**Recursion**

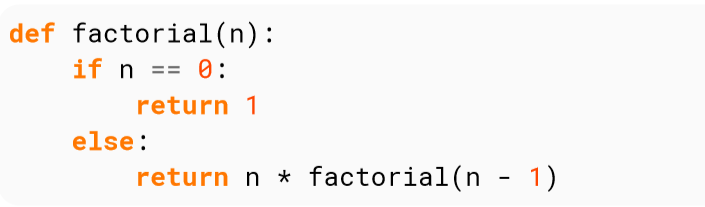
**Overview**

|  |
| --- |
| Today we are going to talk about how to implement a method, once you already have a specification. We will focus on one particular technique; recursion. Recursion is not appropriate for every problem, but it’s an important tool in your software development toolbox. |

Recursion occurs when a function call causes that same function to be called again before the original call terminates. For example, consider the well-known mathematical expression x! (i.e the factorial operation). The factorial operation is defined for all nonnegative integers as follows:

* If the number is 0, then the answer is 1.
* Otherwise, the answer is that number times the factorial of one less than that number

In Python a naive implementation of the factorial operation can be defined as a function as follows:

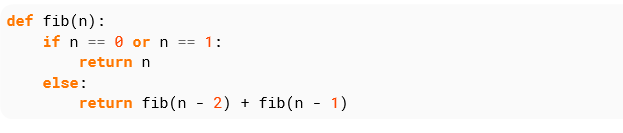


The statement if **n == 0**: return 1 is called a **base case**. This is because it exhibits no recursion. A base case is absolutely required. Without one, you run into infinite recursion. With that said, as long as you have at least one base case, you can have as many cases as you want.

You may also have multiple recursion cases, but we won’t get into that since it’s relatively uncommon and is often difficult to mentally process.

You can also have parallel recursive function calls, For example, consider the Fibonacci sequence which is defined as follows:

* If the number is 0, then the answer is 0.
* If the number is 1, then the answer is 1.
* Otherwise, the answer is the sum of the previous two Fibonacci numbers



Now, let’s cover a few more vocabulary terms:

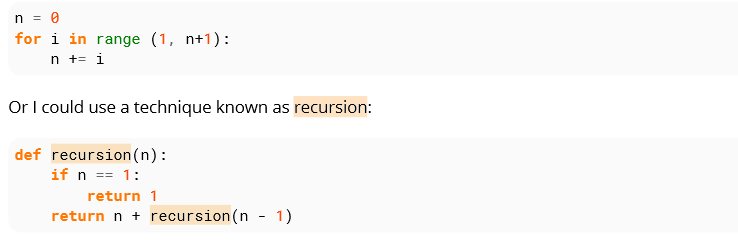
* A tail call is simply a recursive function call which is the last operation to be performed before returning a value. To be clear, return foo(n-1) is a tail call, but return foo(n-1) + 1 is not (since the addition is the last operation).
* Tail call optimization (TCO) is a way to automatically reduce recursion in recursive functions.
* Tail call elimination (TCE) is the reduction of a tail call to an expression that can be evaluated without recursion. TCE is a type of TCO

Tail call optimization is helpful for a number of reasons:

* The interpreter can minimize the amount of memory occupied by environments. Since no computer has unlimited memory, excessive recursive calls would lead to a stack overflow.
* The interpreter can reduce the number of stack frame switches.

Sum of numbers from 1 to n

If i wanted to find out the sum of numbers from 1 to n where n is a natural number, I can do 1+2+3+4 +(several hours later) + n. Alternatively I could write a for loop.



Recursion has advantages over the above methods. Recursion takes less time writing out 1+2+3 for a sum from 1 to 3.

**Task**

Given an integer, write a function using recursion that returns true if the given string is a palindrome, else it should return false. For example, 12321, *madam* are palindromes, but 1451 is not palindrome.

