

# LiDAR

## Light Detection And Ranging

Department of Electrical Engineering, Incheon National University  
**Hwasu Lee, Munkyu Bae**

2024.07.02~05

# 강사 소개



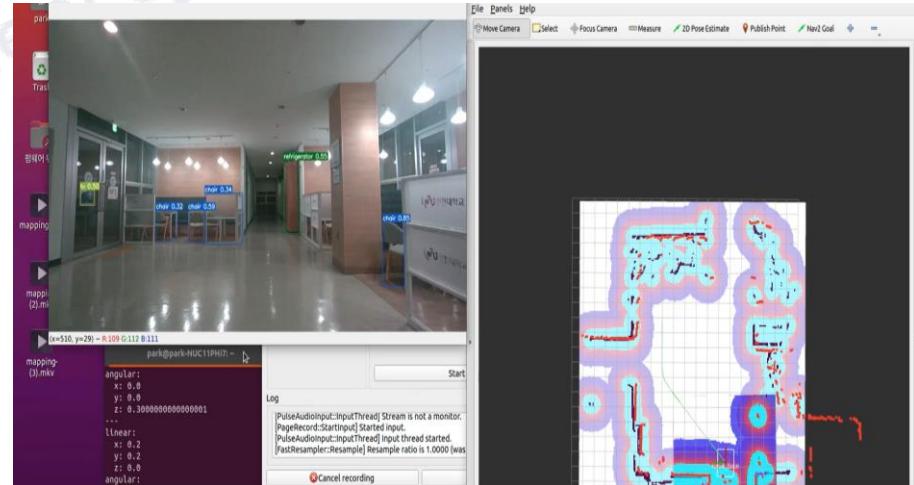
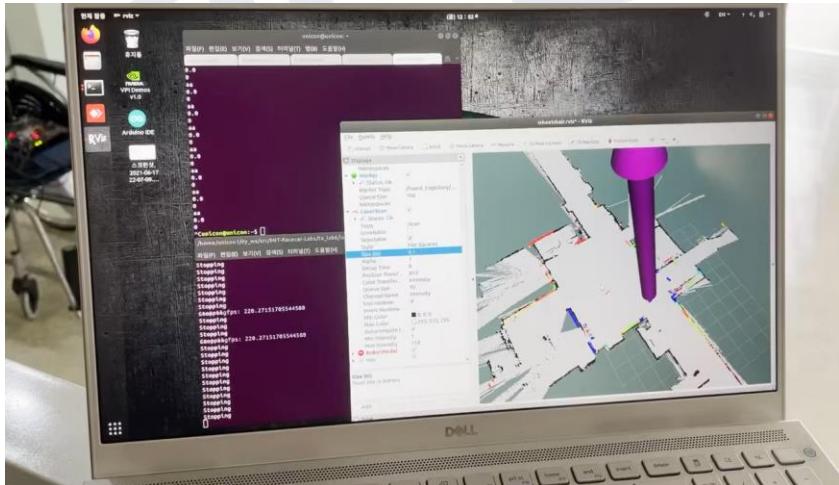
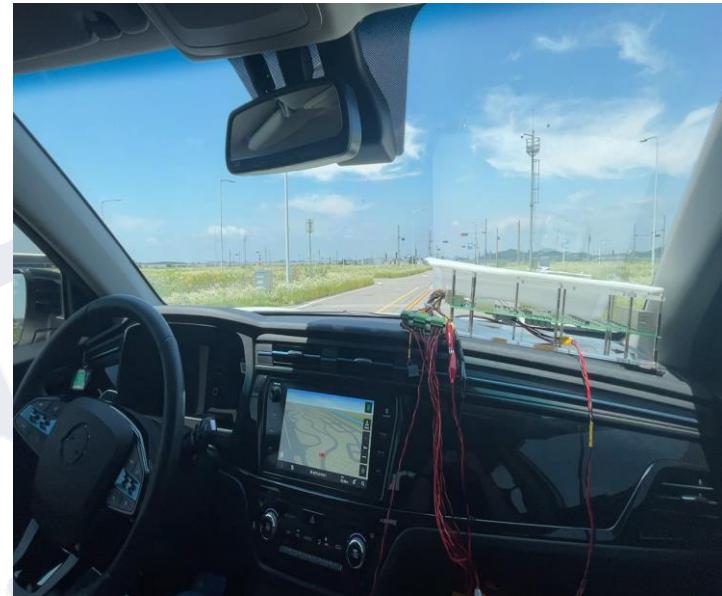
- 이화수 (Hwasu Lee)
- 2015.03 ~ 2022.02: 인천대학교 전기공학과 학사  
2022 ~ 현재: 인천대학교 전기공학과 석사 과정 재학 중
- 무인지능 시스템제어 연구실 소속  
(지도교수: 강창묵 교수, <https://uniconlab.wixsite.com/main>)
- 관심 분야: 예측 제어, 경로 탐색, 심층 강화학습,  
ROS 1 & ROS 2 기반 SLAM 및 Navigation
- 적용 플랫폼: 자율주행 차량, 모바일 로봇 등
- Projects
  - 자율주행 전동휠체어 핵심 기술 개발
  - 순찰로봇의 도심지 적용을 위한 자율주행 기술 개발
  - 지능형 건물 바닥 청소 로봇 플랫폼 개발
  - 인천공항 터미널 폐기물 수집 및 이송로봇 개발

# 강사 소개



- 배문규 (Munkyu Bae)
- ~ 2022.02: 인천대학교 전기공학과 학사  
2022 ~ 현재: 인천대학교 전기공학과 석사 과정 재학 중
- 무인지능 시스템제어 연구실 소속  
(지도교수: 강창욱 교수, <https://uniconlab.wixsite.com/main>)
- 관심 분야: 선형 제어, 비선형 제어, 예측 제어, 센서 퓨전 등
- 적용 플랫폼: 자율주행 차량, 모바일 로봇 등
- Projects
  - 전동휠체어 자율주행 핵심 기술 개발
  - Single-Channel LiDAR 기반 3D mapping 구현
  - 협동로봇을 활용한 언텍트 스토어 운영시스템 개발
  - 동작 분석을 위한 센서 및 알고리즘 개발
  - 자율주행 차량 토크/조향/제구동 통합제어기 개발

# 연구실 수행 프로젝트



# Contents

## 1. Autonomous Driving

- 1) 자율주행 자동차란?
- 2) 자율주행의 3요소
- 3) 자율주행을 위한 센서
- 4) 대표적인 자율주행 기술
- 5) 자율주행 기술의 기대 효과

## 2. Overview of LiDAR

- 1) LiDAR 센서란?
- 2) LiDAR 센서의 원리
- 3) LiDAR 센서의 구조
- 4) 스캐닝 방식에 따른 LiDAR 센서의 종류
- 5) 자율주행 기술 실현을 위한 LiDAR 센서의 역할

## 3. Basics of Python grammar

- 1) Python이란?
- 2) 개발 환경 설정
- 3) 기초 문법
- 4) 제어 구조 (조건문 / 반복문 / 반복 제어)
- 5) 자료 구조 (리스트 / 튜플 / 딕셔너리)
- 6) 함수
- 7) 클래스
- 8) 모듈, 패키지, 라이브러리 활용 (Matplotlib)

# 1. Autonomous Driving

# 1. Autonomous Driving

## 1) 자율주행 자동차(Autonomous Vehicle)란?

- 자율주행 자동차는 운전자 없이도 스스로 상황을 판단하고 주행하는 자동차를 의미
- 자율주행 자동차는 출발지에서 목적지까지 이동하는 경로를 스스로 결정하여 주행함
- 경로상에는 건물, 가로수, 표지판과 같은 정적인 객체들과 주행중인 차량, 보행자 등의 동적인 객체가 존재
- 주변 객체들을 정확하게 인지하여 안전한 주행을 위한 적절한 대응과 최적의 이동 경로를 탐색하기 위한 판단 과정을 수행함
- 주행 과정에서 신호등, 교통 표지판과 같은 교통 법규와 규칙들을 반드시 준수해야 함

# 1. Autonomous Driving

## 자율주행 기술의 단계



자료 출처. <https://happist.com/563575>

## 현재 기술 수준



자료 출처. <https://blog.hyundai-transys.com/387>

# 1. Autonomous Driving

## SAE 자율주행 단계(Levels of Driving Automation)

- 국제자동차기술자협회(SAE International)에서는 운전 자동화에 대한 자율주행 단계를 총 6단계로 구분
- 자율주행 기능이 없는 비자동화 단계부터 완전 자율주행이 가능한 수준까지의 운전 자동화를 포함
- 가장 낮은 단계인 **레벨 0(No Automation)**은 운전과 관련한 모든 조작과 모니터링 활동을 시스템의 개입 없이 운전자가 직접 통제하는 수준임
- **레벨 1(Driver Assistance)**은 차선 위치를 유지하기 위한 조향 제어, 속도 조절을 위한 가·감속 제어와 같은 운전 보조 기능이 추가된 수준을 의미함
- **레벨 2(Partial Automation)**는 자동차가 스스로 주행하는 첫 단계로, 고속도로와 같은 제한적인 상황에서 자율주행 시스템이 차량의 속도와 차선 위치 유지를 위한 제어를 담당함

# 1. Autonomous Driving

## SAE 자율주행 단계(Levels of Driving Automation)

- 레벨 3(Conditional Automation) 단계부터는 운전의 주체가 운전자에서 자율주행 시스템으로 변경
- 레벨 3(Conditional Automation)는 특정 조건에서는 자율주행 시스템이 운전을 완전하게 통제하며, 운전자는 지속적으로 모니터링할 필요가 없지만 시스템의 요청이나 경고에 맞춰 대응할 수 있어야 함
- 레벨 4(High Automation)는 일정하게 정의된 범위와 환경에서 모든 동작을 자율주행 시스템이 담당하여, 운전자는 모니터링하거나 대기할 필요가 없는 수준
- 레벨 5(Full Automation)는 운전에 관련한 모든 범위의 작업을 시스템이 완전히 통제하고 책임지는 완전 자율주행의 단계를 의미

# 1. Autonomous Driving

## 대표적인 자율주행 기업



자료 출처. <https://www.autoherald.co.kr/news/articleView.html?idxno=33375>

정밀지도와 LiDAR 센서를 기반으로 제한된  
영역에서의 완전 자율주행(Level 4)을 지향



자료 출처. <https://topictree.co.kr/newcar/tesla-new-roadster/>

기존에 판매한 수십만대의 차량에서 업로드한 영상  
정보를 바탕으로 운전자 보조 기능(Level 2) 고도화 지향

	<b>Google Waymo</b>	<b>TESLA</b>
<b>핵심 기술</b>	LiDAR + HD Map	Camera(Vision) + Deep Learning
<b>기능</b>	Replace driver	Support driver

# 1. Autonomous Driving

자율주행 참고 영상 – Google Waymo

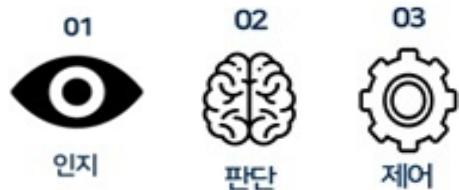


한번 직접 불러 봤습니다

자료 출처. <https://www.youtube.com/watch?v=U3wA3QM1pxc>

# 1. Autonomous Driving

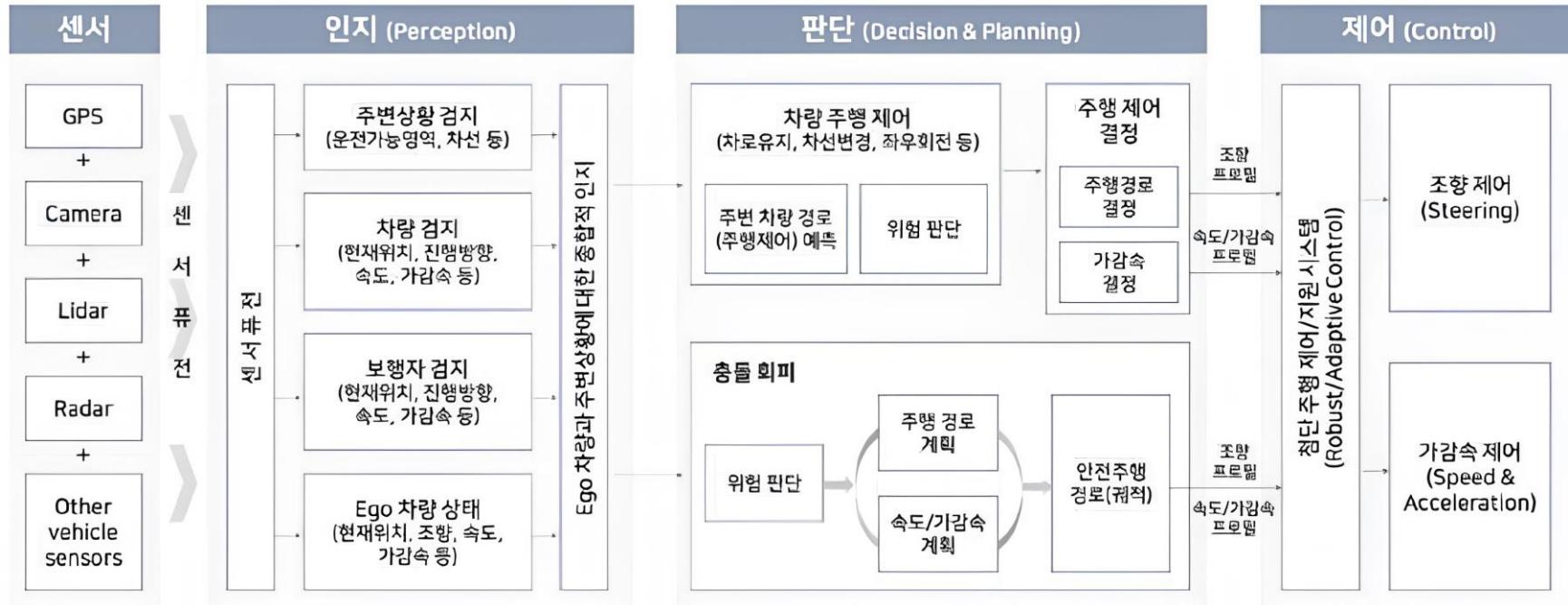
## 2) 자율주행의 3 요소



- 인지(Perception): “인지”는 자율주행 시스템에서 차량 주변 환경을 감지하고 이해하는 역할.  
센서(Camera, Radar, LiDAR)와 알고리즘을 통해 도로의 상태, 교통 표지판, 다른 차량, 보행자 등을 식별하고 그 위치와 속도를 파악함.
- 판단(Decision): “판단”은 인지 과정에서 수집된 정보를 바탕으로 차량의 행동을 결정함.  
이는 경로 계획, 장애물 회피 전략 수립, 안전하고 법규를 준수하는 운전 결정을 포함.  
해당 단계에서는 다양한 시나리오와 예상되는 결과를 평가하여 최적의 운전 계획 결정.
- 제어(Control): “제어”는 결정된 운전 계획에 따라 차량의 스티어링, 가속, 제동 등을 조절.  
해당 과정을 매우 정밀하게 이루어져야 하며, 실시간으로 차량의 동적 상태를 조정하여 안정적이고 효율적으로 차량이 운행되도록 함.

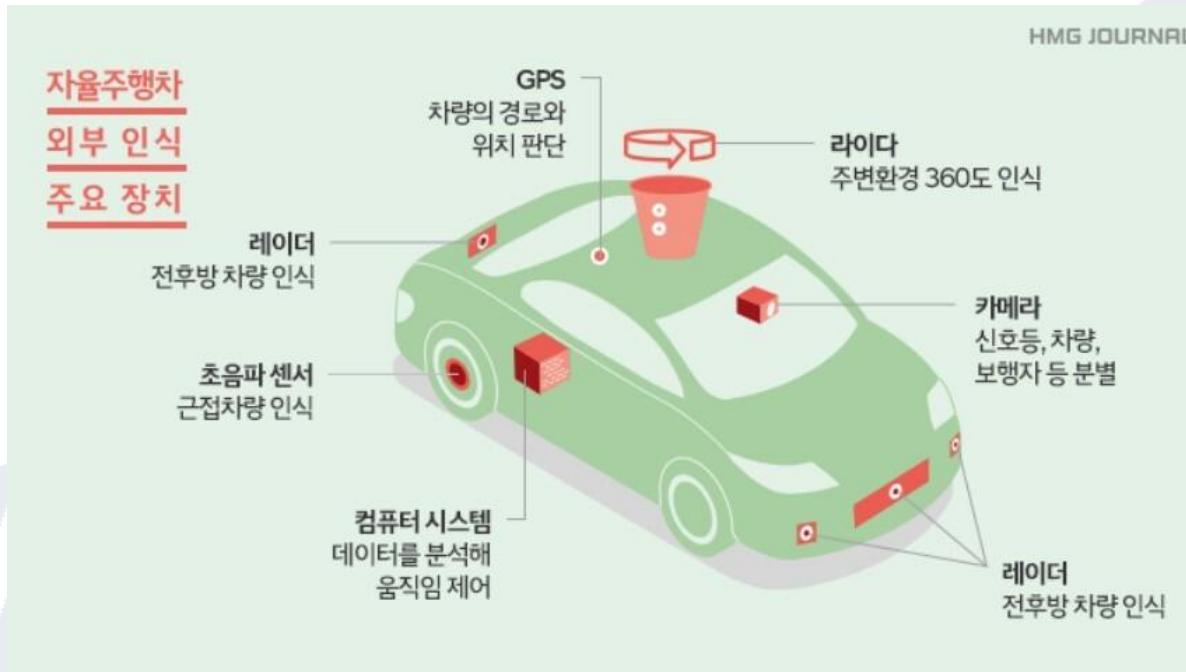
# 1. Autonomous Driving

## 2) 자율주행의 3 요소



# 1. Autonomous Driving

## 3) 자율주행을 위한 센서



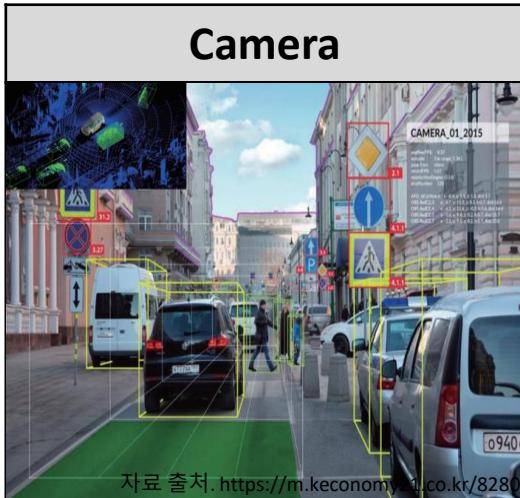
자료 출처. <https://brunch.co.kr/@nostalgia9/444>

- 자율주행 자동차의 안전한 주행을 위해서는 주변 환경을 인지해 주행에 필요한 정보를 제공하는 센서 시스템이 필수적
- 자율주행 자동차의 센서 시스템은 개별 센서의 기능, 비용, 디자인을 비롯한 여러 요인을 고려해 최적으로 설계
- 각 센서의 장점과 단점을 상호 보완할 수 있도록 여러 센서 데이터를 조합하는 센서 퓨전(Sensor fusion) 기술은 자율주행 시스템의 신뢰성과 안정성을 높이는데 매우 중요

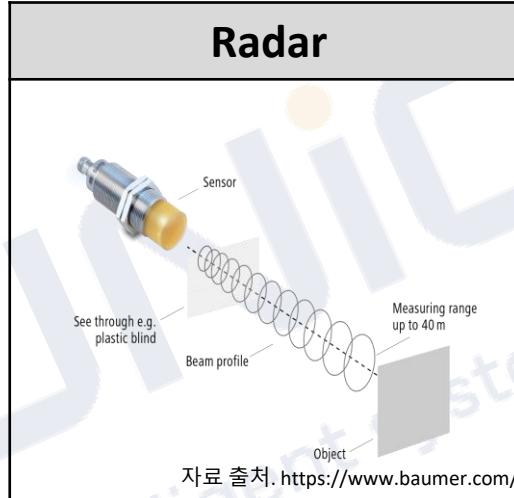
# 1. Autonomous Driving

## 3) 자율주행을 위한 센서

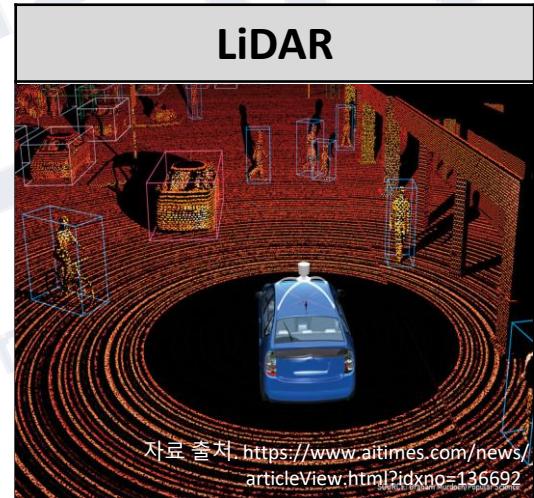
- 인지(Perception)



- 객체 인지 및 위치 추정에 활용
- 텍스처 정보를 포함한 영상 정보 제공
- 30~60 Hz의 빠른 측정 주기
- HD급 카메라를 다수 장착해 FOV 확보
- 고해상도 데이터, 저렴한 비용
- 조명 환경 변화에 취약, 정확한 깊이 추정 불가



- 장애물 회피 및 충돌 감지에 활용
- 전자기파를 이용해 측정한 거리, 속도 정보 제공
- 근거리와 원거리에 대해 모두 측정 가능
- 눈, 비, 조명 등 외부 환경에 강건
- 작은 물체 측정에 취약, 낮은 해상도

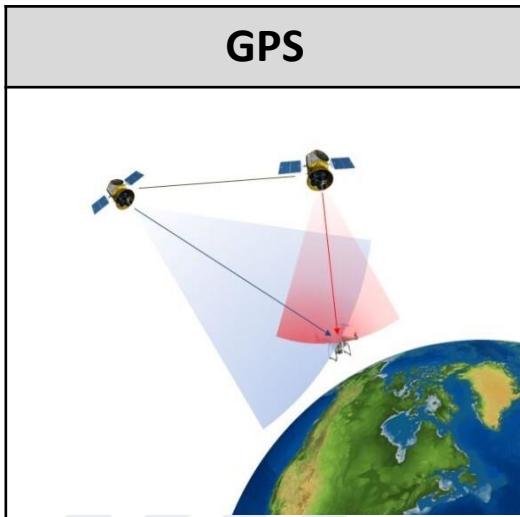


- 객체 인지 및 위치 추정에 활용
- 레이저를 이용해 고해상도의 정확한 3차원 거리 정보 제공
- 10~20 Hz의 주기
- 대상체에 대한 반사도 정보 제공
- 조명 환경 변화에 강건
- 눈, 비, 안개 등 악천후에 민감
- 높은 비용, 제품 양산 및 상용화

# 1. Autonomous Driving

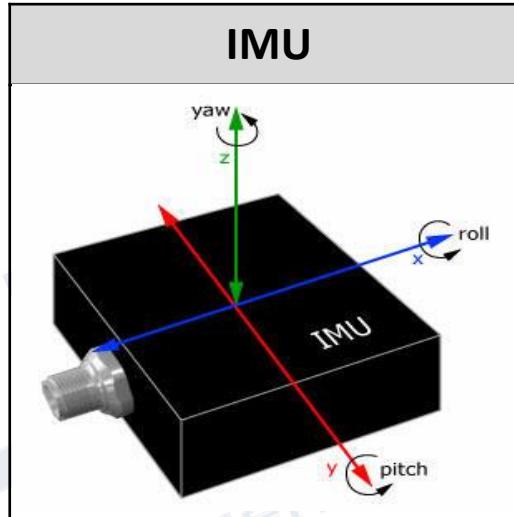
## 3) 자율주행을 위한 센서

- 위치 추정(Localization)



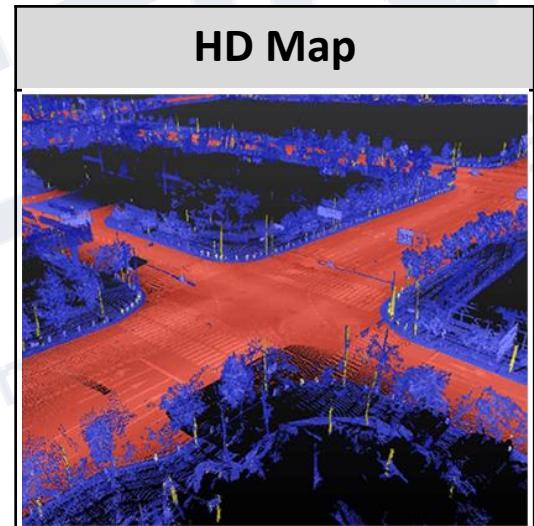
자료 출처. <https://brunch.co.kr/@matthewmin/122>

- 차량의 위치 추정에 필요
- 위성 신호를 이용해 지구좌표계에 따른 절대 위치 정보 제공 (위도, 경도, 고도)
- 1~10Hz 수준의 느린 측정 주기
- 지하/터널/도심 등에서 음영 지역 발생
- 일반적인 GPS는 수 m 오차로 자율주행 용으로는 부정확함
- DGPS, GPS-RTK는 수십 cm 내 정확도



자료 출처. <https://www.usgs.gov/centers/pcmsc/science/inertial-measurement-unit-imu>

- 차량 움직임, 위치 추정에 필요
- 6축 또는 9축 관성(가속도, 각속도, 지자기) 정보를 제공
- 200 Hz 이상의 빠른 측정 주기
- 추측 항법을 통해 차량의 이동 거리 및 방향 추정 가능
- 외부 환경에 강건하나, 시간에 따른 누적 오차 발생

