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

In [1]:
```
def multiplication_table():
    number = int(input("enetr the number:"))

    for i in range(1, 11):
        print("{0} * {1} = {2}".format(number, i, (number * i)))
multiplication_table()
```

```
enetr the number:34
34 * 1 = 34
34 * 2 = 68
34 * 3 = 102
34 * 4 = 136
34 * 5 = 170
34 * 6 = 204
34 * 7 = 238
34 * 8 = 272
34 * 9 = 306
34 * 10 = 340
```

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 prime_numbers(number):
    for i in range(3,number):
        if number % i == 0:
            return False
    return True

def twin_prime(start,end):
    for num in range(start,end):#iterating over numbers from start to end
        if(prime_numbers(num) and prime_numbers(num+2)):#Here we call the above j

#Here we check cosecutive odd numbers is prime or not, by passing these two separ
            print('{0} and {1}'.format(num,num+2))
twin_prime(3,1001)
```

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

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

In [10]:
```
#take input from the user
num = int(input('enter a number'))
def factors (n):

    while n % 2 == 0:
        print(2)
        n = n/2

    for i in range (3,int(n),2):
        if n % i == 0:
            print(i)
            n = n/i

    if n > 2:
        print(n)

factors (num)
```

```
enter a number121
11
11.0
```

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 [4]:
```
import math
n=6
r=3
def nCr(n,r):
    return(math.factorial(n)/(math.factorial(r)*math.factorial(n-r)))

def nPr (n,r):
    return(math.factorial(n)/math.factorial(n-r))

print(nPr(n,r))

print(nCr(n,r))
```

```
120.0
20.0
```

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

In [5]:
```
def decimalToBinary(num):
    """This function converts decimal number
    to binary and prints it"""
    if num > 1:
        decimalToBinary(num // 2)  #it's a recursive function
    print(num % 2,end = '')

number = int(input("Enter the decimal number: "))  #decimal number

decimalToBinary(number)
```

```
Enter the decimal number: 34
100010
```

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]:
```
num = int(input('enter a number'))
sum = 0
a = num
def cubesum(n):
    global sum
    while n > 0:
        var = n%10
        sum += var**3
        n = n//10
    if a == sum:
        return sum

def printarmstrong():
    if cubesum(num):
        print('the entered number:{} is an armstrong'.format(num))

    else:
        print('the entered number:{} is not an armstrong'.format(num))

printarmstrong()
```

```
enter a number54
the entered number:54 is not an armstrong
```

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

In [7]:
```
num = int(input('enter a number:'))
product = 1
def prodDigits(n):
    global product
    while n > 0:
        var = n % 10
        product *= var
        n //= 10
    return product
prodDigits(num)
```

```
enter a number:67
```

Out[7]: 42

8.If all digits of a number n are multiplied by each other repeating with the product, the digit number obtainedone 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 prodDigits(n):
    product = 1
    while n > 0:
        var = n % 10
        n //= 10
        product *= var
    return product
num = int(input('enter a number'))
def MDR():
    global num
    count = 0
    while num >= 10:
        num = prodDigits(num)
        count += 1
    print('mdr:{0}  mpersistence:{1}'.format(num,count))

MDR()
```

```
enter a number87
mdr:0  mpersistence: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. For example proper divisors of 36 are 1, 2, 3, 4, 6, 9, 18

In [9]:
```
num = int(input('enter anumber'))
def sumPdivisors(n):
    sum = 0
    for i in range(1,n):
        if n % i == 0:
            sum += i
    print(sum)

sumPdivisors(num)
```

```
enter anumber45
33
```

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 [10]:
```
num = int(input('enter anumber'))
def perfect(n):
    sum = 0
    for i in range(1,n):
        if n % i == 0:
            sum += i
    if sum == n:
        print('the given number:{} is a perfect number'.format(n))
    else:
        print('the given number:{} is not a perfect number'.format(n))

perfect(num)
```

```
enter anumber37
the given number:37 is not a perfect number
```

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 [11]:
```
#print all the pairs of amicable numbers between 1 and 2000
def perfect(x):
    sum = 0
    for i in range(1,x):
        if x % i == 0:
            sum += i
    return sum
def isamicable(a,b):
    if perfect(a) == b and perfect(b) == a:
        return True
    else:
        return False
#This function print pairs
def Amicable_Pairs(a,b):
    for i in range(0,b):
        for j in range(i + 1,b):
            if isamicable(i,j):
                print('{0} and {1}'.format(i,j))

num1 = 1
num2 = 2000
Amicable_Pairs(num1,num2)
```

```
220 and 284
1184 and 1210
```

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

In [12]:
```
def oddnumbers(num):
    if num % 2 != 0:
        return num

lst = [2, 3, 5, 7, 8]

oddnumber_list = list(filter(oddnumbers,lst))
print(oddnumber_list)
```

```
[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 [13]:
```
def cube(num):
    return num**3

numbers = [1, 2, 3, 4, 5, 6]
cube_elements = list(map(cube,numbers))

print(cube_elements)
```

```
[1, 8, 27, 64, 125, 216]
```

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 [14]:
```
def even_number(num):
    if num%2 == 0:
        return num

def cube_number(num):
    return num**3

numbers = [1, 2, 3, 4, 5, 6]

cube = list(map(cube_number,(filter(even_number,numbers))))
print(cube)
```

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

In [ ]: