Lesson 6, Week 1: Strings II (escape sequences)

AIM

— to understand what an escape sequence is

— to learn the escape sequence `\"`

— to explain the difference between a string with an escape sequence and its formatted appearance

— to learn the escape sequence `\n`

— to learn the escape sequence `\t`

— to learn the escape sequence `\\`

After this lesson, you will be able to

* Say why `'$'` and `\` behave oddly when used to make a string value in Julia
* Define an escape sequence in Julia
* Understand that formatting a string with an escape sequence gets rid of the escape sequence
* Effectively use the escape sequences `\"`, `\n` and `\t`
* Understand that `\` can be used to escape itself

Neither `$` nor `\` come up literally if they're in a string

DEMO: here are two invalid strings: `"\"` and `"$"`

This lesson is about strings like these, where what you see is different from what you type. In fact, it will take us two lessons to cover this topic. In this, we look at what is called *escape sequences*: they all start with a backslash. In the next lesson , we look at what is called *string interpolation*, which involves the dollar character.

What you'll see is that the formatted form of a string is affected by the escape sequences in the string. You already know that the formatted form is different from the string literal itself, because you've seen the difference between `"Hello, world"` and `print("Hello, world")`.

DEMO: a few more examples of the literal vs formatted version of a string

Making and printing a string with the escape sequence

In Julia, all escape sequences start with `\`. A useful and very simple escape sequence is `\"`, which allows you to have double quotes *inside* the formatted version of a string.

DEMO: you cannot get the formatted string `He said "Hello, world"`
with the code `x="He said "Hello, world""; print(x)`.

The reason the code above fails is clear: the first double quote character starts a string, and then the second one ends it. It is worth tarrying a while over the error message syntax: cannot juxtapose string literal. The string has ended with that second double quote, so what follows is new code. Julia could wait to see what the new code might be, but since this is such a common source of bugs, there is a complicated rule for detecting whether this is really the end of a string. Here we have a letter immediately after the double quote, hence the message. A space or a delimiter gives a different error message, and so on.

DEMO: the code you need is `x="He said \"Hello, world\""; print(x)`

We need to read this with some care: the two double quotes we want to see in the formatted version are said to be *escaped* by the backslashes preceding them. The combination `\"` is an escape sequence.

The escape sequences `\n` and `\t`

The escape sequence `\n` shifts the bit of string that follows onto a new line[1].

DEMO: `x="Hello, \nworld"; print(x)`

[1] And therefore is oftened called the newline character—yes, the two characters together are thought of as one character.

With this in hand, we can easily write multiple lines in a single string[2], for example

```
doggerel = "Errors are red, \nsome things are blue, \nI love coding \nand so should you"
println(doggerel)
```

The escape sequence `\t` inserts empty space up to the next tab stop:

```
tabexample = "I \twait"
println(tabexample)
```

Here is a combination of these two escape sequences, showing exactly how the tabbing works:

```
tabstops = "12345678901234567890\nI \twait"
println(tabstops)
```

Clearly, the tabbed spaces are eight characters wide. The shift is to the next available starting point:

```
tabstops = "123456789012345678901234567890\nI will just sit here and \t wait"
println(tabstops)
```

Also if several tabs are inserted, all of them make a shift:

```
tabstops = "123456789012345678901234567890\nI just sit \t\tand wait"
println(tabstops)
```

Escaping the escape: `\\`

Of course, if you want `\` to appear in your string, you must escape it:

```
backslash_eg ="1 backslash 4 is 1\\4"
println(backslash_eg)
```

Review and summary

* The characters `\` and `$` are special and don't appear in a formatted string if you simply type them in.
* To see " and \ in a quoted string, escape them as follows: `\"` and `\\`
* To insert a linebreak in a formatted string, use the escape sequence `\n`
* To insert a tab stop (8 spaces on from the previous stop), use the escape sequence `\t`

[2]Strings can be arbitrarily long—a whole novel, linebreaks and all, could be a single string.