**Guía 1: Introducción al lenguaje Ruby (parte I)**

# Desarrollo

1. String.
   1. Crear un directorio llamado ruby, donde se almacenarán los ejercicios que se llevarán a cabo a lo largo de esta guía.
   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.

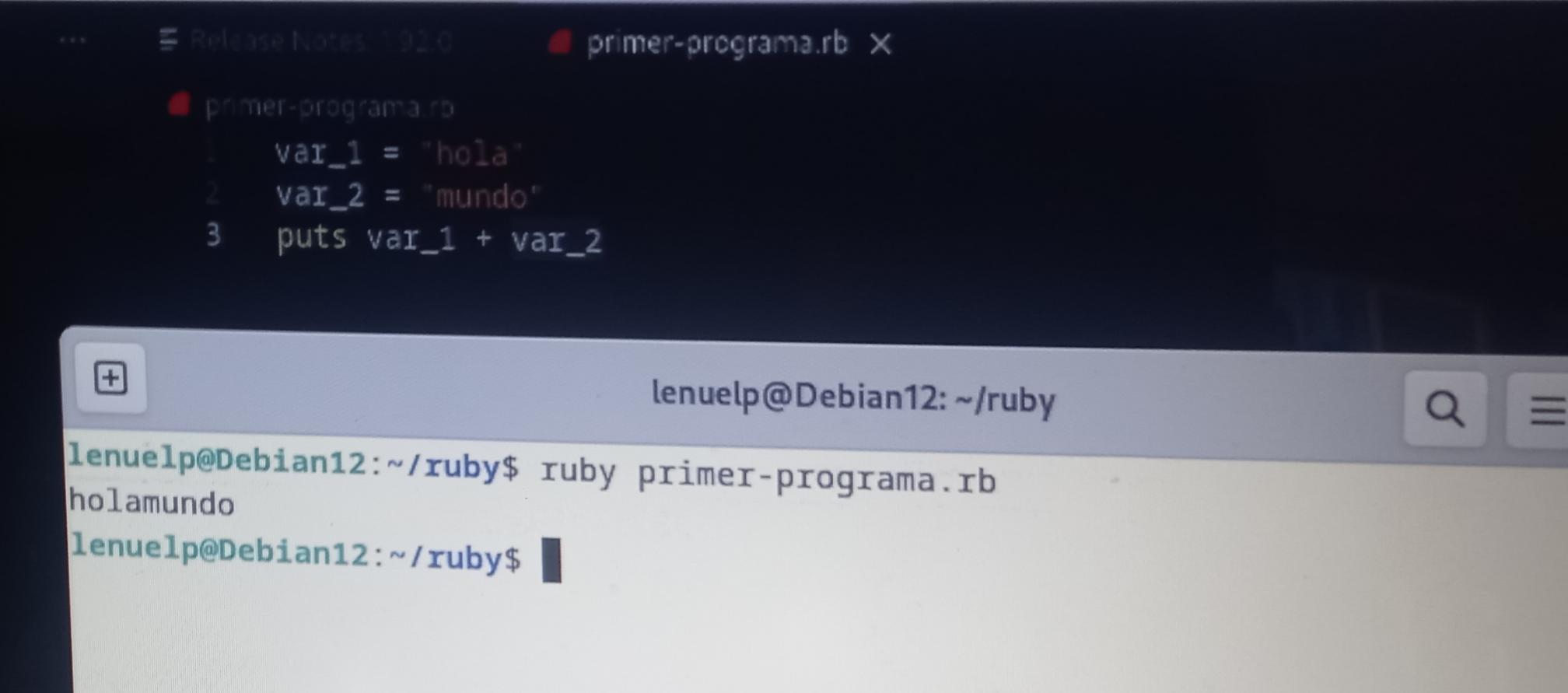
var\_1 = "hola" var\_2 = "mundo"

puts var\_1 + var\_2

* 1. Ejecutar el programa en el terminal.

$ ruby primer\_programa.rb

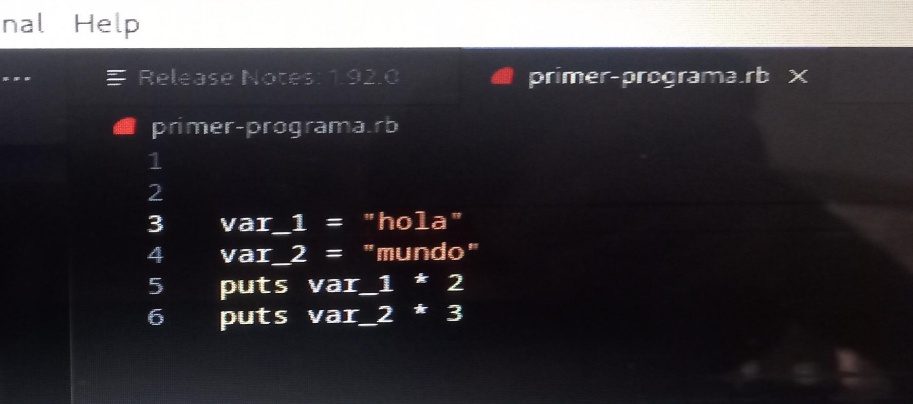
Obtendrá una salida de las 2 variables concatenadas

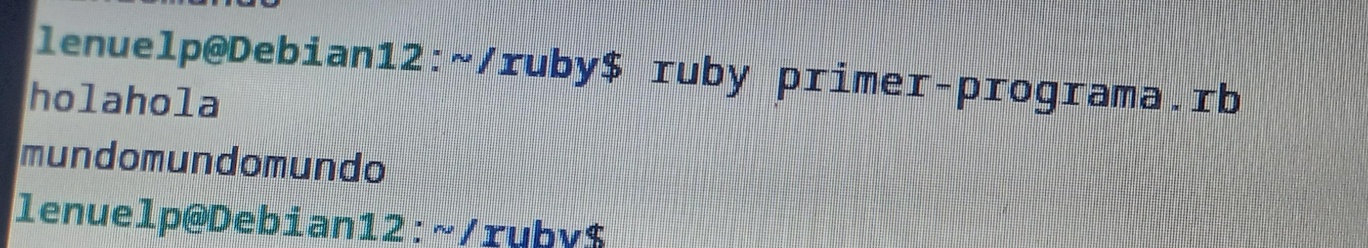


* 1. Editar el archivo creado anteriormente, agregar el siguiente código y ver lo que se muestra por pantalla.

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.





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 ""

