DEV Create account

Eric The Coder
Posted on Dec 14, 2020 • Updated on Aug 8, 2021

67 9 1

My beloved Ruby Cheat Sheet

#ruby #rails #beginners

Follow me!: Follow @EricTheCoder_

Here is my cheat sheet I created along my learning journey. If you have any recommendations (addition/subtraction) let me know.
Naming Conventions

```
#Snake Case for files
customer_import.rb

#Snake Case for Methods, Variables and Symbols
first_name = 'Mike'
def display_customer
  # some code
end
:light_red
```

```
#CapitalCase for Classes and Modules
class ProductManager
  # some code
end
module CustomerSupport
  # some code
end
```

Variables declaration

```
# string
full_name = 'Mike Taylor'

# integer
count = 20

# float
book_price = 15.80

# booleans
active? = true
admin_user? = false

#Array
fruits = ['Appel', 'Orange', 'Banana']

#Hash
fruit_color = { apple: 'red' }

#Array of hash
customers = [
  { id: 1000, name: 'Clark and Son' },
  { id: 1001, name: 'Clean Fast Co' },
  { id: 1002, name: 'Import International' }
]

#Struct
Person = Struct.new(:name, :age)
person1  = Person.new 'mike', 50
person2 = Person.new 'john', 35

#Set to 'Default title' only if nil or false
title = custom_title || 'Default title'
```

```
#Safe navigation operator &. (skip if nil)
name = customer&.first_name
```

print a string to the screen

```
#print with line break
puts 'This string will print on screen'

#print with no line break
print 'The string will print with no line break'

#print var content (debug)
puts customers.inspect
```

string methods

```
# String interpolation
name = 'Mike'
message = "Hello #{name}" # Hello Mike

# get string number of characters
'This is a string'.length  # 16

#check if the string is empty
'Hello World'.empty?   # false
''.empty?   # true

#convert all characters to uppercase
'hello world'.upcase  # HELLO WORLD

#convert all characters to lowercase
'HI'.downcase  # hi

#convert first characters to uppercase and the rest to lowercase
'mikE'.capitalize  # Mike

#remove white space
'  This is a string with space  '.strip

#return a string left justified and padded with a character
'hello'.ljust(20, '.')  # 'hello...............'
```

```
#chaining 2 or more methods
'Hello World'.downcase.include? 'world' # true

#index position (start at postion 0)
'Welcome to this web site'.index('this') # 11

#return string character(s) (start at position 0)
'This is a string'[1]  # h
'This is a string'[0..3]  # This
'This is a string'[-1]  # g (last character)

#replace first sub string
'Hello dog my dog'.sub 'dog', 'cat'. # Hello cat my dog

#replace all sub string
'Hello dog my dog'.gsub 'dog', 'cat'. # Hello cat my cat

#split a string into an array
'Apple Orange Banana'.split ' '  #['Apple', 'Orange', 'Banana']

# get console keyboard input
input = gets

# get input and chomp last char (ex. new line)
input = gets.chomp

# get command-line arguments (ex. ruby main.rb arg1 arg2)
puts ARGV  # ['arg1', 'arg2']

ARGV.each { |option| puts option }
```

Numbers

```
number.round 2.68  # 3
number.floor 2.68  # 2
number.ceil 2.68   # 3

2.next  # 3

puts 3 / 2    # 1 (integers with integer result integer)
puts 3 / 2.0  # 1.5 (float with integer result float)

puts 2.even?  # true
```

```
random_number = rand(1..100)
```

Loop

```
loop do
  puts "Stop loop by using 'break' statement"
  puts "Skip one occurence by using 'next' statement"
end

while number < 100
  puts number
  number += 1
end

# Range
(1..10).each { |i| puts i }
(1..10).each do |i|
  puts i
end

10.times { puts "Hello World" }
```

Conditionals statement

```
# Equal ==   And &&   Or ||   Not !
if action == 1
  puts "action 1"
elsif action < 5
  puts "action not 1 but less than 5"
else
  puts "action greater than 5"
end

#Unless (negated if)
puts 'The user is not active' unless active == true

#Ternary operator
active ? 'The user is active' : 'The user is not active'

#Truthy or falsy
# false and nil equates to false.
# Every other object like 1, 0, "" are all evaluated to true
```

```
  "Not good"
when 1..50
  "Better but not great"
when 51..70
  "Thats good!"
when 71..99
  "Great"
when 100
  "Perfect"
else
  "Score error"
```

