Grammar

// STATEMENTS
STATEMENT -> SIMPLE_STMT | SELECTION_STMT | STATEMENT

SIMPLE_STMT -> ASSIGN | DECLARE

SELECTION_STMT -> IF_STMT | ELIF_STMT | ELSE_STMT | WHILE_STMT

COMPOUND_STMT -> OPEN_SCOPE CLOSE_SCOPE
| OPEN_SCOPE STATEMENT CLOSE_SCOPE

/// OPERATORS
ARITHMETIC_OP -> PLUS | MINUS | DIVISION | MULTIPLY
COMPARISON_OP -> DIFFERENT | EQUALS | GREATER | GREATER_EQUALS | LESS | LESS_EQUALS
LOGICAL_OP -> AND | OR

/// EXPRESSIONS
ARITHMETIC_EXPR -> ARITHMETIC _TERM
| ARITHMETIC_TERM PLUS ARITHMETIC _EXPR
| ARITHMETIC_TERM MINUS ARITHMETIC _EXPR
ARITHMETIC_TERM -> ARITHMETIC _FACTOR
| ARITHMETIC _FACTOR MULT ARITHMETIC _TERM
| ARITHMETIC _FACTOR DIVIDE ARITHMETIC _TERM
ARITHMETIC_FACTOR -> NUMBER_LITERAL
| IDENTIFIER
| L_PAREN ARITHMETIC _EXPR R_PAREN

COMPARISON_EXPR -> COMPARISON _TERM COMPARISON_OP COMPARISON_TERM
COMPARISON_TERM -> LITERAL
| IDENTIFIER
| ARITHMETIC_EXPR
| L_PAREN COMPARISON_TERM R_PAREN

LOGICAL_EXPR -> LOGICAL_TERM
| LOGICAL_EXPR OR LOGICAL_TERM
LOGICAL_TERM -> LOGICAL_FACTOR
| LOGICAL_TERM AND LOGICAL_TERM
LOGICAL_FACTOR -> IDENTIFIER
| LITERAL
| EXPR
| L_PAREN LOGICAL_EXPR R_PAREN

CONDITIONAL_EXPR -> EXPR | LITERAL | IDENTIFIER

EXPR -> ARITHMETIC_EXPR | LOGICAL_EXPR | COMPARISON_EXPR

/// GENERAL
LITERAL -> BOOLEAN_LITERAL | NUMBER_LITERAL | STRING_LITERAL

VAR_TYPE -> TYPE_BOOLEAN | TYPE_NUMBER | TYPE_STRING

ASSIGN -> VAR_TYPE IDENTIFIER VAR_ASSIGN_OPERATOR (LITERAL | IDENTIFIER | EXPR) ENDCOMMAND
| IDENTIFIER VAR_ASSIGN_OPERATOR (LITERAL | IDENTIFIER | EXPR) ENDCOMMAND

DECLARE -> VAR_TYPE IDENTIFIER ENDCOMMAND

/// CONDITIONALS
IF_STMT -> IF CONDITIONAL_EXPR COMPOUND_STMT

ELIF_STMT -> ELIF CONDITIONAL_EXPR COMPOUND_STMT

ELSE_STMT -> ELSE COMPOUND_STMT

/// LOOPS
WHILE_STMT -> WHILE CONDITIONAL_EXPR COMPOUND_STMT