Assignment - 1 (Python)

1) Write a function that inputs a number and prints the multiplication table of that number.

In [25]:
```
def mul(n):
    """
    This function will give the multiplication table of the given numbe
r.
    """
    for i in range(1,11):
        r=i*n
        print("{0} * {1} = {2}".format(i,n,r))

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

```
Enter the number:4
1 * 4 = 4
2 * 4 = 8
3 * 4 = 12
4 * 4 = 16
5 * 4 = 20
6 * 4 = 24
7 * 4 = 28
8 * 4 = 32
9 * 4 = 36
10 * 4 = 40
```

2) Find the twin prime numbers less than 1000

In [1]:
```
odd  = []
prime = []
t_prime = []
for i in range(1,1001):
    if not int (i%2==0):
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
            odd.append(i)
for j in odd:
    if not (j==1):
        for k in range(2,j):
            if not (j == k):
                if (j%k) == 0:
                    break
        else:
            prime.append(j)
            indx = odd.index(j)
            indx1 = prime.index(j)
            val = odd[indx-1]
            val1 = prime[indx1-1]
            if (val == val1):
                t_prime.append(tuple((val1,j)))
print(t_prime)
```

```
[(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)]
```

3) Write a program to find out the prime factors of a number.

In [18]:
```
import pdb

a = []
n = int(input("Enter the number"))
for i in range(2,n+1):
    for i in range(2,n+1):
        if (n%i==0):
            n /= i
            n = int(n)
            #print("i={}".format(i))
            #print("n={}".format(n))
            a.append(i)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
            break
#           pdb.set_trace()
print(a)
```

```
Enter the number100
[2, 2, 5, 5]
```

1. Write a program to implement these formulae of permutations and combinations. Number of permutations of n objects taken r at a time: p(n, r) = n! / (n-r)!. Number of combinations of n objects taken r at a time is: c(n, r) = n! / (r!*(n-r)!) = p(n,r) / r!

In [72]:
```
def fact(m):
    """
    This is a recursive function to find the factorial of the given num
ber.
    """
    return 1 if m == 1 else (m * fact(m-1))

n = int(input("Enter the total number of elements (n): "))
r = int(input("Enter the number of elements taken at a time (r): "))
p = fact(n)/fact(n-r)
c = p/fact(r)
print("The Permutations and combinations of the given values (without r
eplacement) are: {0} and {1}".format(int(p),int(c)))
```

```
Enter the total number of elements (n): 5
Enter the number of elements taken at a time (r): 2
The Permutations and combinations of the given values (without replacem
ent) are: 20 and 10
```

5) Write a function that converts a decimal number to binary number.

In [35]:
```
import pdb

def binary_conversion(n):
    """
    This function will convert the given number to its binary equivalen
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
t.
    """
    print("In the function and the decimal number is: {}".format(n))
    while(n != 0):
        r = n % 2
#         print(r)
        n = int(n/2)
#         print(n)
        b.append(r)
    while (len(b)<4):
        b.append(0)
    b.reverse()
    return b

a=[]
b=[]
d=[]
no=int(input("Enter the decimal number: "))
a = binary_conversion(no)
d=''.join(str(ab) for ab in a)
print(d)
```

```
Enter the decimal number: 22
In the function and the decimal number is: 22
10110
```

1. 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 [74]:
```
def cubesum(n):
    """
    This function will give the sum of cube values of the given numbe
r's digits.
    """
    a = str(n)
    l=len(a)
    aa = list(a)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    cs =0
    for i in range(l):
        dd = aa[i]
        cs = cs + (int(dd) ** 3)
#    print("n = {}".format(n))
    print("Product of digits = {}".format(cs))
    isArmstrong(cs)

def isArmstrong(cs):
    """
    This function will check if the given number is an Armstrong numbe
r.
    """
    if n==cs:
        printArmstrong(n)
    else:
        print("The number {} is NOT an Armstrong number.".format(n))

def printArmstrong(n):
    """
    This function will print the given number (provided only if it is a
n Armstrong number since, only then this function
    will be invoked).
    """
    print("The number {} is an Armstrong number.".format(n))

n=int(input('Enter the number: '))
cubesum(n)
```

```
Enter the number: 153
Product of digits = 153
The number 153 is an Armstrong number.
```

7) Write a function prodDigits() that inputs a number and returns the product of digits of that number.

In [8]:
```
def prod_of_digits(n):
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    """
    This function will give the product of the digits of the given numb
er.
    """
    a = str(n)
    l=len(a)
    aa = list(a)
    cs = 1
    for i in range(l):
        dd = aa[i]
        cs *=  (int(dd))
    print("Product of digits = {}".format(cs))

n=int(input('Enter the number: '))
prod_of_digits(n)
```

```
Enter the number: 25
Product of digits = 10
```

8) 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

In [40]:

