Caesar cipher (part 1)

Overview

In this assignment, we will be creating a Flask application that allows users to encode and decode messages wth a Caesar cipher. A Caesar cipher is a simple form of encoding which *"shifts"* the letters in a message by a predetermined number of spaces (for example, minus 3 spaces) up or down the alphabet. Users are able to return the message back to its original form by shifting the letters the same number of spaces in the opposite direction (for example, plus 3 spaces).

For example, the letter *"m"* becomes *"j"* when shifted minus 3 spaces. When we look at the alphabet this shift becomes a little easier to understand:

$$a\ b\ c\ d\ e\ f\ g\ h\ i\ \boxed{j}\ k\ l\ \boxed{m}\ n\ o\ p\ q\ r\ s\ t\ u\ v\ w\ x\ y\ z$$

Let's apply this to a longer message: *"The quick fox"*. We'll continue to shift the message by minus 3 spaces, although we can choose to shift by different number of spaces. Shifted by minus 3 spaces, we get the following letters:

"T" − 3 ⇒ "Q"	"h" − 3 ⇒ "e"	"e" − 3 ⇒ "b"
"q" − 3 ⇒ "n"	"u" − 3 ⇒ "r"	"i" − 3 ⇒ "f"
"c" − 3 ⇒ "z"	"k" − 3 ⇒ "h"	"f" − 3 ⇒ "c"
"o" − 3 ⇒ "l"	"x" − 3 ⇒ "u"	

Once encoded, our original message of *"The quick fox"* becomes *"Qeb nrfzh clu"*. Since we shifted the message by minus 3 spaces to encode it, we'll need to shift it by positive 3 spaces to decode it:

"Q" + 3 ⇒ "T"	"e" + 3 ⇒ "h"	"b" + 3 ⇒ "e"
"n" + 3 ⇒ "q"	"r" + 3 ⇒ "u"	"f" + 3 ⇒ "i"
"z" + 3 ⇒ "c"	"h" + 3 ⇒ "k"	"c" + 3 ⇒ "f"
"l" + 3 ⇒ "o"	"u" + 3 ⇒ "x"	

It's important to note that the case of a letter (uppercase, or lowercase) should be respected, meaning that uppercase *"T"* becomes uppercase *"Q"*, and lowercase *"t"* becomes lowercase *"q"*. Caesar cipher only alters letters; numbers and other characters are not changed. For example, if our original message is *"The #1 fox"*, the encoded message would be *"Qeb #1 clu"*.

The *"shifting"* of the letters can be done by taking a letter's ASCII code (which is a number) and either adding or subtracting the number of spaces we want to shift by. Let's look at the ASCII code of the letters of the alphabet (both uppercase and lowercase letters, since they have different ASCII codes):

A	65	B	66	C	67	D	68	E	69	F	70
G	71	H	72	I	73	J	74	K	75	L	76
M	77	N	78	O	79	P	80	Q	81	R	82
S	83	T	84	U	85	V	86	W	87	X	88
Y	89	Z	90								

a	97	b	98	c	99	d	100	e	101	f	102
g	103	h	104	i	105	j	106	k	107	l	108
m	109	n	110	o	111	p	112	q	113	r	114
s	115	t	116	u	117	v	118	w	119	x	120
y	121	z	122								

Now let's convert the *"The quick fox"* message once more, but this time we'll convert each letter to its corresponding ASCII code, and then shift the letter by subtracting the number of spaces we want to shift by. This leaves us with a new ASCII code that corresponds to the encoded letter:

