In [16]:
```
# 1. Write a function that inputs a number and prints the multiplicatio
n table of that number

def multab(num):

    """
    This function print multiplication table of given number.
    """

    count = 1

    while (count < 11):                                    #Looping to pr
int multiplication table till 10

        print("{} * {} =" .format(num,count), num * count);
        count += 1


num = int(input("Enter a number "));
multab(num)
```

```
Enter a number 23
23 * 1 = 23
23 * 2 = 46
23 * 3 = 69
23 * 4 = 92
23 * 5 = 115
23 * 6 = 138
23 * 7 = 161
23 * 8 = 184
23 * 9 = 207
23 * 10 = 230
```

In [14]:
```
''' 2. Write a program to print twin primes less than 1000.
If two consecutive odd numbers are both prime then they are known as tw
in primes.'''
```

```
def primecheck(num):

    '''
    This function check if a given number is prime number or not.
    '''
    prime = True;
    for chknum in range (3, num, 2):            # Only odd numbers will
 come as input

        if (num % chknum == 0):

            prime = False;

    if not prime: return 0

    else: return 1


minnum = 3                  # Smallest Twin Prime Number value
maxnum = 1000
count =0
print("Twin Prime Numbers less than {} are :" .format(maxnum))

for num in range (5, maxnum, 2):                    # Looping from 3 to
 1000 for all odd numbers

    if ((primecheck(num) == 1)):

        if (minnum > 0): print(minnum, num); count+=1       # Skip prin
ting if last number was not prime

        minnum = num

    else:

        minnum = 0
```

```
Twin Prime Numbers less than 1000 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
```

In [38]:
```
# 3. Write a program to find out the prime factors of a number. Exampl
e: prime factors of 56 - 2, 2, 2, 7

def prm_fact(num):

    '''
    This function calculate prime factors of given number.
    '''

    for div in range(2, num):

         while (num % div == 0):

                prmfact.append(div)       #Storing and appending values
 to a list
                num = num / div;

    if (num < 1): prmfact.append(num)

    return prmfact

num = int(input("Enter a number: "))
prmfact = []
print("Prime factors of {} are: " .format(num), prm_fact(num))
```

```
Enter a number: 56
Prime factors of 56 are:  [2, 2, 2, 7]
```

In [44]:
```
'''4. Write a program to implement these formulae of permutations and c
ombinations.
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 fact(num):

    '''
    This function calculate factorial of a given number.
```

```
    '''

    return 1 if num == 1 else (num * fact(num-1))

def permcal(obj, rt):

    '''
    This function calculate permutation of a given number.
    '''

    return (fact(obj) / fact(obj - rt))

def combcal(obj,rt):

    '''
    This function calculate combination of a given number.
    '''
    return(permcal(obj,rt) / fact(rt))


obj = int(input("Enter Number of Objects : "));
rt = int(input("Enter Possible orders of objects : "));
print("Permutation of {} objects taken {} at a time is :" .format(obj,r
t), permcal(obj, rt))
print("Combination of {} objects taken {} at a time is :" .format(obj,r
t), combcal(obj, rt))
```

```
Enter Number of Objects : 5
Enter Possible orders of objects : 2
Permutation of 5 objects taken 2 at a time is : 20.0
Combination of 5 objects taken 2 at a time is : 10.0
```

In [32]:
```
# 5. Write a function that converts a decimal number to binary number

def bincal(num):

    '''
    This function converts a given number into binary number.
    '''
```

```
    if num > 1:

        bincal(num // 2)

    print((num % 2), end = '')

num = int(input("Enter a decimal number : "))
print("Binary conversion of decimal number {} is : " .format(num))
bincal(num)
```

```
Enter a decimal number : 34
Binary conversion of decimal number 34 is :
100010
```

In [1]:

```
'''6. Write a function cubesum() that accepts an integer and returns th
e 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.'''

def cubesum(num):

    '''
    This function calculate sum of the cubes of individual digits of gi
ven number.
    '''
    onum = num;
    cubesum = 0 ;
    while (num > 0):

        cnum = num % 10;
        cubesum += cnum ** 3;
        num = num // 10;

    isArmstrong(onum, cubesum)

def isArmstrong(onum, cubesum):
```

```
    '''
    This function verify if a given number is Armstrong number or not.
    '''
    isArmstrong = True if(onum == cubesum) else False

    PrintArmstrong(onum, isArmstrong)

def PrintArmstrong(onum, isArmstrong):

    '''
    This function display the outout.
    '''
    if isArmstrong : print("Number {} is an Armstrong Number".format(on
um))

    else: print("Number {} is not an Armstrong Number".format(onum))


num = int(input("Enter a three digit number: "))
cubesum(num)
```

```
Enter a three digit number: 153
Number 153 is an Armstrong Number
```

In [6]:

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

def prodDigits(num):

    '''
    This function calculates the digit of a given number.
    '''

    onum = num;
    mulnum = 1;
    while (num > 0):

        cnum = num % 10;
        mulnum *= cnum;
```

```
        num = num // 10;

    return mulnum

num = int(input("Enter a number: "))
print("Product of digits of {} is :".format(num), prodDigits(num))
```

