Compare Function

A string is actually an array of `unicode characters`. You can perform almost all the operations we used in an array. You can try it out by yourself.

However, there are some differences. In this article, we will go through some of them which you should be aware of when dealing with a string. These features might vary a lot from one language to another.

String has its own `compare function` (we will show you the usage of compare function in the code below).

However, there is a problem:

> Can we use "==" to compare two strings?

It depends on the answer to the question:

> Does the language support `operator overloading`?

1. If the answer is `yes` (like C++), we `may` use "==" to compare two strings.
2. If the answer is `no` (like Java), we `may not` use "==" to compare two strings. When we use "==", it actually compares whether these two objects are the same object.

Let's run the following example and compare the results:

Immutable or Mutable

`Immutable` means that you can't change the content of the string once it's initialized.

1. In some languages (like C++), the string is `mutable`. That is to say, you

can modify the string just like what you did in an array.
2. In some other languages (like Java), the string is `immutable`. This feature will bring several problems. We will illustrate the problems and solutions in the next article.

You can determine whether the string in your favorite language is immutable or mutable by `testing the modification operation`. Here is an example:

Extra Operations

Compare to an array, there are some extra operations we can perform on a string. Here are some examples:

You should be aware of `the time complexity` of these built-in operations.

For instance, if the length of the string is `N`, the time complexity of both finding operation and substring operation is `O(N)`.

Also, in languages which the string is immutable, you should be careful with the concatenation operation (we will explain this in next article as well).

Never forget to take the time complexity of built-in operations into consideration when you compute the time complexity for your solution.