**HOMEWORK 3**

* **Para qué se utilizan las instrucciones break y next, y dentro de qué**

**¿declaraciones?**

**R//** Las instrucciones "break" y "next" se utilizan en programación para controlar el flujo de ejecución en bucles y condicionales.

1. Bucles "for" y "while": Puedes utilizar "break" y "next" dentro de un bucle "for" o "while" para controlar la iteración y finalización del bucle.

Ejemplo:

for elemento in lista do

if condicion(elemento)

break

end

# Más código aquí

end

while condicion do

if condicion(elemento)

break

end

# Más código aquí

end

1. Bucles "each" y similares: Ruby proporciona métodos como "each" para iterar sobre colecciones. Puedes utilizar "break" y "next" dentro de estos bucles para controlar el flujo.

Ejemplo:

lista.each do |elemento|

if condicion(elemento)

break

end

# Más código aquí

end

1. Bloques de código en condicionales: Puedes utilizar "break" y "next" dentro de bloques de código dentro de condicionales para alterar el flujo de ejecución.

Ejemplo:

if condicion

# Código aquí

break

else

# Código aquí

next

end

En resumen, tanto "break" como "next" se utilizan en Ruby para controlar el flujo de ejecución dentro de bucles y condicionales, permitiendo salir prematuramente de un bucle con "break" o saltar a la siguiente iteración con "next".

* **¿Para qué son estos iteradores? Escribe un ejemplo sencillo de cada uno de ellos:**

**a. Cada iterador**

**b. iterador de tiempos**

**C. recopilar iterador**

**R//** los iteradores son métodos que permiten recorrer y manipular colecciones de datos de manera sencilla y concisa.

1. Iterador "each": El iterador "each" se utiliza para recorrer cada elemento de una colección y ejecutar un bloque de código para cada elemento.

Ejemplo:

numeros = [1, 2, 3, 4, 5]

numeros.each do |numero|

puts numero \* 2

end

#Este ejemplo multiplica cada número de la lista por 2 e imprime el resultado.

1. Iterador "times": El iterador "times" se utiliza para ejecutar un bloque de código un número específico de veces.

Ejemplo:

5.times do |i|

puts "Iteración número #{i+1}"

end

#En este ejemplo, el bloque de código se ejecutará 5 veces, mostrando el número de iteración.

1. Iterador "collect": El iterador "collect" (también conocido como "map") se utiliza para crear una nueva colección a partir de otra aplicando una transformación a cada elemento.

Ejemplo:

numeros = [1, 2, 3, 4, 5]

duplicados = numeros.collect do |numero|

numero \* 2

end

puts duplicados.inspect

# En este ejemplo, se crea una nueva lista llamada "duplicados" que contiene los números originales multiplicados por 2.