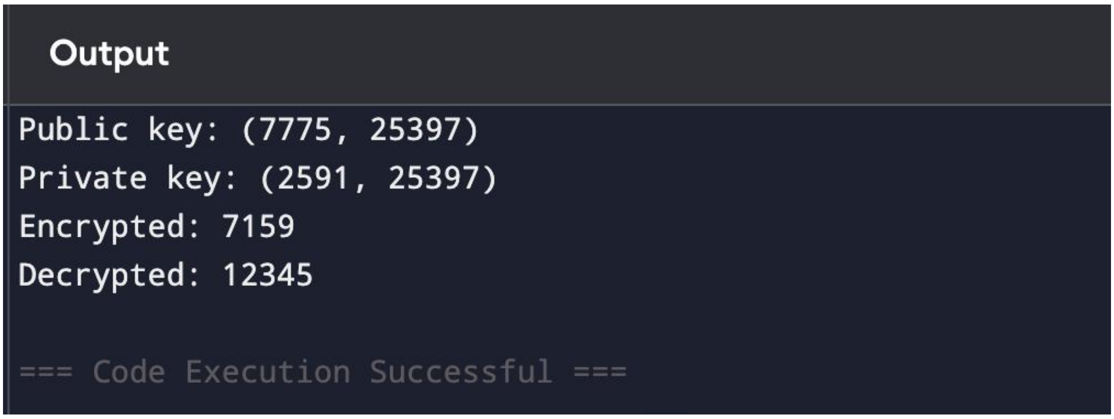
EXP 9: RSA

PROGRAM:

```
import random
from math import gcd

# Check if number is prime
def is_prime(n):
    if n <= 1: return False
    if n <= 3: return True
    if n % 2 == 0 or n % 3 == 0: return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

# Compute modular inverse
def mod_inverse(e, phi):
    for d in range(1, phi):
        if (e * d) % phi == 1:
            return d
    return None

# Generate keys
def generate_keys():
    p = q = 0
    while not (is_prime(p) and is_prime(q) and p != q):
        p = random.randint(100, 300)
        q = random.randint(100, 300)

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

    e = random.randrange(2, phi)
    while gcd(e, phi) != 1:
        e = random.randrange(2, phi)

    d = mod_inverse(e, phi)

    return (e, n), (d, n)

# Encrypt integer
def encrypt_integer(message_int, public_key):
    e, n = public_key
    return pow(message_int, e, n)

# Decrypt integer
def decrypt_integer(cipher_int, private_key):
    d, n = private_key
    return pow(cipher_int, d, n)

# Example usage
public_key, private_key = generate_keys()
print("Public key:", public_key)
print("Private key:", private_key)
```

```
message = 12345
cipher = encrypt_integer(message, public_key)
print("Encrypted:", cipher)

decrypted = decrypt_integer(cipher, private_key)
print("Decrypted:", decrypted)
```

OUTPUT:



```
Public key: (7775, 25397)
Private key: (2591, 25397)
Encrypted: 7159
Decrypted: 12345


=== Code Execution Successful ===
```