Code:

def generate\_key(msg, key):

    key = list(key)

    if len(msg) == len(key):

        return key

    else:

        for i in range(len(msg) - len(key)):

            key.append(key[i % len(key)])

    return "".join(key)

def encrypt\_vigenere(msg, key):

    encrypted\_text = []

    key = generate\_key(msg, key)

    for i in range(len(msg)):

        char = msg[i]

        if char.isupper():

            encrypted\_char = chr((ord(char) + ord(key[i]) - 2 \* ord('A')) % 26 + ord('A'))

        elif char.islower():

            encrypted\_char = chr((ord(char) + ord(key[i]) - 2 \* ord('a')) % 26 + ord('a'))

        else:

            encrypted\_char = char

        encrypted\_text.append(encrypted\_char)

    return "".join(encrypted\_text)

def decrypt\_vigenere(msg, key):

    decrypted\_text = []

    key = generate\_key(msg, key)

    for i in range(len(msg)):

        char = msg[i]

        if char.isupper():

            decrypted\_char = chr((ord(char) - ord(key[i]) + 26) % 26 + ord('A'))

        elif char.islower():

            decrypted\_char = chr((ord(char) - ord(key[i]) + 26) % 26 + ord('a'))

        else:

            decrypted\_char = char

        decrypted\_text.append(decrypted\_char)

    return "".join(decrypted\_text)

text\_to\_encrypt = "She is listening!"

key = "PASCAL"

encrypted\_text = encrypt\_vigenere(text\_to\_encrypt, key)

print(f"Encrypted Text: {encrypted\_text}")

decrypted\_text = decrypt\_vigenere(encrypted\_text, key)

print(f"Decrypted Text: {decrypted\_text}")

Output:

Encrypted Text: Hbq cx fuonjwczc!

Decrypted Text: She is listening!

Code:

def rail\_fence\_encrypt(text, rails):

rail = [['\n' for \_ in range(len(text))] for \_ in range(rails)]

direction\_down = False

row, col = 0, 0

for char in text:

if row == 0 or row == rails - 1:

direction\_down = not direction\_down

rail[row][col] = char

col += 1

row += 1 if direction\_down else -1

ciphertext = ''.join([''.join(row) for row in rail if row])

return ciphertext.replace('\n', '')

def rail\_fence\_decrypt(ciphertext, rails):

rail = [['\n' for \_ in range(len(ciphertext))] for \_ in range(rails)]

direction\_down = None

row, col = 0, 0

for i in range(len(ciphertext)):

if row == 0:

direction\_down = True

if row == rails - 1:

direction\_down = False

rail[row][col] = '\*'

col += 1

row += 1 if direction\_down else -1

index = 0

for i in range(rails):

for j in range(len(ciphertext)):

if rail[i][j] == '\*' and index < len(ciphertext):

rail[i][j] = ciphertext[index]

index += 1

result = []

row, col = 0, 0

for i in range(len(ciphertext)):

if row == 0:

direction\_down = True

if row == rails - 1:

direction\_down = False

result.append(rail[row][col])

col += 1

row += 1 if direction\_down else -1

return "".join(result)

# Fixed input

plaintext = "HELLO WORLD".replace(" ", "") # Removing spaces

rails = 3

# Encryption

ciphertext = rail\_fence\_encrypt(plaintext, rails)

print("\nEncrypted Text:", ciphertext)

# Decryption

decrypted\_text = rail\_fence\_decrypt(ciphertext, rails)

print("Decrypted Text:", decrypted\_text)

Output:

Encrypted Text: HOLELWRDLO

Decrypted Text: HELLOWORLD