String

- String is consider as array of characters in all programming languages.
- In C# String is considered as independent data type.
- A string is an object of type String whose value is text.
- Internally, the text is stored as a sequential read-only collection of Char objects.
- In case of C and C++ there is null character ie '\0' at the end of string.
- 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'). The Length property of a string represents the number of Char objects it contains, not the number of Unicode characters.
- To access the individual Unicode code points in a string, use the StringInfo object.

string vs. System.String

- In C#, the string keyword is an alias for String.
- Therefore, String and string are equivalent, and you can use whichever naming convention you prefer.
- The String class provides many methods for safely creating, manipulating, and comparing strings.
- In addition, the C# language overloads some operators to simplify common string operations.

There are different ways in which we can create string as

1. Declare without initializing.

```
string str1;
```

2. Initialise to null.

```
string str2 = null;
```

3. Initialise to empty string.

```
string str3 = "";
```

4. Initialize as an empty string. Use the Empty constant instead of the literal "".

```
string str4 = System.String.Empty;
```

5. Initialize with a regular string literal.

```
string str5 = "Marvellous Infosystems";
```

6. Initialize with a verbatim string literal.

```
string str6 = @"Marvellous Infosystems";
```

7. Use System.String if you prefer.

```
System.String str7 = "Marvellous Infosystems";
```

8. In local variables (i.e. within a method body)

```
var str8 = "Marvellous Infosystems";
```

9. Use a const string to prevent str9 from being used to store another string value.

```
const string str9 = "Marvellous Infosystems";
```

10. Use String constructor only when creating a string from a char*, char[], or sbyte*.

```
char[] arr = {'M','a','r','v','e','l','l','o','u','s'};
string str10 = new string(arr);
```

Immutability of String Objects :

- String objects are immutable: they cannot be changed after they have been created.
- All of the Stringmethods and C# operators that appear to modify a string actually return the results in a new string object.
- In the following example, when the contents of str1 and str2 are concatenated to form a single string, the two original strings are unmodified. The += operator creates a new string that contains the combined contents.
- That new object is assigned to the variable s1, and the original object that was assigned to s1is released for garbage collection because no other variable holds a reference to it.

Example :

```
string str1 = "Marvellous";
string str2 = "Infosystems";
```

Concatenate str1 and str2. This actually creates a new string object and stores it in str1, releasing the reference to the original object.

```
s1 += s2;
System.Console.WriteLine(str1);     // Marvellous Infosystems
```