```
'''
Public Key is (e,n)
Private Key is (d,n)
'''
def generate_keys():

  from random import randint
  from math import gcd
  from sympy import mod_inverse

  # Generate Two unequal Large Primes of comparable size
  p, q = 877, 751
  #p, q = 6971, 6299

  # For large p and q, n will take centuries to factorize
  n = p*q

  # phi function
  fi_n = (p-1)*(q-1)

  # e and fi_n are relatively prime if their gcd is 1
  while True:
    e = randint(1, fi_n)
    if gcd(fi_n, e) == 1:
      break

  # inverse modulo exists iff e and fi_n are relatively prime
  # Modular Inverse
  d = mod_inverse(e, fi_n)

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


public_key, private_key = generate_keys()
```

```
def encryption(m, public_key):
  e, n = public_key
  c = m**e % n
  return c

c = encryption(ord('A'), public_key)
c
```

```
4610
```

```
def decryption(c, private_key):
  d, n = private_key
  p = c**d % n
  return p

p = decryption(c, private_key)
```

```
p = decryption(c, private_key)
chr(p)
```

```
'A'
```

```
def encrypt_text(plain_text, public_key):
  cipher_text = ''
  for character in plain_text:
    cipher_text += chr(encryption(ord(character), public_key))
  return cipher_text

cipher_text = encrypt_text('Cool', public_key)
cipher_text
```

```
'\U000322c1\U0009b310\U0009b310\U00063da4'
```

```
def decrypt_text(cipher_text, private_key):
  decrypted_text = ''
  for character in cipher_text:
    decrypted_text += chr(encryption(ord(character), private_key))
  return decrypted_text

decrypted_text = decrypt_text(cipher_text, private_key)
decrypted_text
```

```
'Cool'
```

✓ 3s completed at 07:44