8 Caesar Cipher

Jacques Mock Schindler

10.09.2025

In cryptography, encryption is the process of transforming information in a way that, ideally, only authorised parties can decode. This process converts the original representation of the information, known as plaintext, into an alternative form known as ciphertext. Despite its goal, encryption does not itself prevent interference but denies the intelligible content to a would-be interceptor.
From Wikipedia

Ressourcen

cipher.txt — verschlüsselter Text

input.txt — Klartext-Eingabe

caesar_sandbox.pdf — PDF (Sandbox)

caesar_chiffre.ipynb — Jupyter Notebook

The first example of a cipher we will look at is the Caesar cipher.

The Caesar cipher is a simple substitution encryption technique in which each letter of the text to be encrypted is replaced by a letter a fixed number of positions away in the alphabet. For example, using a right letter shift of four, A would be replaced by E, and the word CIPHER would become GMTLIV. The technique is named after Julius Caesar, who used it in his letters. The simplicity of the Caesar cipher makes it a popular source for recreational cryptograms.
From Encyclopedia Britannica

Python Implementation

To implement the Caesar cipher in Python, we build on the string methods offered by Python. The ord function returns the Unicode code point for a given character, and the chr function returns the character that corresponds to a given Unicode code point.

```
1  \CommentTok{\# What is the Unicode code point of the character
   \textquotesingle{}A\textquotesingle{}?}
```

Simple (naïve) Implementation

First, we define a function that encrypts a given plaintext by shifting each letter by a specified number of positions in the alphabet.

```
1  \NormalTok{plain }\OperatorTok{=} \StringTok{"CIPHER"}
2  \NormalTok{shift }\OperatorTok{=} \DecValTok{4}
```

```
1  \KeywordTok{def}\NormalTok{ caesar\_encode(plain, shift):}
2  \NormalTok{    cipher }\OperatorTok{=} \StringTok{""}
3
4      \ControlFlowTok{for}\NormalTok{ char }\KeywordTok{in}\NormalTok{ plain:}
5  \NormalTok{        shifted }\OperatorTok{=}
   \BuiltInTok{ord}\NormalTok{(char) }\OperatorTok{+}\NormalTok{ shift}
6  \NormalTok{        cipher }\OperatorTok{+=}
   \BuiltInTok{chr}\NormalTok{(shifted)}
7
8      \ControlFlowTok{return}\NormalTok{ cipher}
```

Enhanced Implementation

But what happens if we reach the end of the alphabet? For example, if we shift Z by 4 positions, we would go past Z. To handle this, we can use the modulo operator % to wrap around the alphabet. The modulo operator gives the remainder of a division operation, which allows us to "wrap around" when we exceed the length of the alphabet. In the case of the Caesar cipher, we can use it to ensure that our shifted positions stay within the bounds of the alphabet.

Therefore, we will use modulo 26 (the number of letters in the English alphabet) to ensure that our shifted positions wrap around correctly. For example, if we shift Z by 4 positions, we would end up at D. This wrapping behaviour is essential for the Caesar cipher to function correctly.
The calculation for the new position of a letter can be expressed as:

$$x' = (x + n) \mod 26$$

or, in another notation:

$$x' = x \oplus_{26} n$$

Let's implement this in Python:

```
1  \KeywordTok{def}\NormalTok{ caesar\_encrypt\_mod(plain, shift):}
2  \NormalTok{    cipher }\OperatorTok{=} \StringTok{""}
3
4      \ControlFlowTok{for}\NormalTok{ char }\KeywordTok{in}\NormalTok{ plain:}
5  \NormalTok{        shifted }\OperatorTok{=}\NormalTok{
   (}\BuiltInTok{ord}\NormalTok{(char) }\OperatorTok{{-}} \BuiltInTok{ord}\Nor⌋
   malTok{(}\StringTok{\textquotesingle{}A\textquotesingle{}}\NormalTok{)
   }\OperatorTok{+}\NormalTok{ shift) }\OperatorTok{\%} \DecValTok{26}
   \OperatorTok{+} \BuiltInTok{ord}\NormalTok{(}\StringTok{\textquotesingle{}A⌋
   \textquotesingle{}}\NormalTok{)}
6  \NormalTok{        cipher }\OperatorTok{+=}
   \BuiltInTok{chr}\NormalTok{(shifted)}
7
8      \ControlFlowTok{return}\NormalTok{ cipher}
```

Because

```
1  \BuiltInTok{ord}\NormalTok{(}\StringTok{\textquotesingle{}A\textquotesingle⌋
   {}}\NormalTok{)}
```

returns 65, we need to subtract 65 from the result of `ord('A')` to get 0. This gives us the 0-based index of the letter in the alphabet, which is useful for our calculations.

