Ping Pong Game 🎮

💡 Project Idea:

The idea is to build a Ping Pong game using Python where the second player (ai) uses the A* (A-star) algorithm to move intelligently. The computer calculates the optimal movement path to reach the ball and block it, just like a human player would.

🎯 Project Objectives:

The goal is for one player to score 5 points first. The opponent uses the A* (A-star) algorithm to predict the ball's movement and automatically adjusts its paddle. The game supports different difficulty levels based on the speed and responsiveness of the AI, creating an intelligent and challenging gameplay experience.

🤖 PEAS:

Element	Description
Performance Measure	- Win/loss ratio against the human player - Accuracy of ball returns - Response time of the AI paddle - Number of successful interceptions - Point difference between players
Environment	- 2D field with boundaries - A ball following physics rules - Human and AI paddles - Real-time scoring system
Actuators	- Moving the paddle (up/down) - Adjusting paddle speed based on difficulty
Sensors	- Ball position tracking - Ball speed detection - Human paddle position - Collision detection (ball & paddles) - Score monitoring

🧩 ODESA:

Dimension	Description
Observability	**Fully Observable** – The AI sees the ball position, speed, and human paddle position.
Determinism	**Partially Deterministic** – Ball physics are predictable, but human actions introduce randomness.
Episodes	**Sequential** – Each state depends on the previous one (e.g., ball trajectory changes after paddle hits).
Static/Dynamic	**Semi-dynamic** – The environment changes due to player actions, but physics rules remain constant.
Agents	**Multi-agent** – AI vs. human in a **competitive** setting (goal: maximize AI score, minimize human score).

🤖 Agent Type – Goal-Based Agent

The second player (ai) is a **Goal-Based Agent**. Its main goal is to block or return the ball when it gets close. It calculates the ball's position and speed, then decides whether to move up or down to reach the best position. It doesn't move randomly, it selects actions based on achieving its goal, which makes it more efficient and intelligent in handling in-game situations.

🧠 Problem Formulation:

- **Initial State: Paddle at screen center (Y-axis).**
- **Goal State: Paddle aligned with ball's Y-position.**
- **States: All Y-axis positions the paddle can occupy.**
- **Actions: Move up/down (e.g., ±5 pixels).**
- **Transition Model: Action changes paddle Y-position.**
- **Path Cost: Total distance (or number of moves) to reach target position.**