KLunar Navigator: AI for Lunar Exploration

Presented by Faiza Abdullah for ITAI-2372-Artificial Intel Applications.

Project Overview

The KLunar Navigator is an AI-driven rover navigation system for lunar exploration, utilizing Q-learning to autonomously navigate a 10x10 grid from a start point (0,0) to a target (9,9) while avoiding obstacles.

Objectives

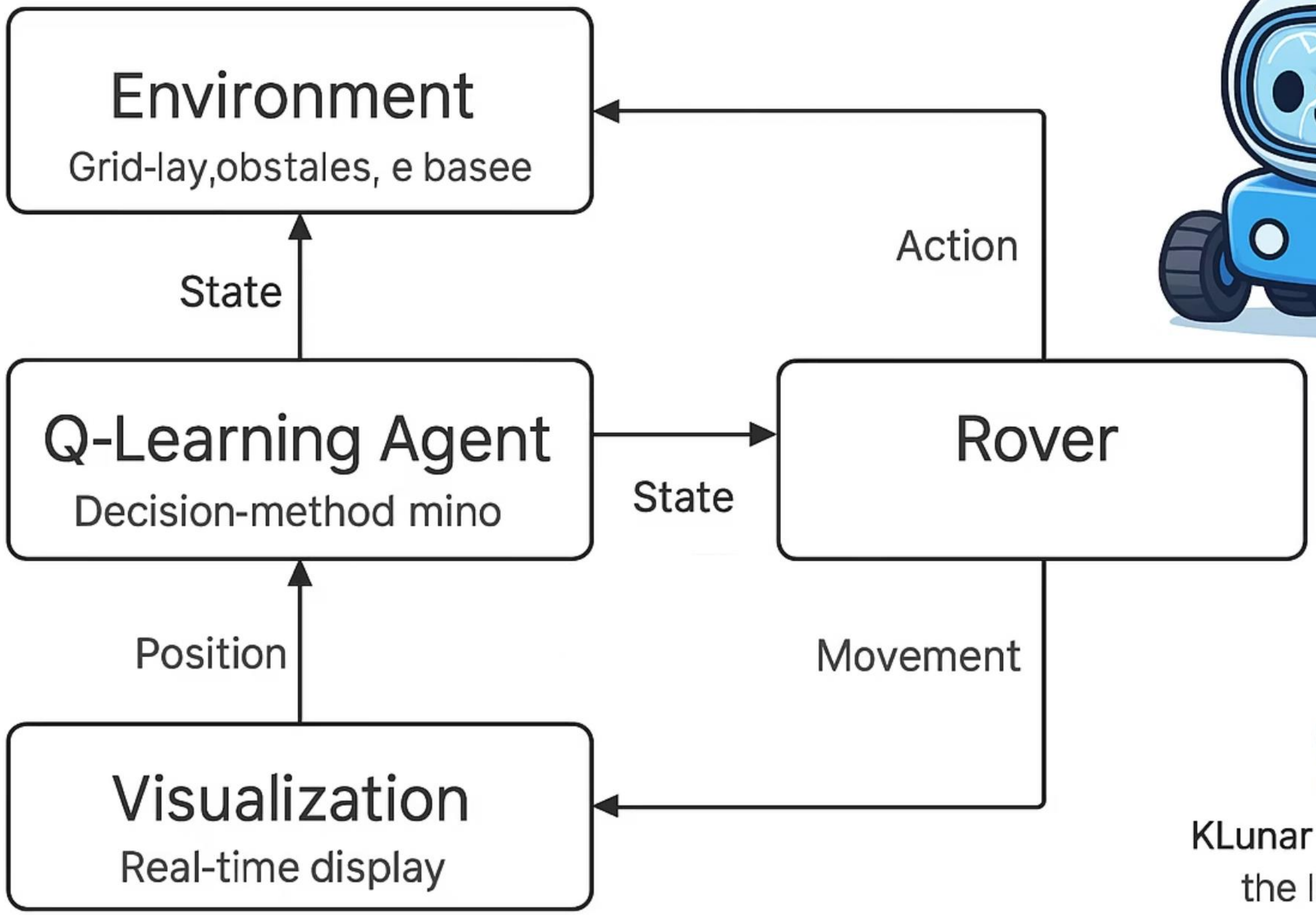
- Develop a functional Q-learning agent.
- Create a Pygame-based visualization.
- Ensure robustness against obstacles.
- Prepare for scalability.

Benefits

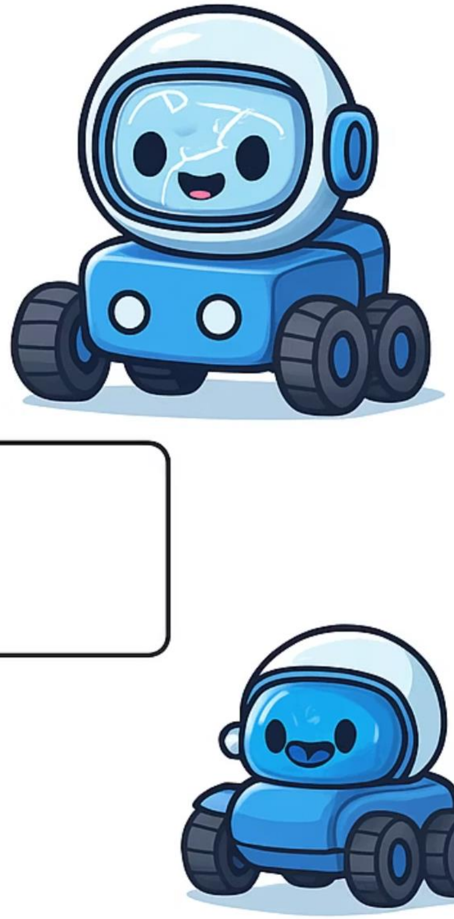
- Enhances RL understanding in robotics.
- Provides a testable prototype.
- Supports educational outreach.

System Architecture

The system integrates several key components: LunarEnvironment, Rover, QLearningAgent, and Visualizer, all managed by a main loop for autonomous navigation and real-time feedback.



Architecture



KLunar Navigator explores the lunar plains, learning to avoid craters and reach its base!

Solution Approach: Core Components

Environment Setup

10x10 grid with start (0,0), target (9,9), and random obstacles.

Rover Movement

Four actions (right, left, down, up) with energy tracking and boundary checks.

Q-Learning Agent

Q-table for 10x10x4 states, learning from rewards and saving data.

Visualization

Pygame window rendering rover, target, and obstacles in real-time.

Technical Details & Main Loop

Main Loop (main.py)

- 100 episodes, 200 max steps per episode.
- Manages setup, loop, and Q-table persistence.

Optimizations

- Reduced obstacles for faster convergence.
- Adjusted learning parameters.
- Max_steps ensures loop completion.

The system is built with Python 3.12, utilizing Pygame, NumPy, and asyncio libraries for efficient simulation and visualization.

Testing Plan: Validation Objectives

Validate the rover's ability to navigate from (0,0) to (9,9), avoid obstacles, and complete 100 episodes within 200 steps each.

1. **Initial Movement**

   Verify correct movement and reward for basic actions.

2. **Obstacle Avoidance**

   Confirm rover stays at current position with negative reward.

3. **Target Reach**

   Ensure target reached with positive reward and episode termination.

