



Ingeniería de Software

1 - Ruby & Active Record

IIC2143-3 Josefa España

jpespana@uc.cl

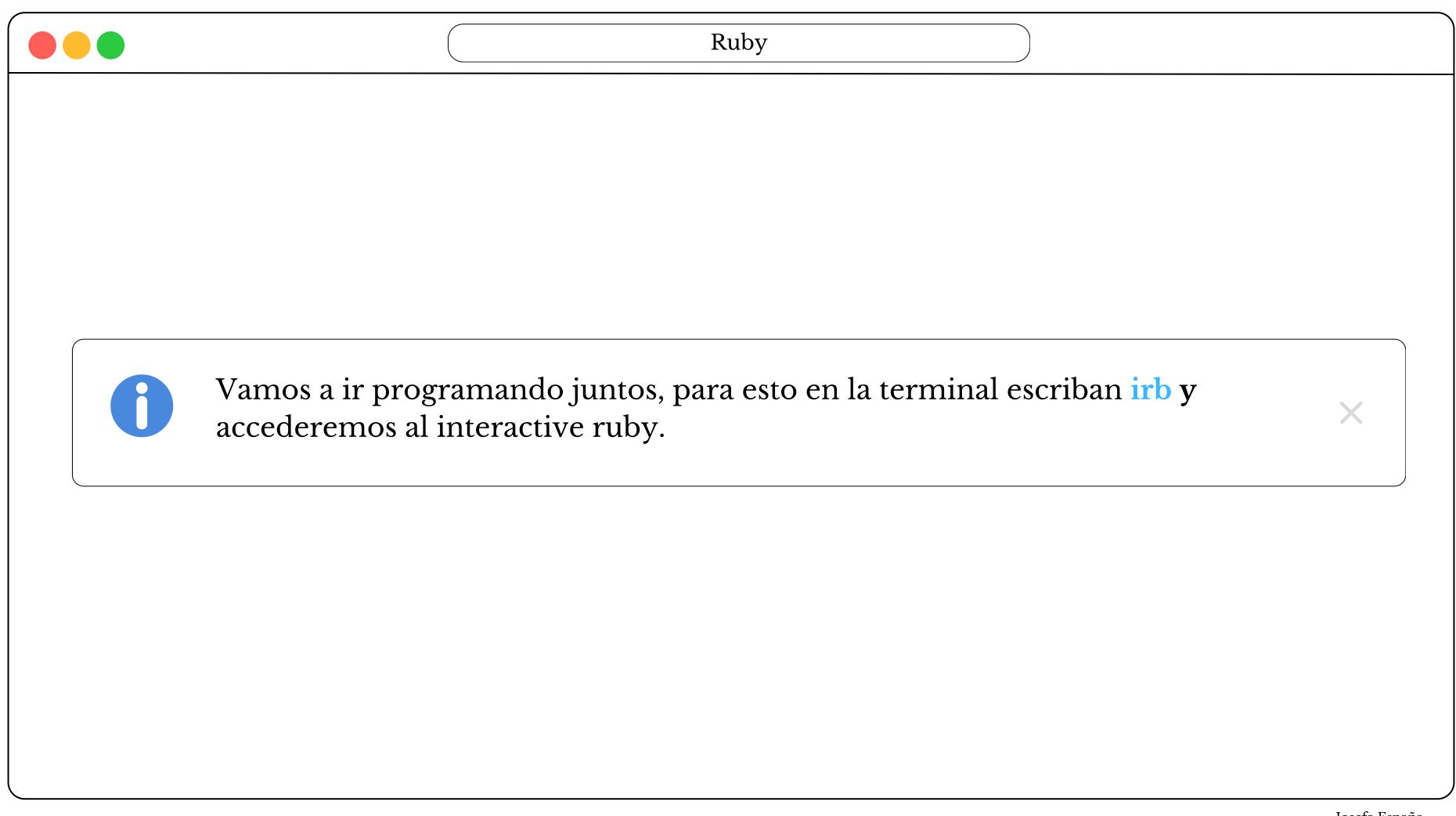


Contenidos

- Comentarios
- Aspectos básicos
- Strings
- Input
- Números
- Comparación
- Métodos
- Condicionales
- Arrays
- Hashes
- OOP
- Herencia
- Convenciones

```
1 # imprimir "Hello, World!" en la consola
2 puts "Hello, World!"
```

```
~/Projects/ruby (0.148s)
ruby ruby.rb
Hello World!
```





Comentarios

```
# Comentarios de una línea
puts 'Hello World!'
puts 'Hello Ruby!' # Comentarios

= begin
Comentario de múltiples líneas
puede tener código de ruby
y no será ejecutado dentro del comentario
= end
```



Aspectos básicos

```
address = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    # imprimir
    p address
    # index
    p address[5]
    new_address = address.reverse!
    p new_address
10
```

