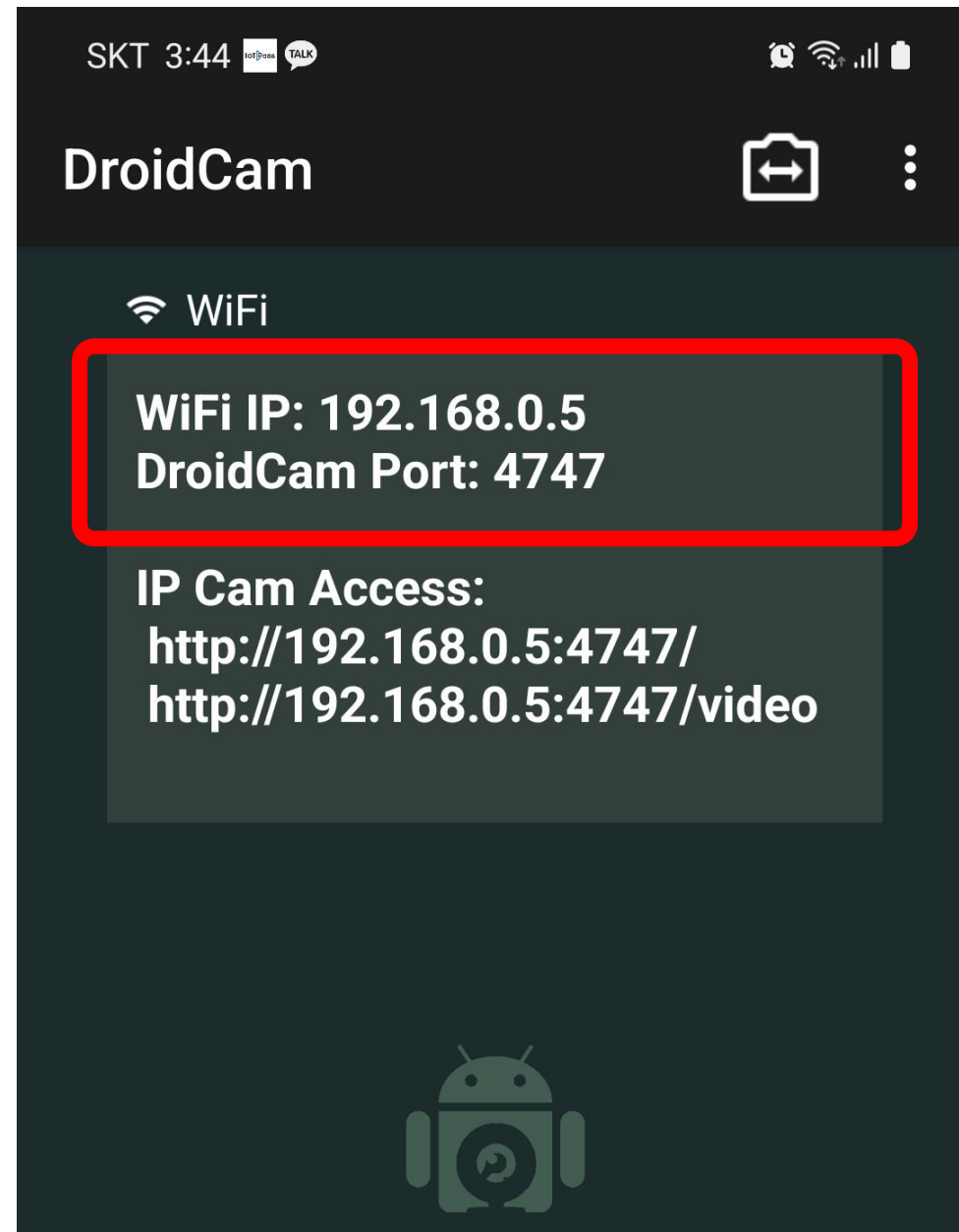
스토어에서 “**드로이드캠**”을 검색하여 설치할게요~

iOS 와 Android폰 아이콘 확인



[단계 1] 드로이드캠과 스마트폰 영상연결하기

스마트폰에 드로이드캠을 설치후 실행시키면 다음과 같이 화면이 나타납니다.



[스마트폰 드로이드캠]

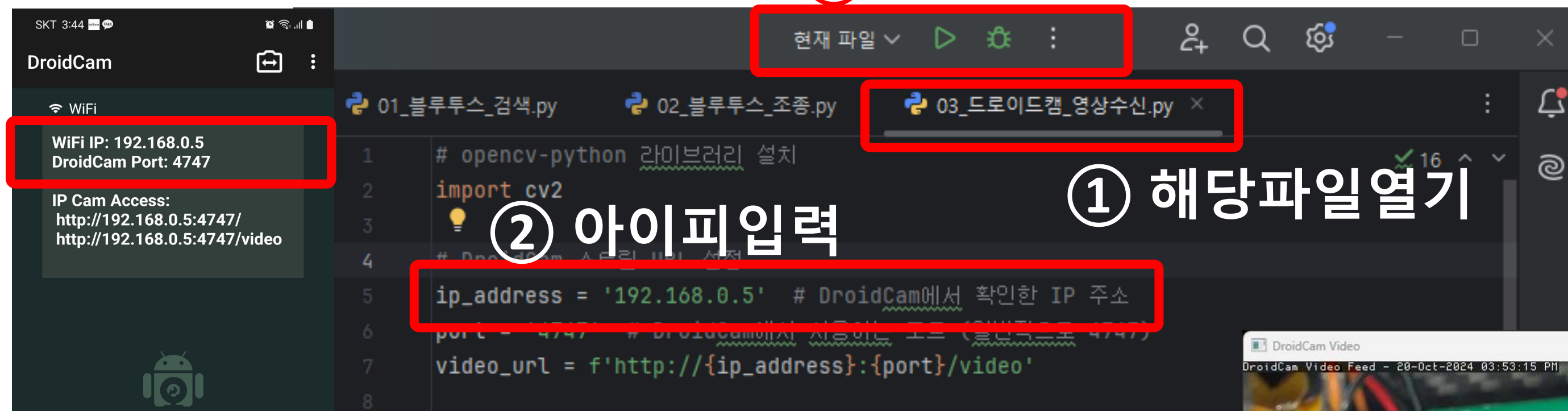
- 무선와이파이를 이용하기 위해서 **스마트폰과 PC(노트북)가 동일한 무선 와이파이**에 연결되어 있어야 함.
- WiFi 번호와 DroidCam Port 번호를 PC의 드로이드 캠에 똑같이 입력합니다.

와이파이 연결상태 꼭! 확인필요

[단계 1] 드로이드캠과 스마트폰 영상연결하기

- 드로이드캠의 WiFi IP 번호를 파이참 프로그램 ip_address에 입력합니다.

③ 현재파일 실행하기



① 해당파일열기

② 아이피입력

[영상 화면]

[스마트폰 화면]



④ 드로이드캠과 영상수신

STEP 2. 케라스모델 확인

파일명 : 02_케라스모델_입력.py

[단계 2] 티처블머신에서 내보내기한 "케라스모델 " 확인하기

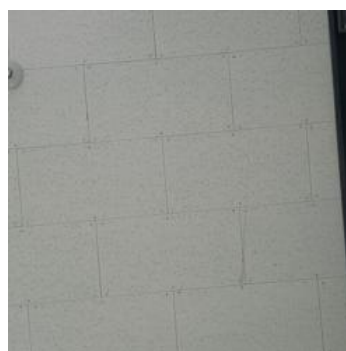
- 케라스모델의 motor,stop,etc 클래스를 인식하는지 모델입력을 해봅니다.



