**Summary of goal to be achieved with program:**

Anton Kusik and I worked together during paired programming to work on getting a program up and running that would allow a user to encrypt or decrypt a phrase or word based on a certain shift variable, using a Caesar cipher. Caesar ciphers will often shift letters of the alphabet certain amounts forward or back in order to send gibberish messages. These messages can be decrypted however by knowing the correct number of letters to shift the message back by.

**References:**

<https://chatgpt.com/share/68efc05e-cbd4-8005-b57c-6a8faeec0d6a>

This is a quick explanation of the basics of Caesar ciphers and some programming aid rendered by ChatGPT.

**A brief description of general assumptions to help solve the problem, as well as key observations from the references:**

A few general assumptions are that lists will be involved to store the basic alphabet layout with punctuations, and to store the user’s input. The program will have to have two different setups, one for encrypting, and the other for decrypting. These setups will have to have the user specify the shift for ease of access. Based on the references, we will also need to implement some sort of “mod” function to allow the lists to loop if the shift is too large.

**Your work showing your path from the question to your solution:**

# Caesar Cipher Implementation

# Shift each letter by a user chosen amount in the alphabet

# Programmer: Shon Hakanson

# Mentor: Anton Kusik

alphabet\_list = alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',

                            'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',

                            " ", ".", ",", "?"]

encrypt\_input\_word = input("Enter a word (no spaces): ")

encrypt\_input\_word = encrypt\_input\_word.lower()

encrypt\_input\_list = list(encrypt\_input\_word)

print("1. Encrypt Word")

print("2. Decrypt Word")

choice = input("Choose an option (1 or 2): ")

# ---- NEW: Require Shift Amount ----

shift = int(input("Enter a shift amount (e.g., 1-26): "))

# ---------- ENCRYPTION ----------

if choice == "1":

    for i in range(len(encrypt\_input\_list)):

        character = encrypt\_input\_list[i]

        if character in alphabet\_list:

            position = alphabet\_list.index(character)

            new\_position = (position + shift) % len(alphabet\_list)  # Uses user shift

            encrypt\_input\_list[i] = alphabet\_list[new\_position]

        else:

            print(f"Character {character} not in alphabet list.")

    print("".join(encrypt\_input\_list))

# ---------- DECRYPTION ----------

elif choice == "2":

    for i in range(len(encrypt\_input\_list)):

        character = encrypt\_input\_list[i]

        if character in alphabet\_list:

            original\_position = alphabet\_list.index(character)

            shifted\_position = (original\_position - shift) % len(alphabet\_list)  # Uses user shift

            encrypt\_input\_list[i] = alphabet\_list[shifted\_position]

        else:

            print(f"Character {character} not in alphabet list.")

    print("".join(encrypt\_input\_list))

**A justification of your answer that explains why you believe this is the most correct answer:**

I believe this is the most correct answer because it solves the problem fairly quickly, while keeping the code manageable and separated. It doesn’t go too complicated with numerical comparisons and such, but just checks for matching characters between lists, and shifts if it finds them. There are most certainly other ways of accomplishing this, but I think the beauty of this version is the simplicity. The program encrypts and decrypts according to user input, keeping the practicality and usability at the forefront.

**Push your code to the SWOSU shared repository and post your link:**

[SwosuCsPythonExamples/Discrete\_Structures/Student\_Work/ShonH\_Disc\_Struct\_Code/Ch4 at main · Goose4244/SwosuCsPythonExamples](https://github.com/Goose4244/SwosuCsPythonExamples/tree/main/Discrete_Structures/Student_Work/ShonH_Disc_Struct_Code/Ch4)