

# Sintaxis básica

En este apartado vamos a repasar la sintaxis básica de Lua.

## Comentarios de una sola línea

Escribe el caracter `-` (guion) dos veces al inicio de la línea:

```
-- Este es un comentario de una sola línea
```

## Comentarios multilínea

Usa la combinación:

```
--[[  
    Este es un comentario en  
    múltiples líneas.  
]]--
```

## Identificadores

Los identificadores son los nombres usados para las variables, funciones, módulos, etc. Lua define una serie de requisitos para la definición de identificadores. Los caracteres permitidos son:

- Letras mayúsculas (A-Z)
- Letras minúsculas (a-z)
- Guiones bajos (\_)
- Números (0-9)

Los identificadores deben comenzar obligatoriamente por una letra mayúscula, minúscula o guion bajo (\_), seguidas de cero o más caracteres alfanuméricos.

Ten en cuenta que Lua diferencia las mayúsculas de las minúsculas, así, el identificador `miVariable` es diferente del identificador `MiVariable`.

Algunos ejemplos de identificadores válidos son:

```
myVariable contador1 Verificar_Valor _temporal
il j indice TEST
```

## Palabras reservadas

Lua, al igual que todos los lenguajes de programación, define una serie de palabras que están reservadas para Lua. Todas estas palabras constituyen las instrucciones básicas del lenguaje de programación de Lua y no se pueden usar para definir identificadores. A continuación se muestra la lista de las palabras reservadas de Lua:

and	break	do	else
elseif	end	false	for
function	if	in	local
nil	not	or	repeat
return	then	true	until
while			

## Variables globales

En Lua las variables son consideradas generalmente como globales. Para que una variable sea considerada como global debe estar declarada en el ámbito global, es decir fuera de cualquier módulo o función:

```
$ lua
> a = 1
> function SetA ()
>> aa = 2
>> end
> print(a)
1
> print(aa)
nil
```

En el ejemplo anterior, se define una variable `a`, que es global a todas las instrucciones. Luego se define una función `SetA` en la que se define a su vez una variable `aa`. Cuando imprimimos la variable `a`, vemos que está definida ya que devuelve un valor. Sin embargo cuando imprimimos la

variable `aa` nos devuelve `nil` que es el identificador de valor nulo. Esto significa que el ámbito donde se definió `aa` no es global, es decir, no están en el mismo ámbito.

Toda variable que haya sido definida, puede ser anulada asignándole el valor `nil`:

```
a = nil  
print(a) --> Devuelve nil
```

Esto significa que esta variable dejará de existir en el ámbito donde se realizó la asignación. Si una variable tiene un valor diferente de `nil`, esta estará disponible en su ámbito.

---

Revisión #4

Creado 23 mayo 2023 06:39:40 por Guillermo

Actualizado 31 julio 2023 12:02:47 por Guillermo