Lua102 cheat sheet

Lexical conventions

- `-- foobar` : comment until end of line
- variable name: _, letters, numbers (cannot start with a number)
- reserved names: `and break do else elseif end false for function if in local nil not or repeat return then true until while`

Types

Type	Literal
number	`123 -2 45.024 9.9e6`
string	`"Foo" 'Bar'`
boolean	`true false`
nil	`nil`
array	`{}`

Operators

Number operator	Meaning
`a + b`	addition
`a - b`	subtraction
`a * b`	multiplication
`a / b`	division
`a % b`	modulo
`a ^ b`	exponentiation
`-a`	opposite

String operator	Meaning
`a .. b`	concatenation
`#a`	length

Comparison operator	Meaning
`a == b`	equal
`a ~= b`	different
`a < b`	strictly smaller
`a > b`	strictly greater
`a <= b`	smaller or equal
`a >= b`	greater or equal

Expressions

Expression	Example
literal	`1`
variable	`foo`
operation	`1 + foo`
(expr)	`(1 + foo)`
function call	`bar(4)`
array indexing	`arr[i]`

Statements

Statement	Meaning
`var = expression`	assign value of expression to variable `var`
`if` control structure	select block of code according to condition

Statement	Meaning
numeric `for` loop	repeat block of code
`while` loop	repeat block of code
`repeat` loop	repeat block of code
`break`	exit current loop
function call	call function, discard return values if any
`return`	exit function
`return expr`	exit function with return value
`local var`	declare local variable `var`
`local var = expr`	declare local variable `var` with initial value

block: sequence of statements

Control structures

```
-- execute block associated with the first expression which is true
-- if all expressions are false, execute elseblock

-- `elseif` blocks: 0 or more
-- `else` block: 0 or 1

if expression1 then
    -- block1
elseif expression2 then
    -- block2
elseif expression3 then
    -- block3
else
    -- elseblock
end
```

```
-- repeat block with variable taking values
-- start, start + step, start + 2*step, start + 3*step, etc.
-- until it goes past finish

-- `step` is optional and defaults to 1

for variable = start,finish,step do
    -- block
end
```

```
-- repeat block as long as condition is true

while expression do
    -- block
end

-- repeat block until condition is true

repeat
    -- block
until condition
```

Functions

```
-- definition
-- parameters: 0 or more, comma separated

function name(param1, param2, param3)
    -- block
end

-- call
-- arguments: 0 or more, comma separated

name(arg1, arg2, arg3)
```

Arrays

Expression	Meaning
`{}`	empty array literal
`{"a", true, 5}`	array with initial values literal
`array[index]`	array indexing (setting/getting)
`#array`	length of array

Array functions	Meaning
`table.insert(array, value)`	insert `value` at end of `array`
`table.insert(array, i, value)`	insert `value` at position `i` in `array`
`table.remove(array)`	remove last element of `array`

Array functions	Meaning
`table.remove(array, i)`	remove element at position `i` in `array`
`table.concat(array, sep)`	make string of array contents with separator `sep`