Ruby Tutorial

...o como pasar un buen rato programando

Lección 1

Lección 2

- 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

Strings y diversión

Los strings (o cadenas de texto) son secuencias de caracteres entre comillas simples o comillas dobles. " (dos comillas simples) no tienen nada: podemos llamarlo string vacío.

```
puts "Hola mundo"
# Se puede usar " o ' para los strings, pero ' es más eficiente.
puts 'Hola mundo'
# Juntando cadenas
puts 'Me gusta' + ' Ruby'
# Secuencia de escape
puts 'Ruby\'s party'
# Repetición de strings
puts 'Hola' * 3
# Definiendo una constante
# Más adelante se hablará de constantes
```

```
PI = 3.1416
puts PI
```

En Ruby las cadenas son mutables: se pueden modificar. Ruby almacena las cadenas como secuencias de bytes.

Strings con acento grave `

Hay unos strings especiales que se diferencian por usar como delimitador el acento grave `:

```
puts `dir`
```

El string entre los acentos, es enviado al sistema operativo como un comando a ser ejecutado. El resultado devuelto por el sistema, es recogido por Ruby. En este caso, al estar puts, obtendremos en pantalla el resultado de `dir`.

Interpolación

Con la interpolación nos referimos al proceso de insertar el resultado de una expresión dentro de un string. La forma de hacerlo es mediante **#{** expresión **}**. Ejemplo:

```
puts "100 * 5 = #{100 * 5}"
```

La sección `#{100*5}` se ejecuta y pone el resultado en esa posición. Es decir, 500.

Comillas simples (') vs comillas dobles (")

La diferencia entre ambas formas, es el tiempo que se toma Ruby en cada una: mientras que con las simples comillas, Ruby hace muy poco; en las dobles comillas, Ruby tiene que hacer más trabajo:

1. busca posibles **substituciones**: las secuencias de escape (las que empiecen por un \) son sustituidas por su valor binario.
2. busca posibles **interpolaciones**: en las secuencias con #{expresión}, se calcula la expresión, y se sustituye el bloque entero por su resultado.

```
def di_adios(nombre)
  resultado = "Buenas noches, #{nombre}"
  return resultado
end
```

```
puts di_adios('Pepe')

=begin
  Como los métodos devuelven el valor
  de la última línea, no hace falta
  el return.
=end

def di_adios2(nombre)
  resultado = 'Buenas noches, ' + nombre
end
puts di_adios2('Pepe')

=begin
  Ahora, en vez de usar ", usamos ',
  utilizando la concatenación de strings
  para obtener el mismo resultado.
=end
```

String#length

String#length devuelve el número de bytes de una cadena.

```
string = "Esto es una cadena"
string.length # => 18
```

string.length devuelve 18. Cuenta todas las letras, incluidos los espacios en blanco.

Other interesting sites

pre dev wiki

Let's open this beast up

d20universo

AIESEC-Madison

Seeing the World from New Perspectives