

[Wikidot.com](http://Wikidot.com)

.wikidot.com

Share on      

[Edit](#) [History](#) [Tags](#) [Source](#)

[Explore »](#)

# Ruby Tutorial

## ...o como pasar un buen rato programando

- [admin](#)
  - [site manager](#)

[Create account](#) or [Sign in](#)



## Lección 1

- [Introducción](#)
- [Instalación](#)
- [El Primer Programa](#)
- [Números en Ruby](#)
- [Strings y diversión](#)
- [Variables](#)
- [Alcance de las variables](#)

## Lección 2

- [Introduciendo Datos](#)
- [Normas en los nombres](#)

- [Los métodos](#)
- [Los métodos: argumentos](#)
- [Rangos](#)
- [Arrays](#)

## Lección 3

- [Bloques](#)
- [Más malabares con strings](#)
- [Expresiones Regulares](#)
- [Condicionales](#)
- [Bucles](#)
- [Números Aleatorios](#)

## Lección 4

- [Clases y Objetos](#)
- [Accesores](#)
- [Ficheros: lectura/escritura](#)
- [Cargando librerías](#)
- [Herencia de clases](#)
- [Modificando clases](#)
- [Congelando objetos](#)
- [Serializando objetos](#)

## Lección 5

- [Control de acceso](#)
- [Excepciones](#)
- [Módulos](#)
- [Constantes](#)
- [Hashes y Símbolos](#)
- [La clase Time](#)

## Lección 6

- [self](#)
- [Duck Typing](#)
- [Azúcar Sintáctico](#)
- [Test de unidades](#)

## contacto

[e-mail](#)



## Variables

Para almacenar un número o un string en la memoria del ordenador, con el fin de usarlos en cálculos posteriores, necesitas dar un nombre a ese número o string. En programación este proceso es conocido como **asignación**.

#Ejemplos de asignaciones

```
s = 'Hello World!'
x = 10
```

Las variables locales en ruby son palabras que:

1. deben empezar con un letra minúscula o un guión bajo (\_)
2. deben estar formadas por letras, números y/o guiones bajos.

Cuando Ruby encuentra una palabra, la interpreta como: una variable local, un método o una palabra clave. Las palabras claves no pueden ser usados como variables. Por ejemplo **def** es una palabra clave: sólo se puede usar para definir un método. **if** también es una palabra clave: gran parte del código consta de instrucciones condicionales que empiezan con **if**, por eso sería muy confuso si pudiese usarse como variable.

Los métodos pueden ser palabras, como **start\_here**, **puts** o **print**. Cuando Ruby encuentra una palabra decide qué es de la siguiente forma:

1. si hay un signo de igualdad (=) a la derecha de la palabra, es una variable local a la que se le asigna un valor.
2. si la palabra es una palabra clave, entonces es una palabra clave. Ruby tiene una lista interna para poder reconocerlas.
3. Si no se cumple ninguno de los anteriores casos, Ruby asume que es un método.

```
# Definición de una constante
PI = 3.1416
puts PI
# Definición de una variable local
myString = 'Yo amo mi ciudad, Vigo'
puts myString

=begin
Conversiones
to_i - convierte a número entero
to_f - convierte a número decimal
to_s - convierte a string
=end

var1 = 5
var2 = '2' #fijarse que es un texto
puts var1 + var2.to_i

=begin
<< marca el comienzo de un string
    y es seguido de ' o ''. Aquí añadimos
    el string junto con el retorno de carro (\n).
=end

a = 'molo '
a<<'mucho.
Molo mazo...'
puts a
```

```
=begin
  ' o " son los delimitadores de un string.
  En este caso, podemos sustituirlos por END_STR.
  END_STR es una constante delimitador de strings.
=end
```

```
a = <<END_STR
This is the string
And a second line
END_STR
puts a
```

En el ejemplo:

```
var2.to_i
```

el punto significa que el método `to_i` es enviado a la variable **var2**, que este caso es un string: transformamos el string en un número para poder sumarlos. Cuando hablemos de objetos, veremos que se puede decir que la **var2** es el receptor de `to_i`. Por lo tanto, cuando aparezca un punto en una posición inexplicable, habrá que interpretarlo como un método (la parte derecha) que es enviado a un objeto (la parte izquierda).

## Interpretación Dinámica

Por interpretación dinámica, se entiende que no hace falta especificar qué tipo de variable se va a manejar: si parece un número, probablemente sea un número; si parece una cadena, probablemente lo sea. El método **class** devuelve el tipo de clase de un objeto:

```
s = 'hello'
s.class # String
```

Otro ejemplo:

```
# Ruby es dinámico
x = 7           #número entero
x = "house"     #string
x = 7.5         #número real
```

page\_revision: 10, last\_edited: 28 Jan 2009, 23:51 GMT-06 (670 days ago)

[EditTags](#) [History](#) [Files](#) [Print](#) [Site tools+](#) [Options](#)

[Help](#) | [Terms of Service](#) | [Privacy](#) | [Report a bug](#) | [Flag as objectionable](#)

Powered by [Wikidot.com](#)

Unless otherwise stated, the content of this page is licensed under [Creative Commons Attribution-ShareAlike 3.0 License](#)

## Other interesting sites



### [Sekrit Agent Sam](#)

given you a number, taken 'way your name



### [Gemexpe](#)



## Karma-Lab

KARMA Technology and the products incorporating it



## Scion: Monsters and Mosh Pits