$$
\begin{aligned}
&\text{"}T\text{"} - 3 \Rightarrow 84 - 3 \Rightarrow 81 \Rightarrow \text{"}Q\text{"} \\
&\text{"}h\text{"} - 3 \Rightarrow 104 - 3 \Rightarrow 101 \Rightarrow \text{"}e\text{"} \\
&\text{"}e\text{"} - 3 \Rightarrow 101 - 3 \Rightarrow 98 \Rightarrow \text{"}b\text{"} \\
&\text{"}q\text{"} - 3 \Rightarrow 113 - 3 \Rightarrow 110 \Rightarrow \text{"}n\text{"} \\
&\text{"}u\text{"} - 3 \Rightarrow 117 - 3 \Rightarrow 114 \Rightarrow \text{"}r\text{"} \\
&\text{"}i\text{"} - 3 \Rightarrow 105 - 3 \Rightarrow 102 \Rightarrow \text{"}f\text{"} \\
&\text{"}c\text{"} - 3 \Rightarrow 99 - 3 \Rightarrow 96 \Rightarrow \text{"}z\text{"} \\
&\text{"}k\text{"} - 3 \Rightarrow 107 - 3 \Rightarrow 104 \Rightarrow \text{"}h\text{"} \\
&\text{"}f\text{"} - 3 \Rightarrow 102 - 3 \Rightarrow 99 \Rightarrow \text{"}c\text{"} \\
&\text{"}o\text{"} - 3 \Rightarrow 111 - 3 \Rightarrow 108 \Rightarrow \text{"}l\text{"} \\
&\text{"}x\text{"} - 3 \Rightarrow 120 - 3 \Rightarrow 117 \Rightarrow \text{"}u\text{"}
\end{aligned}
$$

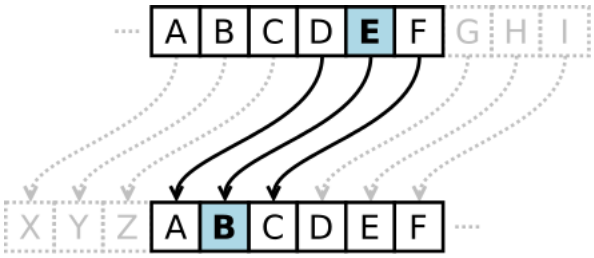
Note that the number ranges in the ASCII table go from 65 to 90 for uppercase letters, and 97 to 122 for lowercase letter. All codes outside of these ranges are not codes for letters (for example, the code for the right square bracket character [is 91.

Whenever a shift causes us to fall outside of the ranges for uppercase or lowercase letters, we must *"wrap around"* to the other end of the range. For example, when we shift the letter *"b"* which has an ASCII code of 98 by minus 3, we should result in 121 (and NOT 95). Think of the ranges as being continous ranges that restart when ever they reach their end: 97, 98, 99, ..., 121, 122, 97, 98, 99, ..., and so on.

If you'd like to play around with a complete Caesar cipher implmenetation, you can head over to this site. The site gives you an option to shift by different numbers (both positive and negative) by updating the *"Shift"* value in the center of the page. If you'd like to learn more about Caesar cipher, I recommend visiting the Wikipedia which has additional examples and information about its history:

> *"In cryptography, a Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. The method is named after Julius*

Caesar, who used it in his private correspondence." https://en.wikipedia.org/wiki/Caesar_cipher



Instructions

Your web application should let a user enter a message which they want to encode along with the shift value which can either be positive or negative by adding a minus sign before it. You should use a `<form>` that accepts the user's input and sends it in a `POST` request. When the user submits their message, you should encode it by shifting all letters by the shift value they provided, then show them the encoded message.

This application is able to both encode and decode messages. All the user has to do to decode a message that was shifted by 3 spaces is provide -3 as the shift value.

Tips

- The `int` function can convert negative numbers: `int("-3") == -3`
- Use the `chr` function to convert a number to a letter (ASCII code to letter): `chr(97) == "a"`
- Use the `ord` function to convert a letter to a number (letter to ASCII code): `ord("a") == 97`
- In order to determine if you need to *"wrap around"* to the beginning or the end of an ASCII code range, you should check if the number you *"shifted"* to falls outside of its corresponding letter range.

  Lowercase letters should fall within 97 to 112. Uppercase letters should fall within 65 to 90. So if you're shifting the letter *"b"* by -3, you'd get 98 - 3 = 95. Because we're working with a lowercase letter, we check that this result falls within the lowercase letter range or 97 - 112, and we can see that it does not, so we would have to wrap around to the end to get the correct result of 121.

  Calculating the wrap around value will require doing some math/logic. The math/logic you use may change depending on if you need to go from the beginning to the end (65 to 90, 97 to 122) or the end back to the beginning (90 to 65, 122 to 97)