Introduction to Ruby

Konstantinos Karasavvas

CITY College

October 22, 2014

Table of contents

Overview

- History
- Philosophy
- Language Characteristics
- Code Example
- REPL or Language Shell
- Basic code conventions
- Ruby installation and RubyGems

History

- Yukihiro Matsumoto
  - *Matz*

History

- Yukihiro Matsumoto
  - *Matz*
- Development started in 1993

History

- Yukihiro Matsumoto
  - *Matz*
- Development started in 1993
- Public in 1995
  - version 0.95

History

- Yukihiro Matsumoto
  - *Matz*
- Development started in 1993
- Public in 1995
  - version 0.95
- Worldwide use 1999
  - version 1.3

History

- Yukihiro Matsumoto
  - *Matz*
- Development started in 1993
- Public in 1995
  - version 0.95
- Worldwide use 1999
  - version 1.3
- Popularised after 2004
  - version 1.8 (1.8.7)
  - ...particularly by Ruby on Rails

History

- Yukihiro Matsumoto
  - *Matz*
- Development started in 1993
- Public in 1995
  - version 0.95
- Worldwide use 1999
  - version 1.3
- Popularised after 2004
  - version 1.8 (1.8.7)
  - ...particularly by Ruby on Rails
- Currently
  - version 2.1
  - ...backward compatible with 1.9 (1.9.3)

History

- Yukihiro Matsumoto
  - *Matz*
- Development started in 1993
- Public in 1995
  - version 0.95
- Worldwide use 1999
  - version 1.3
- Popularised after 2004
  - version 1.8 (1.8.7)
  - ...particularly by Ruby on Rails
- Currently
  - version 2.1
  - ...backward compatible with 1.9 (1.9.3)
- Huge ecosystem and community
  - ...especially in web development

Philosophy

Often people, especially computer engineers, focus on the machines. They think, "By doing this, the machine will run faster. By doing this, the machine will run more effectively. By doing this, the machine will something something something." They are focusing on machines. But in fact we need to focus on humans, on how humans care about doing programming or operating the application of the machines. We are the masters. They are the slaves.

— Yukihiro Matsumoto

Philosophy

Often people, especially computer engineers, focus on the machines. They think, "By doing this, the machine will run faster. By doing this, the machine will run more effectively. By doing this, the machine will something something something." They are focusing on machines. But in fact we need to focus on humans, on how humans care about doing programming or operating the application of the machines. We are the masters. They are the slaves.

— Yukihiro Matsumoto

- Goals
  - productivity
  - simplicity
  - fun to use

Philosophy

Often people, especially computer engineers, focus on the machines. They think, "By doing this, the machine will run faster. By doing this, the machine will run more effectively. By doing this, the machine will something something something." They are focusing on machines. But in fact we need to focus on humans, on how humans care about doing programming or operating the application of the machines. We are the masters. They are the slaves.

— Yukihiro Matsumoto

- Goals
  - productivity
  - simplicity
  - fun to use
- Motto
  - to make programmers happy!

Main Characteristics

- General purpose language

Main Characteristics

- General purpose language
- Interpreted

Main Characteristics

- General purpose language
- Interpreted
- Dynamically and Strongly typed

Main Characteristics

- General purpose language
- Interpreted
- Dynamically and Strongly typed
- Multi-paradigm
  - Imperative / Procedural
  - Object-Oriented
  - Functional

Main Characteristics

- General purpose language
- Interpreted
- Dynamically and Strongly typed
- Multi-paradigm
  - Imperative / Procedural
  - Object-Oriented
  - Functional
- Open Source
  - MRI or CRuby, JRuby, Rubinius, MacRuby, IronRuby, ...
  - RubySpec

Main Characteristics

- General purpose language
- Interpreted
- Dynamically and Strongly typed
- Multi-paradigm
  - Imperative / Procedural
  - Object-Oriented
  - Functional
- Open Source
  - MRI or CRuby, JRuby, Rubinius, MacRuby, IronRuby, ...
  - RubySpec
- Other features
  - Mixins, closures, metaprogramming, garbage collection, package management (gem), ...

Main Characteristics

- General purpose language
- Interpreted
- Dynamically and Strongly typed
- Multi-paradigm
  - Imperative / Procedural
  - Object-Oriented
  - Functional
- Open Source
  - MRI or CRuby, JRuby, Rubinius, MacRuby, IronRuby, ...
  - RubySpec
- Other features
  - Mixins, closures, metaprogramming, garbage collection, package management (gem), ...
- Very Expressive
  - *With great power, comes great responsibility...*

Code Example

```ruby
#
# This is a trivial example
#

# dynamically typed
x = 7
x = "7"

y = "A week has " + x + " days"
puts y

# strongly typed
x = 7
# y = "A week has " + x + " days."
```

Code Example

```ruby
#
# This is a trivial example
#

# dynamically typed
x = 7
x = "7"

y = "A week has " + x + " days"
puts y

# strongly typed
x = 7
# y = "A week has " + x + " days."
```

```
$ ruby code_example.rb
A week has 7 days
```

REPL or Language Shell

- Read-eval-print loop
  - interactive computer programming environment
  - helps learning

REPL or Language Shell

- Read-eval-print loop
  - interactive computer programming environment
  - helps learning

```
$ irb
irb(main):001:0> x = "7"
=> "7"
```

REPL or Language Shell

- Read-eval-print loop
  - interactive computer programming environment
  - helps learning

```
1 $ irb
2 irb(main):001:0> x = "7"
3 => "7"
4 irb(main):002:0> y = "A week has " + x + " days."
5 => "A week has 7 days."
```

REPL or Language Shell

- Read-eval-print loop
  - interactive computer programming environment
  - helps learning

```
$ irb
irb(main):001:0> x = "7"
=> "7"
irb(main):002:0> y = "A week has " + x + " days."
=> "A week has 7 days."
irb(main):003:0> x = 7
=> 7
```

REPL or Language Shell

- Read-eval-print loop
  - interactive computer programming environment
  - helps learning

```
$ irb
irb(main):001:0> x = "7"
=> "7"
irb(main):002:0> y = "A week has " + x + " days."
=> "A week has 7 days."
irb(main):003:0> x = 7
=> 7
irb(main):004:0> y = "A week has " + x + " days."
TypeError: can't convert Fixnum into String
from (irb):4:in `+'
from (irb):4
from :0
irb(main):005:0>
```

Basic coding conventions

- indentation of 2 spaces
  - ...and no tabs

Basic coding conventions

- indentation of 2 spaces
  - ...and no tabs
- snake_case
  - ...but classes and modules use PascalCase

Basic coding conventions

- indentation of 2 spaces
  - ...and no tabs
- snake_case
  - ...but classes and modules use PascalCase
- semicolon is optional
  - ...but is needed to separate two commands in the same line

Basic coding conventions

- indentation of 2 spaces
  - ...and no tabs
- snake_case
  - ...but classes and modules use PascalCase
- semicolon is optional
  - ...but is needed to separate two commands in the same line
- space between operators in assignments
  - ...but not in method arguments default values

Ruby installation and RubyGems (Ubuntu 12.04)

```
1 $ sudo apt-get install ruby
2 $ sudo apt-get install rubygems
```

Ruby installation and RubyGems (Ubuntu 12.04)

```
1 $ sudo apt-get install ruby
2 $ sudo apt-get install rubygems
```

```
1 $ sudo apt-get install ruby1.9.1
```

Ruby installation and RubyGems (Ubuntu 12.04)

```
1 $ sudo apt-get install ruby
2 $ sudo apt-get install rubygems
```

```
1 $ sudo apt-get install ruby1.9.1
```

```
1 $ ruby -v
```

Ruby installation and RubyGems (Ubuntu 12.04)

```
1 $ sudo apt-get install ruby
2 $ sudo apt-get install rubygems
```

```
1 $ sudo apt-get install ruby1.9.1
```

```
1 $ ruby -v
```

```
1 $ sudo gem install rest-client
```

Ruby installation and RubyGems (Ubuntu 12.04)

```
1 $ sudo apt-get install ruby
2 $ sudo apt-get install rubygems
```

```
1 $ sudo apt-get install ruby1.9.1
```

```
1 $ ruby -v
```

```
1 $ sudo gem install rest-client
```

- built-in from 1.9

Ruby installation with Ruby Version Manager

- RVM: multiple Rubies and gem sets
- Installation: http://rvm.io

Ruby installation with Ruby Version Manager

- RVM: multiple Rubies and gem sets
- Installation: http://rvm.io

```
1 $ \curl -sSL https://get.rvm.io | bash -s stable
```

Ruby installation with Ruby Version Manager

- RVM: multiple Rubies and gem sets
- Installation: http://rvm.io

```
$ \curl -sSL https://get.rvm.io | bash -s stable
```

```
$ rvm requirements
$ rvm notes
$ rvm install 1.9.3
$ rvm install jruby
$ rvm use 1.9.3
$ ruby -v
```

Ruby installation with Ruby Version Manager

- RVM: multiple Rubies and gem sets
- Installation: http://rvm.io

```
$ \curl -sSL https://get.rvm.io | bash -s stable
```

```
$ rvm requirements
$ rvm notes
$ rvm install 1.9.3
$ rvm install jruby
$ rvm use 1.9.3
$ ruby -v
```

```
$ rvm --rvmrc --create 1.9.3@myproject
$ gem install sinatra
$ rvm use system
```

Types

- Numbers
- Strings
- Symbols

Numbers

```
$ irb
irb(main):001:0> 3 + 4
=> 7
```

Numbers

```
$ irb
irb(main):001:0> 3 + 4
=> 7
irb(main):002:0> 30 / 2
=> 15
```

Numbers

```
$ irb
irb(main):001:0> 3 + 4
=> 7
irb(main):002:0> 30 / 2
=> 15
irb(main):003:0> 7 / 2
=> 3
```

Numbers

```
$ irb
irb(main):001:0> 3 + 4
=> 7
irb(main):002:0> 30 / 2
=> 15
irb(main):003:0> 7 / 2
=> 3
irb(main):004:0> 7 % 2
=> 1
```

Numbers

```
$ irb
irb(main):001:0> 3 + 4
=> 7
irb(main):002:0> 30 / 2
=> 15
irb(main):003:0> 7 / 2
=> 3
irb(main):004:0> 7 % 2
=> 1
irb(main):005:0> 7.0 / 2
=> 3.5
```

Numbers

```
$ irb
irb(main):001:0> 3 + 4
=> 7
irb(main):002:0> 30 / 2
=> 15
irb(main):003:0> 7 / 2
=> 3
irb(main):004:0> 7 % 2
=> 1
irb(main):005:0> 7.0 / 2
=> 3.5
irb(main):006:0> 7.class
=> Fixnum
```

Numbers

```
$ irb
irb(main):001:0> 3 + 4
=> 7
irb(main):002:0> 30 / 2
=> 15
irb(main):003:0> 7 / 2
=> 3
irb(main):004:0> 7 % 2
=> 1
irb(main):005:0> 7.0 / 2
=> 3.5
irb(main):006:0> 7.class
=> Fixnum
irb(main):007:0> 7.0.class
=> Float
```

Numbers

```
$ irb
irb(main):001:0> 3 + 4
=> 7
irb(main):002:0> 30 / 2
=> 15
irb(main):003:0> 7 / 2
=> 3
irb(main):004:0> 7 % 2
=> 1
irb(main):005:0> 7.0 / 2
=> 3.5
irb(main):006:0> 7.class
=> Fixnum
irb(main):007:0> 7.0.class
=> Float
irb(main):008:0> 7.methods
=> ["%", "odd?", "inspect", ... ]
```

