The Goal:

To be able to find a relationship between how shifting the variable parameters of a Liquidity Protocol's constant function impacts macro-outcomes. This means investigating the link to the divergence (impermanence) loss for liquidity providers as time evolves and investigating the effects of trades on the slippage.

Mechanism Explained:

We will create a class that will inherit the functionalities provided to us by the other teams. Then using these functions, we will specify the parameters that we want to test for a given simulation. From there we will run a time-step based simulation, which will randomly sample actions from traders or liquidity providers, and see how the liquidity pool evolves. We can repeat this process multiple times for a given parameter and obtain a picture on how robust the mechanisms through which the liquidity protocol in question operates.

What does this mean concretely?

1. Initialize the Liquidity Pool
    a. Designate Liquidity Provider, and how much liquidity they provider
    b. Specify the values of the parameters for the Liquidity Protocol
2. Sample a random action (Liquidity or Trade)
    a. Using a pre-defined distribution to model the behaviour of the agent we sample a point from this curve to obtain an action
3. We implement the consequences of the said action on the Liquidity Pool itself
4. We calculate the variables we want to analyse (divergence, slippage) and take account
5. Repeat from Step 2 a "sufficient" number of times

We will do this process many times, for each specification of the parameter of a given liquidity protocol, and obtain a picture of the workings of the liquidity protocol itself.

Extensions and Places to Code

- We can modify the sampling process to be based on and react to the current state of the Liquidity pool
- We can add multiple stakeholders acting in unison
- We can add a delay to the arbitrageur and see how the actions evolve and how the pool diverges from the correct states.