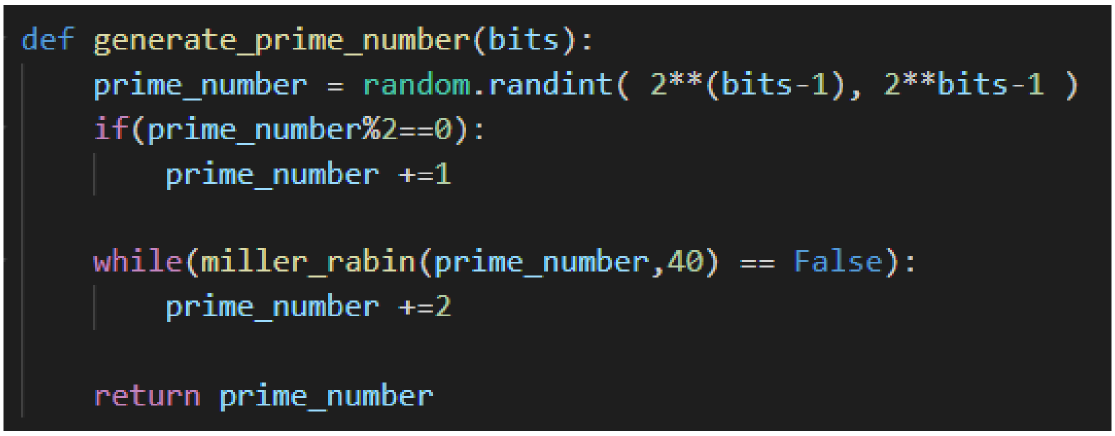
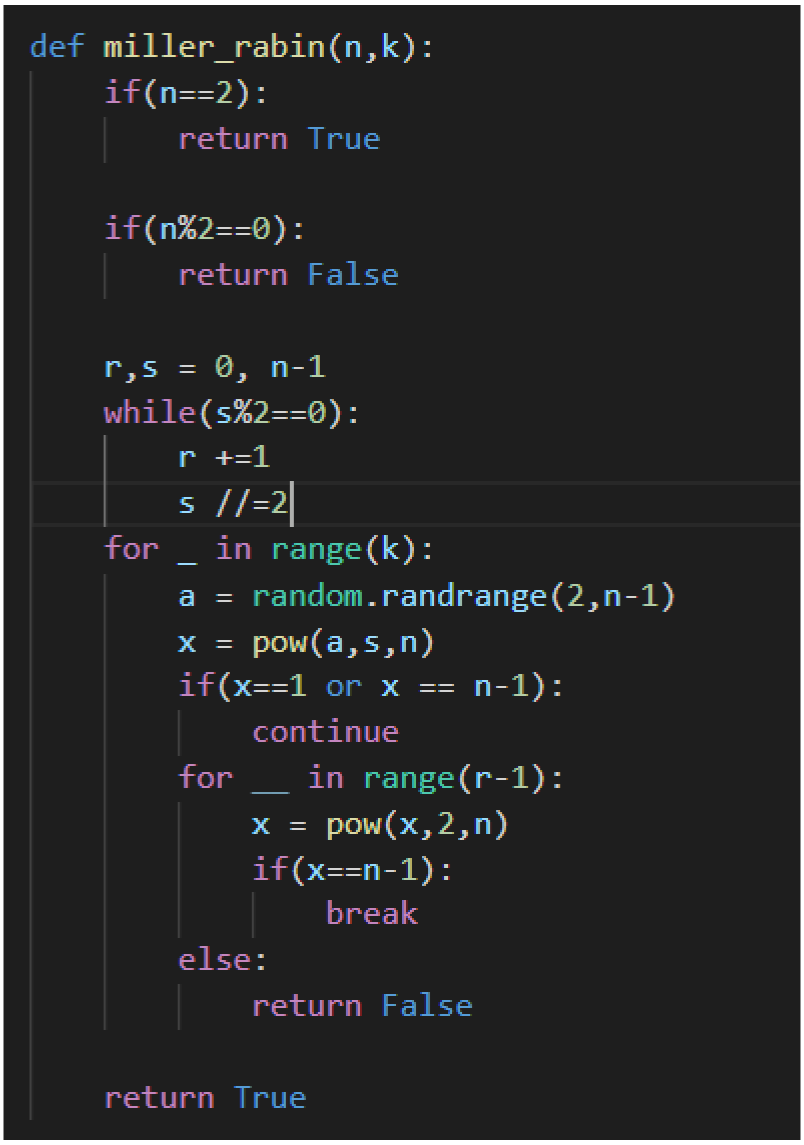
1. 生成 512bit 的質數



