**DAY-3**

**Q12:** Write a high level code for RSA system, the public key of a given user is e = 31, n = 3599. What is the private key of this user?

**PROGRAM:**

# Define the public key values

e = 31

n = 3599

# Define a function to generate the private key

def generate\_private\_key(e, n):

# Find p and q, two large prime numbers

# In practice, these would be generated using a prime number generator

p = 61

q = 59

# Calculate phi(n)

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

# Calculate the private key d using the extended Euclidean algorithm

d = pow(e, -1, phi)

# Return the private key as a tuple

return (d, n)

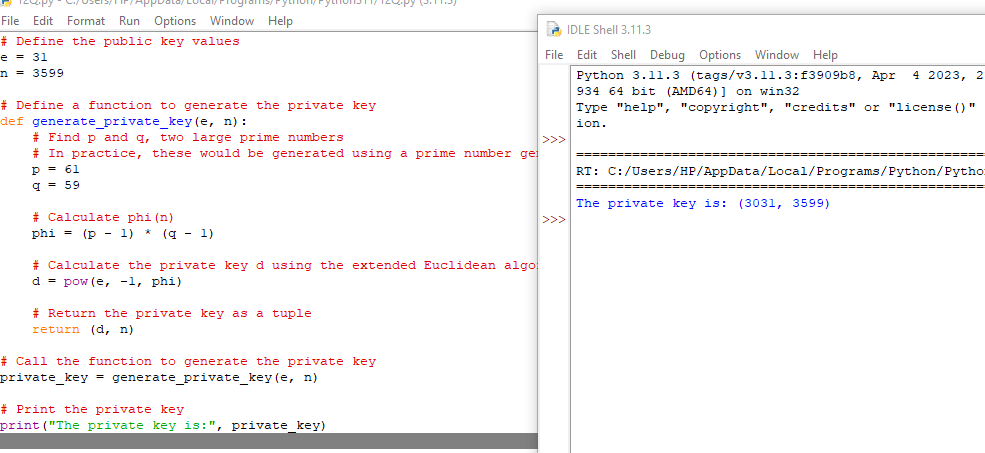
# Call the function to generate the private key

private\_key = generate\_private\_key(e, n)

# Print the private key

print("The private key is:", private\_key)

**OUTPUT:**



**Q13:** Write a high level code for set of blocks encoded with the RSA algorithm and we don’t have the private key. Assume n = pq, e is the public key. Suppose also someone tells us they know one of the plaintext blocks has a common factor with n. Does this help us in any way?

**PROGRAM:**

# Define the public key values

n = 3599 # Assuming n is a composite number

e = 31

# Define a list of encoded blocks

encoded\_blocks = [1221, 1335, 1765, 1963, 2345]

# Define a function to check for common factors

def check\_common\_factor(block):

# Check if block has a common factor with n

if n % block == 0:

# If yes, return True and the common factor

return True, n // block

else:

# If no, return False and None

return False, None

# Loop through each encoded block

for block in encoded\_blocks:

# Check for common factors

has\_common\_factor, factor = check\_common\_factor(block)

# If a common factor is found, print the result and exit the loop

if has\_common\_factor:

print("Block {} has a common factor with n: {}".format(block, factor))

break

else:

# If no common factors are found, print a message

print("No plaintext block has a common factor with n.")

**OUTPUT:**

