

UNIVERSIDAD NACIONAL AUTONOMA DE NICARAGUA



UNAN - León

FACULTAD DE CIENCIAS Y TECNOLOGIA

Carrera: Ingeniería en Telemática

Componente: Software como un servicio

Grupo:1

Docente: Erving Montes

Elaborado por:

- Alli Gissiell Herrera Chow.

Fecha: 07 de agosto de 2024

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

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.

1.3. Ejecutar el programa en el terminal

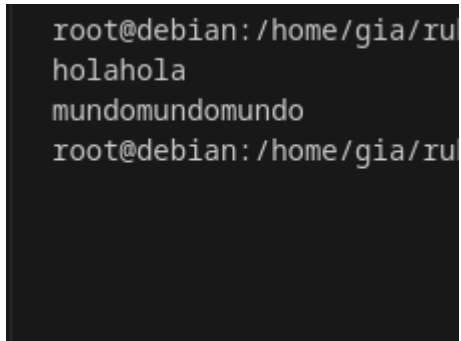
```
primer_programa.rb
1  var_1 = "hola"
2  var_2 = "mundo"
3  puts var_1 + var_2
4

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

gia@debian:~/ruby$ sudo su
[sudo] contraseña para gia:
Lo siento, pruebe otra vez.
[sudo] contraseña para gia:
Lo siento, pruebe otra vez.
[sudo] contraseña para gia:
root@debian:/home/gia/ruby# touch primer_programa.rb
root@debian:/home/gia/ruby# ruby primer_programa.rb
holamundo
root@debian:/home/gia/ruby#
```

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.

A terminal window with a black background and white text. The prompt is 'root@debian:/home/gia/ru'. The code being executed is 'holahola' followed by 'mundomundomundo'. The prompt is shown again at the bottom.

```
root@debian:/home/gia/ru
holahola
mundomundomundo
root@debian:/home/gia/ru
```

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.

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

```
primer_programa_numero.rb
9  puts var_1 - var_2
10 puts "
11
12 #multiplicar
13 puts var_1 * var_2
14 puts "
15
16 #dividir
17 puts var_1 / var_2
18 puts "
19
20 #modulo
21 puts var_1 % var_2
22 puts "
23
24 #numeros aleatorios

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

root@debian:/home/gia/ruby# ruby primer_programa_numero.rb
25

15

100

4

0

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

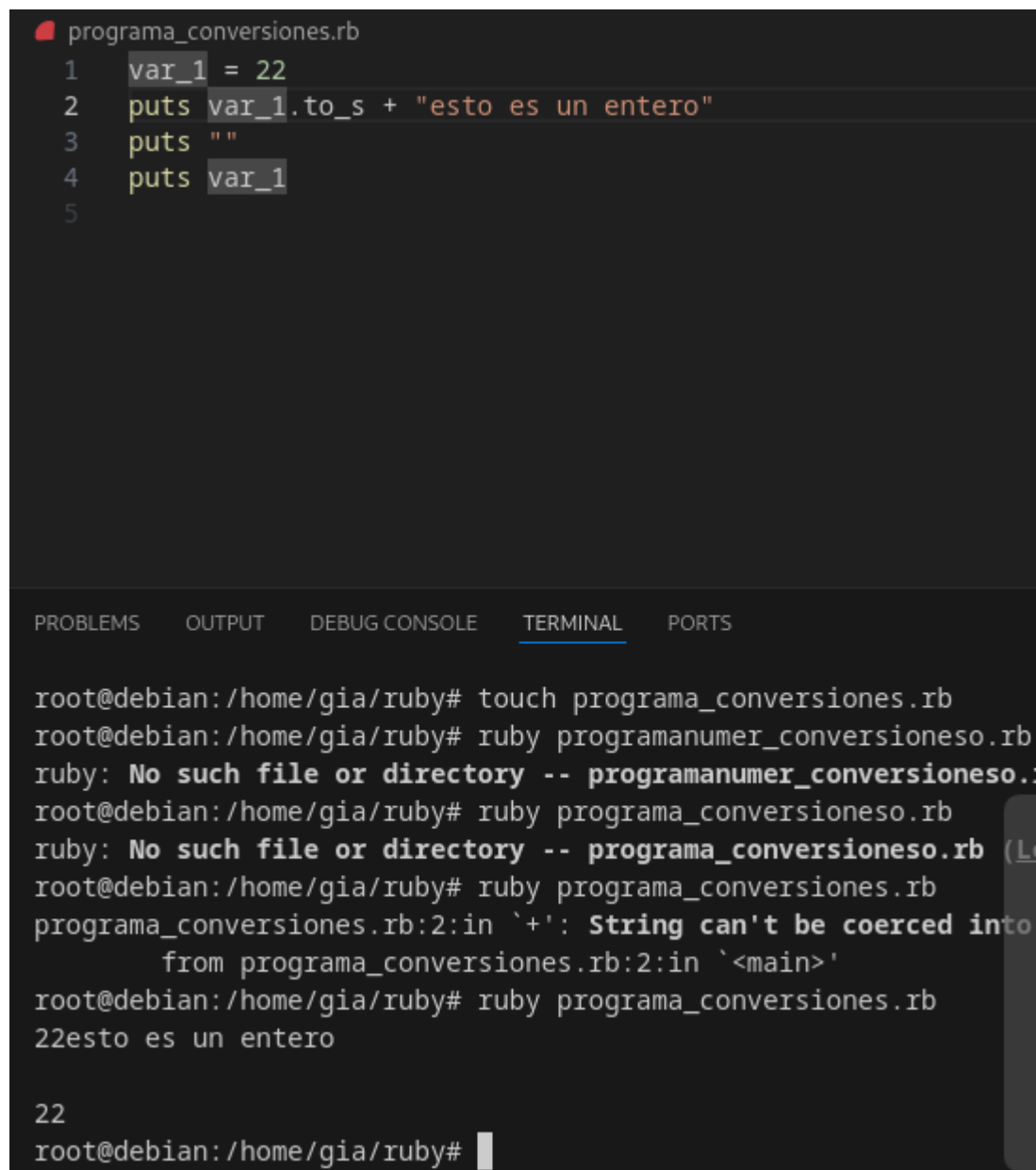
3.Conversiones

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.

3.2 Ejecutar el programa en el terminal.

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

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.



```
programa_conversiones.rb
1  var_1 = 22
2  puts var_1.to_s + "esto es un entero"
3  puts ""
4  puts var_1
5

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

root@debian:/home/gia/ruby# touch programa_conversiones.rb
root@debian:/home/gia/ruby# ruby programanumer_conversioneso.rb
ruby: No such file or directory -- programanumer_conversioneso.
root@debian:/home/gia/ruby# ruby programa_conversioneso.rb
ruby: No such file or directory -- programa_conversioneso.rb (L
root@debian:/home/gia/ruby# ruby programa_conversiones.rb
programa_conversiones.rb:2:in `+': String can't be coerced into
      from programa_conversiones.rb:2:in `'
root@debian:/home/gia/ruby# ruby programa_conversiones.rb
22esto es un entero

22
root@debian:/home/gia/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.

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

```
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_1
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

22Esto es una cadena
La suma de las variables es:
44
22
root@debian:/home/gia/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"

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

root@debian:/home/gia/ruby# ruby leer.rb
root@debian:/home/gia/ruby# ruby leer.rb
ingrese su primer nombre
AGUA de glly
BienvenidoAGUA de glly
disfrute
root@debian:/home/gia/ruby#
```

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

```
leer.rb
1 puts "ingrese su primer nombre"
2 nombre = gets.chomp
3 puts "Bienvenido#{nombre}disfrute"

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

root@debian:/home/gia/ruby# ruby leer.rb
ingrese su primer nombre
agyu gia
Bienvenidoagyu giadisfrute
root@debian:/home/gia/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

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


```
string.rb
2  nombre = gets.chomp
11
12  #miniscula
13  puts "metodo downcase=>" + nombre.downcase
14
15  #intercambia las minusscula por mayuscula(vicerversa)
16  puts "metodo swapcase=>" + nombre.swapcase
17
18  #cambia el primer caracter a mayuscula
19  puts "metodo capitalize=>" + nombre.capitalize
20
21  #devuelve el tamaño del string ingresado
22  puts "metodo length=>" + nombre.length.to_s
23

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

root@debian:/home/gia/ruby# touch string.rb
root@debian:/home/gia/ruby# ruby string.rb
ingrese su nombre
agua de mar
nombre=>agua de mar
metodo reverse=> ram ed auga
metodo upcase=>AGUA DE MAR
metodo downcase=>agua de mar
metodo swapcase=>AGUA DE MAR
metodo capitalize=>Agua de mar
metodo length=>12
root@debian:/home/gia/ruby#
```

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.

```
new.rb
1  iterador=""
2  while iterador.downcase!="s"
3
4      puts "ingrese un nombre"
5      nombre = gets.chomp
6      tamaño = nombre.length
7
8      if(tamaño>=5)
9          puts "su nombre tiene mas de 5 caracteres"
10
11      else
12          puts "su nnombre tiene menos de 5 caracteres"
13      end
14  end
15

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

herador
root@debian:/home/gia/ruby# ruby new.rb
ingrese un nombre
agua de hielo
su nombre tiene mas de 5 caracteres
/npara salir presione la letra s
s
has salido del programa
root@debian:/home/gia/ruby#
```

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.

eje2.rb

```
1  def to_upper_camel_case(text)
2
3      palabra = text.gsub(/[_-]/, ' ').split
4      palabra.map!.with_index do |palabra, index|
5
6          if index == 0
7              palabra.capitalize
8
9          else
10             palabra.upcase
11         end
12     end
13
14     palabra.join
15 end
16
17 puts to_upper_camel_case("hello_word")
18 puts to_upper_camel_case("the-stealth-warrior")
19 puts to_upper_camel_case ("The_stealth_warrior")
20
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
root@debian:/home/gia/ruby# ruby eje2.rb
Helloworld
Thestealthwarrior
Thestealthwarrior
root@debian:/home/gia/ruby#
```