Strings

Strings

```
# Concatenar strings
first_name = 'Josefa'

last_name = 'España'

puts first_name + '' + last_name

# Interpolación de strings
# solo funciona con comillas dobles

puts "Mi primer nombre es #{first_name} y mi apellido es #{last_name}"
```

Strings



```
# Cómo encontrar métodos
10
11
    puts first_name.class
    puts 10.class
12
    puts 10.0.class
13
14
15
    puts first_name.methods
16
    # Métodos comunes
18
    first_name.empty?
    "".nil?
19
20
    sentence = 'Estamos aprendiendo Ruby'
21
22
    sentence.sub('Ruby', 'programación')
```

Strings

Strings

```
# Asignación de variables
24
    first_name = 'Josefa'
25
26
    new_first_name = first_name
    first_name = 'John'
27
28
    new_first_name # sigue siendo Josefa
29
30
    # Escapar caracteres
    # con \ podemos escapar símbolos
31
    puts "My first name is \#{first_name} and my last name is #{last_name}"
32
    puts 'Hey John, \'How are you doing?\''
33
```

Input

Input

```
puts "Qué te gustaría comer?"
food = gets.chomp
puts "Compraré #{food} entonces!"

puts "Ingresa un número y lo multiplicaré por 2"
input = gets.chomp
puts input.to_i * 2
```



Números

```
puts 1 + 2

puts 10 / 4 # retorna un integer, 2.

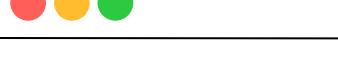
puts 10.0 / 4 # retorna un float

puts 10 / 4.to_f

x = 5

y = 10

puts y / x
```



Números

Números



Números

Josefa España



Comparación

```
# Comparación
    x = y
 3 1 = 2
 4 \ 3 = 3
 5 5 < 2
 6 \quad 2 \leqslant 5
   5 > 2
8 5 8 6
    5 || 6
10
    10.eql?(10.0) # false, compara el tipo de dato
11
12
    10 = 10.0 \# true
```





```
def multiply(first_num, second_num)
        # no necesitamos usar return si queremos devolver
        # el resultado de la última línea
        first_num.to_f * second_num.to_f
    end
    def divide first_num, second_num
        first_num.to_f / second_num.to_f
    end
10
    def subtract(first_num, second_num)
        first_num.to_f - first_num.to_f
12
13
    end
14
    def sum(first_num, second_num)
        first_num.to_f + first_num.to_f
16
17
    end
18
    def remaining(first_num, second_num)
        first_num.to_f % second_num.to_f
20
21
    end
```

Josefa España

Métodos



```
puts "Calculadora simple"
23
    25.times{ print "-" }
25
    puts
    puts "Ingresa el primer número"
26
    num_1 = gets.chomp
27
    puts "Ingresa el segundo número"
28
    num_2 = gets.chomp
29
    puts "El primer número multiplicado por el segundo número es #{multiply(num_1, num_2)}"
    puts "El primer número dividido por el segundo número es #{divide(num_1, num_2)}"
31
    puts "El primer número restado por el segundo número es #{subtract(num_1, num_2)}"
    puts "El primer número más el segundo número es #{sum(num_1, num_2)}"
33
    puts "El resto de la división del primer número por el segundo número es #{remaining(num_1, num_2)}"
34
35
```



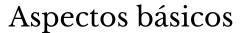
Métodos

```
1  def call_block
2    puts "Start of method"
3    yield
4    yield
5    puts "End of method"
6  end
7
8  call_block { puts "In the block" }
```



Condicionales

```
condition = true
    another_condition = false
   if condition & another_condition # y
        puts "esto se evaluó como verdadero"
   else
        puts "esto se evaluó como falso"
    end
9
    if condition || another_condition # o
        puts "Hola"
    else
12
        puts "Adiós"
13
14
    end
```





```
puts "Calculadora simple"
    20.times{ print "-" }
    puts
    puts "Por favor ingresa tu primer número"
    first_number = gets.chomp
    puts "Por favor ingresa tu segundo número"
    second_number = gets.chomp
    puts "¿Qué quieres hacer?"
    puts "Ingresa 1 para multiplicar, 2 para sumar, 3 para restar"
    user_entry = gets.chomp
    puts "Has seleccionado #{user_entry}"
    if user_entry = "1"
        puts "Has elegido multiplicar"
51
        puts "El resultado es #{multiply(first_number, second_number)}"
52
    elsif user_entry = "2"
        puts "Has elegido sumar"
54
        puts "El resultado es #{add(first_number, second_number)}"
    elsif user_entry = "3"
57
        puts "Has elegido restar"
        puts "El resultado es #{subtract(first_number, second_number)}"
58
    else
59
        puts "Entrada inválida"
60
    end
61
```



Arrays

```
1  a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
2  # indices: 0, 1, 2, 3, 4, 5, 6, 7, 8
3
4  # puts a  # imprimir con salto de línea
5  # print a  # imprimir sin salto de línea y como arreglo
6  # p a  # imprimir con salto de línea e indices
7
8  a.last # 9
9  a.first # 1
```



Arrays

```
# crear un rango de números entre 1 y 100

x = 1..100 # objeto rango

x = (1..100).to_a # objeto arreglo

x.to_a.shuffle # mezclar los números

z = x.to_a.shuffle! # mezclar los números y guardarlos en z como arreglo

x.reverse! # invertir los números y guardarlos en x como arreglo

# !: operador bang, modificará el objeto original
```

Aspectos básicos

Arrays

```
a = ("a".."z").to_a # crear un arreglo de letras de a a z
    # agregar elemento al arreglo
23
    a << 10
    a.unshift("Josefa") # agregar elemento al principio del arreglo
    a.append("Josefa") # agregar elemento al final del arreglo
    a.uniq # eliminar duplicados
26
    a.uniq! # eliminar duplicados y guardarlos en a como arreglo
27
28
    a.empty? # verificar si el arreglo está vacío
    a.include?("Josefa") # verificar si el arreglo contiene a Josefa
30
    a.push("new item") # agregar elemento al final del arreglo
31
    a.pop # eliminar el último elemento del arreglo
32
    a.join # convertir arreglo a cadena de texto
33
    a.join("-") # convertir arreglo a cadena de texto con - entre elementos
    a.split("-") # convertir cadena de texto a arreglo con - entre elementos
```







Hashes

```
1 sample_hash = {'a' \Rightarrow 1, 'b' \Rightarrow 2, 'c' \Rightarrow 3}
    my_details = {'name' ⇒ 'mashrur', 'favcolor' ⇒ 'red'}
    p my_details['favcolor']
    p sample_hash['b']
    another_hash = {a: 1, b: 2, c: 3} # asignar pares clave valor usando símbolos
    p another_hash[:a]
    sample_hash.keys
   sample_hash.values
11 sample_hash.each do |key, value|
        puts "The class for key is #{key.class} and the value is #{value.class}"
12
    end
13
14
    sample_hash[:e] = "Mashrur"
    sample_hash[:c] = "Ruby"
17
    sample_hash.each { |some_key, some_value| puts "The key is #{some_key} and the value is #{some_value}" }
19 sample_hash.select { |k, v| v.is_a?(String) }
    sample_hash.each { |k, v| sample_hash.delete(k) if v.is_a?(String) }
21
```

Aspectos básicos



```
class Student
        attr_accessor :first_name, :last_name, :email, :username
        def initialize(first_name, last_name, username, email, password)
            @first_name = first_name
            @last_name = last_name
            @username = username
            Opassword = password
            aemail = email
10
        end
11
12
        def to_s
            "First name: #{@first_name}, Last name: #{@last_name}, Username: #{@username}, Email: #{email}"
13
14
        end
15
    end
16
    felipe = Student.new("Felipe", "Muñoz", "felipemunoz1", "felipe@email.com", "password")
    john = Student.new("John", "Doe", "john1", "john@email.com", "password1")
    puts felipe
    felipe.last_name = john.last_name
    puts "felipe is altered"
22 puts felipe
```



```
class Song
      def initialize(name, artist, duration)
        @name = name
        @artist = artist
        aduration = duration
      end
      def duration=(duration)
        aduration = duration
10
       end
11
12
      def duration
13
        @duration
14
      end
15
16
      def to_s
        "Song: #@name}--#@artist} (#@duration})"
17
18
      end
    end
19
20
    song = Song.new('Anti-Hero', 'Taylor Swift', 3.2)
    song.duration= 10
    song.duration \# \Rightarrow 10
24 song.duration = 20
```

```
class KaraokeSong < Song</pre>
      def initialize(name, artist, duration, lyrics)
        super(name, artist, duration)
28
        alyrics = lyrics
30
      end
31
      def to_s
32
        super + " [#{@lyrics}]"
33
34
      end
35
    end
36
    karaoke_song = KaraokeSong.new('Anti-Hero',
                                     'Taylor Swift',
38
                                     3.2,
39
                                     "I'm the problem")
40
    karaoke_song.to_s
```



Convenciones

- 1. snake_case para métodos y variables.
- 2. PascalCase para nombres de clases.
- 3. Si el método retorna booleanos, debe ir con un signo de interrogación al final.
- 4. Identación de 2 espacios.

```
4 def is_even?(number)
5 number.even?
6 end
```





Ingeniería de Software

1 - Ruby

IIC2143-3 Josefa España

jpespana@uc.cl