To Do:

- Find a way to increment AnyLogic model time whenever train-function has completed one iteration (or when reset is called). Make sure that the animation is rendered.
- Understand the role of seeds and why despite setting identical seeds outcomes differ between runs → **This is still not fully clear**. I have discovered that for whatever reason, the seed we set in the configuration of the learning algorithm is not used for the ε-greedy policy (it uses system time instead). But even when completely switching off the random elements in the ε-greedy policy I still cannot get identical results between runs. The only way to get identical results between runs is to both switch off the randomness in the ε-greedy policy and to change the reset-function to initialise all state variables to 0 instead of randomising them. This would seem to suggest that despite setting a fixed seed for the AnyLogic simulation, the results of the uniform() function, which are used to initialise the state variables inside reset, are not identical across runs.
- Implement a terminal condition to stop the AnyLogic simulation when train() is finished → **Done**. Implemented two ways of doing this. Either a line in the close() function can be uncommented which will finish the simulation when the training is done. Alternatively, if we do not wish to finish the simulation once training is done (for instance because we wish to load the trained agent), there is also an "exit" button in the main environment now.
- Correct the placement of the cart animation in the main environment → **Done** (kind of; AnyLogic animations, especially their placement inside the environment, still puzzle me.).
- Possibly save some training output to a log file → not sure whether this is useful since the learned policy is saved so in a way we can evaluate the results already. If we do want to save in a log, what information should it contain?
- Implement the loading of the trained policy into a new instance of the MDP and enable it to run for demonstration purposes → **Done** but same problem as with train. The trained agent with the learned policy can be loaded into the environment but as soon as it is told to run, the AnyLogic simulation pauses until the loop is completed.
- Is there a way to suppress or change traceln() messages for specific calls of a given function? When we run the trained agent, some of the

same functions which are also used for training are called and it may be nice to change some of the messages they print to the console based on whether they are called during or after training.

- Compare results of the learning algorithm to those obtained using the OpenAI environment → Need to do this using Ramin's computer which can run the gym server without issues.
- Once the Cartpole example functions properly, begin work on implementing a different, original learning task (e.g. Callcentre) using the same principle.