# Programming Assingment 17

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

In [1]:
```
def evenly_divisible(i,j,k):
    lst=[]
    num=0
    for i in range(i,j+1):
        if i%k ==0:

            lst.append(i)

    for i in lst:
        num +=i
    return num

print("evenly_divisible(1, 10, 2) ➞ ",evenly_divisible(1, 10, 2)  )
print("evenly_divisible(1, 10, 3) ➞ ",evenly_divisible(1, 10, 3)  )
print("evenly_divisible(1, 10, 20) ➞ ",evenly_divisible(1, 10, 20)  )
```

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

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

In [24]:
```
import pdb
def correct_signs(*args):
    #pdb.set_trace()

    arg=str(*args)
    num=''
    nums=[]
    symbol=[]
    result=[]
    arg=arg.replace(' ','')
#seperating numbers in the argument
    for i in range(len(arg)):
        if (arg[i].isdigit()):
            num=num+arg[i]
        elif(not arg[i].isdigit()):
            symbol.append(arg[i])
            nums.append(num)
            num=''
    nums.append(num)

    k=0
    for i in symbol:
```

```
        if(i=='>'):
            result.append(int(nums[k]) > int(nums[k+1]))

        if(i=='<'):
            result.append(int(nums[k]) < int(nums[k+1]))
        if(i=='='):
            result.append(int(nums[k]) == int(nums[k+1]))

        k+=1

    if False in result:
        print (*args, '→', 'False')
    else:
        print (*args, '→', 'True')


correct_signs("13 = 13 > 3 <11<88")
```

```
13 = 13 > 3 <11<88 → 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", "*") → "*shksp**r*"

In [36]:
```
def replace_vowels(words, char):
    sentence=words
    replace_char=char
    sentence_1=''
    vowels=['a','e','i','o','u']
    for i in range(len(sentence)):
        print(sentence[i], i, sentence[i] in vowels )
        if sentence[i] in vowels :
            sentence_1=sentence_1+replace_char
        else:
            sentence_1=sentence_1+sentence[i]
    return sentence_1
print(replace_vowels("the aardvark", "#"))
```

```
t 0 False
h 1 False
e 2 True
  3 False
a 4 True
a 5 True
r 6 False
d 7 False
v 8 False
a 9 True
r 10 False
k 11 False
th# ##rdv#rk
```

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

In [48]:
```
def factorial(n):
```

```
    num=n
    fact=1
    for i in range(1,num+1):
        fact *=i

    return fact
factorial(3)
```

Out[48]: 6

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

In [49]:
```
def hamming_distance(str1, str2):
    first= str1
    second=str2
    distance=0
    for i in range(len(first)):
        if (first[i]==second[i]):
            continue
        else:
            distance +=1
    return distance

hamming_distance("abcde", "bcdef")
```

Out[49]: 5

In [ ]:
```

```