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

In [1]:
```
num=int(input("Enter a number"))  #Taking input from user
i=1
while(i<11):                      #function for computing the table
    print(num,' * ',i,' = ',num*i)
    i+=1
```

```
Enter a number2
2  *  1  =  2
2  *  2  =  4
2  *  3  =  6
2  *  4  =  8
2  *  5  =  10
2  *  6  =  12
2  *  7  =  14
2  *  8  =  16
2  *  9  =  18
2  *  10  =  20
```

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]:
```
def istwinprime(a,b):
    i=2
    flag_a=1
    flag_b=1
    while(i<=a/2):
        if(a%i==0):
            flag_a=0
            break;
        i+=1
    i=2
    while(i<=b/2):
        if(b%i==0):
            flag_b=0
            break;
        i+=1
```

```
    if(flag_a==1 and flag_b==1):
        return 1
    else:
        return 0
```

In [3]:
```
twinPrime=[]
for i in range(3,1000,2):
    k=istwinprime(i,i+2)
    if(k==1):
        twinPrime.append([i,i+2])
```

In [4]:
```
twinPrime
```

Out[4]:
```
[[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 [7]:
```
def isPrime(a):
    i=2
    flag_a=1
    while(i<=a/2):
        if(a%i==0):
            flag_a=0
            break;
        i+=1
    if(flag_a==1):
        return 1
    else:
        return 0
```

In [8]:
```
import pdb
primeFactor=[]
num=int(input("Enter a number"))
k=2
while(num!=1):
    if(num%k==0 and isPrime(k)==1):
        primeFactor.append(k)
        num//=k
    else:
        k+=1
print(primeFactor)
```

```
Enter a number18
```

```
[2, 3, 3]
```

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 [7]:
```
def factorial(num):
    return 1 if num == 1 else (num * factorial(num-1))
n=int(input("enter total number of objects"))
r=int(input("enter number of objects taken"))
nPr=factorial(n)/factorial(n-r)
nCr=nPr/factorial(r)
print("No of permutation ",nPr)
print("No of combination ",nCr)
```

```
enter total number of objects8
enter number of objects taken5
No of permutation  6720.0
No of combination  56.0
```

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

In [8]:
```
num=int(input("Enter a number"))
binary=[]
while(num!=1):
    binary.append(num%2)
    num//=2
binary.append(num)
binary.reverse()
print(binary)
```

```
Enter a number18
[1, 0, 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 [24]:
```
import math
```

```
def cubesum(num):
    Sum=0
    while(num>0):
        a=num%10
        t=a*a*a
        Sum=Sum+t
        num=int(math.floor(num/10))
    return Sum
```

In [27]:
```
nu=int(input('Enter a number'))
if(nu==cubesum(nu)):
    print("Its a Armstrong number")
else:
    print("Not a armstrong number")
```

```
Enter a number371
Its a Armstrong number
```

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

In [30]:
```
import math
def prodDigits(num):
    Sum=1
    while(num>0):
        a=num%10;
        Sum=Sum*a
        num=int(math.floor(num/10))
    return Sum
```

```
24
```

8. 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 [56]:
```
k=341
i=0
while(math.floor(k/10)!=0):      #while(k/10!=0):
    k=prodDigits(k)
    t+=1
```

```
    i+=1
print(k,i)
```

```
2 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 [13]:
```
import math
def sumPdivisors(num):
    proper_divisor=[]
    for i in range(1,math.floor(num/2)+1):
        if(num % i==0):
            proper_divisor.append(i)
    print(proper_divisor)
inp=int(input("Enter a number"))
sumPdivisors(inp)
```

```
Enter a number36
[1, 2, 3, 4, 6, 9, 12, 18]
```

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 [2]:
```
import math
def sumPdivisors(num):
    Sum=0
    for i in range(1,math.floor(num/2)+1):
        if(num % i==0):
            Sum=Sum+i
    return Sum
inp=int(input("Enter a number"))
Perfect_numbers=[]
for i in range(1,inp+1):
    if(i==sumPdivisors(i)):
        Perfect_numbers.append(i)
print(Perfect_numbers)
```

```
Enter a number28
[6, 28]
```

11. Write a function to print pairs of amicable numbers in a range

In [4]:
```
inp=int(input("Enter a number"))
Amicable_numbers=[]
for i in range(1, inp+1):
    for j in range(i+1,inp+1):
        if(i==sumPdivisors(j) and j==sumPdivisors(i)):
                Amicable_numbers.append([i,j])
print(Amicable_numbers)
```

```
Enter a number2000
[[220, 284], [1184, 1210]]
```

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

In [11]:
```
def find_odd(num):
    if(num%2!=0):
        return num
number_list = range(11)
odd_num_lst = list(filter(find_odd, number_list))
print(odd_num_lst)
```

```
[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 [12]:
```
numbers = [1, 2, 3, 4]

def powerOfThree(num):
    return num ** 3

#using map() function
cube = list(map(powerOfThree, numbers))
print(cube)
```

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

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]:
```
def find_even(num):
    if(num%2==0):
```

```
        return num
number_list = range(31)
even_num_lst = list(filter(find_even, number_list))
cube = list(map(powerOfThree, even_num_lst))
print(cube)
```

```
[8, 64, 216, 512, 1000, 1728, 2744, 4096, 5832, 8000, 10648, 13824, 175
76, 21952, 27000]
```