Optional Assignment - Python Functions

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

```
#Defining function
def Mul_table(x):
  print("Multiplication Table of",x,":")
  for i in range(1,11): # iterating over the loop for 10 times
    print(x,"*",i,"=",x*i)

Mul_table(5)
```

```
Multiplication Table of 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
```

2.Write a program to print twin primes less than 1000

```
#Function to check if a number is prome
def prime(n):
  for i in range (2,n):
    if n%i == 0:  # checking if number is divisible
      return False
  return True

#If two consecutive odd numbers are prime its twin primes
def twin_prime(m):
  for i in range (2,m):
      j = i+2 #taking j as next odd number
      if(prime(i) and prime(j)): # checking if both numbers are prime numbers
        print(i,"and",j)

twin_prime(1000)
```

```
3 and 5
5 and 7
11 and 13
17 and 19
```

```
29 and 31
41 and 43
59 and 61
71 and 73
101 and 103
107 and 109
137 and 139
149 and 151
179 and 181
191 and 193
197 and 199
227 and 229
239 and 241
269 and 271
281 and 283
311 and 313
347 and 349
419 and 421
431 and 433
461 and 463
521 and 523
569 and 571
599 and 601
617 and 619
641 and 643
659 and 661
809 and 811
821 and 823
827 and 829
857 and 859
881 and 883
```

3.Write a program to find prime factors of a number

```
#code was reffered from https://www.codesansar.com/python-programming-examples/prime-factors.

def prime_factors(n):
    i = 2
    prime_factors = [] #creating empty set for prime factors
    while i*i <= n:
        if n%i == 0: # checking if number is divisible by 2 at first
            prime_factors.append(i)
            n = n/i #changing n by dividing it by divisor
        else:
            i += 1 # iterating over n in increments of 1

    if n>1:
        prime_factors.append(n)

    return prime_factors

prime_factors(56)
```

```
[2, 2, 2, 7.0]
```

4.Write a program to implement the formula of permutaion and combination

```
#formula of permutaion and combination are using factorial
#code for finding factorial using reccurssion

def fact(n):
  return 1 if n == 1 else (n*fact(n-1))

# permutaion and combination using equation
def npr(n,r):
    return (fact(n)/fact(n-r))

def ncr(n,r):
    return (npr(n,r)/fact(r))

print(npr(9,4))

print(ncr(9,4))
```

```
3024.0
126.0
```

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

```
#code was refferred from https://www.besanttechnologies.com/decimal-to-binary-in-python#:~:te
def decimal_to_binary(n):
  if(n>1):
    decimal_to_binary(n//2) # flat division by discarding the reminder
  print(n%2,end="") # if reminder, then its updated as 1 else updated as zero

decimal_to_binary(50)
```

```
110010
```

6.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

```
def cubesum(n):
    sum=0 #initializing condition
    while(n!=0):
        sum += ((n%10)**3) # taking reminder of subsequent divisions by 10 to get number at t
        n = n//10 # flat division to get next digit
    return sum
```

```
#checking for output of cubesum()
cubesum(101)
```

```
     2
```

```
#Amstrong number is a number for which cubesum(n) = n ;

def PrintArmstrong(n):
  for i in range (1,n):
    if (cubesum(i) == i):
      print (i)

PrintArmstrong(500)
```

```
     1
     153
     370
     371
     407
```

```
#function to check if a number is amstrong number

def isArmstrong(n):
  if (cubesum(n) == n):
    print(n,"is an Amstrong number")
  else:
    print(n,"is not an Amstrong number")

isArmstrong(160)
isArmstrong(370)
```

```
     160 is not an Amstrong number
     370 is an Amstrong number
```

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

```
def prodDigits(n):
  product = 1 #initializing condition
  while (n!=0):
    product *=(n%10) # taking reminder of subsequent divisions
    n=n//10 # flat division to get the next digit
  return product

prodDigits(54)
```

```
     20
```

8)If all digits of a number n are multiplied by each other repeating with the product, the one digit number obtained at last is called the multiplicative digital root of n. The number of times digits need to be multiplied to reach one digit is called the multiplicative persistance of n.

```
#MDR and MP can be defined for two digit numbers onwards

def MDR(n):
  while n >= 10 :
    n = prodDigits(n) # output of prodDigits taken as next number
  return n
MDR(66)
```

```
8
```

```
def MP(n):
  i = 0 # setting iteration as zero
  while n >= 10 :
    n = prodDigits(n)
    i += 1
  return i

MP(66)
```

```
3
```

9)Write a function sumPdivisors() that finds the sum of proper divisors of a number.

```
def sumPdivisors(n):
    a=[] # initializing an empty set
    for i in range(1,n):
        if(n%i==0): # proper divisor if reminder is zero
            a.append(i)
    print(a)
    return sum(a)

sumPdivisors(64)
```

```
[1, 2, 4, 8, 16, 32]
63
```

10)A number is called perfect if the sum of proper divisors of that number is equal to the number

```
#For a perfect number, sum of proper divisors is equal to the number itself

def sumPdivisors(n):
```

```
    a=[] # initializing an empty set
    for i in range(1,n+1):
        if(n%i==0): # proper divisor if reminder is zero
            a.append(i)
    return sum(a)

def perfect_num(n):
    for i in range (1,n+1):
      if (sumPdivisors(i) == i):
        print (i)

perfect_num(500)
```

```
 6
 28
 496
```

11)Two different numbers are called amicable numbers if the sum of the proper divisors of each is equal to the other number

```
def Pdivisors(n):
    a=[] # initializing an empty set
    for i in range(1,n):
        if(n%i==0): # proper divisor if reminder is zero
            a.append(i)
    return a
def Amicable(x,y):
    if(sum(Pdivisors(x))==y and sum(Pdivisors(y))==x):
        print(x,'and',y,'are amicable')
    else:
        print(x,'and',y,'are non amicable')

Amicable(220,284)
Amicable(111,216)
```

```
 220 and 284 are amicable
 111 and 216 are non amicable
```

12)Write a program which can filter odd numbers in a list by using filter function

```
lst=[6,9,11,14,13,76,89,8,15,18]

Oddnum = list(filter(lambda x : (x%2 == 1),lst)) # Using Lambda Function

print(Oddnum)
```

```
 [9, 11, 13, 89, 15]
```

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

```
lst = [1,2,3,4,5,6,7,8,9,10]

NumCube = list(map(lambda x : (x**3),lst)) # Using Lambda Function

print(NumCube)
```

```
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

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

```
lst = [1,2,3,4,5,6,7,8,9,10]

EvenNum = list(filter(lambda x : (x%2 == 0),lst)) # Using Lambda Function
Evencube = list(map(lambda x : (x**3),EvenNum)) # Using Lambda Function

print(Evencube)
```

```
[8, 64, 216, 512, 1000]
```