Jennifer's Environment Diagram Cheat Sheet

Assignment statements:

```
x, y = 2, 3
x, y = y, x
```

(This swaps x and y: x becomes 3 and y becomes 2)

1) Evaluate operands on RHS (from left to right)
2) Bind variables on LHS to values on RHS

Call Expressions:

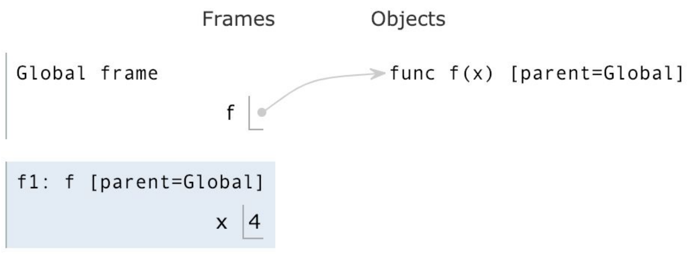
`f(lambda: x)`

(In this example, note that the lambda's parent is global!)

1) Evaluate operator
2) Evaluate operands (from left to right)
3) Apply the function to the arguments, opening a new frame

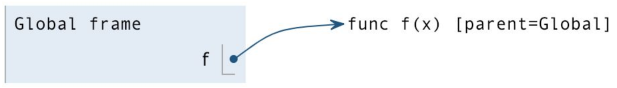
Opening a New Frame:

1) Frame number
2) Frame name
3) Frame's parent
4) Bind parameters to arguments
5) Start executing body of function



def Statements:

1) On RHS: write "func", function name (the intrinsic name), parameters, and parent frame (the frame in which that function is defined)
2) On LHS, bind function object to its name (the bound name)



Lambda Functions:

`lambda x, y: x + y`

Variables before colon are parameters; expression after colon is return value.

```
>>> lambda: 10        >>> (lambda: 10)()
    Function              10
```

Variable Lookup:

- Is variable in current frame?
  - If yes, return its value
  - If no, go to **parent** frame, and ask again (is variable in *this* frame?)
- Procedure repeats until we've found the variable or we hit the global frame. If we can't find the variable in the global frame, this will give an `Error`.

Things to pay attention to:

- Distinguish between functions vs. call expressions. `f` is a function, while `f()` is a call expression.
- When you define a function, do not look into its body (everything that's indented). Only look inside a function after you have called it!
- The parent of a frame is where it was **defined**, *NOT* where it was called! This is where the def statement appeared, or where the lambda expression was evaluated.
- Remember to evaluate all the operators and operands before opening a new frame for that function.
  - `f(lambda: x)`
  - In this example, note that the lambda's parent is global!
- If a function doesn't have a explicit return statement (it doesn't explicitly say `return ______`), then the return value will be `None` implicitly.
- Once we reach a return statement, we immediately exit from that function and don't execute any of the code that appears afterwards.
- **Sanity Checks:**
  - At the very end of your environment diagram, double check to make sure every frame except the global frame has a return value. (If not, something went wrong so backtrack!)
  - When we reach the return value for a frame, we're done with that frame, and we shouldn't be adding any more variable bindings within that frame.