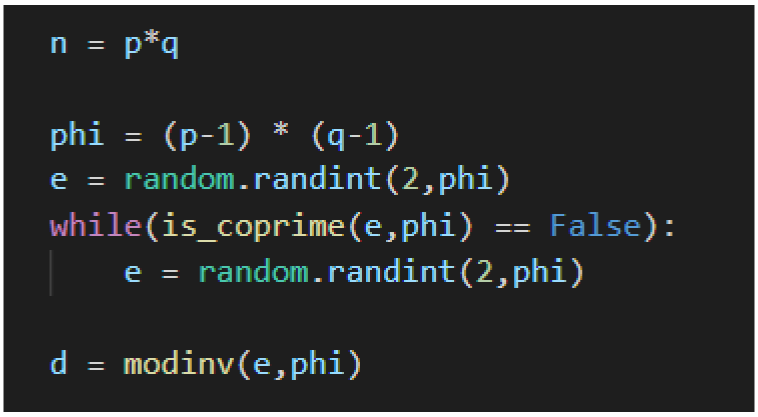
```python
def generate_prime_number(bits):
    prime_number = random.randint( 2**(bits-1), 2**bits-1 )
    if(prime_number%2==0):
        prime_number +=1

    while(miller_rabin(prime_number,40) == False):
        prime_number +=2

    return prime_number
```

2. 用 Miller Rabin 判斷生成的數字是不是質數



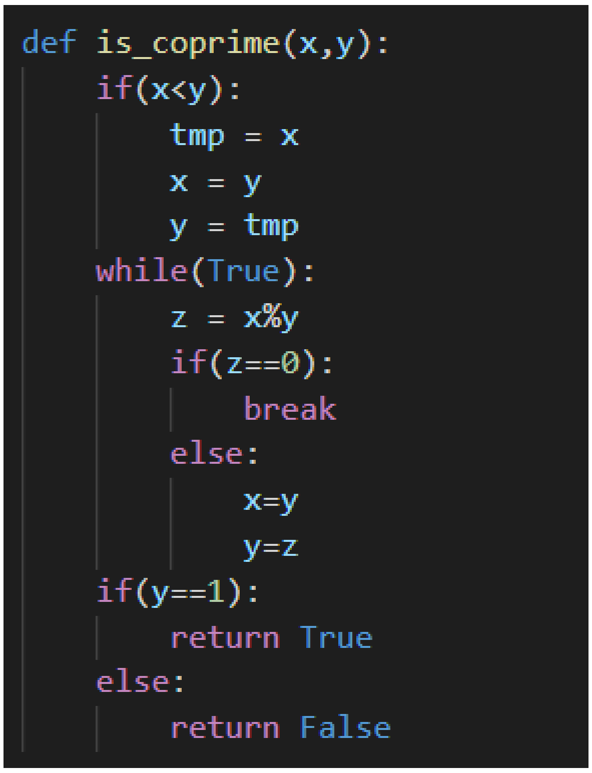
```python
def miller_rabin(n,k):
    if(n==2):
        return True

    if(n%2==0):
        return False

    r,s = 0, n-1
    while(s%2==0):
        r +=1
        s //=2
    for _ in range(k):
        a = random.randrange(2,n-1)
        x = pow(a,s,n)
        if(x==1 or x == n-1):
            continue
        for __ in range(r-1):
            x = pow(x,2,n)
            if(x==n-1):
                break
        else:
            return False

    return True
```

