program for monoalphabetic substitution cipher maps a plaintext alphabet to a ciphertext alphabet, so that each letter of the plaintext alphabet maps to a single unique letter of the ciphertext alphabet.

Program

import string

# Function to encrypt using Monoalphabetic Cipher

def monoalphabetic\_encrypt(plaintext, key\_map):

plaintext = plaintext.lower()

ciphertext = ""

for char in plaintext:

if char.isalpha():

ciphertext += key\_map[char]

else:

ciphertext += char # Keep non-alphabet characters unchanged

return ciphertext

# Function to decrypt using Monoalphabetic Cipher

def monoalphabetic\_decrypt(ciphertext, key\_map):

reverse\_map = {v: k for k, v in key\_map.items()}

ciphertext = ciphertext.lower()

plaintext = ""

for char in ciphertext:

if char.isalpha():

plaintext += reverse\_map[char]

else:

plaintext += char

return plaintext

# Main program

plaintext\_alphabet = list(string.ascii\_lowercase)

ciphertext\_alphabet = list("QWERTYUIOPASDFGHJKLZXCVBNM".lower()) # Example substitution key

# Create mapping dictionary

key\_map = dict(zip(plaintext\_alphabet, ciphertext\_alphabet))

# Input plaintext

plaintext = input("Enter the plaintext: ")

# Encrypt and decrypt

encrypted\_text = monoalphabetic\_encrypt(plaintext, key\_map)

decrypted\_text = monoalphabetic\_decrypt(encrypted\_text, key\_map)

print("\nPlaintext Alphabet : ", ''.join(plaintext\_alphabet))

print("Ciphertext Alphabet: ", ''.join(ciphertext\_alphabet))

print("\nOriginal Text : ", plaintext)

print("Encrypted Text : ", encrypted\_text)

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

output:

