

# 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 →

<code>recur_sum(3)</code>	<code>n=3, waiting for 3 + recur_sum(2)</code>
---------------------------	--

Bottom

Step 2 – call `recur_sum(2)`

Top →

<code>recur_sum(2)</code>	<code>n=2, waiting for 2 + recur_sum(1)</code>
<code>recur_sum(3)</code>	<code>n=3, waiting for 3 + recur_sum(2)</code>

Bottom

Step 3 – call `recur_sum(1)`

Top →

<code>recur_sum(1)</code>	<code>n=1, waiting for 1 + recur_sum(0)</code>
<code>recur_sum(2)</code>	<code>n=2, waiting for 2 + recur_sum(1)</code>
<code>recur_sum(3)</code>	<code>n=3, waiting for 3 + recur_sum(2)</code>

Bottom

Step 4 – call `recur_sum(0)` (base case)

Top →

<code>recur_sum(0)</code>	<code>n=0, returns 0 immediately</code>
<code>recur_sum(1)</code>	<code>n=1, waiting for 1 + recur_sum(0)</code>
<code>recur_sum(2)</code>	<code>n=2, waiting for 2 + recur_sum(1)</code>
<code>recur_sum(3)</code>	<code>n=3, waiting for 3 + recur_sum(2)</code>

Bottom

## Return Phase (stack unwinding)

Step 5 – `recur_sum(0)` returns 0 → pop

Top →

<code>recur_sum(1)</code>	<code>n=1, computes <math>1 + 0 = 1</math></code>
<code>recur_sum(2)</code>	<code>n=2, waiting for <math>2 + \text{recur\_sum}(1)</math></code>
<code>recur_sum(3)</code>	<code>n=3, waiting for <math>3 + \text{recur\_sum}(2)</math></code>

Bottom

Step 6 – `recur_sum(1)` returns 1 → pop

Top →

<code>recur_sum(2)</code>	<code>n=2, computes <math>2 + 1 = 3</math></code>
<code>recur_sum(3)</code>	<code>n=3, waiting for <math>3 + \text{recur\_sum}(2)</math></code>

Bottom

Step 7 – `recur_sum(2)` returns 3 → pop

Top →

<code>recur_sum(3)</code>	<code>n=3, computes <math>3 + 3 = 6</math></code>
---------------------------	---

Bottom

Step 8 – `recur_sum(3)` returns 6 → pop

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

## Summary

Call phase (push):  $3 \rightarrow 2 \rightarrow 1 \rightarrow 0$

Return phase (pop):  $0 \rightarrow 1 \rightarrow 2 \rightarrow 3$

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

Final result = 6.