3. 計算 n,phi,e,d



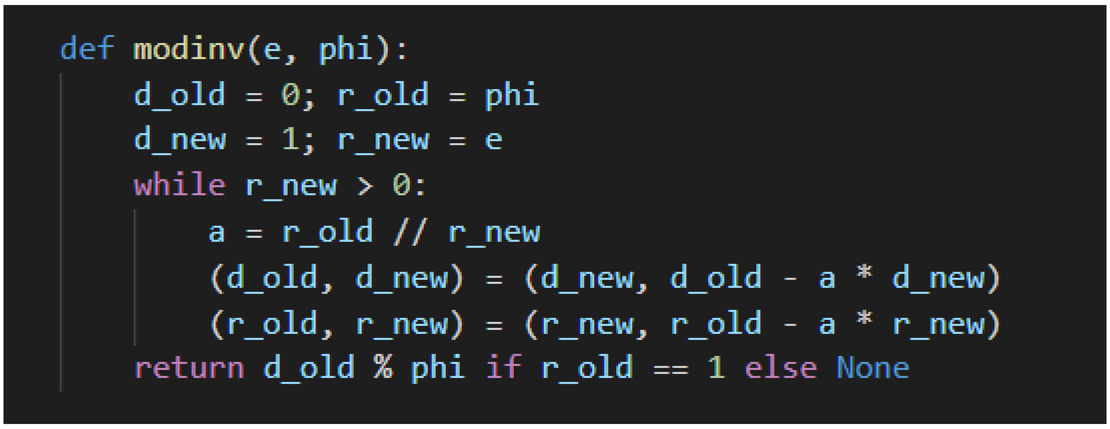
```python
n = p*q

phi = (p-1) * (q-1)
e = random.randint(2,phi)
while(is_coprime(e,phi) == False):
    e = random.randint(2,phi)

d = modinv(e,phi)
```

Is_prime 確保 e 跟 phi 互質



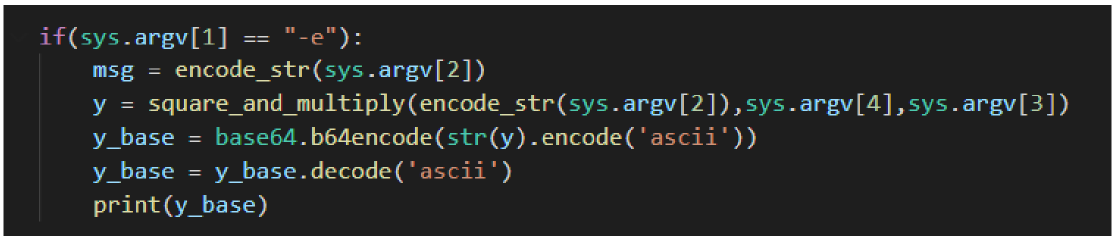
```python
def is_coprime(x,y):
    if(x<y):
        tmp = x
        x = y
        y = tmp
    while(True):
        z = x%y
        if(z==0):
            break
        else:
            x=y
            y=z
    if(y==1):
        return True
    else:
        return False
```

利用 modinv 找到 d



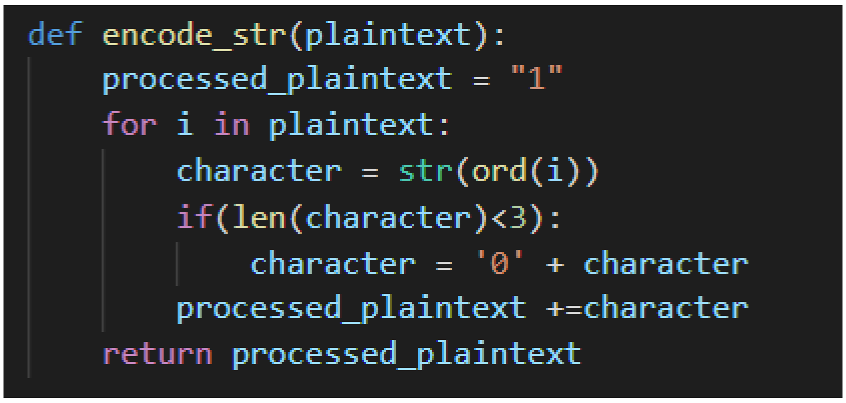
```python
def modinv(e, phi):
    d_old = 0; r_old = phi
    d_new = 1; r_new = e
    while r_new > 0:
        a = r_old // r_new
        (d_old, d_new) = (d_new, d_old - a * d_new)
        (r_old, r_new) = (r_new, r_old - a * r_new)
    return d_old % phi if r_old == 1 else None
```

4. 加密



```python
if(sys.argv[1] == "-e"):
    msg = encode_str(sys.argv[2])
    y = square_and_multiply(encode_str(sys.argv[2]),sys.argv[4],sys.argv[3])
    y_base = base64.b64encode(str(y).encode('ascii'))
    y_base = y_base.decode('ascii')
    print(y_base)
```

