

Introduction to the Course

Implementing the Caesar Cipher

Breaking the Caesar Cipher

Introduction	5 min
Arrays	9 min
Random Numbers and Arrays	11 min
Counting with Arrays	10 min
Developing an Algorithm	5 min
Summary	3 min
Programming Exercise: Breaking the Caesar Cipher	10 min
Practice Quiz: Breaking the Caesar Cipher	11 questions

Object Oriented Caesar Cipher



Review

# Counting Occurrences

Hi, do you want a lollipop today?  
I own many good flavors,  
but banana is outstanding.

- Code to scan text and increment counter

increment a counter for each of the 26 different letters.



Developing an Algorithm

Like

Dislike

Flag

Have a question? Discuss this lecture in the week forums.



Interactive Transcript

Search Transcript

English

**0:03**  
We'd like to automate the cracking or breaking of the CaesarCipher. To do this, we'll rely on frequencies of letters in English text.

**0:14**  
If you're encrypting a message in another language, you'll need to use the frequencies from that language. But the approach will be the same.

**0:22**  
We'll write code to find the character that occurs most frequently in the message being decrypted.

**0:29**  
We'll assume this is the letter E, since E occurs more frequently than any other letter in English text. In Russian for example, the letter O occurs more frequently than E. If our assumption about e is wrong, we won't decrypt the original message. It's possible to use more than just E, but to rely on the frequencies of all the letters, and use statistical approaches to break the CaesarCipher. In some cases these approaches will break other encryption methods, though not the methods used to encrypt data for online shopping and secure transactions.

**1:04**  
Let's look at the code for decryption in two steps. We need to count the occurrences of each letter, A through Z, in the message we're encrypting.

**1:15**  
We'll have code to scan in each letter of the text and [increment a counter for each of the 26 different letters](#).

**1:23**  
Initially, all the counters are zero, since we haven't started scanning the text letter by letter.

**1:29**  
Each counter is numbered from zero to 25, because the counters are array elements.

**1:37**  
The index from a string of 26 letters will helps us find the right index for the counter we'll increment as we scan the text.

**1:46**  
As we scan the message, looking at each character, we'll increment the counter at index seven for H.

**1:54**  
Then as we scan, I will increment the counter at index eight, which is the index of I in our alphabet string.

**2:02**  
We won't increment for the comma, or for the space. Then we'll increment the counter at index three for the D. The counter at index 14 for the O.

**2:15**  
We won't increment for the space because space doesn't occur in the alphabet.

**2:21**  
Then we'll increment the counter index 24 for Y.

**2:27**  
And we'll set the counter at index 14 to two when we scan the second O in the message.

**2:35**  
When we're done scanning every character, we'll have these values for each counter. If you look carefully at these values, you see that our decryption method is likely to fail. The value of the counter with index four is zero. There are no Es in this message, but this is a very unusual message. Now, we'll look at the code for this idea. We scanned the message character by character, using a standard for loop.

**3:01**  
We find where the character occurs in a string of each letter in the alphabet. So that E will be found in index four. Notice that we converted the characters in the message being decrypted to lowercase.

**3:14**  
We used the index in the alphabet to increment the corresponding counter as part of decrypting the message. If the character wasn't in the letters of the alphabet the .index of method returned negative one. And we don't increment any counter.

**3:30**  
The code that uses the idea E occurring most frequently, is straightforward developed from the ideas, algorithm, and code you've just seen.

**3:39**  
As you can see, the code isn't very long. But we've created two helper methods, and relied on the CaesarCipher class to help.

**3:47**  
We've called a method countLetters, which we just discussed. This method will count the occurrences of every character in the string encrypted, with A being stored in the first, or index zero location of the array

**4:01**  
returned by the function, and referenced here by the variable freqs.

**4:06**  
Then we call a method maxIndex which will return the index of their entry infreqs that is the largest. The location we will assume is where the E was shifted.

**4:20**  
We'll find that the distance from this location to E. E has index 4. Since we start with zero for A, and then get B, C, D, E, for one, two, three, four respectively.

**4:33**  
If the maximal index is less than four, we'll need to wrap around from 26, to find the shift used for E.

**4:41**  
If the value dkey was used to encrypt, then 26 minus dkey is used to decrypt, and we return the decryptive string. You'll be ready to use your programming knowledge to finish the task of decrypting. And then apply this knowledge in the mini project with a different cipher. But there's some details we want to highlight. The array freqs in the code we just saw has a relationship between the index and the value in the array. For example, freqs of eight is how often I occurs, since I is the ninth letter and has index eight. Remember, we start with index zero. When looking for a maximum value, as with the maxIndex that we called, and whose implementation you see here, we return the index of the largest value, not the largest value itself. We use the index to find the distance from E.

**5:37**  
Using the existing CaesarCipher class made decryption much more straightforward.

**5:43**  
In general, it is a good idea to use code that has already been developed and tested, rather than reinventing it. Have fun coding.

Downloads

Lecture Video	mp4
Subtitles (English)	WebVTT
Transcript (English)	txt

Would you like to [help us translate](#) the transcript and subtitles into additional languages?