**Ruby - Einführung**

irb --simple-prompt

Objekt: 3, “hallo“

Klasse: Integers, Floats, Strings, Arrays

Methode: + - \* / \*\* % capitalize reverse length

Variablen/Konstanten

Integers (ganze Zahlen) (Bsp. 3 / 2 => 1)

Floats (Fließkommazahlen) (Bsp. 3.0 / 2.0 => 1.5)

- + Addition (Bsp. 1 + 2 => 3)

- - Subtraktion (Bsp. 4 - 3 => 1)

- \* Multiplikation (Bsp. 3 \* 2 => 6)

- / Division (Bsp. 4 / 2 => 2)

- \*\* Exponent (hoch) (Bsp. 3 \*\* 2 => 9)

- % Rest nach Division (Bsp. 5 % 2 => 1)

Strings (Zeichenketten) (Bsp. “Hallo Welt“)

- “Hello“ + “World“ => “Hello World“

- “hi“ \* 3 => “hihihi“

- “1“ + “2“ => “12“

- “1“ \* 2 => “11“ (“1“ wird als Zeichen betrachtet. Nicht als Zahl)

- “hallo“.capitalize => “Hallo“

- “hallo“.reverse => “ollah“

- “hallo“.next => “hallp“

- “hallo“.upcase => “HALLO“

- “HALLO“.downcase => “hallo“

- “Hallo“.swapcase => “hALLO“

- "hallo".length => 5

- "hallo".length + "welt".length => 9

Konvertieren zwischen Klassen

- Integer/Float.to\_s (von Integer/Float zu String) (Bsp. 5.to\_s => “5“)

- Integer/String.to\_f (von Integer/String zu Float) (Bsp. “5“.to\_f => 5.0)

- Float/String.to\_i (von Float/String zu Integer) (Bsp. 5.0.to\_i => 5)

Überprüfung

- Integer/Float/String.is\_a?(Integer/Float/String) (Bsp. “7“.is\_a?(Integer) => false)

Variable (Name den Ruby einem bestimmten Objekt zuordnet) !Variablennamen dürfen nicht mit einem Großbuchstaben beginnen.

- stadt = "Berlin" => "Berlin"

- hey = "Hallo " => "Hallo "

hey = hey + "Welt!" => "Hallo Welt!"

langer Ausdruck:

- num1 = 2 + 4 + 6 + 8 => 20

num1 = num1 / 5 => 4

num2 = 2 \* 3 \* 4 => 24

num3 = num2 - num1 => 20

kurzer Ausdruck:

- num1 = 2 + 4 + 6 + 8 => 20

- num1 /= 5 => 4

- num2 = 2 \* 3 \* 4 => 24

- num2 -= num1 => 20

Konstante (Wert ändert sich nicht/Warnung) !Konstantenname muss mit einem Großbuchstaben beginnen.

- Stadt = "Paris" => "Paris"

Stadt = "Hamburg"

(irb):98: warning: already initialized constant Stadt

(irb):97: warning: previous definition of Stadt was here

=> "Hamburg"

**Programm ausführen**  !Bei Ruby Programmen werden nur die Zeilen mit puts auf dem Bildschirm gezeigt.

- Editor öffnen

- folgendes eingeben: puts "Hallo Welt"

- speichern als hallo.rb (Desktop)

- Terminal/Konsole öffnen: cd desktop

- ruby hallo.rb

Hallo Welt

*Editor*:

num1 = 2 + 4 + 6 + 8

num1 /= 5

num2 = 2 \* 3 \* 4

num2 -= num1

puts "Die Antwort ist " + num2 + "."

*Konsole*:

ruby hallo.rb

Die Antwort ist 20.

Schleifen

Editor:

4.times do

puts "Moin"

end

Konsole:

ruby schleife.rb

Moin

Moin

Moin

Moin

Zähler

Editor:

zaehler = 0

4.times do

zaehler += 1

puts "zaehler = " + zaehler.to\_s

end

Konsole:

ruby zaehler.rb

zaehler = 1

zaehler = 2

zaehler = 3

zaehler = 4

Die Summe von Zahlen (1-11)

Editor:

zaehler = summe = 0

11.times do

zaehler += 1

summe += zaehler

puts "Summe bei " + zaehler.to\_s + " = " + summe.to\_s

end

Konsole:

ruby summe.rb

Summe bei 1 = 1

Summe bei 2 = 3

Summe bei 3 = 6

Summe bei 4 = 10

Summe bei 5 = 15

Summe bei 6 = 21

Summe bei 7 = 28

Summe bei 8 = 36

Summe bei 9 = 45

Summe bei 10 = 55

Summe bei 11 = 66

Benutzereingaben verarbeiten

Editor:

puts “Wie ist Ihr Name?“

name = gets

puts “Hallo “ + name + "!"

Konsole:

ruby name.rb

Wie ist ihr Name?

Hans

Hallo Hans

! (Zeilensprung, weil wir Enter eingegeben haben)

Wie können wir den Zeilensprung verhindern?:

Editor:

puts "Wie ist Ihr Name?"

name = gets

name = name.***chomp***

oder gleich:

Editor:

puts: "Wie ist Ihr Name?"

name = gets.***chomp*** (***chomp*** ist eine String Methode. ***chomp*** kann auf puts angewandt werden, weil es ja irgendwas wiedergibt)

Konditionale

if stadt == "Berlin"

drinking\_age = 18

else

drinking\_age = 21

end

Das sagt aus:

Wenn (if) die stadt gleich (==) "Berlin" ist, dann setze drinking\_age auf 18. Andernfalls (else) setze drinking\_age auf 21.

true (wahr) und false (falsch)

- (=) gleich (Zuweisungsoperator) (Bsp. num = 4 => 4)

- == gleich (Vergleichsoperator) (Bsp. num == 4 => true)

- != nicht gleich (Bsp. 7 != 2 => true)

- > größer als (Bsp. 9 > 8 => true)

- < kleiner als (Bsp. 1 < 3 => true)

- >= größer als oder gleich (Bsp. 7 >= 6 => true)

- <= kleiner als oder gleich (Bsp. 2 <= 3 => true)

Mit ***elsif*** kann man mehr als eine Bedingung einfügen

if alter >= 60

puts "Seniorenpreis"

***elsif*** alter >= 14

puts "Erwachsenenpreis"

***elsif*** alter > 2

puts "Kinderpreis"

else

puts "Frei"

end

Das sagt aus:

Wenn alter 60 ist oder mehr, geben wir Seniorenpreis zurück. Wenn das nicht wahr ist, aber alter 14 ist oder mehr, geben wir Erwachsenenpreis zurück. Wenn das nich wahr ist, aber alter mehr als 2 beträgt, geben wir Kinderpreis zurück. Andernfalls (else) fahren wir umsonst.

! Es wird immer nur die erste Bedingung, die wahr ist, ausgeführt.

While-Schleife

while Konditional

…

end

Editor:

zaehler = 0

while zaehler <= 10

puts "zaehler = " + zaehler.to\_s

zaehler += 1

end

Konsole:

ruby whileschleife.rb

zaehler = 0

zaehler = 1

zaehler = 2

zaehler = 3

zaehler = 4

zaehler = 5

zaehler = 6

zaehler = 7

zaehler = 8

zaehler = 9

zaehler = 10

Kommentar

***#*** das hier wäre jetzt ein Kommentar

Arrays