**CNS Lab**

**Assignment 1**

**Title: To study Shift cipher / Caesar cipher**

**Information:**

1. `encrypt` and `decrypt` Functions:

- The `encrypt` function takes two parameters: `text` (the plaintext to be encrypted) and `shift` (the number of positions each letter should be shifted in the alphabet).

- The `decrypt` function takes two parameters: `encrypted\_text` (the ciphertext to be decrypted) and `shift` (the same shift value used for encryption).

2. Character Processing:

- Both the `encrypt` and `decrypt` functions iterate through each character in the input text (`text` or `encrypted\_text`) and process them one by one.

- They check if the character is an alphabetic character using `char.isalpha()`. If it is not alphabetic (e.g., a digit, space, punctuation), it is left unchanged and appended to the result.

3. Shifting Letters:

- If the character is alphabetic, the code checks whether it is lowercase or uppercase.

- For lowercase letters, it calculates the new character by subtracting the ASCII value of 'a' (or 'A') from the character, applying the shift value, taking the modulus 26 to wrap around the alphabet, and then adding back the ASCII value of 'a' (or 'A'). This effectively shifts the character within the alphabet.

4. Encryption:

- The `encrypt` function uses a positive shift value to encrypt the text. It shifts each letter forward in the alphabet by the specified number of positions.

5. Decryption:

- The `decrypt` function uses a negative shift value to decrypt the text. It shifts each letter backward in the alphabet by the specified number of positions to recover the original plaintext.

6. User Input and Output:

- The code takes user input for the text to be encrypted and the shift value.

- It then calls the `encrypt` function to encrypt the input text and displays the encrypted result.

- After that, it calls the `decrypt` function with the same shift value to decrypt the encrypted text and displays the decrypted result.

**Code:**

def encrypt(text, shift):

    encrypted\_text = ""

    for char in text:

        if char.isalpha():

            if char.islower():

                encrypted\_text += chr((ord(char) - ord('a' ) + shift) % 26 + ord('a'))

            elif char.isupper():

                encrypted\_text += chr((ord(char) - ord('A' ) + shift) % 26 + ord('A'))

        else:

            encrypted\_text += char

    return encrypted\_text

def decrypt(encrypted\_text, shift):

    decrypted\_text = ""

    for char in encrypted\_text:

        if char.isalpha():

            if char.islower():

                decrypted\_text += chr((ord(char) - ord('a' ) - shift) % 26 + ord('a'))

            elif char.isupper():

                decrypted\_text += chr((ord(char) - ord('A' ) - shift) % 26 + ord('A'))

        else:

            decrypted\_text += char

    return decrypted\_text

# Input text and shift value

text = input("Enter the text: ")

shift = int(input("Enter the shift value: "))

# Encrypt the text

encrypted\_text = encrypt(text, shift)

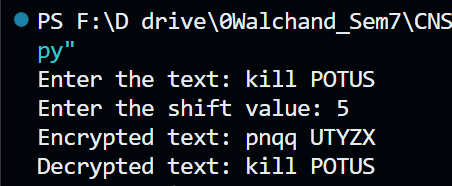
print("Encrypted text:", encrypted\_text)

# Decrypt the encrypted text

decrypted\_text = decrypt(encrypted\_text, shift)

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

**Output:**

****

**Limitations**:

1. Lack of Security: Shift ciphers provide very little security, as there are only 25 possible keys (assuming a standard 26-letter English alphabet). An attacker can easily perform a brute-force attack by trying all possible keys.

2. Vulnerable to Frequency Analysis: Shift ciphers do not hide the frequency distribution of letters in the plaintext. This makes them vulnerable to frequency analysis attacks, where an attacker can analyze the frequency of letters in the ciphertext and make educated guesses about the key.

3. Vulnerable to Known-Plaintext Attacks: If an attacker has access to both the plaintext and ciphertext, they can determine the key used for encryption by comparing the two. This makes shift ciphers vulnerable to known-plaintext attacks.

4. Lack of Key Management: Shift ciphers require a secret key to be effective. If the key is compromised or leaked, all encrypted messages become vulnerable. Key management can be challenging in practice, especially when dealing with multiple parties.

5. Limited Key Space: The key space for shift ciphers is small, making them easily susceptible to brute-force attacks. A determined attacker can quickly try all possible keys to decrypt the message.

6. Ineffective Against Modern Cryptanalysis: Modern cryptographic techniques and computational power can easily break shift ciphers. Cryptanalysis tools and techniques have advanced significantly since the time of Julius Caesar, rendering these ciphers obsolete for secure communications.

7. No Authentication or Integrity Protection: Shift ciphers only provide confidentiality; they do not offer any form of authentication or data integrity protection. This means an attacker can tamper with the ciphertext without detection.

8. Language and Alphabet Dependency: Shift ciphers are designed for a specific alphabet (e.g., English) and may not work well for other languages or character sets with different alphabets and character frequencies.

9. Not Suitable for Large Data: Shift ciphers are inefficient for encrypting large volumes of data because they encrypt one character at a time, leading to a high computational overhead.

10. Prone to Human Error: Shift ciphers rely on users to choose and manage encryption keys correctly. Human errors in key selection or handling can compromise the security of the encryption.