**Introduction to Pebble Language**

**What is Pebble?**  
Pebble is a beginner-friendly, expressive programming language designed to blend simplicity with powerful features. It supports modern syntax styles, including let, const, arrow functions, keyword-style conditionals, and built-in data structures.

**File Extension**

Pebble source code files use the .peb extension.  
**Example:** hello.peb

**How to Run a Pebble Program**

**Prerequisites:** Make sure the Pebble compiler/interpreter is built using flex, bison, and a C backend.

To run a .peb program:

./run hello.peb

This command will compile and execute the Pebble code in hello.peb.

**Sample Program**

**Code in hello.peb:**

var name = "Pebble";

print("Hello from " + name + "!");

**Expected Output:**

Hello from Pebble!

**1. Literals**

Literals represent fixed values that appear directly in code. Pebble supports various literal types for numbers, text, characters, and logical values.

**🔸 Integer Literal**

**Description:** Represents whole numbers without a decimal point.  
**Syntax Example:**

var age = 42

***Explanation****:* Here 42 is an integer literal assigned to the variable age.

**🔸 Float Literal**

**Description:** Represents numbers with a decimal point.  
**Syntax Example:**

var pi = 3.14

***Explanation****:* 3.14 is a floating-point literal (decimal number).

**🔸 String Literal**

**Description:** Represents sequences of characters, enclosed in double quotes (").  
**Syntax Example:**

var greeting = "Hello, World!";

***Explanation****:* "Hello, World!" is a string literal.

**🔸 Char Literal**

**Description:** Represents a single character, enclosed in single quotes ('). Escape sequences like \n are allowed.  
**Syntax Example:**

var newline = '\n';

***Explanation****:* '\n' is a character literal for newline.

**🔸 Boolean Literal**

**Description:** Represents logical true or false.  
**Syntax Example:**

var isReady = true;

***Explanation****:* true is a boolean literal representing truth.

**2. Comparison Operators**

Comparison operators in Pebble are used to compare two values. They return a boolean result (true or false) based on the comparison. Pebble supports both **symbolic** (e.g., ==, !=) and **keyword-style** (e.g., equal, not equal) syntax for readability and flexibility.

**◆ Symbolic Operators**

These use traditional programming symbols (==, !=, >, etc.) to compare values. They are concise and commonly used in most programming languages.

**🔸 == ( *Equal To)***

**Description:**  
Checks if two values are equal. Returns true if they are the same.

**Example:**

var a = 5;

var b = 5;

if (a == b) {

display("Both values are equal");

}

**🔸 != ( *Not Equal To)***

**Description:**  
Checks if two values are **not** equal.

**Example:**

var a = 10;

var b = 20;

if (a != b) {

display("Values are not equal");

}

**🔸 > (*Greater Than)***

**Description:**  
Returns true if the left value is greater than the right one.

**Example:**

if (score > 90) {

display("Excellent!");

}

**🔸 < ( *Less Than)***

**Description:**  
Returns true if the left value is less than the right one.

**Example:**

if (marks < 35) {

display("Fail");

}

**🔸 >= ( *Greater Than or Equal To)***

**Description:**  
Checks if the left value is greater than or equal to the right one.

**Example:**

if (age >= 18) {

display("You can vote");

}

**🔸 <= ( *Less Than or Equal To)***

**Description:**  
Checks if the left value is less than or equal to the right one.

**Example:**

if (temperature <= 0) {

display("Freezing point reached");

}

**◆ Keyword-style Operators**

These use natural language words (equal, not equal, greater, etc.) to compare values. This style improves code readability, especially for beginners or in educational contexts.

**🔸 equal (*Equal To)***

**Description:**  
Returns true if two values are equal.

**Example:**

if (a equal b) {

display("They are equal");

}

**🔸 not equal (*Not Equal To***

**Description:**  
Returns true if two values are different.

**Example:**

if (x not equal y) {

display("They are different");

}

**🔸 greater ( *Greater Than)***

**Description:**  
Returns true if the left-hand value is greater than the right-hand value.

**Example:**

if (score greater 80) {

display("High score!");

}

**🔸 smaller (*Less Than)***

**Description:**  
Returns true if the left-hand value is less than the right-hand value.

**Example:**

if (marks smaller 35) {

display("Fail");

}

**🔸 greater equal (*Greater Than or Equal To)***

**Description:**  
Returns true if the left-hand value is greater than or equal to the right-hand value.

**Example:**

if (age greater equal 18) {

display("You can vote");

}

**🔸 smaller equal ( *Less Than or Equal To)***

**Description:**  
Returns true if the left-hand value is less than or equal to the right-hand value.

**Example:**

if (temp smaller equal 0) {

display("Freezing!");

}