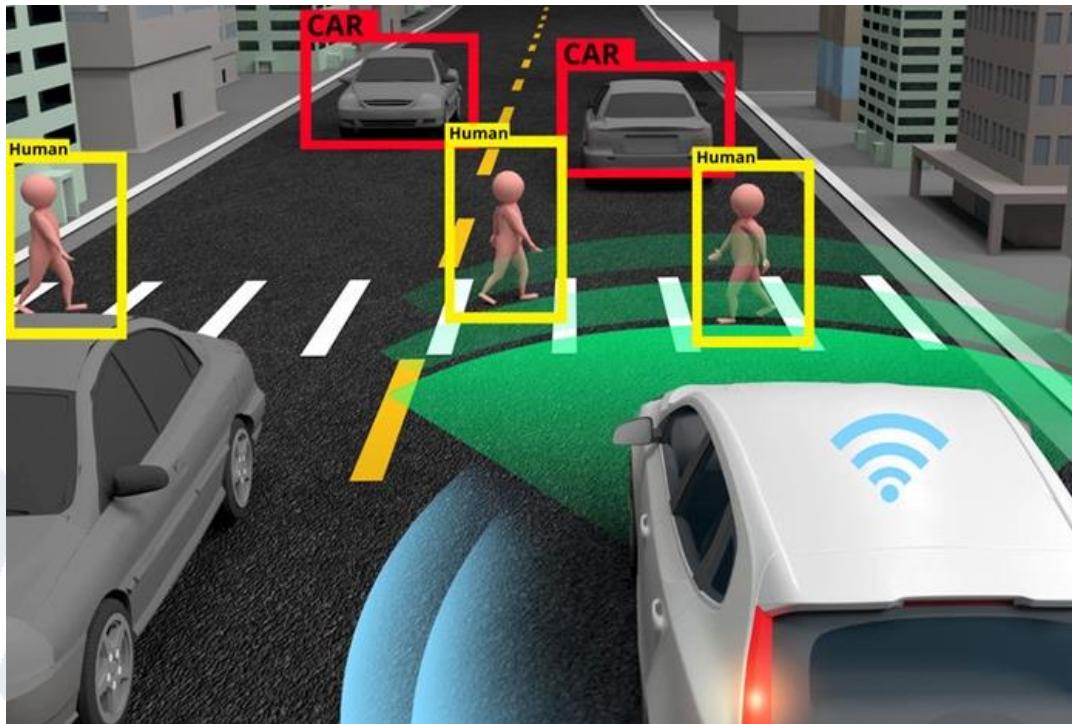


자료 출처. <https://simkkong.tistory.com/14>

- 자율주행 차량을 위한 다양한 도로 정보가 정밀하게 기록되어 있는 지도
- 위치 추정, 경로 계획, 객체 인지에 모두 사용
- 도로 규칙 정보, 구조물 형상 정보, 실시간 교통 정보, 센서 데이터 등
- 구축에 많은 비용이 들고, 지속적인 업데이트가 필요

# 1. Autonomous Driving

- 인지 (Perception)

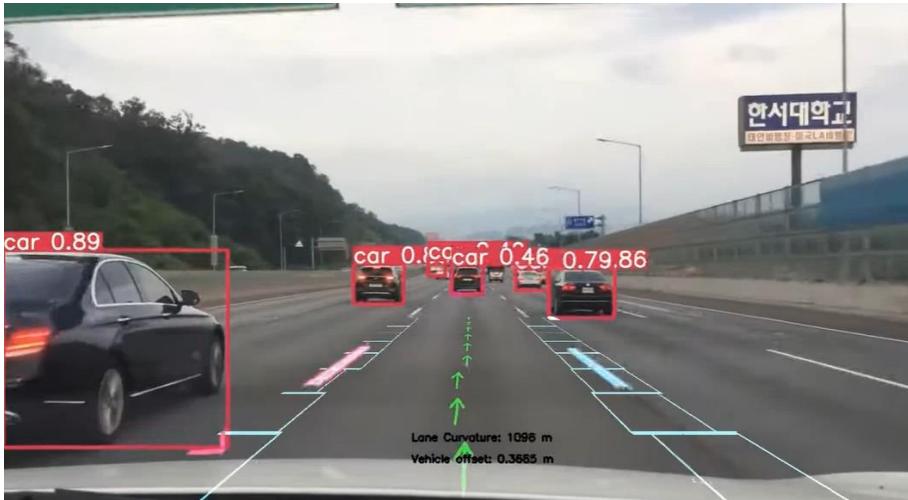


자료 출처. <https://kidd.co.kr/news/210655>

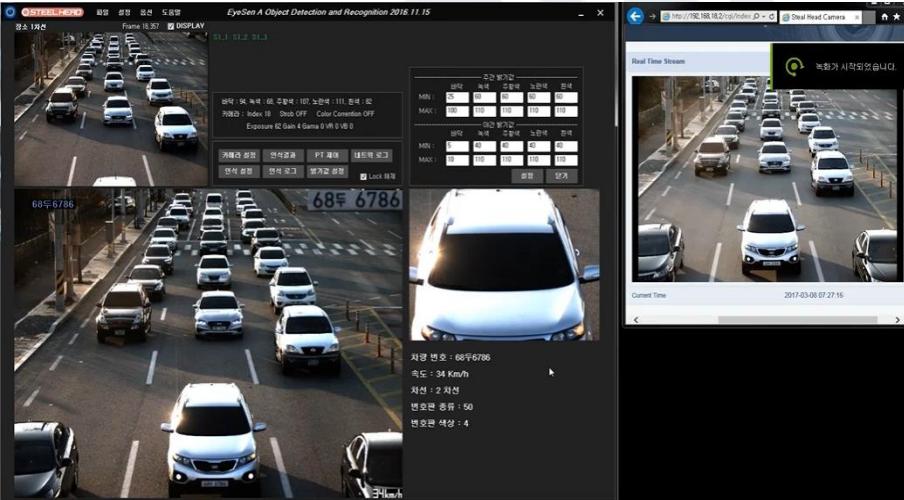
- 차량 주변의 정적 및 동적인 환경을 측정하고 주변 환경을 묘사
- 보행자, 차량, 자전거, 도로 표면, 차선 경계, 교통 신호 등
- 인지 정보를 바탕으로 주행에 대한 제어 방식을 계획하고 결정
- 다양한 센서(Camera, Radar, LiDAR)와 딥러닝 기술을 활용

# 1. Autonomous Driving

- 인지 (Perception) - Camera 센서 기반 객체 인식



자료 출처. <https://www.youtube.com/watch?v=chzq2E75M84>

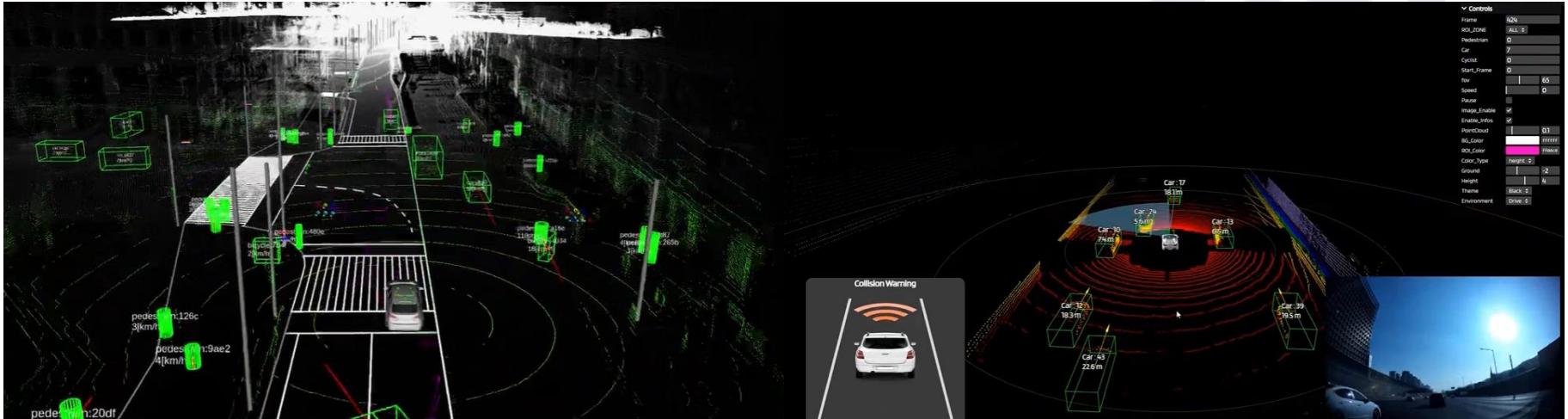


자료 출처. <https://www.youtube.com/watch?v=5zOFzh36CS0>

- 딥러닝 기반의 방법들로 우수한 성능을 보장할 뿐만 아니라, 높은 수준의 의미론적 정보 또한 추론 가능
- 계산 성능(Computing power)의 발달 및 모델 경량화로 On-device AI 가능
- Camera 센서의 내재적 한계점을 내포 : 조도 변화에 민감하며, 객체와의 정확한 거리 측정 불가
- LiDAR, Radar, HD Map 등과의 센서 퓨전을 이용한 상호 보완이 필수적

# 1. Autonomous Driving

- 인지 (Perception) - LiDAR 센서 기반 객체 인식



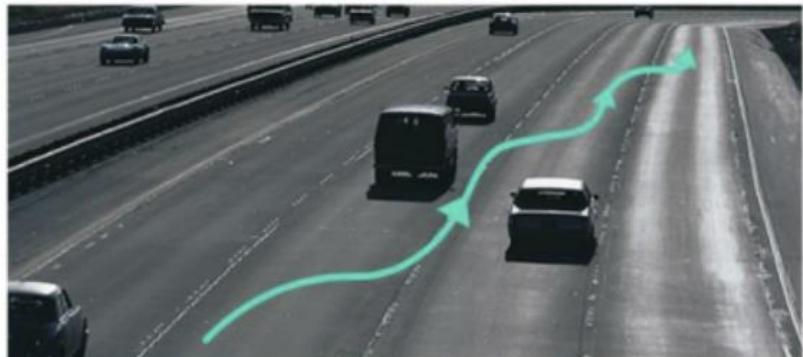
자료 출처. <https://www.youtube.com/watch?v=si9gamz07LA>

자료 출처. <https://www.youtube.com/watch?v=lIDq9W1oSgw>

- LiDAR 기반 객체 인식 기술을 이용하여 대상 객체에 대한 3차원 공간 상의 정확한 위치, 거리, 이동 방향 등을 추정 가능
- 딥러닝 기반의 방법들로 성능 개선이 이루어지고 있으며, Camera 센서의 한계점을 상호 보완하는 형태로 사용됨
- 보다 많은 LiDAR 데이터셋의 확보와 딥러닝 모델의 성능 향상을 위한 연구 개발이 진행 중

# 1. Autonomous Driving

- 판단 (Decision)



전역 및 지역 경로 생성  
(Path Generation)

자료 출처. <https://velog.io/@jaehawilly/posts>



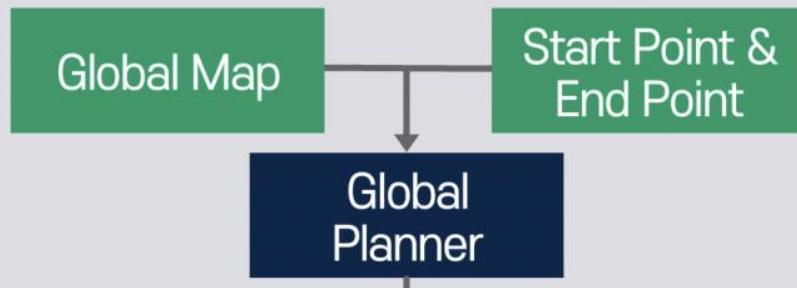
주행상황 판단  
(Decision-Making)

- 지도, 차량의 위치, 도착 위치, 주변 환경 정보를 기반으로 가능한 경로를 예측/생성하는 기술
- 지도를 그래프 구조로 변환하여 최적의 경로를 탐색
- 최근에는 강화학습 기반의 경로 계획 기법들에 대한 연구가 활발히 이루어지고 있음
- 실제 환경에서의 실험이 어렵기 때문에 주로 가상 환경에서 시뮬레이션이 수행됨

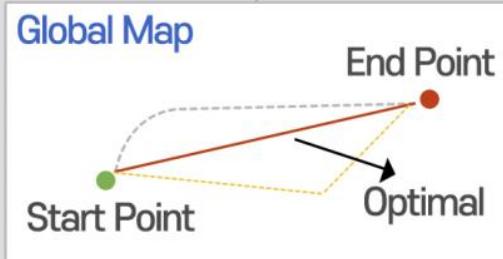
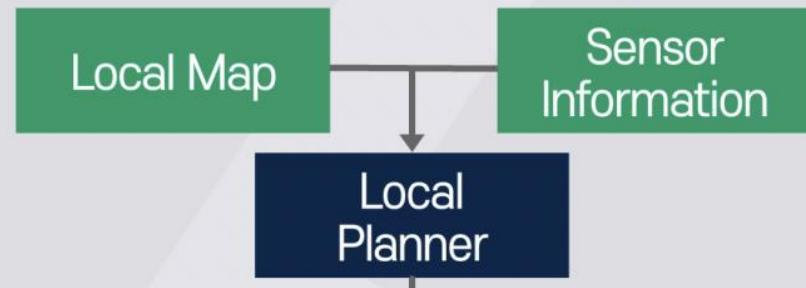
# 1. Autonomous Driving

- 판단 (Decision)

전역 경로계획  
(Global Path Planning)



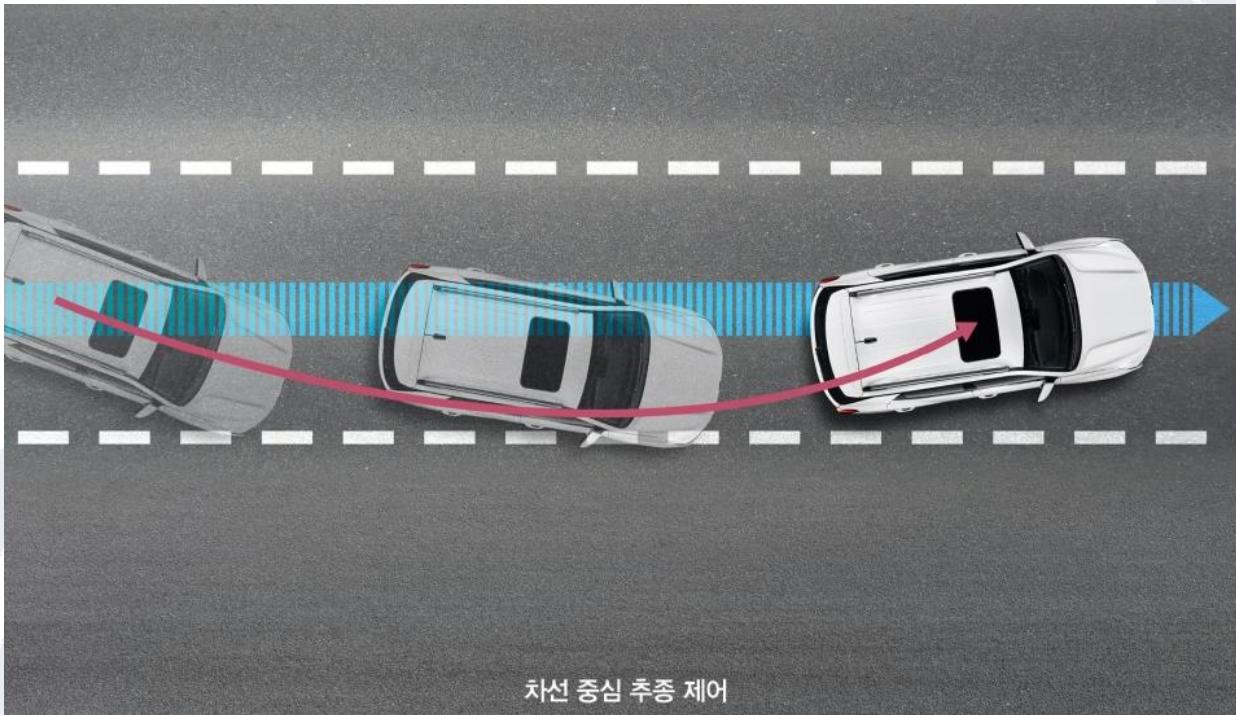
지역 경로계획  
(Local Path Planning)



출발지점에서 목적지점까지 주행할 수 있는 경로생성(Path Generation) 과정을 포함해야 함

# 1. Autonomous Driving

- 제어 (Control)

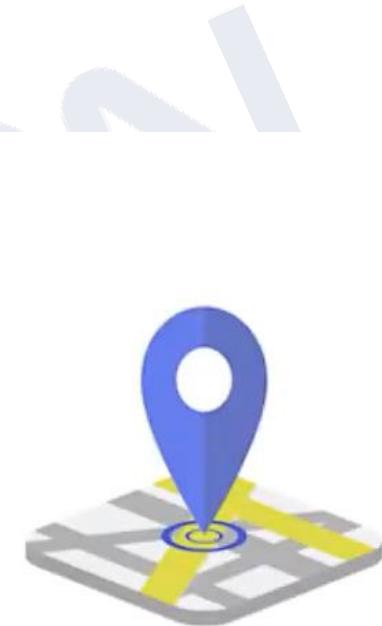
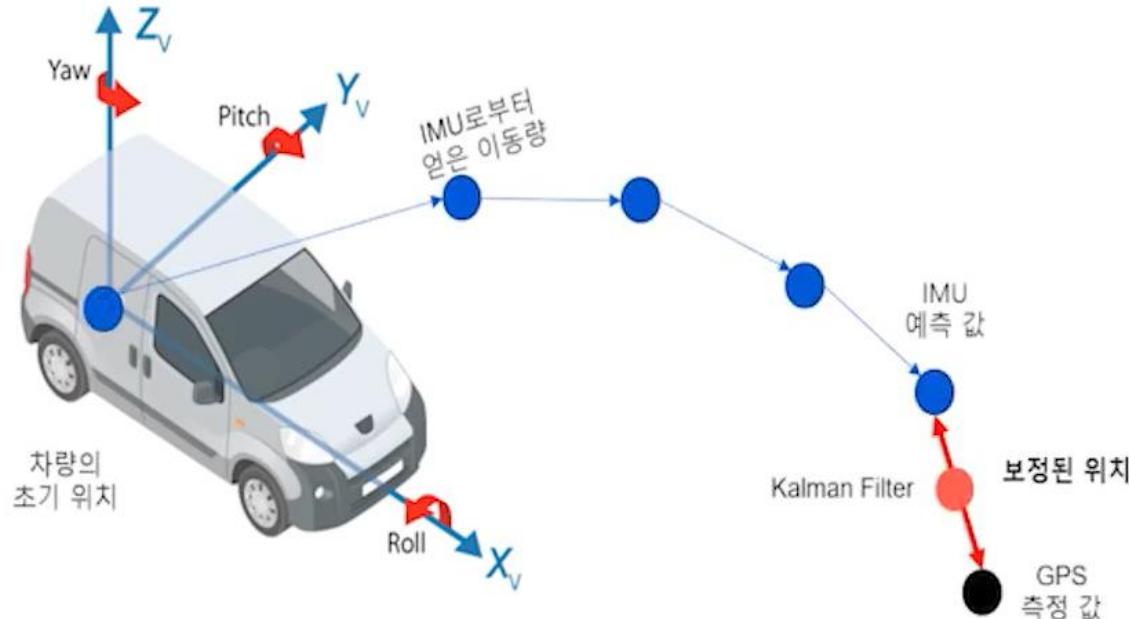


자료 출처. [https://www.motorgraph.com/news/articleView.html?idxno=21621#google\\_vignette](https://www.motorgraph.com/news/articleView.html?idxno=21621#google_vignette)

- ‘판단’ 단계에서 결정된 전역 경로를 올바르게 추종하기 위해 구체적인 속도와 가속도, 조향각 등의 제어를 수행
- 제어 시스템은 차량의 실시간 위치와 계획된 경로 사이의 차이를 지속적으로 분석하여, 필요한 조정을 즉각적으로 수행

# 1. Autonomous Driving

- 위치 추정 (Localization)

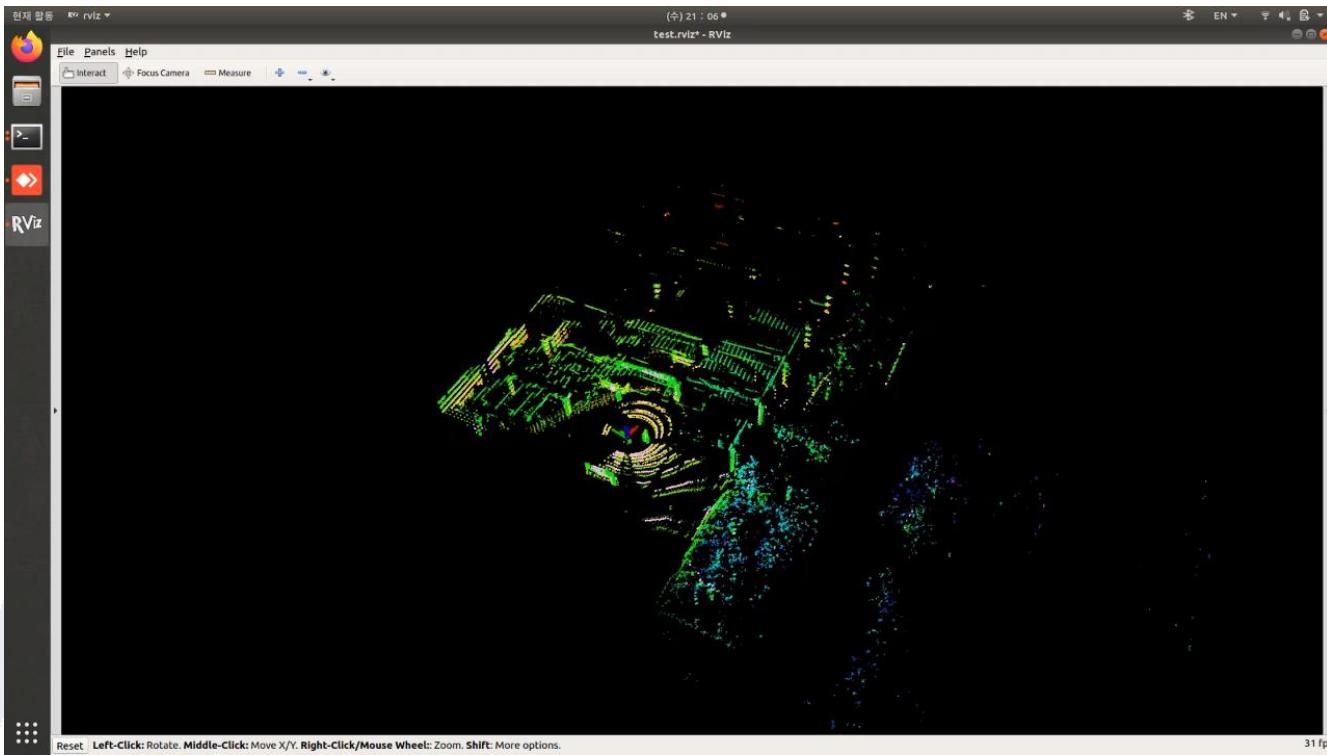


자료 출처. <https://www.youtube.com/watch?v=KQeKVLfgKX0&t=1134s>

- 차량의 위치를 실시간으로 정확하게 추정(예측)하는 기술
- 정확도 향상을 위해 다양한 센서(GPS, IMU 등)를 조합하여 사용
- 일반적으로 GPS의 절대 위치와 IMU 추측 항법(dead reckoning)의 상대 위치를 결합하여 위치를 추정
- GPS는 주기가 느리고 음영 지역이 많아 도심지에서는 신뢰도가 떨어짐

# 1. Autonomous Driving

- SLAM(Simultaneous Localization And Mapping)



자료 출처. <https://uniconlab.wixsite.com/main>

- 연속적인 Camera 센서(Mono, Stereo) 또는 LiDAR 센서로부터 상대적 위치를 추정하는 기술
- 랜드 마크에 대한 특징을 추출하고 대응 관계를 수립하여 프레임 간 차량의 움직임(위치)을 추정
- 위치 추정을 통해 획득한 정보를 기반으로, 측정 데이터들을 누적하여 지도 생성을 동시에 수행
- 누적 오차를 방지하기 위해 정밀하게 제작된 HD Map(High-Definition Map)이 필수적

# 1. Autonomous Driving

### ● 자율주행 시뮬레이션



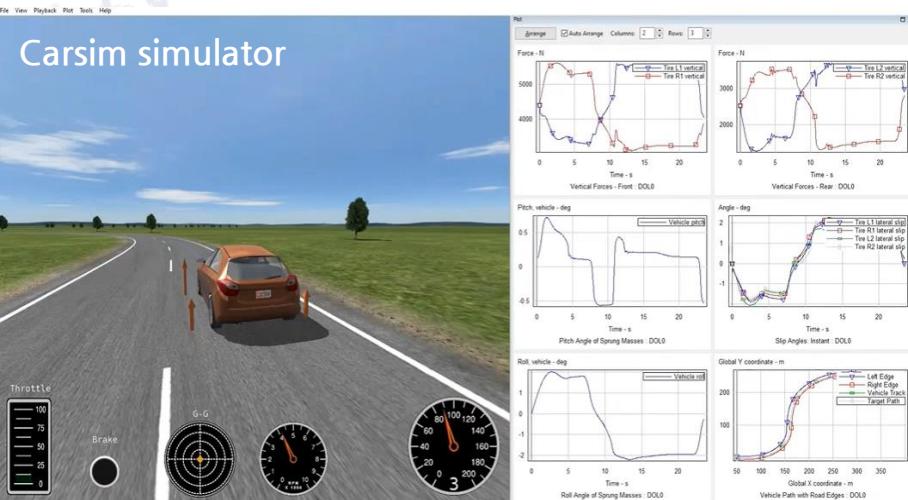
## CARLA simulator



자료 출처. <https://www.youtube.com/watch?v=CauKo089zm0>



자료 출처. <https://www.youtube.com/watch?v=jNYstyOKxIA>



자료 출처. <https://www.youtube.com/watch?v=CauKo089zm0>

# 1. Autonomous Driving

## 4) 대표적인 자율주행 기술

- ADAS (Advanced Driver Assistance System)

- ADAS(Advanced Driver Assistance Systems)는 고급 운전자 보조 시스템으로, 차량의 안전과 운전 편의를 증진하기 위해 설계된 다양한 기술과 기능의 총칭. 이 시스템들은 운전자의 입력을 보완하거나 일부 운전 작업을 자동화하여 사고를 예방하고, 운전 부담을 줄이는 데 중점을 둠.  
ADAS는 각종 센서와 카메라, 레이더, 라이다 같은 기술을 활용하여 차량 주변의 환경 데이터를 수집 및 분석.
- ADAS에는 전방 충돌 경고(FCW), 자동 긴급 제동(AEB), 차선 이탈 경고(LDW), 차선 유지 보조(LKA), 적응형 크루즈 컨트롤(ACC) 등의 기술들이 포함.

# 1. Autonomous Driving

## 4) 대표적인 자율주행 기술 – 횡방향 제어(Lateral control)

- 차선 이탈 방지 보조 (LKA, Lane Keeping Assistance)  
→ 차량이 차선을 이탈하지 않고, 차선 중앙을 유지할 수 있도록 스티어링을 자동 조정.



# 1. Autonomous Driving

## 4) 대표적인 자율주행 기술 – 종방향 제어(Longitudinal control)

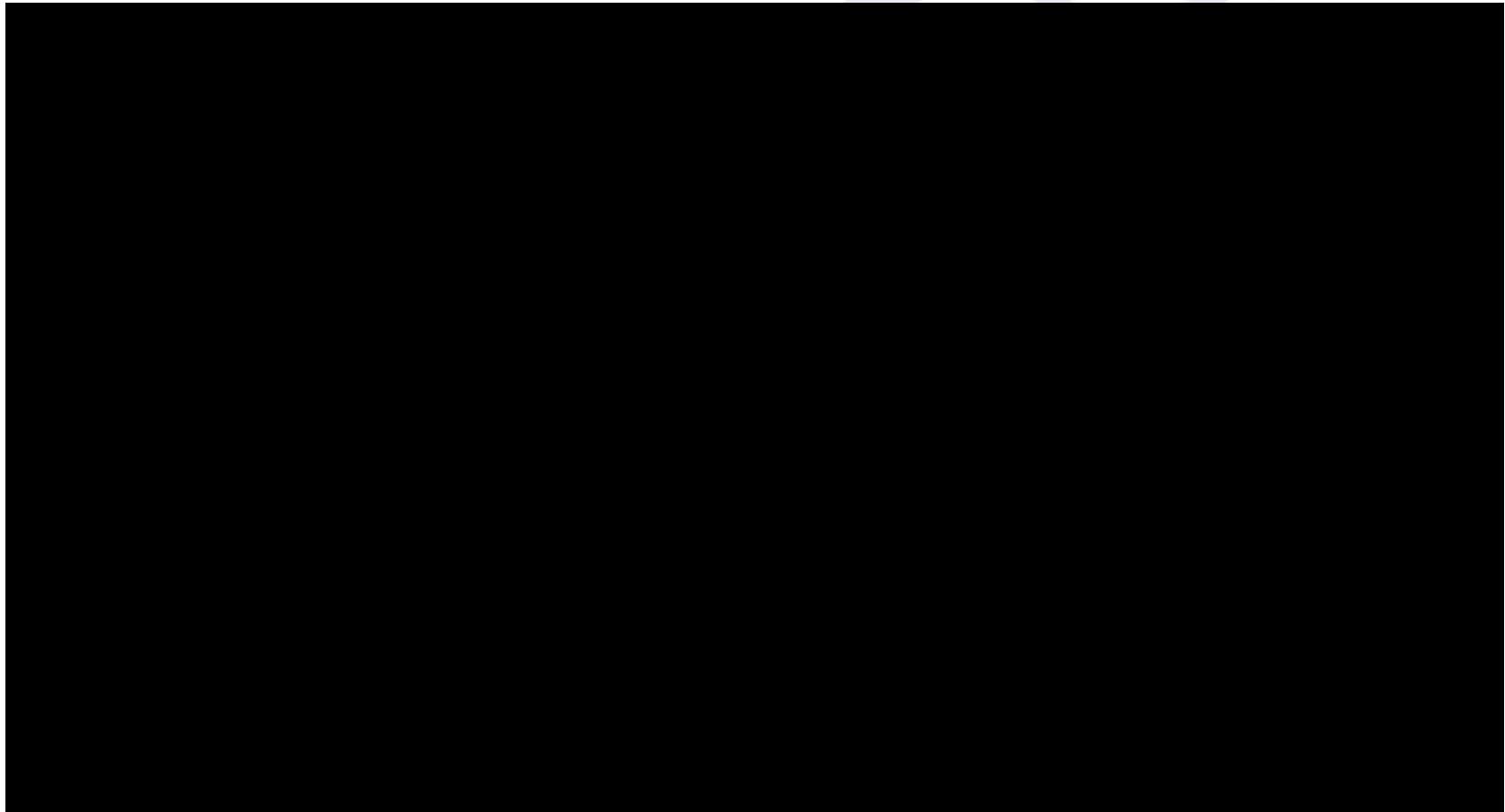
- 적응형 크루즈 컨트롤 (ACC, Adaptive Cruise Control)  
또는 스마트 크루즈 컨트롤 (SCC, Smart Cruise Control)  
→ 앞 차와의 거리를 단계별로 자동 조절하면서, 설정된 속도를 유지하도록 자동 제어.



# 1. Autonomous Driving

## 4) 대표적인 자율주행 기술 – 충돌 제어(Collision control)

- 자동 긴급 제동 (AEB, Autonomous Emergency Braking)  
→ 충돌 위험이 감지되면 차량이 자동으로 제동을 걸어 충돌을 피하거나 충돌의 심각성을 줄임.



# 1. Autonomous Driving

## 5) 자율주행 기술의 기대 효과

- 레벨 5의 완전 자율주행 기술이 실현되면 개인 생활과 사회 전반에 큰 변화를 가져올 것으로 기대됨
- 완전 자율주행 자동차는 피로를 느끼지도 않고, 교통 규칙을 위반하는 일도 없으며, 사람 운전자보다 더 빠른 속도로 의사 결정을 할 수 있게 되어 운전자의 실수로 인한 교통 사고와 피해를 크게 줄이고 안전도를 더욱 높일 수 있음
- 운전자는 목적지까지 이동하는 동안 운전을 위해 집중할 필요 없이 개인 용무를 보거나, 휴식을 취하거나, 오락적인 활동을 수행할 수 있게 되어 자동차가 단순한 이동 수단을 넘어 생활 공간으로 변하게 될 것으로 예상
- 도로 위에 모든 자동차에서 자율 주행이 가능해지면 교통 흐름은 지금보다 더 효율적이고 최적화되어 운송 수단에 의한 에너지 소비와 대기 오염을 줄임으로써 환경을 보존하고 깨끗한 미래를 실현하는데 기여할 수 있음

# 1. Autonomous Driving

Quiz 1) 고해상도의 영상 정보를 빠른 측정 주기로 획득할 수 있어

자율주행 시스템에서 차선, 신호등, 교통 표지판, 차량, 보행자 등을 인식하기 위한 목적으로 사용하는 수동형 센서는?

**카메라 센서**

Quiz 2) 레이저(Laser)를 이용해 정확한 3차원 거리 정보를 측정함으로써

카메라나 레이더가 동작하기 어려운 환경, 상황에서도 더욱 안전한 자율주행을 가능하게 하는 센서는?

**레이저 센서**

# 1. Autonomous Driving

## 학습 정리

- 자율주행 개요

- 자율주행 자동차는 운전자 없이도 스스로 상황을 판단하고 주행하는 자동차를 의미한다.
- 자율주행 기술의 성숙도는 자율주행 기능이 없는 레벨 0부터 완전 자율주행이 가능한 레벨 5까지 여섯 단계로 구분한다.

- 자율주행 시스템

- 자율주행 시스템이 동작하기 위해서는 센서/감지, 위치 추정, 객체 인지, 경로 계획/제어와 같은 요소 기술이 필요하다.

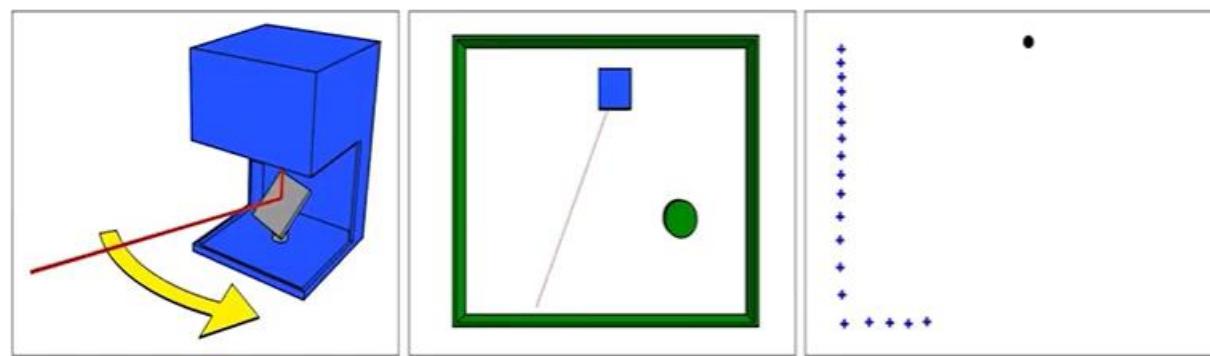
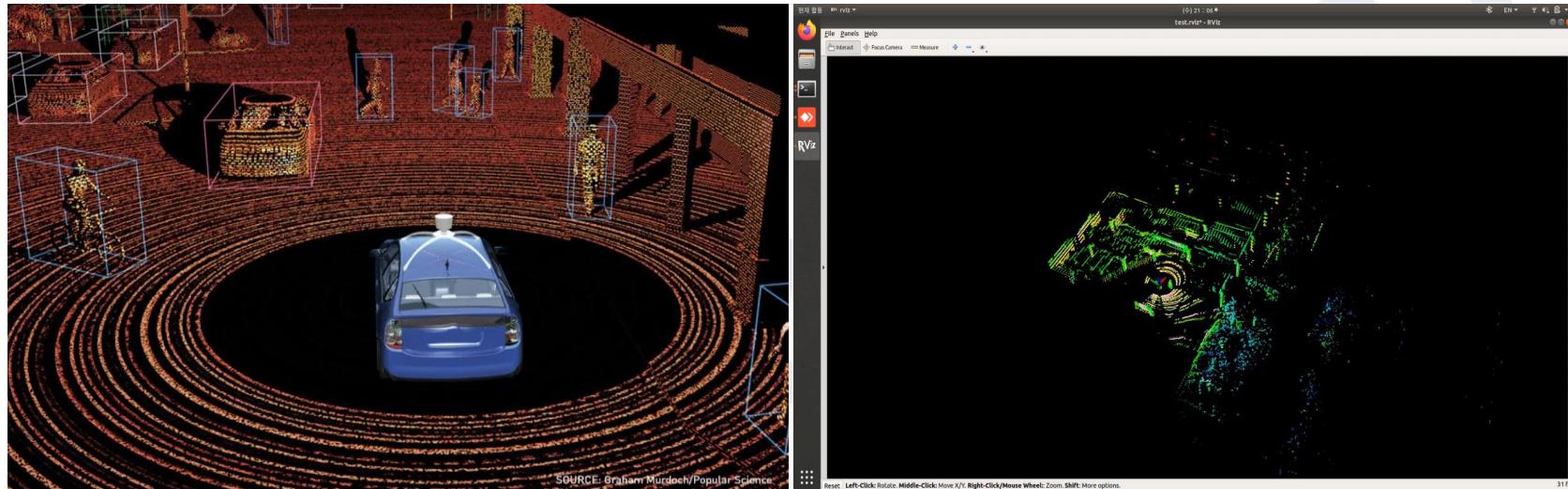
- 자율주행 센서

- 자율주행 자동차는 주행에 필요한 정보를 획득하기 위해 Camera, LiDAR, Radar와 같은 다양한 센서들을 활용해 주변 환경을 인지한다.
- 주행에 필요한 충분한 정보를 얻기 위하여 개별 센서들의 장/단점을 상호 보완하는 방향으로 자율주행 시스템을 구성한다.

## 2. Overview of LiDAR

## 2. Overview of LiDAR

### 1) LiDAR 센서란?



- LiDAR(Light Detection And Ranging) 센서는 빛의 이동 시간 (Time of Flight)을 측정하는 원리를 통해 높은 정밀도의 3차원 공간 정보를 고속으로 획득할 수 있는 센서

## 2. Overview of LiDAR

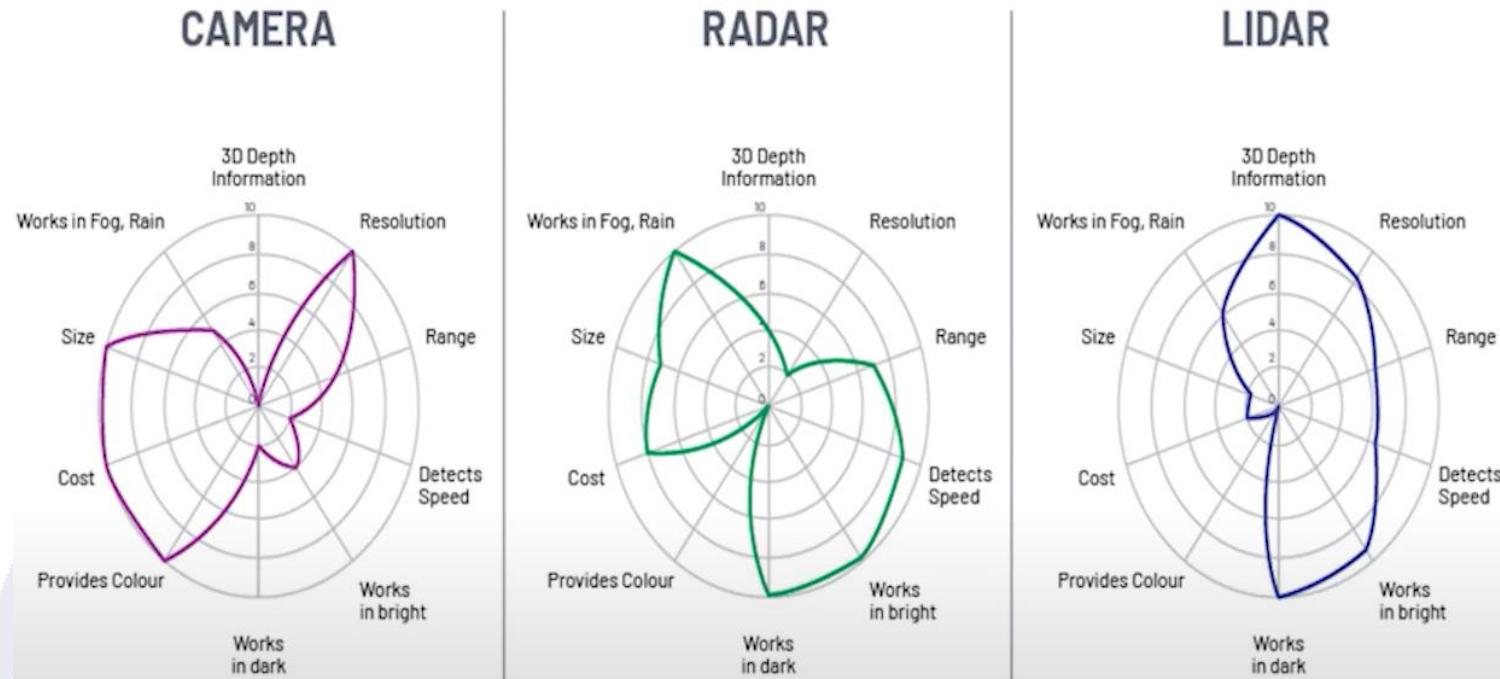
### 1) LiDAR 센서란?



- 자율주행 자동차가 안전하게 주행하기 위해서는 주변 환경을 인지해 주행에 필요한 정보를 제공하는 센서 시스템이 필수적
- 자율주행 자동차의 센서 시스템은 개별 센서의 기능, 비용, 디자인을 비롯한 여러 요인을 고려하여 최적으로 설계
- 각 센서의 장점과 단점을 상호 보완할 수 있도록 여러 센서 데이터를 조합하는 **센서 퓨전 (sensor fusion)** 기술은 자율주행 시스템의 신뢰성과 안정성을 높이는데 매우 중요한 기술

## 2. Overview of LiDAR

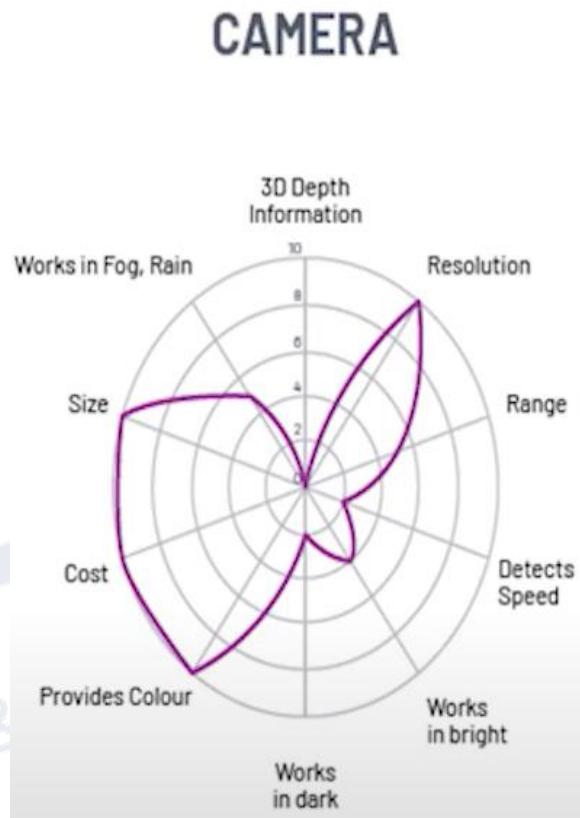
### 1) LiDAR 센서란?



- 자율주행 자동차의 안전한 주행을 위해 주변 환경을 인지하여 주행에 필요한 정보를 제공하는 센서 시스템이 필수적
- 주변 환경 인지를 위해 사용되는 대표적인 센서로는 Camera, Radar, LiDAR 등이 있으며, 각 센서 별 뚜렷한 장/단점이 존재

## 2. Overview of LiDAR

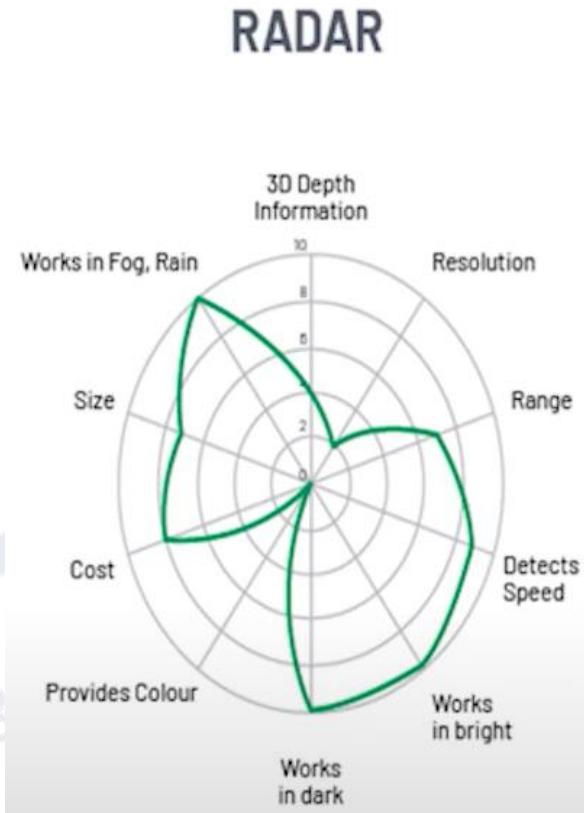
### 1) LiDAR 센서란?



- Camera 센서는 100만 화소급 이상의 고해상도 감지를 통해 감지 대상의 종류를 정확히 감지할 수 있지만, 감지 거리가 짧고 감지 대상의 속도 측정이 어려우며 악천후 환경 및 야간 감지 시 성능이 크게 떨어짐

## 2. Overview of LiDAR

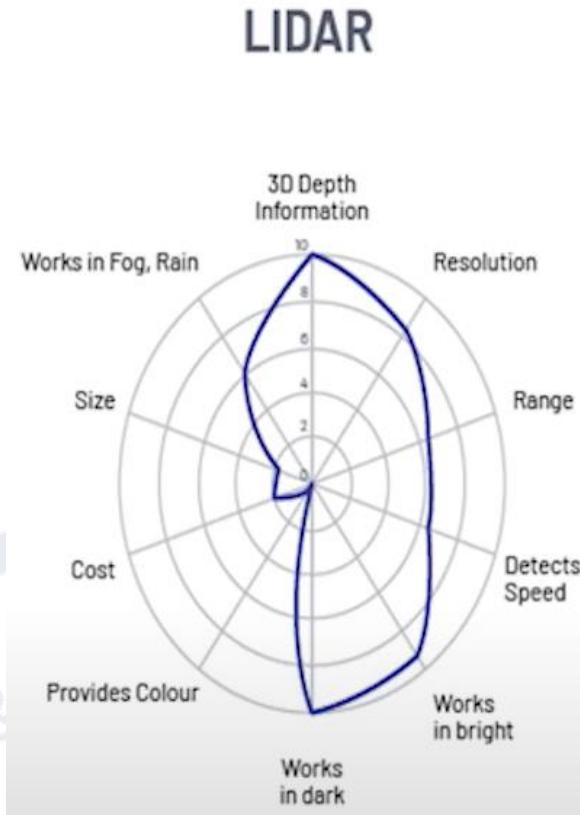
### 1) LiDAR 센서란?



- Radar 센서는 감지 대상의 거리, 속도를 정확하게 측정할 수 있으며, 전자기파를 이용하기 때문에 야간, 악천후 상황에서 열화가 적지만, 낮은 해상도로 인해 객체 구분 및 인식은 제한적

## 2. Overview of LiDAR

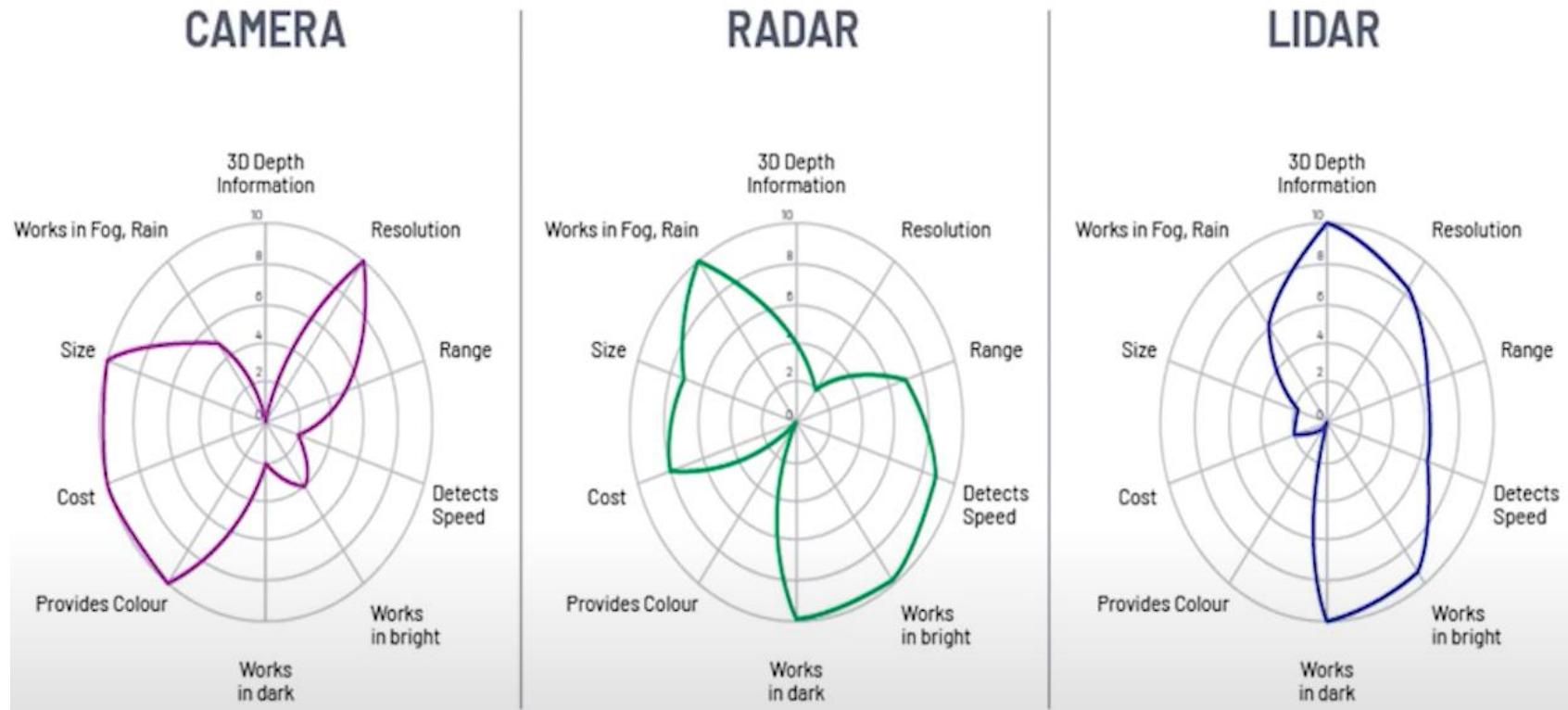
### 1) LiDAR 센서란?



- LiDAR 센서는 카메라 센서 대비 감지 거리가 뛰어나고 야간에도 감지 성능 열화가 전혀 없으며, 레이더 센서 대비 고해상도 감지 및 객체 인식이 가능함으로써 두 센서의 단점을 보완할 수 있음

## 2. Overview of LiDAR

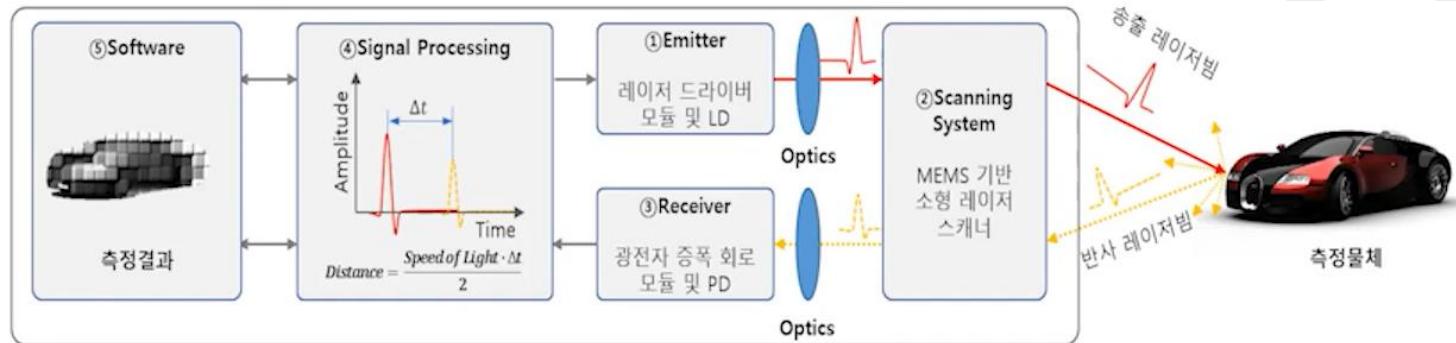
### 1) LiDAR 센서란?



