PYTHON PROGRAMMING
GATE DA/DSA

Agenda:

→ Recursive Function
- ↳ Definition, Examples
- ↳ Types of Recursion
  - Direct Recursion
  - Indirect ,,
  - Head Recursion
  - Tail Recursion
  - Finite Recursion
  - Infinite ,,

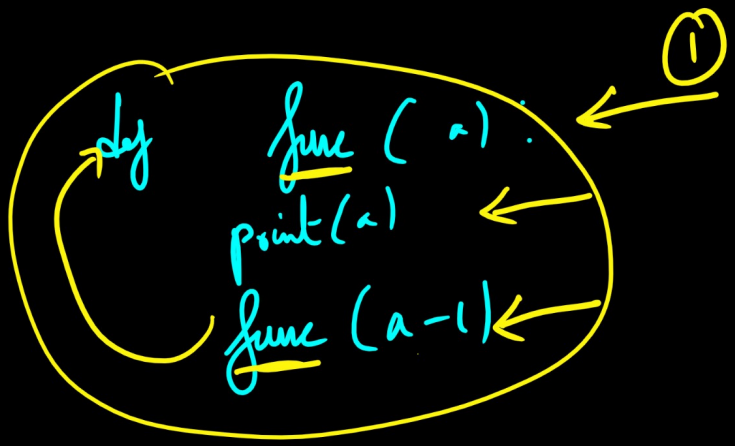
GATE 2024 DA

↓

2 problems in Python

just on Recursion concept.

Recursive Functions: The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called a Recursive Function.



func(10)

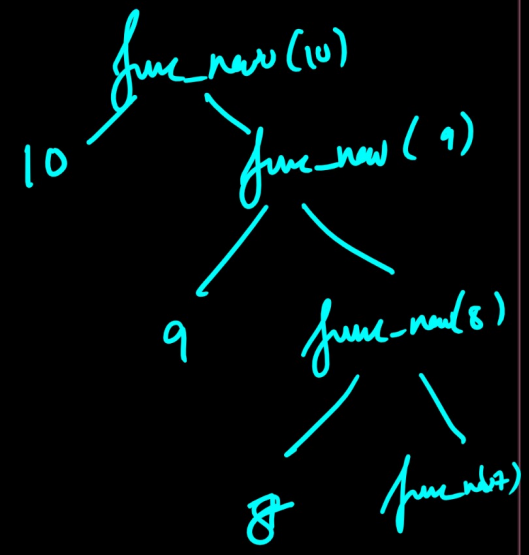
10	4	-3
9	3	⋮
8	2	⋮
7	1	⋮
6	0	
5	-1	
	-2	

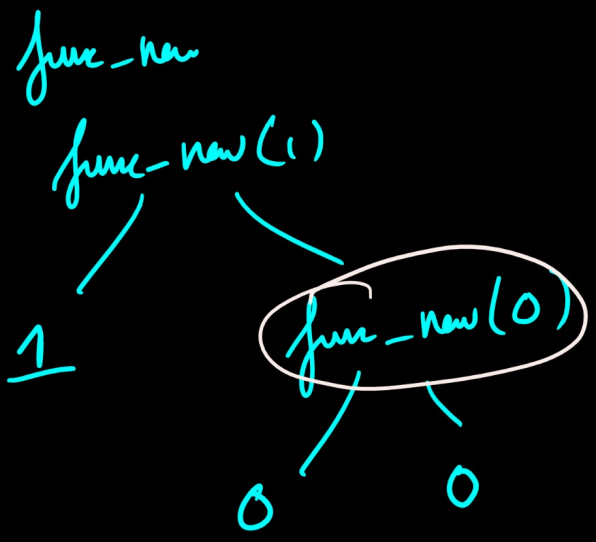
Every recursive function should have base condition inorder to stop the infinite recursion calls.

```
def func_new(a):
  print("Value of a : -->",a)
  if a<=0:
    return a
  else:
    return func_new(a-1)

func_new(10)
```

