

# Ruby & Perl

---

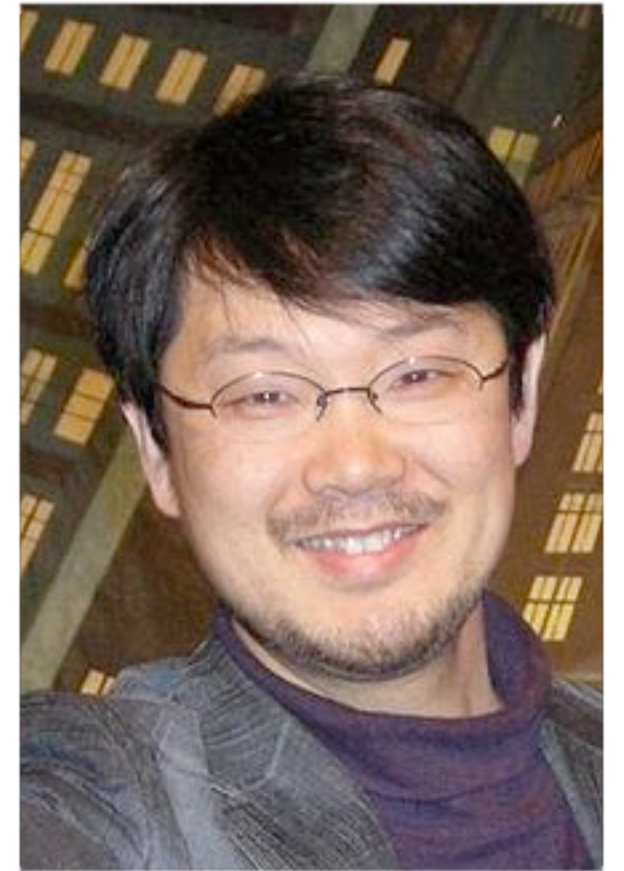
Ein Vergleich im Hinblick auf Erlernbarkeit und Übertragbarkeit auf andere Sprachen

# Kurzprofil

# Ruby

---

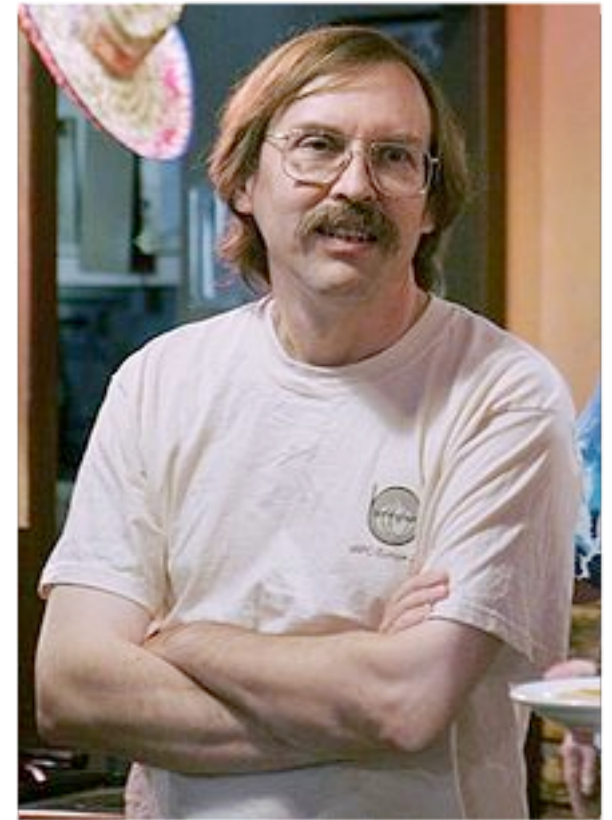
- Shöpfer: Yukihiro “matz” Matsumoto (1995)
- Aktuelle Version : 1.9.x
- Einflüsse : Perl, Smalltalk, Eiffel, Ada und Lisp
- Paradigmen: dynamisch, reflektiv, Objektorientiert
- Mottos: principle of least astonishment (POLA), TIMTOWTDI



# Perl

---

- Schöpfer: Larry Wall (1987)
- Aktuelle Version: 5.10.x (seit )
- Nachfolger: Perl 6, seit 2000 in Entwicklung
- Einflüsse : Awk, C, LISP, Pascal, Sed, Unix-Shell
- Paradigmen: prozedural, modular, teilweise objektorientiert
- Mottos: TIMTOWTDI, makes easy jobs easy and hard jobs possible



# Vergleich

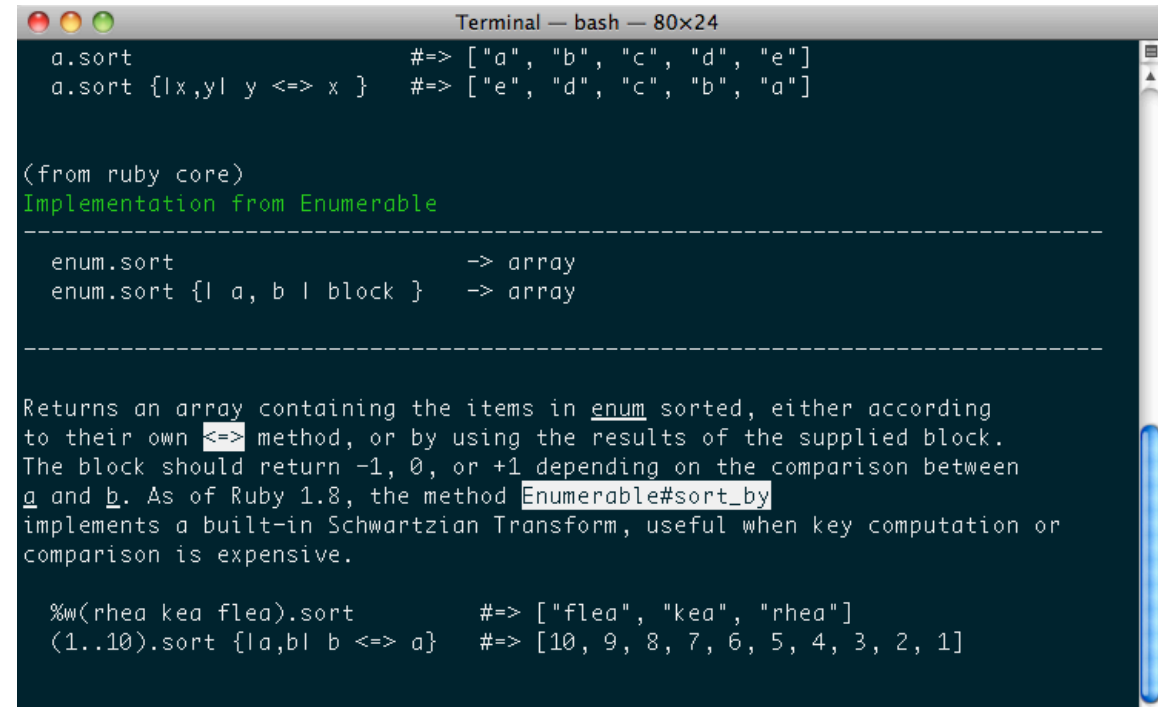
# Implementierungen

---

- MRI (Matz's Ruby Interpreter, in C)
- JRuby
- Mac-Ruby
- Rubinius
- IronRuby
- ...
- Perl-Interpreter in C für Perl 5
- für Perl 6:
- Parrot
- Rakudo
- Pugs

# Ruby - out of the box

- ruby : Der Interpretor
- irb : Interaktive Shell
- ri : Ruby-Dokumentation
- gem : Paketverwaltung



```
Terminal — bash — 80x24
a.sort          #=> ["a", "b", "c", "d", "e"]
a.sort { |x, y| y <=> x }  #=> ["e", "d", "c", "b", "a"]

<from ruby core>
Implementation from Enumerable
-----

enum.sort        -> array
enum.sort { |a, b| block } -> array
-----

Returns an array containing the items in enum sorted, either according
to their own <=> method, or by using the results of the supplied block.
The block should return -1, 0, or +1 depending on the comparison between
a and b. As of Ruby 1.8, the method Enumerable#sort_by
implements a built-in Schwartzian Transform, useful when key computation or
comparison is expensive.

%(rhea kea flea).sort      #=> ["flea", "kea", "rhea"]
(1..10).sort { |a, b| b <=> a }  #=> [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

## RubyGems Documentation Index

### Summary

There are 42 gems installed:

[abstract](#), [actionmailer](#), [actionmailer](#), [actionpack](#), [actionpack](#), [activemodel](#), [activerecord](#), [activerecord](#), [activeresource](#), [activeresource](#), [activesupport](#), [minitest](#), [nokogiri](#), [polyglot](#), [rack](#), [rack-mount](#), [rack-test](#), [rails](#), [rails](#), [railties](#), [rake](#), [rdoc](#), [rspec](#), [rspec-core](#), [rspec-expectations](#), [rspec-mocks](#), [rubygems](#)

### Gems

**abstract 1.0.0** [[rdoc](#)] [[www](#)]

a library which enable you to define abstract method in Ruby

**actionmailer 2.3.8** [[rdoc](#)] [[www](#)] - depends on [actionpack](#).

Service layer for easy email delivery and testing.

**actionmailer 3.0.3** [[rdoc](#)] [[www](#)] - depends on [actionpack](#), [mail](#).

Email composition, delivery, and receiving framework (part of Rails).

**actionpack 2.3.8** [[rdoc](#)] [[www](#)] - depends on [activesupport](#), [rack](#).

Web-flow and rendering framework putting the VC in MVC.

**actionpack 3.0.3** [[rdoc](#)] [[www](#)] - depends on [activemodel](#), [activesupport](#), [builder](#), [erubis](#), [i18n](#), [rack](#), [rack-mount](#), [rack-test](#), [tzinfo](#).

Web-flow and rendering framework putting the VC in MVC (part of Rails).

**activemodel 3.0.3** [[rdoc](#)] [[www](#)] - depends on [activesupport](#), [builder](#), [i18n](#).

A toolkit for building modeling frameworks (part of Rails).

**activerecord 2.3.8** [[rdoc](#)] [[www](#)] - depends on [activesupport](#).

Implements the ActiveRecord pattern for ORM.

**activerecord 3.0.3** [[rdoc](#)] [[www](#)] - depends on [activemodel](#), [activesupport](#), [arel](#), [tzinfo](#).

Object-relational mapper framework (part of Rails).

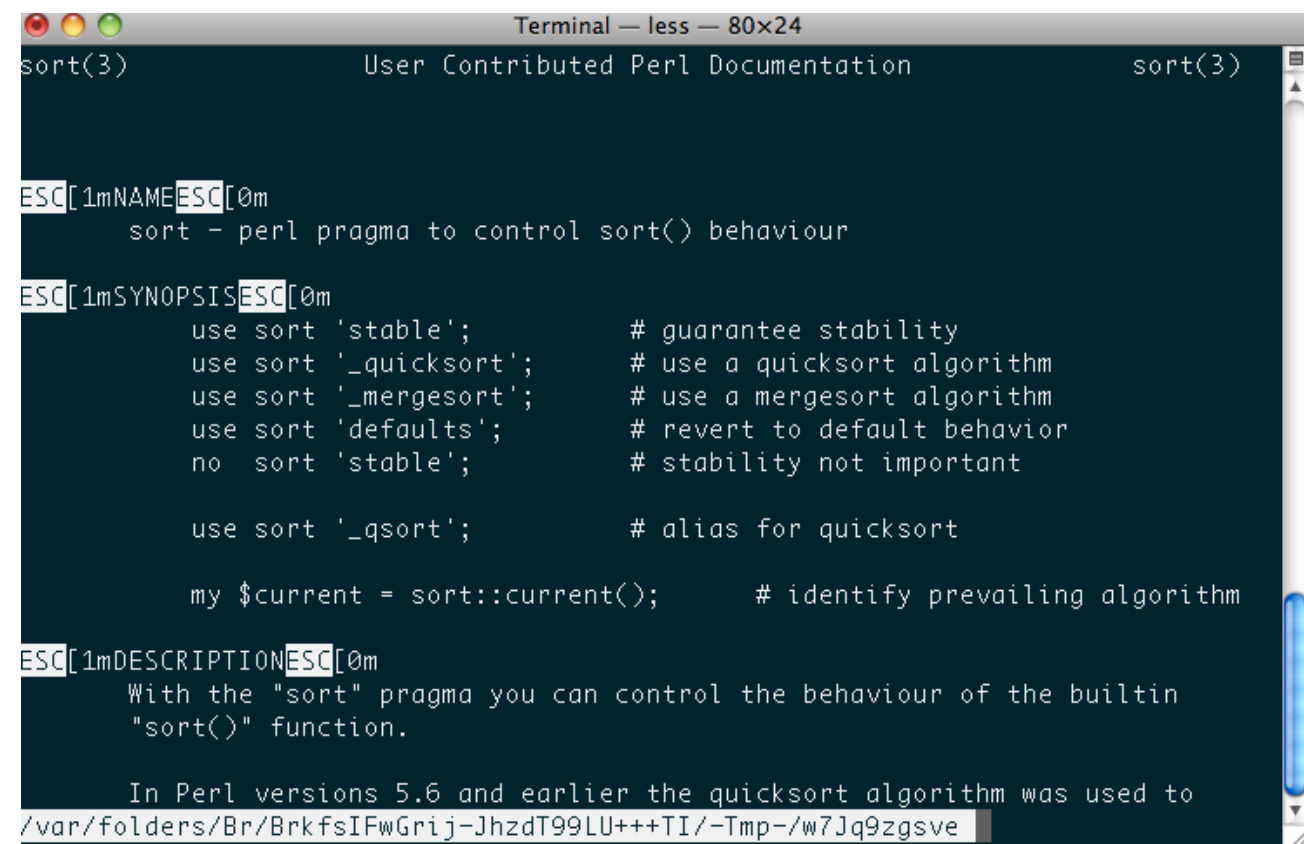
**activeresource 2.3.8** [[rdoc](#)] [[www](#)] - depends on [activesupport](#).

Think Active Record for web resources.

# Perl - out of the box

---

- perl : Der Interpretor
- perldoc : Perl-Dokumentation
- cpan : Paketverwaltung



A terminal window titled "Terminal — less — 80x24" showing the output of the command `perldoc sort`. The window has a dark blue background with white text. The title bar shows standard macOS window controls (red, yellow, green buttons) on the left. The content displays the documentation for the `sort` pragma, including its name, synopsis, and description. The synopsis shows various `use sort` and `no sort` options with their effects. The description explains that the `sort` pragma controls the behavior of the builtin `sort()` function, noting that Perl versions 5.6 and earlier used quicksort.

```
sort(3)                                User Contributed Perl Documentation                                sort(3)

ESC[1mNAMEESC[0m
    sort - perl pragma to control sort() behaviour

ESC[1mSYNOPSISESC[0m
    use sort 'stable';                # guarantee stability
    use sort '_quicksort';            # use a quicksort algorithm
    use sort '_mergesort';            # use a mergesort algorithm
    use sort 'defaults';              # revert to default behavior
    no sort 'stable';                 # stability not important

    use sort '_qsort';                # alias for quicksort

    my $current = sort::current();    # identify prevailing algorithm

ESC[1mDESCRIPTIONESC[0m
    With the "sort" pragma you can control the behaviour of the builtin
    "sort()" function.

    In Perl versions 5.6 and earlier the quicksort algorithm was used to
/var/folders/Br/BrkfsIFwGrij-JhzdT99LU+++TI/-Tmp-/w7Jq9zgsve
```



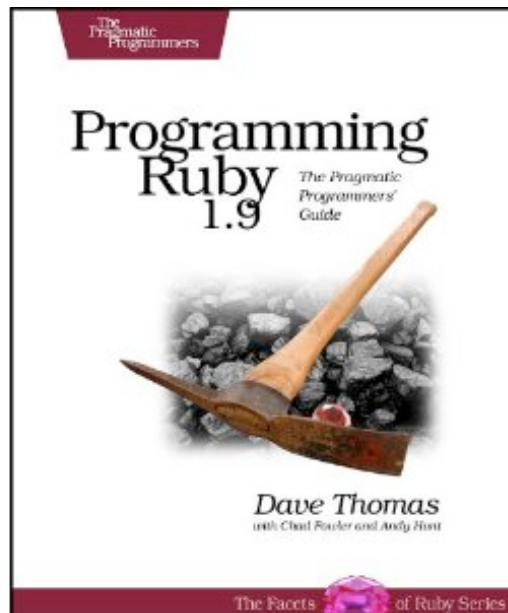
# Erweiterungen

---

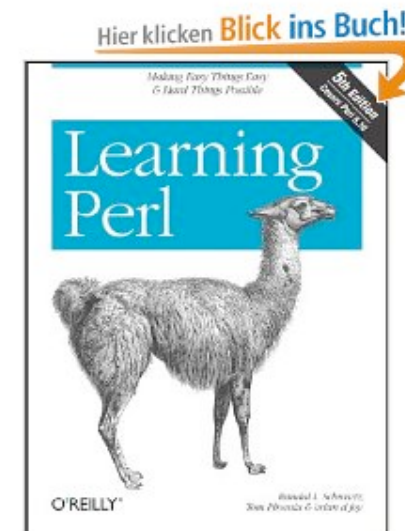
- RubyForge und Ruby Application Archive (RAA): 7000 Anwendungen
- CPAN: 19.000 Anwendungen

# Literatur : Einsteiger / Fortgeschrittene

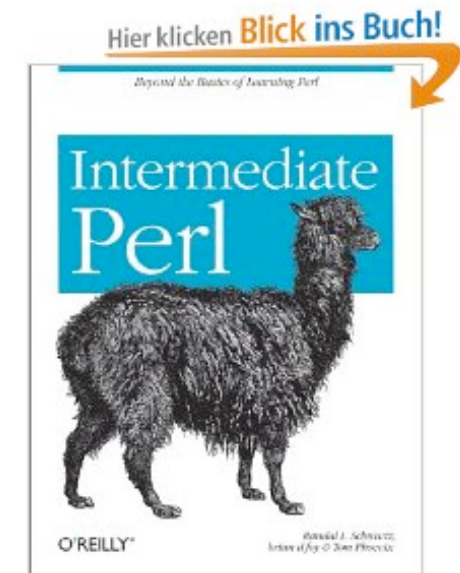
---



- Thomas: Programming Ruby
- ISBN: 978-1934356081

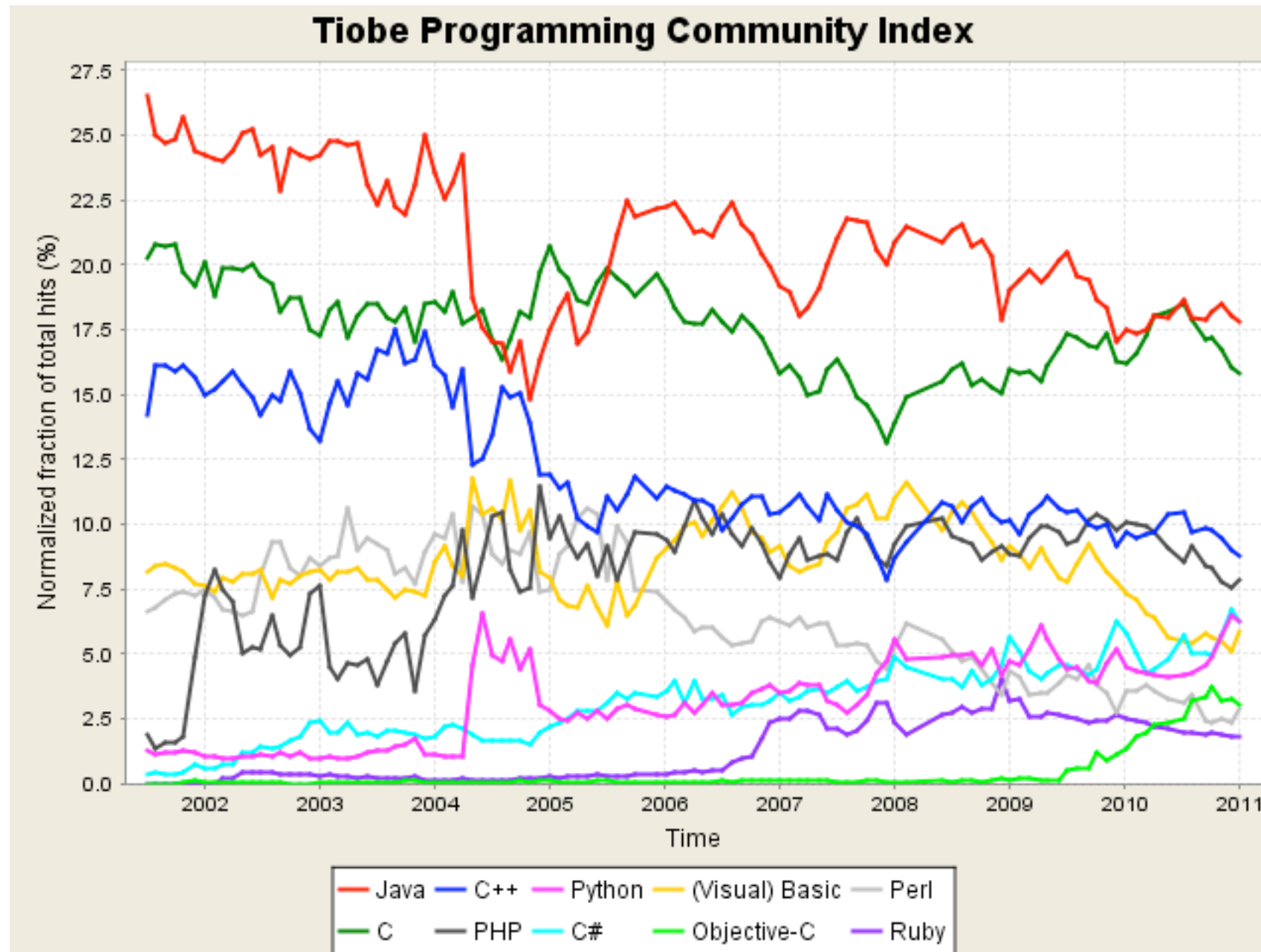


- Schwartz: Learning Perl
- ISBN: 978-0596520106



- Schwartz: Intermediate Perl
- 978-0596102067

# Verbreitung und Entwicklung : Tiobe



# Syntax: Methodendefinition

```
#!/usr/bin/env ruby -w
-
def methode(a,b)-
  puts "a:#{a} und b:#{b}"-
end-
-
methode(1,"test")-
methode(1)
```

a:1 und b:test

**ArgumentError:** wrong number of arguments (1 for 2)

[method methode](#) in `methode.txt` at line 3

[method <main>](#) in `methode.txt` at line 8

Program exited with code #1 after 0.03 seconds.

[copy output](#)

```
#!/usr/bin/env perl -w
-
sub methode {-
  ($a,$b) = @_-
  print "a:". $a. " und b:". $b. "\n";-
}-
-
methode(1,"test");-
methode(3,4,5);-
methode(1);
```

a:1 und b:test  
a:3 und b:4  
a:1 und b:

Program exited with code #0 after 0.04 seconds.

[copy output](#)

# Syntax: Erstellen einer einfachen Klasse

```
#!/usr/bin/env ruby -w
class Greeter
  attr_accessor :name
  def initialize(name)
    @name = name
  end
  def say_hello
    puts "Hello #{name}"
  end
end
g = Greeter.new("Max")
g.say_hello
```

Hello Max

Program exited with code #0 after 0.03 seconds.

[copy output](#)

```
#!/usr/bin/env perl
package Greeter;
sub new {
  my ($class, $name) = @_;
  my $self = {
    _name => $name
  };
  bless($self, $class);
  return $self;
}
sub say_hello {
  $self = shift;
  print "Hello " . $self->{_name} . "\n";
}
package main;
$g = Greeter->new("Max");
$g->say_hello();
```

Hello Max

Program exited with code #0 after 0.03 seconds.

[copy output](#)

# Syntax: Arbeiten mit Encodings

---

```
#!/usr/bin/env ruby-  
# coding: UTF-8-  
# Datei in Latin1 öffnen und als utf-8 bearbeiten und  
# speichern-  
-  
latin1_file = File.open "latin1.txt", "r:iso-8859-1:utf-8"-  
utf8_file = File.open "utf-8.txt", "w:utf-8"-  
-  
latin1_file.each_line do |line|-  
  utf8_file << line-  
end
```

```
#!/usr/bin/perl -w-  
-  
open (LATIN1,"<","latin1.txt");-  
open (UTF8,">:utf8","utf-8.txt");-  
-  
while (<LATIN1) {-  
  print UTF8;-  
}-  
close LATIN1;-  
close UTF8;
```

# Reguläre Ausrücke

---

```
#!/usr/bin/env ruby -w
# coding: utf-8
-
%w{Baum Bär Bier Ärger}.each do |wort|
  puts wort if wort =~ /är/i
end
```

```
#!/usr/bin/env perl -w
use utf8;
-
foreach(qw{Baum Bär Bier Ärger}) {
  print $_ if $_ =~ /är/i
}
```

# Syntax: Erstellen einer Frequenzliste

---

```
#!/usr/bin/ruby -w

# Liest eine Datei ein und erstellt eine Frequenzliste daraus.

#==Author: Andi Neumann
#==Date=11.07.07

h=Hash.new(0)

while zeile=gets
  zeile.chomp!
  zeile.split(/\s/).each do |wort|
    h[word]+=1
  end
end

sorted=h.sort {|a,b|b[1]<=>a[1]}
sorted.each {|frq| puts "#{frq[1]}\t#{frq[0]}\n"}
```

```
#!/usr/bin/perl -w

#Liest eine Datei ein und erstellt daraus eine Frequenzliste
#Autor:Andreas Neumann
#Date:11.07.07

while(<>) {
  chomp;
  foreach (split /\s/,$_){
    $h{$_}++;
  }
}

foreach (sort {$h{$b} <=> $h{$a} } keys %h) {
  print "$h{$_}\t$_\n";
}
```

```
#!/usr/bin/perl -an0
map {$h{$_}++} @F;
map {print "$h{$_}\t$_\n"} (sort {$h{$b}<=>$h{$a}} (keys %h));
```



# Ruby Interpreter + Beispiele für Reflection

---

```
irb(main):001:0> "Weißbier"
=> "Weißbier"
irb(main):002:0> "Weißbier".encoding
=> #<Encoding:UTF-8>
irb(main):003:0> "Weißbier".class
=> String
irb(main):004:0> "Weißbier".methods
=> [:<=>, :==, :===, :eql?, :hash, :casecmp, :+, :*, :%, :[], :[]=, :insert, :length, :
size, :bytesize, :empty?, :=~, :match, :succ, :succ!, :next, :next!, :upto, :index, :ri
ndex, :replace, :clear, :chr, :getbyte, :setbyte, :to_i, :to_f, :to_s, :to_str, :inspec
t, :dump, :upcase, :downcase, :capitalize, :swapcase, :upcase!, :downcase!, :capitalize
!, :swapcase!, :hex, :oct, :split, :lines, :bytes, :chars, :codepoints, :reverse, :reve
rse!, :concat, :<<, :crypt, :intern, :to_sym, :ord, :include?, :start_with?, :end_with?
, :scan, :ljust, :rjust, :center, :sub, :gsub, :chop, :chomp, :strip, :lstrip, :rstrip,
:sub!, :gsub!, :chop!, :chomp!, :strip!, :lstrip!, :rstrip!, :tr, :tr_s, :delete, :squ
eeze, :count, :tr!, :tr_s!, :delete!, :squeeze!, :each_line, :each_byte, :each_char, :e
ach_codepoint, :sum, :slice, :slice!, :partition, :rpartition, :encoding, :force_encodi
ng, :valid_encoding?, :ascii_only?, :unpack, :encode, :encode!, :to_r, :to_c, :>, :>=,
:<, :<=, :between?, :nil?, :!~, :class, :singleton_class, :clone, :dup, :initialize_dup
, :initialize_clone, :taint, :tainted?, :untaint, :untrust, :untrusted?, :trust, :freez
e, :frozen?, :methods, :singleton_methods, :protected_methods, :private_methods, :publi
c_methods, :instance_variables, :instance_variable_get, :instance_variable_set, :instan
ce_variable_defined?, :instance_of?, :kind_of?, :is_a?, :tap, :send, :public_send, :res
pond_to?, :respond_to_missing?, :extend, :display, :method, :public_method, :define_sin
gleton_method, :__id__, :object_id, :to_enum, :enum_for, :equal?, :!, :!=, :instance_ev
al, :instance_exec, :__send__]
```

# Ruby Codebeispiele

```
#!/usr/bin/env ruby -w
# coding: utf-8

class Fixnum
  def the_meaning_of_life
    42
  end
end

puts 1.class
puts 1.the_meaning_of_life
```

Fixnum  
42

```
#!/usr/bin/env ruby -w
# coding: utf-8

for number in 1.upto 10 do
  number.times { |i| print "x" }
  puts "_"
end
```

```
*
*_
**
**_
***
***_
****
****_
*****
*****_
******
******_
*******
*******_
********

```

# Warum Perl?

---

- Riesige Auswahl an Modulen (CPAN)
- ziemlich schnell
- Reguläre Ausdrücke

# Warum kein Perl

---

- schwer wartbarer Code (Lesbarkeit)
- Objektorientierung wirkt aufgesetzt
- Perl 6 immer noch kein Produktivstatus
- Viele Sonderwege: Subroutinendefinition, Objektorientierung, Exceptions, Unit-Testing. Daher Umstieg auf andere Sprachen (Java, Python) schwierig
- Gibt wenig Rückmeldung bei offensichtlichen Fehlern

# Warum Ruby

---

- Gut lesbare Syntax
- Metaprogrammierung
- leichter Umstieg auf Python
- Reguläre Ausdrücke komplett integriert
- Internationalisierung
- Integration in .Net und JVM möglich
- Alles ist ein Objekt