Design of KiddoLang Language

KiddoLang is a simple, beginner-friendly programming language built to help children and new learners understand the core concepts of programming. Its syntax is based on plain English keywords, making it easy to read, write, and understand without prior coding experience.

Instructions to Write KiddoLang Programs

- Every command must end with a semicolon (;)
- Variable names must:
    - Start with an alphabet (a-z, A-Z)
    - May contain letters, numbers, and underscores
    - Cannot contain any special characters (except _)
    - Strings must be enclosed in double quotes ("...")
- There is no need for variable declarations; variables are dynamically created upon assignment.
- Variables can hold numeric values (int/float), strings, or booleans (yes / no)
- Expressions support both arithmetic and boolean logic.
- You can group expressions using parentheses (...)
- Supports nested for loops
- String assignments are allowed but string operations are not supported.
- Automatically skips any comments or empty lines in the program
- Comments should start with "//" and will be considered until the end of the line.

All program logic must be expressed using the supported statements and constructs shown below.

1. Assignment

```
set age to 10;
set name to "Kiddo";
set isHappy to yes;
```

2. Print Statement

```
say age;
say name;
say "Hello, world!";
```

3. Arithmetic Expressions

```
set result to (5 + 3) * 2;
say result;
```

4. Boolean Expressions

```
set p to yes;
set q to no;
say p and q;
say not q;
```

5. Relational Operators

```
set x to 10;
say x > 5;
say x == 10;
```

6. If-Else Statement

```
set score to 85;

when (score > 60) {
  say "You passed!";
} otherwise {
  say "Try again.";
}
```

7. Ternary Operator

```
set grade to score > 90 ? "A" : "B";
say grade;
```

8. For Loop

```
count from 1 to 5 {
  say "Hello!";
}
```

9. While Loop

```
set x to 0;
repeat until (x > 3) {
  say x;
  set x to x + 1;
}
```

Below is our grammar for Parser and Lexer(used EBNF):

```
grammar KiddoLang;

// ======================
//    Parser Rules
// ======================

program       : statement+ ;

statement
   : assignment
   | printStatement
   | ifStatement
   | loopStatement
   | ternaryExpr ';'
   ;
```

```
assignment     : SET ID TO expr SEMI ;

printStatement  : SAY expr SEMI ;

ifStatement    : WHEN LPAREN expr RPAREN block (OTHERWISE block)? ;

loopStatement   : forLoop | whileLoop ;

forLoop        : COUNT FROM expr TO expr block ;

whileLoop      : REPEAT UNTIL LPAREN expr RPAREN block ;

ternaryExpr    : expr QMARK expr COLON expr ;

block          : LBRACE statement+ RBRACE ;

expr
   : expr MULT expr
   | expr DIV expr
   | expr PLUS expr
   | expr MINUS expr
   | expr GT expr
   | expr LT expr
   | expr EQ expr
   | expr AND expr
   | expr OR expr
   | NOT expr
   | LPAREN expr RPAREN
   | ID
   | INT
   | FLOAT
   | YES
   | NO
   | STRING
   ;

// =====================
//    Lexer Rules
```

```
// ======================

// --- Keywords ---
SET          : 'set' ;
TO           : 'to' ;
SAY          : 'say' ;
WHEN         : 'when' ;
OTHERWISE    : 'otherwise' ;
COUNT        : 'count' ;
FROM         : 'from' ;
REPEAT       : 'repeat' ;
UNTIL        : 'until' ;
AND          : 'and' ;
OR           : 'or' ;
NOT          : 'not' ;
YES          : 'yes' ;
NO           : 'no' ;

// --- Operators & Punctuation ---
PLUS         : '+' ;
MINUS        : '-' ;
MULT         : '*' ;
DIV          : '/' ;
GT           : '>' ;
LT           : '<' ;
EQ           : '==' ;
QMARK        : '?' ;
COLON        : ':' ;
LPAREN       : '(' ;
RPAREN       : ')' ;
LBRACE       : '{' ;
RBRACE       : '}' ;
SEMI         : ';' ;

// --- Identifiers & Literals ---
ID           : [a-zA-Z_][a-zA-Z0-9_]* ;
INT          : [0-9]+ ;
FLOAT        : [0-9]+ '.' [0-9]+ ;
```

```
STRING        : '"' (~["\\] | '\\' .)* '"' ;

// --- Whitespace & Comments ---
WS            : [ \t\r\n]+ -> skip ;
COMMENT       : '//' ~[\r\n]* -> skip ;
```