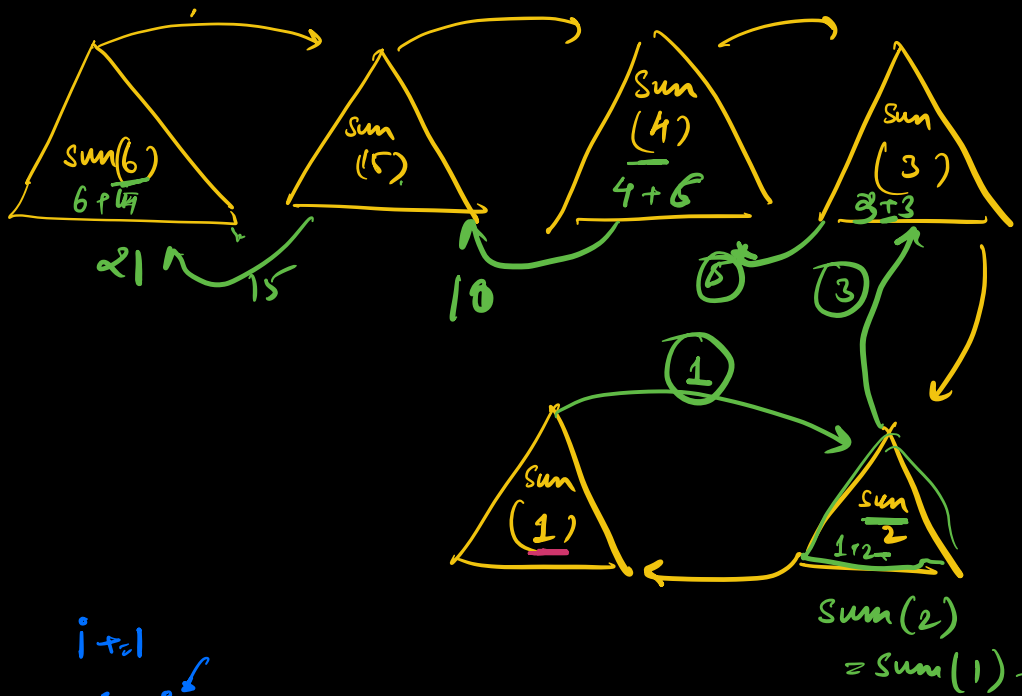
Repetition ⇒ same thing happening again and again

Recursion. → A function is calling itself.

→ Solve the bigger problem by solving smaller problems and using their results. (sub problems)

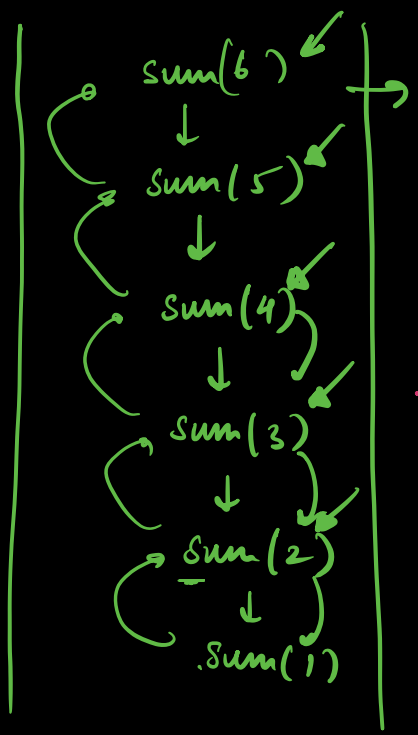
Find sum of first 6 natural numbers

$$sum(6) = sum(5) + 6$$



i += 1

i = i + 1



$sum(n) = sum(n-1) + n$

```
def sum(n):
```

→ (1) How to write recursive code ? * TRUST

(2) How actually recursion works

(3) TC, SC for recursion

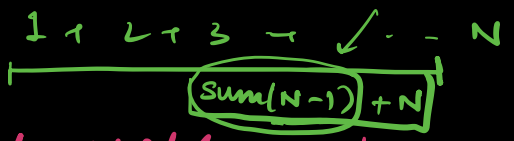
(1) Assumption. ⇒ Just simply assume that your function will solve the problem in hand.

TRUST

Example

```
def sumN(N):
    # Returns sum of first N natural numbers
```

(2) Main Logic:



Solve the current problem by using the result from the subproblem (smaller problem)

Example:

For find sum of N natural no.s

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

(3) Base Condition:

The problem to which you know the answer to.

⇒ where you would want your recursion to stop.

