ÖGOR Summer-Workshop for PhD-candidates and Post-Docs

An introduction to Julia and JuMP for Operations Research

Prof. Dr. Xavier Gandibleux

Nantes Université – France
Département Informatique – Faculté des Sciences et Technique

Topic 1

Getting started

Software environment

Software environment: check list

Base:

- Operating system: **MacOS**, Windows, Linux (ubuntu), etc.
- Browser: **Firefox**, Safari, Chrome, etc.

Apps to install on your computer:

- Language: **Julia 1.9.1**
- Editor: **Visual Studio Code (VSCode)**, Emacs, etc.

Apps to install for optimisation needs:

- Open-source MILP solver: **GLPK**, HiGHS, etc.
- Commercial MILP solver: **Gurobi**, Cplex, etc.

Apps used (installation not mandatory but recommanded):

- interacting with repository: **GitHub Desktop**

Software environment: check list

Base:

- Operating system: **MacOS**, Windows, Linux (ubuntu), etc.
- Browser: **Firefox**, Safari, Chrome, etc.

Apps to install on your computer:

- Language: **Julia 1.9.1**
- Editor: **Visual Studio Code (VSCode)**, Emacs, etc.

Apps to install for optimisation needs:

- Open-source MILP solver: **GLPK**, HiGHS, etc.
- Commercial MILP solver: **Gurobi**, Cplex, etc.

Apps used (installation not mandatory but recommanded):

- interacting with repository: **GitHub Desktop**

Software environment: check list

Base:

- Operating system: **MacOS**, Windows, Linux (ubuntu), etc.
- Browser: **Firefox**, Safari, Chrome, etc.

Apps to install on your computer:

- Language: **Julia 1.9.1**
- Editor: **Visual Studio Code (VSCode)**, Emacs, etc.

Apps to install for optimisation needs:

- Open-source MILP solver: **GLPK**, HiGHS, etc.
- Commercial MILP solver: **Gurobi**, Cplex, etc.

Apps used (installation not mandatory but recommanded):

- interacting with repository: **GitHub Desktop**

Software environment: check list

Base:

- Operating system: **MacOS**, Windows, Linux (ubuntu), etc.
- Browser: **Firefox**, Safari, Chrome, etc.

Apps to install on your computer:

- Language: **Julia 1.9.1**
- Editor: **Visual Studio Code (VSCode)**, Emacs, etc.

Apps to install for optimisation needs:

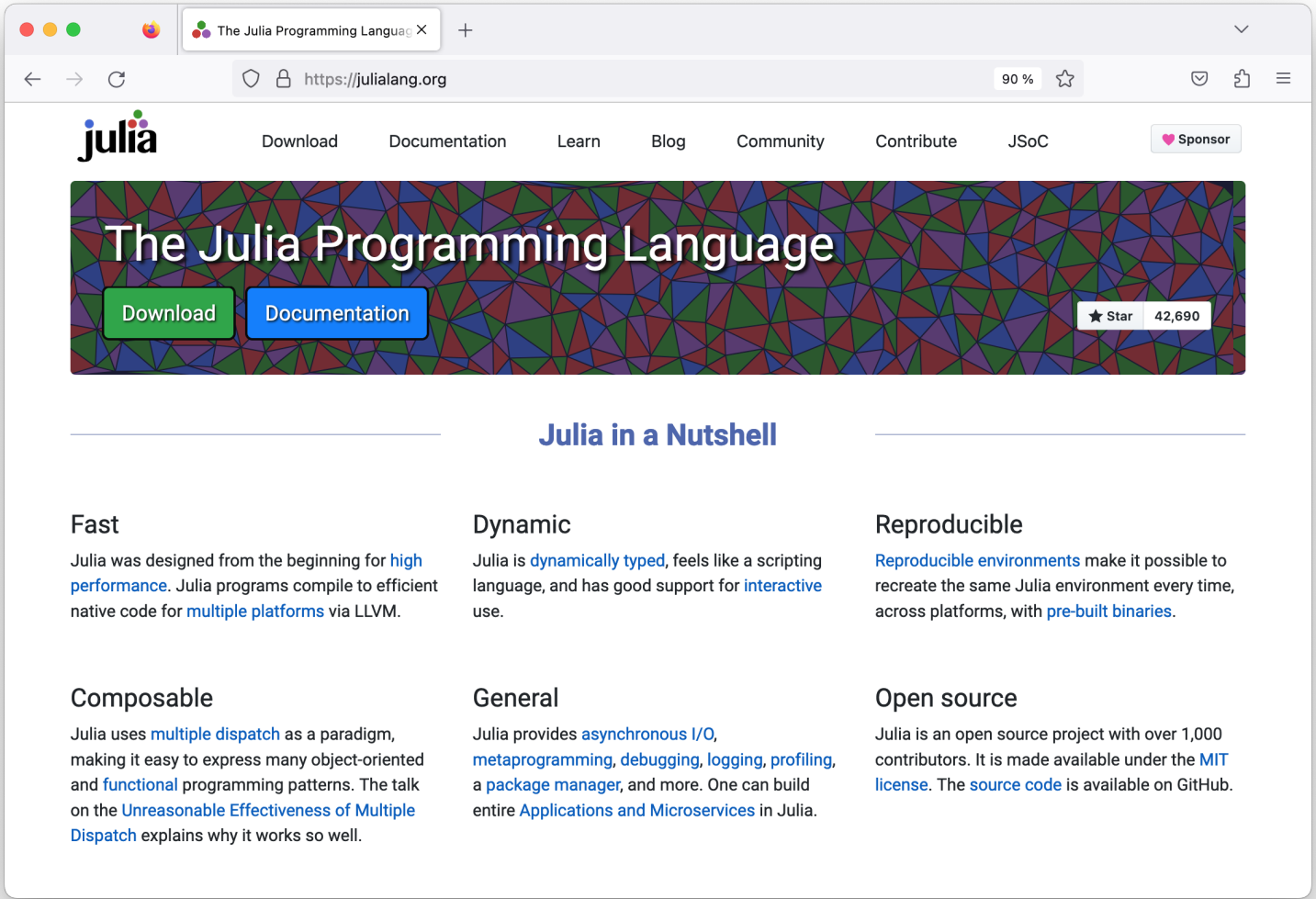
- Open-source MILP solver: **GLPK**, HiGHS, etc.
- Commercial MILP solver: **Gurobi**, Cplex, etc.

Apps used (installation not mandatory but recommanded):

