Lua 101: intro

- `print("Hello World") -- or print('Hello World')` ☺
- Types
  - Dynamically typed
  - nil, Booleans , Numbers, Strings, Tables, Functions, (userdata and threads)
    - If it is not **false** or **nil** then it is true!!!
  - `type(foo)` will return the type name
- Variables
  - Local (to this execution): `local foo = 'foo'`
  - Global (persisted through different executions) `bar = 'bar'`
    - If not explicitly specified, variables are global
  - To delete a variable, set it to nil: `bar = nil; foo = nil`
  - Comments:
    - Inline comments:
      local foo = 3 *-- this is a comment*
    - Block comments
      --[[I am a block
      comment]]
- Tables:
  - associative arrays (array that can be indexed not only by numbers)
  - Index starts at 1
  - # returns the last index (or the size) of an array: `print(#aTable)`

initialization	example	result
`a = {} -- empty table`	`print(a)`	`table: 0x806f048`
`a = {x = 3, y = 1}`	`print(a.x)` `print(a['x'])`	3 3
`a = { foo = {x=12}, bar = {} }`	`print(a['foo']['x'])` `print(a.foo.x)`	12 12

Lua 101: control structures

if then else

```lua
if a == 3 then
  print('three')
else
  print('no!')
end
```

while

```lua
local a = 1
while a < 10 do
  a = a + 1
  print(a)
end
```

repeat

```lua
local a = 1
repeat
  a = a + 1
until a > 10
print(a)
```

generic for

```lua
local days = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"}
for i,v in ipairs(days) do
  print(i .. " - " .. v)
end
-- i, v  out of scope now
```

result

```
1 - Sun
2 - Mon
3 - Tue
4 - Wed
5 - Thu
6 - Fri
7 - Sat
```

```lua
local days = {Sun = 1 , Mon = 2, Tue = 3 , Wed = 4, Thu = 5,
Fri = 6, Sat = 7}
for key,v in pairs(days) do
  print(key .. " - " .. v)
end
-- key, v  out of scope now
```

```
Sun - 1
Wed - 4
Sat - 7
Tue - 3
Thu - 5
Fri - 6
Mon - 2
```

numeric for

```lua
for i=1,3 do
  print(i)
end
```

```lua
for i=1,3,2 do -- step of 2
  print(i)
end
```

result

```
1
2
3
```

```
1
3
```

Lua 101: functions

code	result

`function sum(a,b) --simple function`
`  return a+b`
`end`
`print(sum(5,6))`
	`11`

`function sum_and_sub(a,b) -- multiple returns`
`  local sum = a+b`
`  local diff = a-b`
`  return sum,diff`
`end`
`local sum,diff = sum_and_sub(5,6)`
`print("sum is " .. sum .. " diff is " .. diff)`
	`sum is 11 diff is -1`

`function sum(a,b)`
`  return a+b`
`end`
`function sub(a,b)`
`  return a-b`
`end`
`function eval(f,a,b)`
`  return f(a,b)`
`end`
`local f = sum -- functions are first class citizen`
`print(eval(f,5,6))`
`f = sub`
`print(eval(f,5,6))`
	`11`
	`-1`

Lua 101: easy integration with c

```c
#include <lua.h>
#include <lauxlib.h>
#include <lualib.h>
#include <stdio.h>
#include <stdlib.h>

static int add(lua_State *L) {
    double a=0,b=0;
    printf("I am c\n");
    a = lua_tonumber(L,-1);
    b = lua_tonumber(L,-2);
    lua_pushnumber(L, (a+b));
    return 1;
}

int luaopen_add(lua_State *L){
        lua_register(L,     /* Lua state variable */
                    "add", /* func name as known in Lua */
                    add);  /* func name in this file */

        return 0;
}
```

```
gcc -Wall -shared -fPIC -o add.so -I/usr/include/lua5.2 -llua add.c
```

```
require('add')
local sum = add(5,6)
print(sum)
```

```
I am c
11
```