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

In [1]:
```
def multiplication_table(n):
    for i in range(1,11):
        print(i*n)
n=int(input("Enter the number "))
multiplication_table(n)
```

```
Enter the number 5
5
10
15
20
25
30
35
40
45
50
```

2. 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 [6]:
```
def isprime(num):
    for i in range(2,num):
        if num % i == 0:
            return False
    return True

print("Twins prime less than 1000 :")
for i in range(2,1001):
    j=i+2
    if isprime(i) and isprime(j):
        print("({0},{1})".format(i,j))
```

```
Twin prime less than 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)
```

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

In [1]:
```
import math  #Refrence geeksforgeeks
def prime_factors(n):
    while n%2==0: #if the number is even
```

```
        print(2)
        n=n/2


    for i in range(3,int(math.sqrt(n))+1,2):    #if the number is odd
        while n%i==0:
            print(i)
            n/=i

    if(n>2):  #if the number is prime and greater than 2
        print(n)

num=int(input("Enter the number : "))
prime_factors(num)
```

```
Enter the number : 56
2
2
2
7.0
```

4. 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 [11]:
```
def factorial(n):
    if(n<=1):
        return 1
    else:
        return n*factorial(n-1)

n=int(input("Enter the value of n :"))
r=int(input("Enter the value of r :"))

nPr=factorial(n)/factorial(n-r)   #permutation calculation
nCr=nPr/factorial(n)              #combination calculation

print("Permutation is : {} ".format(nPr))
print("Combination is : {} ".format(nCr))
```

```
Enter the value of n :5
Enter the value of r :2
Permutation is : 20.0
Combination is : 0.16666666666666666
```

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

In [1]:
```
def decimal_to_binary(num):
    bin=' ' #empty string
    while(num>0):
        if(num%2==0):
            bin+='0'
        else:
            bin+='1'
        num=num//2
    binary_num=''.join(reversed(bin))
    print(binary_num)
decimal_to_binary(10)
```

```
1010
```

6. 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 [48]:
```
def cubesum(n):
    sum=0
    while n>0:
        rem=n%10
        sum=sum+rem**3
        n=n//10
    return sum



def isArmstrong(n):
    if n==cubesum(n):
        print("{} is a Armstrong Number".format(n))
    else:
        print("{} is not a Armstrong Number".format(n))
```

```
isArmstrong(153)


print("Armstrong Numbers less than 1000")
def PrintArmstrong():
    for i in range(1,1001):
        if i==cubesum(i):
            print(i)
PrintArmstrong()
```

```
153 is a Armstrong Number
Armstrong Numbers less than 1000
1
153
370
371
407
```

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

In [2]:
```
def prodDigits(n):
    prod=1
    while n>0:
        rem=n%10
        prod=prod*rem
        n=n//10
    return prod
num=int(input("Enter a number "))
prodDigits(num)
```

```
Enter a number 56
```

Out[2]:
```
30
```

8. 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 [3]:
```
def prodDigits(n):
    prod=1
    while n>0:
        rem=n%10
        prod=prod*rem
        n=n//10
    return prod

def MDR():
    k= int(input("Enter the number "))
    while(k//10 >0):
        k=prodDigits(k)
    print("MDR is "+str(k))

MDR()

def MPersistance():
    i=0
    k= int(input("Enter the number "))
    while(k//10 >0):
        k=prodDigits(k)
        i=i+1
    print("MPersistance is "+str(i))
MPersistance()
```

```
Enter the number 341
MDR is 2
Enter the number 341
MPersistance is 2
```

9. 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, 12,18

In [60]:
```
def sumPdivisors(n):
    sum=0
    for i in range(1,n):
        if n%i==0:
            sum=sum+i
```

```
    return sum
sumPdivisors(36)
```

Out[60]: 55

10. 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 [2]:
```
def sumPdivisors(n):
    sum=0
    for i in range(1,n):
        if n%i==0:
            sum=sum+i
    return sum

lst=[]
for i in range(1,1000):
    if sumPdivisors(i)==i:
        lst.append(i)
print(lst)
```

```
[6, 28, 496]
```

11. 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 [5]:
```
def divisors_sum(n):
    sum=0
    for i in range(1,n):
        if n%i==0:
            sum+=i
    return sum

lst=[ ]
n=int(input("Enter the range "))
for i in range(1,n+1):
    for j in range(i+1,n+1):
        if divisors_sum(i)==j and divisors_sum(j)==i:
```

```
            lst.append([i,j])
print(lst)
```

```
Enter the range 3000
[[220, 284], [1184, 1210], [2620, 2924]]
```

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

In [69]:
```
def find_odd_num(num):
    if(num%2!=0):
        return num

lst=[1,2,3,4,5,6,7,8,9,10]
odd_num_list=list(filter(find_odd_num,lst))
print(odd_num_list)
```

```
[1, 3, 5, 7, 9]
```

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

In [70]:
```
def calc_cube(num):
    return num**3
lst=[1,2,3,4,5,6,7,8,9,10]
cube_list=list(map(calc_cube,lst))
print(cube_list)
```

```
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

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

In [72]:
```
def find_even_num(num):
    if(num%2==0):
        return num

def calc_cube(num):
    return num**3

lst=[1,2,3,4,5,6,7,8,9,10]
even_num_list=list(filter(find_even_num,lst))
cube_list=list(map(calc_cube,even_num_list))
print(cube_list)
```

```
[8, 64, 216, 512, 1000]
```