Programming the Robots with Lua

To tell the robots what to do, you need to write a script. The script is written in a language called Lua (v5.3). Lua is a very simple but powerful scripting language. You can find more information on the Lua manual: `https://www.lua.org/manual/5.3/`. The Lua Wikipedia page (`https://en.wikipedia.org/wiki/Lua_(programming_language)`) also contains a nice introductory tutorial.

In the following you will find a short Lua reference with examples. It covers just the essential parts of the language you need to program the robots. These expressions are complemented by the robot-related functions described in the accompanying `footbot.pdf`.

Comments

```lua
-- This is a single-line comment
--[[ This is a multi-line
     comment ]]
```

Variables

```lua
-- Assignment (no declaration needed)
x = 2.55    -- number
x = "ciao"  -- string
```

Note: By default all variables in Lua are global variables. This implies that the information assigned to a variable will remain stored and available until the variable is cleared.

Printing to the ARGoS logger

```lua
log("INFO: x = " .. x)
logerr("ERROR: x = " .. x)
```

Note: In ARGoS, the standard Lua `print()` function does not work.

Conditionals

```lua
-- Simple if/then
if x > 3 then
 logerr("x is too big")
end

-- if/then/elseif/else
if x > 3 then
 logerr("x is too big")
elseif x < 3 then
 logerr("x is too small")
else
 logerr("maybe I just don't like x")
end

-- Checking for equality
if x == 3 then
 log("x is equal to 3")
end

-- Checking for inequality
if x ~= 4 then
 log("x is different from 4")
end
```

```lua
-- Combining conditions with OR
if (x > 3) or (x < 3) then
 logerr("x is not 3")
end

-- Combining conditions with AND
if (x > 3) and (y > 3) then
 logerr("x and y are too big")
end

-- Negating a condition
if not (x > 3) then
 logerr("x is <= 3")
end
```

Loops

```lua
-- while loop to print 1 2 3 4 5
x = 0
while x < 5 do
 x = x + 1
 log(x)
end

-- repeat until loop to print 1 2 3 4 5
x = 0
repeat
 x = x + 1
 log(x)
until x == 5

-- for loop to print 1 2 3 4 5
for x = 1, 5 do
 log(x)
end

-- for loop to print 1 3 5
for x = 1, 5, 2 do
 log(x)
end

-- for loop to print 5 4 3 2 1
for x = 5, 1, -1 do
 log(x)
end
```

Tables

```lua
-- Creating an empty table
t = {}

-- Creating a table with some initial value
t = { x=3 }

-- Using the contents of a table: two equivalent ways
log("t.x = " .. t.x)        -- dot syntax
log("t['x'] = " .. t["x"]) -- string syntax
```

```
-- Tables can be used as arrays
a = { "Huey", "Dewey", "Louie" } -- indices are assigned automatically starting from 1
log(a[1])                        -- prints "Huey"
log(a[2])                        -- prints "Dewey"
log(a[3])                        -- prints "Louie"
log(#a)                          -- prints the number of elements in a

-- Printing the contents of a table: a custom function
function table.print(t)
 for key,value in pairs(t) do
    log(key .. " -> " .. value)
 end
end

-- Tables are always passed by reference!
t1 = { x=3 }
t2 = t1       -- now t2 points to the contents of t1 -> no deep copy
t2.x = 5
log(t1.x)     -- prints 5, not 3!

-- Copying tables, the right way
function table.copy(t)
 local t2 = {}
 for key,value in pairs(t) do
    t2[key] = value
 end
 return t2
end
t1 = { x=3 }
t2 = table.copy(t1)
t2.x = 5
log(t1.x)     -- prints 3
log(t2.x)     -- prints 5

-- Sorting the contents of a simple table
a = { "Huey", "Dewey", "Louie" }
table.sort(a)   -- this operation modifies a!
table.print(a) -- prints "1 -> Dewey", "2 -> Huey", "3 -> Louie"

-- Sorting the contents of a nested table
t = {
 { x=4, label="Huey" },
 { x=56, label="Dewey" },
 { x=0.6, label="Louie" }
}
table.sort(t, function(a,b) return a.x < b.x end) -- sort by x in increasing order
table.sort(t, function(a,b) return a.x > b.x end) -- sort by x in decreasing order
table.sort(t, function(a,b) return a.label < b.label end) -- sort by label in increasing order
```

Functions

```
-- Defining a function
function my_function(p)
 log("Called my_function(" .. p .. ")")
end

-- Returning a value
function my_add(a, b)
 return a + b
end
```

Math

```
-- Setting a 2D vector from length and angle
function vec2_new_polar(length, angle)
  local vec2 = {
    x = length * math.cos(angle),
    y = length * math.sin(angle)
  }
  return vec2
end
-- Using the function
v = vec2_new_polar(2, math.pi/3)

-- Function to sum two 2D vectors (v1 = v1 + v2)
function vec2_sum(v1, v2)
  v1.x = v1.x + v2.x
  v1.y = v1.y + v2.y
end
-- Using the function
v1 = { x=1, y=2 }
v2 = { x=3, y=1 }
vec2_sum(v1, v2)
table.print(v1)   -- prints "x -> 4", "y -> 3"

-- Getting the angle of a 2D vector
function vec2_angle(v)
 return math.atan2(v.y, v.x)
end
```

Random number generator

Note: Do not use `math.random()` to get a random number. Rather, use `robot.random`, as described in `footbot.pdf`.