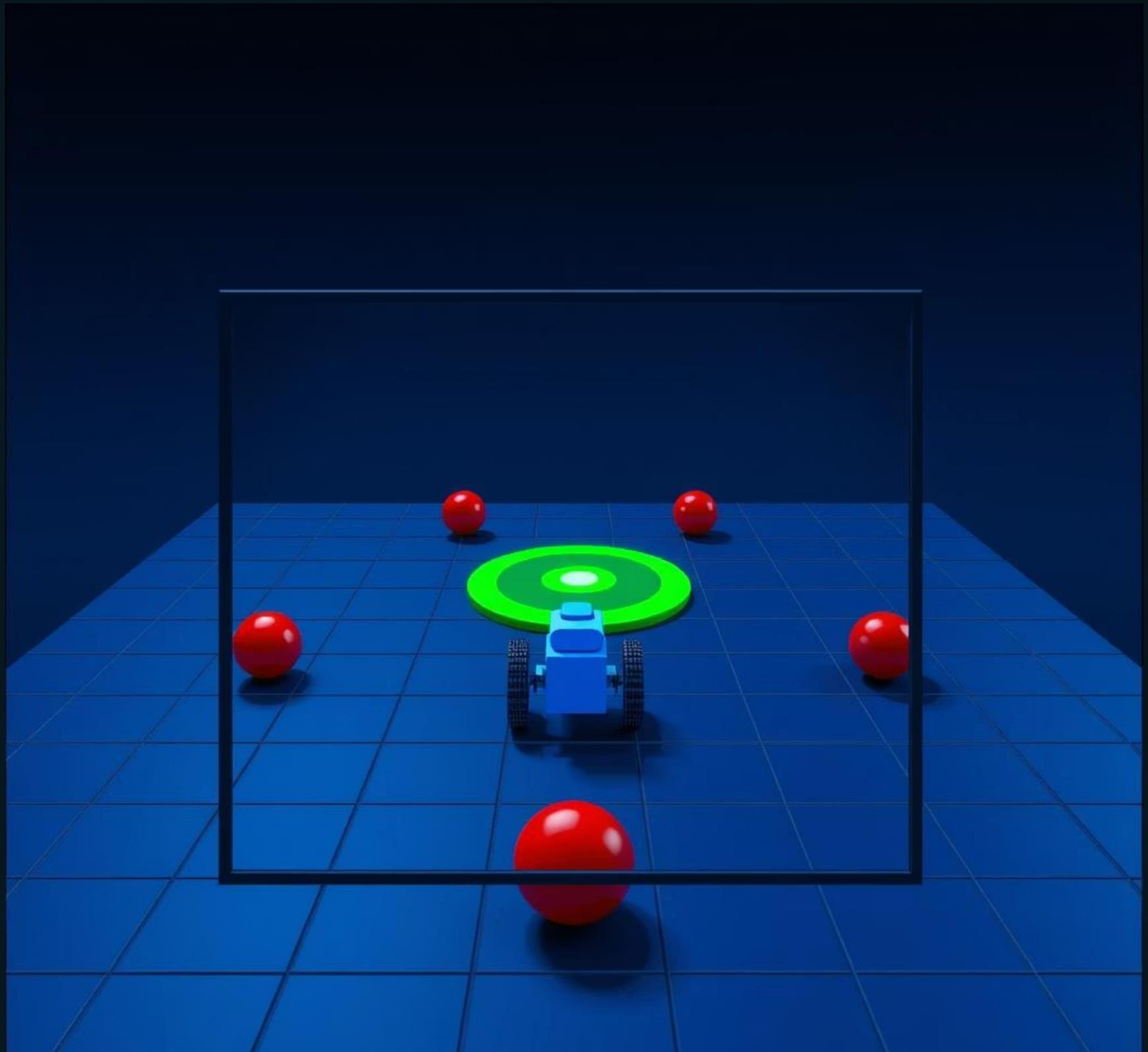
4. **Boundary Check**

   Test edge cases to confirm rover stays within grid.

