41.4 LUA Grundlagen - Operatoren

Autor: Goetz
Quelle: Mein EEP-Forum

Operatoren sind das, was Programme clever macht. Weil sie dafür sorgen, dass der Computer etwas rechnet, prüft oder ändert.

Ich habe in den vorigen Kapiteln schon Operatoren benutzt.

Zum Beispiel das = Zeichen für die Codezeile
John = 3

Oder das + Zeichen für
Gleis = 2+1
Gleis = Gleis+1

Viele dieser Zeichen leuchten sofort ein.
Vermutlich hat sich niemand gefragt, warum ein + Zeichen dafür sorgt, dass zwei Werte addiert werden?
Ist doch logisch. Dafür ist so ein + doch da, oder?
Und dieses = Zeichen bedeutet ”gleich“. Weiß doch jedes Kind.
Oder?

Wir verstehen diese Dinge alle, weil wir sie irgendwann gelernt und längst verinnerlicht haben.
Ein Computer nicht. Der hat überhaupt keine Ahnung. Dem muss man wirklich alles sagen.
Zum Glück haben das die Entwickler von Programmiersprachen für uns schon getan.

Genau das ist nämlich eine Programmier- und eine Skriptsprache:
Ein Programm, welches unsere Texte in Befehle für den Computer übersetzt.
Und je nach Programmiersprache sind diese Übersetzungen unterschiedlich. Deshalb ist Lua etwas anders als Basic. Die Übersetzungen, die dort eingebaut wurden, sind verschieden.

Einer dieser Unterschiede ist das = Zeichen.

In der Umgangssprache ergibt sich aus dem gesamten Satz, ob ich etwas ”gleich“ machen möchte oder fragen will, ob etwas ”gleich“ ist.

In Lua nicht. Da muss ich durch die Schreibweise zeigen, was ich meine.

Wenn ich nur ein = schreibe, dann mache ich etwas gleich.
So, wie in
John = 3

Dabei ist auch die Reihenfolge wichtig. Denn Lua kann nicht mitdenken.
Bei der Zeile oben scheint sofort klar zu sein, was gemeint ist.
Aber was ist bei
John = **Ringo**
Wer wird hier wem gleich gemacht?

Das hat man in Lua festgelegt. Das, was vor dem = Zeichen steht, wird verändert.
Also hat die Variable **John** nach der obigen Codezeile den gleichen Wert, der in **Ringo** gespeichert war.
Der alte Wert in **John** wird also mit dem Wert aus **Ringo** überschrieben. Der Wert in **Ringo** ändert sich dabei nicht!

Und weil diese Reihenfolge fest steht, kann man auch nur
John = 3
schreiben, wenn man möchte, dass in der Variablen **John** der Wert 3 gespeichert wird.
3 = **John geht nicht!**

Nun möchten wir manchmal aber auch wissen, ob der Wert in der Variablen **John** gleich 3 ist oder nicht.

Das geht nicht mit
John = 3
weil man damit den Wert 3 in **John** speichert.
Falls **John** also vorher nicht 3 war, dann werde ich es mit dieser Codezeile nie erfahren.
Denn erstens sagt sie mir nicht, ob eine 3 in **John** gespeichert war. Und zweitens ist die Zahl, die da vorher drin war, jetzt auch noch durch die 3 ersetzt worden.

Deshalb brauchen wir in Lua für die Frage, ob der Wert von **John** vielleicht 3 ist, eine eigene Schreibweise. Nämlich zwei = Zeichen direkt hintereinander. Also
John == 3
Das ist ein Vergleich und als Ergebnis bekomme ich entweder ”wahr“ oder ”falsch”.

Manchmal möchte man auch wissen, ob der Wert in John 3 oder mehr ist.
Die Schreibweise dafür ist
John >= 3
das > Zeichen steht für ”größer”.
Also bekomme ich als Ergebnis dieser Prüfung ”wahr“, wenn der Wert in **John** größer oder gleich 3 ist.
Falls der Wert kleiner als 3 ist bekomme ich ”falsch”.

Natürlich kann ich auch
John <= 3
schreiben und so fragen, ob der Wert in John kleiner als 3 oder gleich 3 ist.

Will man nur wissen, ob der Wert in der Variablen **John** größer als 3 ist, dann schreibt man
John > 3
Falls in der Variablen der Wert 3 steht, dann ist die Antwort diesmal ”falsch”, wohingegen bei
John >= 3
die Antwort ”wahr” wäre.

Abfragen und Berechnungen kann man auch in einer Zeile zusammenfassen
John > 1+2
rechnet zuerst 1+2 aus und prüft dann, ob der Wert in der Variablen **John** größer ist.

Gelegentlich möchte man auch nur wissen, ob in der Variablen **John** irgendeine andere Zahl als 3 gespeichert ist. Egal, ob die nun größer oder kleiner (oder vielleicht überhaupt keine Zahl) ist. Die Schreibweise dafür ist
John ~= 3
also zuerst die Welle und dann das = Zeichen.
Als Ergebnis bekommt man dann ”falsch”, wenn in **John** der Wert 3 steht. Steht in **John** keine 3, dann ist die Antwort ”wahr”.

Die hier gezeigten Reihenfolgen für ~= und >= und <= müssen meines Wissens unbedingt eingehalten werden.

Einfache Rechenaufgaben werden so geschrieben, wie man es vermuten würde
+ zum Addieren
- zum Subtrahieren
* für die Multiplikation
/ für die Division

Bei der Multiplikation muss man darauf achten den Stern * und nicht ein kleines x zu verwenden.
Und bei der Division muss es der Schrägstrich vorwärts / sein. Keinesfalls der Schrägstrich rückwärts \

Im folgenden Kapitel werde ich ein paar spezielle Operatoren erläutern