**Question1. Create a function that takes three arguments a, b, c and returns the sum of the numbers that are evenly divided by c from the range a, b inclusive.**

**Examples**

**evenly\_divisible(1, 10, 20) ➞ 0**

**# No number between 1 and 10 can be evenly divided by 20.**

**evenly\_divisible(1, 10, 2) ➞ 30**

**# 2 + 4 + 6 + 8 + 10 = 30**

**evenly\_divisible(1, 10, 3) ➞ 18**

**# 3 + 6 + 9 = 18**

def evenly\_divisible(a, b, c):

sum = 0

for i in range(a, b+1):

if i % c == 0:

sum += i

return sum

**Question2. Create a function that returns True if a given inequality expression is correct and False otherwise.**

### Examples

**correct\_signs("3 < 7 < 11") ➞ True**

**correct\_signs("13 > 44 > 33 > 1") ➞ False**

**correct\_signs("1 < 2 < 6 < 9 > 3") ➞ True**

def correct\_signs(expr):

parts = expr.split()

for i in range(0, len(parts)-2, 2):

if parts[i+1] == '<':

if not int(parts[i]) < int(parts[i+2]):

return False

elif parts[i+1] == '>':

if not int(parts[i]) > int(parts[i+2]):

return False

return True

**Question3. Create a function that replaces all the vowels in a string with a specified character.**

### Examples

**replace\_vowels("the aardvark", "#") ➞ "th# ##rdv#rk"**

**replace\_vowels("minnie mouse", "?") ➞ "m?nn?? m??s?"**

**replace\_vowels("shakespeare", "\*") ➞ "sh\*k\*sp\*\*r\*"**

def replace\_vowels(txt, char):

vowels = 'aeiouAEIOU'

for vowel in vowels:

txt = txt.replace(vowel, char)

return txt

**Question4. Write a function that calculates the factorial of a number recursively.**

### Examples

**factorial(5) ➞ 120**

**factorial(3) ➞ 6**

**factorial(1) ➞ 1**

**factorial(0) ➞ 1**

def factorial(n):

if n == 0 or n == 1:

return 1

else:

return n \* factorial(n-1)

**Question 5**

**Hamming distance is the number of characters that differ between two strings.**

**To illustrate:**

**String1: "abcbba"**

**String2: "abcbda"**

**Hamming Distance: 1 - "b" vs. "d" is the only difference.**

**Create a function that computes the hamming distance between two strings.**

### Examples

**hamming\_distance("abcde", "bcdef") ➞ 5**

**hamming\_distance("abcde", "abcde") ➞ 0**

**hamming\_distance("strong", "strung") ➞ 1**

def hamming\_distance(str1, str2):

# Check if the strings have the same length

if len(str1) != len(str2):

return None

# Compute the hamming distance

distance = 0

for i in range(len(str1)):

if str1[i] != str2[i]:

distance += 1

return distance