In this module we will be covering how recursion works in PYTHON, and will look at examples of recursive algorithms.

Recursion

A recursive function is a function that calls itself.

A problem can be solved with recursion if it can be broken down into small problems that are identical in structure to the overall problem. Recursion is NEVER required to solve a problem. However, some repetitious problems are more easily solved with recursion than with a loop structure.

Direct recursion is a function that directly calls itself.

Indirect recursion is when function A calls function B, which in turn calls function A.

Recursive Algorithms

Algorithms are sets of logic that perform a particular function that you want to emulate in programming. The Fibonacci series uses recursive functions to create the Fibonacci set of numbers.

Another algorithm using recursion is the Greatest Common Divisor.

The Towers of Hanoi is a mathematical game to teach the power of recursion. The game uses three pegs and set of discs with holes through the centers. All pegs start on the left disk with smallest to largest (largest at the bottom). The middle peg can be used as a temporary holding area for a disk. The object is to get all of the disks on the 3rd peg with the smallest to largest order. The rules that must be follows are:

1. only one disk may be moved at a time

2. a disc cannot be placed on top of a smaller disc

3. all discs must be stored on a peg except while being moved

When you can understand the logic of this game, you can use recursion to solve the problem with MUCH less coding.

Recursion verses a Loop

Both a loop and recursion can be used to solve problems. However, there are several reasons NOT to use recursion

1. recursive calls are less efficient than loops - more overhead

2. a solution may be more evident with the loop than with recursion

Reasons to use recursion instead of a loop:

1. many mathematical problems are more easily solved with recursion