

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 treatment is necessary, as string variables are important for tasks like specifying paths, displaying messages, and documenting functions.

CHARACTERS

The `Char` type is employed to represent individual characters. Characters are written by enclosing them in single quotes, as in `'x'` for the character `x`. Given Julia's support for Unicode characters, `Char` encompasses not only numbers and letters, but also a wide range of symbols.

```
# 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, permitting the representation of 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. ¹

```
x = "Hello, beautiful world"

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

STRING INTERPOLATION

String interpolation lets you embed Julia code directly inside a string. The embedded expression is then evaluated and replaced in the string with its value.

To interpolate an expression, the string must be prefixed with the `$` symbol. If the expression contains spaces, it must be enclosed in curly braces, as in `$()`. 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

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