Implementing Caesar Cipher in Python

Objective:

Students will learn how to implement a basic Caesar Cipher encryption and decryption algorithm in Python. The objective is to gain an understanding of string manipulation, loops, and how encryption algorithms work at a fundamental level.

Background:

A Caesar Cipher is one of the simplest encryption techniques. It works by shifting the letters of the plaintext by a fixed number down the alphabet. For example, with a shift of 3:

- Plaintext: HELLO
- Ciphertext: KHOOR

Each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. The same process is reversed for decryption.

Task 1: Encrypt a Message

Write a Python function “caesar_encrypt(plainText, shiftKey)” that takes a string “plainText” and an integer “shiftKey” as input and returns the encrypted message.

Task 2: Decrypt a Message

Write a Python function “caesar_decrypt(cipherText, shiftKey)” that takes a string “cipherText” and an integer “shiftKey” as input and returns the decrypted message.

Task 3: Handling Edge Cases

Ensure that your implementation handles the following cases:

1. Non-Alphabetic Characters: The cipher should only shift alphabetic characters (both uppercase and lowercase), and other characters (like spaces, punctuation, numbers) should remain unchanged.

2. Negative Shift Values: Your program should correctly handle negative shift values, which will shift the characters in the opposite direction.

3. Large Shift Values: The program should handle shift values larger than 26, ensuring that the shift wraps around the alphabet. For example, a shift of 27 should be equivalent to a shift of 1.

Task 4: Encryption and Decryption with Custom Alphabet

Modify your functions to allow for encryption and decryption using a custom alphabet. For example, instead of using the standard alphabet "ABCDEFGHIJKLMNOPQRSTUVWXYZ", you could use "ZYXWVUTSRQPONMLKJIHGFEDCBA".

Task 5: Cracking the Caesar Cipher

Write a function “caesar_crack(cipherText)” that tries to crack the Caesar cipher without knowing the shift value. This function should return all ***feasible*** decrypted messages.

Task 6: Submit and Document Your Code

- Write clear and concise documentation explaining how your code works, including comments within your code to explain your logic.

- Create a README file that explains how to run your code and provides an overview of your approach.

Extra Credit:

- Implement a user interface (command line or graphical) that allows users to encrypt, decrypt, and crack messages using your Caesar cipher functions.

- Explore and implement the **Vigenère cipher**, which is a more complex cipher based on the Caesar cipher.

Submission:

- Upload your Python code files and README to your GitHub repository.

- Submit the link to your repository.