Numbers

```
$ irb
irb(main):001:0> 3 + 4
=> 7
irb(main):002:0> 30 / 2
=> 15
irb(main):003:0> 7 / 2
=> 3
irb(main):004:0> 7 % 2
=> 1
irb(main):005:0> 7.0 / 2
=> 3.5
irb(main):006:0> 7.class
=> Fixnum
irb(main):007:0> 7.0.class
=> Float
irb(main):008:0> 7.methods
=> ["%", "odd?", "inspect", ... ]
```

- Fixnum
  - .succ
  - .pred
  - .upto
  - .to_f
- Float
  - .round
  - .truncate
  - .to_i
- Operators
  - +, -, *, /, %, **
- Assignment Ops.
  - =, +=, -=, *=, /=, **=

Strings

```
1 $ irb
2 irb(main):001:0> "Ruby"
3 => "Ruby"
```

Strings

```
1 $ irb
2 irb(main):001:0> "Ruby"
3 => "Ruby"
4 irb(main):002:0> 'Ruby'.reverse
5 => "ybuR"
```

Strings

```
$ irb
irb(main):001:0> "Ruby"
=> "Ruby"
irb(main):002:0> 'Ruby'.reverse
=> "ybuR"
irb(main):003:0> "RuBy".downcase
=> "ruby"
```

Strings

```
$ irb
irb(main):001:0> "Ruby"
=> "Ruby"
irb(main):002:0> 'Ruby'.reverse
=> "ybuR"
irb(main):003:0> "RuBy".downcase
=> "ruby"
irb(main):004:0> %{Ruby string}
=> "Ruby string"
```

Strings

```
$ irb
irb(main):001:0> "Ruby"
=> "Ruby"
irb(main):002:0> 'Ruby'.reverse
=> "ybuR"
irb(main):003:0> "RuBy".downcase
=> "ruby"
irb(main):004:0> %{Ruby string}
=> "Ruby string"
irb(main):005:0> "Ruby" << " string"
=> "Ruby string"
```

Strings

```
$ irb
irb(main):001:0> "Ruby"
=> "Ruby"
irb(main):002:0> 'Ruby'.reverse
=> "ybuR"
irb(main):003:0> "RuBy".downcase
=> "ruby"
irb(main):004:0> %{Ruby string}
=> "Ruby string"
irb(main):005:0> "Ruby" << " string"
=> "Ruby string"
irb(main):006:0> "Ruby" + " string"
=> "Ruby string"
```

Strings

```
$ irb
irb(main):001:0> "Ruby"
=> "Ruby"
irb(main):002:0> 'Ruby'.reverse
=> "ybuR"
irb(main):003:0> "RuBy".downcase
=> "ruby"
irb(main):004:0> %{Ruby string}
=> "Ruby string"
irb(main):005:0> "Ruby" << " string"
=> "Ruby string"
irb(main):006:0> "Ruby" + " string"
=> "Ruby string"
irb(main):007:0> n = "Ruby".size
=> 4
```

Strings

```
$ irb
irb(main):001:0> "Ruby"
=> "Ruby"
irb(main):002:0> 'Ruby'.reverse
=> "ybuR"
irb(main):003:0> "RuBy".downcase
=> "ruby"
irb(main):004:0> %{Ruby string}
=> "Ruby string"
irb(main):005:0> "Ruby" << " string"
=> "Ruby string"
irb(main):006:0> "Ruby" + " string"
=> "Ruby string"
irb(main):007:0> n = "Ruby".size
=> 4
irb(main):008:0> %{"Ruby" size is #{n}}
=> "\"Ruby\" size is 4"
```

Strings

```
$ irb
irb(main):001:0> "Ruby"
=> "Ruby"
irb(main):002:0> 'Ruby'.reverse
=> "ybuR"
irb(main):003:0> "RuBy".downcase
=> "ruby"
irb(main):004:0> %{Ruby string}
=> "Ruby string"
irb(main):005:0> "Ruby" << " string"
=> "Ruby string"
irb(main):006:0> "Ruby" + " string"
=> "Ruby string"
irb(main):007:0> n = "Ruby".size
=> 4
irb(main):008:0> %{"Ruby" size is #{n}}
=> "\"Ruby\" size is 4"
```

- String
  - .chomp
  - <=>
  - =~
  - .empty?
  - .include?
  - .index
  - .insert
  - .gsub
  - .length
  - .split
  - .to_i
  - .to_f

Symbols

```
$ irb
irb(main):001:0> str1 = "test"
=> "test"
```

Symbols

```
$ irb
irb(main):001:0> str1 = "test"
=> "test"
irb(main):002:0> str2 = "test"
=> "test"
```

Symbols

```
$ irb
irb(main):001:0> str1 = "test"
=> "test"
irb(main):002:0> str2 = "test"
=> "test"
irb(main):003:0> str1 == str2
=> true
```

Symbols

```
$ irb
irb(main):001:0> str1 = "test"
=> "test"
irb(main):002:0> str2 = "test"
=> "test"
irb(main):003:0> str1 == str2
=> true
irb(main):004:0> str1.equal? str2
=> false
```

Symbols

```
$ irb
irb(main):001:0> str1 = "test"
=> "test"
irb(main):002:0> str2 = "test"
=> "test"
irb(main):003:0> str1 == str2
=> true
irb(main):004:0> str1.equal? str2
=> false
irb(main):005:0> sym1 = :test
=> :test
```

Symbols

```
$ irb
irb(main):001:0> str1 = "test"
=> "test"
irb(main):002:0> str2 = "test"
=> "test"
irb(main):003:0> str1 == str2
=> true
irb(main):004:0> str1.equal? str2
=> false
irb(main):005:0> sym1 = :test
=> :test
irb(main):006:0> sym2 = :test
=> :test
```

Symbols

```
$ irb
irb(main):001:0> str1 = "test"
=> "test"
irb(main):002:0> str2 = "test"
=> "test"
irb(main):003:0> str1 == str2
=> true
irb(main):004:0> str1.equal? str2
=> false
irb(main):005:0> sym1 = :test
=> :test
irb(main):006:0> sym2 = :test
=> :test
irb(main):007:0> sym1.equal? sym2
=> true
```

Symbols

```
$ irb
irb(main):001:0> str1 = "test"
=> "test"
irb(main):002:0> str2 = "test"
=> "test"
irb(main):003:0> str1 == str2
=> true
irb(main):004:0> str1.equal? str2
=> false
irb(main):005:0> sym1 = :test
=> :test
irb(main):006:0> sym2 = :test
=> :test
irb(main):007:0> sym1.equal? sym2
=> true
irb(main):008:0> sym1.object_id
=> 333628
irb(main):009:0> sym2.object_id
=> 333628
```

Symbols

```
$ irb
irb(main):001:0> str1 = "test"
=> "test"
irb(main):002:0> str2 = "test"
=> "test"
irb(main):003:0> str1 == str2
=> true
irb(main):004:0> str1.equal? str2
=> false
irb(main):005:0> sym1 = :test
=> :test
irb(main):006:0> sym2 = :test
=> :test
irb(main):007:0> sym1.equal? sym2
=> true
irb(main):008:0> sym1.object_id
=> 333628
irb(main):009:0> sym2.object_id
=> 333628
```

- Symbol
  - is immutable
  - .to_s
  - .to_i
- Object
  - .object_id

Collections

- Arrays
- Ranges
- Hashes

Arrays

```
$ irb
irb(main):001:0> a = [ 1, 2, 3, 4 ]
=> [1, 2, 3, 4]
```

Arrays

```
$ irb
irb(main):001:0> a = [ 1, 2, 3, 4 ]
=> [1, 2, 3, 4]
irb(main):002:0> a = Array[1, 2, 3]
=> [1, 2, 3]
```

Arrays

```
1 $ irb
2 irb(main):001:0> a = [ 1, 2, 3, 4 ]
3 => [1, 2, 3, 4]
4 irb(main):002:0> a = Array[1, 2, 3]
5 => [1, 2, 3]
6 irb(main):003:0> a = Array.new(5)
7 => [nil, nil, nil, nil, nil]
```

Arrays

```
$ irb
irb(main):001:0> a = [ 1, 2, 3, 4 ]
=> [1, 2, 3, 4]
irb(main):002:0> a = Array[1, 2, 3]
=> [1, 2, 3]
irb(main):003:0> a = Array.new(5)
=> [nil, nil, nil, nil, nil]
irb(main):004:0> [1, 2, 4] == [1, 2, 3]
=> false
```

Arrays

```
$ irb
irb(main):001:0> a = [ 1, 2, 3, 4 ]
=> [1, 2, 3, 4]
irb(main):002:0> a = Array[1, 2, 3]
=> [1, 2, 3]
irb(main):003:0> a = Array.new(5)
=> [nil, nil, nil, nil, nil]
irb(main):004:0> [1, 2, 4] == [1, 2, 3]
=> false
irb(main):005:0> [1, 2, 4] <=> [1, 2, 3]
=> 1
```

Arrays

```
$ irb
irb(main):001:0> a = [ 1, 2, 3, 4 ]
=> [1, 2, 3, 4]
irb(main):002:0> a = Array[1, 2, 3]
=> [1, 2, 3]
irb(main):003:0> a = Array.new(5)
=> [nil, nil, nil, nil, nil]
irb(main):004:0> [1, 2, 4] == [1, 2, 3]
=> false
irb(main):005:0> [1, 2, 4] <=> [1, 2, 3]
=> 1
irb(main):006:0> a = [ 'a', 'b', 'c' ]
=> ["a", "b", "c"]
```

Arrays

```
$ irb
irb(main):001:0> a = [ 1, 2, 3, 4 ]
=> [1, 2, 3, 4]
irb(main):002:0> a = Array[1, 2, 3]
=> [1, 2, 3]
irb(main):003:0> a = Array.new(5)
=> [nil, nil, nil, nil, nil]
irb(main):004:0> [1, 2, 4] == [1, 2, 3]
=> false
irb(main):005:0> [1, 2, 4] <=> [1, 2, 3]
=> 1
irb(main):006:0> a = [ 'a', 'b', 'c' ]
=> ["a", "b", "c"]
irb(main):007:0> a[1]
=> "b"
```

Arrays

```
$ irb
irb(main):001:0> a = [ 1, 2, 3, 4 ]
=> [1, 2, 3, 4]
irb(main):002:0> a = Array[1, 2, 3]
=> [1, 2, 3]
irb(main):003:0> a = Array.new(5)
=> [nil, nil, nil, nil, nil]
irb(main):004:0> [1, 2, 4] == [1, 2, 3]
=> false
irb(main):005:0> [1, 2, 4] <=> [1, 2, 3]
=> 1
irb(main):006:0> a = [ 'a', 'b', 'c' ]
=> ["a", "b", "c"]
irb(main):007:0> a[1]
=> "b"
irb(main):008:0> a.last
=> "c"
irb(main):009:0>
```

Arrays

```
$ irb
irb(main):001:0> a = [ 1, 2, 3, 4 ]
=> [1, 2, 3, 4]
irb(main):002:0> a = Array[1, 2, 3]
=> [1, 2, 3]
irb(main):003:0> a = Array.new(5)
=> [nil, nil, nil, nil, nil]
irb(main):004:0> [1, 2, 4] == [1, 2, 3]
=> false
irb(main):005:0> [1, 2, 4] <=> [1, 2, 3]
=> 1
irb(main):006:0> a = [ 'a', 'b', 'c' ]
=> ["a", "b", "c"]
irb(main):007:0> a[1]
=> "b"
irb(main):008:0> a.last
=> "c"
irb(main):009:0>
```

- Array
  - [-1]
  - .first
  - .at
  - .clear
  - .concat
  - .delete
  - .empty?
  - .fill
  - .flatten
  - .include?

