

Universidad Nacional Autónoma de Nicaragua Unan León



Facultad de Ciencias y Tecnología.

Componente: Software como un servicio

Tema: Practica

Grupo:1

Docente: Erving Montes

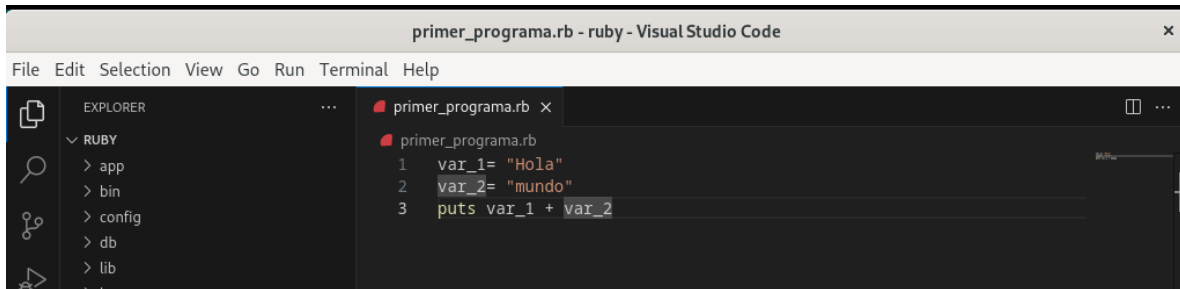
Integrante:

-Harold Steven Garcia Ramos.

Fecha: 8 de agosto de 2024

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

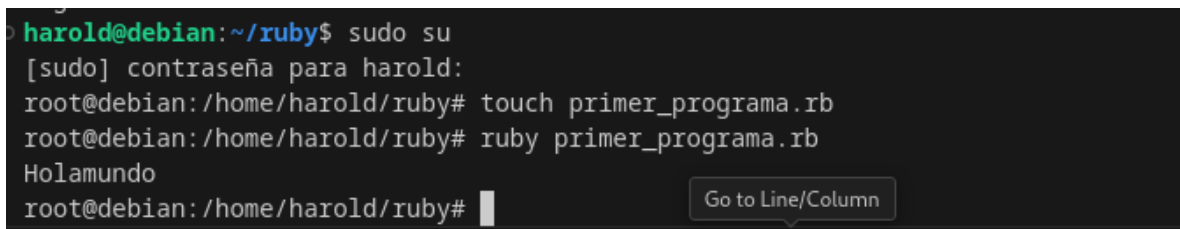
1.2. Crear un programa `primer_programa.rb`, se puede hacer desde el terminal o desde el editor de texto, asignar 2 variables de tipo String para luego imprimir por pantalla las 2 variables concatenadas.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a directory named 'RUBY' containing subdirectories 'app', 'bin', 'config', 'db', and 'lib'. The main editor window is open to 'primer_programa.rb' and contains the following Ruby code:

```
1 var_1= "Hola"
2 var_2= "mundo"
3 puts var_1 + var_2
```

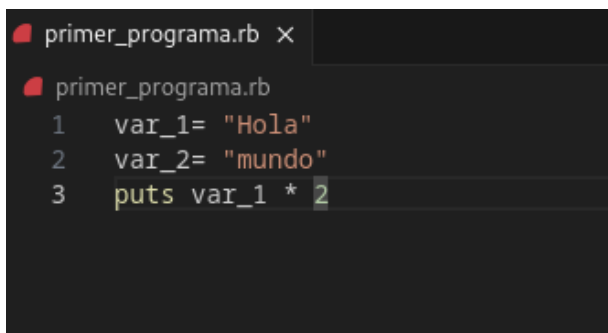
Ejecutar el programa en el terminal.



