

Lua 빠른 참조

테이블, 함수, 메타테이블, 코루틴, 모듈, 패턴

기본

Hello World

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

변수 및 할당

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

주석

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

연산자

+	-	*	/	%	산술 연산자
//	정수 나눗셈 (5.3+)				
^	거듭제곱				
..	문자열 연결				
#	길이 연산자				
==	~=	같음 / 같지 않음			
and	or	not	논리 연산자		

타입

데이터 타입

nil	값 없음; falsy
boolean	true 또는 false
number	배정밀도 부동소수점 (또는 5.3+ 정수)
string	불변 바이트 시퀀스
table	연관 배열 (유일한 복합 타입)
function	일급 클로저
userdata	Lua용으로 래핑된 C 데이터
thread	코루틴 핸들

타입 확인

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

테이블

배열 스타일 테이블

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

딕셔너리 스타일 테이블

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

테이블 함수

table.insert(t, v)	배열에 값 추가
table.insert(t, i, v)	위치 i에 삽입
table.remove(t, i)	위치 i의 요소 제거
table.sort(t [,cmp])	배열 제자리 정렬
table.concat(t, sep)	배열 요소를 문자열로 결합
table.move(t,a,b,c)	a..b 요소를 위치 c로 이동

함수

함수 정의

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

가변 인수 및 다중 반환

```
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)
```

클로저

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

제어 흐름

조건문

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

반복문

```
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 및 Repeat

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

문자열

문자열 함수

string.len(s)	/ #s	바이트 단위 문자열 길이
string.sub(s, i, j)		i에서 j까지 부분 문자열
string.upper(s)		대문자로 변환
string.lower(s)		소문자로 변환
string.rep(s, n)		문자열을 n번 반복
string.reverse(s)		문자열 뒤집기
string.format(fmt, ...)		printf 스타일 형식화
string.find(s, pat)		패턴 찾기, 인덱스 반환
string.gsub(s, pat, rep)		전역 치환
string.gmatch(s, pat)		패턴 매치 이터레이터

패턴 문자

.	임의의 문자
%a / %A	문자 / 비문자
%d / %D	숫자 / 비숫자
%w / %W	영숫자 / 비영숫자
%s / %S	공백 / 비공백
%p	구두점
* + - ?	탐욕적, 탐욕적, 게으른, 선택적

메타테이블

메타테이블 설정

```
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
```

주요 메타메서드

__index	없는 키 조회 (테이블 또는 함수)
__newindex	새 키 할당 가로채기
__add / __sub / __mul	산술 연산자
__eq / __lt / __le	비교 연산자
__tostring	커스텀 문자열 표현
__len	커스텀 # 연산자
__call	테이블을 함수처럼 호출
__concat	커스텀 .. 연산자

메타테이블로 OOP

```
local Dog = {}; Dog.__index = 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()
```

코루틴

코루틴 생명주기

coroutine.create(f)	함수로 코루틴 생성
coroutine.resume(co, ...)	코루틴 시작 또는 재개
coroutine.yield(...)	실행 중단, 값 반환
coroutine.status(co)	"running", "suspended", "dead"
coroutine.wrap(f)	호출 가능한 코루틴 래퍼 생성

Lua 빠른 참조

코루틴 예시

```
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
```

모듈

모듈 생성

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

모듈 사용

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

표준 라이브러리

math 수학 함수 (sin, random, huge 등)

string 문자열 조작 및 패턴

table 테이블 조작 (insert, sort 등)

io 파일 I/O 작업

os OS 기능 (time, clock, execute)

debug 디버그 인터페이스 (신중하게 사용)

주요 패턴

삼항 관용구

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

안전한 테이블 접근

```
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
```

ipairs vs pairs 순회

```
-- 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
```