Comsats University Islamabad,
Attock Campus

Name:
Muhammad Usama

Registration No:
FA24-BSE-062

Course:
Information Security (Lab)

Assignment No:
01

Caesar Cipher Python Implementation

Complete Python Code:

```
# Caesar Cipher Program
# This program encrypts and decrypts text using the Caesar Cipher technique.

def caesar_encrypt(text, shift):
    """Encrypts the given text using Caesar Cipher."""
    encrypted_text = ""  # Store encrypted result

    for char in text:
        # Check if character is uppercase letter if
        char.isupper():
            # Shift character within A-Z range
            encrypted_text += chr((ord(char) - ord('A') + shift) % 26 + ord('A'))

        # Check if character is lowercase letter
        elif char.islower():
            # Shift character within a-z range
            encrypted_text += chr((ord(char) - ord('a') + shift) % 26 + ord('a'))

        else:
            # Keep spaces and special characters unchanged
            encrypted_text += char

    return encrypted_text


def caesar_decrypt(ciphertext, shift):
    """Decrypts the given ciphertext using Caesar Cipher."""
    # Decryption is just encryption with negative shift return
    caesar_encrypt(ciphertext, -shift)


# Main Program
Execution if _name__== '.
main_':
    message = input("Enter your message: ")
    shift_value = int(input("Enter shift value: "))

    encrypted = caesar_encrypt(message, shift_value)
    print("Encrypted Message:", encrypted)

    decrypted = caesar_decrypt(encrypted, shift_value)
    print("Decrypted Message:", decrypted)
```

Line-by-Line Explanation:

1. `# Caesar Cipher Program`
   → Comment describing the purpose of the program.
2. `# This program encrypts and decrypts text...`
   → Explains that the program performs encryption and decryption.

3. `def caesar_encrypt(text, shift):`
   → Defines a function that takes a message (`text`) and a shift value.
4. `encrypted_text = ""`
   → Initializes an empty string to store the encrypted result.
5. `for char in text:`
   → Loops through each character in the input message.
6. `if char.isupper():`
   → Checks if the character is an uppercase letter (A–Z).
7. `ord(char)`
   → Converts the character into its ASCII number.
8. `ord('A')`
   → Gets ASCII value of capital A.
9. `(ord(char) - ord('A'))`
   → Converts letter to a position between 0–25.
10. `+ shift`
    → Adds the shift value to move the letter forward.
11. `% 26`
    → Ensures wrapping around alphabet (Z → A).
12. `+ ord('A')`
    → Converts back to uppercase ASCII range.
13. `chr(...)`
    → Converts ASCII value back to character.
14. `encrypted_text += ...`
    → Appends encrypted letter to result.
15. `elif char.islower():`
    → Checks if character is lowercase (a–z).
16. `ord('a')` logic
    → Same shifting logic applied to lowercase letters.
17. `else:`
    → Handles non-letter characters.
18. `encrypted_text += char`
    → Keeps spaces, numbers, and special characters unchanged.
19. `return encrypted_text`
    → Returns the final encrypted message.

20. `def caesar_decrypt(ciphertext, shift):`
    → Defines decryption function.
21. `return caesar_encrypt(ciphertext, -shift)`
    → Calls encryption function with negative shift to reverse encryption.

22. `if __name__ == "__main__":`
    → Ensures program runs only when executed directly.
23. `message = input(...)`
    → Takes user input for message.
24. `shift_value = int(input(...))`
    → Takes shift value and converts it to integer.
25. `encrypted = caesar_encrypt(...)`
    → Calls encryption function.

26. `print("Encrypted Message:", encrypted)`
    → Displays encrypted text.
27. `decrypted = caesar_decrypt(...)`
    → Calls decryption function.
28. `print("Decrypted Message:", decrypted)`
    → Displays decrypted (original) text.

Security Analysis

The Caesar Cipher is a substitution cipher where each letter is shifted by a fixed number. However, it is NOT secure for modern cryptographic use.

Weaknesses:

1) Only 25 possible keys (brute force easy).

   Example:
   If shift = 3, attacker just tests 1–25 and finds readable text.

2) Vulnerable to frequency analysis attacks.
3) No protection against modern cryptanalysis tools.

4) Not safe for:

- Password protection
- Banking data
- Confidential communication
- Secure messaging

Only suitable for:

- Educational purposes
- Understanding basic encryption concepts