Lesson 1, Week 1: A minimal working example

AIM

* To provide an example in order to start for talking about Julia

---

In this our first lesson, we draw extensively on Lesson 0—the one on what you have to have in place to be ready for this course[1].

I also want to remind you of the course slogan: **small steps, no gaps, make sense always**. The aim is to keep nervous beginners on board!

In this lesson we start from zero, so we can't keep to all parts of the slogan fully. The idea is first to see coding in action in this lesson, and then explain it in detail in lesson 2.

Entering REPL code

Open the REPL, enter `"Hello, world"`.

DEMO: In the video of this lesson, we show you exactly how to do this.

`"Hello, world"` is a string value. Julia has many other kinds of values: we'll see some of them, such as number values and character values[2].

Now enter `mystringexample1 = "Hello, world"`. This is called *assigning a value to a variable.*

IMPORTANT: The `=` sign binds the string value on the right hand side to the variable name on the left. This changed your computer's memory in three places:

- The name `mystringexample1` was put into what is called the namespace[3] .
- The string value `"Hello, world"` was created[4].

[1] Firing up Julia and getting an REPL; editing plain text files and storing them in a special folder for this course.
[2] That is, values that are characters.
[3] Actually, when Julia is running it can have several namespaces, but that is an advanced topic we do not address on this course. Once a name is in a namespace, it will stay there until you shut down the whole namespace. Closing down your Julia session also closes down all the namespaces.
[4] Separately from but in the same way as above.

- The `=` sign between the name on the left and value on the right created a binding between the name and the value.
- By binding the string value to the name, Julia is keeping the string value in your computer's memory so that the value is available in case it is needed later.

Enter `println(mystringexample1)` .

DEMO: `println` is built-in function

What happens when this line is run[5]:
The function `println()` receives the variable name `mystringexample1`, fetches its value (which is a string), reformats it, and on the screen displays the string followed by blank line.

Functions are very, very important in Julia. Many are built-in, such as `println()`, but Julia programs also create many more. You will learn a lot about functions in Julia on this course!

Creating and running a code file

Finally, create `myfirstfile.jl`, as a plain text file (NB!) containing exactly the two lines of code we used above, save it in your course folder[a]. Make sure your course folder is your working directory[b], and enter `include("myfirstfile.jl")`

DEMO: in the video, we show that the result is the same as for the REPL code we used earlier.

[a] That is, create a Julia code file—such files are one of the topics of Lesson 0.
[b] Use `pwd()` to check what your working directory is, and `cd()` to change it.

Congratulations! Your first Julia program! Coding is that simple.

Review and summary

* `"Hello, world"` is a string value.
* `println()` is a function.
* `mystringexample1` is a variable name.
* `=` is assignment: the value of the right hand side binds to the name on the left hand side.
* The function `include()` runs the lines of code it receives from a Julia code file.
* A Julia code file is a plain text file with the extension `.jl`

What we did in this lesson is what we'll do over and over as the course proceeds: some new ideas, some examples (which you to try as the lesson proceeds), and some code files for you to write and execute.

Please do the quiz, the exercise and the self-graded assignment before going on to Lecture 2 of Week 1. They're very short! It really is best to do them before going on.

[5] People also say: "when this line is executed", and "when this line is evaluated".