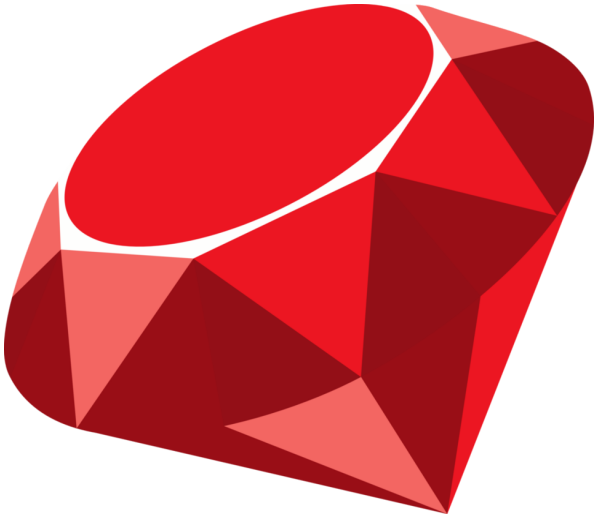
# RUBY

## LENGUAJES DE PROGRAMACIÓN



Hash: **2934**

LP - GEI FIB

Primavera 2019-2020

# Índice de contenidos

**Historia de Ruby** **3**

**Propósito del lenguaje** **4**

**Paradigma de programación** **5**

**Sistema de tipado** **6**

**Sistema de ejecución** **6**

**Principales aplicaciones** **7**

**Características particulares** **7**

**Ejemplos de código ilustrativo** **8**

VARIABLES 8

IF - THEN - ELSE 8

BLOCKS 9

LOOPS 9

ARRAYS 10

CLASES: HERENCIAS Y POLIMORFISMO 10

MÓDULOS 11

**Bibliografía** **12**

Fuentes de información utilizadas 12

Calidad de la información utilizado 12

Referencias 13

# Historia de Ruby

El lenguaje de programación Ruby fue creado por el programador japonés Yukihiro Matsumoto en la década de los 90. Más concretamente, comenzó su desarrollo en 1993 y fue publicado oficialmente en 1995.

Matsumoto deseaba contar con un lenguaje de scripting orientado a objetos fácil de usar. Opciones cercanas como Python o Perl las consideraba a mitad de camino, así que tomando la simplicidad del lenguaje funcional Lisp, la orientación a objetos de SmallTalk, la practicidad de Perl y pinceladas de Eiffel y Ada decidió crear este nuevo lenguaje.

Según su autor, el lenguaje está enfocado en la productividad y diversión del desarrollador, es decir, es un lenguaje creado por y para el desarrollador.

Tras el primer lanzamiento en el 95, a lo largo de los años se han publicado diferentes versiones estables:

- Ruby 1.0: Diciembre 1996
- Ruby 1.2: Diciembre 1998
- Ruby 1.4: Agosto 1999
- Ruby 1.6: Septiembre 2000

Sin embargo, la primera versión de Ruby de larga duración, fue Ruby 1.8, publicado en 2003 y retirado en 2013. Hasta la fecha se han publicado múltiples versiones del lenguaje (véase figura 1). Como dato curioso, a partir del lanzamiento de Ruby 2.0, todos los posteriores se han realizado el día de Navidad de los respectivos años.

Algo que hizo aumentar el interés en el lenguaje Ruby, fue el lanzamiento del Framework Web Rails (conocido como Ruby on Rails, o sus siglas RoR) en 2005.

| Version | Latest teeny version | Initial release date | End of support phase | End of security maintenance phase |
|---|---|---|---|---|
| 1.0 | NA | 1996-12-25[55] | NA | NA |
| 1.8 | 1.8.7-p375[56] | 2003-08-04[57] | 2012-06[58] | 2014-07-01[59] |
| 1.9 | 1.9.3-p551[60] | 2007-12-25[61] | 2014-02-23[62] | 2015-02-23[63] |
| 2.0 | 2.0.0-p648[64] | 2013-02-24[65] | 2015-02-24[64] | 2016-02-24[64] |
| 2.1 | 2.1.10[66] | 2013-12-25[67] | 2016-03-30[68][69] | 2017-03-31[70][71] |
| 2.2 | 2.2.10[72] | 2014-12-25[73] | 2017-03-28[74] | 2018-03-31[71] |
| 2.3 | 2.3.8[75] | 2015-12-25[76] | 2018-06-20[77] | 2019-03-31[77] |
| 2.4 | 2.4.10[78] | 2016-12-25[79] | 2019-04-01[80] | 2020-04-01[80] |
| 2.5 | 2.5.8[81] | 2017-12-25[82] | TBA | TBA |
| 2.6 | 2.6.6[83] | 2018-12-25[84] | TBA | TBA |
| **2.7** | 2.7.1[85] | 2019-12-25[86] | TBA | TBA |
| 3.0 | | 2020[87][88] | TBA | TBA |
| **Legend:** | Old version | Older version, still maintained | **Latest version** | Future release |

*Figura 1:* ***Tabla de versiones de Ruby*** *| Fuente: Wikipedia*

# Propósito del lenguaje

En la propia web de Ruby se describe al mismo como:

> *"Ruby es un lenguaje de programación dinámico y de código abierto enfocado en la simplicidad y productividad. Su elegante sintaxis se siente natural al leerla y fácil al escribirla."*

Ruby es un lenguaje de scripting de propósito general, que tiene como principal objetivo, ser amigable con el programador evitando al mismo situaciones inesperadas. Debido a los nuevos frameworks que se van lanzando, los principales usos que los programadores dan al lenguaje es en el ámbito de las Web Apps (server-side).

# Paradigma de programación

Ruby es bien conocido por ser un lenguaje multiparadigma. A pesar de esto, el paradigma asociado al mismo por defecto, debido a sus fuertes inspiraciones en SmallTalk, es la **Orientación a Objetos**. En Ruby todo es un objeto al que se le pueden asignar propiedades y acciones. De hecho, los tipos básicos o primitivos con los que se construyen objetos son objetos en sí mismos (el número 5 es un objeto, el carácter 'a' es un objeto, etc.) y como tales, pueden recibir métodos y variables de instancia.

```ruby
5.times { print "Nos *encanta* Ruby -- ¡es fuera de serie!" }
```

*Figura 2:* ***Método definido para el objeto "5"*** *| Fuente: Ruby's Website*

Otro paradigma en el que se puede incluir a Ruby es el paradigma **Procedural**. A diferencia de otros lenguajes, en Ruby solo contamos con métodos, los cuales solo pueden ser definidos dentro de clases. Sin embargo nos permite definir métodos aparentemente fuera de ninguna clase, es decir, sin estar asociado a ningún objeto. Esto no es más que una ilusión para el programador, ya que todo método no definido dentro de una clase, se está definiendo como método del objeto principal main. Lo cual nos lleva a que realmente, estamos dentro del paradigma de la Orientación a Objetos, pero desde el punto de vista del programador, podemos tratarlo como Procedural.

```ruby
def procedure1
  "I'm processing stuff.."
end
```

Figura 3: **Ejemplo de función definida fuera de una clase** | Fuente: GitHub Gist

Finalmente, también podemos incluirlo en el paradigma de la programación **funcional**. A diferencia de en OO, donde todo gira entorno a objetos, en el mundo de los lenguajes todo gira entorno a funciones. En Ruby podemos definir objetos que encapsulan funciones lambda. Esto permite el uso/creación de funciones de orden superior como entrada de métodos, recibiendo el objeto que encapsula la función lambda como parámetro, y/o retornarlos como salida.

```ruby
add = -> (a, b) { a + b }

def process(operation, a, b)
  operation.call(a, b)
end
```

Figura 4: **Ejemplo de función de orden superior** | Fuente: GitHub Gist

# Sistema de tipado

Este lenguaje acoge el sistema de tipado coloquialmente conocido como “Duck Typing”, el cual hace referencia a un sistema de tipado dinámico de los datos (en tiempo de ejecución), decidiendo en base a la presencia de ciertos métodos y propiedades si el tipo es válido o no en un determinado punto de la ejecución. Se considera que tiene un tipado fuerte ya que, en el instante en el que el intérprete compila una instrucción para ejecutarla, comprueba si la operación es válida para el tipo en cuestión.

# Sistema de ejecución

Ruby, como se ha mencionado múltiples veces a lo largo del documento, es un lenguaje de scripting interpretado, ya que en tiempo de ejecución, el intérprete compila el código y realiza las comprobaciones pertinentes dinámicamente. El lenguaje cuenta con diferentes intérpretes, pero el de referencia es el creado por el mismo autor del lenguaje, el Matz’s Ruby Interpreter (conocido por sus siglas MRI)

# Principales aplicaciones

Las principales aplicaciones de Ruby se encuentran en el desarrollo web, tanto de front-end como de back-end. Al ser un lenguaje de propósito general, también es utilizado en muchos otros ámbitos, como análisis de datos y prototipado.

Entre los proyectos exitosos en Ruby, tenemos el Framework Metasploit para testear la seguridad de diversos sistemas; o el gestor de paquetes para macOS más conocido, Homebrew, entre muchos otros.

Dado que el framework web más utilizado es Rails, la gran mayoría de las aplicaciones web en Ruby corren sobre el. Este permite desarrollar web de forma rápida a grupos de desarrollo pequenos.

Entre los proyectos exitosos más conocidos desarrollados en Rails, encontramos AirBnB, Hulu, Github, MyFitnessPal, entre muchos otros.

# Características particulares

Ruby es un lenguaje orientado a objetos muy expresivo, para muchos de los más cercanos a programar en “inglés”. Además, al ser multiparadigma, es un lenguaje versátil, apto para todos los gustos. En conjunto, como el autor dice, es un lenguaje fácil para el programador.

Por otro lado esto tiene el (común) problema de todo lenguaje de tal nivel de abstracción, la eficiencia. Ruby tiene problemas velocidad, debido a su tipado dinámico y a ser interpretado. Este presenta problemas de escalabilidad, por ejemplo Twitter, fue inicialmente desarrollado en Ruby, pero debido a sus necesidades de escalabilidad, tuvieron que migrar a Java.

Podríamos decir que Ruby es el lenguaje favorito de las Startups, debido a que permite iniciar proyectos de forma facil y rapida.

Entre otras características, encontramos que el lenguaje cuenta con mecanismos para tratar excepciones y errores, un recolector de basura para liberar memoria de forma automática e incluso la posibilidad de trabajar en multi-hilo sin importar el sistema operativo que haya bajo la ejecución.

# Ejemplos de código ilustrativo

## VARIABLES

```ruby
# Variable Local:
## Son visibles dentro del bloque que las inicializo.
age = 10
_Age = 20

# Variable Global:
## Son visibles desde cualquier bloque.
$age = 10

# Variables de Instancia:
## Todos los objetos de una clase cuentan con la variable,
## pero el valor es privado para cada uno de ellos.
@age = 20

# Variables de Clase:
## Todos los objetos de una clase comparten esa variable
## (incluido el valor).
@@age = 20
```

## IF - THEN - ELSE

```ruby
a = 10;
b = 15;

# if statement
if a % 2 == 0
    puts "Even Number"
end

# if-else statement
if b % 2 == 0
    puts "Even Number"
else
    puts "Odd Number"
end
```

Output:

```
Even Number
Odd Number
```

## BLOCKS

```ruby
# La funcion each permite aplicar un bloque de instrucciones
# a todos los elementos de la misma. En este fragmento podemos
# ver como llama a los elementos 'n' y los escribe.
["Esto", "son", "numeros"].each do |n|
    puts n
end
```

## LOOPS

```ruby
# Bucle for basico
for a in 1..5 do
    puts "Salu2"
end

# Bucle while basico
i = 0
while i < 10
    puts "Hey You"
    i = i + 1
end
```

```ruby
# Se puede hacer uso de breaks para controlar los bucles:
# Bucle do-while basico construido con breaks.
loop do
    val = '7'

    # using boolean expressions
    if val == '7'
        break
    end
end
```

