

Ruby for Beginner

A Programmer's Best Friend

Speaker:
RIZQI NUR ASSYAUFY



RIZQI NUR ASSYAUF

 [bandithijo.github.io](https://github.com/bandithijo)

    [bandithijo](#)  [gmail.com](#)

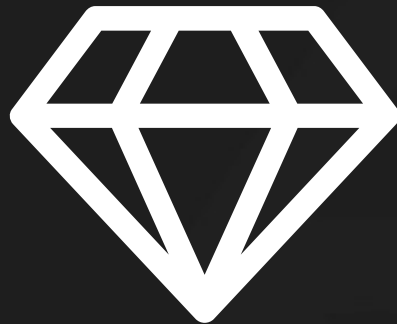
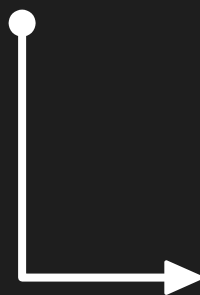
t.me



/GNUlinuxIndonesia /VimID
/JekyllID /ArchLinuxID
/dotfiles_id /ruby_id



bandithijo



- **2017**
Agung Setiawan
IDRails.com
Online Course
- **2019**
VirtualSpirit
Technology
Malaysia
Junior Backend
Ruby on Rails
Developer

Introduction to Ruby

Ruby for Beginner
A Programmer's Best Friend

> Sedikit tentang Ruby

:creator ⇒ Yukihiro “Matz” Matsumoto

:inspired ⇒ [Perl, Smalltalk, Eiffel, Ada, Lisp]

:public_release ⇒ 1995

Matz.said("trying to make Ruby natural, not simple")

_.append("Ruby is simple in appearance,
but is very complex inside,
just like our human body.")

:mass_acceptance ⇒ 2006 < DHH.create(
David Heinemeier Hansson,
Ruby on Rails,
Basecamp, 2003)

Source: About Ruby - <http://www.ruby-lang.org/en/about/>

世界が Ruby を 愛する理由

まつもと ゆきひろ 氏 ネットワーク応用通信研究所 特別研究員

Matz

> Is Ruby Dead Programming Language?

We moved away
from Ruby.
It's dying language...
and it has scalability
problem.

Let's host our projects on
GitHub & **GitLab**, use **Stripe** as
our payment processor, crowdfund
with **KickStarter**, and
livestream on **Twitch**. Also,
write about our industry and
programming on **dev.to**

Source: Nate Hopkins @hopsoft - <https://twitter.com/hopsoft/status/1169706322851328000>

> Is Ruby Dead Programming Language?

```
success_stories("Ruby")
```

```
⇒ {
```

```
Simulations: [Nasa, Motorola, etc.]
```

```
3D_Modeling: [Google Scketchup, etc.]
```

```
Robotics : [Morpha Project (Siemens), etc.]
```

```
Security : [Metasploit Framework, etc.]
```

```
Web Apps : [Basecamp, GitHub, GitLab, etc.]
```

```
etc. : [...] }
```

Source: Ruby Lang Success Stories - <http://www.ruby-lang.org/en/documentation/success-stories/>

> Is Ruby Dead Programming Language?

`Ruby.who("INDONESIA").value`

⇒ [Bukalapak, Cookpad Indonesia, Midtrans, Quipper, Vidio, HIJUP, Alodokter, Apisentris, BBM, BelanjaQu, Binary Academy, Dropsuite, Forstok, GO-JEK, Jualo, Jurnal, Karir.com, KlikDokter, KMK Online, Mekari, Mitrais, Peentar, Qontak, Sejasa.com, Sleekr, Sribulancer, Starqle, Virkea, Wego, WGS, etc...]

Source: :id_ruby - Komunitas Ruby Indonesia - <https://ruby.id>

> Is Ruby Dead Programming Language?

```
Language::Ruby.is_dead(  
  "Ruby is alive and thinking about next 25 years"  
  "dev.to/teaglebuilt/is-ruby-dead-17em"  
  "hackernoon.com/the-state-of-ruby-2019-is-it-dying"  
  "GitHub Blog - Upgrading GitHub to Ruby 2.7"  
  "...")  
⇒ false
```

> Is Ruby Dead Programming Language?

```
Framework::Rails.is_dead(  
  "similartech.com/technologies/ruby-on-rails"  
  "similartech.com/categories/framework"  
  "hackernoon.com/is-ruby-on-rails-dead-in-2019"  
  "dockyard.com/blog/ruby-on-rails-in-2020"  
  "Is Ruby on Rails dying: Myth or Reality?"  
  "...")  
⇒ false
```

> Is Ruby Dead Programming Language?

```
Language::Ruby.is_dead(  
  "Ruby is dead. (rubyisdead.science)"  
⇒ false!
```

```
@jmcharnes.said("isrubydead.com")  
⇒ NO
```

Just because Ruby is not popular anymore,
it doesn't mean Ruby is dead, right?

> Kenapa Memilih Ruby?

Learning::Ruby.is_worth?(:me).value

⇒ {

a: "Membantu menyederhanakan kerumitan"

b: "Membantu menerjemahkan ide/solusi, menjadi sebuah script atau program"

c: "Sintaks yang mudah dipahami"

d: "Ketersediaan library (gem) yang lengkap"

e: "Ketersediaan dokumentasi API yang lengkap"

f: "Dukungan komunitas yang solid"

g: "Open source"

h: "Digunakan oleh Organisasi/Company Besar"

i: "..."

}

> Kenapa Memilih Ruby?

```
PracticalThing::Ruby.can_do(:you)
```

```
⇒ {
```

```
a: "Full-stack web development"
```

```
b: "Web scraping & crawling"
```

```
c: "Static website generators"
```

```
d: "Automation, Backup, & DevOps tools"
```

```
e: "Build your own servers"
```

```
f: "Parsing data, cleaning & filtering"
```

```
g: "API clients (like Twitter's or GitHub's)"
```

```
h: "Report generators (PDF, HTML, CSV, SpreadSheet)"
```

```
i: "Command-line tools"
```

```
j: "Games (Ruby2D, etc.)"
```

```
k: "Data Science"
```

```
l: "Machine Learning & AI"
```

```
}
```

Source: RubyGuide - Jesus Castello - <https://www.rubyguides.com/2019/11/what-can-you-do-with-ruby/>

> What's new on Ruby 2.7?

Ruby.whats_new?(2.7)

⇒ {

a: "Enumerable#tally"

b: "Numbered params for Blocks [Experimental]"

c: "Array#intersection"

d: "Enumerable#filter_map"

e: "Enumerator#produce"

f: "IRB gets a face lift"

g: "Ruby pattern matching [Experimental]"

h: "10x performance ↑ for fiber & thread creation"

i: "Others changes, *go to source"

}

Source: RubyGuide - Jesus Castello - <https://www.rubyguides.com/2019/12/ruby-2-7-new-features/>

> What's new on Ruby 2.7?



Ruby NEVER STOPS Improving!

According to Matz...

**This is the last 2.x version release.
Because next year,
we're getting Ruby 3.0!**

Source: RubyGuide - Jesus Castello - <https://www.rubyguides.com/2019/12/ruby-2-7-new-features/>

> Tips Memasang Ruby for Dev

Groom your app's Ruby env. with Rbenv

Basic GitHub Checkout way:

```
$ cd
$ git clone https://github.com/rbenv/rbenv.git ~/.rbenv
$ echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc # ~/.zshrc
$ echo 'eval "$(rbenv init -)"' >> ~/.bashrc # ~/.zshrc
$ exec $SHELL

$ git clone https://github.com/rbenv/ruby-build.git ~/.rbenv/plugins/ruby-build
$ echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc
# ~/.zshrc

$ exec $SHELL

$ rbenv install 2.7.1 ← 📄
$ rbenv global 2.7.1
$ ruby -v
```

```
🏠 /home/bandithijo
├─ 📁 project_a      ← ruby 2.7.1
├─ 📁 project_b      ← ruby 2.6.3
│   └─ 📁 project_c  ← ruby 2.5.3
└─ 📁 project_d      ← ruby 2.4.2
```

