Guía 1: Introducción al lenguaje Ruby (parte I)

Introducción

Ruby es un lenguaje de programación orientado a objetos de código abierto, reconocido por ser un lenguaje rápido y sencillo. Su creador Yukihiro "Matz" Matsumo mezcló sus lenguajes favoritos (Perl, Smalltalk, Eiffel, Ada y Lisp) para formar un nuevo lenguaje que incorporara lo mejor de cada uno.

En la presente guía el estudiante aprenderá un poco sobre el lenguaje Ruby, podrá trabajar con variables String, números; métodos relacionados con String y cómo se comportan estos objetos dentro del lenguaje.

Objetivos

- Aprender a declarar variables en Ruby.
- Conocer el funcionamiento de los métodos del lenguaje Ruby.
- Distinguir los tipos de objetos que contiene el lenguaje Ruby.

Tiempo

- Una sesión de clase.

Requerimientos

Software	Hardware
Sistema Debian 11 virtualizado en Virtual box con: • Ruby versión 2.4.1 • Rails 5.4.1 • Nodejs • Vscode	Computadora con características: • Memoria RAM mínimo 2GB • Procesador mínimo 2.1 GHz

Referencia

- RubySur. Aprende a programar. rubysur.org/aprende.a.programar/
- Ruby community. Ruby a programmer´s Best friend. https://www.ruby-lang.org/en/documentation/
- Ruby community. Ruby a programmer´s Best friend. https://www.ruby-lang.org/en/documentation/quickstart/

Desarrollo

1. String.
   1.1. Crear un directorio llamado ruby, donde se almacenarán los ejercicios que se llevarán a cabo a lo largo de esta guía.
   1.2. Crear un programa primer_programa.rb, se puede hacer desde el terminal o desde el editor de texto, asignar 2 variables de tipo String para luego imprimir por pantalla las 2 variables concatenadas.

```
var_1 = "hola"
var_2 = "mundo"
puts var_1  + var_2
```

   1.3. Ejecutar el programa en el terminal.

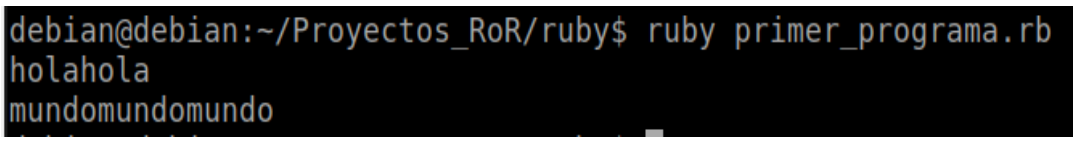
```
$ ruby primer_programa.rb
```

   Obtendrá una salida de las 2 variables concatenadas

   1.4. Editar el archivo creado anteriormente, agregar el siguiente código y ver lo que se muestra por pantalla.

```
var_1 = "hola"
var_2 = "mundo"
puts var_1 * 2
```

Como se observa, se multiplica el contenido de las variables que son de tipo String, por la cantidad de veces que se le diga, en este caso la palabra hola se muestra 2 veces y la palabra mundo 3 veces.



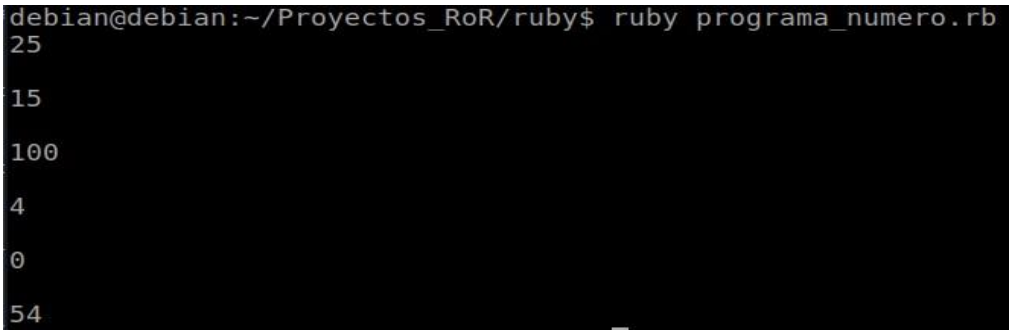
2. Números

2.1. Crear un programa nuevo llamado programa_numero.rb, en el que se asignarán 2 variables enteras para realizar operaciones de aritmética básica.

```
var_1 = 20
var_2 = 5

#suma
puts var_1 + var_2
puts ""

#resta
puts var_1 - var_2
puts ""

#multiplicar
puts var_1 * var_2
puts ""

#dividir
puts var_1 / var_2
puts ""

#modulo
puts var_1 % var_2
puts ""

#números aleatorios
puts rand(100)
```

2.2. Ejecutar el programa en el terminal y observar la salida.

```
$ ruby programa_numero.rb
```



3. Conversiones

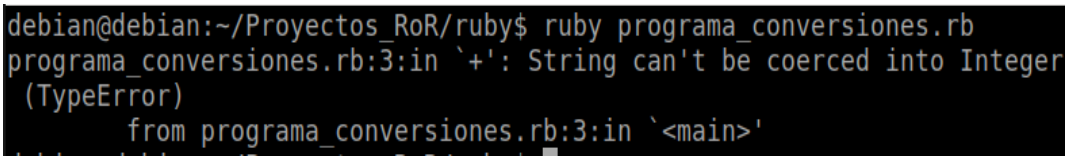
3.1 En Ruby existen distintos métodos que se aplican a objetos como los String, números enteros, etc. Existen métodos especiales de conversiones que se utilizan en diferentes formas o casos, para observar el funcionamiento de estos, crear un archivo programa_conversiones.rb, declarar una variable entera y concatenar con un texto.

```
var_1 = 22
puts var_1 + " Esto es un entero"
```

3.2 Ejecutar el programa en el terminal.

```
$ ruby programa_conversiones.rb
```

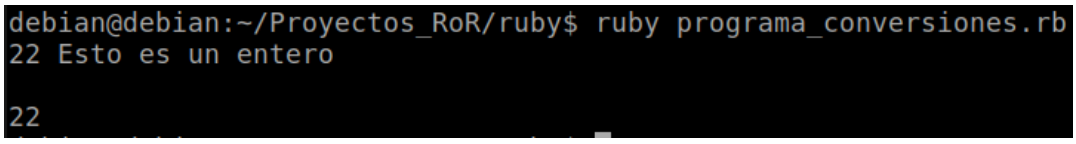
Imprime un error a como se observa en la siguiente figura, esto es debido a que no se puede concatenar un objeto de tipo entero con una cadena de caracteres.



3.3 Para solucionar ese error, hacer uso del método **to_s**, editar el programa y agregar:

```
var_1 = 22
puts var_1.to_s + " Esto es un entero"
puts ""
puts var_1
```

Se obtendrá una salida como en la mostrada en la siguiente figura, como se observa, aunque se ha utilizado el método to_s, la variable **var_1** sigue teniendo el mismo valor entero, pero su representación es como cadena de caracteres.



3.4 Editar nuevamente el programa para hacer uso de los métodos **to_i,** el cual convierte una variable a entero y **to_f,** el cual convierte una variable a flotante.

```
var_1 = 22
var_2 = "22"

puts var_1.to_s + " Esto es un entero"
puts ""

puts var_2 + " Esto es una cadena"
puts "La suma de las variables es:"

puts var_2.to_i + var_1
puts var_2.to_f
```

3.5 Guarda los cambios y ejecutar el programa en el terminal.

```
$ ruby programa_conversiones.rb
```

Se obtendrá la siguiente salida.

```
debian@debian:~/Proyectos_RoR/ruby$ ruby programa_conversiones.rb
22 Esto es un entero

22 Esto es una cadena
La suma de las variables es:
44
22.0
```

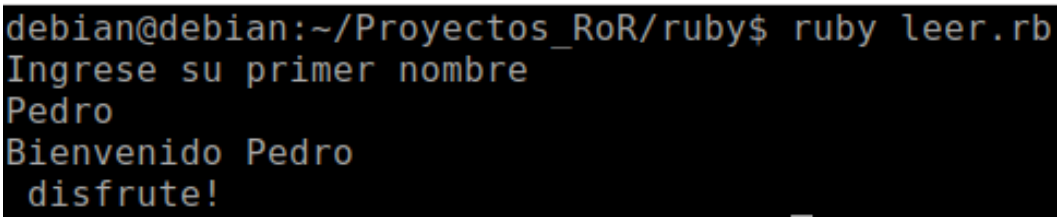
4. Métodos gets y chomp.

