



Lua – Bevezetés

HUZINA PATRIK – PROGRAMTERVEZŐ INFORMATIKUS

PATRIKHUZINA@GMAIL.COM

Lua – Története:

- ▶ A Lua egy könnyűsúlyú szkriptnyelv, melyet az 1993-ban kezdeményezett Roberto Ierusalimschy, Luiz Henrique de Figueiredo és Waldemar Celes alkotta csapat fejlesztett ki.
- ▶ A neve portugálul "holdat" jelent, utalva a fejlesztők brazil származására és a projekt kezdeti szakaszában tett éjszakai munkájukra.
- ▶ Az eredeti célja a beágyazott rendszerekhez való könnyű illeszkedés volt.
- ▶ A Lua-t gyakran használják játékokban és más alkalmazásokban, ahol fontos a gyorsaság és az egyszerűség.
- ▶ Az open-source licenszének köszönhetően a Lua népszerűvé vált a fejlesztői közösségek körében, és ma is széles körben alkalmazzák a programozásban.

Mi is a Lua?

- ▶ A Python mellett a Lua napjaink egyik leggyakrabban használt szkriptnyelve.
- ▶ A Lua egy könnyűsúlyú, beágyazható szkriptnyelv, amely kiemelkedővé vált a programozók körében egyszerűsége, gyorsasága és kiváló beágyazhatósága révén, különösen a játékiparban és a beágyazott rendszerek fejlesztésében.
- ▶ Nagyon hasonló a Pythonhoz, de míg a Pythont széles körben használják, addig a Lua-t inkább beágyazott rendszerekhez és szkripteléshez alkalmazzák.
- ▶ A Lua egy interpretált szkriptnyelv, ami azt jelenti, hogy nincs különálló fordítási lépés, hanem a forráskód közvetlenül futtatásra kerül egy speciális szoftver, az interpreter segítségével.

Miért annyira gyors a Lua?

- ▶ **Könnyűsúlyú tervezés:** A Lua célja a könnyűsúlyú és gyors végrehajtás volt, így a nyelv kialakítása minimálisra van optimalizálva (a forráskód kb. 1.4 MB, míg a lefordított interpreter kb. 350 KB méretű).
- ▶ **Just-In-Time (JIT) fordítás:** Bár az alap Lua interpreter interpretált, léteznek JIT fordítót tartalmazó implementációk is, például a LuaJIT. A JIT fordítás lehetővé teszi, hogy a kód futás közben forduljon le gépi kóddá, ezáltal növelve a végrehajtási sebességet. (A LuaJIT megmaradt az 5.1-es verziónál, napjaink legfrissebb verziója: 5.4.3)
- ▶ **Kicsi memóriahasználat:** A Lua hatékonyan kezeli a memóriát, és a nyelvi konstrukciók kialakítása lehetővé teszi a hatékony erőforrás-kezelést.

Bevezetés a Lua-ba: Alapvető szintaktika

Pythonban való megfelelője:

```
4 >>> a = 6
5 >>> a
6 6
7 >>> a = "hello"
8 >>> len(a)
9 5
0 >>> a
1 'hello'
2 >>> A
3 Traceback (most recent call last):
4   File "<stdin>", line 1, in <module>
5 NameError: name 'A' is not defined
6 >>> "hello " + "world"
7 'hello world'
8 >>> "hello " + 6
9 Traceback (most recent call last):
10  File "<stdin>", line 1, in <module>
11 TypeError: cannot concatenate 'str' and 'int' objects
12 >>> "hello " + str(6)
13 'hello 6'
```

változót nem kell külön deklarálni

```
PS C:\Users\Patrik> lua54
Lua 5.4.2 Copyright (C) 1994-2020 Lua.org, PUC-Rio
> a = 6
> a
6
> a = 'Hello'
> print(a)
Hello
> A
nil
> "hello" .. 'world'
helloworld
> "hello" .. 6
hello6
>
```

a Lua nem dob hibát ismeretlen változó esetén, hanem nil-t!

