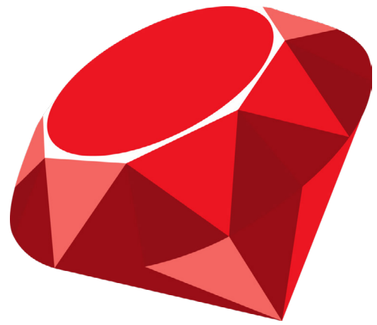

RUBY .rb [2.6.5]

GLIEDERUNG

FAKTEN [1995, Yukihiro „Matz“ Matsumoto]

+ open source
+ Objektorientiert
+ reich an Bibliotheken > Ruby on Rails
+ entwicklerfreundliche Sprache
+ Ruby-Interpreter nötig
+ Dont Repeat Yourself
+ Convention Over Configuration

```ruby
class Testklasse
    def hallosagen(name)
        puts "Hallo #{name}!"
    end
end

testkl = Testklasse.new

testkl.hallosagen("Anton")
```

https://www.ruby-lang.org/en/downloads | apt-get install ruby | GITLAB-HDM:/am180/ruby

„HELLO WORLD“

puts

- Ausgabe der Zeichenkette innerhalb den Anführungszeichen

p

- Kurze Schreibweise
- Ausgabe in Anführungszeichen

print

- Ausgabe der Zeichenkette ohne Zeilenumbruch „\n“

```ruby
puts "Hello World!"

print "Ausgabe ohne Zeilenumbruch - "

p "Die kürzere Variante"

=begin
=> Hello World
=> Ausgabe ohne Zeilenumbruch - "Die kürzere Variante"
=end
```

ZAHLEN

Objekte des Types

- FixNum
  -2^30 bis +2^30 [32bit Maschine]

- BigNum
  -2^62 bis +2^62 [64bit Maschine]

Arithmetische Operatoren

„+“	Addition
„-“	Subtraktion
„*“	Multiplikation
„/“	Division
„%“	Modulo, Division mit Rest
„**“	Potenzfunktion
„+=“	Hochzählen
„-=“	Herunterzählen

```ruby
a = 7   # FixNum
b = 3
bignum = 12345676438901234567890435123456789O # BigNum

addition = a+b
subtraktion = a-b
multiplikation = a*b
division = a/b
modulo = a%b
potenzfunktion = a**b

puts a+=2       # Operatoren ++ und -- gibt es nicht

puts addition, subtraktion, multiplikation
puts division, modulo, potenzfunktion
puts "------------------"

b.upto(a) { |n| puts n } # Methoden der Basisklasse Integer
puts "------------------"
a.downto(b) { |n| puts n }
```

STRINGS

erzeugt durch

- einfache Hochkommata oder
- doppelte Hochkommata oder
- %q oder %Q

Ausdrücken **#{Ausdruck}** können in einem String **nur innerhalb doppelter Hochkommata** definiert werden.

```ruby
mystr = 'Meine Zeichenkette'
mystr2 = %q{Mein Text 2}
myname = 'Anton'
salut = "Hallo #{myname}"
mystr3 = %Q{Hallo #{myname}}

puts mystr
puts mystr2
puts mystr3
puts salut

=begin
Konsolenausgabe:
=> Meine Zeichenkette
=> Mein Text 2
=> Hallo Anton
=> Hallo Anton
=end
```

ARRAYS UND HASHES

Arrays

- Typ der Klasse *Array*
- Erzeugung mittels []
- alternativ über **new** erzeugbar
- in der Länge (.length) unbegrenzt
- bei Bedarf automatisch erweitert
- Indizierung beginnt bei 0

Hashes

- Typ der Klasse *Hash*
- ein Paar aus Schlüssel und Wert
- Erzeugung mittels {}
- alternativ über **new** erzeugbar

```ruby
myarray = [21,5,"Testzeichen"]
myarray[3] = 243
x = 1

for i in myarray
    if x != myarray.length then print "#{i} - "
        else print "#{i}"
    end
    x += 1
end

myHash = Hash.new()
myHash["Key1"] = "Wert1"
myHash[:Key2] = "Wert2"

puts myHash["Key1"]
puts myHash[:Key2]
```

NAMEN UND SYMBOLE

Beginn: Kleinbuchstaben / Unterstrich

- lokale Variable
- Methodenparameter

Beginn: Großbuchstaben

- Klassennamen
- Modulnamen
- Konstanten

Beginn: Klammeraffen @

- Klassenvariablen
- Instanzvariablen

Beginn: Dollarzeichen $

- globale Variablen

```ruby
string = "Thomas"
string1 = "Thomas"
symbol1 = :"Ralf"
symbol2 = :"Ralf"

puts string.object_id, string1.object_id
puts "-----------------"
puts symbol1.object_id, symbol2.object_id

=begin
Konsolenausgabe:
=> 39393320
=> 39393300
=> -----------------
=> 1102948
=> 1102948
=end
```

BEDINGUNGEN

Jede **if**-Anweisung ist mit einem **end**

Operatoren für die Ausdrücke

==	!=
<	<=
>	>=

Ausdrücke verknüpfen

&&	und
||	oder

Statt die Bedingung einer **if**-Anweisung durch **!** zu negieren, bietet Ruby die **unless**-Anweisung

```ruby
x = 5
if x == 0
    puts "0"
elsif x < 0
    puts "kleiner 0"
else
    puts "größer 0"
end

if x == 0 then puts "0"
    elsif x > 0 then puts "kleiner 0"
    else puts "größer 0"
end

puts x if x > 6     # Kurzzeiler
puts "x ist kleiner wie 6" unless x > 6

=begin
Konsolenausgabe:
=> größer 0
=> kleiner 0
=> x ist kleiner wie 6
=end
```