The screenshot shows a terminal window with the following commands and output:

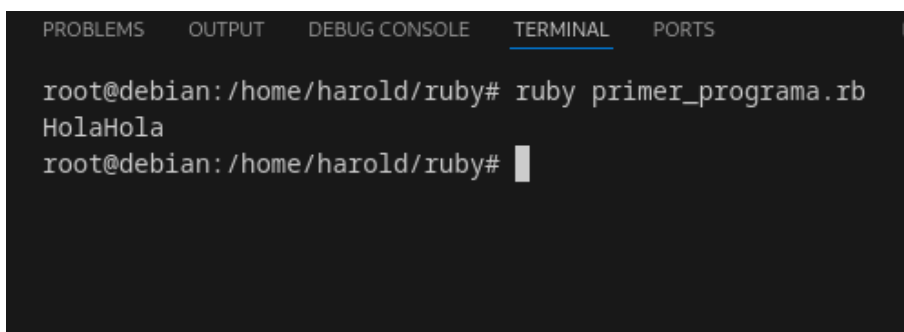
```
harold@debian:~/ruby$ sudo su
[sudo] contraseña para harold:
root@debian:/home/harold/ruby# touch primer_programa.rb
root@debian:/home/harold/ruby# ruby primer_programa.rb
Holamundo
root@debian:/home/harold/ruby#
```

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



The screenshot shows the Visual Studio Code editor with 'primer_programa.rb' open. The code has been updated to:

```
1 var_1= "Hola"
2 var_2= "mundo"
3 puts var_1 * 2
```

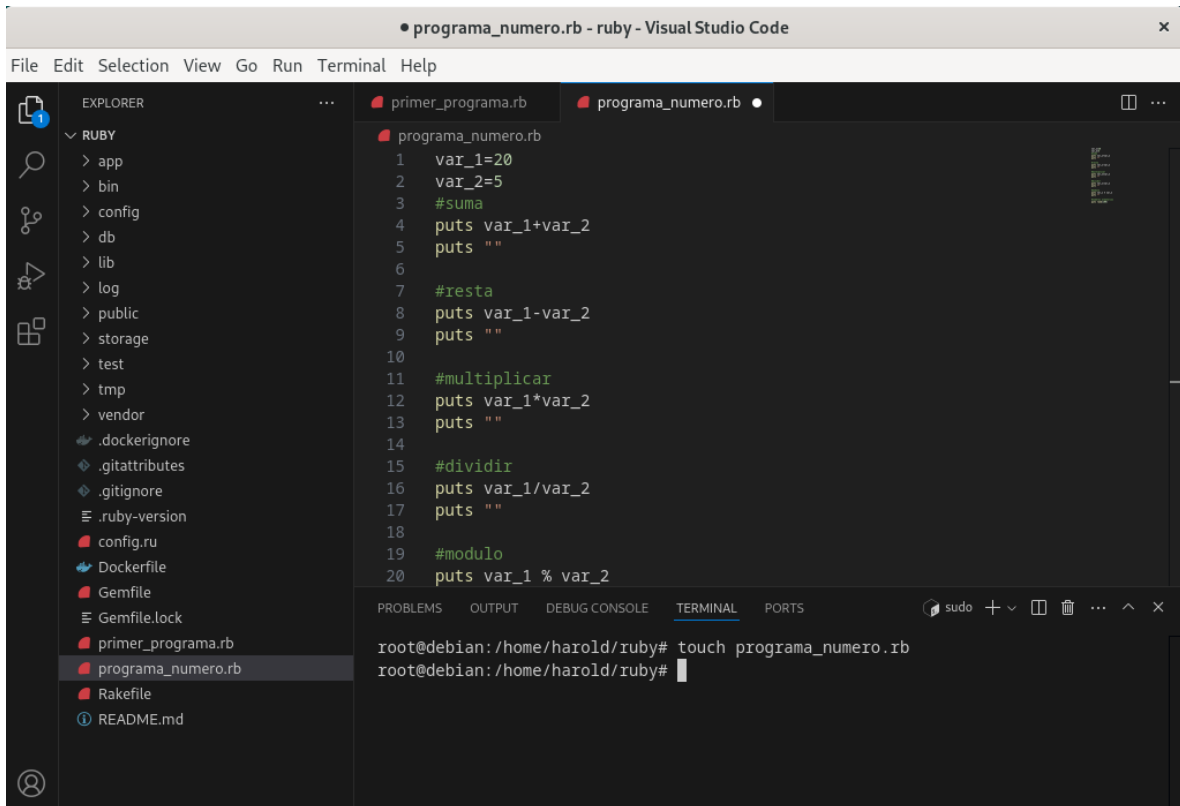


The screenshot shows a terminal window with the following commands and output:

```
root@debian:/home/harold/ruby# ruby primer_programa.rb
HolaHola
root@debian:/home/harold/ruby#
```

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.