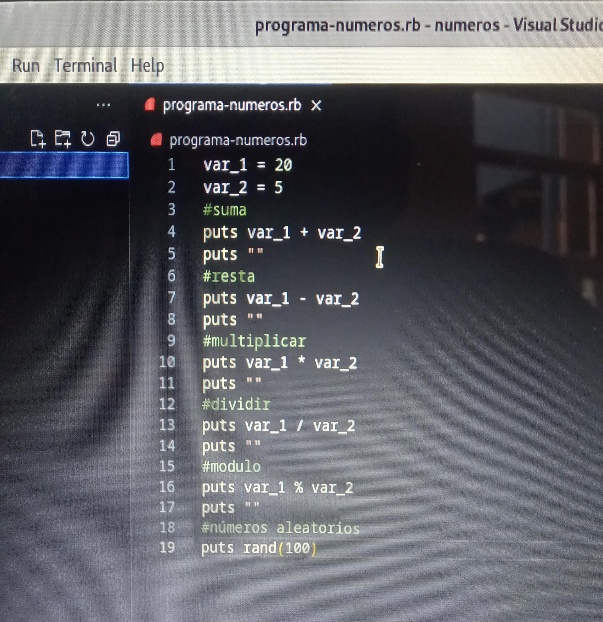
#modulo puts var\_1 % var\_2

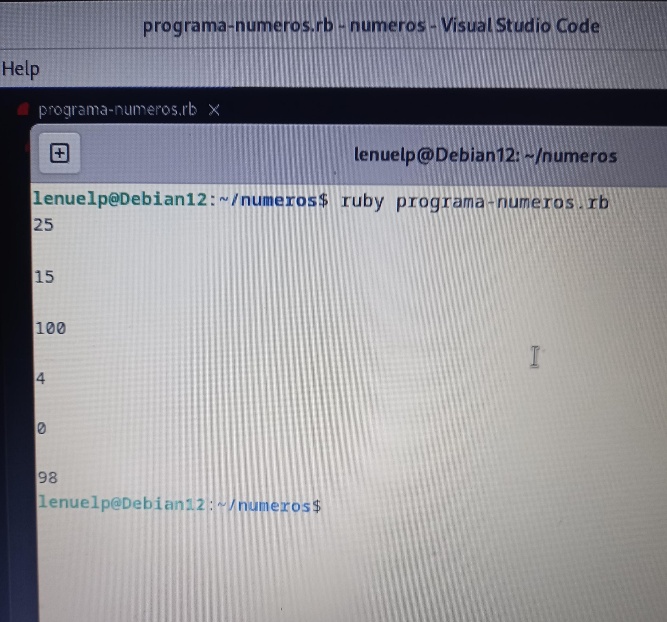
puts ""

#números aleatorios puts rand(100)

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

$ ruby programa\_numero.rb





1. Conversiones
   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.

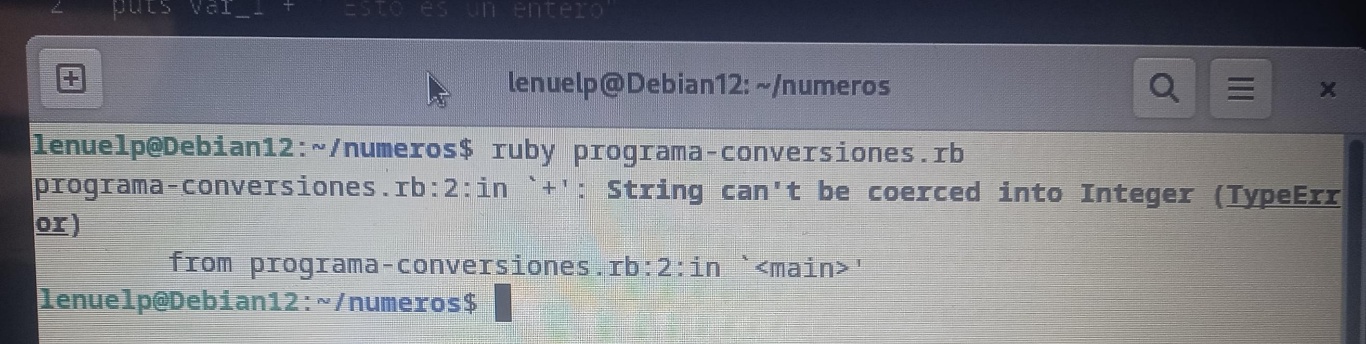
var\_1 = 22

puts var\_1 + " Esto es un entero"

* 1. 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.

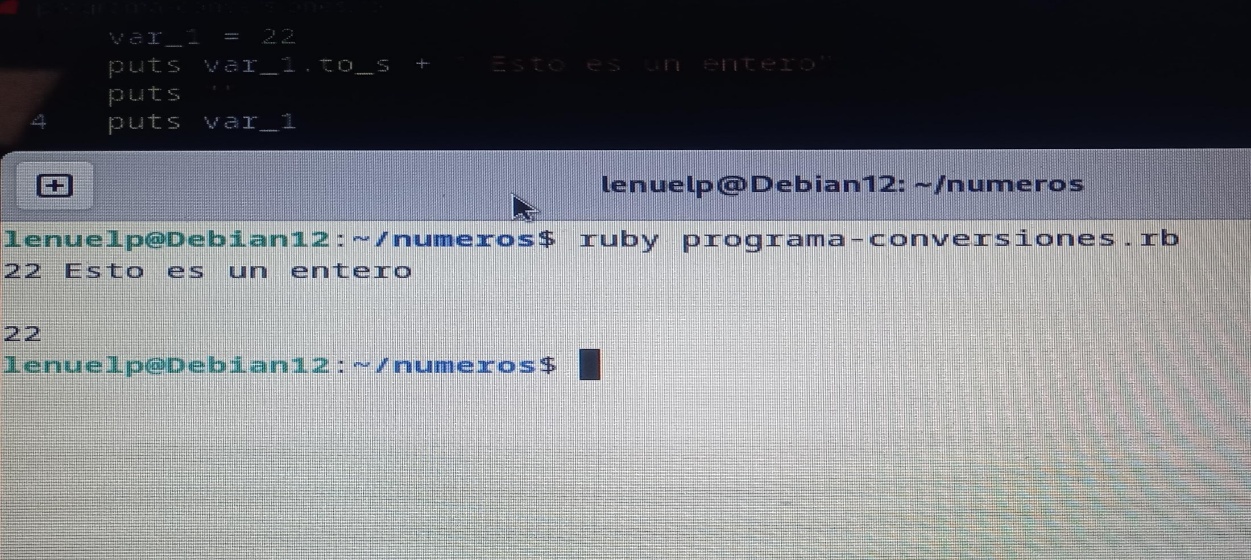


* 1. Para solucionar ese error, hacer uso del método **to\_s**, editar el programa y agregar:

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.



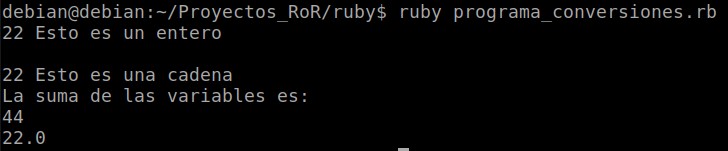
* 1. Editar nuevamente el programa para hacer uso de los métodos **to\_i,** el cual convierte una variable a entero y **to\_f,** el cual convierte una variable a flotante.

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

* 1. Guarda los cambios y ejecutar el programa en el terminal.

$ ruby programa\_conversiones.rb

Se obtendrá la siguiente salida.



1. Métodos gets y chomp.

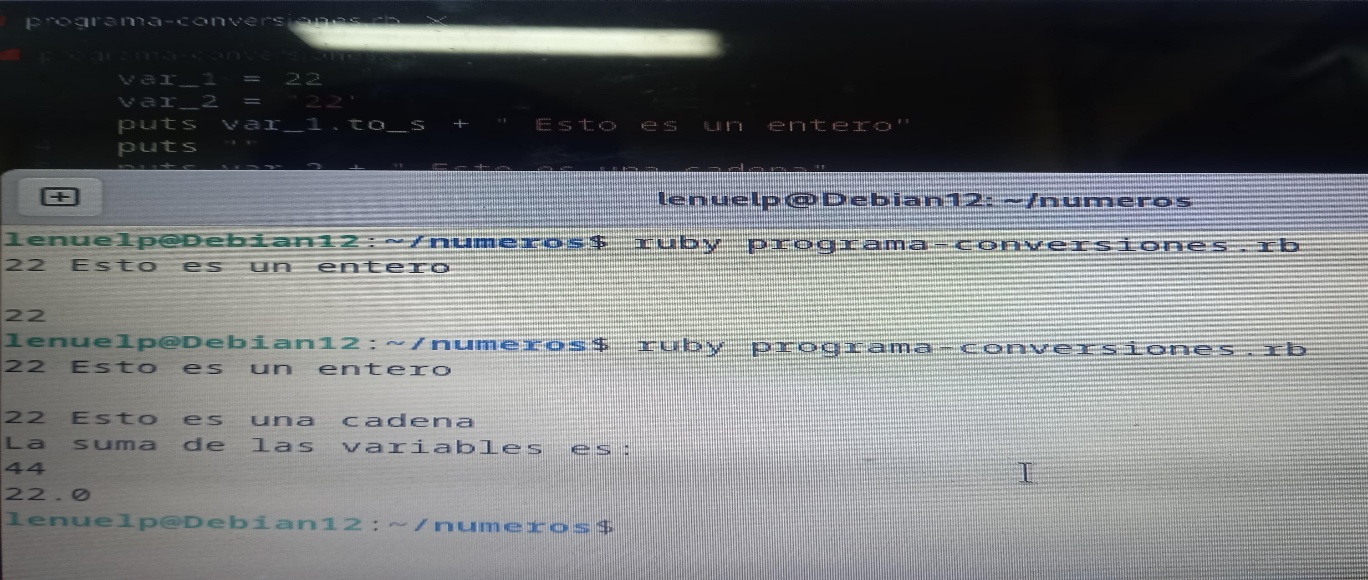
Se ha visto que el método **puts** se utiliza para imprimir en la pantalla; por el contrario, para leer existe el método **gets** que trabaja junto con el método **chomp**, lo que hace este último es eliminar el carácter “enter” al momento de que el método gets lee un dato del teclado.

