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

```
In [1]: def mul_table(num):                    #mul_table fun Created
            for i in range(1,11):
                print( num, 'X' , i ,'=' , num*i)
```

```
In [2]: mul_table(num = int(input()))       # it will prints multiplication table of a num given by user
```

```
55
55 X 1 = 55
55 X 2 = 110
55 X 3 = 165
55 X 4 = 220
55 X 5 = 275
55 X 6 = 330
55 X 7 = 385
55 X 8 = 440
55 X 9 = 495
55 X 10 = 550
```

Question : 2.2. 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 [3]: def Prime(num):                         #prime function takes the num in range, checks the numbs and
        return True for prime num
            for i in range(2, num):
                if num % i == 0:
                    return False
                    break
            else:
                return True


        def generate_twins(first, last):        #This fun will prints twin primes
            for i in range(first, last):
                i = i + 2
                if (Prime(i) and Prime(i)):
                    print(f"({i-2} , {i})",format(i, ""))


        generate_twins(2 ,1000)                  #calling the Function
```

```
(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 [5]: Number = int(input(" Please Enter any Number: "))       #Enter a number/int value by user

        for i in range(2, Number + 1):
            if(Number % i == 0):
                isprime = 1
                for j in range(2, (i //2 + 1)):
                    if(i % j == 0):
                        isprime = 0
                        break

                if (isprime == 1):
                    print(f" %d is prime factors of %d" %(i, Number))   #prints prime factors of a given numb
er
```

```
 Please Enter any Number: 45
 3 is prime factors of 45
 5 is prime factors of 45
```

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! / ((n*(n-r))! = p(n,r) / r!

```
In [5]: import math                                  #importing math function

        n = int(input("The value of n is "))
        r = int(input("The value of r is "))

        p1 = math.factorial(n)/math.factorial(n-r)   # permutation formula
        print("permutation of %d,%d is : %d " %(n,r,p1) )

        c1 =(1/math.factorial(r)) * (math.factorial(n)/math.factorial(n-r))  #combination formula
        print("Combinatation of %d,%d is : %d " %(n,r,c1) )
```

```
The value of n is 9
The value of r is 6
permutation of 9,6 is : 60480
Combinatation of 9,6 is : 84
```

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

```
In [6]: def dec2bin(num):     # crate a function
            if num > 1:
                dec2bin(num // 2)
                print(num % 2 , end ="")     #prints binary num

        num = int(input("Enter a num to convert decimal to binary: "))
        dec2bin(num)     #calling the function
```

```
Enter a num to convert decimal to binary: 55
10111
```

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 [11]:    def cubesum(num):       # create a cubesum func
                sum=0
                while(num!=0):      #while loop
                    i=num%10
                    sum+=(i**3)
                    num//=10
                return sum          #returning the sum value

            i = int(input())
            print (cubesum(i))


            y=cubesum(i)
```

```
407
407
```

```
In [2]: def isArmstrong():
            i = int(input())

            def cubesum(num):   #using cubesum fun in isArmstrong fun (nested functions)
                y=cubesum()
        if y==i:
            print("True" )
        else:
            print("False" )
```

```
True
```

```
In [ ]: def PrintArmstrong(num):     #new function creates
            i=0
            while(num != 0):
                if i == cubesum(i):
                    print(i)
                    i = i+1
                    num = num - 1
                else:
                    i = i+1
                    num = num

        print(PrintArmstrong(999))     #prints armstrong num list in given range
```

```
0
1
153
370
371
407
```

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

```
In [1]: def prodDigits(n):
            prod =1
            while(n != 0):
                prod *= (n%10)
                n = n // 10
            return prod                 #returning the value of products of digits of that num

        n = int(input())
        print (prodDigits(n))          #prints the product of digits of that number
```

```
86
48
```

