**RSA Algorithm**

**Server**

import socket

import random

def modexp(base, exp, mod):

result = 1

base = base % mod

while exp > 0:

if exp % 2 == 1:

result = (result \* base) % mod

exp = exp // 2

base = (base \* base) % mod

return result

def decrypt(ciphertext, private\_key):

d, n = private\_key

decrypted\_message = ''.join([chr(modexp(char, d, n)) for char in ciphertext])

return decrypted\_message

def generate\_keypair(p, q):

n = p \* q

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

j = 1

for i in range(1, n):

e = nn % i

if e != 0:

e = i

break

while True:

t = (e \* j) % nn

if t == 1:

d = j

print(f"Prime Numbers are p= {p} and q= {q}")

print(f"Private key is {d}")

break

j += 1

public\_key = (e, n)

private\_key = (d, n)

return public\_key, private\_key

def generate\_prime():

prime = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]

u\_n = random.sample(prime, len(prime))

p, q = u\_n[1], u\_n[2]

return p,q

**# Key generation**

p,q= generate\_prime()

public\_key, private\_key = generate\_keypair(p, q)

**# Socket setup**

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind(('192.168.1.47', 8080))

server\_socket.listen()

**# Wait for a connection**

client\_socket, addr = server\_socket.accept()

print(f"Connection from {addr}")

**# Send primes to the client**

primes\_message = f"{p},{q}"

client\_socket.send(primes\_message.encode())

**# Receive and decrypt message**

encrypted\_message = client\_socket.recv(1024).decode()

print("Encrypted message : ",encrypted\_message)

ciphertext = [int(char) for char in encrypted\_message.split(',')]

decrypted\_message = decrypt(ciphertext, private\_key)

print("Received and decrypted message:", decrypted\_message)

**# Clean up**

client\_socket.close()

server\_socket.close()

**Client**

import socket

def modexp(base, exp, mod):

result = 1

base = base % mod

while exp > 0:

if exp % 2 == 1:

result = (result \* base) % mod

exp = exp // 2

base = (base \* base) % mod

return result

def encrypt(message, public\_key):

e, n = public\_key

ciphertext = ','.join([str(modexp(ord(char), e, n)) for char in message])

return ciphertext

def generate\_keypair(p, q):

n=p\*q

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

j=1

for i in range(1,n):

e=nn%i

if e!=0:

e=i

break

while True:

t=(e\*j)%nn

if t==1:

d=j

break

j+=1

public\_key = (e, n)

private\_key = (d, n)

return public\_key, private\_key

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client\_socket.connect(('192.168.1.24', 8080))

**# Receive primes from the server**

primes\_message = client\_socket.recv(1024).decode()

p, q = map(int, primes\_message.split(','))

**# Key generation using received primes**

public\_key, private\_key = generate\_keypair(p, q)

print(f"public key is {public\_key[0]} ")

message = str(input("Enter Plaintext: "))

**# Encrypt and send message**

encrypted\_message = encrypt(message, public\_key)

