



EXPLICACIÓN DE LA TAREA HolaRuby

SILVA ROJAS JUAN SEBASTIAN

PARTE 1

```
def sum arr
  result = 0
  arr.each do |e|
    result = result + e.to_i
  end

  result.to_i
end
```

El arreglo se pasa como argumento y con un bucle **each** recorreremos los elementos para sumarlos y guardar dichas sumas en **result**, este mismo contendrá la suma de todos los elementos y es lo que retorna la función.

Resultado del testeo:

```
PS D:\CURSOS\DESARROLLO SOFTWARE\TAREAS\HolaRuby> rspec -e '#sum' .\spec\parte1_spec.rb
Run options: include {:full_description=>/\#sum/}
.....

Finished in 0.0193 seconds (files took 0.38105 seconds to load)
7 examples, 0 failures
```

```
def max_2_sum arr
  addends = arr.max(2)

  sum(addends)
end
```

Usamos la función **max(2)** para capturar los dos elementos mas grandes del arreglo **arr**, y dichos valores lo almacenamos en el arreglo **addends**, luego llamo a la función anterior **sum** para sumar dichos valores y el resultado de esa suma es el valor de retorno

Resultado del testeo:

```
PS D:\CURSOS\DESARROLLO SOFTWARE\TAREAS\HolaRuby> rspec -e '#max_2_sum' .\spec\parte1_spec.rb
Run options: include {:full_description=>/\#max_2_sum/}
.....

Finished in 0.01863 seconds (files took 0.37651 seconds to load)
5 examples, 0 failures
```

```
def sum_to_n? arr, n
  i = 0
  until i == arr.length do
    j=0
    until j == arr.length do
      if(i!=j && n == (arr[i].to_i + arr[j].to_i))
        return true
      end
      j = j+1
    end
    i = i+1
  end

  return false
end
```

Básicamente aquí fijamos un elemento de arreglo **arr[i]** y comprobamos sumándolo con los demás elementos **arr[j]**, y en las iteraciones cuando una suma coincida con el valor de **n**, retornamos **true**, pero si no hubo coincidencias retornamos **false** (penúltima línea)

Resultado del testeo:

```
PS D:\CURSOS\DESARROLLO SOFTWARE\TAREAS\HolaRuby> rspec -e '#sum_to_n' .\spec\parte1_spec.rb
Run options: include {:full_description=>/\#sum_to_n/}
....

Finished in 0.00745 seconds (files took 0.3692 seconds to load)
4 examples, 0 failures
```

PARTE 2

```
def hello(name)
  "Hello, #{name}"
end
```

El argumento se concatena con **"Hello"** usando **#{ }** y se retorna todo el String

Resultado del testeo:

```
PS D:\CURSOS\DESARROLLO SOFTWARE\TAREAS\HolaRuby> rspec -e '#hello' .\spec\parte2_spec.rb
Run options: include {:full_description=>/\#hello/}
..

Finished in 0.00694 seconds (files took 0.37658 seconds to load)
2 examples, 0 failures

PS D:\CURSOS\DESARROLLO SOFTWARE\TAREAS\HolaRuby> 
```

```
def starts_with_consonant? s
  if s.length == 0 or !/[a-zA-Z]/.match? s[0]
    return false
  end

  vowels = %w[a e i o u]

  b = vowels.include? s[0].downcase
  !b
end
```

S es un String y en la primera condicional verificamos si su longitud es cero, también verificamos si el primer elemento no es una letra usando el método **match?**, si la condición se cumple se retorna falso pues no sería consonante.

Luego creamos un array de vocales en minúscula, y el primer elemento de S lo convertimos en minúscula (para que la comparación se pueda dar), y verificamos si s[0] es una vocal, si no lo es entonces b será falso, y retornamos lo opuesto (true) pues la función quiere verificar si es consonante o no.

