* **Python Low-Level API** –
  + Which contains a low-level python interface for interacting and manipulating a learning environment.
  + Unlike the learning environment, the python API is not part of unity, but lives outside and communicates with unity through the communicator.
  + This API is contained in a dedicated mlagents\_envs python package and is used by the python training process to communicate with and control the academy during training.
  + However, it can be used for other purposes as well.
  + For example, you could use the API to use unity as the simulation engine for your own machine learning algorithms.
* **External communicator** –
  + Which connects the learning environment with the python low-level API.
  + It lives within the learning environment.
  + Python trainers which contain all the machine learning algorithms that enable training agents.
  + The algorithms are implemented in python and are part of their own mlagents python package.
  + The package exposes a single command-line utility mlagents-learn that supports all the training methods and options outlined in this document.
  + The python trainers interface solely with the python low-level API.
* **Gym Wrapper (not pictured)-**
  + A common way in which machine learning researchers interact with simulation environments is via a wrapper provided by OpenAI called gym.
  + We provide a gym wrapper in the ml-agents-envs package and instructions for using it with existing machine learning algorithms which utilize gym.
* **PettingZoo Wrapper (not pictured)-**
  + PettingZoo is python API for interacting with multi-agent simulation environments that provides a gym-like interface.
  + We provide a PettingZoo wrapper for Unity ML-Agent’s environments in the ml-agents-envs package and instructions for using it with machine learning algorithms.

The Learning Environment contains two Unity Components that help organize the Unity scene:

* **Agents** - which is attached to a Unity Game Object (any character within a scene) and handles generating its observations, performing the actions it receives and assigning a reward (positive / negative) when appropriate.
  + Each Agent is linked to a Behavior.
* **Behavior** - defines specific attributes of the agent such as the number of actions that agent can take.
  + Each Behavior is uniquely identified by a Behavior Name field.
  + A Behavior can be thought as a function that receives observations and rewards from the Agent and returns actions.
* Behavior can be of one of three types: **Learning, Heuristic or Inference**.
* A **Learning Behavior** is one that is not, yet, defined but about to be **trained**.
* A **Heuristic Behavior** is one that is defined by a **hard-coded set of rules** implemented in code.
* An **Inference Behavior** is one that includes a **trained Neural Network file**.
* In essence, after a Learning Behavior is trained, it becomes an Inference Behavior.
* Every Learning Environment will always have one Agent for every character in the scene.
* While each Agent must be linked to a Behavior, it is possible for Agents that have similar observations and actions to have the same Behavior.
* In our sample game, we have two teams each with their own medic.
* Thus, we will have two Agents in our Learning Environment, one for each medic, but both of these medics can have the same Behavior.
* This does not mean that at each instance they will have identical observation and action values.

**3.3 Code Snippet**