la recursividad es una técnica de programación donde una función se llama a sí misma para resolver un problema

**Componentes de la Recursividad**

**caso base**: la condición en la que la función deja de llamarse a sí misma para evitar un bucle infinito

**llamada recursiva**: la parte de la función donde se llama a sí misma con un subproblema más pequeño

**Ejemplo: Factorial**

el factorial de un número n se define como el producto de todos los números positivos enteros hasta n, se puede definir de manera recursiva asi:

**caso base**: 0! = 1

**llamada recursiva**: n! = n \* (n-1)!

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n - 1)

**Trazando la Recursividad**

es buena idea entender cómo se desarrolla una llamada recursiva paso a paso, por ejemplo, para factorial de 3:

factorial(3) llama a factorial(2)

factorial(2) llama a factorial(1)

factorial(1) llama a factorial(0)

factorial(0) devuelve 1

factorial(1) devuelve 1 \* 1 = 1

factorial(2) devuelve 2 \* 1 = 2

factorial(3) devuelve 3 \* 2 = 6

**Ejemplo: Serie de Fibonacci**

la serie de fibonacci se define de manera recursiva asi:

**caso base**: fib(0) = 0, fib(1) = 1

**llamada recursiva**: fib(n) = fib(n-1) + fib(n-2)

def fibonacci(n):

if n <= 1:

return n

else:

return fibonacci(n-1) + fibonacci(n-2)

**Peligros de la Recursividad**

**exceso de llamadas**: las llamadas recursivas pueden crecer exponencialmente, como en el caso de fibonacci.

**desbordamiento de pila**: si una función recursiva se llama demasiadas veces, puede llevar a un desbordamiento de pila lo que causa que el programa falle

**Recursividad de Cola**

la recursividad de cola ocurre cuando la llamada recursiva es la última operación que realiza una función, es más eficiente porque de esta forma el compilador puede optimizar el uso de la pila

def factorial\_tail(n, acumulador=1):

if n == 0:

return acumulador

else:

return factorial\_tail(n-1, n\*acumulador)

**Ejemplo extra**

**# Factorial**

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n - 1)

**# Fibonacci**

def fibonacci(n):

if n <= 1:

return n

else:

return fibonacci(n-1) + fibonacci(n-2)

**# Recursividad de Cola**

def factorial\_tail(n, acumulador=1):

if n == 0:

return acumulador

else:

return factorial\_tail(n-1, n\*acumulador)

print(factorial(5)) # Salida: 120

print(fibonacci(7)) # Salida: 13

print(factorial\_tail(5)) # Salida: 120