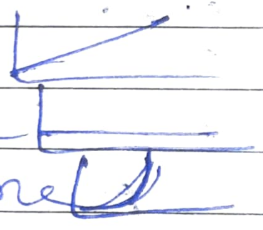
* Recursion

- It is used when the problem can be broken down into smaller, repetitive tasks.

- Advantages :-
  - complex tasks broken into simpler problem
  - code using it is very shorter
  - sequence generation is cleaner with recursion than with iterations
- fibonaci and factorial

* Big O notation :-

time compexity :- lear time
constant time
quadratic time



step 1 :- fastest growing term
step 2 :- take out the coefficient.

$T = an + b = O(n)$
$T = cn^2 + dn + e = O(n^2)$
$T = c = O(1)$.

~~to~~ given array = [1, 4, 3, 2, —— 10]

```
def ~~f~~ stupid_function (given_array) :-
    total = 0 → O(1)
    return total → O(1)
```

$\boxed{T = O(1)}$

DATE: / / PAGE NO:

```
→ def find_sum (given_array):
      total = 0 → O(1)
    for each i in given_array:
        total += i → O(1)
      return total → O(1) x n (because of for
                                loop run n time.)

→ def find_sum (array_2d):
      total = 0 → O(1)
    for each row in array_2d:
        for each i in row:
            total += i → O(1)
      return total → O(1) x n^2
                       (because of 2 for loop
                        run n x n time).
```