SCHLEIFEN

for-Schleife
- iteriert über die Elemente eines Bereichs
- Arrays oder auch Hashes

while-Schleife
- Block wird ausgeführt, solange Bedingung **true** ist

until-Schleife
- Block wird ausgeführt, solange Bedingung **false** ist

```ruby
s1 = [1,2,3,4,5,6]
s2 = 5
s3 = 0

for i in s1
    puts i
end

puts "---------------"

while s2 > 1 do
    puts s2 -= 1
end

puts "---------------"

until s3 >= 4 do
    puts s3 += 1
end
```

METHODEN

Typangabe entfällt

- für Parameter
und Rückgabewerte

return kann entfallen

- können mehr als
einen Rückgabewert haben

~~Überladen~~

- umgehen durch
unterschiedliche Zahl
von Argumenten

- Defaulbeledung
von Argumenten möglich

```ruby
def vorstellen(*args)
    case args.size
        when 1
            puts "Hallo #{args[0]}!"
        when 2
            puts "#{args[0]} ist #{args[1]} Jahre alt!"
        when 3
            puts "#{args[0]} ist #{args[1]} Jahre alt und lebt in #{args[2]}!"
    end
end

vorstellen "Anton"          # möglich, aber ggf. Warnung
vorstellen("Dana", 21)      # eindeutig und besser
vorstellen("Dana", 21, "Stuttgart")

=begin

def methodenname(Argumente)
    Anweisungn
end

Konsolenausgabe:
=> Hallo Anton!
=> Dana ist 21 Jahre alt!
=> Dana ist 21 Jahre alt und lebt in Stuttgart!
=end
```

KLASSEN UND INSTANZEN

Klassendefinition

- *class Classname*
  end

Erzeugung einer Instanz

- Aufruf der Methode new

Instanzvariablen

- innerhalb einer Instanzmethode

Klassenmethoden

- per Default öffentlich

```ruby
class Student
    attr_reader :semester
    attr_writer :semester

    def initialize(name, vorname, fachrichtung, semester=1)
        @name = name              # Instanzvariable
        @vorname = vorname
        @fachrichtung = fachrichtung
        @semester = semester
    end
end

teststudent = Student.new("Mustermann", "Max", "MI-7")

puts teststudent.semester
```

VERERBUNG

Subklasse erbt

- Variablen
- Konstanten
- Methoden (auch *initialize*)

Subklasse hat Zugriff

- Methoden der Superklasse
  - auch private
  - Kann die Sichtbarkeit für eigene Instanzen anpassen

```ruby
require './12_klassen_und_instanzen'

class Student < Person
    attr_reader :semester
    attr_writer :semester

    def statusinfo
        "Status: #{@status}"
    end
end

teststudent = Student.new("Mustermann", "Max", "Student", "mm123")

puts teststudent.statusinfo
```

MODULE

Mehrfach in Klassen verwendbar
- Aufruf der Methode new

Vorgehensweise
- Module in die Klasse integrieren
- Methoden via Instanzen nutzen

```ruby
module TestModule1
    def fnc1
        "Testmodule 1"
    end
end
module TestModule2
    def fnc2
        "Testmodule 2"
    end
end

class Testklasse
    attr_reader :name
    include TestModule1
    include TestModule2

    def initialize(name)
        @name = name
    end
end
testklasseninstanz = Testklasse.new("Max Mustermann")

puts testklasseninstanz.name
puts testklasseninstanz.fnc1, testklasseninstanz.fnc2
```

RUBY ON RAILS [2005, David Heinemeier Hansson]

Framework für Web-Anwendungen
- Aufruf der Methode new

Interesse geht deutlich zurück!
- wenige Fragen auf Stackoverflow
- Wandlung der Web-Entwicklung
- Erfolg von Node.js

QUELLEN

https://www.ruby-lang.org/de/downloads/
https://www.digitalocean.com/community/tutorials/how-to-write-your-first-ruby-program
http://openbook.rheinwerk-verlag.de/ruby_on_rails/
https://jaxenter.de/einfuhrung-und-uberblick-in-die-webentwicklung-mit-ruby-on-rails-9479
https://www.bader-jene.de/wp-content/uploads/Einfuehrung_Ruby_on_Rails.pdf
https://guides.rubyonrails.org/getting_started.html
https://www.javatpoint.com/ruby-on-rails-introduction
http://railsapps.github.io/what-is-ruby-rails.html
https://www.tutorialspoint.com/ruby-on-rails/rails-introduction.htm
https://www.learnenough.com/ruby-on-rails-6th-edition-tutorial
http://b-simple.de/download/ruby.pdf
https://redmonk.com/sogrady/2019/03/20/language-rankings-1-19/
https://t3n.de/news/ruby-on-rails-interesse-842348/
https://syndicode.com/2018/11/01/alles-uber-ruby-werkzeuge-besonderheiten-und-tipps/
https://naturaily.com/blog/who-gives-f-about-rails