* 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.



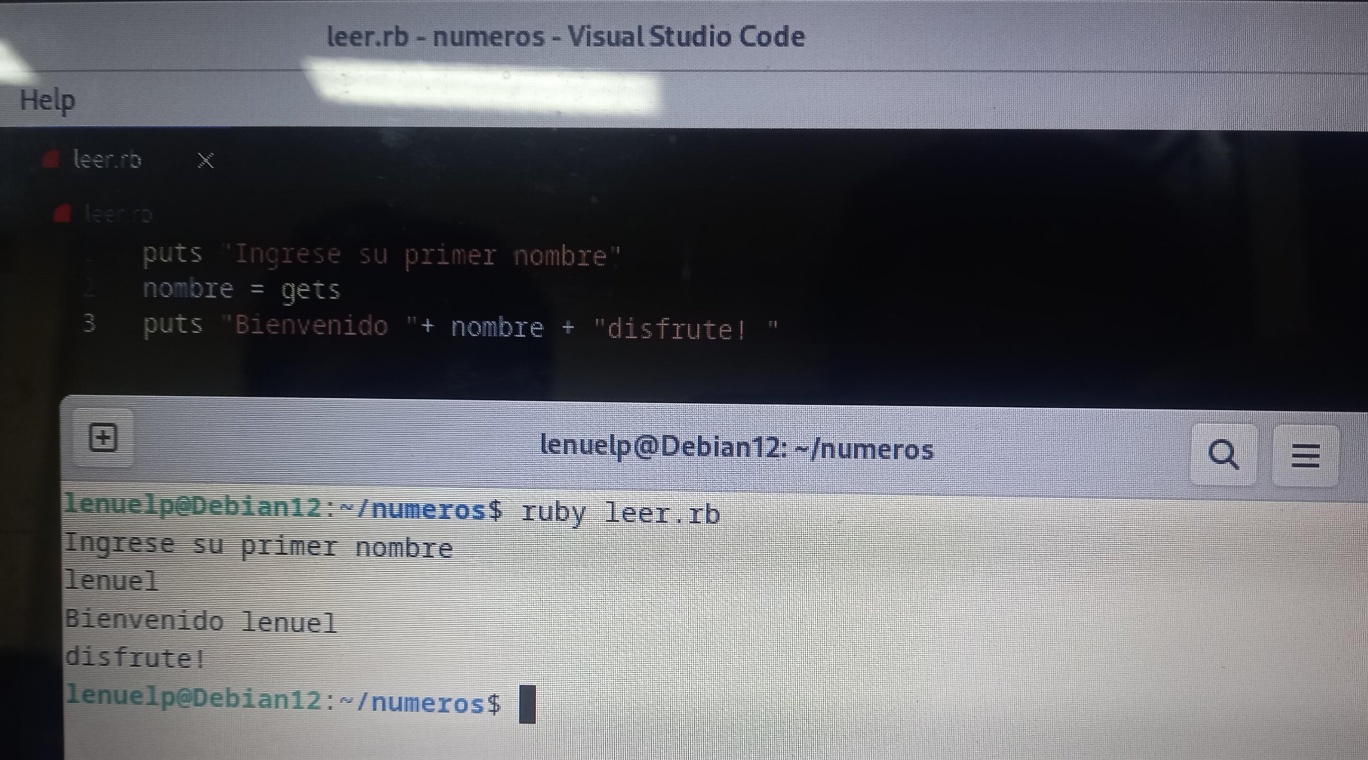
Como se puede observar, el método gets recibe el carácter “enter” como un carácter más de lectura, para solucionar eso es que se utiliza el método chomp.

