



# Empezando con Ruby

Juan Sebastian Velez Posada



/jvelezpo



/rderoldan1

# Globant

- Horario: Lunes a viernes 8 a 18 Flexible
- Beneficio de Work From Home 1 - 2 veces por semana (hay algunas excepciones dependiendo del cliente o asignación, pero no es lo común)
- Plataforma de Training y Capacitaciones, a la cual nuestros Globers tienen acceso todo el tiempo; hay oferta de cursos en todas las áreas, Desarrollo, testing, autodesarrollo, planes de mentores, etc. Tu podrías tomar la cantidad que quieras y cuando quieras, sin límites o restricciones.
- Clases de Inglés sin costo para nuestros empleados.
- Fruta para el desayuno 2 veces por semana.

<https://www.globant.com/>

Análisis numérico - Sistema RMS

# Análisis Numérico

implementación de métodos numéricos

Menú de módulos

Ecuaciones no lineales

Sistemas de ecuaciones

Métodos Directos   Métodos Iterativos

Eliminación   Jacobi

Pivoteo parcial   Gauss Seidel

Pivoteo total   Relajación

Factorización LU

LU GAUSS   LU PARCIAL

Crout   Cholesky

Dolittle   ☐ Obtener Ayuda


Interpolación


Integración

Graficador

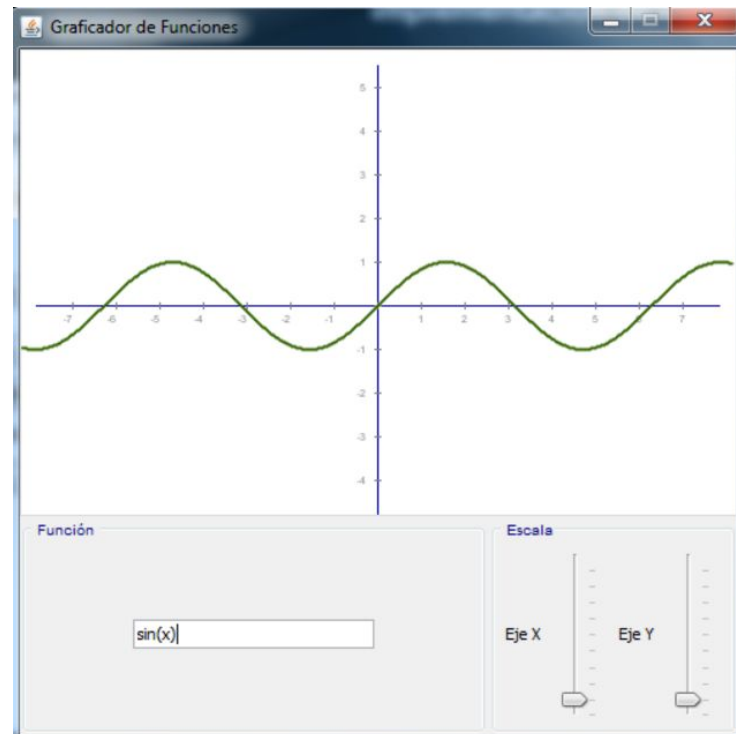
Variables   

x1	x2	x3	x4	B

 **UNIVERSIDAD EAFIT**  
Abierta al mundo

 1960-2010

José Mateo Vidal Barrera  
Rubén Darío Espinosa Roldán  
Sebastián Velez Posada



# Temario

Esta presentación mostrará los siguientes temas:

1. Clases
2. Objetos
3. Atributos
4. Control de acceso
5. Herencia
6. Módulos
7. Ryby meets the web (Sinatra)

**\*\*TIP**

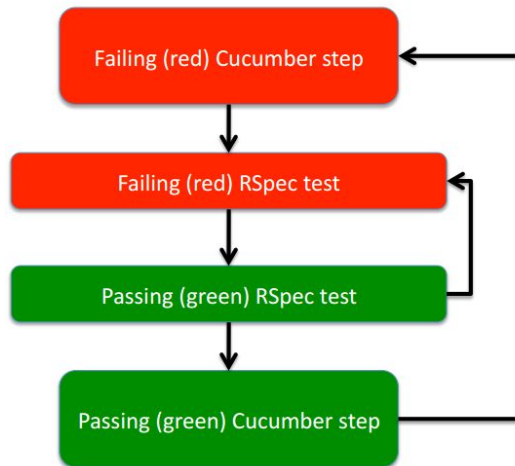
Se recomienda leer y aplicar las buenas practicas en la forma de programación especificadas en

# TDD

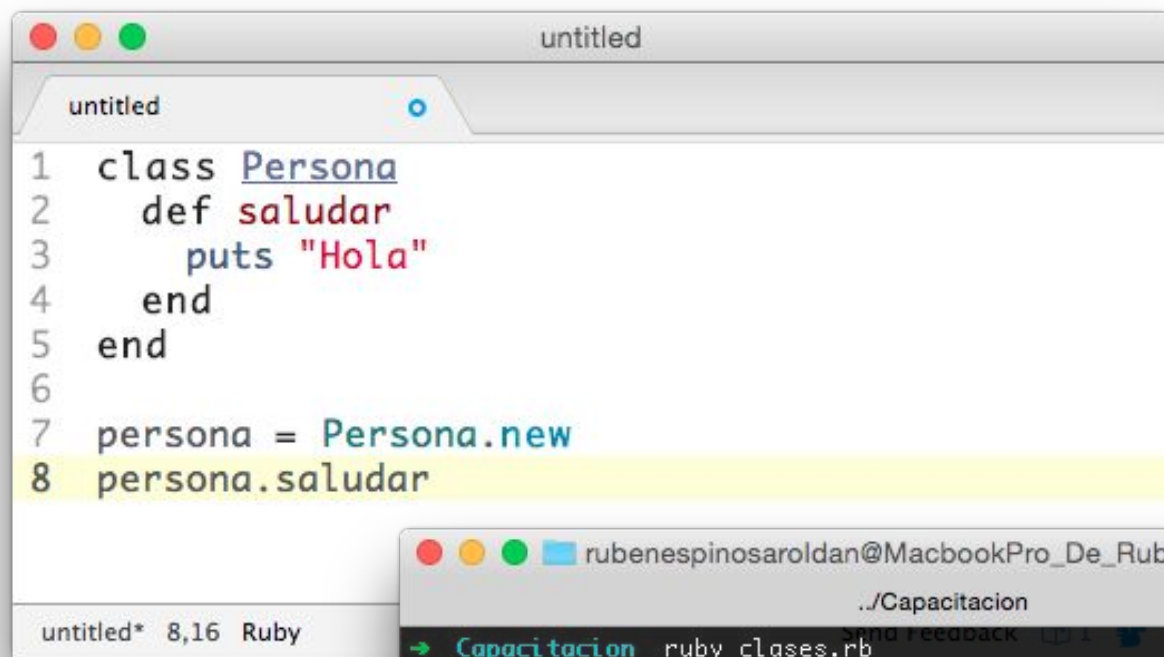
- Test-driven development (TDD)
  - step definitions for new story, may require new code to be written
  - TDD says: write unit & functional tests for that code first, **before** the code itself
  - that is: write tests for the code you wish you had

# Rspec

- RSpec tests individual modules that contribute to those behaviors (test driven development)

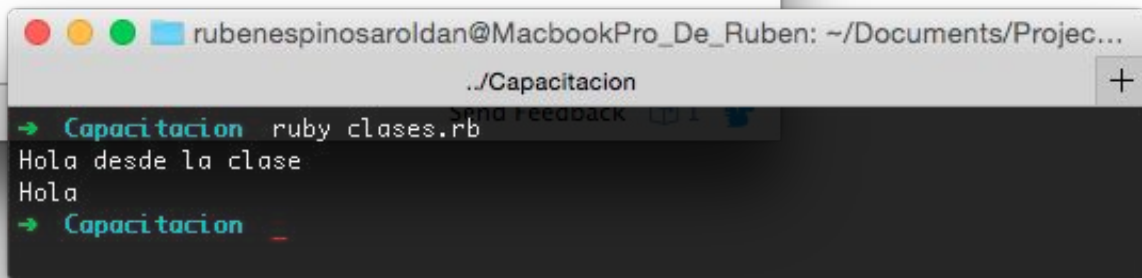


# Classes



```
1 class Persona
2   def saludar
3     puts "Hola"
4   end
5 end
6
7 persona = Persona.new
8 persona.saludar
```

untitled\* 8,16 Ruby



```
rubenespinosaroldan@MacbookPro_De_Ruben: ~/Documents/Proyec...
./Capacitacion
→ Capacitacion ruby clases.rb
Hola desde la clase
Hola
→ Capacitacion _
```

# Objeto != Clase

```
clases.rb x
1 class Persona
2   # metodo de instancia
3   def saludar
4     puts "Hola"
5   end
6
7   # metodo de clase
8   def self.saludar
9     puts 'Hola desde la clase'
10  end
11 end
12
13
14 persona = Persona.new
15 p persona
16
```

```
rubenespinosaroldan@MacbookPro_De_Ruben: ~
./Capacitacion
→ Capacitacion ruby clases.rb
#<Persona:0x00000102885ba0>
→ Capacitacion
→ Capacitacion
→ Capacitacion _
```



# Atributos

```
clases.rb x
1 class Persona
2
3   def nombre
4     @nombre
5   end
6
7   def nombre=(nuevo_nombre)
8     @nombre=nuevo_nombre
9   end
10
11   # metodo de instancia
12   def saludar
13     puts "Hola #{self.nombre}"
14   end
15
16   # metodo de clase
17   def self.saludar
18     puts 'Hola desde la clase'
19   end
20 end
21
22
23 persona = Persona.new
24 persona.nombre = "Ruben"
25 p persona.nombre
26 persona.saludar|
27
```

# Atributos

```
clases.rb
1 class Persona
2   attr_reader :nombre
3
4   def nombre=(nuevo_nombre)
5     @nombre=nuevo_nombre
6   end
7
8   # metodo de instancia
9   def saludar
10    puts "Hola #{self.nombre}"
11  end
12
13  # metodo de clase
14  def self.saludar
15    puts 'Hola desde la clase'
16  end
17 end
18
19
20 persona = Persona.new
21 persona.nombre = "Ruben"
22 p persona.nombre
23 persona.saludar
24
```

# Atributos

```
clases.rb x
1 class Persona
2   attr_reader :nombre
3   attr_writer :nombre
4
5   # metodo de instancia
6   def saludar
7     puts "Hola #{self.nombre}"
8   end
9
10  # metodo de clase
11  def self.saludar
12    puts 'Hola desde la clase'
13  end
14 end
15
16
17 persona = Persona.new
18 persona.nombre = "Ruben"
19 p persona.nombre
20 persona.saludar
21
```

