**Neura Design Plan – Version 1.0**

**Overview**

This document outlines the design plan for Neura, a general-purpose programming language with a focus on readability, simplicity, and futuristic features like native parallelism, event-driven programming, AI support, and interoperability. The core strengths are planned for future implementation, while the current focus is on building a solid foundation with unique syntax and an easy learning curve.

**Core Features**

**1. Here's an updated version of your Variable Declaration documentation with the additional rules you specified. Let me know if you want to refine it further!**

**Variable Declaration in Neura**

* **Syntax: variable\_name: type = value;  
  Type hints are mandatory to specify the data type for each variable.**

**Supported Types**

* **int — Integer values (e.g., 10, -5)**
* **float — Decimal numbers (e.g., 3.14)**
* **str — Strings (e.g., "John", "Hello, world!")**
* **bool — Boolean values (true, false)**
* **dynamic — Can hold any type of value (int, str, bool, etc.) and switch between them.**

**x: int = 10; # Integer variable**

**name: str = "John"; # String variable**

**is\_ready: bool = true; # Boolean variable**

**anything: dynamic = 10; # Dynamic variable**

**anything = "Now I'm a string!"; # Valid for dynamic type**

**2. Conditions**

* Use if, else if, and else blocks for conditional statements.
* Curly braces {} are mandatory for blocks.

*if (x > 5) {*

*say("x is greater than 5");*

*} else if (x == 5) {*

*say("x is equal to 5");*

*} else {*

*say("x is less than 5");*

*}*

**3. Loops**

**For Loop:**

* Uses a unique *for i in start..end by* step syntax.

*for i in 0..10 by 2 {*

*say("Iteration:", i);*

}

**Foreach Loop:**

* Iterate over collections like arrays or lists.

*fruits = ["apple", "banana", "cherry"];*

*foreach fruit in fruits {*

*say("Fruit:", fruit);*

*}*

**4. Functions**

Functions can be defined using fn.

**Standard Function:**

*fn greet(name) {*

*say("Hello", name);*

*}*

**Shorthand Single-Line Function:**

*fn add(a, b) => a + b;*

**5. Input and Output**

**ask(): For user input with type conversion and default values.**

*name = ask("Enter your name:").default("User");*

*age = ask("Enter your age:").asInt();*

***say(): For printing output.***

*say("Hello", name, "You are", age, "years old.");*

**6. String Manipulation and Interpolation**

**Interpolation**

Embed variables directly inside strings using ${}.

*name = "Deekshith";*

*say("Hello, ${name}!"); // Output: Hello, Deekshith!*

**String Methods**

*text = " Hello World! ";*

*trimmed = text.trim(); // Removes leading and trailing spaces*

*upper = text.uppercase(); // Converts to uppercase: "HELLO WORLD!"*

*lower = text.lowercase(); // Converts to lowercase: "hello world!"*

*length = text.length(); // Returns the length of the string*

**7. Type Conversion**

* **ask()** returns input as a string by default. You can convert it to other types.
* Conversion methods: *asInt(), asFloat(), asBool(), asString().*

**Type Conversion Examples**

*age = ask("Enter your age:").asInt();*

*height = ask("Enter your height:").asFloat();*

*status = ask("Are you ready? (0/1)").asBool();*

* **asBool():** Accepts only 0 or 1. Any other input throws an error.
* **.default(value)**: Sets a default value if the input is empty.

**8. Utility Functions**

**wait(seconds)**

Pauses execution for the specified number of seconds.

*wait(5);*

**notify(message, urgency)**

Sends system notifications. Supports urgency levels: low, normal, high.

*notify("Task completed!", urgency: "high");*

**9. Future Core Strengths (Planned for Later Versions)**

These features are in the design plan for future implementation and are currently represented by sample syntax.

**Native Parallelism and Easy Concurrency**

*parallel {*

*download("file1");*

*compress("file2");*

*analyze("file3");*

*}*

**Event-Driven Programming**

*on("data\_received") {*

*say("Data received successfully!");*

*}*

**AI Support**

*model = ai.load\_model("image\_classifier");*

*result = model.predict(image);*

**Interoperability**

*python {*

*import numpy as np*

*data = np.array([1, 2, 3]);*

*}*

**Conclusion**

The Neura aims to combine simplicity with power, providing developers with an intuitive syntax while offering unique strengths like parallelism, concurrency, and future AI integration. The initial focus is to ensure a stable core, with advanced features added iteratively.