Lab: Agent Interaction – Competition Rule

Objective

- To understand how agents interact with each other in a shared environment.
- To simulate competition among agents, where only one survives when two agents meet.
- To learn how simple interaction rules can lead to different system-level outcomes.

Language / Tool

- **Language:** Python
- **Tool:** Jupyter Notebook or Google Colab

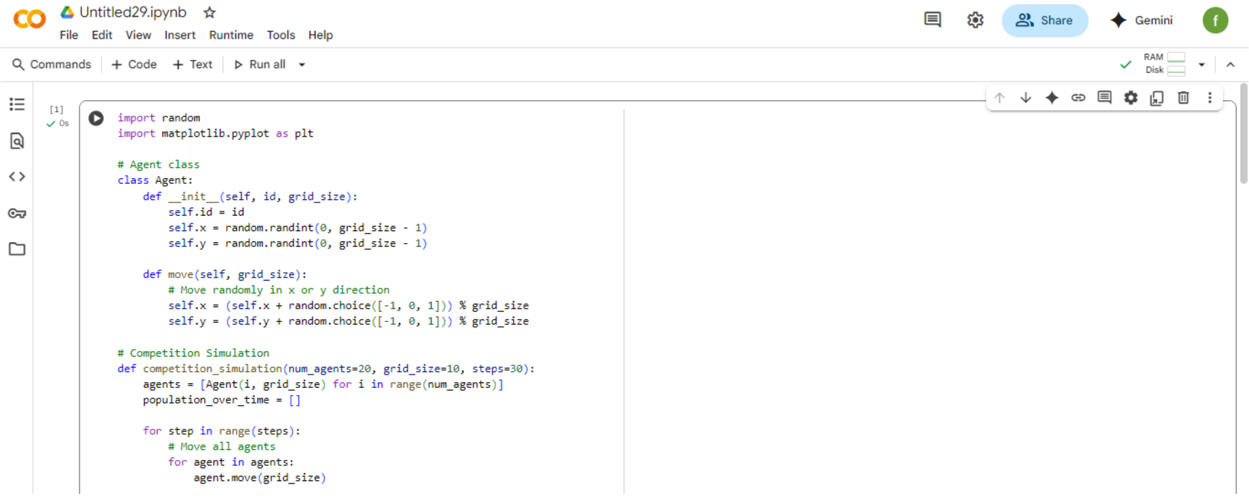
Theory

In Agent-Based Modeling (ABM), agents often do not exist in isolation. They must interact with other agents or with the environment. One important type of interaction is competition.

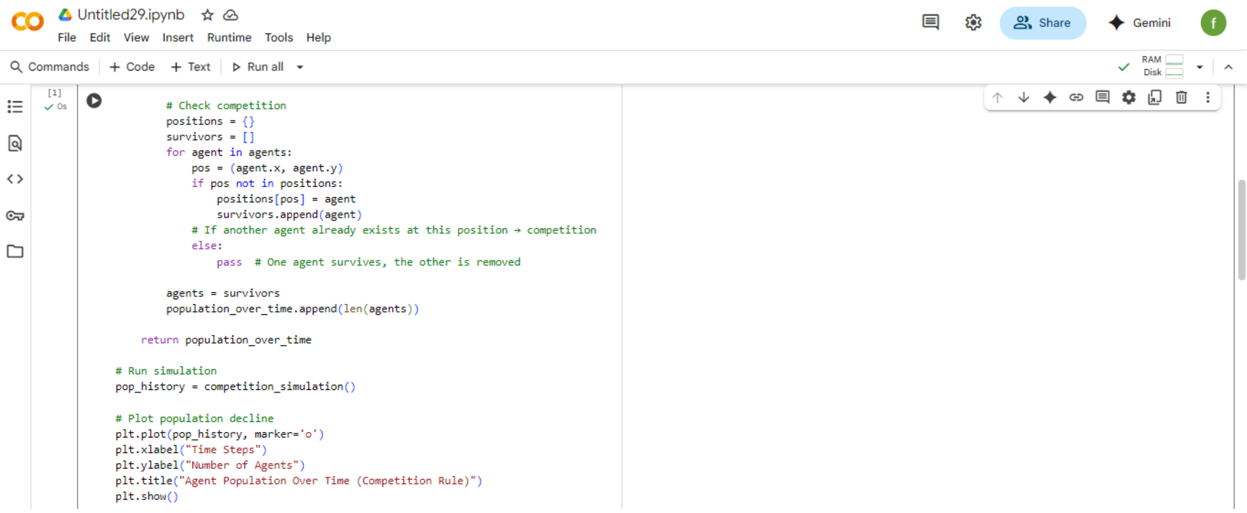
- **Interaction:** When two or more agents come into contact, they follow specific rules (e.g., fighting, cooperating, or ignoring each other).
- **Competition Rule:** If two agents occupy the same position, one of them is removed (survival of the fittest).
- **Emergent Behavior:** By running the simulation multiple times, we can observe how the total number of agents decreases over time due to competition.

Code

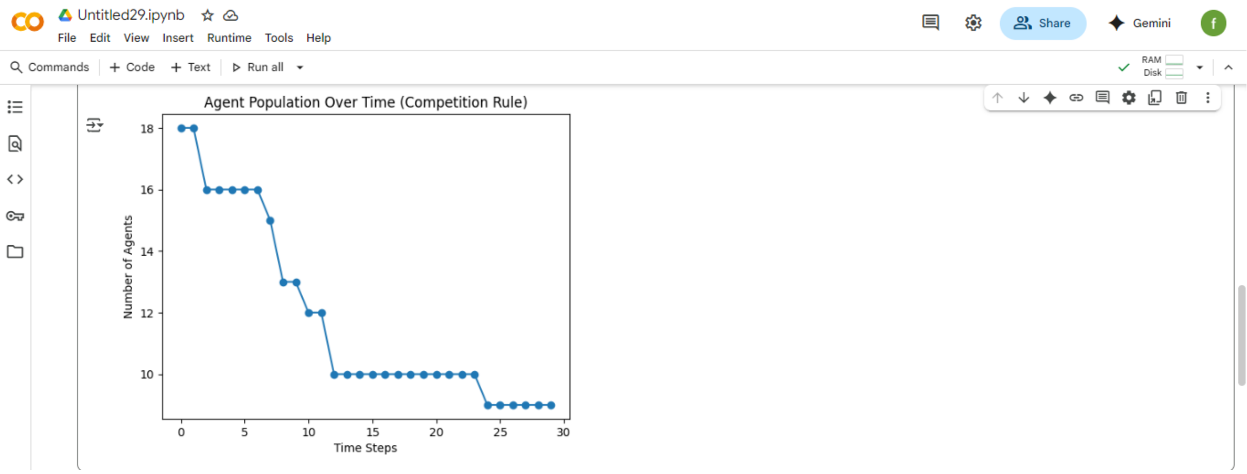
```
import random
import matplotlib.pyplot as plt


# Agent class
class Agent:
    def __init__(self, id, grid_size):
        self.id = id
        self.x = random.randint(0, grid_size - 1)
        self.y = random.randint(0, grid_size - 1)
    def move(self, grid_size):
        # Move randomly in x or y direction
        self.x = (self.x + random.choice([-1, 0, 1])) % grid_size
        self.y = (self.y + random.choice([-1, 0, 1])) % grid_size
# Competition Simulation
def competition_simulation(num_agents=20, grid_size=10, steps=30):
    agents = [Agent(i, grid_size) for i in range(num_agents)]
    population_over_time = []

    for step in range(steps):
        # Move all agents
        for agent in agents:
            agent.move(grid_size)

        # Check competition
        positions = {}
        survivors = []
        for agent in agents:
            pos = (agent.x, agent.y)
            if pos not in positions:
                positions[pos] = agent
                survivors.append(agent)
            # If another agent already exists at this position → competition
            else:
                pass  # One agent survives, the other is removed

        agents = survivors
        population_over_time.append(len(agents))

    return population_over_time

# Run simulation
pop_history = competition_simulation()

# Plot population decline
plt.plot(pop_history, marker='o')
plt.xlabel("Time Steps")
plt.ylabel("Number of Agents")
plt.title("Agent Population Over Time (Competition Rule)")
plt.show()
```


Untitled29.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
Gemini
RAM
Disk
[1]
0s

```
import random
import matplotlib.pyplot as plt

# Agent class
class Agent:
    def __init__(self, id, grid_size):
        self.id = id
        self.x = random.randint(0, grid_size - 1)
        self.y = random.randint(0, grid_size - 1)

    def move(self, grid_size):
        # Move randomly in x or y direction
        self.x = (self.x + random.choice([-1, 0, 1])) % grid_size
        self.y = (self.y + random.choice([-1, 0, 1])) % grid_size

# Competition Simulation
def competition_simulation(num_agents=20, grid_size=10, steps=30):
    agents = [Agent(i, grid_size) for i in range(num_agents)]
    population_over_time = []

    for step in range(steps):
        # Move all agents
        for agent in agents:
            agent.move(grid_size)
```



```
        # Check competition
        positions = {}
        survivors = []
        for agent in agents:
            pos = (agent.x, agent.y)
            if pos not in positions:
                positions[pos] = agent
                survivors.append(agent)
            # If another agent already exists at this position → competition
            else:
                pass  # One agent survives, the other is removed

        agents = survivors
        population_over_time.append(len(agents))

    return population_over_time

# Run simulation
pop_history = competition_simulation()

# Plot population decline
plt.plot(pop_history, marker='o')
plt.xlabel("Time Steps")
plt.ylabel("Number of Agents")
plt.title("Agent Population Over Time (Competition Rule)")
plt.show()
```



Explanation of Code

1. **Agent class**
   - Each agent has a position (x, y) on the grid.
   - The move() method changes its position randomly.
2. **Simulation loop**
   - All agents move.
   - If two agents land on the same position, only one survives.
3. **Population tracking**
   - The number of surviving agents is recorded after each step.
   - A graph shows how the population decreases over time.

Tasks

1. Modify the competition rule:
   - Instead of removing one agent, make them merge into a stronger agent.
   - Track how many merges happen.
2. Track survival:
   - Print which agent survives each encounter.
   - Keep a count of total competitions.
3. Experiment:
   - Change num_agents, grid_size, and steps.
   - Observe how grid size affects the chance of competition.