Ingeniería de Software

Ruby

Juan Pablo Sandoval

Hello World

```
puts "Hello World!”
```

```
>ruby -e 'puts "Hello Ruby!\n"'
Hello Ruby!
```

```
>ruby hello.rb
Hello Ruby!
```

RoadMap

- *Comentarios*
- *Aspectos Básicos*
- *Variables*
- *Funciones*
- *Clases y Objetos*
- *Herencia*
- *Arreglos y Hashes*
- *Estructuras de Control*
- *Ejercicios*

Comentarios

```
# Comentario de una linea
puts "Hello World!"
```

```
puts "Welcome to Ruby!"      # comentario
```

```
=begin
Comentario de multiple lineas
Se puede poner código en ruby dentro del comentario
El código dentro de los comentarios no es interpretado
=end
```

Aspectos Básicos

```
123456
-543
```

```
puts "texto"        # imprime"texto"
puts 'texto'        # imprime"texto"
```

```
puts "texto\n"        # imprime"texto" y enter
puts "suma :#{1+2}" # imprime "suma: 3"
puts 'suma :#{1+2}' # imprime "suma :#{1+2}"
```

Variables

```
MYCONSTANT = "hello"
```

```
var1 = Person.new
var2 = 230
var3 = "hola"
var2 = Array.new
```

Funciones

```
def sum (n1, n2)
      n1 + n2
end

sum( 3 , 4)                    # devuelve 7
sum("cat", "dog")              # devuelve "catdog"
```

Funciones

```
def multiply(val1, val2 )
      result = val1 * val2
      return result
end

value = multiply( 10, 20 )
puts value
# imprime 200
```

Funciones

```ruby
def say_goodnight(name)
    "Good night, #{name}"
end
puts say_goodnight('Ma')   # imprime Good night, Ma
puts say_goodnight'Ma'     # imprime Good night, Ma
```

Funciones anónimas (yield)

```ruby
def call_block                      #función llamada call_block
     yield("hello",2)               # ejecutando la función anonima
end
                                    # enviando una función anonima
call_block { | s, n | puts s*n, "\n" }

# imprime hellohello
```

RoadMap

- *Hello World*
- *Comentarios*
- *Aspectos Básicos*
- *Variables*
- *Funciones*
- ***Clases y Objetos***
- *Herencia*
- *Estructuras de Control*
- *Convenciones*
- *Ejercicios*

Clases y Objetos

```
class BankAccount
    def initialize()
    end

    def test_method
         puts "The class is working"
    end
end

account = BankAccount.new()
account.test_method  # imprime The class is working
```

Clases y Objetos

```ruby
class BankAccount
   def initialize (number)
      @accountNumber = number
   end

   def deposit(amount)
        @accountNumber = @accountNumber + amount
   end
   def withdraw(amount)
        @accountNumber = @accountNumber - amount
   end
   def print
       puts "balance: #{@accountNumber}"
   end
end

account = BankAccount.new(1324)
account.deposit(200)
account.withdraw(100)
```

Clases y Objetos

```
class BankAccount
    ...
    def accountNumber= (number)
        @accountNumber = number
    end
    def accountNumber(number)
        return @accountNumber
    end
    ...
end
account= BankAccount.new(1223)
account.accountNumber= 3
account.accountNumber  # devuelve 3
account.accountNumber =  3 # sigue funcionando con espacios
```

Herencia

```
class Song
  def initialize(name, artist, duration)
      @name = name
      @artist = artist
      @duration = duration
  end
end

song = Song.new("Bicylops", "Fleck", 260)
song.to_s           # que devuelve #<Song:0xe6c>
```

Herencia

```
class Song
  def initialize(name, artist, duration)
      @name = name
      @artist = artist
      @duration = duration
  end
  def to_s                  # sobre-escritura
       "Song: #{@name}--#{@artist} (#{@duration})"
  end
end

song = Song.new("Bicylops", "Fleck", 260)
song.to_s
# devuelve "Song: Bicylops--Fleck (260 )”
```

