Lauren Schmidt
CSC313
World Research Page

World in Lua:
Besides global variables, Lua supports local variables. The following is created local variables with the **local** statement:

```
  j = 10         -- global variable
   local i = 1    -- local variable
```

Unlike global variables, local variables have their *scope* limited to the block where they are declared. A block is the body of a control structure, the body of a function, or a chunk (the file or string with the code where the variable is declared).
Example:

```
  x = 10
   local i = 1        -- local to the chunk

   while i<=x do
     local x = i*2    -- local to the while body
     print(x)         --> 2, 4, 6, 8, ...
     i = i + 1
   end

   if i > 20 then
     local x          -- local to the "then" body
     x = 20
     print(x + 2)
   else
     print(x)         --> 10  (the global one)
   end

   print(x)           --> 10  (the global one)
```

Example with nil:

```
  local a, b = 1, 10
   if a<b then
     print(a)   --> 1
     local a    -- `= nil' is implicit
     print(a)   --> nil
   end          -- ends the block started at `then'
   print(a,b)   -->  1   10
```

Many languages force you to declare all local variables at the beginning of a block (or a procedure), resulting in people thinking it is a bad practice to use declarations in the middle of a block. This is quite the opposite: By declaring a variable only when you need it, you rarely need to declare it without an initial value (and therefore you rarely forget to initialize it). Moreover, you shorten the scope of the variable, which increases readability.

Example with do:

```
  do
    local a2 = 2*a
    local d = sqrt(b^2 - 4*a*c)
    x1 = (-b + d)/a2
    x2 = (-b - d)/a2
   end        -- scope of `a2' and `d' ends here
   print(x1, x2)
```

Do blocks are useful when finer control is needed over the scope of one or more local variables.
Resource: https://www.lua.org/pil/4.2.html