

## 2d. Strings

[Martin Alfaro](#)

PhD in Economics

### INTRODUCTION

This section presents types for text representation, distinguishing between characters and strings. The coverage will be concise, as the website won't focus on string analysis. However, a minimal coverage is necessary as string variables are important for tasks like specifying paths, saving files, and other core functionalities.

### CHARACTERS

In Julia, the `Char` type is used to represent individual characters. A character `x` is defined by using single quotes, as in `'x'`. Given its support for Unicode characters, `Char` encompasses not only numbers and letters, but also a wide range of symbols. This is shown below.

```
# x equals the character 'a'
x = 'a'

# 'Char' allows for Unicode characters
x = 'β'
y = '🐵'
```

Notice that characters must be enclosed in single quotes `' '`, even for symbols like `🐵`. Otherwise, Julia will interpret the expression as a variable.

```
# any character is allowed for defining a variable
🐵 = 2      # 🐵 represents a variable, just like if we had defined x = 2

y = 🐵      # y equals 2, 🐵's value at that moment (not 🐵 itself)
z = '🐵'    # z equals the character 🐵 (entirely independent of the 🐵 variable )
```

### STRINGS

We'll rarely use the type `Char` directly. Instead, we'll work with the so-called type `String`. This is an ordered collection of characters, making it possible to represent text.

Strings can be defined through either double quotes `" "` or triple quotes `""" """`. The latter is particularly convenient for handling newlines, such as when the text has to span multiple lines. <sup>1</sup>

```
x = "Hello, beautiful world"

x = """Hello, beautiful world"""
```

## **STRING INTERPOLATION**

**String interpolation** allows you to embed Julia code within a string, which is then evaluated and replaced in the string with its value.

To interpolate an expression, you must simply prefix the string with the `$` symbol. If the expression contains spaces, you'll need to enclose it in curly braces, like `$( )`. Both cases are exemplified below.

```
number_students = 10

output_text      = "There are $(number_students) students in the course"
```

```
julia> output_text
"There are 10 students in the course"
```

```
number_matches  = 50
goals_per_match  = 2

output_text      = "Last year, Messi scored $(number_matches * goals_per_match) goals"
```

```
julia> output_text
"Last year, Messi scored 100 goals"
```

---

### **FOOTNOTES**

<sup>1</sup>. For more on the differences between double and triple quotes, see [here](#)