def gcd(a, b): # calculates GCD of a and d

while b != 0:

c = a % b

a = b

b = c

return a

def modinv(a, m): # calculates modulo inverse of a for mod m

for x in range(1, m):

if (a \* x) % m == 1:

return x

return None

def coprimes(a): # calculates all possible co-prime numbers with a

l = []

for x in range(2, a):

if gcd(a, x) == 1 and modinv(x, phi) != None:

l.append(x)

for x in l:

if x == modinv(x, phi):

l.remove(x)

return l

def encrypt\_block(m): # encrypts a single block

c = m \*\* e % n

return c

def decrypt\_block(c): # decrypts a single block

m = c \*\* d % n

return m

def encrypt\_string(s): # applies encryption

return ''.join([chr(encrypt\_block(ord(x))) for x in list(s)])

def decrypt\_string(s): # applies decryption

return ''.join([chr(decrypt\_block(ord(x))) for x in list(s)])

if \_\_name\_\_ == "\_\_main\_\_":

p = int(input('Enter prime p: '))

q = int(input('Enter prime q: '))

print("Choosen primes:\np=" + str(p) + ", q=" + str(q) + "\n")

n = p \* q

print("n = p \* q = " + str(n) + "\n")

phi = (p - 1) \* (q - 1)

print("Euler's function (totient) [phi(n)]: " + str(phi) + "\n")

print("Choose an e from a below coprimes array:\n")

print(str(coprimes(phi)) + "\n")

e = int(input())

d = modinv(e, phi) # calculates the decryption key d

print("\nYour public key is a pair of numbers (e=" + str(e) + ", n=" + str(n) + ").\n")

print("Your private key is a pair of numbers (d=" + str(d) + ", n=" + str(n) + ").\n")

s = input("Enter a message to encrypt: ")

print("\nPlain message: " + s + "\n")

enc = encrypt\_string(s)

print("Encrypted message: ", enc, "\n")

dec = decrypt\_string(enc)

print("Decrypted message: " + dec + "\n")