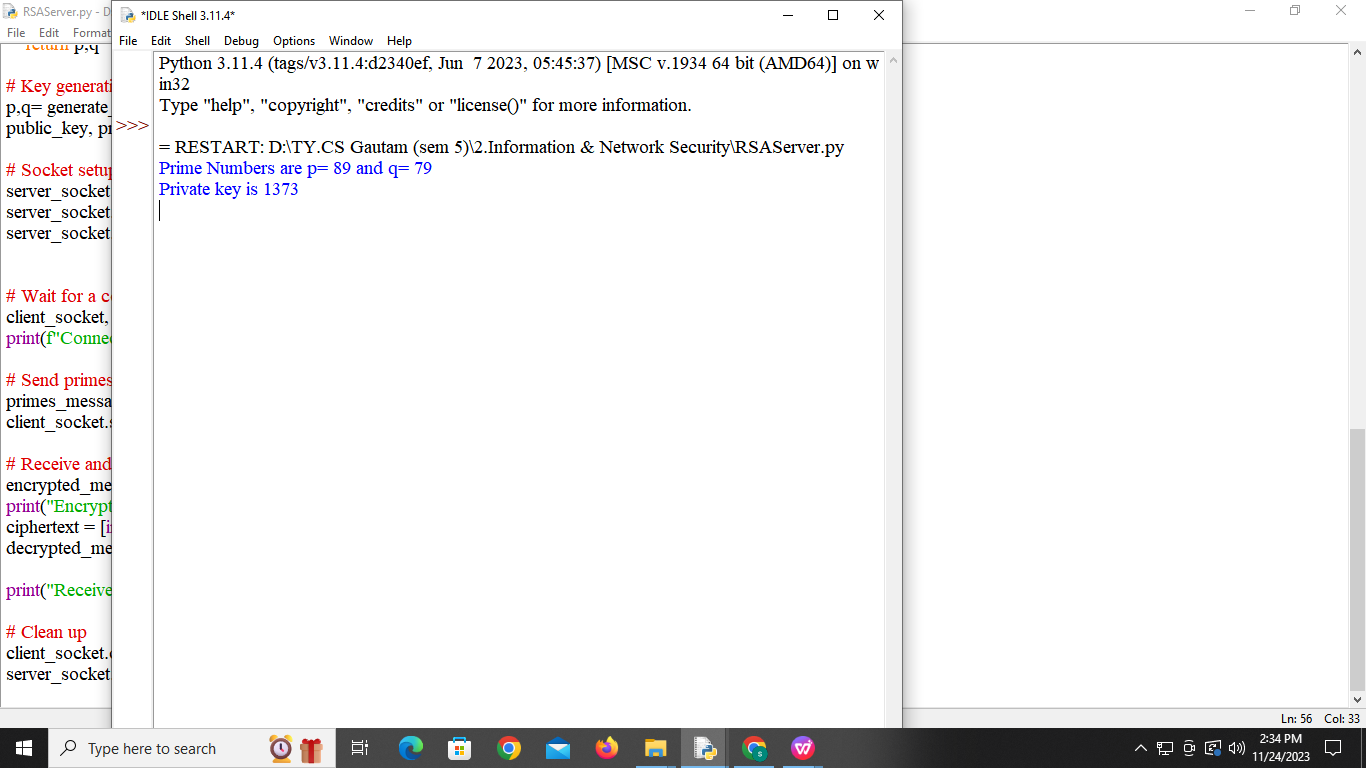
print("Encrypted Message: ",encrypted\_message)

client\_socket.send(encrypted\_message.encode())

client\_socket.close()

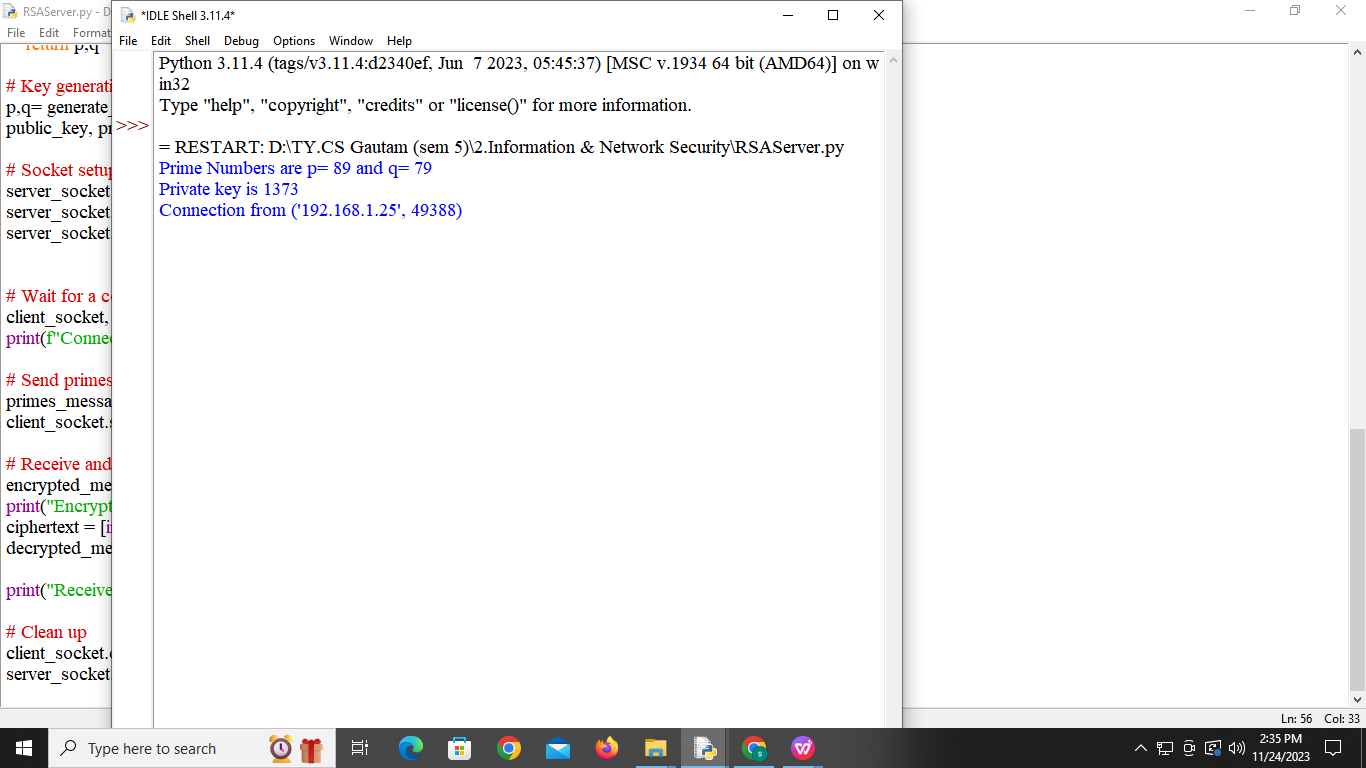
**Output**

**First run server program**

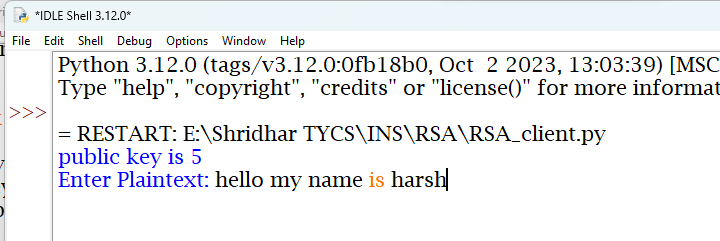
****

**Then Run client program**

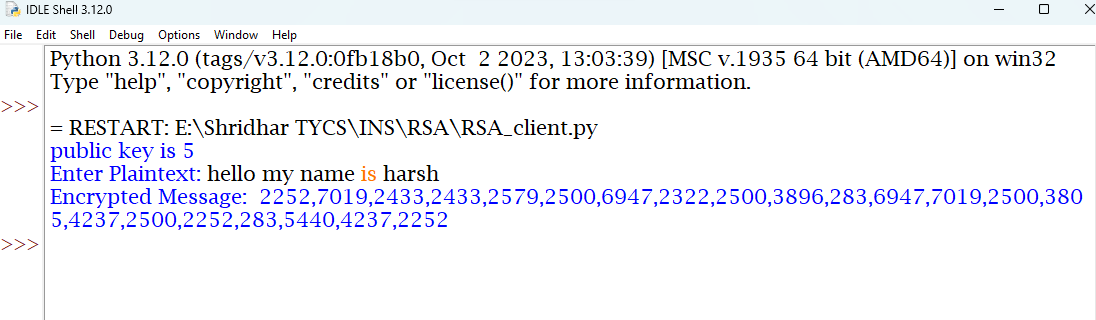
**After execution of client program client connected to the server**

****

**Enter message into plain text from client side**



**Program encrypt the message into ASCCII code**



**Encrypted message pass to server and server decrypt that message into plain text**

