Recursion is made for trees and Divide and Conquer type of problem statements

Recursion

↳ function call itself directly or indirectly

factorial

↳ $5! = 5 * 4 * 3 * 2 * 1 = 120$

In recursion based problem statements Base case condition will be always present that represents the condition where code is meant to e terminated

Base case condition

↳ code will terminate

$T(n)$ — C

```
fact(n):
    if n==0 or n==1:
        return 1
    else:
        result = n * fact(n-1)
        return result
```

Small problem

$n = 0$ or $n = 1$

Since factorial is 0 and 1 is equals to 1

↳ 1

Base case condition

Recursion $n > 1$

↳ Recursive call

$n * T(n-1)$

fact(5) → 120

$n = 5$

$5 * fact(4)$ → 24

$4 * fact(3)$ → 6

$3 * fact(2)$ → 2

$2 * fact(1)$ → 1

$1! = 1$

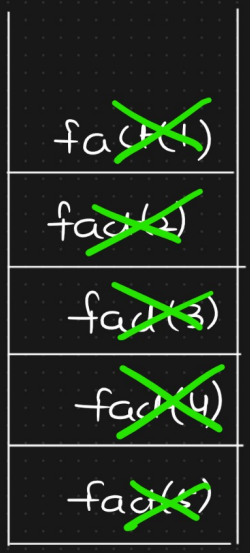
Code will stop

In recursion, recursive function calls are stored in the form of Stack where last recursive call is returned first as soon as base condition(termination state) is meet

Stack → LIFO
(Last In first Out)

⇓

to store function call



→ Empty stack

Recurrence Relation
(factorial of number)

$$T(n) = \begin{cases} 1 & n = 1 \\ n * T(n-1) & n > 1 \end{cases}$$

$$T(n) = n * T(n-1)$$

↳ Substitution method