Resultado del testeo:

```
PS D:\CURSOS\DESARROLLO SOFTWARE\TAREAS\HolaRuby> rspec -e '#starts_with_consonant?' .\spec\parte2_spec.rb
Run options: include {:full_description=>/\#starts_with_consonant\?/}
.....

Finished in 0.0078 seconds (files took 0.37335 seconds to load)
5 examples, 0 failures
```



```
def binary_multiple_of_4? s
  if s.is_a?(String)

    return false if s == ''
    return false if /[a-zA-z]/.match? s

    if s.to_i(2) % 4 == 0
      return true
    end

    return false
  end

  'Input must be a string'
end
```

Verificamos primero si el argumento es un String, de no serlo retornamos un mensaje "Input must be a string", pero de serlo se hace lo siguiente:

- Primero verificamos si es un numero binario valido, para ello verificamos si es un String vacío y si contiene alguna letra, de cumplirse una de las dos condiciones entonces no sería un binario y retornamos falso

- Luego, si es binario, lo transformamos a un numero en base 10 (**to_i(2)**) y verificamos si es múltiplo de cuatro para retornar true, de no serlo se retorna false

Resultado del testeo:

```
PS D:\CURSOS\DESARROLLO SOFTWARE\TAREAS\HolaRuby> rspec -e '#binary_multiple_of_4?' .\spec\parte2_spec.rb
Run options: include {:full_description=>/\#binary_multiple_of_4\?/}
...


Finished in 0.00761 seconds (files took 0.3574 seconds to load)
3 examples, 0 failures
```

PARTE 3

```
class BookInStock
  def initialize(isbn,price)
    if isbn.empty? || price <=0
      raise ArgumentError,'Invalid value'
    end
    @isbn = isbn
    @price = price
  end

  def price_as_string
    "$#{format("%.2f", @price.to_f)}" #Especifica que debe imprimirse con 2 decimales
  end

  attr_accessor :isbn, :price
end
```



Una opción que nos brinda ruby es definir los getters y setters en una sola línea (flecha roja) en este caso creamos los getters y setter para **isbn** y **price**.

Resultado del testeo:

```
PS D:\CURSOS\DESARROLLO SOFTWARE\TAREAS\HolaRuby> rspec -e 'getters y setters' spec/parte3_spec.rb
Run options: include {:full_description=>/getters\ y\ setters/}
....

Finished in 0.00451 seconds (files took 0.3763 seconds to load)
4 examples, 0 failures
```

```
class BookInStock
  def initialize(isbn,price)
    if isbn.empty? || price <=0
      raise ArgumentError, 'Invalid value'
    end
    @isbn = isbn
    @price = price
  end

  def price_as_string
    "$#{format("%.2f", @price.to_f)}" #Especifica que debe imprimirse con 2 decimales
  end

  attr_accessor :isbn, :price
end
```

En el constructor ponemos una condicional para verificar la validez de los atributos, si hay algún valor invalido lanza un error (Invalid value) pero si son válidos inicializa los atributos

Resultado del testeo:

```
PS D:\CURSOS\DESARROLLO SOFTWARE\TAREAS\HolaRuby> rspec -e 'constructor' spec/parte3_spec.rb
Run options: include {:full_description=>/constructor/}
...

Finished in 0.00482 seconds (files took 0.3775 seconds to load)
3 examples, 0 failures
```

```
class BookInStock
  def initialize(isbn,price)
    if isbn.empty? || price <=0
      raise ArgumentError, 'Invalid value'
    end
    @isbn = isbn
    @price = price
  end

  def price_as_string
    "$#{format("%.2f", @price.to_f)}" #Especifica que debe imprimirse con 2 decimales
  end

  attr_accessor :isbn, :price
end
```

Para la función **price_as_String** usaremos la función **format** en donde el primer parámetro especificamos la cantidad de decimales y en el segundo, el valor que deseamos imprimir con esa cantidad de decimales y a ese valor le añadimos el \$ y lo retornamos

Resultado del testeo:

```
PS D:\CURSOS\DESARROLLO SOFTWARE\TAREAS\HolaRuby> rspec -e '#price_as_string' spec/parte3_spec.rb
Run options: include {:full_description=>/\#price_as_string/}
....

Finished in 0.00789 seconds (files took 0.37422 seconds to load)
4 examples, 0 failures
```