Bevezetés a Lua-ba: Alapvető szintaktika

Pythonban való megfelelője:



```
#Ez egy egysoros komment
```

```
#Ez  
#Egy  
#Többsoro komment
```

```
a = ""  
Ez egy többsoros  
sztring  
""
```



```
--Ez egy egysoros komment
```

```
--[[Ez  
Egy  
Többsoro komment]]
```

```
a = [[  
Ez egy többsoros  
sztring  
]]
```

Bevezetés a Lua-ba: Alapvető szintaktika

Pythonban való megfelelője:

```
1 #!/usr/bin/env python3
2
3
4 def main():
5     print("Hello, World!")
6
7
8 if __name__ == "__main__":
9     main()
```

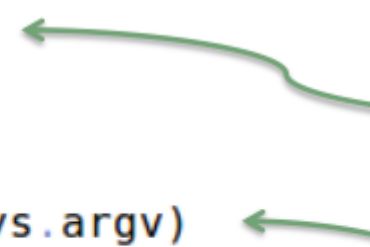


```
#!/usr/bin/env
function main()
    print("Hello, World!")
end
main()
```


Bevezetés a Lua-ba: Alapvető szintaktika

Pythonban való megfelelője:

```
3 import sys
4
5
6 def main():
7     print(sys.argv)
8
9
10 if __name__ == "__main__":
11     main()
```

A diagram with two green arrows. The first arrow starts from the Python code line 'import sys' and points to the Lua code line 'arg = {}'. The second arrow starts from the Python code line 'print(sys.argv)' and points to the Lua code line 'for i, v in ipairs(arg) do'.

```
arg = {}
print(arg)

for i, v in ipairs(arg) do
    print("Argumentum #" .. i .. ":", v)
end
```

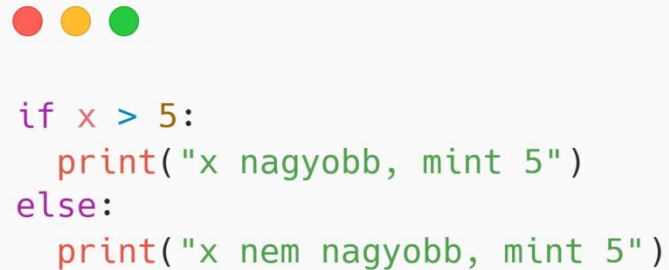
```
-- Kiírjuk a fájlnevet (nulladik index)
print("Fájlnev:", arg[0])

-- Kiírjuk a többi parancssori argumentumot
for i = 1, #arg do
    print("Argumentum #" .. i .. ":", arg[i])
end
```

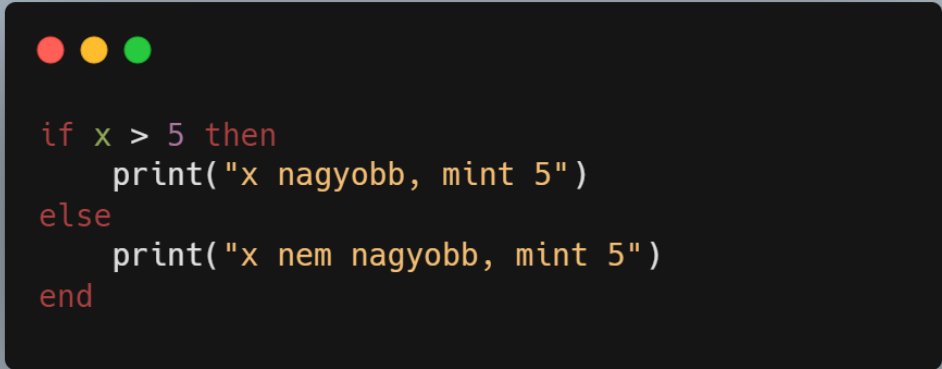

Bevezetés a Lua-ba: Alapvető szintaktika

► Feltételes szerkezet:

Pythonban való megfelelője:



```
if x > 5:  
    print("x nagyobb, mint 5")  
else:  
    print("x nem nagyobb, mint 5")
```



```
if x > 5 then  
    print("x nagyobb, mint 5")  
else  
    print("x nem nagyobb, mint 5")  
end
```

Bevezetés a Lua-ba: Alapvető szintaktika

► For ciklus:

Pythonban való megfelelője:



```
for i in range(1, 6):  
    print(i)
```



```
for i = 1, 5 do  
    print(i)  
end
```

Bevezetés a Lua-ba: Alapvető szintaktika

- While ciklus: (Akkor fut le ha a feltétel igaz. A feltétel a ciklus elején értékelődik ki.)

Pythonban való megfelelője:

```
count = 0

while count < 5:
    print(count)
    count += 1
```

```
local count = 0
while count < 5 do
    print(count)
    count = count + 1
end
```

Luaban nincs count +=, count -=, count++, count--

Bevezetés a Lua-ba: Alapvető szintaktika

- Repeat Until ciklus: (Akkor fut le ha a feltétel hamis. A ciklus végén értékelődik ki a feltétel. Vagyis: ha a feltétel igaz, akkor kilép a ciklusból.)

Pythonban való megfelelője:

```
repeatCount = 0

while True:
    print(repeatCount)
    repeatCount += 1
    if repeatCount >= 5:
        break
```

```
local repeatCount = 0
repeat
    print(repeatCount)
    repeatCount = repeatCount + 1
until repeatCount >= 5
```

Bevezetés a Lua-ba: Táblák

- ▶ A tábla a Lua legrugalmasabb adatszerkezete. Hogy miért?
Ami Pythonban lista, set, tuple, dictionary, az Lua-ban egy szimpla tábla.
- ▶ Lua specialitás, hogy az indexelést egy táblánál nem 0-ról kezdjük hanem 1-ről.
- ▶ Alapvető szintaktika:

Pythonban való megfelelője:

```
a = [8, 2, 6]

for n in a:
    print(a)
```

```
a = {8, 2, 6}

for i = 1, #a do
    print(a[i])
end
```

Bevezetés a Lua-ba: Táblák

- Szótár (dictionary), halmaz (set) és tuple definiálása Lua-ban és Python-ban:

Python-ban való megfelelője:



```
my_dict = {'a': 1, 'b': 2, 'c': 3} #Szótár  
my_set = {1, 2, 3} # Halmaz  
my_tuple = (1, 2, 3) # Tuple
```



```
myTable = {a=1, b=2, c=3} -- Szótárak  
mySet = {1, 2, 3} -- Halmazok  
myTuple = {1, 2, 3} -- Tuple
```

setmetatable
(haladó)

Bevezetés a Lua-ba: Táblák (folyt.)

- Szótár (dictionary), halmaz (set) és tuple definiálása Lua-ban és Python-ban:

Lehetőségek egy tábla tartalmának szép kiíratására:

```
#!/usr/bin/env lua

local helper = require("Helper")

function main()
    t = { apple = "green", orange = "orange", banana = "yellow", 5, 6, 7 }
    print(helper.dump(t))
end

main()
```

```
local helper = {} -- The main table

-- pretty print a table
function helper.dump(o)
    if type(o) == "table" then
        local s = "{ "
        for k, v in pairs(o) do
            if type(k) ~= "number" then
                k = "'" .. k .. "'"
            end
            s = s .. "[" .. k .. "] = " .. helper.dump(v) .. ", "
        end
        return s .. "} "
    else
        return tostring(o)
    end
end

return helper
```


Bevezetés a Lua-ba: Táblák

- ▶ A Lua specialitása, hogy kétféleképpen lehet bejárni a táblákat.

1) ipairs: Itt fontos, hogy a tábla tartalma logikailag összefüggő legyen, azaz a tábla kulcsai egymást kövessék. (Pld. egy lista adatszerkezet ezt kielégíti.)

2) pairs: Nála nem fontos, hogy logikailag összefüggő legyen a sorrend. Egyedül azt nézi, hogy a tábla elemeinek a hash értéke különbözik-e és ez alapján végigiterálja a táblát.

```
table = {'a' = 'a', 'b' = 1, 'c' = {}, 'teszt',  
        'super'}  
#table = 2  
for k, v in pairs(table) do  
    print(k) -- 1, 2, 'a', 'b', 'c'  
    print(v) -- 'teszt', 'super', 'a', 1, table  
end
```

```
table = {10, 9, 8, 7, 6}  
#table = 5  
for k, v in ipairs(table) do  
    print(k) -- 1, 2, 3, 4, 5  
    print(v) -- 10, 9, 8, 7, 6  
end
```