Se ha visto que el método **puts** se utiliza para imprimir en la pantalla; por el contrario, para leer existe el método **gets** que trabaja junto con el método **chomp**, lo que hace este último es eliminar el carácter “enter” al momento de que el método gets lee un dato del teclado.

4.1 Crear un programa leer.rb y agregar el siguiente código.

```
puts "Ingrese su primer nombre"
nombre = gets
puts "Bienvenido "+ nombre + "disfrute! "
```

Al ejecutar el programa se obtiene la siguiente salida.

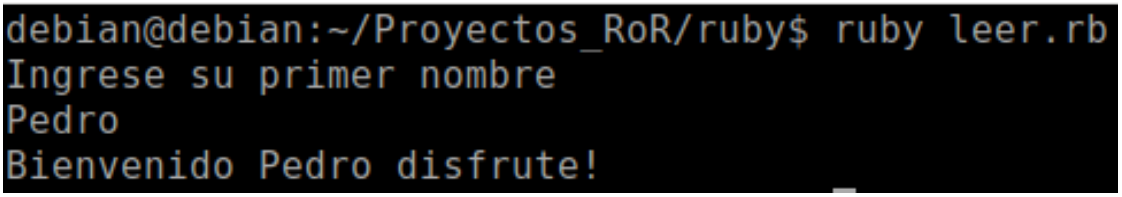


Como se puede observar, el método gets recibe el carácter “enter” como un carácter más de lectura, para solucionar eso es que se utiliza el método chomp.

4.2 Editar el programa anterior y utilizar el método chomp al momento de leer el nombre.

```
puts "Ingrese su primer nombre"
nombre = gets.chomp

puts "Bienvenido #{nombre} disfrute! "
```

Al ejecutar el programa se observa la diferencia, cuando se usa el método **chomp** que ya no captura el enter como un carácter más, de igual forma se observa como la variable **nombre** es impresa de una forma distinta a las anteriores.



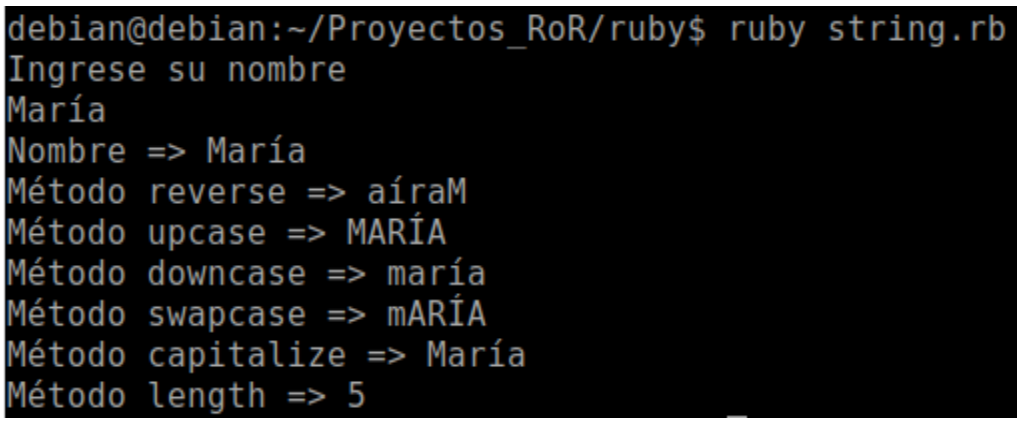
5. Métodos de String

Como se menciona anteriormente, en Ruby existen distintos métodos que se pueden aplicar a cada uno de los objetos del lenguaje, en esta sección se conocerá sobre los métodos relacionados a los String.