ott.png



st.png



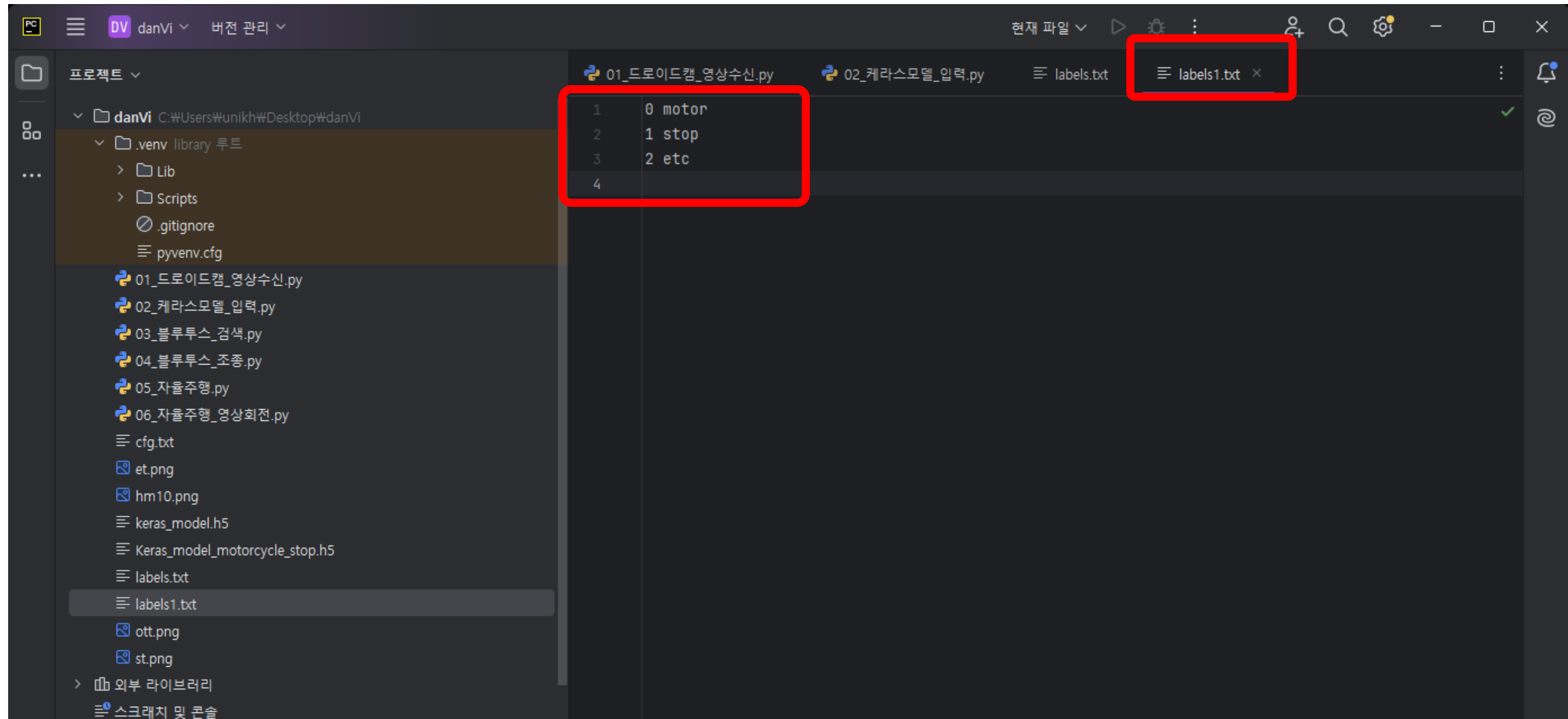
et.png

```
def load_and_predict_frame(model, frame):  
    class_name = class_names[index].strip()  
    confidence_score = prediction[0][index]  
  
    return class_name, confidence_score  
  
# PNG 이미지 파일 경로  
image_path = "ott.png"  
# OpenCV로 이미지를 읽어들이 (BGR 형식으로 변환됨)  
frame = cv2.imread(image_path)  
# 이미지가 정상적으로 읽어졌는지 확인  
if frame is None:  
    print("이미지를 읽을 수 없습니다.")  
else:  
    class_name, confidence_score = load_and_predict_frame(model, frame)
```

실행 C:\Users\unikh\Desktop\Danvi_racer\.venv\Scripts\python.exe C:\Users\unikh\Desktop\Danvi_racer\03_드로이드캠_영상수신.py

[단계 2] 티처블머신에서 내보내기한 "케라스모델 " 확인하기

- 케라스모델의 motor,stop,etc 클래스를 인식하는지 모델입력을 해봅니다.



```
1 0 motor
2 1 stop
3 2 etc
4
```


[단계 2] 티처블머신에서 내보내기한 “케라스모델 ” 확인하기

- 케라스모델에서 학습된 이미지가 첫번째 클래스인 “motor” 이미지로 99% 학습된 것을 확인할수 있다.

```
def load_and_predict_frame(model, frame):  
    # 모델 예측  
    prediction = model.predict(data)  
    # 가장 높은 확률을 가진 클래스의 인덱스  
    index = np.argmax(prediction)  
    # 예측된 클래스 이름과 신뢰도 점수 반환  
    class_name = class_names[index].strip()  
    confidence_score = prediction[0][index]  
    return class_name, confidence_score  
  
image_path = "ott.png"  
frame = cv2.imread(image_path)  
# 이미지가 정상적으로 읽어졌는지 확인  
if frame is None:  
    print("이미지를 읽을 수 없습니다.")  
else:
```

② 영상이 확인되면 x표로 닫기

① ott.png 수정

③ motor 클래스로 99% 인식

[단계 2] 티처블머신에서 내보내기한 "케라스모델 " 확인하기

- 이미지의 파일명을 "st.png" 으로 바꾸어 stop이미지가 확인되면 99% 인식됨을 알수있다.

① st.png 수정

② 영상이 확인되면 x표로 닫기

③ stop 클래스로 99% 인식

[단계 2] 티처블머신에서 내보내기한 "케라스모델 " 확인하기

- 이미지의 파일명을 "et.png" 으로 바꾸어 기타이미지가 인식되면 70%로 인식된다.
- 기타이미지의 학습율을 높이려면 티처블머신에서 다시 교육시킬수 있다.

① et.png 수정

② 영상이 확인되면 x표로 닫기

③ etc 클래스로 70% 인식

```
def load_and_predict_frame(model, frame):  
    # 모델 예측  
    prediction = model.predict(data)  
    # 가장 높은 확률을 가진 클래스의 인덱스  
    index = np.argmax(prediction)  
    # 예측된 클래스 이름과 신뢰도 점수 반환  
    class_name = class_names[index].strip()  
    confidence_score = prediction[0][index]  
    return class_name, confidence_score  
  
image_path = "et.png"  
frame = cv2.imread(image_path)  
if frame is None:  
    print("이미지를 읽을 수 없습니다.")  
else:
```

```
C:\Users\unikh\Desktop\Danvi_racer\.venv\Scripts\python.exe C:\Users\unikh\Desktop\Danvi_racer\04_케라스모델_입력.py  
2024-10-25 10:49:31.269074: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use avail  
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the  
1/1 [-----] - 1s 578ms/step  
etc 0.7083412  
종료 코드 0(으)로 완료된 프로세스
```

STEP 3. 블루투스 검색하기

파일명 : 03_블루투스_검색.py

[단계 3] 블루투스를 검색하기

- 단비의 전원을 켜고, 블루투스의 이름을 확인한다.

```
1 # bleak 라이브러리 설치
2 > import ...
3
4
5 # 주변 BLE 장치 스캔 (비동기 호출을 동기식으로 처리)
6 def scan_ble_devices_sync(): 1개의 사용 위치
7     loop = asyncio.new_event_loop() # 새로운 이벤트 루프 생성
8     asyncio.set_event_loop(loop)   # 현재 이벤트 루프로 설정
9     devices = loop.run_until_complete(BleakScanner.discover()) # 비동기 함수 실행
10    loop.close() # 이벤트 루프 종료
11
12 # 스캔된 장치 출력
13 for device in devices:
14     print(device)
15
16 # 동기식 함수 호출
17 scan_ble_devices_sync()
```

[단계 3] 블루투스를 검색하기

- 자신의 단비 블루투스 이름을 확인하고, 맥 어드레스를 복사한다.

