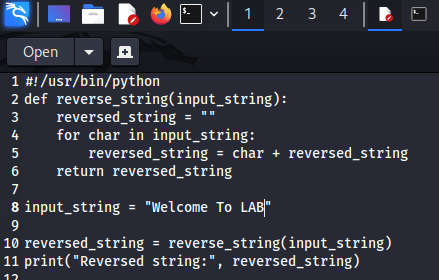
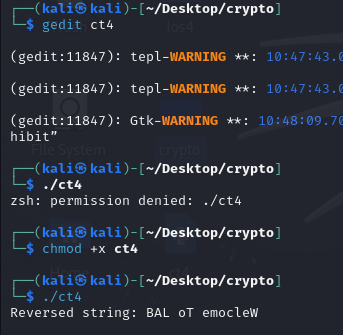
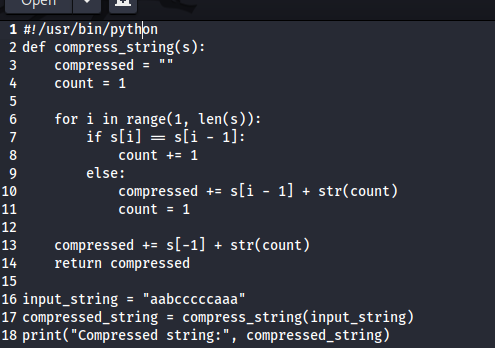
**1. Write a Python program to reverse the content of the string.**

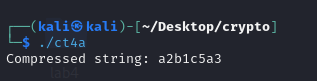
**Do not use built in**

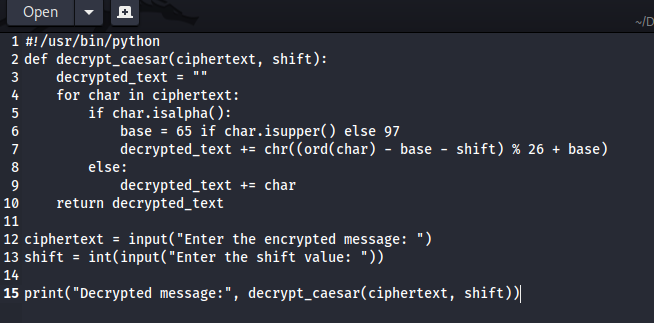
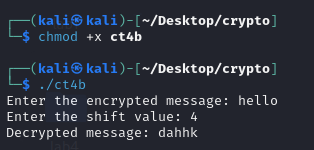




**2. Create a program that performs basic string compression using the counts of repeated characters. For example, the string “aabcccccaaa” would become “a2b1c5a3”.**