5.1 Crear un nuevo programa string.rb y agregar el siguiente código:

```
puts "Ingrese su nombre"
nombre = gets.chomp

    #Imprime el nombre ingresado
    puts "Nombre => "+ nombre

    # convierte al revés el nombre
    puts "Método reverse => " + nombre.reverse

    #Mayúscula
    puts "Método upcase => " + nombre.upcase

    #Minúscula
    puts "Método downcase => " + nombre.downcase

    #Intercambia las minúsculas por mayúscula (viceversa)
    puts "Método swapcase => " + nombre.swapcase

    #cambia el primer carácter a mayúscula
    puts "Método capitalize => " + nombre.capitalize

    #devuelve el tamaño del string ingresado
    puts "Método length => " + nombre.length.to_s
```

5.2 Ejecute el programa en el terminal y observar el comportamiento de los métodos.



6. Condicionales y bucles

6.1 Los condicionales y los bucles en Ruby funcionan de la misma manera que en otros lenguajes de programación, para ver el funcionamiento, crear un programa nuevo y agregar el siguiente código.

```
iterador = " "

while iterador.downcase != "s"

puts "Ingrese un nombre"
nombre = gets.chomp
tamaño = nombre.length

    if (tamaño >= 5 )

     puts "Su nombre tiene más de 5 caracteres"

    else

     puts "Su nombre tiene menos de 5 caracteres"

    end
puts "\nPara salir presione la letra S"


iterador = gets.chomp
end
puts "Ha salido del programa"
```

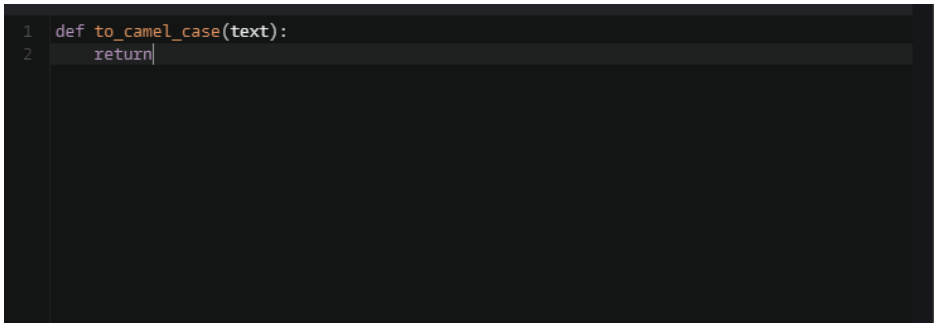
6.2 Ejecutar e interactuar con el programa para ver su funcionamiento.

```
debian@debian:~/Proyectos_RoR/ruby$ ruby condicionales.rb
Ingrese un nombre
Francisco
Su nombre tiene mas de 5 caracteres

Para salir presione la letra S
n
Ingrese un nombre
juan
Su nombre tiene menos de 5 caracteres

Para salir presione la letra S
n
Ingrese un nombre
Rafael
Su nombre tiene mas de 5 caracteres

Para salir presione la letra S
s
Ha salido del programa
```

Ejercicios propuestos para ser entregados al docente

1. Realizar cada uno de los enunciados de la guía, probar su funcionamiento y analizar cada uno de los programas planteados.

2. Complete el método/función para que convierta las palabras delimitadas por guiones/guiones bajos en mayúsculas y minúsculas. La primera palabra dentro de la salida debe estar en mayúsculas solo si la palabra original estaba en mayúsculas (conocido como Upper Camel Case, también conocido como caso Pascal). Las siguientes palabras deben estar siempre en mayúscula.



Ejemplos
"the-stealth-warrior" se convierte en "the Stealth Warrior"

"The_Stealth_Warrior" se convierte en "The_Stealth_Warrior"

"The_Stealth-Warrior" se convierte en "The_Stealth-Warrior"