- 따라서, 3단계 이상의 자율주행 기능 구현 시 전방 및 측/후방 감지 신뢰성 확보를 위해 기존의 Camera, Radar 센서와 함께 LiDAR 센서를 추가로 적용이 필요

## 2. Overview of LiDAR

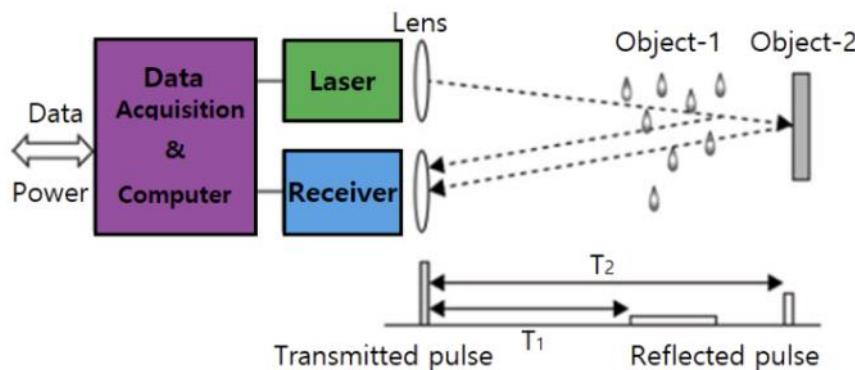
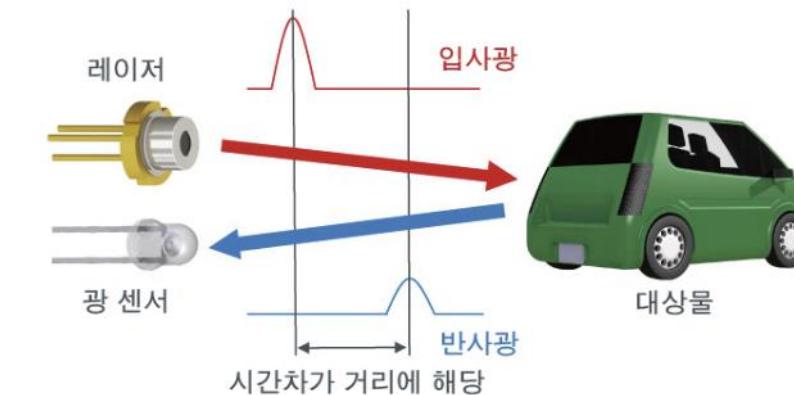
### 1) LiDAR 센서란?



- LiDAR 센서는 자율주행 자동차, 모바일 로봇, 드론 및 산업용 기기 등의 응용 분야에서 핵심 센서로 활용됨
- LiDAR 센서 시스템은 크게 측정 방식(measurement process), 발광 소자(emitter laser), 수광 소자(receiver photodetector), 빔 스티어링(beam steering)의 4가지 핵심 기술로 구성됨
- LiDAR 센서의 측정 방식으로는 비행 시간을 측정해 거리를 계산하는 ToF(Time-of-Flight) 방식과 주파수의 위상 변위를 이용해 거리를 측정하는 FMCW(Frequency Modulated Continuous Wave) 방식이 대표적으로 존재

## 2. Overview of LiDAR

### 2) LiDAR 센서의 원리



자료 출처. [https://freecon.co.kr/?act=board&bbs\\_code=solu&bbs\\_mode=view&bbs\\_seq=22](https://freecon.co.kr/?act=board&bbs_code=solu&bbs_mode=view&bbs_seq=22)

$$d = \frac{c \times t}{2}$$

- **d:** 목표물까지의 거리
- **c:** 빛의 속도  
(공기 중에서 대략 299,792,458 m/s)
- **t:** 레이저 빔이 목표물에 도달 후,  
다시 돌아오는데 걸린 전체 시간

- LiDAR 센서는 레이저 빛을 조사한 뒤, 그 빛이 물체에 반사되어 돌아오는 반사광의 비행 시간을 측정하여 대상체까지의 거리를 측정하는 원리를 가짐

## 2. Overview of LiDAR

### LiDAR 센서의 거리 측정 방식

- ToF 측정 방식은 송수신 소자 및 시스템 구성이 단순하여, 시스템 구현을 위한 기술 난이도가 낮고 저가로 구성이 가능하지만, 신호 처리를 통한 감지 성능(SNR) 향상에 한계가 있고, 짧은 시간에 고출력 펄스 형태로 광원을 수신해야 한다는 특징이 있음
- FMCW 측정 방식은 주파수 변조를 통해 일관된(Coherent) 방식의 신호 처리가 가능하며 감지 성능(SNR)을 극대화 할 수 있고, 도플러 효과를 통해 속도 검출이 가능하다는 특징이 있음. 그러나, 주파수 변조를 위해 시스템 구성이 복잡하고, 전장용 신뢰성 확보가 어려우며, 소형화 및 저가화에 한계가 있다는 특징이 있음.

## 2. Overview of LiDAR

### LiDAR 센서의 거리 측정 방식



\* ToF(time-of-flight) 비행 시간을 측정해 거리를 계산 방법

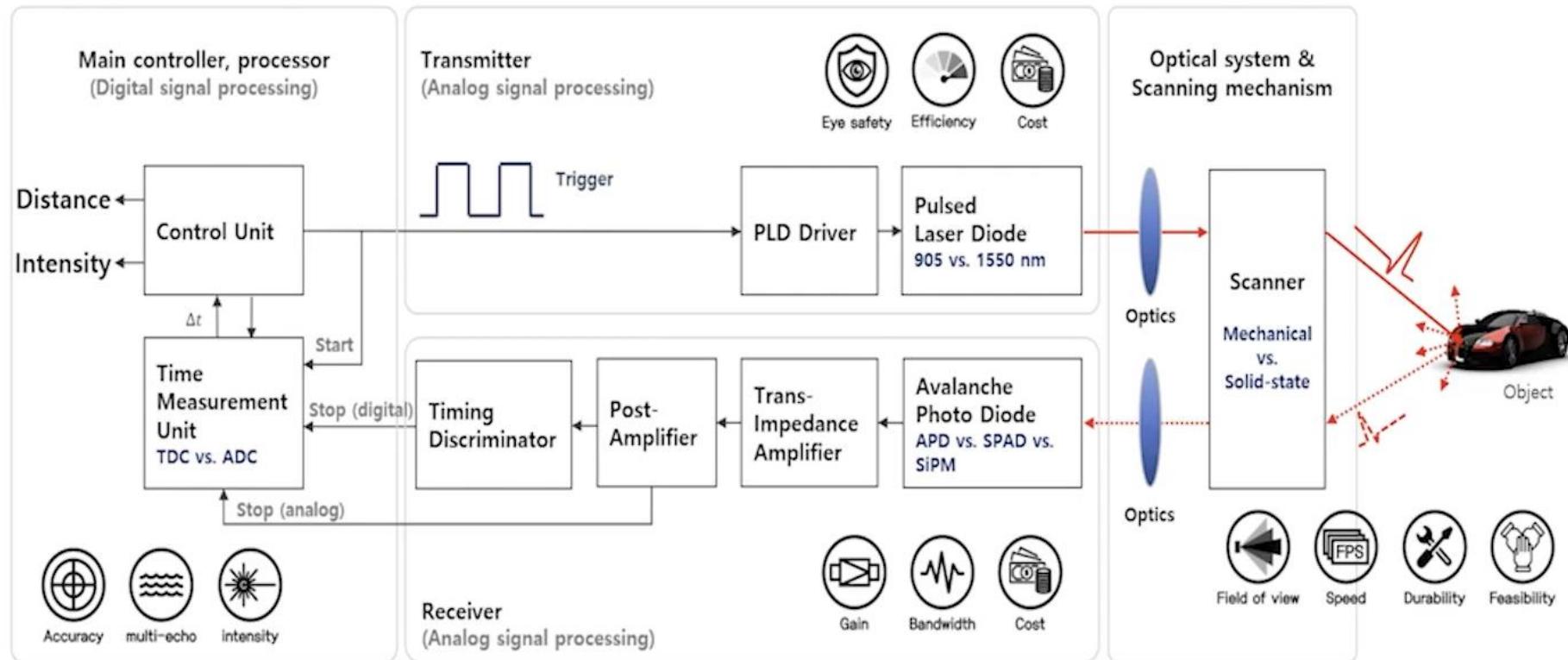
\* FMCW(Frequency Modulated Continuous Wave) 주파수의 위상 변위를 이용해 거리를 측정하는 방법

거리 계산	원리	신호처리 (SNR 개선)	시스템 복잡도 및 가격	차량용 신뢰성	속도 계산
TOF*	단순 시간 계산	열세	우세	우세	열세
FMCW*	주파수(파장) 변조	우세	열세	열세	우세

Copyright © 2021 SOSLAB, Professional LiDAR Sensor Provider All rights reserved.

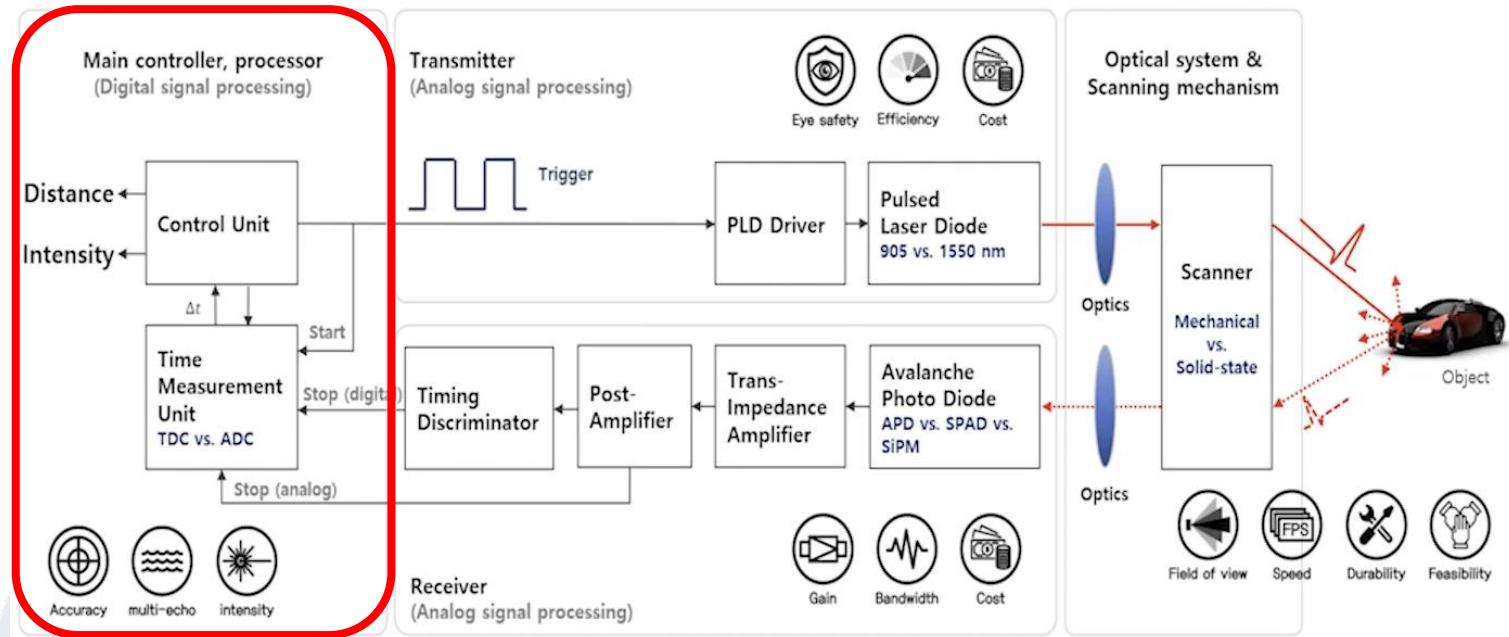
## 2. Overview of LiDAR

### 3) LiDAR 센서의 구조



## 2. Overview of LiDAR

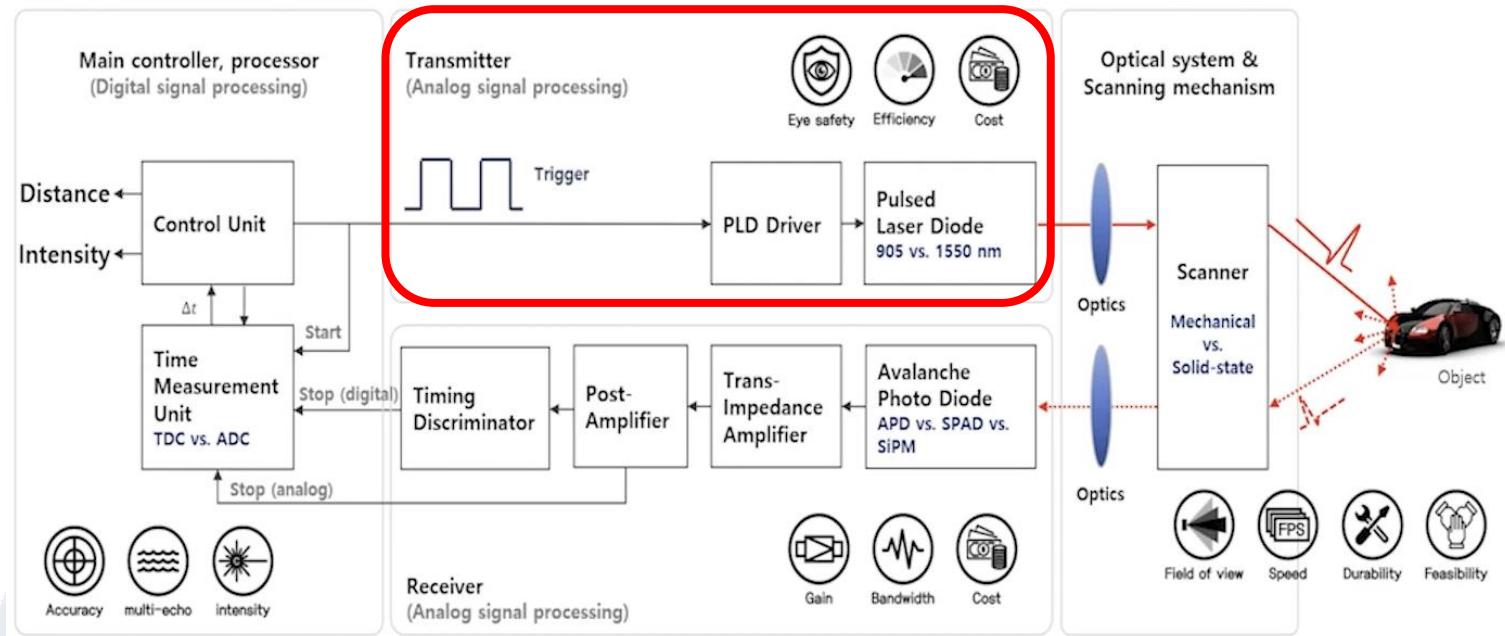
### 3) LiDAR 센서의 구조 – Main controller



- 역할: Main controller는 LiDAR 센서의 모든 작동을 제어하고, 데이터 처리를 관리하며, 시스템 간의 통신을 조정
- 기능: 시스템의 타이밍을 제어하고, 수신된 데이터를 분석하여 사용자가 원하는 형태로 데이터를 처리하는 역할을 수행

## 2. Overview of LiDAR

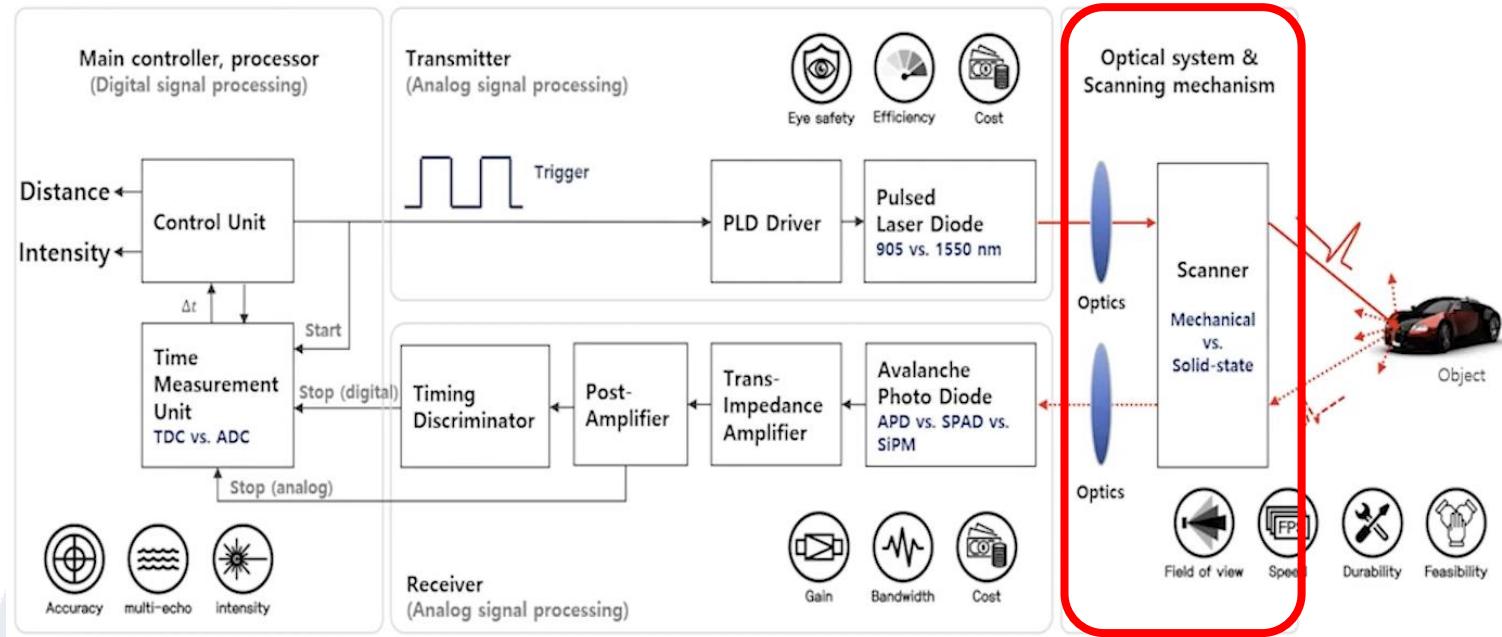
### 3) LiDAR 센서의 구조 – Transmitter(발신기)



- 역할: Transmitter는 발광 소자(emitter)를 통해 레이저 빔을 발사하여 물체에 도달하도록 하며, 물체에 반사되어 Receiver로 돌아오는 방식으로 거리를 측정
- 특성: 일반적으로 레이저 다이오드가 사용되며, 파장은 주로 적외선 범위에서 선택됨

## 2. Overview of LiDAR

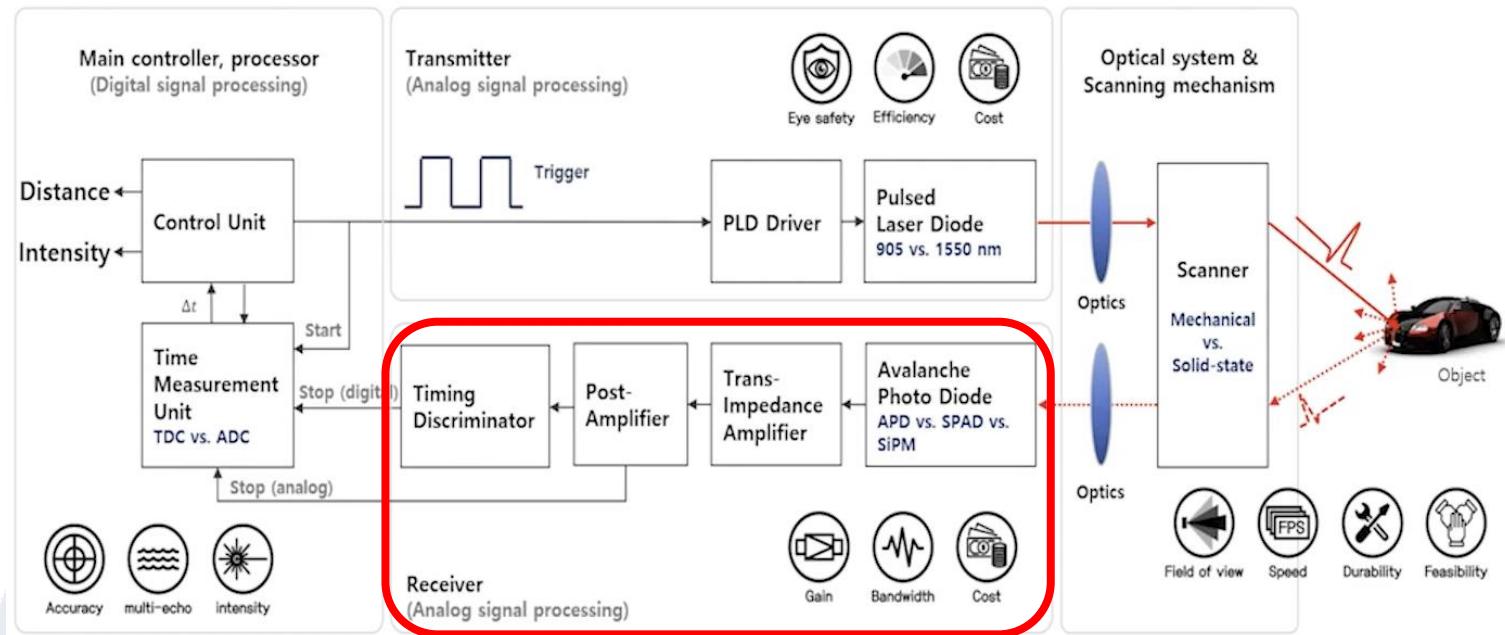
### 3) LiDAR 센서의 구조 – Scanner(스캐너)



- 역할: Scanner는 발사된 레이저 빔의 방향 조절을 수행하며, 이를 통해 LiDAR는 주변 환경을 360도 또는 필요한 각도만큼 스캔 가능. 이러한 기술을 ‘빔 스티어링 (Beam Steering)’이라고 함
- 기능: Scanner는 Spinning과 같은 기계적 회전 방식이나 MEMS Mirror와 같은 전자적 방식으로 구현될 수 있으며, 스캔 범위와 정밀도는 사용되는 Scanner의 종류에 따라 달라질 수 있음

## 2. Overview of LiDAR

### 3) LiDAR 센서의 구조 – Receiver(수신기)



- 역할: Receiver는 수광 소자(photodetector)를 통해 반사된 레이저 빔을 감지하고 수신하며, 해당 정보를 바탕으로 물체의 위치, 형태, 그리고 거리에 대한 계산이 가능
- 기능: Receiver는 고감도의 광학 센서를 사용하여 아주 약한 빛의 신호도 포착 가능

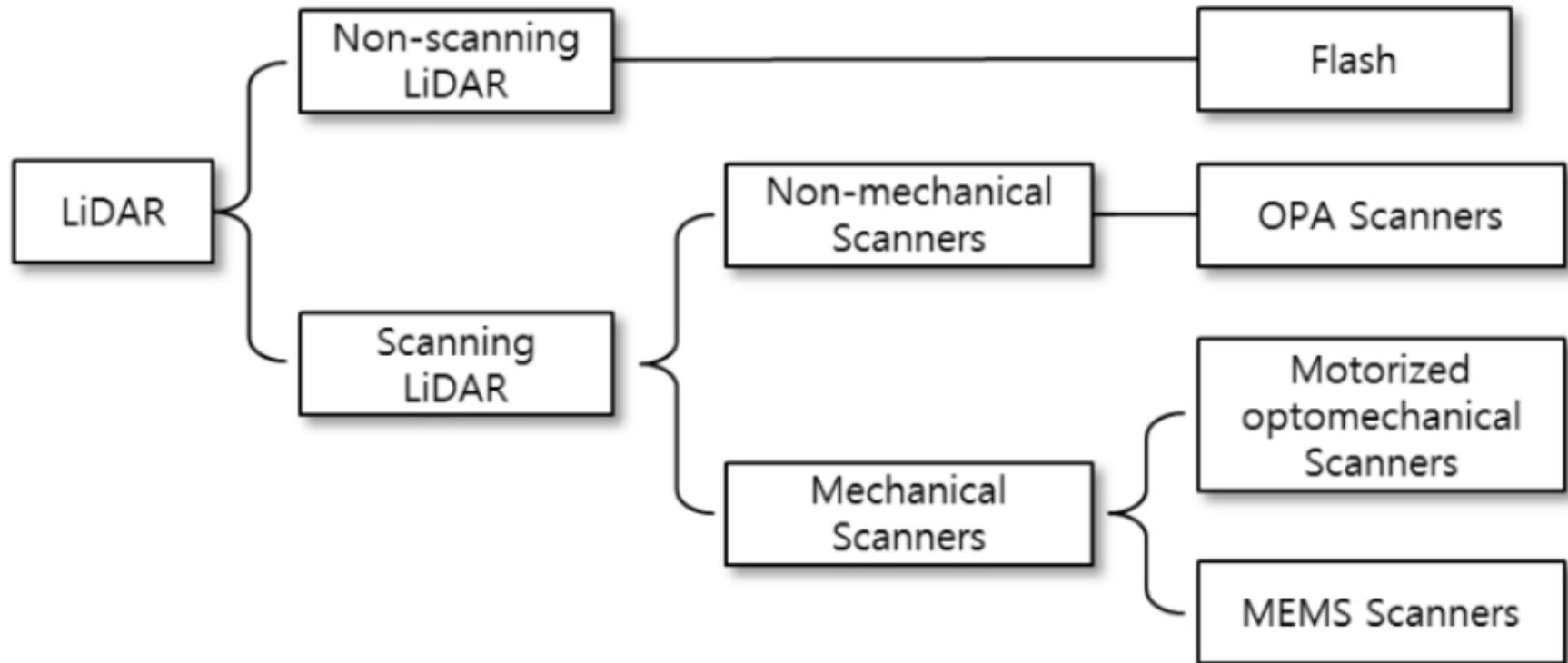
## 2. Overview of LiDAR

### LiDAR 센서의 기술 구성 요소

- 발광 소자(emitter)는 목표 물체로 레이저 빛을 발사하는 역할을 수행
- 수광 소자(photodetector)는 반사되어 돌아오는 빛을 검출하는 역할을 수행
- 빔 스티어링(beam steering)은 발광 소자로부터 발사된 LiDAR 센서의 레이저 빛을 스캐너(Scanner)를 통해 특정 방향으로 정밀하게 조절하는 역할을 수행
- LiDAR 시스템에는 빛을 원하는 방향으로 조사할 수 있는 기계식(mechanical), 고정형(solid-state) 스캐너 장치들을 이용해 특정 공간 또는 목표 지점에 대해 주기적으로 측정을 반복하여 주변 환경에 대한 3차원 공간 정보를 획득

