Strings

A string in C# is an object of type System.String. It can be represented as an array of chars. The Char data type represents a character in .NET and the text is stored as a sequential read-only collection of Char objects. There is no null-terminating character at the end of a C# string; therefore a C# string can contain any number of embedded null characters ('\0').

* String objects have fixed length (String.Length)
* String‘s elements can be accessed directly by index
  + The index is in the range [0...Length-1]

The string class defined in the .NET base class library represents text as a series of Unicode characters that is **immutable.** Strings are stored in the dynamic memory

Although string is a reference type, the equality operators (== and !=) are defined to compare the values of string objects, not references. This makes testing for string equality more intuitive.

string a = "hello";  
string b = "h";  
// Append to contents of 'b'  
b += "ello";  
Console.WriteLine(a == b);  
Console.WriteLine((object)a == (object)b);

// the result is:  
// True  
// False

This displays "True" and then "False" because the content of the strings are equivalent, but a and b do not refer to the same string instance.

# String class

The string class provides methods and properties to work with strings.   
  
The string class has methods to clone a string, compare strings, concatenate strings, and copy strings. This class also provides methods to find a substring in a string, find the index of a character or substring, replace characters, spilt a string, trim a string, and add padding to a string. The string class also provides methods to convert a string characters to uppercase or lowercase.

# Useful methods:

string.Compare(firstString, secondString);  
firstString.CompareTo(secondString);

* If the first string is bigger, the result is 1. If the first string is smaller, the result is -1.
* If both strings are equal, the result is 0. The number essentially indicates how much larger the first string is.

string.Concat(firstString, " and ", secondString);

* Concatenates strings

string copySample = string.Copy(sample);

* makes a copy of the string

string replace = stringSample.Replace( stringToReplace, stringToReplaceWith);

* Replaces every string stringToReplace with stringToReplaceWith in the string stringSample. The result is returned to the string replace

string substring = text.Substring(0, 5);

* Gets the first 5 symbols from the text and returns them to the substring
* This method throws nice exceptions, so watch out for the indexes

string[] words = stringToSplit.Split(' ');

* splits the stringToSplit by white space and returns an array of the elements to the array words

string[] words = stringToSplir.Split(charArray , StringSplitOptions.RemoveEmptyEntries);

* splits the stringToSplit by white space and returns an array of the elements to the array words
* if empty strings are found they won’t be returned to the array

//Split string  
 string stringToSplit = "This is an example for string split";  
 string[] words = stringToSplit.Split(' ');  
 foreach (string word in words)  
 {  
 Console.WriteLine(word);  
 }