# Reinforcement Learning -
# Final Project - 2024

The main goal of this final assignment is to summarize the key topics (theoretical & practical) that we have covered in the course integrating some theoretical and practical aspects.

Part A - Tabular Methods

Part B - Deep RL methods.

In this exercise, you will solve two variations of the "Minigrid" environment. MiniGrid is a lightweight, 2D grid-world environment with goal-oriented tasks. The agent in these environments is a triangle-like agent with a discrete action space. The tasks involve solving different maze maps and interacting with different objects such as doors, keys, or boxes.

Minigrid documentation:

https://minigrid.farama.org/environments/minigrid/

In this assignment there are two minigrid environments:

1. Random**Empty**Env_10
2. Random**Key**MEnv_10

In both parts you can use reward shaping. Which means that you can change the original rewards system.

E.g. "If door was closed and now is open - reward = 10"

E.g. "-1 for every movement"

The new rewards system should be for **events** and for logics such holding flags etc,

**Part A - Q-Learning**

To render the game (random actions), use the following notebook:

https://colab.research.google.com/drive/1t-X7y03xD8u4hYMA5p0YpUv9k0c7HF3s?usp=sharing

To run the code you need the minigrid_x.py in your temporary disk.

https://drive.google.com/file/d/1pjyRwXMnDKBbbxKGjl20bL78s4Um3kyu/view?usp=sharing

In this part, try to "solve" it as fast as you can (less episodes) - this is a competition part.

- Analyze the env. MDP? Episodic? Discrete/continuous action? Discrete/continuous states? Fully/partial observable?
- You need to solve it with Q-LEARNING
- Show relevant graphs comparing the different approaches
- How to represent the state? State size? Q-table size?
- Different hyperparameters: Learning rate, Epsilon
- Choose discount factor
- Different initializations
- Different considerations for the two environments
- Approaches for Exploration-Exploitation tradeoff.
- Graphs for the training stage and
- For the inference stage – Average steps to finish episode
- Evaluate both the good and bad points of your approach

**Part B - DEEP RL**

Here, use the following notebooks:

ENV1 - EMPTY ENV

https://colab.research.google.com/drive/1g07dzScKCpGxzO3TujmRsCy5vH6aUBIH?usp=sharing

ENV2 - KEY DOOR ENV

https://colab.research.google.com/drive/1V4BKBaW_GYS0BeO-PaSaURy6nBpZ-vXt?usp=sharing

To run the code you need the minigrid_x.py in your temporary disk.

https://drive.google.com/file/d/13_yph8slZ6mqur7d1tsC1EB_pO8sdz4a/view?usp=sharing

Note: the files are different from Part A.

**Note: There are a lot of implementations on the web of these algorithms. We want to see your different considerations while you solve the environments**

- Solve it with different algorithms from the DEEP RL part of the course
- Discuss the advantages/disadvantages of the approaches relating to your minigrid mission.
- Show relevant graphs comparing the different approaches
- Add preprocessing to the input image
- Different hyperparameters: Learning rate, Epsilon, replay buffer size, target network update rate, initializations' etc.
- Different considerations for the two environments
- Approaches for Exploration-Exploitation tradeoff.
- Graphs for the training stage
- For the inference stage – Last 100 episodes "average num of steps to finish episode"
- Evaluate both the good and bad points of your approach

Guidelines:

1. You are not allowed to use an existing RL libraries. Write the algorithm yourself.

2. You are not allowed to use deep reinforcement learning approaches or RL algorithms that you haven't learnt in the course.

3. For the most successful experiment, in each exercise, Show (In the notebook and report) the number of steps it took the agent to solve the environment, and a convergence graph (Rewards according to the number of episodes). All training outputs will be displayed in the notebook.

4. Add video clips (In the notebook) that shows the agent in the middle of the training process and after the learning completed and converged,

5. Even if you did not fully solve an environment - supply graphs describing the average rewards on X episodes (after finishing the training phase).
   Submit a final report in pdf format when your ID numbers are included in the file name.

6. You need to submit your code as Google Colab notebooks links. Such that the course staff can run it. Also, add the links to the report.

7. Write a clean code! Separate code cells and make extensive use of text cells. **A sloppy notebook will result in a lower grade.**

8. Do not change anything in the notebook after the final submission date. A notebook that has been run/changed after the submission date will be automatically disqualified.

**Due Date:**

## Final Project submission is due April 7th at 23:59.

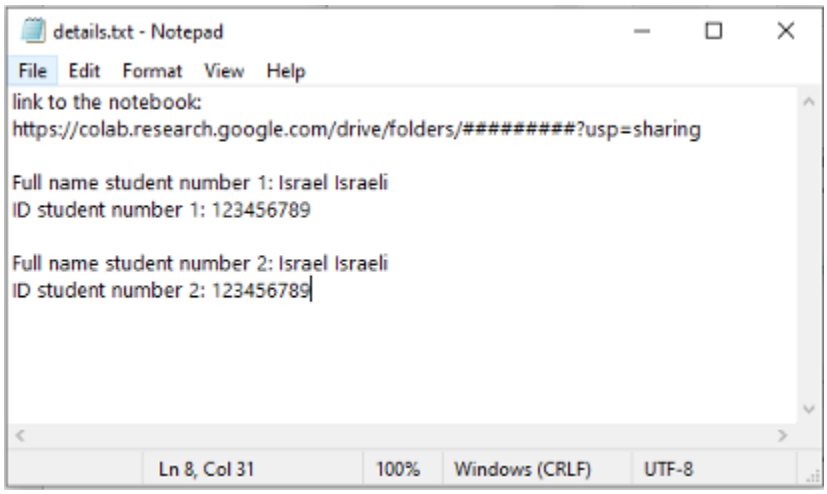
The report will be submitted to the submission box. The file name of the report will contained your IDs as following: **report_ID1_ID2.pdf**

**Do not forget to include the project title, your name and ID in this file.**
**Max number of pages: 10 (but you don't have to use them all !!).**

The notebook will be presented in an orderly and clean manner, it will contain separate code cells and text cells that explain the actions performed.
** **Very important** ** - when submitting, the notebooks will contain **all the outputs** relevant to the training results.

To the submission box you will also submit an **explainer** file that contains instructions and explanations on how to operate your notebook and other relevant details that need to be known to those who want to use your notebook.

Enter your details in the text file named **submit.txt**, the address of your notebook, the names and ID of the two partners, as follows:




To sum up, in the submission box, submit the following files:

1. details.txt
2. report_ID1_ID2.pdf
3. explainer.txt

**Team size**
The project will be performed in groups of 2 students.

**Academic Integrity**

Team/Student may not copy code from other teams/students. Copying answers or code from other students for a project is a violation of the university's honor code and will be treated as such. All suspicious activity will be reported to the head of the department and the university authorities.

Giving code to another student is also considered a violation. Students are responsible for protecting their own work from copying.

If you build some of your code on existing work and utilize existing code (your own or code found on the web), you must give proper attribution to all existing work that you used and make it clear what you changed and contributed. Any unattributed or uncited work that you use will be considered a breach of academic honesty and dealt with according to the course policy in the syllabus.