­­6.Affine Caesar

Program:

def gcd(a, b):

while b:

a, b = b, a % b

return a

def mod\_inverse(a, m):

for i in range(1, m):

if (a \* i) % m == 1:

return i

return None

def affine\_caesar\_encrypt(text, a, b):

encrypted\_text = ""

for char in text:

if char.isalpha():

shift = ord('A') if char.isupper() else ord('a')

encrypted\_char = chr((a \* (ord(char) - shift) + b) % 26 + shift)

encrypted\_text += encrypted\_char

else:

encrypted\_text += char

return encrypted\_text

def affine\_caesar\_decrypt(ciphertext, a, b):

mod\_inv = mod\_inverse(a, 26)

if mod\_inv is None:

return "Error: 'a' value is not valid (no modular inverse exists)"

decrypted\_text = ""

for char in ciphertext:

if char.isalpha():

shift = ord('A') if char.isupper() else ord('a')

decrypted\_char = chr((mod\_inv \* ((ord(char) - shift) - b)) % 26 + shift)

decrypted\_text += decrypted\_char

else:

decrypted\_text += char

return decrypted\_text

# Main program

plaintext = input("Enter the text:")

a = int(input("Enter the A value:"))

b = int(input("Enter the B value:"))

encrypted\_text = affine\_caesar\_encrypt(plaintext, a, b)

print("Encrypted:", encrypted\_text)

decrypted\_text = affine\_caesar\_decrypt(encrypted\_text, a, b)

print("Decrypted:", decrypted\_text)

Output:

