In [1]:
```
print("*Eg.1 Multiplication Table*")
num=int(input ("Enter the number whose multiplication table you want::"
))
#print(num)
print("Multiplication Table for {0} is:".format(num))
lst1 = list(range(1,11))
#print(lst1)
lst2=list(map(lambda x: x*num,lst1))
for ele in lst2:
    print(ele)
```

```
*Eg.1 Multiplication Table*
Enter the number whose multiplication table you want::2
Multiplication Table for 2 is:
2
4
6
8
10
12
14
16
18
20
```

In [2]:
```
import math as m
def prime_factor (num):

    while(num % 2== 0):

        print(2)
        num= num/2

    for i in range(3,int(m.sqrt(num))+1,2):

        while(num%i==0):
```

```
            print(i)
            num=num/i

    if num>2:
        print (int(num))
print("*Eg.3 Prime factors*")
num = int(input("Enter a number whose prime factors you want to find:-"
))
prime_factor(num)
```

```
*Eg.3 Prime factors*
Enter a number whose prime factors you want to find:-36
2
2
3
3
```

In [3]:

```
def convert (num):

    lst = [0]*num
    #print(len(lst))
    i= 0

    while(num>0):

        lst[i] = num % 2
        num = int(num/2)
        i +=1

    for j in range(i - 1, -1, -1):
        print(lst[j], end = "");

print("*Eg.5*")
num = int(input("Enter the number which you want to convert into binary
 ::  "))
convert(num)
```

```
*Eg.5*
Enter the number which you want to convert into binary ::  25
```

```
11001
```

In [4]:
```
def cubesum (num):
    lst = list(map(int,str(num)))   #int to list conversion
    cube = list(map(lambda x: x**3,lst))
    #print(cube)
    #print(len(cube))
    sum = 0
    for i in range(len(cube)):
        sum +=cube[i]

    print("The sum of the cubes of individual digits of number is = {}
.".format(sum))
    isArmstrong(num,sum)



def isArmstrong(x,y):
    if x == y:
        PrintArmstrong(x)

    else:
        print("Hence the given number {} is not a ArmStrong Number.".fo
rmat(x))


def PrintArmstrong(j):
    print("Hence the given number {} is a ArmStrong Number.".format(num
))

print("*Eg.6 Armstrong Numbers*")
num = int(input("Enter the number which you want to check whether it is
 Armstrong or not :: "))
cubesum (num)
```

```
*Eg.6 Armstrong Numbers*
Enter the number which you want to check whether it is Armstrong or not
:: 365
The sum of the cubes of individual digits of number is = 368.
```

Hence the given number 365 is not a ArmStrong Number.

In [5]:
```
from functools import reduce

def prodDigits(num):

    lst = [int(x) for x in str(num)]
    #print(lst)
    product = reduce(lambda a,b : a*b ,lst)
    #print(type(product))
    return(product)

print("*Eg.7*")
num = input("Enter the number whose product of digits you want ::")
#print(num)
#print(type(num))
print("The product of digits is ::",prodDigits(num))
```

```
*Eg.7*
Enter the number whose product of digits you want ::28
The product of digits is :: 16
```

In [6]:
```
def MDR(y):
    a = prodDigits(y)
    MPersistence(a)


def MPersistence(num):
    global count  #We need to use global keyword in a function if we wa
nt to do assignments / to change them (variables)
    x = str(num)
    if len(x) == 1:
        print("MDR = {} , MPersistence = {}".format(x,count))

    else:
        count +=1
        MDR(x)
```

```
num = int(input('''Enter the number whose multiplicative digital root a
nd
multiplicative persistence you want to calculate :: '''))

print("Eg.8")
count=0
MPersistence(num)
```

```
Enter the number whose multiplicative digital root and
multiplicative persistence you want to calculate :: 365
Eg.8
MDR = 0 , MPersistence = 2
```

In [7]:
```
def sumPdivisors(num):
    lst = []

    for i in range(1,(num-1)):

        if num % i == 0:
            lst.append(i)
    #print("List of divisors are ::",lst)
    x = sum(lst)

    return x

print("Eg.9")

num = int(input('''Enter the number whose sum of proper divisors you wa
nt to calculate :: '''))
print("Sum = ",sumPdivisors(num))
```

```
Eg.9
Enter the number whose sum of proper divisors you want to calculate ::
225
Sum =  178
```

In [8]:
```
def is_perfect(i):
    sum = sumPdivisors(i)
    #print(sum)
```

```
    if sum == i:
        lst1.append(i)

print("Eg.10")
lst1 = []
num1,num2 = [int(x) for x in input("Enter two value: ").split()]
for i in range(num1,(num2+1)):
    is_perfect(i)
print("Perfect numbers in a given range are ::",lst1)
```

```
Eg.10
Enter two value: 25 30
Perfect numbers in a given range are :: [28]
```

In [10]:
```
lst = []
num1,num2 = [int(x) for x in input("Enter two value: ").split()]

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

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

        sum1 = sumPdivisors(i)
        sum2 = sumPdivisors(j)
        if sum1 == j and sum2 == i:
            lst.append(i + j)

print(lst)
```

```
Enter two value: 201 300
[504, 504]
```

In [18]:
```
def filter(n):

    if n % 2 == 0:
        return False
    else:
        return True
print("Eg.12")
```

```
lst = [int (i) for i in input("Enter the list of elements ::").split()]
lst2 = []
for ele in lst:
    if filter(ele):
        lst2.append(ele)
print(lst2)
```

```
Eg.12
Enter the list of elements ::1 2 3 4 5 6 7 8 9 10
[1, 3, 5, 7, 9]
```

In [20]:
```
print("Eg.13")
def cubes (n):
    return n**3

lst = [int (i) for i in input("Enter the list of elements ::").split()]
new_lst = list(map(cubes,lst))
print(new_lst)
```

```
Enter the list of elements ::1 2 3 4 5 6 7 8 9 10
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

In [1]:
```
print("Eg.14")

def find_even(num):
    if num % 2 == 0:
        return num

def cubes (n):
    return n**3

lst1 = [int (i) for i in input("Enter the list of elements ::").split
()]
even = list(filter(find_even,lst1))
new_lst = list(map(cubes,even))
print(new_lst)
```

```
Eg.14
Enter the list of elements ::1 2 3 4 5 6 7 8 9 10
[8, 64, 216, 512, 1000]
```

[0, 64, 216, 512, 1000]

In [ ]: