Ruby Programming Language Cheatsheet

A comprehensive cheat sheet covering essential Ruby syntax, data structures, control flow, and object-oriented programming concepts.

Ruby Basics

Syntax

Comments	`# This is a single-line comment`
	`=begin`
	`This is a multi-line comment`
	`=end`

Variables	`variable_name = value` (snake_case)
Constants	`CONSTANT_NAME = value` (UPPER_SNAKE_CASE)
String Interpolation	`"Hello, #{variable_name}!"`
Blocks	`do ... end` or `{ ... }`

Methods	`def method_name(arg1, arg2)`
	`  # method body`
	`  return value`
	`end`

Data Types

- **Numbers:** Integers (`1, 2, 3`), Floats (`1.0, 2.5`), Rational (`1/2`)
- **Strings:** `"Hello, world!"` , `'Single quotes'`
- **Booleans:** `true` , `false`
- **Symbols:** `:symbol_name` (immutable strings)
- **Arrays:** `[1, 2, "three"]`
- **Hashes:** `{ :key1 => "value1", "key2" => 2 }`
- **Nil:** `nil` (represents absence of value)

Operators

Arithmetic	`+, -, *, /, %, **` (exponentiation)
Comparison	`==, !=, >, <, >=, <=, <=>`
Logical	`&& (and), || (or), ! (not)`
Assignment	`=, +=, -=, *=, /=, %=, **=`

Control Flow

Conditional Statements

If Statement

```
if condition
  # code to execute if true
elsif other_condition
  # code to execute if other_condition is true
else
  # code to execute if false
end
```

Unless Statement

```
unless condition
  # code to execute if condition is false
else
  # code to execute if condition is true
end
```

Ternary Operator

```
condition ? true_value : false_value
```

Case Statement

```
case variable
when value1
  # code to execute if variable == value1
when value2
  # code to execute if variable == value2
else
  # code to execute if no other value matches
end
```

Loops

While Loop

```
while condition
  # code to execute while true
end
```

Until Loop

```
until condition
  # code to execute until true
end
```

For Loop

```
for variable in collection
  # code to execute for each element
end
```

Each Iterator (Array/Hash)

```
array.each do |element|
  # code to execute for each element
end

hash.each do |key, value|
  # code to execute for each key-value pair
end
```

Loop Control

`break` - exits the loop.

`next` - skips the current iteration.

Exception Handling

```
begin
  # code that might raise an exception
rescue SpecificError => e
  # code to handle specific exception
rescue => e
  # code to handle other exceptions
ensure
  # code that always executes (optional)
end
```

Object-Oriented Programming

Classes and Objects

Class Definition

```ruby
class ClassName
  # attributes and methods
end
```

Creating Objects

```ruby
object = ClassName.new
```

Attributes (Instance Variables)

```ruby
class ClassName
  attr_accessor :attribute1, :attribute2  # getter and setter
  attr_reader :attribute3  # getter only
  attr_writer :attribute4  # setter only
end
```

Instance Methods

```ruby
class ClassName
  def method_name(arg1, arg2)
    # method body
  end
end
```

Class Methods

```ruby
class ClassName
  def self.method_name
    # method body
  end
end
```

Constructor (initialize method)

```ruby
class ClassName
  def initialize(arg1, arg2)
    @attribute1 = arg1
    @attribute2 = arg2
  end
end
```

Inheritance

Inheriting from a Class

```ruby
class SubClass < SuperClass
  # override methods or add new ones
end
```

Calling Superclass Methods

```ruby
def method_name(arg1, arg2)
  super(arg1, arg2)
  # additional code
end
```

Modules

Module Definition

```ruby
module ModuleName
  # constants and methods
end
```

Including Modules

```ruby
class ClassName
  include ModuleName
end
```

Extending Modules

```ruby
class ClassName
  extend ModuleName
end
```

Using Modules as Namespaces

```ruby
ModuleName::CONSTANT
ModuleName.method_name
```

Common Methods

String Methods

`length`	Returns the length of the string.
`upcase`	Converts the string to uppercase.
`downcase`	Converts the string to lowercase.
`strip`	Removes leading and trailing whitespace.
`split(delimiter)`	Splits the string into an array based on the delimiter.
`include?(substring)`	Checks if the string contains the substring.

Array Methods

`length` or `size`	Returns the number of elements in the array.
`push(element)` or `<< element`	Adds an element to the end of the array.
`pop`	Removes and returns the last element of the array.
`shift`	Removes and returns the first element of the array.
`unshift(element)`	Adds an element to the beginning of the array.
`include?(element)`	Checks if the array contains the element.

Hash Methods

`length` or `size`	Returns the number of key-value pairs in the hash.
`keys`	Returns an array of all keys in the hash.
`values`	Returns an array of all values in the hash.
`has_key?(key)`	Checks if the hash contains the key.
`has_value?(value)`	Checks if the hash contains the value.
`delete(key)`	Deletes the key-value pair from the hash.