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

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

total\_sum = 0

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

if num % c == 0:

total\_sum += num

return total\_sum

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

print(evenly\_divisible(1, 10, 20)) # Output: 0

print(evenly\_divisible(1, 10, 2)) # Output: 30

print(evenly\_divisible(1, 10, 3)) # Output: 18

Exmaple usage:

evenly\_divisible(1, 10, 20) ➞ 0

evenly\_divisible(1, 10, 2) ➞ 30

evenly\_divisible(1, 10, 3) ➞ 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

ANS: def correct\_signs(expression):

try:

return eval(expression)

except:

return False

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

print(correct\_signs("3 < 7 < 11")) # Output: True

print(correct\_signs("13 > 44 > 33 > 1")) # Output: False

print(correct\_signs("1 < 2 < 6 < 9 > 3")) # Output: True

Example usage: correct\_signs("3 < 7 < 11") ➞ True

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

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

ANS: def replace\_vowels(s, replacement\_char):

vowels = "AEIOUaeiou"

return ''.join(replacement\_char if char in vowels else char for char in s)

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

print(replace\_vowels("the aardvark", "#")) # Output: "th# ##rdv#rk"

print(replace\_vowels("minnie mouse", "?")) # Output: "m?nn?? m??s?"

print(replace\_vowels("shakespeare", "\*")) # Output: "sh\*k\*sp\*\*r\*"

EXAMPLE USAGE:

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

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

replace\_vowels("shakespeare", "\*") ➞ "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

**ANS : def factorial(n):**

**if n == 0 or n == 1:**

**return 1**

**else:**

**return n \* factorial(n - 1)**

**# Test the function**

**if \_\_name\_\_ == "\_\_main\_\_":**

**print(factorial(5)) # Output: 120**

**print(factorial(3)) # Output: 6**

**print(factorial(1)) # Output: 1**

**print(factorial(0)) # Output: 1**

**Example usage:**

**factorial(5) ➞ 120**

**factorial(3) ➞ 6**

**factorial(1) ➞ 1**

**factorial(0) ➞ 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

**ANS: def hamming\_distance(str1, str2):**

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

**raise ValueError("Input strings must have the same length.")**

**return sum(c1 != c2 for c1, c2 in zip(str1, str2))**

**# Test the function**

**if \_\_name\_\_ == "\_\_main\_\_":**

**print(hamming\_distance("abcde", "bcdef")) # Output: 5**

**print(hamming\_distance("abcde", "abcde")) # Output: 0**

**print(hamming\_distance("strong", "strung")) # Output: 1**

**Example usage:**

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

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

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