Bevezetés a Lua-ba: Sztring

- ▶ Mivel a Python rendelkezik a programozási nyelvekben az egyik legtöbb sztring manipulációs lehetőséggel, ezért szemmel látható, hogy a Lua ehhez képest „primitív”-nek tűnhet. De az alapvető dolgok megtalálhatók benne.

```
myString = "Hello, World!"

length = len(myString)

concatenatedString = "Hello" + "World!"

subString = myString[7:11 + 1] # Jobbról nyílt

isFound = "World" in myString

upperCase = myString.upper()
lowerCase = myString.lower()

replacedString = myString.replace("Hello", "Hi")

reversedString = myString[::-1]

trimmedString = myString.strip()

parts = myString.split(',')

formattedString = "Name: {}, Age: {}".format(name, age)

#Vagy:
formattedString = f"Name: {name}, Age: {age}"
```

```
myString = "Hello, World!"

length = #myString

concatenatedString = "Hello" .. "World!"

subString = string.sub(myString, 7, 11) -- Zárt intervallum

isFound = string.find(myString, "World")

upperCase = string.upper(myString)
lowerCase = string.lower(myString)

replacedString = string.gsub(myString, "Hello", "Hi")

reversedString = string.reverse(myString)

trimmedString = string.trim(myString)

parts = {}
for part in string.gmatch(myString, "([^\,]+)") do
    table.insert(parts, part)
end

formattedString = string.format("Name: %s, Age: %d", name, age)
```

Bevezetés a Lua-ba: local

- ▶ A Lua-ban a local láthatósági módosítást jelent. És ellentétben más programozási nyelvekkel, ha egy fájlban definiálok valamit localban azt más fájl nem láthatja de az adott fájlban belül lévő dolgok sem (amik hátrébb voltak a végrehajtásban).
