The Ultimate Ruby Cheatsheet

Learning Ruby can be overwhelming with all the bits & pieces you have to remember.

That's why I put together this reference for you!

It will help refresh your memory & quickly review what you need to know to write Ruby programs.

Have fun!

Strings

A string is a sequence of characters inside two quotation marks (`""`). Used to represent text & data.

Example:

```
"I like chocolate"
```

You can also use single quotation marks (`''`).

Important methods:

- size
- empty?
- include?
- gsub
- split

More methods:

https://www.rubyguides.com/2018/01/ruby-string-methods/

Hashes

A hash (`{}`) is a key-value pair (`a => b`) data structure. Used as a dictionary. You can access hash elements by their keys. Keys are unique.

Example:

```
# Create
```

```ruby
h = { a: 1, b: 2, c: 3 }
# Access
h[:a]
# Set
h[:test] = 10
```

Important methods:

- key?
- fetch
- new (for default values)
- merge

More methods:

http://ruby-doc.org/core-2.6.4/Hash.html

Symbol

A static string used for identification, one common example is hash keys. They always start with a colon (`:bacon`). Symbols are never used for their content (the individual characters).

Learn more:

https://www.rubyguides.com/2018/02/ruby-symbols/

Nil

A singleton class (only one object allowed) that represents a default or "not found" kind of value.

Evaluates to "false" in a conditional context.

Learn more:

- https://www.rubyguides.com/2018/01/ruby-nil/
- https://www.rubyguides.com/2019/02/ruby-booleans/

Array

An object used to represent a list of objects. An array can contain any kind of object (`a = [1, "abc", []]`), including other arrays.

You access array elements with their index (`a[0]`) & nested arrays with `a[0][0]` .

Important methods:

- size
- empty?
- push / pop
- join
- flatten

More methods:

http://ruby-doc.org/core-2.6.4/Array.html

Enumerable

A Ruby module used to iterate over the elements of any class that implements the `each` method, like Array, Range & Hash.

Important methods:

- map
- select
- inject

More:

https://www.rubyguides.com/2016/03/enumerable-methods/

File

A class that helps you work with files in Ruby. Anything from reading them, writing to them or even getting info about them, like the file size.

Important methods:

- read
- write

More:

https://www.rubyguides.com/2015/05/working-with-files-ruby/

Regular Expression

If you're looking to find patterns, substrings, or something specific inside a string, then a regular expression may be what you're looking for.

They can be used to validate email addresses & phone numbers. Or to extract information from text.

Example:

```
"aaaa1".match?(/[0-9]/)
# true
"".match?(/[0-9]/)
# false
```

Learn more:

https://www.rubyguides.com/2015/06/ruby-regex/

Ruby Gems & Bundler

Ruby gems are packages you can download to use in your Ruby programs.

These packages give you new functions.

For example, in Rails you can easily add authentication with the Devise gem, or pagination with the Kaminari gem.

Learn more:

https://www.rubyguides.com/2018/09/ruby-gems-gemfiles-bundler/

Classes & Object-Oriented Programming

Ruby is an Object-Oriented Programming language. We think of everything as an object. Objects are created from their blueprints, classes.

Objects can know things & do things. You tell objects to do things with methods.

Important methods:

- class
- include / extend

Learn more:

- https://www.rubyguides.com/2019/02/ruby-class/

- https://www.youtube.com/watch?v=LuTTUNnSj6o&list=PL6Eq_d2HYExeKIi4d9rUEoD6qSiKS4vfe&index=2

Types Of Variables

A variable is a label for an object that we can use to access that object.

We use different kinds of variables in Ruby.

Here's a list:

- Local variables (`something`)
- Instance variables (`@something`)
- Constants (`Something` / `SOMETHING`)
- Global variables (`$something`)

The main difference is from what locations you can access them.

%w, %i, %q, %r, %x

There is a way to create objects with a special kind of syntax, the percentage symbol.

Examples:

```
array_of_strings = %w(apple orange coconut)
array_of_symbols = %i(a b c)
string = %q(things)
regular_expression = %r([0-9])
```

Remember that the percentage symbol (`%`) is also used as the modulo mathematical operator.

Use of Parenthesis

Parenthesis & semicolons are not required in Ruby, but they can be used.

Some basic rules:

- DON'T USE parenthesis when defining a method with no arguments => `def foo`
- USE parenthesis with method arguments => `def foo(a, b, c)`
- USE parenthesis when you want to change the precedence, or priority, of an operation => `(a.size + b.size) * 2`

Readability is one use for parenthesis, while changing order of operations is another.

Syntax Examples

Method definition

```ruby
def apple(a,b,c)
  # method body
end
```

More about methods:

- https://www.rubyguides.com/2018/06/rubys-method-arguments/
- https://www.rubyguides.com/2019/06/ruby-method-definition/

Class definition

```ruby
class Fruit
  # methods
end
```

More about classes:

- https://www.rubyguides.com/2019/02/ruby-class/
- https://www.rubyguides.com/2019/01/what-is-inheritance-in-ruby/

Ternary operator

```ruby
true ? "yes" : "no"
```

String interpolation

```ruby
fruit = "orange"
puts "I have an #{fruit}. Would you like a slice of it?"
```

More about interpolation:

- https://www.rubyguides.com/2019/07/ruby-string-concatenation/

Each with block

```ruby
[1,2,3].each do |n|
  puts n
end
```

If / Else

```ruby
n = 20
if n > 1
  puts "Greater than 1"
else
  puts "Less than 1"
end
```

Case statement

```ruby
case 20
when 1..20
  puts "Between 1 & 20"
when 21..40
  puts "Between 21 & 40"
else
  puts "Not within a valid range"
end
```

Where `1..20` is a `Range` object.

Thank You

Review these often until it becomes built into your brain.

If you haven't read it yet, you may also enjoy my [Ruby book](#).

Thanks for reading & have a nice day!

- Jesus Castello