The Third Sprint: Core Functionality

This sprint focuses on building the foundational logic and arithmetic operations. You'll need to define functions that can work with numerical **atoms**, which are a specific type of data structure you've likely created in a previous sprint.

Core Task List

1. **Number Functions:** Implement the following arithmetic and relational functions. These functions should take numerical atoms as arguments, perform the specified operation, and return the result. Return a new atom containing the result rather than a simple number. Use lowercase function names for consistency.
    - `add`: Adds two numbers.
    - `sub`: Subtracts one number from another.
    - `mul`: Multiplies two numbers.
    - `div`: Divides one number by another.
    - `mod`: Returns the remainder of a division.
    - `lt`: Checks if one number is less than another.
    - `gt`: Checks if one number is greater than another.
    - `lte`: Checks if one number is less than or equal to another.
    - `gte`: Checks if one number is greater than or equal to another.
2. **Equality and Logical Functions:**
    - `eq`: Define an equality function that compares two atoms of any type (not just numbers) and returns a boolean value (true or false).
    - `not`: Implement the logical `not` function. This should take a boolean atom and return the opposite value.
3. **Errors/Exceptions**:
    - If your function cannot return a value you don't have to crash.
    - DivisionByZero
    - "Not a number"
    - Consider returning a symbol or a string

Important: Do **not** implement `and` or `or` at this stage. We will address short-circuiting logic in a later sprint.

Testing

You should be able to test your new functionality programmatically.