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

num=int (input("Enter a number: "))
table=range(1,11)
for i in table:
    print('{} * {} = {}'.format(num,i,num*i))
```

In [ ]:
```
"""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"""
start=3
stop=1000
primes=[]
prime=True
for i in range (start,stop):
    for j in range (2,i):
        if (i%j==0):
            prime=False
            break
        else:

            prime =True
    if (prime):
        primes.append(i)
twinprime=[]
for i in range(0,len(primes)):
    for j in range (i,len(primes)):
        if (abs(primes[i]-primes[j])==2):
            print ('({},{}),'.format(primes[i],primes[j]),end=' ')
```

In [ ]:
```
"""3. Write a program to find out the prime factors of a number. Example: prim
e factors of 56 -
2, 2, 2, 7"""
factors=[]
num=int(input('Enter a number: '))
for i in range (1,num):
    if (num%i==0):
        factors.append(i)
isprime=[]
for ele in factors:
    isPrime=True
    for j in range (2,ele):
        if (ele%j==0):
            isPrime=False
            break
        else:
            isPrime=True
    if isPrime:
        isprime.append(ele)
print (isprime)
```

In [ ]:
```
"""4. Write a program to implement these formulae of permutations and combinat
ions.
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!"""


n=int (input('Number of permutations of n objects: '))
r=int (input('Number of permutations of n objects taken r at a time: '))

def Permutation(n,r):
    return (factorial(n)/factorial(n-r))
def Combination(n,r):
    return (Permutation(n,r)/factorial(r))


def factorial(num):
    if num==1:
        return 1
    else:
        return num*factorial(num-1)



print ('Permutation of () and () is ' .format(n,r),end ='')
print ( int (Permutation (n,r)))
print ('Combination of () and () is ' .format(n,r),end ='')
print (int (Combination(n,r)))
```

In [ ]:
```
"""5. Write a function that converts a decimal number to binary number"""

num=int(input('Enter a number: '))
def dectoBinary(n):
    if (n>1):
        dectoBinary(n//2)
    print (n%2,end=' ')
dectoBinary(num)
```

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


num=int (input('Enter a number: '))
ll=[int(i) for i in str(num)] # converting int type into list as int is not it
erable object

# print (ll)
def cubesum(ll):
    sum=0
    for ele in ll:
        sum+=ele**3
    return sum
armstrongPrg=cubesum(ll)
if (armstrongPrg==num):
    print ("() is an armstrong number".format(num))
else :
    print ("() is not an armstrong number".format(num))
```

In [ ]:
```
"""7. Write a function prodDigits() that inputs a number and returns the produ
ct of digits of that
number."""

def prodDigits():
    num=int(input("Please, Enter a number: "))
    ll=[int(x) for x in str(num)]
    prod=1
    for ele in ll:
        prod*=ele
    print ('The product of bigits of () is () '.format(num,prod))
prodDigits()
```

In [ ]:
```
"""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 multiplica
tive
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"""

num=int(input("Please, Enter a number: "))

def prodDigits(num):
    ll=[int(x) for x in str(num)]
    prod=1
    for ele in ll:
        prod*=ele
    MDR(prod)

def MDR(num):

    if num < 10:
        print ('MDR is: {} and Mpersistence is {}: '.format(num,count))
    else:

        count =count+1
        prodDigits(num)

count=0
prodDigits(num)
```

In [ ]:
```
"""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, excep
t the
number itself. For example proper divisors of 36 are 1, 2, 3, 4, 6, 9, 18"""

num=int (input('Please, enter a number'))
ll=[]
def sumPdivisors(num):
    for i in range (1,num):
        if (num%i==0):
            ll.append(i)
    print (ll)

sumPdivisors(num)
sum=0
for ele in ll:
    sum+=ele
print ("sum of proper divisors of () is ()".format(num,sum))
```

In [ ]:
```
"""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"""

num1=int (input('Start: '))
num2=int (input('Stop:  '))
listOfPerfectNumber=[]
sum=0
for i in range(num1,num2):
    for j in range(1,i):
        if (i%j==0):
            sum+=j
    if (sum==i):
        listOfPerfectNumber.append(i)
        sum=0
    else:
        sum=0
for ele in listOfPerfectNumber:
    print (ele)
```

In [11]:
```
"""11. Two different numbers are called amicable numbers if the sum of the pro
per divisors of
each is equal to the other number. For example 220 and 284 are amicable number
s.
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"""

import pdb
num1=int (input ('Enter the first number: '))
num2=int (input ('Enter the second number: '))
ll=[]
l2=[]
for i in range (num1,num2):
    ll.append(i)
# print (ll)

for ele in ll:
    sum=0
    for i in range (1,ele):
        if (ele%i==0):
            sum+=i
    if sum in ll:
        l2.append(ele)
def amicableNumbers (num):
    for ele in num:
        sum=0
        for i in range (1,ele):
            if (ele%i==0):
                sum+=i
        if sum in num:
#             print (sum)
            found= amicableFound(sum)
            check (sum,found)

def amicableFound(num):
    for ele in l2:
        if (ele==num):
            return ele

def check(sum,ele):
    sum1=0
    for i in range (1,ele):
        if (ele%i==0):
            sum1+=i
    if (sum1==sum):
        print ("() and () are amicable".format(sum,ele))


amicableNumbers(l2)
```

```
Enter the first number: 200
Enter the second number: 300
```

In [ ]:
```
"""12. Write a program which can filter odd numbers in a list by using filter
 function"""
ll=range(2,50)

odd=list(filter( lambda x: x%2 !=0,ll))
print (odd)
```

In [15]:
```
"""13. Write a program which can map() to make a list whose elements are cube
 of elements in
a given list"""


ll=list (range(1,10))
# print (ll)
def cube(num):
    if (num**3 in ll):
        return num


```