# All About Strings

This chapter introduces you to the basic concepts of character strings in R.

**We'll cover the following**

## Creating Character Strings

In R, we can express character strings by surrounding text with double quotes:

“learning is fun”,

or we can also surround text with single quotes:

‘learning is fun’.

Double_Quotes | Single_Quotes

```r
cat("learning is fun")
```

Creating string using double quotes

## Empty String

The most basic type of string is the **empty** string produced by consecutive quotation marks: `""`.

> `""` is a string with no characters in it, hence the name **empty string**:

Double_Quotes Single_Quotes

```
cat("")
```

Empty string using double quotes

## Escape Sequences

A sequence that starts with a `\` in a string is called an **escape sequence**. It allows us to include special characters in our strings.

Common escape sequences are:

| Escape sequence | Usage |
|---|---|
| `\n` | newline |
| `\t` | tab |
| `\\` | backslash |
| `\'` | Single quote (') |
| `\"` | Double quote (") |

> You saw one escape sequence previously: `\n` is used to denote a new line.

Each escape sequence is considered one character.

'\n' | '\t' | '\\'

```
cat("Learning\nis\nfun")
```

Using newline escape sequence

## Basic Operations with Strings

The following are some of the basic methods for string manipulation. Have a look at their codes.

### Length of a String

The length of a string can be found using a simple method `nchar()`. Let's find the length of the string "learning is fun".

```
nchar("learning is fun")
```

Example of how to use nchar()

Empty spaces or tabs, i.e. `\t`, are also considered characters and are added in the total length of the string.

#### Why Do We Not Use `length()` Here?

The keyword `length()` gives the length of **R objects** for which this method has been defined. It returns the number of **elements** in the object, so for a *single string* it will return 1.

```
length("learning is fun")
```

Example of how to use length()

### Concatenating Two Strings

Two strings can be concatenated using `paste()`.

```
paste("learning", "is", "fun")
```

Example of how to use paste()

By default, `paste()` inserts a single space between pairs of strings.

However, this default setting can be overridden using the `sep` argument in `paste()`.

For example, we can set a `.` or an empty string `""` as a separator.

sep="." sep=""

```
paste("learning", "is", "fun", sep = ".")
```

paste() with sep = "." argument

## Duplicating Strings

We can duplicate the same string multiple times using the keywords `replicate()` or `rep()`.

### Syntax for `replicate()`

```
replicate(<numberOfTimes>, <stringToReplicate>)
```

### Syntax for `rep()`

```
rep(<stringToReplicate>, <numberOfTimes>)
```

The parameter `numberOfTimes` specifies how many times to repeat the parameter `stringToReplicate`.

replicate() rep()

```
replicate(3, "learning is fun")
```

Example of how to use replicate()

Both `rep` and `replicate` perform the same task. However, the order of the arguments is the opposite.

---

In the next lesson we will be learning an important concept: the difference between `cat()` and `print()`. It is particularly important to understand this concept early on as we'll be moving on to more difficult concepts later in the course.