karthik.krishnaswamy17@gmail.com_op1

April 21, 2019

In [ ]:
```
#1) Write a function that inputs a number and prints the multiplication table of that
```

In [1]:
```
def mult(n):
    """
        Function that prints the multiplication table of that number.
    """
    for i in range (1,11):
        print("{0} * {1} ={2}".format(n,i,n*i))

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

```
Enter the number:5
5 * 1 =5
5 * 2 =10
5 * 3 =15
5 * 4 =20
5 * 5 =25
5 * 6 =30
5 * 7 =35
5 * 8 =40
5 * 9 =45
5 * 10 =50
```

In [2]:
```
#2)Write a program to print twin primes less than 1000.
#If two consecutive odd numbers are both prime then they are known as twin primes.
```

In [3]:
```
import numpy as np
def prime(n):
    """
    Find Prime CF of two numbers using (sieve of eratosthenes Alogrithm)
    """
    p=np.arange(n)
    p[0]=0
    p[1]=0
    for i in range (2,n):
        p[i]=1
```

```
            for i in range (2,n):
                if p[i]==1:
                    j=2
                while(i*j<n):
                    p[i*j]=0
                    j+=1

            print("Twin prime numbers till {}".format(n))
            for i in range (n-1):
                if p[i]==1 and p[i+2]==1:
                    print(i,i+2)

        prime(1000)
```

```
Twin prime numbers till 1000
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 [ ]: #3) Write a program to find out the prime factors of a number. Example: prime factors
        #2, 2, 2,7)
```

```
In [7]: import math
        def primefactors(n):
            c=0
            for i in range (2,int(math.sqrt(n))+1):
                    while (n%i==0):
                        n=n/i
                        c+=1
                        print(i)
            if c==0 :
                print("{} is a prime number.No prime factors for prime number.".format(n))

        i=int(input("Enter any number to find prime factors:"))
        primefactors(i)
```

```
Enter any number to find prime factors:15
3
```

```
In [8]: #4) 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
        #combinations of n objects taken r at a time is: c(n, r) = n! / (r!*(n-r)!) = p(n,r) /
```

```
In [9]: def fact(num):
            """
            Old School method to find factorial of a number without using recursive.
            """
            if num==0 or num==1:
                return 1
            else:
                p=1
                for i in range (2,num+1):
                    p*=i
                return p

        def permutation(n,r):
            result=fact(n)/fact(n-r)
            print("Permutation of p({0},{1})={2}".format(n,r,result))

        def combination(n,r):
            result=fact(n)/(fact(r)*fact(n-r))
```

```
            print("Combination of p({0},{1})={2}".format(n,r,result))

        n=int(input("Enter number of Objects:"))
        r=int(input("Enter the time of objects taken at time: "))
        combination(n,r)
        permutation(n,r)
```

```
Enter number of Objects:6
Enter the time of objects taken at time: 3
Combination of p(6,3)=20.0
Permutation of p(6,3)=120.0
```

```
In [ ]: #5)Write a function that converts a decimal number to binary number
```

```
In [12]: import numpy as np
         def dec2bin(n):
             bin=[]
             while (n>0):
                 r=int(n%2)
                 n=int(n/2)
                 bin.append(r)
             bin.reverse()

             # Adding the numbers by using power of 10's using one's twos'....
             res = sum(d * 10**i for i, d in enumerate(bin[::-1]))
             return res
         n=int(input("Enter number in decimal format:"))
         print("{} in binary:{}".format(n,dec2bin(n)))
```

```
Enter number in decimal format:112
112 in binary:1110000
```

```
In [ ]: #6)Write a function cubesum() that accepts an integer and returns the sum of the cubes
        #individual digits of that number. Use this function to make functions PrintArmstrong(
        #isArmstrong() to print Armstrong numbers and to find whether is an Armstrong number.
```

```
In [13]: def isArmstrong(s,n):
             if s==n:
                 print("{} is Armstrong number".format(n))
             else:
                 print("{} is not a Armstrong number".format(n))

         def cubesum(n):
             l=[int(x) for x in str(n)]
             s=0
             for ele in l:
                 s+=ele**3
```

```
            isArmstrong(s,n)

        def PrintArmstrong(n):
            cubesum(n)

