Problem Statement for Programming Language Experimentation

Context: I am designing a programming language that allows any data type to be either pointed to or quoted. The goal is to enable flexible access to functions and objects through a loose call signature, defined by the user. This programming model experiments with the idea that both objects and functions share the same nature, and hence can be accessed and called interchangeably.

Problem Definition: In conventional programming languages, strict function signatures and object types limit flexibility and dynamic interactions between data types. There is a need to experiment with a model where functions and objects can be treated equivalently, allowing user-defined, loose call signatures to enhance flexibility and reduce constraints in programming.

Objective: To implement a programming language where data types must be either pointed to or quoted, facilitating dynamic access to objects and functions. The programming model will allow functions to be treated as objects and vice versa, enabling flexible and user-defined call signatures.

Example:

- `Signal` can be defined as both a function and an object.
- It can be accessed or invoked as either:
  - `Signal()` : As a function call.
  - `Signal == value` : As an object access.

This experimental approach aims to create a more dynamic and flexible programming paradigm where functions and objects are interchangeable in their usage.

Outcome: The experiment will provide insight into how loose call signatures can be implemented, fostering flexibility in function-object interactions, and offering a novel way to approach programming language design.