

### What is Ruby?

Ruby is object-oriented language, which means that everything in Ruby is an object, even the most basic data types.

- It is simple & readable language
- Implemented based on python and perl
- Ruby follows convention over configuration and DRY

### Convention Over Configuration and DRY

**COC:** Reduces need for extensive setup & makes coding quicker & simple.

Ex: method names and variable names should match the logic

**DRY:** Encourage reducing repetition in code to make it more maintainable & efficient.

### History

Developed by "Yukihiro Matsumoto" in the mid - 1990s in Japan.

### Features of Ruby

It is Object-oriented programming language known for easy syntax & focus on simplicity & productivity.

Supports for blocks, iterators, mixins for code reuse.

Follows DRY & COC

### Why everything in ruby is Object?

Everything in ruby is object including datatypes because, language is designed to treat all entities uniformly.

Allowing for consistent behaviour & functionality across datatypes.

### Variables

Use to store values in memory, allowing to access & modify data as you need.

#### Types

**Local variable:** Accessible within scope, they defined within a method or block

**Instance variable(@):** Associated with instance of class, accessible across methods in instance.

**Class variable(@@):** Accessible among all instance of class.

**Global variable(\$):** Accessed anywhere in program.

### OOPs concept

**Class:** A blueprint for creating objects; it defines attributes and methods. In Ruby, classes are created using the class keyword. **Object:** An instance of a class, holding unique data and behaviors as defined by its class.

**Encapsulation:** Bundles data and methods within class, controlling access to them Ex: attr\_reader, attr\_writer, attr\_accessor

**Inheritance:** Allows class to inherit behaviours & attributes from another class.

**Polymorphism:** It is the ability to represent an operator or function in different ways

**Abstraction:** It hides the complexity of a class by modelling classes appropriate to the problem.

### Data Types In Ruby

1. **Strings:** Represented by single('Hi') or double quotes("Hi")

2. **Numbers:** It can be integers(10) or float(3.14)

3. **Arrays:** Collection of objects. [1,2,3]

4. **Hashes:** Key-value pair. {"name"=>"John"}

5. **Symbols:** Light weight, immutable strings are used as identifiers (:name)

### Initialize method

It is special instance method in ruby called when a new object is created.

### RVM(Ruby version Manager)

It is a command line tool allows easily to install, manage & work with different ruby environments

### Loops

**Loops** allows you to repeat set of instructions multiple times.

1. **for loop:** Used to iterator over range/ collection

2. **while loop:** Runs code as long as condition is true.

3. **Until loop:** Opposite of while loop. Runs code until a specific condition becomes true.

4. **Nested for loop:** For loop inside other for loop. Allows to perform iterators over multi-dimensional DS.

### Array

\* Array is collection of objects, they hold objects like integer, number, hash, string, symbol etc..

\* Index starts with 0

### Operators

Operators are symbols used to perform operations

**Unary Operator:** ! ~ +

**Arithmetic Operator:** + - / % \*\*

**Assignment Operator:** = += -= /= %=\*\*=

**Comparison Operator:** == != > >= < <= ===

**Bitwise Operator:** & | ^ ~ << >>

**Logical Operator:** && || ! and or

**Ternary Operator:** ? !

**Range:** ... ..

### Hash

\* Hash is Data structure which stored key-value pair

\* Useful for storing data with unique identifiers(key), as you can access by key

### Object

Object is instance of class that combines data(attributes) & behaviour(methods).

Everything in ruby is object link numbers, strings, arrays etc.

### Methods

Set of instructions/ code that perform specific task.

Used to encapsulate reusable code, making program more organised

They can take arguments, perform actions & return value.

### Class methods Vs Instance methods

**Class methods:** Methods belongs to class itself, allowing them to call without creating instance of class.

\* We use self keyword in class

**Instance methods:** Belongs to instance of class & can only called on an object created from that class.

### Method Overloading & Method Overriding

**Overloading:** A class containing more methods with same name with different parameters

**Overriding:** Parent class and child class having same name with different functionalities. when you call, it executes parent class only.

### Constant & Iterators

**Constants** are variables that cannot change once we declare.

**Iterators** are methods that naturally loop over a give set of data & allow you to operate on each element in the collection.

### Module Vs Mixins

**Modules:** Containers for methods and constants that help organize code and avoid naming conflicts. Modules can't create objects on their own.

**Mixins:** A technique for adding module methods to a class as instance methods, allowing multiple classes to share functionality without inheritance.

### Hash VS Array

**Hash:** Key-Value pair combination. Key should be unique

**Array:** Collection of strings, numbers. Index starts with '0'. It is Heterogenous

### Class VS Module

**Class:** Class is a blueprint of object

\* Contains methods & attributes for its instance.

\* These methods are used as objects

**Module:** It is collection of class

### Self

- Self refers to current object instance

- It is used to access methods & variables within object itself.

### nil vs false

**nil:** Object represents nothing or no value

**false:** Boolean value that explicitly represents falsehood.

### Difference between == and ===?

**== operator:** Checks value equality.

Compare 2 value objects to see if they are same.

**=== Operator(Case equality):** Used in case statement & check whether object is in or matches another object.

Ex: Numeric === 5 #true because 5 is number

### sub VS gsub

**sub:** It will replace first matching item

Ex: I love cats. cats O/p: I love dogs. cats  
are great are great

**gsub:** It will replace all matching item in sentence

Ex: I love cats. cats O/p: I love dogs.  
are great dogs are great

### puts vs print vs p

**puts:** Output a string with newline at end.

**print:** Output a string without newline

**p:** Output value is more readable form, shows exact values including quotes.

### Procs VS lamda

**Proc(Procedure):** It is code block that stored in local variable, not in method.

**Lamda:** It is Short block of code which takes in parameters & returns value.

### Include VS Extend

**Include:** We need objects to call methods

**Extend:** We dont need to objects to call methods, directly we can call with class.

### eq!? Vs equal? methods

**eq!?**: Used in hases to check if key have same values & type, allowing them match.

Ex: `puts "hello".eq!?("hello") #True`

**equal?**:Used for key comparison. Checks if two variables reference same object in memory.

Ex: `a="hello, c=a`

`puts a.equal?(c) #True(same object)`

### Each Vs Map

**Each**: executes the given block for each element of the array, then returns the array itself.

**Map**: executes the given block for each element of the array, but returns a new array whose values are the return values of each iteration of the block.



By **Akshaya** (akki\_1)  
[cheatography.com/akki-1/](https://cheatography.com/akki-1/)

Published 14th November, 2024.

Last updated 14th November, 2024.

Page 3 of 3.

Sponsored by **ApolloPad.com**

Everyone has a novel in them. Finish

Yours!

<https://apollopad.com>