

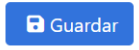
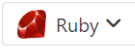
```
1 var_1 = "hola"  
2 var_2 = "mundo"  
3 puts var_1 + var_2  
4
```

Entrada del programa

### Salida del programa

holamundo

[Execution complete with exit code 0]



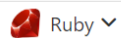
```
1 var_1 = "hola"  
2 var_2 = "mundo"  
3 puts var_1 * 2
```

Entrada del programa

### Salida del programa

holahola

[Execution complete with exit code 0]



Ruby ▾



Ejecute



Guardar

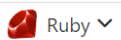
```
1 var_1 = 20
2 var_2 = 5
3 #suma
4 puts var_1 + var_2
5 puts ""
6 #resta
7 puts var_1 - var_2
8 puts ""
9 #multiplicar
10 puts var_1 * var_2
11 puts ""
12 #dividir
13 puts var_1 / var_2
14 puts ""
15 #modulo
16 puts var_1 % var_2
17 puts ""
18 #números aleatorios
19 puts rand(100)
20
```

Entrada del programa

**Salida del programa**

```
25
15
100
4
0
1
```

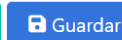
[Execution complete with exit code 0]



Ruby ▾



Ejecute



Guardar



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

Entrada del programa

**Salida del programa**


```
main.rb:2:in `+': String can't be coerced into Integer (TypeError)
    from main.rb:2:in `<main>'
```

[Execution complete with exit code 1]

 Ruby ▾ 

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

 Ejecute

 Guardar

Entrada del programa

#### Salida del programa

22 Esto es un entero


22

[Execution complete with exit code 0]

 Ruby ▾ 

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

 Ejecute

 Guardar


Entrada del programa

#### Salida del programa

22 Esto es un entero


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

[Execution complete with exit code 0]

 Ruby ▾ 

```
1 puts "Ingrese su primer nombre"
2 nombre = gets
3 puts "Bienvenido "+ nombre + "disfrute! "
4
```

 Ejecute

 Guardar

Josh

#### Salida del programa

Ingrese su primer nombre  
Bienvenido Joshdisfrute!

[Execution complete with exit code 0]



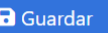
Ruby ▾



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



Ejecute



Guardar

Josh

### Salida del programa

```
Ingrese su primer nombre
Bienvenido Josh disfrute!
```

[Execution complete with exit code 0]



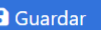
Ruby ▾



```
1 puts "Ingrese su nombre"
2 nombre = gets.chomp
3 #Imprime el nombre ingresado
4 puts "Nombre => " + nombre
5 # convierte al revés el nombre
6 puts "Método reverse => " + nombre.reverse
7 #Mayúscula
8 puts "Método upcase => " + nombre.upcase
9 #Minúscula
10 puts "Método downcase => " + nombre.downcase
11 #Intercambia las minúsculas por mayúscula (viceversa)
12 puts "Método swapcase => " + nombre.swapcase
13 #cambia el primer carácter a mayúscula
14 puts "Método capitalize => " + nombre.capitalize
15 #devuelve el tamaño del string ingresado
16 puts "Método length => " + nombre.length.to_s
```



Ejecute




Guardar

Josh


### Salida del programa

```
Ingrese su nombre
Nombre => Josh
Método reverse => hsoJ
Método upcase => JOSH
Método downcase => josh
Método swapcase => jOSH
Método capitalize => Josh
Método length => 4
```

[Execution complete with exit code 0]

Ruby 

 Ejecute

 Guardar

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

Josh


### Salida del programa

```
main.rb:12:in `<main>': undefined method `chomp' for nil:NilClass
Ingrese un nombre
Su nombre tiene menos de 5 caracteres
```


Para salir presione la letra S

[Execution complete with exit code 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.

Ruby 

 Ejecute

 Guardar

```
1 def convert_to_camel_case(s)
2   words = s.split(/_|-/ ) # Dividir la cadena en palabras por guiones bajos o guiones
3   # Convertir la primera palabra a mayúscula si la original estaba en mayúscula
4   if words[0] =~ /^[A-Z]/
5     words[0] = words[0].capitalize
6   else
7     words[0] = words[0].downcase
8   end
9   # Convertir las palabras restantes a mayúscula
10  words = words.map(&:capitalize)[1..-1]
11  # Unir las palabras en una sola cadena
12  camel_case_string = words.join("")
13  camel_case_string
14 end
15 # Ejemplos de uso:
16 puts convert_to_camel_case("hello_world") # Salida: helloWorld
17 puts convert_to_camel_case("HELLO_WORLD") # Salida: HelloWorld
18 puts convert_to_camel_case("my-python-example") # Salida: myPythonExample
19 puts convert_to_camel_case("MY-PYTHON-EXAMPLE") # Salida: MyPythonExample
```

Entrada del programa

### Salida del programa

```
World
World
PythonExample
PythonExample
```

[Execution complete with exit code 0]