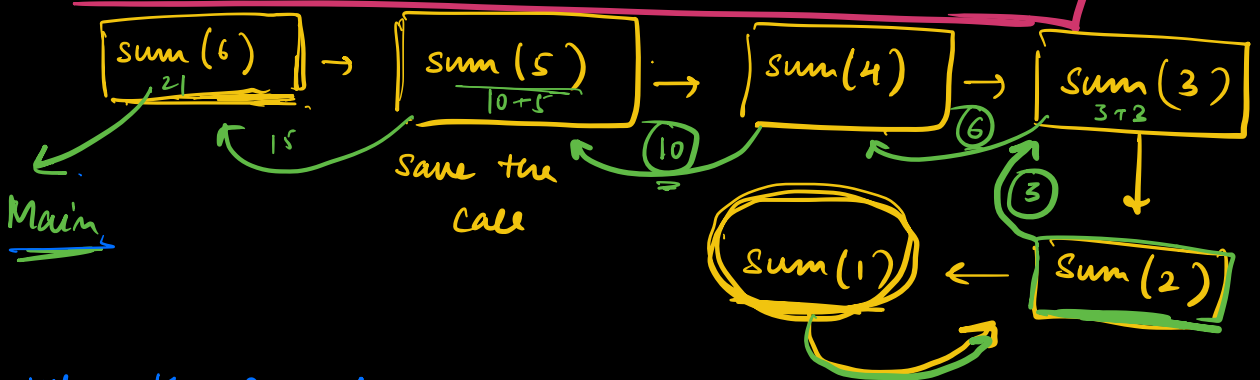
Example

```
if N == 1:
    return 1
```

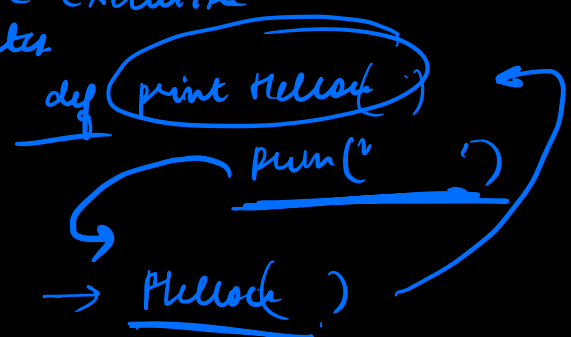
* TERMINATING POINT

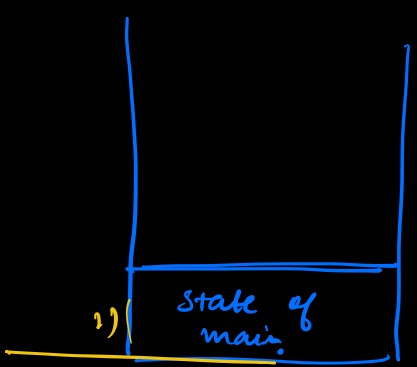
```
if N == 0:
    return 0
```

```
def sumN ( n ) :
    # This fn returns sum of first
      N natural no.s
    if n == 1 :
        return 1
    return sum(n-1) + n
```

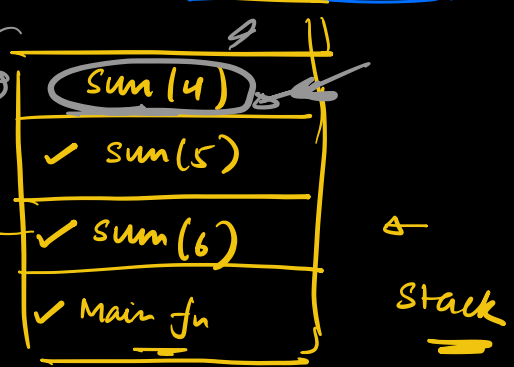


When the execution completes Returns the control to the Calling fn





State of all the fn Calls is saved!



Q Factorial

0!

$$n! = n \times (n-1) \times (n-2) \ldots 1$$

Write a recursive code that returns value of (n!)

$$n! = n \times (n-1)!$$

```
def fact(n):
    Assumption -
    if n == 0:
        return 1
    return n * fact(n-1)
```

Q Fibonacci sequence

0 1 2 3 4 5 6 7 8 ...
0 1 1 2 3 5 8 13

```
def fib(n):
    # returns the nth fibonacci number
    if n <= 1:
        return n
    return fib(n-1) + fib(n-2)
```

```
if n == 1
    return 0
if n == 2
    return 1
```

