Imports and Languages in Open Evolve

After reading through the example code, the nature of how Open Evolve interacts with programming libraries is interesting. It would stand to reason that the LLM used to power the evolution of the program is the one providing context for how the library works itself. That means that a custom built library is NOT intended for the Open Evolve framework unless its given in the context sent to the Open Evolve in the first place, however I am currently unaware of how deep the system prompt can go and whether the AI can efficiently navigate a code base if given the entire code base as context.

It seems that in the Open Evolve example programs the most common python libraries are tested, numpy, matplotlib, scipy. These example problems are all more math focused tasks so it would make sense for the focus of their imports to be the same. Outside the LLMs request which you could say the LLMs being leveraged to create an output that matches more or less what the user is looking for. However this may be hiding the inadequacies of the system of open evolve, like many artificial Intelligence applications do so. More common libraries must be tested.

The only two programming languages that are shown as examples are Rust and Python. While an LLM should be able to write perfectly adequate C,C++,C#Java, and other programming languages based on my experience with them, it is unknown how Open Evolves efficiency with pointers, nested classes, programming language specific pitfalls and errors(like HTML 5+”5”) and other challenges that the language poses.

This means for our research, the evaluation script of Open Evolve must look after how many instances a library or language incorrectly writes or iterates over and save it as a data point, relative to the specific problem at hand.