# Week 14 Assignment - Recursion

## Overview

This week's assignment will focus on using recursion to implement two algorithms. Make sure to create a github repository for this assignment named **IS211_Assignment14**. All development should be done in this repository.

## Useful Reminders

1. Read the assignment over a few times. At least twice. It always helps to have a clear picture of the overall assignment when understanding how to build a solution.
2. Think about the problem for a while, and even try writing or drawing a solution using pencil and paper or a whiteboard.
3. Before submitting the assignment, review the "Functional Requirements" section and make sure you hit all the points. This will not guarantee a perfect score, however.

## Part I - Implement the Fibonnaci Sequence

One of this week's quiz questions referred to the Fibonnaci sequence. This sequence of numbers is defined such that the $n^{th}$ number of the sequence is simply the sum of the two previous numbers in the sequence. In formal terms, $F_n = F_{n-1} + F_{n-2}$, where $F_n$ is the $n^{th}$ Fibonnaci number. Write a function in *recursion.py*, called *fibonnaci*, which will accept one integer parameter (lets call it *n*) and returns the $n^{th}$ element of the Fibonnaci sequence.

## Part II - Implement Euclid's GCD Algorithm

The greatest common divisor, or GCD, of two integers is the largest number that divides both of them with no remainder. Euclid's algorithm is one method to find the GCD of two numbers. Mathematically, we know that if *r* is the remainder when *a* is divided by *b*, then *gcd(a, b) = gcd(b, r)*. Write a *recursive* function called *gcd* that takes parameters *a* and *b* and returns their greatest common divisor. Think about what the base case is for this algorithm.

## Part III - String Comparison

When comparing strings, we can always use the == operator in Python. However, let us define our own function to compare two strings, but lets do so recursively. Write a function called compareTo(s1, s2) that will:

- a negative number if s1 < s2,
- 0 if s1 == s2, and
- a positive number if s1 > s2

Again, think about what the base case is here, which will help clarify how to implement the recursion.