Arrays, cont.

```
$ irb
irb(main):001:0> a = [1, 2, 3]
=> [1, 2, 3]
```

Arrays, cont.

```
$ irb
irb(main):001:0> a = [1, 2, 3]
=> [1, 2, 3]
irb(main):002:0> a[2] = 4
=> 4
```

Arrays, cont.

```
$ irb
irb(main):001:0> a = [1, 2, 3]
=> [1, 2, 3]
irb(main):002:0> a[2] = 4
=> 4
irb(main):003:0> a
=> [1, 2, 4]
```

Arrays, cont.

```
$ irb
irb(main):001:0> a = [1, 2, 3]
=> [1, 2, 3]
irb(main):002:0> a[2] = 4
=> 4
irb(main):003:0> a
=> [1, 2, 4]
irb(main):004:0> a[4] = 3
=> 3
```

Arrays, cont.

```
$ irb
irb(main):001:0> a = [1, 2, 3]
=> [1, 2, 3]
irb(main):002:0> a[2] = 4
=> 4
irb(main):003:0> a
=> [1, 2, 4]
irb(main):004:0> a[4] = 3
=> 3
irb(main):005:0> a
=> [1, 2, 4, nil, 3]
```

Arrays, cont.

```
$ irb
irb(main):001:0> a = [1, 2, 3]
=> [1, 2, 3]
irb(main):002:0> a[2] = 4
=> 4
irb(main):003:0> a
=> [1, 2, 4]
irb(main):004:0> a[4] = 3
=> 3
irb(main):005:0> a
=> [1, 2, 4, nil, 3]
irb(main):006:0> a.compact.sort
=> [1, 2, 3, 4]
```

Arrays, cont.

```
$ irb
irb(main):001:0> a = [1, 2, 3]
=> [1, 2, 3]
irb(main):002:0> a[2] = 4
=> 4
irb(main):003:0> a
=> [1, 2, 4]
irb(main):004:0> a[4] = 3
=> 3
irb(main):005:0> a
=> [1, 2, 4, nil, 3]
irb(main):006:0> a.compact.sort
=> [1, 2, 3, 4]
irb(main):007:0> a << 5
=> [1, 2, 3, 4, 5]
```

Arrays, cont.

```
$ irb
irb(main):001:0> a = [1, 2, 3]
=> [1, 2, 3]
irb(main):002:0> a[2] = 4
=> 4
irb(main):003:0> a
=> [1, 2, 4]
irb(main):004:0> a[4] = 3
=> 3
irb(main):005:0> a
=> [1, 2, 4, nil, 3]
irb(main):006:0> a.compact.sort
=> [1, 2, 3, 4]
irb(main):007:0> a << 5
=> [1, 2, 3, 4, 5]
irb(main):007:0> a << "foo"
=> [1, 2, 4, nil, 3, "foo"]
```

Arrays, cont.

```
$ irb
irb(main):001:0> a = [1, 2, 3]
=> [1, 2, 3]
irb(main):002:0> a[2] = 4
=> 4
irb(main):003:0> a
=> [1, 2, 4]
irb(main):004:0> a[4] = 3
=> 3
irb(main):005:0> a
=> [1, 2, 4, nil, 3]
irb(main):006:0> a.compact.sort
=> [1, 2, 3, 4]
irb(main):007:0> a << 5
=> [1, 2, 3, 4, 5]
irb(main):007:0> a << "foo"
=> [1, 2, 4, nil, 3, "foo"]
```

- Array
  - .max
  - .min
  - .reverse
  - .each
  - .map
  - .reject

as stacks and as queues

```
$ irb
irb(main):001:0> a = [1, 2, 3, 4]
=> [1, 2, 3, 4]
```

as stacks and as queues

```
1 $ irb
2 irb(main):001:0> a = [1, 2, 3, 4]
3 => [1, 2, 3, 4]
4 irb(main):002:0> a.push 5
5 => [1, 2, 3, 4, 5]
```

as stacks and as queues

```
1 $ irb
2 irb(main):001:0> a = [1, 2, 3, 4]
3 => [1, 2, 3, 4]
4 irb(main):002:0> a.push 5
5 => [1, 2, 3, 4, 5]
6 irb(main):003:0> a.pop
7 => 5
8 irb(main):004:0> a
9 => [1, 2, 3, 4]
```

as stacks and as queues

```
1  $ irb
2  irb(main):001:0> a = [1, 2, 3, 4]
3  => [1, 2, 3, 4]
4  irb(main):002:0> a.push 5
5  => [1, 2, 3, 4, 5]
6  irb(main):003:0> a.pop
7  => 5
8  irb(main):004:0> a
9  => [1, 2, 3, 4]
10 irb(main):005:0> a.shift
11 => 1
12 irb(main):006:0> a
13 => [2, 3, 4]
```

as stacks and as queues

```
1  $ irb
2  irb(main):001:0> a = [1, 2, 3, 4]
3  => [1, 2, 3, 4]
4  irb(main):002:0> a.push 5
5  => [1, 2, 3, 4, 5]
6  irb(main):003:0> a.pop
7  => 5
8  irb(main):004:0> a
9  => [1, 2, 3, 4]
10 irb(main):005:0> a.shift
11 => 1
12 irb(main):006:0> a
13 => [2, 3, 4]
14 irb(main):007:0> a.unshift 1
15 => [1, 2, 3, 4]
```

as stacks and as queues

```
1  $ irb
2  irb(main):001:0> a = [1, 2, 3, 4]
3  => [1, 2, 3, 4]
4  irb(main):002:0> a.push 5
5  => [1, 2, 3, 4, 5]
6  irb(main):003:0> a.pop
7  => 5
8  irb(main):004:0> a
9  => [1, 2, 3, 4]
10 irb(main):005:0> a.shift
11 => 1
12 irb(main):006:0> a
13 => [2, 3, 4]
14 irb(main):007:0> a.unshift 1
15 => [1, 2, 3, 4]
```

- Array
  - stack (LIFO)
    - push, pop
    - unshift, shift
  - queue (FIFO)
    - push, shift
    - unshift, pop

as sets

```
$ irb
irb(main):001:0> a = [ 0, 1, 3 ]
=> [0, 1, 3]
```

as sets

```
1 $ irb
2 irb(main):001:0> a = [ 0, 1, 3 ]
3 => [0, 1, 3]
4 irb(main):002:0> b = [ 0, 2, 4 ]
5 => [0, 2, 4]
```

as sets

```
$ irb
irb(main):001:0> a = [ 0, 1, 3 ]
=> [0, 1, 3]
irb(main):002:0> b = [ 0, 2, 4 ]
=> [0, 2, 4]
irb(main):003:0> a + b
=> [0, 1, 3, 0, 2, 4]
```

as sets

```
$ irb
irb(main):001:0> a = [ 0, 1, 3 ]
=> [0, 1, 3]
irb(main):002:0> b = [ 0, 2, 4 ]
=> [0, 2, 4]
irb(main):003:0> a + b
=> [0, 1, 3, 0, 2, 4]
irb(main):004:0> a & b
=> [0]
```

as sets

```
1  $ irb
2  irb(main):001:0> a = [ 0, 1, 3 ]
3  => [0, 1, 3]
4  irb(main):002:0> b = [ 0, 2, 4 ]
5  => [0, 2, 4]
6  irb(main):003:0> a + b
7  => [0, 1, 3, 0, 2, 4]
8  irb(main):004:0> a & b
9  => [0]
10 irb(main):005:0> a | b
11 => [0, 1, 3, 2, 4]
```

as sets

```
$ irb
irb(main):001:0> a = [ 0, 1, 3 ]
=> [0, 1, 3]
irb(main):002:0> b = [ 0, 2, 4 ]
=> [0, 2, 4]
irb(main):003:0> a + b
=> [0, 1, 3, 0, 2, 4]
irb(main):004:0> a & b
=> [0]
irb(main):005:0> a | b
=> [0, 1, 3, 2, 4]
irb(main):006:0> a - b
=> [1, 3]
```

as sets

```
$ irb
irb(main):001:0> a = [ 0, 1, 3 ]
=> [0, 1, 3]
irb(main):002:0> b = [ 0, 2, 4 ]
=> [0, 2, 4]
irb(main):003:0> a + b
=> [0, 1, 3, 0, 2, 4]
irb(main):004:0> a & b
=> [0]
irb(main):005:0> a | b
=> [0, 1, 3, 2, 4]
irb(main):006:0> a - b
=> [1, 3]
irb(main):007:0> (a + b).sort
=> [0, 0, 1, 2, 3, 4]
```

as sets

```
1  $ irb
2  irb(main):001:0> a = [ 0, 1, 3 ]
3  => [0, 1, 3]
4  irb(main):002:0> b = [ 0, 2, 4 ]
5  => [0, 2, 4]
6  irb(main):003:0> a + b
7  => [0, 1, 3, 0, 2, 4]
8  irb(main):004:0> a & b
9  => [0]
10 irb(main):005:0> a | b
11 => [0, 1, 3, 2, 4]
12 irb(main):006:0> a - b
13 => [1, 3]
14 irb(main):007:0> (a + b).sort
15 => [0, 0, 1, 2, 3, 4]
```

- Array
  - append
  - intersection
  - union / add
  - complement / subtract

Ranges

```
1 $ irb
2 irb(main):001:0> (1..3)
3 => 1..3
```

Ranges

```
1 $ irb
2 irb(main):001:0> (1..3)
3 => 1..3
4 irb(main):002:0> (1..3).class
5 => Range
```

Ranges

```
1 $ irb
2 irb(main):001:0> (1..3)
3 => 1..3
4 irb(main):002:0> (1..3).class
5 => Range
6 irb(main):003:0> (1..3).to_a
7 => [1, 2, 3]
```

Ranges

```
1 $ irb
2 irb(main):001:0> (1..3)
3 => 1..3
4 irb(main):002:0> (1..3).class
5 => Range
6 irb(main):003:0> (1..3).to_a
7 => [1, 2, 3]
8 irb(main):004:0> (1...3).to_a
9 => [1, 2]
```

Ranges

```
$ irb
irb(main):001:0> (1..3)
=> 1..3
irb(main):002:0> (1..3).class
=> Range
irb(main):003:0> (1..3).to_a
=> [1, 2, 3]
irb(main):004:0> (1...3).to_a
=> [1, 2]
irb(main):005:0> ('ab'..'ad').to_a
=> ["ab", "ac", "ad"]
```

Ranges

```
1  $ irb
2  irb(main):001:0> (1..3)
3  => 1..3
4  irb(main):002:0> (1..3).class
5  => Range
6  irb(main):003:0> (1..3).to_a
7  => [1, 2, 3]
8  irb(main):004:0> (1...3).to_a
9  => [1, 2]
10 irb(main):005:0> ('ab'..'ad').to_a
11 => ["ab", "ac", "ad"]
```

- Range
  - .max
  - .min
  - .each
  - .map
  - .reject

Hashes

```
1 $ irb
2 irb(main):001:0> h1 = { :key => 'value' }
3 => {:key=>"value"}
```

Hashes

```
1 $ irb
2 irb(main):001:0> h1 = { :key => 'value' }
3 => {:key=>"value"}
4 irb(main):002:0> h2 = { key: 'value' }
5 => {:key=>"value"}
```

Hashes

```
$ irb
irb(main):001:0> h1 = { :key => 'value' }
=> {:key=>"value"}
irb(main):002:0> h2 = { key: 'value' }
=> {:key=>"value"}
irb(main):003:0> h1[h2] = "key is object!"
=> "key is object!"
irb(main):004:0> h1
=> {:key=>"value", {:key=>"value"}=>"key is object!"}
```

