Tham Khảo Nhanh Lua

Bảng, hàm, metatables, coroutines, modules

Cơ Bản

Kiểu Dữ Liệu & Biến

```
local x = 42          -- number
local s = "hello"     -- string
local b = true        -- boolean
local n = nil         -- nil
local t = {}          -- table
```

Kiểu Dữ Liệu

nil	Giá trị vắng mặt
boolean	true hoặc false
number	Số nguyên và số thực (float 64-bit)
string	Chuỗi bất biến (UTF-8)
table	Cấu trúc dữ liệu duy nhất (mảng + dict)
function	First-class function
userdata	Dữ liệu C tùy chỉnh
thread	Coroutine

Toán Tử

+ - * / // %	Số học (// là chia nguyên)
^	Lũy thừa
== ~= < > <= >=	So sánh (~= là không bằng)
and or not	Logic
..	Nối chuỗi
#	Độ dài chuỗi/mảng

Luồng Điều Khiển

If / Else

```
if x > 0 then
    print("positive")
elseif x == 0 then
    print("zero")
else
    print("negative")
end
```

Vòng Lặp

```
for i = 1, 10 do print(i) end
for i = 10, 1, -1 do print(i) end   -- step -1

while x > 0 do x = x - 1 end

repeat
    x = x + 1
until x >= 10
```

Vòng Lặp Qua Bảng

```
-- ipairs: indexed (array part)
for i, v in ipairs(t) do print(i, v) end

-- pairs: all keys
for k, v in pairs(t) do print(k, v) end
```

Hàm

Định Nghĩa Hàm

```
function greet(name)
    return "Hello, " .. name
end

local square = function(x) return x * x end

-- Multiple return values
function minmax(a, b)
    return math.min(a,b), math.max(a,b)
end
local lo, hi = minmax(3, 7)
```

Varargs & Closures

```
function sum(...)
    local total = 0
    for _, v in ipairs({...}) do total = total + v end
    return total
end

function counter(start)
    local n = start
    return function() n = n + 1; return n end
end
local next = counter(0)
next()  -- 1, next() -- 2
```

Bảng (Tables)

Mảng

```
local a = {10, 20, 30}   -- 1-indexed!
print(a[1])              -- 10
a[4] = 40
print(#a)                -- 4
```

Ánh Xạ (Dictionary)

```
local person = { name = "Alice", age = 30 }
print(person.name)       -- field access
print(person["age"])     -- bracket access
person.city = "NYC"      -- add field
```

Phương Thức Bảng Chuẩn

table.insert(t, v)	Thêm v vào cuối bảng
table.insert(t, i, v)	Chèn v tại vị trí i
table.remove(t, i)	Xóa và trả về phần tử tại i
table.concat(t, sep)	Nối tất cả thành chuỗi
table.sort(t)	Sắp xếp in-place
table.unpack(t)	Trả về phần tử như nhiều giá trị

Strings

Hàm String

string.len(s) / #s	Độ dài chuỗi
string.upper(s) / lower(s)	Chuyển đổi hoa/thường
string.rep(s, n)	Lặp lại chuỗi n lần
string.sub(s, i, j)	Cắt chuỗi (1-based)
string.find(s, pat)	Tìm pattern, trả về vị trí
string.match(s, pat)	Trích xuất match
string.gsub(s, pat, rep)	Thay thế toàn cục
string.format('%d %.2f', n, f)	Định dạng printf-style

Pattern Matching

```
local d, m, y = string.match("2024-03-15",
    "(%d+)-(%d+)-(%d+)")
-- %d digit, %a letter, %s space, %w word char
-- + one or more, * zero or more, ? optional
```

Metatables

Thiết Lập Metatable

```
local mt = {
    __index = function(t, k) return 0 end,
    __tostring = function(t)
        return "Vec(" .. t.x .. "," .. t.y .. ")"
    end,
    __add = function(a, b)
        return {x = a.x+b.x, y = a.y+b.y}
    end
}
local v = setmetatable({x=1, y=2}, mt)
```

Metamethods Phổ Biến

__index	Tra cứu key bị thiếu
__newindex	Gán key mới
__add / __sub / __mul	Toán tử số học
__eq / __lt / __le	So sánh
__tostring	Chuyển đổi chuỗi
__call	Gọi bảng như hàm
__len	Toán tử #

OOP với Metatables

```
Animal = {}
Animal.__index = Animal

function Animal.new(name, sound)
    return setmetatable({name=name, sound=sound}, Animal)
end

function Animal:speak()
    print(self.name .. " says " .. self.sound)
end

local cat = Animal.new("Cat", "meow")
cat:speak()
```

Coroutines

Coroutine Cơ Bản

```
local co = coroutine.create(function(a, b)
    print("start", a, b)
    local c = coroutine.yield(a + b)
    print("resumed", c)
end)

coroutine.resume(co, 10, 20)  -- start 10 20
coroutine.resume(co, 99)      -- resumed 99
```

Hàm Coroutine

coroutine.create(f)	Tạo coroutine mới
coroutine.resume(co, ...)	Tiếp tục thực thi
coroutine.yield(...)	Tạm dừng và trả giá trị
coroutine.status(co)	suspended/running/dead/normal
coroutine.wrap(f)	Trả về hàm resume tiện lợi

Modules

Tạo Module

```
-- mymodule.lua
local M = {}

function M.greet(name)
    return "Hello, " .. name
end

return M
```

Tham Khảo Nhanh Lua

Sử Dụng Module

```
local mod = require("mymodule")
print(mod.greet("Alice"))

-- Standard libraries:
local math = require("math")
local io   = require("io")
local os   = require("os")
```

Standard Libraries

`math`	Hàm toán học (sin, cos, floor, random...)
`string`	Thao tác chuỗi và pattern matching
`table`	Thao tác bảng (insert, remove, sort)
`io`	File I/O và console
`os`	Thời gian, biến môi trường, lệnh hệ thống
`math.random / randomseed`	Sinh số ngẫu nhiên