Lesson 7, Week 1: Strings III (string interpolation)

AIM

— To explain what string interpolation is, how it works and what it's for.

— To give examples of some of the main uses.

After this lesson, you will be able to

* Explain what string interpolation is and what is for.
* Use the `$()` syntax to do string interpolation.
* Explain what inputs can be used for string interpolation, and give some examples of such inputs.

What is string interpolation, and what is it for?

String interpolation is a method for adding something to a string literal. Why not just make the literal and be done? There are several reasons. One of them is to make a table of values that you have computed. Very often, you want to use formatted strings in the table. But you don't have the values until they are computed. So the code tells Julia to make the formatted string in two steps: first make the part that doesn't require the computed value, and then when the value is available interpolate it. Another is to make labels. Often one needs to label many things with labels that change only a little each time—interpolation is ideal for just adding the changed bit each time.

The `$()` syntax

The `$()` is very simple: you just put it in a literal string. For example, if you had the string `"zxywvut"` and you wanted to interpolate the cry `Help!` in place of the `w`, you could just use `"zxy$("Help!")vut"`.

Important point: the input to `$()` (that is, the inside of the parentheses) must be either a string value, as here, or something that can be formatted as a string value.

Using a variable as input to string interpolation

As we saw, string interpolation is most useful when it uses values not available at the time of writing the code. Here's one that just gives the value of a variable[1]:
`"The value of trackedvariable is $(trackedvariable)"`.

DEMO: using this, for sevaral values of `trackedvariable`.

For example, suppose somehow the variable `todaysdate` was a string with the correct date for today. String interpolation can then be used to label a prediction temperature with today's date.

Shortcut: in some cases, you can omit the parentheses around the input to `$()`. For example
`"The value of trackedvariable is $trackedvariable"`
Why? Well, in the full `$()` syntax, the parentheses are there to delimit exactly the input to be interpolated. So in cases like this, where the `$` character is unambiguously followed by a variable name, Julia decides simply to use that variable as the input for the interpolation. That is, when the input to be interpolated is nothing but a variable name, you can leave off the parentheses[2].

Using an expression as input to string interpolation

Any expression that can be formatted as a string can be the input. This includes characters by themselves and in expressions, the operators `*` and `^` and the shift arithmetic on characters we saw in the previous lesson.

DEMO: `temp = "aa", n = '1'; "the value is $(temp*n)` (note that `n` has a character value—numerical value for `n` doesn't work[3]), this is because of how `*` works.

Because simple numbers so often occur, they can be interpolated, as in `"a$(22)b"` and `"a$(2.222)b"`. This can even include some arithmetic, as in `x=3; "a$(x+39)b"`

[1] Useful to keep track of a computation!

[2] Provided that the variable name is unambiguous—for example, followed by a space or a comma or a full stop. But following it with an exclamation mark would not work. An exclamation mark can be part of a valid name in Julia, so there are two possible names here, the one with and the one without the exclamation mark. Ambiguous!

[3] We'll see later how to get the string version of a numerical value.

Including the character `'$'` in a string literal

Escape it! For example (we must use `print` or `println` — recall that the escaping is only visible after the string is formatted):

```
println("That'll be \$4.99, please")
```

Review and summary

* A literal string can include interpolations via the `$()` syntax
* The input to `$()` is anything that can be formatted as a string: characters, concatenations, character arithmetic, numbers, and number arithmetic
* Interpolations are particularly useful for making strings with values that only become known after the code for making the string was written.
* To include the character value `'$'` in a string value, escape it with `\$`.