In [6]:

```
#1.Table of any number
n=int(input("enter the no."))
for i in range(1,11):
    print(n,'x',i,'=',n*i)
```

```
enter the no.6
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
```

In [7]:

```
#2.Twin Prime numbers
def prime_no(l,u):
    lis=[]
    for i in range(l,u+1):
         if i>1:
                for j in range(2,i):
                    if i%j==0:
                        break
                else:
                    lis.append(i)
    return lis


def odd_no(lis):
    lis2=[]
    for i in lis:
        if i%2==1:
            lis2.append(i)
    return lis2

x=prime_no(1,1000)
y=odd_no(x)
print('The twin prime numbers are:')
for i in range(len(y)-1):
    a=y[i]
    b=y[i+1]
    if b-a==2:
        print(a,b)
```

```
The twin prime numbers 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 [8]:

```
#3.Prime factors of a number
x=int(input("enter the number:"))
l=[]
for i in range(2,x+1,1):
    while x%i==0:
        l.append(i)
        x=x/i;
print(l)
```

```
enter the number:36
[2, 2, 3, 3]
```

In [17]:

```
#4.1 Permutation
def p(n,r):
    def factorial(n):
        fact=1
        if n==0:
            return 1
        for i in range(1,n+1):
            fact=fact*i
        return fact
    x=factorial(n)
    y=factorial(n-r)
    return x/y
```

In [18]:

```
p(10,7)
```

Out[18]:

```
604800.0
```

In [19]:

```
#4.2 Combination
def c(n,r):
    def factorial(n):
        fact=1
        if n==0:
            return 1
        for i in range(1,n+1):
            fact=fact*i
        return fact
    x=factorial(n)
    y=factorial(n-r)
```

```
    z=factorial(r)
    return x/(z*y)
```

In [20]:

```
c(10,7)
```

Out[20]:

120.0

In [21]:

```
#5.decimal to binary
s=int(input("enter the number"))
l=""
w=""
while s>0:
    x=s%2
    l=l+str(x)
    s=int(s/2)
for i in l:
    w=i+w
print(w)
```

```
enter the number08
1000
```

In [22]:

```
#6.1 sums cube of individual digit of numer
def cubesum(n):
    k=str(n)
    x=len(k)
    l=list(k)
    cube=0
    for i in l:
        cube=cube+(int(i)**3)
    return cube
```

In [23]:

```
cubesum(3)
```

Out[23]:

27

In [24]:

```
#6.2 to print armstrong no. in given range
def PrintArmstrong(ll,ul):
    if ll and ul<=1000:
        for i in range(ll,ul+1):
            x=cubesum(i)
            if x==i:
                print(x)
    else:
        print("error:Please enter upper and lower limit within 3 digit range only")
```

In [26]:

```
PrintArmstrong(1,1000)
```

```
1
153
370
371
407
```

In [27]:

```
#6.3 to find if it is an armstrong number
def isArmstrong(n):
    y=cubesum(n)
    if n==y:
        print(n,"is an armstrong number")
    else:
        print(n,"is not an armstrong number")
```

In [28]:

```
isArmstrong(154)
```

```
154 is not an armstrong number
```

In [1]:

```
#7.Product of digits in a string
def prodDigits(n):
    k=str(n)
    l=list(k)
    m=1
    for i in l:
        m=m*int(i)
    return m
```

In [30]:

```
prodDigits(154)
```

Out[30]:

```
20
```

In [18]:

```
#8.MDR
def MDR(n):
    k=len(str(n))
    l=list(str(n))
    if k==1:
        print("MDR:",n)
    else:
        while(k>1):
            mul=prodDigits(n)
            k=len(str(mul))
            n=mul
        print("MDR:",mul)
```

In [19]:

```
MDR(341)
```

```
MDR: 2
```

In [15]:

```
#8.2 MPersistence
def MPersistence(n):
    k=len(str(n))
    count=0
    l=list(str(n))
    if k==1:
```

```
        print("MPersistence",0)
    else:
        while(k>1):
            count+=1
            mul=prodDigits(n)
            k=len(str(mul))
            n=mul
        print("MPersistence:",count)
```

In [16]:

```
MPersistence(342)
```

```
MPersistence: 2
```

In [21]:

```
#9.Sum of proper divisors of number
def sumPdivisors(n):
    sum_p=0
    for i in range(1,n):
        if n%i==0:
             sum_p+=i
    return sum_p
```

In [32]:

```
sumPdivisors(36)
```

Out[32]:

```
55
```

In [22]:

```
#10.to find perfect number i.e sum of all divisor equal to that number

def Perfect_no(ll,ul):
    for i in range(ll,ul+1):
            x=sumPdivisors(i)
            if x==i:
                print(x)
```

In [23]:

```
Perfect_no(1,1000)
```

```
6
28
496
```

In [24]:

```
#11. amicable numbers i.e sum of proper divisor is equal to other number
def amicable(x,y):
    a=sumPdivisors(x)
    b=sumPdivisors(y)
    if a==y and b==x:
        print("They are amicable numbers")
    else:
        print("They are not amicable numbers")
```

In [25]:

```
amicable(220,284)
```

```
amicable(220,284)
```

```
They are amicable numbers
```

In [33]:

```
#12.filter odd number using filter function
n=int(input("enter size of list:"))
l=[]
nl=[]
for i in range(n):
    l.append(int(input()))
def filter_odd(x):
    if x%2==0:
        return x
even_no_list=list(filter(filter_odd,l))
print(even_no_list)
```

```
enter size of list:5
1
2
3
4
5
[2, 4]
```

In [34]:

```
#13.map function to apply cube function
n=int(input("enter size of list:"))
l=[]
for i in range(n):
    l.append(int(input()))
def cube(x):
    return x**3
cubes=list(map(cube,l))
print(cubes)
```

```
enter size of list:5
1
2
3
4
5
[1, 8, 27, 64, 125]
```

In [35]:

```
#14.using map and filter function
n=int(input("enter size of list:"))
l=[]
nl=[]
for i in range(n):
    l.append(int(input()))
def filter_odd(x):
    if x%2==0:
        return x
even_no_list=list(filter(filter_odd,l))
print(even_no_list,"are the even numbers found using filter function")
def cube(x):
    return x**3
cubes=list(map(cube,even_no_list))
print(cubes,"are the cube of numbers using map function")
```

```
enter size of list:5
1
2
3
4
5
[2, 4] are the even numbers found using filter function
```

```
[8, 64] are the cube of numbers using map function
```