SIMULATING A DELIVERY ROBOT

Introduction

In this assignment, you'll be working on a robot simulation using Python. The goal is to implement a program that simulates a robot navigating through an environment, delivering packages, and avoiding obstacles.

Task

You will simulate the robot's movement through a grid-based environment, avoiding obstacles, delivering packages, and displaying the simulation visually.

Requirements

Environment Setup:

- Define a grid size for the environment (10 x 10)
- Create a list of obstacle positions. Positions: [(3, 4), (5, 7), (8, 2)]
- Create a list of package delivery positions. Positions: [(1, 2), (4, 6), (7, 8)]
- Set the initial position of the robot.

Agent Initialization:

- Initialize an empty list to keep track of the robot's path.
- Initialize an empty list to store delivered packages.

Simulation Loop:

- Implement a simulation loop that continues until all packages are delivered.

Inside the loop:

- Display the environment using Matplotlib. Example: Obstacles as black squares ("ks"). Empty cells as black circles ("ko"). Delivered packages as green circles ("go"). Robot's path as a blue line ("b-"). Update the visualization after each iteration using plt.pause(0.5).
- Agent Behavior:
    1. Check if there are remaining packages. If packages are present:
    2. Check if the robot's current position matches the next package's position.
    3. If yes, deliver the package, update the path and the delivered packages list.
- Valid Moves:
    1. Calculate valid moves for the robot based on the current position and obstacles. Valid moves are steps that stay within the grid and don't collide with obstacles.
    2. Exclude the previous position from valid moves to avoid going backward.

3. How is the next move selected? As a first step, choose the move randomly among the valid moves. Is it efficient? Can you think of other ways to select moves?
4. Update the robot's position and the path.

Time Tracking

- Keep track of the time (iterations) the simulation has run.

Final Visualization

After all packages are delivered, create a final visualization.

Display the environment, delivered packages, and robot's path.

Tips

You'll use object-oriented programming principles to organize your code.