Hashes

```
1  $ irb
2  irb(main):001:0> h1 = { :key => 'value' }
3  => {:key=>"value"}
4  irb(main):002:0> h2 = { key: 'value' }
5  => {:key=>"value"}
6  irb(main):003:0> h1[h2] = "key is object!"
7  => "key is object!"
8  irb(main):004:0> h1
9  => {:key=>"value", {:key=>"value"}=>"key ↩
       is object!"}
10 irb(main):005:0> h1.delete(h2)
11 => "key is object!"
12 irb(main):006:0> h1
13 => {"key"=>"value"}
```

Hashes

```
$ irb
irb(main):001:0> h1 = { :key => 'value' }
=> {:key=>"value"}
irb(main):002:0> h2 = { key: 'value' }
=> {:key=>"value"}
irb(main):003:0> h1[h2] = "key is object!"
=> "key is object!"
irb(main):004:0> h1
=> {:key=>"value", {:key=>"value"}=>"key is object!"}
irb(main):005:0> h1.delete(h2)
=> "key is object!"
irb(main):006:0> h1
=> {"key"=>"value"}
irb(main):007:0> h1.size
=> 1
```

Hashes

```
$ irb
irb(main):001:0> h1 = { :key => 'value' }
=> {:key=>"value"}
irb(main):002:0> h2 = { key: 'value' }
=> {:key=>"value"}
irb(main):003:0> h1[h2] = "key is object!"
=> "key is object!"
irb(main):004:0> h1
=> {:key=>"value", {:key=>"value"}=>"key is object!"}
irb(main):005:0> h1.delete(h2)
=> "key is object!"
irb(main):006:0> h1
=> {"key"=>"value"}
irb(main):007:0> h1.size
=> 1
irb(main):008:0> h1.keys
=> ["key"]
```

Hashes

```
$ irb
irb(main):001:0> h1 = { :key => 'value' }
=> {:key=>"value"}
irb(main):002:0> h2 = { key: 'value' }
=> {:key=>"value"}
irb(main):003:0> h1[h2] = "key is object!"
=> "key is object!"
irb(main):004:0> h1
=> {:key=>"value", {:key=>"value"}=>"key is object!"}
irb(main):005:0> h1.delete(h2)
=> "key is object!"
irb(main):006:0> h1
=> {"key"=>"value"}
irb(main):007:0> h1.size
=> 1
irb(main):008:0> h1.keys
=> ["key"]
irb(main):009:0> h1.values
=> ["value"]
```

Hashes

```
$ irb
irb(main):001:0> h1 = { :key => 'value' }
=> {:key=>"value"}
irb(main):002:0> h2 = { key: 'value' }
=> {:key=>"value"}
irb(main):003:0> h1[h2] = "key is object!"
=> "key is object!"
irb(main):004:0> h1
=> {:key=>"value", {:key=>"value"}=>"key is object!"}
irb(main):005:0> h1.delete(h2)
=> "key is object!"
irb(main):006:0> h1
=> {"key"=>"value"}
irb(main):007:0> h1.size
=> 1
irb(main):008:0> h1.keys
=> ["key"]
irb(main):009:0> h1.values
=> ["value"]
```

- Hash
  - v. 1.8
  - v. 1.9
  - .clear
  - .length
  - .empty?
  - .key?
  - .value?
  - .index
  - .store
  - .to_a
  - .to_s
  - .sort

Structure

- Methods
- Blocks
- Procs and Lambdas

Methods

```
1  def power( a, n=1 )
2    a ** n
3  end
4
5  def divide( a, b )
6    return a / b, a % b
7  end
8
9  puts power 5, 2
10 puts power( 3 )
11
12 d, m = divide 5, 2
13 puts "Result is #{d} and modulo is: #{m}"
```

Methods

```ruby
def power( a, n=1 )
  a ** n
end

def divide( a, b )
  return a / b, a % b
end

puts power 5, 2
puts power( 3 )

d, m = divide 5, 2
puts "Result is #{d} and modulo is: #{m}"
```

```
$ ruby power_method.rb
25
3
Result is 2 and modulo is: 1
```

Splat operator and variable parameters

```
$ irb
1.9.3-p362 :001 > first, *center, last = [1, 2, 3, 4, 5]
 => [1, 2, 3, 4, 5]
```

Splat operator and variable parameters

```
$ irb
1.9.3-p362 :001 > first, *center, last = [1, 2, 3, 4, 5]
 => [1, 2, 3, 4, 5]
1.9.3-p362 :002 > first
 => 1
```

Splat operator and variable parameters

```
$ irb
1.9.3-p362 :001 > first, *center, last = [1, 2, 3, 4, 5]
 => [1, 2, 3, 4, 5]
1.9.3-p362 :002 > first
 => 1
1.9.3-p362 :002 > center
 => [2, 3, 4]
```

Splat operator and variable parameters

```
$ irb
1.9.3-p362 :001 > first, *center, last = [1, 2, 3, 4, 5]
 => [1, 2, 3, 4, 5]
1.9.3-p362 :002 > first
 => 1
1.9.3-p362 :002 > center
 => [2, 3, 4]
```

```
def say( what, *people )
  people.each { |person| puts "#{what} #{person}!" }
end

say "Hi", "Kostas", "Alex"
```

Splat operator and variable parameters

```
$ irb
1.9.3-p362 :001 > first, *center, last = [1, 2, 3, 4, 5]
 => [1, 2, 3, 4, 5]
1.9.3-p362 :002 > first
 => 1
1.9.3-p362 :002 > center
 => [2, 3, 4]
```

```
def say( what, *people )
  people.each { |person| puts "#{what} #{person}!" }
end

say "Hi", "Kostas", "Alex"
```

```
$ ruby methods_params.rb
Hi Kostas!
Hi Alex!
```

Blocks

- is simply a chunk of code between curly brackets { and } or the keywords **do** and **end**
- it can take parameters after the start of the block between vertical bars
- a block can be passed to a method as the last parameter

Blocks

- is simply a chunk of code between curly brackets { and } or the keywords **do** and **end**
- it can take parameters after the start of the block between vertical bars
- a block can be passed to a method as the last parameter

```
1 [1, 2, 3].each { |n| print n, " " }
2
3 # will output: 1 2 3
```

Blocks

- is simply a chunk of code between curly brackets { and } or the keywords **do** and **end**
- it can take parameters after the start of the block between vertical bars
- a block can be passed to a method as the last parameter

```ruby
[1, 2, 3].each { |n| print n, " " }

# will output: 1 2 3
```

```ruby
n = "foo"
[1, 2, 3].each { |n| print n, " " }
print n

# v.1.9 - will output: 1 2 3 foo
```

Method that takes a block as a parameter

```
1 def test
2     puts "You are in the method"
3     yield
4     puts "You are again back to the method"
5 end
6
7 test { puts "You are in the block" }
```

Method that takes a block as a parameter

```
1 def test
2     puts "You are in the method"
3     yield
4     puts "You are again back to the method"
5 end
6
7 test { puts "You are in the block" }
```

```
1 $ ruby test.rb
2 You are in the method
3 You are in the block
4 You are again back to the method
```

Block / yield can also take parameters

```
1 def test
2     yield 5
3     puts "You are in the method test"
4     yield 100
5 end
6 test { |i| puts "You are in the block with value #{i}" }
```

Block / yield can also take parameters

```ruby
def test
    yield 5
    puts "You are in the method test"
    yield 100
end
test { |i| puts "You are in the block with value #{i}" }
```

```
$ ruby test.rb
You are in the block with value 5
You are in the method test
You are in the block with value 100
```

Explicitly binding a block to a method

```
1 def test( &block )
2       puts "You are in the method"
3       block.call
4       puts "Back to the method and block is of class #{block.class}"
5 end
6
7 test { puts "You are in the block" }
```

Explicitly binding a block to a method

```
1 def test( &block )
2       puts "You are in the method"
3       block.call
4       puts "Back to the method and block is of class #{block.class}"
5 end
6
7 test { puts "You are in the block" }
```

```
1 $ ruby test.rb
2 You are in the method
3 You are in the block
4 Back to the method and block is of class Proc
```

Procs

- a block is a `Proc`
- a `Proc` is bound to a variable explicitly
  - a block is not
- multiple `Proc` objects can be passed to a method
  - only one block can be passed to a method

Procs

- a block is a Proc
- a Proc is bound to a variable explicitly
  - a block is not
- multiple Proc objects can be passed to a method
  - only one block can be passed to a method

```
1 def test_blocks(some_proc)
2   puts some_proc.call
3 end
4
5 some_new_proc = Proc.new { puts "in the Proc !" }
6 test_blocks(some_new_proc)
7
8 # Output:
9 # in the Proc !
```

Lambdas

- a `lambda` is a `Proc`
- a `lambda` checks the number of parameters passed to it
  - a `Proc` does not

Lambdas

- a `lambda` is a `Proc`
- a `lambda` checks the number of parameters passed to it
  - a `Proc` does not

```
1  def test_parameter_handling(code)
2    code.call(1,2)
3  end
4
5  l = lambda {|a,b,c| puts "#{a} is a #{a.class}, #{b} is a #{b.↩
       class}, #{c} is a #{c.class}" }
6  p = Proc.new {|a,b,c| puts "#{a} is a #{a.class}, #{b} is a #{b.↩
       class}, #{c} is a #{c.class}" }
7
8  test_parameter_handling (p)
9  test_parameter_handling (l)
10
11 # Output:
12 # 1 is a Fixnum, 2 is a Fixnum,  is a NilClass
13 # ArgumentError: wrong number of arguments (2 for 3)
```

Lambdas, cont.

- a `lambda` is a `Proc`
- a `return` keyword inside a `lambda` returns to the enclosing method
  - a `return` keyword inside `Proc` returns outside the enclosing method

Lambdas, cont.

- a `lambda` is a `Proc`
- a `return` keyword inside a `lambda` returns to the enclosing method
  - a `return` keyword inside `Proc` returns outside the enclosing method

```
1  def my_method
2    puts "before proc"
3    my_proc = lambda do
4      puts "inside proc"
5      return
6    end
7    my_proc.call
8    puts "after proc"
9  end
10
11 my_method
12 # Output:
13 # before proc
14 # inside proc
15 # after proc
```

Lambdas, cont.