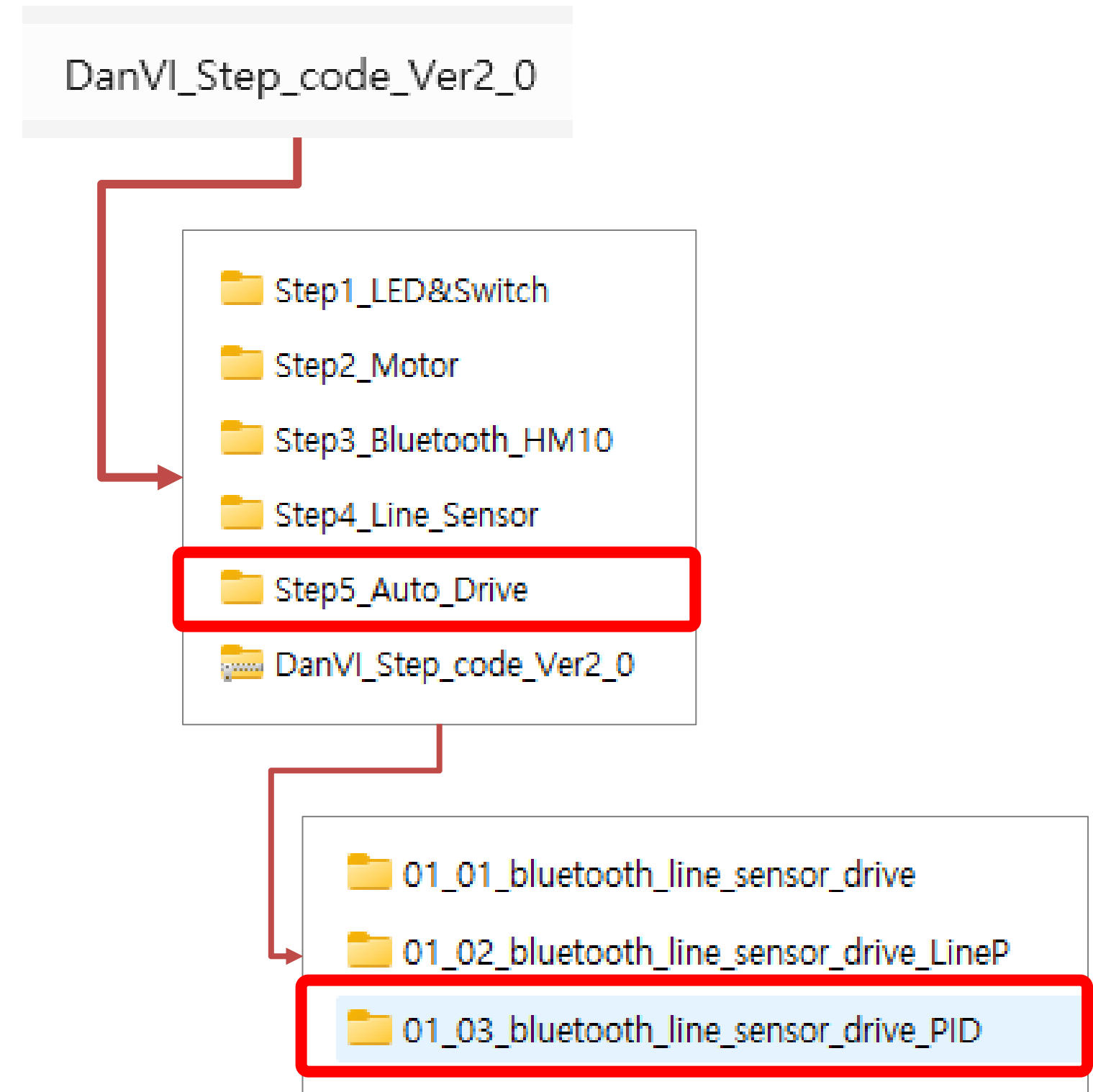
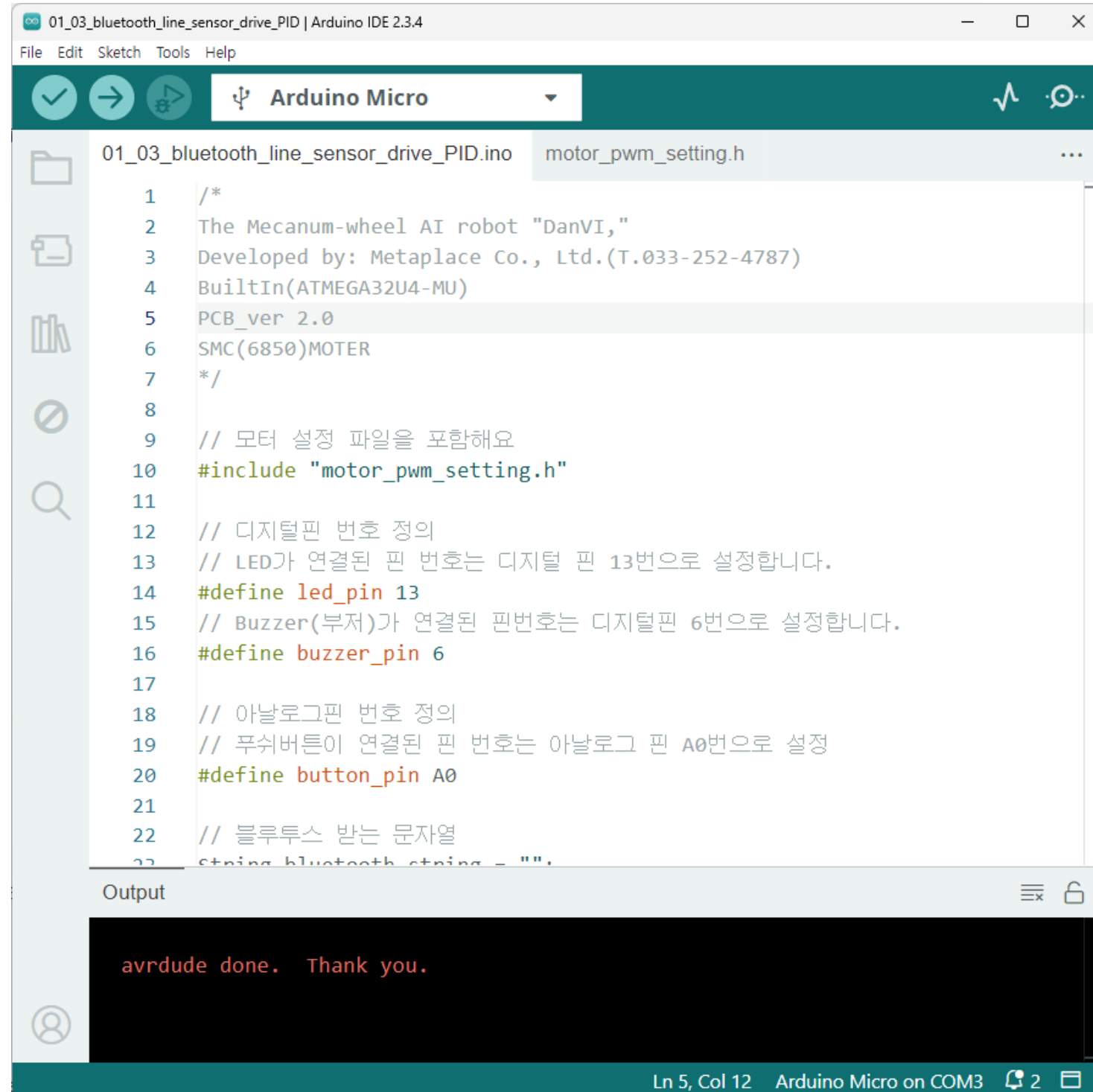
```
09_자율주행.py 9
09_자율주행_영상회전.py 10
cfg.txt 11
hm10.png 12
keras_model.h5 13
labels.txt 14
ott.png 15
설치할 라이브러리.txt 16
01_블루투스_검색 x
C:\Users\unikh\Desktop\Danvi_racer\.venv_new\Scripts\python.exe C:\Users\unikh\Desktop\Dan
57:08:7C:EF:46:DD: None
05:75:E5:92:5B:3A: None
18:93:D7:33:21:74: danvi6850
71:60:26:EE:F4:E4: None
10:04:5E:88:47:C5: None
4B:79:01:31:04:7C: None
```


STEP 4. 블루투스로 조종하기

파일명 : 04_블루투스_조종.py

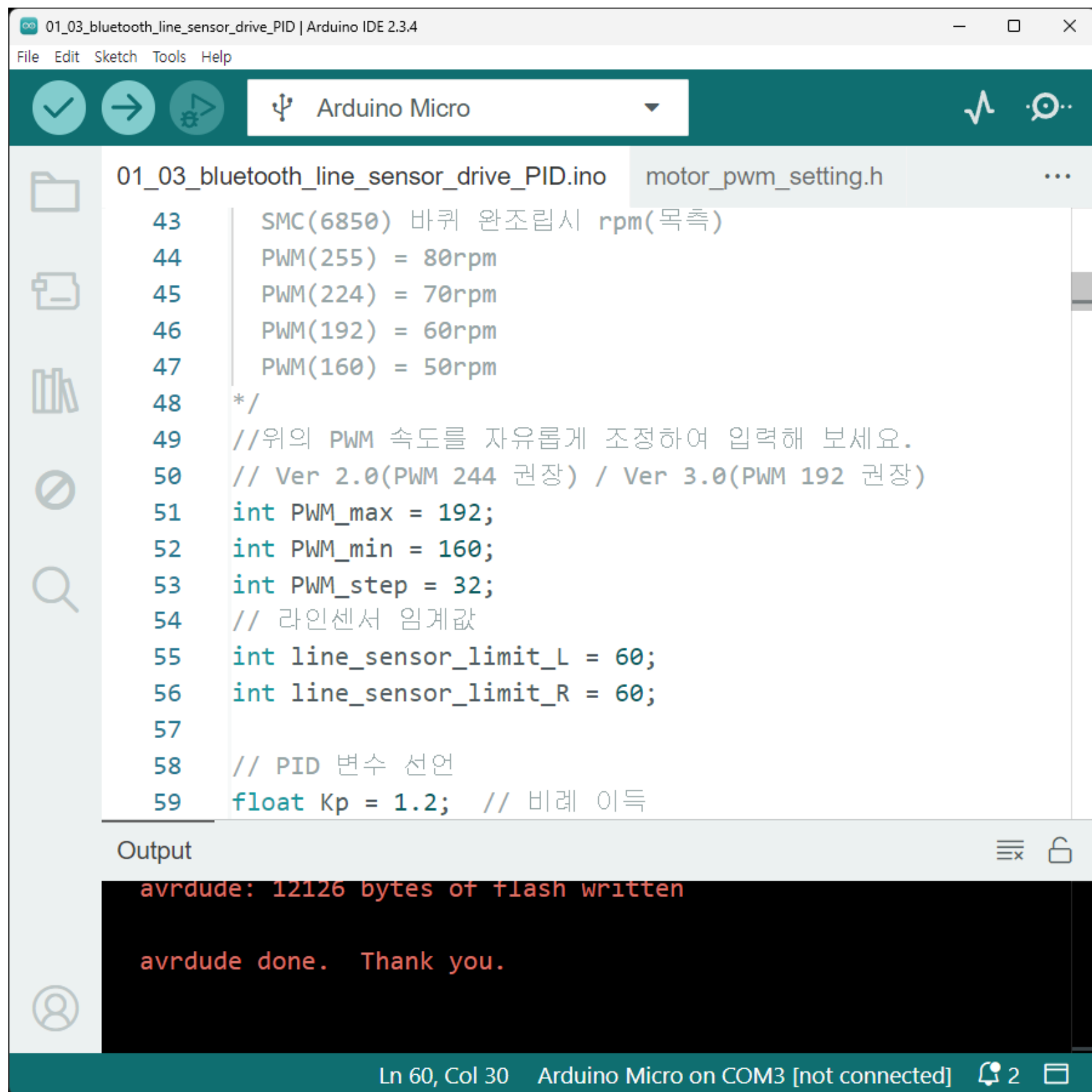
[단계 4] 블루투스로 조종하기

- AutoDrive 프로그램 소스를 업로드 시킨다.



[단계 4] 블루투스로 조종하기

- 라인센서 값을 확인하여 라인트레이서 작동을 확인한다.



```
01_03_bluetooth_line_sensor_drive_PID.ino  motor_pwm_setting.h
43   SMC(6850) 바퀴 완조립시 rpm(목측)
44   PWM(255) = 80rpm
45   PWM(224) = 70rpm
46   PWM(192) = 60rpm
47   PWM(160) = 50rpm
48   */
49   //위의 PWM 속도를 자유롭게 조정하여 입력해 보세요.
50   // Ver 2.0(PWM 244 권장) / Ver 3.0(PWM 192 권장)
51   int PWM_max = 192;
52   int PWM_min = 160;
53   int PWM_step = 32;
54   // 라인센서 임계값
55   int line_sensor_limit_L = 60;
56   int line_sensor_limit_R = 60;
57
58   // PID 변수 선언
59   float Kp = 1.2; // 비례 이득
```

Output

```
avrdude: 12126 bytes of flash written
avrdude done. Thank you.
```

Ln 60, Col 30 Arduino Micro on COM3 [not connected]

- `PWM_max`, `PWM_min`, `PWM_step` 변수는 PWM 신호의 최대, 최소값과 단계(step)를 설정합니다.
- 이 값들은 모터의 속도를 제어하기 위한 범위와 변화 간격을 지정합니다.

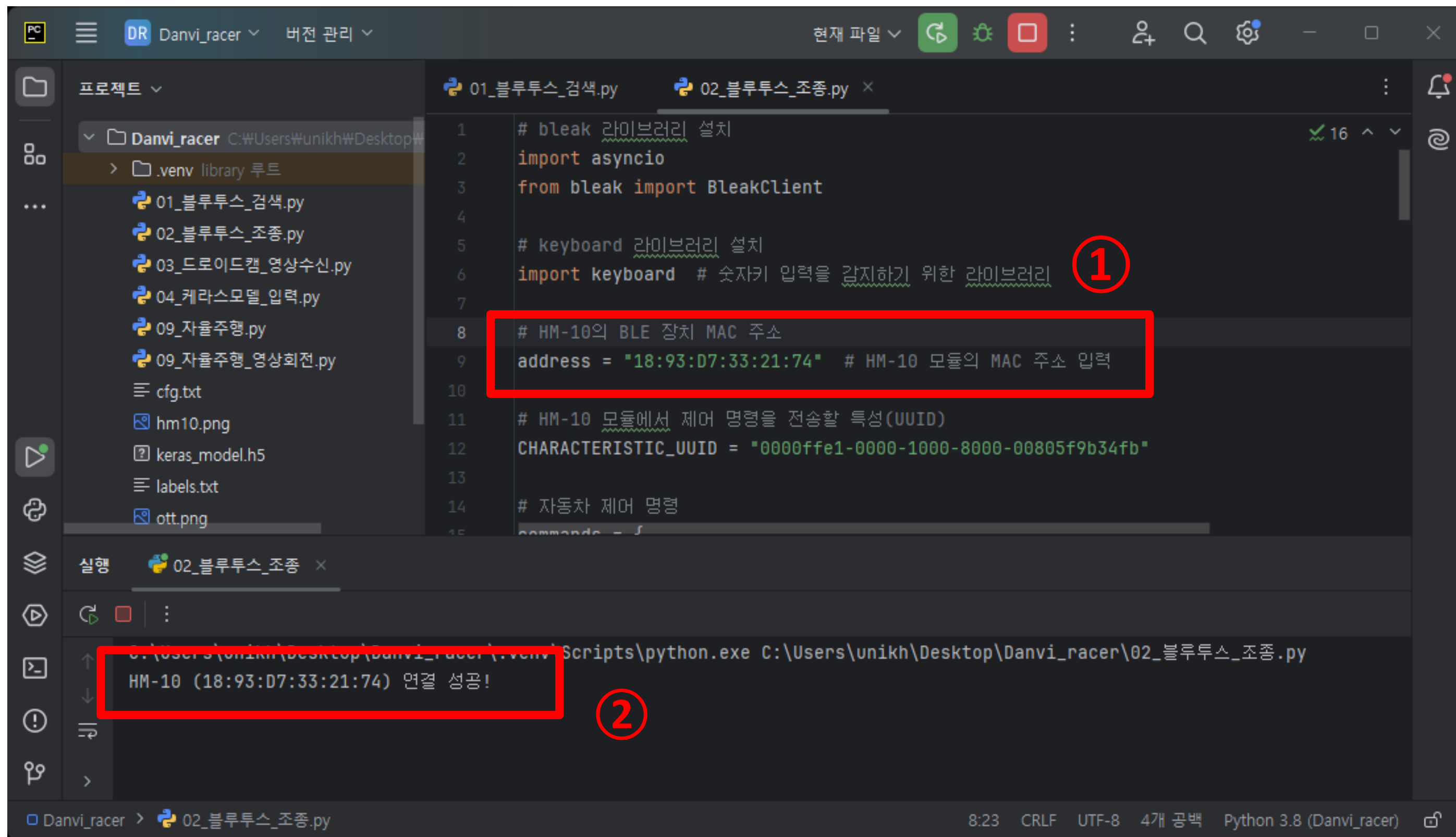
```
c 코드 복사
int PWM_max = 192; // PWM 최대값
int PWM_min = 160; // PWM 최소값
int PWM_step = 32; // PWM 변화 단계
```

- `line_sensor_limit_L`, `line_sensor_limit_R` 변수는 왼쪽(L)과 오른쪽(R) 라인 센서의 임계값을 설정합니다.
- 이 값들은 라인 센서가 얼마나 민감하게 라인을 감지할지를 결정합니다.

```
c 코드 복사
int line_sensor_limit_L = 60; // 왼쪽 라인 센서 임계값
int line_sensor_limit_R = 60; // 오른쪽 라인 센서 임계값
```

[단계 4] 블루투스 조종하기

- 복사한 맥 어드레스 주소를 9라인에 주소를 붙여넣기 한다.



The screenshot shows a Python IDE with two files open: `01_블루투스_검색.py` and `02_블루투스_조종.py`. The code in `02_블루투스_조종.py` is as follows:

```
1 # bleak 라이브러리 설치
2 import asyncio
3 from bleak import BleakClient
4
5 # keyboard 라이브러리 설치
6 import keyboard # 숫자키 입력을 감지하기 위한 라이브러리
7
8 # HM-10의 BLE 장치 MAC 주소
9 address = "18:93:D7:33:21:74" # HM-10 모듈의 MAC 주소 입력
10
11 # HM-10 모듈에서 제어 명령을 전송할 특성(UUID)
12 CHARACTERISTIC_UUID = "0000ffe1-0000-1000-8000-00805f9b34fb"
13
14 # 자동차 제어 명령
15 commands = f
```

Red annotations highlight the `import keyboard` line (circled with '1') and the `address = "18:93:D7:33:21:74"` line (boxed). The terminal at the bottom shows the command `C:\Users\unikh\Desktop\Danvi_racer\.venv\Scripts\python.exe C:\Users\unikh\Desktop\Danvi_racer\02_블루투스_조종.py` and the output `HM-10 (18:93:D7:33:21:74) 연결 성공!`, which is also boxed and circled with '2'.