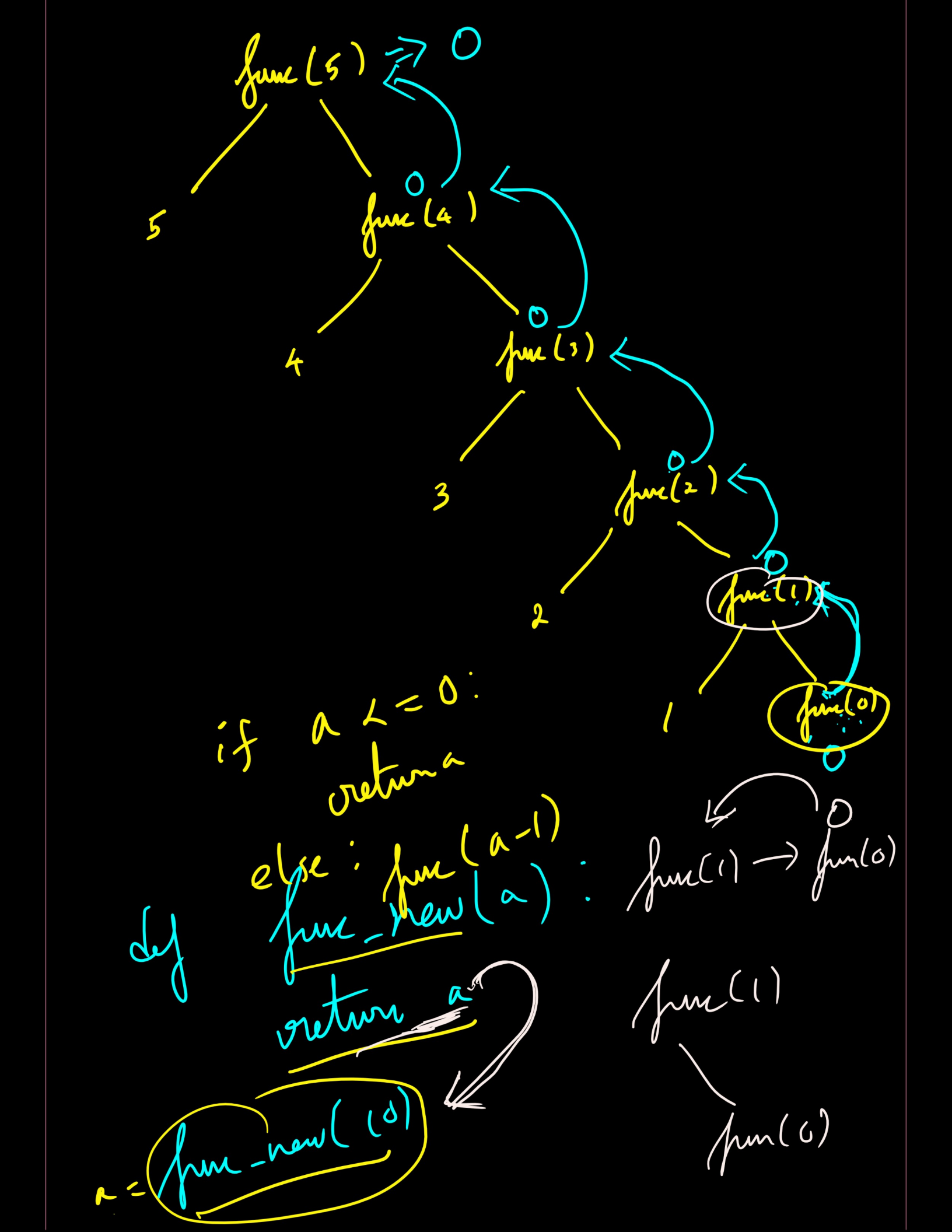
No more recursion

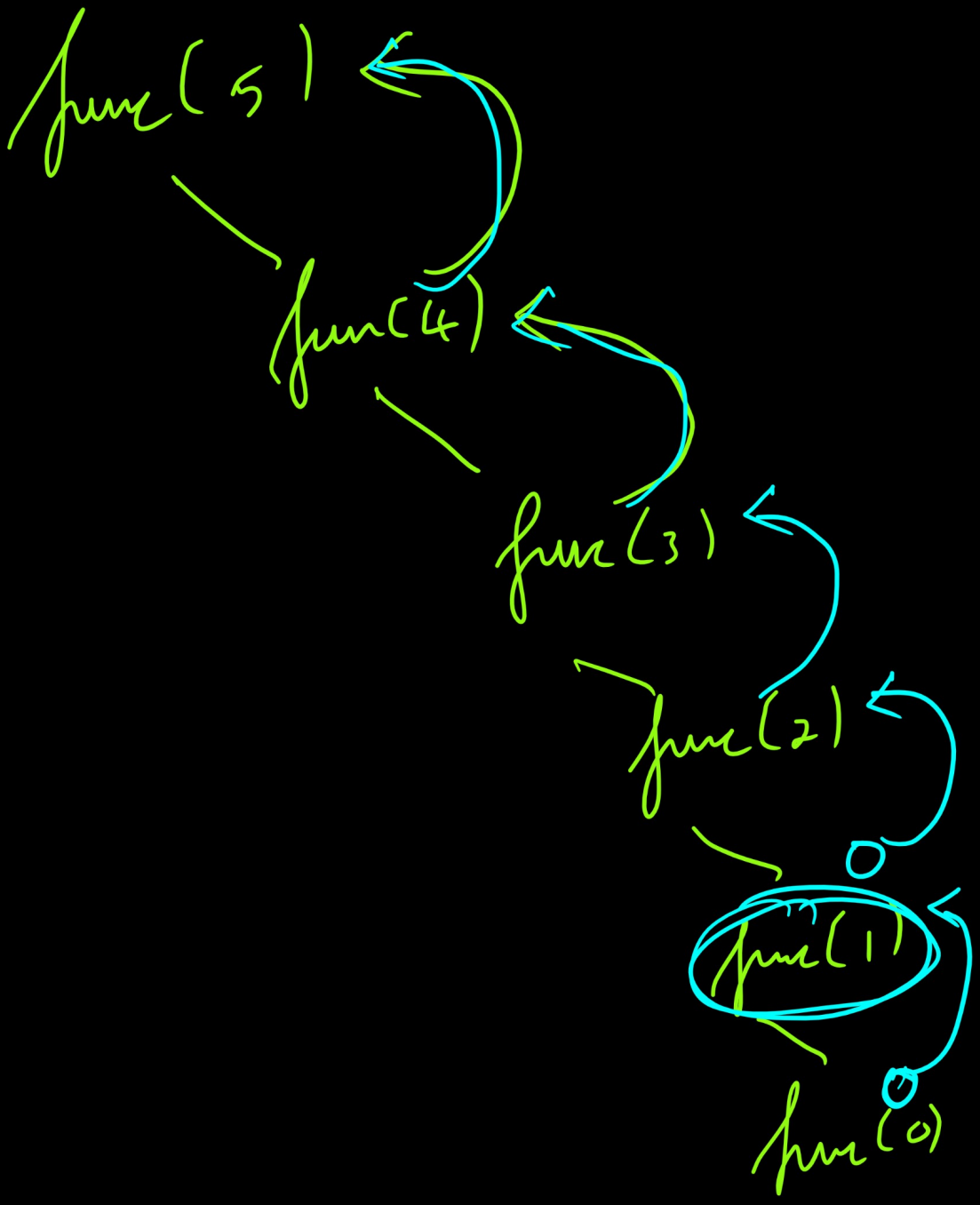




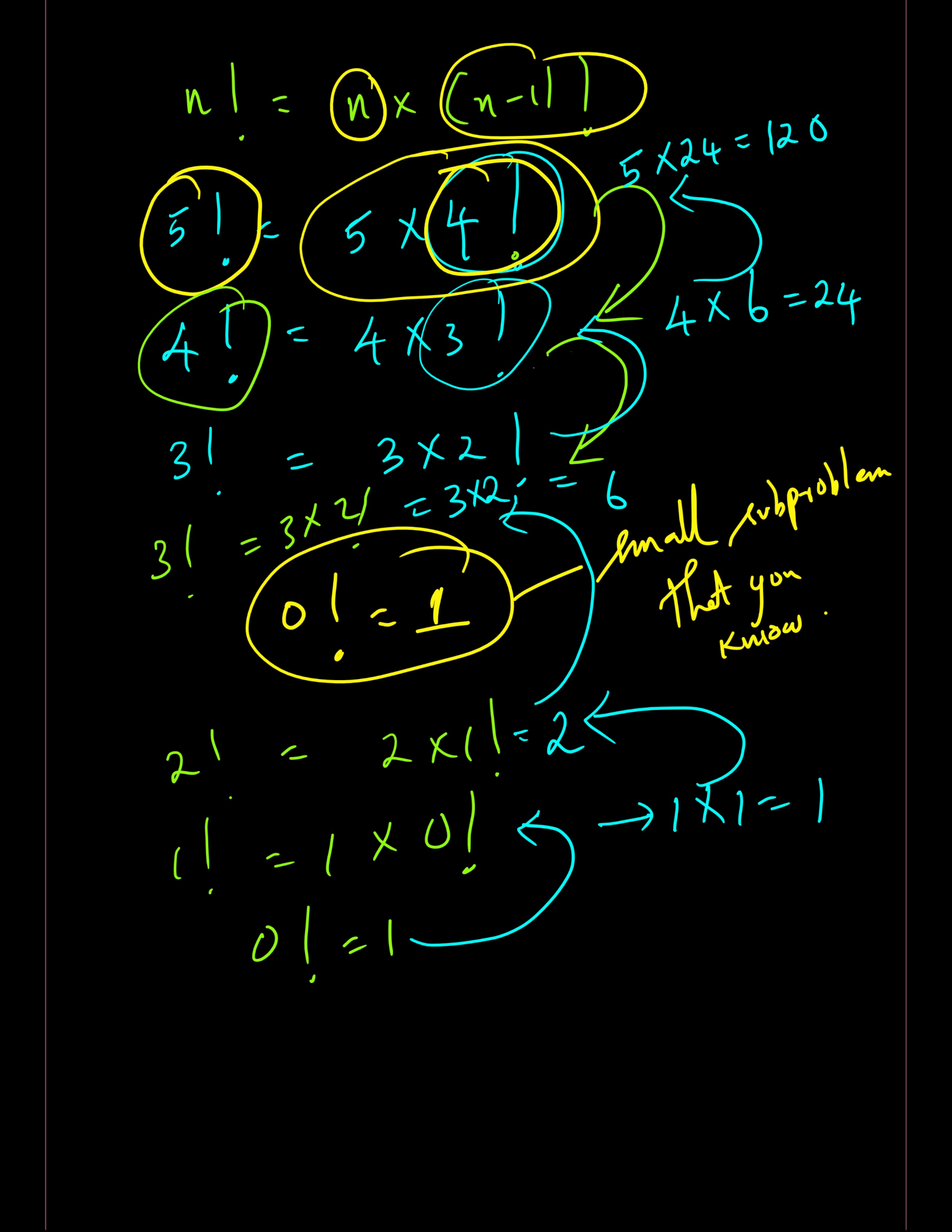
10 9 8 7 6 5 4 3 2 1

if a <= 0:
  return a


func(5)
5
func(4)
4
func(3)
3
func(2)
2
func(1)
1
func(0)
if a <= 0:
return a
else: func(a-1)
def func_new(a):
return a
a = func_new(0)
func(1) → func(0)
func(1)
func(0)

