### fib

Let's take a look at a classic recursive problem. The fibonacci sequence!

# Write a method fib(n) that takes in a number and returns the nth number of

# the fibonacci sequence.

# In the fibonacci sequence, the 1st number is 1 and the 2nd number is 1. To generate the

# next number in the sequence, we take the sum of the previous two fibonacci numbers.

# For example the first 5 numbers of the fibonacci sequence are `1, 1, 2, 3, 5`

# Examples:

# fib(1) # => 1

# fib(2) # => 1

# fib(3) # => 2

# fib(4) # => 3

# fib(5) # => 5

# fib(6) # => 8

# fib(7) # => 13

To get a fibonacci number, we need to take the sum of the previous two. Take a look at the following ways we can describe fib.

# fib(5) = fib(4) + fib(3)

# fib(4) = fib(3) + fib(2)

# fib(3) = fib(2) + fib(1)

# fib(2) = 1 base case

# fib(1) = 1 base case

In general:

fib(n) = fib(n - 1) + fib(n - 2)

Finally, let's implement fib using code:

def fib(n)

return 1 if n == 1 || n == 2

fib(n - 1) + fib(n - 2)

end

This should feel like magic! To make sense of recursive code like fib, use abstraction and your comfort with helper methods. Recursion is only "different" from using regular helper methods because we are using the same method as the helper. However, you can use abstraction in the same way. If we wanted to soolve fib(5) we can decompose it into fib(4) + fib(3)! Take a second to appreciate the beauty of our recursive fib method! So cool.

### When is recursion appropriate?

Recursion allows us to solve problems in an elegant way. However, recursion is a tool that is only appropriate for certain problems. Look to the structure of a problem to figure out if it can be solved recursively. **Recursion is used to solve problems that can be decomposed into smaller versions of the**same**problem.** For example we can decompose fib(n) into fib(n - 1) + fib(n - 2). Intuitively, we know that fib(n - 1) is a "smaller" or "easier" problem than fib(n). The easiest subproblem is fib(1) or fib(2) because the answer is simply 1; this is an assumption in the fibonacci sequence. We use the easiest subproblems as the base case in recursion.