```
Enter a number: 345
Product of digits of 345 is : 60
```

In [7]:

```
# 8. Using the function prodDigits() of previous exercise write functio
ns MDR() and MPersistence() that input a number
# and return its multiplicative digital root and multiplicative persist
ence respectively.

def MPersistence(onum, count):

    '''
    This function display the Multiplicative Persistence of a given num
ber.
    '''

    print("Multiplicative Persistence of {0} is : {1}".format(onum, cou
nt))

def MDRCalc(num):

    '''
    This function calculate the Multiplicative digital root of a given
 number.
    '''

    onum = num;
    count = 0;
    while(num > 9):

        mulnum = prodDigits(num);
        num = mulnum;
        count += 1
```

```
    print("Multiplicative Digital Root of {0} is : {1} ".format(onum, n
um))

    MPersistence(onum, count)

num = int(input("Enter a number: "))
MDRCalc(num)
```

```
Enter a number: 86
Multiplicative Digital Root of 86 is : 6
Multiplicative Persistence of 86 is : 3
```

In [8]:

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

def sumPdivisors(num):

    '''
    This function calculate the sum of proper divisors of given number.
    '''
    sum = 0;
    for i in range(1,num):

        if (num % i == 0):

            sum = sum + i;

    return sum

num = int(input("Enter a number : "))
print("Sum of proper divisors of {} is :".format(num), sumPdivisors(num
))
```

```
Enter a number : 36
Sum of proper divisors of 36 is : 55
```

In [9]:
```
''' 10. A number is called perfect if the sum of proper divisors of tha
t 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.'''

def Perfctno(num1, num2):

    '''
    This function calculates perfect number of a given number.
    '''
    pno = ();

    for dnum in range (num1, num2 + 1):

        retsum =sumPdivisors(dnum)

        if (retsum == dnum):

            pno1 = (dnum,)           # Converting integer to tuples to
 store valid value to pno tuple.
            pno = pno + pno1;        # Creating tuple of all perfect nu
mbers.

    return pno

num1 = int(input("Enter Start Range number : "))
num2 = int(input("Enter End Range Number : "))
print("Perfect numbers between {0} and {1} are :".format(num1, num2), P
erfctno(num1, num2))
```

```
Enter Start Range number : 1
Enter End Range Number : 500
Perfect numbers between 1 and 500 are : (6, 28, 496)
```

In [10]:
```
''' 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'''

def amicableNo(num1, num2):

    '''
    This function find amicable numbers between a range of numbers.
    '''

    print("Amicable numbers between {0} and {1} are :".format(num1, num
2))
    pno = ();
    pno1 = ();
    pno2 = ();

    for dnum in range (num1, num2 + 1):

        retsum1 = sumPdivisors(dnum)

        if ((retsum1 > num1) & (retsum1 < num2)):

            retsum2 = sumPdivisors(retsum1)

        if (retsum2 == dnum) & (retsum1 != retsum2) & (retsum1 < retsum
2):

            pno1 = (retsum1, retsum2)

            print(pno1)

num1 = int(input("Enter Start Range number : "))
num2 = int(input("Enter End Range Number : "))
amicableNo(num1, num2)
```

```
Enter Start Range number : 200
Enter End Range Number : 1300
Amicable numbers between 200 and 1300 are :
(220, 284)
(1184, 1210)
```

In [10]:
```
# 12. Write a program which can filter odd numbers in a list by using f
ilter function

def findOddNum(num):

    '''
    This function verify if a given number is odd or not.
    '''

    if (num % 2 == 1):

        return num

num1 = int(input("Enter Start Range number : "))
num2 = int(input("Enter End Range Number : "))

listnum = range(num1, num2)

odd_num = list(filter(findOddNum, listnum))

print("Odd Numbers in given list {} are : ".format(listnum), odd_num)
```

```
Enter Start Range number : 1
Enter End Range Number : 10
Odd Numbers in given list range(1, 10) are :  [1, 3, 5, 7, 9]
```

In [3]:
```
# 13. Write a program which can map() to make a list whose elements are
 cube of elements in a given list.

num1 = int(input("Enter Start Range number : "))
num2 = int(input("Enter End Range Number : "))

ranlist = range(num1, num2)

finalList = list(map(lambda x: x**3 , ranlist))

print("List of cube elements are: ", finalList)
```

```
Enter Start Range number : 1
Enter End Range Number : 10
```

```
List of cube elements are:  [1, 8, 27, 64, 125, 216, 343, 512, 729]
```

In [13]:

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

def evenNum(num):

    '''
    This function verify if a given number is even number or not.
    '''

    if (num % 2 == 0):

        return num

num1 = int(input("Enter Start Range number : "))
num2 = int(input("Enter End Range Number : "))

ranlist = range(num1, num2)

finalList = list(map(lambda x: x**3, (list(filter(evenNum, ranlist)))))

print(finalList)
```

```
Enter Start Range number : 1
Enter End Range Number : 10
[8, 64, 216, 512]
```