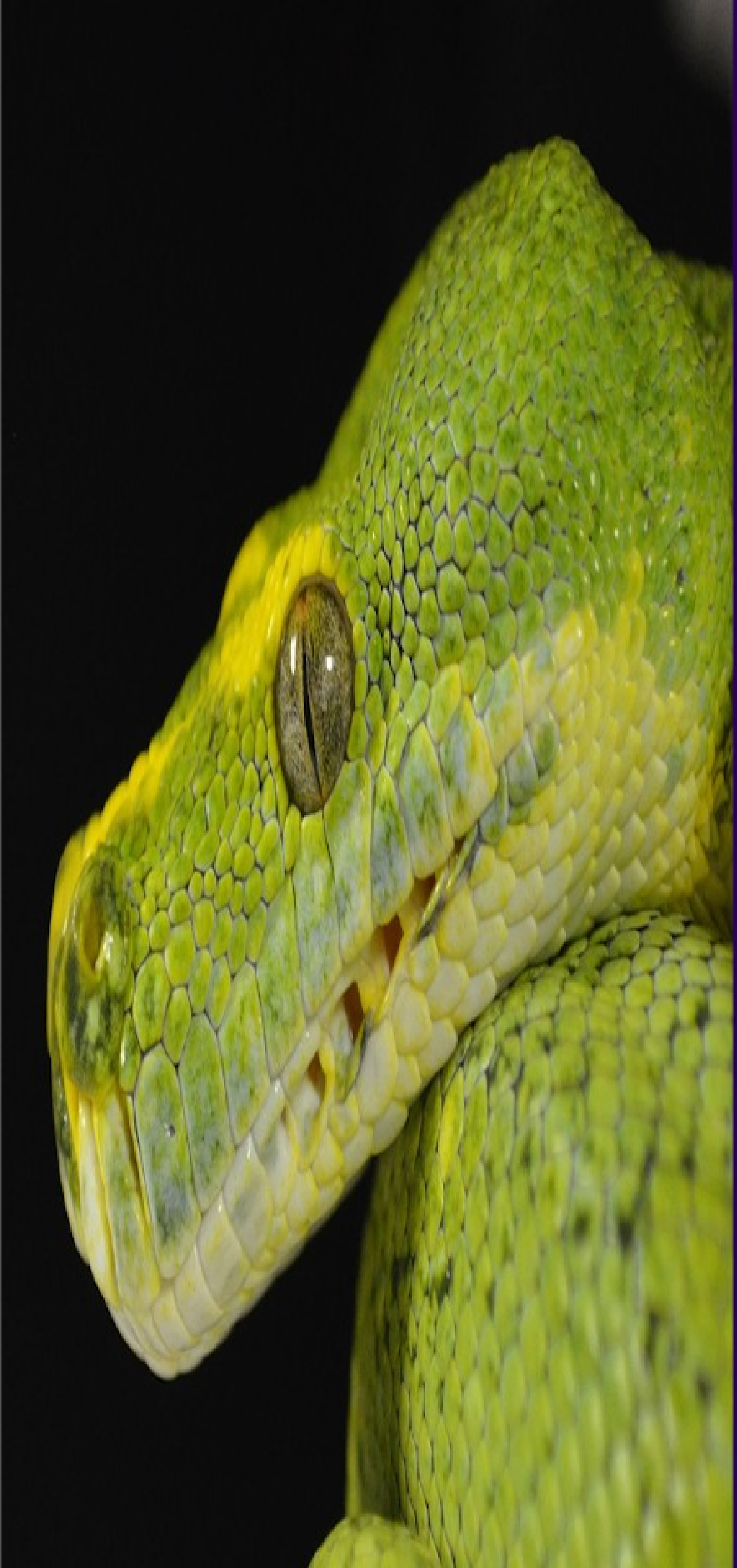


Python

Recursion

Asha Tutorials

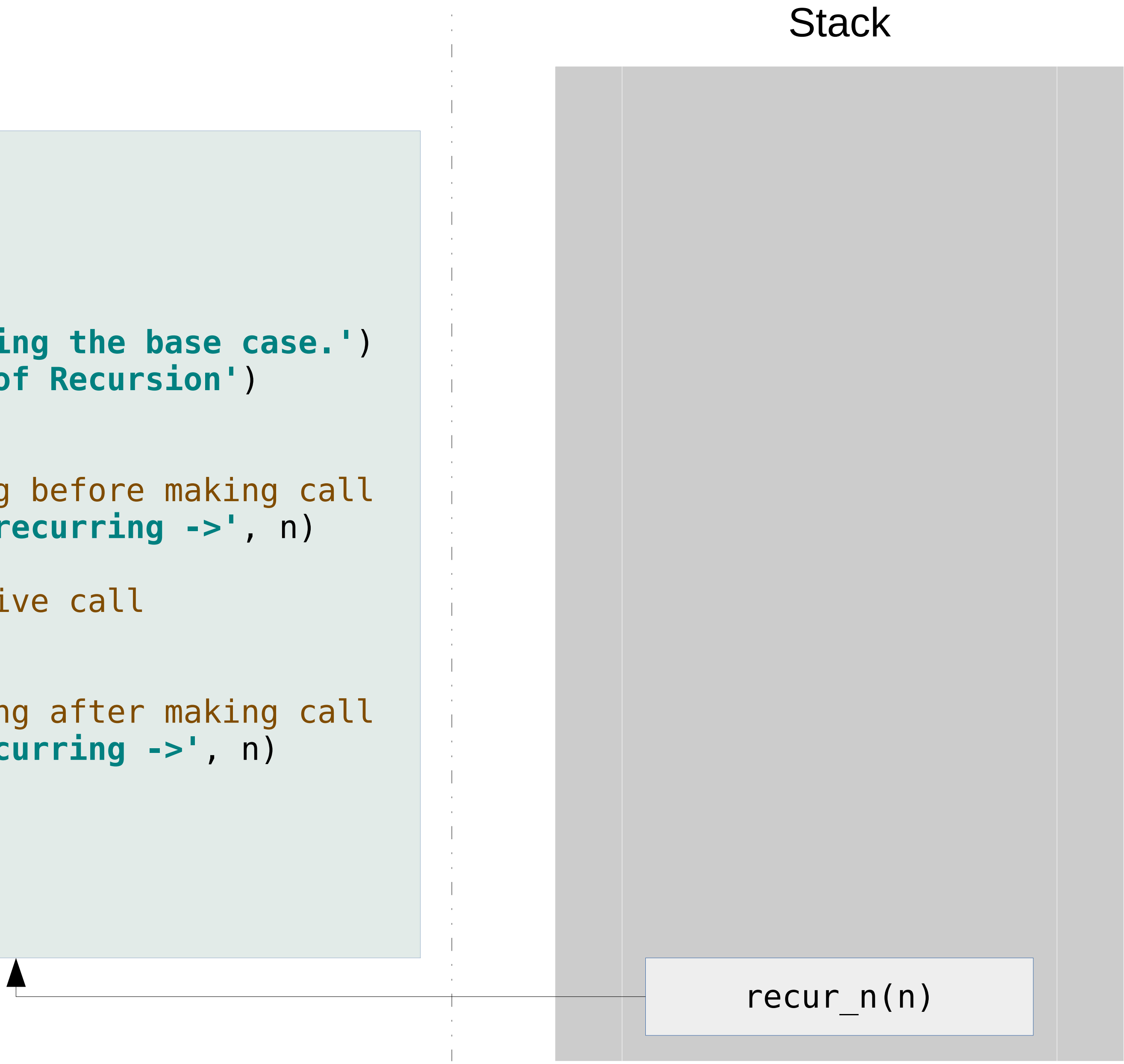
Recursion

Code :

```
def recur_n(n):
    """ trace the execution sequence of n recurrence steps """

    if n == 0:
        print('Solving the base case. End of recursive calls.')
        return

    print('Started recurring ->', n)

    recur_n(n-1)

    print('Ended recurring ->', n)


recur_n(5)
```

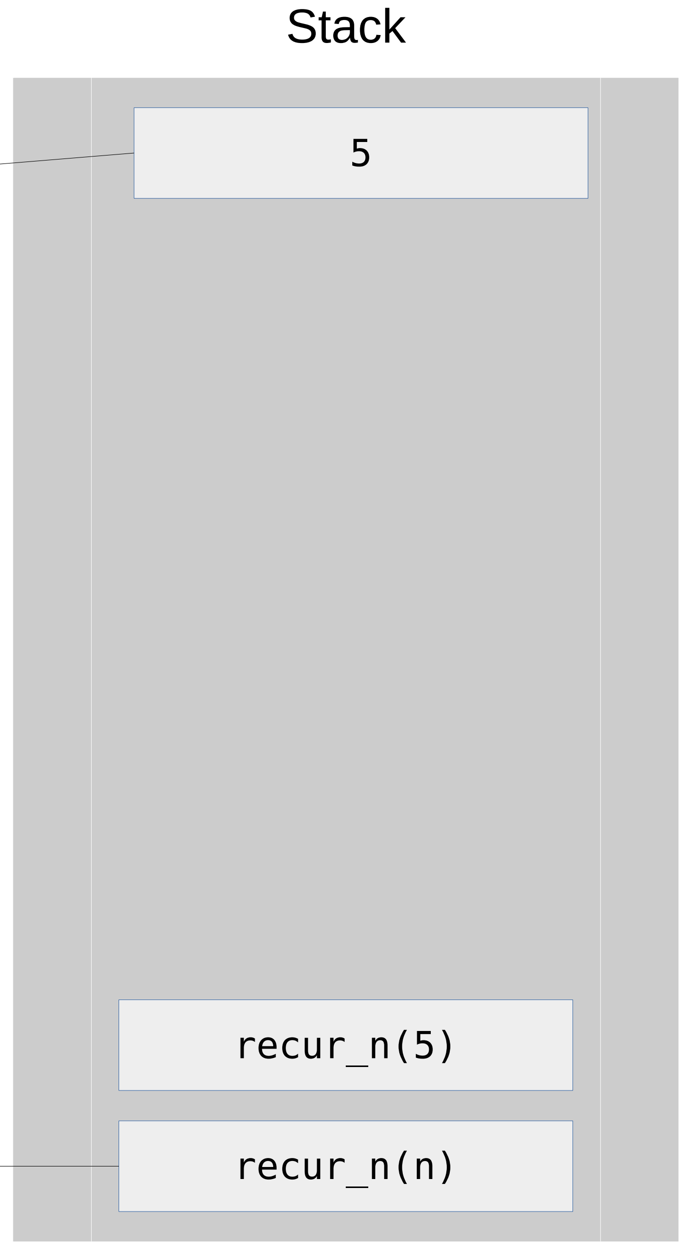
Stack Trace

```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



Stack Trace

```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



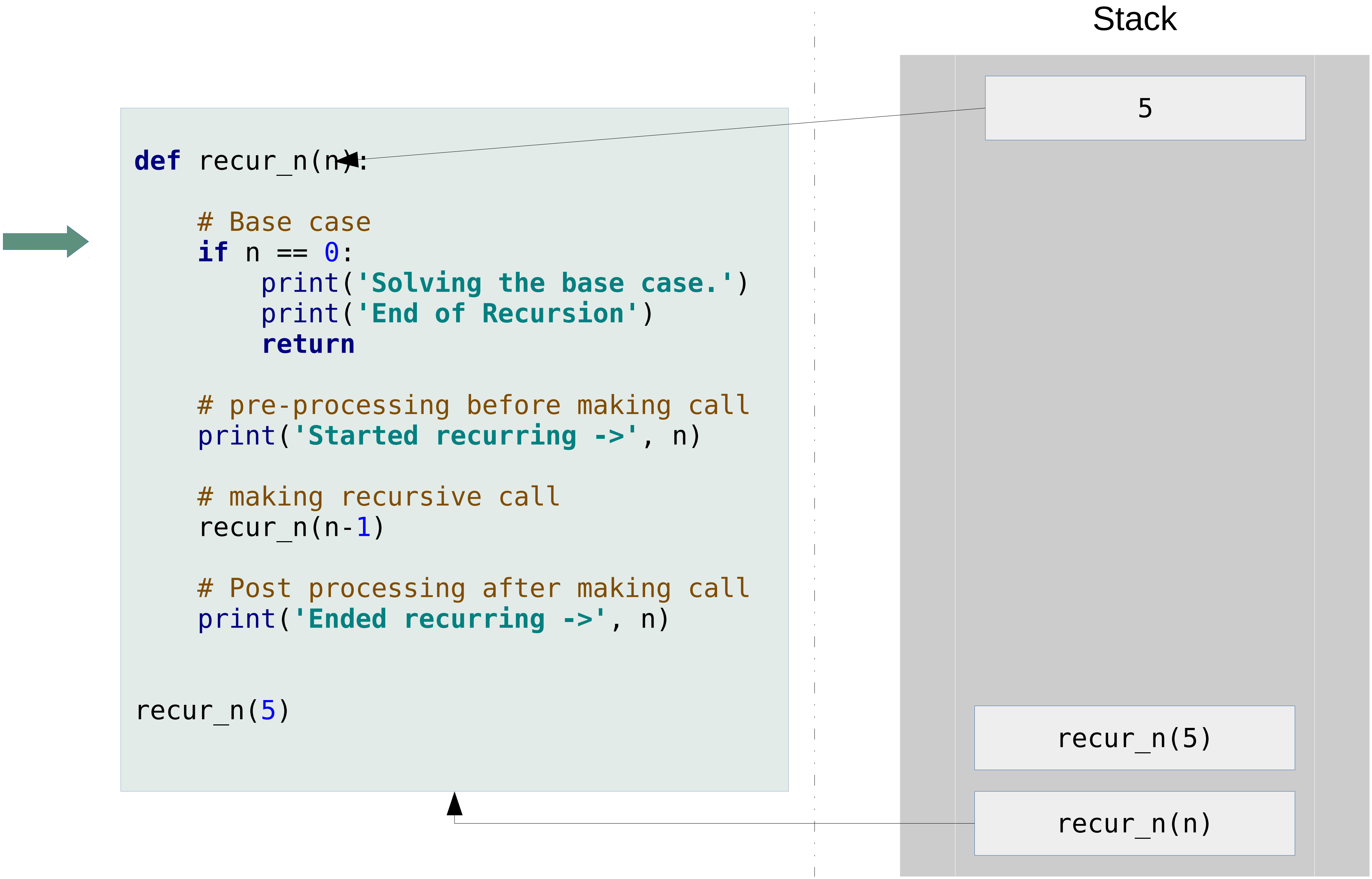
Stack

5

recur_n(5)

recur_n(n)

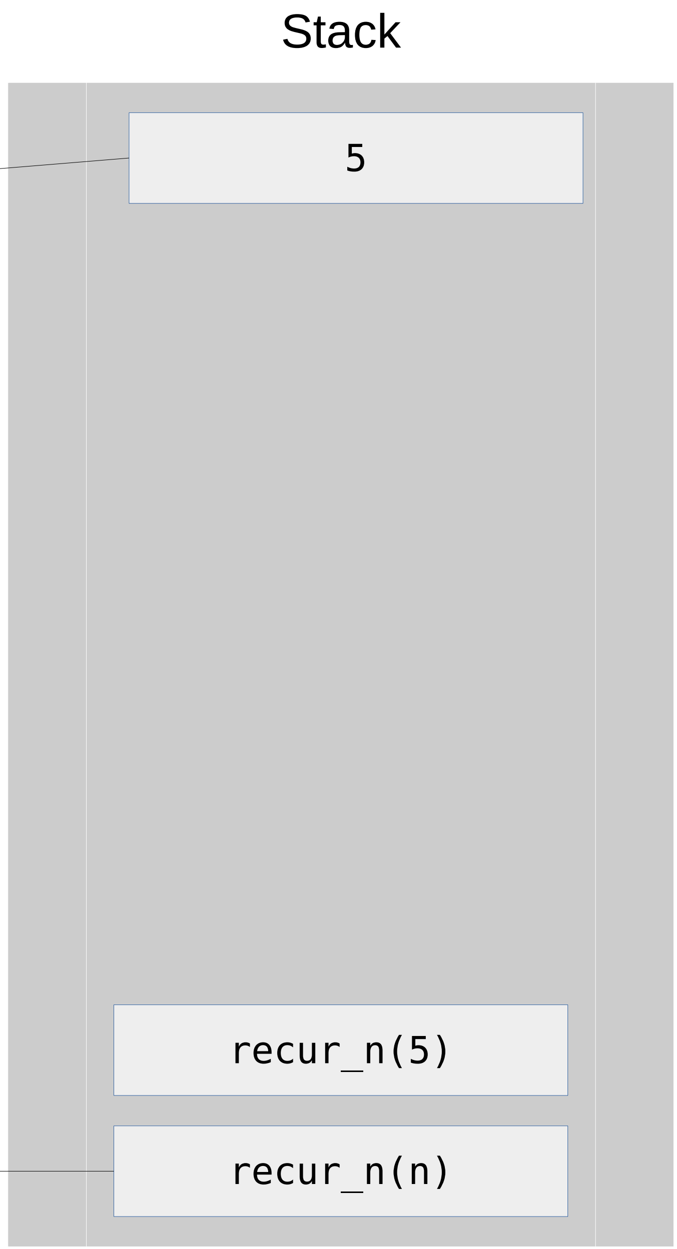
Stack Trace



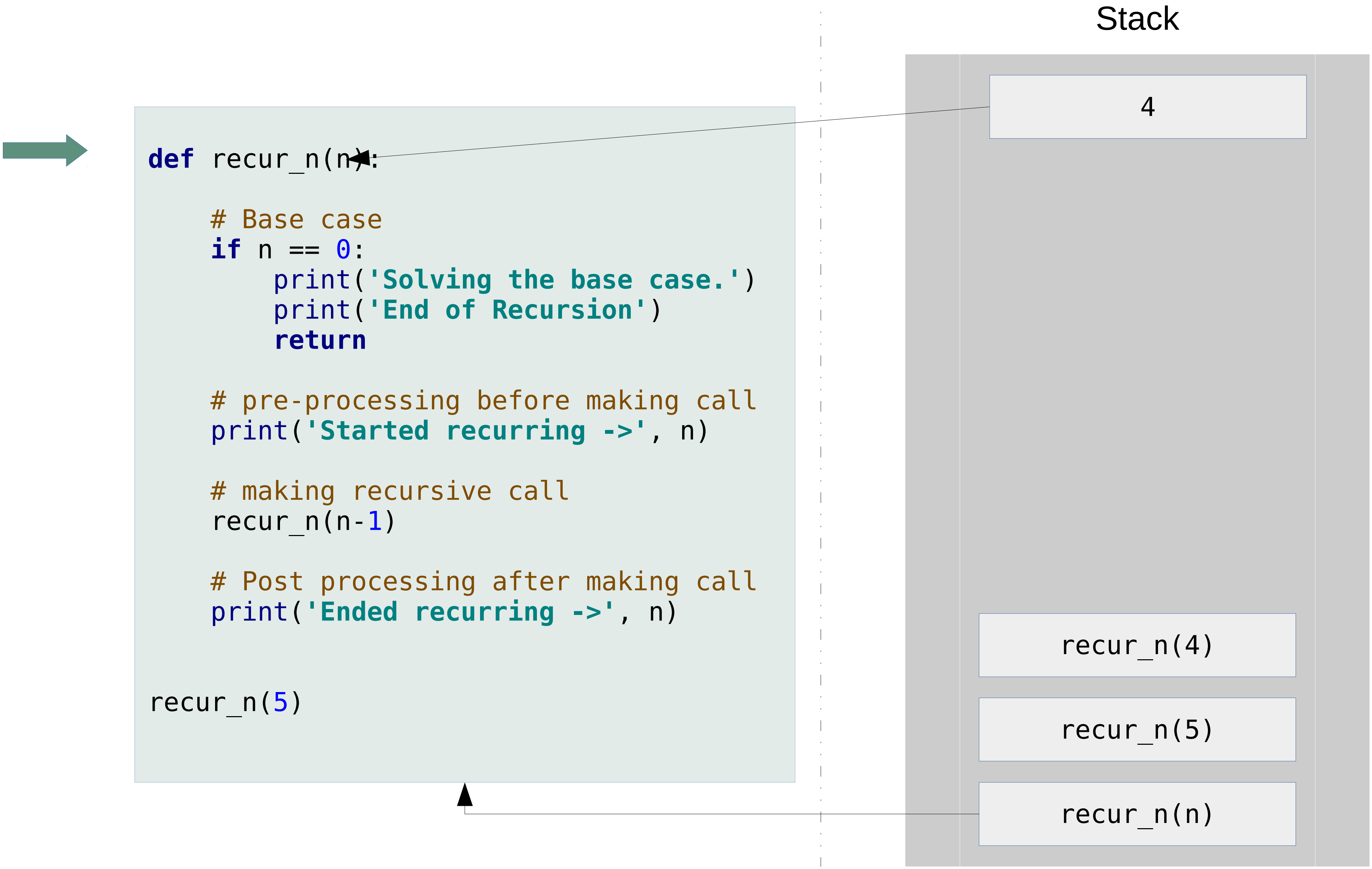
```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```

Stack Trace

```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



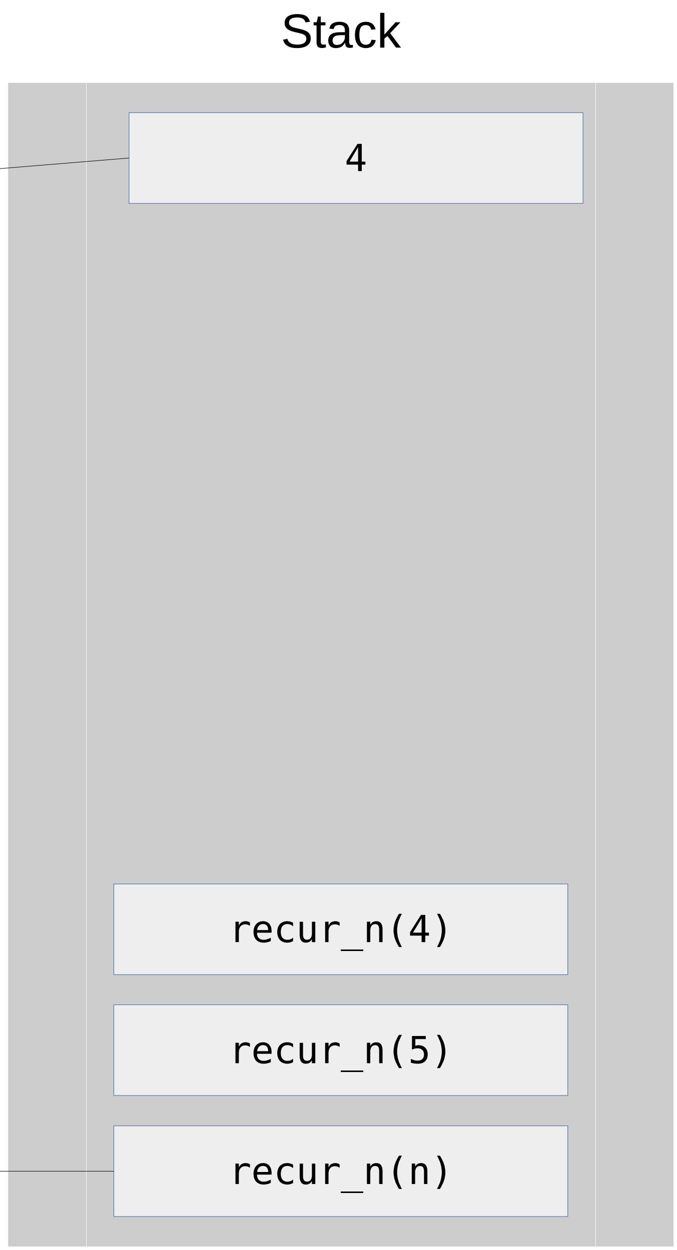
Stack Trace



```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```

Stack Trace

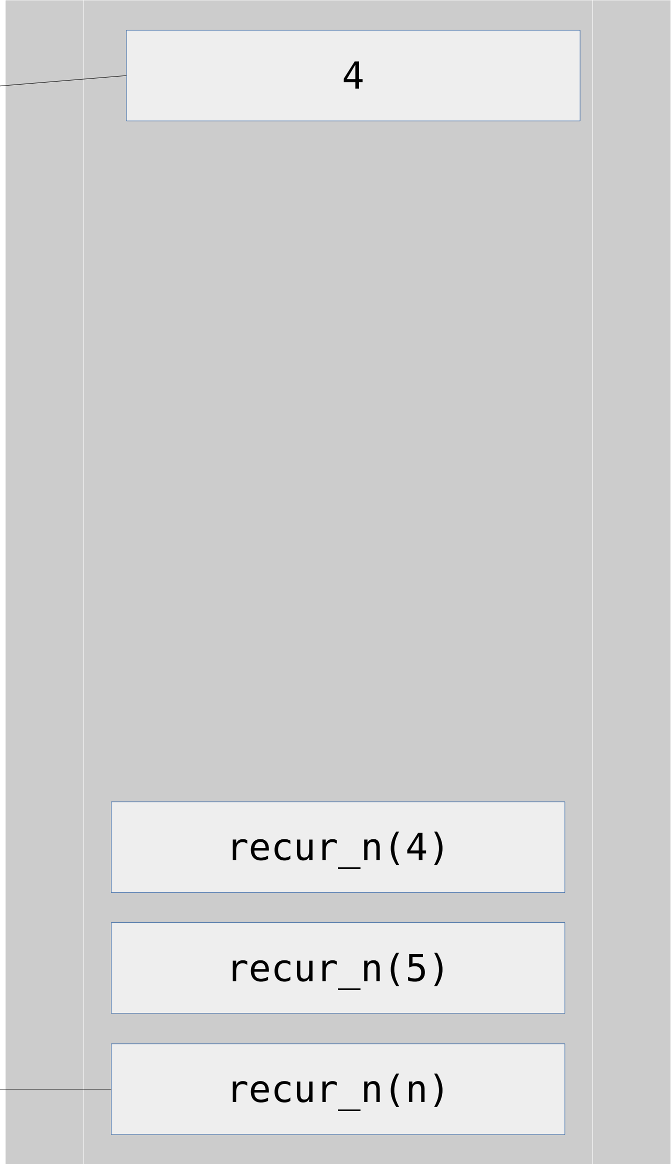
```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



Stack Trace

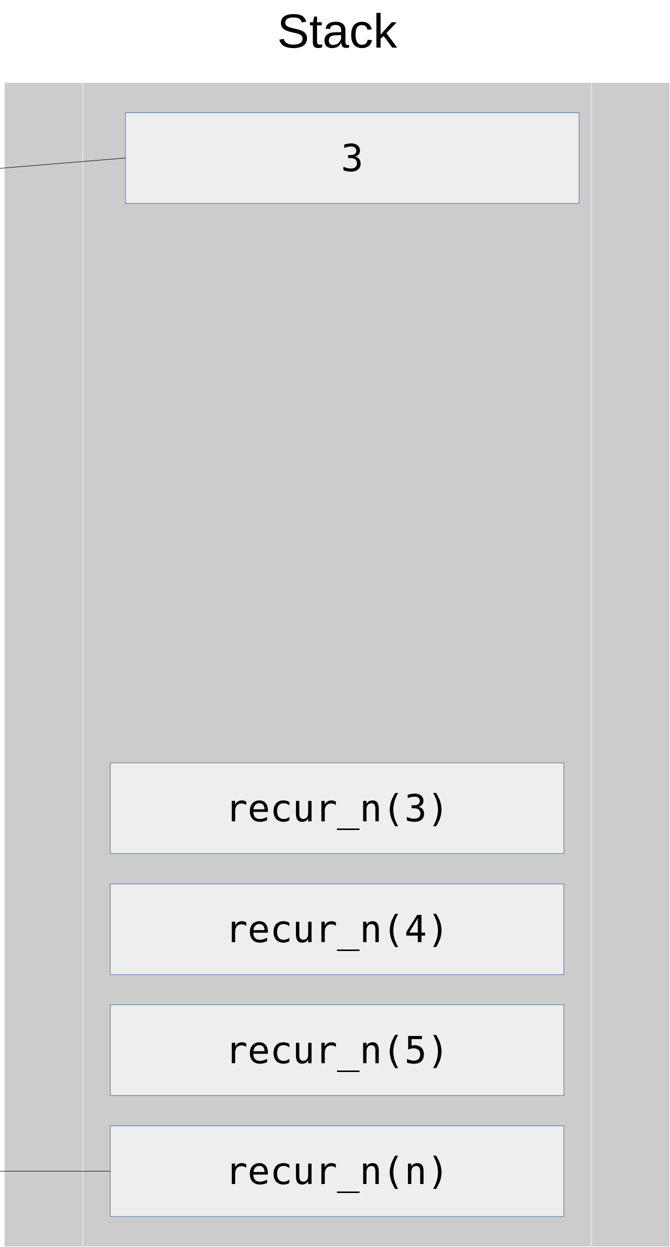
```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```

Stack



Stack Trace

```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



Stack

3

recur_n(3)

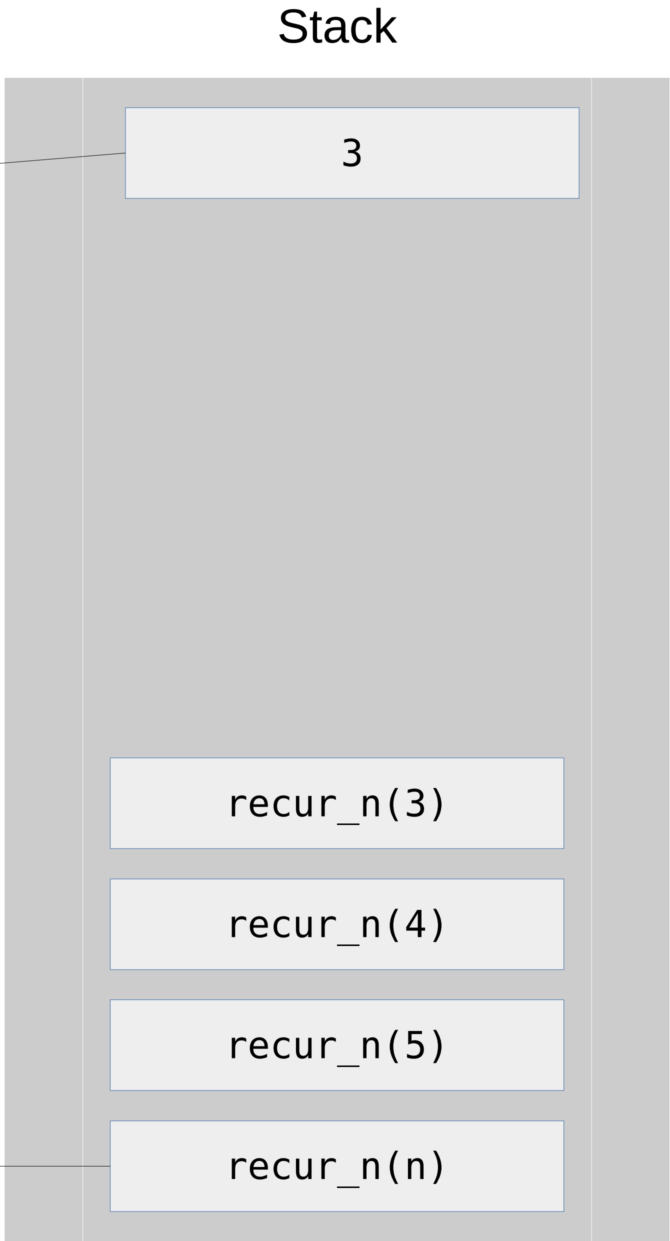
recur_n(4)

recur_n(5)