Source: Rbenv (GitHub) - <https://github.com/rbenv/rbenv/blob/master/README.md>

Source: GoRails.com - <https://gorails.com/setup/ubuntu/20.04#ruby-rbenv>

> Tips Memasang Ruby for Dev

Groom your app's Ruby env. with Rbenv

Verify that rbenv is properly set up:

```
$ curl -fsSL \
https://github.com/rbenv/rbenv-installer/raw/master/bin/rbenv-doctor | bash
```

```
https://github.com/rbenv/rbenv-installer/raw/master/bin/rbenv-doctor | bash
Checking for `rbenv' in PATH: /home/bandithijo/.rbenv/bin/rbenv
Checking for rbenv shims in PATH: OK
Checking `rbenv install' support: /home/bandithijo/.rbenv/plugins/ruby-build/
bin/rbenv-install (ruby-build 20200520-10-g157c719)
Counting installed Ruby versions: 8 versions
Checking RubyGems settings: OK
Auditing installed plugins: OK
```

```
🏠 /home/bandithijo
├─ 📁 project_a      ← ruby 2.7.1
├─ 📁 project_b      ← ruby 2.6.3
│   └─ 📁 project_c  ← ruby 2.5.3
└─ 📁 project_d      ← ruby 2.4.2
```

Source: Rbenv (GitHub) - <https://github.com/rbenv/rbenv/blob/master/README.md>

Source: GoRails.com - <https://gorails.com/setup/ubuntu/20.04#ruby-rbenv>

Ruby Style Guide

Ruby for Beginner
A Programmer's Best Friend

> Ruby Style Guide

Adalah panduan menulis kode Ruby dengan “best practice” sehingga sesama programmer Ruby dapat menulis kode yang dapat dipelihara satu sama lain.

Indentation

Gunakan **2** spasi per level indentasi.

```
# bad - four spaces
def some_method
  do_something
end
```



```
# good
def some_method
  do_something
end
```



Source: The Ruby Style Guide - <https://rubystyle.guide/>

> Ruby Style Guide

Should I Terminate Expression with ; ?

Don't use ; to terminate statements and expressions.

```
# bad  
puts 'foobar'; # superfluous semicolon
```



```
# good  
puts 'foobar'
```



Source: The Ruby Style Guide - <https://rubystyle.guide/>

> Ruby Style Guide

One Expression Per Line

```
# bad  
puts 'foo'; puts 'bar' # two expressions on the same line
```



```
# good
```

```
puts 'foo'  
puts 'bar'
```



```
puts 'foo', 'bar' # this applies to puts in particular
```



Source: The Ruby Style Guide - <https://rubystyle.guide/>

> Ruby Style Guide

Spaces & Operators

Use spaces **around operators**, **after commas**, **colons** and **semicolons**

```
# bad
sum=1+2
a,b=1,2
class FooError<StandardError;end
```



```
# good
sum = 1 + 2
a, b = 1, 2
class FooError < StandardError; end
```



Source: The Ruby Style Guide - <https://rubystyle.guide/>

> Ruby Style Guide

Spaces & Operators (Exceptions)

Exponent operator

```
# bad  
e = M * c ** 2
```



```
# good  
e = M * c**2
```



Safe navigation operator

```
# bad  
foo &. bar  
foo &.bar  
foo&. bar
```



```
# good  
foo&.bar
```



Slash in rational literals

```
# bad  
o_scale = 1 / 48r
```



```
# good  
o_scale = 1/48r
```



Source: The Ruby Style Guide - <https://rubystyle.guide/>

> Ruby Style Guide

Spaces & Braces

No! spaces after `(`, `[` or before `]`, `)`
Use spaces around `{` and before `}`

bad

```
some( arg ).other  
[ 1, 2, 3 ].each{|e| puts e}
```



good

```
some(arg).other  
[1, 2, 3].each { |e| puts e }
```



Source: The Ruby Style Guide - <https://rubystyle.guide/>

> Ruby Style Guide

Spaces & Braces (Exceptions)

Hash literals

```
# good - space after { and before }  
{ one: 1, two: 2 }
```



```
# good - no space after { and before }  
{one: 1, two: 2}
```



String interpolation expressions

```
# bad  
"From: #{ user.first_name }, #{ user.last_name }"
```



```
# good  
"From: #{user.first_name}, #{user.last_name}"
```



Source: The Ruby Style Guide - <https://rubystyle.guide/>

> Ruby Style Guide

CamelCase for Classes

Use CamelCase for **classes** and **modules**.
(Keep acronyms like HTTP, RFC, XML uppercase).

bad

```
class Someclass  
  # some code  
end
```



```
class Some_Class  
  # some code  
end
```



```
class XmlSomething  
  # some code  
end
```



good

```
class SomeClass  
  # some code  
end
```



```
class SomeXML  
  # some code  
end
```



```
class XMLSomething  
  # some code  
end
```



Source: The Ruby Style Guide - <https://rubystyle.guide/>

> Ruby Style Guide

Snake Case for Symbols, Methods, & Variables

Use snake_case for **symbols**, **methods** and **variables**.

```
# bad
:'some symbol'
:SomeSymbol
:someSymbol
```



```
someVar = 5
```



```
def someMethod
  # some code
end
```



```
def SomeMethod
  # some code
end
```



```
# good
:some_symbol
```



```
some_var = 5
```



```
def some_method
  # some code
end
```



Source: The Ruby Style Guide - <https://rubystyle.guide/>

> Ruby Style Guide

Lebih lengkapnya silahkan kunjungi,

The Ruby Style Guide

<https://rubystyle.guide>



IRB

(Interactive Ruby Shell)

Ruby for Beginner
A Programmer's Best Friend

> Apa itu IRB?

IRB adalah kependekan dari **Interactive Ruby**.

Sebuah shell yang dapat kita gunakan untuk bermain-main dan bereksplorasi dengan Ruby dan langsung melihat hasilnya. Definisi ini diistilahkan dengan REPL (Read-Eval-Print-Loop).

Untuk menjalankan IRB, cukup tulis di terminal dengan perintah:

```
$ irb  
irb(main):001:0> _
```

Ayo kita coba jalankan operasi matematika!

```
irb(main):001:0> 1 + 2 * (3**4) / 5 - 6  
⇒ 27
```

Source: RubyGuides - <https://www.rubyguides.com/2018/12/what-is-a-repl-in-ruby/>

> Customisasi IRB

Secara default, bentuk dari "prompt" IRB akan seperti ini

```
$ irb  
irb(main):001:0> _
```

Secara pribadi, saya kurang suka melihat prompt yang terlalu panjang.

Kita bisa menyederhanakan dengan memanggil dengan cara berbeda.

```
$ irb --prompt simple  
>> _
```

Untuk membuat prompt "simple" menjadi default yang digunakan, cukup tambahkan pada file ~/.irbc (buka dengan text editor).

```
IRB.conf[:PROMPT_MODE] = :SIMPLE
```

Nah, sekarang apabila teman-teman memanggil irb, maka otomatis akan menggunakan "simple" prompt.

Source: Rake Routes - <https://www.rakeroutes.com/customize-your-irb/>



.rb

(dot rb file)

Ruby for Beginner
A Programmer's Best Friend

> .rb

Ruby memiliki file ekstensi .rb

Tujuannya agar Ruby interpreter dapat mengenali kalau file yang kita jalankan adalah file Ruby.

nama_file.rb

```
puts 'Helo, rubyist!'
puts 'Selamat belajar bahasa pemrograman Ruby.'
```

Untuk menjalankan file .rb

```
$ ruby nama_file.rb
```

```
Helo, rubyist!
Selamat belajar bahasa pemrograman Ruby.
```

Variabel

> Apa itu Variabel?

Digunakan untuk memberikan nama & menyimpan nilai (value)

In Ruby, Variable is **just a label**.

...a name for something that you can use to reference this value in your Ruby programs.

Just like the name we give to real-world things.
When I say "apple" you know what I'm talking about.
I don't have to describe it to you.
That's what variables do!

```
nama_lengkap = 'Yukihiro Matsumoto' ← String
tahun_lahir   = 1995                  ← Integer
penjajahan    = 3.5                   ← Float
bumi_datar     = false                 ← Boolean
kamar_kosong   = nil                  ← NilClass
tim_dev        = ['budi', 1, true]     ← Array
kontak_dev     = { 'nama': 'budi' }    ← Hash
```

Source: RubyGuides - <https://www.rubyguides.com/ruby-tutorial/variables/>

> Mengecek Class dari Variabel

Darimana kita tahu `variable` tertentu termasuk dalam Class apa?

