Towers of Hanoi Trace:
I have print statements throughout the function to help trace:

```
def moves(n, left):
        print("here " + str(n))
        if n == 0:
            return
        moves(n-1, not left)
        print("here1 " + str(n))
        if left:
            print(str(n) + ' left')
        else:
            print(str(n) + ' right')
        print("here2 " + str(n))
        moves(n-1, not left)
print(moves(3, True))
```

says here 3, 2, 1, then 0
that means it goes through the function and adds to the stack:
moves(3, True) moves(2, False) moves(1, True) → first in stack, last to be added
here1 1 → now after the recursion part
1 left → executes if left: n=1
here2 1 → n is still 1
here 0 → n is 0, goes through stack again (n=1 function has already happened, so it goes to n=2)
here1 2 → n is now 2 (going through the stack) → not true
prints "2 right" because it is move(2, False)

here 2 2 → goes past print statement for 2 right
now it calls moves(n-1, not left)
n is now 1 and moves(1, True) is added to the stack
here 1 → n is now 1
here 0 → (n) is now 0 again
stack: moves(1, True), moves(3, True)
↳ last to be added, so this is what will be called

prints here1 1 → means it's starting after where it left off
n = 1

prints "1 left" → if True statement is followed through

prints here2 1 → n is still = 1

calls moves(n-1, not left) → n = 0, left is False
stack just has moves(3, True) now

here 0 → now returns so it goes to functions in stack

here 1 3 → n is now = 3

prints 3 left because the boolean is True

calls moves(n-1, not left) → moves(2, False) is added to stack

here 2
here 1
here 0
} calls moves(1, True) and then goes to n=0

Stack: moves(1, True), moves(2, False)
↓ at the top

here 1 1 → n=1

prints 1 left because left = True

moves(n-1, not left) → n=0, goes back to stack because of return

moves(2, False) is now called so n=2

goes to else statement → prints "2 right"

moves(n-1, not left) is called → Stack: moves(1, True)

n resets to 0 after first moves(n-1, not left) is called

moves(1, True) is called from where it left off

n=1, left = true so it prints "1 left"

calls moves(n-1, not left) so n=0.

Stack = empty, return is None,

Recursion ended!