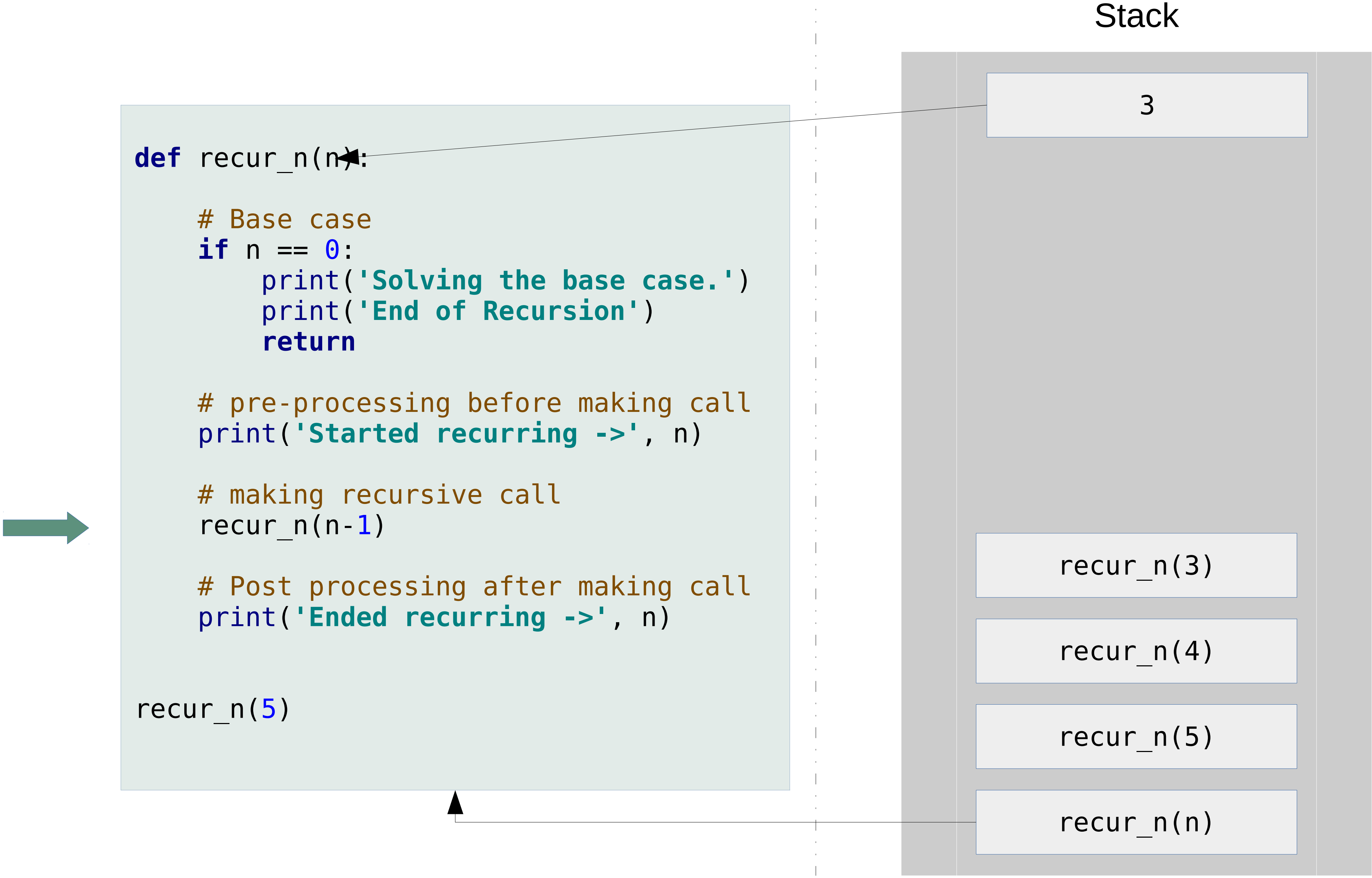
recur_n(n)

Stack Trace

```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



Stack Trace



```python
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```

Stack

3

recur_n(3)

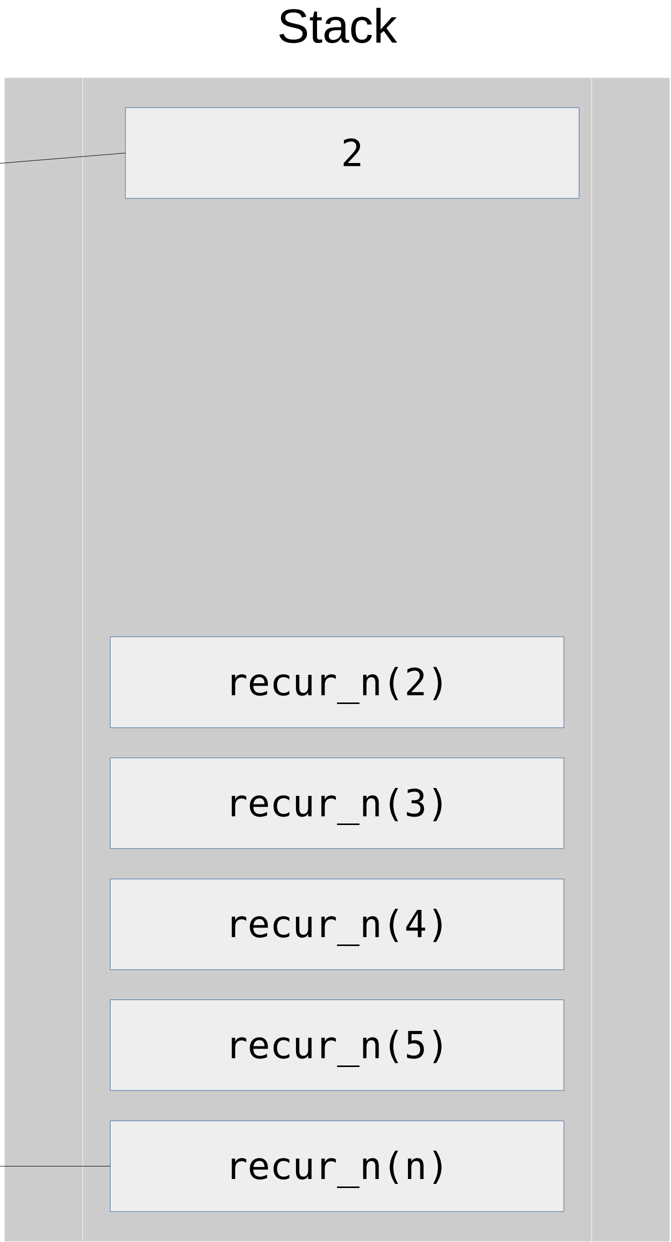
recur_n(4)

recur_n(5)

recur_n(n)

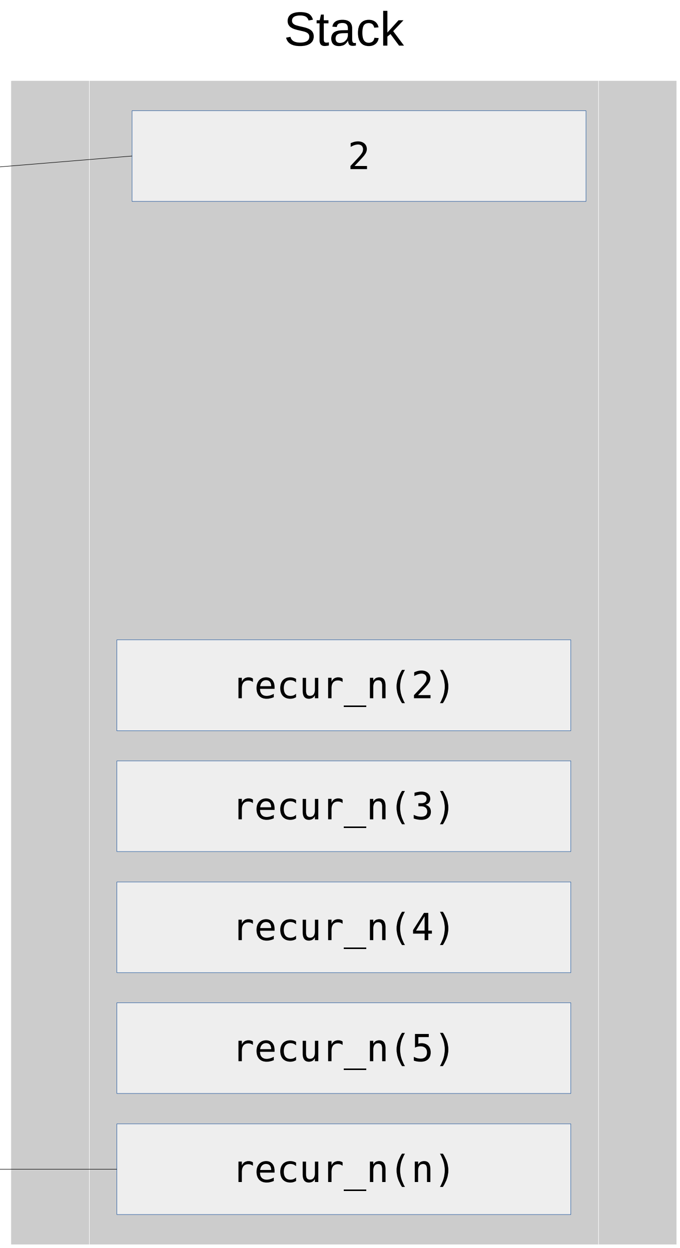
Stack Trace

```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



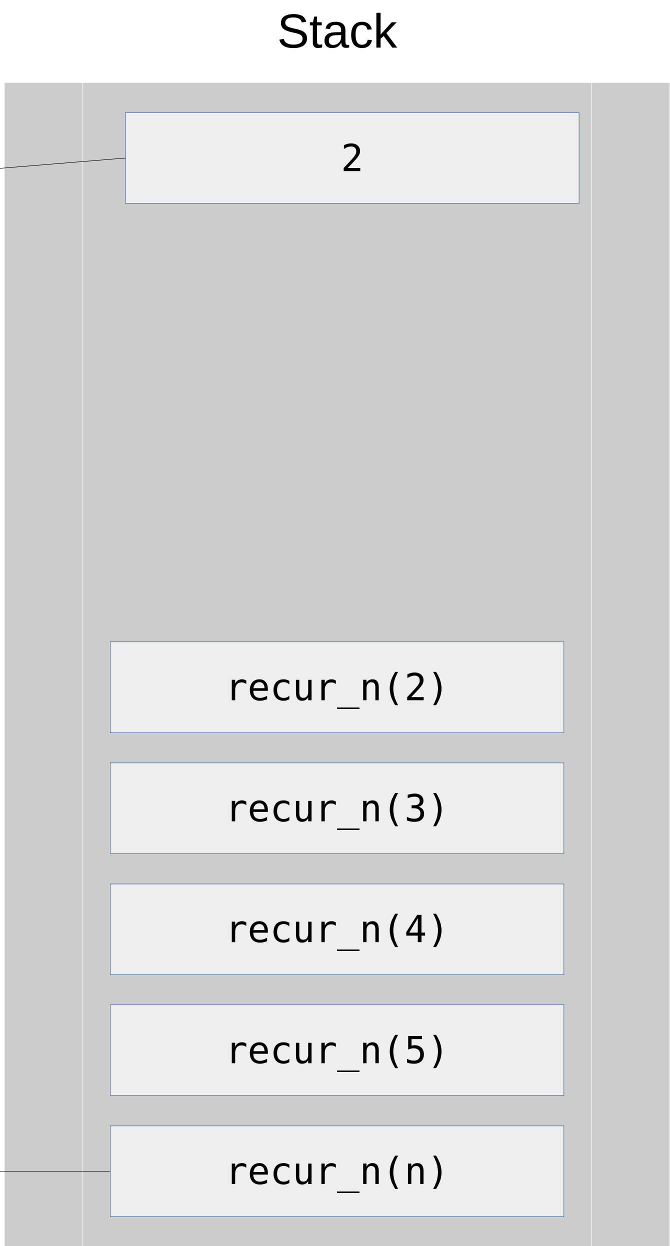
Stack Trace