Hence the calculation shown above.

For the decryption, we can simply subtract the shift value instead of adding it:

```
1  \KeywordTok{def}\NormalTok{ caesar\_decrypt\_mod(plain, shift):}
2  \NormalTok{    cipher }\OperatorTok{=} \StringTok{""}
3
4      \ControlFlowTok{for}\NormalTok{ char }\KeywordTok{in}\NormalTok{ plain:}
5  \NormalTok{        shifted }\OperatorTok{=}\NormalTok{
   (}\BuiltInTok{ord}\NormalTok{(char) }\OperatorTok{{-}} \BuiltInTok{ord}\Nor⌋
   malTok{(}\StringTok{\textquotesingle{}A\textquotesingle{}}\NormalTok{)
   }\OperatorTok{{-}}\NormalTok{ shift) }\OperatorTok{\%} \DecValTok{26}
   \OperatorTok{+} \BuiltInTok{ord}\NormalTok{(}\StringTok{\textquotesingle{}A⌋
   \textquotesingle{}}\NormalTok{)}
6  \NormalTok{        cipher }\OperatorTok{+=}
   \BuiltInTok{chr}\NormalTok{(shifted)}
7
8      \ControlFlowTok{return}\NormalTok{ cipher}
```

For convenience, we implement one function for both encryption and decryption.

```
1  \KeywordTok{def}\NormalTok{ caesar(text : }\BuiltInTok{str}\NormalTok{,
   shift : }\BuiltInTok{int}\NormalTok{,
   encrypt}\OperatorTok{=}\VariableTok{True}\NormalTok{)
   }\OperatorTok{{-}\textgreater{}} \BuiltInTok{str}\NormalTok{:}
2  \NormalTok{    text }\OperatorTok{=}\NormalTok{ text.upper()}
3  \NormalTok{    result }\OperatorTok{=} \StringTok{""}
4
5      \ControlFlowTok{if}\NormalTok{ encrypt:}
6          \ControlFlowTok{for}\NormalTok{ char }\KeywordTok{in}\NormalTok{
   text:}
7  \NormalTok{            shifted }\OperatorTok{=}\NormalTok{
   (}\BuiltInTok{ord}\NormalTok{(char) }\OperatorTok{{-}} \BuiltInTok{ord}\Nor
   malTok{(}\StringTok{\textquotesingle{}A\textquotesingle{}}\NormalTok{)
   }\OperatorTok{+}\NormalTok{ shift) }\OperatorTok{\%} \DecValTok{26}
   \OperatorTok{+} \BuiltInTok{ord}\NormalTok{(}\StringTok{\textquotesingle{}A
   \textquotesingle{}}\NormalTok{)}
8  \NormalTok{            result }\OperatorTok{+=}
   \BuiltInTok{chr}\NormalTok{(shifted)}
9      \ControlFlowTok{else}\NormalTok{:}
10         \ControlFlowTok{for}\NormalTok{ char }\KeywordTok{in}\NormalTok{
   text:}
11 \NormalTok{            shifted }\OperatorTok{=}\NormalTok{
   (}\BuiltInTok{ord}\NormalTok{(char) }\OperatorTok{{-}} \BuiltInTok{ord}\Nor
   malTok{(}\StringTok{\textquotesingle{}A\textquotesingle{}}\NormalTok{)
   }\OperatorTok{{-}}\NormalTok{ shift) }\OperatorTok{\%} \DecValTok{26}
   \OperatorTok{+} \BuiltInTok{ord}\NormalTok{(}\StringTok{\textquotesingle{}A
   \textquotesingle{}}\NormalTok{)}
12 \NormalTok{            result }\OperatorTok{+=}
   \BuiltInTok{chr}\NormalTok{(shifted)}
13
14     \ControlFlowTok{return}\NormalTok{ result}
```

Breaking Caesar Cipher

There are two main methods to break a Caesar cipher: 1. Brute Force Attack: Try all possible shifts (1-25) and see which one produces a meaningful result. 2. Frequency Analysis: Analyse the frequency of letters in the ciphertext and compare it to the expected frequency of letters in the language.