```
def prod_of_digits(n):
    """
    This function will give the multiplicative digitak root and the mul
tiplicative persistence of the given number.
    """
    a = str(n)
    l=len(a)
    aa = list(a)
    c = 0
    while (l > 1):
        if not (c==0):
            aa = list(str(cs))
#             print("aa={}".format(aa))
        cs = 1
        for i in range(l):
            dd = aa[i]
#             print("dd={}".format(dd))
            cs *=  (int(dd))
#             print("cs={}".format(cs))
        l = len(str((cs)))
#         print("l={}".format(l))
        c += 1
    print("The multiplicative digital root = {}".format(cs))
    print("The multiplicative persistence = {}".format(c))

n=int(input('Enter the number: '))
prod_of_digits(n)
```

```
Enter the number: 362
The multiplicative digital root = 8
The multiplicative persistence = 3
```

9) Write a 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, 18

In [26]:
```
a = []

def sumPdivisors(n):
    """
    This function will give the sum of proper divisors of the given num
ber.
    """
    s = 0
    for i in range(1,n):
        if (n % i ==0):
            a.append(i)
            s += i
    print("The sum of proper divisors of the number is {0} and the prop
er divisors are {1}".format(s,a))

n = int(input("Enter the Number: "))
sumPdivisors(n)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Enter the Number: 2924
The sum of proper divisors of the number is 2620 and the proper divisor
s are [1, 2, 4, 17, 34, 43, 68, 86, 172, 731, 1462]
```

10) A number is called perfect if the sum of proper divisors of that number is equal to the number. 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 [4]:
```
import pdb
a = []

def sumPdivisors(m,n):
    """
    This function will give the perfect numbers in a given range (This
 function requires 2 input numbers).
    """
    for i in range(int(m),int(n)):
        s = 0
        for j in range(1, i):
#             pdb.set_trace()
            if (i % j == 0):
                s += j
#                 print("i={0}, j={1}, s={2}".format(i,j,s))
        if (s == i):
#             print("Inside the condition for appending - i={0}, j={1},
 s={2}".format(i,j,s))
            a.append(i)
#            pdb.set_trace()
    print(a)


m,n = input("Enter two numbers for the range: ").split()
sumPdivisors(m,n)
```

```
Enter two numbers for the range: 1 30
[6, 28]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

1. Two different numbers are called amicable numbers if the sum of the proper divisors of each is equal to the other number. For example 220 and 284 are amicable numbers. Sum of proper divisors of 220 = 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

In [37]:
```
import pdb
a = []
b = []
def sumPdivisors(m,n):
    """
    This function will give the amicable numbers within a given range.
    """
    for i in range(int(m),int(n)):
        s = 0
        for j in range(1, i):
            if (i % j == 0):
                s += j
#         pdb.set_trace()
        for k in range(int(m),int(n)):
            t = 0
            if (k == s):
                for l in range(1, k):
                    if (k % l == 0):
                        t += l
                if (i == t):
#                   pdb.set_trace()
                    if not (i == k):
                        a.append(list((i,k)))
#    print(a)
    p = len(a)
    for o in range(0,p,2):
        b.append(a[o])
    print(b)

m,n = input("Enter two numbers for the range: ").split()
sumPdivisors(m,n)
```

```
Enter two numbers for the range: 1 10000
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
[[220, 284], [1184, 1210], [2620, 2924], [5020, 5564], [6232, 6368]]
```

1. Write a program which can filter odd numbers in a list by using filter function

In [62]:
```
l = [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27
, 28, 29, 30]
m = list(filter(lambda a: (a % 2 != 0), l))
print("These are the odd numbers in the list: ")
for a in range(len(m)):
    print(m[a], end=" ")
```

```
These are the odd numbers in the list:
11 13 15 17 19 21 23 25 27 29
```

1. Write a program which can map() to make a list whose elements are cube of elements in a given list

In [61]:
```
l = [1, 2, 3, 4, 5]
m = list(map(lambda a: (a**3), l))
print("These are the cube values of the elements in the list: ")
for a in range(len(m)):
    print(m[a], end=" ")
```

```
These are the cube values of the elements in the list:
1 8 27 64 125
```

1. Write a program which can map() and filter() to make a list whose elements are cube of even number in a given list

In [68]:
```
l = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
m = list(filter(lambda a: (a % 2 == 0), l))
print("These are the even numbers in the list: ")
for a in range(len(m)):
    print(m[a], end=" ")
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
n = list(map(lambda a: (a**3), m))
print("\nThese are the cube values of the elements in the list of even
 numbers: ")
for a in range(len(n)):
    print(n[a], end=" ")
```

```
These are the even numbers in the list:
2 4 6 8 10
These are the cube values of the elements in the list of even numbers:
8 64 216 512 1000
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD