I. GRAMMAR OF NILANG

- Terminals in quotes or lower case.
  - ident, symbols
  - num, numbers
  - null, empty statement
  - expr, native julia expression.
  - expr::Int, native julia expression with integer return value.
  - expr::Bool, native julia expression with boolean return value.
- { }* indiceates zero or more repetitions.
- [ ] indicates zero or one repetitions.

```
Function ::= 'function' Fname '(' Fargs [',' ident '...']; Fargs ')' 'where' '{' Symbols '}'
                 Statements
              'end'

Fname ::= ident
          | '(' ident '::' ident ')'

Fargs ::= ident, Fargs
          | ident '::' ident, Fargs
          | ident '=' ident, Fargs
          | ident '::' ident '=' expr, Fargs
          | null

Symbols ::= ident, Symbols
          | ident


Statements ::= Ifstmt Statements
          | Whilestmt Statements
          | Forstmt Statements
          | Callstmt Statements
          | Instrstmt Statements
          | Revstmt Statements
          | @anc Statements
          | @routine Statements
          | @safe Statements
          | null

Ifstmt ::= 'if' '(' expr::Bool ',' expr::Bool ')'
                 Statements
              ['else'
                 Statements]
              'end'

Whilestmt ::= 'while' '(' expr::Bool ',' expr::Bool ')'
                 Statements
              'end'

Forstmt ::= 'for' ident '=' expr::Int[':' expr::Int] ':' expr::Int
                 Statements
              'end'

Callstmt ::= ident '(' Dataviews ')'

Instrstmt ::= ident '+=' ident ['(' Dataviews ')']
          | ident '-=' ident ['(' Dataviews ')']
          | ident '⊻=' ident ['(' Dataviews ')']
          | ident '.+=' ident ['.' '(' Dataviews ')']
          | ident '.-=' ident ['.' '(' Dataviews ')']
          | ident '.⊻=' ident ['.' '(' Dataviews ')']

Revstmt ::= '~' '(' Statements ')'

@routine ::= '@routine' ident 'begin'
                 Statements
              'end'
          | '@routine' ident

@anc ::= '@anc' ident '=' expr
          | '@deanc' ident '=' expr

@safe ::= '@safe' expr

Dataviews ::= Dataview, Dataviews
          | null

Dataview ::= Dataview '[' expr::Int ']'
          | ident {'.' ident}*
          | ident '(' Dataview ')'
          | Constant

Constant ::= num | 'π'
```