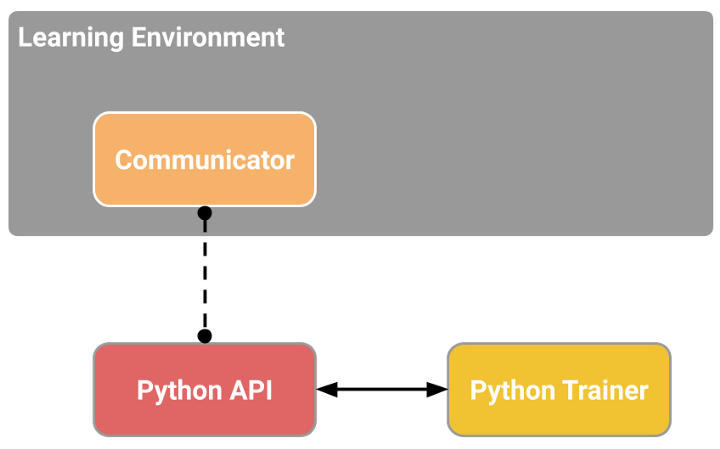
ML-Agents

Unity ML-Agents is a toolkit for the game engine Unity that allows us to create environments using Unity or use pre-made environments to train our agents.

The six components:



1. **Learning Environment**: which contains the Unity scene (the environment) and the environment elements (game characters).
2. **Python Low-level API**: which contains the low-level Python interface for interacting and manipulating the environment. It’s the API we use to launch the training.
3. **External Communicator**: connects the Learning Environment (made with C#) with the low-level Python API (Python).
4. **The Python trainers**: the Reinforcement algorithms made with PyTorch (PPO, SAC…).
5. **The Gym wrapper**: to encapsulate the RL environment in a wrapper.
6. **The ‘PettingZoo’ wrapper**: It is the multi-agent version of the gym wrapper.

Inside the Learning Component, we have two important elements:

1. **Agent Component**: the actor of the scene. We’ll train the agent by optimizing its policy (which will tell us what action to take in each state). The policy is called the Brain.
2. **Academy**: This component orchestrates agents and their decision-making processes. Think of this Academy as a teacher who handles Python API requests.

The Academy will be the one that will send the order to our Agents and ensure that agents are in sync:

* Collect Observations
* Select your action using your policy
* Take the Action
* Reset if you reached the max step or if you’re done.

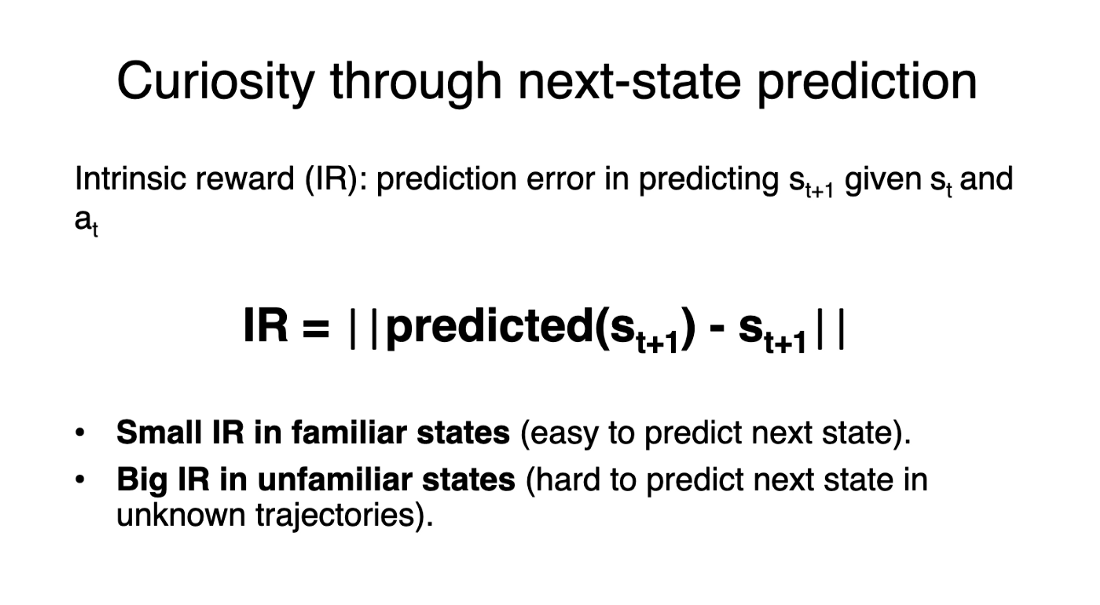
**Curiosity**:

To understand what Curiosity is, we first need to understand the two major problems with RL:

1. sparse rewards problem: that is, most rewards do not contain information, and hence are set to zero.
2. The extrinsic reward function is handmade; in each environment, a human has to implement a reward function. But how we can scale that in big and complex environments?

A solution to these problems is to develop a reward function intrinsic to the agent, i.e., generated by the agent itself. The agent will act as a self-learner since it will be the student and its own feedback master.

This intrinsic reward mechanism is known as Curiosity because this reward pushes the agent to explore states that are novel/unfamiliar. To achieve that, our agent will receive a high reward when exploring new trajectories.



Using Curiosity will push our agent to favour transitions with high prediction error (which will be higher in areas where the agent has spent less time, or in areas with complex dynamics) and consequently better explore our environment.