```python
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



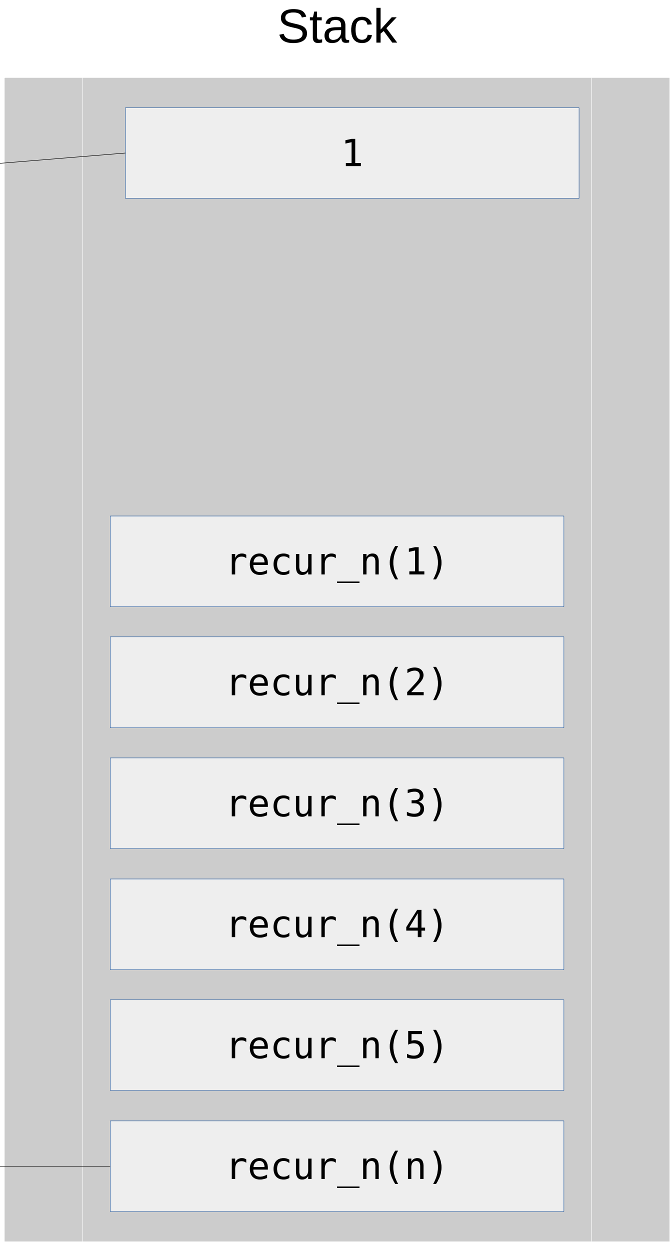
Stack Trace

```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



Stack Trace

```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



Stack

1

recur_n(1)

recur_n(2)

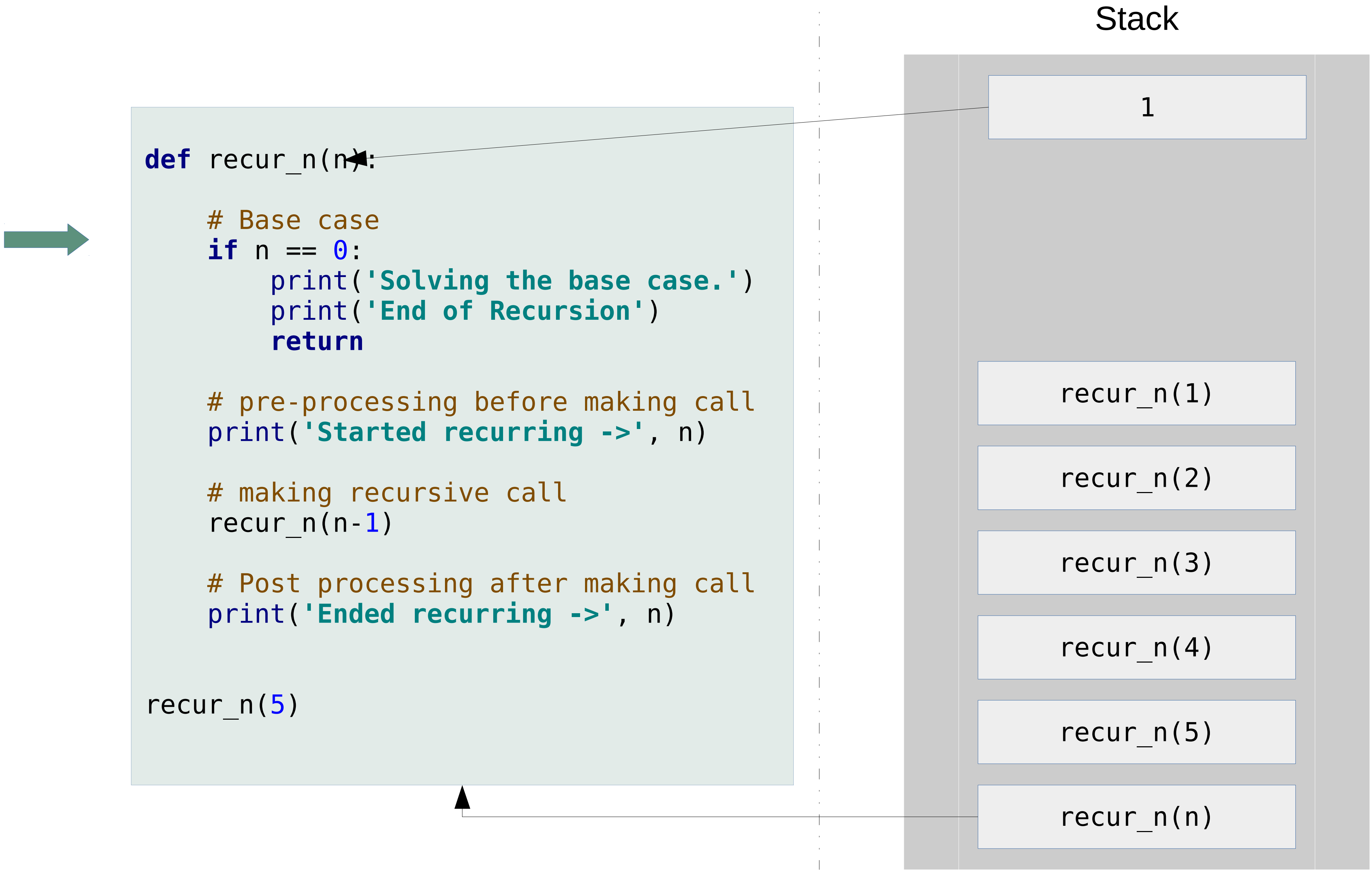
recur_n(3)

recur_n(4)

recur_n(5)