* 1. 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.



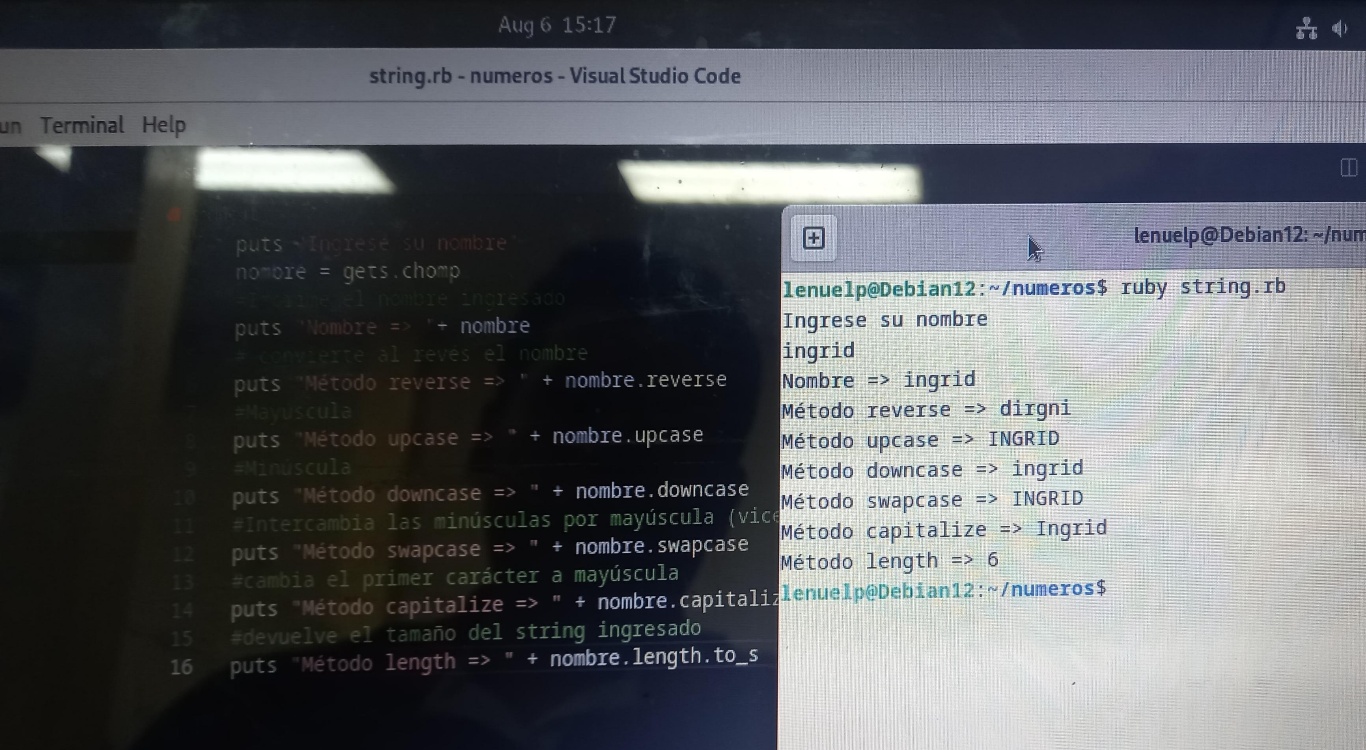
1. Métodos de String

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.

