Tokens:

+
-
*
/
>
>=
<
<=
!=
==
=
(
)
[
]
->
<-
"
space
newline
while
if
else
input
display
int
string
list
declare
,
;

Lexic:

Alphabet:
 - both uppercase and lowercase letters from the english alphabet (a-z A-Z)
 - all the digits (0-9)

Lexic:
 - operators: + - * / > >= < <= == != =
 - separators: () [] -> <- ; space newline , " '
 - reserved words:
  while for if else input display int string list declare
 - identifiers -> cannot start with a digit, cannot have any letters after digits
  - identifier::= letter {letter} {digit}
  - letter::= 'a' | 'b' | ... | 'z' | 'A' | 'B' | ... | 'Z'
  - digit::= '0' | '1' | ... | '9'
Constants:
 -int::='0' | ["+" | "-"]non_zero_digit{digit}
 -non_zero_digit::= '1' | '2' | ... | '9'

-char::="""letter | digit"""
-string::="""{char}"""

Syntax:

program = {statement_list}
list_declaration=("int" | "string") "list" "["nr"]"
type= "int" | "string" | list_declaration
declaration= "declare" type identifier {"=" (expression | array_expression)}
declaration_list = declaration | declaration "," declaration_list
relation= ">" | ">=" | "<" | "<=" | "!=" | "=="
condition= expression relation expression
operation = "+" | "-" | "*" | "/"
while_statement = "while" "(" condition ")" compound_statement
if_statement = "if" "(" condition ")" compound_statement
io_statement= ("INPUT" | "DISPLAY) "("identifier")"
int_expression = int | identifier | int_expression operation int_expression | "("int_expression operation int_expression ")"
string_expression = string | identifier | string_expression "+" string_expression
expression = int_expression | string_expression
array_expression = "[" {expression ","} expression"]"
assignment_statement = identifier "=" (expression | array_expression)
statement = (declaration|assignment_statement|if_statement|while_statement|return_statement)
statement_list = statement | statement statement_list
compound_statemet = "->" statement_list "<-"
return_statement = "return" expression