

COMSATS UNIVERSITY ISLAMABAD
ATTOCK CAMPUS



INFORMATION SECURITY LAB

ASSIGNMENT NO: 1

SUBMITTED BY:

AROOJ NOREEN

FA24-BSE-046

SUBMITTED TO:

MS AMBREEN GUL

DEADLINE:

February 28, 2026

Table of Contents:

INTRODUCTION.....	2
PROGRAM CODE:	2
CODE EXPLANATION:.....	4
SECURITY ANALYSIS:.....	6
CODE SCREENSHOTS:	6
OUTPUT:	7

INTRODUCTION

The **Caesar Cipher** is one of the earliest and simplest encryption techniques in classical cryptography. It is a type of substitution cipher in which each letter of the plaintext is shifted forward or backward by a fixed number of positions in the alphabet.

For example, if the shift value is 3:

A → D
B → E
C → F

The shift value acts as the key for both encryption and decryption. Only alphabetic characters are shifted, while spaces, numbers, and special symbols remain unchanged.

Although the Caesar Cipher is useful for understanding basic encryption concepts, it is not secure for modern communication because it has only 26 possible keys and can easily be broken using brute force or frequency analysis.

PROGRAM CODE:

```
# Caesar Cipher Implementation
# Lab Assignment 1 - Information Security
```

```
def caesar_encrypt(text, shift):
    encrypted_text = ""

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

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

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

    return encrypted_text

def caesar_decrypt(ciphertext, shift):
    decrypted_text = ""

    for char in ciphertext:
        # Check if character is uppercase letter
        if char.isupper():
            decrypted_char = chr((ord(char) - ord('A') - shift) % 26 + ord('A'))
            decrypted_text += decrypted_char

        # Check if character is lowercase letter
        elif char.islower():
            decrypted_char = chr((ord(char) - ord('a') - shift) % 26 + ord('a'))
            decrypted_text += decrypted_char

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

    return decrypted_text

# Main Program
```

```
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)
```

CODE EXPLANATION:

1. Function Definition

```
def caesar_encrypt(text, shift):
```

This line defines the encryption function that takes the original message (text) and the shift value (shift) as input parameters.

2. Initialize Result Variable

```
encrypted_text = ""
```

An empty string is created to store the encrypted output.

3. Loop Through Characters

```
for char in text:
```

This loop processes each character of the input message individually.

4. Check Uppercase Letters

```
if char.isupper():
```

This condition checks whether the character is an uppercase letter.

5. Apply Shift to Uppercase

```
encrypted_char = chr((ord(char) - ord('A') + shift) % 26 + ord('A'))
```

This line shifts the uppercase character using ASCII values and modular arithmetic to ensure it remains within the alphabet range.

6. Check Lowercase Letters

```
elif char.islower():
```

This condition verifies if the character is lowercase.

7. Apply Shift to Lowercase

```
encrypted_char = chr((ord(char) - ord('a') + shift) % 26 + ord('a'))
```

This performs shifting for lowercase letters while preserving alphabet boundaries.

8. Preserve Special Characters

```
else:
```

```
    encrypted_text += char
```

Non-alphabetic characters such as spaces and symbols remain unchanged.

9. Return Encrypted Text

```
return encrypted_text
```

Returns the final encrypted message.

10. Decryption Function

The decryption function follows the same logic as encryption, except that the shift value is subtracted instead of added. This restores the original message.

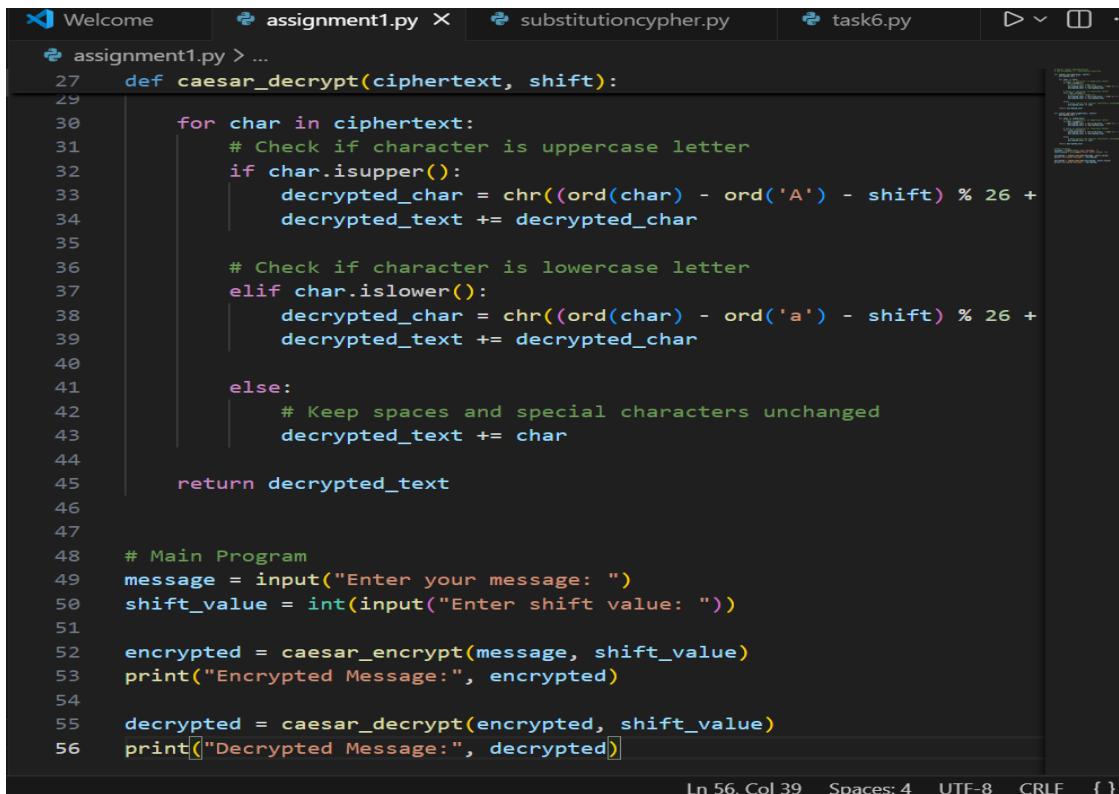
SECURITY ANALYSIS:

The Caesar Cipher provides basic encryption by shifting characters using a fixed key. However, the algorithm is vulnerable to brute force and frequency analysis attacks because it has only 26 possible keys. Therefore, it is not suitable for modern secure communication systems.

CODE SCREENSHOTS:

A screenshot of a code editor window titled "assignment1.py". The code implements a Caesar Cipher with functions for encryption and decryption. The editor interface includes tabs for "Welcome", "assignment1.py", "substitutioncypher.py", "task6.py", and a search bar at the top. The status bar at the bottom shows file statistics: Ln 56, Col 39, Spaces: 4, UTF-8, CRLF, and Python.

```
1 # Caesar Cipher Implementation
2 # Lab Assignment 1 - Information Security
3
4 def caesar_encrypt(text, shift):
5     encrypted_text = ""
6
7     for char in text:
8         # Check if character is uppercase letter
9         if char.isupper():
10             # Shift within range A-Z
11             encrypted_char = chr((ord(char) - ord('A') + shift) % 26 +
12             encrypted_text += encrypted_char
13
14         # Check if character is lowercase letter
15         elif char.islower():
16             # Shift within range a-z
17             encrypted_char = chr((ord(char) - ord('a') + shift) % 26 +
18             encrypted_text += encrypted_char
19
20         else:
21             # Keep spaces and special characters unchanged
22             encrypted_text += char
23
24     return encrypted_text
25
26
27 def caesar_decrypt(ciphertext, shift):
28     decrypted_text = ""
```



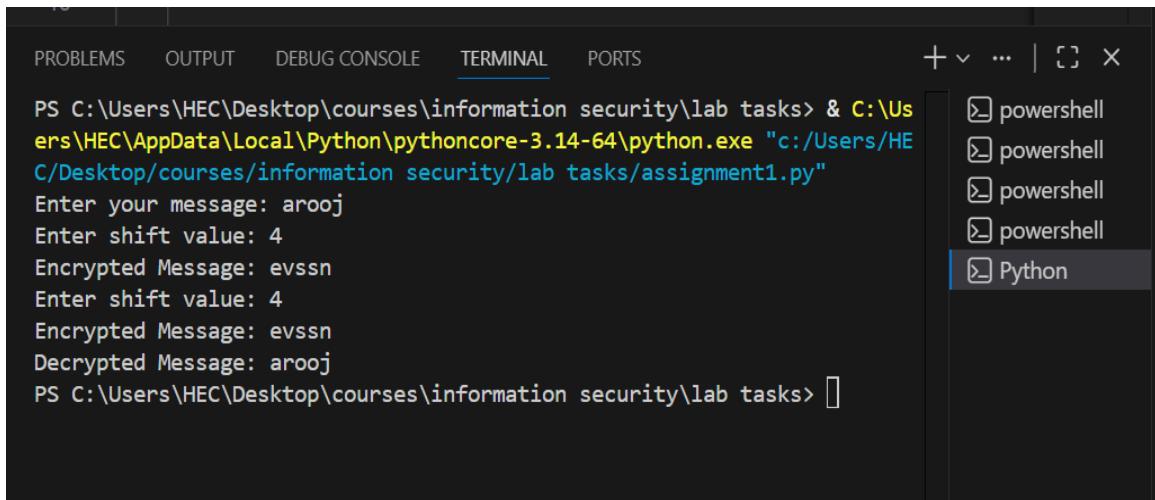
```

1 Welcome  2 assignment1.py X  3 substitutioncypher.py  4 task6.py  5 D v  6 
7 assignment1.py > ...
8
9     def caesar_decrypt(ciphertext, shift):
10
11         for char in ciphertext:
12             # Check if character is uppercase letter
13             if char.isupper():
14                 decrypted_char = chr((ord(char) - ord('A') - shift) % 26 +
15                                         decrypted_text += decrypted_char
16
17             # Check if character is lowercase letter
18             elif char.islower():
19                 decrypted_char = chr((ord(char) - ord('a') - shift) % 26 +
20                                         decrypted_text += decrypted_char
21
22             else:
23                 # Keep spaces and special characters unchanged
24                 decrypted_text += char
25
26
27     return decrypted_text
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48     # Main Program
49     message = input("Enter your message: ")
50     shift_value = int(input("Enter shift value: "))
51
52     encrypted = caesar_encrypt(message, shift_value)
53     print("Encrypted Message:", encrypted)
54
55     decrypted = caesar_decrypt(encrypted, shift_value)
56     print("Decrypted Message:", decrypted)

```

Ln 56, Col 39 Spaces: 4 UTF-8 CRLF { }

OUTPUT:



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... | [] X

```

PS C:\Users\HEC\Desktop\courses\information security\lab tasks> & C:\Users\HEC\AppData\Local\Python\pythoncore-3.14-64\python.exe "c:/Users/HEC/Desktop/courses/information security/lab tasks/assignment1.py"
Enter your message: arooj
Enter shift value: 4
Encrypted Message: evssn
Enter shift value: 4
Encrypted Message: evssn
Decrypted Message: arooj
PS C:\Users\HEC\Desktop\courses\information security\lab tasks>

```