Array access

```
fruits = ['Apple', 'Orange', 'Banana']
fruits = %w(Apple Orange Banana)

fruits.length # 3

fruits.first  # Apple
fruits.last   # Banana

fruits[0]     # Apple
fruits[-2]    # Orange
fruits[3]     # nil
fruits[1..2]  # ['Orange', 'Banana']

# iteration
fruits.each do { |fruit| puts fruit }

fruits.each_with_index do |fruit, index|
  puts fruit  # Apple
  puts index  # 0
end
```

Array Methods

```
fruits.include? 'Orange'  # true
[1, 5, 2, 4, 3].sort  # [1, 2, 3, 4, 5]
[1, 2, 3].reverse  # [3, 2, 1]

fruits.push 'Strawberry' # append at the end
fruits <<  'Raspberry' # append at the end
fruits.unshift 'Strawberry' # Append in front
fruits.delete_at(0) # remove first element
fruits.shift  # remove the first element

fruits.join ', '  # 'apple, orange, banana'

# Add in a new array
array1 = %w(dog cat bird)
array2 = %w(fish hamster)
array3 = array1 + array2 #['dog', 'cat', 'bird', 'fish', 'hamster']

# Concat in the same array
array1.concat array2
puts array1  #['dog', 'cat', 'bird', 'fish', 'hamster']

# Constructing arrays with * splat operator
puts ['dog', *array2, 'bird']  #['dog', 'fish', 'hamster', bird']
```

Convert between type

```
123.to_s   # convert number to string "123"
"123".to_i # convert string to integer 123
"123".to_f # convert string to float 123.0

#convert to array
(1..10).to_a  # [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
('a'..'e').to_a # ['a', 'b', 'c', 'd', 'e']
```

Hash

```
product = {}
product['title'] = "Mac Book Pro"
product[:price] = 1599.99
product = { 'title' => 'Mac Book Pro', 'price' => 1599.99 }
product = { title: 'Mac Book Pro', price: 1599.99 }
puts product.fetch(:cost, 0)  # return default value 0
product.keys   # [:title, :price]
product.values # ['Mac Book Pro', 1599.99]

product.each do |key, value|
  puts key
  puts value
end
```

Date and time

```
christmas = Time.new(2020, 12, 25)  #
puts christmas.wday # return 5 (Thursday)

now = Time.now  # current time: 2020-12-13 03:08:15 +0000
now.year    # 2020
now.month   # 12
now.day     # 13
now.hour    # 3
now.min     # 8
now.sec     # 15
now.sunday? # true

past = Time.now - 20  # return current time minus 20 seconds
past_day = Time.now - 86400 # 60 secs * 60 mins * 24 hours
past_day = Time.now - 1.day # work only in Rail

#Format Time
# %d    Day of the month (01..31)
# %m    Month of the year (01..12) Use %-m for (1..12)
# %k    Hour (0..23)
# %M    Minutes
# %S    Seconds (00..60)
# %I    Hour (1..12)
# %p    AM/PM
# %Y    Year
# %A    Day of the week (name)
# %B    Month (name)

time = Time.new
time.strftime("%d of %B, %Y")    # "25 of December, 2020"
```

Regular Expression (editor: www.rubular.com)

```
zip_code = /\d{5}/
"Hello".match zip_code  # nil
"White House zip: 20500".match zip_code  # 20500
"White House: 20500 and Air Force: 20330".scan zip_code # ['20500', '20330']
"Apple Orange Banana".split(/\s+/) # ['Apple','Orange', 'Banana']
```

Functions

```
end
puts greeting('Paul')  # Hello Paul

# variable number of arguments
def greeting(*names)
  names.each { |name| puts name }
end

#naming parameters
def display_product(price, options = {})
  puts price, options[:hidden], options[:rounded]
end
display_product 1599, hidden: false, rounded: true
```

Map, Select, Detect, Reduce and Count

```
#map (return a modified array)
names = ['paul', 'john', 'peter']
names_capitalize = names.map do |name|
  name.capitalize
end
# ['Paul', 'John', 'Peter']

# short hand version
names_capitalize = names.map { |name| name.capitalize }

# Symbol to proc
names_capitalize = names.map &:capitalize

#select (return all match)
products = [
  { name: 'Mac Book Pro', active: true, price: 1599.99 },
  { name: 'iWatch', active: false, price: 599.99 },
  { name: 'iPad Pro', active: true, price: 699.99 },
]
active_products = products.select { | product | product[:active] }

#Detect (return first match)
first_active_product = products.detect { | product | product[:active] }

# Reduce (return one)
total = products.reduce(0) do |total, product|
  total = total + product[:price]
end
puts total  # 2899.97
puts nb_products # 1
```

Module

```
# Static module method
module Display
  def self.hello
    puts 'Hello'
  end
end
Display.hello

# Class Mix in
module Display
  def hello
    puts 'Hello'
  end
end

require_relative 'display.rb'
class Customer
  include Display
end
Customer.new.hello

# Module as namespace
module Person
  class Customer
    def initialize(name)
      @name = name
    end
  end
end
customer = Person::Customer.new('Mike Taylor')

# Constant
module Contact
  ACCESS_KEY = 'abc123'
  class Person
      ACCESS_KEY = '123abc'
  end
end
puts Contact::ACCESS_KEY
puts Contact::Person::ACCESS_KEY
```

```
...
def initialize
  greeting
end

private
  def greeting
    puts 'hello'
  end
end
```

OOP

```
# class declaration
class Product

end

# object instantiation
product = Product.new

# class declaration with constructor and instance variables
class Product
  def initialize(name, price, active)
    @name = name
    @price = price
    @active = active
  end
end
product = Product.new 'Mac Book Pro', 1599, true

# Getter and Setter
class Product
  # set
  def price=(value)
    @price = value
  end
  # get
  def price
    @price
  end
end

# attribute accessor (shorthand get & set)
```

```
  attr_writer :price  # write only
  ...
end
...
puts product.price  # 1599

# instance method
class Product
  ...
  def price_with_tax
    # reference to @price directly is not recommended
    self.price + (self.price * tax_percent / 100)
    # self keyword is optional
  end
end
...
puts product.price_with_tax # 1838.85

# private method
class Product
  ...
  private
    def profit
      ...
    end
end
...
puts product.profit # NOT ALLOWED

#static class method and static class variable (use self keyword)
def self.calc_tax(amount)
 @@count = 1
end
puts Product::calc_tax(1599.99)

# Constant
class Product
  MIN_PRICE = 100

  def price=(price)
    if price < MIN_PRICE
      @price = MIN_PRICE
    else
      @price = price
    end
```

```ruby
# Inheritance
class Customer < Person
  attr_accessor :number

  def initialize(name, number)
    # super call the parent same method name
    # when call without parentheses then all arguments are pass
    # if call with empty arguments () then no arguments pass
    super(name)
    @number = number
  end

  def price=(price)
    # super call the parent price method
    super(price)
    @price += 100
  end
end
```

File I/O

```ruby
# Read
text = File.read('exemple.txt')

# Read by lines
lines = File.readlines("exemple.txt")
lines.each do |line|
  puts "Line: #{line}"
end

# Write
File.write('exemple.txt', 'text to write...')

File.open “index.htm ”, “ ”) do | file |
  file.puts ‘text to writ ’
end

#read csv
require 'csv'
table = CSV.parse(File.read("products.csv"), headers: true)
table[0]["id"] # 1000
table[0]["name"] # "Mac Book Pro"
```

```
  { name: "IPad Pro", price: 799 }
]
CSV.open("products.csv", "w", headers: products.first.keys) do |csv|
  products.each { |product| csv << product.values }
end
```

Errors/Exceptions Handling

```
# Raise exception and output error message
raise "This is an exception"

# Debut variable value
raise products.inspect # [{:id=>10, :name=>"ipad pro"},{:id=>20, :name=>"Mac

# Exception handling
begin
  # Any exceptions here ex. 0 / 1
  0 / 1
rescue
  # ...will make this code to run
  puts "Exception"
  do_something()
end

# Exception object
begin
  0 / 1
rescue ZeroDivisionError => e
  puts e.class.name
  puts e.message
end
```

Top comments (9)

Tom Mulkins • Feb 22 '22

Thanks for putting this out there.

