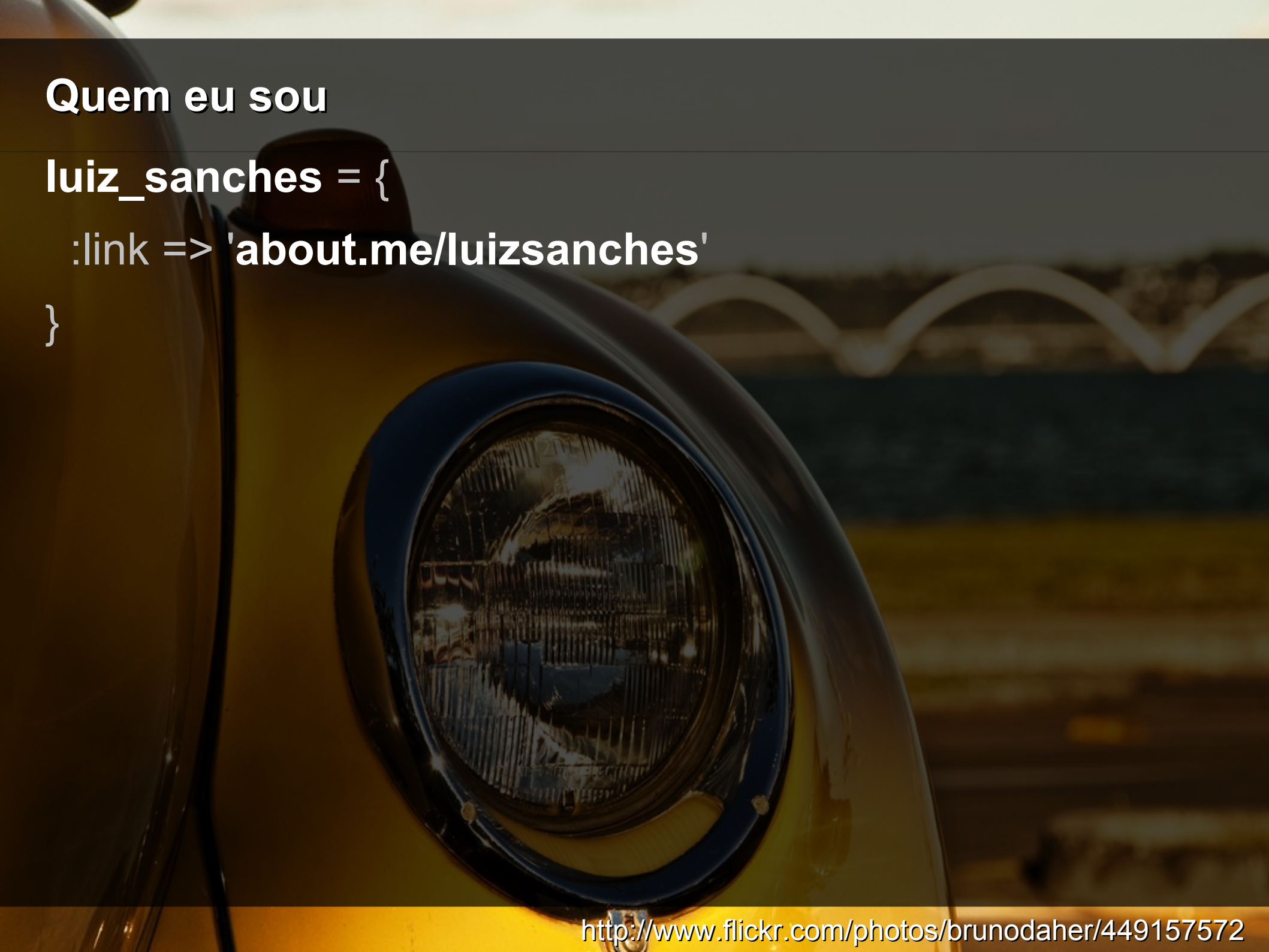


Ruby – praticamente falando





Quem eu sou

```
luiz_sanches = {  
  :link => 'about.me/luizsanches'  
}
```


Linguagem

=begin

É qualquer e todo sistema de signos que serve de meio de comunicação de ideias ou sentimentos.

Fonte: pt.wikipedia.org

=end

A linguagem Ruby

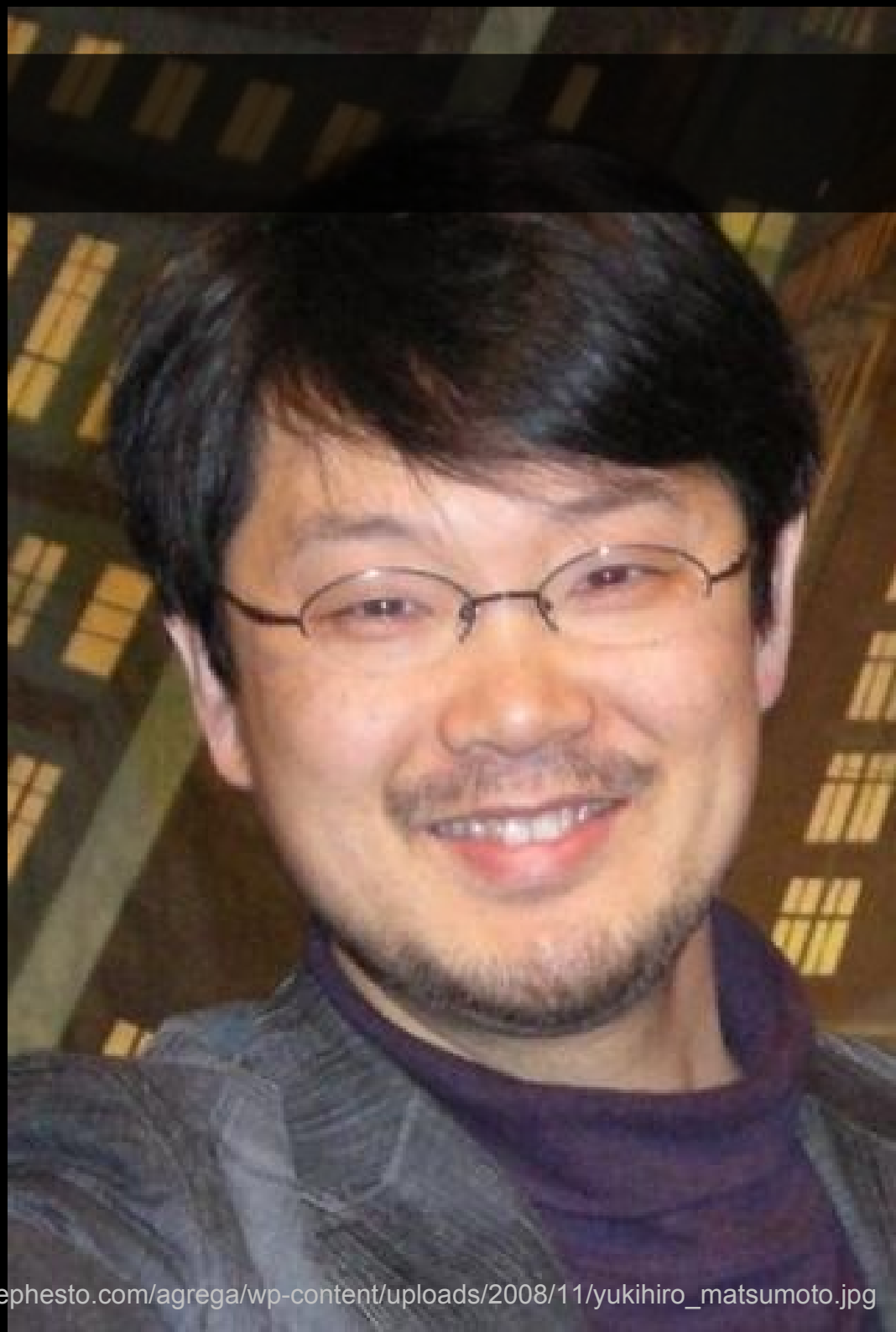
```
foi_inspirada_em = [  
  'Perl',  
  'Smalltalk',  
  'Eiffel',  
  'Ada',  
  'Lisp'  
]
```



Origem

Em **1995**, no **Japão**, Yukihiro "**Matz**" Matsumoto queria uma **linguagem** de script poderosa, totalmente **orientada** a objetos e a **pessoas**.

Ruby era a pedra zodiacal de um amigo de Matz.



Instalação

Linux e Mac OSX - rvm.io

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

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

Windows - railsinstaller.org

OBS.: Baixe o railsinstaller-3.2.1.exe com Ruby 2.2

Uso

Linux e Mac OSX

Abrir um console de comando

```
$ ruby -v
```

```
$ ruby arquivo.rb
```

```
$ irb
```

Windows

Abrir o prompt do ms-dos: botão iniciar → executar → cmd

```
C:\> ruby -v
```

```
C:\> ruby arquivo.rb
```

```
C:\> irb
```

Editores

- **Vim** - vim.org
- **Sublime Text** - sublimetext.com
- **Atom** - atom.io

Argumentos na linha de comando

```
# ola.rb
```

```
nome = ARGV[0]
```

```
print 'Ola ', nome
```

```
$ ruby ola.rb fulano
```

```
Ola fulano
```

Obtendo dados do usuário pelo terminal

ola.rb

```
print 'Digite seu nome: '
```

```
nome = gets.chomp
```

```
print 'Ola ', nome
```

\$ ruby ola.rb

Ola fulano

Input / Output

escrever.rb

```
print 'Escreva um texto: '
```

```
texto = gets
```

```
File.open('arquivo.txt', 'w') do |f|
```

```
  f << texto
```

```
end
```

\$ ruby escrever.rb

ler.rb

```
puts File.read('arquivo.txt')
```

\$ ruby ler.rb

irb (Ruby interativo)

```
irb(main):001:0> numero = 10  
=> 10
```

```
irb(main):002:0> if numero % 2 == 0
```

```
irb(main):003:1>   puts 'par'
```

```
irb(main):004:1> else
```

```
irb(main):005:1*   puts 'impar'
```

```
irb(main):006:1> end
```

```
par
```

```
=> nil
```

irb (Ruby interativo)

=begin

A primeira seção, onde está escrito irb(main), mostra o nome do programa que estamos rodando. A segunda seção mostra o número da linha, um contador de quantas linhas de Ruby nós digitamos. A terceira seção é um nível de profundidade. Toda vez que você abrir uma expressão que precisa ser fechada, o nível de profundidade vai aumentar uma unidade. E toda vez que o Irb detectar que seu código não está terminado, o fim do console vai virar um asterisco.

Fonte: why.carlosbrando.com/expansion-pak-1.html

=end

Comentando

Eu sou um comentário de uma linha

=begin

Eu sou um comentário

De várias linhas

=end

Seus tipos de dados são:

1.class # **Fixnum**

100000000000.class # **Bignum**

1.0.class # **Float**

'palavra'.class # **String**

:nome.class # **Symbol**

(1..10).class # **Range**

[1, 'dois'].class # **Array**

traduz = {'um' => 'one', 'dois' => 'two'}.class # **Hash**

/expressao/.class # **Regexp**

true.class # **TrueClass**

false.class # **FalseClass**

nil.class # **NilClass**

Mas no final, TUDO é

self.class # Object

1.class # Integer

1.class.superclass # Numeric

1.class.superclass.superclass # Object

1.class.superclass.superclass.superclass # **BasicObject**

Variáveis

sou_local = 'meu escopo é local'

@sou_de_instancia = 'sirvo ao objeto instanciado'

@@tenho_classe = 'sirvo a minha classe'

Constante = 'devo ser constante, mas aviso se mudar'

\$sou_global = 'sou globalizada e indesejada'

Numerics

sou_inteiro = 101

tambem_sou_inteiro = -453

sou_big_inteiro = 2000000000

tambem_sou_um_big_inteiro = 2_000_000_000

sou_float = 1.25

continuo_sendo_float = 2_000_000_000.45

Strings

'Quero saber quem fez essa bagunça?'

suspeito = 'Apolônio'

'Será que foi o ' + suspeito + '?' # **concatenação**

'Mas o ' << suspeito << ' não é disso.'

"Tudo indica ter sido mesmo o #{suspeito}." # **interpolação**

'culpado! ' * 3

condenacao=<<HEREDOC

Se o #{suspeito} for condenado, sofrerá as consequências
e passará um ano sem hamburger e batata frita.

HEREDOC

Symbols

`simbolo_nao_e = ['string', CONSTANTE=0, variavel=nil]`

:simbolo_e_um_identificador_que_representa_ele_mesmo

`Nome = "nome" # constante`

`nome = "nome" # variável`

`:nome # símbolo`

strings são mutáveis, assim como floats. Exemplo:

`'nome'.object_id # -606174638`

`'nome'.object_id # -606183698`

symbols são únicos, assim como inteiros. Exemplo:

`:nome.object_id # 166898`

`:nome.object_id # 166898`

Coleções

```
array = [1, 'dois', 3, 'quatro']  
array[1]
```

```
# adiciono elementos  
feira = []  
feira.push 'arroz'  
feira.push 'feijão'  
feira << 'morango'  
feira  
# retiro o último elemento  
feira.pop  
feira  
# retiro o primeiro elemento  
feira.shift  
feira
```

```
Hash = { 1 => 'um', 2 => 'dois' }  
hash[1]
```

```
idades = {  
  'Belém' => 'PA',  
  'Recife' => 'PE'  
}  
idades['Belém']
```

```
info = {  
  :nome => 'Manoel Carlos',  
  :email => 'manoel@mail.com'  
}  
info[:email]  
info.keys  
info.values
```

Expressões Regulares - aurelio.net/regex

(=~) corresponde e (!~) não-corresponde

er = /^[0-9]/

'123' =~ er # 0

'123' !~ er # false

'abc' =~ er # nil

'abc' !~ er # true

'me acha'.index(/me/) # 0

'gato'.gsub(/g/,'p') # pato

coisas = %w(laranja azeitona anel manga livro carro caneta)

coisas.grep(/**^a**/) # ["azeitona", "anel"]

coisas.grep(/**a\$**/) # ["laranja", "azeitona", "manga", "caneta"]

coisas.grep(/**^a.*a\$**/) # ["azeitona"]

Se

n1, n2 = 5, 10

if n1 > n2

puts "#{n1} é maior que #{n2}"

elsif n2 > n1

puts "#{n2} é maior que #{n1}"

else

puts 'Os números são iguais'

end

unless n1 > n2 **# senão**

puts "#{n2} é maior que #{n1}"

end

puts "#{n1} é " + (n1 % 2 == 0 ?
'par' : 'ímpar') **# operador ternário**

puts “só imprimo se a condição for
verdadeira” **if n1 > n2**

Caso

```
menu = :saldo  
case menu  
when :promocoes  
  puts 'nossas promoções'  
when :creditos  
  puts 'inserir créditos'  
when :saldo  
  puts 'consultar saldo'  
else  
  puts 'ouça mais música'  
end
```

```
opcao = 15  
faixa =  
  case opcao  
  when 0..10  
    "de 0 a 10"  
  when 11..20  
    "de 11 a 20"  
  else  
    "não encontrada"  
  end  
  puts "Faixa #{faixa}"
```


For

```
for r in (1..5)
```

```
  puts r
```

```
end
```

```
for r in (1...5)
```

```
  puts r
```

```
end
```

```
for a in ('a'..'f')
```

```
  puts a
```

```
end
```

```
1.upto(5) do |u|
```

```
  puts u
```

```
end
```

```
5.downto(1) { |d| puts d }
```

```
3.times { puts 'Tá safo!' }
```



Faça

w = 1

while w < 5 # enquanto

puts w

w += 1 # em Ruby não tem ++ nem --

end

u = 1

until u == 5 # até que

puts u

u += 1

end

Blocos e Procs

```
fala = Proc.new { puts 'oi' }  
fala.call
```

```
chama = Proc.new do  
  puts 'ei'  
  puts 'vem aqui!'  
end  
chama.call
```

```
despede = proc { puts 'tchau' }  
despede.call
```

```
sauda = lambda { |nome| puts "Olá #{nome}" }  
sauda.call 'Fábio'
```


Blocos e Iteradores

```
compras = ['arroz', 'feijão', 'açúcar']
```

```
compras.each do |item|  
  puts item  
end
```

```
compras.each { |item| puts item }
```

```
documentos = { :rg => '86474837', :cpf => '3653364645' }
```

```
documentos.each { |chave, valor| puts "#{chave} = #{valor}" }
```

Métodos

```
def grita  
  puts 'TO GRITANDO!'  
end  
grita
```

```
def soma(n1, n2)  
  n1 + n2  
end  
soma(5, 6)  
soma 7, 3
```

```
def executo_bloco(n)  
  yield(n)  
end  
executo_bloco(4) { |i| i * i }  
executo_bloco(4) { |i| i + i }
```

```
nome = 'Marta'
```

```
# (?) predicados
```

```
nome.include? 'M'
```

```
nome.include? 'f'
```

```
# (!) destrutivos
```

```
puts nome.upcase, nome
```

```
puts nome.upcase!, nome
```

Classes e herança

```
class Pessoa
```

```
  def initialize(nome) # construtor
```

```
    @nome = nome # de instância
```

```
  end
```

```
  def nome # get
```

```
    @nome
```

```
  end
```

```
  def nome=(novo_nome) # set
```

```
    @nome = novo_nome
```

```
  end
```

```
end
```

```
class Homem < Pessoa
```

```
  def initialize(nome)
```

```
    super(nome)
```

```
  end
```

```
end
```

```
# instanciando um objeto
```

```
fulano = Homem.new('Ambrósio')
```

```
puts fulano.nome
```

```
fulano.nome = 'Vanderlucio'
```

```
puts fulano.nome
```


Metaprogramação e variáveis de classe

```
class Pessoa
```

```
  attr_accessor :nome
```

```
  @@pessoas = 0 # de classe
```

```
  def initialize(nome)
```

```
    @nome = nome
```

```
    @@pessoas += 1
```

```
  end
```

```
  def Pessoa.quantidade
```

```
    @@pessoas
```

```
  end
```

```
end
```

```
class Homem < Pessoa
```

```
end
```

```
  # instanciando objetos
```

```
  fulano = Homem.new('Ambrósio')
```

```
  sicrano = Homem.new('Mariano')
```

```
  puts fulano.nome
```

```
  puts sicrano.nome
```

```
  puts Pessoa::quantidade
```


Visibilidade

```
class Visibilidade
  def metodo_publico
    puts 'sou um método público'
  end

  protected

  def metodo_protegido
    puts 'sou um método protegido'
  end

  private

  def metodo_privado
    puts 'sou um método privado'
  end
end
```

Tipagem dinâmica e forte

posso ser o que eu quiser

variavel = 'sou uma string'

variavel = 10

mas não posso abusar

valor1 = 100

valor2 = '200'

soma = valor1 + valor2 # **erro na certa!**

soma = valor1 + valor2.to_i # agora funfa!

junta = valor1.to_s + valor2 # também funfa!

Duck Typing

```
def calcular(a, b, c)  
    (a + b) * c  
end
```

```
calcular(1, 2, 3)
```

```
calcular('mangas ', 'e uvas, ', 2)
```

```
calcular([1, 2, 3], [4, 5, 6], 2)
```


Açúcar Sintático (Legibilidade)

quando você faz

```
calculo = 2 + 3
```

```
apelido = 'Farol'
```

```
class Mulher
```

```
  attr_writer :nome
```

```
end
```

```
fulana = Mulher.new
```

```
fulana.nome = 'Ana'
```

Ruby está fazendo

```
calculo = 2.+(3) → 2.send '+', 3
```

```
apelido = String.new('Farol')
```

```
class Mulher
```

```
  def nome=(nome)
```

```
    @nome=nome
```

```
  end
```

```
end
```

```
fulana = Mulher.new
```

```
fulana.nome=('Ana')
```


Módulos e composição

```
class Ave
  def voar
    'bate as asas'
  end
end
sabia = Ave.new
sabia.voar
```

```
module Mamifero
  def mergulhar
    'prende a respiração'
  end
  module_function :mergulhar
  public :mergulhar
end
Mamifero.mergulhar
```

```
class AveSelvagem < Ave
  include Mamifero
end
pato = AveSelvagem.new
pato.voar
pato.mergulhar
```

```
# Mais um pouco de módulos
Math::PI
Math.sqrt(9)
Math.class # Module
```


Exceções

```
valor1 = 1
```

```
valor2 = '2'
```

```
begin
```

```
  puts valor1 + valor2
```

```
rescue TypeError => motivo
```

```
  puts "Deu bronca! TypeError: #{motivo}"
```

```
ensure
```

```
  puts "De qualquer forma, tô por aqui"
```

```
end
```

Depuração – pryrepl.org

instalar a gem

```
gem install pry
```

iniciar um console melhorado

```
pry
```

depurar código

```
# debug.rb
```

```
require 'pry'
```

```
linguagem = 'ruby'
```

```
binding.pry
```

```
puts linguagem
```

executar

```
ruby debug.rb
```


Testes unitários

teste_calculadora.rb

```
require 'test/unit'
require './calculadora'
class TesteCalculadora < Test::Unit::TestCase
  def setup
    @calc = Calculadora.new
  end
  def test_soma
    assert_equal(4, @calc.soma(1, 3), '1 + 3 = 4')
  end
  def test_subtrai
    assert_equal(2, @calc.subtrai(5, 2), '5 - 2 = 3')
  end
  def teardown
    @calc = nil
  end
end
```

calculadora.rb

```
class Calculadora
  def soma(a, b)
    a + b
  end

  def subtrai(a, b)
    a - b
  end
end
```

No terminal

\$ ruby teste_calculadora.rb

Gemologia

=begin

É a especialidade da Geologia que estuda o caráter físico e químico dos materiais de valores gemológicos.

Fonte: pt.wikipedia.org

=end

RubyGems

- **Gerenciador de pacotes** (programas e bibliotecas) do Ruby
- **Pacote = Gema**
- **Gerencia dependências** (similar ao apt-get do Debian)
- Fonte padrão das gemas: **rubygems.org**

Exemplos:

gem install rails

gem install mongoid

gem list

gem uninstall dbi

Referências

> Site oficial da linguagem

ruby-lang.org

> Ruby a Partir de Outras Linguagens

ruby-lang.org/pt/documentation/ruby-from-other-languages

> Experimente Ruby no navegador

tryruby.org

> Aprenda a programar de Chris Pine

jmonteiro.com/aprendaaprogramar

> Tutorial de Ruby do TaQ

eustaquiorangel.com/downloads

> O (comovente) guia de Ruby do Why

why.carlosbrando.com

> The Little Book Of Ruby

sapphiresteel.com/ruby-programming/The-Little-Book-Of-Ruby.html

Mas afinal, a quem puts pertence?

Kernel.puts('Ele é um método do módulo Kernel do Ruby, assim como **gets**.')

Kernel.puts('Só confirmando que tudo em Ruby é')

Kernel.class # Module

Kernel.class.superclass # **Object**

Obrigado!

> slideshare.net/luizsanches/ruby-praticamente-falando

> github.com/luizsanches/ruby-praticamente-falando

