Ruby Tutorial

...o como pasar un buen rato programando

Lección 1

Lección 2

Lección 3

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

Números en Ruby

Juguemos con los números

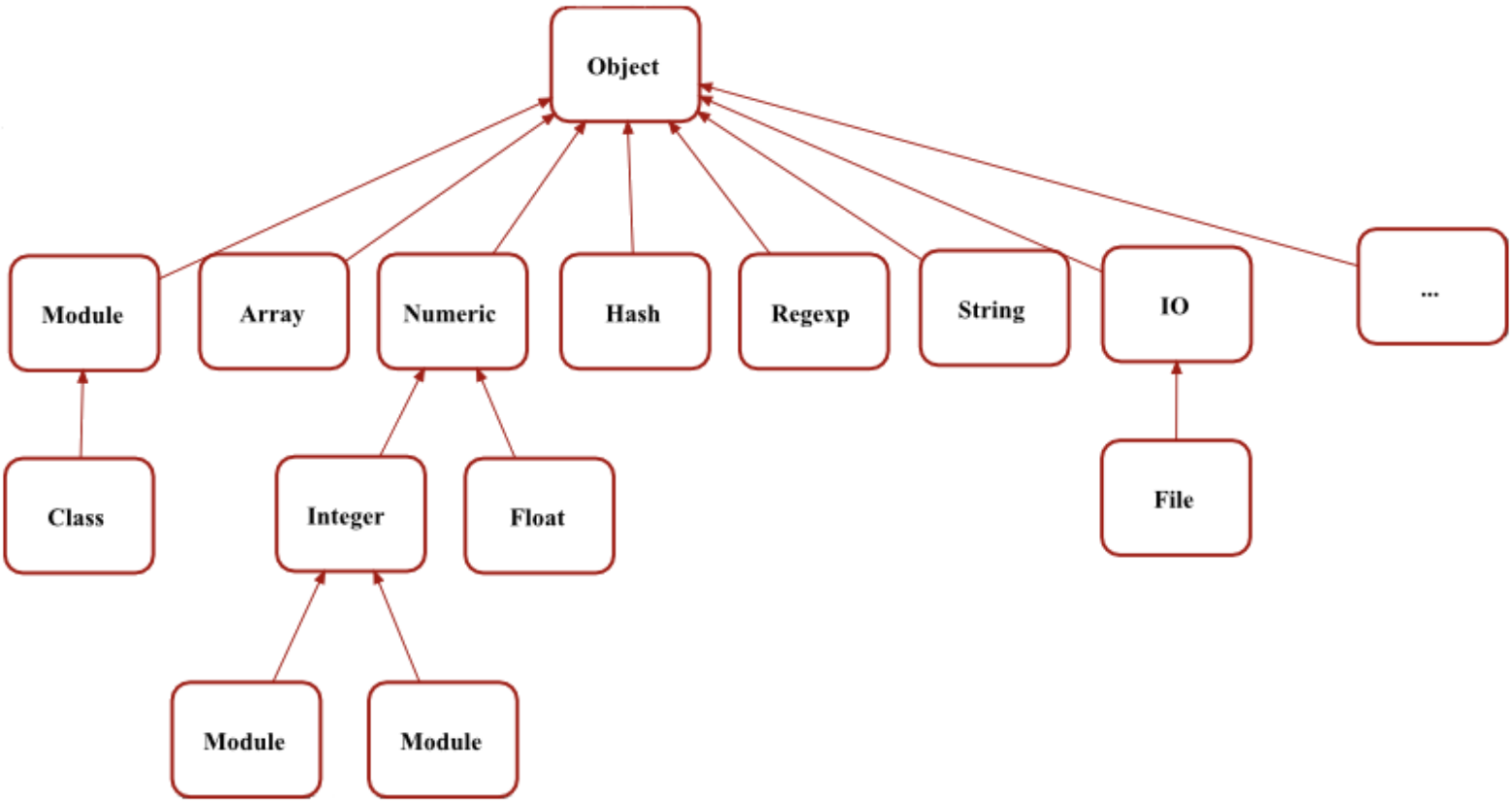
En Ruby, los números sin la coma decimal son llamados enteros, y los enteros con decimales son llamados coma-flotantes, o más sencillamente, flotantes.

```
puts 1 + 2
puts 10 - 11
puts 2 * 3
#Divisón: cuando divides dos enteros, obtienes un entero:
puts 3 / 2
#si quieres obtener el resultado de decimal,
#al menos uno de los dos tiene que ser decimal
puts 3.0 / 2
puts 3 / 2.0

puts 1.5 / 2.6
```

Los números en Ruby son objetos de la clase **`Fixnum`** o **`Bignum`**: estas clases representan números enteros de distintos tamaños. Ambas clases descienden de la clase **Integer** (en inglés, integer=entero). Los números coma-flotantes son objetos de la clase Float (en inglés, float=flotante).

Una diagrama de la **jerarquía de las clases** (Donald Craig):



Veamos el ejemplo que Peter Cooper nos propone, sacado de su libro "Beginning Ruby" (no importa que todavía no seas capaz de entender todo el código):

```
=begin
Problema del tablero de ajedrez:
si en la primera casilla ponemos un grano,
y duplicamos la cantidad de granos en la siguiente,
y así hasta rellenar el tablero,
¿cuántos granos tendremos?
=end
```

```
granos = 1
64.times do |escaque|
puts "En el escaque #{escaque+1} hay #{granos}"
granos *= 2
end
```

Al final tenemos $2.2.2...2.2 = 2^{64}$ granos en la última casilla…¡trillones de granos! Esto demuestra que Ruby es capaz de manejar números extremadamente grandes, y a diferencia de otros lenguajes de programación, no hay límites en esos números. Ruby hace esto gracias a las distintas clases antes mencionadas:

- **`Fixnum`** - maneja los números pequeños
- **`Bignum`** - maneja los números grandes (en inglés, big=grande).

Ruby escogerá cuál usar, y tú únicamente tendrás que preocuparte por lo que quieras hacer con ellos.

Operadores y precedencia

Echémosle un ojo a los operadores de Ruby (Half Fulton - The Ruby Way). Están ordenados de mayor a menor rango de precendencia; dicho de otra forma, los de la parte superior de la tabla, son los primeros en ejecutarse.

:	Alcance (scope)
[]	Índices
**	Exponentes
+ - ! ~	Unarios: pos/neg, no,…
* / %	Multiplicación, División,…
+ -	Suma, Resta,…
« »	Desplazadores binarios,…
&	'y' binario
|, ^	'or' y 'xor' binarios
> >= < <=	Comparadores
== === <=> != =~ !~	Igualadad, inegualdad,…
&&	'y' booleano
||	'o' booleano
.. …	Operadores de rango
= (+=, -=,...)	Asignadores
?:	Decisión ternaria
not	'no' booleano
and, or	'y', 'o' booleano

Destacar que:

1. Los paréntesis funcionan de la misma forma que en las matemáticas: cualquier cosa dentro de ellos es calculado en primer lugar. O dicho más técnicamente: tienen más precedencia.
2. Los operadores incremento y decremento (++ y —) no están disponibles en Ruby, ni en su forma "pre" ni en su forma "post".

El operador módulo

El operador módulo, que nos da el resto de una división, se comporta como sigue:

```
puts (5 % 3)     # imprime  2
puts (-5 % 3)    # imprime  1
```

```
puts (5 % -3)    # imprime -1
puts (-5 % -3)   # imprime -2
```

Ejercicio

Escribir un programa que diga cuantos minutos hay en un año de 365 días.

Other interesting sites

L4D Map Database

L4DMapDB.com

TENNESSEESPOKES

Guayadeque

A Complete Linux Music Manager