1. Rules for Evaluating Expressions with Environmental Diagrams

Remember: The purpose of drawing environmental diagram is to visualize how STk evaluates expressions.

Preparational work: Start by drawing **Global Environment**, set it as your **Current Environment**

The RULES:

(1) If the expression is a **constant**, self-evaluate, no need to change the world

(2) If the expression is a **symbol**, find binding in Current Env. If not found, go to the env. that contains the current env and try to find it there. Repeat process until the binding is found. Error if no binding found.

(3) If expression is a **procedure call**, then:

- If the procedure being called is a **lambda procedure**, follow these **lambda-proc call rules**:
  (a) If procedure being called is a symbol, find the function-bubble that the symbol binds to. If it is a lambda expression, create a new function-bubble in the current environment.
  (b) Evaluate the arguments by recursively applying these same rules **(*INSIDE OUT RULE*)**
  (c) Find out which environment the function you are calling lives in, create a new environment in that environment you found. bind formal parameters to their respective arguments (THIS IS THE ONLY TIME YOU CREATE A NEW Environment)
  (d) Set the new Environment as your **current Environment**, evaluate the body of the procedure by recursively applying these same rules
  (e) Go back to the previous current environment after procedure call finishes.
- If the procedure being called is a **primitive**, then follow these rules:
  (a) Evaluate the procedure and arguments by recursively applying these same rules
  (b) Apply the procedure by magic

(4) If the expression is a **lambda expression**, then create a bubble inside the current environment. The bubble should contain information about parameter and body.

(5) If the expression is a **define expression**, convert so it does not use syntactic shortcut, then add a new binding to the **current Environment**

(6) If the expression is a **set! expression**, then change the *first available binding*

(7) If the expression is a **let expression**, convert it to a lambda procedure call, then follow the **lambda-proc call rules**