In [16]:
```
# multiplication table

num=int(input("Enter any number:"))
print("Multiplication Table of",num)
def multable(num):
    for i in range(1,11):
        print("{0} X {1} = {2}".format(num,i,num*i))
multable(num)
```

```
Enter any number:5
Multiplication Table of 5
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
5 X 10 = 50
```

In [24]:
```
#Twin Prime Numbers

print("Twin Prime Numbers less than 1000 are:")
p=[]
for i in range(1,1000):
    if i>1:
        for j in range(2,i):
            if(i%j==0):
                break
        else:
            p.append(i)
for i in range(0,len(p)-1):
    if(p[i+1]-p[i]<=2):
        print(p[i],p[i+1])
```

```
Twin Prime Numbers less than 1000 are:
2 3
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 [5]:
```
#Prime Factors of Number

num=int(input("Enter any number:"))
F=[]
P=[]
PF=[]
for i in range(1,num+1):
    if(num%i==0):
        F.append(i)
for i in range(0,len(F)):
    if(F[i]>1):
        for j in range(2,F[i]):
            if(F[i]%j==0):
                break
        else:
            P.append(F[i])
#print(F)
#print(P)
for k in range(0,len(P)):
    while(num%P[k]==0):
        PF.append(P[k])
        num=num/P[k]
print("Prime factors of number are:",PF)
```

```
Enter any number:56
Prime factors of number are: [2, 2, 2, 7]
```

In [24]:
```
#Permutation Combination
num1=int(input("Enter any positive number:"))
num2=int(input("ENter any positive number less than num1 : "))
def factorial(n):
    fact=1
    for i in range(1,n+1):
        fact=fact*i
    return fact
def permutation(num1,num2):

    per=int(factorial(num1)/factorial(num1-num2))
    print("Permutation of",num1 ,"and", num2,"is",per)
permutation(num1,num2)

def combination(num1,num2):

    comb=int(factorial(num1)/factorial(num2)*factorial(num1-num2))

    print("Combination of",num1 ,"and", num2,"is",comb)
combination(num1,num2)
```

```
Enter any positive number:9
ENter any positive number less than num1 : 7
Permutation of 9 and 7 is 181440
Combination of 9 and 7 is 144
```

In [4]:
```
#Decimal TO binary

num=int(input("Enter any number : "))
def decimalToBinary(num):
    if(num>1):
        decimalToBinary(num//2)
    print(num%2,end=' ')
decimalToBinary(num)
```

```
Enter any number : 56
1 1 1 0 0 0
```

In [26]:
```
#ArmStrong Number

num=int(input("Enter any number:"))
def cubsum(num):
    sum=0
    while (num!=0):

        sum=sum+(num%10)**3
        num=num//10

    return sum

cubsum(num)
def isArmStrong(num):
    if num==cubsum(num):
        print(num,"is ArmStrong Number")
    else:
        print(num,"is not ArmStrong number")
isArmStrong(num)
def PrintArmstrong(a,b):
    for i in range(a,b+1):
        if i==cubsum(i):
            print(i)

a=int(input("Enter lower range number : "))
b=int(input("Enter upper range number : "))
print("ArmStrong numbers between",a,"and",b,"are:")
PrintArmstrong(a,b)
```

```
Enter any number:153
153 is ArmStrong Number
Enter lower range number : 100
Enter upper range number : 1000
ArmStrong numbers between 100 and 1000 are:
153
370
371
407
```

In [4]:
```
#Product of digits of number

num=int(input("Enter any number :"))

def proDigits(num):
    prod=1
    while (num!=0):
        prod=prod*(num%10)
        num=num//10
    return prod

proDigits(num)
print("Product of digits for given number",num,"is",proDigits(num))
```

```
Enter any number :56
Product of digits for given number 56 is 30
```

In [27]:
```
# MDR() and MPersistence()

a=int(input("Enter any number :"))
def proDigits(a):
    prod=1
    numdigit=[]
    while (a>=1):
        r=a%10
        numdigit.append(r)
        a=a//10
    for i in numdigit:
        prod=prod*i
    return prod

def MDR(a):
    count=0
    while a>10:
        a=proDigits(a)
        count =count+1
    return (a,MPersistence(count))
print("MDR of given number is ",MDR(a))

def MPersistence(count):
    return count
```

```
Enter any number :98
MDR of given number is  (4, 3)
```

In [28]:
```
#Sum of Proper Divisors
num=int(input("Enter any number :"))
def  sumPdivisorss(num):
    s=0
    for i in range(1,num):
        if(num%i==0):
            s=s+i
    print("sum of all divisors ",num,"is :",s)

sumPdivisorss(num)
```

```
Enter any number :36
sum of all divisors  36 is : 55
```

In [29]:
```
#Perfect Number

print("Perfect numbers between 1 to 1000 are:")
for i in range(1,1001):
    sum=0
    for j in range(1,i):
            if(i%j==0):
                sum=sum+j
    if(sum==i):
        print(i)
```

```
Perfect numbers between 1 to 1000 are:
6
28
496
```

In [45]:
```
#Amicable Number

def amicable(a,b):
    for i in range(a,b):
            sum=0
            sum1=0
            for j in range(1,i):
                if(i%j==0):
                    sum=sum+j
            if(sum!=i):
                for k in range(1,sum):
                    if(sum%k==0):
                        sum1=sum1+k
                if(i==sum1):
                    print(i,"and",sum,"are amicable numbers")
a=int(input("Enter lower range number :"))
b=int(input("Enter upper range number :"))
amicable(a,b)
```

```
Enter lower range number :1
Enter upper range number :1000
220 and 284 are amicable numbers
284 and 220 are amicable numbers
```

In [8]:
```
# odd numbers in a list by using filter function

lst = [1, 29,36,49,52,11,67,89,34,56,78,23,14,51,98,71,37]
odd_list=list(filter(lambda X :(X%2!=0),lst))
print(odd_list)
```

```
[1, 29, 49, 11, 67, 89, 23, 51, 71, 37]
```

In [14]:
```
#Map() to map cube of elements in a list

lst = [1, 2, 3, 4, 5,6,7,8,9]
cube_lst=list(map(lambda X : X**3,lst))
print(cube_lst)
```

```
[1, 8, 27, 64, 125, 216, 343, 512, 729]
```

In [22]:
```
#Map() and Filter()

lst = [1, 2, 3, 4, 5,6,7,8,9,10]
new_lst=list(filter(lambda X :(X%2==0),lst))
cube_lst=list(map(lambda X:X**3,new_lst))
print(new_lst)
print(cube_lst)
```

```
[2, 4, 6, 8, 10]
[8, 64, 216, 512, 1000]
```

In [ ]:

In [1]:
```
!jupyter nbconvert --to html PythonOptional_Assignment.ipynb
```

```
[NbConvertApp] Converting notebook PythonOptional_Assignment.ipynb to html
[NbConvertApp] Writing 287935 bytes to PythonOptional_Assignment.html
```

In [ ]: