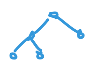
RECURSION 1

Agenda :

→ Recursion (Basics)
→ steps
→ problems (fact, fibo, power, sum)

Why recursion ?

→ Decision Trees 
→ Dynamic programming
→ Merge sort / Quick sort
→ Graphs
→ Backtracking

Disclaimer :

i) It might take some time to digest ~~this~~ ~~topic~~.
ii) A lot of dry runs are required.

☆ A function is calling inside itself.

Observations :

i) Smaller dolls (sub problem)
ii) Similar dolls (same problem)
iii) Tinniest doll (end case stop / base condition / stop condn)

Recursion :

i) Solving problem using smaller instance of same problem.

ii) $f(n) \rightarrow f(n-1) \rightarrow f(n-2) \dots$ base condn

Problem : Find sum of first N-natural numbers

- Iteration
- Recursion

$\Rightarrow$ $sum(5) \Rightarrow \underbrace{1 + 2 + 3 + 4}_{sum(4)} + 5$

$sum(4) + 5$

★ Steps of recursion :

i) Assumption : Decide what you want from function.

ii) Main logic : Solving assumption using smaller instance.

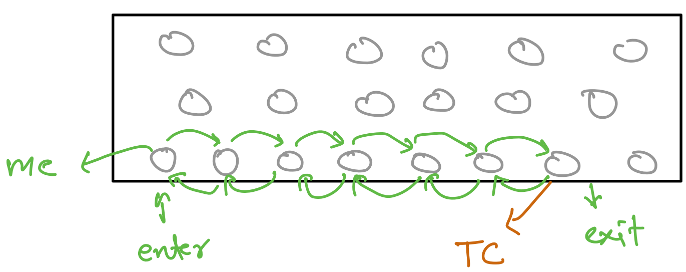
15

$$
\begin{aligned}
\text{Sum}(5) &= 5 + \text{Sum}(4) \quad 10\\
\text{Sum}(4) &= 4 + \text{Sum}(3) \quad 6\\
\text{Sum}(3) &= 3 + \text{Sum}(2) \quad 3\\
\text{Sum}(2) &= 2 + \text{Sum}(1)\\
\underline{\text{Sum}}(1) &= 1
\end{aligned}
$$

↳ Base condition

iii) Base condition : Once this base condition is hit we need not go further into recursive calls.

★ DTC :



↳ Base condition.

★ Sum of first N- natural numbers :

Any   add (n) => n + add (n-1)

Recursive tree :

15

add (5)   5 + add (4)

10   add (4)   4 + add (3)

6   add (3)   3 + add (2)

3   add (2)   2 + add (1)

1   add (1)   Base Condition

(1)

★ $\boxed{add(5) = n + add(n-1)}$

↳ Recurrence relation

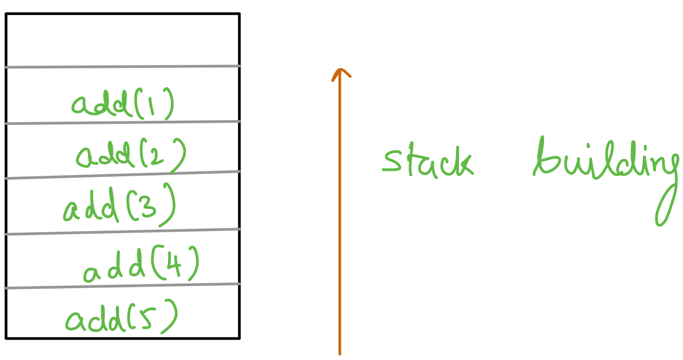
#

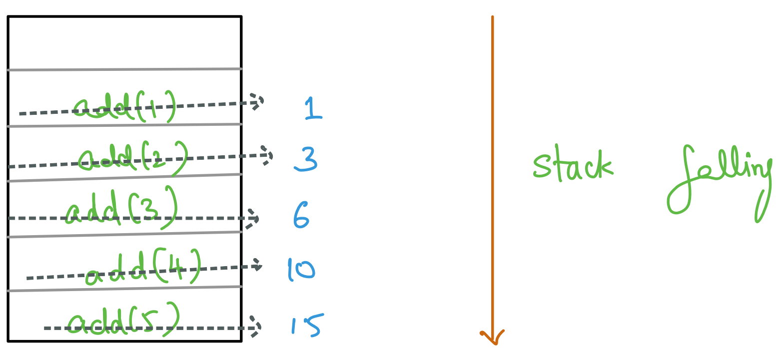
```
def add(5):
    # Base condition
```

```
if n == 1:
    return 1

return n + add(n-1)
```

★ Memory stack :





★ **Factorial**

\# $fact(0) = 1$

\# $fact(1) = 1$

\# $fact(3) = 3 \times 2 \times 1 = 6$

\# $fact(5) = 5 \times 4 \times 3 \times 2 \times 1 = 120$

\# $fact(5) = 5 \times fact(4)$

$$\boxed{\#\quad fact(n) = n \times fact(n-1)}$$

↳ Recurrence relation

120

\# $fact(5) = 5 \times fact(4)$

24

$fact(4) = 4 \times fact(3)$

6

$fact(3) = 3 \times fact(2)$

2

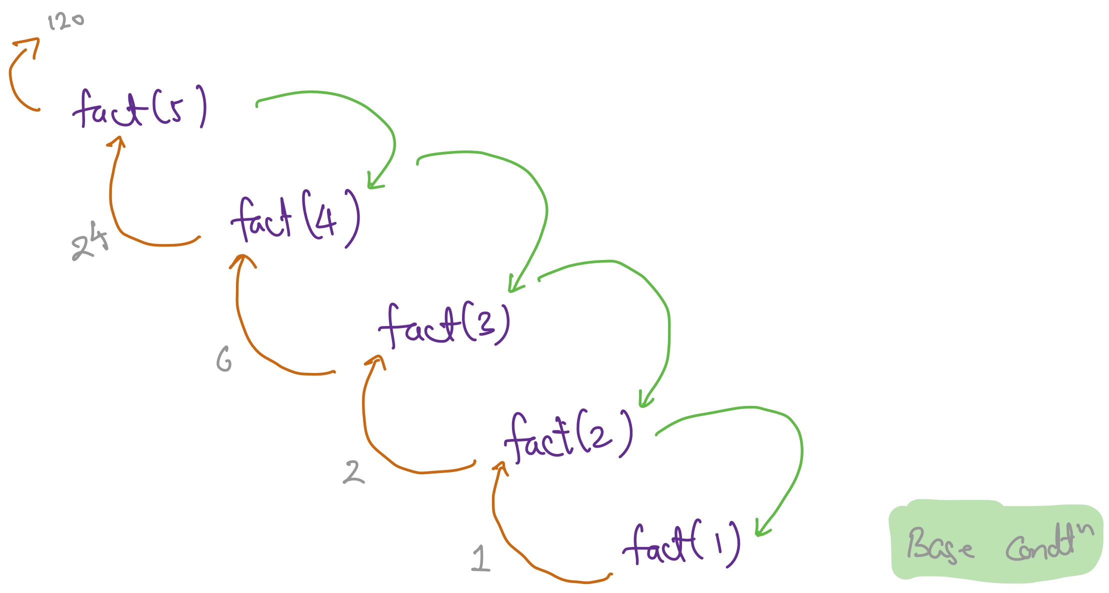
$fact(2) = 2 \times fact(1)$

1

$fact(1) = 1$

↳ Base condition

☆ Recursive Tree



HW : Draw memory stack

```
def factorial(n) :
        # Base condition
    if  n == 1  or  n == 0 :
            return  1
    return  n x factorial(n-1)
```

☆ Fibonacci Series :

N =	0	1	2	3	4	5	6	7
fib (N) =	0	1	1	2	3	5	8	13

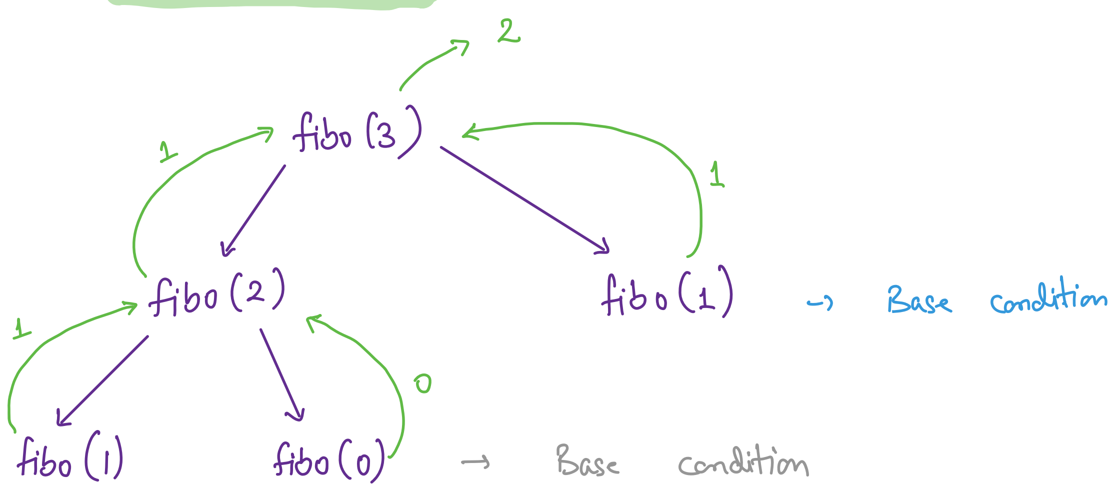
★ formula :

$$fibo(n) = fibo(n-1) + fibo(n-2)$$

☆ Base Condition :

if n == 0 or n == 1 :
return n

☆ Recursive Tree :



★ **Power**

power (a , n)

power (2, 3)

power (2,3) = 2 * power (2,2)
power (2,2) = 2 * power (2,1)
power (2,1) = 2 * power (2,0)
power (2,0) = 1
↳ Base condition

★ Recurrence relation :

power (a, n) = a * power (a , n-1)

★ power (2,3)

power (2,3) = 2 × 2^2 → power (2,2)
power (2,2) = 2 × 2^1 → power (2,1)
power (2,1) = 2 × 2^0 → power (2,0)
↳ 1

Ans

power (2,1) = 2
power (2,2) = 2 × 2 = 4
power (2,3) = 2 × 4 = 8