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

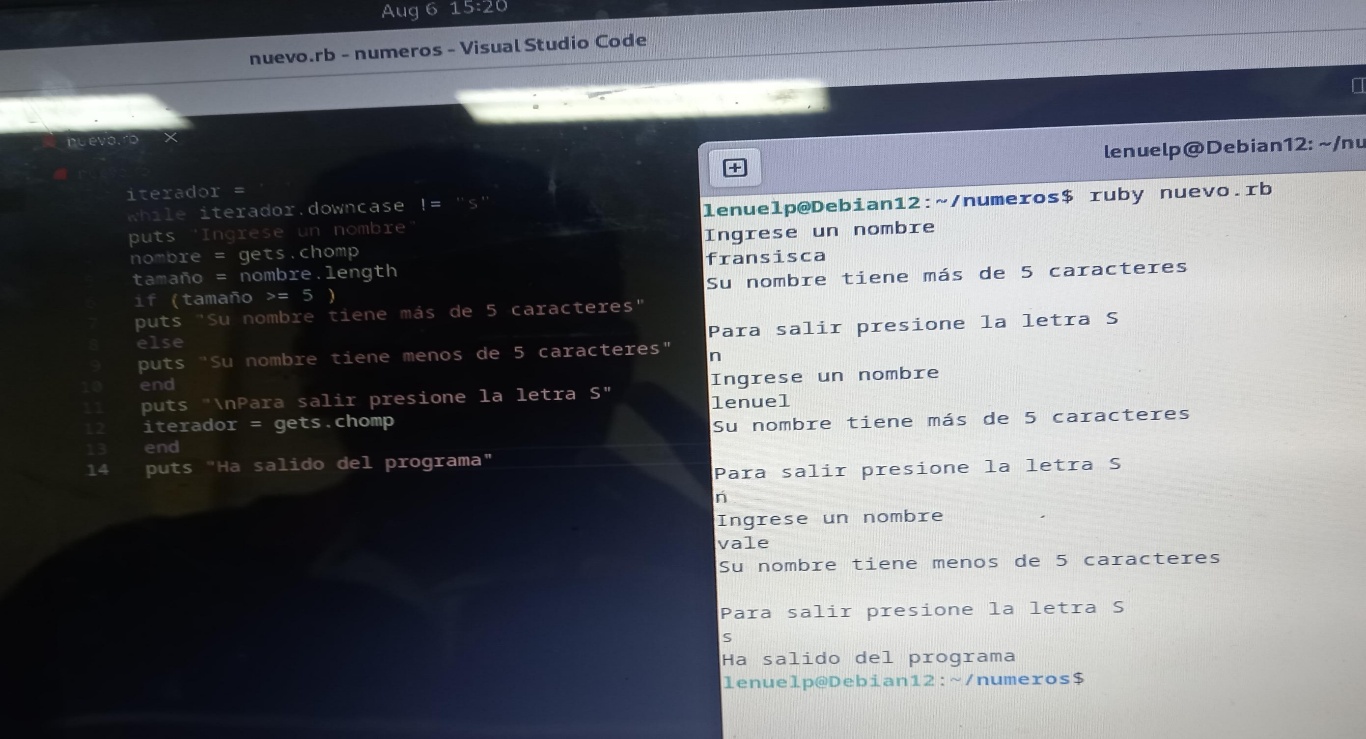
|  |
| --- |
| puts "Ingrese su nombre" nombre = gets.chomp  #Imprime el nombre ingresado puts "Nombre => "+ nombre  # convierte al revés el nombre  puts "Método reverse => " + nombre.reverse  #Mayúscula  puts "Método upcase => " + nombre.upcase  #Minúscula  puts "Método downcase => " + nombre.downcase    #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 |

