Further Examples of Recursion

→ Linear Recursion:

- when a recursive function is designed so that each invocation of the body makes at most one new recursive call
- a consequence of this is that any recursion trace will appear as a singular sequence of calls

Reversing a Sequence w/ Recursion

→ consider problem of reversing n elements of a sequence, S, so first element becomes last, etc.

→ We can solve this using linear recursion, observing that the reversal of a sequence can be achieved by swapping first & last elements and recursively reversing remaining elements

```
def reverse(S, start, stop):
  if start < stop - 1:
    S[start], S[stop-1] = S[stop-1], S[start]
    reverse(S, start+1, stop-1)
```

→ outside of base cases; if n is even, we eventually reach start == stop case, if n is odd, we will eventually reach start == stop-1

→ this implies that the algorithm is guaranteed to terminate after $1 + \lfloor \frac{n}{2} \rfloor$ recursive calls

Recursive Algorithms for Computing Powers

→ a trivial recursive definition follows from the fact $x^n = x \cdot x^{n-1}$ for $n > 0$

$$power(x,n) = \begin{cases} 1 & \text{if } n = 0 \\ x \cdot power(x, n-1) & \text{otherwise} \end{cases}$$

```
def power(x,n):
  if n == 0:
    return 1
  else:
    return x * power(x,n-1)
```

→ a recursive call to power(x,n) runs in O(n)

→ parameter decreases by one w/ each call and constant work performed at each of n+1 levels