## 2. Overview of LiDAR

### 4) 스캐닝 방식에 따른 LiDAR 센서의 종류



자료 출처. <https://veganwithbacon.tistory.com/324>

## 2. Overview of LiDAR

### LiDAR 센서의 스캐닝 방식 – 기계식(mechanical) 스캐닝

- LiDAR 센서의 스캐닝 방식은 기계식 회전(mechanical) 방식과 고정형(solid-state) 방식으로 구분됨
- 기계식 회전 방식은 송수신 모듈을 모터에 장착하여 직접 회전하는 방식과, 송수신 모듈은 고정하고 반사 거울을 모터에 장착하여 스캐닝하는 방식이 대표적
- 두 방식 모두 시스템 구성에 필요한 기술 난이도가 상대적으로 낮기 때문에 현재 제품화 되어 판매 중인 LiDAR 센서는 모두 기계식 회전 방식
- 그러나 기계식 회전 방식의 LiDAR 센서는 스캐닝을 위한 모터 및 구조물이 추가되어 사이즈가 크고, 고가의 재료비가 필요하다는 한계점을 가짐

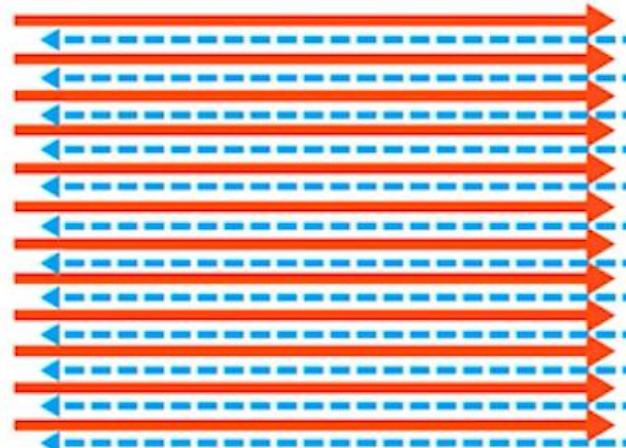
## 2. Overview of LiDAR

### LiDAR 센서의 스캐닝 방식 – 고정식(solid-state) 스캐닝

- 고정형 스캐닝 방식은 전자기적 공진 특성을 이용한 MEMS Mirror 스캐닝 방식과, 스캐너 없이 직접 송수신을 하는 Flash 방식으로 구분
- MEMS Mirror 방식은 스캐너 사이즈가 작고 상대적으로 저렴한 단가로 구성할 수 있지만, 기계적 회전 방식 대비 감지 성능이 떨어지며, 차량용 진동/내구 신뢰성에 취약함
- Flash 방식은 스캐너가 없기 때문에 사이즈, 단가, 신뢰성 측면에서 우수하지만, 감지 성능은 가장 떨어짐
- Optical Phase Array (OPA) 방식은 기술 성숙도가 낮아 자율주행 차량용으로는 고려하지 않음

## 2. Overview of LiDAR

### LiDAR 센서의 스캐닝 방식 – Spinning LiDAR 예시



광원/수광 모듈 세트  $\times N$  개: N 채널 라이다



## 2. Overview of LiDAR

LiDAR 센서의 스캐닝 방식 – Spinning LiDAR 예시



## 2. Overview of LiDAR

### LiDAR 센서의 스캐닝 방식 – Spinning LiDAR의 특징

#### 장점

- 한 줄로 정렬된 레이저 소자들이 모터를 통해 360도 회전함으로써 전방향에 대한 거리를 측정할 수 있어 고성능을 가짐
- 한 줄에 레이저 소자를 16개 집적하는 경우 16채널, 32개를 배치하는 경우 32채널의 해상도로 분류됨

#### 단점

- Spinning LiDAR는 많은 소자를 필요로 하므로 시스템 구성 비용이 높아짐
- 많은 소자를 필요로 하는 시스템의 특성으로 인해 소형화가 어려움
- 모터가 직접 회전하는 방식의 한계점으로 인해 내구도가 쉽게 저하될 수 있음
- 360도를 회전하는 LiDAR의 구조상, 자동차 범퍼 내 탑재 불가

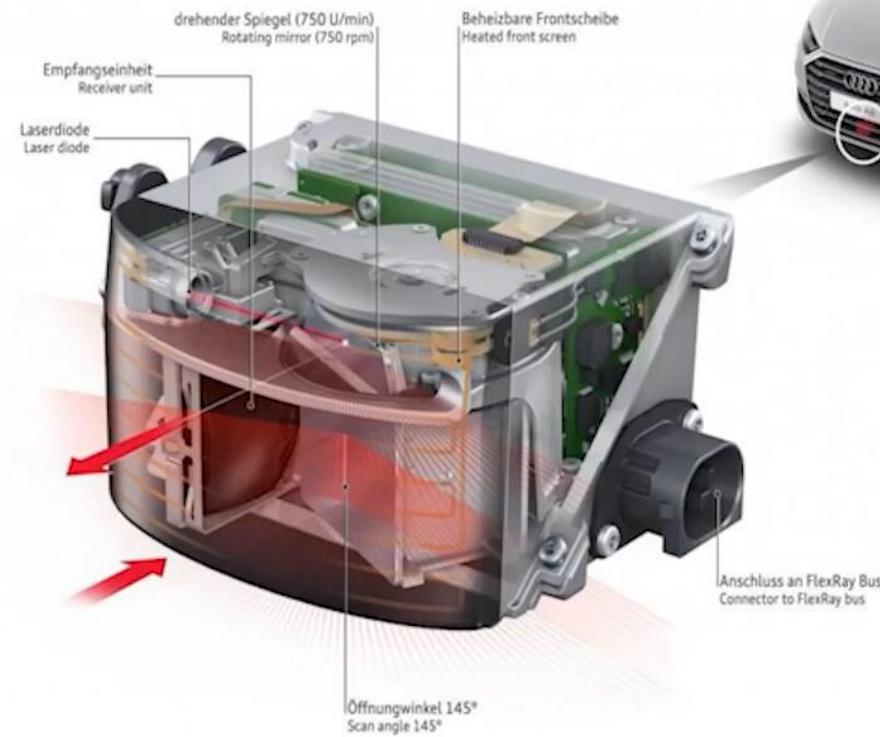
## 2. Overview of LiDAR

### LiDAR 센서의 스캐닝 방식 – Polygon Mirror Scanning LiDAR 예시



**ibeo**  **Valeo**  
automotive

  
**Audi**



## 2. Overview of LiDAR

### LiDAR 센서의 스캐닝 방식 – Polygon Mirror Scanning LiDAR 특징



#### 장점

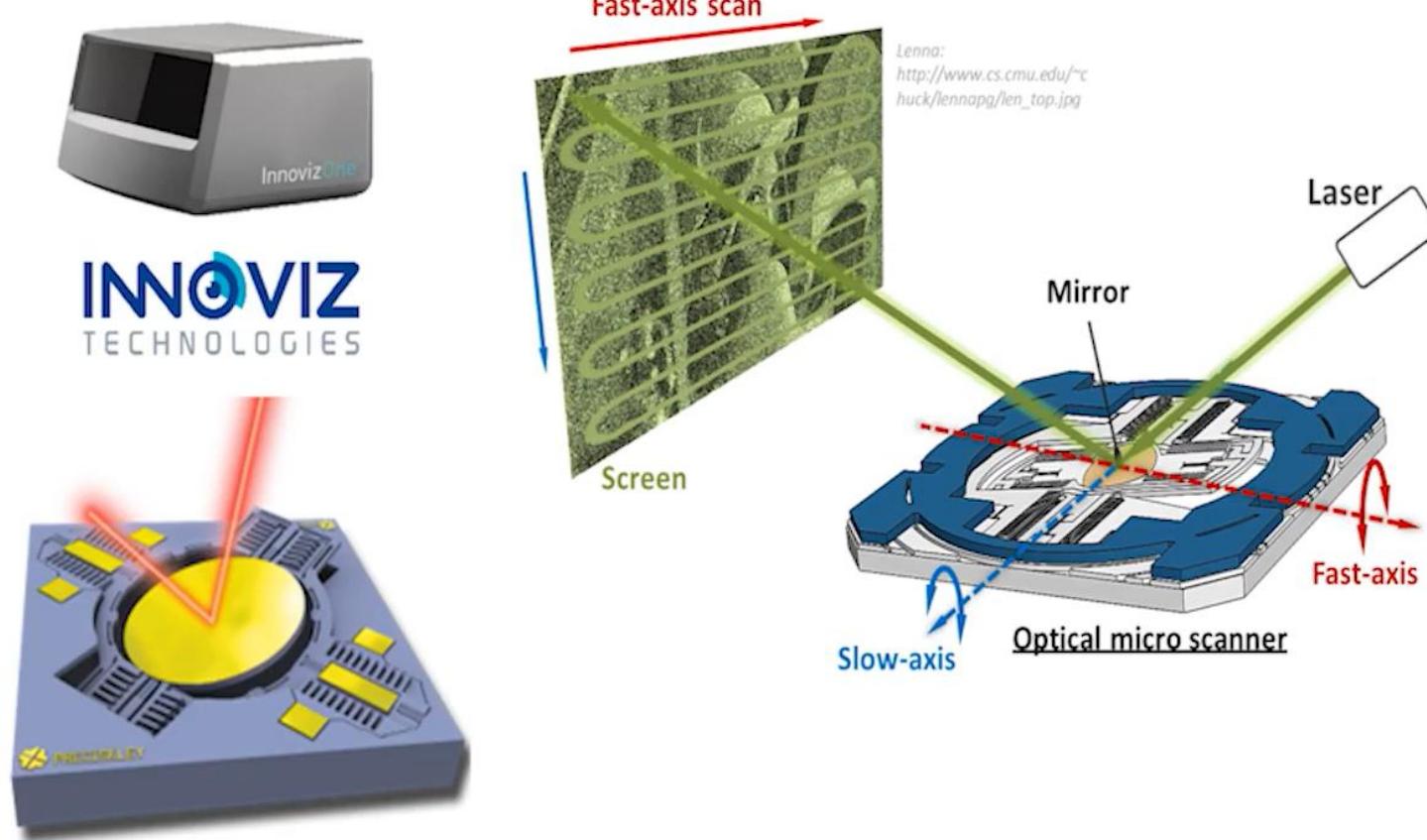
- 레이저 소자들이 직접 회전하는 것이 아닌 거울과 같은 반사체가 회전하여 거리 데이터를 취득함으로써, 최소한의 레이저 소자를 바탕으로 주변 환경에 대한 스캐닝이 가능
- 최소한의 레이저 소자를 기반으로 구성되므로 소형화가 가능해져, 차량에도 적용 가능

#### 단점

- 여전히 모터가 직접 회전하는 방식의 한계점으로 인해 내구도가 쉽게 저하될 수 있음
- Spinning LiDAR 대비 해상도가 낮음

## 2. Overview of LiDAR

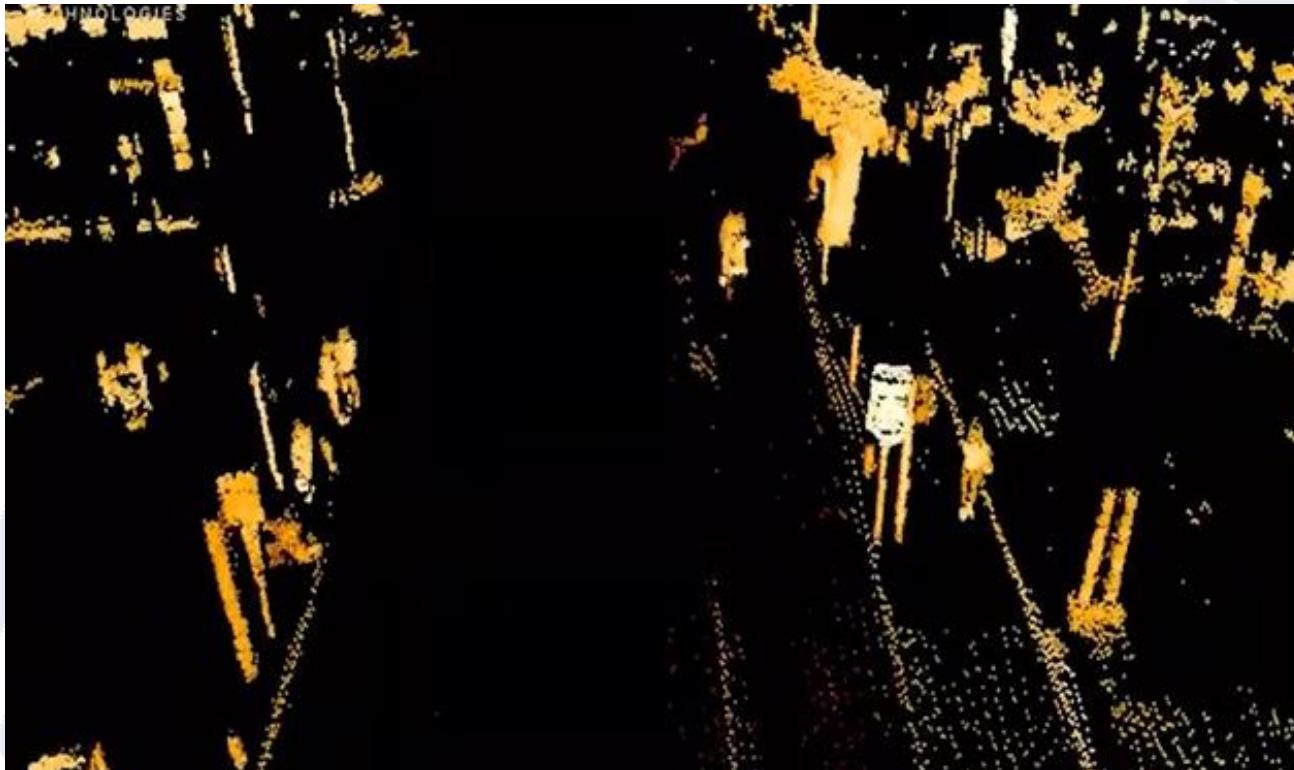
### LiDAR 센서의 스캐닝 방식 – MEMS Mirror Scanning LiDAR 예시



- MEMS actuator라는 아주 작은 actuator를 이용하여 레이저 신호를 2축으로 조사함으로써 면 광원을 생성하는 기술

## 2. Overview of LiDAR

LiDAR 센서의 스캐닝 방식 – MEMS Mirror Scanning LiDAR 예시



- 이스라엘의 ‘이노비즈’사에서 제작한 MEMS Mirror Scanning 방식의 LiDAR 예시

## 2. Overview of LiDAR

### LiDAR 센서의 스캐닝 방식 – MEMS Mirror Scanning LiDAR 특징

#### 장점

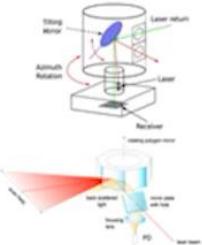
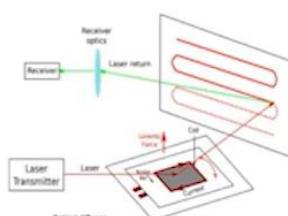
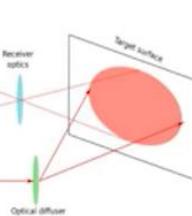
- 적은 레이저를 바탕으로 고해상도의 3차원 데이터 취득 가능 (Polygon Mirror LiDAR 대비)
- 소형화, 생산성 향상, 가격 절감 등을 통해 양산성 확보가 가능 (Spinning LiDAR 대비)

#### 단점

- 화각이 작아 멀티 광원 사용 및 광정렬 문제 발생
- 저속 축에 대한 내구성 이슈가 존재

## 2. Overview of LiDAR

### 스캐닝 방식에 따른 LiDAR 센서의 특징 정리

스캐닝방식	기계식 회전	MEMS Mirror	Flash
시스템 구성			
크기 및 가격	열세	보통	우세
차량용 신뢰성	보통	열세	우세
해상도	보통	보통	우세
감지 거리	우세	보통	열세

Copyright © 2022 SOSLAB, Professional LiDAR Sensor Provider All rights reserved.

- 기계식 회전 방식은 긴 감지 거리와 높은 해상도를 가지지만, 크기 및 가격 측면에서 제품화가 어렵다는 한계점을 가지고 있음.
- MEMS Mirror 또는 Flash 방식을 통해 크기 및 가격을 낮춤으로써 실제 차량에 들어갈 수 있는 형태의 LiDAR 센서에 대한 개발이 활발히 진행되고 있음

## 2. Overview of LiDAR

### 5) 자율주행 기술 실현을 위한 LiDAR 센서의 역할



- 차량 주변의 동적인 환경을 측정하고 주변 환경을 묘사
- 보행자, 차량, 자전거, 도로 표면, 차선 경계, 교통 신호 등을 인지
- 인지 정보를 바탕으로 주행에 대한 제어 방식을 계획하고 결정
- 다양한 센서(Camera, Radar, LiDAR)와 딥러닝 기술을 활용

## 2. Overview of LiDAR

### Camera 기반 객체 인식 기술의 한계

- 객체 인식 및 추적 알고리즘은 대부분 2D Camera 센서를 기반으로 개발되어 왔으며, 상용화된 자율주행 2단계 차량에 적용되어 Lane Departure Warning (LDW), Lane Keeping Assist(LKA), Traffic Sign Recognition (TSR) 등의 ADAS 기능을 수행
- Camera 센서 기반 객체 인식 알고리즘은 100M 픽셀 이상급의 고해상도 데이터를 바탕으로 높은 인식률을 얻을 수 있으며, 기존에 촬영된 수 많은 이미지를 바탕으로 보행자, 차량, 이륜차, 차선 등의 다양한 객체를 딥러닝(deep learning) 기반의 학습을 통해 성능을 지속해서 향상할 수 있다는 장점이 있음
- 하지만 조도 환경이 매우 높거나 (예: 태양광이 Camera 센서로 직접 조사되는 경우 등) 낮은 경우 (예: 야간, 터널 환경 등) 인식률이 현저히 떨어지며, 인식에 실패했을 경우 정확도 부족으로 인해 자율주행 기능이 정지되거나 사고가 발생하는 상황을 초래할 수 있음

## 2. Overview of LiDAR

### LiDAR 기반 객체 인식 기술

- LiDAR 센서 출력인 3차원 점 군(3D Point-Cloud)을 기반으로 하는 객체 인식 알고리즘은 Camera 센서에 비해 낮은 해상도를 사용하므로 인식률은 다소 떨어지나, 인식률 성능이 조도 환경에 크게 영향을 받지 않으며 객체 인식에 실패하더라도 정확한 거리 데이터를 이용하여 사고를 방지할 수 있다는 장점이 있음
- 최근에는 딥러닝 기반의 방법들로 대상 객체에 대한 3차원 공간 상의 정확한 위치, 거리, 이동 방향 등의 추정이 가능해지고 있어 Camera 센서 기반 객체 인식 기술의 한계점을 보완하는 형태로 활용되고 있음
- 또한 딥러닝 기반 객체 인식 기술의 성능을 높이기 위해 더 많은 LiDAR 데이터셋의 확보와 딥러닝 모델의 개선을 위한 연구도 활발하게 진행되고 있음
- 이러한 관점에서 라이다 점 군 데이터를 이용한 객체 인식 기술은 3단계 이상의 자율주행 기능 구현에 필수적임

## 2. Overview of LiDAR

LiDAR 기반 객체 인식 기술 예시



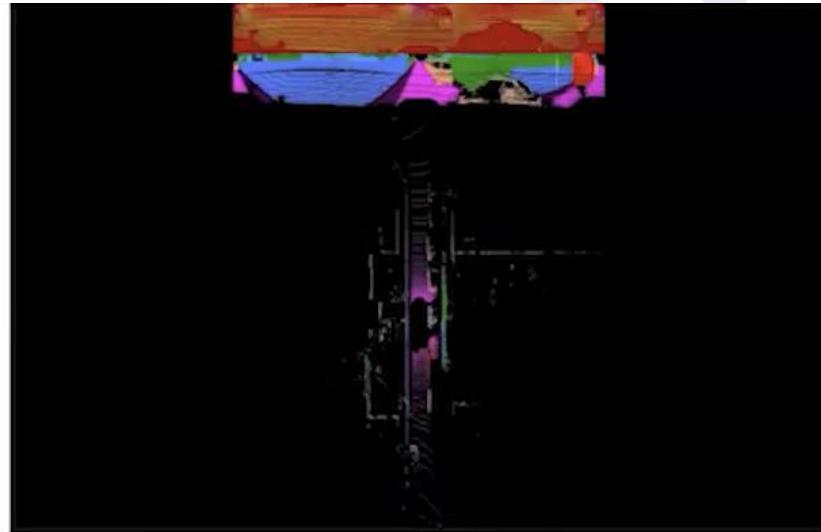
- 국내 자율주행 스타트업 서울 로보틱스의 LiDAR 기반 객체 인식 기술 데모 영상
- 360도 회전이 가능한 Spinning LiDAR를 기반으로 객체 인식 수행

## 2. Overview of LiDAR

### LiDAR 기반 객체 인식 기술 최신 동향 – 인공지능 기반 객체 인식



라이다 3차원 객체 인식 기술 ©[NVIDIA](#)



라이다 3차원 의미론적 분할 기술 ©[NVIDIA](#)

- 최근에는 Camera 센서 기반의 객체 인식 분야 뿐만 아니라, LiDAR 센서를 기반으로 하는 객체 인식 분야에서도 딥러닝을 통한 특징 추출 기술들을 활용하여 높은 수준의 인지 성능을 보이는 알고리즘들이 개발되고 있음
- 딥러닝 모델의 성능은 데이터셋의 양과 품질에 좌우되기 때문에 Waymo, Uber ATG 등 자율주행 기업들을 중심으로 더 많은 LiDAR 데이터셋의 확보를 위한 연구가 활발하게 진행되고 있음

## 2. Overview of LiDAR

### LiDAR 기반 객체 인식 기술 최신 동향 – 자율주행 데이터셋

구분	DATASET	Link
국 외	KITTI [Geiger2013]	<a href="http://www.cvlibs.net/datasets/kitti/">http://www.cvlibs.net/datasets/kitti/</a>
	NuScenes (Aptiv, Scale and nuTonomy Inc.)	<a href="https://scale.ai/open-datasets/nuscenes">https://scale.ai/open-datasets/nuscenes</a>
	Apollo Open Transform (Baidu Inc.)	<a href="http://data.apollo.auto/">http://data.apollo.auto/</a>
	Waymo Opendataset (Waymo)	<a href="https://waymo.com/open">https://waymo.com/open</a>
국 내	KAIST Urban (KAIST)	<a href="http://irap.kaist.ac.kr/dataset/">http://irap.kaist.ac.kr/dataset/</a>
	KODAS (LX 공간정보연구원)	<a href="http://kodas.or.kr/">http://kodas.or.kr/</a>
	HD Map Dataset (NAVERLABS)	<a href="https://hdmap.naverlabs.com/">https://hdmap.naverlabs.com/</a>



© The KITTI Vision Benchmark



© Lyft Level 5



© ApolloScape

- 대표적인 데이터셋인 KITTI 데이터셋 뿐만 아니라, Google Waymo의 오픈 데이터셋과 같은 다양한 데이터셋을 기반으로 인공지능을 통한 객체 인식 기술들이 빠르게 발전하고 있음

## 2. Overview of LiDAR

### LiDAR 기반 객체 인식 기술 최신 동향 – 자율주행 데이터셋



데이터 공유센터 (<http://www.kotsa.or.kr/avds>)

The screenshot shows the homepage of the AI Hub website. At the top, there is a navigation bar with links: AI 데이터찾기, AI 개발지원, 참여하기, 정보공유, 고객지원, AI 허브소개, 로그인, 회원가입, and a three-dot menu icon. Below the navigation bar, there are six colored boxes representing different data categories: 한국어 (74종), 영상이미지 (65종), 헬스케어 (51종), 재난안전환경 (46종), 농축수산 (29종), and 교통물류 (28종).

자율주행 공개 데이터셋 제공 (<https://aihub.or.kr/>)

- 국내 데이터 공유센터를 통해 자율주행 데이터셋을 수집 및 분석할 수 있도록 지원
- AIHub을 통해 국가 사업을 통해 수집한 다양한 데이터셋을 공개함으로써 인공지능 모델 개발의 속도를 높이고자 함

## 2. Overview of LiDAR

Quiz 3) LiDAR 시스템에서 레이저를 원하는 방향으로 조사(조향)하기 위한  
핵심 기술의 명칭은?

