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

In [99]:

```
def multiplication():
    number = int(input("Enter Number"))
    for i in range(1,21):
        print (str(number) + " * " + str(i) + " =   {0}".format(number * i))
multiplication()
```

```
Enter Number10
10 * 1 =   10
10 * 2 =   20
10 * 3 =   30
10 * 4 =   40
10 * 5 =   50
10 * 6 =   60
10 * 7 =   70
10 * 8 =   80
10 * 9 =   90
10 * 10 =   100
10 * 11 =   110
10 * 12 =   120
10 * 13 =   130
10 * 14 =   140
10 * 15 =   150
10 * 16 =   160
10 * 17 =   170
10 * 18 =   180
10 * 19 =   190
10 * 20 =   200
```

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 [100]:

```
num1 = 1000

nums = [ele for ele in range(2,num1) if all(ele%i != 0 for i in range(2,ele))] ## list of prime num
bers upto 1000
                                        ## If all ele%i is false for all elements then only take ti
t element
twinprimes = []                         ## as a prime number
for i in range(0,len(nums)-1):
    if(abs(nums[i] - nums[i+1]) == 2):  ## calculating the absolute values of consequtive prime num
bers
        twinprimes.append((nums[i],nums[i+1])) ## If the difference is 2 then append to the twinpri
mes list
print ("*"*50)
print ("List of twin primes are: ",twinprimes)
print ("*"*50)
print ("Number of twin primes between {0} and {upto} is {count}".format(0,upto = num1,count = len(t
winprimes)))
```

```
**************************************************
List of 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)]
**************************************************
Number of twin primes between 0 and 1000 is 35
```

..

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

In [115]:

```
import math
num1 = int(input("Enter the number to get prime factors"))
primefactors = []
while(num1%2 == 0):              ## continue dividing by 2 and adding the 2 to prime factor list
    primefactors.append(2)
    num1 = num1/2                ## As the number is divisble by 2 divide the num with 2
for i in range(3,int(math.sqrt(num1))): ## Next looping from 3 to sqrt of num because max it can t
o the value of sqrt value
    while(num1%i == 0):                 ## If number is divisible i
        primefactors.append(i)          ## Then add to primefactors list
        num1 = num1/i
    else:
        i = i+2                         ## If the number is not divisible then increase 3 by 5
if (num1>2):                            ## If number is greater than 2
    primefactors.append(int(num1))      ## directly append to prime factors list
print (primefactors)
## Reference from "https://www.geeksforgeeks.org/print-all-prime-factors-of-a-given-number/"
```

```
Enter the number to get prime factors36
[2, 2, 9]
```

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 [102]:

```
def p(n,r):
    if ( n > 0):
        fact = factorial(n)          ##n!                    ## n P r = n!/(n-r)!
        deno = factorial(abs(n-r))   ## (n-r)!
    return fact/deno

def factorial(n):
    return 1 if (n == 0 or n == 1) else n*factorial(n-1)
def c(n,r):
    num = p(n,r)                                  ## n c r = npr/r!
    denom = factorial(r)
    return num/denom
n = int(input("Enter the value of n"))
r = int(input("Enter the value of r"))
print ("No of permutation are {0}".format(p(n,r)))
print ("No of combinations are {0}".format(c(n,r)))
```

```
Enter the value of n6
Enter the value of r3
No of permutation are 120.0
No of combinations are 20.0
```

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

In [116]:

```
def dec2bin(num):
    '''
    This returns a binary number for a decimal number
    '''
    numbers = []                ###To store the binary digits
    while (num > 0):
        numbers.append(num%2)   ###Storing the remainder when num is divided by 2
        num = int(num/2)        ##Dividing the num by 2 and stroing the result as num
```

```
    if (len(numbers)< 4 ):      ## If the length of binary digits is less than 4 it will append a 0
to it
        numbers.append(0)
    return " ".join(str(i) for i in numbers[::-1])## As the join method takes only string convertin
g list of integers to string

number = int(input("Enter the positive decimal Number: "))
if (number < 0):
    print ("oops! please enter the positive number")
else:
    print ("Binary conversion of {0} is {1}".format(number,dec2bin(number)))
```

```
Enter the positive decimal Number: 10
Binary conversion of 10 is 1 0 1 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 [104]:

```
def cubesum(num):
    '''
    This function return the cube sum of the input number
    '''
    ## list(str(num)) --> This will convert integer num to string and string to list
    ##for loop will loop through the above list and cube of element is calculated and stored in tu
ple
    ## sum is a method which will return sum of all elements in a tuple or list
    return sum((int(i)*int(i)*int(i) for i in list(str(num))))

def PrintArmstrong(num):
    '''
    This will print armstrong number
    '''
    armstrong_nums = []
    sum_of_cubes = list(map(cubesum,[i for i in range(number+1)])) ## Getting the cube sum of numbe
rs of a list
    # comparing the cube sum matches with the real number
    #zip command will get 1 element from 1 list and another from 2nd list

    for i ,x in zip(sum_of_cubes,[i for i in range(number+1)]):
        if (int(i) == int(x)):
                armstrong_nums.append(x)
        else:
                pass
    return armstrong_nums
def isArmstrong(num):
    '''
    This checks a number is armstrong number or not
    '''
    cube_sum = cubesum(num)
    return True if (cube_sum == num) else False


number = int(input("Enter the number to find the cube sum of all digits: "))
if (number >= 0):
    print ("Cubesum of the number is {0}: ".format(cubesum(number)))
    print ("Armstrong number upto {0} are {1} ".format(number,PrintArmstrong(number)))
    if (isArmstrong(number)):
        print ("{0} is an Armstrong Number".format(number))
    else:
        print ("{0} is not an Armstrong Number".format(number))
else:
    print ("Oops! please enter the positive number")

##Reference : https://stackoverflow.com/questions/32946714/arithmetic-operations-in-a-list
```

```
Enter the number to find the cube sum of all digits: 156
Cubesum of the number is 342:
Armstrong number upto 156 are [0, 1, 153]
```

```
156 is not an Armstrong Number
```

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

In [105]:

```
def prodDigits(num = None):
    if (num == None):           # This loop can be used if no argument was passed to this function
        num = int(input("Enter a number to find product of all digits"))
    numlist = list(str(num))    # converting non iterable int object to string object so it can  be
converted to list
    prod = 1
    for i in numlist:
        prod *= int(i)          #looping through the string objects and converting them to int
                                #As prod is not possible in string objects
    return prod
print (prodDigits())
```

```
Enter a number to find product of all digits345
60
```

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 [106]:

```
def MDR():
    num = input("Enter the number to find the MDR") #convert input to list to find the length of
input
    digitalroots = []
    while(len(num) > 1):                        # length of num is not greater than 1 then call
prodDigits function

        digitalroots.append(int(num))           # appending the digital root to the list
        num = str(prodDigits(int(num)))         # stores the output of prodDigts in num
                                                # converting output of prodDigits to str or else whi
looping
                                                # as int object does not have len argument
    else:
        digitalroots.append(int(num))           # If length is 1 then num is directly adding to the d
igital roots list
    return digitalroots

def MPersistence(digitalroots):
    m_persistence_count = 0
    m_persistence_count = len(digitalroots) - 1# removing the first input number
    return m_persistence_count
digitalroots = MDR()
print ("The MDR for {number} is {roots} and MPersistence is {persistance}"\
                .format(number = digitalroots[0], roots = digitalroots[1:] , persistance = MPersiste
nce(digitalroots)))
```

```
Enter the number to find the MDR341
The MDR for 341 is [12, 2] and MPersistence is 2
```

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 [107]:

```
def sumPdivisors(num = None):
    try:
        if num == None:
            num = int(input("Enter number to find proper divisors: "))
        divisors = [i for i in range(1,num) if num%i == 0] # If the number is divisble in range of
1 to 35 then add to divisors list
        return (num,divisors,sum(divisors))
    except :
        print ("oops!",sys.exc_info()[0],"occured")

divisors = sumPdivisors()
print ("Sum of divisors for {number} is {sum}".format(number = divisors[0],sum = divisors[2]))
```

```
Enter number to find proper divisors: 52
Sum of divisors for 52 is 46
```

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 [108]:

```
def perfectnumbers(num):
    perfect_numbers = []
    for i in range(1,num+1):
        p_divisors = sumPdivisors(i)
        if (p_divisors[0] == p_divisors[2]):
            perfect_numbers.append(i)
        else:
            pass
    return perfect_numbers
Number = int(input("Enter the range to find perfect numbers"))
print ("The perfect Numbers upto {0} are {1}".format(Number,perfectnumbers(Number)))
```

```
Enter the range to find perfect numbers100
The perfect Numbers upto 100 are [6, 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.

In [109]:

```
def amicable(num):
    divisors = {}
    amicable = []
    for i in range(1,num+1):         ## Calculating the positive divisors sum by using above functic
n
        Pdivisors = sumPdivisors(i)
        divisors[Pdivisors[0]] = Pdivisors[2] ## storing the sum in a dictionary with key as num an
d value as sum
    for key,val in divisors.items():  ## looping through the items of a dictionary
        for i,j in divisors.items():
            if val == i and j == key and key != val: ## If divisor sum is equal to num value and\
                amicable.append(key)                  ## then check the above number sum value is eq
al to key of above sum value
                amicable.append(i)                    ## if key and value are having same number the
it cannot be amicable
    amicable = list(set(amicable))                    ## It will give duplicates of amplicable
numbers ex:(220,284) ,(284,220)
                                                      ## removing the duplicates using set and
converting to list again
    for i in range(-1,len(amicable)-1,2):             ## looping from reverse order from a list
1210,1184,284,220
        print ("{0} and {1} are amicable numbers".format(amicable[i-1],amicable[i]))
```

```
                                                    ## printing amicable in particular order from
lower to higher number
amicable(int(input("Enter the number to find amicable numbers upto that range: ")))
```

```
Enter the number to find amicable numbers upto that range: 2000
220 and 284 are amicable numbers
1184 and 1210 are amicable numbers
```

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

In [110]:

```
def oddnumbers(x):
    if (x%2 != 0):
        return True
odd = list(filter(oddnumbers, [i for i in range(int(input("Enter the number to find odd numbers upt
o that range: "))+1)]))
odd
```

```
Enter the number to find odd numbers upto that range: 10
```

Out[110]:

```
[1, 3, 5, 7, 9]
```

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

In [111]:

```
def cube(num):
    return num*num*num
cubes = list(map(cube,[i for i in range(int(input("Enter the number to find cubes upto that range:
"))+1)]))
cubes
```

```
Enter the number to find cubes upto that range: 10
```

Out[111]:

```
[0, 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 [113]:

```
def even(num):
    return True if num%2 == 0 else False                    ## maps list of 10 numbers to cube
function
even_cubes = list(filter(even,map(cube, [i for i in range(int(input("Enter number to find cubes: ")
)+1)])))
even_cubes                                                  ## filter the cube values which ar
even number
```

```
Enter number to find cubes: 10
```

Out[113]:

```
[0, 8, 64, 216, 512, 1000]
```