SOS Style Guide

The SOS language has a very flexible syntax, which gives the programmer many options for enhancing readability. However, it also allows for quite monstrous code if used poorly. This style guide is meant to establish a standard for readable SOS.

1. Declarations

Include spaces before and after the `:`, `->`, and `=`. If the function is short, then it should be on the same line. If it would take more than one line, the expression should start on the next line and be indented by four spaces.

```
var : type = expr...
fxn : (type arg1, type arg2, ...) -> = short expr...
fxn : (type arg1, type arg2, ...) -> =
    long expr...
    more expr...
```

2. Expressions

Most of an expression should fit on a single line. If a expression that should be a single line is very long, any additional lines needed should be indented by four spaces. The expressions that do not require indenting are if/else expressions and sequencing.

For sequencing, two short expressions may share a line. In this case, the `;` should be preceded and followed by a space. Otherwise, the second expression should be on a new line at the same indentation, while the semicolon is at the end of the first expression.

For if/else, the then expression and the else expression may be on their own lines. If so, the `then` may share a line with `if` or the beginning of the then expression, and the `else` may share a line with the beginning of the else expression. The `if` should always share a line with the if expression.

```
expr ; expr
if expr then
longer expr
else longer expr
```

3. Spacing

Spaces between identifiers and keywords are encouraged in most cases. Using more than one space is generally undesirable unless this is used to align similar elements in different lines for readability. No space may be used for simple arithmetic. The main exceptions are struct and array access, function application, and unary operators, where the identifier and symbol should always be directly adjacent.

```
struct.field       // not struct . field
array[element]     // not array [ element ]
function(args...)  // not function ( args )
-number            // not - number
```

Statements should all be on separate lines, and should be grouped into related blocks of statements that are separated by a blank line. For instance, several associated variable definitions may be on adjacent lines, followed by a blank line.

4. Identifiers

Identifiers should be lowercase and use underscores to separate words. Camel case is discouraged, but can be used instead of underscores as long as it is used consistently.