Import Section

```
import math
```

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

```
def multiplication_table(num):
    for sel in range(1,11):
        print(num,'X',sel ,'=',num*sel)
multiplication_table(1000)
```

```
1000 X 1 = 1000
1000 X 2 = 2000
1000 X 3 = 3000
1000 X 4 = 4000
1000 X 5 = 5000
1000 X 6 = 6000
1000 X 7 = 7000
1000 X 8 = 8000
1000 X 9 = 9000
1000 X 10 = 10000
```

Question 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

```
%%time
def twinprimes(num):
    if num == 1:
        return []
    else:
        primelist = [2]
        twin_prime = []
        for sel in range(3,num+1,2):
            for val in range(2,math.ceil(sel/2)+1):
                if sel%val == 0:
                    break
                elif val == math.ceil(sel/2):
                    primelist.append(sel)
                    if primelist[-1] - primelist[-2] == 2:
                        twin_prime.append((primelist[-2],primelist[-
1]))
        print(twin_prime)


twinprimes(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)]
CPU times: user 6.68 ms, sys: 211 µs, total: 6.89 ms
Wall time: 6.75 ms
```

Question 3: Write a program to find out the prime factors of a number. Example: prime factors of 56 - 2, 2, 2, 7

```
%%time
def primefact(n):
    primefactlist=[]
    # Print the number of two's that divide n
    while n % 2 == 0:
        primefactlist.append(str(2)),
        n = n / 2
    # n must be odd at this point
    # so a skip of 2 ( i = i + 2) can be used
    for i in range(3,int(math.sqrt(n))+1,2):
        # while i divides n , print i and divide n
        while n % i== 0:
            primefactlist.append(str(i)),
            n = n / i
    # Condition if n is a prime
    # number greater than 2
    if n > 2:
        primefactlist.append(str(int(n)))
    return ','.join(primefactlist)

primefact(56)
```

```
CPU times: user 17 µs, sys: 2 µs, total: 19 µs
Wall time: 21 µs
```

```
'2,2,2,7'
```

Question 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 of combinations of n objects taken r at a time is: c(n, r) = n! / (r!*(n-r)!) = p(n,r) / r!

```
%%time
def permutate_combi(n,r):
    if n < r or n <= 0 or r <=0:
        return 'please give n values greater than r or give n and r
greater than 0'
    else:
        n_fact=1
        n_r_fact = 1
        r_fact = 1
        for val in range(1,n+1):
            n_fact*=val
            if val == n-r:
```

```
                n_r_fact = n_fact
            if val == r:
                r_fact = n_fact
        print("p(n,r) : ",n_fact/n_r_fact)
        print("c(n,r) : ",n_fact/(n_r_fact * r_fact))

permutate_combi(10,2)
```

```
p(n,r) :  90.0
c(n,r) :  45.0
CPU times: user 164 µs, sys: 192 µs, total: 356 µs
Wall time: 233 µs
```

Question 5: Write a function that converts a decimal number to binary number

```
%%time
def deci_bin_convert(num):
    bin=[]
    while num != 0:
        bin.append(str(num%2))
        num = int(num/2)
    print(",".join(bin[::-1]))
deci_bin_convert(2000)
```

```
1,1,1,1,1,0,1,0,0,0,0
CPU times: user 39 µs, sys: 10 µs, total: 49 µs
Wall time: 46 µs
```

Question 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.

```
# https://www.quora.com/What-is-an-Armstrong-number

%%time
def cubesum(num,power=3):
    sums = 0
    while num != 0:
        mul = 1
        for i in range(1,power+1):
            mul *= (num%10)
        sums+=mul
        num = int(num/10)
    return sums

def printarmstrong(num):
    armstring = []
```

```
    for i in range(1,num+1):
        if i == cubesum(i,len(str(i))):
            armstring.append(str(i))
    return ', '.join(armstring)



def isarmstrong(num):
    if num == cubesum(num,len(str(num))):
        string = 'it is armstring number'
    else:
        string = 'it is not an armstrong number'
    return string

print(cubesum(153))
print(printarmstrong(2000))
print(isarmstrong(1634))
```

```
153
1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634
it is armstring number
CPU times: user 4.03 ms, sys: 54 µs, total: 4.08 ms
Wall time: 4.06 ms
```

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

```
%%time
def prodDigits(num):
    prods = 1
    while num != 0:
        prods *= (num%10)
        num = int(num/10)
    return prods
prodDigits(123456789)
```

```
CPU times: user 26 µs, sys: 1e+03 ns, total: 27 µs
Wall time: 30 µs

362880
```

Question 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. Example: 86 -> 48 -> 32 -> 6 (MDR 6, MPersistence 3) 341 -> 12->2 (MDR 2, MPersistence 2)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

```
def MDR(num):
    if len(str(num)) == 1:
        return num
    else:
        while len(str(num)) != 1:
            num = prodDigits(num)
        else:
            return num
def MPersistance(num):
    if len(str(num)) == 1:
        return 1
    else:
        val = 0
        while len(str(num)) != 1:
            num = prodDigits(num)
            val+=1
        else:
            return val

print('MDR of {} is {} '.format(86,MDR(86)))
print('MPersistance of {} is {} '.format(86,MPersistance(86)))
print('MDR of {} is {} '.format(341,MDR(341)))
print('MPersistance of {} is {} '.format(341,MPersistance(341)))
```

```
MDR of 86 is 6
MPersistance of 86 is 3
MDR of 341 is 2
MPersistance of 341 is 2
```

Question 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

```
%%time
def sumPdivisors(num):
    sums=0
    for val in range(1,int(num/2)+1):
        if num % val == 0:
```

```
            sums+=val
    return sums

sumPdivisors(36)
```

```
CPU times: user 14 µs, sys: 0 ns, total: 14 µs
Wall time: 16.2 µs

55
```

Question 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

```
def perfectnum(range1,range2):
    for sel in range(range1,range2+1):
        #print('--------------')
        #print(sel)
        sums=0
        for val in range(1,int(sel/2)+1):
            if sel%val == 0:
                #print(val)
                sums+=val
        if sel==sums:
                print(sel,end = ' ,')
perfectnum(1,1000)
```

```
6 ,28 ,496 ,
```

Question 11: 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 = 284Sum of proper divisors of 284 = 1+2+4+71+142 = 220Write a function to print pairs of amicable numbers in a range

```
def amicable_nums(num1,num2):
    amiclist=[]
    for val in range(num1,num2+1):
        sums = 0
        sum2=0
        for sel in range(1,int(val/2)+1):
            if val%sel==0:
                sums+=sel
        for val1 in range(1,int(sums/2)+1):
            if sums%val1==0:
                sum2+=val1
```

```
        if (val == sum2) and (val != sums):
            amiclist.append(tuple(sorted((val,sums))))
    print(sorted(set(amiclist)))

amicable_nums(1,10000)
```

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

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

```
def oddremover(var):
    if var%2==0:
        return var
inptlist=[1,2,3,4,5,6,7,8,90,10,11]
print(list(filter(oddremover,inptlist)))
```

```
[2, 4, 6, 8, 90, 10]
```

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

```
def cubes(var):
    return var*var*var
inptlist=[1,2,3,4,5,6,7,8,90,10,11]
print(list(map(cubes,inptlist)))
```

```
[1, 8, 27, 64, 125, 216, 343, 512, 729000, 1000, 1331]
```

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

```
inptlist=[1,2,3,4,5,6,7,8,90,10,11]
print(list(map(cubes,list(filter(oddremover,inptlist)))))
```

```
[8, 64, 216, 512, 729000, 1000]
```