

# AutoCar3 시작하기

AutoCar3는 한백전자에서 개발한 머신러닝 기반 자율주행차 실습장비입니다.

사용자가 머신러닝을 접목한 자율주행 기술 습득에 집중할 수 있도록 한백전자에서 개발한 Pop을 비롯해 다양한 머신러닝 모델 및 라이브러리가 내장되어 있습니다.

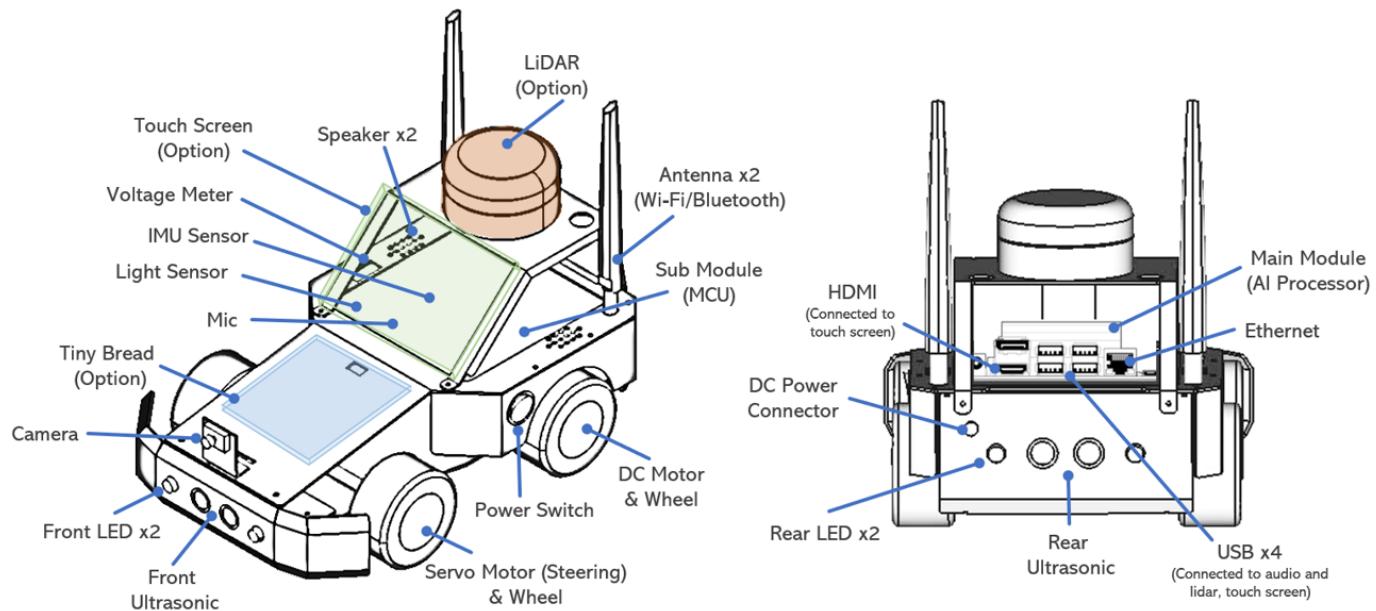
## AutoCar3 구조

AutoCar3에는 바퀴나, 모터, 프레임과 같은 기구물 외에 많은 전자장치가 포함되어 있습니다.

- 전원부: 동작에 필요한 전원을 안정적으로 공급
- 서브 모듈: 각종 센서와 액추에이터를 직접 제어
  - DC모터, 서보 모터, 부저, LED, 초음파 센서, IMU 센서, 빛 센서 제어
  - ST마이크로일렉트로닉스의 고성능 MCU(STM32F4) 사용. FreeRTOS로 운영
- 메인 모듈: 서브 모듈에 제어 명령을 내리고 인공지능 학습 및 예측 실행
  - 카메라, 오디오, 커넥티비티(이더넷, Wi-Fi, 블루투스), 라이다(옵션), 터치 스크린(옵션) 제어
  - 엔비디아의 Jetson Nano(또는 Xavier NX) 사용. 우분투 리눅스로 운영
- 모듈간 통신: 서브 모듈과 메인 모듈은 안정적인 CAN을 통해 연결됨

### CAN(Controller Area Network)

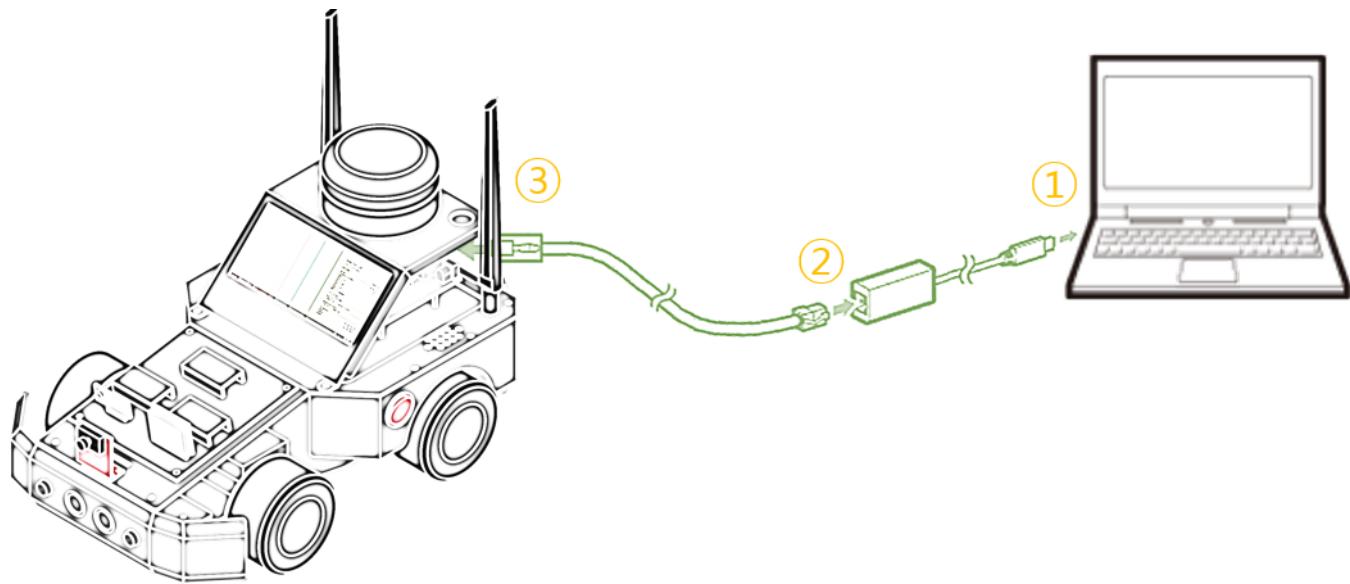
여러 컨트롤러 사이 간편하게 데이터를 교환하는 버스 시스템으로 잡음에 강해 자동차에서 주로 사용



## AutoCar3 구조

제공되는 USB 이더넷 어댑터를 PC에 연결 ① 한 후 이더넷 케이블로 양쪽 이더넷 포트를 연결 ②,③ 합니다.

PC가 랩톱이라면 이더넷 포트가 없는 경우가 많고, 데스크탑이라면 인터넷에 연결되어 있으므로 별도의 USB 타입 이더넷 어댑터 제공

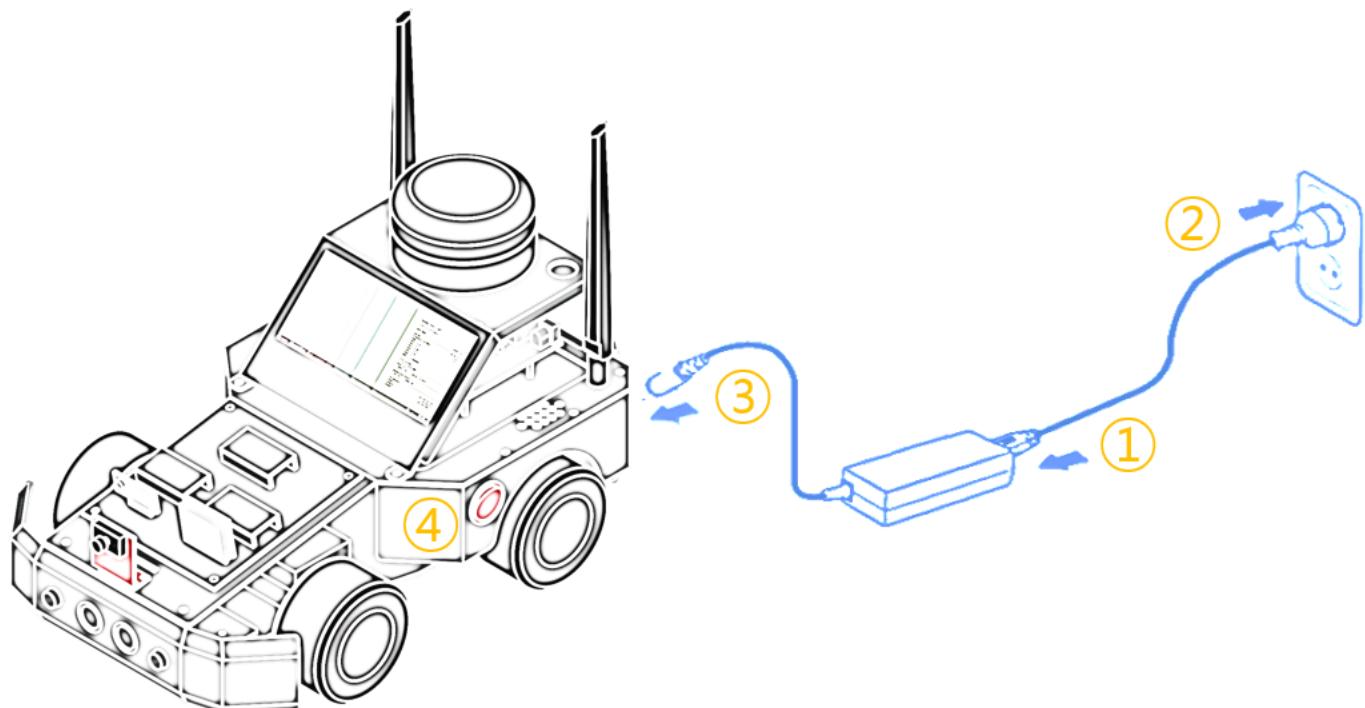


## AutoCar3 부팅

처음 시작하거나 내장된 배터리 전원이 충분치 않으면 전원 어댑터에 콘센트 연결선 ①을 꼽고 콘센트에 연결 ②합니다.

반대쪽을 AutoCar3의 전원 포트 ③에 연결한 후 전원 스위치를 ON ④ 시킵니다.

AutoCar3 전원 포트 쪽은 가볍게 밀어 넣으면, 전원이 켜지면 멈출 것. (얇게 꼽힘)



AutoCar3에 전원이 공급되면 메인 모듈은 범용 리눅스 운영체제를, 서브 모듈은 FreeRTOS로 구현한 펌웨어를 시작합니다.

전원이 공급된 후 약 5초 정도 지나면 서브 모듈에서 짧은 부저 음이 울리는데, 이후부터 CAN을 통해 다음 기능을 사용할 수 있습니다.

- IMU 센서값(가속도, 각속도, 지자기, 오일러, 쿼터니언) 읽기
- 앞/뒤 초음파 센서값(장애물 거리) 읽기
- 빛 센서값 읽기

- 앞/뒤 LED 켜고 끄기
- 뒤 좌/우 DC 모터 구동
- 앞 서보 모터(스티어링) 구동

메인 모듈의 리눅스는 약 1분정도 더 기다려야 사용할 수 있으며 주요 기능은 다음과 같습니다.

리눅스는 윈도우처럼 커널을 비롯해 많은 디바이스 드라이버의 초기화를 진행해 더 많은 시간 소요.

- **우분투** 리눅스를 수정해 인공지능 자율주행 교육에 최적화한 환경 제공
  - zsh(Oh-my-zsh 포함)과 tmux로 현대적인 CLI 환경 사용
  - SSH Server, Jupyter Server 활성화
- 엔비디아의 임베디드 GPU 개발환경인 **JetPack(CUDA, CuDNN, TensorRT, Tensorflow, Pytorch 등)** 포함
  - 객체 분류 및 탐지를 위해 사전 학습된 ResNet, Yolo 모델 제공
  - Pop.AI를 통해 좀 더 쉬운 머신러닝 실습 환경 제공
- 좀 더 쉽게 AutoCar3를 제어하고 간편하게 인공지능 기술을 자율주행에 적용하는 **Pop 라이브러리** 포함
  - Pop을 통해 서브 모듈 기능 사용
  - numpy, matplotlib, pyqt5, opencv-python 등 파이썬 라이브러리 사전 설치

## AutoCar3 전원

AutoCar3의 전자 장치는 외부 전원과 배터리로 움직이며, 외부 전원을 연결하면 배터리 충전과 전자 장치의 전원 공급이 동시에 이뤄집니다.

- 외부 전원
  - 어댑터로 220V AC를 19V DC로 변환해 AutoCar3에 공급
  - AutoCar3의 전원부는 이를 각 부품이 요구하는 전압으로 변환해 공급
    - 충전 회로에는 배터리 충전 전압인 16.8V로 변환해 전달
- 배터리
  - 외부 전원을 연결하지 않으면 배터리 전원 사용
    - 외부 전원을 연결하면 배터리는 충전 상태로 바뀌고, 외부 전원 사용
  - 충전 전압, 충전 종지 전압: 16.8V
  - 정격 전압: 14.8V (정격 용량 7000mA)
  - 방전 종지 전압: 10.6V

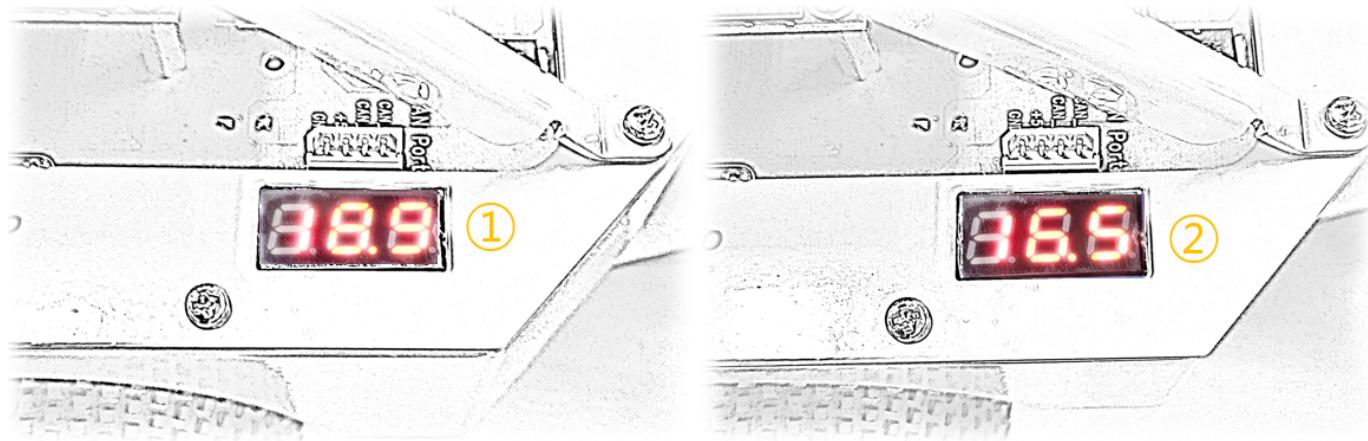
충전 전압은 배터리를 충전하는데 필요한 전압, 충전 종지 전압은 배터리가 완충 되었을 때 전압.

정격 전압은 배터리를 처음 생산해 출고할 때 전압, AutoCar3는 메인 모듈의 소모 전력이 높아 이 값보다 작으면 배터리 충전 요구.

방전 종지 전압은 충전에 필요한 최소 전압으로 이보다 작으면 더 이상 충전되지 않으므로 주의해야 함.

AutoCar3에는 사용자가 현재 전압을 확인할 수 있도록 외부 전원 ① 또는 배터리 전원 ②에 대한 전압 상태를 표시합니다.

AutoCar3에 연결된 어댑터 및 배터리 출고값에 따라 0.1V ~ 0.3 정도의 편차 발생



## 배터리 방전 주의

배터리를 사용할 때 정격 전압 이하가 되면 방전 경고음이 울립니다.

이때부터는 가급적 AutoCar3의 사용을 멈추고 외부 전원을 연결해 배터리를 충전합니다.

- 전원 스위치를 OFF 시키고, 전원 어댑터를 AutoCar3에 연결한 후 2시간 이상 충전할 것
- 완충까지는 5시간 이상 소요 (외부 온도에 따라 다름)

## PC 네트워크 설정

이더넷 케이블로 연결된 PC와 AutoCar3가 서로 통신하려면 PC의 이더넷 어댑터에 대한 TCP/IPv4 네트워크 설정이 필요합니다.

AutoCar3는 다음과 같이 사전에 네트워크 주소가 설정되어 있습니다.

- IP주소: 192.168.101.101
- 서브넷 마스크: 255.255.255.0

따라서 PC의 네트워크 설정은 다음과 같아야 합니다. (단, IP주소는 변경 가능)

- IP주소: 192.168.101.120
- 서브넷 마스크: 255.255.255.0

마스크 비트의 일종인 서브넷 마스크는 IP주소와 비트 AND 연산한 결과를 로컬과 리모트 네트워크 구분 사용.

로컬 네트워크는 직접, 리모트 네트워크는 게이트웨이를 경유해 AutoCar3에 데이터 전송.  
호스트 주소는 로컬 네트워크에서 유일해야 함.

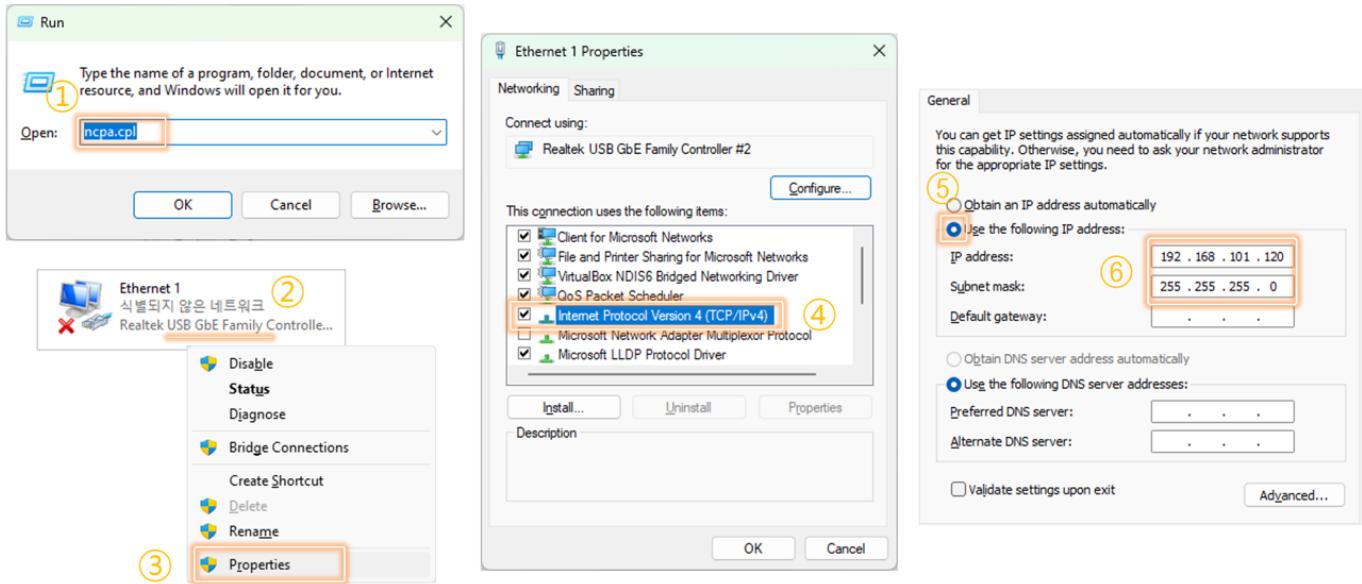
서브넷 마스크 255.255.255.0은 IP주소 앞 3자리를 네트워크 그룹으로, 나머지 1자리를 고유한 호스트 주소로 구분합니다.

- 네트워크 그룹: 192.168.101.0
- 호스트 주소: PC는 120, AutoCar3는 101
  - 0, 255는 예약되어 있고, 1(또는 254)은 로컬에서 리모트 네트워크를 연결하는 게이트웨이의 IP주소로 사용합니다.

PC에 연결된 USB 이더넷 어댑터의 네트워크 설정은 다음과 같습니다.

- 실행창(윈도우+R)에서 **ncpa.cpl** ① 명령으로 제어판의 네트워크 설정 애플릿을 실행합니다.
- AutoCar3와 연결된 USB 이더넷 어댑터 ②의 속성 ③을 선택합니다.

- 인터넷 프로토콜 버전 4(TCP/IPv4) ④ 를 더블클릭한 후 다음 IP주소 사용 ⑤ 으로 IP주소와 서브넷 마스크 ⑥ 를 입력합니다.



PC 네트워크 설정이 끝나면 통신이 가능한지 확인하기 위해 명령 프롬프트를 실행한 후 **ping** 명령으로 AutoCar3의 응답을 확인합니다.

```
ping 192.168.101.101
```

통신이 가능하다면 결과는 다음과 같아야 합니다.

```
Pinging 192.168.101.101 with 32 bytes of data:
Reply from 192.168.101.101: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.101.101:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

결과가 다르면 처음부터 다시 시도해 보세요.

PC의 USB 포트에 USB 이더넷 어댑터가 잘 꽂혀 있는지 확인.  
양쪽 이더넷 포트에 이더넷 케이블이 잘 꽂혀 있는지 확인.  
PC의 USB 이더넷 어댑터에 네트워크 설정이 맞는지 확인. 간혹 이더넷 케이블 내부가 끊어진 경우가 있음. 다른 케이블 사용.

## AutoCar3 원격 접속

PC와 AutoCar3가 통신 가능한 상태라면 PC에 설치한 VSCode와 Remote SSH 확장을 통해 AutoCar3에 원격 접속할 수 있습니다.

## 연결 설정 만들기

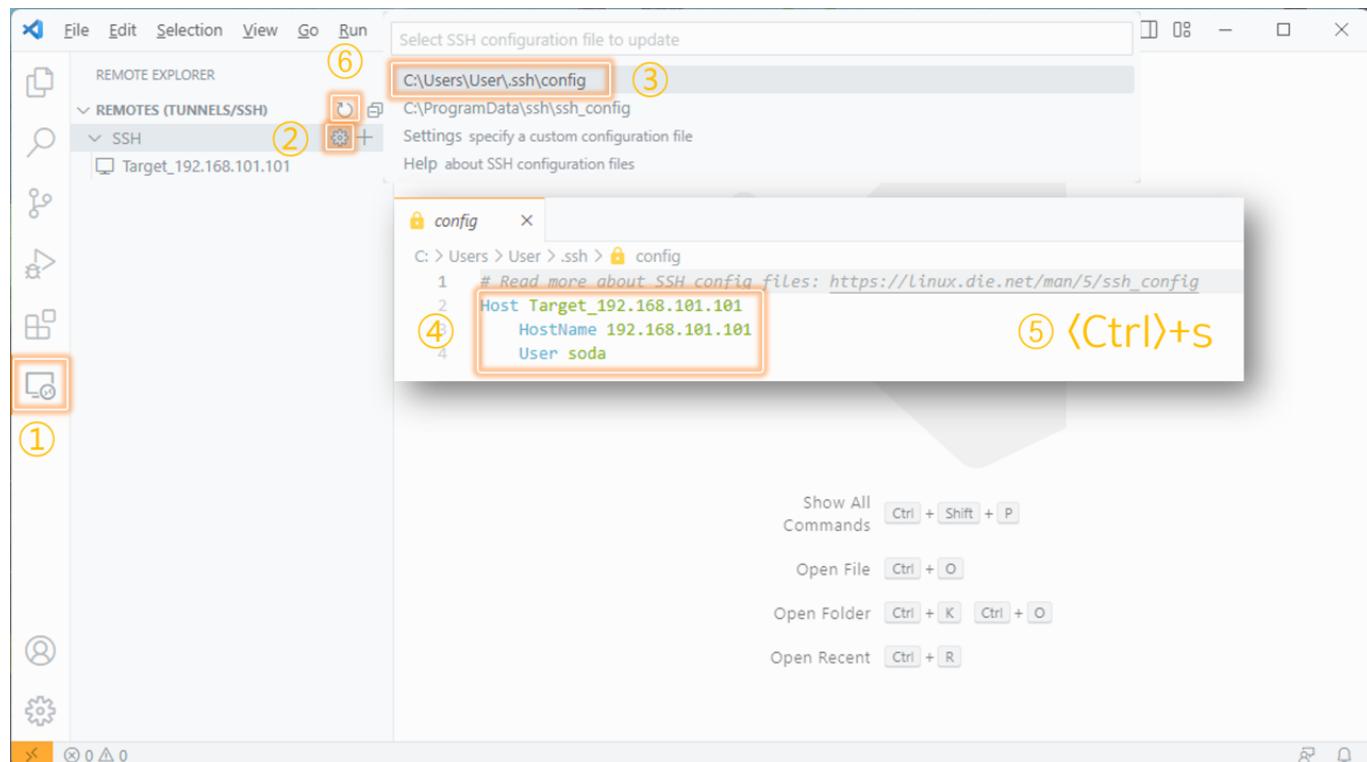
VSCode로 AutoCar3에 원격 접속하려면 AutoCar3의 IP주소와 계정 정보(IP, 패스워드)가 필요합니다. 연결할 때마다 매번 입력하는 것보다 연결 설정 파일을 만든 후 이를 이용하면 편리합니다.

AutoCar3의 기본 계정 정보는 다음과 같습니다.

- ID: soda
- 패스워드: soda

VSCode를 실행하고 왼쪽 사이드바에서 **Remote Explorer** ①을 선택한 후 다음 순서대로 AutoCar3 연결 설정을 만들습니다.

- SSH 항목으로 마우스 커서를 옮긴 후 **Open SSH Config File** ②을 누릅니다.
- 목록이 표시되면 홈 폴더 경로의 **.ssh\config** ③을 선택합니다.
- 파일이 열리면 AutoCar3 연결에 필요한 설정 ④을 입력합니다.
  - Host는 연결 이름이며, HostName은 AutoCar3의 IP 주소, User는 AutoCar3의 계정 이름입니다.
- 연결 설정 입력이 끝나면 **+s** ⑤를 눌러 파일을 저장합니다.
  - 홈 폴더 경로의 .ssh 폴더 안에 config란 이름으로 저장됨
- REMOTES 항목으로 마우스 커서를 옮긴 후 **Refresh** ⑥를 누르면 SSH 항목 아래 설정한 연결 이름이 표시됩니다.



홈 폴더(또는 디렉토리)는 사용자를 위한 전용 작업공간

원도우: C:/User/<계정 이름>  
리눅스: /home/<계정 이름>  
Mac: /Users/<계정 이름>

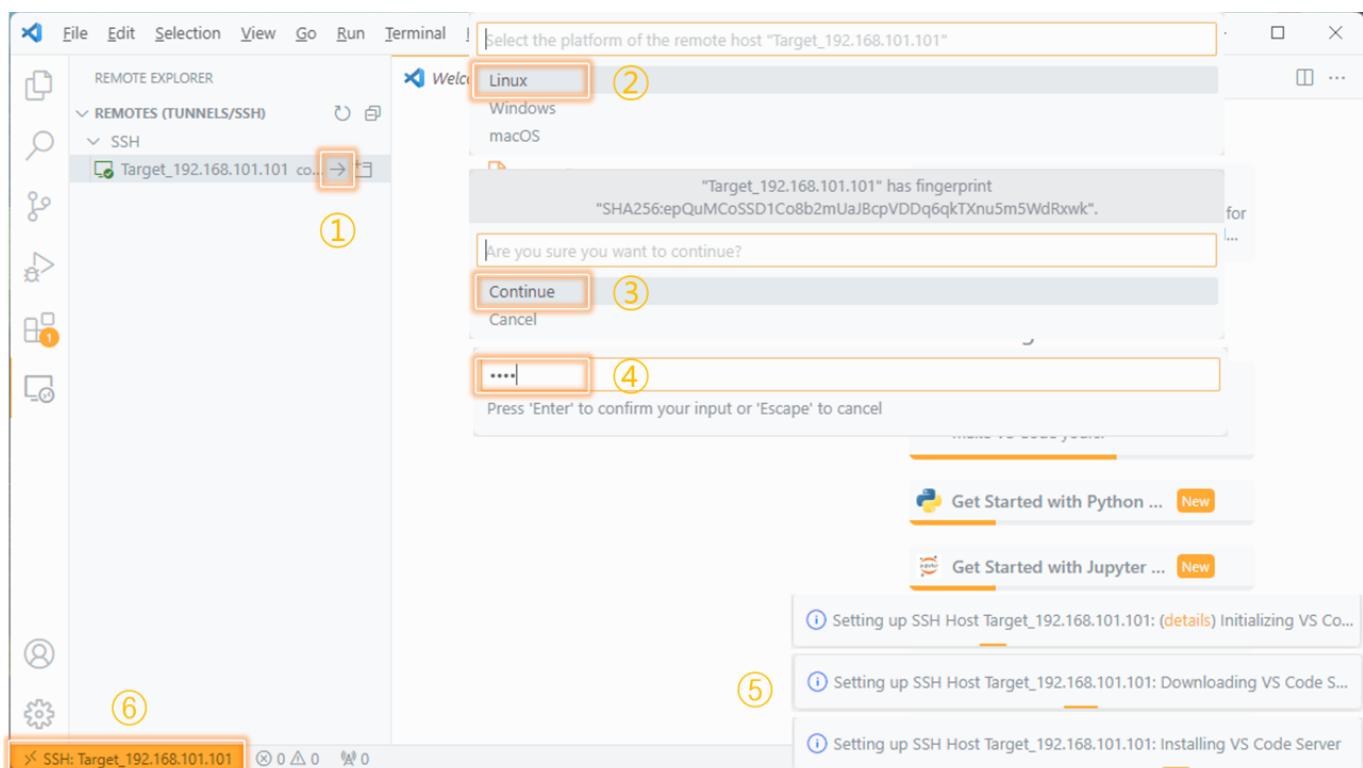
## 원격 접속

처음 AutoCar3에 원격 접속할 때는 몇 가지 절차를 더 진행하는데, VSCode는 AutoCar3에서 실행되는 **VS Code Server**가 없으면 자동으로 인터넷 저장소에서 이를 다운받아 설치합니다.

AutoCar3 접속은 사이드바의 **Remote Explorer**에서 다음과 같이 진행합니다.

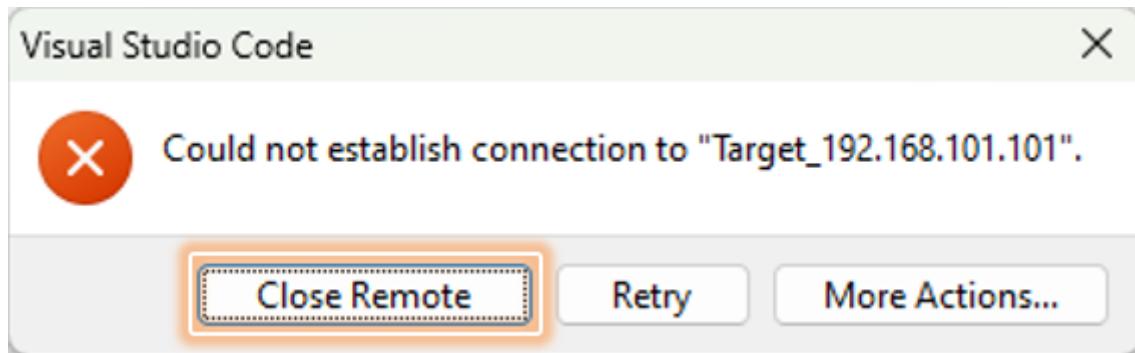
### 각 단계를 진행할 때 시간을 지체하면 연결이 실패하므로 주의할 것

- 앞서 만든 연결 이름(Target\_192.168.101.101)에서 **Connect in Current Window** ①를 선택합니다.
- 처음 연결이라면 플랫폼 선택창이 표시되는데, **Linux** ②를 선택합니다.
- 처음 연결이라면 암호화 통신에 필요한 키 교환 창이 표시되는데, **Continue** ③를 선택합니다.
  - 현재 AutoCar3 연결에 대한 키 정보는 홈 폴더의 .ssh 폴더 안에 known\_hosts 파일로 저장됩니다.
- AutoCar3 패스워드 입력창이 표시되면 **soda** ④를 입력합니다.
  - 연결 설정 파일에는 보안을 위해 AutoCar3의 soda 계정에 대한 패스워드가 저장되어 있지 않습니다.
- 처음 연결했다면 자동으로 **VS Code Server**를 다운로드한 후 설치 ⑤ 합니다.
- 연결 절차가 성공적으로 끝나면 상태 표시줄에 연결 이름 ⑥이 표시됩니다.



## 문제 해결

연결 실패 창이 표시되면 **Close Remote**를 선택한 후 처음부터 다시 시도합니다.



문제가 계속되면 다음 사항을 확인합니다.

- PC가 인터넷에 연결되어 있는가?
- PC의 USB 이더넷 어댑터의 네트워크 설정이 옳바른가?
- AutoCar3의 연결 설정이 옳바른가?
- 명령 프롬프트에서 **ping 192.168.101.101** 명령을 실행하면 AutoCar3가 정상적으로 응답하는가?
- AutoCar3가 켜져 있는가? 켜져 있다면 메인 모듈이 시작될 만큼 충분한 시간이 흘렀는가?

그래도 문제가 해결되지 않으면 다음 내용을 실행한 후 원격 접속을 다시 시도합니다.

- VSCode를 종료한 후 탐색기를 통해 VSCode 루트의 data 폴더 아래 만들어진 **user-data** 폴더를 삭제합니다.
  - user-data 폴더에는 VSCode의 실행 상태(빠른 실행을 위한 캐시 데이터) 및 사용자 환경 설정이 저장되어 있습니다.
- 홈 폴더의 .ssh 폴더 안에 들어있는 **known\_hosts**, **known\_hosts.old** 파일을 삭제합니다.
  - 앞서 진행한 접속 절차에 따라 해당 파일이 없을 수도 있습니다.

## AutoCar3 설정

AutoCar3의 메인 모듈은 우분투 리눅스로 운영되며 미리 기본적인 설정이 되어 있습니다.

### Wi-Fi로 인터넷 연결

AutoCar3는 이더넷과 Wi-Fi를 지원합니다. 이더넷은 PC와 연결되어 있으므로 인터넷 연결은 Wi-Fi를 권장합니다.

또한 주행 실습은 대부분 배터리 전원을 사용하므로 공유기가 준비되어 있다면 PC와도 Wi-Fi로 연결할 필요가 있습니다.

AutoCar3를 Wi-Fi 공유기에 연결하면 VSCode가 AutoCar3에 원격 접속되어 있어야 하고, 관리자에게 Wi-Fi 공유기 이름(SSID)과 패스워드를 확인해야 합니다.

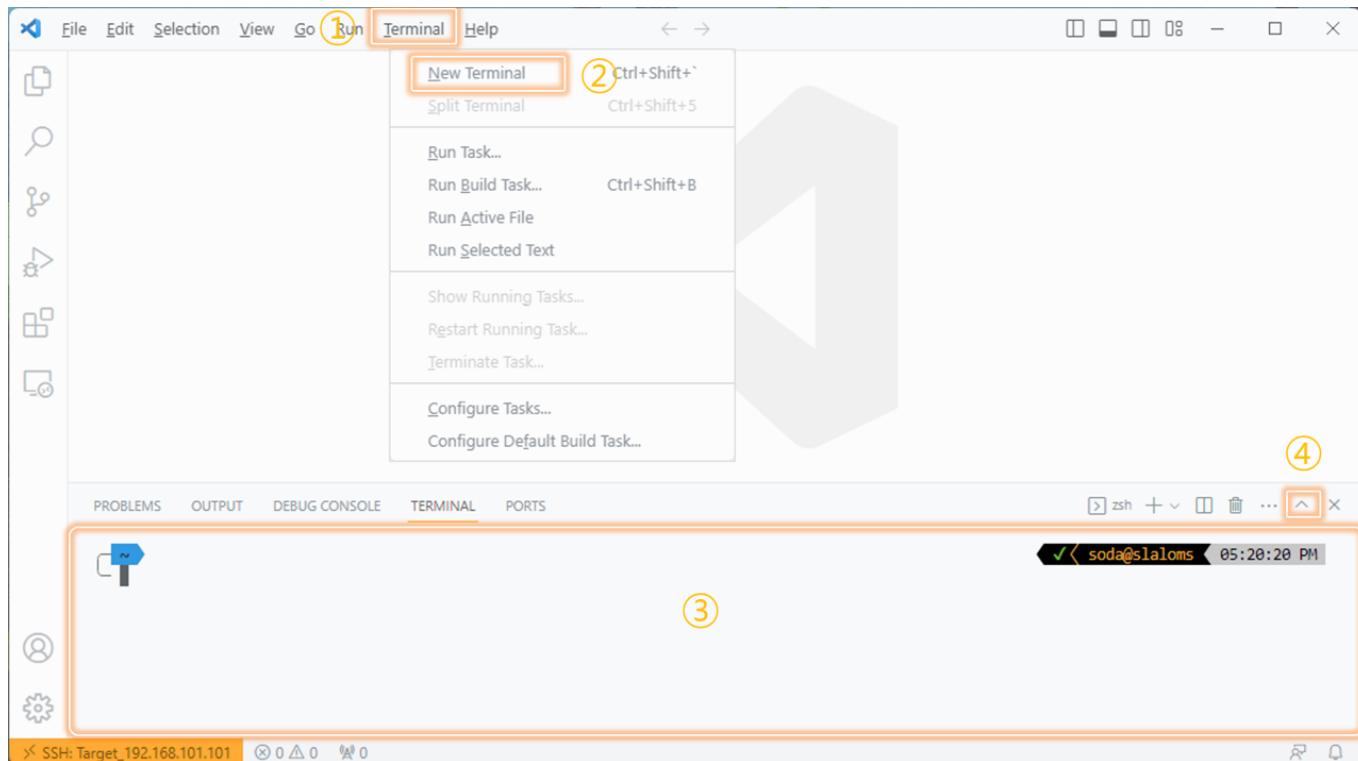
사전 준비가 끝나면 다음과 같이 AutoCar3의 리눅스 쉘을 실행합니다.

**쉘은 사용자의 표준 입력을 해석해 운영체제의 핵심 영역인 커널로 전달하고, 응답을 표준 출력으로 사용자에게 돌려줍니다.**

표준 입력은 키보드, 표준 출력은 모니터의 텍스트 모드 영역을 가리키며, 이를 명령 라인 인터페이스(CLI)라 합니다.

AutoCar3의 기본 쉘은 현대적인 zsh이며, oh-my-zsh이 함께 설정되어 있습니다.

- VSCode 메뉴의 **Terminal** ①에서 **New Terminal** ②을 선택합니다.
- 터미널창 ③이 표시되면, 리눅스 명령을 실행할 수 있습니다.
- 터미널창에 포함된 **Maximize Panel Size** ④을 선택하면 창 크기가 최대화됩니다.



네트워크 관리자인 nmcli 명령을 이용해 AutoCar3를 다음과 같이 해당 공유기에 연결합니다.

sudo는 root(관리자) 권한으로 해당 명령을 실행할 때 사용합니다.

리눅스는 시스템 설정을 변경하려면 root 권한이 필요합니다.

처음 sudo가 포함된 명령을 실행하면 패스워드를 요구하는데, soda 계정의 패스워드인 soda를 입력합니다.

- AutoCar3의 Wi-Fi 어댑터 이름은 **wlan0**이며, **device** 옵션으로 확인합니다.

```
nmcli device
```

- 연결 가능한 Wi-Fi 공유기를 찾을 때는 **wifi list** 옵션을 추가합니다.

```
nmcli device wifi list
```

- **list** 대신 **connect <이름>** 및 **password <패스워드>** 옵션으로 해당 Wi-Fi 공유기에 연결합니다.

```
sudo nmcli device wifi connect HBE_RSP password hanback91!
```

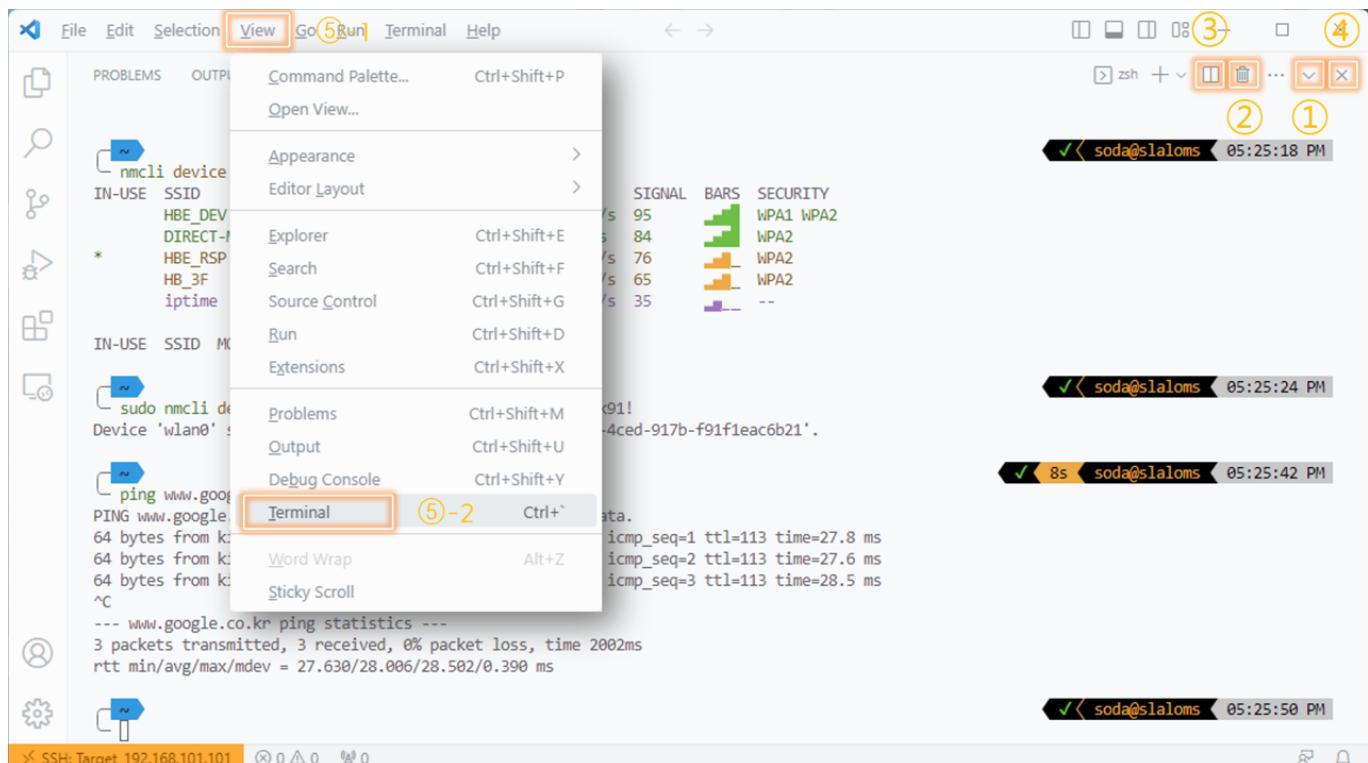
Wi-Fi 공유기 이름이 HBE\_RSP, 패스워드는 hanback91!로 가정. (연결을 완료하는데 약간의 시간 필요)

- 성공적으로 공유기에 연결되면 ping 명령으로 인터넷에 연결되는지 확인합니다.

```
ping www.google.co.kr
```

최대화된 터미널창 크기를 원래 크기로 되돌리면 **Restore Panel Size** ①를 선택합니다.

- Kill Terminal** ②은 현재 터미널을 종료합니다.
- Split Terminal** ③은 현재 터미널 창을 2개로 분할합니다.
- Close Panel** ④은 터미널을 종료하지 않고 터미널 창을 포함한 패널을 닫습니다.
  - 다시 터미널 창을 표시하려면 메뉴에서 **View > Terminal** ⑤을 선택합니다.



## 기존 패키지를 최신 버전으로 업그레이드

많은 오픈소스 패키지(라이브러리, 응용프로그램 등)들이 미리 설치되어 있지만 시간이 지나면 이들의 업그레이드가 필요합니다.

패키지 관리자인 apt 명령을 이용해 다음과 같이 설치된 패키지들의 최신 버전이 있다면 업그레이드합니다.

- 배포 서버로부터 패키지 목록 업데이트

```
sudo apt update
...
Reading package lists... Done
Building dependency tree
Reading state information... Done
1 package can be upgraded. Run 'apt list --upgradable' to see it.
```

- 패키지 목록 업데이트 과정에서 업그레이드된 패키지가 있다면 최신 버전으로 업그레이드

```
sudo apt -y upgrade
```

만약 패키지를 업그레이드할 때 마지막에 다음 오류가 표시되면 무시합니다.  
데비안 패키지와 엔비디아 L4T 패키지 사이 호환성 문제로 사용에는 문제가 없습니다.

Errors were encountered while processing:

nvidia-l4t-bootloader

E: Sub-process /usr/bin/dpkg returned an error code (1)

## 파이썬 및 주피터 확장 설치

우리의 작업은 새로운 파이썬 또는 노트북 파일을 만든 후 AutoCar3에 내장된 파이썬 인터프리터나 주피터랩의 파이썬 커널을 통해 이를 실행합니다. 이때 원격 접속한 VSCode를 위해 AutoCar3에 파이썬과 주피터 확장을 설치하면 다음과 같은 장점이 있습니다.

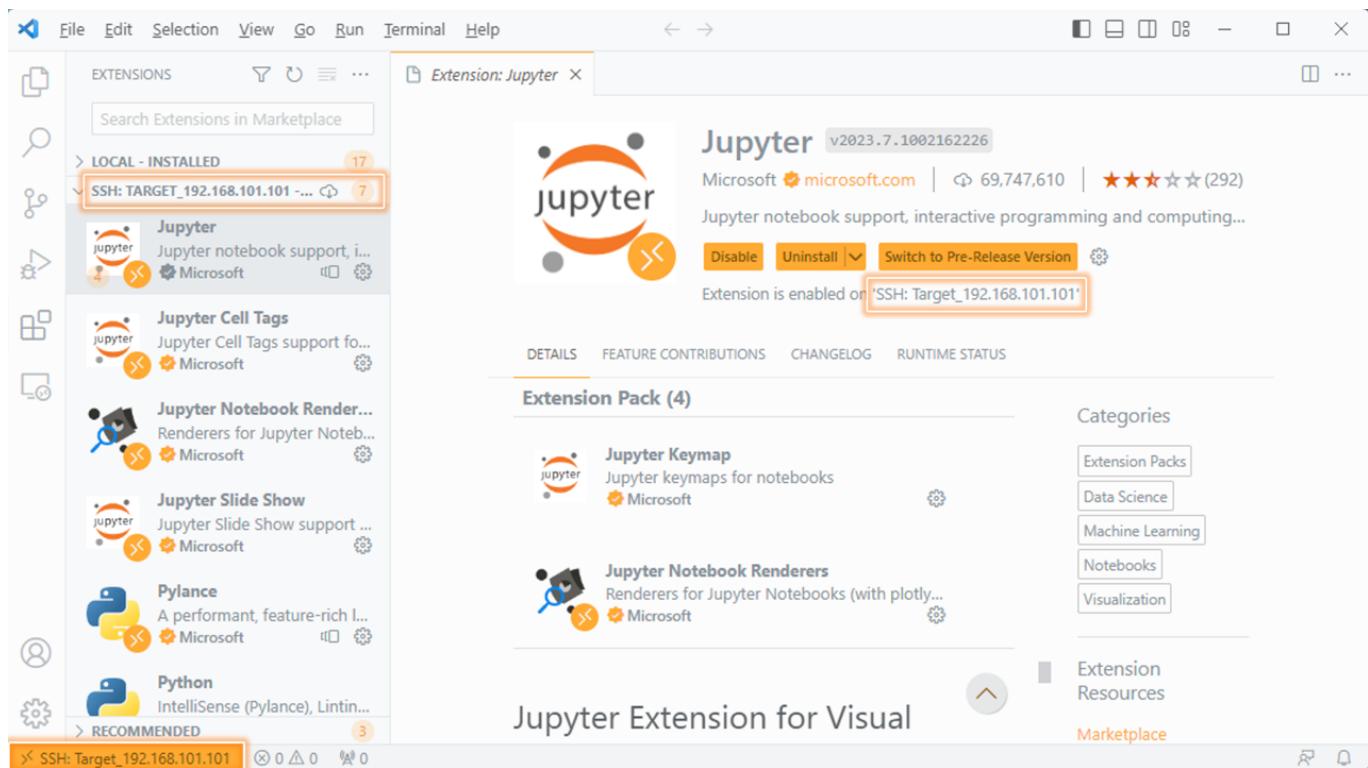
- 파이썬 확장
  - 미리 파이썬 구문의 유효성 검사(Linting) 및 필요한 키워드나 함수, 클래스 또는 메소드 추천 (IntelliSense)을 지원합니다.
  - 코드 탐색(Navigation) 및 형식 맞춤(Formatting), 구조 변경(Refactoring) 지원합니다.
  - 실행 및 실시간 디버깅 지원합니다.
- 주피터 확장
  - 셀 단위로 구문을 작성하고 실행하는 주피터 환경 지원합니다.
    - 파이썬 확장 기능 포함합니다.
    - 이미지, 멀티미디어 렌더링 지원합니다.
    - 간단한 GUI 환경인 Widgets 지원합니다.
  - 파이썬 코드에서 직접 주피터 노트북처럼 셀 단위로 구분해 실행하는 환경 지원합니다.
    - 특별한 주석(%%)으로 노트북 셀 구분합니다.

파이썬 및 주피터를 비롯해 파이썬 개발자들이 선호하는 다음 확장들을 **Extensions**의 검색 상자를 통해 설치합니다.

현재 VSCode는 AutoCar3에 원격 접속된 상태이므로 대부분의 확장들은 AutoCar3에 설치되지만 일부 확장은 PC에 설치됨

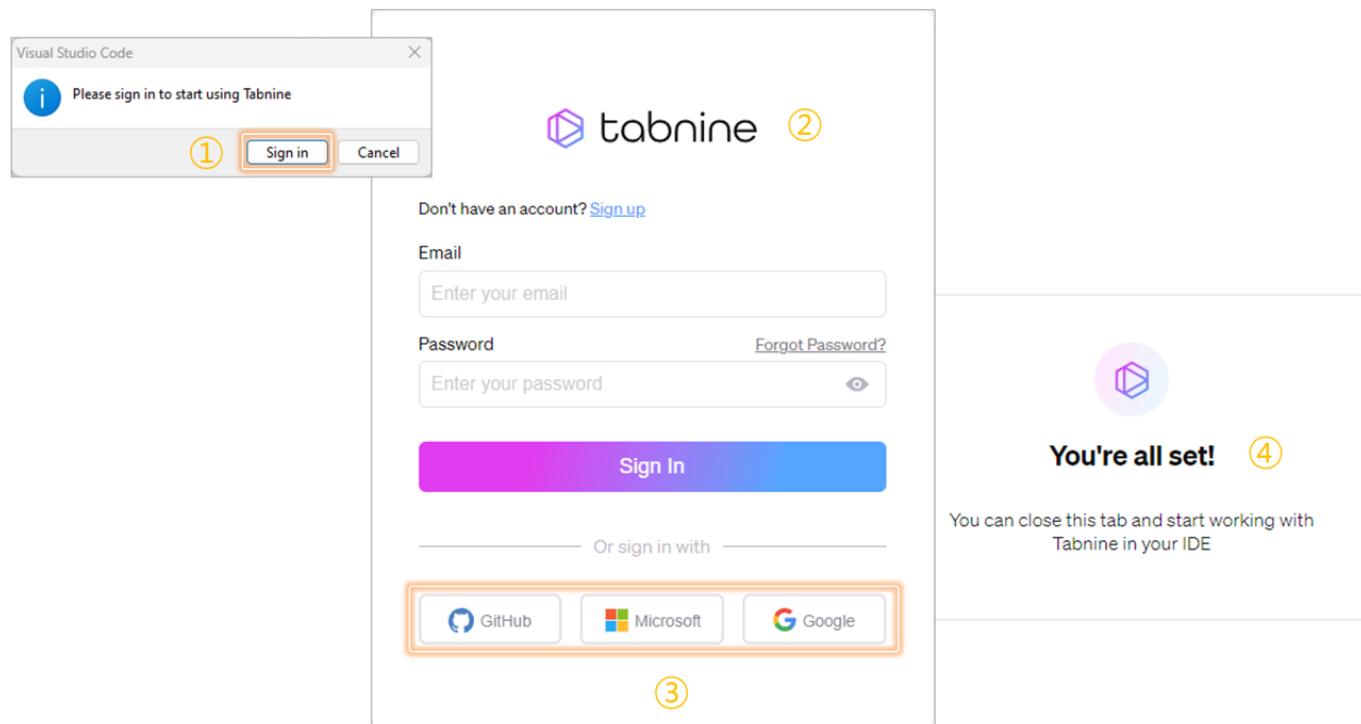
- **ms-python.python**
  - Microsoft에서 배포하는 Python 공식 확장으로 상황에 따른 구문 추천 도우미인 Pylance도 함께 설치됩니다.
- **ms-toolsai.jupyter**
  - Microsoft에서 배포하는 Jupyter 공식 확장으로 Notebook Renderers, Slide Show, Cell Tags도 함께 설치됩니다.
  - Keymap은 PC에 설치되어 주피터 노트북의 단축키 정보를 VSCode에 연결합니다.
- **TabNine.tabnine-vscode**
  - 머신러닝 기반 코드 도우미로 서명을 하면 무료 사용이 가능합니다.
- **usernamehw.errorlens**
  - PC에 설치되면, 구문 오류가 발생한 줄 옆에 자세한 설명을 표시합니다.

- **KevinRose.vsc-python-indent**
  - PC에 설치되면, 파이썬 들여쓰기 시각화를 지원합니다.
- **PKief.material-icon-theme**
  - PC에 설치되면, 인기 있는 아이콘 테마 중 하나입니다.



TabNine 확장을 설치한 후 인증 창이 표시되면 **Sign In** 버튼 ① 을 누른 후 다음과 같이 진행합니다.

- 웹 브라우저에 TabNine 인증 페이지 ② 가 표시됩니다.
- 하단에서 본인과 관계된 계정 링크 ③ 중 하나를 선택해 인증합니다.
- 선택한 계정으로 인증이 끝나면 완료 페이지 ④ 가 표시됩니다.



## VSCode 설정

VSCode는 다양한 기능을 제공하는 강력한 코드 편집기입니다. 하지만 기본 설정은 모든 사용자에게 적합하지 않을 수 있습니다. 따라서 자신의 작업 방식에 맞게 VSCode를 설정할 필요가 있습니다.

다음은 추천 설정으로 **Activity Bar > Manage > Settings**를 통해 진행합니다.

VSCode/data/user-data/User/setting.json에 자동 저장되며, 설정을 다른 사람과 공유하려면 이 파일 복사

- Commonly Used
  - **Auto Save**를 **onFocusChange**로 변경하면 창의 포커스가 바뀔 때마다 편집 중인 파일이 자동 저장됩니다.
  - **Font Family**의 입력 상자를 통해 폰트를 변경하며, 한글/영문 폰트 이름을 쉼표(,)로 구분하면 함께 사용할 수 있습니다.
    - 영문은 <https://www.nerdfonts.com/font-downloads>에서 배포하는 'JetBrainsMono Nerd Font'를 추천합니다.
    - 한글은 [https://dalseo.daegu.kr/index.do?menu\\_id=00003453](https://dalseo.daegu.kr/index.do?menu_id=00003453)에서 배포하는 DalseoHealing을 추천합니다.
- Text Editor
  - **Mouse Wheel Zoom**을 체크한 후 편집창에서 <Ctrl>을 누른 채 마우스 휠을 위/아래로 조작하면 폰트 크기가 커지거나 작아집니다.
    - 상단의 Search settings 상자에 **Mouse Wheel Zoom**를 입력하면 항목을 쉽게 찾을 수 있습니다.

## 새 프로젝트

원격 접속한 VSCode와 AutoCar3에 설치한 확장을 이용해 AutoCar3 제어용 프로젝트를 만든 후 실행해 봅니다.

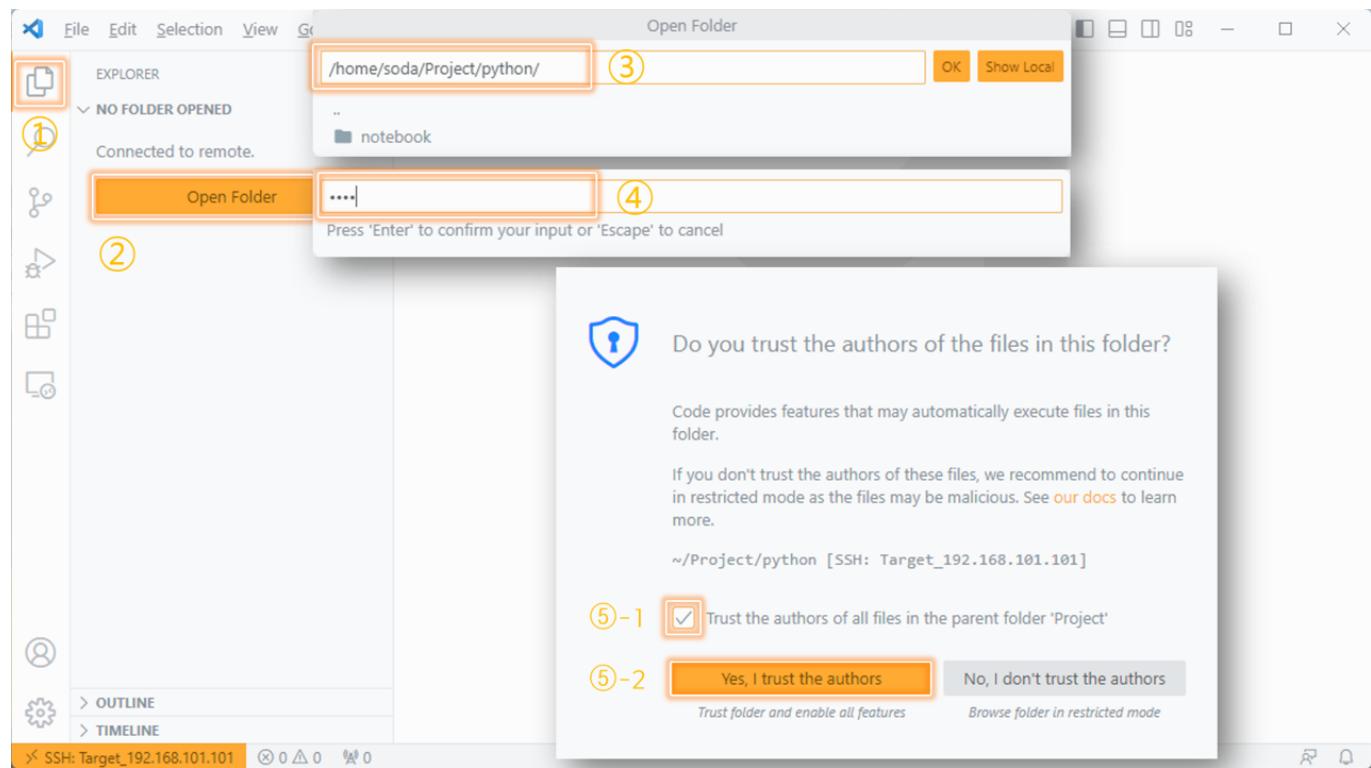
### 작업공간 선택

작업공간(Workspace)은 사용자가 프로젝트를 진행할 때 관련된 여러 소스 코드 파일과 리소스를 한데 모아 놓은 폴더입니다.

파이썬 또는 노트북 파일로 AutoCar3 제어 프로그램을 작성하려면 먼저 작업공간을 선택해야 합니다.

VSCode에서 AutoCar3의 작업공간을 선택하는 방법은 다음과 같습니다.

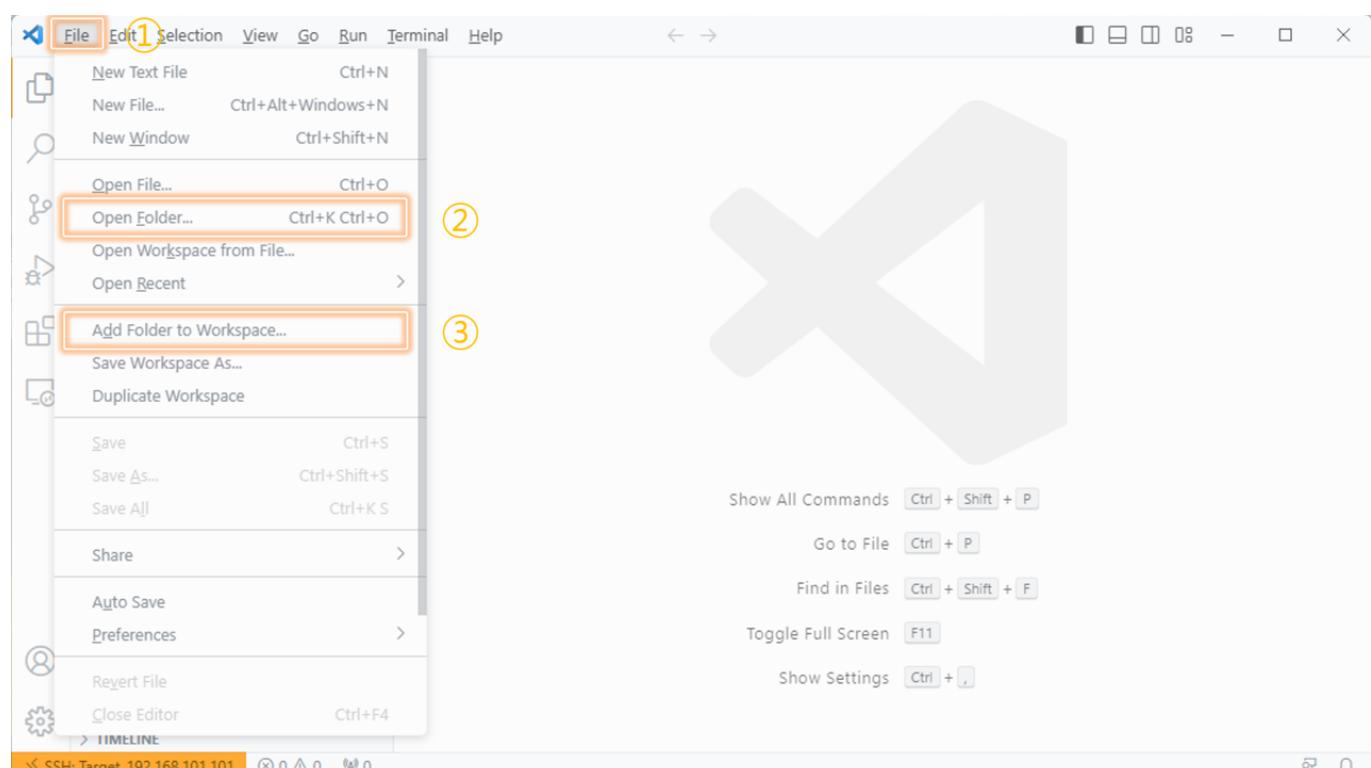
- 사이드 바에서 **Explorer** ① 를 실행합니다.
- **Open Folder** ② 를 선택합니다.
- 폴더 선택창이 표시되면 /home/soda/Project/python 폴더 ③ 를 선택합니다.
  - 현재 VSCode는 AutoCar3에 원격 접속된 상태이므로 선택 경로는 AutoCar3의 리눅스 파일시스템 경로입니다.
- 패스워드 입력창이 표시되면 **soda** ④ 를 입력합니다.
- 선택한 폴더에 대한 신뢰성 확인이 표시되면 **Trust the authors of all files**와 **Yes, I trust the authors** ⑤ 를 선택합니다.



VSCode는 종료했다가 다시 시작해도 이전 작업공간을 자동으로 선택합니다. 만약 작업공간을 변경하려면 다시 새로운 폴더를 열어야합니다.

단, 작업공간이 선택된 상태에서는 **Explorer > Open Folder**를 사용할 수 없으므로 메뉴 **File** ①에서 다음과 같이 선택합니다.

- **Open Folder** ②로 새 폴더를 선택합니다.
  - 기존 폴더를 닫고 새 폴더를 작업공간으로 사용합니다.
- **Add Folder to Workspace** ③로 새 폴더를 추가합니다.
  - 기존 폴더를 닫지 않고 새로운 폴더를 추가하는 것으로 여러 폴더를 하나의 작업공간에 둘 수 있습니다.



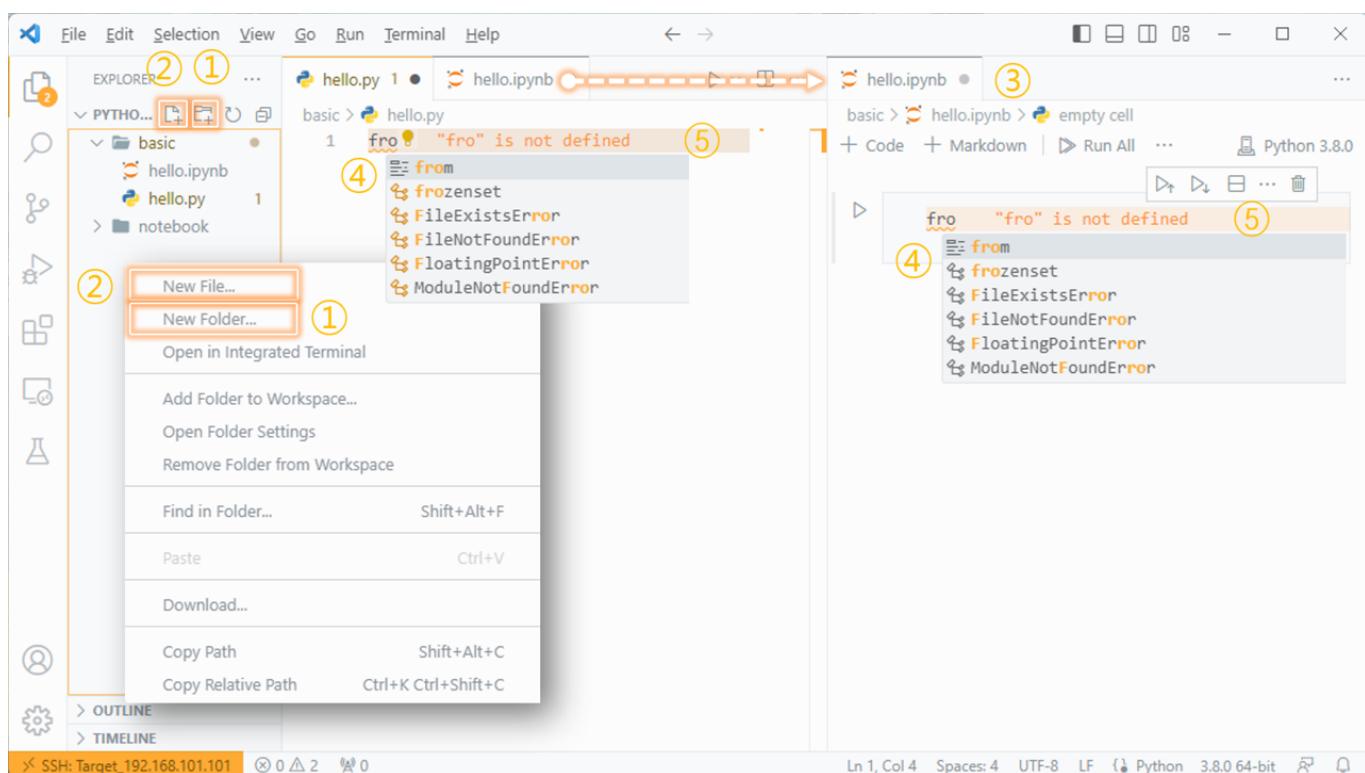
## 새 파일 만들기

작업공간이 선택되면 파이썬 코드 작성을 위해 새 파일을 만드는데, 작업공간 루트 또는 새로 만든 하위 폴더 모두 가능합니다.

확장자 **.py**는 파이썬 스크립트를, **.ipynb**는 주피터 노트북 파일을 의미합니다.

**Explorer**에서 도구 모음 또는 팝업 메뉴(마우스 오른쪽 클릭)를 통해 다음과 같이 새 파일을 만들어 봅니다.

- **New Folder**를 선택한 후 폴더 이름으로 **basic** ① 을 입력합니다.
- basic 폴더를 선택한 상태에서 **New File**을 통해 **hello.py** 와 **hello.ipynb** ② 를 차례로 만듭니다.
- 편집 영역에서 두 번째 만든 hello.ipynb의 탭을 마우스로 끌어 오른쪽 끝에 최대한 가깝게 놓으면 편집창이 분할 ③ 됩니다.
- hello.ipynb의 첫 번째 셀에 파이썬 구문을 입력하면 자동으로 코드를 추천 ④ 하며, 구문 오류가 있으면 설명 ⑤ 이 표시됩니다.
- hello.py도 동일하게 자동 코드 추천 및 구문 오류 설명 표시 기능을 사용할 수 있습니다.



편집창을 넓게 사용하기 위해 **Side bar**를 닫은 후 다음과 같이 앞서 만든 파일에 AutoCar3 주행과 관련된 구문을 추가합니다.

```
from pop.Pilot import AutoCar
from time import sleep

car = AutoCar()

car.setSpeed(80)
car.forward()
car.steering = 1.0
sleep(2)
car.backward()
car.steering = -1.0
```

```
sleep(2)
car.steering = 0.0
car.stop()
```

- 먼저 pop.Pilot 모듈의 AutoCar 클래스와 time 모듈의 sleep 함수를 로드 ① 합니다.
  - pop에 포함된 모든 모듈은 한백전자에서 제공하며, time 모듈은 파이썬 표준입니다.
- AutoCar 클래스로 AutoCar 객체(인스턴스) ② 를 만듭니다.
- setSpeed()와 forward() 메소드로 속도는 80, 뒷바퀴는 전진하게 하고, steering 프로퍼티로 앞바퀴를 최대 오른쪽 회전 ③-1 시킵니다.
- time() 함수로 2초 대기합니다.
  - time() 함수에 소수점 인자를 사용하면 밀리초 단위 지연이 가능합니다.
- backward()로 뒷바퀴를 후진으로 바꾸고, steering 프로퍼티로 앞바퀴를 최대 왼쪽으로 회전 ③-2 시킵니다.
- time() 함수로 2초 대기합니다.
- steering 프로퍼티로 앞바퀴는 중앙에 맞추고 stop() 메소드로 뒷바퀴를 정지 ③-3 시킵니다.

The screenshot shows a dual-interface setup for Python development. On the left, PyCharm displays a script named 'hello.py' with the following code:

```
basic > hello.py > ...
1  from pop.Pilot import AutoCar
2  from time import sleep
3
4  car = AutoCar()
5
6  car.setSpeed(80)
7  car.forward()
8  car.steering = 1.0
9  sleep(2)
10 car.backward()
11 car.steering = -1.0
12 sleep(2)
13 car.steering = 0.0
14 car.stop()
15
```

Yellow circles numbered ① through ③-3 are overlaid on the code to indicate specific steps: ① covers the first two lines, ② covers the line 'car = AutoCar()', ③-1 covers the block from 'car.setSpeed(80)' to 'sleep(2)', ③-2 covers the block from 'car.backward()' to 'sleep(2)', and ③-3 covers the final 'car.stop()' line.

On the right, a Jupyter Notebook titled 'hello.ipynb' is open, showing the same code in a notebook cell. The cell output area contains the code again, indicating it has been run. The status bar at the bottom of the interface shows 'Ln 15, Col 1' and other details.

주피터 노트북은 셀을 편집하는 셀 선택 모드와 해당 셀에 코드를 입력하는 코드 입력 모드입니다.

- 주피터에서 셀 추가는 상단의 **+ Code** ⑥ 를 누릅니다.
  - 셀 제거는 **Del** 키를 누릅니다.

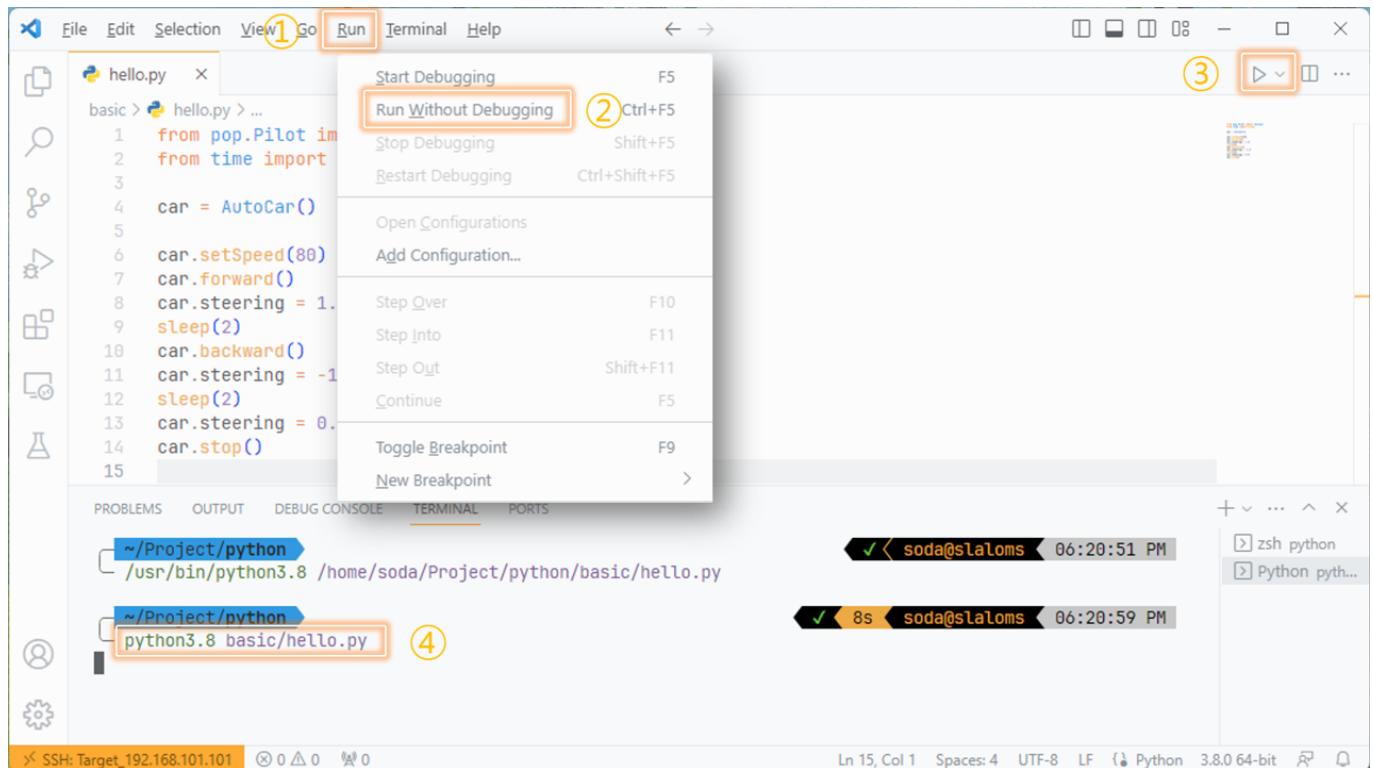
## 실행

코드 작성이 완료되었으면 파이썬 인터프리터 또는 주피터 커널을 이용해 실행합니다.

파이썬 스크립트(.py)는 다음과 같이 실행합니다.

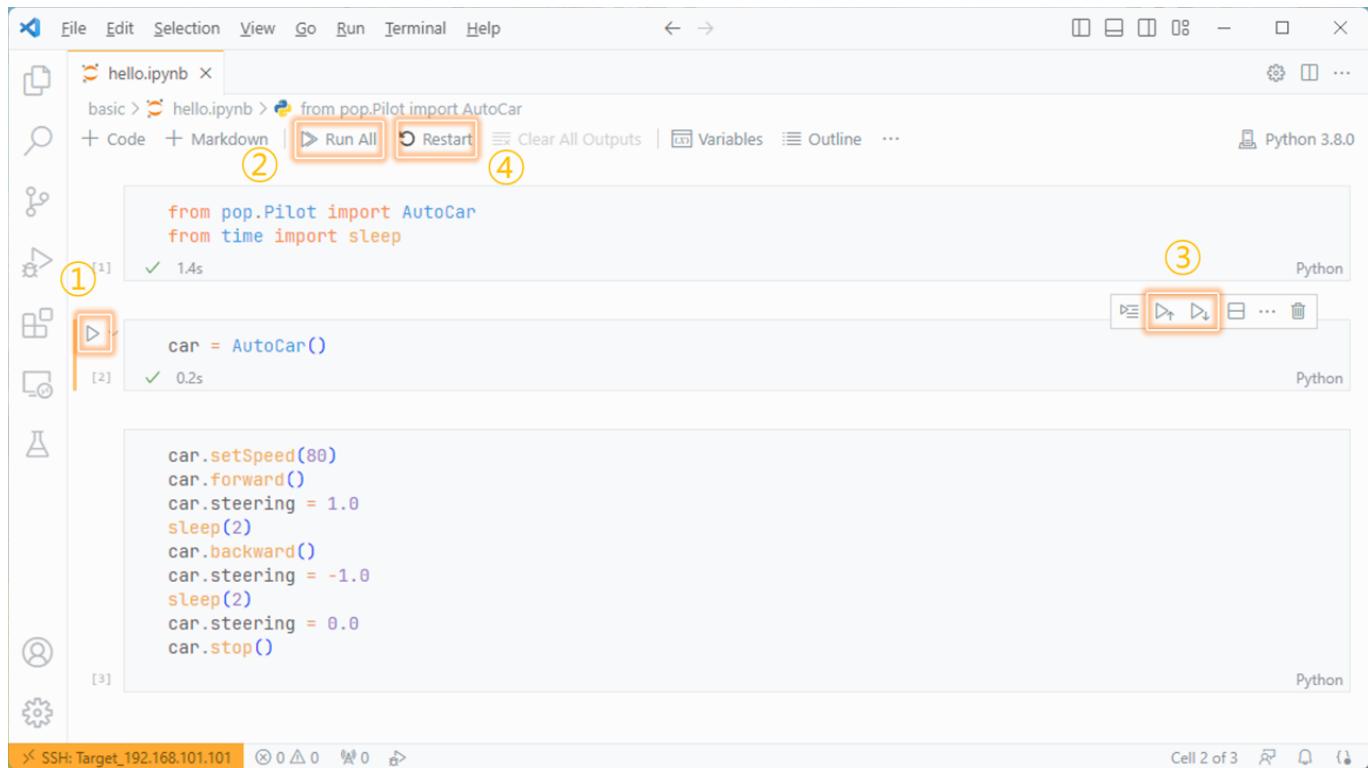
- 메뉴의 **Run** ① 에서 **Run Without Debugging** ② 또는 도구 모음의 **Run Python File** ③ 을 선택합니다.
  - 터미널에 자동으로 파이썬 인터프리터와 스크립트 파일 경로가 입력되어 실행되며, 두 번째 방법을 권장합니다.

- 가장 권장하는 방법으로 터미널에서 사용자가 직접 파이썬 인터프리터와 스크립트 파일 경로를 입력 ④ 해 실행합니다.
  - 현재 폴더와 파이썬 스크립트 파일 위치를 고려해 경로를 입력합니다.



주피터 노트북(.ipynb)은 다음과 같이 창에 표시된 도구를 이용해 실행합니다.

- 셀 왼쪽 옆의 **Excute cell** ① 을 누르면 현재 셀이 실행됩니다.
- 헤더의 **Run All** ② 를 누르면 전체 셀이 실행됩니다.
- 셀 오른쪽 도구 모음에서 **Excute Above Cells** 또는 **Excute Cell and Below\*\*** ③ 를 선택하면 현재 셀과 위 또는 아래 모든 셀을 실행합니다.
  - Excute Above Cells**는 첫 번째 셀부터 현재 셀 전까지 차례로 실행합니다.
  - \*\***Excute Cell and Below\*\*\***는 현재 셀부터 마지막 셀까지 차례로 실행합니다.
- 오랜 시간 실행되는 셀을 강제 종료하려면 헤더의 **Restart** ② 를 눌러 파이썬 커널을 다시 시작합니다.
  - 실행과 관련된 셀을 처음부터 다시 시작해야 합니다.



주피터 노트북은 셀을 관리하는 선택모드와 코드를 입력하는 입력모드로 나눠지는데, 다음은 주피터 노트북의 유용한 단축키 모음입니다.

- 공통
  - <Ctrl> + <Enter>: 현재 셀을 실행합니다.
  - <Shift> + <Enter>: 현재 셀을 실행하고, 다음 셀로 커서를 옮깁니다. 만약 다음 셀이 없다면 새로 추가합니다.
- 선택모드
  - A: 현재 셀 위에 새로운 셀을 추가합니다.
  - B: 현재 셀 아래에 새로운 셀을 추가합니다.
  - D: 현재 셀을 삭제합니다.
  - M: 현재 셀을 코드에서 마크다운으로 변경합니다.
  - Y: 현재 셀을 마크다운에서 코드로 변경합니다.
  - C: 현재 셀을 복사합니다.
  - V: 복사한 셀을 붙여 넣습니다.
  - X: 현재 셀을 잘라냅니다.
- 입력모드
  - VScode의 편집 단축키를 그대로 사용합니다.

## VSCode 원격 실행 절차 더 알아보기

지금까지 PC와 AutoCar3를 TCP/IP 네트워크로 연결한 후 VSCode 원격 개발환경을 통해 코드를 작성하는 과정을 알아보았습니다.

이런 원격 개발환경을 사용하면 실제 모든 작업은 AutoCar3에서 이뤄지고, PC는 AutoCar3의 모니터와 키워드 역할만 수행합니다.

물론 VSCode 자체는 PC에서 실행되지만, AutoCar3에 설치된 VS Code Server를 통해 VSCode에서 수행하는 모든 작업이 AutoCar3에서 반영됩니다.

따라서 PC 성능은 VSCode를 실행할 정도면 충분합니다.

