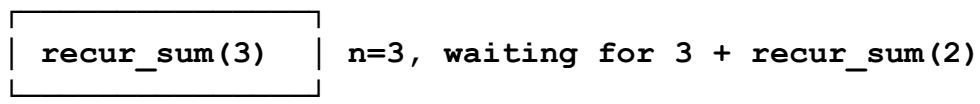
Recursion Note

```
def recur_sum(n):
    if n == 0:
        return 0
    else:
        return n + recur_sum(n-1)
print(recur_sum(3))
```

Call Phase (stack growth) (LIFO, always from top)

Step 1 — call recur_sum(3)

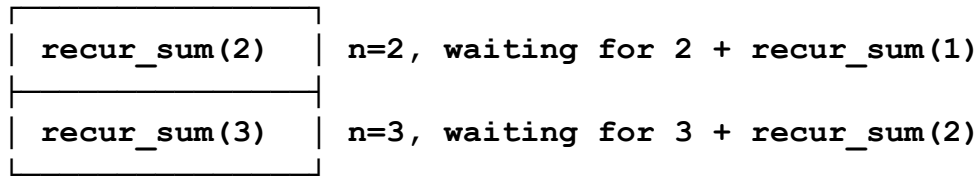
Top →



Bottom

Step 2 — call recur_sum(2)

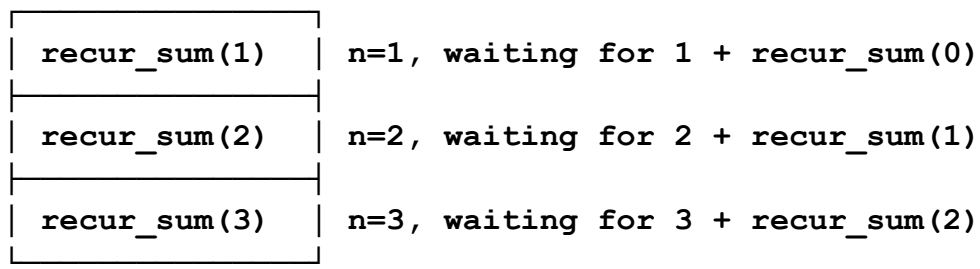
Top →



Bottom

Step 3 — call recur_sum(1)

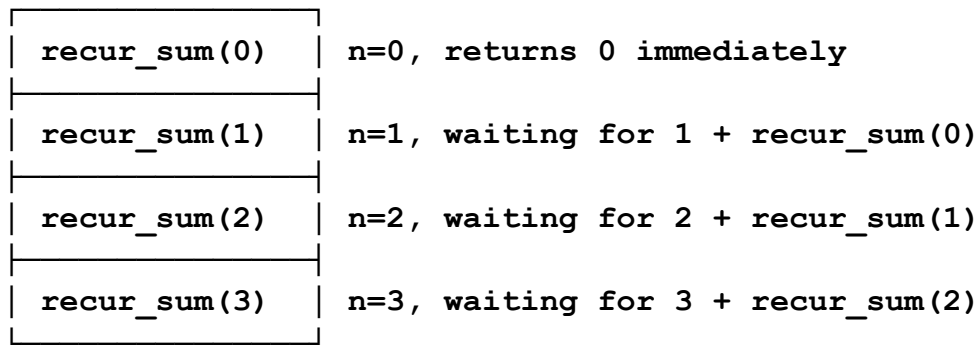
Top →



Bottom

Step 4 — call recur_sum(0) (base case)

Top →

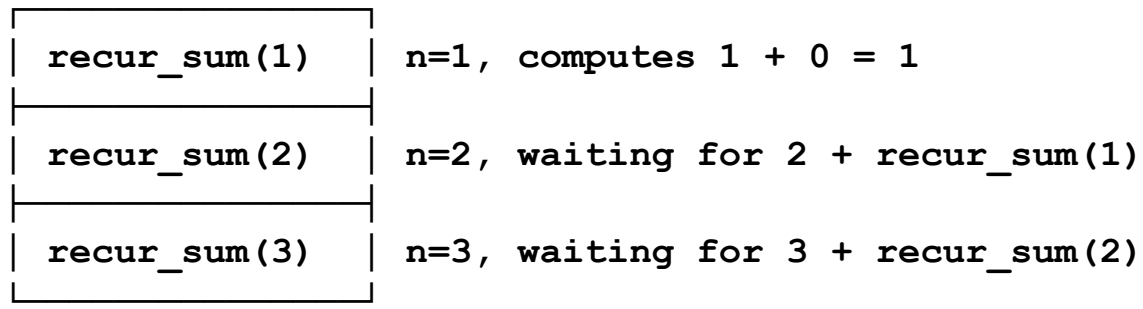


Bottom

Return Phase (stack unwinding)

Step 5 — recur_sum(0) returns 0 → pop

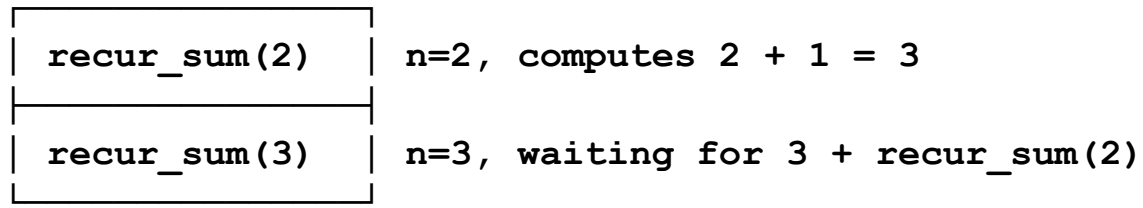
Top →



Bottom

Step 6 — recur_sum(1) returns 1 → pop

Top →



Bottom

Step 7 — recur_sum(2) returns 3 → pop

Top →

Bottom

Step 8 — recur_sum(3) returns 6 → pop

(Empty stack → back to main program, print outputs 6)

Summary

Call phase (push): 3 → 2 → 1 → 0
Return phase (pop): 0 → 1 → 2 → 3

At each return, the frame adds its own n to the value returned from deeper down.

Final result = 6.