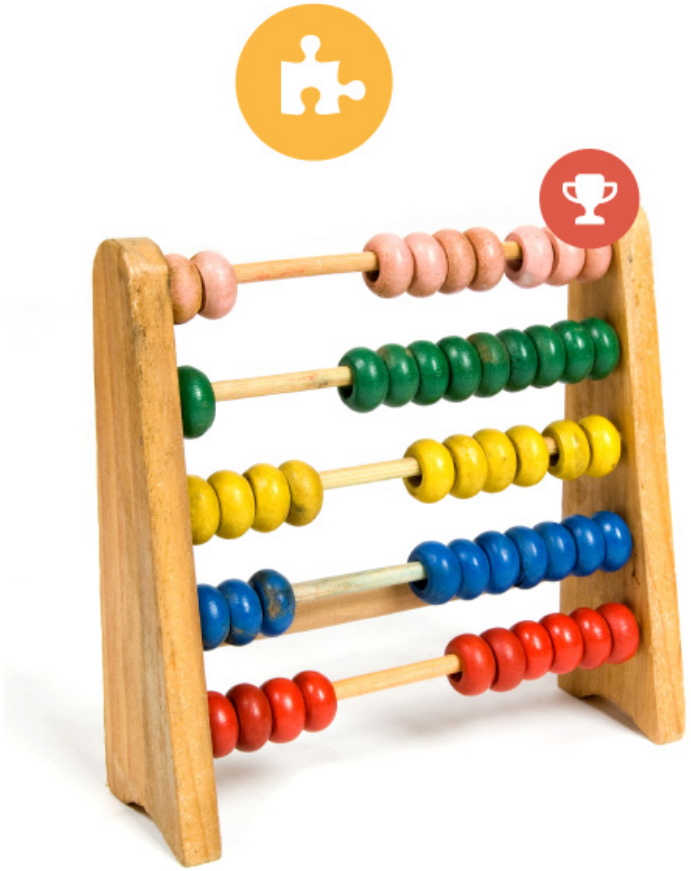
Programación en Lenguaje Ruby

Strings

Delio Tolivia

Gijón crece 2016 / 2019
AYTO. DE GIJÓN / CCOO / FADE / UGT
Ayuntamiento de Gijón
Gijón impulsa Empleo
dicampus TALENTO FORMACIÓN



Índice

Introducción a String

- Las cadenas de texto o ***strings*** son fundamentales para comunicarse con el usuario y por tanto serán muy utilizadas desde nuestros primeros pasos al programar.
- Se pueden construir cadenas de forma literal mediante el uso de comillas dobles o simples.

```
2.0.0-p648 > "January"
 => "January"
2.0.0-p648 > 'January'
 => "January"
```

- Se usa la \ para escapar elementos como las “ por ejemplo
- Se puede usar la sintaxis %Q[…….] y no hará falta escapar nada dentro de ella
- Todas las cadenas son instancias de la clase ***String***.
- La diferencia entre comillas dobles y simples es que las primeras permiten secuencias de escape como ***\n*** (salto de línea).

Longitud de un String

- Para saber la longitud de una cadena disponemos del método ***length*** (podemos saber todos los métodos llamando a ***methods***).

```
2.0.0-p648 > "January".length
 => 7
```

Interpolación de Strings (I)

- Es esencial saber remplazar placeholders (marcadores de posición) por los valores que representan:

```
2.0.0-p648 > a=1
 => 1
2.0.0-p648 > b=4
 => 4
2.0.0-p648 > puts "The number #{a} is less than #{b}"
The number 1 is less than 4
 => nil
```

- Los placeholders no son variables, son bloques de código Ruby que serán evaluados e insertados en esa localización

Interpolación de Strings (II)

- Aquí definimos un método que recibe una cadena y que como resultado inserta en el placeholder la longitud de la misma

```
2.0.0-p648 > def string_length_interpolater(incoming_string)
2.0.0-p648 ?>     "The string you just gave me has a length of
 #{incoming_string.length}"
2.0.0-p648 ?>   end
 => nil
2.0.0-p648 > string_length_interpolater("hello")
 => "The string you just gave me has a length of 5"
```

Búsqueda de un String (I)

