Lua Syntax Docs V1.0

The most simple Lua docs ever

Basic Lua

View or Print

Print

```
print("Hello World!")
```

> Hello World!

Semicolons

Semicolons are not required. But if you like, you may put it in the end of the line.

```
someFunc(STH)
```

or

```
someFunc(STH);
```

Comments

Anything after `--` is considered as comment.

So, you can write `----------` in your code instead of bad looking `//////////` . (Yay!)

```
-- Single line comment
--[[
  Multi-line
  comments
]]
```

Newline

```
io.write("string", "\n")
io.write("string\n")
io.write("hi")
io.write("hello")
```

> string
>
> string
>
> hihello

Backslash (`\`) examples

`\n` : Newline
`\b` : Backspace
`\t` : tab
`\\` : Prints out `\`
`\"` : Prints out `"`
`\'` : Prints out `'`

Print long strings with newlines and whitespace

```
longString = [[
  am the super
  mega long string
  you have ever seen in your
  life.
]]

io.write(longString, "\n")
```

> am the super
> mega long string
> you have ever seen in your
> life.

Combine strings

```
name = "I "
longString = [[
  am the super
  mega long string
  you have ever seen in your
  life.
]]
longString = name .. longString

io.write(longString, "\n")
```

> I am the super
> mega long string
> you have ever seen in your
> life.

Use the operator `..` to combine strings.

Simple Lua Functions

Variables

No need to put `int`, `float`, etc. Lua doesn't have those. This example automatically makes `name` a String.

```
name = "cozyplanes"
```

or

```
name = 'cozyplanes'
```

Size of string

```
name = "cozyplanes"
io.write("Size of string: ", #name, "\n")
```

> Size of string: 10

`#VARIABLE_NAME` returns the size of string.

Change assigned value during code

```
name = "cozyplanes"
io.write("Size of string: ", #name, "\n")

name = "Ian"
io.write("My name is ", name, "\n")
```

> Size of string: 10
> My name is Ian

Get variable type

```
someVar = 98212547682 + 1
io.write("Type of someVar: ", type(someVar), "\n")
```

> Type of someVar: number

Float precision

Float values are accurate for 13 digits under decimal

```
floatPrecision = 1.999999999999 + 0.0000000000005
io.write(floatPrecision)
```

> 1.9999999999995

After the 13th value, it gets deleted

```
floatPrecision = 1.999999999999 + 0.00000000000058
io.write(floatPrecision)
```

> 1.9999999999995

This example above lost 0.0000000000008

Errors in Lua

```
io.write(type(madeUpVar), "\n")
```

> nil

Math

Math operators and functions

Basic operators

```
io.write(5 + 3, "\n") -- Add
io.write(5 - 3, "\n") -- Subtract
io.write(5 * 3, "\n") -- Multiply
io.write(5 / 3, "\n") -- Divide
io.write(5 % 3, "\n") -- Remainder
```

> 8
>
> 2
>
> 15
>
> 1.6666666666667
>
> 2

Advanced functions

`floor`	`ceil`	`max`	`min`	`sin`	`cos`	`tan`	`asin`
`acos`	`exp`	`log`	`log10`	`pow`	`sqrt`	`random`	`randomseed`

Examples:

```lua
io.write(math.floor(2.345), "\n")
io.write(math.ceil(2.345), "\n")
io.write(math.max(2, 3), "\n")
io.write(math.min(2, 3), "\n")
io.write(math.pow(8, 2), "\n") -- 8^2
io.write(math.sqrt(64), "\n") -- root 64
io.write(math.random(), "\n") -- Any random number
io.write(math.random(10), "\n") -- 1 ~ 10
io.write(math.random(5, 100), "\n") -- 5 ~ 100
```

> 2
>
> 3
>
> 3
>
> 2
>
> 64
>
> 8
>
> 0.84018771676347 (random)
>
> 4 (random)
>
> 80 (random)

Increase variable value by n

In other languages,

```lua
number += 1
```

is the same as

```lua
number = number + 1
```

However, in Lua, it only supports

```lua
number = number + 1
```

Format strings

```lua
print(string.format("Pi = %.10f", math.pi))
```

> Pi = 3.1415926536

Conditionals

Operators

Relational Operators	`>`	`<`	`>=`	`<=`	`==`	`~!` (same as `!=`)
Logical Operators	`and`	`or`	`not`			

if / elseif with Logical Operator : and

```
age = 15

if age < 14 then
  io.write("You are probably a script kiddy", "\n")
elseif (age >= 14) and (age < 19) then
  io.write("You are probably a teenager", "\n")
else
  io.write("You are probably an adult", "\n")
end
```

> You are probably a teenager

if / elseif with Logical Operator : or

```
age = 10

if (age < 14) or (age > 67) then
  io.write("You shouldn't work", "\n")
end
```

> You shouldn't work

local variable

`local` variables are only valid inside a function that has the variable.

The `localVar` below is only valid inside of the `if` statement.

```
age = 15

if age < 19 then
  io.write("Not an adult", "\n")
  local localVar = 100
end

print(localVar)
```

> Not an adult
>
> nil

Format variable to another type

```
print(string.format("not true = %s", tostring(not true)))
```

> not true = false

Ternary Operator

```
myage = 15

canVote = myage > 18 and true or false
io.write("Can I vote? ", tostring(canVote), "\n")
```

> Can I vote? false

Difference with other languages

In other languages,

```
canVote = myage > 18 ? true : false
```

However, in Lua, ternary operator is `and` / `or` instead of `?` / `:`

Strings

The `quote` in the following examples is the following.

Declare quote like this in the top of code if you are following with the examples.

```
quote = "Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo."
```

Useful String Functions

Length

```
io.write("Quote Length : ", string.len(quote), "\n")
```

or

```
io.write("Quote Length : ", #quote, "\n")
```

Both results:

> Quote Length : 215

Replace

```
io.write("Replace error with success : ", string.gsub(quote, "error", "success"), "\n")
```

> Replace error with success : Sed ut perspiciatis unde omnis iste natus success sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Index

Each characters will get an index.

For example, `sed ut` in the quote, the index is like the following...

	`s`	`e`	`d`		`u`	`t`
index	0	1	2	3	4	5

You can search the index of the quote, like this example

```
io.write("Index of ut : ", string.find(quote, "ut"), "\n")
```

> Index of ut : 5

Uppercase

```
io.write("Quote to Uppercase : ", string.upper(quote), "\n")
```

> Quote to Uppercase : SED UT PERSPICIATIS UNDE OMNIS ISTE NATUS ERROR SIT VOLUPTATEM ACCUSANTIUM DOLOREMQUE LAUDANTIUM, TOTAM REM APERIAM, EAQUE IPSA QUAE AB ILLO INVENTORE VERITATIS ET QUASI ARCHITECTO BEATAE VITAE DICTA SUNT EXPLICABO.

Lowercase

```
io.write("Quote to Lowercase : ", string.lower(quote), "\n")
```

> Quote to Lowercase : sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Looping

while

```
i = 1

while (i <= 10) do
  io.write(i)
  i = i + 1
end
```

> 12345678910

Break during while loop

```
i = 1

while (i <= 10) do
  io.write(i)
  i = i + 1

  if i == 8 then
    break -- leaves the while loop when i variable becomes 8
  end
end -- ends with 7
```

> 1234567

repeat - until

`repeat` is the `do while` in other languages.

In the example below, it takes input until the user inputs 15.

When the user inputs 15, the program terminates.

```
repeat
  io.write("Enter your guess : ")
  guess = io.read() -- Take input from user, simple as that

until
  tonumber(guess) == 15 -- Convert guess to number
```

> (Varies by input)
>
> Example output:
>
> Enter your guess : 10
>
> Enter your guess : 17
>
> Enter your guess : 15

for

```
for i = 1, 10 , 1 do -- starting value / ending value / increment by n
  io.write(i)
end
```

> 12345678910

Cycle through values in Table (See Table Section)

```lua
months = {"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"}

for key, value in pairs(months) do
  io.write(value, " ")
end
```

> Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

Table

Tables is the concept of Arrays or Dictionaries in other programming languages.

One Dimensional Table

```lua
aTable = {}

for i = 1, 10 do
  aTable[i] = i -- Increment i and assign those values to i
       --|-- The very first i will have the index of 1
end

io.write("First : ", aTable[1], "\n") -- Searches i of the index of 1
```

> First : 1

Number of items

```lua
aTable = {}

for i = 1, 10 do
  aTable[i] = i
end

io.write("Number of items : ", #aTable, "\n")
```

> Number of items : 10

Insert a value in Table

```lua
aTable = {}

for i = 1, 10 do
  aTable[i] = i
end

table.insert(aTable, 1, 0) -- target table / index / value to be changed

io.write("Tenth (index) value : ", aTable[10], "\n")
```

> Tenth (index) value : 9

Convert a Table to a String

```
aTable = {}

for i = 1, 10 do
  aTable[i] = i
end

table.insert(aTable, 1, 0) -- table is the library

print(table.concat(aTable, ", ")) -- Use the function concat()
```

> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Remove a value from Table

```
aTable = {}

for i = 1, 10 do
  aTable[i] = i
end

table.insert(aTable, 1, 0) -- table is the library

table.remove(aTable, 1) -- target table / index
print(table.concat(aTable, ", "))
```

> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Multi Dimensional Table

```
aMultiTable = {}

for i = 0, 9 do
  aMultiTable[i] = {}
  for j = 0, 9 do
    aMultiTable[i][j] = tostring(i) .. tostring(j)
  end
end

io.write("Table[][] : ", aMultiTable[0][0]) -- Play with the values in aMultiTable[][]
io.write("Table[][] : ", aMultiTable[1][3])
```

> Table[][] : 00
>
> Table : 13

Print all values

```
aMultiTable = {}

for i = 0, 9 do
  aMultiTable[i] = {}
  for j = 0, 9 do
    io.write(aMultiTable[i][j], " : ")
  end
  print()
end
```

To be continued..... (`28:06`)

Misc.

License (Unlicense)

```
Copyright 2018 cozyplanes
All rights reserved.

This is free and unencumbered software released into the public domain.

Anyone is free to copy, modify, publish, use, compile, sell, or
distribute this software, either in source code form or as a compiled
binary, for any purpose, commercial or non-commercial, and by any
means.

In jurisdictions that recognize copyright laws, the author or authors
of this software dedicate any and all copyright interest in the
software to the public domain. We make this dedication for the benefit
of the public at large and to the detriment of our heirs and
successors. We intend this dedication to be an overt act of
relinquishment in perpetuity of all present and future rights to this
software under copyright law.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE SOFTWARE.

For more information, please refer to <http://unlicense.org>
```

Contact

If you have questions, don't hesitate to contact to me at cozyplanes@tuta.io

Hope this document helps to you. If you liked this, make sure to leave a star at

https://github.com/cozyplanes/LuaDocs