1 Task 01

You are tasked with creating an AI for the game of chess. To solve the problem using Reinforcement Learning, you have to frame the game of chess as a Markov Decision Process (MDP). Describe both the game of chess formally as a MDP, also formalize the respective policy.

State Space S: Every possibility of the chess pieces on the checkerboard is a state in our state space.
Action Space A: Every possible move of each chess piece of our team – invalid moves are ignored
State Transition function: Partly known (own moves) and unknown (opponent moves) – stochastic distribution of (valid) opponent moves
Reward Function: There is only a (unbiased) reward for transitioning to the terminal state: 1 or -1 for Win or Lose. One could define more (biased) rewards, for example for capturing opponents' pieces or losing own ones.

Our policy pi(a|s) is the probability of performing the chess move a € A when we have a specific board configuration s € S. The policy has to be defined for every state-action pair.

2 Task 02

Check out the LunarLander environment on OpenAI Gym: Check out this Link!. Describe the environment as a MDP, include a description how the policy is formalized.

State Space S: All possible states s are given by a vector with the x- and y-coordinates, the velocities in both directions, the angle and the angular velocity of the lander and the information whether its legs are touching the ground.

Action Space A: Do nothing, fire left orientation engine, fire main engine, fire right orientation engine.

State Transition Function: It's deterministic but it's unknown(Physics-based world? Which gravitational force? How powerful are the engines? …)

Reward Function:

- Landing in the Landing Zone: +100…140
- Transitioning to a crashed state: -100
- Transitioning to a not crashed state: +100
- Each leg that touches the ground +10

- Firing main engine: -0.3
- Maximum Reward (Solved) is +200

Our policy pi(a|s) is the probability of performing one of the four actions a € A when the lander is in a specific state s € S. The policy has to be defined for every state-action pair.

3 Task 03

Discuss the Policy Evaluation and Policy Iteration algorithms from the lecture. They explicitly make use of the environment dynamics (p(s′, r | s, a)).

- Explain what the environment dynamics (i.e. reward function and state transition function) are and give at least two examples.

The dynamics of an environment describes how the state changes when you take an action.

For Self-Driving-Cars the environment dynamics is everything that happens in the real world (e.g. Physics, other agents, consequences of own actions, …) that influences the state-transition function. The reward function can be partly defined by the user or engineer (or whoever…).

In contrast, Go has more restricted environment dynamics, because there is only you and your opponents' actions which influence the environment. The state-transition function defined and restricted by the game rules. It's partly unknown, because we don't know the opponents' actions. The reward function is only 1 or -1 for a winning or losing state.

- Discuss: Are the environment dynamics generally known and can practically be used to solve a problem with RL?

Especially in real life settings, the state transition functions are mostly unknown, because of the complexity of real world and number of other agents.
Even without knowing the whole environment dynamics, it can still be possible to (partly-)solve RL problems, e.g. in chess, because we can fill the part of the enemy. In other cases the possibilities are rather limited e.g. a simulation of a robotic application, which always comes short of the real world.