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

In [1]:

```
#static input

number=int(input("enter the number for the table :"))

#using for loop
for i in range (1,11):
    print("{} x {} = {}".format(number,i,number*i))
```

```
enter the number for the table :11
11 x 1 = 11
11 x 2 = 22
11 x 3 = 33
11 x 4 = 44
11 x 5 = 55
11 x 6 = 66
11 x 7 = 77
11 x 8 = 88
11 x 9 = 99
11 x 10 = 110
```

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 [1]:

```
def printTwinPrime(n):


    prime= [True for i in range(n+2)]
    p=2


    while (p*p<=n+1):


        if (prime[p]==True):
            for i in range (p*2,n+2,p):
                prime[i]=False
        p+=1

    for p in range(2,n-1):
        if prime[p] and prime[p + 2]:
            print("(",p,",",(p+2),")",end='')


if __name__=='__main__':

        n=1000



        printTwinPrime(n)
```

```
( 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 )( 28
1 , 283 )( 311 , 313 )( 347 , 349 )( 419 , 421 )( 431 , 433 )( 461 , 463 )( 521 , 523 )( 569 , 571 )( 599 , 6
01 )( 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

In [2]:

```
import math


def primeFactors(n):


    while n%2==0:
        print("2") ,
        n=n/2
    for i in range(3,int(math.sqrt(n))+1,2):

        while n%i==0:
            print (i),
            n=n/i

    if n>2:
        print (n)



n=int(input("Enter the any nuumber:"))
primeFactors(n)
```

```
Enter the any nuumber:12
2
2
3.0
```

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!

CODE FOR THE COMBINATIONS

In [7]:

```
#defining the function



def nCr(n,r):
    return(fact(n)/(fact(r)*fact(n-r)))

def fact(n):

    res = 1

    for i in range (2,n+1):
        res=res*i
    return res

#dynamic input

n=int(input("enter the 'n ' valve:"))
r=int(input("enter the 'r' valve:"))


#print the values
print(int(nCr(n,r)))
```

```
enter the 'n ' valve:15
enter the 'r' valve:4
1365
```

CODE FOR THE permutations

In [8]:

```
#defining the function


def nPr(n,r):
    return(fact(n)/fact(n-r))

def fact(n):

    res = 1

    for i in range (2,n+1):
        res=res*i
    return res




#dynamic input
n=int(input("enter the 'n ' valve:"))
r=int(input("enter the 'r' valve:"))


#printing the output
print(int(nPr(n,r)))
```

```
enter the 'n ' valve:15
enter the 'r' valve:4
32760
```

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

In [10]:

```
#defining the function
#here dtb means decimal to binary

def dtb(num):
    if num>1:
        dtb(num//2)
    print(num%2,end='')

num=int(input("enter any decimal no:"))

dtb(num)
```

```
enter any decimal no:45
101101
```

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 [8]:

```
#https://www.programiz.com/python-programming/examples/armstrong-number

num=int(input("Enter the number : "))
order=len(str(num))

#inititating the sum
sum=0
#finding the sum of cubes of ecah digits
temp=num
while temp>0:
    digit=temp%10
    sum+=digit**order
    temp//=10


  #printing the results

if num==sum:
    print(num,"is an Armstrong number.")
else:
        print(num,"is not an armstrong number.")
```

```
Enter the number : 153
153 is an Armstrong number.
```

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

In [22]:

```
def prodOfDigits(num):
    product = 1

    while (num!= 0):
        product = product * (num % 10)
        num = num // 10

    return product

# Dynamic input
num = int(input("Enter  the number :  "))
print(prodOfDigits(num))
```

```
Enter  the number :  1214
8
```

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 [ ]:

```
#https://codereview.stackexchange.com/questions/156769/repeatedly-multiplying-digits-until-a-single-digit-is-obtained
```

In [37]:

```
def getMDRAndMPersistence(num):  #defing the function of mrd and mpersistence
    count=0
    while(True):
        count += 1
        prodOfDigits = prodDigits(num)
        if(prodOfDigits < 10):
            return [prodOfDigits, count]
        num = prodOfDigits

    return [prodOfDigits, count]

def MDR(num):
   #defining the mrd
    if(num < 10):
        return num
    return getMDRAndMPersistence(num)[0]
def MPersistence(num):
    if(num <= 10):                              #defining the m presistance
        return 1
    return getMDRAndMPersistence(num)[1]

#dynmaic input

num = int(input('Enter a valid number : '))
#printing the results
print('MRD',MDR(num),'\n','Mpersistence',MPersistence(num))
```

```
Enter a valid number : 341
MRD 2
 Mpersistence 2
```

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 [23]:

```
#Vhttps://www.geeksforgeeks.org/sum-of-all-proper-divisors-of-a-natural-number/

import math
def sumPdivisors(num):
    result=0
    i=2
    while i<= (math.sqrt(num)) :
        if (num % i==0) :
            if (i == (num/i)) :
                result = result+i;
            else :
                result =result+(i + num/i);
        i = i +1
    return (result + 1);
num=int(input("Enter the  number : "))
print(sumPdivisors(num))
```

```
Enter the  number : 36
55.0
```

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 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 [44]:

```
#https://www.geeksforgeeks.org/perfect-number/
def isPerfect( num ):
                  # To store sum of divisors
    sum = 1
 # Find all divisors and add them
    i = 2
    while i * i <= num:
        if num % i == 0:
            sum = sum + i + num/i
        i += 1
 # If sum of divisors is equal to
    # n, then n is a perfect number

    return (True if sum == num and num!=1 else False)

# static input

print("Below are all perfect numbers till 10000")
num = 2
for num in range (1000):
    if isPerfect (num):
        print(num , " is a perfect number")
```

```
Below are all perfect numbers till 10000
6  is a perfect number
28  is a perfect number
496  is a perfect number
```

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.

In [12]:

```
#https://www.sanfoundry.com/python-program-check-numbers-amicable/

x=int(input('Enter number 1: '))
y=int(input('Enter number 2: '))
sum1=0
sum2=0
for i in range(1,x):
    if x%i==0:
        sum1+=i
for j in range(1,y):
    if y%j==0:
        sum2+=j
if(sum1==y and sum2==x):
    print(x,'&',y,'are Amicable numbers!.')
else:
    print(x,'&',y, 'are Not Amicable numbers!.')
```

```
Enter number 1: 220
Enter number 2: 284
220 & 284 are Amicable numbers!.
```

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

In [1]:

```
#https://www.geeksforgeeks.org/filter-in-python/

a=[1,2,3,4,5,6]
r=filter(lambda x: x % 2, a)
print(list(r))
```

```
[1, 3, 5]
```

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

In [66]:

```
#https://www.geeksforgeeks.org/filter-in-python/

lst=[1,2,3,4,5]
sqrdNum=map(lambda x: x**3,lst)
for i in sqrdNum:
    print(i)
```

```
1
8
27
64
125
```

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 [3]:

```
#https://www.geeksforgeeks.org/filter-in-python/


lst=[1,2,3,4,5,6,8,9,10,19,18,17]
EvenNum=list(filter(lambda x: (x%2==0),lst))
print(EvenNum)
sqrdNum=list(map(lambda x: x**3,EvenNum))
for i in sqrdNum:
    print(i)
```

```
[2, 4, 6, 8, 10, 18]
8
64
216
512
1000
5832
```

In [ ]:

```
#reference from
#greeksforgeeks
#stackoverflow
#stackexchange
#and some python cheat codes
```

In [ ]:

In [ ]: