Applied Cryptography and Network Security (CSI3002)

LAB ASSESSMENT – 4

Name	**: RITHIV.R**
Reg No	**: 19MIC0113**
Slot	**: L27+L28**

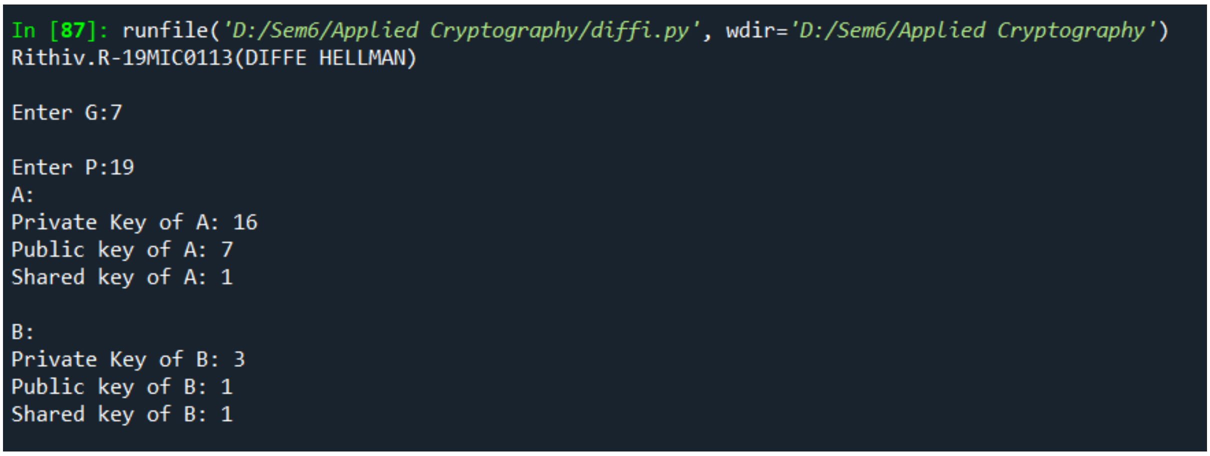
1.DIFF HELMAN KEY EXCHANGE:

Code:

```
# -*- coding: utf-8 -*-
"""
@author: Rihiv R
"""
print('Rithiv.R-19MIC0113(DIFFE HELLMAN)')

import random

def difffunc(v1,v2,v3):
    return int(pow(v1,v2))%v3



def publickey(g,p,val):
    return difffunc(g,p,val)

def sharedkey(y,x,p):
    return difffunc(y, x, p)

g = int(input('Enter G:'))
p = int(input('Enter P:'))

xa = random.randint(0,p)
xb = random.randint(0,p)

ya = publickey(g, p, xa)
yb = publickey(g, p, xb)
```

```
k1 = sharedkey(yb, xa, p)
k2 = sharedkey(ya, xb, p)


print('A:')
print('Private Key of A:',xa)
print('Public key of A:',ya)
print('Shared key of A:',k1)


print('\nB:')
print('Private Key of B:',xb)
print('Public key of B:',yb)
print('Shared key of B:',k2)
```

Output:



```
In [87]: runfile('D:/Sem6/Applied Cryptography/diffi.py', wdir='D:/Sem6/Applied Cryptography')
Rithiv.R-19MIC0113(DIFFE HELLMAN)

Enter G:7

Enter P:19
A:
Private Key of A: 16
Public key of A: 7
Shared key of A: 1

B:
Private Key of B: 3
Public key of B: 1
Shared key of B: 1
```

2.RSA:

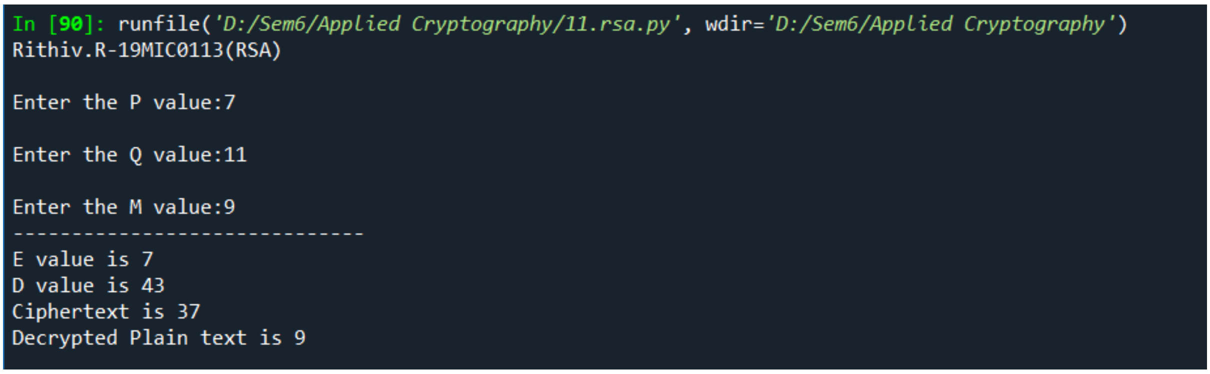
Code:

```
# -*- coding: utf-8 -*-
"""
@author: Rihiv R
"""
print('Rithiv.R-19MIC0113(RSA)')
```

```
import math


n = 0

et = 0

e = 0

d = 0

ciphertext = 0

decryptedtext = 0


def calc_e(p,q):

    global e,et,n

    n = p*q

    et = (p-1)*(q-1)

    for i in range(2,et):

        if(math.gcd(i, et)==1):

            e = i

            print('E value is',e)

            break



def calc_d():

    global e,d,et

    for i in range(et):

        if((i*e)%et==1):

            d = i

            print('D value is',d)

            break
```

```
def encryption(m):
    global e,n,ciphertext
    ciphertext = int(pow(m,e))%n
    print('Ciphertext is',ciphertext)

def decryption():
    global ciphertext,d,n,decryptedtext
    decryptedtext = int(pow(ciphertext,d))%n
    print('Decrypted Plain text is',decryptedtext)

p = int(input('Enter the P value:'))
q = int(input('Enter the Q value:'))
m = int(input('Enter the M value:'))

for i in range(30):
    print('-',end='')
print()

calc_e(p, q)
calc_d()
encryption(m)
decryption()
```

Output:



```
In [90]: runfile('D:/Sem6/Applied Cryptography/11.rsa.py', wdir='D:/Sem6/Applied Cryptography')
Rithiv.R-19MIC0113(RSA)

Enter the P value:7

Enter the Q value:11

Enter the M value:9
------------------------------
E value is 7
D value is 43
Ciphertext is 37
Decrypted Plain text is 9
```

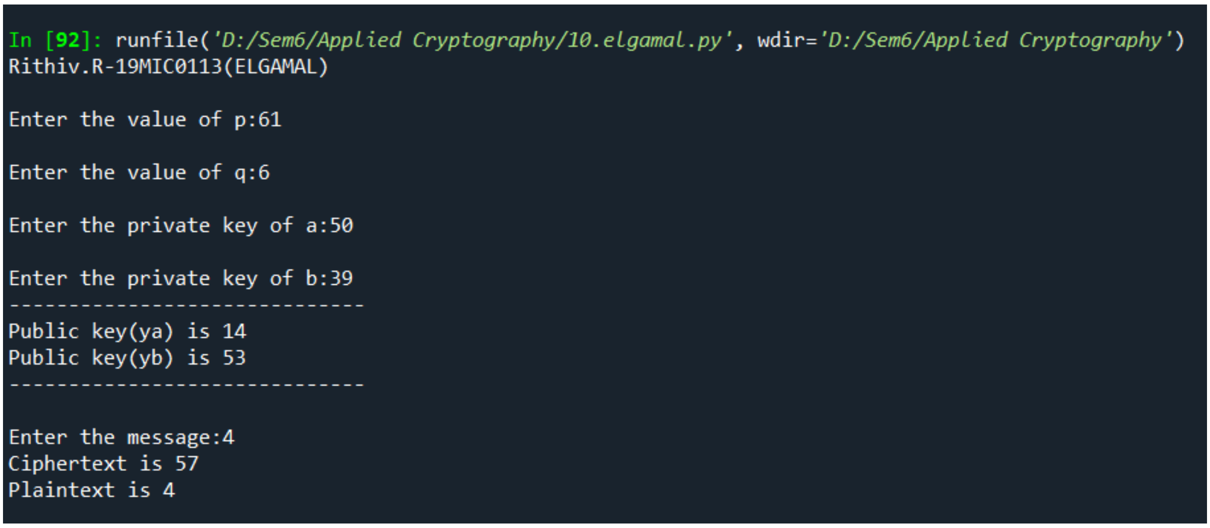
3. ELGAMAL:

Code:

```
# -*- coding: utf-8 -*-
"""
@author: Rihiv R
"""

print('Rithiv.R-19MIC0113(ELGAMAL)')
ya = 0
yb = 0
message = 0
yec = 0
dec = 0

def cal_pub_key(xa,xb,g,p):
    global ya,yb
    ya = int(pow(g,xa)%p)
    yb = int(pow(g,xb)%p)
    for i in range(30):
        print('-',end='')
    print()
    print('Public key(ya) is',ya)
    print('Public key(yb) is',yb)

def encryption(xb,p):
    global message,ya,yec
    message = int(input('Enter the message:'))
    yec = message*int(pow(ya,xb))%p
    print('Ciphertext is',int(yec))
```

```
def decryption(xa,p):
    global yb,dec
    sub = p-1-xa
    dec = (yec%p) * (int(pow(yb,sub))%p)
    if(dec>p):
        dec = dec%p
    print('Plaintext is',int(dec))


p = int(input('Enter the value of p:'))
g = int(input('Enter the value of q:'))
xa = int(input('Enter the private key of a:'))
xb = int(input('Enter the private key of b:'))
cal_pub_key(xa=xa,xb=xb,p=p,g=g)
for i in range(30):
    print('-',end='')
print()
encryption(xb=xb,p=p)
decryption(xa=xa,p=p)
```

Output:



```
In [92]: runfile('D:/Sem6/Applied Cryptography/10.elgamal.py', wdir='D:/Sem6/Applied Cryptography')
Rithiv.R-19MIC0113(ELGAMAL)

Enter the value of p:61

Enter the value of q:6

Enter the private key of a:50

Enter the private key of b:39
------------------------------
Public key(ya) is 14
Public key(yb) is 53
------------------------------

Enter the message:4
Ciphertext is 57
Plaintext is 4
```

4. AES:

Code:

```
"""

@author: Rihiv R

"""


print('RITHIV.R-19MIC0113(AES)')

keymatrix = []

words = []


sbox = {

'00':{'00':'63','01':'7c','02':'77','03':'7b','04':'f2','05':'6b','06':'6f','07':'c5','08':'30'
,'09':'01','0a':'67','0b':'2b','0c':'fe','0d':'d7','0e':'ab','0f':'76'},

'10':{'00':'ca','01':'82','02':'c9','03':'7d','04':'fa','05':'59','06':'47','07':'f0','08':'ad'
,'09':'d4','0a':'a2','0b':'af','0c':'9c','0d':'a4','0e':'72','0f':'c0'},

'20':{'00':'b7','01':'fd','02':'93','03':'26','04':'36','05':'3f','06':'f7','07':'cc','08':'34',
'09':'a5','0a':'e5','0b':'f1','0c':'71','0d':'d8','0e':'31','0f':'15'},

'30':{'00':'04','01':'c7','02':'23','03':'c3','04':'18','05':'96','06':'05','07':'9a','08':'07
','09':'12','0a':'80','0b':'e2','0c':'eb','0d':'27','0e':'b2','0f':'75'},

'40':{'00':'09','01':'83','02':'2c','03':'1a','04':'1b','05':'6e','06':'5a','07':'a0','08':'5
2','09':'3b','0a':'d6','0b':'b3','0c':'29','0d':'e3','0e':'2f','0f':'84'},

'50':{'00':'53','01':'d1','02':'00','03':'ed','04':'20','05':'fc','06':'b1','07':'5b','08':'6a
','09':'cb','0a':'be','0b':'39','0c':'4a','0d':'4c','0e':'58','0f':'cf'},
'60':{'00':'d0','01':'ef','02':'aa','03':'fb','04':'43','05':'4d','06':'33','07':'85','08':'45'
,'09':'f9','0a':'02','0b':'7f','0c':'50','0d':'3c','0e':'9f','0f':'a8'},

'70':{'00':'51','01':'a3','02':'40','03':'8f','04':'92','05':'9d','06':'38','07':'f5','08':'bc'
,'09':'b6','0a':'da','0b':'21','0c':'10','0d':'ff','0e':'f3','0f':'d2'},

'80':{'00':'cd','01':'0c','02':'13','03':'ec','04':'5f','05':'97','06':'44','07':'17','08':'c4'
,'09':'a7','0a':'7e','0b':'3d','0c':'64','0d':'5d','0e':'19','0f':'73'},

'90':{'00':'60','01':'81','02':'4f','03':'dc','04':'22','05':'2a','06':'90','07':'88','08':'46
','09':'ee','0a':'b8','0b':'14','0c':'de','0d':'5e','0e':'0b','0f':'db'},

'a0':{'00':'e0','01':'32','02':'3a','03':'0a','04':'49','05':'06','06':'24','07':'5c','08':'c2
','09':'d3','0a':'ac','0b':'62','0c':'91','0d':'95','0e':'e4','0f':'79'},

'b0':{'00':'e7','01':'c8','02':'37','03':'6d','04':'8d','05':'d5','06':'4e','07':'a9','08':'6
c','09':'56','0a':'f4','0b':'ea','0c':'65','0d':'7a','0e':'ae','0f':'08'},

'c0':{'00':'ba','01':'78','02':'25','03':'2e','04':'1c','05':'a6','06':'b4','07':'c6','08':'e8
','09':'dd','0a':'74','0b':'1f','0c':'4b','0d':'bd','0e':'8b','0f':'8a'},

'd0':{'00':'70','01':'3e','02':'b5','03':'66','04':'48','05':'03','06':'f6','07':'0e','08':'61
','09':'35','0a':'57','0b':'b9','0c':'86','0d':'c1','0e':'1d','0f':'9e'},

'e0':{'00':'e1','01':'f8','02':'98','03':'11','04':'69','05':'d9','06':'8e','07':'94','08':'9b
','09':'1e','0a':'87','0b':'e9','0c':'ce','0d':'55','0e':'28','0f':'df'},

'f0':{'00':'8c','01':'a1','02':'89','03':'0d','04':'bf','05':'e6','06':'42','07':'68','08':'41'
,'09':'99','0a':'2d','0b':'0f','0c':'b0','0d':'54','0e':'bb','0f':'16'},
   }
```

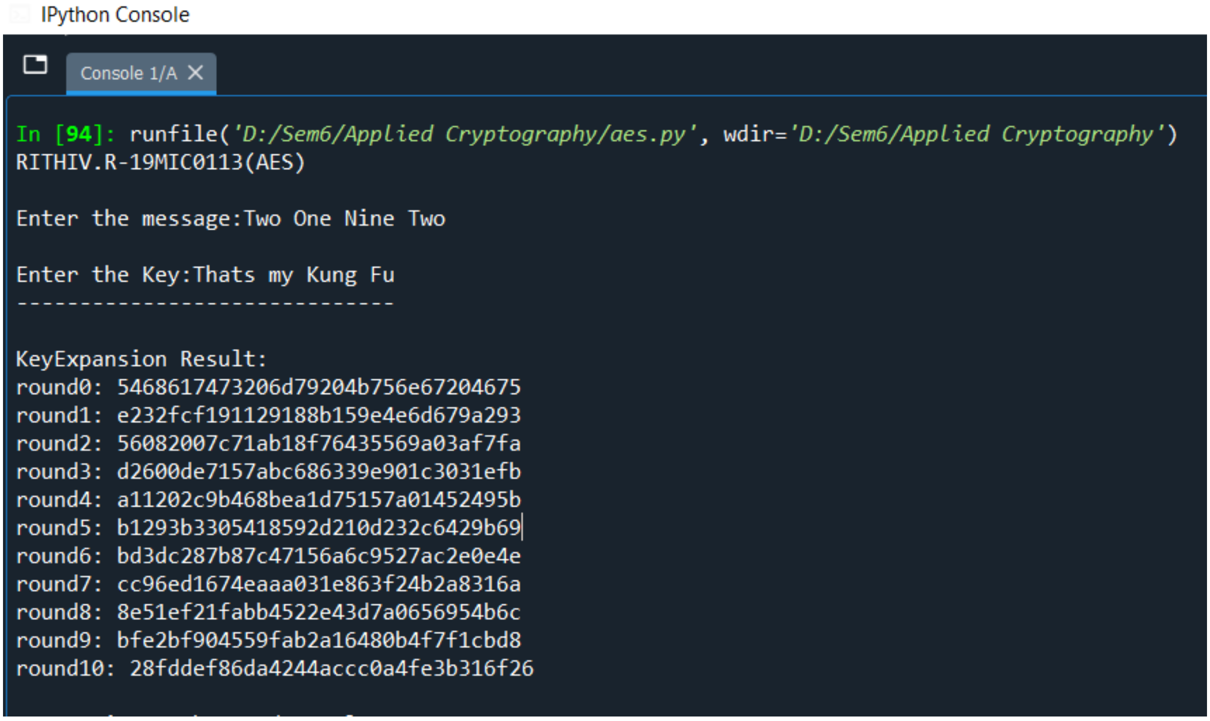
```
rconst = [['01','00','00','00'],
    ['02','00','00','00'],
    ['04','00','00','00'],
    ['08','00','00','00'],
    ['10','00','00','00'],
    ['20','00','00','00'],
    ['40','00','00','00'],
    ['80','00','00','00'],
    ['1b','00','00','00'],
    ['36','00','00','00'],
    ]

hexa = {'0':'0000','1':'0001','2':'0010','3':'0011','4':'0100','5':'0101','6':'0110','7':'0111','8':'1000','9':'1001','a':'1010','b':'1011','c':'1100','d':'1101','e':'1110','f':'1111'}


roundkeys = []

mi1 = [
     ['02','03','01','01'],
     ['01','02','03','01'],
     ['01','01','02','03'],
     ['03','01','01','02'],
    ]

IR = '100011011'
```

```
def strtohex(val):
    return val.encode('utf-8').hex()



def hextobin(val):
    return "{0:08b}".format(int(val, 16))



def xor(str1,str2):
    strtemp = ''
    for i,j in zip(str1,str2):
        if(i==j):
            strtemp = strtemp + '0'
        else:
            strtemp = strtemp + '1'
    return strtemp


def bintohexa(val):
    num = int(val, 2)
    hexnum = format(num, 'x')
    return hexnum


def rotword(arr):
    array = []
    array.extend([i for i in arr[1:]])
    array.append(arr[0])
    return array

def subword(arr):
    tempo = []
    for i in arr:
        if(len(i)==1):
            i = '0'+i
        x = i[0]+'0'
        y ='0'+i[1]
        tempo.append(sbox[x][y])
    return tempo

def sub1word(arr):
    x = arr[0]+'0'
    y = '0'+arr[1]
    return sbox[x][y]

def xorpart(arr1,arr2):
    myarray = []
    for i,j in zip(arr1,arr2):
        x1 = hextobin(i)
        x2 = hextobin(j)
        resultxor = xor(x1,x2)
        resulthex = bintohexa(resultxor)
        myarray.append(resulthex)
    return myarray

def g(w,r):
    val1 = rotword(w)
    val2 = subword(val1)
    x = xorpart(val2, r)
    return x



def keyexpansion(key):
    hexa = ''
    temp = []
    for i in key:
        value = strtohex(i)
        hexa= hexa+value
        temp.append(value)

    temp1 = []
    for i in range(4):
        myword = [j for j in temp[i*4:(i*4+4)]]
        temp1.append(myword)
        words.append(myword)

    for i in range(10):
        gc = g(words[-1],rconst[i])
        wc1 = xorpart(gc, words[-4])
        wc2 = xorpart(wc1,words[-3])
        wc3 = xorpart(wc2,words[-2])
        wc4 = xorpart(wc3,words[-1])
        words.extend([wc1,wc2,wc3,wc4])

    for j,i in enumerate(words):
        for k2,k1 in enumerate(i):
            if(len(k1)==1):
                words[j][k2]= '0'+k1

    for i in range(11):
        str1 = ''
        for j in words[i*4:i*4+4]:
            for k1 in j:
                str1 = str1 + k1
        roundkeys.append(str1)

    print('\nKeyExpansion Result:')
    for j,i in enumerate(roundkeys):
        print('round'+str(j)+':',i)




def addround(array,temp2):
    statematrix = [[0 for i in range(4)] for i in range(4)]
    for i in range(4):
        for j in range(4):
            x = bintohexa(xor(hextobin(array[i][j]),hextobin(temp2[i][j])))
            statematrix[j][i]=x.rjust(2,'0')
    return statematrix

def substitue(statematrix):
    submatrix = [[0 for i in range(4)] for i in range(4)]
    for c1,i in enumerate(statematrix):
        for c2,j in enumerate(i):
            submatrix[c2][c1]=sub1word(j)
    return submatrix

def shiftrow(submatrix):
    myarray = []
    for j,i in enumerate(submatrix):
        if(j==0):
            myarray.append(i)
        else:
            shift = []
            shift.extend(i[j:])
            shift.extend(i[:j])
            myarray.append(shift)
    return myarray

def polynomial(string,val):
    result = []
    for i,j in zip(range(8,0,-1),string):
        if(j=='1'):
            result.append(i+val)
    return result

def mixedbox(matrix):
    mymain = []
    for j1,i1 in enumerate(mi1):
        myar = []
        for j2,i2 in enumerate(i1):
            mya1 = [mi1[j1][k] for k in range(4)]
            mya2 = [matrix[k][j2] for k in range(4)]
            mainlist = []
            for k1,k2 in zip(mya1,mya2):
                if(k1=='01'):
                    mainlist.extend(polynomial(hextobin(k2),0))
                elif(k1=='02'):
                    mainlist.extend(polynomial(hextobin(k2),1))
                elif(k1=='03'):
                    mainlist.extend(polynomial(hextobin(k2),1))
                    mainlist.extend(polynomial(hextobin(k2),0))
            setter = set(mainlist)
            counter = {}
            for i in setter:
                counter[i] = mainlist.count(i)
            mul=[]
            for i,j in sorted(counter.items(),reverse=True):
                if j%2!=0:
                    mul.append(i)
            result = '00'
            if(len(mul)!=0):
                length=mul[0]
                mul_string=["0"]*(length)
                for i in range(length+1):
                    if i in mul:
                        mul_string[i-1]="1"
                mul_string.reverse()
                mul_string="".join(mul_string)
                result=division(mul_string)
            myar.append(bintohexa(result))
        mymain.append(myar)
    return mymain

def division(bits):
    while(len(bits))>8:
        irr_poly = "100011011"
        bits = bin(int("0b"+bits,2)^int("0b"+irr_poly,2))[2:]
    while(len(bits)<8):
        bits = '0'+bits
    return bits

def makearr(val):
    temp = []
    temp2 = [[0 for i in range(4)] for i in range(4)]
    roundtemp = roundkeys[val]
    for j,i in enumerate(roundkeys[val]):
        if((j+1)%2==0):
            temp.append(roundtemp[j-1:j+1])
    counter = 0
    for i in range(4):
        for j in range(4):
            temp2[j][i] = temp[counter]
            counter=counter+1
    return temp2

def encryption(message):
    print('\nEncryption Each Round Result:')
    x1 = []
    for i in message:
        x1.append(strtohex(i))
    array = [[0 for i in range(4)] for i in range(4)]
    counter=0
    for i in range(4):
        for j in range(4):
            array[j][i] = x1[counter]
            counter+=1
    statematrix = []
    temp2 = makearr(0)
```

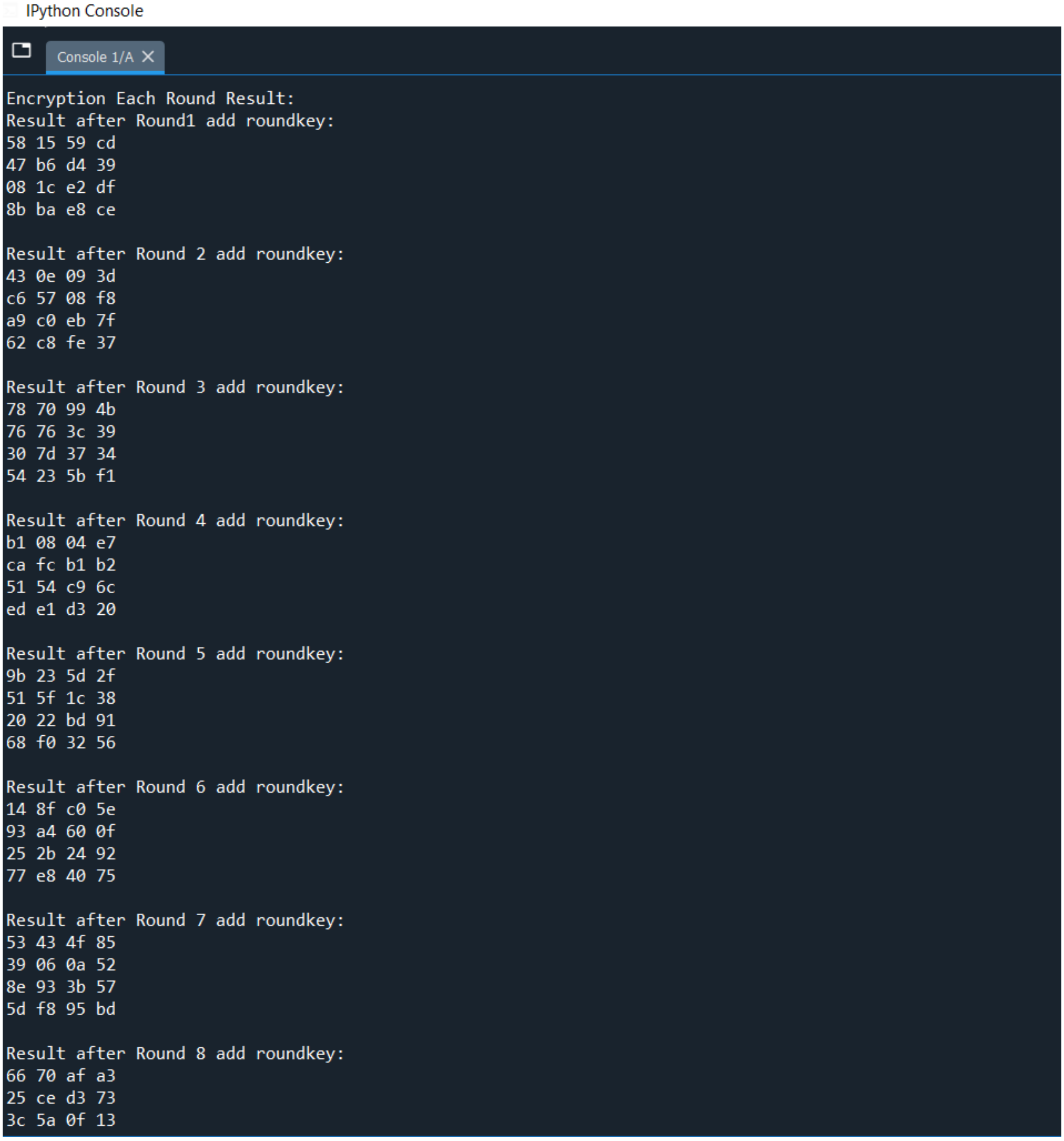
```
statematrix = addround(array, temp2)
submatrix = substitue(statematrix)
shiftmatrix = shiftrow(submatrix)
mixedb = mixedbox(shiftmatrix)
temp2 = makearr(1)
statematrix = addround(mixedb, temp2)
print('Result after Round1 add roundkey:')
for i in range(4):
   for j in range(4):
      print(statematrix[j][i],end=' ')
   print()
print()
for i in range(2,10):
   submatrix = substitue(statematrix)
   shiftmatrix = shiftrow(submatrix)
   mixedb = mixedbox(shiftmatrix)
   temp2 = makearr(i)
   statematrix = addround(mixedb, temp2)
   print('Result after Round '+str(i)+' add roundkey:')
   for i in range(4):
      for j in range(4):
         print(statematrix[j][i],end=' ')
      print()
   print()
submatrix = substitue(statematrix)
shiftmatrix = shiftrow(submatrix)
```

```
    temp2 = makearr(10)
    statematrix = addround(shiftmatrix,temp2)
    print('Result after Round10 add roundkey:')
    for i in range(4):
        for j in range(4):
            print(statematrix[j][i],end=' ')
        print()
    print()
    print('\nThe Encrypted Cipher Text by AES is:')
    for i in range(4):
        for j in range(4):
            print(statematrix[i][j],end=' ')
    print()


message = input('Enter the message:')
key = input('Enter the Key:')
for i in range(30):
    print('-',end='')
print()
keyexpansion(key)
encryption(message)
```

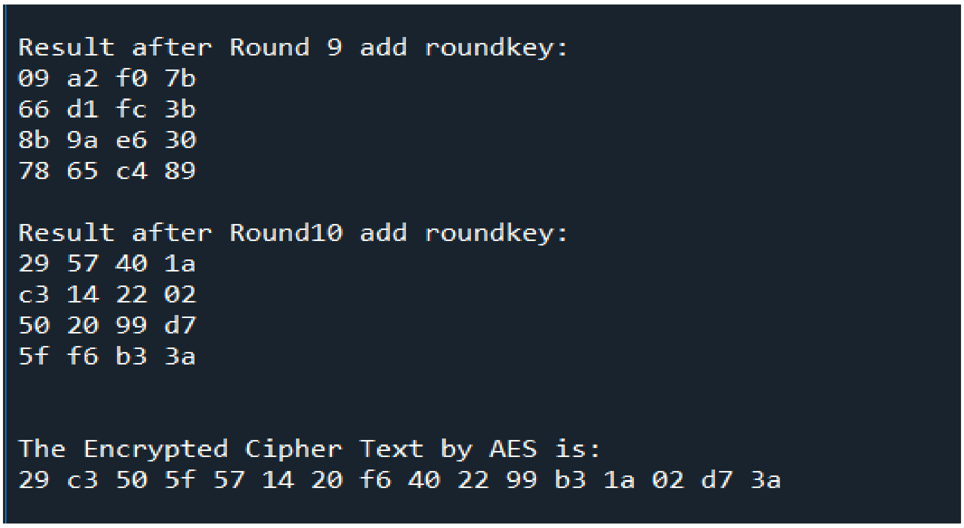
Output:



```
In [94]: runfile('D:/Sem6/Applied Cryptography/aes.py', wdir='D:/Sem6/Applied Cryptography')
RITHIV.R-19MIC0113(AES)

Enter the message:Two One Nine Two

Enter the Key:Thats my Kung Fu
------------------------------

KeyExpansion Result:
round0: 5468617473206d79204b756e67204675
round1: e232fcf191129188b159e4e6d679a293
round2: 56082007c71ab18f76435569a03af7fa
round3: d2600de7157abc686339e901c3031efb
round4: a11202c9b468bea1d75157a01452495b
round5: b1293b3305418592d210d232c6429b69
round6: bd3dc287b87c47156a6c9527ac2e0e4e
round7: cc96ed1674eaaa031e863f24b2a8316a
round8: 8e51ef21fabb4522e43d7a0656954b6c
round9: bfe2bf904559fab2a16480b4f7f1cbd8
round10: 28fddef86da4244accc0a4fe3b316f26
```



```
Encryption Each Round Result:
Result after Round1 add roundkey:
58 15 59 cd
47 b6 d4 39
08 1c e2 df
8b ba e8 ce

Result after Round 2 add roundkey:
43 0e 09 3d
c6 57 08 f8
a9 c0 eb 7f
62 c8 fe 37

Result after Round 3 add roundkey:
78 70 99 4b
76 76 3c 39
30 7d 37 34
54 23 5b f1

Result after Round 4 add roundkey:
b1 08 04 e7
ca fc b1 b2
51 54 c9 6c
ed e1 d3 20

Result after Round 5 add roundkey:
9b 23 5d 2f
51 5f 1c 38
20 22 bd 91
68 f0 32 56

Result after Round 6 add roundkey:
14 8f c0 5e
93 a4 60 0f
25 2b 24 92
77 e8 40 75

Result after Round 7 add roundkey:
53 43 4f 85
39 06 0a 52
8e 93 3b 57
5d f8 95 bd

Result after Round 8 add roundkey:
66 70 af a3
25 ce d3 73
3c 5a 0f 13
```



```
Result after Round 9 add roundkey:
09 a2 f0 7b
66 d1 fc 3b
8b 9a e6 30
78 65 c4 89

Result after Round10 add roundkey:
29 57 40 1a
c3 14 22 02
50 20 99 d7
5f f6 b3 3a


The Encrypted Cipher Text by AES is:
29 c3 50 5f 57 14 20 f6 40 22 99 b3 1a 02 d7 3a
```