**factorial**

Recall our factorial problem from earlier in the course:

# Write a method `factorial(n)` which takes a number and returns the factorial of n.

# A factorial is the product of all whole numbers between 1 and n, inclusive.

# For example, `factorial(5)` is 5 \* 4 \* 3 \* 2 \* 1 = 120.

How can we solve this problem using recursion? Notice that the structure of factorial has us take decreasing numbers similar to the countDown. However, this time we need to keep multiplying them together.

If we lay out the math used to calculate the factorial of some numbers, we'll notice a pattern:

# factorial(5) = 5 \* 4 \* 3 \* 2 \* 1

# factorial(4) = 4 \* 3 \* 2 \* 1

# factorial(3) = 3 \* 2 \* 1

# factorial(2) = 2 \* 1

# factorial(1) = 1 (base case)

In the outline above, notice that as the input number becomes smaller and smaller, the problem we solve also becomes smaller. There are less multiplications that need to take place!

Let's see the pattern programmatically. We can find the factorial of a number by using the factorial of another number:

# factorial(5) = 5 \* factorial(4)

# factorial(4) = 4 \* factorial(3)

# factorial(3) = 3 \* factorial(2)

# factorial(2) = 2 \* factorial(1)

# factorial(1) = 1 (base case)

Or in general, if n is some number:

* factorial(n) = n \* factorial(n - 1)
* factorial(1) = 1

Now let's implement factorial with some recursive Ruby:

def factorial(n)

return 1 if n == 1

n \* factorial(n - 1);

end

factorial(5) # => 120

Note that mathematically, factorial(0) = 1 so we could have also used that as the base case.

**Solving a Problem Recursively:**

Because every recursive problem must have a base and recursive case, we can follow these steps to help us write a recursive method:

1. Identify the base case in the problem and code it. The base case should explicity handle the scenario(s) where the arguments are so trivially "small", that we immediately know the result without further calculation. Be sure it works by testing it.
2. Solve the next level of the problem, using the result of the base case. Test it.
3. Modify the code in step 2, generalizing it for every level of the problem.