Recursion & Arrays

**What is recursion?**

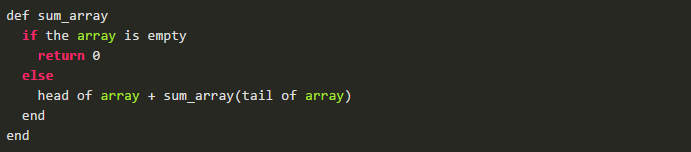
Recursion in computer science is a method where the solution to a problem depends on solutions to smaller instances of the same problem (as opposed to iteration).

## An example in pseudocode

Let's look at an example in pseudocode and talk about what is happening. We're going to sum all of the numbers in an array.

The usual flow of a recursive function is as follows:

* Perform an operation on the first element (head) of the array.
* Call the same function (hence recursion) with the remaining elements (tail) of array.
* Have a guard clause to stop recursion from happening when we've reached the end.



What happens is sort of like this:

array = [1,2,3,4,5]

1 + 2 + 3 + 4 + 5

1 + 2 + 3 + 9

1 + 2 + 12

1 + 14

15

## Recursion vs Iteration

In Ruby it is often preferable to avoid recursion and use iteration instead. Ruby (and most imperative programming languages) have very useful language constructs to aid with iterating over data.  
Recursion can end up being slower and use more memory than its iterative counterpart for a number of reasons

Certain languages, especially functional languages which deal with immutable data (Elixir, Haskell, etc), don't have language constructs at all to deal with iteration and all of it is performed through recursion. These languages are optimized for recursion using a technique called [tail call optimization](http://en.wikipedia.org/wiki/Tail_call) and also come with handy pattern matching to deal with the head and tail of arrays.

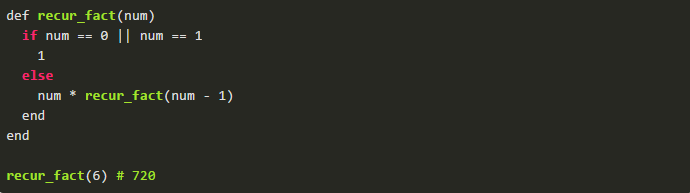
## Why recursion in Ruby?

Despite what was said above, we wanted to see what recursion might look like to solve a number of problems in Ruby. Because Ruby doesn't have an easy way of grabbing the head and tail of an array through pattern matching, when we’ve added a head\_tail method to the Array class to aid in this.

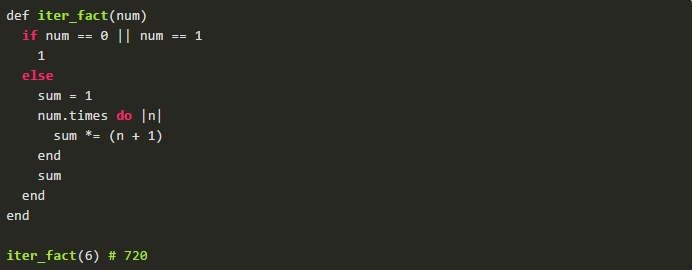


## Recursion Examples

### **Factorials**

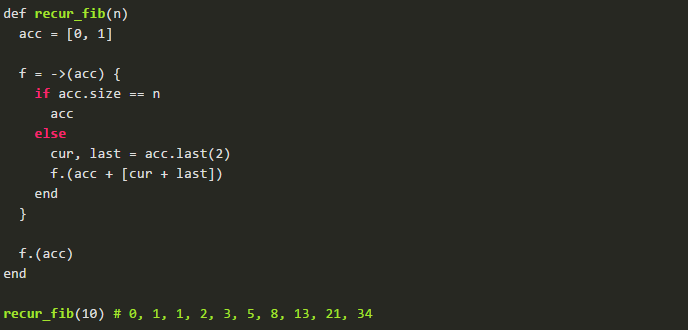
Factorials in math are written like 6! and are the result of 6 \* 5 \* 4 \* 3 \* 2 \* 1.  
  


Let’s look at the same method done in an iterative way:

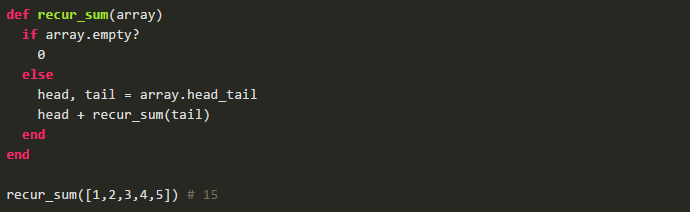


### **Fibonacci**

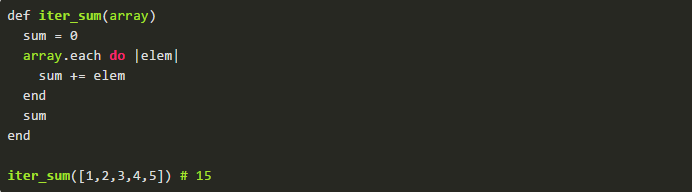
I heard a funny definition of Fibonacci that goes like this: "Fibonacci - A problem used to teach recursion in computer science." The first 10 Fibonacci numbers go like this:   
0, 1, 1, 2, 3, 5, 8, 13, 21, 34 - The next number is the sum of the previous 2 numbers.



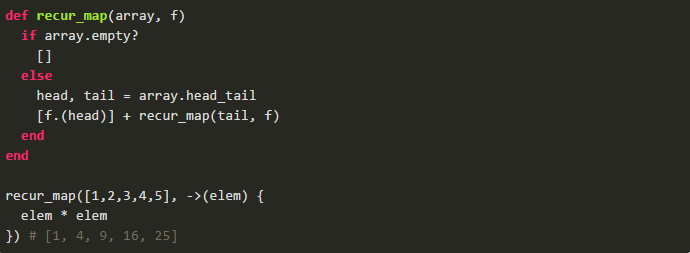
**Summing Array:**



Iteratively done:



**Mapping Array:**



Iteratively done:

