

REFERENSI CEPAT LUA

Tabel, fungsi, metatable, coroutine, modul, pola

Dasar

Hello World

```
print("Hello, Lua!")
```

Variabel & Penugasan

```
local name = "Lua" -- local variable
x = 10 -- global (avoid)
local a, b = 1, 2 -- multiple assignment
a, b = b, a -- swap values
```

Komentar

```
-- single line comment
--[[ multi-line
comment ]]
```

Operator

+ - * / % Operator aritmatika
// Pembagian lantai (5.3+)
^ Perpangkatan
.. Penggabungan string
Operator panjang
== ~= Sama / tidak sama
and or not Operator logika

Tipe

Tipe Data

nil Tidak ada nilai; palsu
boolean true atau false
number Float presisi ganda (atau integer di 5.3+)
string Urutan byte immutable
table Array asosiatif (satu-satunya tipe majemuk)
function Closure kelas pertama
userdata Data C yang dibungkus untuk Lua
thread Handle coroutine

Pemeriksaan Tipe

```
print(type(42)) -- "number"
print(type("hi")) -- "string"
print(type(nil)) -- "nil"
print(type({})) -- "table"
```

Tabel

Tabel Gaya Array

```
local fruits = {"apple", "banana", "cherry"}
print(fruits[1]) -- "apple" (1-indexed)
table.insert(fruits, "date")
table.remove(fruits, 2) -- remove "banana"
print(#fruits) -- length
```

Tabel Gaya Kamus

```
local user = {name = "Alice", age = 30}
user.email = "agb.com" -- add field
user["name"] = "Bob" -- bracket access
user.age = nil -- remove field
```

Fungsi Tabel

table.insert(t, v) Tambahkan nilai ke array
table.insert(t, i, v) Sisipkan di posisi i
table.remove(t, i) Hapus elemen di posisi i
table.sort(t [, cmp]) Urutkan array di tempat
table.concat(t, sep) Gabungkan elemen array menjadi string
table.move(t, a, b, c) Pindahkan elemen dari a..b ke posisi c

Fungsi

Definisi Fungsi

```
local function add(a, b)
  return a + b
end
local mul = function(a, b) return a * b end
print(add(2, 3)) -- 5
```

Variadic & Beberapa Nilai Kembali

```
local function sum(...)
  local s = 0
  for _, v in ipairs({...}) do s = s + v end
  return s
end
local function swap(a, b) return b, a end
local x, y = swap(1, 2)
```

Closure

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

Alur Kontrol

Kondisional

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

Perulangan

```
for i = 1, 10 do print(i) end
for i = 10, 1, -1 do print(i) end
for k, v in pairs(tbl) do print(k, v) end
for i, v in ipairs(arr) do print(i, v) end
```

While & Repeat

```
while x > 0 do x = x - 1 end
repeat
  x = x + 1
until x >= 10
```

String

Fungsi String

string.len(s) / #s Panjang string dalam byte
string.sub(s, i, j) Substring dari i ke j
string.upper(s) Konversi ke huruf kapital
string.lower(s) Konversi ke huruf kecil
string.rep(s, n) Ulangi string n kali
string.reverse(s) Balik string
string.format(fmt, ...) Pemformatan gaya printf
string.find(s, pat) Temukan pola, kembalikan indeks
string.gsub(s, pat, rep) Substitusi global
string.gmatch(s, pat) Iterator atas kecocokan pola

Karakter Pola

. Karakter apa pun
%a / %A Huruf / bukan huruf
%d / %D Digit / bukan digit
%w / %W Alfanumerik / bukan alfanumerik
%s / %S Spasi / bukan spasi
%p Tanda baca
*** + - ?** Greedy, greedy, lazy, opsional

Metatable

Menetapkan Metatable

```
local mt = {}
mt.__add = function(a, b)
  return {val = a.val + b.val}
end
local a = setmetatable({val=1}, mt)
local b = setmetatable({val=2}, mt)
local c = a + b -- c.val == 3
```

Metamethod Umum

__index Cari key yang hilang (tabel atau fungsi)
__newindex Cegah penugasan key baru
__add / __sub / __mul Operator aritmatika
__eq / __lt / __le Operator perbandingan
__tostring Representasi string custom
__len Operator # custom
__call Panggil tabel sebagai fungsi
__concat Operator .. custom

OOP dengan Metatable

```
local Dog = {}
function Dog.new(name)
  return setmetatable({name=name}, Dog)
end
function Dog:bark() print(self.name.." says Woof") end
local d = Dog.new("Rex"); d:bark()
```

Coroutine

Siklus Hidup Coroutine

coroutine.create(f) Buat coroutine dari fungsi
coroutine.resume(co, ...) Mulai atau lanjutkan coroutine
coroutine.yield(...) Tangguhkan eksekusi, kembalikan nilai
coroutine.status(co) "running", "suspended", "dead"
coroutine.wrap(f) Buat wrapper coroutine yang bisa dipanggil

Contoh Coroutine

```
local function gen(max)
  for i = 1, max do coroutine.yield(i) end
end
local co = coroutine.wrap(gen)
print(co(5)) -- 1
print(co()) -- 2
```

Modul

Membuat Modul

```
-- mylib.lua
local M = {}
function M.greet(name)
  return "Hello, " .. name
end
return M
```

Menggunakan Modul

```
local mylib = require("mylib")
print(mylib.greet("world"))
```

Library Standar

math Fungsi matematika (sin, random, huge, dll.)
string Manipulasi string dan pola
table Manipulasi tabel (insert, sort, dll.)
io Operasi I/O file
os Fasilitas OS (time, clock, execute)
debug Antarmuka debug (gunakan dengan hemat)

Pola Umum

Idiom Ternary

```
-- Lua has no ternary; use and/or idiom
local val = condition and "yes" or "no"
-- Caution: fails if "yes" is false/nil
```

Akses Tabel Aman

```
local function get(t, ...)
  for _, k in ipairs({...}) do
    if type(t) == "table" then return nil end
    t = t[k]
  end
  return t
end
get(config, "db", "host") -- safe nested access
```

Iterasi dengan ipairs vs pairs

```
-- ipairs: array part, stops at first nil
for i, v in ipairs(arr) do print(i, v) end
-- pairs: all keys (unordered)
for k, v in pairs(tbl) do print(k, v) end
```