**빔 스티어링 (Beam Steering)**

**또는**

**스캐닝 (Scanning)**

## 2. Overview of LiDAR

### 학습 정리

- LiDAR 센서의 동작 원리 및 핵심 기술

- LiDAR 센서는 빛의 이동 시간을 측정하여 높은 정확도와 정밀도의 3차원 공간 정보를 고속으로 취득할 수 있는 능동형 센서

- LiDAR 시스템은 측정 방식, 발광 소자, 수광 소자, 빔 스티어링의 4가지 핵심 기술로 구성

- LiDAR 센서의 활용

- 자율주행 시스템에서는 차량, 보행자, 자전거 등과 같은 동적인 객체를 인지하기 위한 목적으로 LiDAR 센서를 활용

- 정확한 거리 인식을 통해 사고를 방지할 수 있어 3단계 이상 자율주행 기능 구현에 필수적

- LiDAR 학습용 데이터셋의 중요성

- 최근 딥러닝 기반의 LiDAR 객체 인식 방법들은 높은 성능을 보이며 활발하게 연구가 진행

- 딥러닝 모델의 성능은 데이터의 양과 품질에 좌우되기 때문에 양질의 데이터셋을 구축하는 것이 매우 중요

### **3. Basics of Python grammar**

### 3. Basics of Python grammar

#### 1) Python이란?



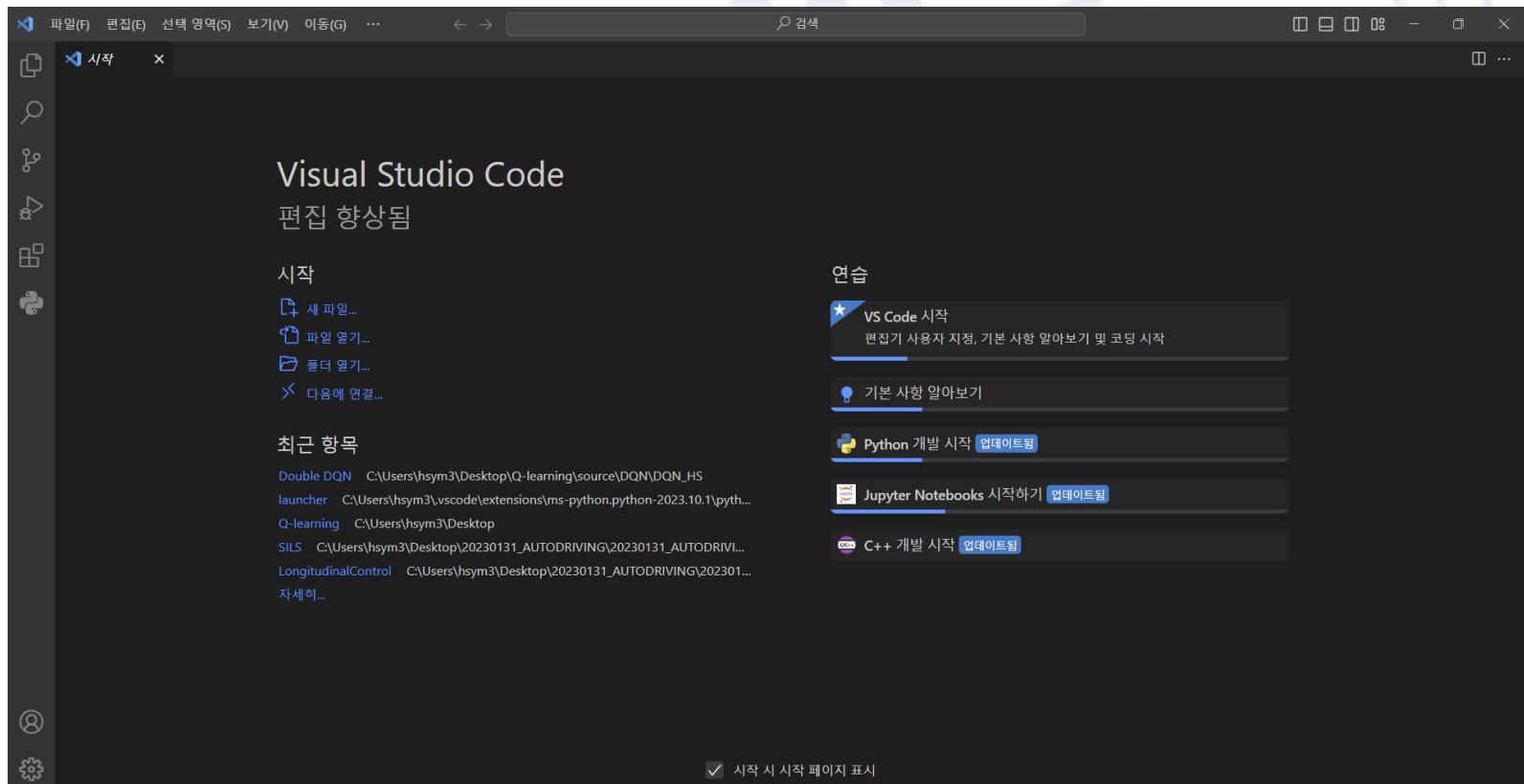
- Python(파이썬)은 코드의 가독성을 높이고, 작업을 보다 간단하게 수행할 수 있도록 1991년 '귀도 반 로섬(Guido van Rossum)'에 의해 개발된 프로그래밍 언어
- Python은 interpreter(인터프리터) 언어로, 코드를 작성한 후 별도의 컴파일 과정 없이 바로 실행이 가능하여 개발 과정을 더 빠르고 유연하게 함
- 오픈 소스 언어로서 수많은 라이브러리와 프레임워크가 개발되어 다양한 작업을 쉽고 효율적으로 처리 가능

# 3. Basics of Python grammar

## 2) 개발 환경 설정

### ▪ 개발 환경(IDE) : Visual Studio Code (VS Code)

- VS Code는 마이크로소프트에서 개발한 경량화 된 텍스트 에디터이며,  
확장성, 디버깅 툴, Git 통합 등의 다양한 기능을 바탕으로 원활한 개발 환경 구축 가능.



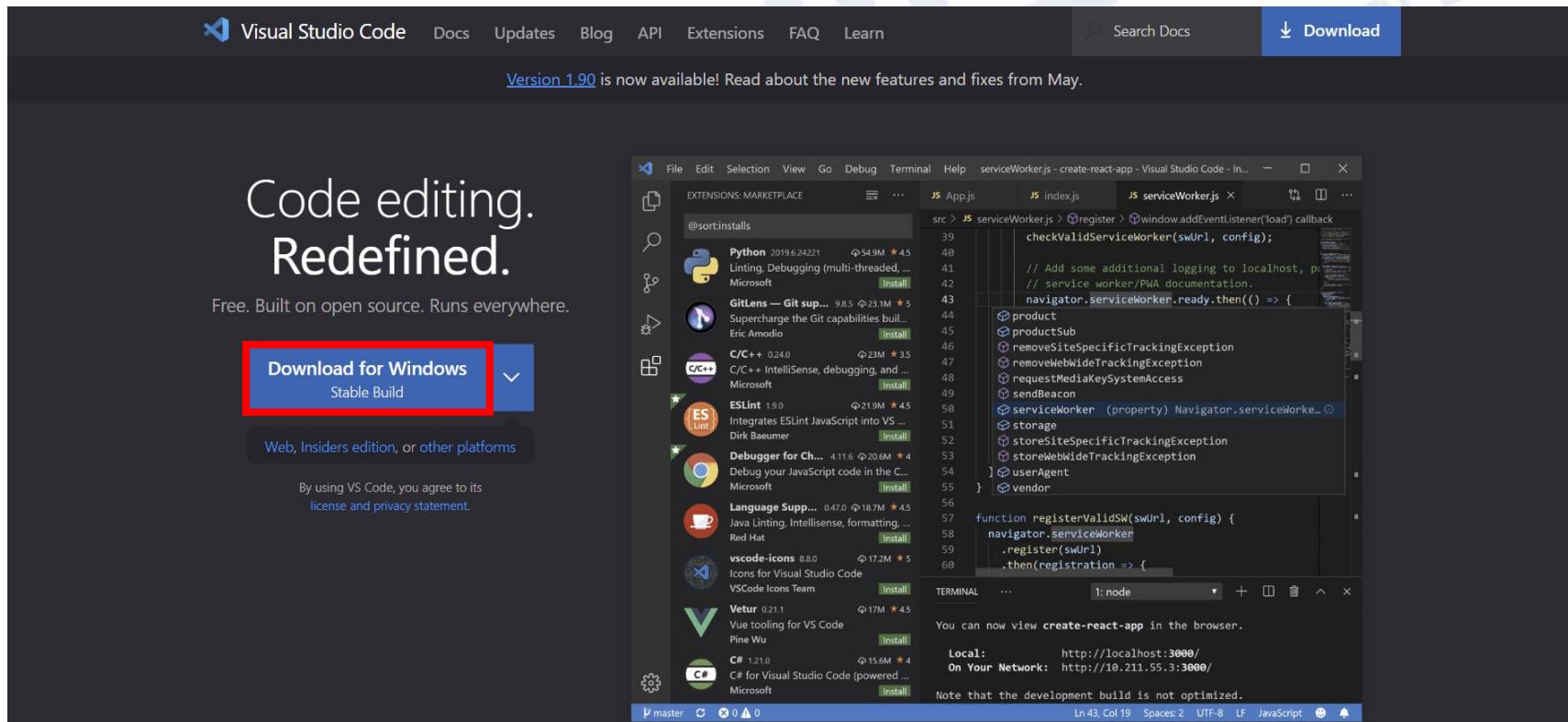
# 3. Basics of Python grammar

## 2) 개발 환경 설정

- 개발 환경(IDE) 설치 : Visual Studio Code (VS Code) [1/2]

- VS Code download link : <https://code.visualstudio.com/>

- Download link에서 Download for Windows를 클릭하여 설치 파일을 다운로드

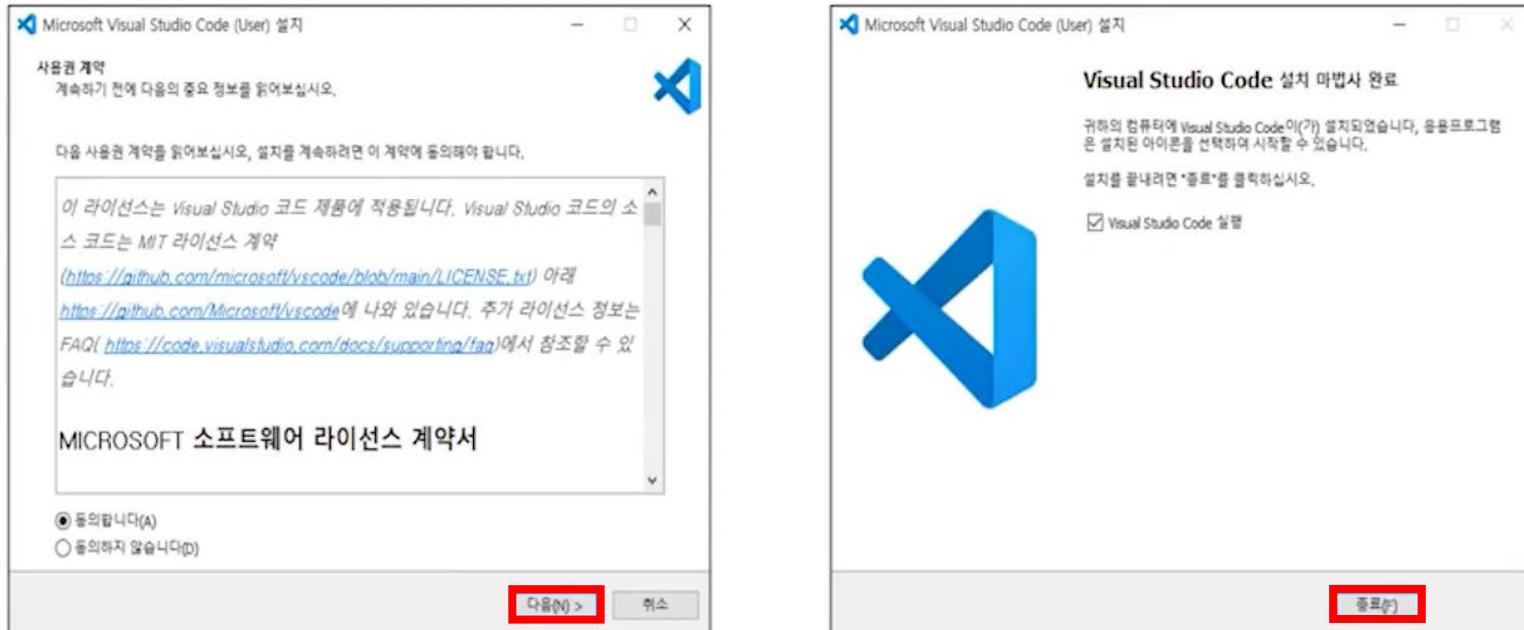


### 3. Basics of Python grammar

#### 2) 개발 환경 설정

- 개발 환경(IDE) 설치 : Visual Studio Code (VS Code) [2/2]

- VS Code 설치 파일(VSCodeUserSetup-x64-1.xx.x.exe)을 실행
- 사용권 계약의 “동의합니다” 선택 후, 다음 버튼을 클릭하여 설치를 진행
- 설치 파일에서 제공하는 기본 설정으로 설치를 완료



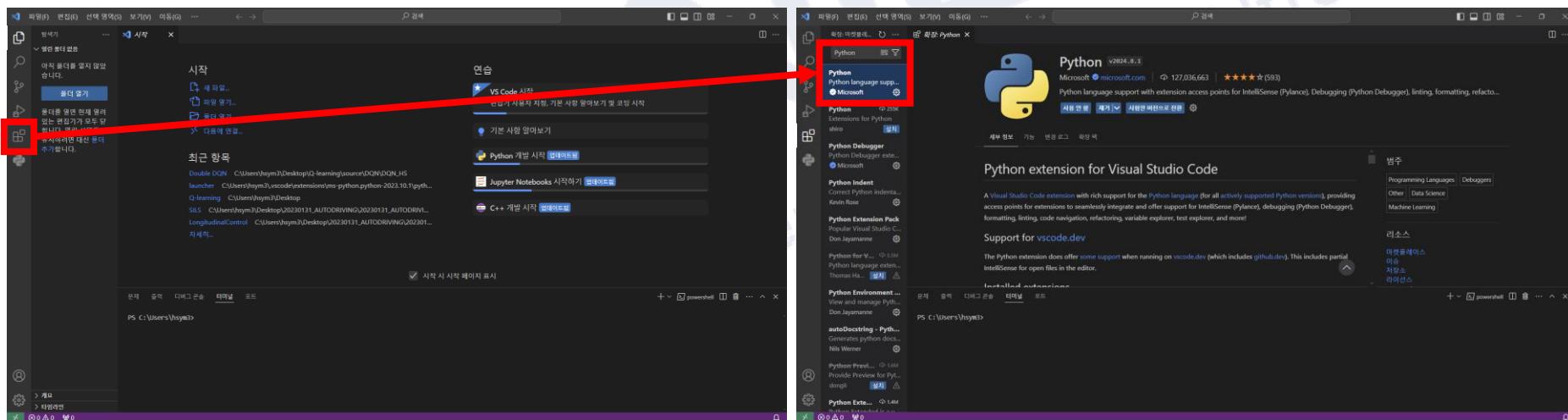
# 3. Basics of Python grammar

## 2) 개발 환경 설정

- VS Code 설정 : Python 기능 설치

- VS Code를 실행한 뒤, 왼쪽 아이콘에서 Extensions 를 클릭

- Extensions 창에서 Python을 검색 후, “install”을 클릭하여 VS Code의 Python 기능을 설치



- VS Code 개발 환경 내에 Python 언어를 인지할 수 있는 기능을 추가하는 과정
- 실제로 Python 언어를 설치한 것은 아님에 유의

# 3. Basics of Python grammar

## 2) 개발 환경 설정

### ▪ Python 설치 [1/3]

- Python download link : <https://www.python.org/downloads/windows/>

- Download link에서 파이썬 3.7.9(버전 무관) - Windows x86-64 executable installer를 다운로드

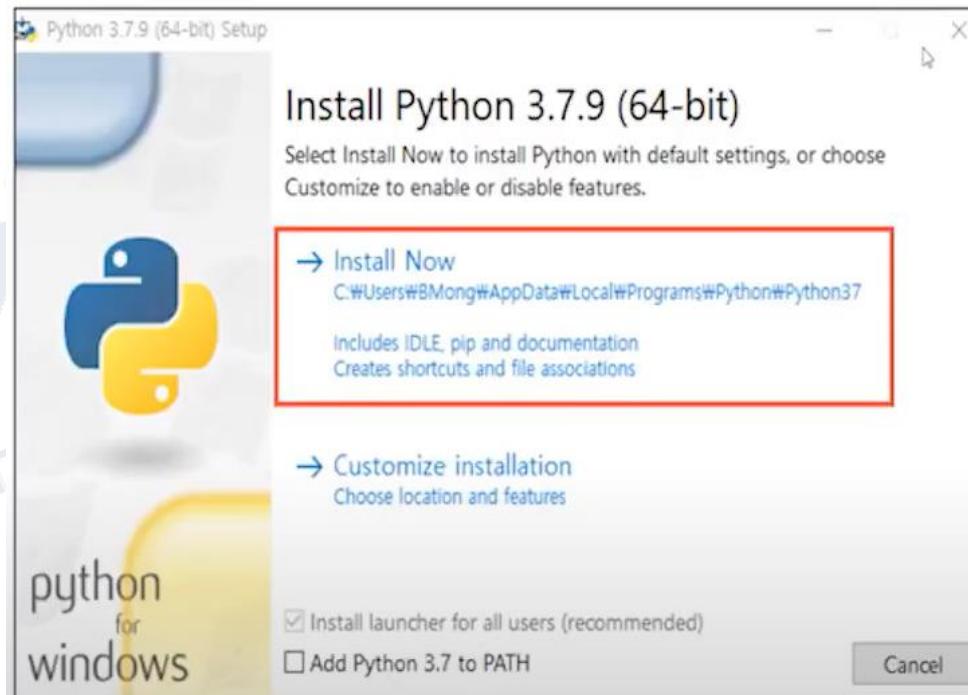
- [Download Windows x86-64 embeddable zip file](#)
- [Download Windows x86-64 executable installer](#)
- [Download Windows x86-64 web-based installer](#)
- [Python 3.5.10 - Sept. 5, 2020](#)  
**Note that Python 3.5.10 cannot be used on Windows XP or earlier.**
  - No files for this release.
- [Python 3.7.9 - Aug. 17, 2020](#)  
**Note that Python 3.7.9 cannot be used on Windows XP or earlier.**
  - [Download Windows help file](#)
  - [Download Windows x86 embeddable zip file](#)
  - **[Download Windows x86 executable installer](#)**
  - [Download Windows x86 web-based installer](#)
  - [Download Windows x86-64 embeddable zip file](#)
  - [Download Windows x86-64 executable installer](#)
  - [Download Windows x86-64 web-based](#)
- [Python 3.9.0a2 - Dec. 18, 2019](#)
  - [Download Windows help file](#)
  - [Download Windows x86 embeddable zip file](#)
  - [Download Windows x86 executable installer](#)
  - [Download Windows x86 web-based installer](#)
  - [Download Windows x86-64 embeddable zip file](#)
  - [Download Windows x86-64 executable installer](#)
  - [Download Windows x86-64 web-based](#)
- [Python 3.7.6rc1 - Dec. 11, 2019](#)
  - [Download Windows help file](#)
  - [Download Windows x86 embeddable zip file](#)
  - [Download Windows x86 executable installer](#)
  - [Download Windows x86 web-based installer](#)
  - [Download Windows x86-64 embeddable zip file](#)
  - [Download Windows x86-64 executable](#)

### 3. Basics of Python grammar

#### 2) 개발 환경 설정

- Python 설치 [2/3]

- 설치 파일을 실행하여 아래 그림과 같이 **Install launcher for all users (recommended)**를 체크
- **Install Now**를 클릭하여 설치를 진행

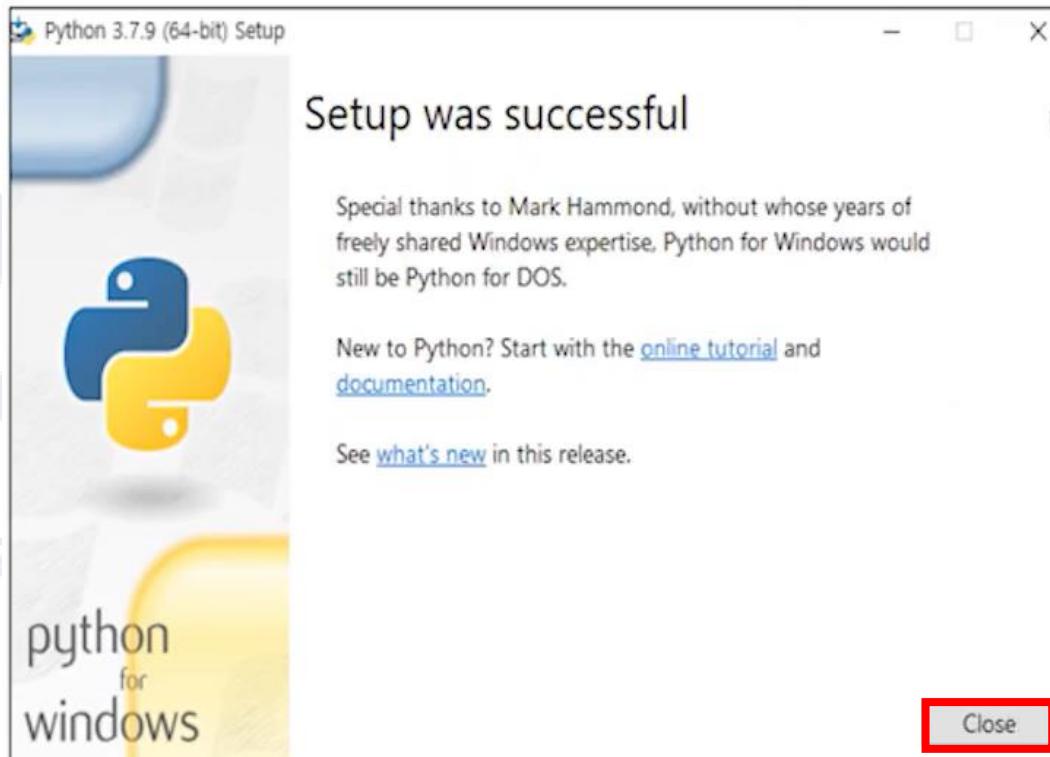


### 3. Basics of Python grammar

#### 2) 개발 환경 설정

- Python 설치 [3/3]

- 아래 그림과 같은 화면이 나오면 Python 설치 완료

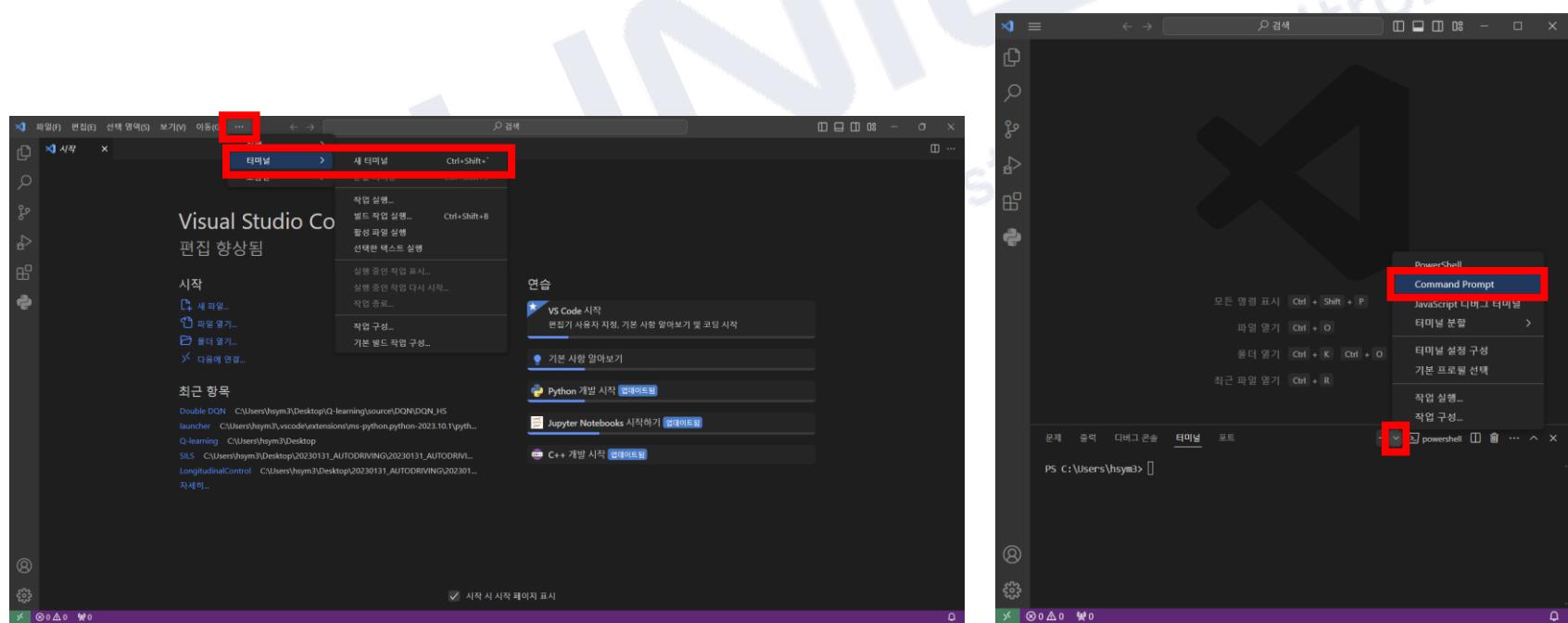


# 3. Basics of Python grammar

## 2) 개발 환경 설정

### ▪ Visual Studio Code 설정 [1/2]

- Python 확인 및 패키지 설치를 위하여 VS Code에서 Terminal을 실행 (단축키 : Ctrl + Shift + `)
- 생성된 Terminal에서 토글을 클릭하여 명령 프롬프트(Command Prompt)를 선택

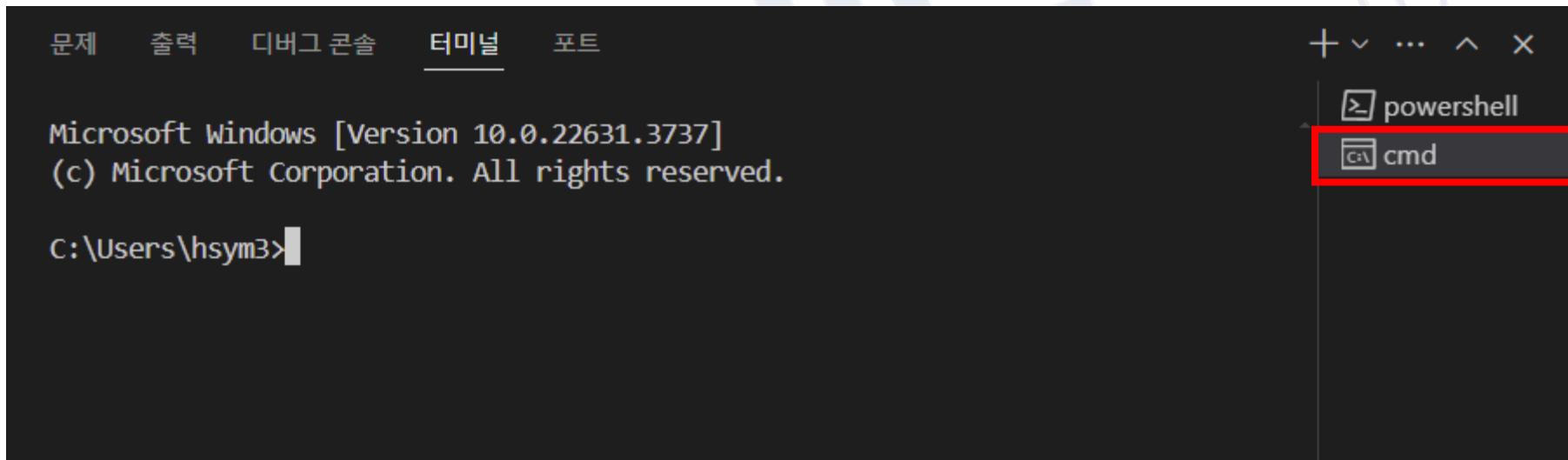


### 3. Basics of Python grammar

#### 2) 개발 환경 설정

- Visual Studio Code 설정 [2/2]

- Window에서 사용하는 cmd(command)창을 동일하게 사용 가능



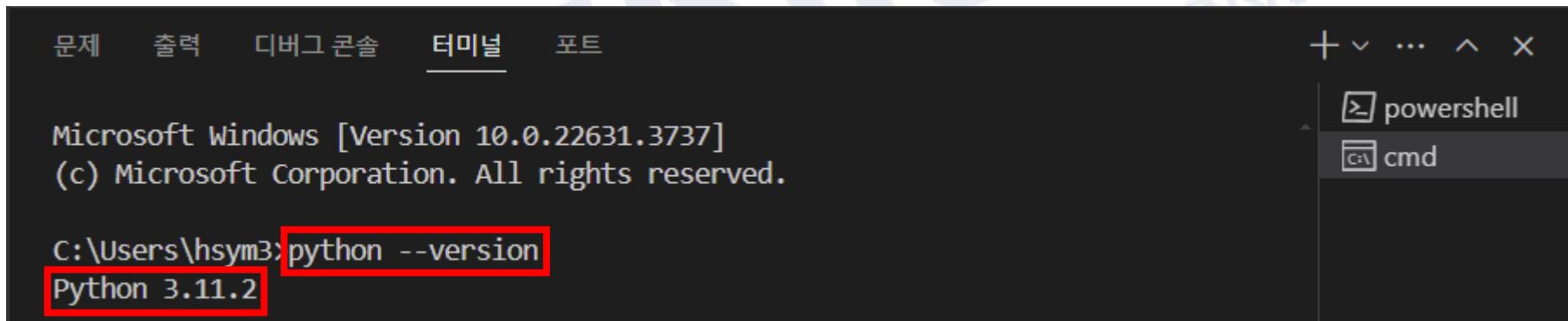
### 3. Basics of Python grammar

#### 2) 개발 환경 설정

- Python 설치 확인

- VS Code의 명령 프롬프트에서 아래 command를 입력하여 Python 설치를 확인

```
$ python --version
```



A screenshot of the VS Code interface showing a terminal window. The terminal tab is selected at the top. The terminal content shows the output of a command-line session on Microsoft Windows. The session starts with system information, followed by the command \$ python --version, and ends with the Python version 3.11.2. The command and its output are highlighted with a red box.

