```
#Multiplication table upto 10
def mulp_table(n):
  for i in range(1,11):
    print("{0} * {1} = {2}\n".format(n,i,n*i))
num=(int)(input("Enter a number"))
mulp_table(num)    #calling func
```

```
Enter a number68
68 * 1 = 68

68 * 2 = 136

68 * 3 = 204

68 * 4 = 272

68 * 5 = 340

68 * 6 = 408

68 * 7 = 476

68 * 8 = 544

68 * 9 = 612

68 * 10 = 680
```

```
#Print Twin Prime
import math
def isPrime(n):
  isdiv=False
  for i in range(2,math.floor(math.sqrt(n))+1):
    if n%i==0:
      isdiv=True
      break
  if isdiv:
    return False
  else :
    return True
i=3
while i<1000:
  if (isPrime(i) and isPrime(i+2)):
    print("({0},{1})".format(i,i+2))
    i+=2
  else:
    i+=2
```

```
(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)
```

```
#Prime factors of number

n = (int)(input("Enter a number :"))
m=n
i=2
print("Prime factors are :")
while i<=m:
  if n%i==0:
    print("{0}".format(i))
    n=n//i
    i-=1              #so that we keep on trying dividing by same number untill its not divis
  i+=1
```

```
Enter a number :36
Prime factors are :
2
2
3
3
```

```
#Permutaion and Combinations
def fact(n):
   if n<=1: return 1
   else: return n*fact(n-1)
def perm(n,r):
  per=fact(n)/fact(n-r)
  return int(per)
def comb(n,r):
  com=perm(n,r)/fact(r)
  return int(com)

n=int(input("Enter the value of n: "))
r=int(input("Enter the value of r: "))
print("Permutaions are {0} and combinations are {1}".format(perm(n,r),comb(n,r)))
```

```
Enter the value of n: 10
Enter the value of r: 6
Permutaions are 151200 and combinations are 210
```

```
#Decimal to Binary
def convert(n):
  if n>0:
    convert(n//2)         #we use recursion and print remainder after recursion call bcz we n
    print(n%2,end="")
n= int(input("Enter the integer decimal number: "))
convert(n)
```

```
Enter the integer decimal number: 12
1100
```

```
#Cube Sum and Armstrong Number
def cubesum(n):
  sum=0;
  while(n>0):
    sum+=((n%10)**3)
    n=n//10
  print("Sum is",sum)
  return sum

def isArmstrong(n):
  asum=cubesum(n)
  if n==asum:
    return True
```

```
  else:
    return False

def PrintArmstrong(n):
  if isArmstrong(n):
    print(n," is a armstrong no.")
  else:
    print("Not a armstrong no.")


n=int(input("Enter the num :"))
PrintArmstrong(n)
```

```
Enter the num :153
Sum is 153
153  is a armstrong no.
```

```
#Product of Digits
def prodDigits(n):
  prod=1
  while n>0:
    prod*=(n%10)
    n=n//10
  return prod

n = int(input("Enter a number: "))
print("Product of digit is: ",prodDigits(n))
```

```
Enter a number: 96
Product of digit is:  54
```

```
#Multiplicative digital root and multiplicative persistence
def MDR(n):
  x=prodDigits(n)
  count=1
  while x>9:
    x=prodDigits(x)
    count+=1
  print("MDR is {0} and MPersistence is {1}".format(x,count))

def MPersistence(n):
  MDR(n)

n = int(input("Enter the number :"))
MPersistence(n)
```

```
Enter the number :36
MDR is 8 and MPersistence is 2
```

```
#Sum of Proper Divisors
```

```
def sumPdivisors(n):
  sum=0
  for i in range(1,(n//2)+1):
    if n%i==0:
      sum+=i
  return sum

n = int(input("Enter the number: "))
print("Sum of proper divisors of {0} is {1}".format(n,sumPdivisors(n)))
```

```
Enter the number: 85
Sum of proper divisors of 85 is 23
```

```
#Perfect number
def perfectno(x,y):
  for i in range(x,y):
    s= sumPdivisors(i)
    if s==i:
      print(i)

x = int(input("Enter the range's starting number: "))
y = int(input("Enter the range's ending number: "))
print("Perfect no are:")
perfectno(x,y)
```

```
Enter the range's starting number: 1
Enter the range's ending number: 300
Perfect no are:
6
28
```

```
#Amicable number
def AmicableNo(m,n):
  for i in range(m,n):
    s1=sumPdivisors(i)
    if s1<n and s1!=i:            #s1!=i to eliminate same no pairs like 6 and 6
      s2=sumPdivisors(s1)
      if s2==i and i<s1 :         #i<s1 to eliminate reoccuring pair
        print(i,"and",s1)
x = int(input("Enter the range's starting number: "))
y = int(input("Enter the range's ending number: "))
print("Amicable Pairs are:")
AmicableNo(x,y)
```

```
Enter the range's starting number: 1
Enter the range's ending number: 300
Amicable Pairs are:
220 and 284
```

```
#filter odd no
def oddNO(n):
```

```
  if n%2==1:
    return n
lst=[]
n=int(input("Enter no of elements in a list: "))
print("Enter list element , integer only: ")
for i in range(n):
  x=int(input())
  lst.append(x)
odd_lst=list(filter(oddNO,lst))
print("Odd element list is: ",odd_lst)
```

```
Enter no of elements in a list: 5
Enter list element , integer only:
1
2
3
4
5
Odd element list is:  [1, 3, 5]
```

```
#Map cube of elements
def cube(n):
  return n**3
lst=[]
n=int(input("Enter no of elements in a list: "))
print("Enter list element , integer only: ")
for i in range(n):
  x=int(input())
  lst.append(x)

cubeLst=list(map(cube,lst))
print("Cube list is: ",cubeLst)
```

```
Enter no of elements in a list: 5
Enter list element , integer only:
1
2
3
4
5
Cube list is:  [1, 8, 27, 64, 125]
```

```
#Map and Filter
lst=[]
n=int(input("Enter no of elements in a list: "))
print("Enter list element , integer only: ")
for i in range(n):
  x=int(input())
  lst.append(x)
even_cube_lst=list(map(cube,filter(lambda x: x%2==0,lst)))
print("Even element cube list is: ",even_cube_lst)
```

```
Enter no of elements in a list: 5
Enter list element , integer only:
1
2
3
4
5
Even element cube list is:  [8, 64]
```