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

In [1]:

```
# 1. Code of A function that inputs a number and prints the multiplication table of that number
def multiplier():

    '''This function prints the multiplication table of a number'''

    n=int(input("enter the number: "))

    for i in range(1,11):
        c=n*i
        print("{}*{}={}".format(n,i,c))
multiplier()
```

```
enter the number: 4
4*1=4
4*2=8
4*3=12
4*4=16
4*5=20
4*6=24
4*7=28
4*8=32
4*9=36
4*10=40
```

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 [2]:

```
2.# a program to print twin primes less than 1000. If two consecutive odd numbers are
#both prime then they are known as twin primes
a=1
b=1000
lst=[]
for no in range(a,b):
    if no>1:
        isdivisible=False
        for i in range(2,no):
            if no % i==0:
                isdivisible=True
        if not isdivisible:
            lst.append(no)
#print("Total prime numbers less than 1000 are:\n",lst)
print('\n ***twin primes are:')
for i in range(0,len(lst)-1):
    if lst[i+1] - lst[i]==2:
        print(lst[i],lst[i+1])
```

```
 ***twin primes are:
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. A program to find out the prime factors of a number. Example: prime factors of 56 -2, 2, 2, 7

In [3]:

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

num=int(input('enter a number: '))
number=num
lst=[]
a=2
while number>1:
    if number%a==0:
        lst.append(a)
        number=number/a
    else:
        a+=1

print('prime factors of', num, 'are:',lst)
```

```
enter a number: 56
prime factors of 56 are: [2, 2, 2, 7]
```

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 [4]:

```
#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!

def facto(n):
    fact_value=1

    for ele in range(1,n+1):
        fact_value=ele*fact_value
    return fact_value

def permtn(n,r):
```

```
    npr=facto(n)/facto(n-r)   #p(n,r)=n!/(n-r)!
    print('npr value is:',npr)


def combn(n,r):
    ncr=facto(n)/(facto(r)*facto(n-r))    #c(n,r)=n!/(r!*(n-r)!)
    print('\nncr value is:',ncr)

permtn(4,2)
combn(4,2)
```

```
npr value is: 12.0

ncr value is: 6.0
```

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

In [5]:

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

def dec_bin(n):
    bin_no=[]
    k=0
    while n>0:
        no=n%2
        bin_no.append(no)
        #print (bin_no)
        n=int(n/2)
        #print(n)
        k+=1
        #print(k)

    #print(bin_no)
    print(list(reversed(bin_no)))

dec_bin(32)
```

```
[1, 0, 0, 0, 0, 0]
```

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 [6]:

```
def cubesum(n):
        sum=0
        num=n
        while num>0:
            num1=num%10
            num2=num1**3
            sum=sum+num2
            num=num//10
        return sum

def PrintArmstrong(n):
    totalsum=cubesum(n)
    if n==totalsum:
        return n

def isArmstrong(n):
    if n==PrintArmstrong(n):
        print(n, 'is an Armstrong number')
    else:
```

```
    else:
        print(n, 'is not an Armstrong number')

isArmstrong(300)
isArmstrong(370)
```

```
300 is not an Armstrong number
370 is an Armstrong number
```

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

In [7]:

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

def prodDigits(n):

    num=n
    prod=1
    while num>=1:
        d=num%10
        prod*=d
        num=num//10
    return prod


prodDigits(86)
```

Out[7]:

```
48
```

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 [8]:

```
def MDR(n):
    l=len(str(n))
    MDR1=n
    while l>1:
        if MDR1>0:
            MDR1=prodDigits(MDR1)
        l=len(str(MDR1))
    print ('MDR is:', MDR1)

MDR(86)


def MPersistence(n):
    count=0
    l=len(str(n))
    MDR1=n
    count=0
```

```
    count=0
    while l>1:
        if MDR1>0:
            MDR1=prodDigits(MDR1)
            count+=1
        l=len(str(MDR1))
    print('MPersistence is: ', count)

MPersistence(86)
```

```
MDR is: 6
MPersistence is:  3
```

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.

In [12]:

```
def sumPdivisors(n):
    sum=0
    i=1
    #print('proper divisors are: ')
    while i<n:
        if n%i==0:
            #print(i)
            sum=sum+i
        i+=1
    print('Sum of proper divisors of {} is : {}'.format(n,sum))

sumPdivisors(25)
```

```
Sum of proper divisors of 25 is : 6
```

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 [13]:

```
a=int(input('enter the number: '))
b=int(input('enter a number greater than a : '))

for i in range (a,b):
    sum_div=0
    for j in range(1,i):
        if i%j==0:
            sum_div+=j
            if sum_div==i:
                print(i)
```

```
enter the number: 0
enter a number greater than a : 30
6
24
28
```

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 [14]:

```
def sumPdivisors(n):
    sum=0
    i=1
    #print('proper divisors are: ')
    while i<n:
        if n%i==0:
            #print(i)
            sum=sum+i
        i+=1
    return sum

def amicable(a,b):
    s1=sumPdivisors(a)
    s2=sumPdivisors(b)

    if s1==b and s2==a:
        return True
    else:
        return False

def amicablepair(p,q):
    pairs=0
    if i in range(p,q):
        for j in range(i+1,q):
            if amicable(i,j):
                print(i,j)
                pairs=pairs+1
    print('total amicable pairs:',pairs)

amicable(220,284)
#amicablepair(100,300)
```

Out[14]:

```
True
```

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

In [15]:

```
list1=[1,2,3,4,5,6,7,8]

odd_num=filter(lambda x: x%2!=0, list1)
print(list(odd_num))
```

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

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

In [16]:

```
list2=[1,2,3,4,5,6,7,8,9,10]

cube_ele=map(lambda x: x**3, list2)
print(list(cube_ele))
```

```
[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 [17]:

```
list3=[1,2,3,4,5,6]

even=filter(lambda a:a%2==0,list3)

cube_even=map(lambda a:a**3, even)
print(list(cube_even))
```

```
[8, 64, 216]
```