Steps to thinking recursively

We have our recursive sum function

Step 1) Know what the function is supposed to do, not what it currently does.

- If we have a *sum* function, we know it must return an integer *sum* from 1 to size n

Step 2) Pick a smaller problem and assume our function will work for it.

- A subproblem is any problem smaller than the original. In this case, smaller means less input values.
- Our problem is to sum values from 1 to n, a subproblem could be summing all numbers from 1 to a value *less than* n

- If sumTo(n) was our original problem, these are all considered subproblems because they are smaller versions of the original problem

```
sumTo(n-1)
sumTo(n-2)
sumTo(n-3)
...
sumTo(1)
```

- We've got to pick an appropriate subproblem, usually one that's as close to our original as possible. Obviously these will not all work. In this case, since our problem solves for n, the next closest subproblem would be solving for n-1
- Using n-1 as our subproblem

```
sum(n):
        return sumTo(n-1)
```

Because of our subproblem selection, we already have the sum of all values from 1 to n-1. All we need to do now is make that final leap.

Step 3) Take the answer to your subproblem, and use it to solve the original problem.

- How can we take the solution to the subproblem and use it to solve the original problem?
- We have solved for the sum from 1 to n-1, but how do we use it to solve all the way to n?
- All we need to do is add the current value of n to the sum of the values up to n-1
- Return sumTo(n-1) + n
  - You took the subproblem and found how it is used to solve the original problem. Congratulations, you have just found a *recurrence relation.*

Step 4) Base case

- Your function right now is calling itself, which means it will likely run forever. We need to give it a termination condition
- The base case prevents more function calls if it has reached it. To pick a base case, think of the following.
    - What is the EASIEST POSSIBLE PROBLEM that requires no extra calculation?
    - n = 0, because 0 + 0 is 0, so if there are no values left to sum up, then we know we are done.
- That is it