**Expansion Table**

s = "11110000101010101111000010101010"

print("Input: "+s)

sbox = [32, 1, 2, 3, 4, 5,

4, 5, 6, 7, 8, 9,

12, 13, 14, 15, 16, 17,

16, 17, 18, 19, 20, 21,

20, 21, 22, 23, 24, 25,

24, 25, 26, 27, 28, 29,

28, 29, 30, 31, 32, 1

]

out = ""

for i in sbox:

out += str(s[i-1])

print()

print("Output: ")

print(out)

**s\_box**

s\_box = [

[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7],

[0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],

[4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],

[15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13]

]

def substitute(binary\_input):

"""Substitute input bits using the S-box"""

decimal\_input = int(binary\_input, 2)

input\_bits = format(decimal\_input, '06b')

row = int(input\_bits[0] + input\_bits[5], 2)

col = int(input\_bits[1:5], 2)

output\_decimal = s\_box[row][col]

return output\_decimal

binary\_input = input("Enter your digit in binary form: ")

output\_decimal = substitute(binary\_input)

print(output\_decimal)

**Shift Cipher**

plain\_text = input("What is your plaintext: ")

key = int(input("What is your shift: "))

def caesar(plain\_text, key):

cipher\_text = ""

for ch in plain\_text:

if ch.isalpha():

stay\_in\_alphabet = ord(ch) +key

if stay\_in\_alphabet > ord('z'):

stay\_in\_alphabet -= 26

final\_letter = chr(stay\_in\_alphabet)

cipher\_text += final\_letter

print("Your ciphertext is: ", cipher\_text)

return cipher\_text

caesar(plain\_text, key)

**RSA**

import random

def decrypt(cipher\_text, private\_key):

    d, n = private\_key

    plain\_text = [chr(pow(char, d, n)) for char in cipher\_text]

    return ''.join(plain\_text)

def encrypt(plain\_text, public\_key):

    e, n = public\_key

    cipher\_text = [pow(ord(char), e, n) for char in plain\_text]

    return cipher\_text

def gcd(a, b):

    if b == 0:

        return a

    else:

        return gcd(b, a % b)

def generate\_keys(p, q):

    n = p \* q

    phi = (p - 1) \* (q - 1)

    e = random.randrange(1, phi)

    while gcd(e, phi) != 1:

        e = random.randrange(1, phi)

    d = mod\_inverse(e, phi)

    return ((e, n), (d, n))

def mod\_inverse(e, phi):

    d = 0

    x1 = 0

    x2 = 1

    y1 = 1

    temp\_phi = phi

    while e > 0:

        temp1 = temp\_phi // e

        temp2 = temp\_phi - temp1 \* e

        temp\_phi = e

        e = temp2

        x = x2 - temp1 \* x1

        y = d - temp1 \* y1

        x2 = x1

        x1 = x

        d = y1

        y1 = y

    if temp\_phi == 1:

        return d + phi

p = 61

q = 53

public\_key, private\_key = generate\_keys(p, q)

message = "Hello!"

cipher\_text = encrypt(message, public\_key)

print("Cipher text:", cipher\_text)

plain\_text = decrypt(cipher\_text, private\_key)

print("Plain text:", plain\_text)

**DES**

!pip install pycryptodome

!pip install crypto

from Crypto.Cipher import DES

from secrets import token\_bytes

def get\_key():

  key = token\_bytes(8)

  print("key is (key)")

  return key

def encrypt(msg,key):

  cipher = DES.new(key,DES.MODE\_EAX)

  nonce = cipher.nonce

  cipher\_text,tag = cipher.encrypt\_and\_digest(msg.encode('ascii'))

  return cipher\_text,tag,nonce

msg = "Hello"

key = get\_key()

ct,tg,nc = encrypt(msg,key)

print(f"Plaintext: {msg} \nCipherText:{ct}")

def decrypt(ct,key,nonce,tag):

  cipher = DES.new (key,DES.MODE\_EAX,nonce=nonce)

  plain\_text = cipher.decrypt(ct)

  return plain\_text

decrypt(ct,key,nc,tg)

**Monoalphabetic cipher.**

cipher\_key = {'A':'D', 'B':'E', 'C':'F', 'D':'G', 'E':'H',

              'F':'I', 'G':'J', 'H':'K', 'I':'L', 'J':'M',

              'K':'N', 'L':'O', 'M':'P', 'N':'Q', 'O':'R',

              'P':'S', 'Q':'T', 'R':'U', 'S':'V', 'T':'W',

              'U':'X', 'V':'Y', 'W':'Z', 'X':'A', 'Y':'B', 'Z':'C'}

plain\_text = input("Enter the message to be encrypted: ")

plain\_text = plain\_text.upper()

cipher\_text = ""

for letter in plain\_text:

  if letter in cipher\_key:

    cipher\_text += cipher\_key[letter]

print("The encrypted text is: ", cipher\_text)

decipher\_key = {'D':'A', 'E':'B', 'F':'C', 'G':'D', 'H':'E',

              'I':'F', 'J':'G', 'K':'H', 'L':'I', 'M':'J',

              'N':'K', 'O':'L', 'P':'M', 'Q':'N', 'R':'O',

              'S':'P', 'T':'Q', 'U':'R', 'V':'S', 'W':'T',

              'X':'U', 'Y':'V', 'Z':'W', 'A':'X', 'B':'Y', 'C':'Z'}

cipher\_text = input("Enter the message to be decrypted: ")

plain\_text = ""

for letter in cipher\_text:

  if letter in decipher\_key:

    plain\_text += decipher\_key[letter]

print("The decrypted text is: ", plain\_text)

**Caesar Cipher.**

def encryptMe(text, key):

    l = len(text)

    tem = ""

    for i in text:

        if(ord(i)==32):

            tem = tem + " "

            continue

        #small letter asci code 97 - 122

        if(ord(i)>96):

            l = 0

            m = 0

            if(ord(i)+key> 122):

                l = 122 - ord(i)

                l = l + key - 1

                m = 97

                i = chr(l + m)

                tem = tem + i

                continue

            i = chr(ord(i)+ key + l + m)

            tem = tem + i

            #operation for Capital Lettter

        elif(ord(i)<ord('Z')+1):

            l = 0

            m = 0

            if(ord(i)+key>ord('Z')):

                l = ord('Z') - ord(i)

                l = l + key - 1

                m = ord('A')

                i = chr(l + m)

                tem = tem + i

                continue

            i = chr(ord(i)+ key + l + m)

            tem = tem + i

    return tem

def decriptMe(text, key):

    l = len(text)

    tem = ""

    for i in text:

        if(ord(i)==32):

            tem = tem + " "

            continue

        if(ord(i)>96):

            l = 0

            m = 0

            if(ord(i)-key< ord('a')):

                l =  ord(i) - ord('a')

                l = l - key - 1

                m = ord('z')

                i = chr(m-l)

                tem = tem + i

                continue

            i = chr(ord(i)- key)

            tem = tem + i

        elif(ord(i)<ord('Z')+1):

            l = 0

            m = 0

            if(ord(i)-key>ord('Z')):

                l =  ord(i) - ord('Z')

                l = l - key - 1

                m = ord('A')

                i = chr(m-l)

                tem = tem + i

                continue

            i = chr(ord(i)- key)

            tem = tem + i

    return tem

if \_\_name\_\_ == '\_\_main\_\_':

    kk = encryptMe("We Love Bangladesh", 3)

    print(kk)

    ll = decriptMe(kk, 3)

    print(ll)