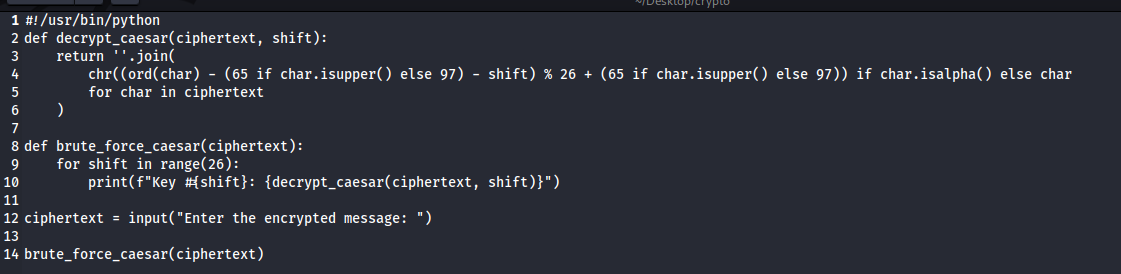


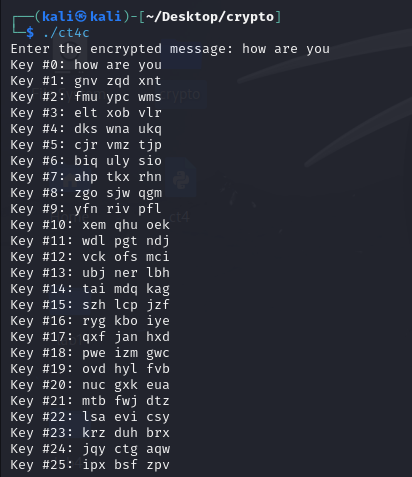


**3. Get the Caesar cipher from the user Decrypt the cipher**

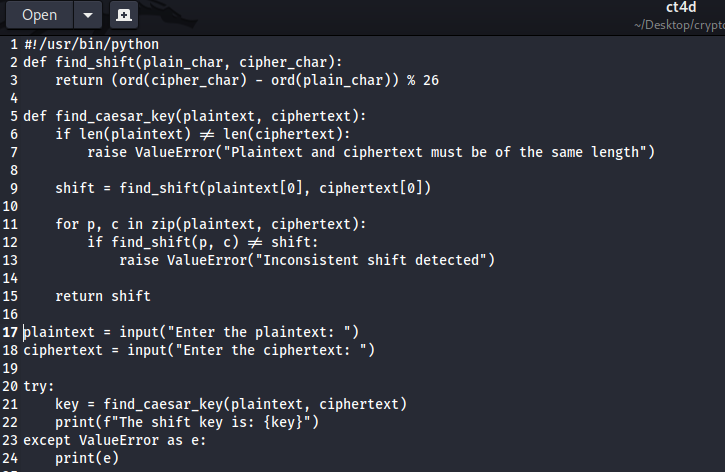
**4. Get the cipher encrypted using shift cipher. Identify the key used to encrypt using brute force**

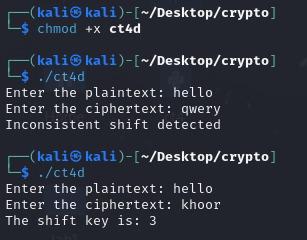
**ie all the values in the key space**



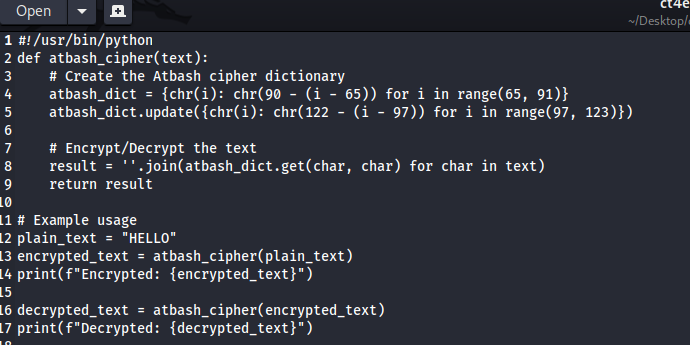


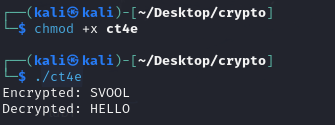
**5. Find the k value , Provided cipher text and plain text**





**6. Encrypt and decrypt the string using Atbash cipher**





**7. Encrypt and decrypt using Affine cipher**

**add validation**

#!/usr/bin/python

def gcd(a, b):

while b != 0:

a, b = b, a % b

return a

def mod\_inverse(a, m):

for x in range(1, m):

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

return x

return None

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

if gcd(a, 26) != 1:

raise ValueError("a and 26 must be coprime.")

result = ""

for char in text:

if char.isalpha():

offset = 65 if char.isupper() else 97

result += chr(((a \* (ord(char) - offset) + b) % 26) + offset)

else:

result += char

return result

def affine\_decrypt(cipher, a, b):

if gcd(a, 26) != 1:

raise ValueError("a and 26 must be coprime.")

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

if a\_inv is None:

raise ValueError("Multiplicative inverse for a does not exist.")

result = ""

for char in cipher:

if char.isalpha():

offset = 65 if char.isupper() else 97

result += chr(((a\_inv \* ((ord(char) - offset) - b)) % 26) + offset)

else:

result += char

return result

# Example usage

a = 5

b = 8

plain\_text = "HELLO WORLD"

encrypted\_text = affine\_encrypt(plain\_text, a, b)

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

decrypted\_text = affine\_decrypt(encrypted**\_**text, a, b)

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