A Short and Incomplete Introduction to Julia

Part 0: Introduction

Riccardo Murri <riccardo.murri@uzh.ch>
S3IT: Services and Support for Science IT,
University of Zurich

September 12, 2019

Welcome!

Prerequisites

This course assumes a basic experience with computer programming.

Any language should do, as long as you are already familiar with the concepts of variables and functions.

What about you?

Name / Affiliation / Interest in Julia? /
Other known programming languages?

Course outline

1. Julia basics
2. Array manipulation and plotting
3. How to query tabular data

A word of warning

Julia is a language with many features — it's not possible to cover all in few hours.

The course starts with the basics, and aims to teach you enough to be able to do common tasks that you would use MATLAB or Python for, and being able to search further on the Internet.

We will not, unfortunately, see the more advanced and interesting stuff.

(This is also the first time I use this material, so I will be grateful for *any* feedback! No, really, it *is* important.)

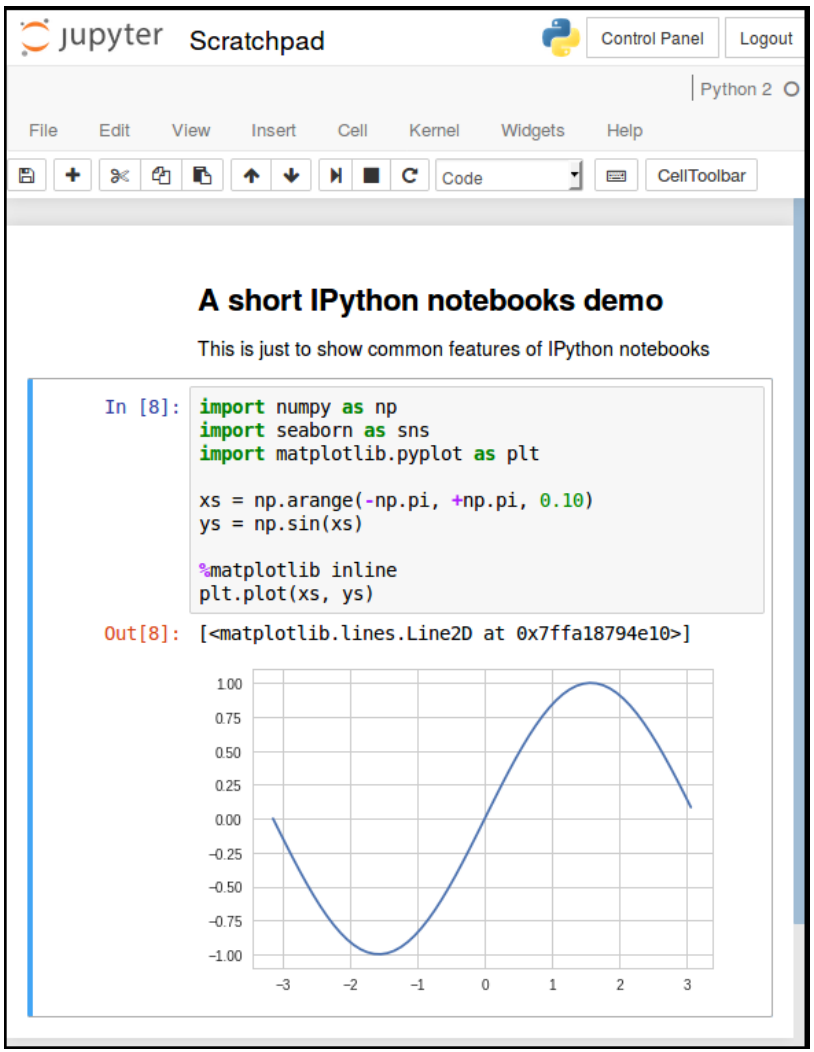
Next steps

The course will be structured as a mixture of slides and hands-on sessions for practicing Julia programming.

So, the very first step is making sure you can access the Jupyter server for running the exercise notebooks.

How to run Julia code

The Jupyter notebook, I



An appealing way of interacting with Julia is through *Jupyter notebooks*.

Notebooks are made of "cells", which come in two flavors:

- documentation cells, containing text formatted according to the Markdown conventions;
- code cells, containing arbitrary Julia code

The Jupyter notebook, II

To run Julia code in the notebook:

- Type your code in a cell besides the `In [ ]:` (multiple lines are allowed)
- Press **Ctrl+Enter** to evaluate the cell (prompt changes to `In [*]:`) — or press **Alt+Enter** to evaluate the code *and* open a new code cell.
- When the Julia kernel has done computing, the result appears *under* the code cell marked with a `Out [ ]:` label.

The Julia REPL, I

Julia features an interactive "shell" for evaluating expressions and statements immediately.

Julia interaction is started by invoking the command `julia` in a terminal window.

```
$ julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.2.0 (2019-08-20)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |
```

julia> ← *this is where you enter commands*

The Julia REPL, II

Expressions can be entered at the Julia REPL prompt; they are evaluated and the result is printed:

```
julia>   2+2
4
```

(The acronym REPL indeed means “Read-Eval-Print-Loop”.)

Throughout these slides, all code marked with either ‘In [*]’ or ‘`julia>` ’ can also be entered and evaluated in the Julia notebook cells.

Getting help from Julia

Typing `?` followed by a word
searches the built-in help for that topic.

```
In [1]:  ?julia
Out[1]:
search:

  Welcome to Julia 1.2.0. The full manual is available at

  https://docs.julialang.org/

   [...]
```

Works both in the REPL and in the Notebook.

Use it to find help about any topic in Julia, especially built-in functions and types (e.g., try '`?print`').