Herencia

```
class KaraokeSong < Song
  def initialize(name, artist, duration, lyrics)
    super(name, artist, duration)
    @lyrics = lyrics
  end
end
song = KaraokeSong.new("My Way", "Sinatra", 225, "And now, the...")
song.to_s
#devuelve "Song: My Way--Sinatra (225)”
```

Herencia

```ruby
class KaraokeSong < Song
  def initialize(name, artist, duration, lyrics)
    super(name, artist, duration)
    @lyrics = lyrics
  end
  def to_s
    super + " [#{@lyrics}]"
  end
end
song = KaraokeSong.new("My Way", "Sinatra", 225, "And now, the...")
song.to_s
#devuelve "Song: My Way--Sinatra (225) [And now, the …]”
```

Variables y Métodos de Clase

```ruby
class Song
  @@total_plays = 0
  def initialize(name, artist, duration)
    @name = name
    @artist = artist
    @duration = duration
    @plays = 0
  end
  def play
    @plays += 1
    @@total_plays += 1
  end
  def printReport
    puts "this song play #{@plays} times"
  end
  def self.printPlaysReport
    puts "all songs play #{@@total_plays} times"
  end
end
```

```ruby
song1 = Song.new
song1.play
song2 = Song.new
song2.play

song1.printReport
# this song play 1 times
song2.printReport
# this song play 1 times
Song.printPlaysReport
# all songs play 2 times
```

Arreglos

```
days_of_week = Array[ "Mon", "Tues", "Wed", "Thu", "Fri", "Sat", "Sun" ]
days_of_week.at(0)        # devuelve "Mon"
days_of_week.size         # devuelve 7
days_of_week.empty?       # devuelve false
```

Arreglos

```
days_of_week = [ "Mon", "Tue", "Wed", "Thu", "Fri" ]
days_of_week[0]          # devuelve "Mon"
days_of_week[1]          # devuelve "Tue"
```

Arreglos

```
days1 = ["Mon", "Tue", "Wed"]
days2 = ["Thu", "Fri", "Sat", "Sun"]
days = days1 + days2

# ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
```

Arreglos

```
colors = ["red", "green", "blue"]

# antes ["red", "green", "blue"]

colors[1] = "yellow"        # asigna "yellow"
colors

# después ["red", "yellow", "blue"]
```

Hash

```
inst = { "a" => 1, "b" => 2 }
inst["a"]  # devuelve  1
inst["c"]  # devuelve  2

# al especificar 0 en el constructor, el Hash inicializa los
valores por defecto en 0

inst = Hash.new(0)
inst["a"] # devuelve  0
inst["a"] += 1
inst["a"] # devuelve  1
```

Estructuras de Control

```
if count > 10
      puts "Try again"
elsif tries == 3
      puts "You lose"
else
      puts "Enter a number"
end
```

Estructuras de Control

```
while weight < 100 and num_pallets <= 30
    pallet = next_pallet()
    weight += pallet.weight
    num_pallets += 1
end
```

Iteradores

```
animals = ["ant", "bee", "cat", "dog", “elk"]

# forma 1
animals.each {|animal| puts animal }

# forma 2
for animal in animals do
    puts animal
end
```

Otros iteradores

```
3.times { print "X " }
1.upto(5) {|i| print i, " " }
99.downto(95) {|i| print i, " " }
50.step(80, 5) {|i| print i, " " }
```

Material Adicional

- *Documentación de Ruby: https://www.ruby-lang.org/es/documentation/*
- *Prueba Ruby en tu navegador: https://try.ruby-lang.org/*
- *Aprende a programar: https://pine.fm/LearnToProgram/*
- *Otros libros: https://github.com/EbookFoundation/free-programming-books/blob/main/books/free-programming-books-langs.md#ruby*

Para la siguiente clase

- *Instalar Ruby on rails (https://guides.rubyonrails.org/getting_started.html).*
- *Instalar Postman (https://www.postman.com/downloads/)*
- *¡Traer sus computadoras para programar!*