21. def pad(text):

# Add PKCS#7 padding to the plaintext

padding\_length = 8 - (len(text) % 8)

padding = bytes([padding\_length] \* padding\_length)

return text + padding

def unpad(text):

# Remove PKCS#7 padding from the plaintext

padding\_length = text[-1]

return text[:-padding\_length]

def encrypt\_cbc(plaintext, key):

# Generate a random initialization vector

iv = os.urandom(8)

# Create the 3DES cipher object and initialize with the key and IV

cipher = DES3.new(key, DES3.MODE\_CBC, iv)

# Pad the plaintext and encrypt it in CBC mode using 3DES

padded\_plaintext = pad(plaintext)

ciphertext = cipher.encrypt(padded\_plaintext)

# Prepend the IV to the ciphertext

return iv + ciphertext

def decrypt\_cbc(ciphertext, key):

# Extract the IV from the ciphertext

iv = ciphertext[:8]

# Create the 3DES cipher object and initialize with the key and IV

cipher = DES3.new(key, DES3.MODE\_CBC, iv)

# Decrypt the ciphertext in CBC mode using 3DES and remove the padding

padded\_plaintext = cipher.decrypt(ciphertext[8:])

plaintext = unpad(padded\_plaintext)

return plaintext

# Define the plaintext message

plaintext = b"meet me at the usual place at ten rather than eight oclock"

# Define the initial key

key = b"\x01\x23\x45\x67\x89\xAB\xCD\xEF\xFE\xDC\xBA\x98\x76\x54\x32\x10\x01\x23\x45\x67\x89\xAB\xCD\xEF"

# Encrypt the plaintext message using CBC mode with 3DES

ciphertext = encrypt\_cbc(plaintext, key)

# Decrypt the ciphertext message using CBC mode with 3DES

decrypted\_plaintext = decrypt\_cbc(ciphertext, key)

print(f"Plaintext: {plaintext}")

print(f"Ciphertext: {ciphertext}")

print(f"Decrypted plaintext: {decrypted\_plaintext}")

22. pt=str(input("ENTER THE PLAIN TEXT : "))

cipher=""

letter="abcdefghijklmnopqrstuvwxyz"

common=max(set(pt),key=pt.count)

print("COMMON LETTER : "+common)

if common in letter:

com=letter.find(common)

key=com-6

#print("key = "+common+" - g = "+key)

if (key<0):

key=26-key

for i in pt:

if i in letter:

pos=letter.find(i)

new\_pos=(pos+key)%26

new\_char=letter[new\_pos]

cipher+=new\_char

print("CIPHER TEXT : "+cipher)

output:

ENTER THE PLAIN TEXT : cryptography

COMMON LETTER : p

CIPHER TEXT : lahycxpajyqh

23. import numpy as np

def hill\_cipher(message, key):

msg\_num = [ord(char) - 65 for char in message.upper()]

msg\_len = len(msg\_num)

while msg\_len % len(key) != 0:

msg\_num.append(23)

msg\_len += 1

msg\_mat = np.reshape(msg\_num, (-1, len(key)))

enc\_mat = np.matmul(msg\_mat, key) % 26

enc\_num = np.reshape(enc\_mat, (-1,)).tolist()

ciphertext = ''.join([chr(num + 65) for num in enc\_num])

return ciphertext