Menggunakan method `class()`

```
>> nama = 'Yukihiko Matsumoto'
>> nama.class
=> String

>> umur = 55
>> umur.class
=> Integer

>> tim_dev = ['Rizal', 'Reza', 'Rizqi']
>> tim_dev.class
=> Array
```

INFO!

Cara memanggil sebuah method dengan menggunakan tanda `.` (dot)

> Scope/Jangkauan Variabel?

Local variable

```
team_name = "Elite Lokal"
```

Global variable (prefix: \$)

```
$team_name = "Elite Global"
```

Instance variable (prefix: @)

```
@fav_food = "Indomie Instance"
```

Class variable (prefix: @@)

```
@@fav_drama = "Itaewon Class"
```

Constant variable (Huruf Besar)

```
PHI = 3.14
```

Source: RubyGuides - <https://www.rubyguides.com/ruby-tutorial/variables/>

> Scope/Jangkauan Variabel?

The difference between them?

It's on their "**scope**".

A variable scope answers this question:

"From where can I access this variable?"

This is only going to matter when you start learning about Object-Oriented Programming.

Source: RubyGuides - <https://www.rubyguides.com/ruby-tutorial/variables/>

Source: RubyGuides - <https://www.rubyguides.com/2019/03/ruby-scope-binding/>

Puts, Print, P Console

Ruby for Beginner
A Programmer's Best Friend

> Puts, Print, P

puts

Menampilkan isi dari object & menambahkan newline “\n” di akhir.

```
>> nama_lengkap = 'Yukihiro Matsumoto'  
>> puts nama_lengkap  
Yukihiro Matsumoto  
⇒ nil
```

print

Menampilkan isi dari object tanpa newline “\n” di akhir.

```
>> print nama_lengkap  
Yukihiro Matsumoto⇒ nil
```

p

Menampilkan isi dari object & mengembalikan nilai dari object tsb.

```
>> p nama_lengkap  
"Yukihiro Matsumoto"  
⇒ "Yukihiro Matsumoto"
```

Source: RubyGuides - <https://www.rubyguides.com/2018/10/puts-vs-print/>

Working with String (a)

.Working with String (a)

- > String Concatenation
- > String Interpolation
- > Mengecek Method pada Class
- > String Method yang Biasa Digunakan
- > Escaping Character

> String Concatenation

Combining multiple string

a. Menggunakan operator +

```
>> nama_depan      = 'Yukihiro'
>> nama_belakang   = 'Matsumoto'
>> nama_lengkap    = nama_depan + nama_belakang

>> nama_lengkap
"YukihiroMatsumoto"

>> puts nama_depan + ' ' + nama_belakang
"Yukihiro Matsumoto"

>> puts 'Yukihiro' + ' ' + 'Matsumoto'
"Yukihiro Matsumoto"
```

Source: RubyGuides - <https://www.rubyguides.com/2019/07/ruby-string-concatenation/>

> String Concatenation

Combining multiple string

b. Menggunakan operator +=

```
>> nama = ''  
  
>> nama += 'Yukihiro'  
>> nama += ' '  
>> nama += 'Matsumoto'  
  
>> puts nama  
"Yukihiro Matsumoto"
```

Source: RubyGuides - <https://www.rubyguides.com/2019/07/ruby-string-concatenation/>

> String Concatenation

Combining multiple string

c. Menggunakan concat() method

```
>> nama = ''  
  
>> nama.concat('Yukihiro')  
>> nama.concat(' ')  
>> nama.concat('Matsumoto')  
  
>> puts nama  
"Yukihiro Matsumoto"
```

Source: RubyGuides - <https://www.rubyguides.com/2019/07/ruby-string-concatenation/>

> String Concatenation

Combining multiple string

d. Menggunakan operator << (alias dari concat() method)

```
>> nama = ''  
  
>> nama << 'Yukihiro'  
>> nama << ''  
>> nama << 'Matsumoto'  
  
>> puts nama  
"Yukihiro Matsumoto"  
  
>> puts nama << 'Yukihiro' << ' ' << 'Matsumoto'  
"Yukihiro Matsumoto"
```

```
>> 'berat badan = ' << 70  
# TypeError: no implicit conversion of Fixnum into String
```

Source: RubyGuides - <https://www.rubyguides.com/2019/07/ruby-string-concatenation/>

> String Concatenation

Combining multiple string

d. Menggunakan operator `<<` (alias dari `concat()` method)

Problem:

```
>> 'berat badan = ' << 70  
⇒ "berat badan = F"
```

Solusi:

```
>> 'tinggi badan = ' << 70.to_s  
⇒ "tinggi badan = 70"
```

Menggunakan `to_s` method untuk mengkonversi Integer ke String.

Source: RubyGuides - <https://www.rubyguides.com/2019/07/ruby-string-concatenation/>

Source: RubyGuides - <https://www.rubyguides.com/2018/09/ruby-conversion-methods/>

> String Concatenation

Combining multiple string

e. Menggunakan `prepend()` method (kebalikan dari `concat`)

```
>> nama = ''  
  
>> nama.prepend('Yukihiro')  
>> nama.prepend(' ')  
>> nama.prepend('Matsumoto')  
  
>> puts nama  
"Matsumoto Yukihiro"
```

📄 If you're thinking there is an `append` method, well there isn't for `String`. `append` method only for arrays.

Source: RubyGuides - <https://www.rubyguides.com/2019/07/ruby-string-concatenation/>

> String Interpolation

Interpolation or merging of variables into strings is a powerful technique. It allows you to "templatize" a string.

a. Menggunakan "#{"

```
>> nama_depan      = 'Yukihiko'  
>> nama_belakang   = 'Matsumoto'  
>> umur_sekarang   = 55  
  
>> puts "Nama: #{nama_depan} #{nama_belakang}. Umur: #{umur}"  
"Nama: Yukihiko Matsumoto. Umur: 55"
```

! Syarat: String harus menggunakan tanda petik ganda ("...")

Source: RubyGuides - <https://www.rubyguides.com/ruby-tutorial/variables/>

> Mengecek Method pada Class

Darimana kita tahu method apa yang tersedia dari sebuah Class?

Menggunakan `methods()` method

```
>> nama = 'Yukihiro Matsumoto'
>> nama.class
=> String

>> nama.methods
=> [ :unicode_normalize, :unicode_normalize!, :ascii_only?, :unpack,
    :unpack1, :to_r, :shellsplit, :encode, :encode!, :shellescape, :
    %, :include?, :*, :+, :pretty_print, ..., ..., ... ]
```

Berapa banyak jumlah method yang tersedia? (misal: String)

```
>> nama.class
=> String

>> nama.methods.count
=> 192
```

> String Method yang Biasa Digunakan

.upcase()

```
>> nama = 'Yukihiro Matsumoto'  
>> nama.upcase  
⇒ "YUKIHIRO MATSUMOTO"
```

.downcase()

```
>> nama = 'Yukihiro Matsumoto'  
>> nama.downcase  
⇒ "yukihiro matsumoto"
```

.capitalize()

```
>> nama = 'yukihiro matsumoto'  
>> nama.capitalize  
⇒ "Yukihiro matsumoto"
```

.reverse()

```
>> nama = 'yukihiro matsumoto'  
>> nama.reverse  
⇒ "otomustam orihikuy"
```

.length()

```
>> nama = 'Yukihiro Matsumoto'  
>> nama.length  
⇒ 18
```

More: Ruby-Doc - <https://ruby-doc.org/core-2.7.0/String.html>

> String Method yang Biasa Digunakan

.sub()

```
>> nama = 'Yukihiro Matz'  
>> nama.sub('Matz', 'Akita')  
⇒ "Yukihiro Akita"
```

.gsub()

```
>> nama = 'Budi Budi Bayu'  
>> nama.gsub('Budi', 'Bayu')  
⇒ "Bayu Bayu Bayu"
```

.strip()

```
>> nama = '    matsumoto    '  
>> nama.strip  
⇒ "matsumoto"
```

.split()

```
>> nama = 'Budi Budi Bayu'  
>> nama.split(' ')  
⇒ ["Budi", "Budi", "Bayu"]
```

dan masih banyak lagi...

More: Ruby-Doc - <https://ruby-doc.org/core-2.7.0/String.html>

> Escaping Character

backslashed
notation

Bagaimana cara melakukan escaping character pada String?

```
>> 'malam jum'at'
```

```
SyntaxError: unexpected local variable or method...
```

```
>> 'malam jum\'at'  
"malam jum'at"
```

Alternatif gunakan double quote untuk escaping single quote:

```
>> "malam jum'at"  
"malam jum'at"
```

Contoh-contoh escaping character

```
\' : single quote  
\" : double quote  
\\ : backslash  
\\n : new line  
\\r : carriage return
```

```
\\t : tab  
\\b : backspace  
\\f : form feed  
\\v : vertical tab  
est.
```

More: AppSignal - <https://blog.appsignal.com/2016/12/21/ruby-magic-escaping-in-ruby.html>

Working with String (b)

.Working with String (b)

> Get Input from User (Console)

> Get Input from User (Console)

Ruby use `gets()` method to get the user input (as a string).

`.gets`

```
>> print 'Nama kamu siapa? '; nama = gets
Yukihiro←
>> nama
⇒ "Yukihiro\n"
```

"\n" didapatkan saat kita menekan tombol enter.

Bagaimana cara menghilangkannya?

Gunakan `String#chomp` method.

`.gets.chomp`

```
>> puts "Nama kamu siapa? "; nama = gets.chomp
Yukihiro←
>> nama
⇒ "Yukihiro"
```

Source: Ruby-Doc - <https://ruby-doc.org/core-2.7.1/IO.html>

Working with Number

Ruby for Beginner
A Programmer's Best Friend

.Working with Number

- > Arithmetic Operation
- > Order Arithmetic Operation
- > Loop with Times
- > Upto & Downto

> Arithmetic Operation

Nothing new here...

```
>> 5 + 5  
⇒ 10
```

```
>> 10 * 2  
⇒ 20
```

```
>> 2**3  
⇒ 8
```

```
>> 10 % 2  
⇒ 0
```

```
>> 10.even?  
⇒ true
```

```
>> 4.odd?  
⇒ false
```

```
>> 10 / 2.0  
⇒ 5.0
```

```
>> 9.0 / 3  
⇒ 3.0
```

Source: Ruby-Doc - <https://ruby-doc.org/core-2.7.1/I0.html>

> Order Arithmetic Operation

Perhatikan urutan dari pengerjaan operator.

Gunakan jembatan “PEMDAS” untuk mengingatnya.

Parenthesis	()
Exponential	**
Multiplication	*
Division	/
Addition	+
Substraction	-

```
>> 1 + 2 * 3**4 / (5 - 6)  
⇒ -161
```

Source: Ruby-Doc - <https://ruby-doc.org/core-2.7.1/I0.html>

> Loop with times

Salah satu method yang mudah untuk melakukan looping / perulangan terhadap angka.

.times

```
?> 3.times do  
?>   puts 'DILo Balikpapan'  
>> end
```

```
DILo Balikpapan  
DILo Balikpapan  
DILo Balikpapan
```

```
?> 2.times do |n|  
?>   puts "Dilo ke-#{n}"  
>> end
```

```
DILo ke-0  
DILo ke-1
```

.times (one line)

```
>> 3.times { puts 'DILo Bppn' }
```

```
DILo Bppn  
DILo Bppn  
DILo Bppn
```

```
>> 2.times { |n| puts n }
```

```
0  
1
```

> Upto & Downto

Salah satu method yang mudah untuk melakukan looping / perulangan terhadap angka dengan rentang tertentu.

.upto().each

```
?> 1.upto(3).each do  
?>   puts 'HelloDev'  
>> end
```

```
HelloDev  
HelloDev  
HelloDev
```

```
?> 1.upto(3) do |n|  
?>   puts n  
>> end
```

```
1  
2  
3
```

.downto().each

```
?> 3.downto(1).each do  
?>   puts 'HelloDev'  
>> end
```

```
HelloDev  
HelloDev  
HelloDev
```

```
?> 3.downto(1).each do |n|  
?>   puts n  
>> end
```

```
3  
2  
1
```

Comparison & Logical Operators

.Comparison & Logical Operators

- > Comparison Operators
- > Logical Operators

> Comparison Operators

Membandingkan 2 (atau lebih) objek.

Dan membuat keputusan berdasarkan hasil perbandingan.

Operator	Deskripsi
<	Kurang dari
<=	Kurang dari sama dengan
>	Lebih dari
>=	Lebih dari sama dengan
==	Sama dengan
!=	Tidak sama dengan

📄 All these operators are methods,
and they return a boolean value.

Source: RubyGuides - <https://www.rubyguides.com/2018/07/ruby-operators/>

> Logical Operators

Membandingkan 2 (atau lebih) condition.

Dan membuat keputusan berdasarkan hasil perbandingan.

A	B	A && B	A B
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

```
if (Condition A) && (Condition B)
  # statement
end
```

```
if (Condition A) || (Condition B)
  # statement
end
```

Source: RubyGuides - <https://www.rubyguides.com/2018/07/ruby-operators/>

Branching /Condition

. Branching / Condition

- > If
- > Unless
- > If/Unless for One Statement
- > If, Else
- > If, Elsif, Else
- > If with Multiple Condition
- > If Nested (bertingkat)
- > Case
- > Ternary Operator

> if Statements

```
if condition
  statement 1
  statement 2
end
```

```
if condition then statement 1; statement 2 end
```

```
if condition; statement 1; statement 2 end
```

📄 Ruby executes code if the conditional is true.
True is interpreted as anything that isn't false or nil.

Source: RubyGuides - <https://www.rubyguides.com/ruby-tutorial/ruby-if-else/>

> unless Statements

```
if !condition      # ← negasi dari condition
  statement
end
```

Sama dengan,

```
unless condition
  statement
end
```

```
unless condition then statement end
```

```
unless condition; statement end
```

📄 Ruby executes code if the condition is **true**.
True is interpreted as anything that isn't false or nil.

Source: RubyGuides - <https://www.rubyguides.com/ruby-tutorial/ruby-if-else/>

> if/unless (reversed form)

```
statement if condition  
statement 1; statement 2; statement n; if condition
```

```
statement unless condition  
statement 1; statement 2; statement n; unless condition
```

Ruby executes code if the condition is true.
True is interpreted as anything that isn't false or nil.

Source: RubyGuides - <https://www.rubyguides.com/ruby-tutorial/ruby-if-else/>

Collection

Ruby for Beginner
A Programmer's Best Friend

Looping

Ruby for Beginner
A Programmer's Best Friend

Method

Ruby for Beginner
A Programmer's Best Friend

EXTENDED

Ruby for Beginner
A Programmer's Best Friend

.File IO

.Error Handling

.Object Oriented Programming

Ruby for Beginner
A Programmer's Best Friend

.Module

Ruby for Beginner
A Programmer's Best Friend

> Tips mencari dengan Google

Gunakan prefix "ruby" sebagai awalan kata kunci untuk mencari di Google.

 **ruby**

Kemudian, diikuti dengan keyword apa yang mau dicari. Misal, tentang perulangan dengan method "each".

 **ruby each loop**

Dengan cara ini, kita dapat membuat hasil pencarian yang lebih spesifik untuk bahasa pemrograman Ruby.

> Referensi Belajar Ruby

YouTube:

- GoRails
- Jesus Castello
- Drifting Ruby
- Nate Berkopec
- TechmakerTV
- Decypher Media
- zayne
- Code School
- FreeCodeCamp.org
- Sekolah Koding
- Agung Setiawan

WebPage:

- Ruby in 20 minutes
- RubyGuides.com

Online Course:

- IDRails.com

> Referensi Group Telegram ➤ untuk Bertanya seputar Ruby

- Ruby Indonesia
- Rails Indonesia

> Referensi Channel Telegram ➤ untuk Berita tentang Ruby

- Ruby/Rails Inside

: Terima_Kasih

> Made with ❤ by Rizqi Nur Assyaufi
@BanditHijo
📅 2020/11