- Otro escenario común es la búsqueda de caracteres, subcadenas o palabras en otro string. Para ello tenemos el método ***include?***:

```
2.0.0-p648 > "[Luke:] I can't believe it. [Yoda:] That is why you fail.".include?
 ("Yoda")
 => true
```

- Para saber si una cadena empieza por otra tenemos el método ***start_with?***

```
2.0.0-p648 > "Ruby is a beautiful language".start_with?("Ruby")
 => true
```

- Para comprobar si termina con otra tenemos el método ***end_with?***

```
2.0.0-p648 > "I can't work with any other language but Ruby".end_with?("Ruby")
 => true
```

- Es común terminar el nombre de los métodos que devuelven un booleano con **?**

Búsqueda de un String (II)

- Para conocer el índice donde se encuentra un determinado carácter, subcadena o cadena tenemos el método ***index***

```
2.0.0-p648 > "I am a Rubyist".index("R")
 => 7
2.0.0-p648 > "I am a Rubyist".index("Ruby")
 => 7
```

Cambios en un String

- Para pasar todas las letras de una cadena a mayúsculas tenemos el método ***upercase***:

```
2.0.0-p648 >   puts 'i am in lowercase'.upcase
I AM IN LOWERCASE
 => nil
```

- Para pasar todo a minúsculas tenemos ***downcase***:

```
2.0.0-p648 > 'This is Mixed CASE'.downcase
 => "this is mixed case”
```

- Existe el método ***swapcase*** para intercambiar mayúsculas por minúsculas y viceversa:

```
2.0.0-p648 > "ThiS iS A vErY ComPlEx SenTeNcE".swapcase
 => "tHIs Is a VeRy cOMpLeX sENtEnCe"
```

Partición de un String

- En múltiples ocasiones se querrá partir una cadena larga en palabras o subcadenas utilizando un separador como el espacio, la coma, etc. Para ellos disponemos del método ***split*** que recibe como argumento el separador. Este método nos devolverá un ***array*** con los elementos separados:

```
2.0.0-p648 > 'Fear is the path to the dark side'.split(' ')
 => ["Fear", "is", "the", "path", "to", "the", "dark", "side"]
```

Ejercicio

- Hacer el Ejercicio 3 del pdf de ejercicios
- Hacer el Ejercicio 4 del pdf de ejercicios

Concatenación de Strings

- Se pueden concatenar utilizando el operador **+**

```
2.0.0-p648 > 'Concate'+'nating'
 => "Concatenating"
```

- También disponemos del método ***concat***

```
2.0.0-p648 > 'Concate'.concat('nating')
 => "Concatenating"
```

- Con el operador **<<** (la diferencia de este es que modifica la primera cadena concatenándole la segunda, los otros métodos crean una cadena nueva)

```
2.0.0-p648 > 'Concate'<<'nating'
 => "Concatenating"
```

Remplazar una subcadena

- El método ***sub*** remplaza por la subcadena indicada como segundo argumento la primera aparición de la subcadena indicada como primer argumento:

```
2.0.0-p648 > "I should look into your problem when I get time".sub('I','We')
 => "We should look into your problem when I get time"
```

- Para substituir todas las apariciones tenemos el método ***gsub***:

```
2.0.0-p648 > "I should look into your problem when I get
 time".gsub('I','We')
 => "We should look into your problem when We get time"
```

- Se pueden usar expresiones regulares para las búsquedas:

```
2.0.0-p648 > 'Replacing'.gsub(/[aeiou]/,'1')
 => "R1pl1c1ng"
```

Ejercicio

- Realizar el ejercicio 7 del pdf de ejercicios

Búsquedas con expresiones regulares

- Para buscar una cadena utilizando expresiones regulares tenemos el método ***match***. Por ejemplo para buscar la primera aparición de un carácter que tiene antes un espacio en blanco:

```
2.0.0-p648 > 'Ruby Is Pretty Brilliant'.match(/ ./)
 => #<MatchData " I">
```

- Podemos indicar como segundo parámetro a partir de que índice de la cadena tiene que buscar:

```
2.0.0-p648 :009 > 'Ruby Is Pretty Brilliant'.match(/ ./,9)
 => #<MatchData " B">
```