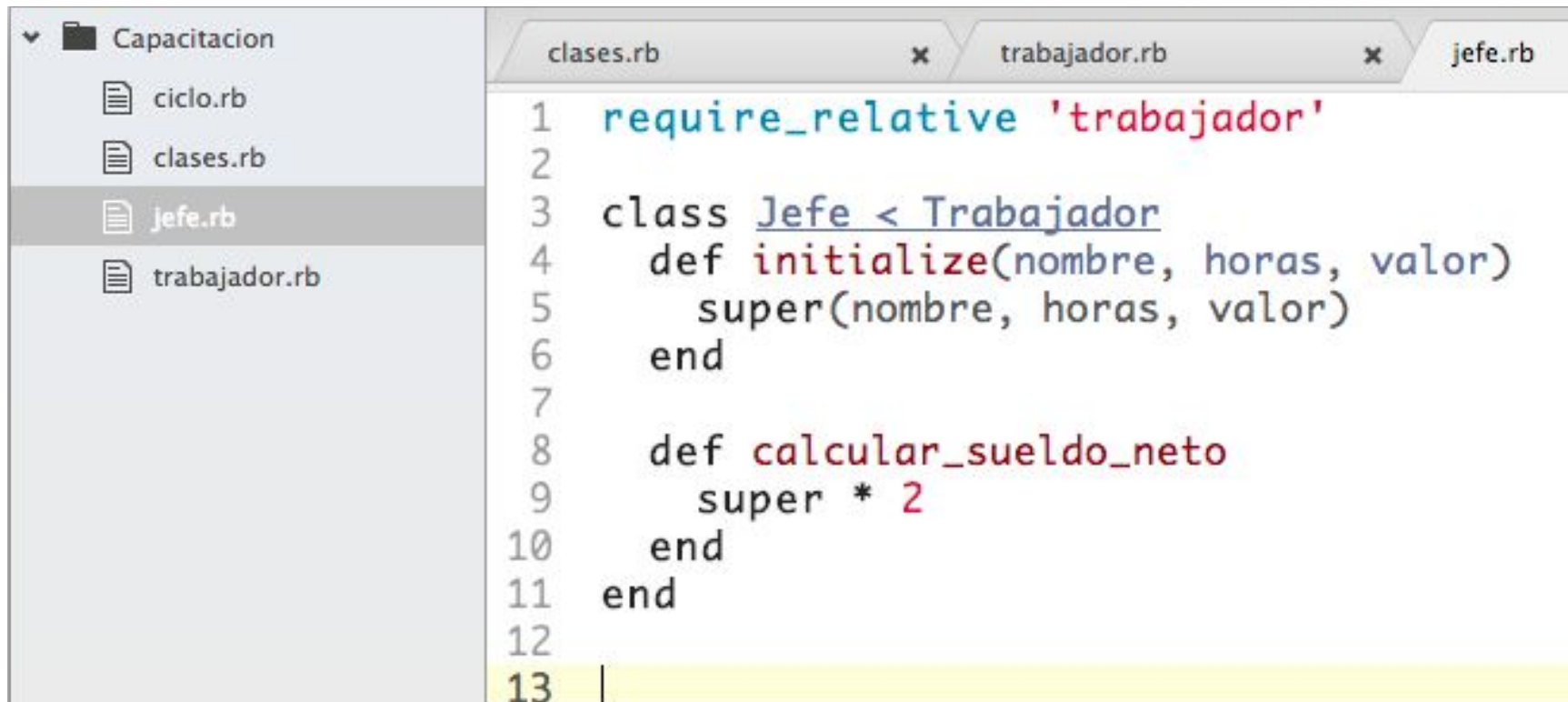
# Atributos

```
clases.rb x
1 class Persona
2   attr_accessor :nombre
3
4   # metodo de instancia
5   def saludar
6     puts "Hola #{self.nombre}"
7   end
8
9   # metodo de clase
10  def self.saludar
11    puts 'Hola desde la clase'
12  end
13 end
14
15
16 persona = Persona.new
17 persona.nombre = "Ruben"
18 p persona.nombre
19 persona.saludar
20
```

# Control de acceso (encapsulamiento)

```
clases.rb x nomina.rb
1 class Trabajador
2   attr_accessor :nombre, :horas, :valor
3   RTE_FTE = 0.05
4
5   def initialize(nombre, horas, valor)
6     self.nombre = nombre
7     self.horas = horas
8     self.valor = valor
9   end
10
11   private
12   def calcular_pago
13     self.horas * valor
14   end
15
16   protected
17   def calcular_rtefte
18     (self.horas * valor) * RTE_FTE
19   end
20
21   public
22   def calcular_sueldo_netto
23     calcular_pago - calcular_rtefte
24   end
25 end
26
```

# Herencia



```
1  require_relative 'trabajador'
2
3  class Jefe < Trabajador
4    def initialize(nombre, horas, valor)
5      super(nombre, horas, valor)
6    end
7
8    def calcular_sueldo_neto
9      super * 2
10   end
11 end
12
13 |
```

# Classes & inheritance

```
class SavingsAccount < Account # inheritance
  # constructor used when SavingsAccount.new(...) called
  def initialize(starting_balance=0)      # optional argument
    @balance = starting_balance
  end
  def balance                             # instance method
    @balance                             # instance var: visible only to this object
  end
  def balance=(new_amount)                # note method name: like setter
    @balance = new_amount
  end
  def deposit(amount)
    @balance += amount
  end
  @@bank_name = "MyBank.com"             # class (static) variable
  # A class method
  def self.bank_name # note difference in method def
    @@bank_name
  end
  # or: def SavingsAccount.bank_name ; @@bank_name ; end
end
```

# Question

(a) `my_account.@balance`

(b) `my_account.balance`

(c) `my_account.balance()`

Which ones are correct:

All three

Only (b)

(a) and (b)

(b) and (c)



# Instance variables: shortcut

```
class SavingsAccount < Account
  def initialize(starting_balance)
    @balance = starting_balance
  end
  def balance
    @balance
  end
  def balance=(new_amount)
    @balance = new_amount
  end
end
```

# Instance variables: shortcut

```
class SavingsAccount < Account
  def initialize(starting_balance)
    @balance = starting_balance
  end

  attr_accessor :balance
end
```

**attr\_accessor is just a plain old method that uses metaprogramming...not part of the language!**

# Question

```
class String
  def curvy?
    !("AEFHIKLMNTVWXYZ".include?(self.upcase))
  end
end
```

String.curvy?("foo")

"foo".curvy?

self.curvy?("foo")

curvy?("foo")

# Review: Ruby's Distinguishing Features (So Far)

- Object-oriented with no multiple-inheritance
  - everything is an object, even simple things like integers
  - class, instance variables invisible outside class
- Everything is a method call
  - usually, only care if receiver responds to method
  - most “operators” (like +, ==) actually instance methods
  - Dynamically typed: objects have types; variables don't
- Destructive methods
  - Most methods are nondestructive, returning a new copy
  - Exceptions: <<, some destructive methods (eg merge vs. merge! for hash)
- Idiomatically, {} and () sometimes optional

# Loops—but don't think of them that way

```
["apple", "banana", "cherry"].each do |string|  
  puts string  
end                                     # apple banana cherry
```

```
for i in (1..10) do  
  puts i  
end                                   # 1 2 3 4 5 6 7 8 9 10
```

```
1.upto 10 do |num|  
  puts num  
end                                   # 1 2 3 4 5 6 7 8 9 10
```

```
3.times { print "Rah, " } # Rah, Rah, Rah,
```

# If you're iterating with an index, you're probably doing it wrong

- Iterators let objects manage their own traversal

- `(1..10).each do |x| ... end`  
`(1..10).each { |x| ... }`  
`1.upto(10) do |x| ... end`

=> range traversal

- `my_array.each do |n| ... end`

=> array traversal

- `hsh.each_key do |key| ... end`  
`hsh.each_pair do |key,val| ... end`

=> hash traversal

- `10.times {...} # => iterator of arity zero`  
• `10.times do ... end`

# “Expression orientation”

```
x = ['apple', 'cherry', 'apple', 'banana']
```

```
x.sort          # => ['apple', 'apple', 'banana', 'cherry']  
x.uniq.reverse  # => ['banana', 'cherry', 'apple']  
x.reverse!      # => modifies x
```

```
x.map do |fruit|  
  fruit.reverse  
end.sort        # => ['ananab', 'elppa', 'elppa', 'yrrehc']
```

```
x.collect { |f| f.include?("e") }  
          # [true, true, true, false]
```

```
x.any? { |f| f.length > 5 }      # true
```

# “Expression orientation”

Which string will **NOT** appear in the result of:

```
['banana', 'anana', 'naan'].map do |food|  
  food.reverse  
end.select { |f| f.match /^a/ }
```

naan

ananab

anana

The above code won't run due to syntax error(s)



