1. wap to take input a number and return the multiplication of number

In [749…
```
def pro(i):
    a = [i*n for n in range(1,11)]
    return a

print(pro(2))
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

2. wap to pint the twin prime less than 1000

In [750…
```
def twinprime(maxrange):
    pair=[]
    a = []
    b= []
    for x in range(2,maxrange):
        for y in range(2,(x//2)+1):
            if(x % y) == 0:
                break
        else:
            pair.append(x)
    pair.sort()
    for dig in range(len(pair)):
        if pair[dig]+2 in pair:
            a.append(pair[dig])
            b.append(pair[dig]+2)
    item = zip(a,b)
    print(list(item),end=" ")
```

In [751…
```
twinprime(1000)
```

```
[(3, 5), (5, 7), (11, 13), (17, 19), (29, 31), (41, 43), (59, 61), (71, 73), (101, 103), (107, 109), (137, 13
9), (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), (65
9, 661), (809, 811), (821, 823), (827, 829), (857, 859), (881, 883)]
```

3.wap for the print the prime factors of the number

In [752…
```
def primefactor(num):
    prime=[]
    b = []
    for x in range(2,(num//2)+1):
        for y in range(2,(x//2)+1):
            if(x % y) == 0:
                break
        else:
            prime.append(x)
    for x in range(len(prime)):
        if(num%prime[x]==0):
            while(num%prime[x]==0):
                num = num//prime[x]
                b.append(prime[x])
    return(b)
```

In [753…
```
primefactor(56)
```

Out[753…
```
[2, 2, 2, 7]
```

5. wap to covert the decimal to binary number

In [754…
```
def dectobi(num):
    rem=[]
    while num>0:
        a = num%2
        rem.append(a)
        num = num//2
    return(''.join(map(str,rem[::-1])))
```

In [755…
```
dectobi(6)
```

Out[755…
```
'110'
```

12. wap to filter odd number from the list

In [756…
```
def odd(list_1):
        if(list_1%2 == 0):
            return False
        return True

r_list =[1,3,2,5,7,9,6,5,13,1,41,15,11,19,17,83]

task = list(filter(odd,r_list))
for responce in task:
    print(responce,end=" ")
```

```
1 3 5 7 9 5 13 1 41 15 11 19 17 83
```

7. wap to print the product of the digit

In [757…
```
def prodDigits(num):
    prod = 1
    while num>1:
        a = num%10
        prod = prod*a
        num = num//10
    return(prod)
```

In [758…
```
prodDigits(145)
```

Out[758…
```
20
```

9.wap to print the sum of proper divisors of the number

In [759…
```
def sumPdivisors(num):
    sum1 = 0
    for x in range(1,(num//2)+1):
        if num%x == 0:
            sum1 = sum1+x
    return(sum1)
```

In [760…
```
sumPdivisors(36)
```

Out[760…
```
55
```

10. wap to print the perfect number from the range

In [761…
```
def perfectnum(min_range,max_range):
    for num in range(min_range,max_range+1):
        sum1 = 0
        for x in range(1,(num//2)+1):
            if num%x == 0:
                sum1 = sum1+x
                if sum1 == num:
                    print(num,end=' ')
```

In [762…
```
perfectnum(20,30)
```

```
24 28
```

8. wap for the multiplicative digital roots and its persistence

In [763…
```
def MDR(num):
    count = 0
    while num>10:
        count = count+1
        pro = prodDigits(num)    #using prodDigits()
        print("digit:",pro)
        print("term:",count)
        if(pro>10):
            pass
        num = pro
```

In [764…
```
MDR(86)
```

```
digit: 48
term: 1
digit: 32
term: 2
digit: 6
term: 3
```

6. wap for print cube sum , armstrong and isArmstrong

In [765…
```
def sumcube(num):
    sum1 = 0
    while num>=1:
        a = num%10
        cube = a**3
        sum1 = sum1+cube
        num = num//10
    return(sum1)

def armstrong(num):
    digit = sumcube(num)
    if digit == num:
        return(num)

def isarmstrong(num):
    a = armstrong(num)
    if a:
        return("yes its armstrong")
    else:
        return("sorry")
```

In [766…
```
sumcube(153)
```

Out[766…
```
153
```

In [767…
```
armstrong(153)
```

Out[767…
```
153
```

In [768…
```
isarmstrong(153)
```

Out[768…
```
'yes its armstrong'
```

11. wap for print the pair of the amicable number in the given range

In [769…
```
def amicable(min_range,max_range):
    for num in range(min_range,max_range+1):
        sum1 = 0
        sum2 = 0
        for x in range(1,(num//2)+1):
            if num%x == 0:
                sum1 = sum1+x
        num2 = sum1
        if num2<= max_range:
            for x in range(1,(num2//2)+1):
                if num2%x == 0:
                    sum2 = sum2+x
            if sum2 == num and num != num2:
                print(num,num2)
```

In [770…
```
amicable(200,300)
```

```
220 284
284 220
```

4. c(n/r) = n!/r!(n-r)! = p(n!/(n-r)!) proof by evaluate the equation

In [771…
```
def formula(n,r):
    nf,df,rf = 1,1,1
    drv = (n-r)
    while drv>=1:
        df = df*drv
        drv = drv-1
    while n>=1:
        nf = nf*n
        n = n-1
    while r>=1:
        rf = rf*r
        r = r-1

    per = (nf//df)//rf
    com = nf//(rf*df)
    print(per,com)        #c(n/r) = n!/r!(n-r)! = p(n!/(n-r)!)
```

In [772…
```
formula(4,3)
```

```
4 4
```

13. wap which can map() to make a list whose element are cube of element in a given list

In [773…
```
def cubes(r_list):
     return(r_list**3)

r_list = [1,2,4,3]

task = map(cubes,r_list)
print(list(task))
```

```
[1, 8, 64, 27]
```

14.wap which can map() and filter () to make a list whose element are cube of element of even number in a given list

In [774…
```
def even(list_1):
        if(list_1%2 == 0):
            return True
        return False

r_list =[2,5,7,8,6,5]

task = list(filter(even,r_list))
f_list = []
for x in task:
    f_list.append(x**3)
print(f_list)
```

```
[8, 512, 216]
```