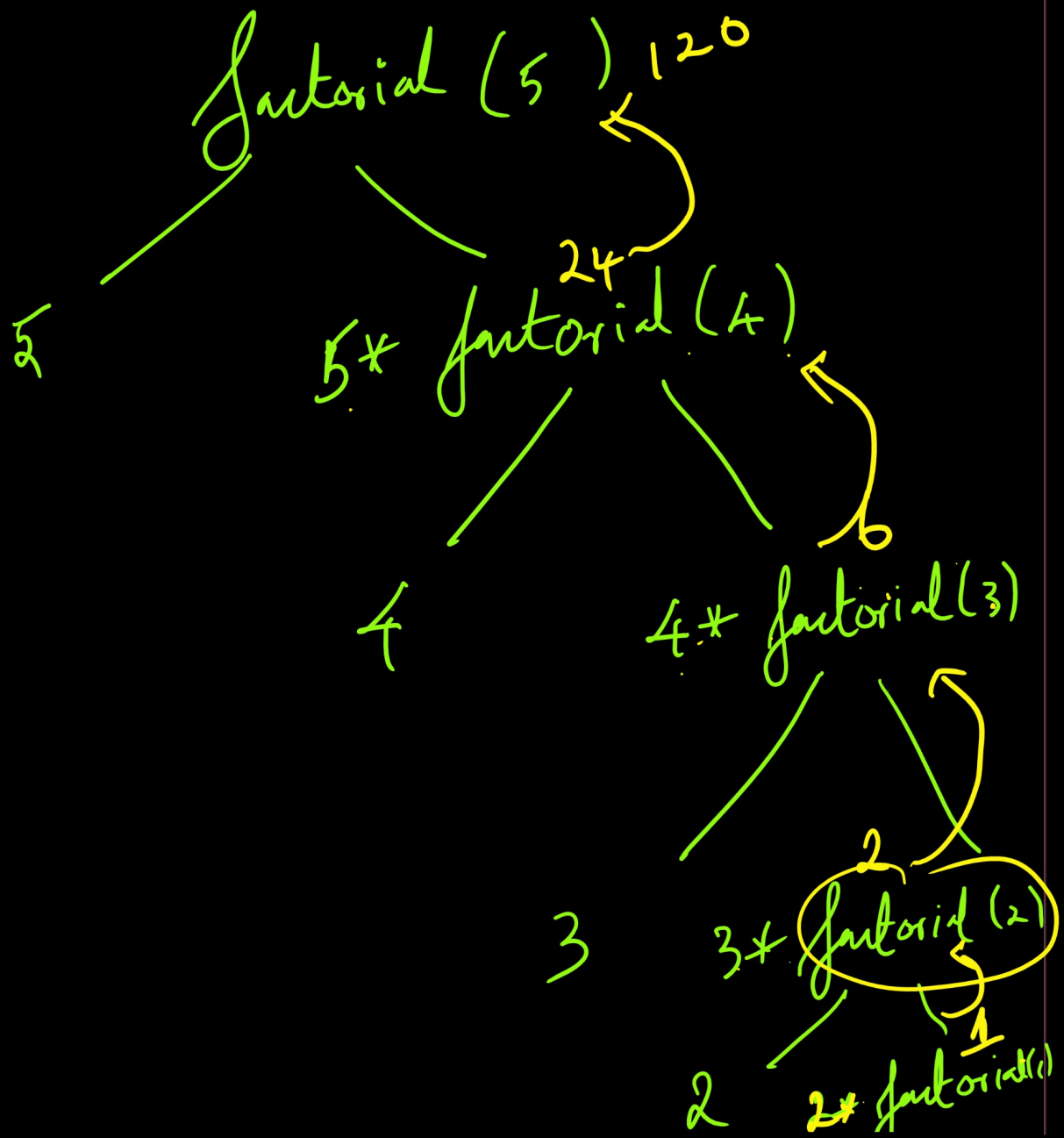


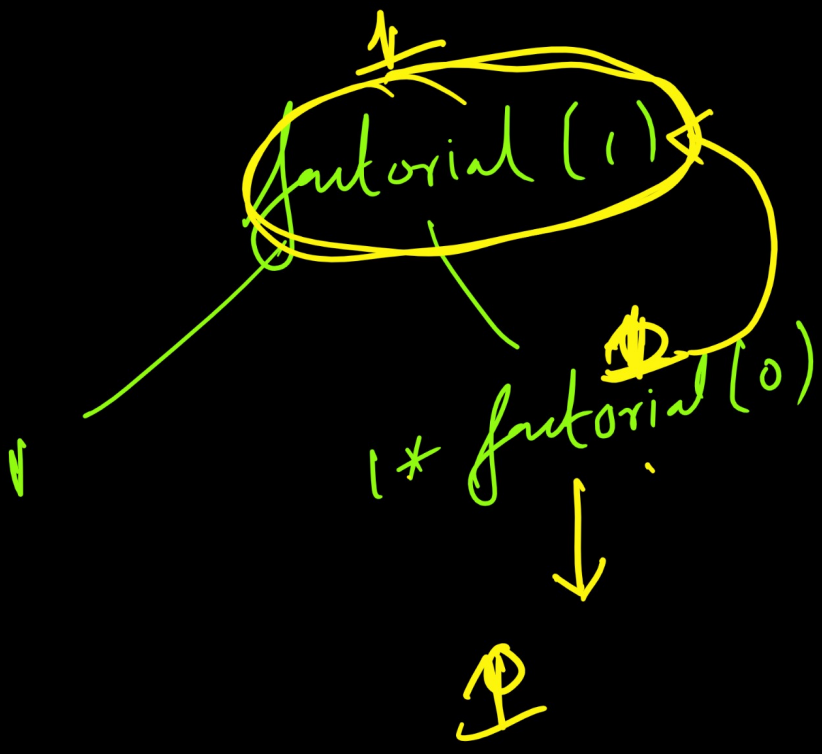
Factorial of a number:


n! = n x (n-1)!
5! = 5 x 4!
4! = 4 x 3!
3! = 3 x 2!
2! = 2 x 1!
1! = 1 x 0!
0! = 1
5 x 24 = 120
4 x 6 = 24
3! = 3 x 2! = 3 x 2 = 6
small subproblem that you know.

```
def factorial(n):
  print("value of n: -->",n)
  if n==0:
    return 1
  return n*factorial(n-1)

ret_val = factorial(5)
print("ret val: -->", ret_val)
```





Sum of first 'n' numbers starting from 1.

Sum of first 5.

sum(5)

5 + sum(n-1)

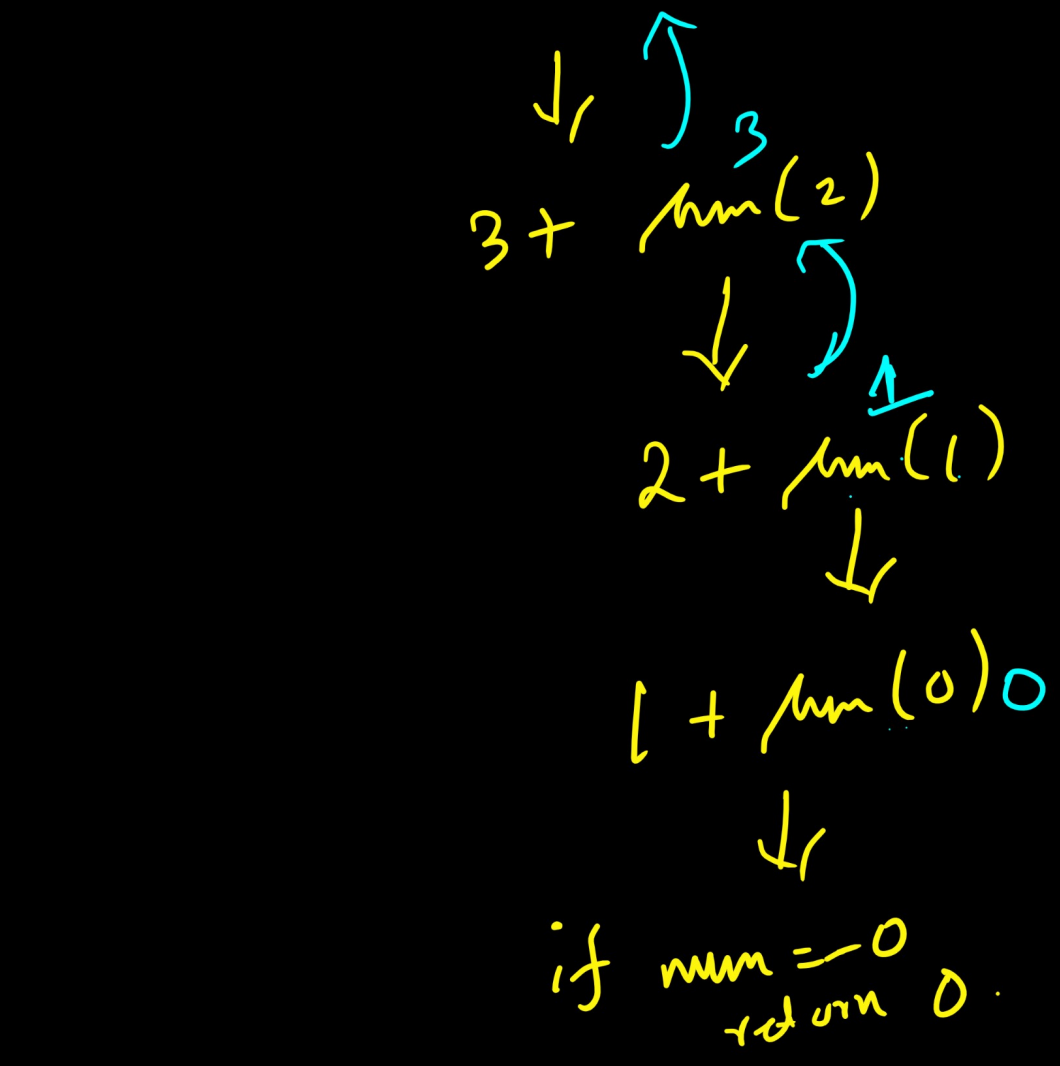
15

5 + sum(4)

10

4 + sum(3)

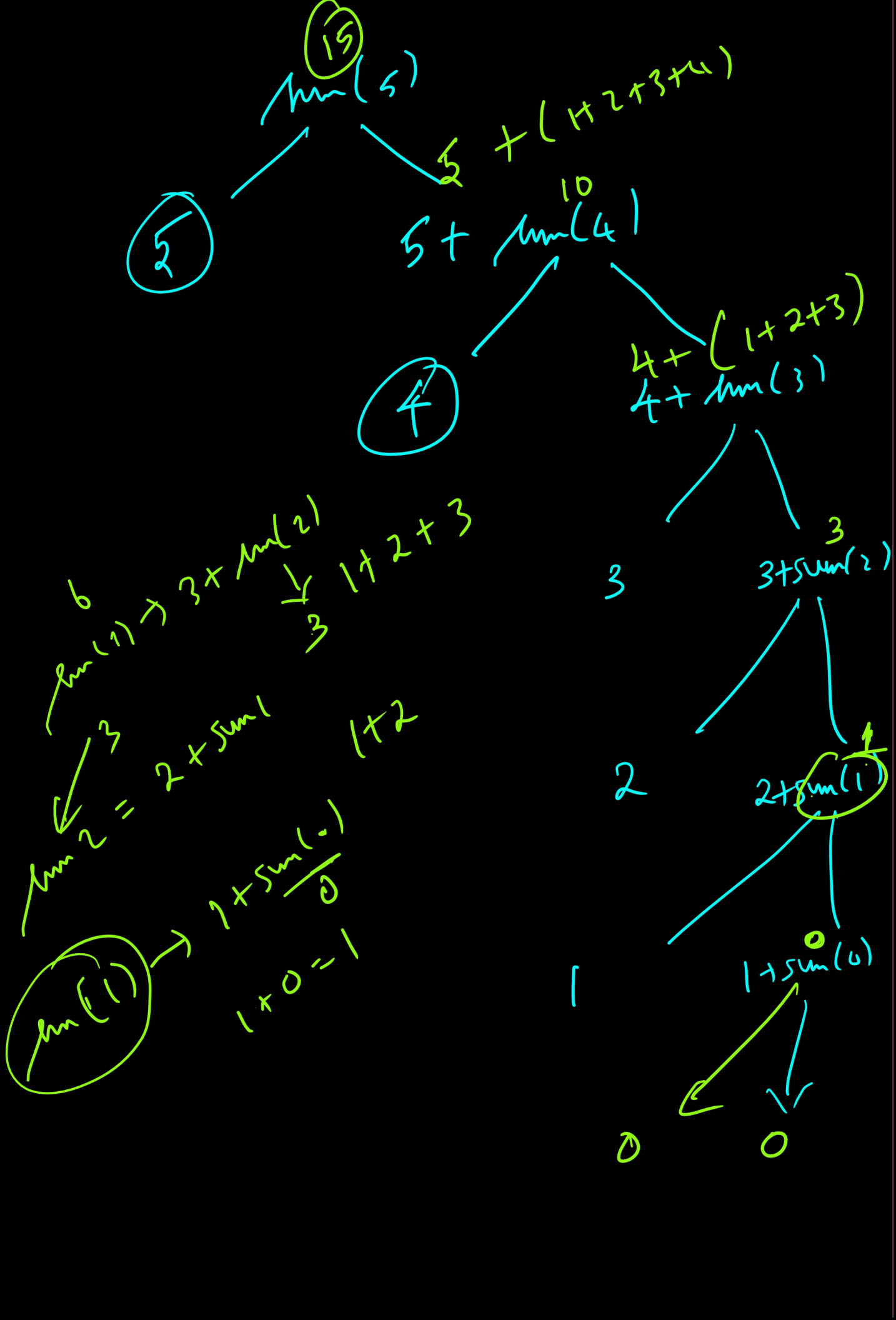
6




```
def sum_n_numbers(n):
  print("n val : -->",n)
  if n==0:
    return 0
  return n+sum_n_numbers(n-1)

ret_val = sum_n_numbers(5)
print(ret_val)
```

sum(5)


15
sum(5)
5 + (1+2+3+4)
5
10
5 + sum(4)
4
4 + (1+2+3)
4 + sum(3)
6
sum(3) → 3 + sum(2)
3
1+2+3
3
3
3+sum(2)
sum 2 = 2 + sum 1
1+2
2
2+sum(1)
sum(1) → 1 + sum(0)
0
1 + 0 = 1
1
1+sum(0)
0
0
1

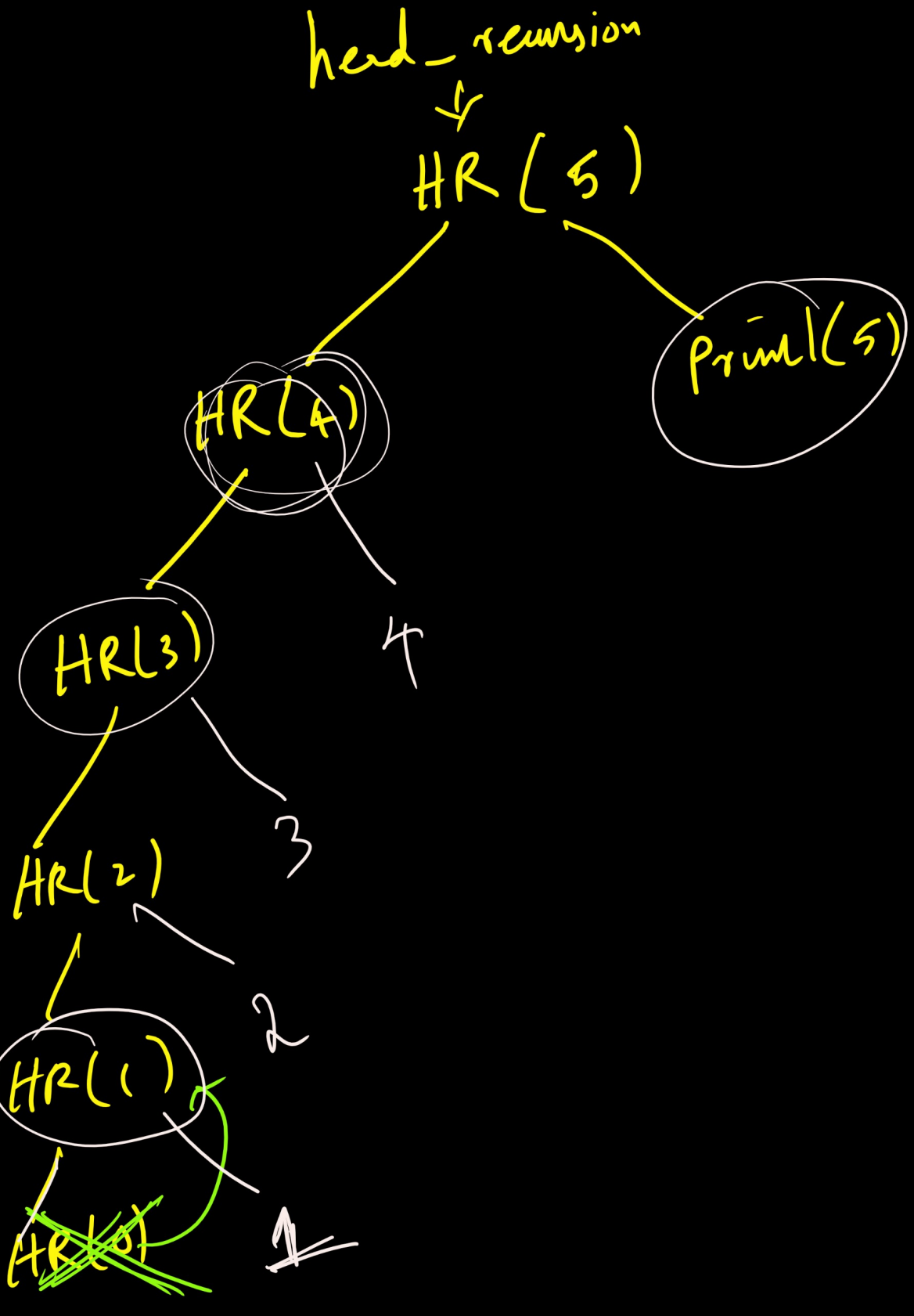
Head Recursion:

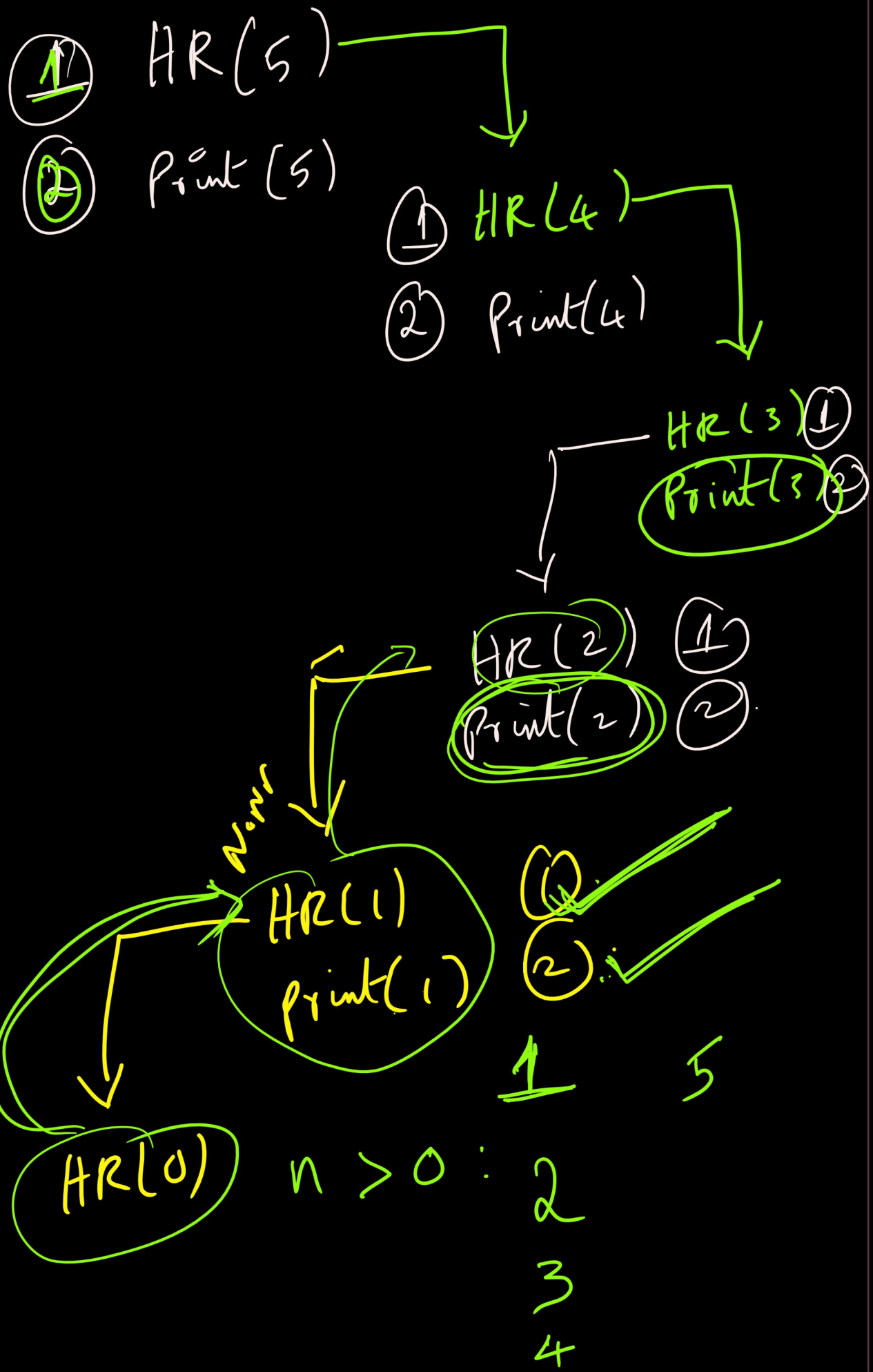
In head recursion, the recursive call is made at the beginning of the function body.

Tail Recursion:

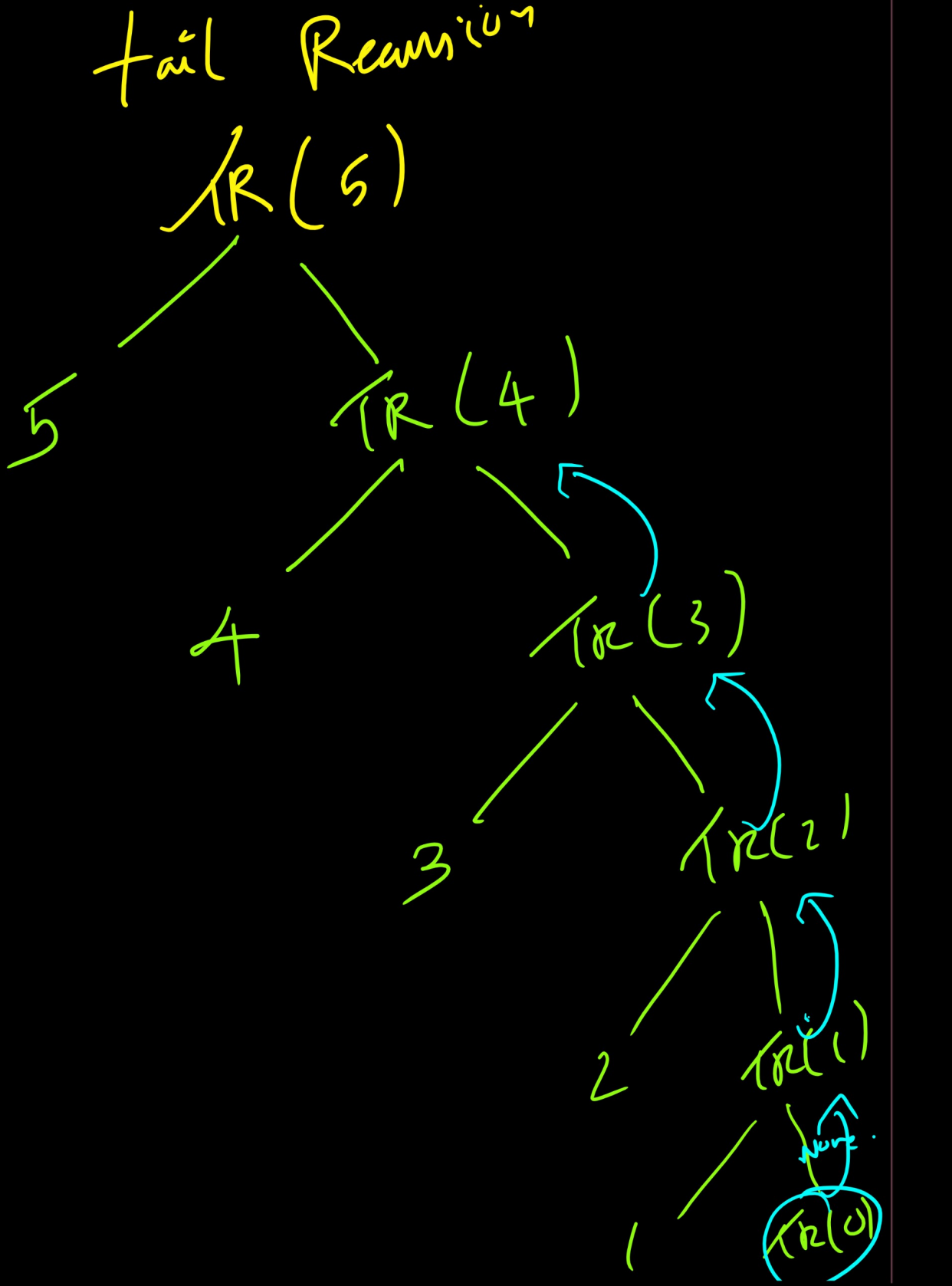
Tail recursion occurs when the recursive call is last thing executed by the function.

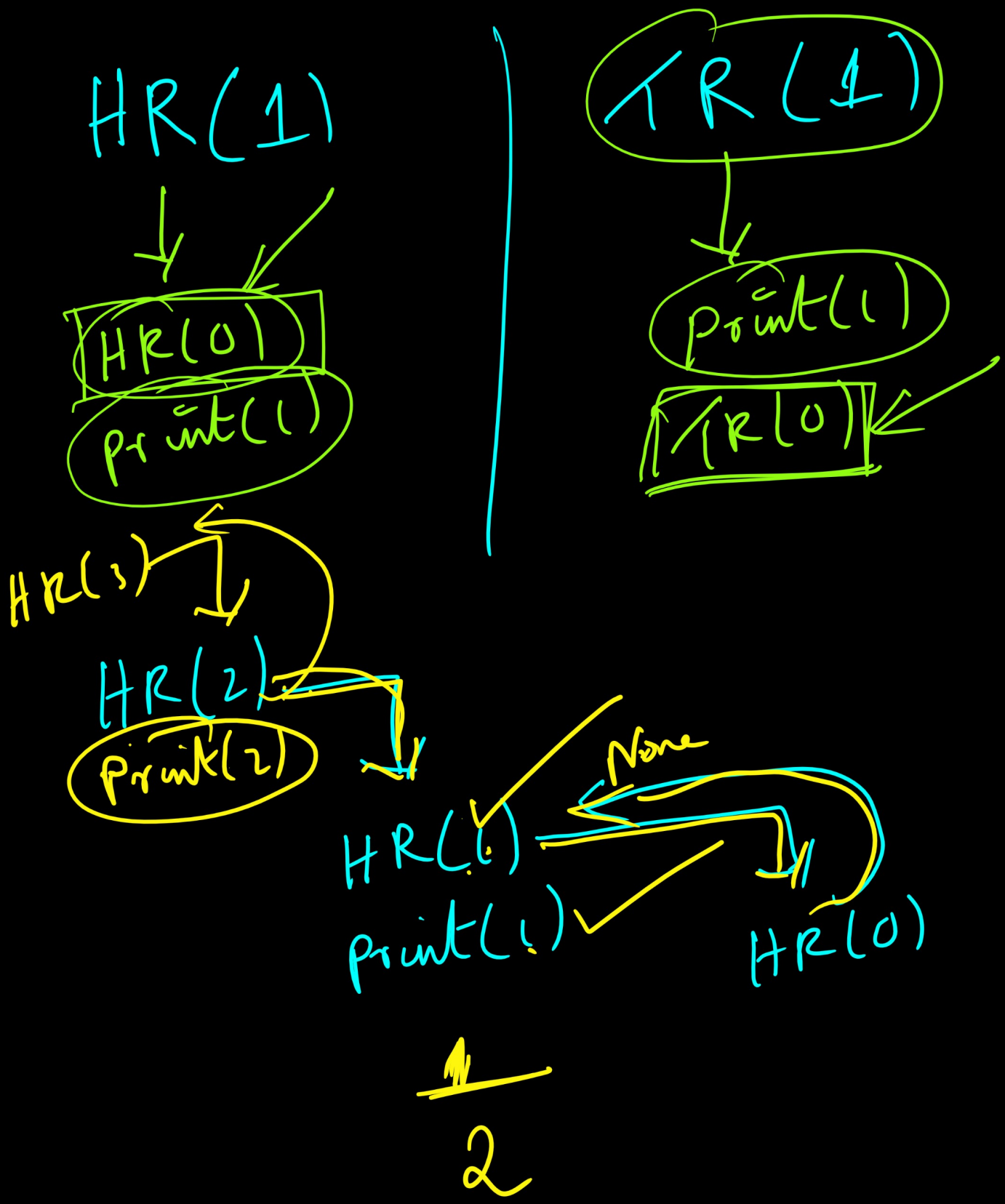
```
def head_recursion(n):
  if n>0:
    head_recursion(n-1)
    print("n val: -->", n)
```




HR(5)
Print(5)
HR(4)
Print(4)
HR(3)
Print(3)
HR(2)
Print(2)
HR(1)
Print(1)
HR(0)
n > 0 :
1
5
2
3
4

```
def tail_recursion(n):
  if n>0:
    print("n val: -->", n)
    tail_recursion(n-1)
```




HR(1)
HR(0)
Print(1)
HR(3)
HR(2)
Print(2)
TR(1)
Print(1)
TR(0)
None
HR(1)
Print(1)
HR(0)

1

2

Finite Recursion:

Finite Recursion terminates after a finite number of recursive calls.

Infinite Recursion: ← base condition won't get satisfied

function calls itself indefinitely.

Direct Recursion:

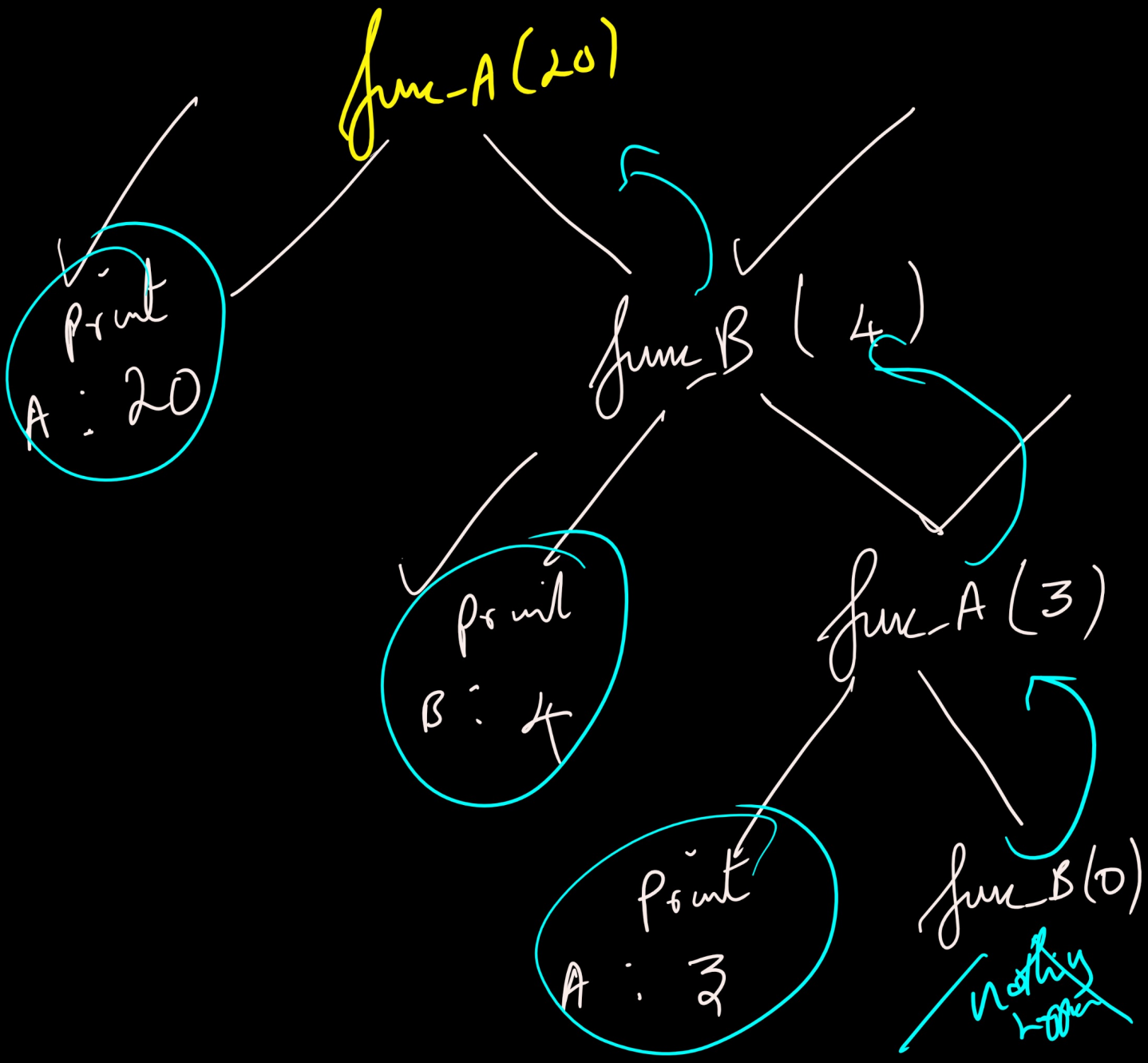
Direct Recursion occurs when a function calls itself directly.

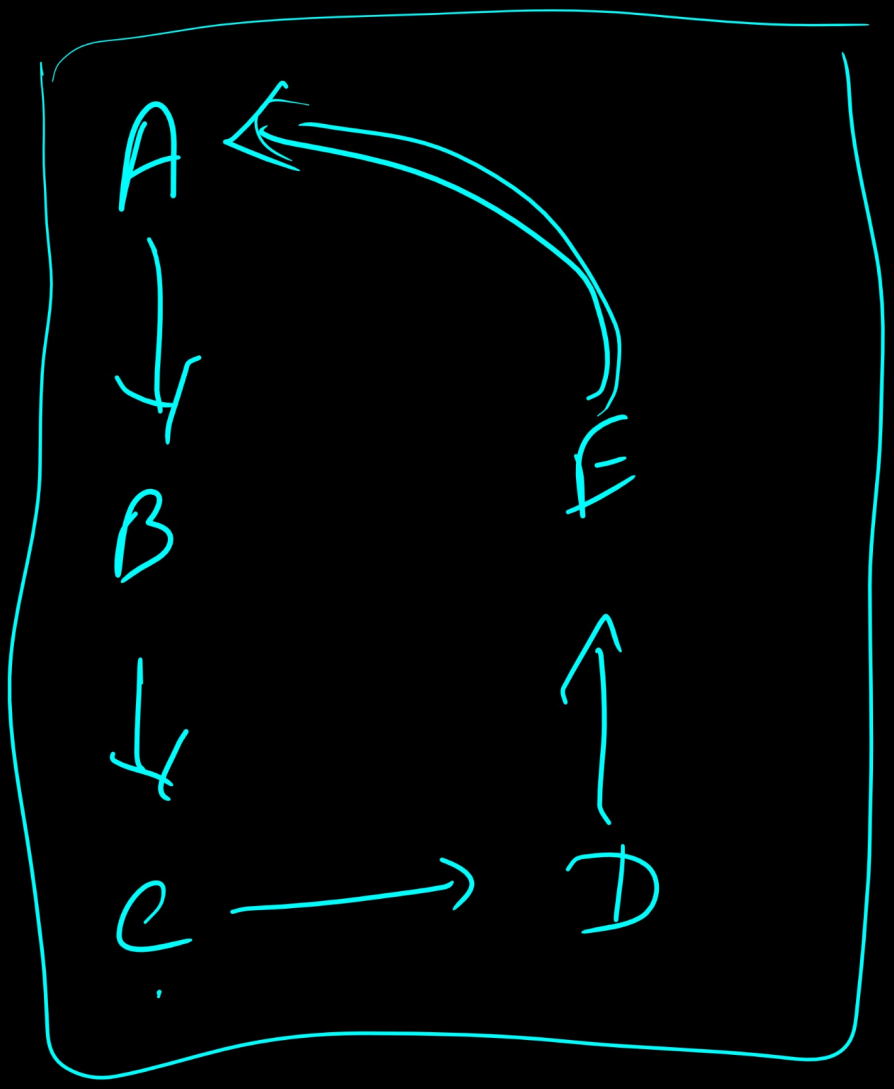
Indirect Recursion:

Indirect recursion occurs when a function calls another function that eventually calls first function again (directly / indirectly).

```
def func_A(n):
  if n>0:
    print("In func A: -->", n)
    func_B(n//5)

def func_B(n):
  if n>0:
    print("In func B: -->", n)
    func_A(n-1)
```