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



1. Condicionales y bucles
   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.

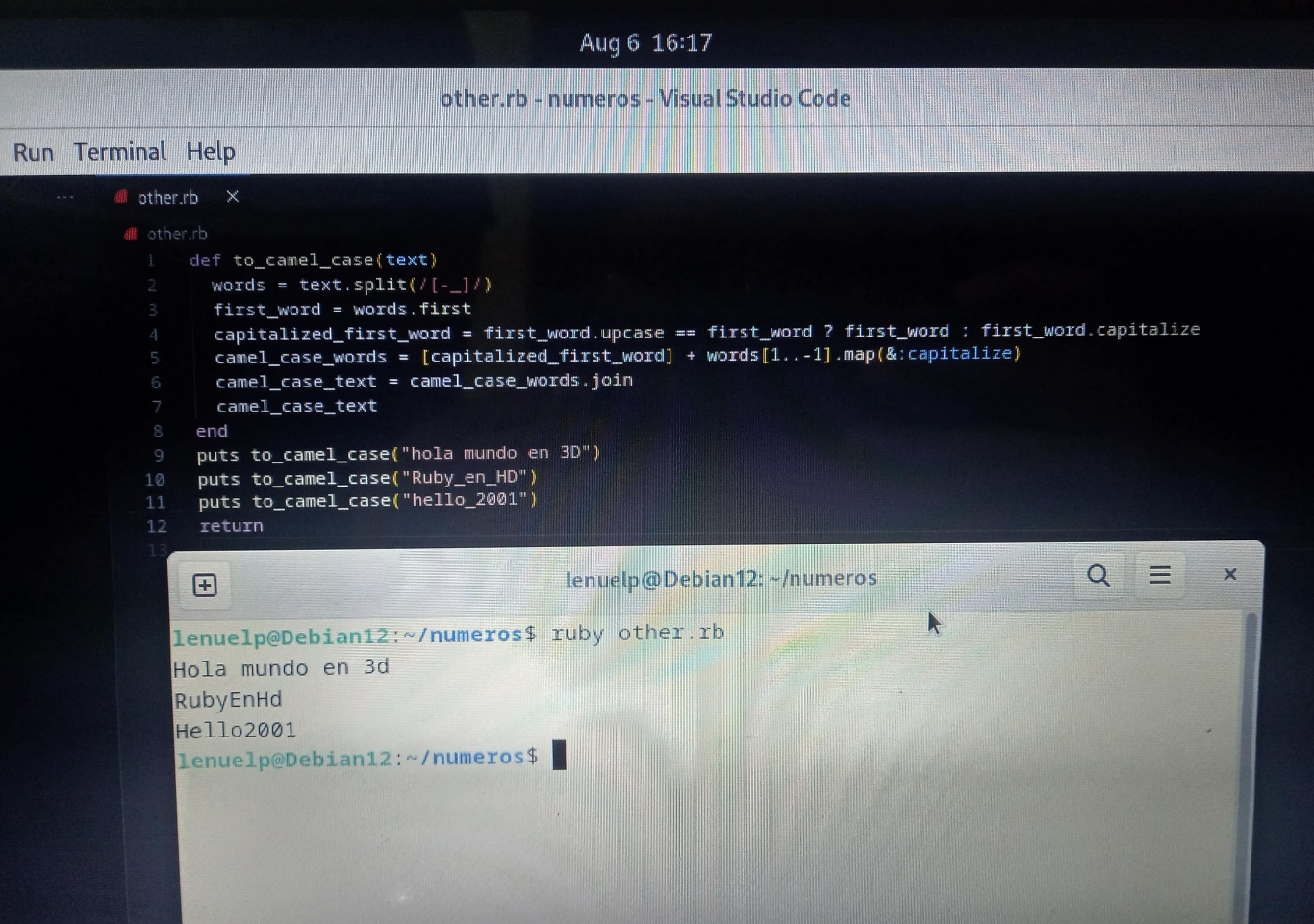
|  |
| --- |
| iterador = " "    while iterador.downcase != "s"    puts "Ingrese un nombre" nombre = gets.chomp  tamaño = nombre.length  if (tamaño >= 5 )    puts "Su nombre tiene más de 5 caracteres" else  puts "Su nombre tiene menos de 5 caracteres"    end  puts "\nPara salir presione la letra S"    iterador = gets.chomp end  puts "Ha salido del programa" |

* 1. Ejecutar e interactuar con el programa para ver su funcionamiento.

**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.

* + 1. 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"