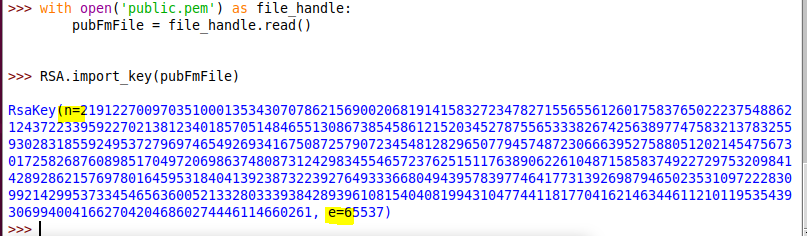
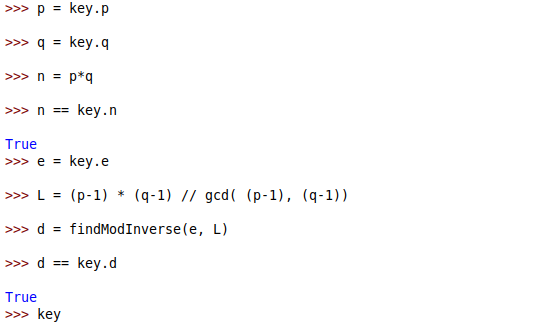
# Cryptography Homework 5a—Create an RSA Private/Public Key Pair in Python--KEY

## Hand In

Create a key, then export and save both the public and private keys. To answer these questions, you will need the gcd and findModInverse functions from cryptomath.py.

1. The private\_key and public\_key variables both contain base64 encoded text. The private key is much longer than the public key. What does the private key need to have that the public key does not (and should not) have?  
     
   The public key only contains n and e. It must not contain d, p, q, or u. If it did, the key is compromised.
2. Verify your answer to question 1. Read the public.pem file you saved into a variable (for example, “pub”,) then convert it to a key with the method RSA.import\_key(pub). What does the public key contain? Do the same thing for the private key. What does the private key contain?  
     
   
3. Check to see if the RSA key generation agrees with what was covered in the Crypto 4 module. Extract p, q, and d from the private key (private\_key.p, etc.)
   1. Compute n = p \*q. Is the result the same as private\_key.n ?  
        
      see below
   2. Compute L = lcm( (p-1) \* (q-1) ) Remember that lcm(a, b) = a \* b // gcd(a, b)  
        
      see below
   3. Compute d = e-1 mod L. Is the result the same as private\_key.e ?  
        
      see below

Note: The lecture notes use lambda = lcm( (p-1) \* (q-1) ). In Python lambda is a reserved word, so I’ve changed it to L.



For verification, here are the full values