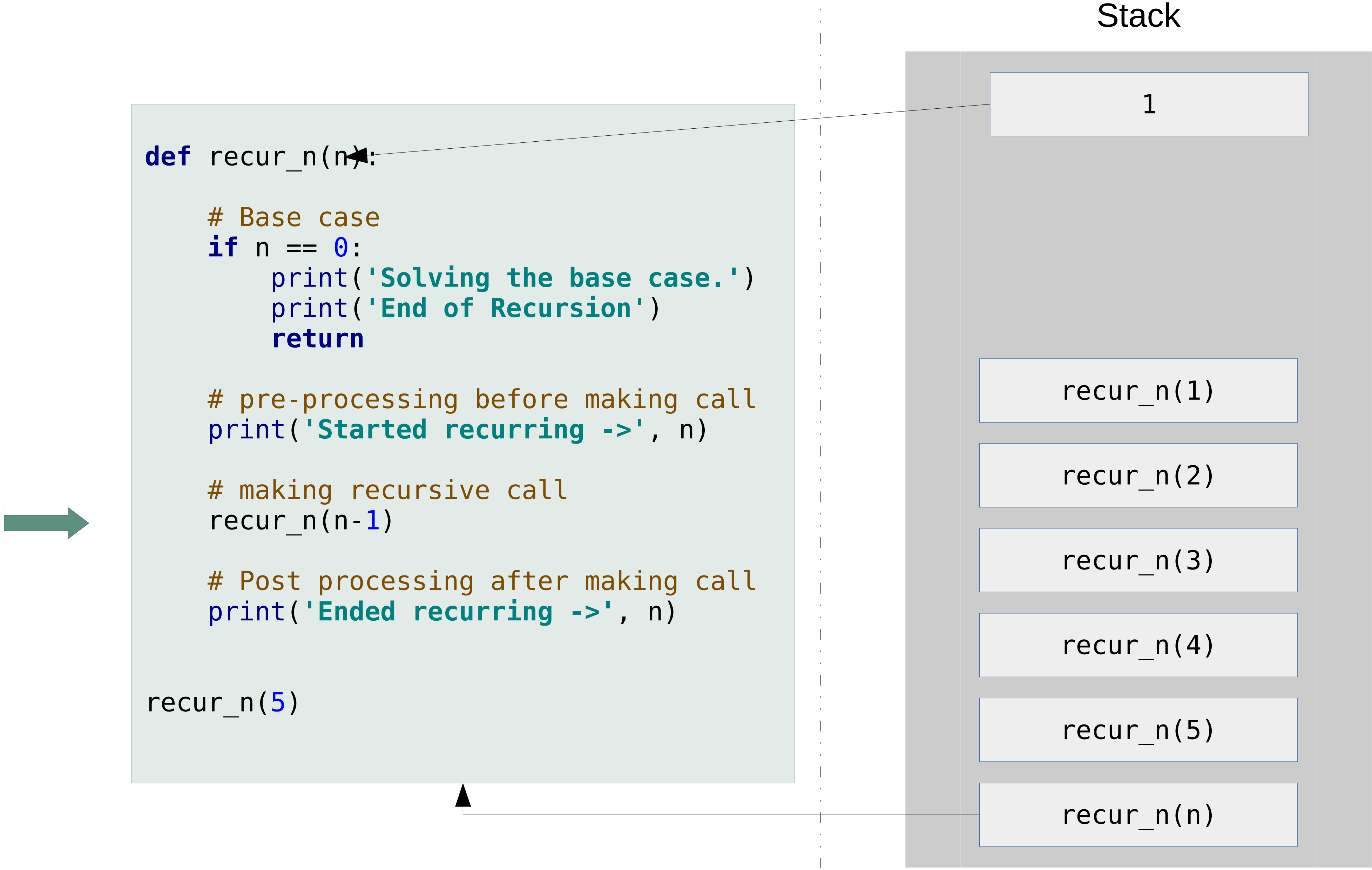
recur_n(n)

Stack Trace



```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```

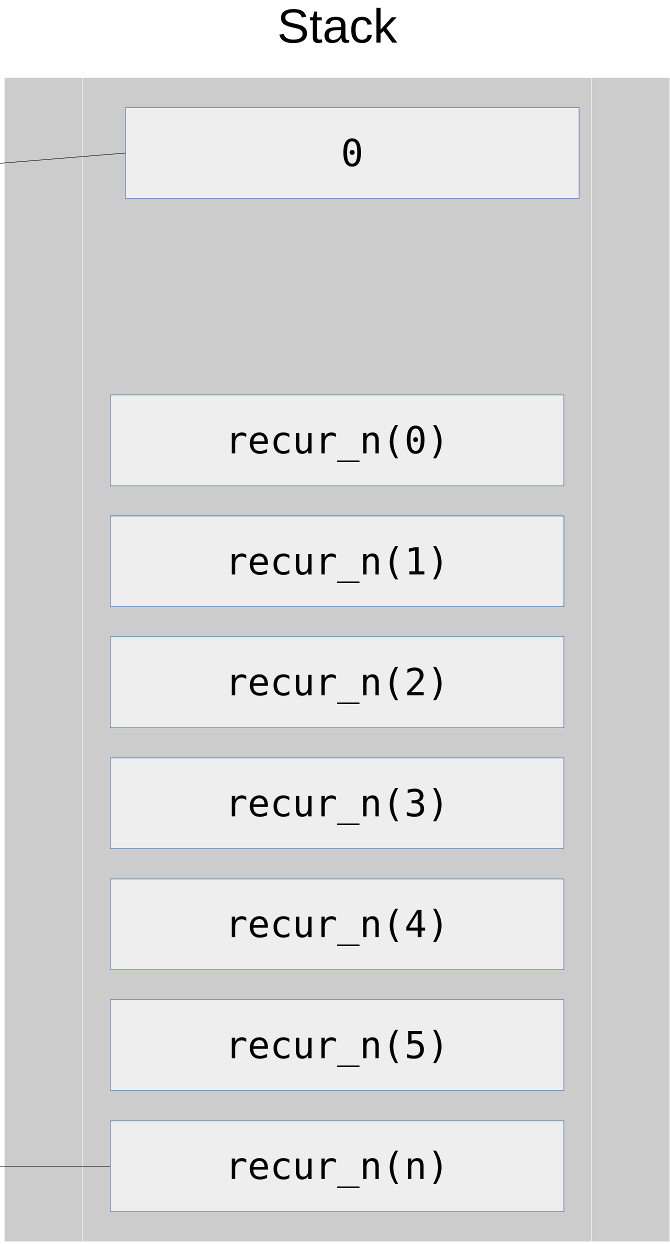
Stack Trace



```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```

Stack Trace

```python
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



Stack

0

recur_n(0)

recur_n(1)

recur_n(2)

recur_n(3)

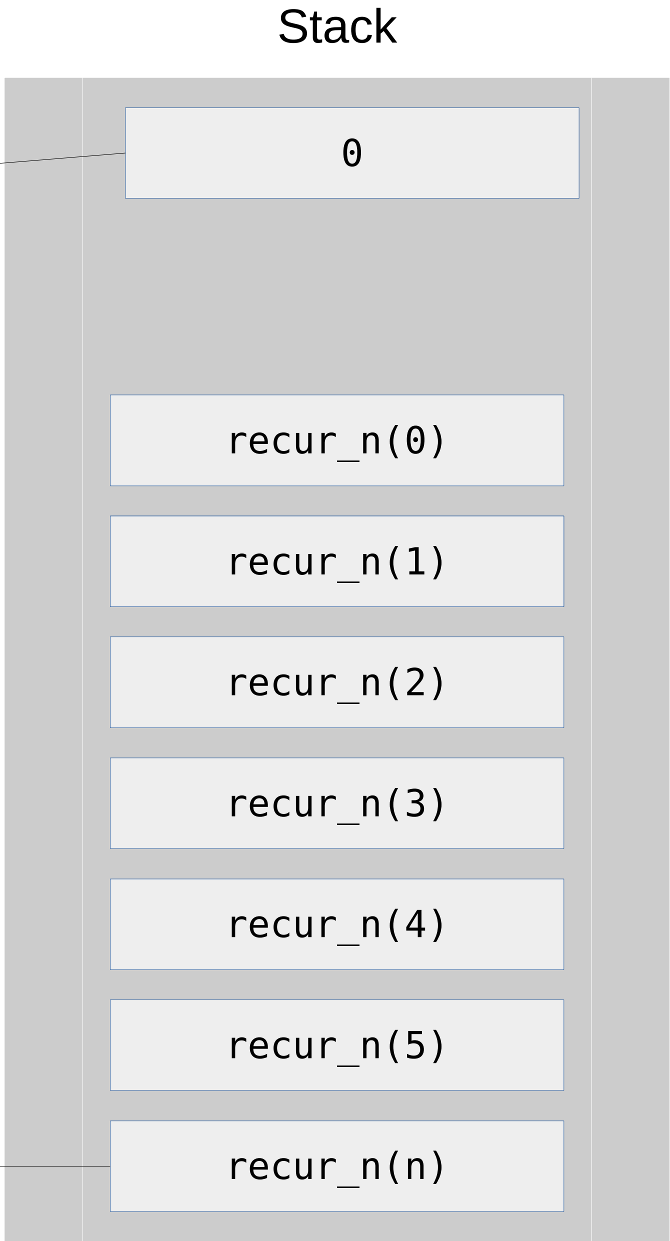
recur_n(4)

recur_n(5)

recur_n(n)

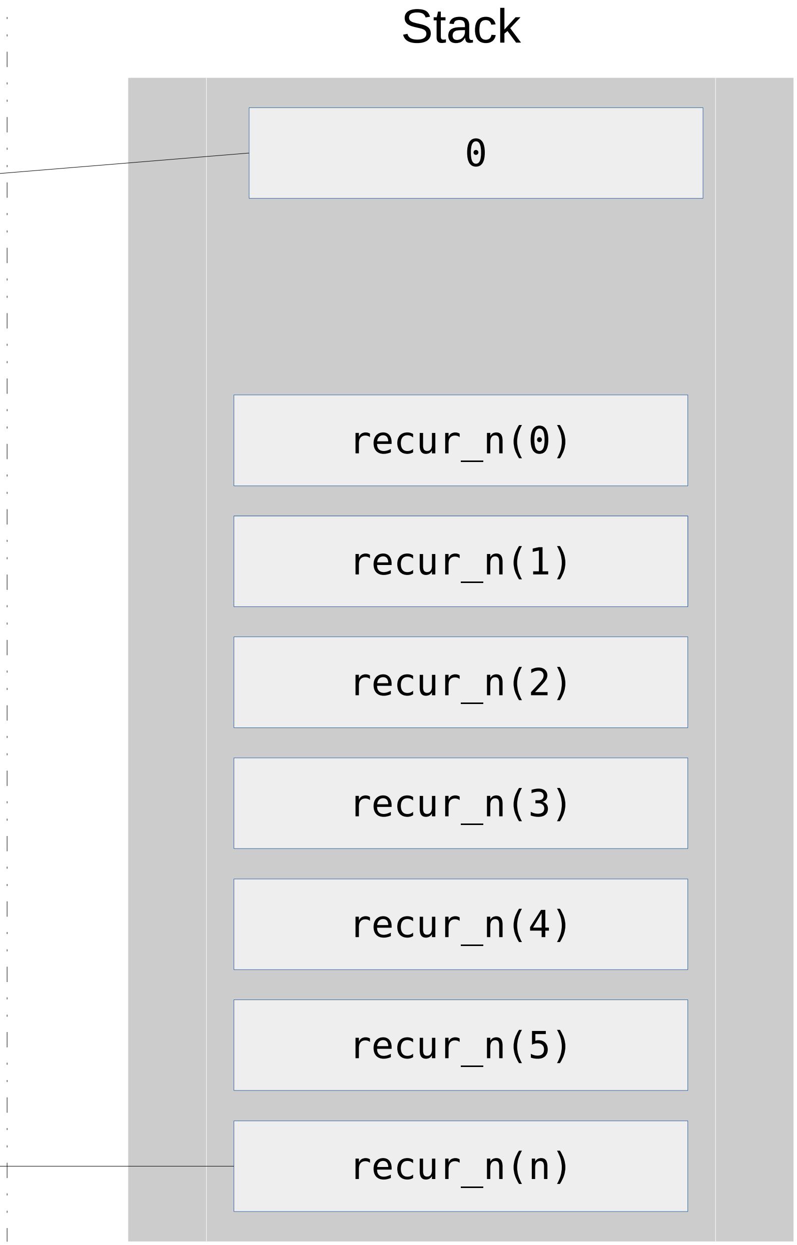
Stack Trace

