LAB-8

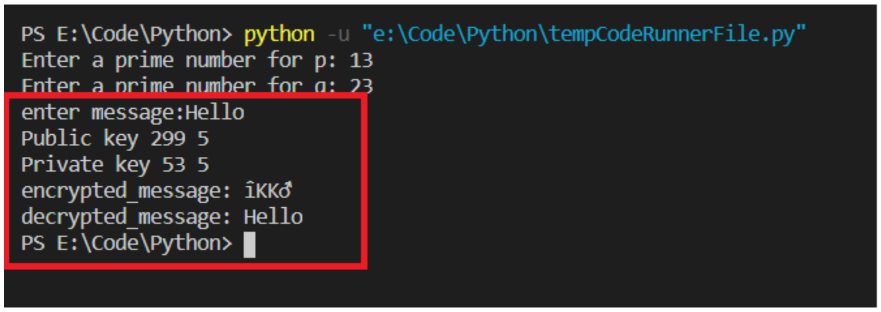
IMPLEMENTATION OF RSA:

CODE:

```
import math

def encrypted_msg(e,msg):

    encrypt = ''
    for c in msg:
        ch = ord(c)
        encrypt += chr((pow(ch,e)) % n)
    return encrypt

def decrypted_msg(encrypt,msg):

    decrypt = ''
    for c in encrypt:
        ch = ord(c)
        decrypt += chr((pow(ch,d) % n))
                 # P = (C ^ d) % n
    return decrypt

def public(z: int):
    e = 2
    while e < z:
        # check if this is the required `e` value
        if math.gcd(e, z)==1:
            return e
        # else : increment and continue
        e += 1

def private(e: int, z: int):
    # d -> ed = 1(mod z)        ; 1 < d < z
    d = 2
    while d < z:
        if ((d*e) % z)==1:

            return d
        d += 1

p = int(input("Enter a prime number for p: "))
q = int(input("Enter a prime number for q: "))
```

```
def Check(prime):
    if prime==2:
        return True
    elif prime<2 or prime%2==0:
        return False
    elif prime>2:
        for i in range(2,prime):
            if not(prime%i):
                return False
    return True

check_p = Check(p)
check_q = Check(q)
while check_p==False or check_q==False :
    print("Enter only prime numbers")
    p = int(input("Enter a prime number for p: "))
    q = int(input("Enter a prime number for q: "))
    check_p = Check(p)
    check_q = Check(q)

message = input("enter message:")    #input message
n=p*q # n value
z = (p-1)*(q-1)
e = public(z)                       #public key
d = private(e, z)                   #private key
print("Public key",n,e)
print("Private key",d,e)

encrypt= encrypted_msg(e,message)
decrypt=decrypted_msg(encrypt,d)

print("encrypted_message:", encrypt)
print("decrypted_message:" , decrypt)
```

RESULT:



```
PS E:\Code\Python> python -u "e:\Code\Python\tempCodeRunnerFile.py"
Enter a prime number for p: 13
Enter a prime number for q: 23
enter message:Hello
Public key 299 5
Private key 53 5
encrypted_message: îKKő
decrypted_message: Hello
PS E:\Code\Python>
```