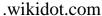
Wikidot.com





Ruby Tutorial

...o como pasar un buen rato programando

- admin
 - o 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

Usando librerías

Cuando escribes tus primeros programas, se tiende a guardar todo el código en un mismo fichero. Pero a medida que creces como programador, tus programas también lo hacen, y en algún momento te drás cuenta que tener un único fichero contiendo todo el código, es poco práctico. Es mucho más fácil dividir el código en grupos y colocar cada grupo en un fichero: una **librería** es un fichero que contiene métodos y clases para su uso a posteriori. Para poder usar estas librerías, necesitas de los métodos **require** y **load**.

require

El método **require** lee una única vez el fichero especificado:

hola.rb

```
puts "Hola a todos!"
```

Al usar varias veces require:

```
require 'hola' # Hola a todos!
require 'hola' #
require 'hola' #
```

Si volvemos a usarlo, omite su lectura: todas las clases y métodos del fichero, ya están alamacenados en memoria. Por tanto nos ahorramos el tiempo de otra lectura. Hay que fijarse en un detalle a la hora de usar require:

```
require 'hola' # bien
```

require da por hecho que la extensión del fichero es .rb, buscando el fichero 'hola.rb'. Aunque no fuese .rb, Ruby seguiría buscando entre las demás extensiones hasta encontrar un fichero del nombre 'hola'.

load

El método **load** lee el fihcero indicado tantas veces como aparezca la instrucción. ¿Difícil? No tanto, veamos un ejemplo:

```
load 'hola.rb' # Hola a todos!
load 'hola.rb' # Hola a todos!
load 'hola.rb' # Hola a todos!
```

Cada vez que se lee, todas los métodos del fichero se vuelven a leer, y están disponibles para su uso. Cuando se usa load es porque se espera que algo haya cambiado en el fichero.

RAILS: en Rails se suele usar load. Por ejemplo:mientras se desarrolla, lo que significa estar probando la aplicación y haciendo cambios al mismo tiempo, los cambios se actualizan al releer el fichero. Usar require no tendría el mismo efecto, puesto que el fichero se ha leído una vez.

Otro ejemplo

moto.rb

class Moto

```
def initialize(marca, color)
    # Atributos (variables del objeto)
    @marca = marca
    @color = color
end

def arrancar
    if (@estado_motor)
        puts 'Motor encendido'
    else
        @estado_motor = true
        puts 'Arrancando el motor'
    end
end
```

end

Si ahora queremos usar la clase Moto, en otro fichero:

```
require 'moto.rb'
m = Moto.new('Yamaha', 'rojo')
m.arrancar
```

page_revision: 5, last_edited: 26 Jul 2009, 18:02 GMT-05 (491 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 <u>Creative Commons Attribution-ShareAlike 3.0 License</u>

Other interesting sites



The Remnantz



Mikity



Divided Skies



•••••• ••••••