I have one suggestion:

Consider making "dealing with Nil values" its own section. :)

This is great!

Norris Mei • Oct 4 '23

Thank you for this summary! I'm brushing up my Ruby skills and this cheat sheet jogged a lot of things for me.

For the exception handling part, I think you meant 1 / 0 for ZeroDivisionError. It's written as 0 / 1 in two lines and one comment, which is just 0 and shouldn't produce an error.

Sylwia Vargas • Feb 22 '21

Oh wow!! I love this. I'm going to read it in depth tomorrow to my morning coffee!

Khaireddine Hamdi • Jul 23 '21

This article is very great, thanks

Lee • Dec 15 '20

Have added it to my snippets :D

Oleg Puzanov • Dec 15 '20

This is truly helpfull. Also want to have the same for PHP and Python

Eric The Coder • Dec 23 '20

Updated the cheat cheat with couples more info related to array contact and couples other things.

zachee • Feb 21 '22

arm Arm PROMOTED •••

[Learn How Ruth Amos Supercharges Developer Creativity]

👦🏽 Tap into your child-inspired creativity.
🤝 Embrace fearless problem-solving.
💡 Create ground-breaking solutions.
🔑 Level up your design and development skills.
🏆 Learn from an award-winning engineer and innovator.

DEV

Create account

Eric The Coder

Businessman and blogger #Javascript, #Python and #PHP. My favorite frameworks/librairies are #React, #Laravel, and #Django. I am also a fan of #TailwindCSS

LOCATION
Canada

JOINED
Sep 3, 2020

More from Eric The Coder

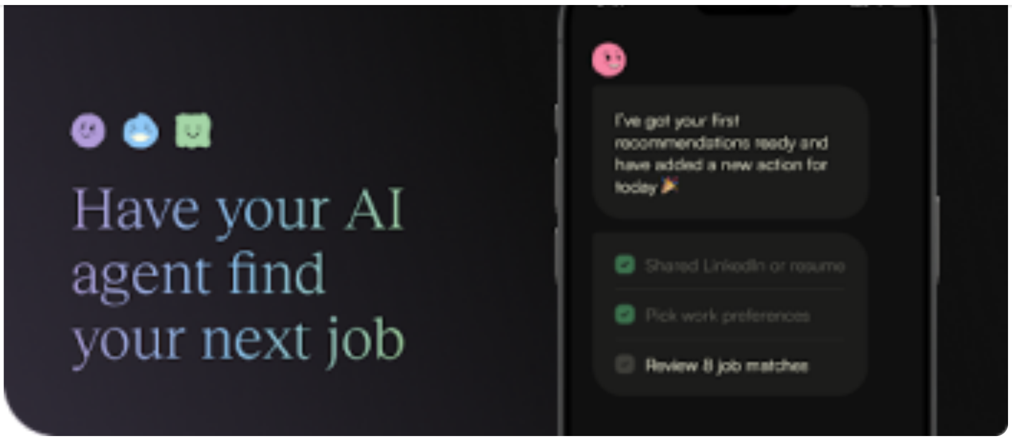
Python : String Manipulations
#python #tutorial #beginners

Python : Crash Course
#python #tutorial #beginners

PHP crash course : require, include, files manipulation and enumerations
#php #backends #tutorial #beginners

Commit PROMOTED

Is looking for your next developer job actually a brutal process?


Have your AI agent find your next job
I've got your first recommendations ready and have added a new action for today 🎉
Shared LinkedIn or resume
Pick work preferences
Review 8 job matches

Commit is an AI Agent that actually finds jobs for you

Commit's AI Talent Agent does the career-advancement heavy lifting for you – researching, finding, and applying to jobs that fit your developer profile perfectly.

It is still in beta, but we are fast-tracking DEV readers who sign up via this link:

Get started now!