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 [2]: def MDR(n):                  #create function that  multiplicative digital root of number
            a = prodDigits(n)   # prodDigits(n) can be stored into the value of "a"
            if (a > 9):
                MDR(a)
            else:
                print("MDR =", a)

        num = int(input())
        MDR(num)                    #calling the MDR fun
```

```
86
MDR = 6
```

```
In [3]: def mper(num):       #Create a fun that prints Mpersistence of a num
            count=1
            while(num > 9):
                if prodDigits(num) >9:  #using prodDigits fun in mper fun
                    count += 1
                    num= prodDigits(num)
                else:
                    break
            return count

        n = int(input())
        print("Mper =",mper(num))    #prints mper value
```

```
86
mper = 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 [2]: def sumpdiv(n):
            m=[0]                       #empty list
            for i in range(1, n):
                if(n % i ) == 0):
                    m.append(i)
                    print(i)            #prints proper divisors a num
            return sum(m)


        n = int(input())
        print(sumpdiv(n))
```

```
220
1
2
4
5
10
11
20
22
44
55
110
284
```

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 [11]: lower = int(input("enter a lower number :"))
         upper = int(input("enter a upper number :"))
         for n in range(lower, upper+1):                  # n range between lower, upper values given by user
                 result = 0
                 for i in range(1, n):
                     if(n % i ) == 0):
                         result += i                      #result = result +i , result store the final solution
                 if n == result:                          #the value of n is completely equal to the value of res
         ult
                     print(result)
```

```
enter a lower number :2
enter a upper number :9999
6
28
496
8128
```

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 [1]: def sumpdiv(n):          #the fun is used to find the sum of proper divisors of a num
            m=[0]
            for i in range(1, n):
                if(n % i ) == 0):
                    m.append(i)
            return sum(m)

        def amicable_pair(number):    #it will prints the pairs of amicable num
            result = []               #empty lists
            for x in range(1,number):
                y = sumpdiv(x)        #the value of sumpdiv(x) stores into y
                if sumpdiv(y) == x and x != y:
                    result.append(tuple(sorted((x,y))))   #append will used to add the values into the tuple
            return set(result)


        print(amicable_pair(10000))    #prints pairs of amicable in given range 'n'
```

```
{(1184, 1210), (220, 284), (5020, 5564), (6232, 6368), (2620, 2924)}
```

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

```
In [11]: def is_odd(n):   #fun will help to find odd num in the list
             return n%2!= 0


         x=range(1,100,1)     #range(lower value, upper value, step size )
         odds=list(filter(is_odd,x))  #filter odd value from the list
         print(odds)
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49,
51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97,
99]
```

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

```
In [3]: def update(n):     # n**3 means = value (n)3
            return n**3

        nums = [*range(1,10,1)]
        cubes = list(map(update, nums)) #map fun is to connect with nums list
        print(cubes)
```

```
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, 1331, 1728, 2197, 2744, 3375, 4096, 4913, 5832, 68
59, 8000, 9261, 10648, 12167, 13824, 15625, 17576, 19683, 21952, 24389, 27000, 29791, 32768, 359
37, 39304, 42875, 46656, 50653, 54872, 59319, 64000, 68921, 74088, 79507, 85184, 91125, 97336, 1
03823, 110592, 117649, 125000, 132651, 140608, 148877, 157464, 166375, 175616, 185193, 195112, 2
05379, 216000, 226981, 238328, 250047, 262144, 274625, 287496, 300763, 314432, 328509, 343000, 3
57911, 373248, 389017, 405224, 421875, 438976, 456533, 474552, 493039, 512000, 531441, 551368, 5
71787, 592704, 614125, 636056, 658503, 681472, 704969, 729000, 753571, 778688, 804357, 830584, 8
57375, 884736, 912673, 941192, 970299]
```

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 [8]: def is_even(n):   #fun is used to filter/seperate even value from list
            return (n%2== 0

        def cube(n):      #fun is used to filter/seperate odd value from list
            return n**3

        nums = [*range(1,10,1)]
        even_nums = list(filter(is_even,nums))   #used to filter even num from list and store into even_nums
```