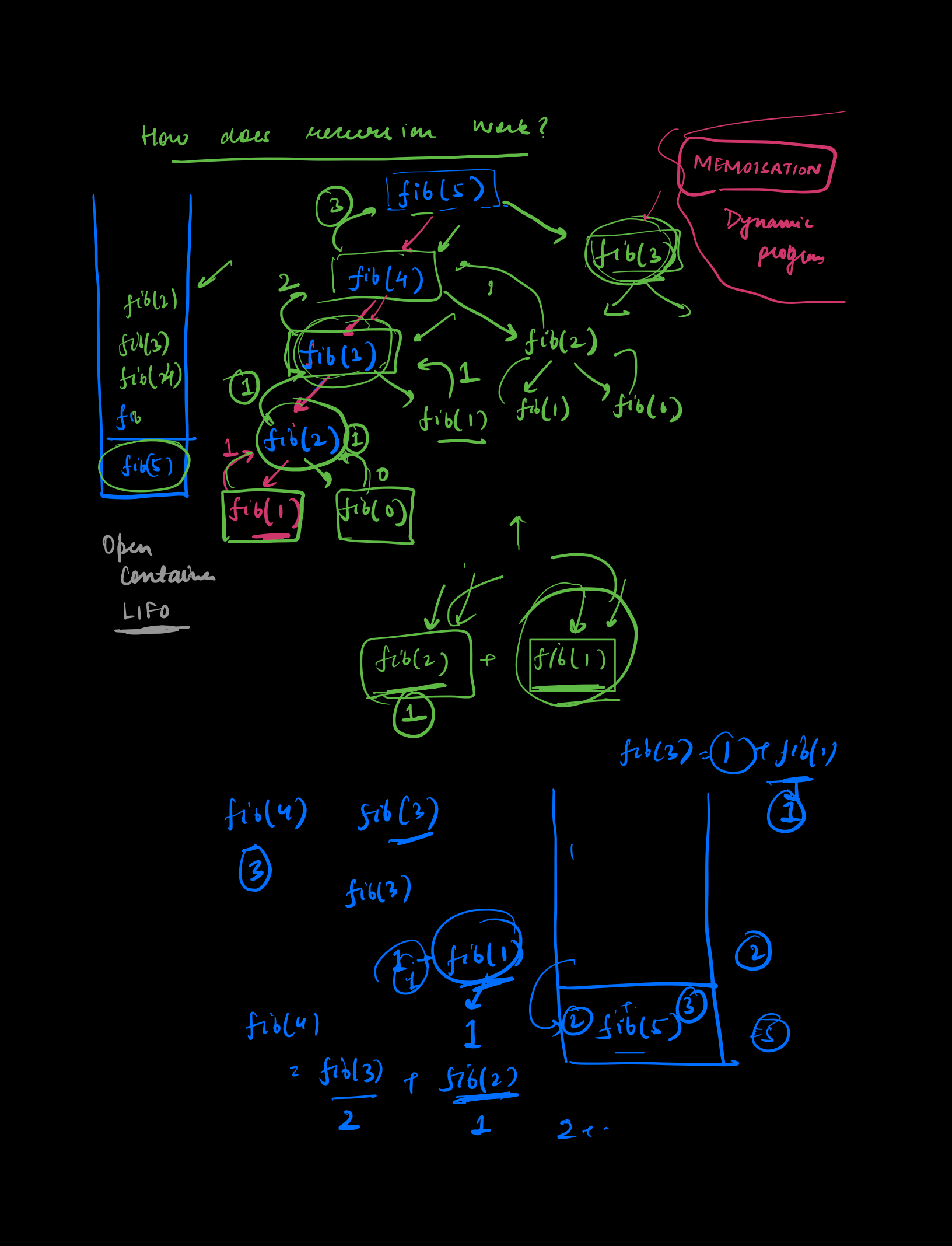
n = 2

fib(3) 1 + 1

fib(2) 3 + 2

a + b

fib(3) 2


How does recursion work?
MEMOISATION
Dynamic programming
fib(5)
fib(4)
fib(3)
fib(2)
fib(1)
fib(0)
fib(3)
fib(2)
fib(1)
fib(1)
fib(0)
fib(2)
fib(1)
fib(0)
fib(2)
fib(1)
Open
Contains
LIFO
fib(4) fib(3)
fib(3)
fib(1)
fib(4)
= fib(3) + fib(2)
fib(5)
fib(3) = 1 + fib(1)

```
def printDec(N):
    # it will the nos in
      decreasing order starting from N
    if N==0:
        return
    print(N)
    printDec(N-1)
```

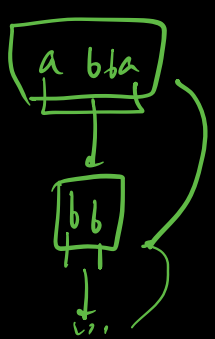
9 8 7 6 5 4 3 2 1

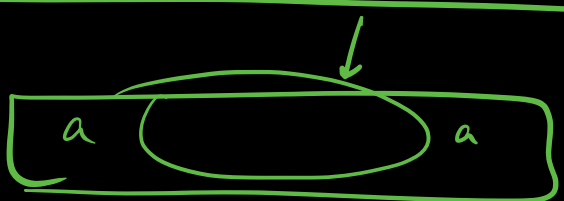
Q Check if a string is Palindrome or not (WITH RECURSION)

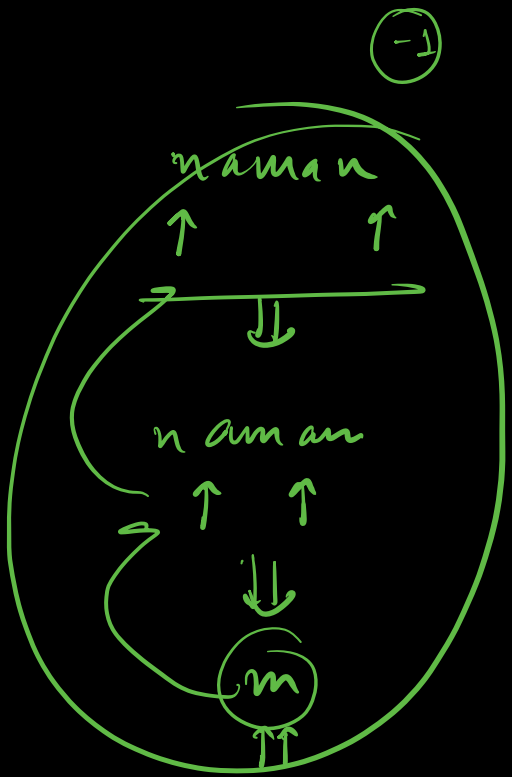
```
def solve(s):
```

```
def solve(s):
    if len(s)<=1:
        return True
    if s[0]==s[-1] and solve(s[1:-1])
        return True
    return False
```

O(N)







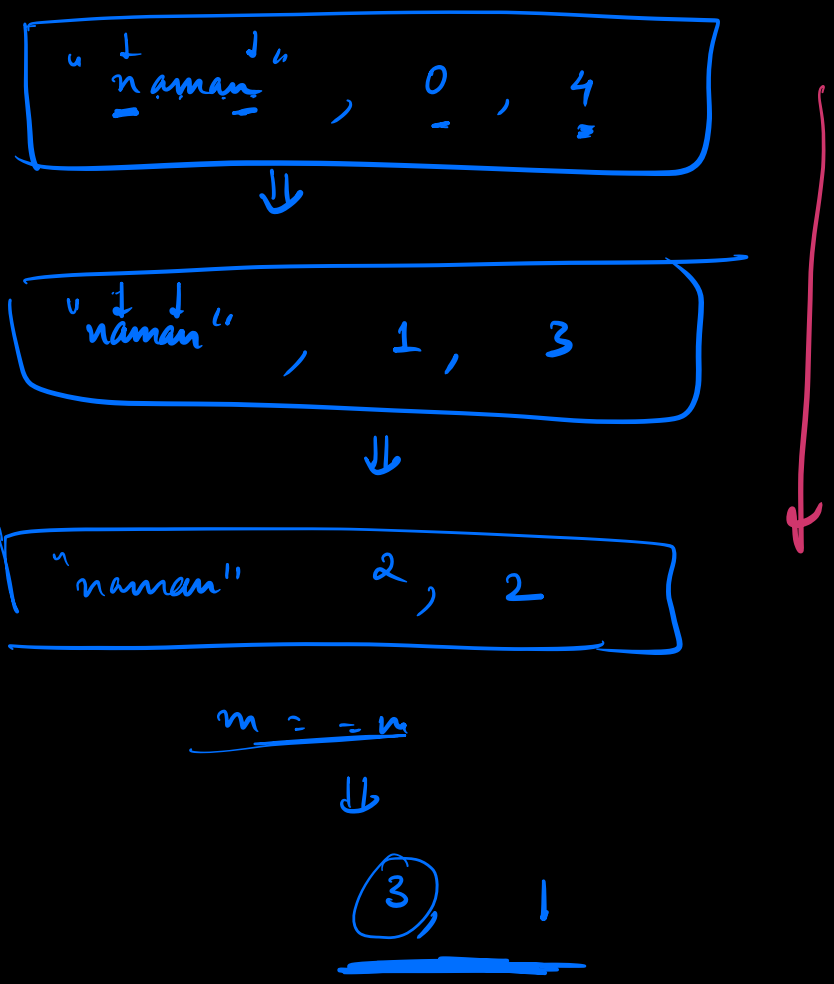

```
solve(s):
    return isPalindrome(s, 0, len(s) - 1)
```

```
bool isPalindrome(s, st, end)
    # Returns True if
    sub. string from start
    to end is a palindrome
```


```
if s[st] == s[end]
    and isPalindrome(s, st+1, end-1)
    return True
return False
```

Base Cond:

```
if start >= end:
    return True
```


"naman", 0, 4
"naman", 1, 3
"naman", 2, 2
n == n
3
1