1. Purpose

Noggin is a custom language, and this document will help explain the lexical grammar rules used in Noggin, specifically how source code is broken into tokens.

2. Character Set

Acceptable characters in Noggin will align with python's acceptable list which is that of ASCII / Unicode characters that include:

- Alphabets: All capital (A-Z) and small (a-z) alphabets.
- Digits: All digits 0-9.
- Special Symbols: Python supports all kind of special symbols like, " ' l ; : ! ~ @ # $ % ^ ` & * ( ) _ + - = { } [ ] \ .
- White Spaces: White spaces like tab space, blank space, newline, and carriage return.
- Other: All ASCII and UNICODE characters are supported by Python that constitutes the Python character set.

3. Whitespace and Comments

- New lines are visually impacting only, everything will be separated by semicolons (except one-line comments)
- Skip all whitespace when creating tokens
- Comments will start with // and end after a new line, or start with /* and end with */

4. Token Types

Punctuation / Delimiters

- LEFT_PAREN
- RIGHT_PAREN
- LEFT_BRACE
- RIGHT_BRACE
- COMMA
- DOT
- EOF
- SEMICOLON

Operators

- MINUS
- PLUS

- SLASH
- STAR
- BANG
- BANG_EQUAL
- EQUAL
- EQUAL_EQUAL
- GREATER
- GREATER_EQUAL
- LESS
- LESS_EQUAL

Literals

- IDENTIFIER
- STRING
- NUMBER

Keywords

- AND
- CLASS
- DEF
- ELSE
- FALSE
- FOR
- IF
- NULL
- OTHER
- OR
- PRINT
- RETURN
- SUPER
- THIS
- TRUE
- VAR
- WHILE

5. Keywords

Keywords are case-sensitive, all lowercase:

- and
- class
- def

- else
- false
- for
- if
- null
- or
- print
- return
- super
- this
- true
- var
- while

6. Identifiers

- Identifiers cannot perfectly match keywords
- Identifiers must start with a letter or underscore, can include digits after the first character

7. Literals

Literal types:

- Integer literals
    - 10, 5, 25000
- Floating-point literals
    - 10.0, 688.992, 80.11102
- String literals (both “ “ and ‘ ‘)
    - “Hello world”, ‘Hi gang’
- Boolean (binary value)
    - true, false
- Null
    - No value

8. Operators

Operator : token_type

- - : MINUS
- + : PLUS
- / : SLASH
- * : STAR

- ! : BANG
- != : BANG_EQUAL
- = : EQUAL
- == : EQUAL_EQUAL
- > : GREATER
- >= : GREATER_EQUAL
- < : LESS
- <= : LESS_EQUAL

9. Punctuation/Delimiters

Punctuation/Delimiter : token_type

- (: LEFT_PAREN
-) : RIGHT_PAREN
- { : LEFT_BRACE
- } : RIGHT_BRACE
- , COMMA
- . DOT
- ; SEMICOLON

10. Error Handling

Noggin will print an error statement and quit the program when encountering a lexical error during scanning.

Including but not limited to:

- Unexpected Character
- Unterminated String