
🌟 Custom Human-Readable Programming Language

A clean, natural-language-inspired language designed for readability and clarity, resembling English prose.

🧠 Language Philosophy

- Use **natural English syntax** like let, be, is greater than, define function, etc.
- Make programs feel like **pseudocode**, but executable.
- Encourage clarity, readability, and accessibility for non-programmers.

✅ Core Language Features

Category	**Feature Description**
✅ Variables	English-like declaration: let x be 5;
✅ Arithmetic	Supports +, -, *, /
✅ Strings	Double-quoted string support with escape sequences: "Hello\nWorld"
✅ Conditionals	if, then, else, with rich comparisons like is greater than
✅ Loops	repeat X times, repeat until, while loops
✅ Functions	define function NAME with PARAMS, returns TYPE, end function
✅ Boolean logic	and, or, not with true, false literals
✅ Input/Output	ask, print, run
✅ Type system	Basic types: number, text, boolean
✅ Pattern matching	match … case … endmatch for conditional branching
✅ Error handling	try … catch … endtry
✅ Comments	Inline: # comment

Keywords

Keyword	Purpose	Token
let	Declare variable	T_LET
be	Assign value	T_BE
set	Alternative assignment	T_SET
is	Used in comparisons	T_IS
null	Null literal	T_NULL
if, then, else	Conditionals	T_IF, T_THEN, T_ELSE
while, repeat, until, do	Loops	T_WHILE, T_REPEAT, etc.
define, function, end, with, returns	Functions	T_DEFINE, T_FUNCTION, T_END, etc.
return, give	Return from function	T_RETURN
and, or, not	Logical operators	T_AND, T_OR, T_NOT
true, false	Boolean literals	T_TRUE, T_FALSE
ask, print, run	Input/output/system calls	T_ASK, T_PRINT, T_RUN
match, case, endmatch	Pattern matching	T_MATCH, T_CASE, T_END_MATCH
try, catch, endtry	Error handling	T_TRY, T_CATCH, T_END_TRY
into	For ask ... into ...	T_INTO
number, text, boolean	Types	T_TYPE_NUM, T_TYPE_TEXT, T_TYPE_BOOL

🔎 Comparison Phrases

These are **multi-word tokens** Lex recognizes as single units:

Phrase	**Meaning**	**Token**
is equal to	Equality	T_EQ
is not	Inequality	T_NEQ
is greater than	Greater than	T_GT
is less than	Less than	T_LT

🧱 Symbols & Punctuation

Symbol	**Meaning**	**Token**
+	Addition	T_PLUS
-	Subtraction	T_MINUS
*	Multiplication	T_MUL
/	Division	T_DIV
(,)	Grouping	T_LPAREN, T_RPAREN
{, }	Code blocks	T_LBRACE, T_RBRACE
[,]	List indexing	T_LBRACKET, T_RBRACKET
;	Statement terminator	T_SEMI
,	Separator	T_COMMA
:	Used in match/case, objects	T_COLON

🔢 Literals and Identifiers

Type	Example	Token
String	"Hello"	T_STRING
Integer	42	T_NUM
Real number	3.14	T_RNUM
Identifier	myVar, age	T_ID

🧾 Comments

- Any line starting with # is ignored
- Example:
- let name be "Alice"; # This is a comment

🧪 Sample Program

```
define function greet with name
  let msg be "Hello, ";
  if name is not null then
    print msg;
    print name;
  else
    print "Guest";
end function
```
