In [ ]:
```
# Taken Logics help from GeeksforGeeks,
# removed error with the help of StackOverflow,
```

In [ ]:
```
# Input a number and print multiplication of the number
```

In [1]:
```
def Multiplication_Table(num):
    print("Multiplication Table for {}".format(num))
    print("Multiplication Table for {}".format(num))
    for i in range(1,n+1):
        print("{} * {} =".format(num,i),num*i)
```

In [2]:
```
Multiplication_Table(3)
```

```
Enter the range for the multiplication table 10
Multiplication Table for 3
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
```

In [3]:
```
# Program to print twin prime number
# If two consecutive odd numbers are both prime then they are known as twin primes
#GeeksforGeeks
```

In [13]:
```
def isprime(n):
    flag=0
    # Check for prime number
    for i in range(2,n):
        if(n%i==0):
            flag=1
            break
    if flag==1:
        return False
    else:
        return True

# To print Prime Numbers
#Appending the prime numbers in list
prime_numbers=[]
print("Prime Numbers are:")
n=int(input("enter the range to print prime numbers"))
for i in range(2,n):
    if isprime(i):
        prime_numbers.append(i)
print(prime_numbers)
#To print twin prime numbers
print("\nAll Twin prime between range 1 to {}".format(n))
for i in range(len(prime_numbers)-1):
    if (prime_numbers[i+1]-prime_numbers[i]==2):
        print(prime_numbers[i],prime_numbers[i+1])
```

```
Prime Numbers are:
enter the range to print prime numbers10
[2, 3, 5, 7]


All Twin prime between range 1 to 10
3 5
5 7
```

In [5]:
```
#Print the prime factor of a number
#GeeksforGeeks
```

In [2]:
```
def prime_factor(n):
    pf=[]
    while n%2==0:
        pf.append(2)
        n=n/2
    for i in range(3,int(n),2):
        while n%i==0:
            pf.append(i)
            n=n/i
    if(n>1):
        pf.append(int(n))
    print(pf)

n=(int(input("Enter the number to print prime factor\n")))
prime_factor(n)
```

```
Enter the number to print prime factor
56
[2, 2, 2, 7]
```

In [ ]:
```
# To implement these formulae of permutations and combinations.
#GeeksforGeeks
```

In [3]:
```
def factorial(z):
    fact=1
    for i in range(1,z+1):
        fact=fact*i
    return fact
# Number of permutations of n objects taken r at a time: p(n, r) = n! / (n-r)!.
def permutation(n,r):
    perfactorial(n)/factorial(n-r)
    return per
# Numberofcombinations of n objects taken r at a time is: c(n, r) = n! / (r!*(n-r)!) = p (n,r) / r!
def combination(n,r):
    com=permutation(n,r)/factorial(r)
    return com
print("Permutaion and Combination\n")
print("Enter the value of n\n",(n))
print("Enter the value of r\n",(r))
print("The Permutaion of {} and {} is".format(n,r))
print(permutation(n,r))
print("The Combination of {} and {} is".format(n,r))
print(combination(n,r))
```

```
Permutaion and Combination

enter the value of n
10
enter the value of r
6
The Permutaion of 10 and 5 is
30240.0
The Combination of 10 and 5 is
252.0
```

In [ ]:
```
# Function that converts a decimal number to binary number
```

In [4]:
```
def convert_dec_to_Binary(n):
    b=[]
    while(n>=1):
        r=n%2
        n=n//2
        b.append(int(r))
    n=-1
    b.reverse()
    print("After Conversion from Decimal to Binary")
    for i in b:
        print(i,end='')

n=(int(input("Enter the number\n")))
convert_dec_to_Binary(n)
```

```
Enter the number
10
After Conversion from Decimal to Binary
1010
```

In [ ]:
```
# 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 [8]:
```
import math
def cubesum(temp):
    s=0
    while temp!=0:
        rem=temp%10
        s +=int(math.pow(rem,3))
        temp=int(temp/10)

    return s


def isArmstrong():
    flag=0
    num=(int(input("Enter the Number to find Armstrong number\n")))
    if(num==cubesum(num)):
        flag=1
    else:
        flag=0
    return flag

def PrintArmstrong():
    if(isArmstrong()==1):
        print("It is an Armstrong Number")
    else:
        print("It is not an Armstrong Number")

PrintArmstrong()
```

```
Enter the Number to find Armstrong number
153
It is an Armstrong Number
```

In [ ]:
```
# Function prodDigits() that inputs a number and returns the product of digits of that numbe
r
```

In [9]:
```
def prodDigits(temp):
    s=0
    if temp==0:
        return 0
    while temp!=1:
        rem=temp%10
        s *=rem
        temp=int(temp/10)

    return s

num=int(input("Enter the Number \n"))
print("The product of digits of {} is".format(num),prodDigits(num))
```

```
Enter the Number
153
The product of digits of 153 is 15
```

In [ ]:
```
# 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
#Hackerrnka
```

In [10]:
```
def prodDigits(temp):
    s=0
    if temp==0:
        return 0
    while temp!=1:
        rem=temp%10
        s *=rem
        temp=int(temp/10)

    return s

def MDR():
    num=int(input("Enter the Number to find perfect Numbers \n"))
    temp1=num
    c=0
    while temp1!=0:
        mdr=prodDigits(temp1)
        if mdr<10:
            print("Multiplication Digit Root of {} is".format(num),mdr)
            break

        temp1=mdr

def MPersistence():
    num=int(input("Enter the Number to find Multiplicative Persistance\n"))
    temp1=[]
    while num not in temp1:
        temp1.append(num)
        num=prodDigits(num)
    print("Multiplicative Persistence of {} is".format(temp1[0]),len(temp1)-1)

MDR()
print("")
MPersistence()
```

```
Enter the Number to find Multiplication Digit Root
86
Multiplication Digit Root of 86 is 6

Enter the Number to find Multiplicative Persistance
86
Multiplicative Persistence of 86 is 3
```

In [ ]:
```
# 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, 9,12,18
```

In [3]:
```
def sumPdivisors(num):
    s=0
    for i in range(1,num):
        if num%i==0:
            s+=i
    return s


num=int(input("Enter The Number \n"))

print("The proper Divisors of {} is".format(num),sumPdivisors(num))
```

```
Enter The Number
10
The proper Divisors of 10 is 8
```

In [ ]:
```
# A number is called perfect if the sum of proper divisors of that number is equal to the nu
mber.
# 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 [133]:
```
def perfectNum():
    n=int(input("Enter The Range to find perfect Numbers \n"))
    print("The perfect numbers between 1 to {} is".format(n))
    for i in range(1,n+1):
        if(i ==sumPdivisors(i)):
            print(i)


perfectNum()
```

```
Enter The Range to find perfect Numbers
50
The perfect numbers between 1 to 50 is
6
28
```

In [ ]:
```
# Two different numbers are called amicable numbers if the sum of the proper divisors of eac
h is equal to the other number.
# For example 220 and 284 are amicable numbers.
# Proper divisors of 220 are 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 and 110 so 1+2+4+5+10+11+20+22+44+55+110 = 284
# Sum of proper divisors of 284 = 1+2+4+71+142 = 220
# Write a function to print pairs of amicable numbers in a range
# GeeksforGeeks
```

In [4]:
```
def Amicable():
    n=int(input("Enter The Range to find Amicable Numbers.The first Amicable number pair is
(220,284).So plese enter the range starting from 300 \n"))
    print("The Amicable Numbers between 1 to {} is".format(n))
    for i in range(1,n+1):
        for j in range(i+1,n+1):
            if(sumPdivisors(i)==j and sumPdivisors(j)==i):
                print(i,j)
Amicable()
```

```
Enter The Range to find Amicable Numbers.The first Amicable number pair is (220,284).So plese
enter the range starting from 300
300
The Amicable Numbers between 1 to 300 is
220 284
```

In [26]:
```
# Write a program which can filter odd numbers in a list by using filter function
# map() and filter() syntax and calling from AppliedAICourse Tutorial
```

In [5]:
```
def find_odd_number(num):
    if num%2!=0:
        return num

lst=[]
n=(int(input("Enter the range of list\n")))
for i in range(1,n+1):
    lst.append(i)
print("The elements in list are\n",lst)
odd_num=list(filter(find_odd_number,lst))
print("The odd Number in List are\n",odd_num)
```

```
Enter the range of list
10
The elements in list are
 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
The odd Number in List are
 [1, 3, 5, 7, 9]
```

In [ ]:
```
# Write a program which can map() to make a list whose elements are cube of elements in a gi
ven list
# map() and filter() syntax and calling from AppliedAICourse Tutorial
```

In [6]:
```
#Given List
lst=[]
n=int(input("Enter the range of list\n"))
for i in range(1,n+1):
    lst.append(i)
print("The elements in list are\n",lst)

def Cube_of_Ele(num):
    return num**3

Cube = list(map(Cube_of_Ele,lst))
print('The cube of elements in a list are\n',Cube)
```

```
Enter the range of list
10
The elements in list are
 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
The cube of elements in a list are
 [1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

In [ ]:
```
# Write a program which can map() and filter() to make a list whose elements are cube of eve
n number in a given list
```

In [7]:
```
#Given List
lst=[]
n=int(input("Enter the range of list\n"))
for i in range(1,n+1):
    lst.append(i)
print("The elements in list are\n",lst)

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

def cube(num):
    return num**3

even_number=list(filter(even_num,lst))
cube_of_even_number=list(map(cube,even_number))
print("The cube of even number in a list are\n",cube_of_even_number)
```

```
Enter the range of list
10
The elements in list are
 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
The cube of even number in a list are
 [8, 64, 216, 512, 1000]
```

In [ ]:
```

```