```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



Stack Trace

```python
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



Stack

0

recur_n(0)

recur_n(1)

recur_n(2)

recur_n(3)

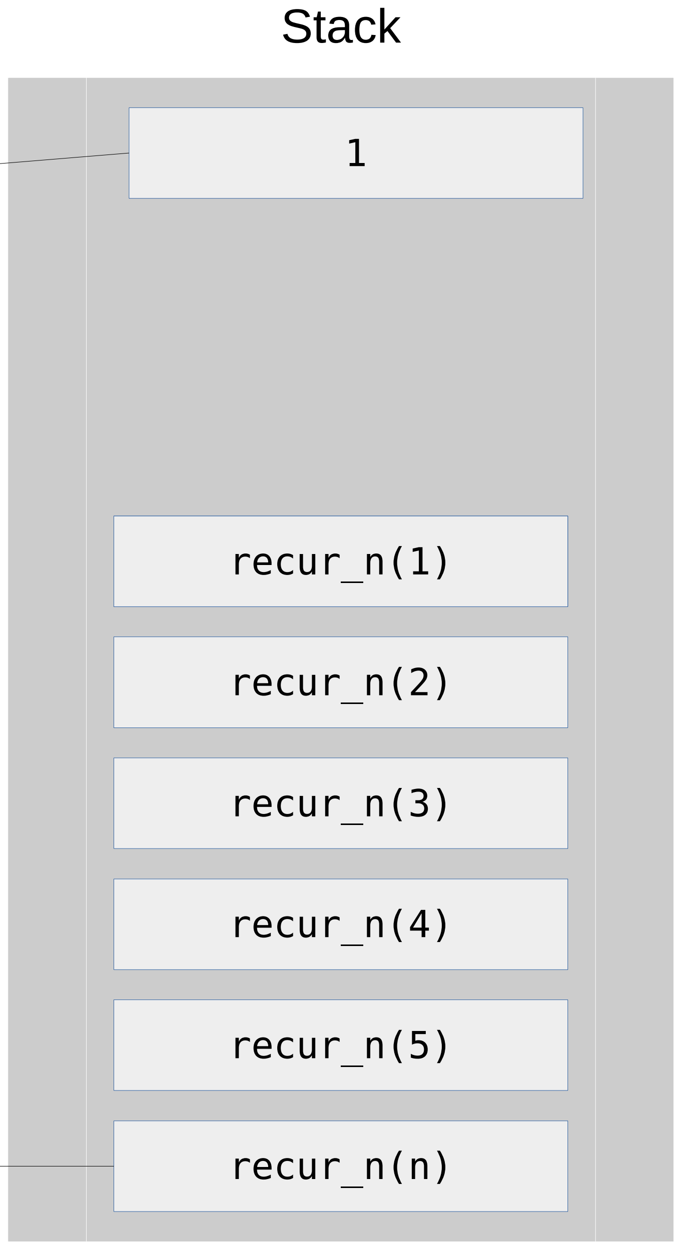
recur_n(4)

recur_n(5)

recur_n(n)

Stack Trace

```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



Stack

1

recur_n(1)

recur_n(2)

recur_n(3)

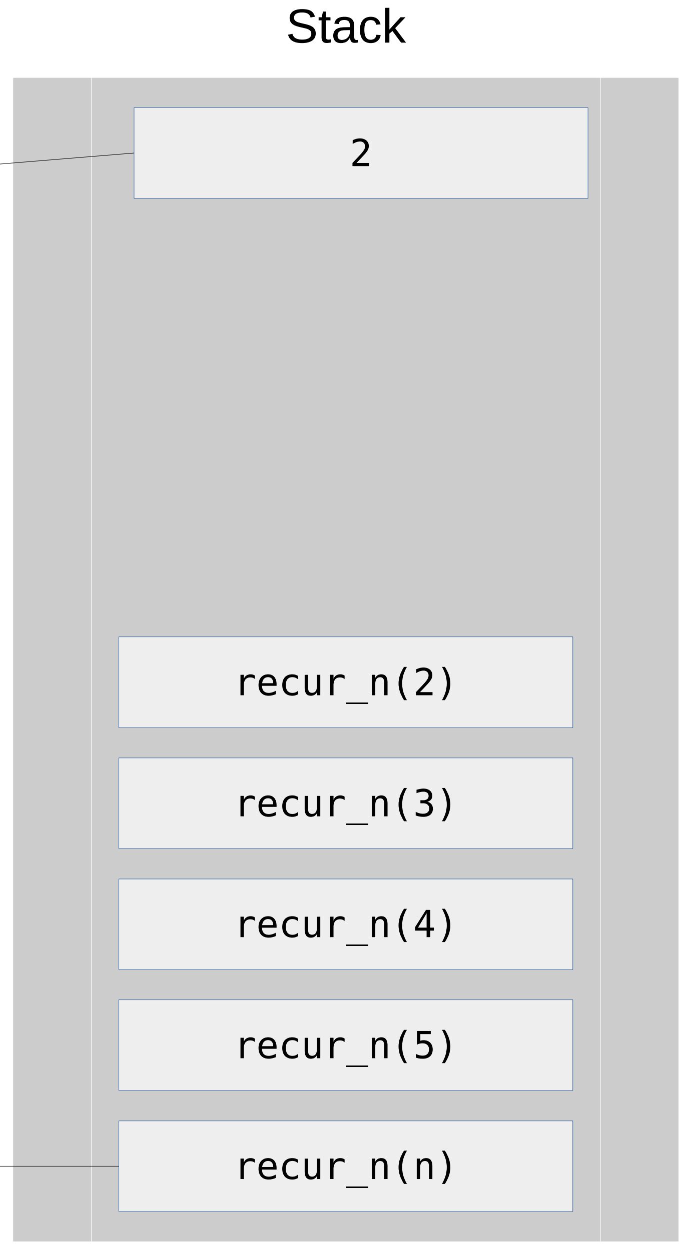
recur_n(4)

recur_n(5)

recur_n(n)

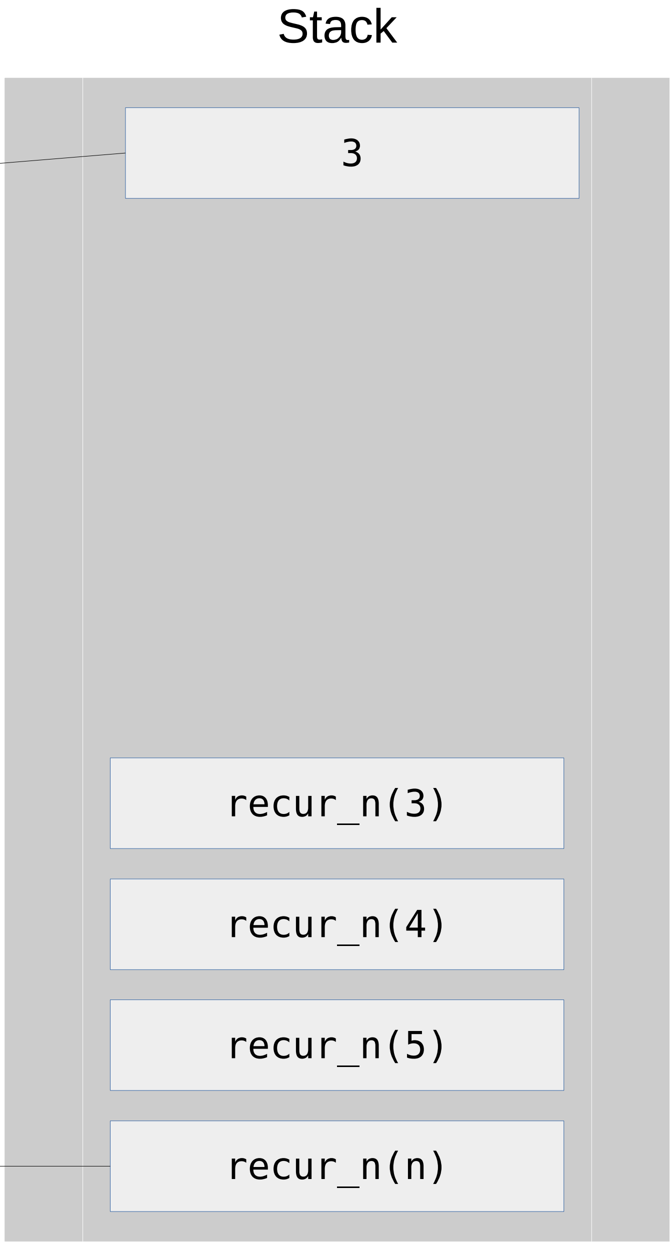
Stack Trace

```python
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



Stack Trace

