In [27]:
```
import pandas as pd
import numpy as np
```

1. Write a function that inputs a number and prints the multiplication table of that number

In [1]:
```
print("***calculating the multiplication table***")
x=int(input("Please Provide a number"))
for i in range(1,11):
    print("{} x {} = {}".format(i,x,(x*i)))
```

```
***calculating the multiplication table***
Please Provide a number5
1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
4 x 5 = 20
5 x 5 = 25
6 x 5 = 30
7 x 5 = 35
8 x 5 = 40
9 x 5 = 45
10 x 5 = 50
```

1. Write a program to print twin primes less than 1000. If two consecutive odd numbers are both prime then they are known as twin primes

In [5]:
```
print("***Printing Twin Primes below 1000***")

def check_prime(num):
    for i in range(2,num//2):
        if(num % i == 0):
            return 0
    return 1
```

```
for i in range(3,1000,2):
    if(check_prime(i) & check_prime(i+2)):
        print("({},{})".format(i,i+2))
```

```
***Printing Twin Primes below 1000***
(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)
```

1. Write a program to find out the prime factors of a number. Example: prime factors of 56 - 2, 2, 2, 7

In [30]:
```
from functools import reduce

def check_prime(num):
    for i in range(2,num//2):
        if(num % i == 0):
            return 0
    return 1

def next_prime(num):
    n=num
    if (num <=2):
        return num+1
    elif (num % 2 == 0):
        return -1
    else:
        while True:
            if (check_prime(n+2) == 1):
                return (n+2)
            else :
                n =n +2

num=int(input("Enter a number to find the prime factors:"))

i=2
factors=[]
n=num
while(i<n//2):
    if(num % i == 0):
        factors.append(i)
        num = num /i
    else:
        i=next_prime(i)
```

```
if(factors and reduce((lambda x, y: x * y), factors) == n):
    print("Prime Factors of {} are: {}".format(n,factors))
else:
    print("{} can not be prime factorized".format(n))
```

```
Enter a number to find the prime factors:56
Prime Factors of 56 are: [2, 2, 2, 7]
```

1. Write a program to implement these formulae of permutations and combinations. Number of permutations of n objects taken 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!*(n-r)!) = p(n,r) / r!

In [17]:
```
def factorial(num):
    fact = 1
    for i in range(1,num+1):
       fact*=i
    return fact

print("Enter value for n & r for calculating P(n,r) and C(n,r)")
n,r=int(input("n=")),int(input("r="))
pnr=factorial(n)/factorial(n-r)
cnr=pnr/factorial(r)
print("P({},{})={}".format(n,r,pnr))
print("C({},{})={}".format(n,r,cnr))
```

```
Enter value for n & r for calculating P(n,r) and C(n,r)
n=5
r=2
P(5,2)=20.0
C(5,2)=10.0
```

1. Write a function that converts a decimal number to binary number

In [65]:
```
def conv_to_bin(num):
    n=num
    str1=""
    while (n >= 1):
```

```
        str1=str1+str(n % 2)
        n=n//2

    return (str1[::-1])

num=int(input("Enter a number for converting to binary:"))
print("{} in binary ={}".format(num,conv_to_bin(num)))
```

```
Enter a number for converting to binary:5
5 in binary =101
```

1. Write a function cubesum() that accepts an integer and returns the sum of the cubes of individual digits of that number. Use this function to make functions PrintArmstrong() and isArmstrong() to print Armstrong numbers and to find whether is an Armstrong number.

In [3]:
```
import math
def cubesum(num):
    n=num
    sum=0
    while(n>0):
        sum+=math.pow((n%10),3)
        n//=10
    return sum

def isArmstrong(num):
    if (cubesum(num) == num):
        return 1
    else:
        return 0

def PrintArmstrong(num):
    if(isArmstrong(num)):
        print("{} is an Amtrong number".format(num))
    else:
        print("{} is not an Amtrong number".format(num))

x=int(input("Please Enter a number to check if that is an amstrong numb
```

```
er:"))
PrintArmstrong(x)
```

```
Please Enter a number to check if that is an amstrong number:153
153 is an Amtrong number
```

1. Write a function prodDigits() that inputs a number and returns the product of digits of that number.

In [3]:
```
def prodDigits(num):
    prod=1
    n=num
    while(n>0):
        prod*=n%10
        n//=10
    return prod

x=int(input("Please enter a number :"))
print("Product of all digits of {} = {}".format(x,prodDigits(x)))
```

```
Please enter a number :56
Product of all digits of 56 = 30
```

1. If all digits of a number n are multiplied by each other repeating with the product, the one digit number obtained at last is called the multiplicative digital root of n. The number of times digits need to be multiplied to reach one digit is 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 MPersistence() that input a number and return its multiplicative digital root and multiplicative persistence respectively

In [8]:
```
def MDR(num):
    n=num
    while(n>=9):
        n=prodDigits(n)
    return n
```

```
def MPersistence(num):
    n=num
    cnt=0
    while(n>=9):
        n=prodDigits(n)
        cnt+=1
    return cnt

x=int(input("Please enter a number :"))
print("multiplicative digital root of {0} = {1}\nmultiplicative persist
ence of {2} = {3}".format(x,MDR(x),x,MPersistence(x)))
```

```
Please enter a number :86
multiplicative digital root of 86 = 6
multiplicative persistence of 86 = 3
```

1. Write a function sumPdivisors() that finds the sum of proper divisors of a number. Proper divisors of a number are those numbers by which the number is divisible, except the number itself. For example proper divisors of 36 are 1, 2, 3, 4, 6, 9, 18

In [21]:
```
def sumPdivisors(num):
    prop_div=0
    for i in range(1,(num//2)+1):
        if(num % i ==0):
            prop_div+=i
    return prop_div

x=int(input("Please enter a number :"))
print("Sum of Proper diviors of {} is={}".format(x,sumPdivisors(x)))
```

```
Please enter a number :284
Sum of Proper diviors of 284 is=220
```

1. A number is called perfect if the sum of proper divisors of that number is equal to the number. For example 28 is perfect number, since 1+2+4+7+14=28. Write a program to print all the perfect numbers in a given range

In [17]:
```
print("***Program to print perfect numbers in a range***")
low,high=int(input("Please Provide range start value=")),int(input("Ple
ase Provide range End value="))

print("The Perfect numbers in this range-")
for i in range(low,high+1):
    if(i == sumPdivisors(i)):
        print(i)
```

```
***Program to print perfect numbers in a range***
Please Provide range start value=1
Please Provide range End value=500
The Perfect numbers in this range-
6
28
496
```

1. Two different numbers are called amicable numbers if the sum of the proper divisors of each is equal to the other 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 a range

In [24]:
```
def amicable(num):
    sum=sumPdivisors(num)
    if(sum >num and num!=sum and sumPdivisors(sum) == num):
        print("({},{})".format(num,sum))
    return

print("***Program to print amicable numbers in a range***")
low,high=int(input("Please Provide range start value=")),int(input("Ple
ase Provide range End value="))

print("The Amicable numbers in this range-")
for i in range(low,high+1):
    amicable(i)
```

```
***Program to print amicable numbers in a range***
Please Provide range start value=5
Please Provide range End value=1000
The Amicable numbers in this range-
(220,284)
```

1. Write a program which can filter odd numbers in a list by using filter function

In [44]:
```
def oddlistfunc(n):
    if(n%2 == 1):
        return True
    else:
        return False

xlist=np.random.randint(0,100,15)
print("List:",xlist)
newlist=list(filter(oddlistfunc,xlist))
print("After Filtering odd list is=",newlist)
```

```
List: [10 67 65 40 21 20 14  5 86 88 61 88 12 88 61]
After Filtering odd list is= [67, 65, 21, 5, 61, 61]
```

1. Write a program which can map() to make a list whose elements are cube of elements in a given list

In [43]:
```
xlist=np.random.randint(1,26,5)
print("List:",xlist)
newlist=list(map(lambda x:x**2,xlist))
print(newlist)
```

```
List: [ 1 12  4 10 15]
[1, 144, 16, 100, 225]
```

1. Write a program which can map() and filter() to make a list whose elements are cube of even number in a given list

In [53]:
```
def evenlistfunc(n):
    if(n%2 == 0):
        return True
    else:
        return False

xlist=np.random.randint(1,15,5)
print("List:",xlist)

finallist=list(map(lambda x:x**3,list(filter(evenlistfunc,xlist))))
print(finallist)
```

```
List: [ 8  2  5 10  7]
[512, 8, 1000]
```