Object Oriented Programming   Mickael Osorio J143040002   Jean Jedidia Leong J143040003

Reflection Report

Group Project – Reinforcement Learning and Object-Oriented Programming

Introduction

This group project helped us better understand **reinforcement learning** and **object-oriented programming** using **Python** and the **Gymnasium** library. The project was divided into three parts. First, we installed and tested Gymnasium. Then, we worked on the FrozenLake environment using Q-learning. Finally, we created our own project to apply object-oriented programming concepts in practice.

Before starting this project, we only had a basic theoretical understanding of reinforcement learning. We knew terms like agent, environment, and reward, but it was not very clear how they all work together in a real program. Working on this project helped us see how an agent actually learns through interaction and experience.

Another important objective was to improve our programming skills. In particular, we wanted to learn how to write cleaner and better-structured code using object-oriented programming. By progressing through the different parts of the project, we slowly became more confident and comfortable with both the concepts and the tools.

Part 1 – Installation and First Tests

The first part of the project was about installing **Python 3** and **Gymnasium**, then running a simple environment to check that everything worked correctly. Even though this part did not count much for the final grade, it was essential for the rest of the project.

We learned how to create and use a **virtual environment**, which helped us manage dependencies and avoid conflicts with other Python projects. This was a useful introduction to good development practices that are often used in real-world projects.

Running the sample code and the MountainCar environment allowed us to confirm that Gymnasium was correctly installed. Seeing the environment run without errors gave us confidence to move on to more complex tasks. This part showed us that a proper setup is the base of any successful programming project.

Part 2 – FrozenLake and Q-Learning

The second part of the project focused on solving the **FrozenLake-v1** environment using **Q-learning**. The objective was to reach a **success rate higher than 70%**, while respecting the constraints given in the project description. We were only allowed to adjust exploration-related parameters.

At the beginning, the agent failed most of the time. FrozenLake is a stochastic environment, meaning that actions do not always have the expected outcome. Even when the agent chose a good action, it could still slip and lose the episode. This made learning more difficult and sometimes frustrating.

By tuning the exploration parameters, we understood how important the balance between **exploration** and **exploitation** is. If the agent explores too much, it behaves randomly. If it explores too little, it may stop improving. Finding the right balance required several tests and careful observation of the results.

We also learned that **evaluation is just as important as training**. Training alone does not show the real performance of an agent. By running a fixed evaluation phase with almost no exploration, we were able to measure how well the agent actually learned the task. This part helped us clearly understand how reinforcement learning works in practice.

Part 3 – OOP Project: Warehouse Robot

The third part of the project was the most important and had the highest impact on the final grade. The goal was to design and implement a project that clearly demonstrates **object-oriented programming principles** such as abstraction, inheritance, and polymorphism.

We decided to create a **Warehouse Robot environment**. In this environment, a robot moves on a grid and must reach a target position. To make the project more interesting, we created two versions of the environment. The basic version is simple, while the advanced version includes obstacles and a battery limit, which increases the difficulty.

This part allowed us to apply OOP concepts in a concrete way. We used **abstraction** by defining base classes for agents, which describe what an agent should do without giving implementation details. We used **inheritance** to create different types of agents, such as a random agent and a greedy agent, while reusing common code. **Polymorphism** allowed the same training logic to work with different agents and environments without any changes.

We also implemented a Trainer class to manage training and evaluation. This helped separate responsibilities and made the code easier to understand. By following the Gymnasium interface (reset, step, render), our custom environment remained compatible with reinforcement learning tools.

Challenges Faced

During the project, we faced several challenges. One of the main difficulties was understanding why an agent was not learning properly. In reinforcement learning, problems are not always obvious. The issue can come from rewards, exploration strategy, or simple implementation mistakes.

Another challenge was creating a custom environment that follows the Gymnasium API correctly. We had to be careful with action spaces, observation spaces, and return values. Even small errors could cause unexpected behavior or crashes.

Working as a group also required good communication. We had to divide tasks, share progress, and help each other when needed. This improved our teamwork and collaboration skills.

What We Learned

Through this project, we learned many important things:

- How reinforcement learning agents interact with environments
- How Q-learning works in practice
- Why exploration and evaluation are important
- How to apply object-oriented programming correctly
- How to structure code in a clean and modular way

We also learned that progress often comes after several failures. Many attempts did not work at first, but each mistake helped us better understand the problem.

Conclusion

To conclude, this group project was a very valuable learning experience. It helped us connect theoretical concepts with real implementations and improve both our programming and teamwork skills. By combining reinforcement learning with object-oriented programming, we gained a better understanding of how intelligent systems are designed.

Although the project was challenging at times, it was also very rewarding. We now feel more confident using Gymnasium, reinforcement learning algorithms, and object-oriented programming concepts. The experience gained from this project will be useful for future studies and projects.