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

# 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.

# Define a function, function1 hat 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.

def function1(a,b,c):

sum = 0

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

if i%c == 0:

sum = sum + i

return sum

# Call function, function1 with different inputs and check the results

print(function1(1,10,20)) # The output should be 0

print(function1(1, 10, 2)) # The output should be 30

print(function1(1, 10, 3)) # The output should be 18

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

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

# Define a function, function1

def function1(s1):

result = eval(s1)

if result:

return True

else:

return False

# call the function, function1 with different inputs and check the results

string1 = "3 < 7 < 11"

print(function1(string1)) # The output should be True

string1 = "13 > 44 > 33 > 1"

print(function1(string1)) # The output should be False

string1 = "1 < 2 < 6 < 9 > 3"

print(function1(string1)) # The output should be 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\*"

# Create a function that replaces all the vowels in a string with a specified character

# Define a function, function1

def function1(s1, c):

s1 = s1.replace('a',c)

s1 = s1.replace('e',c)

s1 = s1.replace('i',c)

s1 = s1.replace('o',c)

s1 = s1.replace('u',c)

return s1

# Call the function, function1 with different inputs and check the results

string1 = "the aardvark"

char = '#'

print(function1(string1, char)) #The output should be "th# ##rdv#rk"

string1 = "minnie mouse"

char = '?'

print(function1(string1, char)) #The output should be "m?nn?? m??s?"

string1 = "shakespeare"

char = '\*'

print(function1(string1, char)) #The output should be "sh\*k\*sp\*\*r\*"

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

# Write a function that calculates the factorial of a number recursively.

# Define a function, function1 which returns the factorial of a number

def function1(n):

if n == 1:

return 1

else:

return n \* function1(n - 1)

# Call the function with different inputs and check the results

print(function1(1)) # The output should be 1

print(function1(2)) # The output should be 2

print(function1(3)) # The output should be 6

print(function1(4)) # The output should be 24

print(function1(5)) # The output should be 120

**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

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

from scipy.spatial.distance import hamming

# Define a function, function1 to calculate hamming distance

def function1(s1,s2):

hamming\_distance = hamming(list(s1),list(s2))\*len(s1)

return hamming\_distance

# Call the function, function1 with different inputs given and check the result

s1, s2 = "abcde", "bcdef"

print(function1(s1, s2)) # The output should be 5

s1, s2 = "abcde", "abcde"

print(function1(s1, s2)) # The output should be 0

s1, s2 = "strong", "strung"

print(function1(s1, s2)) # The output should be 1