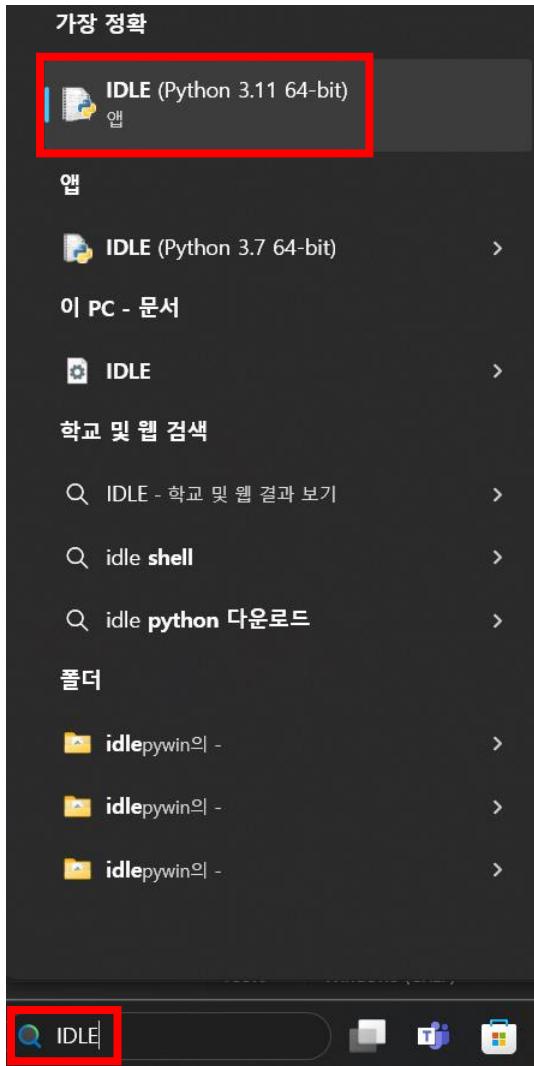
```
문제    출력    디버그 콘솔    터미널    포트 + v ... ^ x
Microsoft Windows [Version 10.0.22631.3737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hsym3>python --version
Python 3.11.2
```

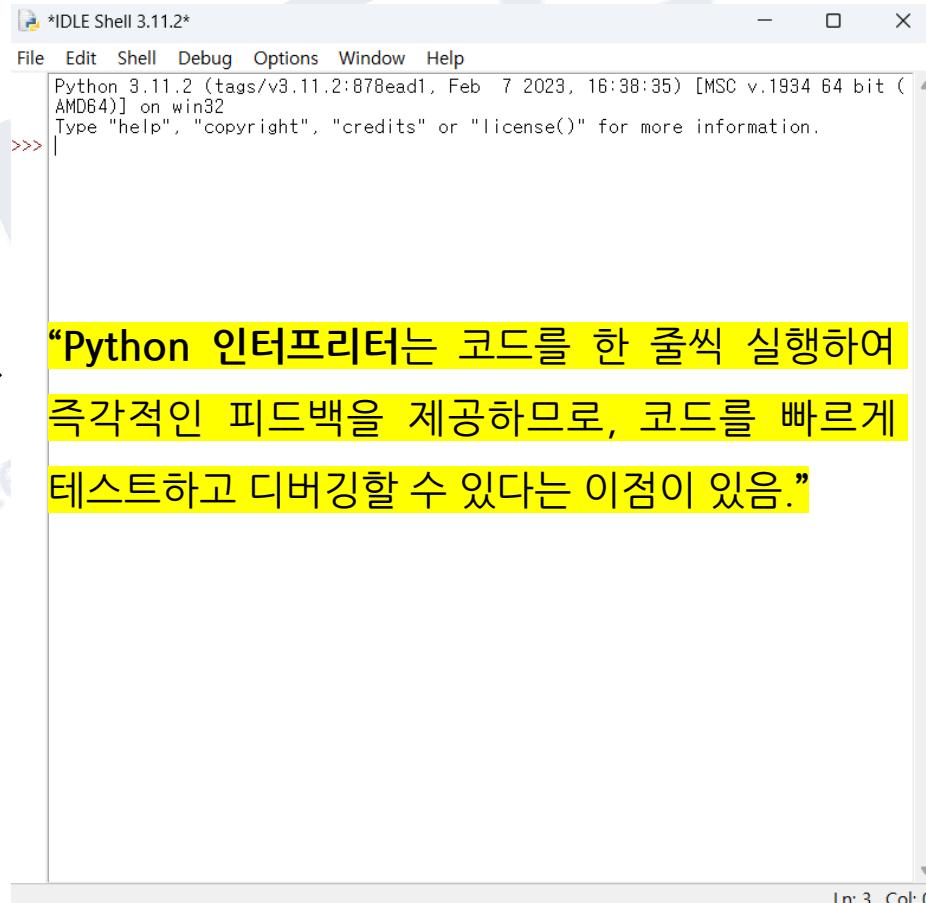
- Python 3.7.9가 확인되면 올바르게 설치된 것

### 3. Basics of Python grammar

#### 3) 기초 문법 – Python 인터프리터 사용법 (1)



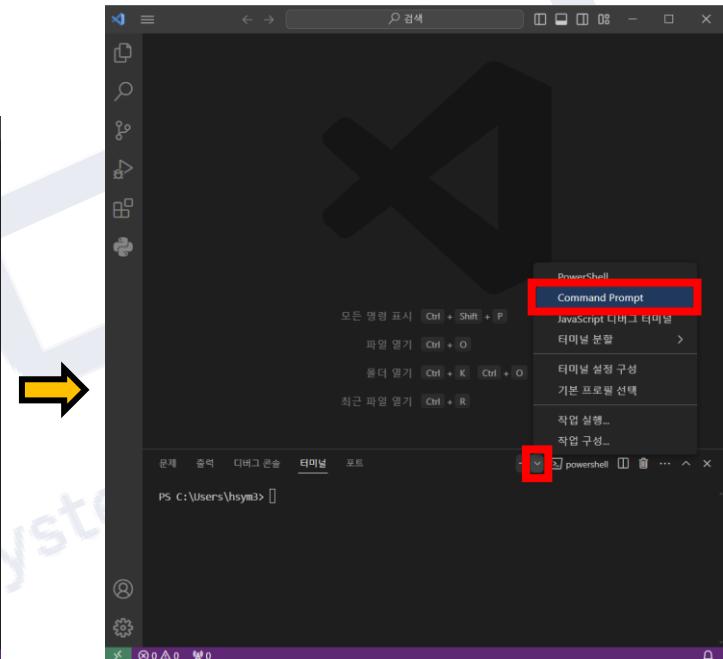
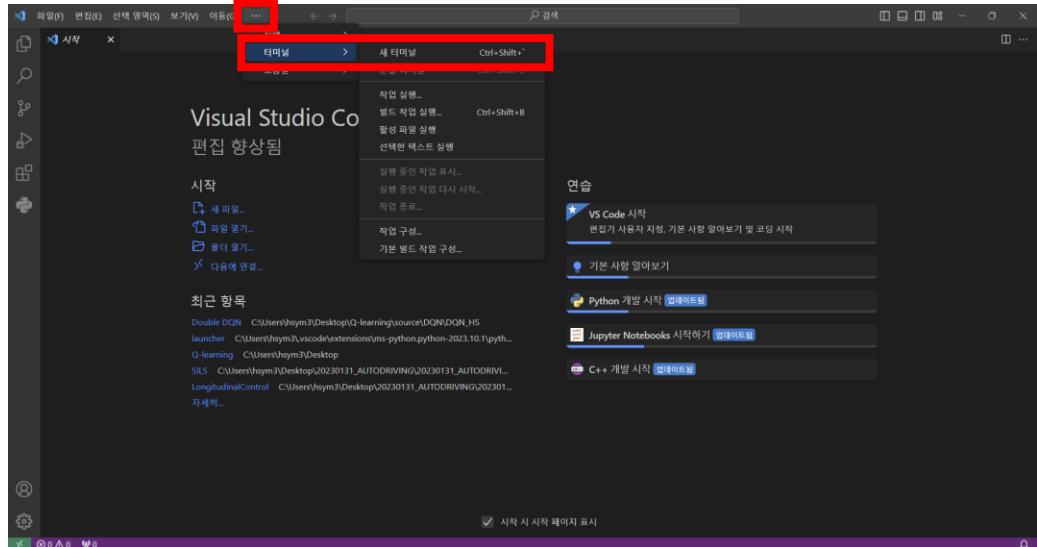
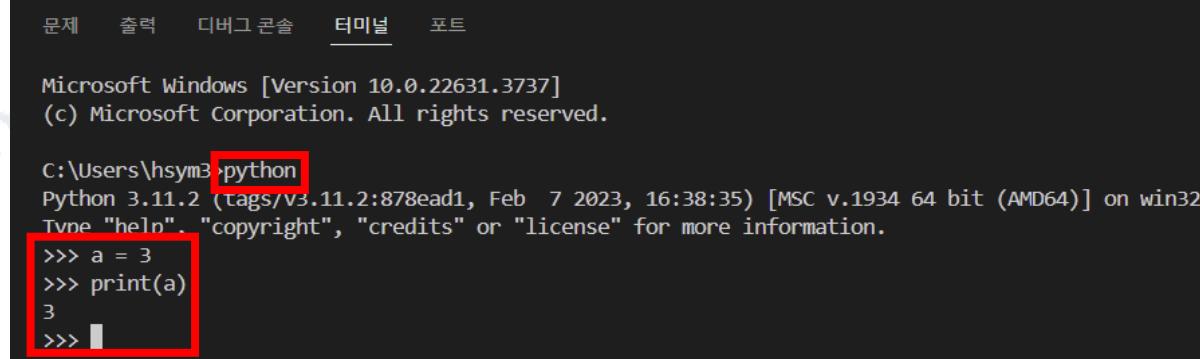
- 윈도우 왼쪽 아래 검색 명령창에 “IDLE”을 검색한 뒤,  
왼쪽 그림과 같이 검색된 Python 앱을 클릭하여 실행.



The screenshot shows the \*IDLE Shell 3.11.2\* window. The title bar says "\*IDLE Shell 3.11.2\*". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays Python version information: "Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32". It also shows the prompt "Type >>>" followed by a blank command line. A large yellow arrow points from the left side of the slide towards this window. To the right of the window, a yellow box contains the following text:  
**"Python 인터프리터는 코드를 한 줄씩 실행하여  
즉각적인 피드백을 제공하므로, 코드를 빠르게  
테스트하고 디버깅할 수 있다는 이점이 있음."**

# 3. Basics of Python grammar

## 3) 기초 문법 – Python 인터프리터 사용법 (2)

A screenshot of a terminal window titled 'Microsoft Windows [Version 10.0.22631.3737]'. The window title bar includes tabs for '문제', '출력', '디버그 콘솔', '터미널', and '포트'. The terminal content shows:

```

Microsoft Windows [Version 10.0.22631.3737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hsym3>python
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.

>>> a = 3
>>> print(a)
3
>>>

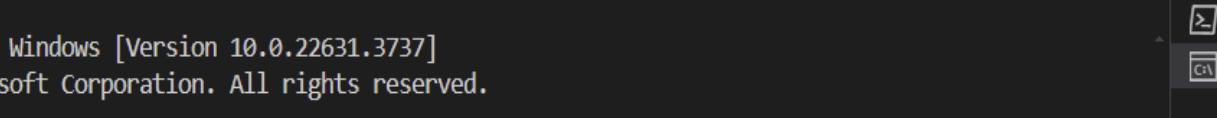
```

The bottom portion of the terminal window is highlighted with a red box.

# 3. Basics of Python grammar

## 3) 기초 문법 - 변수

- 변수: 변수는 데이터를 저장하는 컨테이너로서, 프로그램 내에서 데이터를 다루기 위해 사용됨.  
C 또는 C++ 언어와는 달리 Python에서는 변수를 선언할 때 특별한 명령어나 자료형을 지정할 필요가 없으며, 변수에 값을 할당하는 순간 해당 변수가 생성됨.



```
Microsoft Windows [Version 10.0.22631.3737]
(c) Microsoft Corporation. All rights reserved.

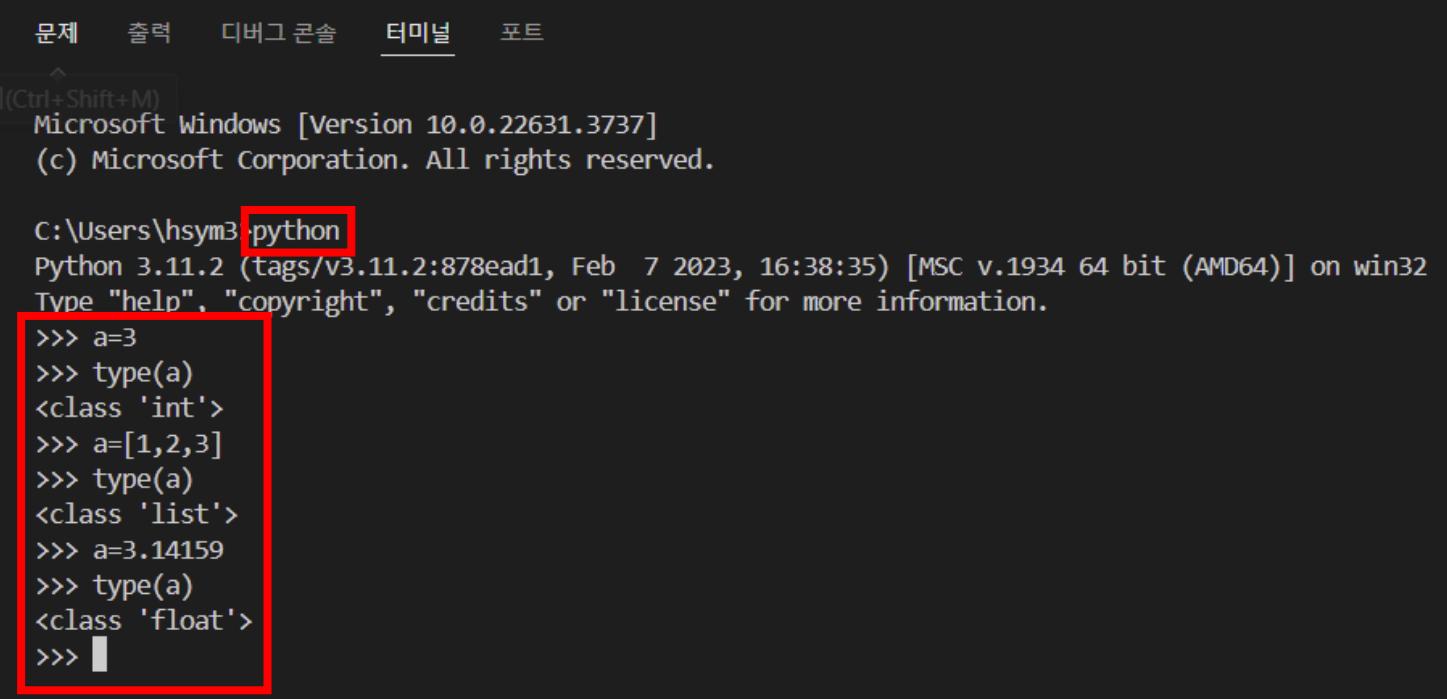
C:\Users\hsym3>python
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.

>>> x=5
>>> x
5
>>>
```

### 3. Basics of Python grammar

#### 3) 기초 문법 - 자료형

- 자료형(타입, Type) : 프로그래밍 언어에서 변수 또는 값이 가질 수 있는 데이터의 종류를 의미
- 자료형 → 정수형(Integer), 부동 소수점 수(Float), 문자열(String), 불리언(Boolean), 리스트(List), 튜플(Tuple), 딕셔너리(Dictionary) 등의 다양한 자료형이 존재



```
문제    출력    디버그 콘솔    터미널    포트
(Ctrl+Shift+M)
Microsoft Windows [Version 10.0.22631.3737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hsym\python
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.

>>> a=3
>>> type(a)
<class 'int'>
>>> a=[1,2,3]
>>> type(a)
<class 'list'>
>>> a=3.14159
>>> type(a)
<class 'float'>
>>> 
```

### 3. Basics of Python grammar

#### 3) 기초 문법 - 변수와 자료형

- 정수형(Integer)
  - ex) `x = 10` → 정수 10을 변수 x에 할당.
- 부동 소수점 수(Float)
  - ex) `y = 3.14` → 실수 3.14를 변수 y에 할당.
- 문자열(String)
  - ex) `name = "Alice"` → 작은 따옴표(' ') 또는 큰 따옴표(" ")를 통해 문자열 구성.
- 불리언(Boolean)
  - 조건문에서 주로 사용되며, 'True' 또는 'False'만을 가지는 자료형.  
→ 리스트(List), 튜플(Tuple), 딕셔너리(Dictionary)은 뒤에서 설명 예정.

### 3. Basics of Python grammar

#### 3) 기초 문법 - 변수와 자료형 예시

- 정수형(Integer) 예시

```
>>> a=3
>>> type(a)
<class 'int'>
>>> 
```

- 부동 소수점 수(Float) 예시

```
>>> a=3.14159
>>> type(a)
<class 'float'>
>>> 
```

- 문자열(String) 예시

```
>>> a='my name is "Hwasu Lee"'
>>> type(a)
<class 'str'>
>>> 
```

- 불리언(Boolean) 예시

```
>>> a="Hwasu"
>>> if a=="Hwasu": print("True")
...
True
>>> 
```

# 3. Basics of Python grammar

## 3) 기초 문법 – 기본 연산자

### 1. 산술 연산자

- `+` : 덧셈
- `-` : 뺄셈
- `\*` : 곱셈
- `/` : 나눗셈 (결과는 항상 부동 소수점 수)
- `//` : 정수 나눗셈 (결과는 정수, 소수점 이하 버림)
- `%` : 나머지
- `\*\*` : 거듭제곱

### 2. 비교 연산자

- `==` : 같다
- `!=` : 같지 않다
- `>` : 크다
- `<` : 작다
- `>=` : 크거나 같다
- `<=` : 작거나 같다

### 3. 논리 연산자

- `and` : 논리적 AND
- `or` : 논리적 OR
- `not` : 논리적 NOT

### 4. 할당 연산자

- `=` : 값을 변수에 할당
- `+=` : 변수에 값을 더한 후 결과를 다시 변수에 할당
- `-=` : 변수에서 값을 뺀 후 결과를 다시 변수에 할당
- `\*=` : 변수에 값을 곱한 후 결과를 다시 변수에 할당
- `/=` : 변수를 값으로 나눈 후 결과를 다시 변수에 할당
- `%=` : 변수를 값으로 나눈 나머지를 다시 변수에 할당
- `//=` : 변수를 값으로 나눈 몫을 다시 변수에 할당
- `\*\*=` : 변수를 값으로 거듭제곱한 결과를 다시 변수에 할당

# 3. Basics of Python grammar

## 3) 기초 문법 - 주석

## ■ 주석

- 주석은 코드 내에서 해설이나 설명을 제공하는 부분으로, 코드 실행 시 무시됨.
  - 주석은 '#' 기호 뒤에 작성되며, 코드의 가독성을 높이고 다른 개발자들이 코드를 이해하는 데 도움을 줌.
  - 주석을 여러 줄 작성할 경우, 보통 문자열("“ 주석 ”” or “‘ 주석 ’”)을 사용하여 작성되지만 일반적으로는 단일 줄 주석(#이 주로 사용됨.

```
19 class AStarPlanner:
20
21     def planning(self, sx, sy, gx, gy):
22
23         # show graph
24
25         if show_animation: # pragma: no cover
26             plt.plot(self.calc_grid_position(current.x, self.min_x),
27                      self.calc_grid_position(current.y, self.min_y),
28                      'ro')
29
30             # for stopping simulation with the esc key.
31             plt.gcf().canvas.mpl_connect('key_release_event',
32                                         lambda event: [exit(
33                                             0) if event.key == 'escape'
34                                         else None])
35
36             if len(closed_set.keys()) % 10 == 0:
37                 plt.pause(0.001)
38
39
40             if current.x == goal_node.x and current.y == goal_node.y:
41                 print("Find goal")
42                 goal_node.parent_index = current.parent_index
43                 goal_node.cost = current.cost
44                 break
45
46
47             # Remove the item from the open set
48             del open_set[c_id]
```

```
C:\Users\hsym3\Desktop\SLAM & Navigation source code\PythonRobotics-master\PythonRobotics-  
19 class AStarPlanner:  
20  
21     def planning(self, sx, sy, gx, gy):  
22         """  
23             A star path search  
24  
25             input:  
26                 s_x: start x position [m]  
27                 s_y: start y position [m]  
28                 gx: goal x position [m]  
29                 gy: goal y position [m]  
30  
31             output:  
32                 rx: x position list of the final path  
33                 ry: y position list of the final path  
34         """  
35  
36         start_node = self.Node(self.calc_xy_index(sx, self.min_x),  
37                               self.calc_xy_index(sy, self.min_y), 0.0, -1)  
38         goal_node = self.Node(self.calc_xy_index(gx, self.min_x),  
39                               self.calc_xy_index(gy, self.min_y), 0.0, -1)
```

### 3. Basics of Python grammar

#### 3) 기초 문법 – 입·출력 함수

- `input()`

- `input()` 함수는 사용자로부터 입력을 받는 데 사용.
- 입력 받은 데이터는 항상 문자열 형태로 반환되며, 다른 데이터 타입이 필요한 경우 변환 과정 필요.

```
name = input("Enter your name: ")
print("Hello, " + name + "!")
```

- `name`이라는 변수에 사용자로부터 문자열 타입의 데이터를 직접 입력 받아 출력하는 구조.

```
age = input("Enter your age: ")
age = int(age) # 문자열을 정수로 변환
print("You are", age, "years old.")
```

- `age`라는 변수에 사용자로부터 문자열 타입의 데이터를 직접 입력 받은 뒤,  
`int()` 타입의 데이터 형태로 변환하여 출력하는 구조.

### 3. Basics of Python grammar

#### 3) 기초 문법 – 입·출력 함수

- `print()`

- `print()` 함수는 화면에 출력을 표시하는 데 사용되는 함수.
- 기본적인 사용법은 매우 간단하나, 다양한 옵션을 통해 출력 형식을 보다 세밀하게 제어 가능.

```
print("Hello, World!")
```

- 가장 기본적인 형태의 `print()` 구조.
- 출력으로 “Hello, World!” 문자열을 화면에 출력.

```
print("Hello", "World", 123)
# 출력: Hello World 123
```

- `print()` 함수는 여러 인수를 쉼표로 구분하여 출력이 가능하며, 각 인수는 공백으로 구분되어 출력됨.
- 출력으로 “Hello World 123” 문자열을 화면에 출력.

```
print("Hello", "World", sep=", ")
# 출력: Hello, World
```

- `print()` 함수는 ‘`sep`’이라는 키워드 인수를 통해 여러 인수 사이에 출력할 문자열을 지정 가능.
- 출력으로 “Hello, World” 문자열을 화면에 출력.

### 3. Basics of Python grammar

#### 4) 제어 구조 – 반복문

- **반복문** : 프로그래밍에서 특정 코드 블록을 조건에 따라 여러 번 실행할 수 있게 하는 구조.  
  
반복문을 통해 같은 작업을 반복해서 수행하는 코드를 간결하게 작성할 수 있으며,  
데이터 처리, 자동화 작업, 반복적인 태스크 처리 등 다양한 곳에서 활용 가능.
- 대표적으로는 ‘for’문을 통한 반복문과, ‘while’문을 통한 반복문이 있음.
- ‘for’ 반복문 : ‘for’ 반복문은 주어진 시퀀스(리스트, 튜플, 딕셔너리, 문자열 등)의 각 요소에  
대해 코드 블록을 반복 실행하거나, 정해진 횟수만큼 반복하는 경우 사용.
- ‘while’ 반복문 : ‘while’ 반복문은 주어진 조건이 참(True)인 동안 반복해서 코드를 실행.

### 3. Basics of Python grammar

#### 4) 제어 구조 - 반복문

- ‘for’ 반복문 : ‘for’ 반복문은 주어진 시퀀스(리스트, 튜플, 딕셔너리, 문자열 등)의 각 요소에 대해 코드 블록을 반복 실행하거나, 정해진 횟수만큼 반복하는 경우 사용.

```
for 변수 in 시퀀스:
    실행할 코드
    ↪ 들여쓰기 ('Tab 키' or 'Spacebar 키 4번')
```

- 예제)
 

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

```
# 0부터 4까지 출력
for i in range(5):
    print(i)
```

`range(n) → 0 부터 n-1까지 반복`

`range(1, n+1) → 1부터 n까지 반복`

### 3. Basics of Python grammar

#### 4) 제어 구조 – 반복문

- ‘while’ 반복문 : ‘while’ 반복문은 주어진 조건이 참(True)인 동안 반복해서 코드를 실행.

while 조건:

실행할 코드

- 예제)

```
x = 5
while x > 0:
    print(x)
    x -= 1 # x 값을 1씩 감소
```

while True:

```
user_input = input("종료하려면 'exit'을 입력하세요: ")
```

```
if user_input.lower() == 'exit':
```

```
    print("프로그램을 종료합니다.")
```

```
    break
```

```
else:
```

```
    print(f"입력한 메시지: {user_input}")
```

### 3. Basics of Python grammar

#### 4) 제어 구조 – 조건문

- 조건문 : 프로그래밍에서 코드의 실행 흐름을 제어하기 위한 구조.

조건문을 통해 특정 조건이 참(True)일 때와 거짓(False)일 때, 다른 코드를 실행 가능.