Testing Success Criteria

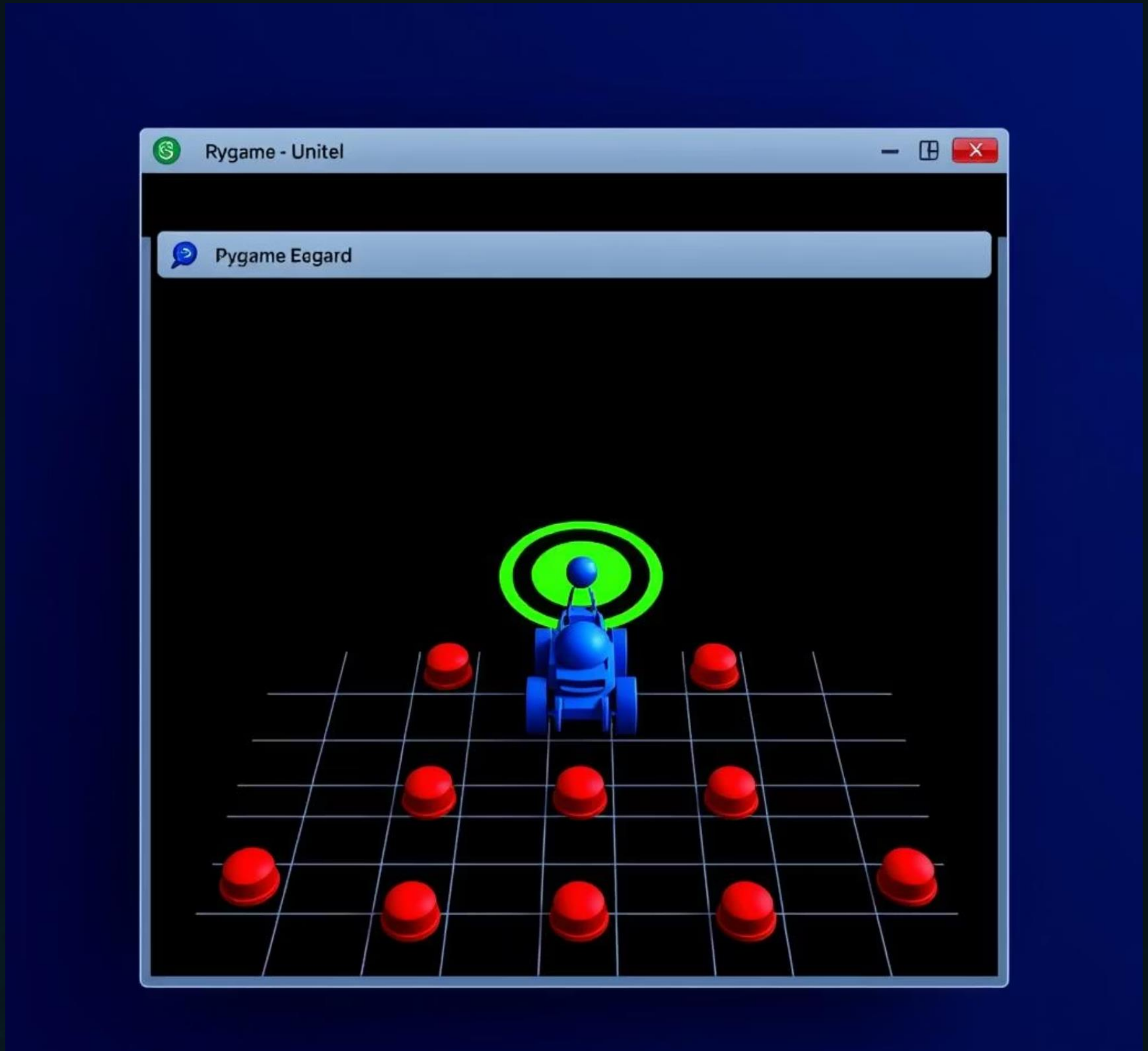
Visual Confirmation

