In [12]:
```
# Que1. Write a function that inputs a number and prints the multiplication table

def multi(num):
    """ This function produces multiplication table of a number
    """
    for i in range(1,11):
        print("{}X{} =".format(num,i),i*num)

# Take the number as input
number=int(input("Enter the Number : "))

# Call the multiplication function
multi(number)
```

```
Enter the Number : 19
19X1 = 19
19X2 = 38
19X3 = 57
19X4 = 76
19X5 = 95
19X6 = 114
19X7 = 133
19X8 = 152
19X9 = 171
19X10 = 190
```

In [13]:
```
# Que2. Write a program to print twin primes less than 1000. If two consecutive 
# known as twin primes

#Below code prints the pairs of twin prime numbers below 1000.
lastprime=-1

for i in range(1,1000):
    for j in range(2,i):
        if i%j == 0:
            break
        elif j == i-1 :
            if (lastprime != -1) & (lastprime == i-2):
                print('[',lastprime,i,']')
            lastprime=i
```

```
[ 3 5 ]
[ 5 7 ]
[ 11 13 ]
[ 17 19 ]
[ 29 31 ]
[ 41 43 ]
[ 59 61 ]
[ 71 73 ]
[ 101 103 ]
[ 107 109 ]
[ 137 139 ]
[ 149 151 ]
[ 179 181 ]
[ 191 193 ]
[ 197 199 ]
[ 227 229 ]
[ 239 241 ]
[ 269 271 ]
[ 281 283 ]
[ 311 313 ]
[ 347 349 ]
[ 419 421 ]
[ 431 433 ]
[ 461 463 ]
[ 521 523 ]
[ 569 571 ]
[ 599 601 ]
[ 617 619 ]
[ 641 643 ]
[ 659 661 ]
[ 809 811 ]
[ 821 823 ]
[ 827 829 ]
[ 857 859 ]
[ 881 883 ]
```

In [14]:
```
# Que3. Write a program to find out the prime factors of a number. Example: prime

import math

def isPrime(num):
    """ This function checks whether a number is prime or not and returns True/Fa
    """
    if  (num == 1) or  (num == 2):
        return True
    for i in range(2,num):
        if num %i == 0:
            return False
            break
        elif (i == num-1):
            return True

def primFact(num):
    """ This function finds out prime factors of a number
    """
    primefact=[]
    while not isPrime(num):
        for i in range(2,num+1):
            if num%i == 0:
                primefact.append(i)
                num=num//i
                if isPrime(num):
                    primefact.append(num)
                break
    return primefact

# Take the number as input and print the prime factors
number=int(input("Please enter the number: "))

#Print the prime factors of the number
print("Prime factors of the number {} are : ".format(number), primFact(number))
```

```
Please enter the number: 56
Prime factors of the number 56 are :  [2, 2, 2, 7]
```

In [15]:
```
# Que4. Write a program to implement these formulae of permutations and combinat
# r at a time: p(n, r) = n! / (n-r)!.
# Number of combinations of n objects taken r at a time is: c(n, r) = n! / (r!*(

from functools import reduce

def mult(num1,num2):
    """Returns multiplications of two numbers
    """
    return num1*num2

def pnr(n,r):
    """ Returns npr for a give n & r
    """
    return reduce(mult,range(1,n+1))/reduce(mult,range(1,n-r+1))

def cnr(n,r):
    """ Returns ncr for a give n & r
    """
    return pnr(n,r)/reduce(mult,range(1,r+1))

# Take inputs
num=int(input("Please enter number of Objects n : "))
fre=int(input("Please enter the no. of Objects taken at a time : "))

# print npr and ncr outputs
print('Number of permutations p({},{}) = '.format(num,fre),pnr(num,fre))
print('Number of combinations c({},{}) ='.format(num,fre),cnr(num,fre))
```

```
Please enter number of Objects n : 50
Please enter the no. of Objects taken at a time : 3
Number of permutations p(50,3) =  117600.0
Number of combinations c(50,3) = 19600.0
```

In [16]:
```
# Que5. Write a function that converts a decimal number to binary number

import numpy as np

binnum=[]
def dectobin(num):
    """ This finction take a number and produces its binary equivalent
    """
    binnum.append(int(num%2))
    num=np.floor(num/2)
    return binnum if num == 0 else dectobin(num)

# Take number as input
num=int(input("Enter the number : "))

# Print binary equivalent of the number entered
print("Binary equivalent of decimal {} is : ".format(num),dectobin(num)[::-1])
```

```
Enter the number : 23
Binary equivalent of decimal 23 is :  [1, 0, 1, 1, 1]
```

In [17]:
```
# Que6. Write a function cubesum() that accepts an integer and returns the sum of
# Use this function to make functions PrintArmstrong() and isArmstrong() to print
# Armstrong number.

def cubesum(num):
    """ This function returns the sum of cube of digits of a number
    """
    cube=0
    for i in str(num):
        cube = cube + int(i)**3
    return(cube)

def isArmstrong(num):
    """ This function checks whether a number is Armstrong number or not
    """
    if num == cubesum(num):
        return True

def PrintArmstrong(num):
    """ This function prints all Armstrong number beolow the number provided
    """
    armst=[]
    for i in range(1,num+1):
        if isArmstrong(i):
            armst.append(i)
    return armst

# Take the number as input
num=int(input("Enter the range for Armstrong Number : "))

# Print Armstrong number
print("Armstrong numbers are :",PrintArmstrong(num))
```

```
Enter the range for Armstrong Number : 300
Armstrong numbers are : [1, 153]
```

In [67]:
```
# Que7. Write a function prodDigits() that inputs a number and returns the produc

def prodDigits(num):
    """ This function returns the product of digits of a number
    """
    prod=1
    for i in str(num):
        prod=int(i)*prod
    return prod

# Take the number as input
num=int(input("Enter a number : "))

# Print product of digits
print("Product of Digits is : ",prodDigits(num))
```

```
Enter a number : 234
Product of Digits is :  24
```

In [69]:
```
# Que8. If all digits of a number n are multiplied by each other repeating with t
#last is called the multiplicative digital root of n. The number of times digits
#called the multiplicative persistance of n.
#Example: 86 -> 48 -> 32 -> 6 (MDR 6, MPersistence 3)
# 341 -> 12->2           (MDR 2, MPersistence 2)
# Using the function prodDigits() of previous exercise write functions MDR() and
# its multiplicative digital root and multiplicative persistence respectively


def MDR(num):
    """Returns the multiplicative digital root of a number using prodDigits() fu
    """
    return num if len(str(num)) == 1 else MDR(prodDigits(num))

def MPersistence(num,pers):
    """Returns the multiplicative persitence of a number using prodDigits() funct
    """
    return pers if len(str(num)) == 1 else MPersistence(prodDigits(num),pers+1)

# Take the number as input.
number=int(input("Enter  The Number: "))

#Print MDR of the number
print("MDR of {} is : ".format(number),MDR(number))

#Print MPersistence of the number
print("MPersistence of {} is : ".format(number),MPersistence(number,0))
```

```
Enter  The Number: 2347
MDR of 2347 is :  6
MPersistence of 2347 is :  4
```

In [33]:
```
# Que9. Write a function sumPdivisors() that finds the sum of proper divisors of
# those numbers by which the number is divisible, except the number itself.  For
# 1, 2, 3, 4, 6, 9, 12, 18

def sumPdivisors(num):
    """This function finds out the sum of the proper divisors of a given number
    """
    sumpdiv=0
    proplist=[]

    for i in range(1,num):
        if num%i == 0:
            proplist.append(i)
            sumpdiv=i+sumpdiv
    return  sumpdiv


#Take the number as input
num=int(input("Enter the number : "))

#Print sum of Proper Divisors
print("sum of Proper divisors of {} is : ".format(num),sumPdivisors(num))
```

```
Enter the number : 284
sum of Proper divisors of 284 is :  220
```

In [81]:
```
# Que10. A number is called perfect if the sum of proper divisors of that number
# perfect number, since 1+2+4+7+14=28. Write a program to print all the perfect n


def printperfect(low,upp):
    """ This function checks whether a number is perfect or not
    """
    perf=[]
    for i in range(low,upp+1):
        if i == sumPdivisors(i):
            perf.append(i)
    return perf

# Take the range numbers as input
low=int(input("Enter the lower range : "))
upp=int(input("Enter the upper range : "))

# Print all the perfect numbers in the range
print("Perfect numbers in range {}-{} are : ".format(low,upp),printperfect(low,up
```

```
Enter the lower range : 10
Enter the upper range : 100
Perfect numbers in range 10-100 are :  [28]
```

In [36]:
```
# Que11. Two different numbers are called amicable numbers if the sum of the prop
# number. For example 220 and 284 are amicable numbers. Sum of proper divisors of
# 220 = 1+2+4+5+10+11+20+22+44+55+110 = 284 Sum of proper divisors of
# 284 = 1+2+4+71+142 = 220 Write a function to print pairs of amicable numbers in


def amic(low,upp):
    """ This function finds the amicable numbers in a given range
    """
    amic=[]
    for i in range(low,upp+1):
        for j in range(low,upp+1):
            if ((sumPdivisors(i) == j) & (sumPdivisors(j) == i)):
                amic.append((i,j))
    return amic

# Enter the range limits
low=int(input("Enter the lower range : "))
upp=int(input("Enter the upper range : "))

# Print Amicable numbers
print("Amicable numbers in range {}-{} are : ".format(low,upp), amic(low,upp))
```

```
Enter the lower range : 500
Enter the upper range : 1000
Amicable numbers in range 500-1000 are :  []
```

In [24]:
```
# Que12. Write a program which can filter odd numbers in a list by using filter f

def checkeven(num):
    """ This function checks whether a number is even or not and returns True/Fal
    """
    if num%2 == 0:
        return True

#List to check
numberlist=[1,2,3,4,5,6,7,8,9,10]

#Print filtered odd numbers list
print("Filtered odd numbers list is : " ,list(filter(checkeven,numberlist)))
```

```
Filtered odd numbers list is :  [2, 4, 6, 8, 10]
```

In [22]:
```
# Que13. Write a program which can map() to make a list whose elements are cube o

def cube(num):
    """ This function returns the cube of a number
    """
    return num**3

#List to map
numbers=[1,2,3,4,5,6,7,8,9,10]

# Print cube elemnets
print("Mapped Cube list is : " , list(map(cube,numbers)))
```

```
Mapped Cube list is :  [1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

In [37]:
```
# Que14. Write a program which can map() and filter() to make a list whose elemen

numbers=[1,2,3,4,5,6,7,8,9,10]

print("Cube of even numbers of the list is :", list(map(cube,list(filter(checkeve
```

```
Cube of even numbers of the list is : [8, 64, 216, 512, 1000]
```