- ▶ Példa erre:

```
function printStr()  
    print(str) -- Nem ismeri, nem tudja kiírni  
end  
  
local str = 'Teszt'  
  
function printStr2()  
    print(str) -- Output: Teszt  
    print(teszt()) -- Nem ismeri, nem tudja kiírni  
end  
  
local function teszt()  
    return 'Teszt Funkció'  
end  
  
print(teszt()) -- Teszt Funkció
```

Bevezetés a Lua-ba: Classok

- ▶ A Lua nem annyira híres az objektum-orientáltságról, de nem zárja ki teljesen. Van rá lehetőség.
- ▶ Példa erre:

Pythonban való megfelelője:

```
#Python osztály definíció:
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    # Metódus:
    def say_hello(self):
        print(f"Hello, my name is {self.name} and I am {self.age} years old")

# Példa az osztály használatára:
person1 = Person("John", 30)
person1.say_hello()
```

```
-- Lua osztály definíció
Person = {name = "", age = 0}

-- Konstruktor függvény
function Person:new(name, age)
    local obj = {}
    setmetatable(obj, self)
    self.__index = self
    obj.name = name
    obj.age = age
    return obj
end

-- Metódus
function Person:sayHello()
    print("Hello, my name is " .. self.name .. " and I am " .. self.age .. " years old.")
end

-- Példa az osztály használatára
local person1 = Person:new("John", 30)
person1:sayHello()
```

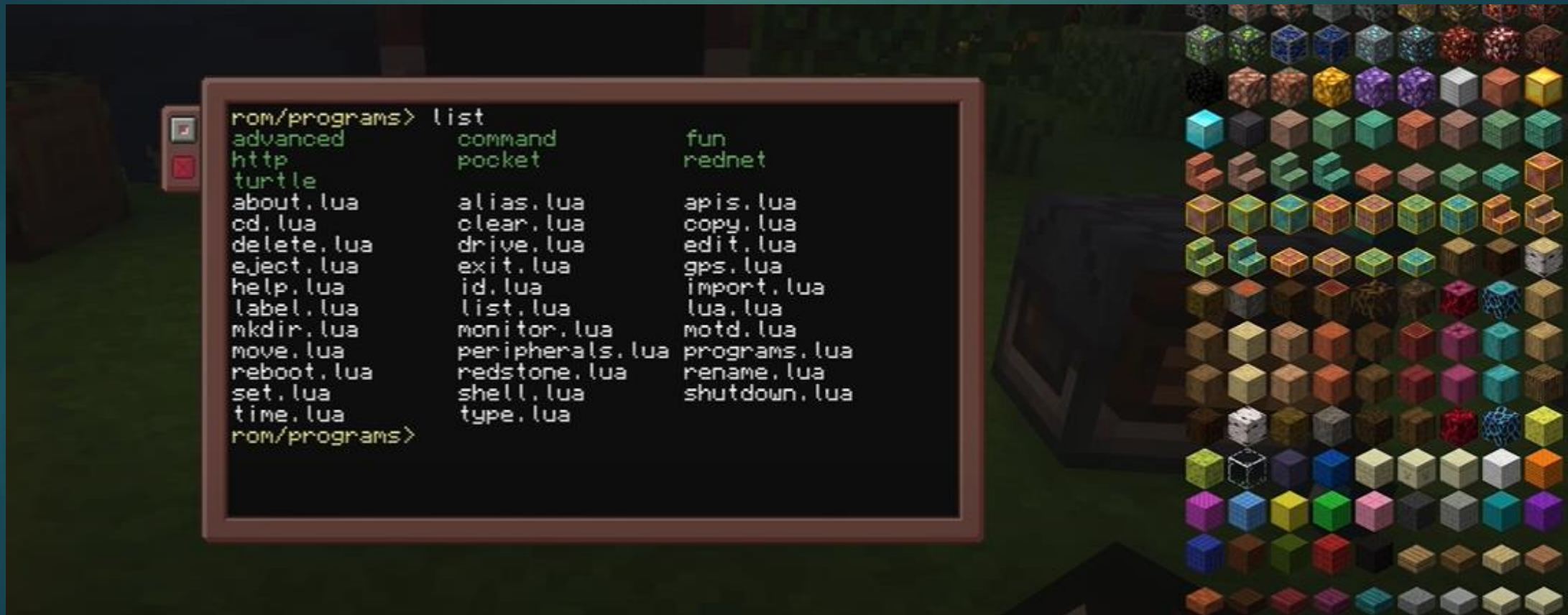
: -ot használunk . helyett!

Mire jó a Lua? Mire használják?

- ▶ A Lua egy könnyűsúlyú, beágyazható szkriptnyelv.
- ▶ Kiválóan alkalmazható játékfejlesztésben és egyéb beágyazott rendszerek tervezésénél.
- ▶ A nyelv kompakt mérete és kivételesen gyors teljesítménye miatt ideális választás játékmotorokban.
- ▶ Emellett széles körben használják beágyazott rendszerek fejlesztéséhez, beleértve mobilalkalmazásokat, okos eszközöket és ipari vezérlőket.
- ▶ A Lua egyszerűsége és könnyen tanulhatósága miatt gyakran preferált olyan projektekben, ahol a gyors és hatékony fejlesztés kiemelten fontos.

Hol használják a Lua-t?

Gyakorlati példa #1 – Minecraft MOD



Hol használják a Lua-t?

Gyakorlati példa #2 – CSGO Hack CFG

Text Example

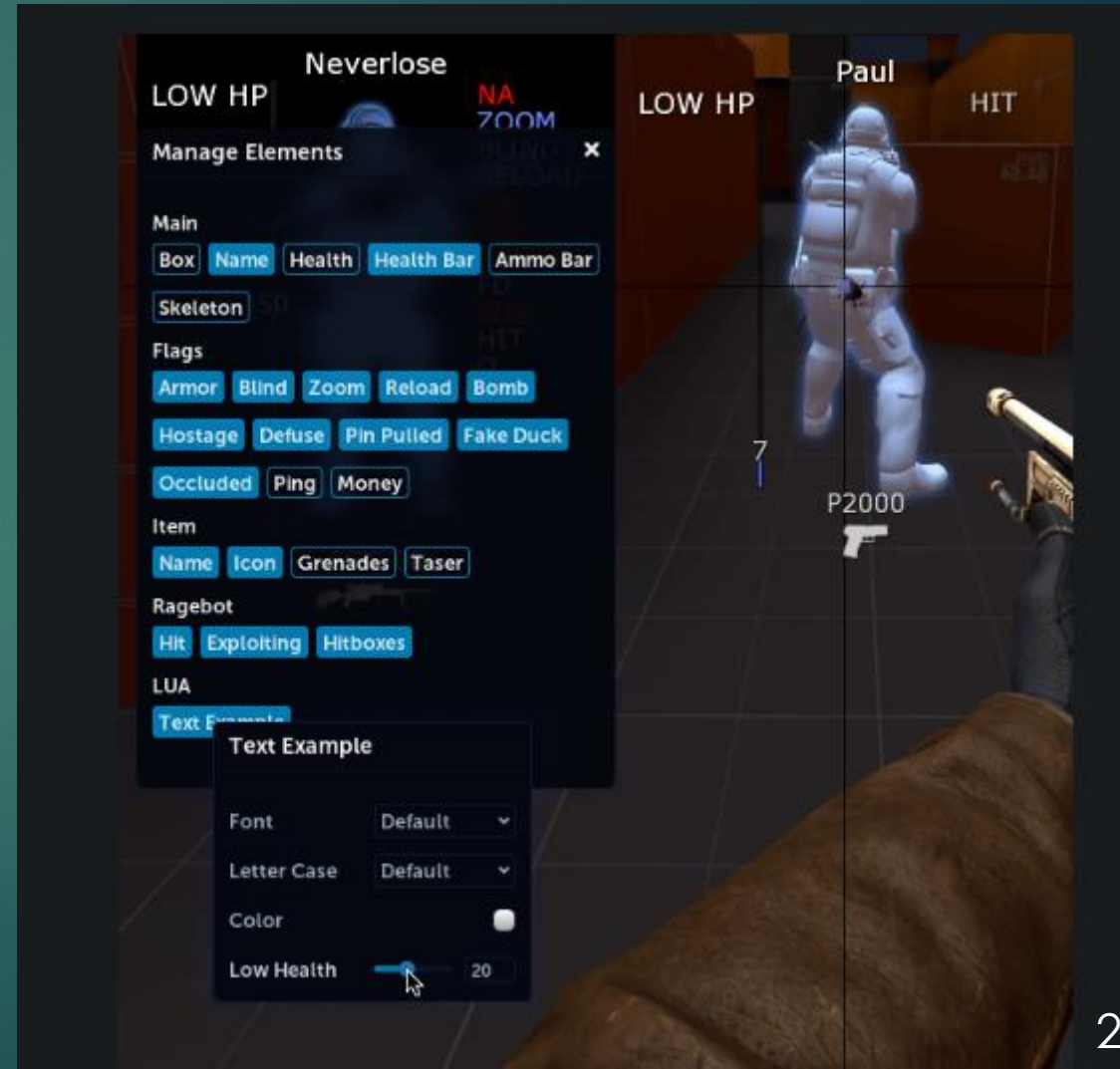
```
local low_health = 20

local text_example = esp.enemy:new_text("Text Example", "LOW HP", function(player)
    local health = player.m_iHealth
    if health > low_health then
        return
    end
    return "LOW HP"
end)

local group_ref = text_example:create()

local slider_ref = group_ref:slider("Low Health", 0, 50, low_health)

slider_ref:set_callback(function(slider_ref)
    low_health = slider_ref:get()
end, true)
```