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

```
Enter the number:407
407 is Armstrong number
```

```
In [ ]: #Write a function prodDigits() that inputs a number and returns the product of digits
```

```
In [14]: def prodDigits(n):
             l=[int(x) for x in str(n)]
             p=1
             for ele in l:
                 p*=ele
             return p

         if __name__ == '__main__':
             n=int(input("Enter the number to find products of digits:"))
             print("{} is the product of each digits in {}".format(prodDigits(n),n))
```

```
Enter the number to find products of digits:143
12 is the product of each digits in 143
```

```
In [ ]: #8 Find digital Multiplicative digital root of n.
```

```
In [29]: #import prodDigits
         n=int(input("Enter the number:"))
         c=0
         if n<9:
             print("MDR:{} MPersistence:{}".format(n,c))
         else:
             while(n>9):
                 n=prodDigits(n)
                 if n>=0:
                     c+=1
             print("MDR:{} MPersistence:{}".format(n,c))
```

```
Enter the number:341
MDR:2 MPersistence:2
```

```
In [ ]: #9 Write a function sumPdivisors() that finds the sum of proper divisors of a number.
```

```
In [30]: def sumPdivisors(n):
             l=[]
             s=0
             if n==1:
                 return [[1],1]
             else:
                 for i in range (1,int(n/2)+1):
                     if (n%i==0):
                         l.append(i)
                         s+=i
                 return [l,s]

         if __name__ == "__main__":
             n=int(input("Enter the number:"))
             o=sumPdivisors(n)
             print("Divisor of {} :{}\nSum of these Divisor:{}".format(n,o[0],o[1]))
```

```
Enter the number:18
Divisor of 18 :[1, 2, 3, 6, 9]
Sum of these Divisor:21
```

```
In [ ]: #10 Write a program to print all the perfect numbers in a given range
```

```
In [35]: #import sumPdivisors

         def isPerfect(n):
             r=sumPdivisors(n)
             if r[1]==n :
                 print("Yeah! {} is  a perfect number".format(n))
             else:
                 print("Sorry! {} is  not a perfect number".format(n))

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

```
Enter the number:8128
Yeah! 8128 is  a perfect number
```

```
In [ ]: #11 Write a function to print pairs of amicable numbers in a range
```

```
In [37]: #from code_9 import sumPdivisors
         def checkAmicable(n):
             a=dict()
             b=dict()
             for i in range (n):
                 o=sumPdivisors(i)[1]
                 if o>1 and o!=i:
```

```
                    a.update({i:o})

            # Old Fashion For Loop
            # for k in a.keys():
            #     for k1,v in a.items():
            #         if(k==v and a[v]==k1):
            #             print(k,a[k])

            # Comprehension list
            c={k:a[k] for k in a.keys() for k1,v in a.items() if (k==v and a[v]==k1)}
            print("Amicable pairs:\n{}".format(c))

        checkAmicable(int(input("Enter the range to find amicable numbers:")))
```

```
Enter the range to find amicable numbers:10000
Amicable pairs:
{220: 284, 284: 220, 1184: 1210, 1210: 1184, 2620: 2924, 2924: 2620, 5020: 5564, 5564: 5020, 62
```

```
In [ ]: #12 Write a program which can filter odd numbers in a list by using filter function
```

```
In [38]: def oddNumbers(n):
             if n%2!=0:
                 return True

         n=int(input("Enter the range:"))
         l=range(n)
         print("Odd numbers under the range {}:".format(n))
         print(list(filter(oddNumbers,l)))
```

```
Enter the range:30
Odd numbers under the range 30:
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29]
```

```
In [ ]: #13 Write a program which can map() to make a list whose elements are cube of elements
```

```
In [39]: def cubeElements(n):
             return n**3

         if __name__ == "__main__":
             n=int(input("Enter the range:"))
             l=range(n)
             o=map(cubeElements,l)
             print("List of cube elements:\n{}".format(list(o)))
```

```
Enter the range:10
List of cube elements:
[0, 1, 8, 27, 64, 125, 216, 343, 512, 729]
```

In [40]:
```
#14 Write a program which can map() and filter() to make a list whose elements are cu
#even number in a given list
```

In [41]:
```
#from code_13 import cubeElements

def evenNumbers(n):
    if n%2==0:
        return True

n=int(input("Enter the range:"))
l=range(1,n)

evenList=list(filter(evenNumbers,l))
cube_of_even_list=list(map(cubeElements,evenList))

print("Even number under {}:\n".format(evenList))
print("Cube of even number list:\n{}".format(cube_of_even_list))
```

```
Enter the range:15
Even number under [2, 4, 6, 8, 10, 12, 14]:

Cube of even number list:
[8, 64, 216, 512, 1000, 1728, 2744]
```