1. Theoretical Analysis

The code simulates an environment where an "agent" learns the probability distribution of objects in a grid-like world, predicting the locations of objects based on historical data. The program aims to enhance object-detection performance using a probabilistic model. The theoretical basis is a kind of artificial intelligence where the agent iterates over multiple random worlds to build a probability map of object locations.

The program has four main components:

- **Environment Creation**: Generates a 5x5 grid where each cell is either occupied by an object (1) or is free (0).
- **Training Agent**: The agent trains by simulating many worlds, learning the probability of each cell having an object.
- **Sorting Probabilities**: The probability distribution is sorted, helping the agent search cells in decreasing the likelihood of finding an object.
- **Performance Evaluation**: The agent's accuracy is tested in new worlds, measuring how well it detects objects based on the sorted probabilities.

2. Data Structure

- **Grid (world)**: A 2D list representing a 5x5 grid where `1` indicates an object, and `0` indicates free space.
- **Probability**: A 2D list of integers representing the learned probabilities of each cell containing an object.
- **Object Positions**: A list of tuples representing coordinates where objects are present.
- **Sorted Indices**: A list of grid coordinates sorted by likelihood, helping the agent prioritize cell-checking.

3. Algorithm to Function Representation

Function	Purpose
create_environment()	Generates a random 5x5 grid with objects placed at some cells randomly.
find_object_positions(world)	Identifies positions of all objects (1) in the generated grid.
train_agent()	Trains the agent by iterating over random environments and updating a probability grid.
sort_index_probability(probability_learned)	Sorts indices by learned probabilities to optimize the search order.

check_performance(world, probable_index_sorted)	Measures success and failure in detecting objects in a test grid.
show_performance(world, probability, probable_index, visited_indices)	Visualizes the grid, marking each cell as checked, missed, or successful.
main()	Initializes training, sorts probabilities, tests performance in simulated worlds, and visualizes results.

4. Implementation

The main algorithm flow consists of:

1. **Training Phase**: The agent is trained by repeatedly generating worlds and recording object locations, resulting in a probability map of each cell.
2. **Sorting**: The cells are sorted based on probabilities in ascending order.
3. **Testing Phase**: The agent tests its predictions in random environments, and success rates are recorded.
4. **Visualization**: Each cell in the grid is visually represented, showing the agent's current view and checked cells.

5. Input Test Cases Format

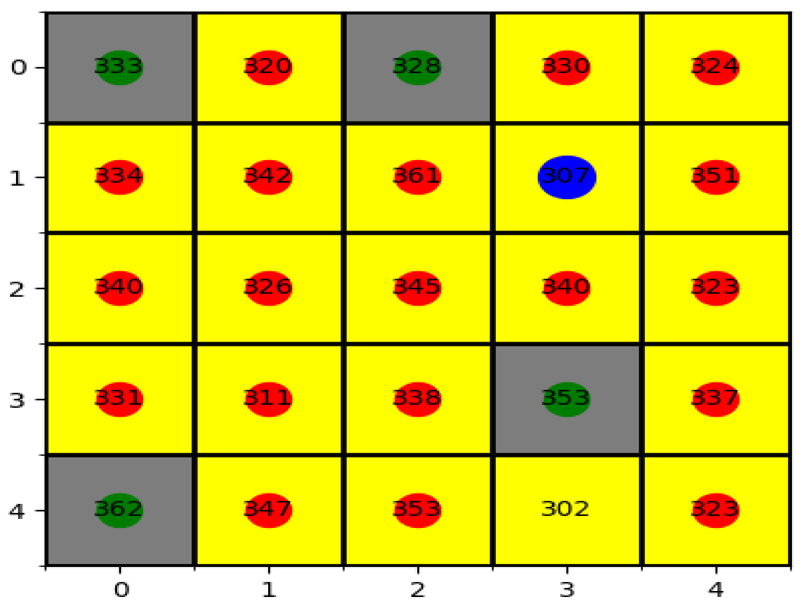
This code doesn't take traditional inputs. Instead, it simulates environments internally.

6. Output Format

The main output is a visual representation and performance statistics.

For each session:

- The sorted indices (cells ranked by likelihood) are displayed as a list.
- Success and failure rates are printed in percentage format.
- The visualization shows the grid with object positions, checked cells, successful finds (green), and failures (red).


0
1
2
3
4
333
320
328
330
324
334
342
361
307
351
340
326
345
340
323
331
311
338
353
337
362
347
353
302
323
0
1
2
3
4