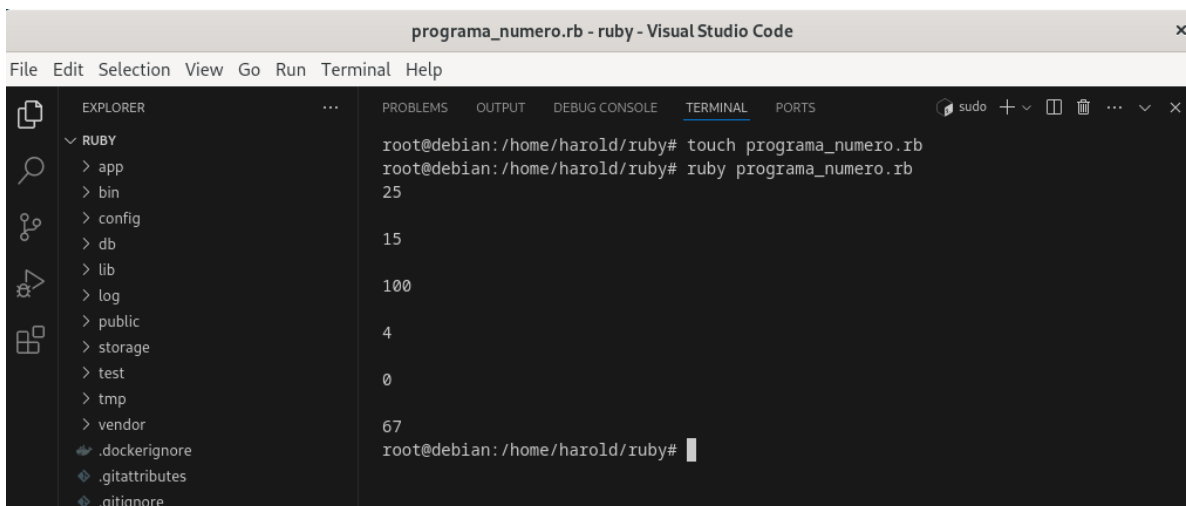


The screenshot shows the Visual Studio Code interface with the file explorer on the left and the editor in the center. The file explorer shows a project structure with a 'RUBY' folder containing 'programa_numero.rb'. The editor displays the code for 'programa_numero.rb' with the following content:

```
1 var_1=20
2 var_2=5
3 #suma
4 puts var_1+var_2
5 puts ""
6
7 #resta
8 puts var_1-var_2
9 puts ""
10
11 #multiplicar
12 puts var_1*var_2
13 puts ""
14
15 #dividir
16 puts var_1/var_2
17 puts ""
18
19 #modulo
20 puts var_1 % var_2
```

The terminal at the bottom shows the command 'touch programa_numero.rb' being executed, creating the file.

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

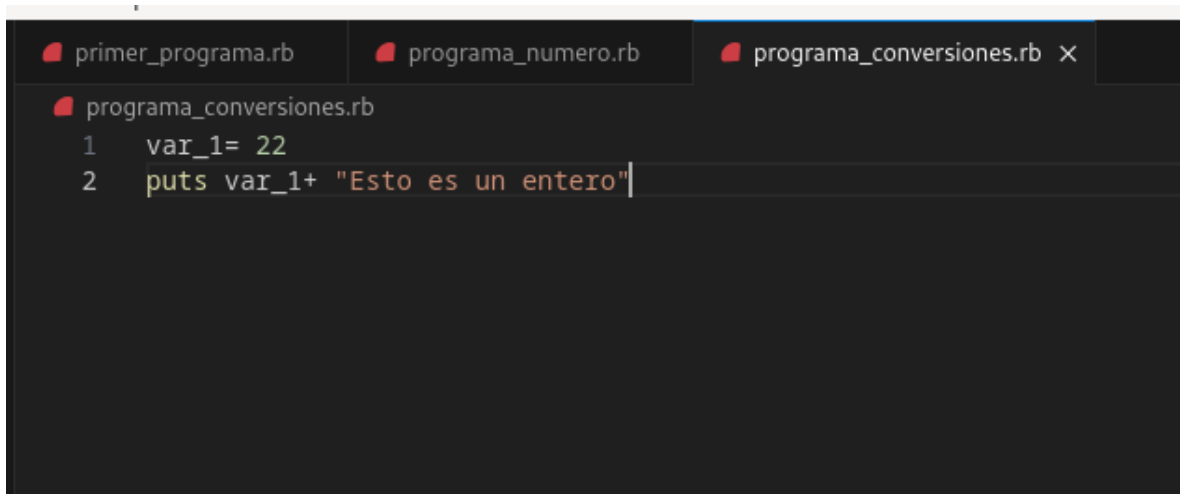


The screenshot shows the Visual Studio Code interface with the terminal at the bottom. The terminal displays the output of the program execution:

```
root@debian:/home/harold/ruby# touch programa_numero.rb
root@debian:/home/harold/ruby# ruby programa_numero.rb
25
15
100
4
0
67
root@debian:/home/harold/ruby#
```

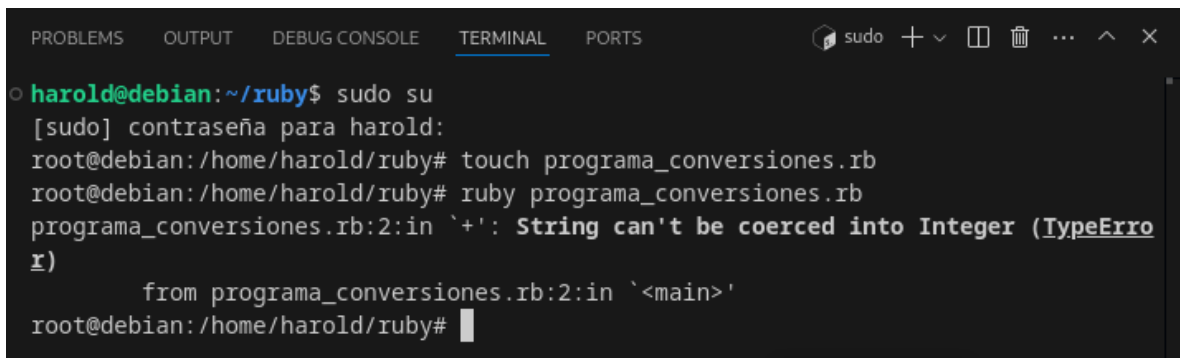
3.Conversiones

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.



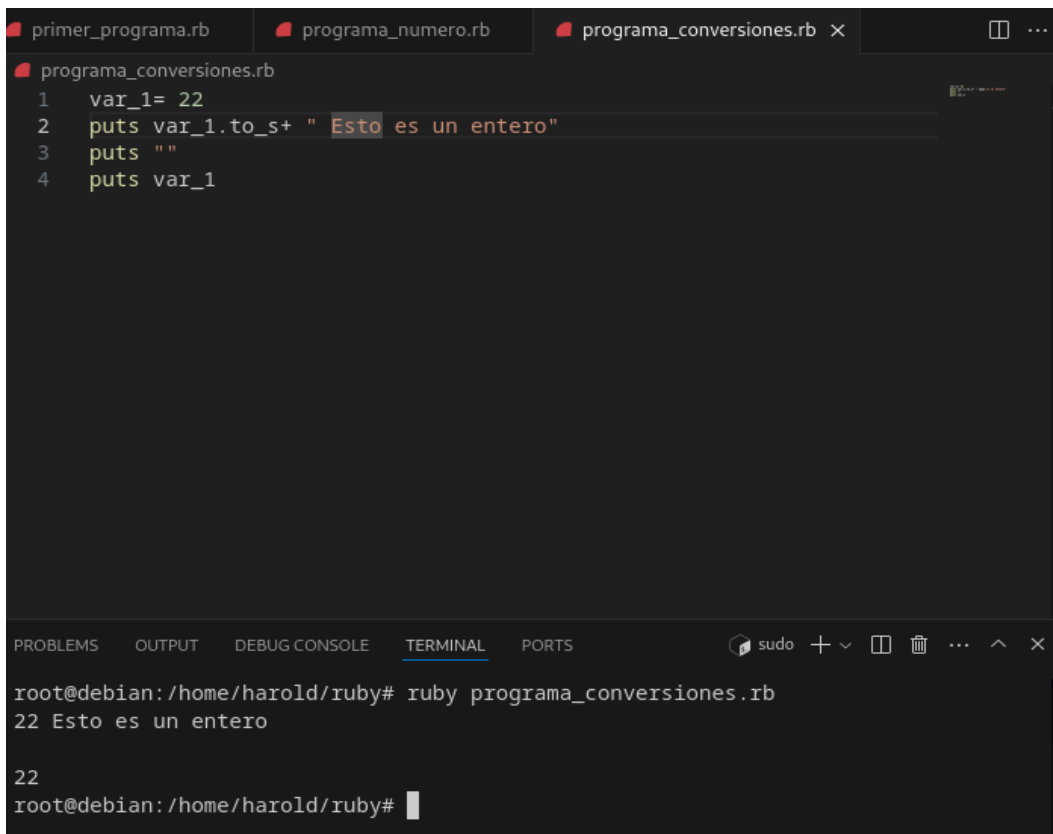
```
programa_conversiones.rb
1  var_1= 22
2  puts var_1+ "Esto es un entero"
```

3.2 Ejecutar el programa en el terminal



```
harold@debian:~/ruby$ sudo su
[sudo] contraseña para harold:
root@debian:/home/harold/ruby# touch programa_conversiones.rb
root@debian:/home/harold/ruby# ruby programa_conversiones.rb
programa_conversiones.rb:2:in `+': String can't be coerced into Integer (TypeError)
    from programa_conversiones.rb:2:in `<main>'
root@debian:/home/harold/ruby#
```

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



The screenshot shows a code editor with three tabs: `primer_programa.rb`, `programa_numero.rb`, and `programa_conversiones.rb`. The active tab is `programa_conversiones.rb`, which contains the following code:

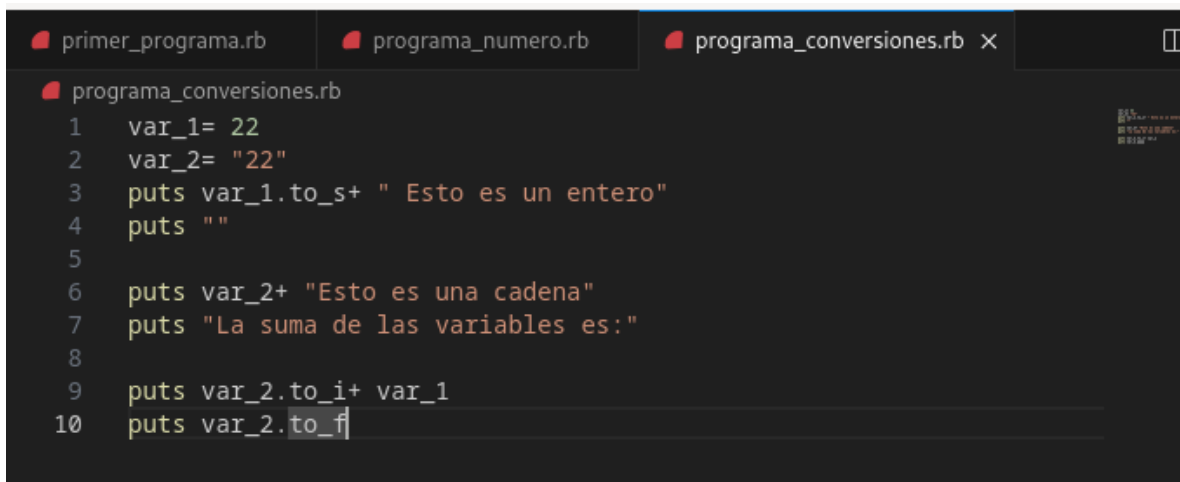
```
1 var_1= 22
2 puts var_1.to_s+ " Esto es un entero"
3 puts ""
4 puts var_1
```

Below the code editor is a terminal window with the following output:

```
root@debian:/home/harold/ruby# ruby programa_conversiones.rb
22 Esto es un entero

22
root@debian:/home/harold/ruby#
```

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.



The screenshot shows a code editor with three tabs: `primer_programa.rb`, `programa_numero.rb`, and `programa_conversiones.rb`. The active tab is `programa_conversiones.rb`, which contains the following code:

```
1 var_1= 22
2 var_2= "22"
3 puts var_1.to_s+ " Esto es un entero"
4 puts ""
5
6 puts var_2+ "Esto es una cadena"
7 puts "La suma de las variables es:"
8
9 puts var_2.to_i+ var_1
10 puts var_2.to_f
```

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

```
22
root@debian:/home/harold/ruby# ruby programa_conversiones.rb
22 Esto es un entero

22 Esto es una cadena
La suma de las variables es:
44
22.0
root@debian:/home/harold/ruby#
```

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.