[단계 4] 블루투스로 조종하기

- 2번을 누르면 전진! 5번을 누르면 중지!
- 블루투스연결이 되었으므로 파이썬에서 조종이 가능합니다.

```
1 # bleak 라이브러리 설치
2 import asyncio
3 from bleak import BleakClient
4
5 # keyboard 라이브러리 설치
6 import keyboard # 숫자키 입력을 감지하기 위한 라이브러리
7
8 # HM-10의 BLE 장치 MAC 주소
9 address = "18:93:D7:33:21:74" # HM-10 모듈의 MAC 주소 입력
10
11 # HM-10 모듈에서 제어 명령을 전송할 특성(UUID)
12 CHARACTERISTIC_UUID = "0000ffe1-0000-1000-8000-00805f9b34fb"
13
14 # 자동차 제어 명령
15 commands = {
```

실행 02_블루투스_조종

```
0' 명령 전송 완료
2'2' 명령 전송 완료
'2' 명령 전송 완료
5'5' 명령 전송 완료
'5' 명령 전송 완료
```



**[주의] 커서를 아래쪽에 한번 "클릭" 하고
2번을 누르면 전진, 5번을 누르면 중지하는지 확인합니다~**

STEP 5. 자율주행로봇의 영상확인

파일명 : 05_자율주행_영상회전.py

- 스마트폰과 거치대 연결하기

① 거치대와 스마트폰을 준비



② 결합하기

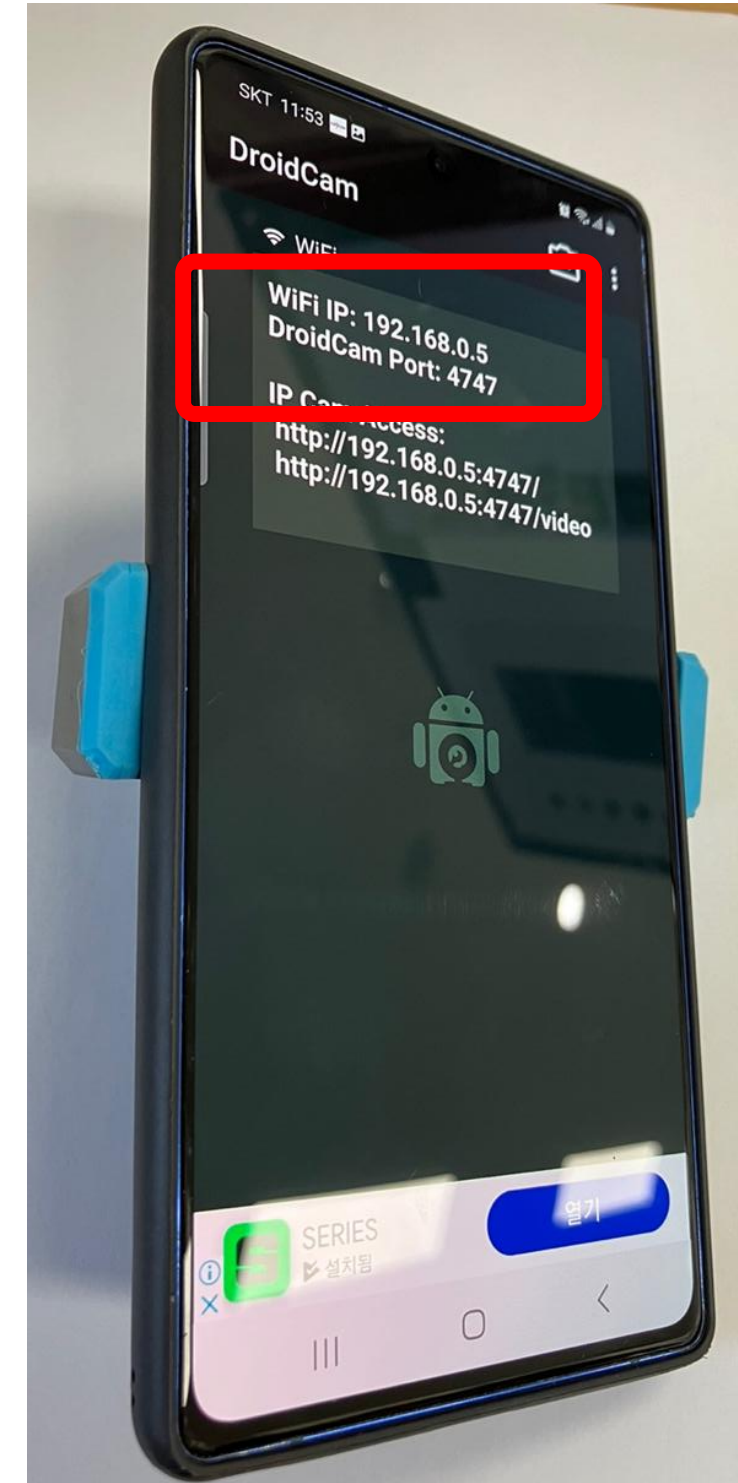


- 스마트폰과 거치대 연결하기

① 라인트레이서 프로그램을 업로드한 단비는 움직이죠?
옆으로 높게하여 놓아주세요.



② 스마트폰의 "드로이드캠" 을
켜주세요.



- PC(노트북) 파이썬 프로그램과 단비 연결하기

③ 현재파일을 ▶ 실행합니다.

The screenshot shows a Python IDE with a file explorer on the left and a terminal at the bottom. The code in the editor includes:

```
ip_address = '192.168.0.5' # DroidCam에서 확인한 IP 주소
port = 47471 # DroidCam에서 사용하는 포트 (기본적으로 4747)
video_url = f'http://{ip_address}:{port}/video'

# OpenCV를 사용하여 스트림 열기
cap = cv2.VideoCapture(video_url)

if not cap.isOpened():
    print("스트림을 열 수 없습니다.")
    exit()

# NumPy 배열의 출력 옵션 과학적 표기법 비활성화
np.set_printoptions(suppress=True)
```

The terminal window shows the following execution logs:

```
0.004114
'2' 명령 전송 완료
1/1 [=====] - 0s 23ms/step
etc 0.51453614
'2' 명령 전송 완료
1/1 [=====] - 0s 21ms/step
stop 0.51155376
'2' 명령 전송 완료
```

Annotations on the image include:

- ① 파이썬 파일열기 "자율주행_영상회전.py" (Python file opening "autonomous_driving_video_rotation.py")
- ② 드로이드캠의 WiFi 번호입력 (DroidCam WiFi ID input)
- ② 스마트폰과 연결 화면 확인하기 (Check connection screen with smartphone)

- PC(노트북) 파이썬 프로그램과 단비 연결하기

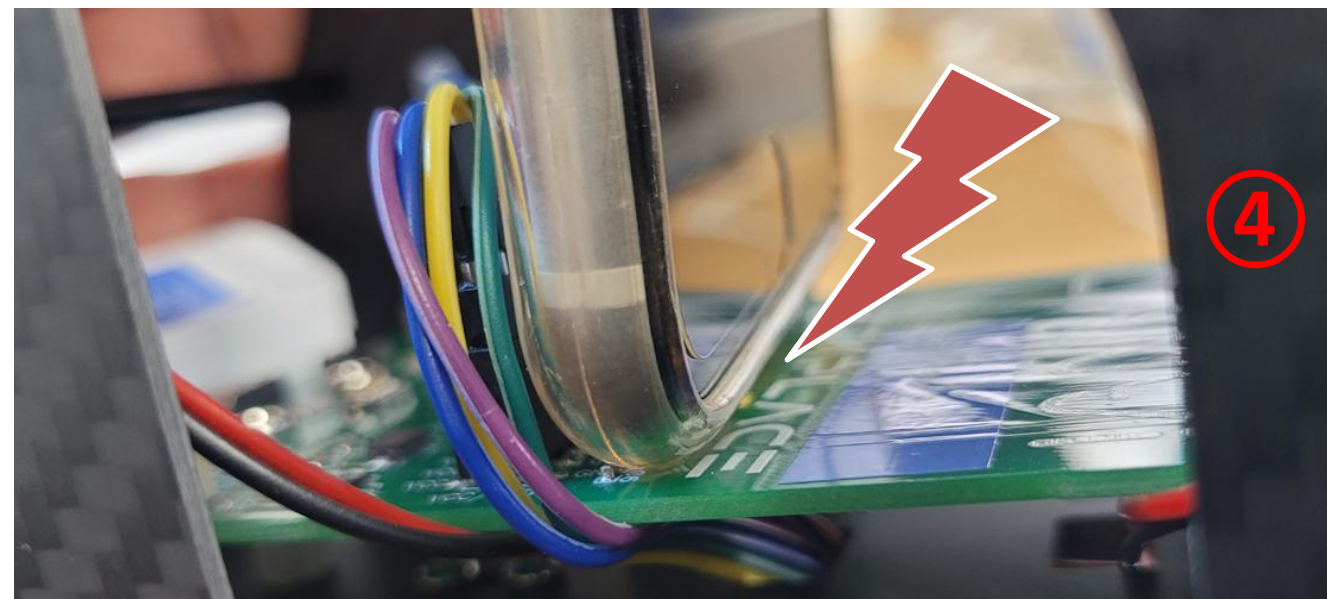
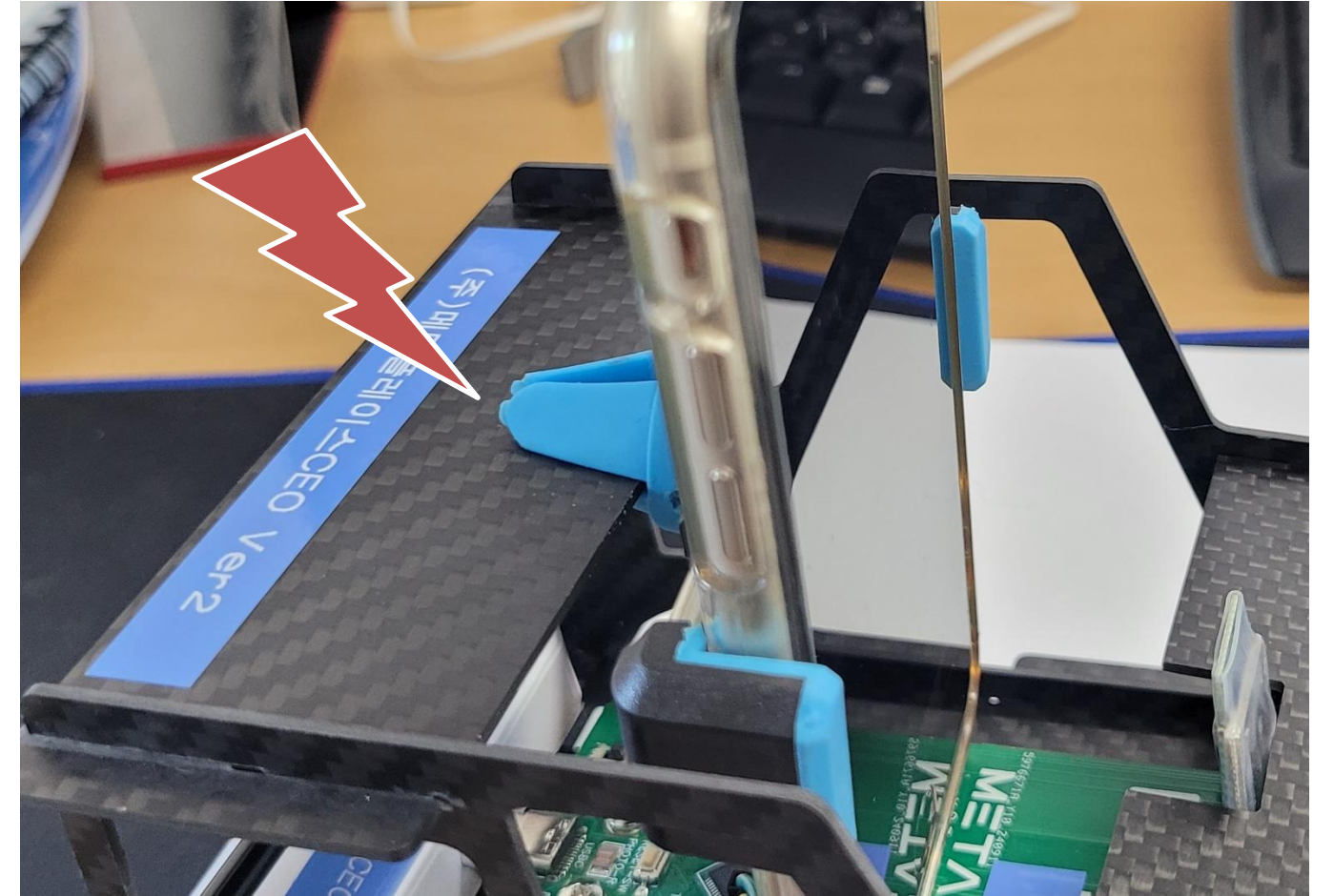
① 거치대와 스마트폰을 준비



② 결합하기

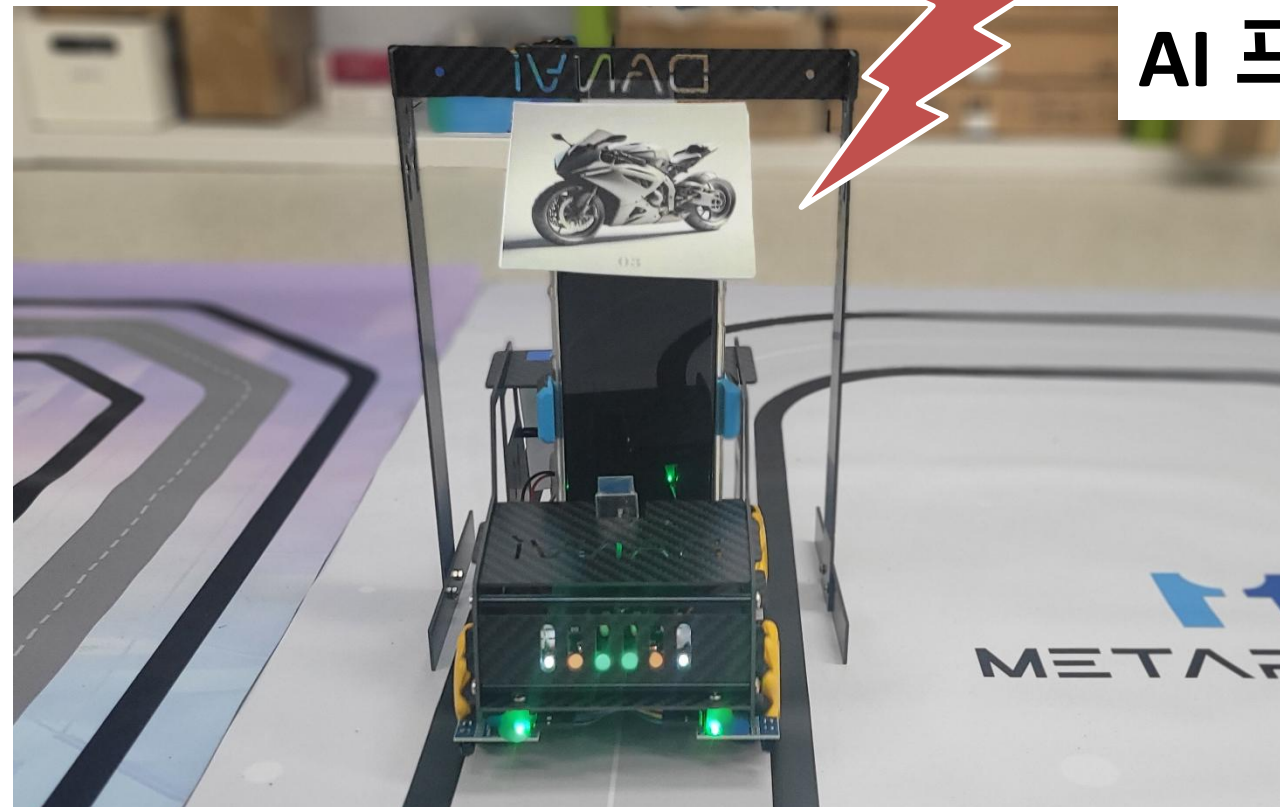
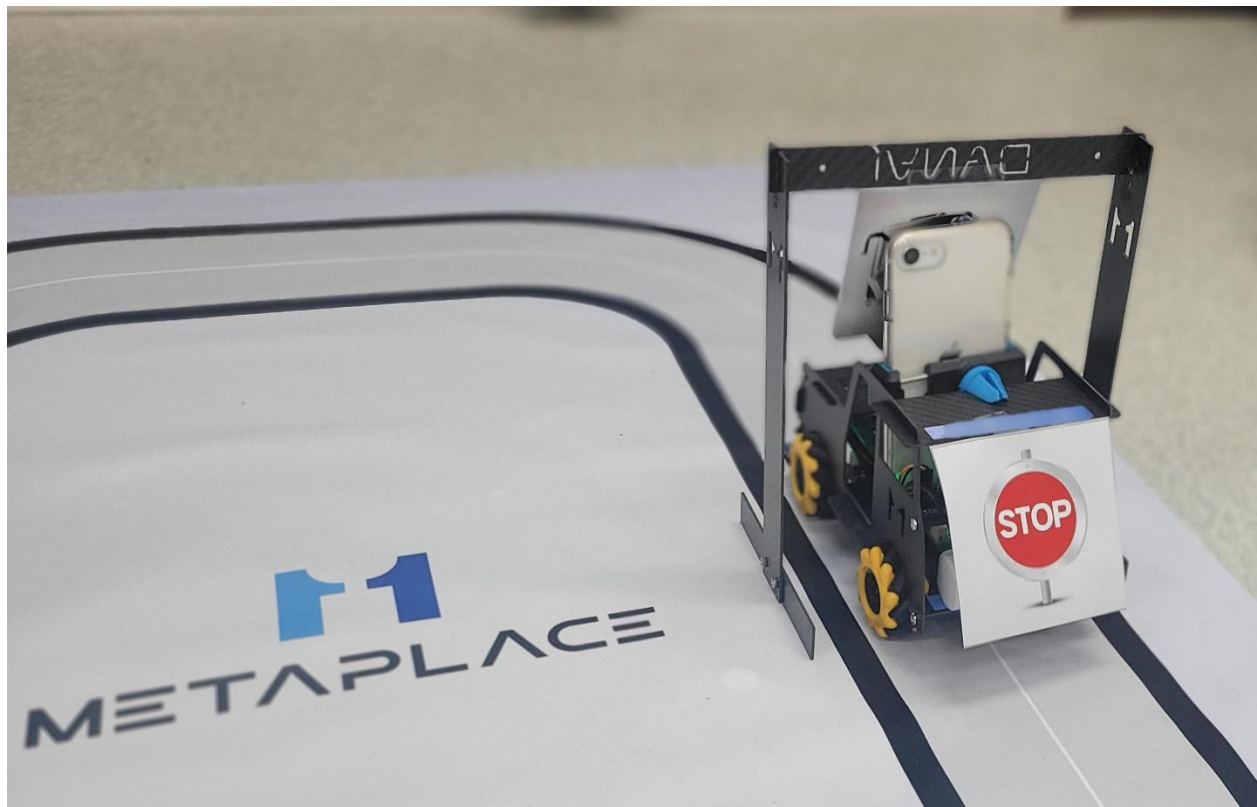
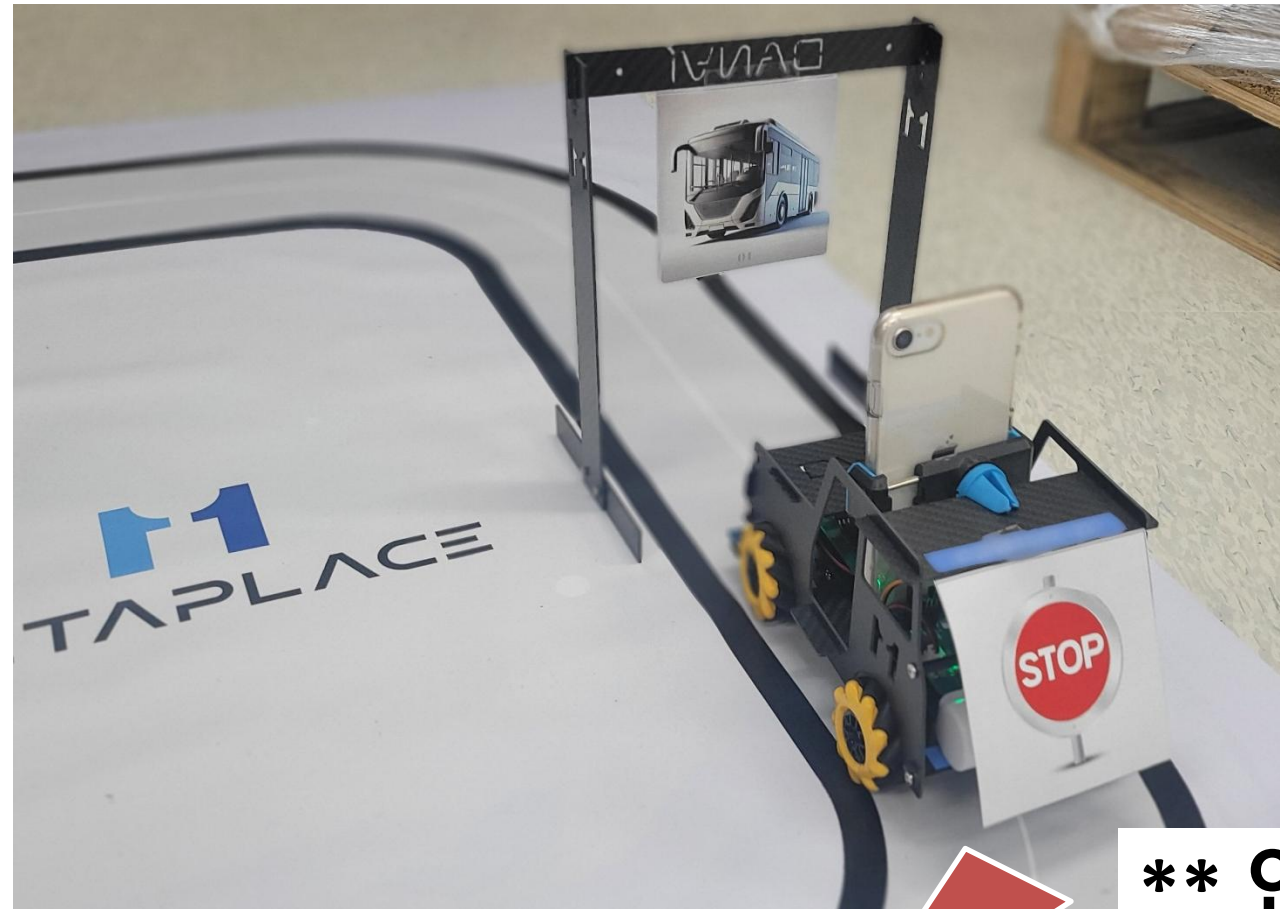


