Step 2: Choose Language Paradigm for "Pear"

Choosing the right programming paradigm(s) is a crucial step in defining the character and capabilities of the "Pear" programming language. Each paradigm brings its strengths and trade-offs, and the combination of paradigms can make "Pear" versatile and adaptable to different programming scenarios. Here are the main paradigms we can consider for "Pear":

1. **Imperative Paradigm:** The imperative paradigm focuses on describing the sequence of operations that a program should perform. It involves using statements and control structures like loops and conditionals to change program state. This paradigm is widely used in procedural programming.

2. **Object-Oriented Paradigm:** The object-oriented paradigm centres around organizing code into classes and objects, allowing encapsulation, inheritance, and polymorphism. It promotes code reusability and modularity.

3. **Functional Paradigm:** The functional paradigm treats computation as the evaluation of mathematical functions. It emphasizes immutability and avoids side effects, making it easier to reason about code and facilitate parallel processing.

4. **Declarative Paradigm:** Declarative programming focuses on specifying what the program should achieve, rather than how to achieve it. It includes languages like SQL for database queries.

Based on our objectives, we can consider a combination of paradigms to create a well-rounded language. For example:

Object-Oriented + Imperative + Functional: This combination allows for a balance between traditional imperative programming, the benefits of object-oriented design, and the simplicity and expressiveness of functional programming.

Imperative + Declarative: Combining these paradigms can be helpful for tasks where specifying the desired outcome in a declarative manner simplifies the code, but certain operations still require an imperative approach.

Ultimately, the choice of paradigms will heavily influence the design of the language and its syntax. We can leverage the best aspects of each paradigm to make "Pear" a powerful, flexible, and user-friendly language that suits a wide range of programming tasks.