Hol használják a Lua-t?

Gyakorlati példa #3 – micro text editor

closeOthers

Close all the panes except the current one, i.e. close all the other panes.

I asked here (zyedidia/micro#2996) how to do it and [dmaluka](#) gave a working solution. Thanks!

Usage

Add the following line to your `bindings.json`:

```
"Alt-q": "lua:closeOthers.closeOthers",
```

```
VERSION = "0.1.0"

-- from https://github.com/zyedidia/micro/issues/2996

local micro = import("micro")

function closeOthers(bp)
    local others = {}
    local iter = micro.CurTab().Panes()
    for _, p in iter do
        if p:ID() ~= bp:ID() then
            table.insert(others, p)
        end
    end
    for _, p in pairs(others) do
        p:Quit()
    end
end
```

Hol használják a Lua-t?

Gyakorlati példa #4 – Roblox Pluginok

```
local Player = game.Players.LocalPlayer
door = script.Parent.Parent.Door
door2 = script.Parent.Parent.Door2

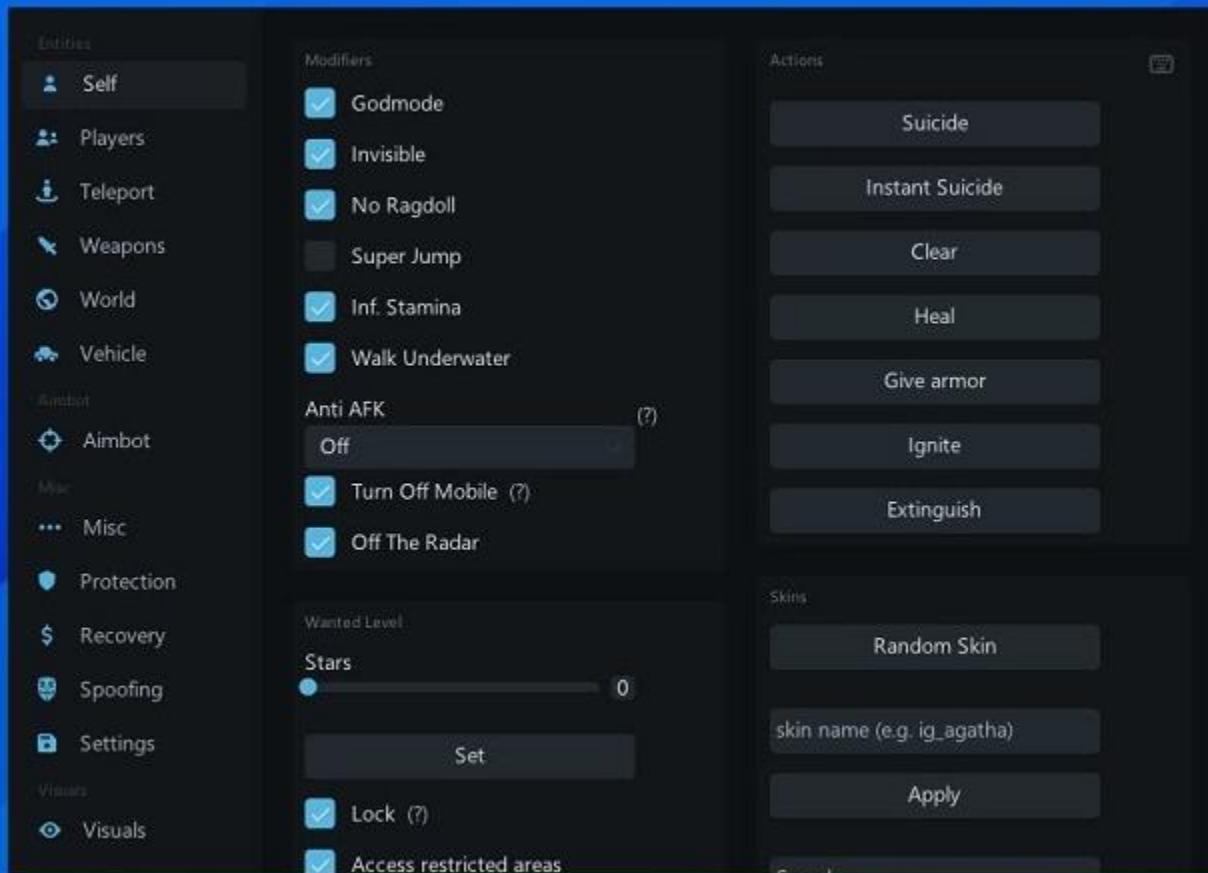
function onClicked()
    for i = 1, 70 do
        door.CFrame = door.CFrame*CFrame.new(0, 0, 0)
        door2.CFrame = door2.CFrame*CFrame.new(0, 0, 0)
        wait()
    end
    wait(3)
    for i = 1, 70 do
        door.CFrame = door.CFrame*CFrame.new(0, 0, 0)
        door2.CFrame = door2.CFrame*CFrame.new(0, 0, 0)
        wait()
    end
end

if Player.UserId == 325345634 then
    script.Parent.ClickDetector.MouseClick:Connect(function()
        onClicked()
    end)
end
```



Hol használják a Lua-t?

Gyakorlati példa #5 – GTA 5 Modmenü



Kódrészlet: MTA Inventory Rendszer (+600 játékos alatt stressz tesztelve)

<https://github.com/HuzinaPatrik/luainventory>

```
--- ...
function renderInventory()
    -- ...
    font = exports['cr_fonts']:getFont("Poppins-SemiBold", 16)
    font2 = exports['cr_fonts']:getFont("Poppins-Medium", 12)
    font3 = exports['cr_fonts']:getFont("Poppins-Regular", 12)
    font4 = exports['cr_fonts']:getFont("Rubik-Regular", 11)
    font5 = exports['cr_fonts']:getFont("RobotoB", 13)

    x, y = getNode("Inventory", "x"), getNode("Inventory", "y")
    UpdatePos("stack", {x + 10, y + 10, 55, 20 + 2})
    UpdateAlpha("stack", tocolor(242,242,242,alpha))
    local _x, _y = x, y

    local w, h = drawnSize["bg"][1], drawnSize["bg"][2]
    local _w, _h = w, h
    local cx, cy = x, y

    local line = 1
    local column = 1
    local startX = _x + (10)
    local _startX = startX
    local startY = _y + (50) -- 28

    local data = cache[elementType][elementID][invType]

    if not data then
        checkTableArray(elementType, elementID, invType)
    end

    -- ...
end
```

```
--- ...
function renderInventory()
    -- ...
    if (invElement and isElement(invElement) and invElement.type == "player" and
    invElement ~= localPlayer) then
        local money = tonumber(invElement:getData("char >> money") or 0)
        local green = "#7cc576"
        if money < 0 then
            green = "#d23131"
        end
        shadowedText(exports['cr_dx']:formatMoney(money) .. " $", _x + 7, _y - 16, _x
        + 7 + 168, _y - 16, tocolor(0, 0, 0, alpha), 1, font5, "left", "center", false, false,
        false, true)
        dxDrawText(green .. exports['cr_dx']:formatMoney(money) .. " $", _x + 7, _y -
        16, _x + 7 + 168, _y - 16, tocolor(242, 242, 242,alpha), 1, font5, "left", "center",
        false, false, false, true)
    end

    dxDrawRectangle(_x, _y, w, h, tocolor(51,51,51,alpha * 0.8))
    dxDrawRectangle(_x + 10, _y + 10, 55, 20, tocolor(41, 41, 41,alpha * 0.9))

    if isInSlot(_x + w - 10 - 15, _y + 10, 15, 15) then
        dxDrawImage(_x + w - 10 - 15, _y + 10, 15, 15, "assets/images/close.png", 0,
        0, tocolor(255, 59, 59, alpha))
    else
        dxDrawImage(_x + w - 10 - 15, _y + 10, 15, 15, "assets/images/close.png", 0,
        0, tocolor(255, 59, 59, alpha * 0.6))
    end

    if invType == specTypes["vehicle"] or invType == specTypes["vehicle.in"] then
        local v = iconPositions[1]
        local ax, ay, name, typeID = unpack(v)
        local name = "vehicle"
        local w, h = 20,20
        dxDrawImage(_x + ax, _y + ay, w, h, "assets/images/"..name.."png", 0,0,0,
        tocolor(255,59,59,alpha))
    elseif invType == specTypes["object"] then
        local v = iconPositions[1]
        local ax, ay, name, typeID = unpack(v)
        local name = "safe"
        local w, h = 20,20
        dxDrawImage(_x + ax, _y + ay, w, h, "assets/images/"..name.."png", 0,0,0,
        tocolor(255,59,59,alpha))
    else
        for k,v in pairs(iconPositions) do
            local ax, ay, name, typeID, w, h = unpack(v)
            if invType == typeID or isInSlot(_x + ax, _y + ay, w, h) then
                dxDrawImage(_x + ax, _y + ay, w, h, "assets/images/"..name.."png",
                0,0,0, tocolor(255,59,59,alpha))
            else
                dxDrawImage(_x + ax, _y + ay, w, h, "assets/images/"..name.."png",
                0,0,0, tocolor(242,242,242,alpha * 0.6))
            end
        end
    end

    -- ...
end
```



És a végére egy tévhit megcáfolása, miszerint a Lua csak pár soros szkriptek megírására szolgálhat:

```
1733 function onQuit()
1734     local items = getItems(source, 1)
1735
1736     for k, v in pairs(items) do
1737         local id, itemid, value, count, status, dutyitem, premium, nbt = unpack(items[k])
1738
1739         if itemid == 317 then
1740             deleteItem(source, k, itemid, true)
1741         end
1742     end
1743 end
1744 addEventHandler("onPlayerQuit", root, onQuit)
```

```
3921 if disableChatInteract then
3922     if type(disableChatInteract) == "table" then
3923         if disableChatInteract[1] == "chat" then
3924             outputChatBox(disableChatInteract[2], 255,255,255,true)
3925         elseif disableChatInteract[1] == "box" then
3926             exports['cr_inboxbox']:addBox(disableChatInteract[2][1], disableChatInteract[2][2])
3927         end
3928     end
3929 else
3930     local syntax = getServerSyntax("I
3931     exports['cr_inboxbox']:addBox("err
3932 end
3933
3934 cancelMove()
3935 end
```

```
4883
4884 ignoreWeaponInteractions = {
4885     [1] = true,
4886     [3] = true,
4887     [4] = true,
4888     [22] = true,
4889     [23] = true,
4890     [24] = true,
4891     [28] = true,
4892     [32] = true,
4893     [16] = true,
4894     [17] = true,
4895     [18] = true,
4896     [41] = true,
4897     [43] = true,
4898     [14] = true,
4899     [46] = true,
4900 }
```

```
2229
2230     exports['cr_chat']:createMessage(localPlayer, "levesz egy maszkot a fejéről", 1) -- // ?!
2231
2232     localPlayer:setData("char >> customName", oldCustomName)
2233
2234     removeCommandHandler("maskname", changeMaskName)
2235     removeCommandHandler("maskname", changeMaskPosition)
2236 end
2237 end
2238 end
```

Köszönöm
szépen a
figyelmet!