③ 보드에 닿지않게 단비에 끼워주기



④ 보드에 닿지않도록 조심하면 좋아요!

[단계5] 자율주행로봇 단비를 전용매트에 놓아 자율주행하도록 합니다.



**** 인공지능카드는 AI 프레임에 붙여주세요~**

- PC(노트북) 파이썬 프로그램과 단비 연결하기

The image shows a Python IDE (likely PyCharm) with a project named 'Danvi_racer'. The code editor displays a function `control_car_sync()` that processes video frames, predicts class names and confidence scores, and sends commands to a client based on these predictions. The terminal window shows the execution output, including confidence scores and command transmissions. A video window titled 'Rotated Video' shows a camera feed of a mat, with red text overlaid on it.

```
def control_car_sync():  
    # 1개의 사용 위치  
    # 10 프레임마다 예측  
    if frame_counter % 10 == 0:  
        class_name, confidence_score = load_and_predict_frame(model, frame)  
        label = f"{class_name}: {confidence_score:.2f}"  
        cv2.putText(frame, label, org=(10, 30), cv2.FONT_HERSHEY_SIMPLEX, fontScale=1, color=(0, 255, 0), thickness=2)  
  
        print(class_name[2:], confidence_score)  
  
        if (class_name[2:] in ['Class 1', 'motor', 'stop']) and (confidence_score >= 0.97):  
            print('stop')  
            send_command_sync(client, key='5')  
        else:  
            send_command_sync(client, key='2')  
  
        frame_counter += 1  
  
    # 자원 해제  
    cap.release()  
    cv2.destroyAllWindows()  
  
finally:  
    # 주석 기 마다 연결 해제
```

Terminal Output:

```
'2' 명령 전송 완료  
1/1 [=====] - 0s 21ms/step  
etc 0.94031113  
'2' 명령 전송 완료  
1/1 [=====] - 0s 20ms/step  
etc 0.92046314  
'2' 명령 전송 완료  
1/1 [=====] - 0s 21ms/step  
etc 0.9273797  
'2' 명령 전송 완료  
1/1 [=====] - 0s 24ms/step  
etc 0.9354396  
'2' 명령 전송 완료
```

Rotated Video Window Text:

② 자율주행 매트에서 보이는 영상을 확인하기

① 영상에서 보이는 인식율을 확인해봅시다!

[단계5] 자율주행하는 AI로봇의 영상

```
def control_car_sync():
    # 10 프레임마다 예측
    if frame_counter % 10 == 0:
        class_name, confidence_score = load_and_predict_frame(model, frame)
        label = f"{class_name}: {confidence_score:.2f}"
        cv2.putText(frame, label, org=(10, 30), cv2.FONT_HERSHEY_SIMPLEX, fontScale=1, color=(0, 255, 0), thickness=2)

        print(class_name[2:], confidence_score)

        if (class_name[2:] in ['Class 1', 'motor', 'stop']) and (confidence_score >= 0.97):
            print('stop')
            send_command_sync(client, key='5')
        else:
            send_command_sync(client, key='2')


    frame_counter += 1

# 자원 해제
cap.release()
cv2.destroyAllWindows()

finally:
    # 주석 처리된 코드가 있음
```

실행 콘솔:

```
'5' 명령 전송 완료
1/1 [=====] - 0s 23ms/step
motor 0.9999566
stop
'5' 명령 전송 완료
1/1 [=====] - 0s 22ms/step
motor 0.99995756
stop
'5' 명령 전송 완료
1/1 [=====] - 0s 22ms/step
motor 0.99996185
stop
'5' 명령 전송 완료
```



① motor 클래스가 인식되고, 장애물을 발견하였으니 단비로봇은 "5번" 중지합니다.