Brute Force Attack

```
1  \KeywordTok{def}\NormalTok{ caesar\_bruteforce(ciphertext:
   }\BuiltInTok{str}\NormalTok{) }\OperatorTok{{-}\textgreater{}}
   \VariableTok{None}\NormalTok{:}
2      \ControlFlowTok{for}\NormalTok{ shift }\KeywordTok{in}
   \BuiltInTok{range}\NormalTok{(}\DecValTok{1}\NormalTok{,
   }\DecValTok{26}\NormalTok{):}
```

```
3  \NormalTok{        decrypted\_text }\OperatorTok{=}\NormalTok{
   caesar(ciphertext, shift,
   encrypt}\OperatorTok{=}\VariableTok{False}\NormalTok{)}
4        \BuiltInTok{print}\NormalTok{(}\SpecialStringTok{f"Shift
   }\SpecialCharTok{\{}\NormalTok{shift}\SpecialCharTok{\}}\SpecialStringTok{:
   }\SpecialCharTok{\{}\NormalTok{decrypted\_text}\SpecialCharTok{\}}\SpecialS⌟
   tringTok{"}\NormalTok{)}
```

Frequency Analysis

In the English language, certain letters appear more frequently than others. For example, the letter 'E' is the most common letter in English text, followed by 'T', 'A', 'O', 'I', and 'N'. By analysing the frequency of letters in the ciphertext, we can make educated guesses about which letters correspond to which in the plaintext.

For a more accurate analysis, see the following frequency table:

Letter	Frequency
E	12.02%
T	9.10%
A	8.12%
O	7.68%
I	7.31%
N	6.95%
S	6.28%
R	6.02%
H	5.92%
D	4.32%
L	4.03%
C	2.78%
U	2.76%
M	2.41%
W	2.36%
F	2.23%
G	2.02%
Y	1.97%
P	1.93%
B	1.49%
V	0.98%
K	0.77%
J	0.15%
X	0.15%
Q	0.10%
Z	0.07%

Therefore, we need a function that counts the frequency of each letter in the ciphertext. The result should be a dictionary where the keys are the letters and the values are the counts of each letter. Or even better, the frequencies in percent.

To keep things simple, we assume that the input text only contains uppercase letters from A to Z and no spaces or punctuation.

```
1  \KeywordTok{def}\NormalTok{ letter\_frequency(text:
   }\BuiltInTok{str}\NormalTok{) }\OperatorTok{{-}\textgreater{}}
   \BuiltInTok{dict}\NormalTok{:}
2  \NormalTok{    frequency }\OperatorTok{=}\NormalTok{ \{\}}
3  \NormalTok{    total\_letters }\OperatorTok{=} \DecValTok{0}
4
5      \ControlFlowTok{for}\NormalTok{ char }\KeywordTok{in}\NormalTok{ text:}
6          \ControlFlowTok{if}\NormalTok{ char }\KeywordTok{not}
   \KeywordTok{in}\NormalTok{ frequency:}
7  \NormalTok{            frequency[char] }\OperatorTok{=} \DecValTok{1}
8          \ControlFlowTok{else}\NormalTok{:}
9  \NormalTok{            frequency[char] }\OperatorTok{+=} \DecValTok{1}
10 \NormalTok{        total\_letters }\OperatorTok{+=} \DecValTok{1}
11
12     \ControlFlowTok{for}\NormalTok{ key, value }\KeywordTok{in}\NormalTok{
   frequency.items():}
13 \NormalTok{        frequency[key] }\OperatorTok{=}\NormalTok{ (value
   }\OperatorTok{/}\NormalTok{ total\_letters) }\OperatorTok{*} \DecValTok{100}
14
15
16     \ControlFlowTok{return}\NormalTok{ frequency}
```

After finding the most frequent letter in the ciphertext, we can assume that it corresponds to the letter 'E' in the plaintext. By calculating the shift needed to convert the most frequent letter to 'E', we can then decrypt the entire ciphertext using that shift value.

```
1  \KeywordTok{def}\NormalTok{ find\_shift(char: }\BuiltInTok{str}\NormalTok{,
   e }\OperatorTok{=}
   \StringTok{\textquotesingle{}E\textquotesingle{}}\NormalTok{)
   }\OperatorTok{{-}\textgreater{}} \BuiltInTok{int}\NormalTok{:}
2      \ControlFlowTok{return}\NormalTok{ (}\BuiltInTok{ord}\NormalTok{(char)
   }\OperatorTok{{-}} \BuiltInTok{ord}\NormalTok{(e)) }\OperatorTok{\%}
   \DecValTok{26}
```

After guessing the shift, we can use our existing caesar function to decrypt the ciphertext. We can call the function with the encrypted text and the guessed shift value to obtain the original plaintext.

If needed, the implementation of the find_shift function can exchange the default value for the most frequent letter to another letter.