# Sinatra



<http://www.sinatrarb.com/>

<https://github.com/sinatra/sinatra>

```
1. rubenepinosaroldan@MacbookPro_De_Ruben: ~ (zsh)  
→ - gem install sinatra_
```

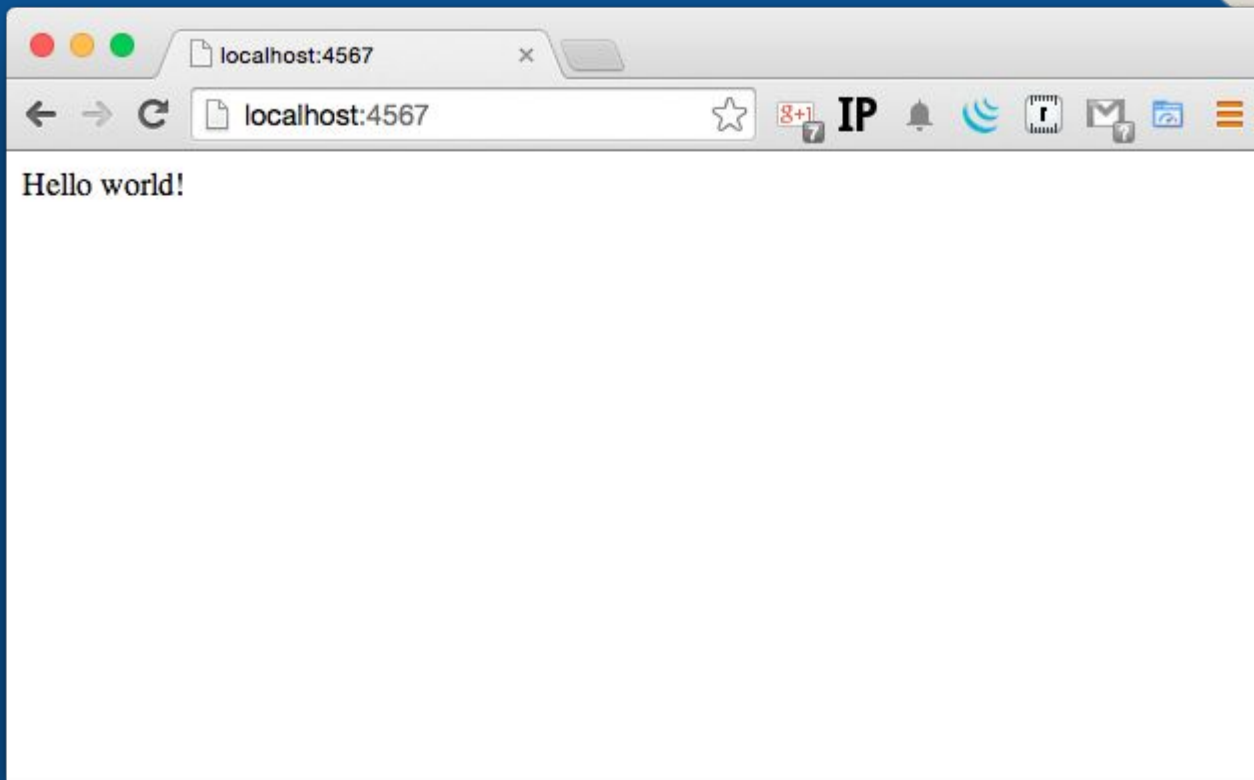
# Sinatra



```
hello_sinatra.rb - /Users/rubenespinosaroldan/Docum...
hello_sinatra.rb
1  # myapp.rb
2  require 'sinatra'
3
4  get '/' do
5    'Hello world!'
6  end
7
hello_sinatra.rb* 5,18 Ruby Send Feedback 1
```

ruby hello\_sinatra.rb

# Sinatra



# Rails - Homework

<https://github.com/rails/rails>

Read the doc and do the Getting Started

