{desafío} latam_

Simplificando el flujo

Objetivos

- Simplificar el código de un script en Ruby utilizando buenas prácticas y la regla de condición en positivo
- Conocer sintaxis de if ternario en Ruby
- Refactorizar flujo condicional en Ruby

Variantes de IF

If en una línea (inline)

En Ruby, también es posible utilizar versiones cortas de la instrucción `if` y `unless` de la siguiente forma: `action if condition`. A este tipo de expresiones se le conoce como *if inline* por estar en la misma línea.

Analicemos los siguientes ejemplos:

```
puts 'Ingresa tu edad: '
edad = gets.to_i

puts 'Eres mayor de edad!' if edad >= 18
```

```
puts 'Ingresa tu nombre: '
nombre = gets.chomp

puts "Hola! #{nombre}!" if nombre != ''
```

Esta forma de la instrucción `if` es más limitada ya que no tenemos manejo del flujo cuando no se cumple la condición (`elsif` y `else`), sin embargo, es una muy buena solución cuando nos enfrentamos a evaluaciones sencillas como las anteriores.

Operador ternario

El operador ternario es una variante de `if` que permite operar en base a dos caminos en condiciones simples.

La lógica es la siguiente:

`si_es_verdadero ? entonces_esto : sino_esto`

```
edad = 18

puts edad >= 18 ? "Mayor de edad" : "Menor de edad"
```

Lo anterior se lee: Si la edad es mayor o igual a 18, imprime "Mayor de edad"; sino, imprime "Menor de edad".

Refactorizar

Cuando comenzamos a operar con condicionales es común que caigamos en redundancias innecesarias. Analicemos el siguiente ejemplo:

```
mayor_de_edad = true
zurdo = false

if mayor_de_edad == true
  if zurdo == true
    puts "Mayor de edad y zurdo!"
  else
    puts "Mayor de edad pero no zurdo!"
  end
else
  if zurdo == true
    puts "Menor de edad y zurdo!"
  else
    puts "Menor de edad pero no zurdo!"
  end
end
```

Reemplacemos los `if` anidados por condiciones múltiples:

```
if mayor_de_edad == true && zurdo == true
  puts "Mayor de edad y zurdo!"
elsif mayor_de_edad == true && zurdo == false
  puts "Mayor de edad pero no zurdo!"
elsif mayor_de_edad == false && zurdo == true
  puts "Menor de edad y zurdo!"
else
  puts "Menor de edad y no zurdo!"
end
```

Ahora podemos refactorizar las comparaciones en los condicionales que son inncesarias.

¿Cómo?

Recordemos que la instrucción `if` espera que el resultado se evalúe como `true` o `false`. Por lo tanto:

```
mayor_de_edad = true

if mayor_de_edad == true
  puts "Mayor de edad"
end
```

Es lo mismo que:

```
mayor_de_edad = true

if mayor_de_edad
  puts "Mayor de edad"
end
```

La comparación `mayor_de_edad == true` es redundante ya que la variable por sí sola se puede evaluar como `true` o `false`. Apliquemos esto en el ejemplo:

```
mayor_de_edad = true
zurdo = false

if mayor_de_edad && zurdo
  puts "Mayor de edad y zurdo!"
elsif mayor_de_edad && zurdo == false
  puts "Mayor de edad pero no zurdo!"
elsif mayor_de_edad == false && zurdo
  puts "Menor de edad y zurdo!"
else
  puts "Menor de edad y no zurdo!"
end
```

Podemos también refactorizar la comparación para evaluar que una variable booleana sea falsa, simplemente negando la condición:

```
mayor_de_edad = true
zurdo = false

if mayor_de_edad && zurdo
  puts "Mayor de edad y zurdo!"
elsif mayor_de_edad && !zurdo
  puts "Mayor de edad pero no zurdo!"
elsif !mayor_de_edad && zurdo
  puts "Menor de edad y zurdo!"
else
  puts "Menor de edad y no zurdo!"
end
```