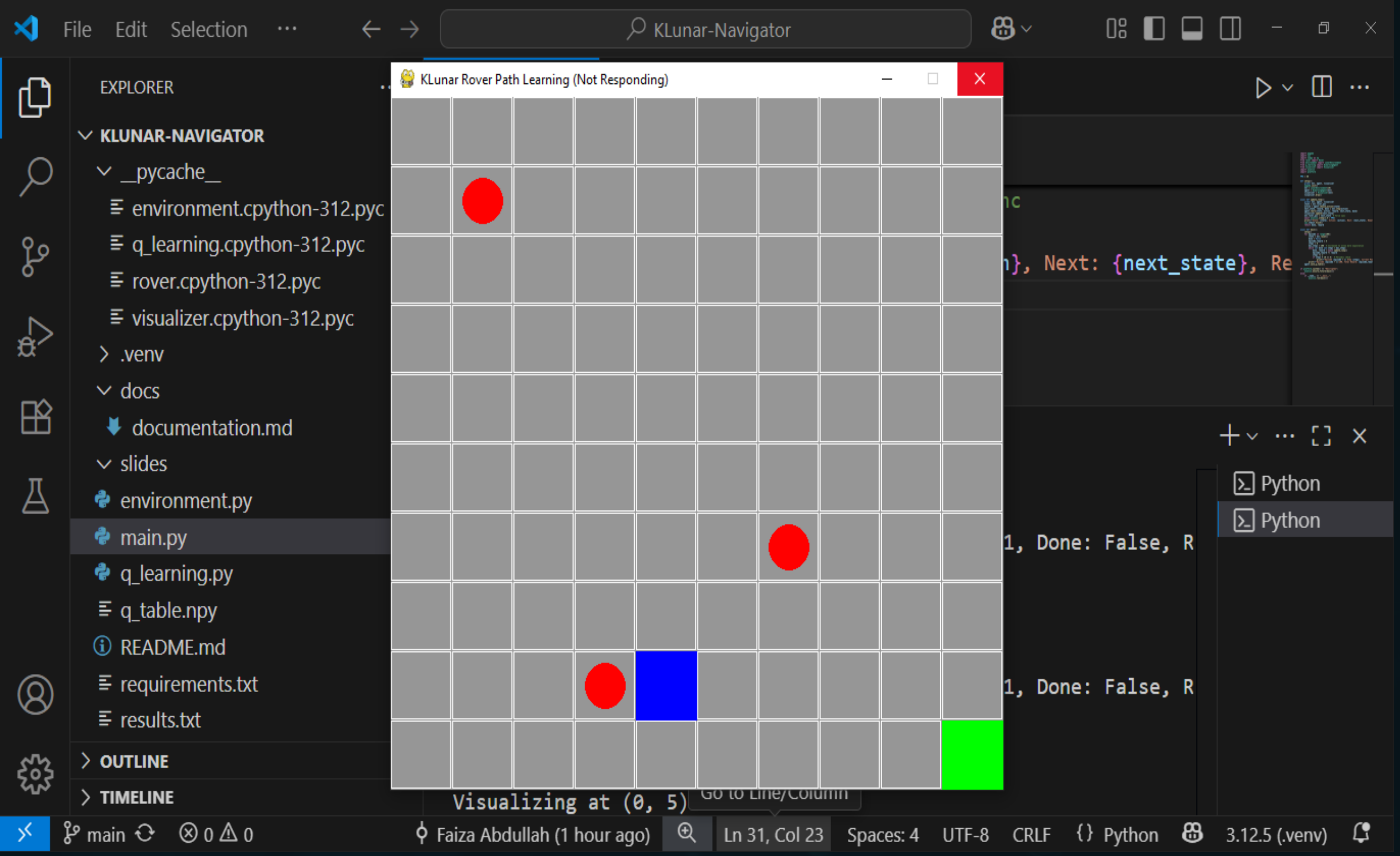
Rover moves visibly toward (9,9) in Pygame.

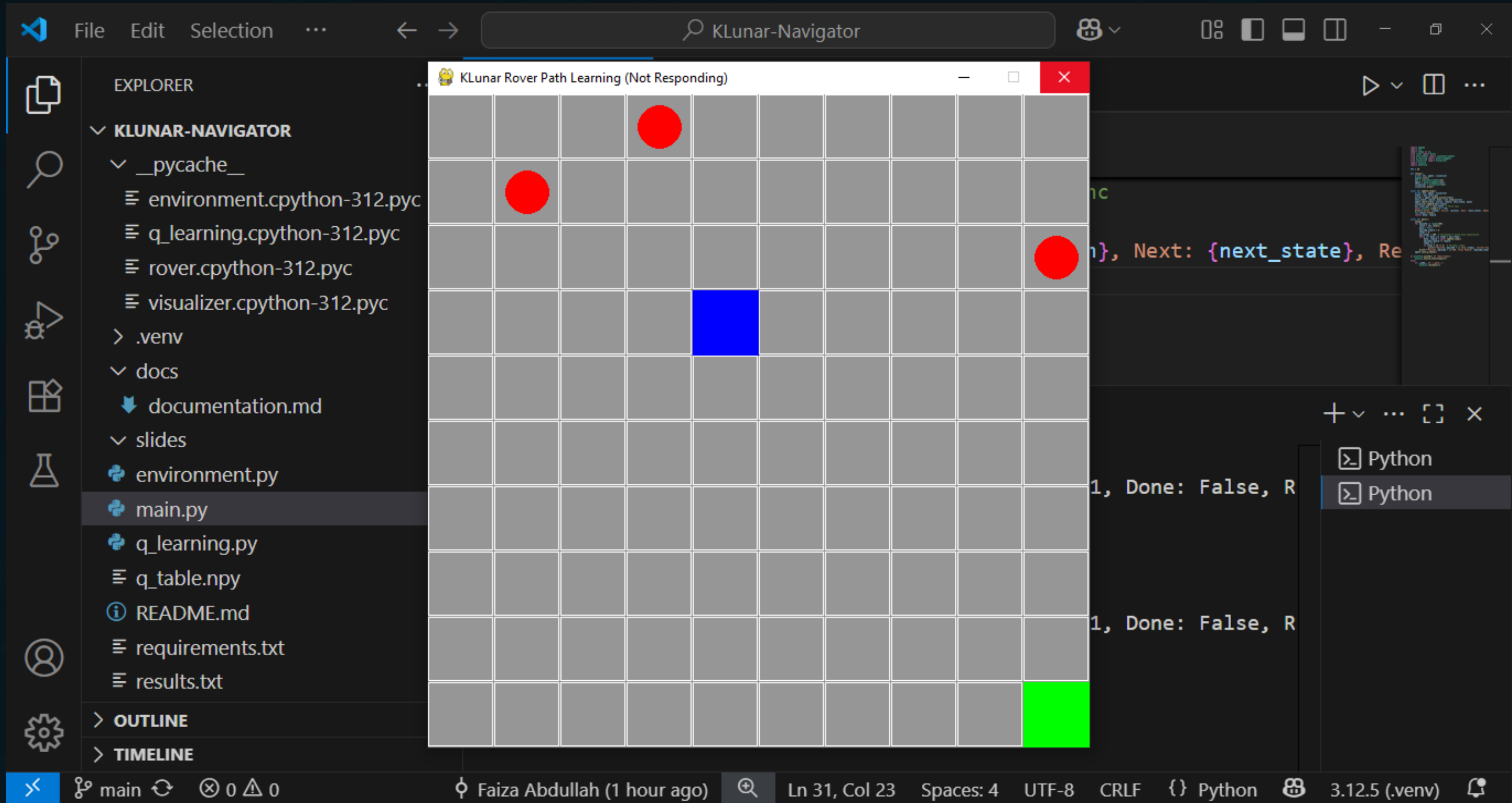


Episode Completion

100 episodes complete within 20,000 total steps.




File Edit Selection ...
KLunar-Navigator
KLunar Rover Path Learning (Not Responding)
EXPLORER
KLUNAR-NAVIGATOR
pycache
environment.cpython-312.pyc
q_learning.cpython-312.pyc
rover.cpython-312.pyc
visualizer.cpython-312.pyc
.venv
docs
documentation.md
slides
environment.py
main.py
q_learning.py
q_table.npy
README.md
requirements.txt
results.txt
OUTLINE
TIMELINE
}, Next: {next_state}, Re
1, Done: False, R
1, Done: False, R
Python
Python
Visualizing at (0, 5)
main
Faiza Abdullah (1 hour ago)
Ln 31, Col 23
Spaces: 4
UTF-8
CRLF
{} Python
3.12.5 (.venv)


File Edit Selection ...
KLunar-Navigator
KLunar Rover Path Learning (Not Responding)
EXPLORER
KLUNAR-NAVIGATOR
pycache
environment.cpython-312.pyc
q_learning.cpython-312.pyc
rover.cpython-312.pyc
visualizer.cpython-312.pyc
.venv
docs
documentation.md
slides
environment.py
main.py
q_learning.py
q_table.npy
README.md
requirements.txt
results.txt
OUTLINE
TIMELINE
}, Next: {next_state}, Re
1, Done: False, R
1, Done: False, R
Python
Python
main
Faiza Abdullah (1 hour ago)
Ln 31, Col 23
Spaces: 4
UTF-8
CRLF
{} Python
3.12.5 (.venv)

Future Enhancements

Dynamic Obstacles

Implement real-time obstacle generation for complex scenarios.

Multi-Rover Coordination

Develop algorithms for multiple rovers to cooperate.

3D Terrain Simulation

Upgrade visualization and navigation to a realistic 3D environment.

These enhancements will further bridge the gap between conceptual simulation and real-world space exploration.