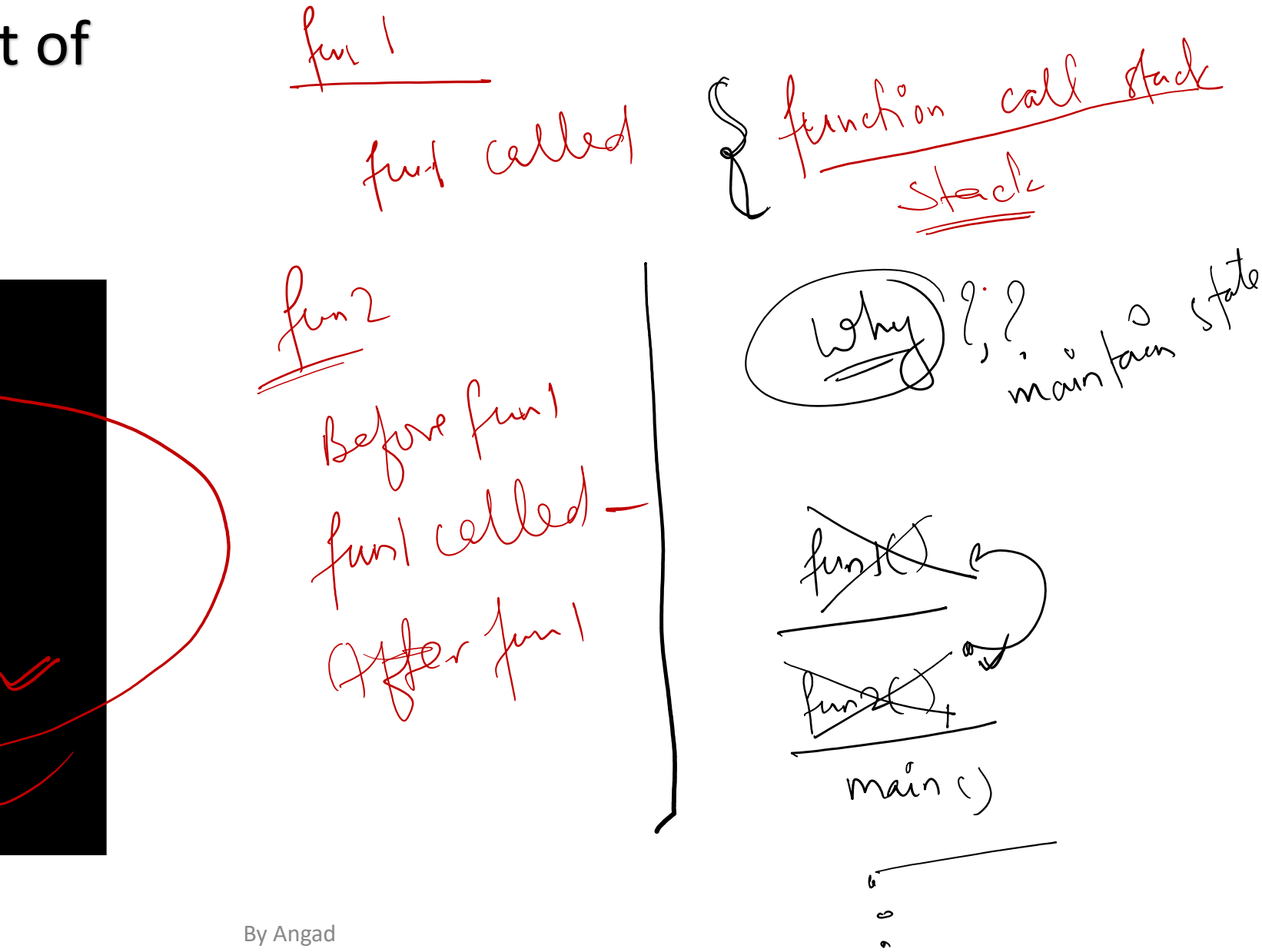
What will be the output of this code?

```
# First Basic Example
def fun1():
    print("fun1() Called")


def fun2():
    print("Before fun1()")
    fun1()
    print("After fun1()")
```

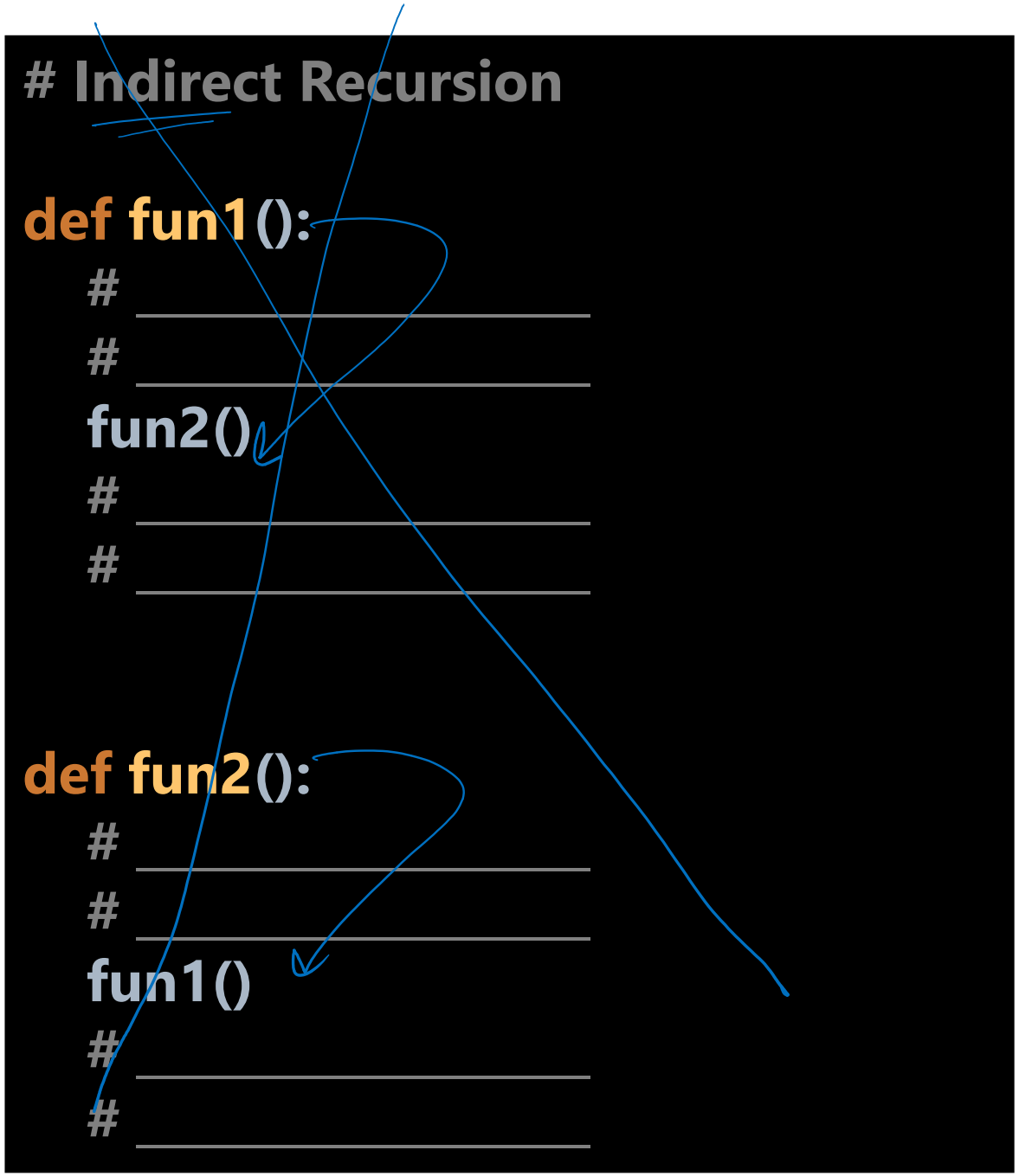
main → fun2()



Recursion: A function calls itself Directly or Indirectly

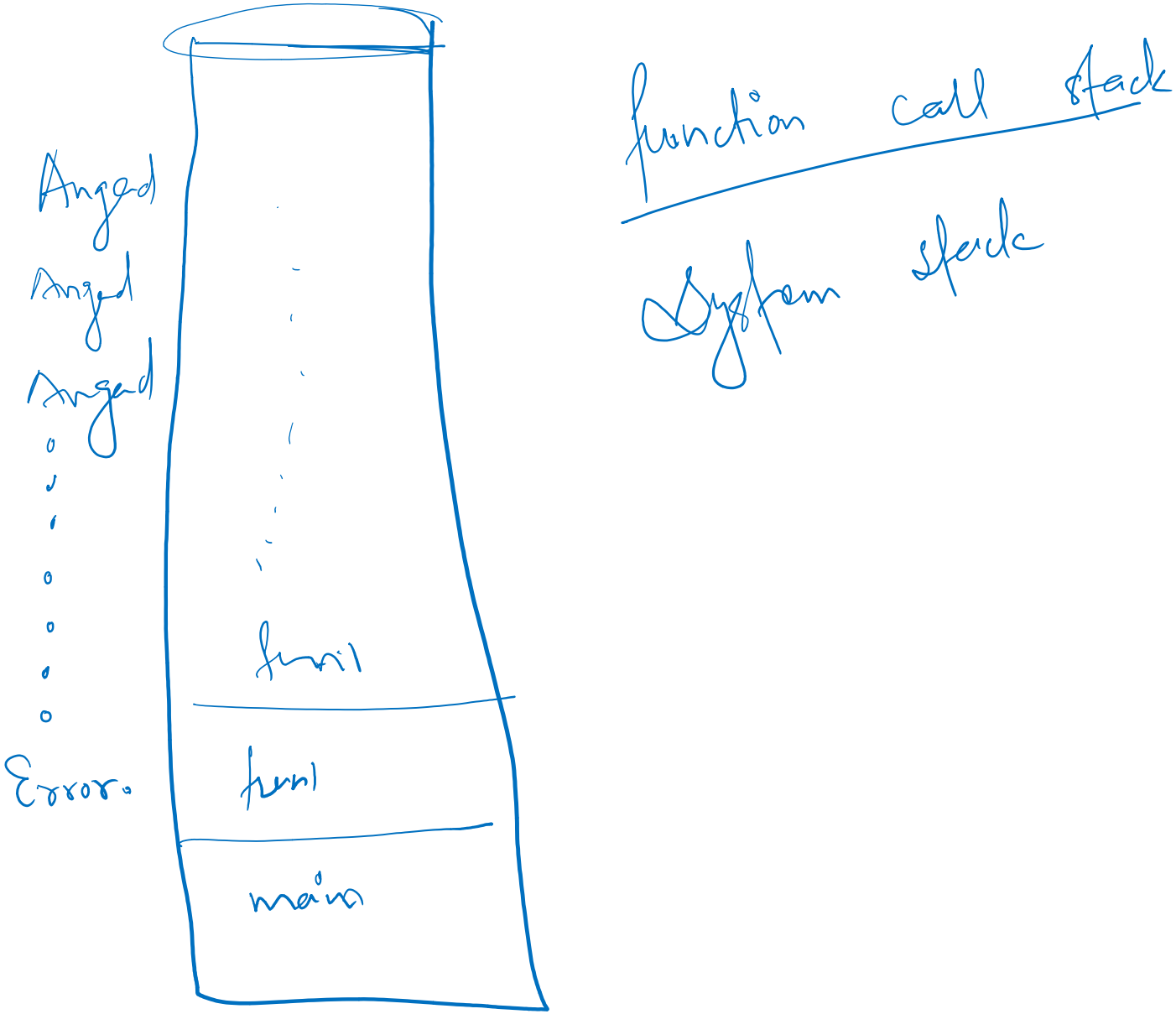
```
# Direct Recursion

def fun1():
    # _____
    # _____
    fun1()
    # _____
    # _____
```

```
# Indirect Recursion

def fun1():
    # _____
    # _____
    fun2()
    # _____
    # _____


def fun2():
    # _____
    # _____
    fun1()
    # _____
    # _____
```



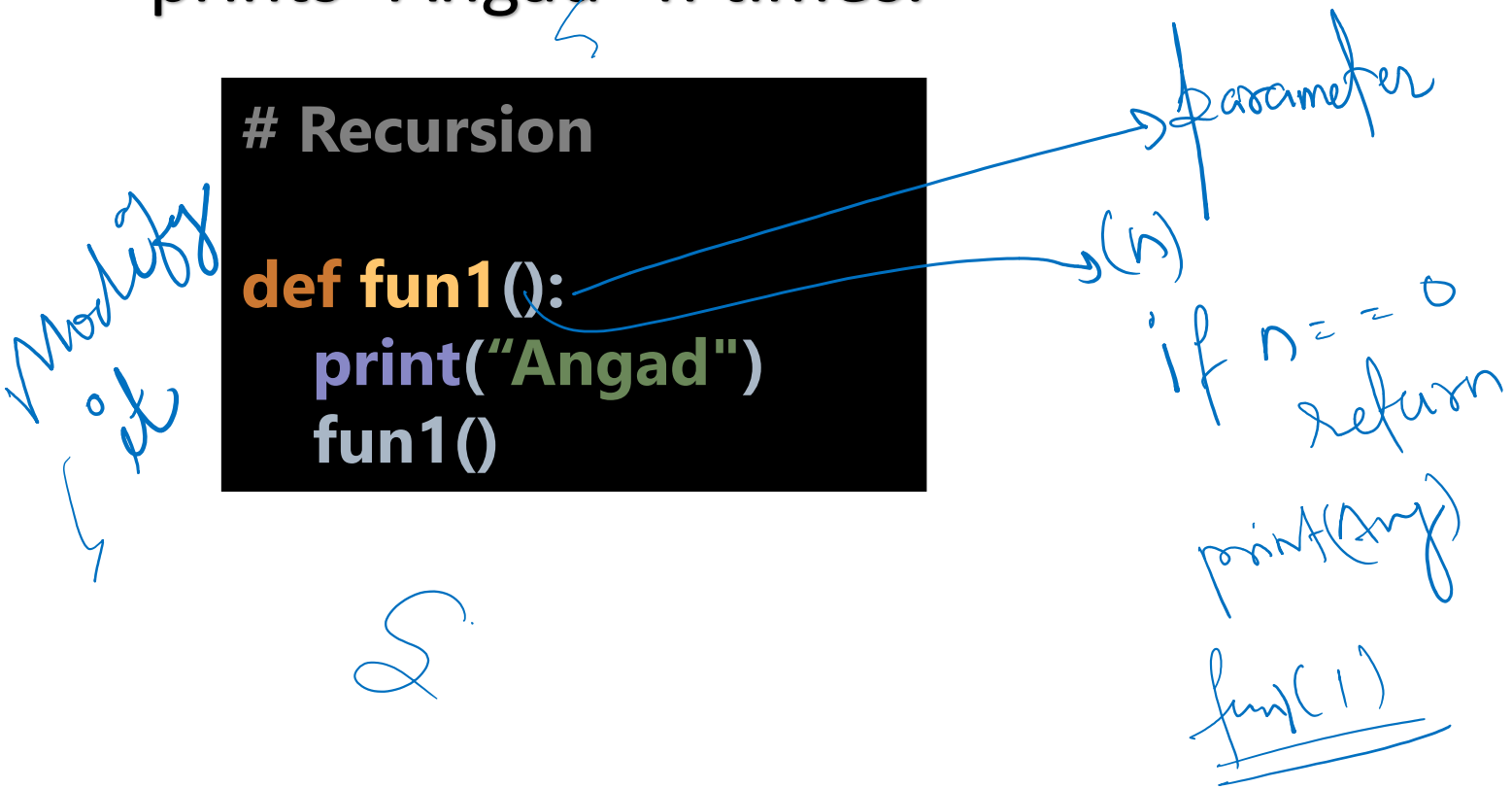
What will be the output of this code?

```
# Recursion

def fun1():
    print("Angad")
    fun1()
```



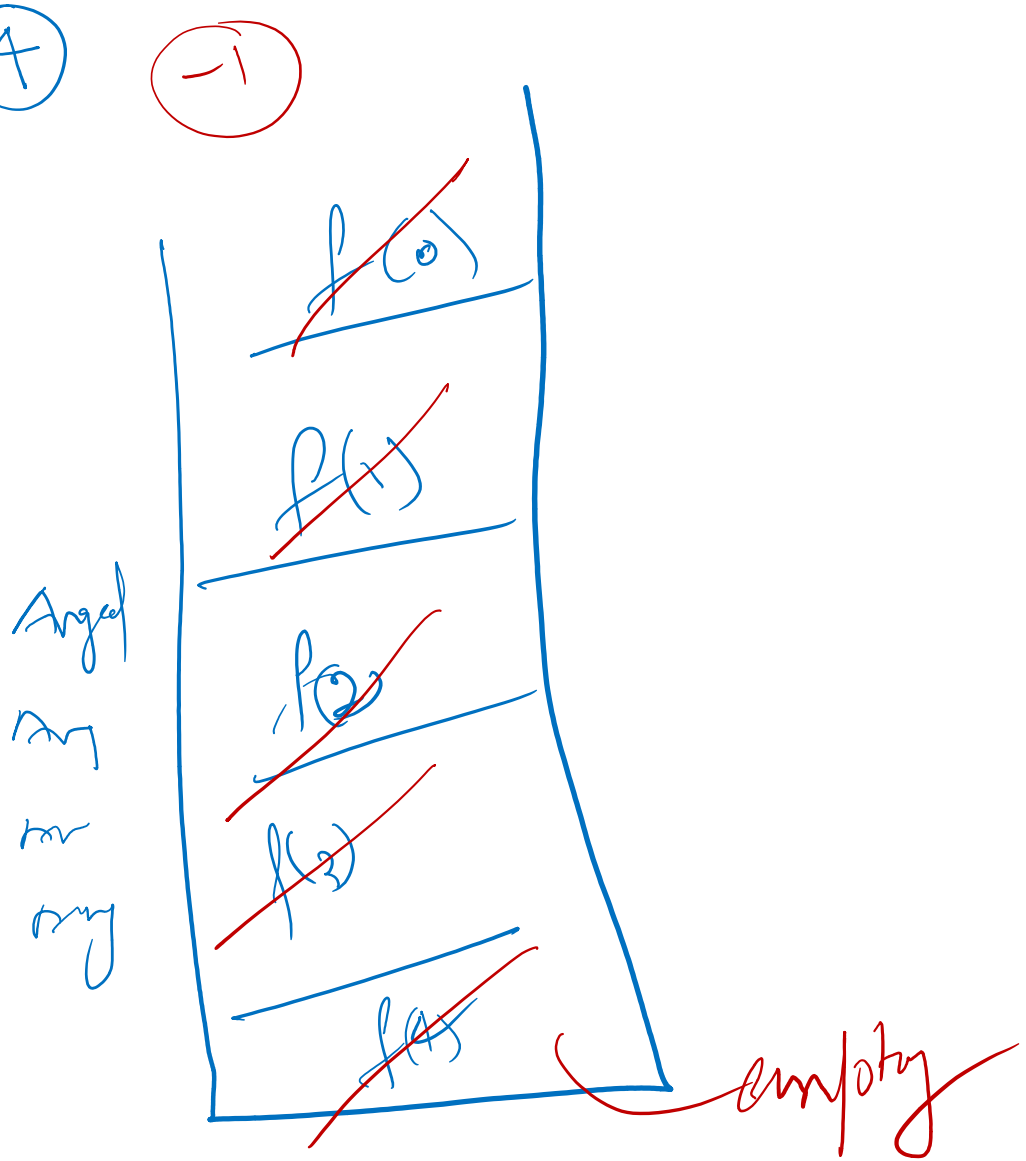
Write a function which prints “Angad” n times.

```
# Recursion

def fun1():
    print("Angad")
    fun1()
```



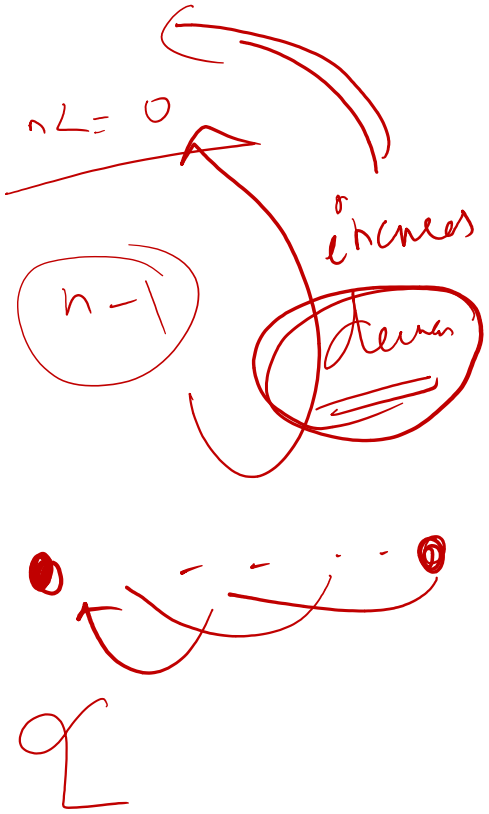
Different Cases for inputs like Zero, Negative

```
def fun1(n):
    # Base Case
    if n == 0:
        return
    print("Angad")

    # Recursive Case
    fun1(n-1)
```



Typical Structure of Recursive Functions

```
# Typical Recursive Function

def funX(.....parameters.....):
    # Base Case
    # _____________________
    # _____________________

    #Recursive Call
    funX(.....parameters.....)
    # _____________________
    # _____________________
```



With at least on change to the parameter so that call approaches towards base case.

Applications of Recursion

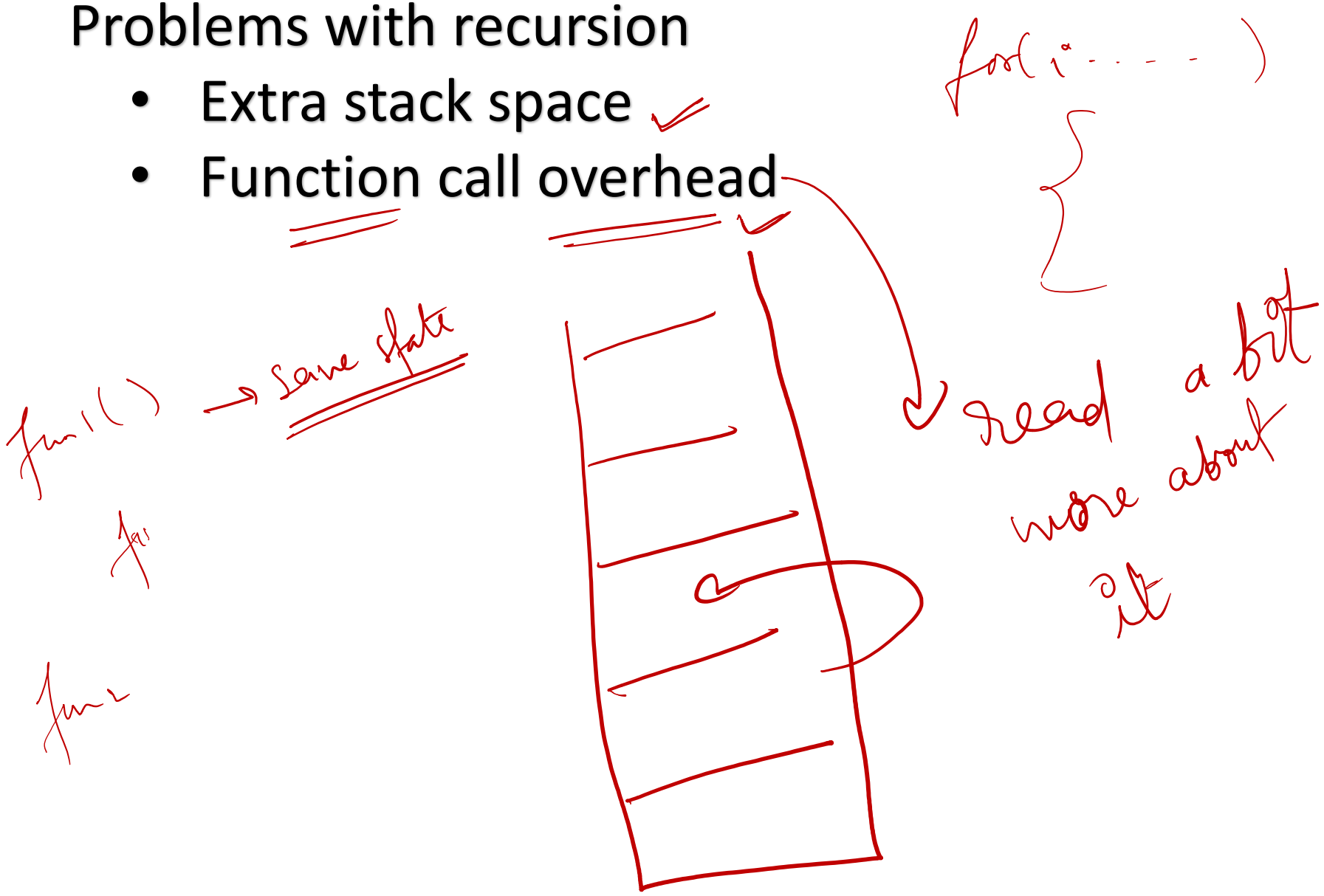
- Many Alogs are base on recursion
  - DP → dynamic programming
  - Backtracking
  - Diving & Conquer
- Many problems are inherently recrusive
  - Tower of Hanoi
  - DFS Based Traversals
    - DFS of Graph
    - Tree DFS Traversals

Solid on Recursion

Easy for you

Problems with recursion

- Extra stack space
- Function call overhead

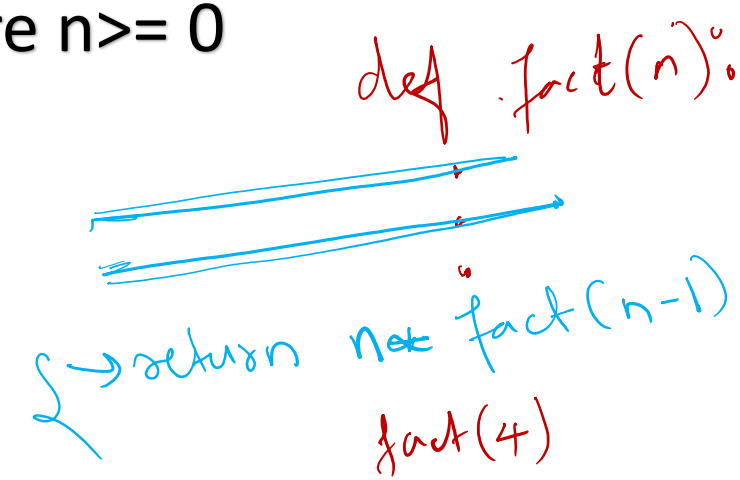


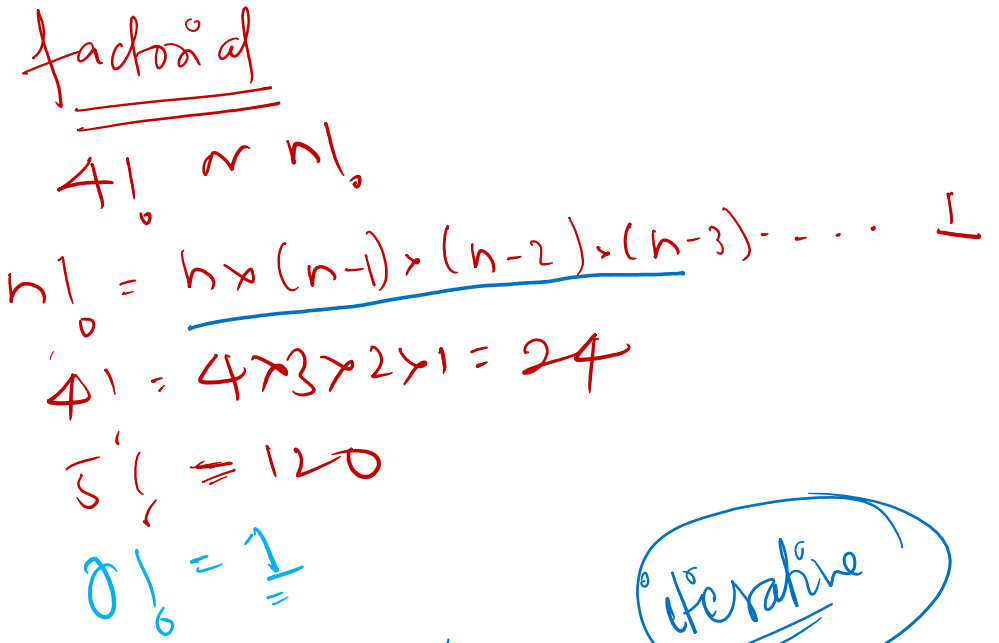
Writing Base Cases in Recursion

Write recursive solution for it

Factorial n where n>= 0

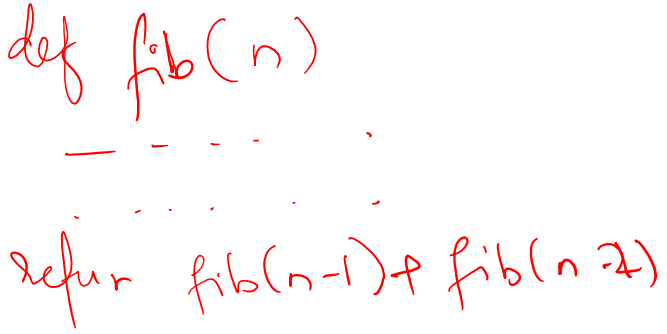
- I/P: n=4
- O/P: 24
- I/P: n=0
- O/P: 1

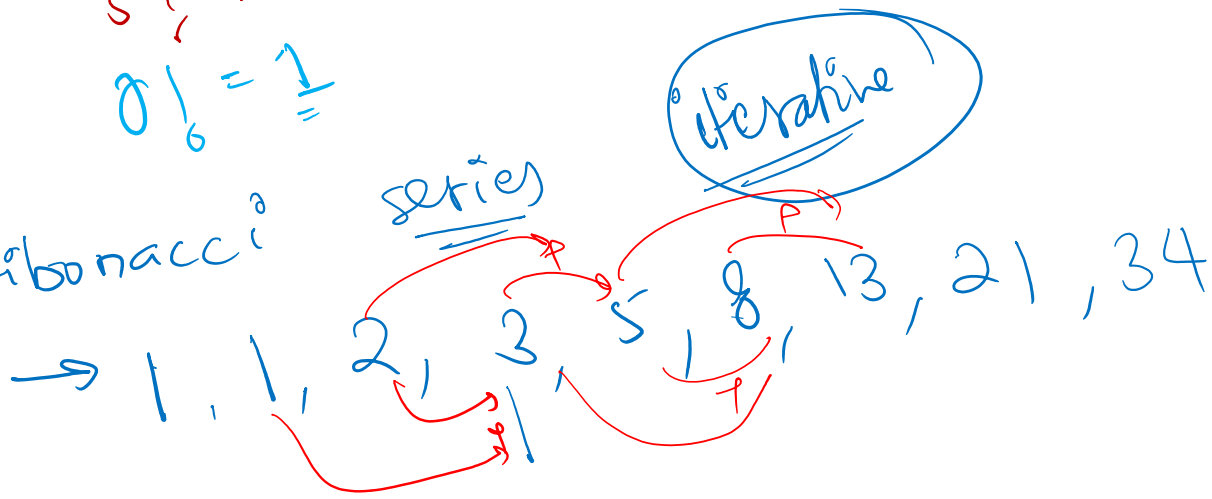




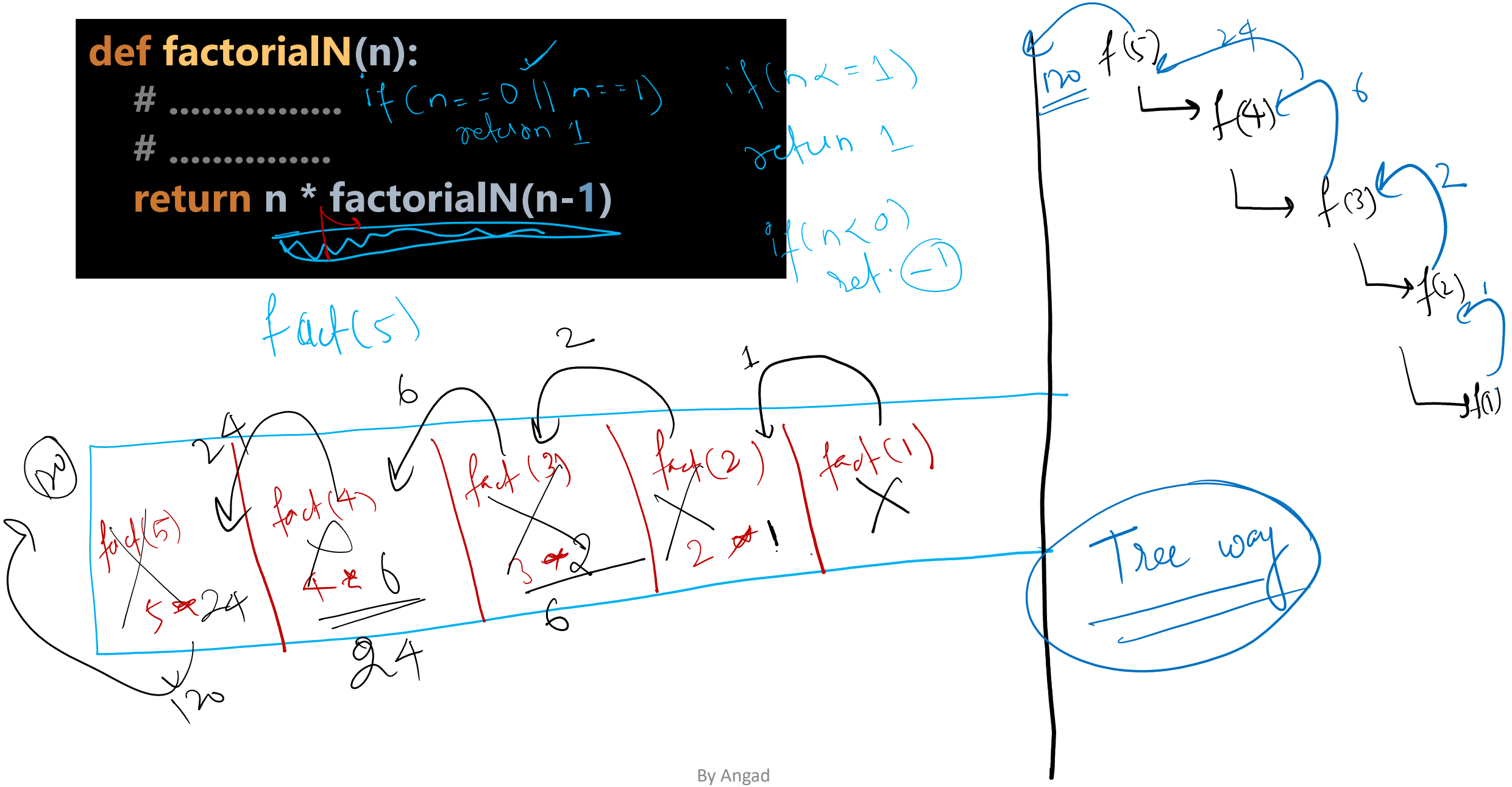
N-th Fibonacci Number where n>= 0

- I/P: n=4
- O/P: 3
- I/P: n=0
- O/P: 0





```
def factorialN(n):
    # ..............
    # ..............
    return n * factorialN(n-1)
```

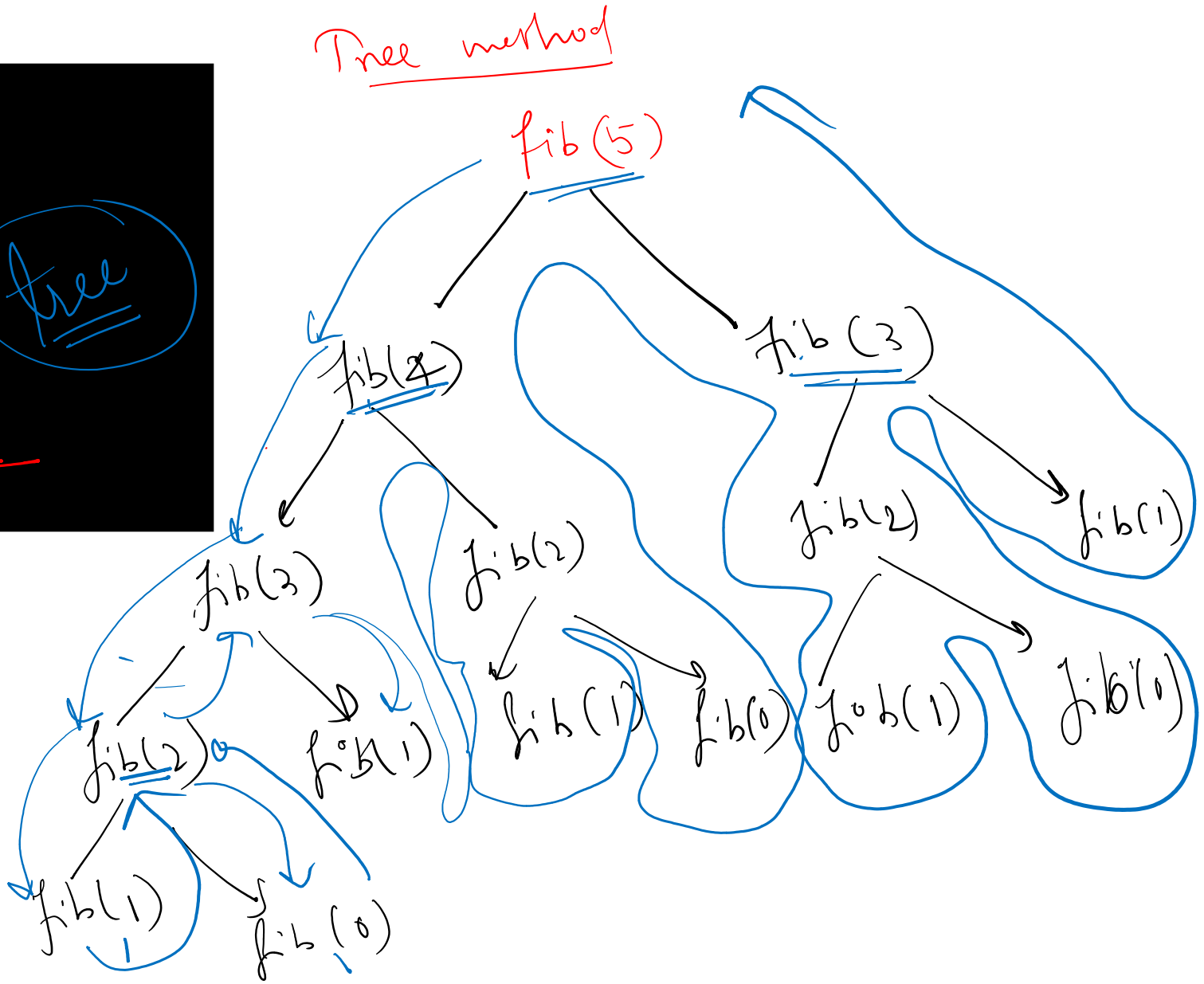


```
def fib(n):
    # .....
    # .....
    # .....
    # .....
    return fib(n-1) + fib(n-2)
```

if(n<=1) return n;

tree

Draw recursion tree for fib(6)



Tail Recursion

A recursive function is called tail recrusive if the function does not do anything after the last recursive call .

```
def fun1(n):
    if n == 0:
        return
    print("Angad")
    fun1(n-1)
```

```
def fun2(n):
    # ..............
    # ..............
    fun2(n-1)
    print(n, end="")
```

```
def factorialN(n):
    # ..............
    # ..............
    return n * factorialN(n-1)
```

Tail Call Elimination

eliminate recursion

Recursive to non recursive

- C++, Java

```
def fun1(n):
    if n == 0:
        return
    print(n, end="")
    fun1(n-1)
```

n = n-1

goto 1

Python does not do it.

Extra space

func()

non-tail Recursion we can't do it

Conversion to Loop

```
def fun1(n):
    if n == 0:
        return
    print(n, end="")
    fun1(n-1)
```

```
def fun1(n):
    while n != 0:
        print(n, end="")
        n = n-1
```

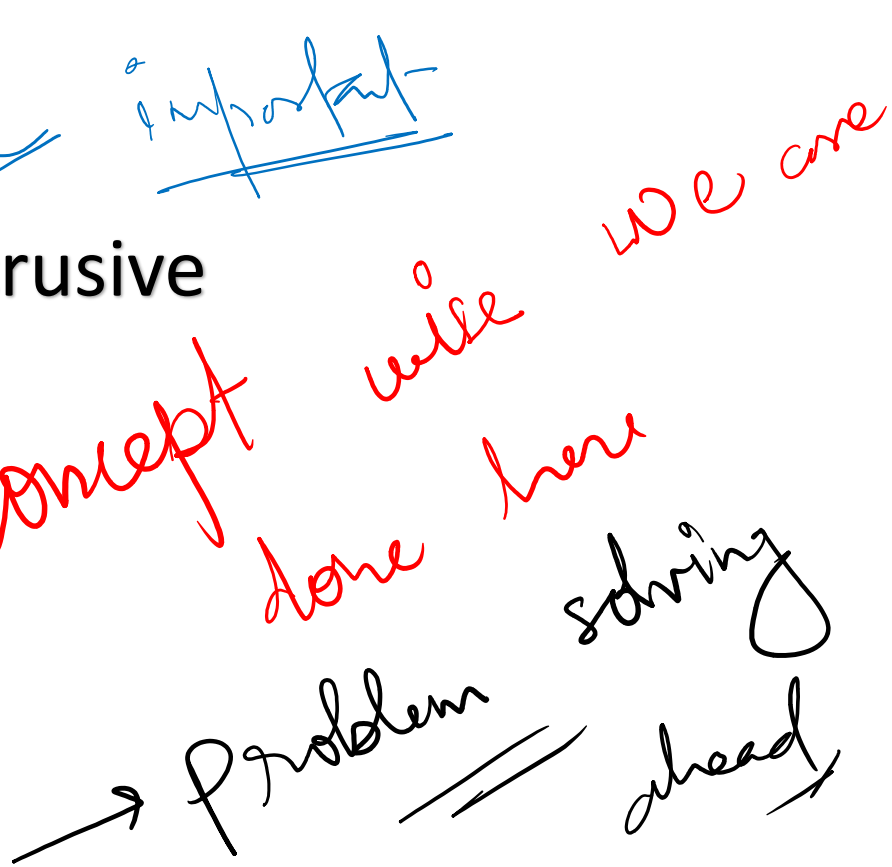
- Replace If with while
- Changes values of Parameters at the end

Examples of Tail Recursion

Quick Sort

Post Order Tree Traversal

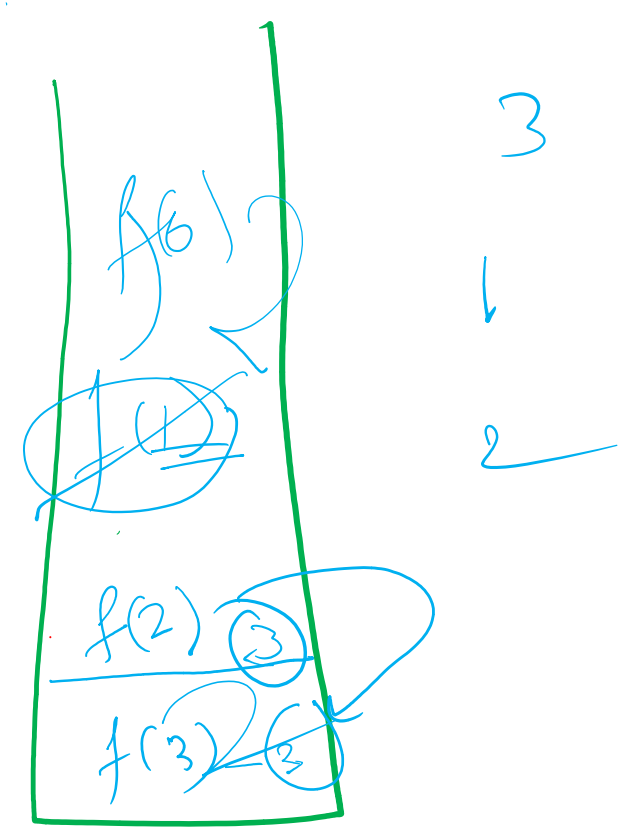
- Merge Sort is not Tail recrusive



Recursion Practice : Finding Output

Find Output of the below program

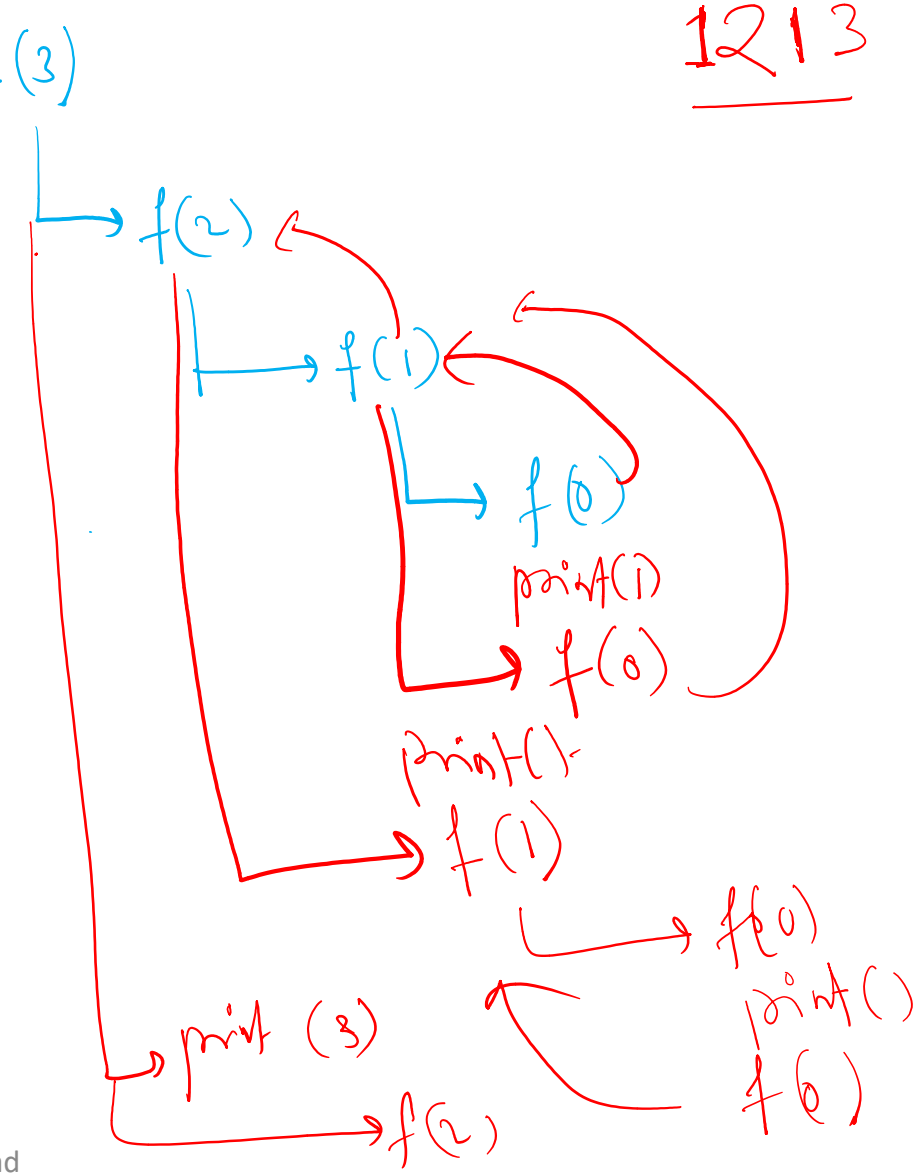
```
def prinFun(n):
    if n == 0:
        return
    print(n)
    prinFun(n - 1)
    print(n)

prinFun(3)
```



Recursion Practice : Finding Output

Find Output of the below program

```
def printFun2(n):
    if n == 0:
        return
    printFun2(n - 1)
    print(n)
    printFun2(n - 1)
```

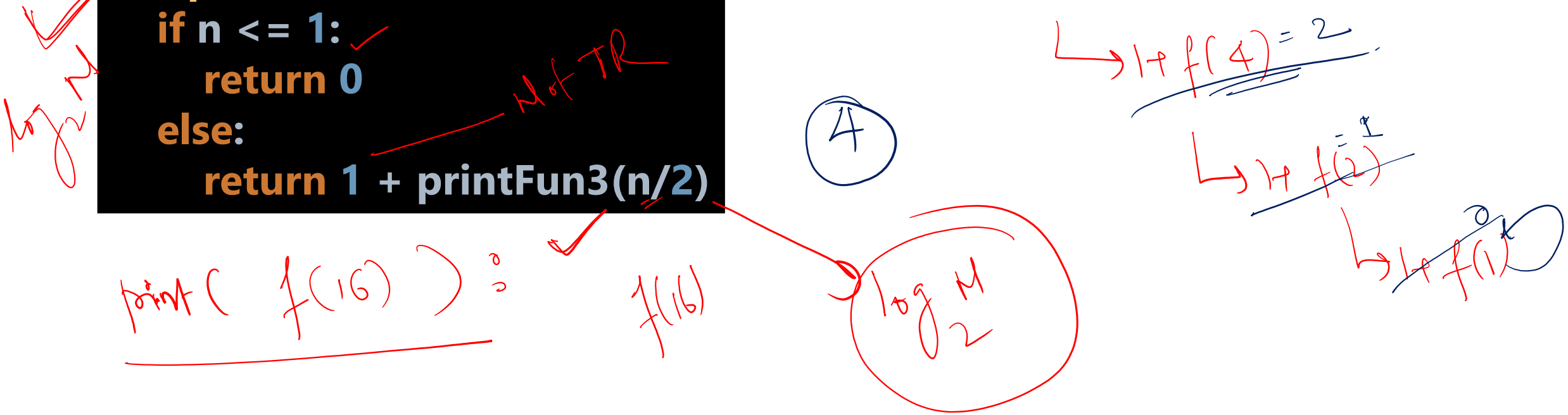


1213 (12)

Recursion Practice : Finding Output

Find Output of the below program

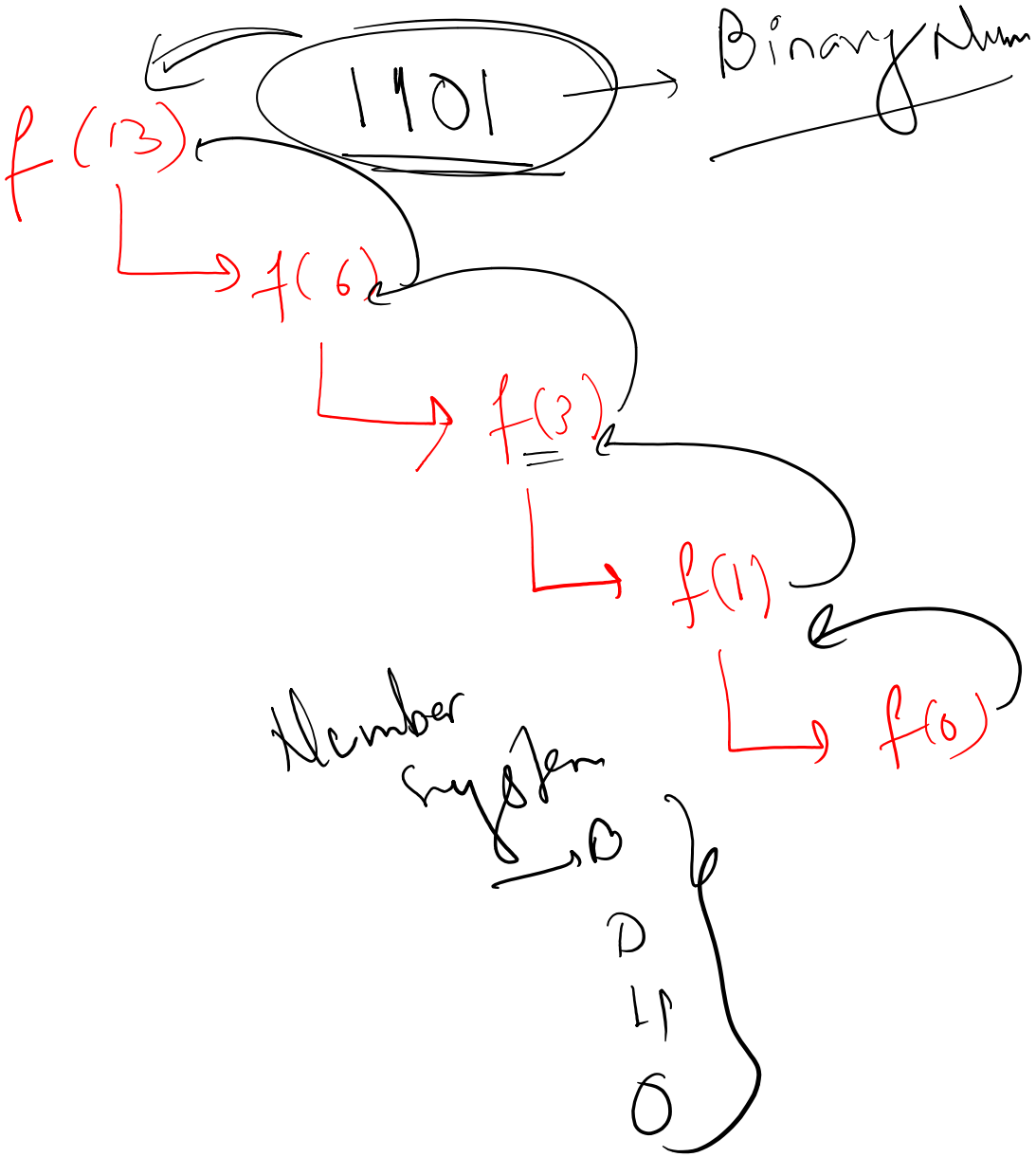
```
def printFun3(n):
    if n <= 1:
        return 0
    else:
        return 1 + printFun3(n/2)
```



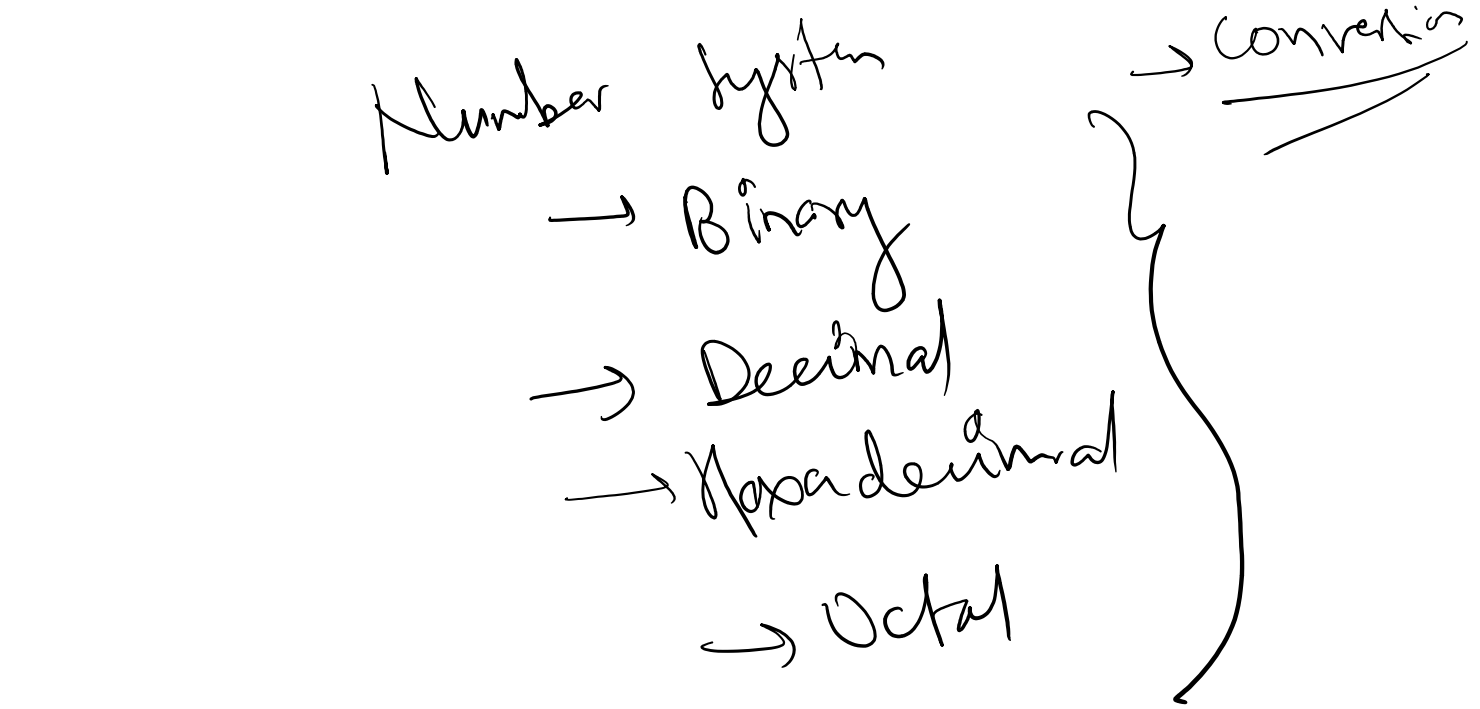
Recursion Practice : Finding Output

Find Output of the below program

```
def printFun4(n):
    if n == 0:
        return
    printFun4(n // 2)
    print(n % 2)
```



Recursion Practice : Finding Output

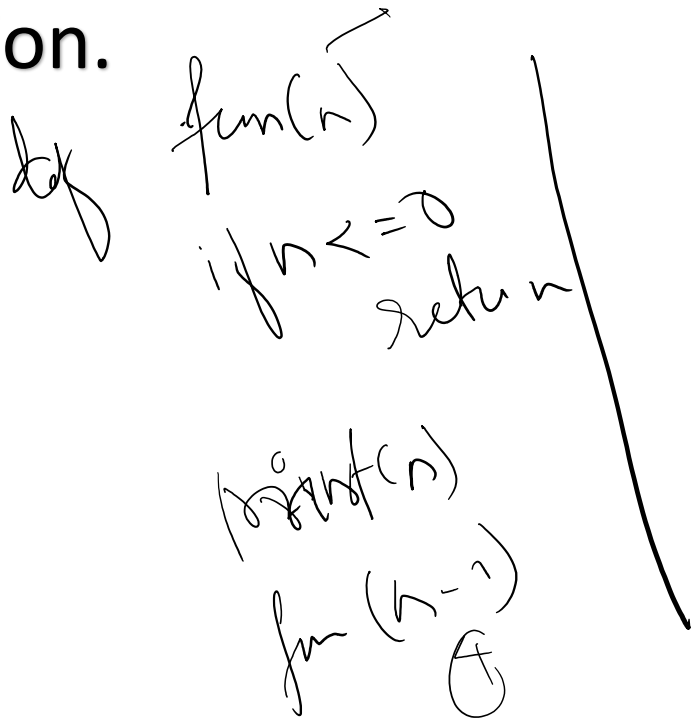


Recursion Practice : Writing Programs

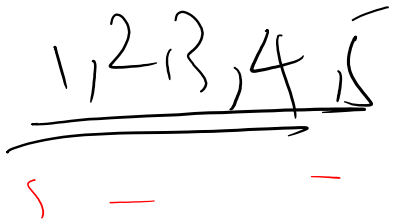
Print n to 1 using Recursion.

- I/P: n=5
- O/P: 5 4 3 2 1

- I/P: n=2
- O/P: 2 1



Recursion Practice : Writing Programs



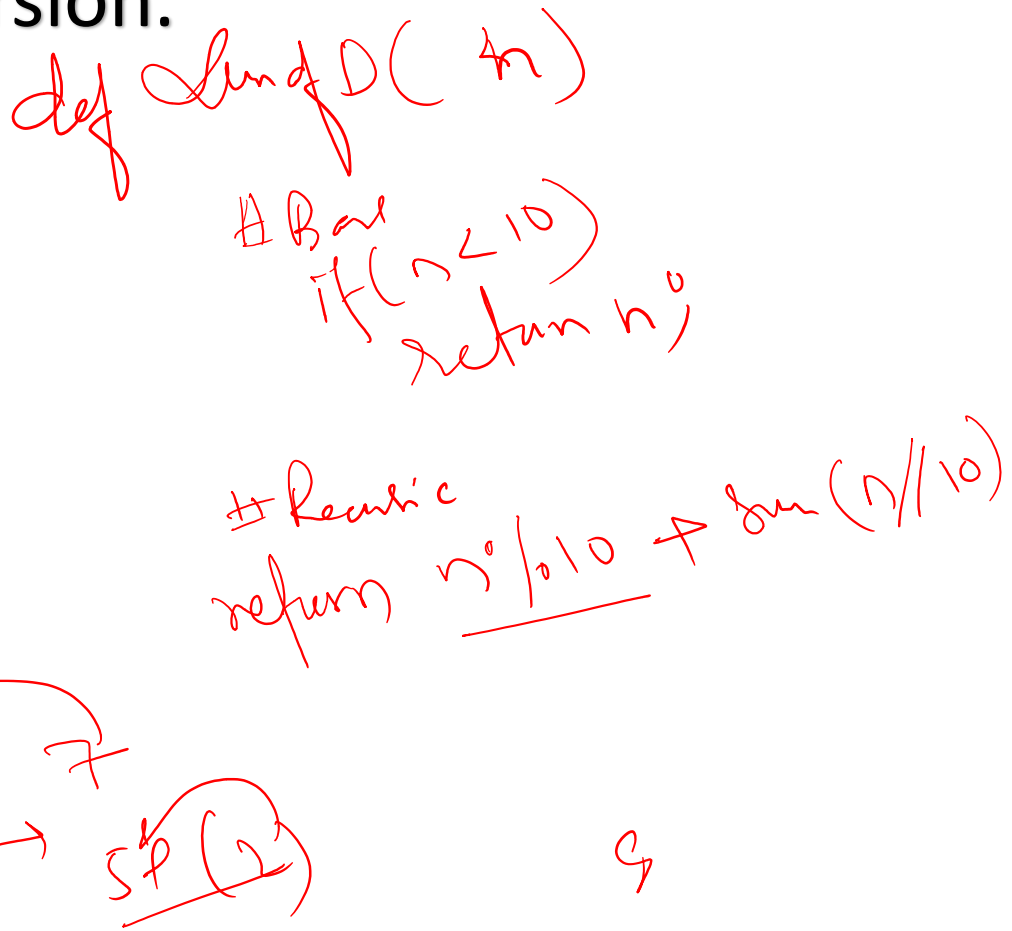
```
def print1toN(n):
    if n == 0:
        return
    print1toN(n - 1)
    print(n)
```

Recursion Practice : Writing Programs

Sum of Digits using Recursion.

- I/P: n=253
- O/P: 10

- I/P: n=984
- O/P: 21



Recursion Practice : Writing Programs

Palindrome Check using Recursion.

- I/P: abbcbba
- O/P: YES

- I/P: abba
- O/P: YES

- I/P: geeks
- O/P: NO

Recursion Practice : Writing Programs

```
def pal(str, start, end):
    if start >= end
        return True:
    return (str[start] == str[end]) and
        pal(str, start+1, end-1)
```

1st and 2nd | 1st or 2nd