Caesar cipher challenge

# Code snippets .

Creates an empty dictionary and adds the key–value pairing A : C to it

| 1  2  3 | caesar = {}  caesar["A"] = "C" |
| --- | --- |

Creates a dictionary that holds alphabet letters with the corresponding encrypted pair

| 1  2  3  4  5  6  7 | caesar = {"A": "C",  "B": "D",  "C": "E"}    caesar["D"] = "F" # adds the key-value pairing to the dictionary  print(caesar) |
| --- | --- |

Converts an ASCII value to its equivalent character

| 1 | chr(66) # this will return the character B |
| --- | --- |

Iterates through a string

| 1  2  3 | word = "pizza"  for letter in word:  print(letter) |
| --- | --- |

Concatenates a string with another string value

| 1 | letter = "a"  anotherletter = "b"  letter = letter + anotherletter # letter is assigned ab |
| --- | --- |

# Scenario . Caesar cipher encryption program

A program needs to be created that allows the user to enter a piece of plain text that will be encrypted using a user-defined key.

**The program should:**

* Prompt for the encryption key.
* Prompt for the plain text to be encrypted.
* Store the plain text in upper case.
* Encrypt the plain text message.
* Display the encrypted text for the user.

The encryption will be based on a Caesar cipher. Each letter entered as plain text will be shifted forwards using the number given as the encryption key.

**Example program input and output:**

| What is the encryption key?  3  Enter your text to encrypt  Hello world  KHOOR ZRUOG  >>> |
| --- |

In this example, the encryption key entered is 3. Each letter from the plain text Hello world is shifted over by three letters of the alphabet.

**The program should work in the following way:**

1. Prompt the user to enter an encryption\_key between 1 and 25.
2. Use a function to populate a dictionary that will form the Caesar cipher. The function should:
   1. Accept the encryption\_key as an argument.
   2. Create a dictionary called caesar.
   3. Add the space (“ “) character to the dictionary.
   4. For each letter of the alphabet, it should create a dictionary pair that contains the plain text letter as the key and the encrypted letter as the key-value pair.
   5. If the encrypted letter is higher than Z, then it should go back 26 letters to mimic a Caesar’s wheel.
   6. Return the caesar dictionary to the main program.
3. Prompt the user to enter their plaintext message.
4. Ensure the message is stored in upper case.
5. Generate the ciphertext by taking each letter in plaintext and generating a new encrypted letter based on the cipher dictionary.
6. Display the ciphertext to the user.

# Task . Create your program

* Using the guidance in the scenario, create your Caesar cipher encryption program.
* Use the code snippets on page one and any other programs that you have created to support you with this activity.
* Remember to test often and break the problem down into smaller steps if they become too challenging. For example, instead of encrypting the entire message, see if you can encrypt a single character.

**Note:** You completed a similar challenge in Lesson 27 – String handling II, which might provide some support for this challenge.

# Task . Test your program

* Test all aspects of your program by entering boundary, normal, and erroneous data.
* Add any appropriate data validation checks to your program.

# Explorer tasks . Improve your encryption program

Improve your encryption program using this list:

* Provide the user with a menu system.
* Provide an additional option to decrypt the program.

Code:

|  |
| --- |