- a `lambda` is a `Proc`
- a `return` keyword inside a `lambda` returns to the enclosing method
  - a `return` keyword inside `Proc` returns outside the enclosing method

```ruby
def my_method
  puts "before proc"
  my_proc = lambda do
    puts "inside proc"
    return
  end
  my_proc.call
  puts "after proc"
end

my_method
# Output:
# before proc
# inside proc
# after proc
```

```ruby
def my_method
  puts "before proc"
    my_proc = Proc.new do
      puts "inside proc"
      return
    end
    my_proc.call
    puts "after proc"
end

my_method

# Output:
# before proc
# inside proc
```

Control Flow

- Conditionals
- Loops
- Iterators
- Exceptions

if and unless

```
1  x = 1
2  if( x > 1 )
3    puts "x > 1"
4  elsif x < 1
5    puts "x < 1"
6  else
7    puts "x is 1"
8  end
9  # Output:
10 # x is 1
```

if and unless

```ruby
x = 1
if( x > 1 )
  puts "x > 1"
elsif x < 1
  puts "x < 1"
else
  puts "x is 1"
end
# Output:
# x is 1
```

```ruby
str = ""
puts "Empty" if str.empty?

# Output:
# Empty
```

if and unless

```ruby
x = 1
if( x > 1 )
  puts "x > 1"
elsif x < 1
  puts "x < 1"
else
  puts "x is 1"
end
# Output:
# x is 1
```

```ruby
x = 1
unless x == 1
  puts "x is not 1"
else
  puts "x is 1"
end

# Output:
# x is 1
```

```ruby
str = ""
puts "Empty" if str.empty?

# Output:
# Empty
```

if and unless

```ruby
x = 1
if( x > 1 )
  puts "x > 1"
elsif x < 1
  puts "x < 1"
else
  puts "x is 1"
end
# Output:
# x is 1
```

```ruby
x = 1
unless x == 1
  puts "x is not 1"
else
  puts "x is 1"
end

# Output:
# x is 1
```

```ruby
str = ""
puts "Empty" if str.empty?

# Output:
# Empty
```

```ruby
url = nil
url = "default" unless url
puts url

# Output:
# default
```

case and conditional operators

```ruby
age = 5
case age
when 0 .. 2
    puts "baby"
when 3 .. 6
    puts "little child"
when 7 .. 12
    puts "child"
when 13 .. 18
    puts "youth"
else
    puts "adult"
end

# Output:
# little child
```

case and conditional operators

```ruby
age = 5
case age
when 0 .. 2
    puts "baby"
when 3 .. 6
    puts "little child"
when 7 .. 12
    puts "child"
when 13 .. 18
    puts "youth"
else
    puts "adult"
end

# Output:
# little child
```

```ruby
str = "test"
if str[1] == 'e' && str.size > 3
  puts "both true"
end

# Output:
# both true
```

case and conditional operators

```ruby
age = 5
case age
when 0 .. 2
    puts "baby"
when 3 .. 6
    puts "little child"
when 7 .. 12
    puts "child"
when 13 .. 18
    puts "youth"
else
    puts "adult"
end

# Output:
# little child
```

```ruby
str = "test"
if str[1] == 'e' && str.size > 3
  puts "both true"
end

# Output:
# both true
```

- ==, ===, != or <>, !==, >=, <=, >, <
- && (and), || (or), ! (not)
- Ternary: a ? b : c

for, while and until

```ruby
for i in [1, 2] do
  print i, " "
end

for s in ["one", "two"] do
  print s, " "
end

for i in (1..2) do
  print i, " "
end

# Output:
# 1 2
# one two
# 1 2
```

for, while and until

```ruby
for i in [1, 2] do
  print i, " "
end

for s in ["one", "two"] do
  print s, " "
end

for i in (1..2) do
  print i, " "
end

# Output:
# 1 2
# one two
# 1 2
```

```ruby
x = 0
while x < 2
  x += 1          # not x.succ
  print x, " "
end

while x == 2 do puts "x is now 3"; x += 1 end

begin
  puts "at least once"
end while x < 0

until x == 3 do puts(x) end

# Output:
# 1 2 x is now 3
# at least once
```

Iterators

```ruby
[1, 2].each do |i|
  print i, " "
end

(3..4).each { |i| print i, " " }

5.upto(6) { |i| print i, " " }

2.times { print "7 " }

(8..10).step(2) { |i| print i.to_s + " "  }

11.step(15, 2) { |i| print "#{i} "}

# Output:
# 1 2 3 4 5 6 7 7 8 10 11 13 15
```

Iterators, cont.

```ruby
numbers = (1..10)

puts numbers.include?(1)

even = []
numbers.each { |n| even << n if n.even? }

odd = numbers.select { |n| n.odd? }

even_squares = even.map { |n| n ** 2 }

odd_squares = numbers.reject { |n| n.even? }.map! { |n| n ** 2 }

puts odd_squares.to_s

# Output:
# true
# [1, 9, 25, 49, 81]
```

Exception Handling

- predictability
  - file existence vs file deleted while reading
- `raise` an exception when you identify an external problem:
  - the server send invalid data, or
  - out of disk space
- ...that the user needs to handle, or
  - sometimes the program can handle

Exception Handling, cont.

```ruby
def calc( val1, val2 )
  begin
    result = val1 / val2
  rescue TypeError, NoMethodError => e
    puts( "#{e.class}: #{e}" )
    puts( "One of the values is not a number!" )
    result = nil
  rescue Exception => e
    puts( "#{e.class}: #{e}" )
    result = nil
  end
  return result
end
```

More Structure

- Classes
- Classes: Behind the Scenes
- Modules
- Files

Introduction

```
1  class Thing
2    def name
3      @name
4    end
5
6    def name=( str )
7      @name = str
8    end
9  end
10
11 t = Thing.new
12 t.name = "Sword"
13
14 puts t                    #<Thing:0x000000012fd0e0>
15 puts t.inspect            #<Thing:0x000000012fd0e0 @name="Sword">
```

Introduction

```
1  class Thing
2    def name
3      @name
4    end
5
6    def name=( str )
7      @name = str
8    end
9  end
10
11 t = Thing.new
12 t.name = "Sword"
13
14 puts t                    #<Thing:0x000000012fd0e0>
15 puts t.inspect            #<Thing:0x000000012fd0e0 @name="Sword">
```

- getters/setters
- instance variables
- default constructor

Introduction, cont.

```ruby
class Thing
  attr_accessor :name, :description

  def initialize options={}
    @name = options[:name]
    @description = options[:description] || "No description"
  end
end

sword = Thing.new :name => "Sword",
                  :description => "A longsword."

bow = Thing.new :name => "Bow"

p sword  #<Thing:0x.. @name="Sword", @description="A longsword.">
p bow    #<Thing:0x.. @name="Bow", @description="No description">
```

Introduction, cont.

```
1  class Thing
2    attr_accessor :name, :description
3
4    def initialize options={}
5      @name = options[:name]
6      @description = options[:description] || "No description"
7    end
8  end
9
10 sword = Thing.new :name => "Sword",
11                  :description => "A longsword."
12
13 bow = Thing.new :name => "Bow"
14
15 p sword  #<Thing:0x.. @name="Sword", @description="A longsword.">
16 p bow    #<Thing:0x.. @name="Bow", @description="No description">
```

- attr_accessor, attr_reader and attr_writer
- constructors, parameters and default values

Inheritance

```ruby
class Thing
  attr_accessor :name, :description

  def initialize( options={} )
    default = { :name => "A thing",
                :description => "No description" }
    options = default.merge(options)

    @name = options[:name]
    @description = options[:description]
  end
end
```

- proper method argument defaults

Inheritance, cont.

```
14 class Weapon < Thing
15   attr_accessor :damage
16
17   def initialize( options={} )
18     default = { :damage => 0 }
19     options = default.merge(options)
20
21     @damage = options[:damage]
22     super(options)
23   end
24 end
25
26 p Weapon.new :name => "Sword", :damage => 10
```

- single class inheritance
- super

Duck Typing

- Semantics are determined by an object's methods
  - ...and not by its inheritance
  - ...or interfaces for other languages

Duck Typing

- Semantics are determined by an object's methods
  - ...and not by its inheritance
  - ...or interfaces for other languages

"If it walks like a duck, and quacks like a duck, it is a duck."

Duck Typing

- Semantics are determined by an object's methods
  - ...and not by its inheritance
  - ...or interfaces for other languages

"If it walks like a duck, and quacks like a duck, it is a duck."

```
1  class Duck
2    def quack
3      "quack!"
4    end
5  end
6
7  class Person
8    def quack
9      "make quack sound"
10   end
11 end
12
13 d = Duck.new
14 p = Person.new
15
16 [d, p].each do |obj|
17   puts obj.quack
18 end
```

Access Control

```ruby
class MyClass
  def method_one
  end

  protected
  def method_two
  end

  private
  def method_three
  end

  public
  def method_four
  end
end
```

Access Control

```ruby
class MyClass
  def method_one
  end

  protected
  def method_two
  end

  private
  def method_three
  end

  public
  def method_four
  end
end
```

```ruby
class MyClass
  def method_one
  end

  def method_two
  end

  def method_three
  end

  def method_four
  end

  public    :method_one,
            :method_four
  protected :method_two
  private   :method_three
end
```

Access Control, cont.

- public
  - can be accessed by any object
- protected
  - can be accessed from inside the defining class and its subclasses
- private
  - can be accessed only from inside the defining class

Class variables, methods and inheritance

```ruby
class Thing
  @@num_of_things = 0

  def initialize
    @@num_of_things += 1
  end

  def self.num_of_things
    @@num_of_things
  end
end

t1 = Thing.new
t2 = Thing.new

puts Thing.num_of_things
```

Class variables, methods and inheritance

```ruby
class Thing
  @@num_of_things = 0

  def initialize
    @@num_of_things += 1
  end

  def self.num_of_things
    @@num_of_things
  end
end

t1 = Thing.new
t2 = Thing.new

puts Thing.num_of_things
```

```ruby
class Treasure < Thing
  @@num_of_things = 0

  def initialize
    @@num_of_things += 1
  end

  def self.num_of_things
    @@num_of_things
  end
end

tr1 = Treasure.new
puts Treasure.num_of_things
puts Thing.num_of_things
```

Constants and scope resolution operator

```ruby
class Circle
  PI = 3.1415926535

  def initialize( radius )
    @radius = radius
  end

  def circumference
    @radius * 2 * PI
  end
end

c = Circle.new( 5 )
printf( "Circumference is %.2f\n", c.circumference )

puts "PI is #{Circle::PI}"

# Circumference is 31.42
# PI is 3.1415926535
```

Partial Classes

```
1  class Dog
2    def bark
3      puts( "woof" )
4    end
5  end
6
7  class Dog
8    def bite
9      puts( "yum" )
10   end
11 end
```

Partial Classes

```
1  class Dog
2    def bark
3      puts( "woof" )
4    end
5  end
6
7  class Dog
8    def bite
9      puts( "yum" )
10   end
11 end
```

- classes can be reopened
- ...and new methods added
- this applies to all classes!

Introduction

- Everything is an object
- Every object has a class

```
1 class Person
2   def talk
3   end
4 end
```

Method lookup path: object

```
irb(main):005:0> aragorn = Person.new
=> #<Person:0x0000000285c3a0>
```

Method lookup path: object

```
irb(main):005:0> aragorn = Person.new
=> #<Person:0x0000000285c3a0>
irb(main):006:0> aragorn.class
=> Person
```

Method lookup path: object

```
irb(main):005:0> aragorn = Person.new
=> #<Person:0x0000000285c3a0>
irb(main):006:0> aragorn.class
=> Person
irb(main):007:0> Person.superclass
=> Object
```

Method lookup path: object

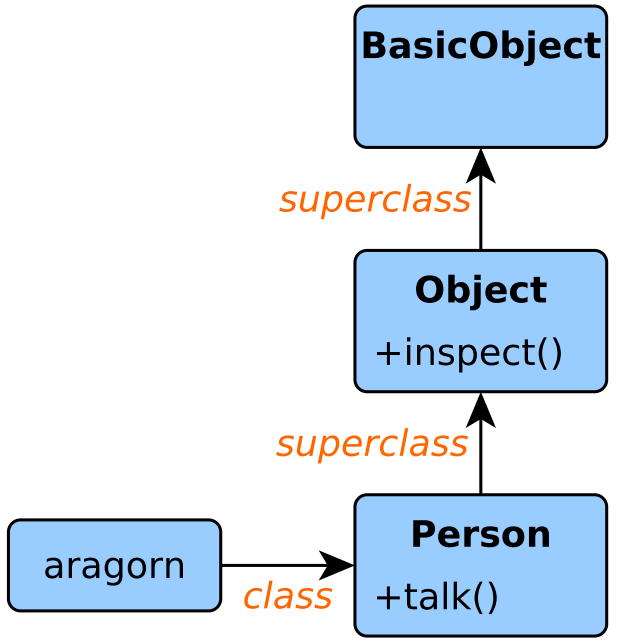
```
irb(main):005:0> aragorn = Person.new
=> #<Person:0x0000000285c3a0>
irb(main):006:0> aragorn.class
=> Person
irb(main):007:0> Person.superclass
=> Object
irb(main):008:0> Object.superclass
=> BasicObject
```

Method lookup path: object

```
1  irb(main):005:0> aragorn = Person.new
2  => #<Person:0x0000000285c3a0>
3  irb(main):006:0> aragorn.class
4  => Person
5  irb(main):007:0> Person.superclass
6  => Object
7  irb(main):008:0> Object.superclass
8  => BasicObject
9  irb(main):009:0> BasicObject.superclass
10 => nil
```

Method lookup path: object

```
1  irb(main):005:0> aragorn = Person.new
2  => #<Person:0x0000000285c3a0>
3  irb(main):006:0> aragorn.class
4  => Person
5  irb(main):007:0> Person.superclass
6  => Object
7  irb(main):008:0> Object.superclass
8  => BasicObject
9  irb(main):009:0> BasicObject.superclass
10 => nil
```



Method lookup path: class object

- A class is also an object!

Method lookup path: class object

- A class is also an object!

```
irb(main):005:0> aragorn = Person.new
=> #<Person:0x0000000285c3a0>
irb(main):006:0> aragorn.class
=> Person
```

Method lookup path: class object

- A class is also an object!

```
irb(main):005:0> aragorn = Person.new
=> #<Person:0x0000000285c3a0>
irb(main):006:0> aragorn.class
=> Person
irb(main):007:0> Person.class
=> Class
```

Method lookup path: class object

- A class is also an object!

```
1 irb(main):005:0> aragorn = Person.new
2 => #<Person:0x0000000285c3a0>
3 irb(main):006:0> aragorn.class
4 => Person
5 irb(main):007:0> Person.class
6 => Class
7 irb(main):008:0> Class.superclass
8 => Module
```

Method lookup path: class object

- A class is also an object!

```
1  irb(main):005:0> aragorn = Person.new
2  => #<Person:0x0000000285c3a0>
3  irb(main):006:0> aragorn.class
4  => Person
5  irb(main):007:0> Person.class
6  => Class
7  irb(main):008:0> Class.superclass
8  => Module
9  irb(main):009:0> Module.superclass
10 => Object
```

Method lookup path: class object

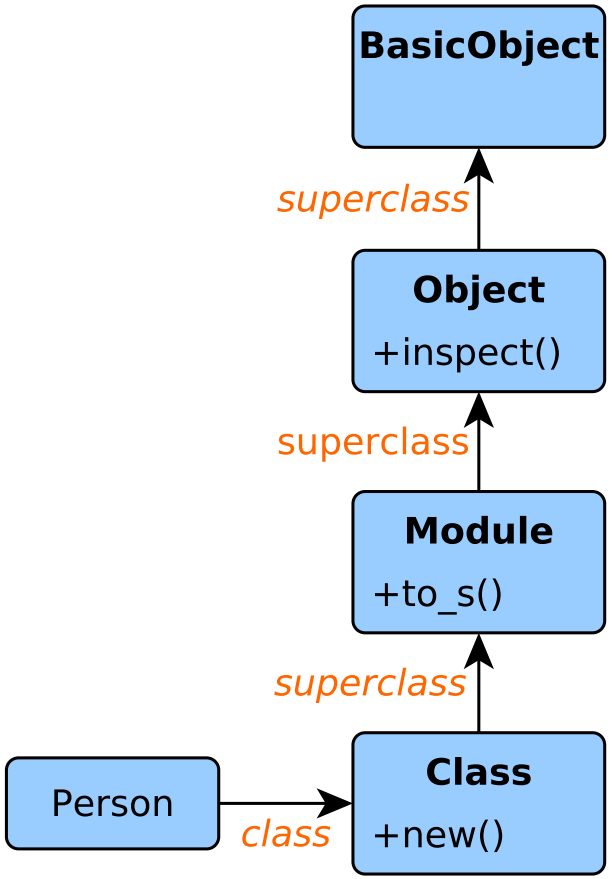
- A class is also an object!

```
1  irb(main):005:0> aragorn = Person.new
2  => #<Person:0x0000000285c3a0>
3  irb(main):006:0> aragorn.class
4  => Person
5  irb(main):007:0> Person.class
6  => Class
7  irb(main):008:0> Class.superclass
8  => Module
9  irb(main):009:0> Module.superclass
10 => Object
11 irb(main):010:0> Object.superclass
12 => BasicObject
```

Method lookup path: class object

- A class is also an object!

```
1  irb(main):005:0> aragorn = Person.new
2  => #<Person:0x0000000285c3a0>
3  irb(main):006:0> aragorn.class
4  => Person
5  irb(main):007:0> Person.class
6  => Class
7  irb(main):008:0> Class.superclass
8  => Module
9  irb(main):009:0> Module.superclass
10 => Object
11 irb(main):010:0> Object.superclass
12 => BasicObject
```



Singleton Methods

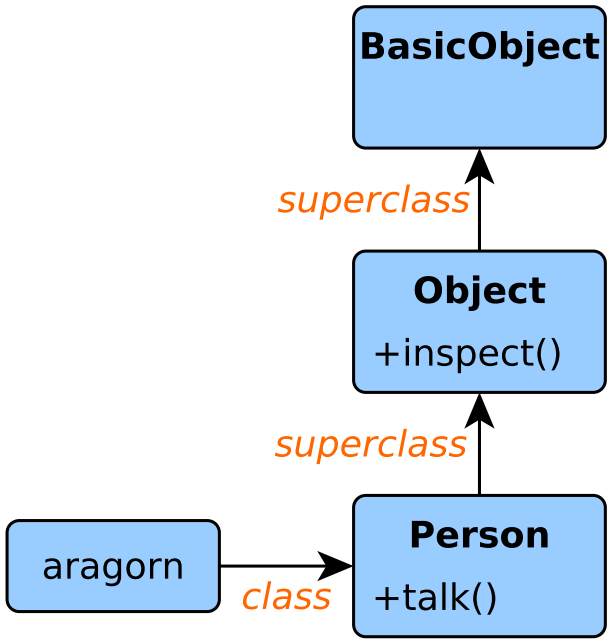
- A class defines the behaviour of their instances
- Behaviour of person instances is placed in the Person class
- Ruby allows unique behaviour to individual objects!

Singleton Methods

- A class defines the behaviour of their instances
- Behaviour of person instances is placed in the Person class
- Ruby allows unique behaviour to individual objects!

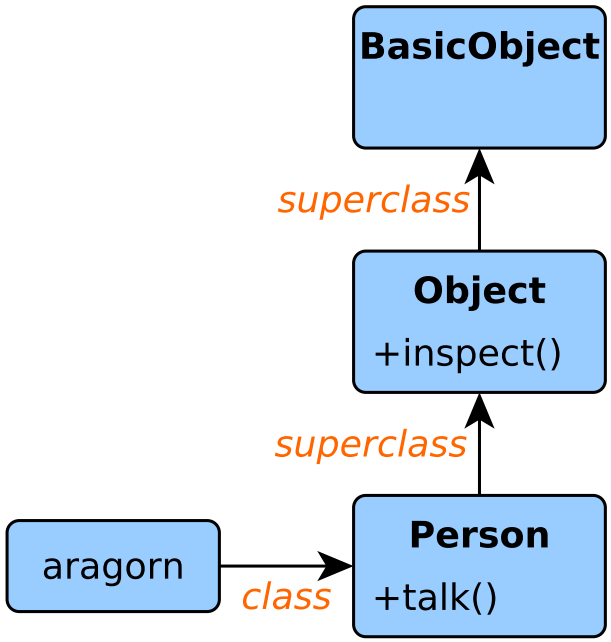
```ruby
aragorn = Person.new
legolas = Person.new

def aragorn.aka
  "Strider"
end

aragorn.aka
# => "Strider"

legolas.aka
# => NoMethodError: undefined method `aka' ...
```

Eigenclasses (singleton classes, metaclasses, ghost classes)

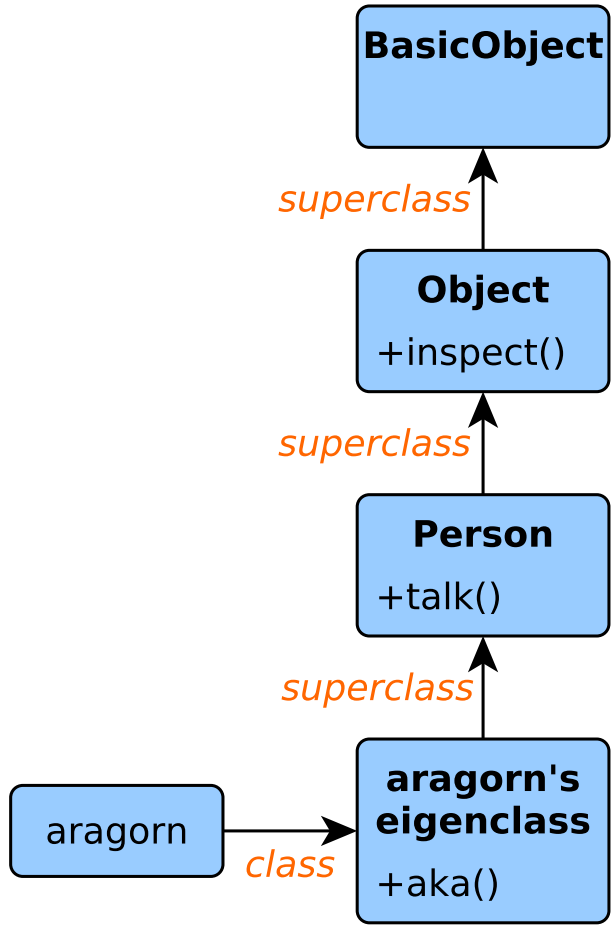


- Where is `aka`?

Eigenclasses (singleton classes, metaclasses, ghost classes)



- There it is!



Eigenclasses and the `class <<` syntax

- "one's very own" class
- anonymous class

Eigenclasses and the `class <<` syntax

- "one's very own" class
- anonymous class

```
1 def aragorn.aka
2   "Strider"
3 end
```

Eigenclasses and the `class <<` syntax

- "one's very own" class
- anonymous class
- The `class <<` syntax opens the eigenclass of whatever object you pass to it

```
1 def aragorn.aka
2   "Strider"
3 end
```

```
1 class << aragorn
2   def aka
3     "Strider"
4   end
5 end
```

Class Methods, revisited

- classes are also objects, so
  - ...they can have singleton methods

Class Methods, revisited

- classes are also objects, so
  - ...they can have singleton methods
- In fact, what we call class methods are singleton methods for classes!

Class Methods, revisited

- classes are also objects, so
  - ...they can have singleton methods
- In fact, what we call class methods are singleton methods for classes!

```
1 def Person.closest_dna
2   "chimpanzee"
3 end
```

Class Methods, revisited

- classes are also objects, so
  - ...they can have singleton methods
- In fact, what we call class methods are singleton methods for classes!

```
1 def Person.closest_dna
2   "chimpanzee"
3 end
```

```
1 class Person
2   def self.closest_dna
3     "chimpanzee"
4   end
5 end
```

Class Methods, revisited

- classes are also objects, so
  - ...they can have singleton methods
- In fact, what we call class methods are singleton methods for classes!

```ruby
def Person.closest_dna
  "chimpanzee"
end
```

```ruby
class << Person
  def closest_dna
    "chimpanzee"
  end
end
```

```ruby
class Person
  def self.closest_dna
    "chimpanzee"
  end
end
```

Class Methods, revisited

- classes are also objects, so
  - ...they can have singleton methods
- In fact, what we call class methods are singleton methods for classes!

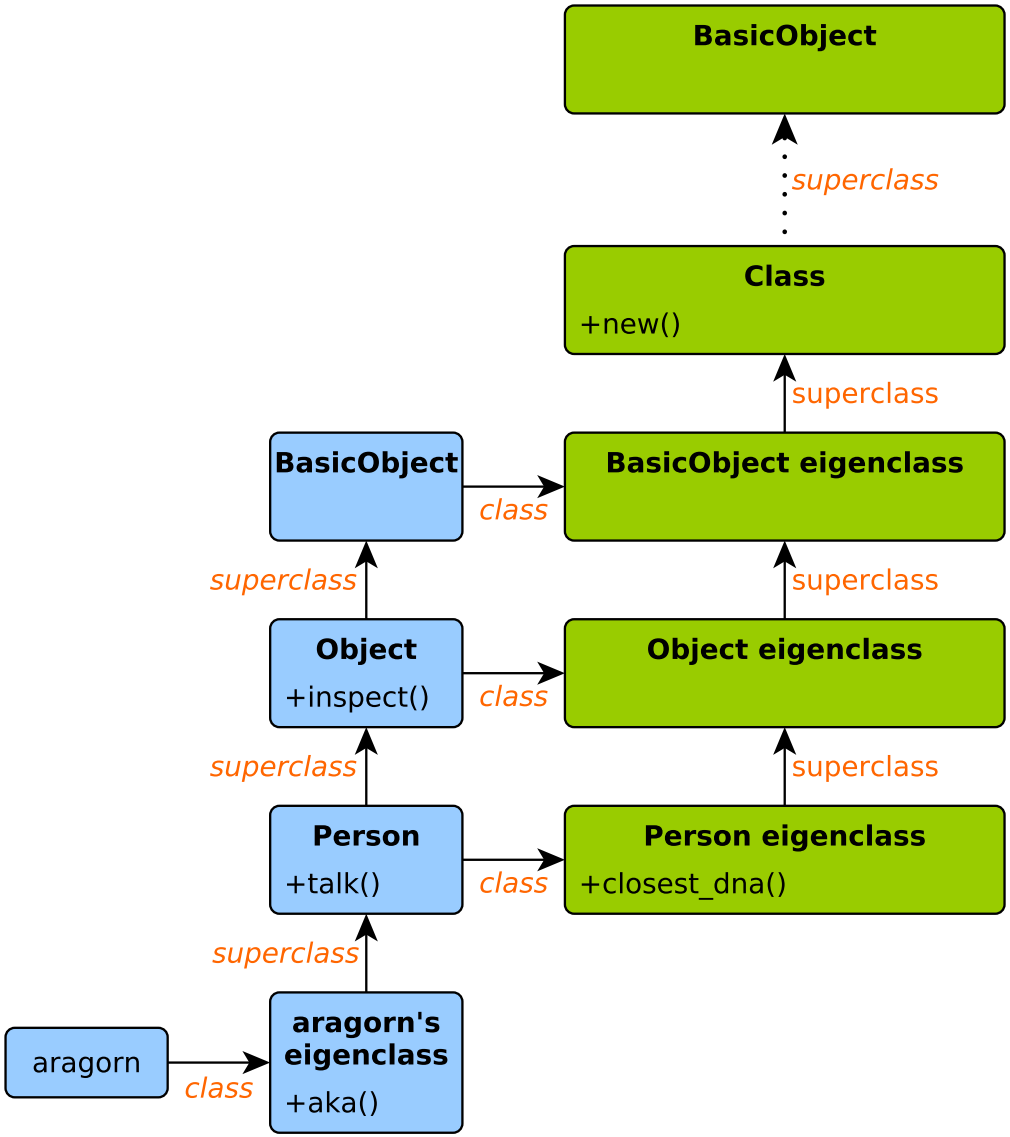
```ruby
def Person.closest_dna
  "chimpanzee"
end
```

```ruby
class << Person
  def closest_dna
    "chimpanzee"
  end
end
```

```ruby
class Person
  def self.closest_dna
    "chimpanzee"
  end
end
```

```ruby
class Person
  class << self
    def closest_dna
      "chimpanzee"
    end
  end
end
```

Complete lookup path



Modules

- A way of grouping together methods, classes and constants
  - namespaces

Modules

- A way of grouping together methods, classes and constants
  - namespaces

```
irb:001> Math.sqrt(9)
 => 3.0
```

Modules

- A way of grouping together methods, classes and constants
  - namespaces

```
irb:001> Math.sqrt(9)
 => 3.0
irb:002> Math::PI
 => 3.141592653589793
```

Modules

- A way of grouping together methods, classes and constants
  - namespaces

```
irb:001> Math.sqrt(9)
 => 3.0
irb:002> Math::PI
 => 3.141592653589793
irb:003> include Math
 => Object
```

Modules

- A way of grouping together methods, classes and constants
  - namespaces

```
irb:001> Math.sqrt(9)
 => 3.0
irb:002> Math::PI
 => 3.14159265358979 3
irb:003> include Math
 => Object
irb:004> sqrt(5)
 => 2.23606797749979
```

Modules

- A way of grouping together methods, classes and constants
  - namespaces

```
irb:001> Math.sqrt(9)
 => 3.0
irb:002> Math::PI
 => 3.14159265358979 3
irb:003> include Math
 => Object
irb:004> sqrt(5)
 => 2.23606797749979
irb:005> PI
 => 3.14159265358979 3
```

Modules

- A way of grouping together methods, classes and constants
  - namespaces

```
irb:001> Math.sqrt(9)
 => 3.0
irb:002> Math::PI
 => 3.141592653589793
irb:003> include Math
 => Object
irb:004> sqrt(5)
 => 2.23606797749979
irb:005> PI
 => 3.141592653589793
```

```
module MyTrig
  PI = 3.14
  def MyTrig.sqrt(n)
    # ..
  end
end

puts MyTrig::PI
puts Math::PI

# Output:
# 3.14
# 3.141592653589793
```

Modules, cont.

- A way of adding extra functionality to classes
  - mixins

Modules, cont.

- A way of adding extra functionality to classes
  - mixins

```
1 module MyLogger
2   def debug msg
3     puts "#{self.class}: " +
4          "#{msg}"
5   end
6   def error
7     # ..
8   end
9 end
```

Modules, cont.

- A way of adding extra functionality to classes
  - mixins

```ruby
module MyLogger
  def debug msg
    puts "#{self.class}: " +
         "#{msg}"
  end
  def error
    # ..
  end
end
```

```ruby
require_relative "my_logger"

class Person
  include MyLogger
  attr_accessor :name

  def initialize name
    debug "Inside init!"
    @name = name
  end
end

p = Person.new( "10" )
```

Mixins, another example

- Even more useful when mixin code interacts with class
  - `Comparable` provides `<`, `<=`, `==`, `>=`, `>`, `between?`
  - `Comparable` assumes class implements `<=>`

Mixins, another example

- Even more useful when mixin code interacts with class
  - `Comparable` provides `<`, `<=`, `==`, `>=`, `>`, `between?`
  - `Comparable` assumes class implements `<=>`

```
1  class Person
2    include Comparable
3
4    attr_accessor :name
5
6    def initialize name
7      @name = name
8    end
9
10   def to_s; "#{@name}"; end
11
12   def <=>(other)
13     @name <=> other.name
14   end
15 end
```

Mixins, another example

- Even more useful when mixin code interacts with class
  - `Comparable` provides `<`, `<=`, `==`, `>=`, `>`, `between?`
  - `Comparable` assumes class implements `<=>`

```ruby
class Person
  include Comparable

  attr_accessor :name

  def initialize name
    @name = name
  end

  def to_s; "#{@name}"; end

  def <=>(other)
    @name <=> other.name
  end
end
```

```ruby
p1 = Person.new( "Kostas" )
p2 = Person.new( "Alex" )
p3 = Person.new( "Dave" )

if p1 > p2
  puts "Yep, K > A"
end

puts [p1, p2, p3].sort
```

Files

- Organise code into multiple files
  - `load`
  - `require`
  - `require_relative`

Files

- Organise code into multiple files
  - `load`
  - `require`
  - `require_relative`
- Load path
  - `$LOAD_PATH` or `$:`

System Interaction

- File System
- Arguments
- Environment
- Shebang (hashbang)
- Ruby Command Line

File

```
1 f = File.new("filename", "r")
2 f.each_line { |l| puts l }
3 f.close
```

File

```
1 f = File.new("filename", "r")
2 f.each_line { |l| puts l }
3 f.close
```

```
1 File.open("filename", "r") do |f|
2    f.each_line { |l| puts l }
3 end
```

File

```ruby
f = File.new("filename", "r")
f.each_line { |l| puts l }
f.close
```

```ruby
File.open("filename", "r") do |f|
   f.each_line { |l| puts l }
end
```

```ruby
IO.foreach("filename") { |l| puts l }
```

File

```ruby
f = File.new("filename", "r")
f.each_line { |l| puts l }
f.close
```

```ruby
File.open("filename", "r") do |f|
   f.each_line { |l| puts l }
end
```

```ruby
IO.foreach("filename") { |l| puts l }
```

- File
  - .size
  - .exists?
  - .directory?
  - .writable?
  - .delete
  - .rename
  - .chown
  - .ctime

Dir

```
1 Dir.foreach("/usr/bin") do |entry|
2     puts entry
3 end
```

Dir

```ruby
Dir.foreach("/usr/bin") do |entry|
   puts entry
end
```

```ruby
Dir.entries("/usr/bin").join(' ')
```

Dir

```ruby
Dir.foreach("/usr/bin") do |entry|
   puts entry
end
```

```ruby
Dir.entries("/usr/bin").join(' ')
```

```ruby
Dir["/usr/bin/*"]
```

Dir

```ruby
Dir.foreach("/usr/bin") do |entry|
   puts entry
end
```

```ruby
Dir.entries("/usr/bin").join(' ')
```

```ruby
Dir["/usr/bin/*"]
```

- Dir
  - .chdir
  - .mkdir
  - .pwd

Arguments

- Command-line arguments
  - `ARGV` array
  - `ARGV.each { |a| puts a }`
- Check `optparse` from standard library

Environment

- Reading OS's environment variables
  - `ENV` hash
  - `ENV.each { |k,v| puts "#{k}=#{v}" }`

Shebang (hashbang)

```
1 #!/usr/bin/env ruby
2
3 puts "Hello World!"
```

Shebang (hashbang)

```
1 #!/usr/bin/env ruby
2
3 puts "Hello World!"
```

```
1 $ chmod 755 file.rb
2 $ ./file.rb
3 Hello World
4 $
```

Ruby Command Line

- Command line options
  - `-v`

Ruby Command Line

- Command line options
  - `-v`
  - `-e`
    - `$ ruby -e 'puts "Hi"'`

Ruby Command Line

- Command line options
  - `-v`
  - `-e`
    - `$ ruby -e 'puts "Hi"'`
  - `-n`
    - `while gets; ... end`
    - `$ echo -e "foo\nbar" | ruby -ne 'puts $_'`

Ruby Command Line

- Command line options
  - `-v`
  - `-e`
    - `$ ruby -e 'puts "Hi"'`
  - `-n`
    - `while gets; ... end`
    - `$ echo -e "foo\nbar" | ruby -ne 'puts $_'`
  - `-p`
    - operates on `$_` and writes it at every iteration
    - `$ echo -e "foo\nbar" | ruby -pe 'chomp'`

Miscellaneous

- Regular Expressions
- Web Services
- Concurrency: a bird's eye view
- Metaprogramming: a bird's eye view

Regular Expressions

- A pattern that can be matched against a string
  - `r = Regexp.new("pattern")`

Regular Expressions

- A pattern that can be matched against a string
  - `r = Regexp.new("pattern")`
  - `r = /pattern/`

Regular Expressions

- A pattern that can be matched against a string
  - `r = Regexp.new("pattern")`
  - `r = /pattern/`
  - `r = %r{pattern}`

Regular Expressions

- A pattern that can be matched against a string
  - `r = Regexp.new("pattern")`
  - `r = /pattern/`
  - `r = %r{pattern}`
  - Operators: `=~` and `!~`

Regular Expressions

- A pattern that can be matched against a string
  - `r = Regexp.new("pattern")`
  - `r = /pattern/`
  - `r = %r{pattern}`
  - Operators: `=~` and `!~`

```
1 /lo/ =~ "hello"                    # => 3
```

Regular Expressions

- A pattern that can be matched against a string
  - `r = Regexp.new("pattern")`
  - `r = /pattern/`
  - `r = %r{pattern}`
  - Operators: `=~` and `!~`

```
1 /lo/ =~ "hello"                    # => 3
2 "hello" =~ /lo/                    # => 3
```

Regular Expressions

- A pattern that can be matched against a string
  - `r = Regexp.new("pattern")`
  - `r = /pattern/`
  - `r = %r{pattern}`
  - Operators: `=~` and `!~`

```
1 /lo/ =~ "hello"                    # => 3
2 "hello" =~ /lo/                    # => 3
3 "hello" =~ /hi/                    # => nil
```

Regular Expressions

- A pattern that can be matched against a string
  - `r = Regexp.new("pattern")`
  - `r = /pattern/`
  - `r = %r{pattern}`
  - Operators: `=~` and `!~`

```ruby
/lo/ =~ "hello"                    # => 3
"hello" =~ /lo/                    # => 3
"hello" =~ /hi/                    # => nil
m = /ll/.match("hello")            # => #<MatchData "ll">
```

Regular Expressions

- A pattern that can be matched against a string
  - `r = Regexp.new("pattern")`
  - `r = /pattern/`
  - `r = %r{pattern}`
  - Operators: `=~` and `!~`

```
1 /lo/ =~ "hello"                                    # => 3
2 "hello" =~ /lo/                                    # => 3
3 "hello" =~ /hi/                                    # => nil
4 m = /ll/.match("hello")                            # => #<MatchData "ll">
5 "#{m.pre_match}->#{m[0]}<-#{m.post_match}"
6 # => "he->ll<-o"
```

Regular Expressions

- A pattern that can be matched against a string
  - `r = Regexp.new("pattern")`
  - `r = /pattern/`
  - `r = %r{pattern}`
  - Operators: `=~` and `!~`

```ruby
/lo/ =~ "hello"                                    # => 3
"hello" =~ /lo/                                    # => 3
"hello" =~ /hi/                                    # => nil
m = /ll/.match("hello")                            # => #<MatchData "ll">
"#{m.pre_match}->#{m[0]}<-#{m.post_match}"
# => "he->ll<-o"

if "hello" =~ /hi/
  # ..
end
```

Regular Expressions, cont.

- All characters match themselves except:
  - . | ( ) [ ] { } + \ ^ $ * ?

Regular Expressions, cont.

- All characters match themselves except:
  - . | ( ) [ ] { } + \ ^ $ * ?

```
1 "hello" =~ /^el/                    # => nil
```

Regular Expressions, cont.

- All characters match themselves except:
  - . | ( ) [ ] { } + \ ^ $ * ?

```
1 "hello" =~ /^el/                    # => nil
2 "hello " =~ /lo$/                   # => nil
```

Regular Expressions, cont.

- All characters match themselves except:
  - . | ( ) [ ] { } + \ ^ $ * ?

```
1 "hello" =~ /^el/                    # => nil
2 "hello " =~ /lo$/                   # => nil
3 "hello " =~ /[aeiou]/               # => 1
```

Regular Expressions, cont.

- All characters match themselves except:
  - . | ( ) [ ] { } + \ ^ $ * ?

```
1 "hello" =~ /^el/                    # => nil
2 "hello " =~ /lo$/                   # => nil
3 "hello " =~ /[aeiou]/               # => 1
4 "hello world" =~ /\s/               # => 5
```

Regular Expressions, cont.

- All characters match themselves except:
  - . | ( ) [ ] { } + \ ^ $ * ?

```
1 "hello" =~ /^el/                  # => nil
2 "hello " =~ /lo$/                 # => nil
3 "hello " =~ /[aeiou]/             # => 1
4 "hello world" =~ /\s/             # => 5
5 "hello world" =~ /\s[[:digit:]]/  # => nil
```

Regular Expressions, cont.

- All characters match themselves except:
  - . | ( ) [ ] { } + \ ^ $ * ?

```
1 "hello" =~ /^el/                  # => nil
2 "hello " =~ /lo$/                 # => nil
3 "hello " =~ /[aeiou]/             # => 1
4 "hello world" =~ /\s/             # => 5
5 "hello world" =~ /\s[[:digit:]]/  # => nil
6 "bat bit bot bet" =~ /(b.t )*/    # => 0
```

Regular Expressions, cont.

- All characters match themselves except:
  - . | ( ) [ ] { } + \ ^ $ * ?

```
1 "hello" =~ /^el/                    # => nil
2 "hello " =~ /lo$/                   # => nil
3 "hello " =~ /[aeiou]/               # => 1
4 "hello world" =~ /\s/               # => 5
5 "hello world" =~ /\s[[:digit:]]/    # => nil
6 "bat bit bot bet" =~ /(b.t )*/      # => 0
7 print $~                            # bat bit bot  => nil
```

Regular Expressions, cont.

- All characters match themselves except:
  - . | ( ) [ ] { } + \ ^ $ * ?

```
1 "hello" =~ /^el/                          # => nil
2 "hello " =~ /lo$/                         # => nil
3 "hello " =~ /[aeiou]/                     # => 1
4 "hello world" =~ /\s/                     # => 5
5 "hello world" =~ /\s[[:digit:]]/          # => nil
6 "bat bit bot bet" =~ /(b.t )*/            # => 0
7 print $~                                  # bat bit bot  => nil
8 "bat bit bot bet" =~ /(b.t ){2}/          # => 0
```

Regular Expressions, cont.

- All characters match themselves except:
  - . | ( ) [ ] { } + \ ^ $ * ?

```
1 "hello" =~ /^el/                          # => nil
2 "hello " =~ /lo$/                         # => nil
3 "hello " =~ /[aeiou]/                     # => 1
4 "hello world" =~ /\s/                     # => 5
5 "hello world" =~ /\s[[:digit:]]/          # => nil
6 "bat bit bot bet" =~ /(b.t )*/            # => 0
7 print $~                                  # bat bit bot  => nil
8 "bat bit bot bet" =~ /(b.t ){2}/          # => 0
9 print $~                                  # bat bit => nil
```

Regular Expressions, cont.

- All characters match themselves except:
  - . | ( ) [ ] { } + \ ^ $ * ?

```
1  "hello" =~ /^el/                    # => nil
2  "hello " =~ /lo$/                   # => nil
3  "hello " =~ /[aeiou]/               # => 1
4  "hello world" =~ /\s/               # => 5
5  "hello world" =~ /\s[[:digit:]]/    # => nil
6  "bat bit bot bet" =~ /(b.t )*/      # => 0
7  print $~                            # bat bit bot  => nil
8  "bat bit bot bet" =~ /(b.t ){2}/    # => 0
9  print $~                            # bat bit => nil
10 "mississipi" =~ /(...)\1/i          # => 1
```

Regular Expressions, cont.

- All characters match themselves except:
  - . | ( ) [ ] { } + \ ^ $ * ?

```
1  "hello" =~ /^el/                          # => nil
2  "hello " =~ /lo$/                         # => nil
3  "hello " =~ /[aeiou]/                     # => 1
4  "hello world" =~ /\s/                     # => 5
5  "hello world" =~ /\s[[:digit:]]/          # => nil
6  "bat bit bot bet" =~ /(b.t )*/            # => 0
7  print $~                                  # bat bit bot  => nil
8  "bat bit bot bet" =~ /(b.t ){2}/          # => 0
9  print $~                                  # bat bit => nil
10 "mississipi" =~ /(...)\1/i                # => 1
11 print $~                                  # ississ => nil
```

REST (rest-client)

```
1 require 'rest_client'
2
3 url = 'https://api.twitter.com/1/users/show.json'
4
5 response = RestClient.get(url,
6                 { :params => { :screen_name => 'TwitterAPI' } })
7
8 puts response.body
```

SOA/SOAP (savon – v2)

```ruby
require 'savon'

url = 'http://wsf.cdyne.com/WeatherWS/Weather.asmx?WSDL'

client = Savon.client(wsdl: url)

puts client.operations

puts client.call(:get_weather_information)
```

Concurrency: a bird's eye view

- Processes

Concurrency: a bird's eye view

- Processes
- Threads

Concurrency: a bird's eye view

- Processes
- Threads
  - green threads (v1.8)

Concurrency: a bird's eye view

- Processes
- Threads
  - green threads (v1.8)
  - native threads (v1.9)

Concurrency: a bird's eye view

- Processes
- Threads
  - green threads (v1.8)
  - native threads (v1.9)
    - GIL: data modified only by one thread at a time

Concurrency: a bird's eye view

- Processes
- Threads
  - green threads (v1.8)
  - native threads (v1.9)
    - GIL: data modified only by one thread at a time
    - ...but another thread could do I/O

Concurrency: a bird's eye view

- Processes
- Threads
  - green threads (v1.8)
  - native threads (v1.9)
    - GIL: data modified only by one thread at a time
    - ...but another thread could do I/O
  - JRuby, IronRuby, MacRuby, Rubinius, ...

Concurrency: a bird's eye view

- Processes
- Threads
  - green threads (v1.8)
  - native threads (v1.9)
    - GIL: data modified only by one thread at a time
    - ...but another thread could do I/O
  - JRuby, IronRuby, MacRuby, Rubinius, ...
- Fibers

Concurrency: a bird's eye view

- Processes
- Threads
  - green threads (v1.8)
  - native threads (v1.9)
    - GIL: data modified only by one thread at a time
    - ...but another thread could do I/O
  - JRuby, IronRuby, MacRuby, Rubinius, ...
- Fibers
  - *lightweight* threads

Concurrency: a bird's eye view

- Processes
- Threads
  - green threads (v1.8)
  - native threads (v1.9)
    - GIL: data modified only by one thread at a time
    - ...but another thread could do I/O
  - JRuby, IronRuby, MacRuby, Rubinius, ...
- Fibers
  - *lightweight* threads
  - ...but controlled by the developer

Metaprogramming: a bird's eye view

- DSLs: writing code that writes code

Metaprogramming: a bird's eye view

- DSLs: writing code that writes code
- Classes (and other language constructs) definition

Metaprogramming: a bird's eye view

- DSLs: writing code that writes code
- Classes (and other language constructs) definition
  - C, Pascal, ...

Metaprogramming: a bird's eye view

- DSLs: writing code that writes code
- Classes (and other language constructs) definition
  - C, Pascal, ...
  - ...at compile time

Metaprogramming: a bird's eye view

- DSLs: writing code that writes code
- Classes (and other language constructs) definition
  - C, Pascal, ...
  - ...at compile time
  - Java, ...

Metaprogramming: a bird's eye view

- DSLs: writing code that writes code
- Classes (and other language constructs) definition
  - C, Pascal, ...
  - ...at compile time
  - Java, ...
  - ...at compile time and provides read access at runtime

Metaprogramming: a bird's eye view

- DSLs: writing code that writes code
- Classes (and other language constructs) definition
  - C, Pascal, ...
  - ...at compile time
  - Java, ...
  - ...at compile time and provides read access at runtime
  - Ruby

Metaprogramming: a bird's eye view

- DSLs: writing code that writes code
- Classes (and other language constructs) definition
  - C, Pascal, ...
  - ...at compile time
  - Java, ...
  - ...at compile time and provides read access at runtime
  - Ruby
  - ...at compile time and provides read/write access at runtime