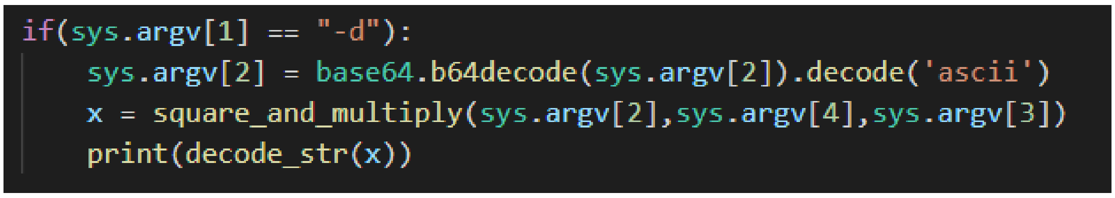
將明文編碼成數字



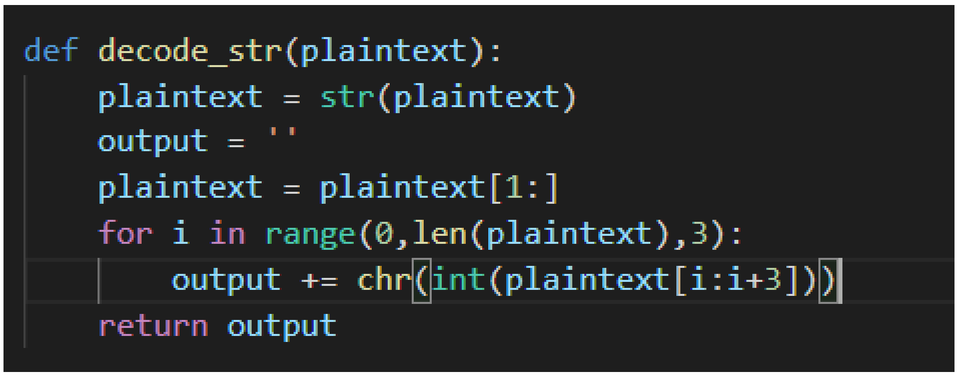
```python
def encode_str(plaintext):
    processed_plaintext = "1"
    for i in plaintext:
        character = str(ord(i))
        if(len(character)<3):
            character = '0' + character
        processed_plaintext +=character
    return processed_plaintext
```

加密完再轉成 Base64

5. 解密



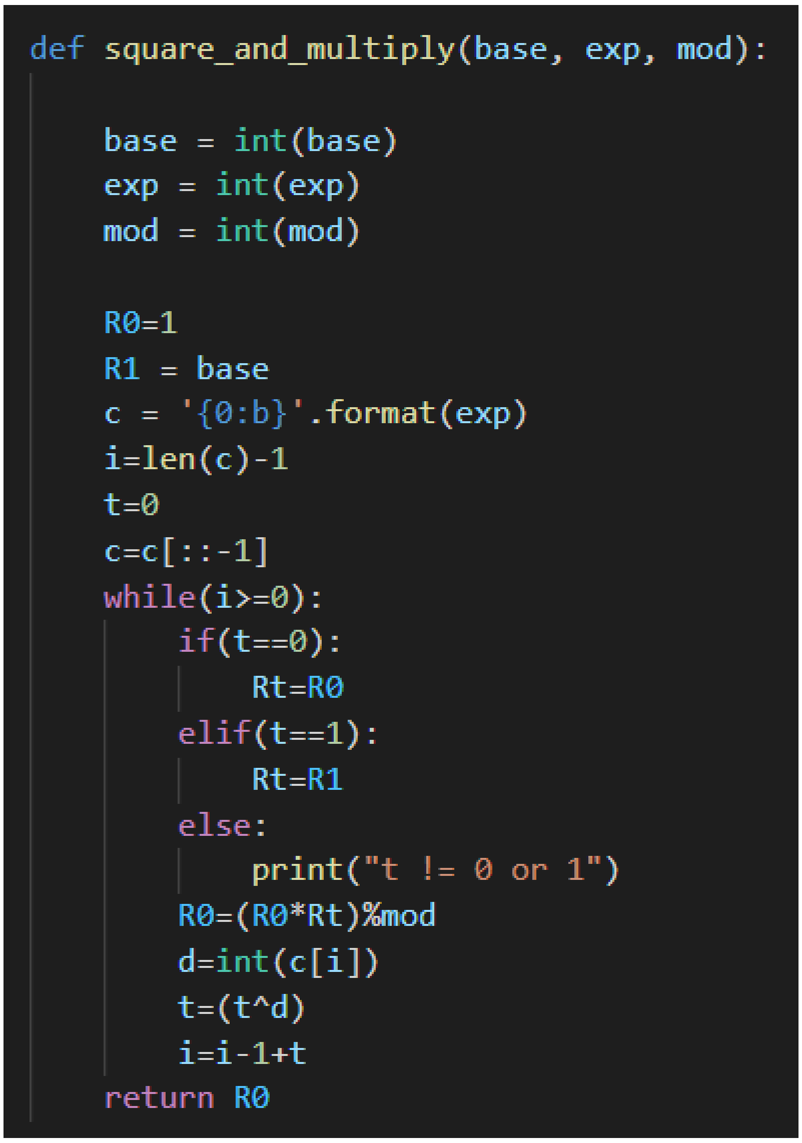
```python
if(sys.argv[1] == "-d"):
    sys.argv[2] = base64.b64decode(sys.argv[2]).decode('ascii')
    x = square_and_multiply(sys.argv[2],sys.argv[4],sys.argv[3])
    print(decode_str(x))
```



```python
def decode_str(plaintext):
    plaintext = str(plaintext)
    output = ''
    plaintext = plaintext[1:]
    for i in range(0,len(plaintext),3):
        output += chr(int(plaintext[i:i+3]))
    return output
```

