The keywords are: `array`, `assert`, `bool`, `else`, `false`, `float`, `fn`, `if`, `image`, `int`, `let`, `print`, `read`, `return`, `show`, `sum`, `then`, `time`, `to`, `true`, `type`, `write`.

Type Syntax

```
type : int
     | bool
     | float
     | <variable>
     | <type> [ , ... ]
     | { <type> , ... }
```

Expressions

```
expr : <integer>
     | <float>
     | true
     | false
     | <variable>
     | { <expr> , ... }         // Tuple literal
     | [ <expr> , ... ]         // Array Literal
     | ( <expr> )
     | <expr> + <expr>
     | <expr> - <expr>
     | <expr> * <expr>
     | <expr> / <expr>
     | <expr> % <expr>
     | - <expr>                 // unary negation
     | <expr> < <expr>
     | <expr> > <expr>
     | <expr> == <expr>
     | <expr> != <expr>
     | <expr> <= <expr>
     | <expr> >= <expr>
     | <expr> && <expr>
     | <expr> || <expr>
     | ! <expr>
     | <expr> { <integer> }     // Tuple Index
     | <expr> [ <expr> , ... ]  // Array index
     | if <expr> then <expr> else <expr>
     | array [ <variable> : <expr> , ... ] <expr>
     | sum [ <variable> : <expr> , ... ] <expr>
     | expr : <variable> ( <expr> , ... )
```

Statements (Like Commands but in functions only)

```
stmt : let <lvalue> = <expr>
     | assert <expr> , <string>
     | return <expr>
```

Commands

```
cmd  : read image <string> to <argument>
     | write image <expr> to <string>
     | type <variable> = <type>
     | let <lvalue> = <expr>
     | assert <expr> , <string>
     | print <string>
     | show <expr>
     | time <cmd>
     | fn <variable> ( <binding> , ... ) : <type> { ;
         <stmt> ; ... ;
       }
```

Arguments, Lvalues, and Bindings

```
argument : <variable>
         | <variable> [ <variable> , ... ]

lvalue : <argument>
       | { <lvalue> , ... }

binding : <argument> : <type>
        | { <binding> , ... }
```