# Strings – Introduction to Strings

## 1. What is a String?

A **string** is a sequence of **characters** stored together to represent **textual data** such as words, sentences, or symbols.

In most programming languages, a string is treated as an **array (or collection) of characters**.

Example:

```
"CodeZapra"
```

Each character (C, o, d, e, Z, a, p, r, a) occupies a position in the string.

## 2. Why Do We Need Strings?

Strings are essential because programs often deal with **text-based data**, such as:

- User names and passwords
- Messages and descriptions
- File names and URLs
- Search queries and commands

Without strings, handling text data would be extremely difficult and inefficient.

## 3. Characteristics of Strings

Important characteristics of strings include:

1. **Sequence of Characters**

   A string is made up of individual characters arranged in order.

2. **Index-Based Access**

   Each character can be accessed using an index, starting from 0.

3. **Immutable or Mutable (Language Dependent)**

   - In some languages (like Python, Java), strings are **immutable**
   - In others (like C), strings can be modified

4. **Contiguous Memory Storage**

   Characters are stored in consecutive memory locations.

5. **Length Property**

   Every string has a defined length representing the number of characters.

## 4. String Indexing

Just like arrays, strings use **zero-based indexing**.

Example string:

```
"HELLO"
```

| **Index** | **0** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|---|
| Char | H | E | L | L | O |

- First character → index 0
- Last character → index length − 1

## 5. Memory Representation of Strings

Strings are stored as **character arrays** in memory.

Example:

```
"ABC"
```

Memory layout:

```
A | B | C
```

In some languages (like C), an additional **null character (\0)** is used to mark the end of the string.

## 6. Types of Strings

### 1️⃣ Single-Line Strings

- Contain characters in a single line
- Example: "Hello World"

### 2️⃣ Multi-Line Strings

- Can span across multiple lines

- Used for paragraphs or long text

## 7. Basic Operations on Strings

Strings support several fundamental operations, such as:

- Traversal (visiting each character)
- Comparison
- Concatenation (joining strings)
- Reversal
- Substring extraction
- Searching characters or patterns

These operations form the base for advanced string algorithms.

## 8. Advantages of Using Strings

- Easy handling of text data
- Built-in support in most languages
- Efficient storage of characters
- Useful for communication-based applications

## 9. Limitations of Strings

- Modification can be costly in immutable strings
- Memory usage increases with large text
- Some string operations are time-consuming

## 10. Real-World Examples of Strings

- Username and password validation
- Email addresses
- Search engine queries
- Text editors
- Chat applications

## 11. Importance of Strings in DSA

Strings play a major role in:

- Pattern matching
- Text processing
- Data validation
- Competitive programming
- Interview problems

Understanding strings is essential before learning advanced topics like **substring search**, **anagrams**, and **palindromes**.

## 12. Summary

- Strings are sequences of characters
- They store textual data efficiently
- Characters are accessed using indexes
- Strings are widely used in real-world applications
- They are a core topic in DSA