**Tail Recursion**

**1. Definition**

Tail recursion is a special type of recursion where the recursive call is the last operation in the function. The key characteristic is that no additional computation is needed after the recursive call returns.

**2. Example Implementation: Factorial with Tail Recursion**

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

*# Base case*

if n <= 1:

return accumulator

*# Tail recursive call - no computation after this call*

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

*# Usage*

result = factorial\_tail(5) *# 120*

**3. Stack Visualizatin for Tail Recursion**

factorial\_tail(5, 1)

└── factorial\_tail(4, 5)

└── factorial\_tail(3, 20)

└── factorial\_tail(2, 60)

└── factorial\_tail(1, 120)

Returns: 120

**4. Advantages of Tail Recursion**

* Can be optimized by compilers into iteration
* No stack frame needs to be maintained
* More memory efficient than regular recursion