# **Assignment 17**

#### 1.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**

In [37]:

**def** evenDivisible(a,b,c):  
 divList **=** []  
 **for** num **in** range(a,b**+**1):  
 **if** num**%c** == 0:  
 divList**.**append(num)  
 print(f'{a,b,c} ➞ {sum(divList)}')  
  
evenDivisible(1,10,20)  
evenDivisible(1,10,2)  
evenDivisible(1,10,3)

(1, 10, 20) ➞ 0  
(1, 10, 2) ➞ 30  
(1, 10, 3) ➞ 18

#### 2.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**

In [42]:

**def** checkEquality():  
 in\_string **=** input('Enter the inequality: ')  
 out\_bool **=** eval(in\_string)  
 print(f'{in\_string} ➞ {out\_bool}')  
  
for x **in** range(3):  
 checkEquality()

Enter the inequality: 3 < 7 < 11  
3 < 7 < 11 ➞ True  
Enter the inequality: 13 > 44 > 33 > 1  
13 > 44 > 33 > 1 ➞ False  
Enter the inequality: 1 < 2 < 6 < 9 > 3  
1 < 2 < 6 < 9 > 3 ➞ True

#### 3.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", "\*") ➞ "shksp\*\*r"**

In [20]:

**def** replaceVowels():  
 vowels **=** ['a','e','i','o','u','A','E','I','O','U']  
 in\_string **=** input("String: ")  
 in\_string\_copy **=** in\_string  
 in\_char **=** input('Replacement character: ')  
 **for** ele **in** in\_string:  
 **if** ele **in** vowels:  
 in\_string **=** in\_string**.**replace(ele,in\_char)  
 print(f'{in\_string\_copy} {in\_char} ➞ {in\_string}')  
   
for x **in** range(3):  
 replaceVowels()

String: the aardvark  
Replacement character: #  
the aardvark # ➞ th# ##rdv#rk  
String: minnie mouse  
Replacement character: ?  
minnie mouse ? ➞ m?nn?? m??s?  
String: shakespeare  
Replacement character: \*  
shakespeare \* ➞ sh\*k\*sp\*\*r\*

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

**Examples:**  
**factorial(5) ➞ 120**  
**factorial(3) ➞ 6**  
**factorial(1) ➞ 1**  
**factorial(0) ➞ 1**

In [26]:

**def** factorial(n):  
 **if** n**==**0:  
 **return** 1  
 **return** n **\*** factorial(n**-**1)  
  
print(f'factorial(5) ➞ {factorial(5)}')  
print(f'factorial(3) ➞ {factorial(3)}')  
print(f'factorial(1) ➞ {factorial(1)}')  
print(f'factorial(0) ➞ {factorial(0)}')

factorial(5) ➞ 120  
factorial(3) ➞ 6  
factorial(1) ➞ 1  
factorial(0) ➞ 1

#### 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

In [31]:

**def** genHamDistance():  
 in\_string\_1 **=** input('Enter the String\_1: ')  
 in\_string\_2 **=** input('Enter the String\_2: ')  
 **if** len(in\_string\_1) **==** len(in\_string\_2):  
 count **=** 0  
 **for** i **in** range(len(in\_string\_1)):  
 **if** in\_string\_1[i] **!=** in\_string\_2[i]:  
 count **=** count**+**1  
 print(f'Hamning Distance b/w {in\_string\_1} and {in\_string\_2} ➞ {count}')  
 **else**:  
 print('Both Strings Must be of Same Length')  
  
for x **in** range(3):  
 genHamDistance()

Enter the String\_1: abcde  
Enter the String\_2: bcdef  
Hamning Distance b/w abcde and bcdef ➞ 5  
Enter the String\_1: abcde  
Enter the String\_2: abcde  
Hamning Distance b/w abcde and abcde ➞ 0  
Enter the String\_1: strong  
Enter the String\_2: strung  
Hamning Distance b/w strong and strung ➞ 1