# PROJECT TITLE

OBSTACLE DETECTION USING LIDAR DATA

## TEAM MEMBERS:

Siya Katoch Victor, Shreya U

## TEAM NAME:

Coding Divas

## DATE:

06.08.2025

## INTRODUCTION

In this project, we built an obstacle detection system for a robot using 3D LIDAR data. The LIDAR sensor returns a list of 3D coordinates (x, y, z) representing points in the robot's surroundings. Our goal was to detect and return only those points which are within a 10-meter radius from the robot's origin (0, 0, 0).

## TOOLS USED

Programming Language: Python

Libraries: math (for distance calculation)

## PROJECT STRUCTURE

The project contains a Python function `find\_obstacles` that identifies all points within a 10-meter range. These points are considered as obstacles. The z-values are ignored since we only work in 2D (x, y).

## HOW IT WORKS

1. The function calculates the Euclidean distance from the origin using x and y coordinates.  
2. If the distance is less than 10 meters, the point is marked as an obstacle.  
3. All such points are collected and returned as a list.

## RUNNING THE CODE

To run the code, save it in a Python file (e.g., `lidar\_solution1.py`) and execute it using a Python interpreter.  
Example:

```python  
python lidar\_solution1.py  
```

## TESTING

Sample Inputs and Outputs:

Test Case 1:  
Input: [(10, 0, 0), (-10, 0, 0), (0, 10, 0), (6, 8, 0), (8, 6, 0)]  
Output: []

Test Case 2:  
Input: [(10, 0, 0), (1, 1, 0), (0, 10, 0), (6, 8, 0), (8, 6, 0), (1, -5, 0)]  
Output: [(1, 1, 0), (1, -5, 0)]

Custom Run Output:  
Input: [(10, 0, 0), (1, 1, 0), (6, 3, 0), (5, -5, 0)]  
Output: [(1, 1, 0), (6, 3, 0), (5, -5, 0)]

## TEST RESULTS

The function passes all the sample and custom test cases correctly. It successfully detects all points within the 10-meter range.

## CONCLUSION

Our LIDAR-based obstacle detection system effectively detects and returns 3D points within a 10-meter range. It uses simple Euclidean distance calculations to determine proximity, and the implementation is efficient and accurate.