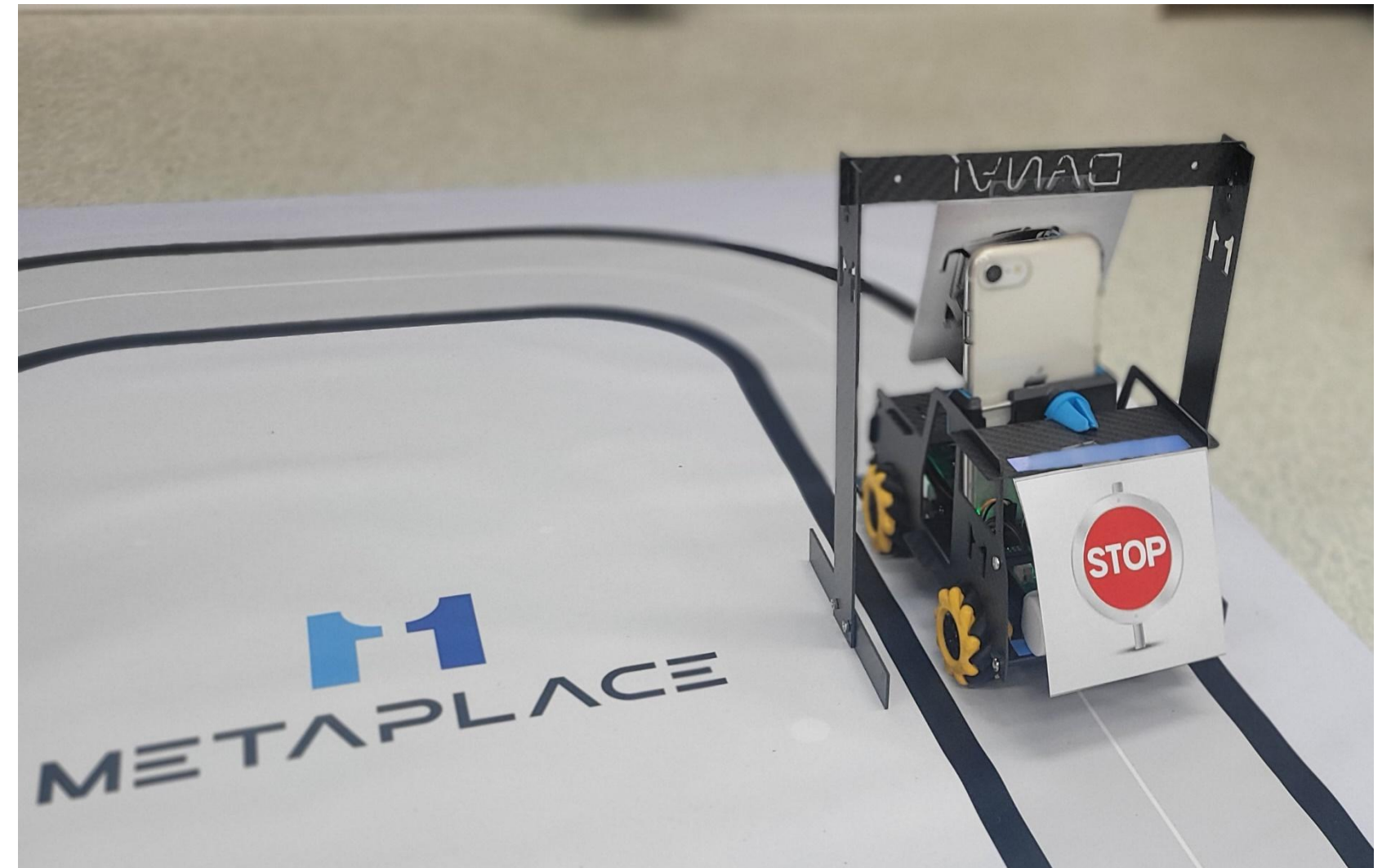
② motor 이미지 앞에서 단비가 멈추는지 확인하세요!

[단계5] 자율주행로봇 단비를 전용매트에 놓아 자율주행하도록 합니다.

① 적당한 거리에서 멈추었나요?



① "2"번 키보드를 눌러
앞으로 직진하도록 하세요!



[단계5] 자율주행하는 AI로봇의 영상

The image shows a Python IDE interface with a code editor and a terminal. The code editor displays a function `def control_car_sync():` which handles car control logic based on frame confidence scores. The terminal shows the execution output, including confidence scores and command transmissions.

```
def control_car_sync():
    # 1개의 사용 위치
    # 10 프레임마다 예측
    if frame_counter % 10 == 0:
        class_name, confidence_score = load_and_predict_frame(model, frame)
        label = f"{class_name}: {confidence_score:.2f}"
        cv2.putText(frame, label, org=(10, 30), cv2.FONT_HERSHEY_SIMPLEX, fontScale=1, color=(0, 255, 0), thickness=2)

        print(class_name[2:], confidence_score)

        if (class_name[2:] in ['Class 1', 'motor', 'stop']) and (confidence_score >= 0.97):
            print('stop')
            send_command_sync(client, key='5')
        else:
            send_command_sync(client, key='2')

    frame_counter += 1

# 자원 해제
cap.release()
cv2.destroyAllWindows()

finally:
    # 주석 기 마다 여겨 해피
```

Terminal Output:

```
'2' 명령 전송 완료
1/1 [=====] - 0s 20ms/step
etc 0.9756301
'2' 명령 전송 완료
1/1 [=====] - 0s 20ms/step
etc 0.9725186
'2' 명령 전송 완료
1/1 [=====] - 0s 20ms/step
etc 0.99875283
'2' 명령 전송 완료
1/1 [=====] - 0s 20ms/step
etc 0.99887496
'2' 명령 전송 완료
```

The video window titled "Rotated Video" shows a white bus driving on a road, with a vertical timestamp on the left side: "JroidCam Video Feed - 25-Oct-2024 09:47:05 AM".

[단계5] 자율주행하는 시로봇의 영상

The image shows a Python IDE window with a project named 'Danvi_racer'. The code in the editor defines a function `control_car_sync()` that processes video frames to detect a motor and issue commands. The console shows the execution output, including confidence scores and commands sent to the robot. To the right, a 'Rotated Video' window displays a camera feed of a white robot car.

```
def control_car_sync():
    # 1개의 사용 위치
    # 10 프레임마다 예측
    if frame_counter % 10 == 0:
        class_name, confidence_score = load_and_predict_frame(model, frame)
        label = f"{class_name}: {confidence_score:.2f}"
        cv2.putText(frame, label, org=(10, 30), cv2.FONT_HERSHEY_SIMPLEX, fontScale=1, color=(0, 255, 0), thickness=2)

        print(class_name[2:], confidence_score)

        if (class_name[2:] in ['class 1', 'motor', 'stop']) and (confidence_score >= 0.97):
            print('stop')
            send_command_sync(client, key='5')
        else:
            send_command_sync(client, key='2')

    frame_counter += 1

# 자원 해제
cap.release()
cv2.destroyAllWindows()

finally:
    # 주석 시 이 코드를 해제
```

실행 결과:

```
'5' 명령 전송 완료
1/1 [=====] - 0s 22ms/step
motor 0.99795234
stop
'5' 명령 전송 완료
1/1 [=====] - 0s 20ms/step
motor 0.9981505
stop
'5' 명령 전송 완료
1/1 [=====] - 0s 24ms/step
motor 0.9978085
stop
'5' 명령 전송 완료
```

[단계5] 자율주행하는 AI로봇의 영상

The image shows a code editor window with a Python script named `control_car_sync()`. The code is designed to process video frames, detect objects, and send commands to a car. It includes a loop that checks for objects every 10 frames, prints their class names and confidence scores, and sends 'stop' or '2' commands based on the detected class and confidence level.

```
def control_car_sync():
    # 1개의 사용 위치
    # 10 프레임마다 예측
    if frame_counter % 10 == 0:
        class_name, confidence_score = load_and_predict_frame(model, frame)
        label = f"{class_name}: {confidence_score:.2f}"
        cv2.putText(frame, label, org=(10, 30), cv2.FONT_HERSHEY_SIMPLEX, fontScale=1, color=(0, 255, 0), thickness=2)

        print(class_name[2:], confidence_score)

        if (class_name[2:] in ['Class 1', 'motor', 'stop']) and (confidence_score >= 0.97):
            print('stop')
            send_command_sync(client, key='5')
        else:
            send_command_sync(client, key='2')

    frame_counter += 1

# 자원 해제
cap.release()
cv2.destroyAllWindows()

finally:
    # 조금 더 기다려 줘야 함
```

The terminal window below the code editor shows the execution output, indicating that the '5' command is being sent successfully and the car is stopping. The output includes the following lines:

```
'5' 명령 전송 완료
1/1 [=====] - 0s 25ms/step
stop 0.9999943
stop
'5' 명령 전송 완료
1/1 [=====] - 0s 22ms/step
stop 0.99999416
stop
'5' 명령 전송 완료
1/1 [=====] - 0s 21ms/step
stop 0.999995
stop
'5' 명령 전송 완료
```

The video window on the right shows a rotated video feed of a red prohibition sign (a red circle with a diagonal slash) on a white background, which is the output of the car's camera.

메카넘휠 AI 로봇 "단비"

Mecanum Wheel Robot AI DanVI

THANK YOU

궁금한 부분이 있으시면 아래의 메일로 연락주세요.

E-mail : metaplace@naver.com

Tel : 033-252-4787