## ARRAYS

```ruby
# Los arrays pueden contener elementos de distintos tipos.

myArray = ["Como", "Me", "Gusta", "Ruby"]

myArray[1]  ## Retorna "Como"
myArray[-1] ## Retorna "Ruby" (los indices son ciclicos)
```

## CLASES: HERENCIAS Y POLIMORFISMO

```ruby
# Super clase Salu2
class Salu2

    def initialize

        puts "Esto es la superclase"
    end

    def super_method

        puts "Metodo de la superclase"
    end
end

# Clase Hola, subclase de Salu2
class Hola < Salu2

    def initialize

       puts "Esto es una subclase"
    end
end

# Creacion de un objeto de superclase
Salu2.new

# Creacion de un objeto de subclase
sub_obj = Hola.new

# Mediante herencia, podemos llamar a metodos de
# la superclase desde la subclase
sub_obj.super_method
```

# MÓDULOS

```ruby
# Un modulo contiene variables, metodos, constantes y
# variables de clase. Se definen del mismo modo que una clase

# Creacion del modulo LP
# Utilizamos el nombre del modulo com prefijo
module LP

    Nota = 10;

    def LP.welcome
        puts "Welcome to LP!!"
    end

    def LP.tutorial
        puts "Leete el manual"
    end

    def LP.truth
        puts "Haskell > C++"
    end

end

# displaying the value of
# module constant
puts LP::Nota

# calling the methods of the module
LP.welcome
LP.tutorial
LP.truth
```

Output:

```
10
Welcome to LP!!
Leete el manual
Haskell > C++
```

# Bibliografía

## Fuentes de información utilizadas

Para realizar este documento me he basado principalmente en la documentación oficial del propio lenguaje en Español y en Inglés. Además también me he apoyado en una entrevista realizada al autor para la parte de historia y propósito del lenguaje.

Las muestras de código son adaptadas por mí en base a webs que tratan de iniciar a personas en la codificación de Ruby como Geeks for Geeks.

** Todas las fuentes utilizadas provienen de internet, dada la imposibilidad de consultar una biblioteca en esta la "Era del confinamiento".

## Calidad de la información utilizado

La información obtenida, se basa principalmente en la web del propio lenguaje y en una entrevista realizada al autor, por lo tanto, debo darle total veracidad y calidad máxima a la misma.

En cuanto a otras webs utilizadas, en ocasiones pecan de dar su mera opinión, pero dado que utilizan ejemplos para su demostración, podía yo mismo comprobar que eso era así.

Finalmente, en cuanto a las muestras de código, he comprobado su funcionamiento y o mismo, así que considero que la calidad de los mismos es correcta, porque hacen lo que dicen hacer.

# Referencias


- Hacker Noon. (2020, febrero 2). Interview with Yukihiro Matsumoto: Ruby is Designed for Humans, not Machines. Recuperado 13 de mayo de 2020, de Hacker Noon Interview

- Geeks for Geeks. (2020, marzo 23). Ruby Tutorial. Recuperado 13 de mayo de 2020, de Ruby Tutorial

- Geeks for Geeks. (2019, noviembre 21). Polymorphism in Ruby. Recuperado 13 de mayo de 2020, de Polymorphism Ruby

- Geeks for Geeks. (2018, julio 30). Ruby | Types of Variables. Recuperado 13 de mayo de 2020, de Ruby Types of Variables

- Farsi, M. (2019, mayo 11). Ruby is a Multi-paradigm programming language - rubycademy. Recuperado 11 de mayo de 2020, de Ruby is a Multi-paradigm programming language

- EDUCBA. (s. f.). Uses Of Ruby. Recuperado 11 de mayo de 2020, de Uses of Ruby

- Wikipedia contributors. (2020, mayo 8). Ruby (programming language). Recuperado 11 de mayo de 2020, de Ruby (programming language)