조건문은 프로그램의 논리적 흐름을 제어하는 데 필수적.

- 대표적으로 ‘if’, ‘elif’, ‘else’문을 통해 조건문을 구성하며, 하나의 조건문을 형성할 때 ‘if’, ‘elif’, ‘else’문을 모두 사용할 필요는 없음.

이러한 조건문은 크게 ‘조건식’과 ‘블록’으로 구성됨

- 조건식 : 조건식은 참(True) 또는 거짓(False)으로 평가될 수 있는 표현식을 의미.

비교 연산자(‘==’, ‘!=’, ‘>’, ‘<’, ‘>=’, ‘<=’)나 논리 연산자(‘and’, ‘or’, ‘not’)를 사용하여 작성.

- 블록 : 조건이 참(True)일 때 실행될 코드를 의미하며, 들여쓰기를 사용하여 블록을 구분.

# 3. Basics of Python grammar

## 4) 제어 구조 – 조건문

- 단일 if문

**if** 조건식:  
    실행할 코드

```
x = 10
if x > 5:
    print("x is greater than 5")
```

- if-elif-else문

**if** 조건식1:  
    실행할 코드 (조건식1이 참일 때)
   
**elif** 조건식2:  
    실행할 코드 (조건식2가 참일 때)
   
**else:**  
    실행할 코드 (모든 조건이 거짓일 때)

```
x = 7
if x > 10:
    print("x is greater than 10")
elif x > 5:
    print("x is greater than 5 but less than or equal to 10")
else:
    print("x is 5 or less")
```

- if-else문

**if** 조건식:  
    실행할 코드 (조건이 참일 때)
   
**else:**  
    실행할 코드 (조건이 거짓일 때)

```
x = 3
if x > 5:
    print("x is greater than 5")
else:
    print("x is 5 or less")
```

### 3. Basics of Python grammar

#### 실습

- 반복문(for 반복문 / while 반복문) 및 조건문(if / elif / else) 실습

→ 첫 번째 실습 : “for 반복문과 input 함수를 통해 1부터 입력 받은 정수까지 모든 수의 합을 구하는 코드 작성”

(이 때, 입력 받은 수가 자연수가 아니라면, “Wrong Number”를 출력하도록 코드 구성)

→ 두 번째 실습 : “while 반복문과 input 함수를 통해 1부터 입력 받은 정수까지 모든 수의 곱을 구하는 코드 작성”

(이 때, 입력 받은 수가 자연수가 아니라면, “Wrong Number”를 출력하도록 코드 구성)

→ 세 번째 실습 : “조건문과 input 함수를 통해 입력으로 받은 자연수가 홀수인지 짝수인지 구하는 코드 작성”

(이 때, 입력 받은 수가 자연수가 아니라면, “Wrong Number”를 출력하도록 코드 구성)

### 3. Basics of Python grammar

#### 실습

→ 첫 번째 실습 : “for 반복문과 input 함수를 통해 1부터 입력 받은 정수까지 모든 수의 합을 구하는 코드 작성 ”

```
n = int(input("Enter a positive integer: "))

if n < 1:
    print("Wrong Number")
else:
    total = 0
    for i in range(1, n + 1):
        total += i
    print(f"The sum of numbers from 1 to {n} is: {total}")
```

### 3. Basics of Python grammar

#### 실습

→ 두 번째 실습 : “ while 반복문과 input 함수를 통해 1부터 입력 받은 정수까지 모든 수의 곱을 구하는 코드 작성 ”

```
n = int(input("Enter a positive integer: "))

if n < 1:
    print("Wrong Number")
else:
    product = 1
    i = 1
    while i <= n:
        product *= i
        i += 1
    print(f"The product of numbers from 1 to {n} is: {product}")
```

### 3. Basics of Python grammar

#### 실습

→ 세 번째 실습 : “ 조건문과 input 함수를 통해 입력으로 받은 자연수가 홀수인지 짝수인지 구하는 코드 작성 ”

```
n = int(input("Enter a natural number: "))

if n <= 0:
    print("Wrong Number")
elif n % 2 == 0:
    print(f"{n} is an even number.")
else:
    print(f"{n} is an odd number.")
```

### 3. Basics of Python grammar

#### 5) 자료 구조 (리스트 / 튜플 / 딕셔너리)

- 리스트(List)

- 리스트는 순서가 있는 시퀀스로, 다양한 데이터 타입의 요소를 포함할 수 있음
- 요소(원소)의 추가 append, 삭제 pop 또는 remove, 수정이 자유로움
- 대괄호 '[]' 또는 'list()'를 통해 생성하며, 인덱싱과 슬라이싱으로 요소에 접근 가능
- 인덱싱은 특정 리스트(ex. arr = [1, 2, 3])를 구성하는 원소들의 인덱스를 통해 접근하는 방식

**이 때, 리스트의 첫 번째 원소가 가지는 인덱스는 1이 아닌 “ 0 ”임을 유의!!**

ex) arr[0] = 1, arr[1] = 2, arr[2] = 3

- 슬라이싱은 인덱싱과 유사하게 리스트의 인덱스를 통해 접근하는 방식.

ex) new\_arr = arr[0:2] → print(new\_arr) → [1, 2]

ex) new\_arr = arr[:3] → print(new\_arr) → [1, 2, 3]

### 3. Basics of Python grammar

#### 5) 자료 구조 (리스트 / 튜플 / 딕셔너리)

##### 리스트(List) 주요 내장 함수

- **append(x)** : 리스트의 끝에 항목 ‘x’를 추가.
- **insert(i, x)** : 리스트의 i번째 인덱스에 항목 ‘x’를 추가.
- **pop(i)** : 리스트의 i번째 인덱스에 항목을 제거하고, 그 값을 반환.  
  
만약 i가 지정되지 않으면, 리스트의 마지막 항목을 제거하고 반환.
- **remove(x)** : 리스트에서 값 ‘x’를 찾아 제거.
- **index(x)** : 리스트에서 값 ‘x’가 위치한 인덱스를 반환.
- **sort(key=None, reverse=False)** : 리스트를 오름차순으로 정렬.
- **reverse()** : 리스트의 항목 순서를 거꾸로 뒤집음.
- **clear()** : 리스트의 모든 항목을 제거.

### 3. Basics of Python grammar

#### 5) 자료 구조 (리스트 / 튜플 / 딕셔너리)

##### 리스트(List) 예제 코드

```
>>> arr = []
>>> type(arr)
<class 'list'>
>>> arr.append('Hwasu')
>>> arr.append('Munkyu')
>>> print(arr)
['Hwasu', 'Munkyu']
>>> 
```

```
>>> print(arr[0])
Hwasu
>>> print(arr[1])
Munkyu
>>> arr[0] = 'Chang Mook'
>>> print(arr)
['Chang Mook', 'Munkyu']
>>> name = arr.pop()
>>> print(name)
Munkyu
>>> print(arr)
['Chang Mook']
>>> 
```

arr = ['Hwasu', 'Munkyu']  
 0번 인덱스 1번 인덱스  


Python에서 인덱스(index)는  
0번부터 시작!

arr = [1, 2, 3, 4, 5, 6, 7]  
 arr[:7] = ??

슬라이싱을 통해 arr 리스트의  
0번 인덱스 값부터 6번 인덱스  
값까지 반환!

ex) [1, 2, 3, 4, 5, 6, 7]

```
>>> arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> print(arr)
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> arr2 = arr[:6]
>>> print(arr2)
[1, 2, 3, 4, 5, 6]
>>> arr3 = arr[2:8]
>>> print(arr3)
[3, 4, 5, 6, 7, 8]
>>> 
```

### 3. Basics of Python grammar

#### 5) 자료 구조 (리스트 / 튜플 / 딕셔너리)

- 리스트 컴프리헨션(List Comprehension)
  - 리스트 컴프리헨션은 파이썬에서 리스트를 생성하는 간결하고 효율적인 방법.
  - 전통적인 for문을 통한 반복문을 사용하는 것보다 더 짧고 가독성 좋은 코드를 작성 가능.
  - 리스트 컴프리헨션을 사용하면 기존의 리스트나 다른 반복 가능한(iterable) 객체를 기반으로 새로운 리스트를 만들 수 있음.

```
[expression for item in iterable if condition]
```

- ‘expression’은 각 항목에 대해 실행할 표현식을 의미.
- ‘item’은 반복 과정에서 사용되는 변수를 의미.
- ‘iterable’은 반복 가능한 객체(예: 리스트, 문자열, range 등)를 의미.
- ‘condition’은 선택 사항으로, 각 항목에 대해 참(True)인 경우에만 expression을 적용.

### 3. Basics of Python grammar

#### 5) 자료 구조 (리스트 / 튜플 / 딕셔너리)

- 리스트 컴프리헨션(List Comprehension) 예제

```
numbers = [1, 2, 3, 4, 5]
doubled = [x * 2 for x in numbers]
print(doubled) # 출력: [2, 4, 6, 8, 10]
```

- 기존 리스트의 각 항목에 2를 곱하여 새로운 리스트 생성

- 리스트 컴프리헨션을 사용하면
  - 코드가 간결하고 가독성 있게
  - 구성되며, 보다 빠른 속도로
  - 실행된다는 이점이 있음.

```
numbers = [1, 2, 3, 4, 5, 6]
even_numbers = [x for x in numbers if x % 2 == 0]
print(even_numbers) # 출력: [2, 4, 6]
```

- 조건문을 사용하여 짝수만 필터링

```
numbers = [1, 2, 3, 4, 5]
result = [x * 2 if x % 2 == 0 else x * 3 for x in numbers]
print(result) # 출력: [3, 4, 9, 8, 15]
```

- 조건식에 if-else를 사용하여 값을 변환 (값이 짝수인 경우 2배, 아닌 경우 3배)

### 3. Basics of Python grammar

#### 5) 자료 구조 (리스트 / 튜플 / 딕셔너리)

- **튜플(Tuple)**

- 튜플은 순서가 있는 시퀀스이지만, 한 번 생성된 후에는 수정 불가 (불변 자료형)
- 소괄호 '()' 또는 'tuple()'을 통해 생성하거나, 괄호 없이 요소를 나열하여 생성 가능
- 주로 데이터가 변경될 위험이 없어야 하는 상황에서 유용하며,  
함수에서 여러 값을 한 번에 반환할 때 주로 사용됨

#### 튜플(Tuple) 예제 코드

```
dimensions = (1920, 1080)
print(dimensions[0]) # 1920 출력
# dimensions[0] = 2560 # 에러 발생, 튜플은 수정할 수 없음
```



```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> █
```

### 3. Basics of Python grammar

#### 5) 자료 구조 (리스트 / 튜플 / 딕셔너리)

- 딕셔너리(Dictionary)

- 딕셔너리는 ‘key(키):value(값)’ 쌍으로 데이터를 저장하는 자료 구조
- 중괄호 ‘{}’ 또는 ‘dict()’를 통해 생성하며, ‘key’를 통해 빠르게 데이터에 접근 가능
- 딕셔너리의 각 ‘key’ 값은 고유해야 하며, 중복 불가
- 리스트(List)와 동일하게 ‘key’와 ‘value’ 값을 추가, 삭제, 수정할 수 있음

#### 딕셔너리(Dictionary) 예제 코드

```
student_grades = {'Alice': 90, 'Bob': 85, 'Eve': 88}  
student_grades['Alice'] = 95 # Alice의 점수를 수정  
print(student_grades['Alice']) # 95 출력
```

key    value



### 3. Basics of Python grammar

#### 실습

- 자료구조(리스트 / 튜플 / 딕셔너리) 실습

→ 첫 번째 실습 : “리스트 내의 정수 중 가장 큰 수를 출력하는 코드 작성”

(단, 파이썬 내장 함수인 `max()` 함수 사용 불가)

→ 두 번째 실습 : “리스트 컴프리헨션과 `input()` 함수를 이용하여 입력 받은 수까지의 자연수 중,

3의 배수 또는 5의 배수인 숫자의 제곱을 리스트로 생성하는 코드 작성”

→ 세 번째 실습 : “크기가 서로 다른 자연수로 구성된 리스트를 오름차순으로 정렬하는 코드 작성”

(단, 파이썬 내장 함수인 `sort()` 함수 사용 불가)

### 3. Basics of Python grammar

#### 실습

- 첫 번째 실습 : “리스트 내의 정수 중 가장 큰 수를 출력하는 코드 작성”
- 주의사항 : Python 내장 함수인 max() 함수를 사용하지 않고 코드를 작성

```
# 예제 리스트
example_list = [23, 1, 45, 34, 75, 54, 24]

if not example_list: # 리스트가 비어있는 경우 처리
    maximum_value = None
else:
    maximum_value = example_list[0] # 리스트의 첫 번째 요소를 최대값으로 초기 설정
    for number in example_list:
        if number > maximum_value:
            maximum_value = number

print("가장 큰 수는:", maximum_value)
```

### 3. Basics of Python grammar

#### 실습

→ 두 번째 실습 : “리스트 컴프리헨션과 input() 함수를 이용하여 입력 받은 수까지의 자연수 중,  
3의 배수 또는 5의 배수인 숫자의 제곱을 리스트로 생성하는 코드 작성 ”

```
# 사용자가 입력한 숫자까지의 범위에서 3의 배수 또는 5의 배수인 숫자의 제곱을 리스트로 생성
n = int(input("Enter a positive integer: "))

multiples_of_3_or_5_squares = [x**2 for x in range(1, n + 1) if x % 3 == 0 or x % 5 == 0]

print(multiples_of_3_or_5_squares)
```

### 3. Basics of Python grammar

#### 실습

→ 세 번째 실습 : “ 크기가 서로 다른 자연수로 구성된 리스트를 오름차순으로 정렬하는 코드 작성 ”

→ 주의사항 : 파이썬 내장 함수인 `sort()` 함수를 사용하지 않고 코드를 작성

```
# 정렬할 리스트
numbers = [64, 25, 12, 22, 11]

# 선택 정렬 알고리즘
n = len(numbers)
for i in range(n):
    # 현재 위치에서 최소값의 인덱스를 찾음
    min_idx = i
    for j in range(i+1, n):
        if numbers[j] < numbers[min_idx]:
            min_idx = j
    # 현재 위치와 최소값의 위치를 교환
    numbers[i], numbers[min_idx] = numbers[min_idx], numbers[i]

print("Sorted list:", numbers)
```

# 3. Basics of Python grammar

## 6) 함수

- **함수(Function)**

- 코드를 조직화하고 재사용하기 위해 사용
- 복잡한 프로그램을 관리하기 쉽도록 부분으로 나눌 수 있으며,

코드의 중복을 줄이고 프로그램의 가독성을 향상시킬 수 있음

- ‘def’ 키워드를 통해 정의하며, return이 없는 경우 함수는 ‘None’ 값을 반환
- 함수의 입력이 되는 매개변수는 반드시 있어야 하는 것은 아님

```
def 함수명(매개변수):
    # 수행할 코드
    return 반환값
```

### 함수 예제 코드

```
def greet(name):
    return f"Hello, {name}!"

message = greet("Alice")
print(message) # Hello, Alice!
```

```
def describe_pet(animal_type, pet_name):
    print(f"I have a {animal_type} and its name is {pet_name}.") 

describe_pet("hamster", "Harry")
describe_pet(pet_name="Daisy", animal_type="dog")
```

### 3. Basics of Python grammar

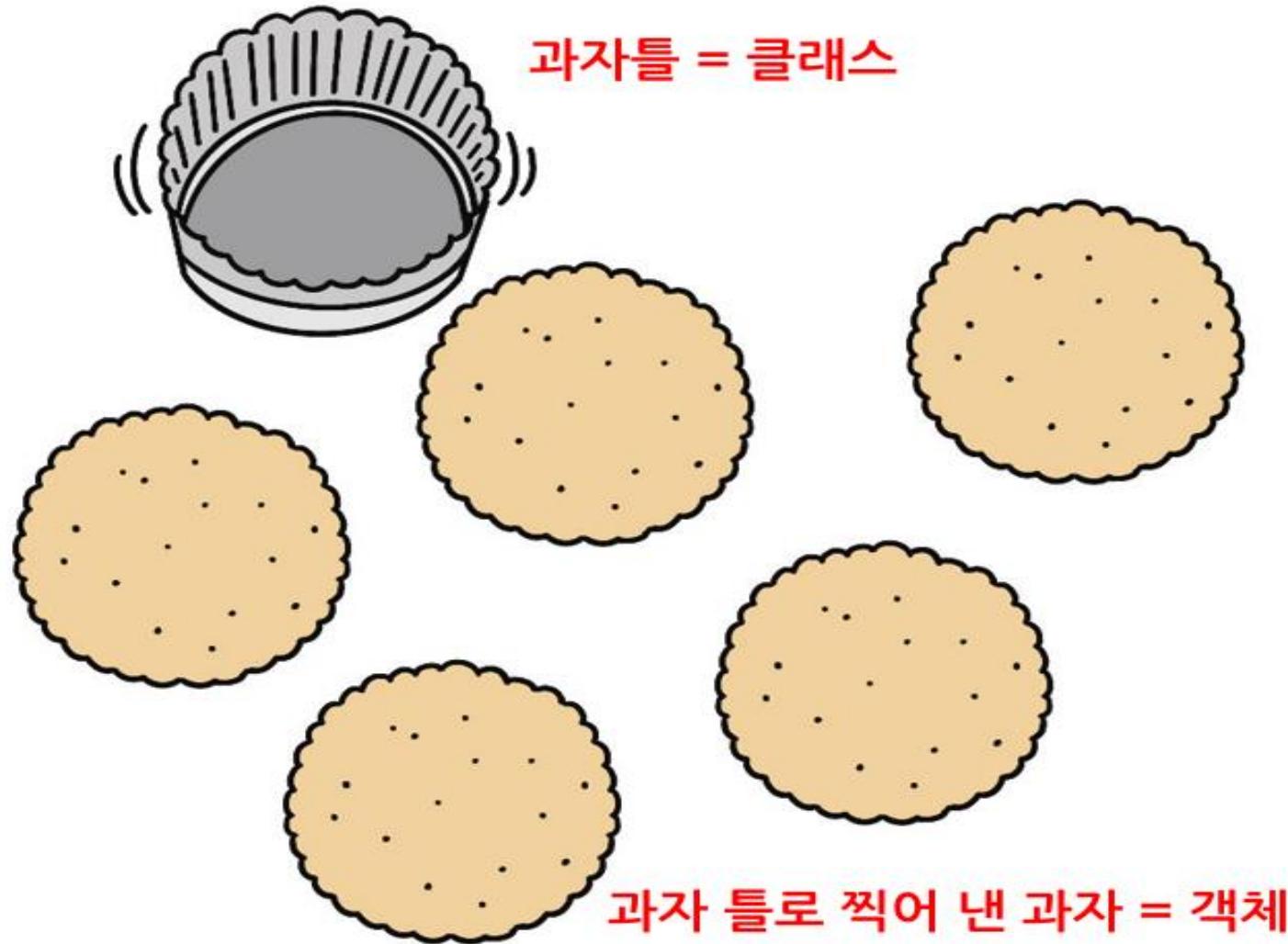
#### 7) 클래스

- **클래스(Class)**

- 클래스는 데이터와 함수를 하나로 묶어, 새로운 타입을 정의
- 데이터 구조를 만들고, 해당 구조에 관련된 특정 기능들을 조직화 및 모듈화 가능
- 클래스는 객체를 생성하는 “틀” 또는 “모델”과 같은 역할을 수행
- 클래스의 주요 구성 요소는 ‘속성(Attribute)’과 ‘메서드(Method)’로 구성됨
  - 속성(Attribute) : 클래스에 속한 데이터를 의미하며, 변수로 표현됨
  - 메서드(Method) : 클래스의 객체가 수행할 수 있는 함수를 의미

### 3. Basics of Python grammar

#### 7) 클래스



### 3. Basics of Python grammar

#### 7) 클래스 예시

```
class Car:  
    def __init__(self, make, model, year):  
        self.make = make  
        self.model = model  
        self.year = year } 인스턴스 속성  
  
    def display_info(self):  
        print(f"Car: {self.year} {self.make} {self.model}") } 메서드  
  
# 객체 생성 및 사용  
my_car = Car("Toyota", "Corolla", 2021)  
my_car.display_info()
```

### 3. Basics of Python grammar

#### 7) 클래스

- **클래스 - 상속(inheritance)**

- 상속은 기존 클래스의 (인스턴스) 속성과 메서드를 물려받아 새로운 클래스를 생성하는 것을 의미
- 이를 통해 코드의 재사용성을 높이고, 코드의 관리를 용이하게 할 수 있음

→ 상속의 주요 이점 : 코드 중복 감소, 프로그램의 구조화와 확장성 증대, 유지보수가 용이

```
class Car:  
    def __init__(self, make, model, year):  
        self.make = make  
        self.model = model  
        self.year = year  
  
    def display_info(self):  
        print(f"Car: {self.year} {self.make} {self.model}")  
  
# 객체 생성 및 사용  
my_car = Car("Toyota", "Corolla", 2021)  
my_car.display_info()
```

앞서 작성하였던 기존 클래스

# 3. Basics of Python grammar

## 7) 클래스

- 클래스 - 상속(inheritance)

기존에 작성하였던 클래스  
(부모 클래스)

부모 클래스를 상속 받는 클래스  
(자식 클래스)

```

class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

    def display_info(self):
        print(f"Car: {self.year} {self.make} {self.model}")

class ElectricCar(Car): # Car 클래스를 상속받는 ElectricCar 클래스
    def __init__(self, make, model, year, battery_size):
        super().__init__(make, model, year) # 부모 클래스의 생성자 호출
        self.battery_size = battery_size

    def display_battery(self):
        print(f"This car has a {self.battery_size}-kWh battery.")

# 상속받은 클래스의 객체 생성 및 사용
my_tesla = ElectricCar("Tesla", "Model S", 2019, 75)
my_tesla.display_info()
my_tesla.display_battery()

```

### 3. Basics of Python grammar

#### 실습

- **함수(Function) 및 클래스(Class) 실습**

→ 첫 번째 실습 : “ 리스트를 입력으로 받아, 해당 리스트의 평균값을 출력으로 도출하는 함수 만들기 ”

→ 두 번째 실습 : “ 클래스를 통해 덧셈과 뺄셈이 가능한 계산기 만들기 ”

→ 세 번째 실습 : “ 상속(inheritance)을 통해 사칙연산이 가능한 계산기 만들기 ”

(만드시 두 번째 실습에서 작성한 클래스를 상속받을 것)

### 3. Basics of Python grammar

#### 실습

- 첫 번째 실습 : “리스트를 입력으로 받아, 해당 리스트의 평균값을 출력으로 도출하는 함수 만들기”
- 주의사항 : Python 내장 함수인 sum() 함수를 사용하지 않고 코드를 작성

```
# 숫자 리스트의 평균을 계산하는 함수를 정의합니다.

def calculate_average(numbers):
    total = 0
    count = 0
    for number in numbers:
        total += number
        count += 1
    if count == 0:
        return 0
    else:
        return total / count

# 숫자 리스트를 생성합니다.
number_list = [10, 20, 30, 40, 50]

# 함수를 호출하여 평균을 계산하고 출력합니다.
average = calculate_average(number_list)
print("The average of the list is:", average)
```

### 3. Basics of Python grammar

#### 실습

→ 두 번째 실습 : “ 덧셈과 뺄셈이 가능한 계산기 만들기 ”

```
class SimpleCalculator:  
    def __init__(self):  
        self.result = 0  
  
    def add(self, a, b):  
        """두 수를 더하고 결과를 반환합니다."""  
        self.result = a + b  
        return self.result  
  
    def subtract(self, a, b):  
        """첫 번째 수에서 두 번째 수를 빼고 결과를 반환합니다."""  
        self.result = a - b  
        return self.result  
  
# 계산기 객체 생성  
calc = SimpleCalculator()  
  
# 덧셈 예시  
print("10 + 5 =", calc.add(10, 5)) # 15  
  
# 뺄셈 예시  
print("10 - 5 =", calc.subtract(10, 5)) # 5
```

# 3. Basics of Python grammar

## 실습

→ 세 번째 실습 : “상속(inheritance)을 통해 사칙연산이 가능한 계산기 만들기”

→ 주의사항 : 반드시 두 번째 실습에서 작성한 클래스를 상속받을 것

```

class SimpleCalculator:
    def __init__(self):
        self.result = 0

    def add(self, a, b):
        """두 수를 더하고 결과를 반환합니다."""
        self.result = a + b
        return self.result

    def subtract(self, a, b):
        """첫 번째 수에서 두 번째 수를 빼고 결과를 반환합니다."""
        self.result = a - b
        return self.result

```

```

# SimpleCalculator 클래스를 상속받는 ExtendedCalculator 클래스 정의
class ExtendedCalculator(SimpleCalculator):
    def multiply(self, a, b):
        """두 수를 곱하고 결과를 반환합니다."""

        self.result = a * b
        return self.result

```

```

def divide(self, a, b):
    """첫 번째 수를 두 번째 수로 나누고 결과를 반환합니다. 0으로 나누는 것을 방지합니다."""

    if b == 0:
        return "Error! Division by zero."
    else:
        self.result = a / b
        return self.result

```

```

# 확장된 계산기 객체 생성
calc = ExtendedCalculator()

# 사칙연산 예시
print("10 + 5 =", calc.add(10, 5))          # 15
print("10 - 5 =", calc.subtract(10, 5))       # 5
print("10 * 5 =", calc.multiply(10, 5))        # 50
print("10 / 5 =", calc.divide(10, 5))         # 2.0
print("10 / 0 =", calc.divide(10, 0))         # Error! Division by zero.

```

# QnA

E-mail : [leehwasu9696@inu.ac.kr](mailto:leehwasu9696@inu.ac.kr)  
[mkbae97@gmail.com](mailto:mkbae97@gmail.com)

Mobile : 010-8864-5585  
010-6218-9259



청주대학교  
미래형자동차 인력양성사업단

자율주행 분야

# 단기집중 교육과정



6월 11일(화)  
오후 4시~8시

리눅스 설치,  
ROS 설치 및 ROS 개발환경 구축

융합관  
410호

\*개인 노트북 필수 지참(대여불가)

온라인  
사전교육

Python 교육(ROS 기반)



파이썬 기초      파이썬 중급      ROS 기초강의

7월 2일~  
5일(화~금)  
오후 1시~6시

라이다 센서 교육

새천년종합정보관  
107A호

경진대회 참여학생

7월 8일~  
10일(월~수)  
오후 1시~6시

ROS 활용 교육(Python 기반)

융합관  
410호

\*개인 노트북 필수 지참(대여불가)

7월 15일~  
17일(월~수)  
오후 1시~6시

Python OpenCV 설치 및 응용

새천년종합정보관  
409호

\*개인 노트북 필수 지참(대여불가)



청주대학교  
CHEONGJU UNIVERSITY

**Thank you.**