Recursion

Exercises

Exercise 1

Create a recursive function to find the sum of the first n natural numbers.

- **Base Case**: If n is 1, return 1.
- **Recursive Step**: Return n plus the function called with n-1.

Exercise 2

Write a recursive function to find the nth number in the Fibonacci sequence.

- **Base Case**: If n is 0, return 0. If n is 1, return 1.
- **Recursive Step**: Return the sum of the function called with n-1 and n-2

Exercise 3

Write a recursive function to reverse a string.

- **Base Case**: If the string is empty or has only one character, return the string.
- **Recursive Step**: Return the last character of the string plus the function called with the substring excluding the last character.

Exercise 4

Identify the **Base Cases** and the **Recursive Case** of the following recursive functions and code them in Python:

1. one that counts down from a given number to 1.
2. one that finds the minimum value in an array of integers.
3. one that checks if a list of numbers is sorted in ascending order.
4. one that calculates x raised to the power of n (i.e. x^n).
5. one that checks if a string is a palindrome.

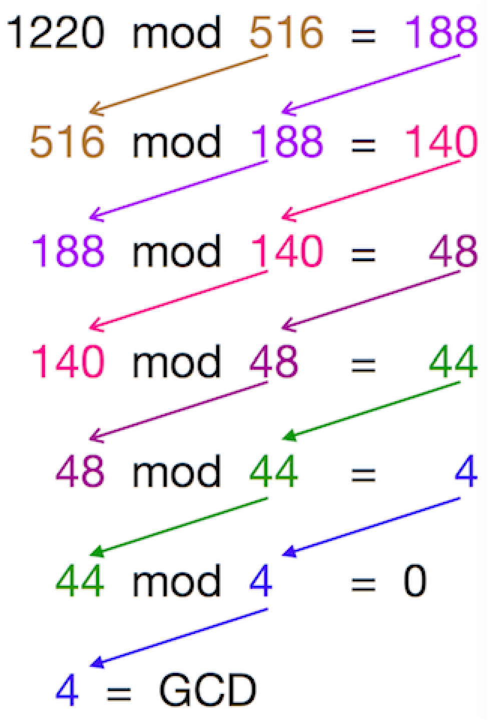
Exercise 5 – List-ception

Calculates the sum of all numbers in a list containing other lists using a recursive function.

Examples:

- *[[1,2,3], [4,5,6]] gives 21*
- *[[1,2,3, 4, 5, 6]] gives 21*
- *[1,2, [3,4,[5,6]]] gives 21*

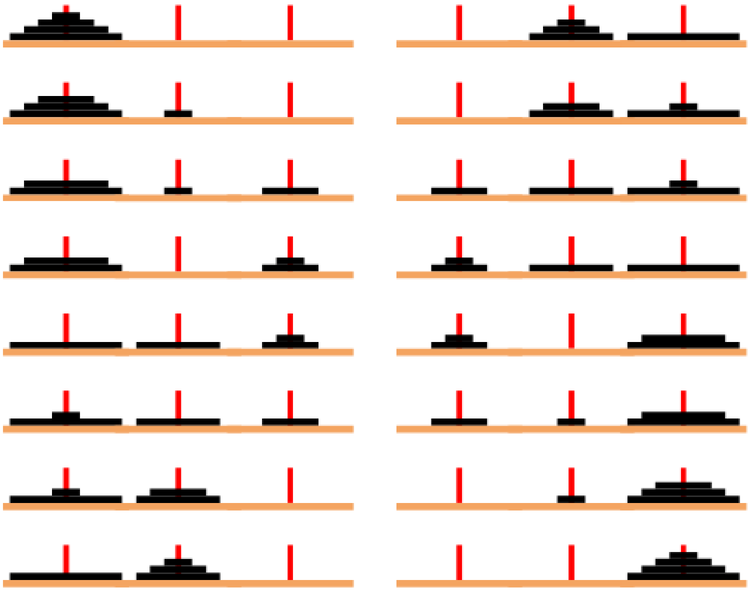
Exercise 6 – Greatest Common Divisor

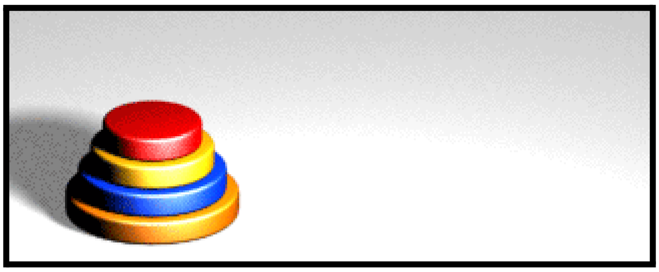


Given two non-negative integers, a and b, create a recursive function to find their Greatest Common Divisor (GCD). The Euclidean algorithm is a method for finding the GCD of two numbers that is based on the principle that the GCD of two numbers also divides their difference.

1. Start with two integers, a and b, where a > b.
2. Find the remainder of a divided by b. Call this remainder r.
3. Replace a with b and b with r.
4. Repeat steps 2 and 3 until b becomes 0. The non-zero remainder at this stage will be the GCD of a and b.
5. If b becomes 0, then a is the GCD.

Exercise 7 - Tower Of Hanoï





You are given three pegs: source (A), auxiliary (B), and target (C). Initially, there are n disks stacked on peg A, with the largest disk at the bottom and the smallest at the top. The objective is to move all the disks from peg A to peg C using peg B as an auxiliary, following these rules:

1. Only one disk can be moved at a time.
2. A disk can only be placed on top of a larger disk or an empty peg.
3. You cannot place a disk on top of a smaller disk.

Your task is to determine the sequence of moves required to transfer all n disks from peg A to peg C.

The idea is to move n-1 disks to the auxiliary peg, then move the largest disk to the target peg, and finally move the n-1 disks from the auxiliary peg to the target peg. Here's a step-by-step method:

1. Move n-1 Disks from Source to Auxiliary: Treat the problem as moving n-1 disks from peg A to peg B, using C as auxiliary.
2. Move the Last Disk from Source to Target: Move the remaining disk on peg A (the largest one) to peg C.
3. Move n-1 Disks from Auxiliary to Target: Now treat the problem as moving n-1 disks from peg B to peg C, using A as auxiliary.