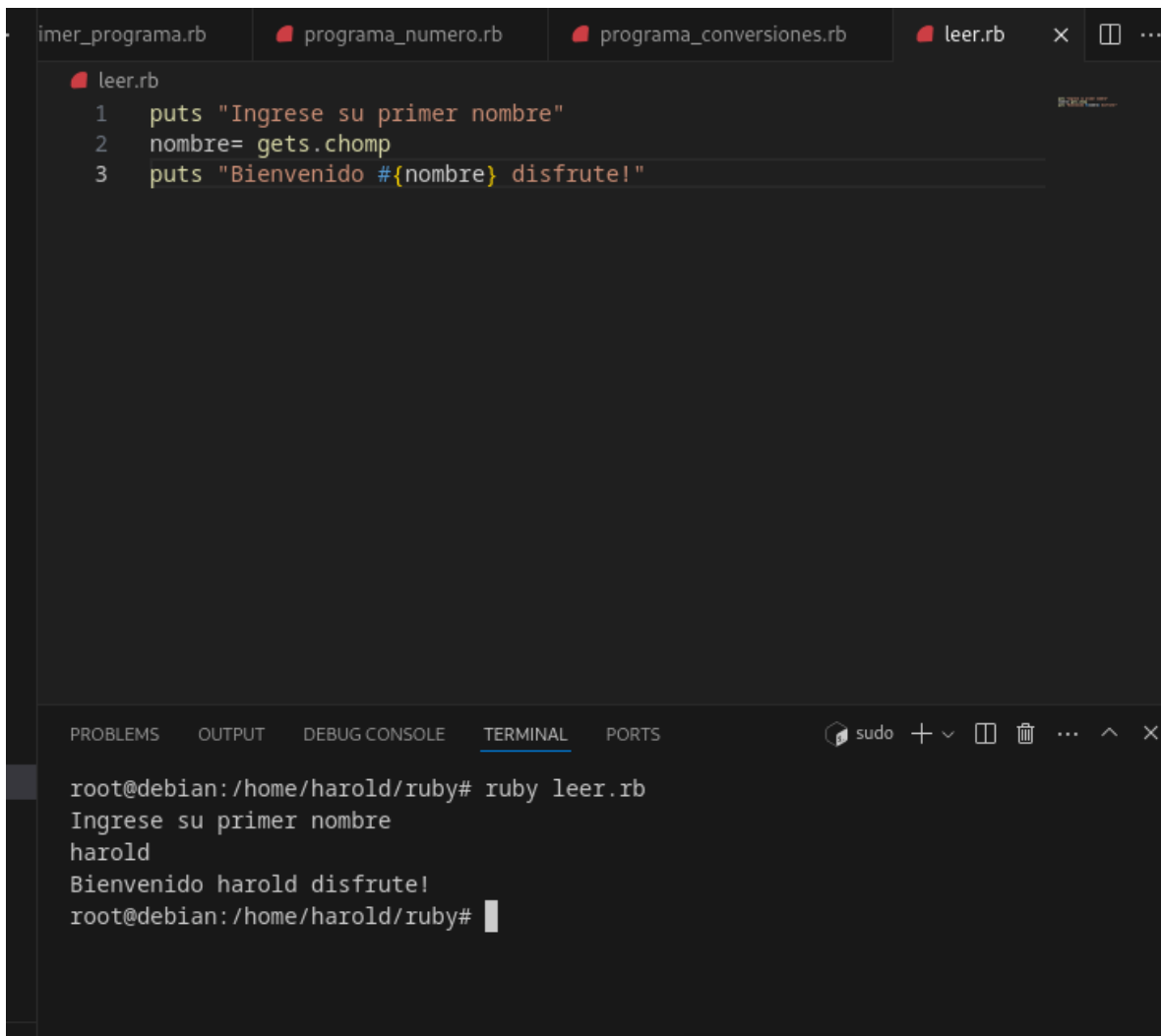
4.1 Crear un programa leer.rb y agregar el siguiente código.

```
leer.rb
1 puts "Ingrese su primer nombre"
2 nombre= gets
3 puts "Bienvenido "+ nombre+ " disfrute!"
```

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.

```
root@debian:/home/harold/ruby# touch leer.rb
root@debian:/home/harold/ruby# ruby leer.rb
Ingrese su primer nombre
harold
Bienvenido harold
disfrute!
root@debian:/home/harold/ruby#
```

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



The image shows a code editor with four tabs: `imer_programa.rb`, `programa_numero.rb`, `programa_conversiones.rb`, and `leer.rb`. The `leer.rb` tab is active, displaying the following Ruby code:

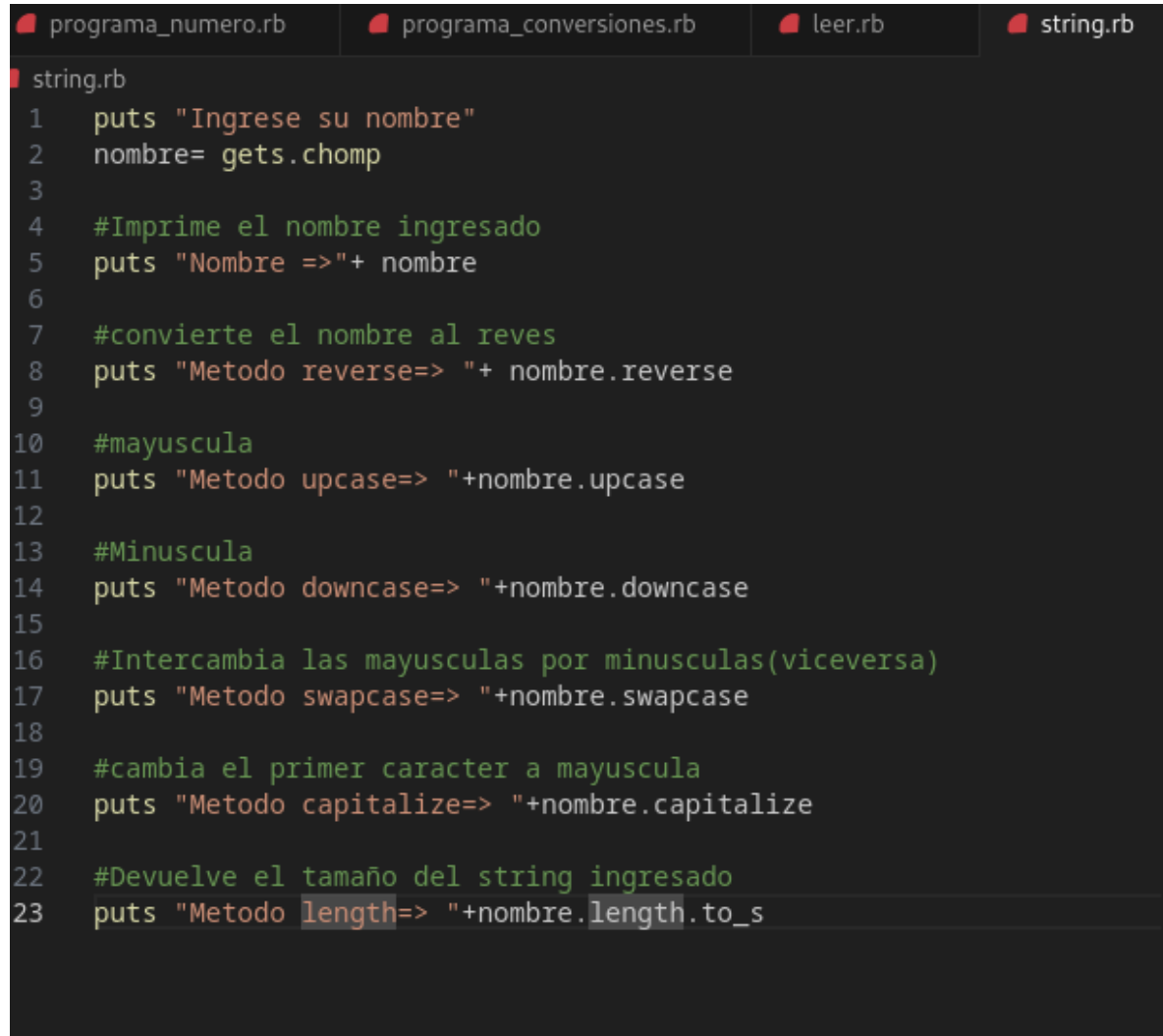
```
1 puts "Ingrese su primer nombre"
2 nombre= gets.chomp
3 puts "Bienvenido #{nombre} disfrute!"
```