```python
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```



Stack

3

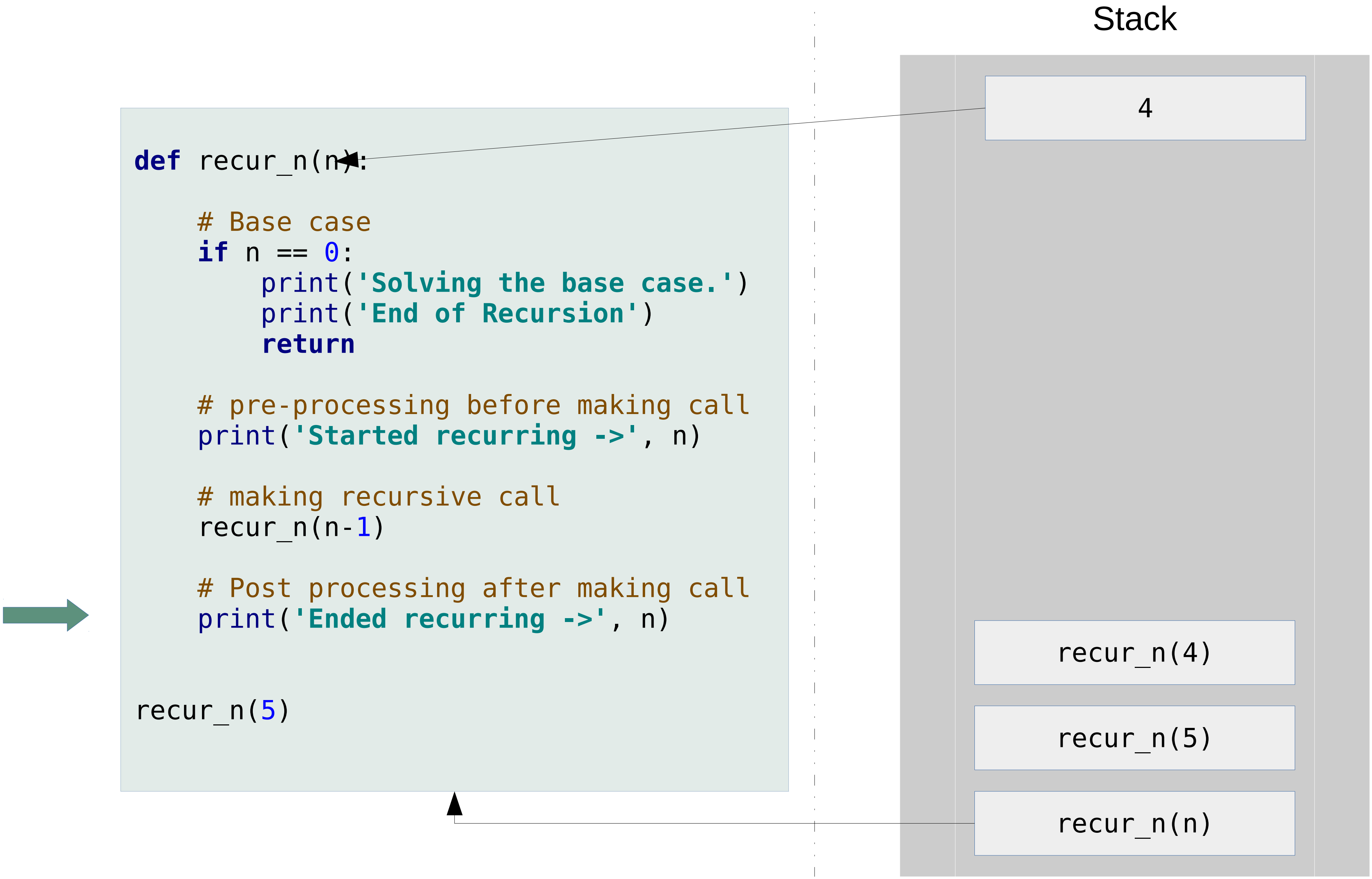
recur_n(3)

recur_n(4)

recur_n(5)

recur_n(n)

Stack Trace



```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```

Stack

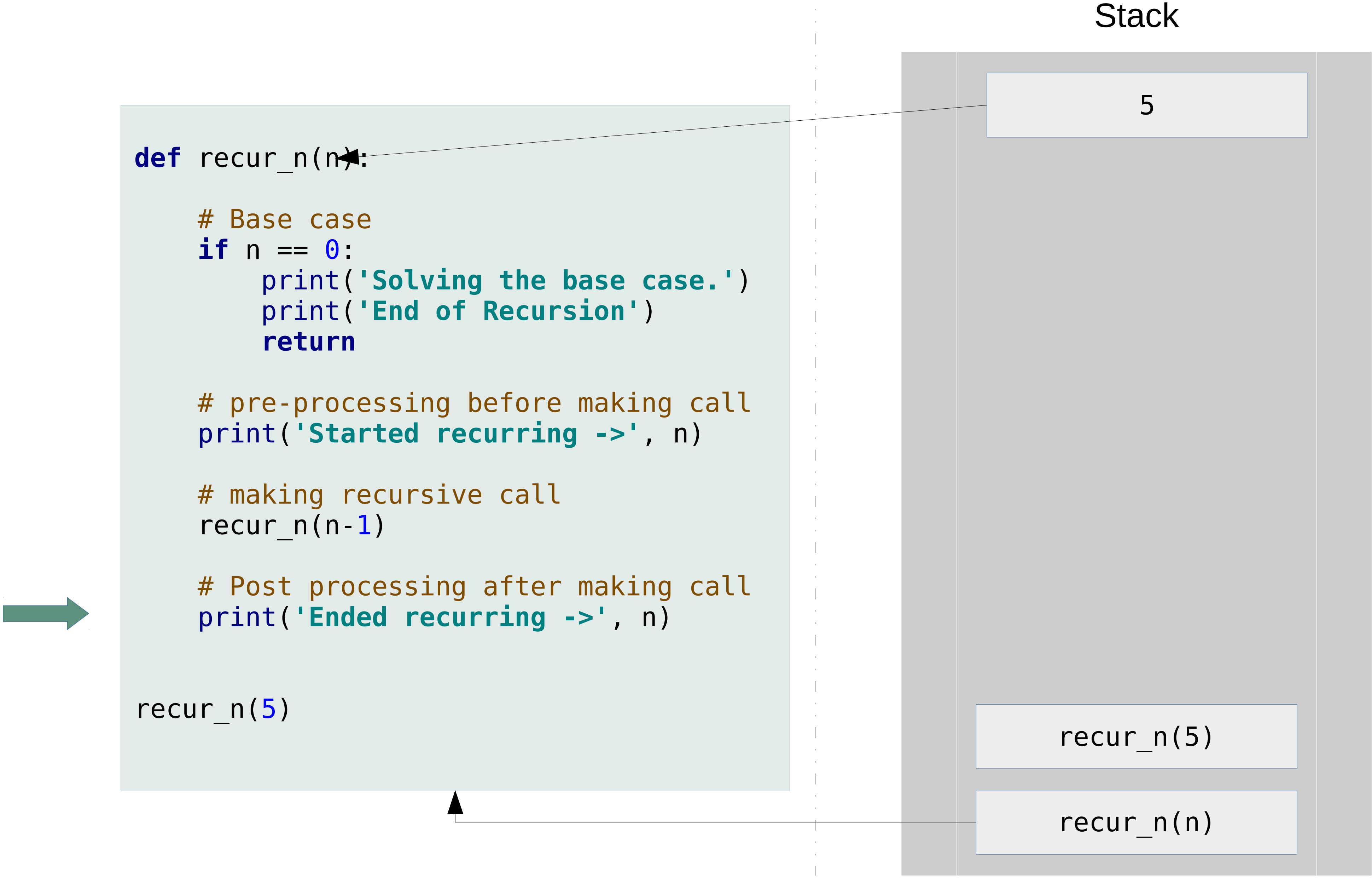
4

recur_n(4)

recur_n(5)

recur_n(n)

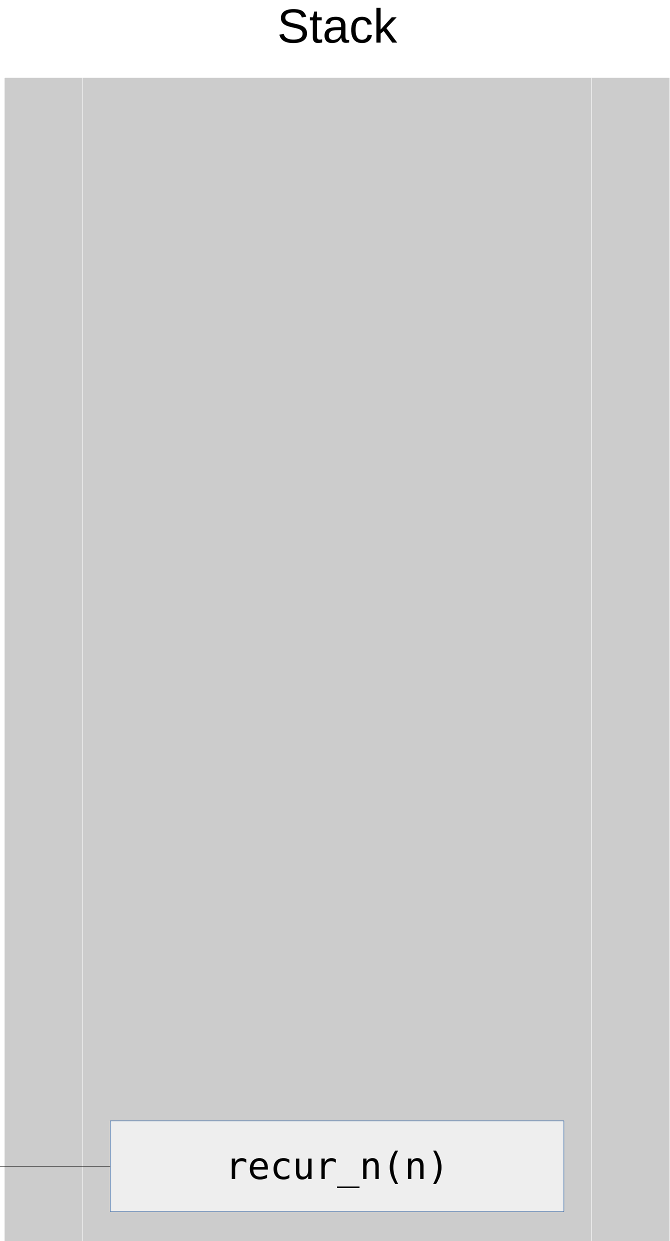
Stack Trace



```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```

Stack Trace

```
def recur_n(n):

    # Base case
    if n == 0:
        print('Solving the base case.')
        print('End of Recursion')
        return

    # pre-processing before making call
    print('Started recurring ->', n)

    # making recursive call
    recur_n(n-1)

    # Post processing after making call
    print('Ended recurring ->', n)


recur_n(5)
```


Stack

recur_n(n)

End