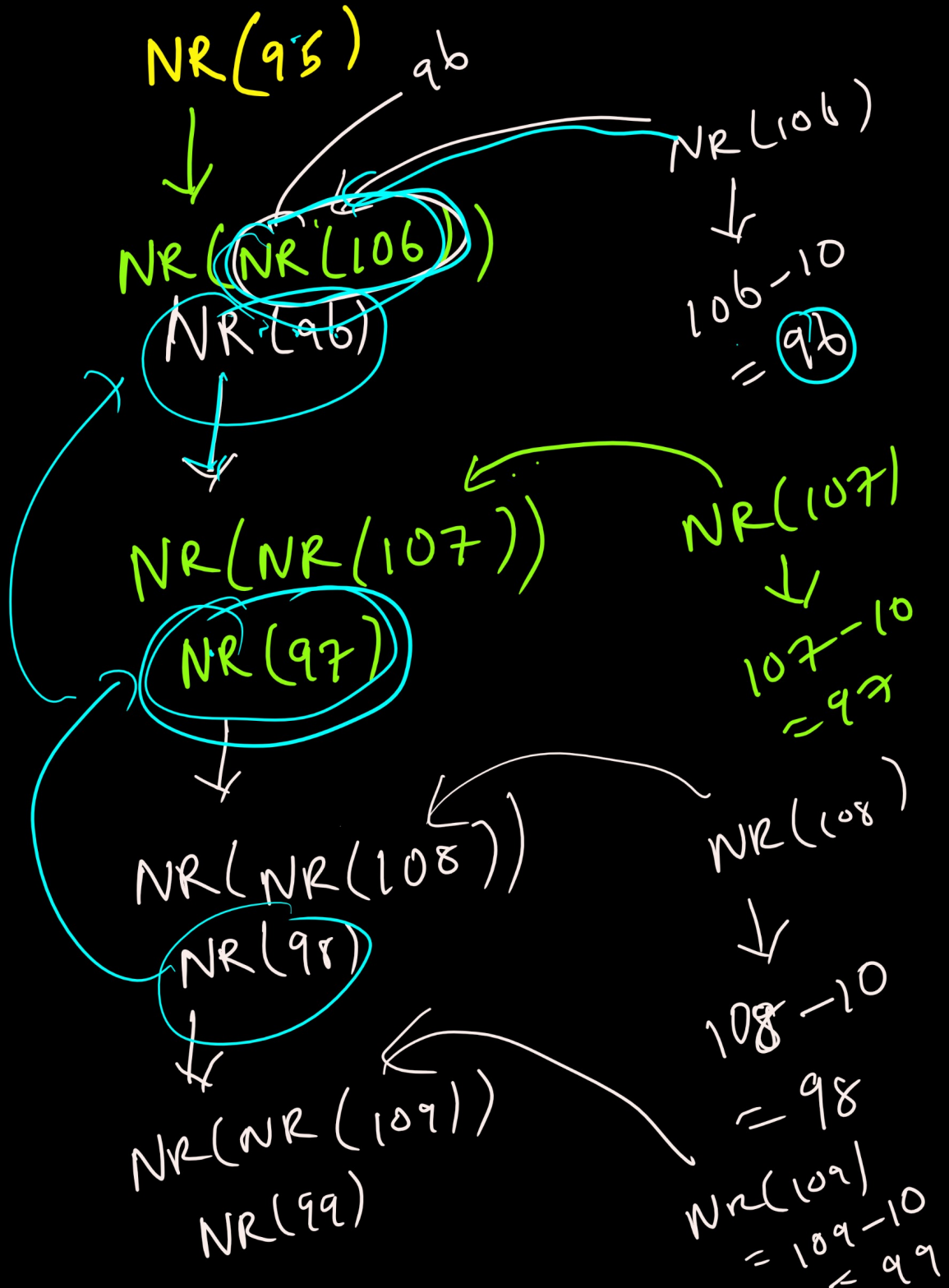


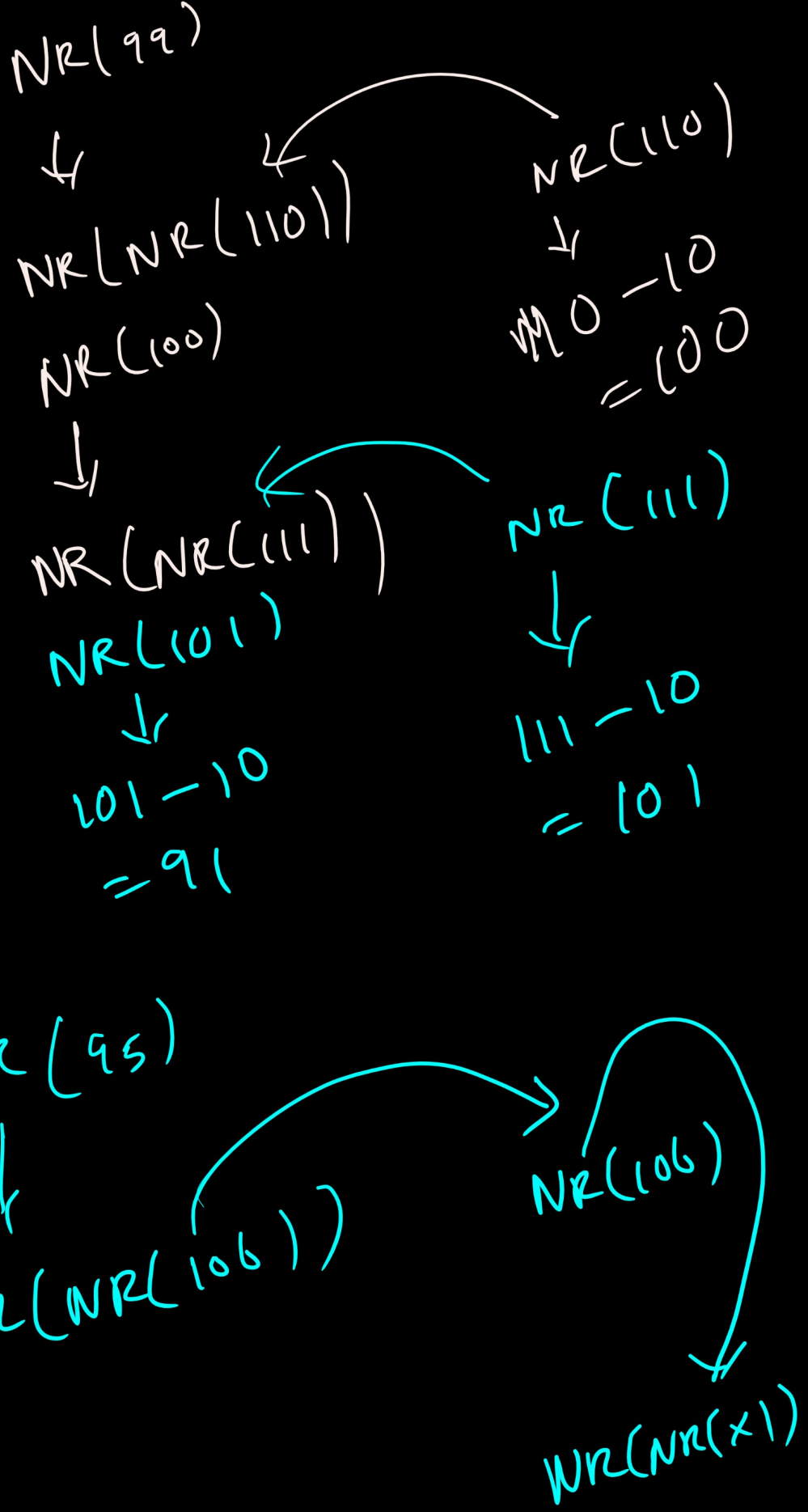
Nested Recursion :-

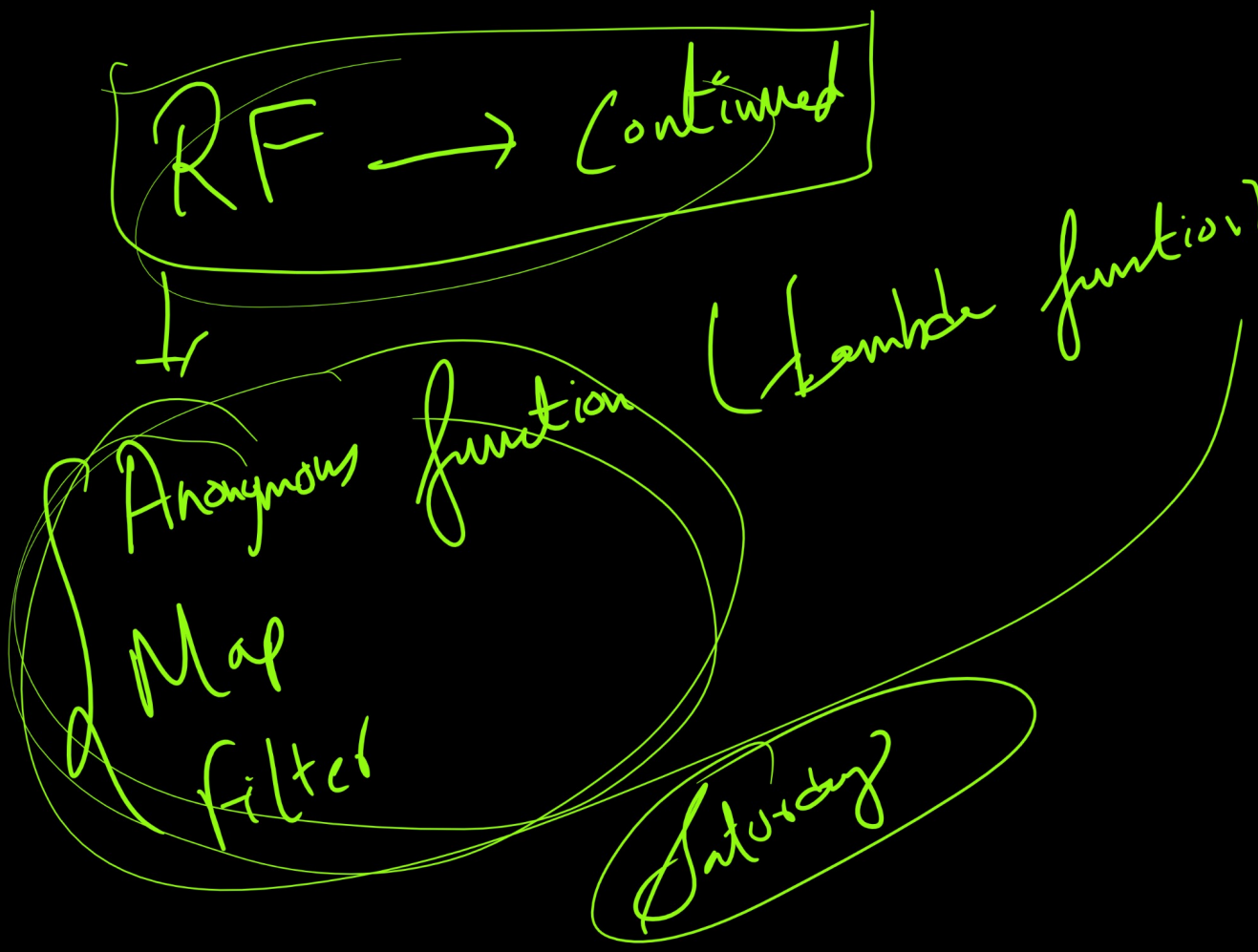
In this recursion, a recursive function will pass the parameter as a recursive call. That means recursion inside recursion.

```
def nested_recursion(n):
  if n>100:
    return n-10
  return(nested_recursion(nested_recursion(n+11))

nested_recursion(95)
```




NR(99)
NR(NR(110))
NR(100)
NR(NR(111))
NR(101)
101-10
=91
NR(110)
110-10
=100
NR(111)
111-10
=101
NR(95)
NR(NR(106))
NR(106)
NR(NR(x1))


RF → Continued
In
Anonymous function (lambda function)
Map
filter
Saturday