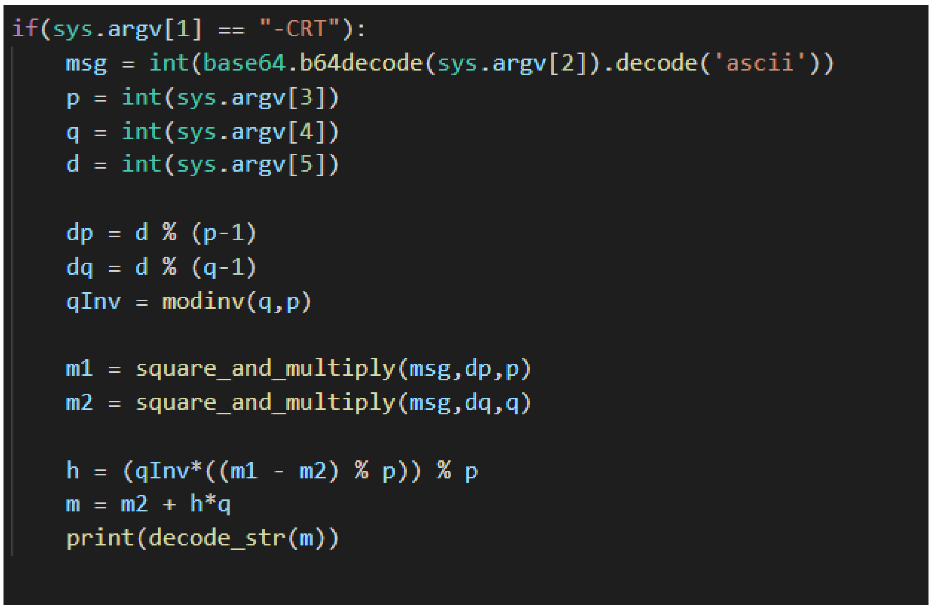
先將密文轉回數字

6. 大數的冪次方



```python
def square_and_multiply(base, exp, mod):

    base = int(base)
    exp = int(exp)
    mod = int(mod)

    R0=1
    R1 = base
    c = '{0:b}'.format(exp)
    i=len(c)-1
    t=0
    c=c[::-1]
    while(i>=0):
        if(t==0):
            Rt=R0
        elif(t==1):
            Rt=R1
        else:
            print("t != 0 or 1")
        R0=(R0*Rt)%mod
        d=int(c[i])
        t=(t^d)
        i=i-1+t
    return R0
```

7. CRT 加速



```python
if(sys.argv[1] == "-CRT"):
    msg = int(base64.b64decode(sys.argv[2]).decode('ascii'))
    p = int(sys.argv[3])
    q = int(sys.argv[4])
    d = int(sys.argv[5])

    dp = d % (p-1)
    dq = d % (q-1)
    qInv = modinv(q,p)

    m1 = square_and_multiply(msg,dp,p)
    m2 = square_and_multiply(msg,dq,q)

    h = (qInv*((m1 - m2) % p)) % p
    m = m2 + h*q
    print(decode_str(m))
```