```
Alphabet
a-z
A-Z
0-9
-

Lexic

operators: +, -, *, /, %, ==, <, <=, >, >=, =, &&, ||
separators: {}, (), ;, space, newline, ", ', :, ., ","(actual comma)
reserved words: let, if, else, while,  print, readI32, readU32, readStr, readBool, readF32,

identifier = (letter|"_"){letter|digit|"_"}
letter = "A"|"B"|"C"|..."Z"|"a"|"b"|...|"z"
digit = "0"|"1"|"2"|...|"9"

int_constant = ["+"|"-"] non_zero_digit {digit} | "0"
non_zero_digit = "1"|"2"|..|"9"
string_constant = """{letter|digit|"_"|" "}"""
bool_constant = "true"|"false"
float_constant = int_constant ["." digit{digit}]

int_array_constant = "[" [{int_constant","}int_constant] "]"
string_array_constant = "[" [{string_constant","}string_constant] "]"
bool_array_constant = "[" [{bool_constant","}bool_constant] "]"
float_array_constant = "[" [{float_constant","}float_constant] "]"
```

Tokens

```
+
-
*
/
%
==
<
<=
>
>=
=
&&
||
{
}
(
)
;
```

```
space
newline
,
'
:
.
,
let
if
else
while
print
readI32
readU32
readStr
readBool
readF32
i32
u32
str
bool
f32
array
```

Syntax

```
Program         ::= Statement{ Statement } ;

Statement       ::= DeclStatement
                | Assignment
                | Input
                | Output
                | Conditional
                | Loop
                | Comment ;

DeclStatement   ::= "let" Identifier ":" Type ["=" Expression] ";" ;

Type            ::= "i32"
                |   "u32"
                |   "bool"
                |   "f32"
                |   "str"
                |   "array["Type";" NonZeroDigit{Digit}"]" ;

Assignment      ::=  Identifier"=" Expression ";" ;
```

```
Input           ::= "readI32()"|"readU32()"|"readStr()"|"readBool()"|"readF32()";

Output          ::= "print" ( Expression{, Expression} ) ";" ;

Conditional     ::= "if" "(" Expression ")" "{" Program "}" [ "else" "{" Program "}" ] ;

Loop            ::= "while" "(" Expression ")" "{" Program "}" ;

Comment         ::= "//" { Any Character Except Newline } ;

Expression      ::= Term { Operator Term } ;
Term            ::= Identifier
                | IntConstant
                | StringLiteral
                | BoolConstant
                | FloatConstant
                | ArrayConstant
                | "("Expression")";

Identifier      ::= ("_" | Letter) {"_" | Letter | Digit};

IntConstant     ::= [ "+" | "-" ] ( NonZeroDigit { Digit } ) | "0" ;

StringLiteral   ::= """ { Any Character Except """ } """ ;

BoolConstant    ::= "true" | "false" ;

FloatConstant   ::= IntConstant [ "." { Digit } ] ;

ArrayConstant   ::= IntArrayConstant
                | StringArrayConstant
                | BoolArrayConstant
                | FloatArrayConstant ;

IntArrayConstant ::= "[" [ IntConstant { "," IntConstant } ] "]" ;

StringArrayConstant ::= "[" [ StringLiteral { "," StringLiteral } ] "]" ;

BoolArrayConstant ::= "[" [ BoolConstant { "," BoolConstant } ] "]" ;

FloatArrayConstant ::= "[" [ FloatConstant { "," FloatConstant } ] "]" ;

Operator        ::= "+" | "-" | "*" | "/" | "%" | "==" | "<" | "<=" | ">" | ">=" | "=" | "&&

Letter          ::= "A" | "B" | "C" | ... | "Z" | "a" | ... "z" ;
```

```
NonZeroDigit    ::= "1" | "2" | ... | "9" ;

Digit           ::= "0" | "1" | "2" | ... | "9" ;
```