Below the code editor is a terminal window with tabs for `PROBLEMS`, `OUTPUT`, `DEBUG CONSOLE`, `TERMINAL` (selected), and `PORTS`. The terminal shows the execution of the `leer.rb` program:

```
root@debian:/home/harold/ruby# ruby leer.rb
Ingrese su primer nombre
harold
Bienvenido harold disfrute!
root@debian:/home/harold/ruby#
```

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:

A screenshot of a code editor with a dark background. At the top, there are four tabs: 'programa_numero.rb', 'programa_conversiones.rb', 'leer.rb', and 'string.rb'. The 'string.rb' tab is active. The code in the editor is as follows:

```
1 puts "Ingrese su nombre"
2 nombre= gets.chomp
3
4 #Imprime el nombre ingresado
5 puts "Nombre =>" + nombre
6
7 #convierte el nombre al reves
8 puts "Metodo reverse=> " + nombre.reverse
9
10 #mayuscula
11 puts "Metodo upcase=> "+nombre.upcase
12
13 #Minuscula
14 puts "Metodo downcase=> "+nombre.downcase
15
16 #Intercambia las mayusculas por minusculas(viceversa)
17 puts "Metodo swapcase=> "+nombre.swapcase
18
19 #cambia el primer caracter a mayuscula
20 puts "Metodo capitalize=> "+nombre.capitalize
21
22 #Devuelve el tamaño del string ingresado
23 puts "Metodo length=> "+nombre.length.to_s
```


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

```
Ingrese su nombre
harold
Nombre =>harold
Metodo reverse=> dlorah
Metodo upcase=> HAROLD
Metodo downcase=> harold
Metodo swapcase=> HAROLD
Metodo capitalize=> Harold
Metodo length=> 6
root@debian: /home/harold/ruby#
```

Ln 23, Col

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.

```
nuevo.rb
1  iterador=""
2  while iterador.downcase != "s"
3      puts "Ingrese un nombre"
4      nombre= gets.chomp
5      tamaño= nombre.length
6
7      if (tamaño>=5)
8          puts "Su nombre tiene mas de 5 caracteres"
9      else
10         puts "Su nombre tiene menos de 5 caracteres"
11     end
12
13     puts "\nPara salir presione la letra S"
14
15     iterador= gets.chomp
16 end
17 puts "Ha salido del programa"
```

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

```
root@debian:/home/harold/ruby# touch nuevo.rb
root@debian:/home/harold/ruby# ruby nuevo.rb
Ingrese un nombre
sofia
Su nombre tiene mas de 5 caracteres

Para salir presione la letra S

Ingrese un nombre
ally
Su nombre tiene menos de 5 caracteres

Para salir presione la letra S

Ingrese un nombre
joysi
Su nombre tiene mas de 5 caracteres

Para salir presione la letra S
s
Ha salido del programa
root@debian:/home/harold/ruby#
```

Ejercicios propuestos.

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.

```

programa2.rb
1  def to_camel_case(text)
2
3      texto= text.split(/[_-]/,)
4      nuevo= texto.map!.with_index do |texto, num|
5          if num==0
6              texto.downcase
7          else
8              texto.capitalize!
9          end
10     end
11
12     return nuevo.join(' ')
13 end
14 puts "Ingrese un texto con guiones como separador"
15 texto= gets.chomp
16 puts to_camel_case(texto)
17 puts to_camel_case("Esta-es-otra-prueba-con_texto-previo")

```

```

programa2.rb
1  def to_camel_case(text)
2
3      texto= text.split(/[_-]/,)
4      nuevo= texto.map!.with_index do |texto, num|
5          if num==0
6              texto.downcase
7          else
8              texto.capitalize!
9          end
10     end
11
12     return nuevo.join(' ')
13 end
14 puts "Ingrese un texto con guiones como separador"
15 texto= gets.chomp
16 puts to_camel_case(texto)
17 puts to_camel_case("Esta-es-otra-prueba-con_texto-previo")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

root@debian:/home/harold/ruby# ruby programa2.rb
Ingrese un texto con guiones como separador
hola-a-todos
hola A Todos
esta Es Otra Prueba Con Texto Previo
root@debian:/home/harold/ruby#

```