- interacting with repository: **GitHub Desktop**

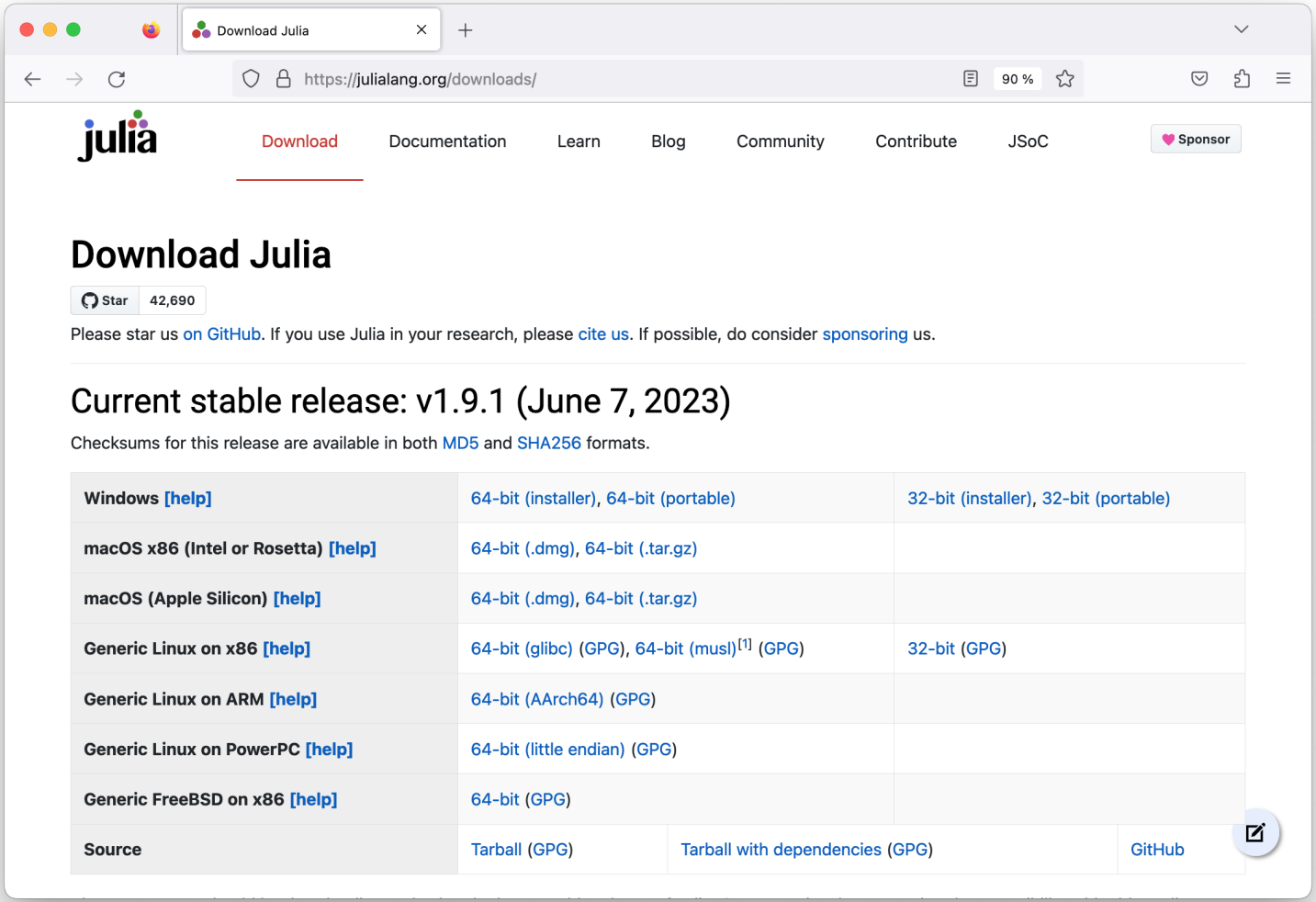
Language: Julia 1.9.1

`https://julialang.org/`



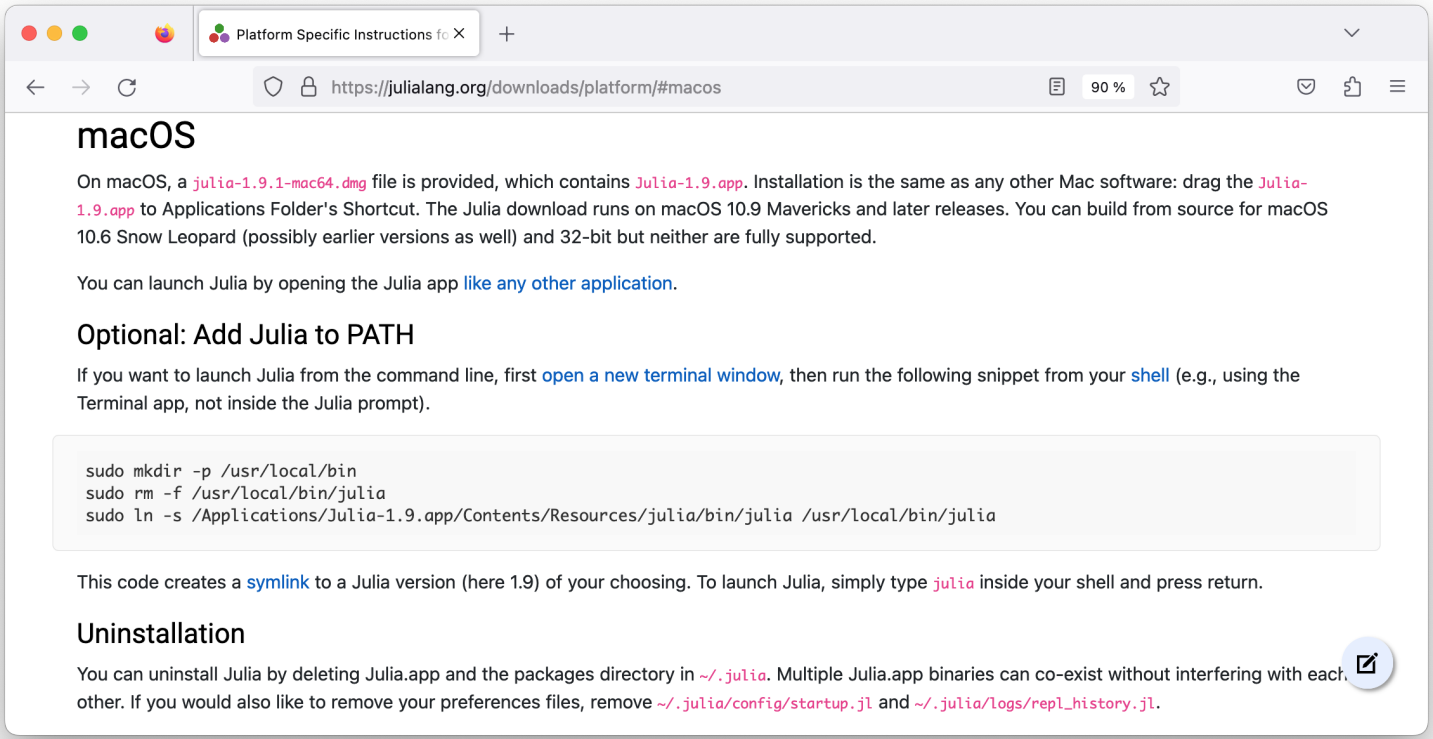
Language: Julia 1.9.1

`https://julialang.org/downloads/`



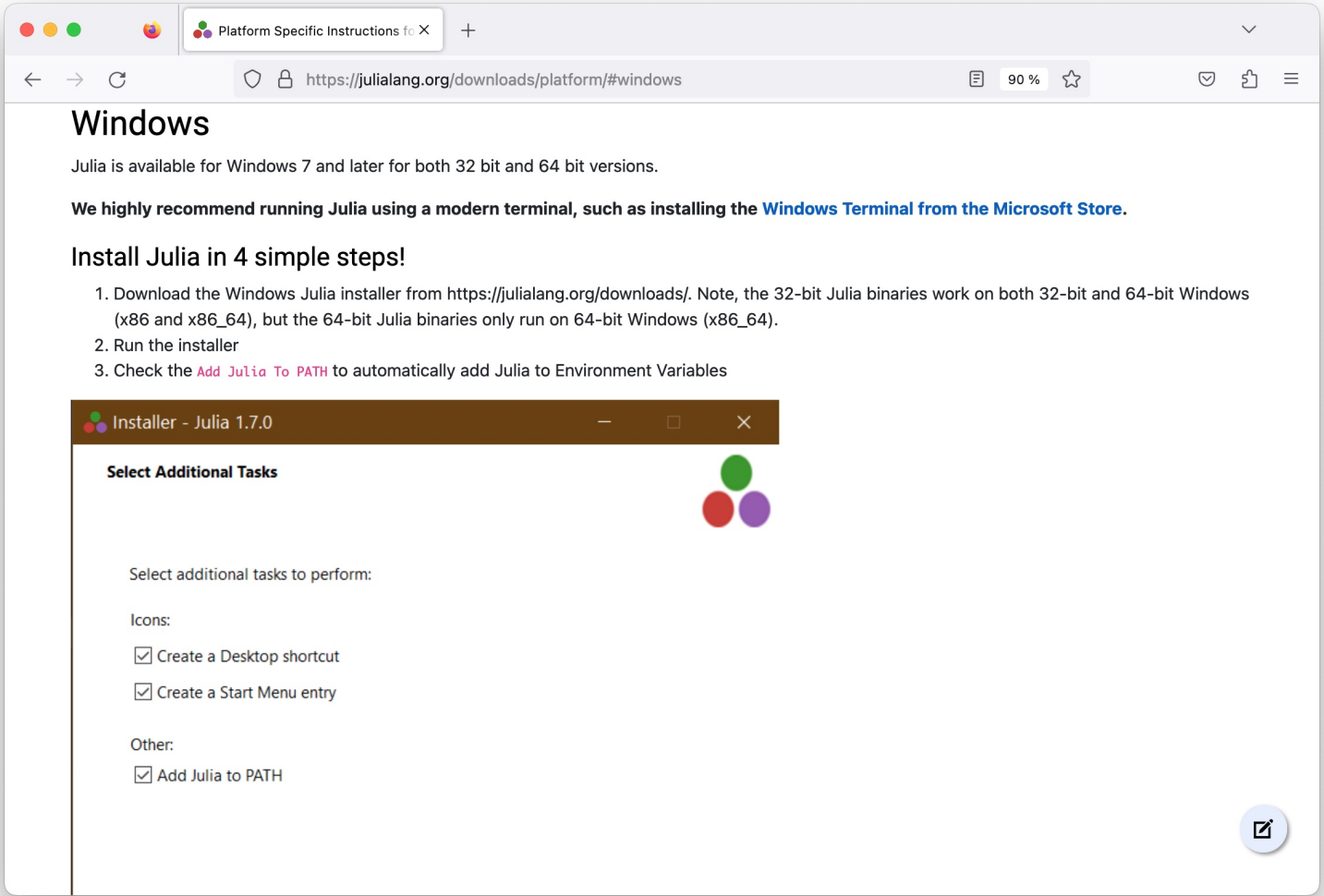
Language: Julia 1.9.1

`https://julialang.org/downloads/`



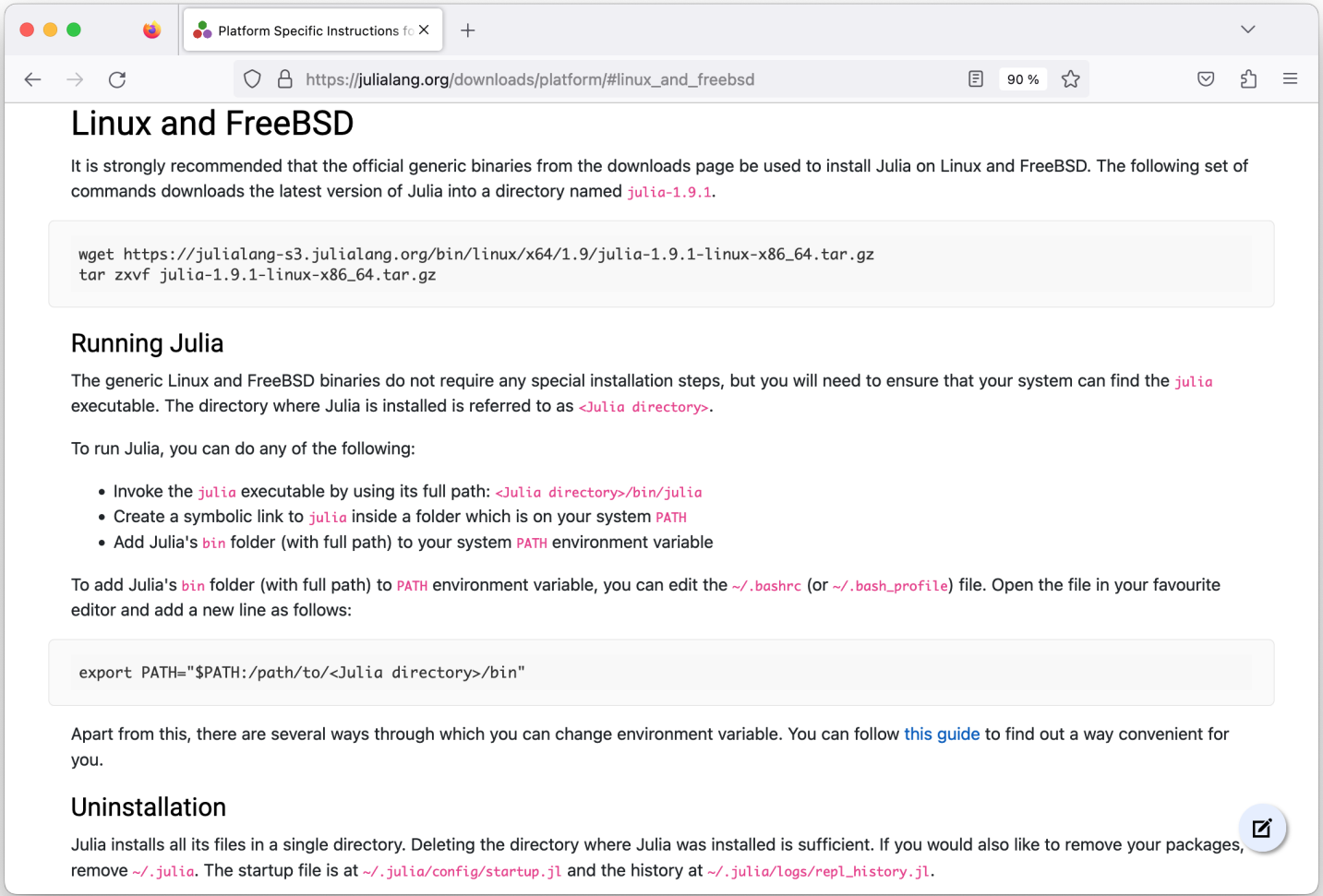
Language: Julia 1.9.1

`https://julialang.org/downloads/`



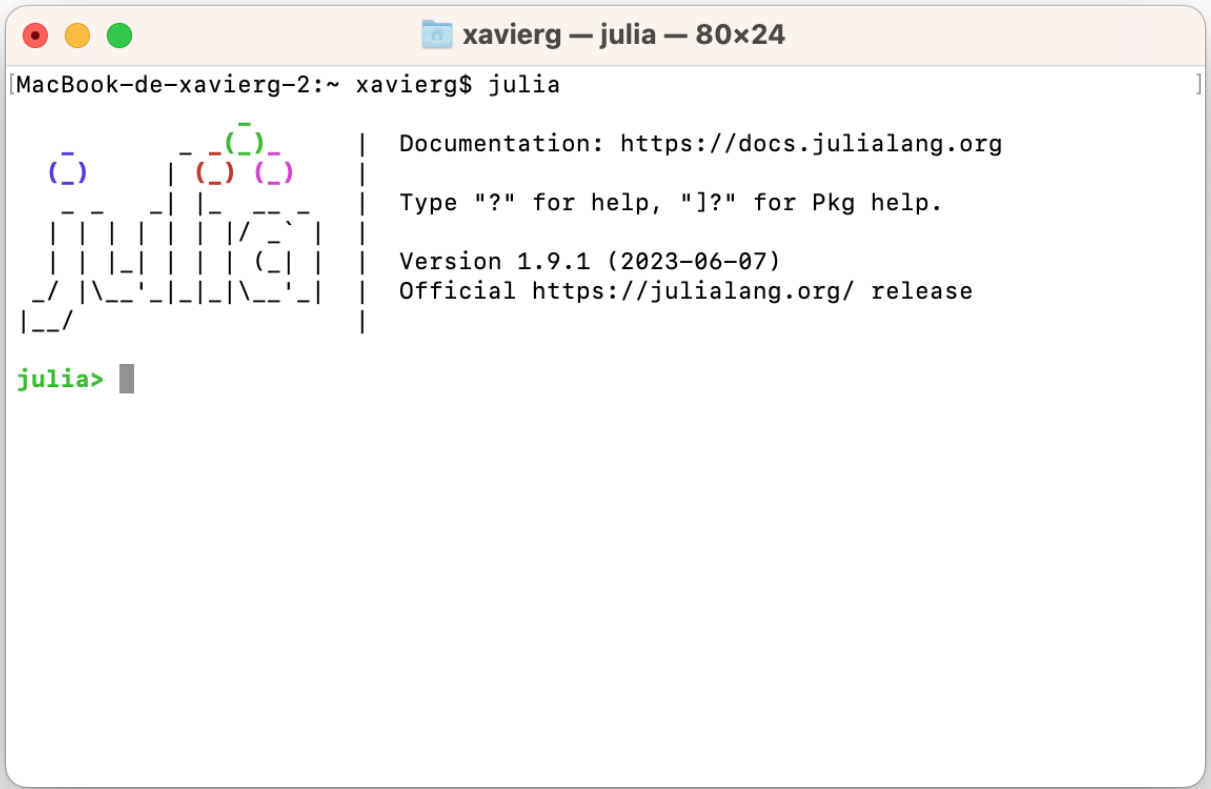
Language: Julia 1.9.1

`https://julialang.org/downloads/`



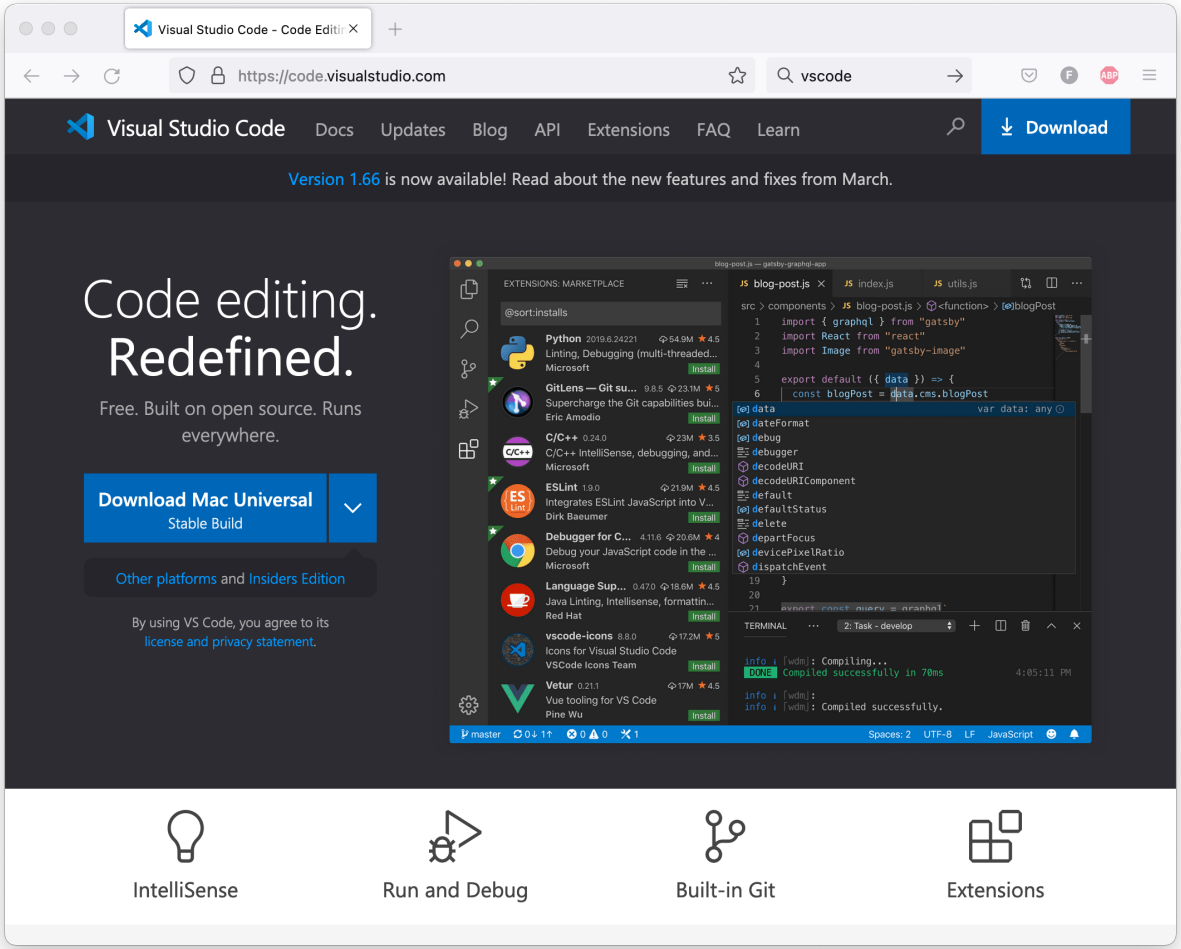
Language: Julia 1.9.1

Just click on the app icon



Editor: Visual Studio Code

`https://code.visualstudio.com/`

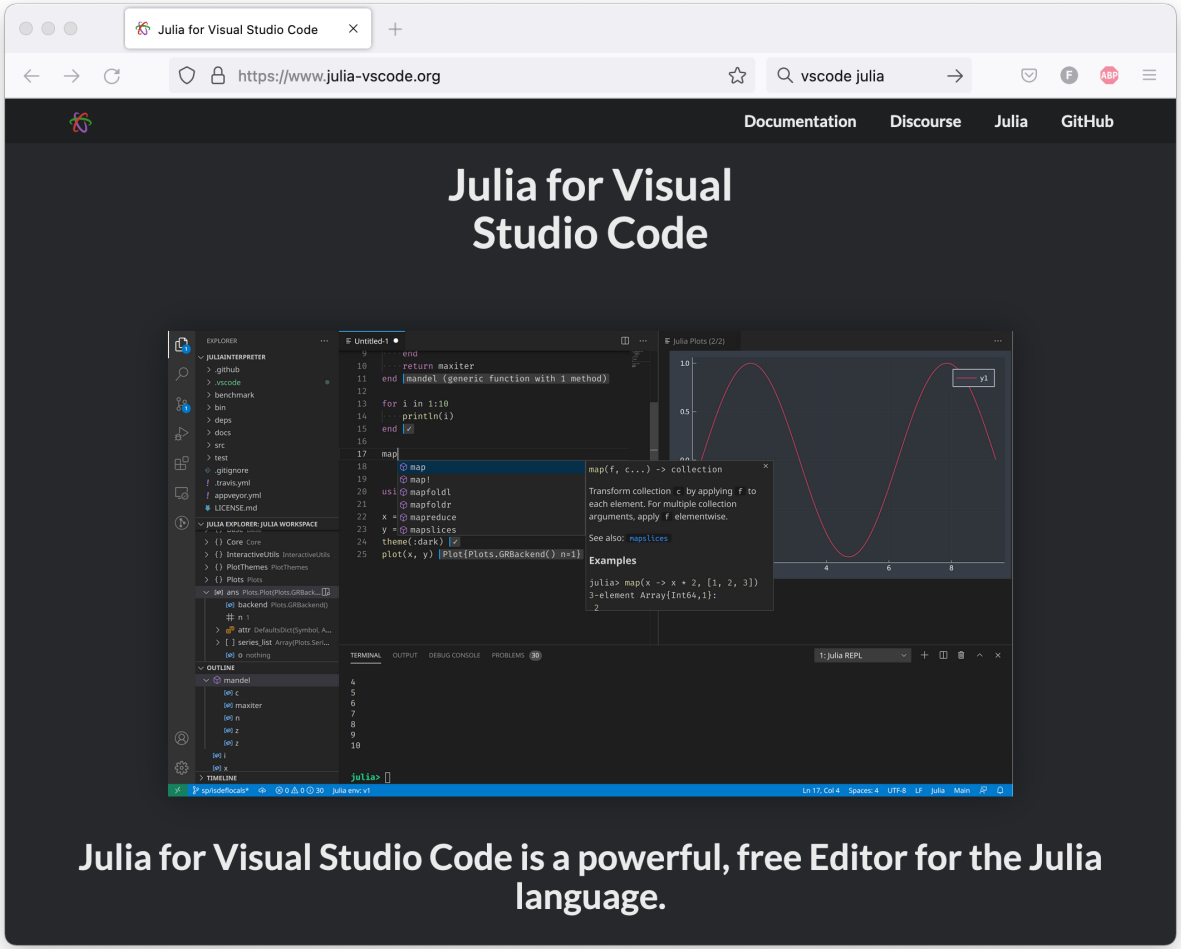


extended for Julia

`https://www.julia-vscode.org/`

Editor: Visual Studio Code

`https://code.visualstudio.com/`



extended for Julia

`https://www.julia-vscode.org/`

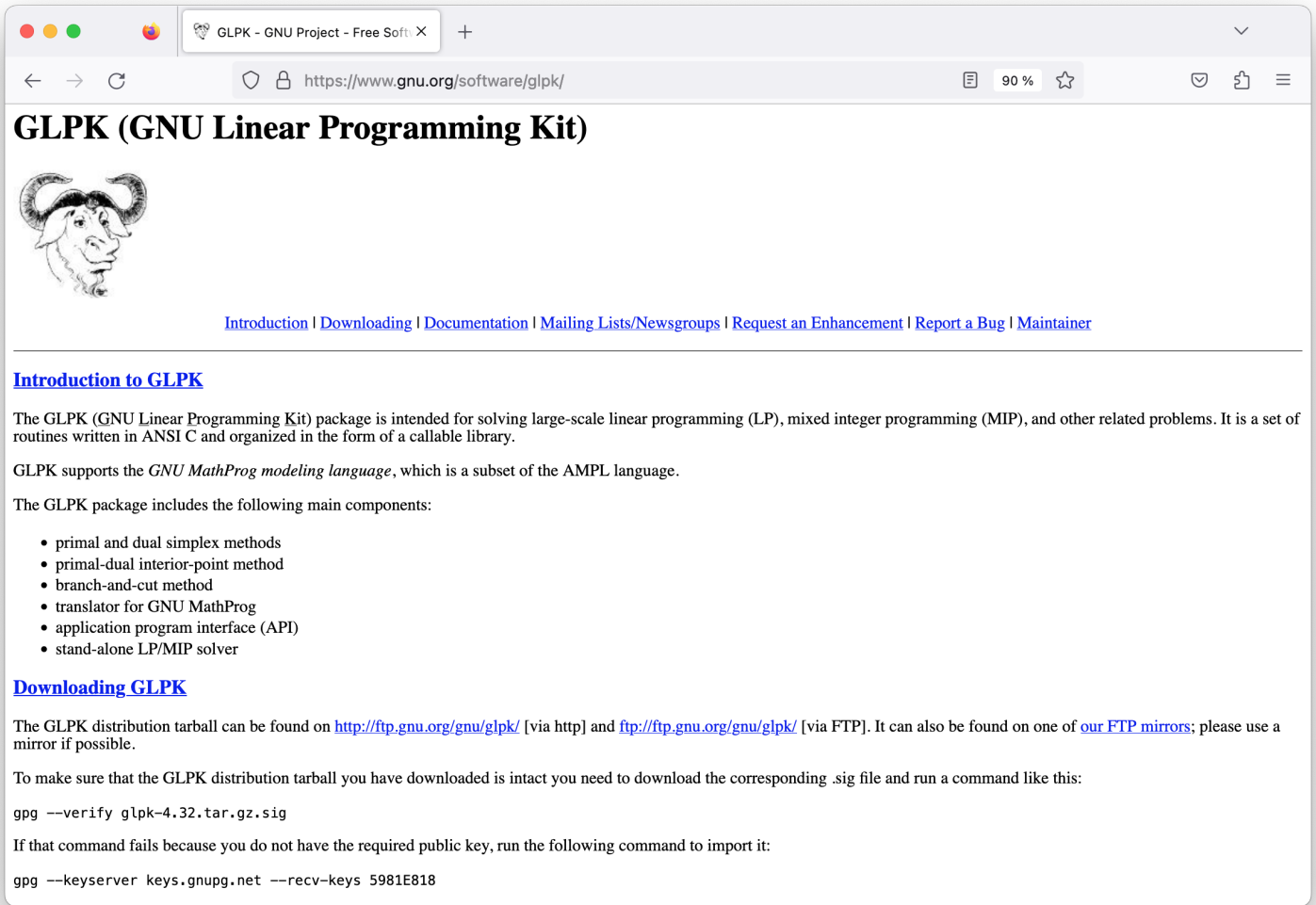
Editor: Visual Studio Code

`https://code.visualstudio.com/`

extended for Julia

Open Source MILP solver: GLPK

`https://www.gnu.org/software/glpk/`



Installing GLPK

On macOS

The easy way to install GLPK with one of these two package manager:

- homebrew
  - install homebrew: `https://brew.sh/`
  - install GLPK: `https://formulae.brew.sh/formula/glpk`
- macport
  - install macport: `https://www.macports.org/`
  - install GLPK: `https://ports.macports.org/port/glpk/`

On windows

Follow the instructions given here:

- `http://winglpk.sourceforge.net/`

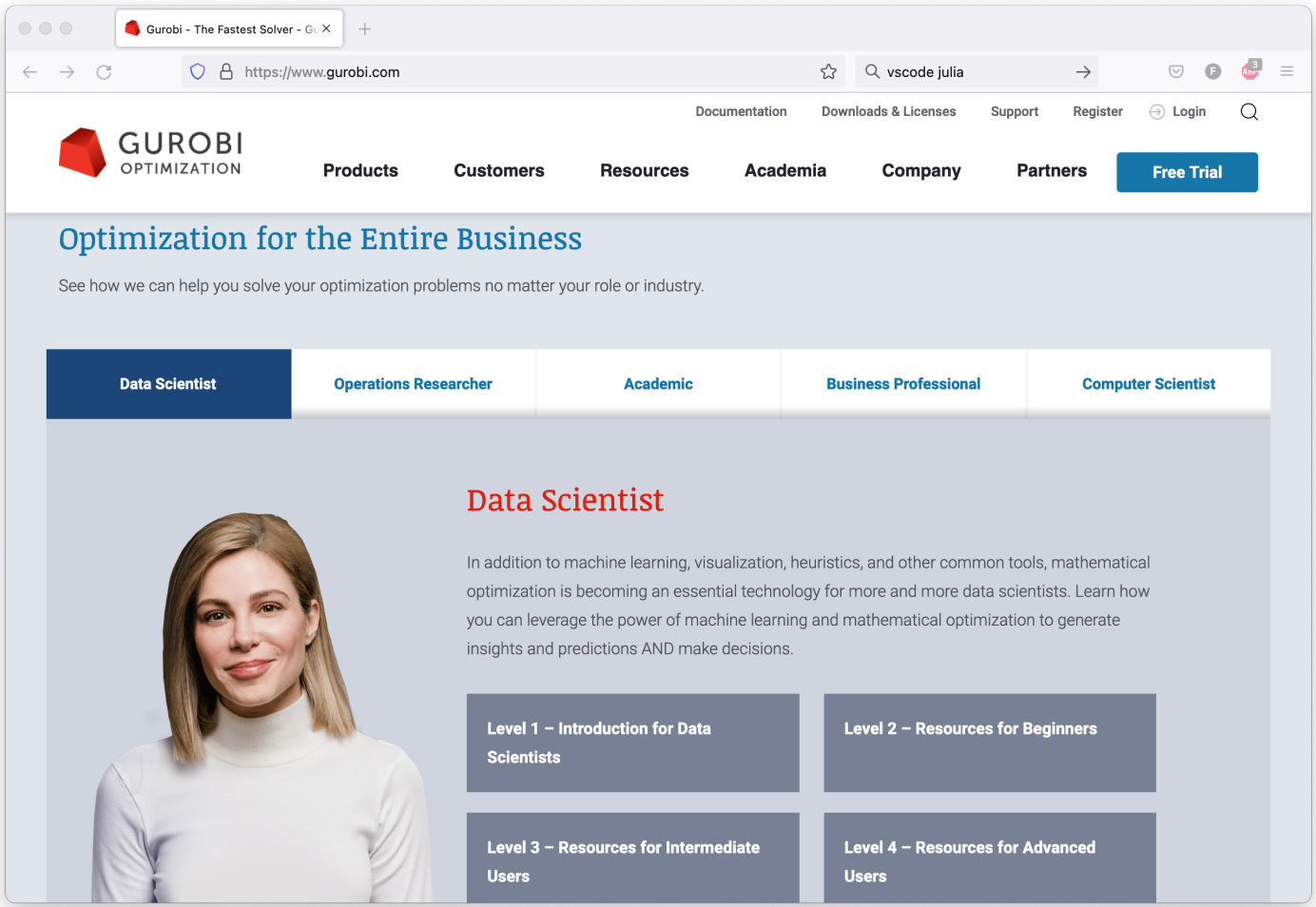
On linux (ubuntu)

Follow the instructions given here:

- `https://howtoinstall.co/en/glpk`

Commercial MILP solver: Gurobi

`https://www.gurobi.com/`



Installing Gurobi (academic license)

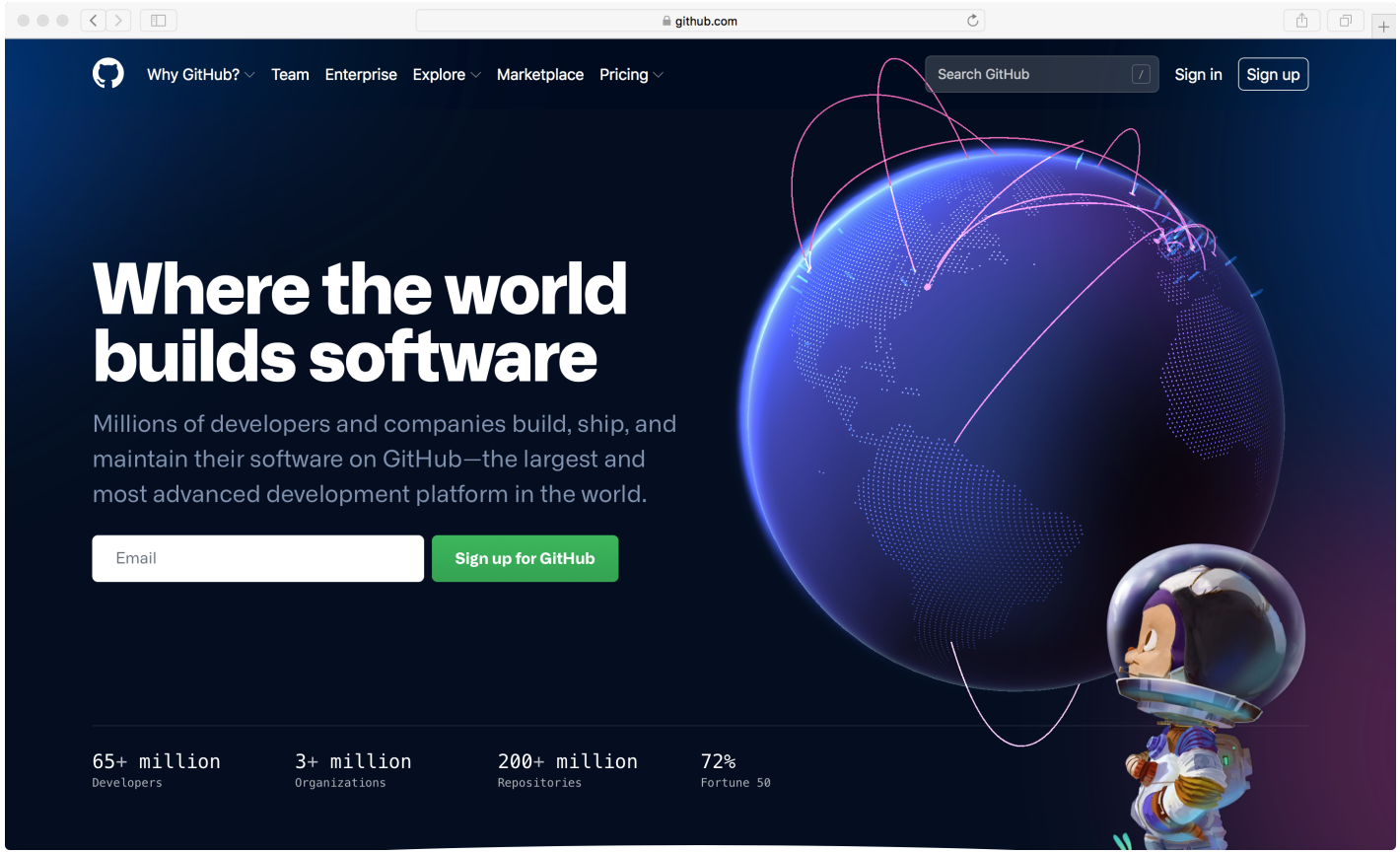
Gurobi software downloads and license center:
`https://www.gurobi.com/downloads/`

Video for

- macOS:
  `https://www.youtube.com/watch?v=ZcL-NmckTxQ`
- windows:
  `https://www.youtube.com/watch?v=fQVxuWOiPpI&t=0s`
- linux (ubuntu):
  `https://www.youtube.com/watch?v=yNmeG6Wom1o&t=0s`

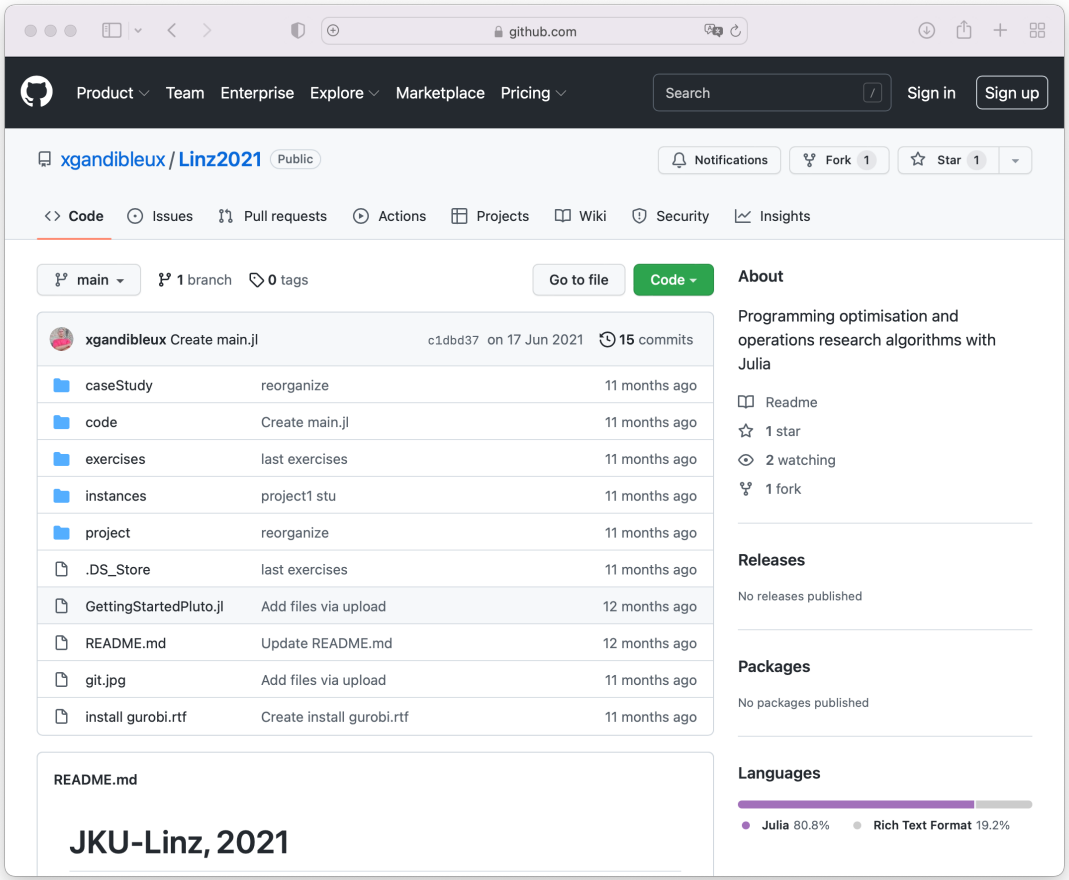
Repository: GitHub and the app GitHub Desktop

`https://github.com` and `https://desktop.github.com`



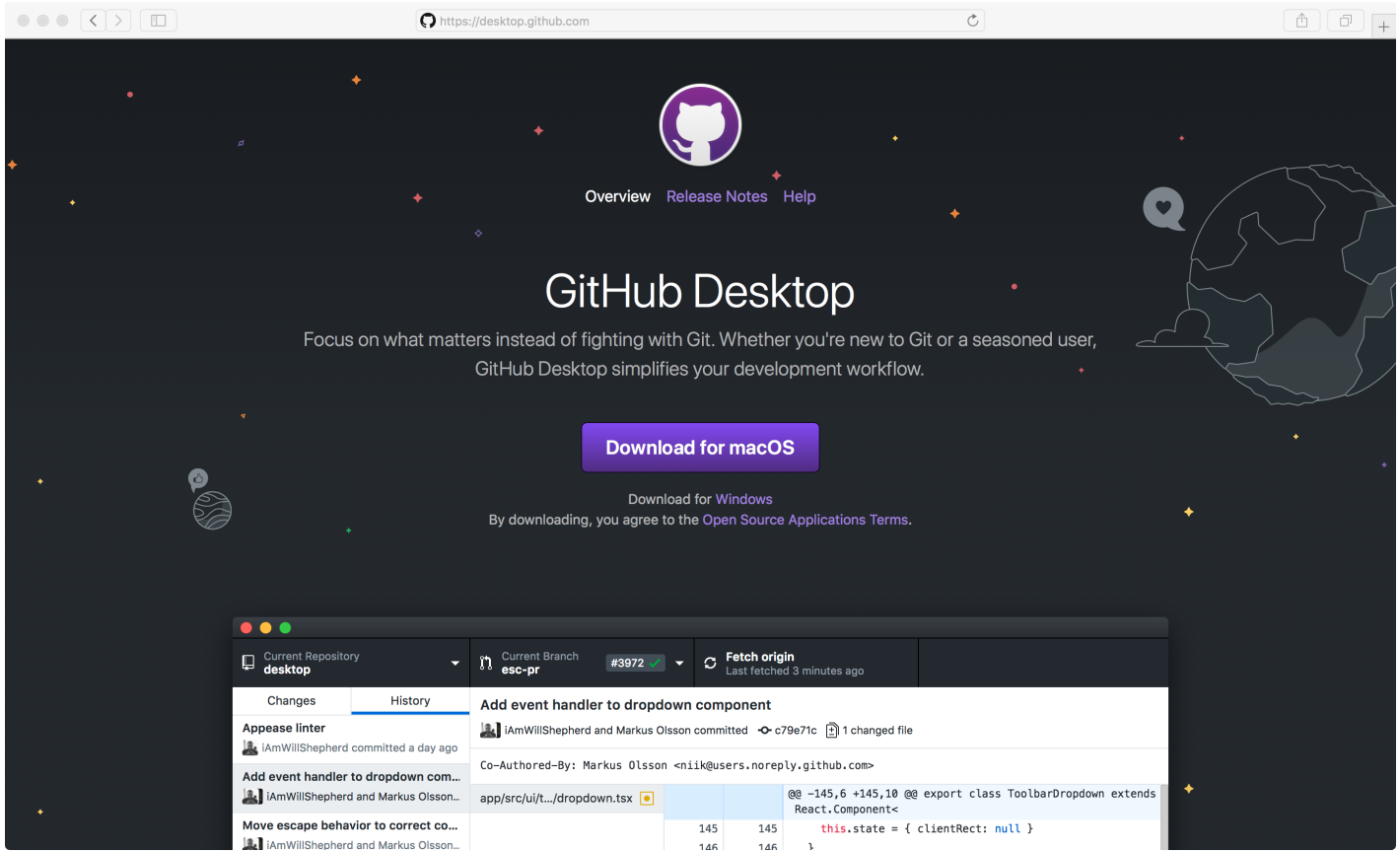
Repository: GitHub and the app GitHub Desktop

`https://github.com` and `https://desktop.github.com`



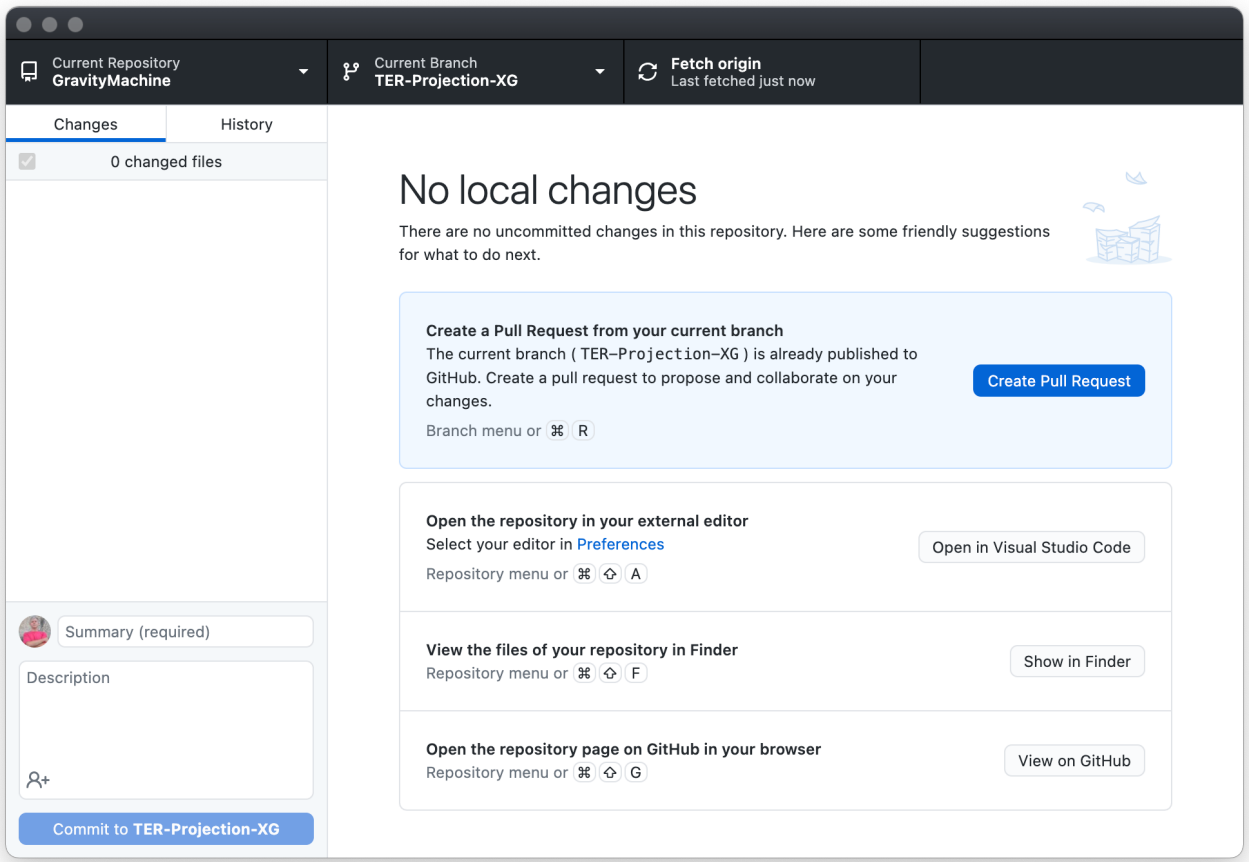
Repository: GitHub and the app GitHub Desktop

`https://github.com` and `https://desktop.github.com`



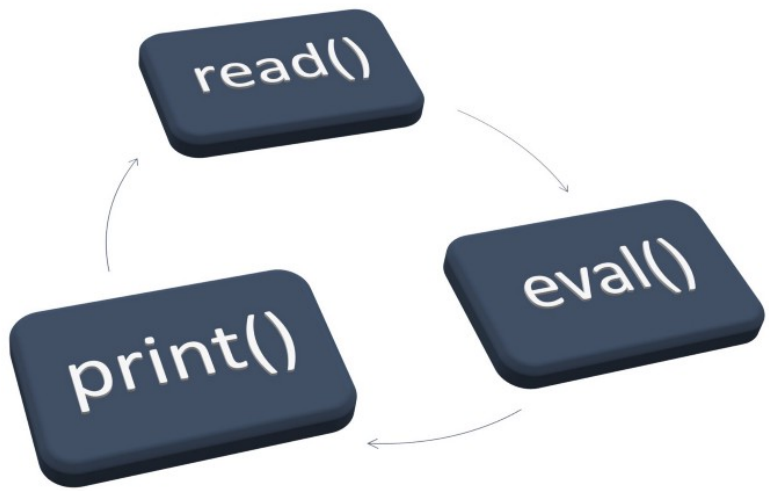
Repository: GitHub and the app GitHub Desktop

`https://github.com` and `https://desktop.github.com`



Getting started

Read Evaluate Print Loop



REPL: normal mode

Just click on the app icon

REPL: normal mode (commands)

Some commands to know:

Load (and run) a program	`include("filename.jl")`
Interrupt execution	`[Ctrl] + [C]`
Recall last result	`ans`
Previous-run command	`Up` key
Search backwards into history	`[Ctrl] + [r]`
Tab-completion	characters `+ [Tab]`
Clear screen	`[Ctrl] + [L]`
Exit REPL	`exit()` or `[Ctrl] + [D]`

REPL: normal mode (commands)

Some commands to know:

Load (and run) a program	`include("filename.jl")`
Interrupt execution	`[Ctrl] + [C]`
Recall last result	`ans`
Previous-run command	`Up` key
Search backwards into history	`[Ctrl] + [r]`
Tab-completion	characters `+ [Tab]`
Clear screen	`[Ctrl] + [L]`
Exit REPL	`exit()` or `[Ctrl] + [D]`

REPL: normal mode (commands)

Some commands to know:

Load (and run) a program	`include("filename.jl")`
Interrupt execution	`[Ctrl] + [C]`
Recall last result	`ans`
Previous-run command	`Up` key
Search backwards into history	`[Ctrl] + [r]`
Tab-completion	characters + `[Tab]`
Clear screen	`[Ctrl] + [L]`
Exit REPL	`exit()` or `[Ctrl] + [D]`

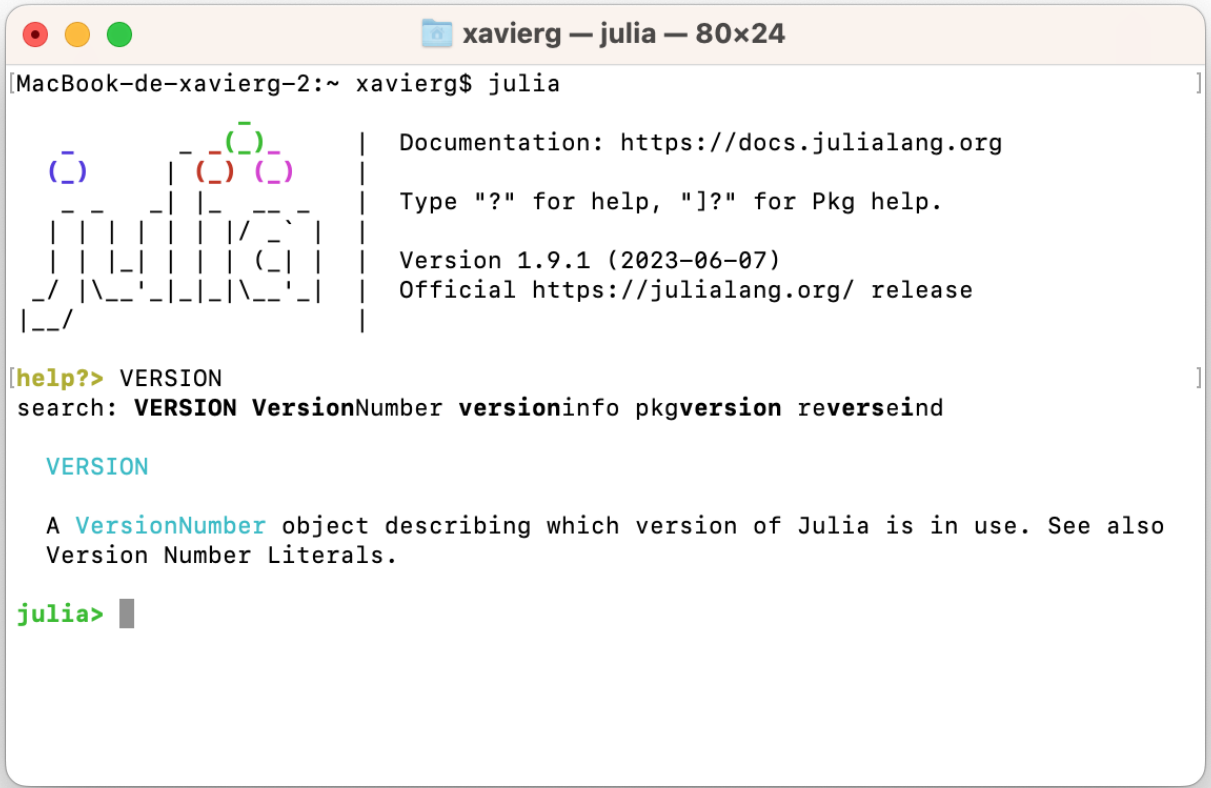
REPL: normal mode (commands)

Some commands to know:

Load (and run) a program	`include("filename.jl")`
Interrupt execution	`[Ctrl] + [C]`
Recall last result	`ans`
Previous-run command	`Up` key
Search backwards into history	`[Ctrl] + [r]`
Tab-completion	characters `+ [Tab]`
Clear screen	`[Ctrl] + [L]`
Exit REPL	`exit()` or `[Ctrl] + [D]`

REPL: help mode

```
julia> ?
```



REPL: shell mode

```
julia> ;
```

Return to normal mode: `[Backspace]` on empty line

REPL: package manager mode

```
julia> ]
```

Return to normal mode: [Backspace] on empty line

REPL: package manager mode (commands)

Some commands to know:

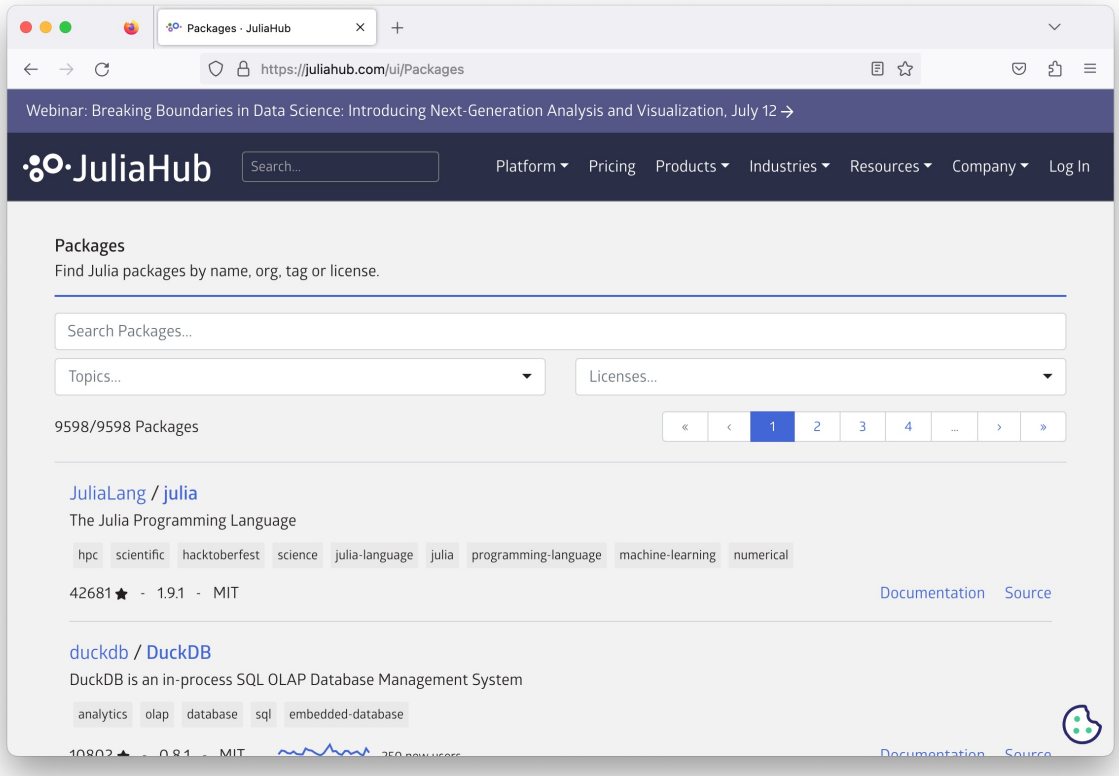
add package(s) to project	`add`
remove package(s) from project	`remove` or `rm`
summarize contents	`status` or `st`
update packages	`update` or `up`
help online	`help`

Packages

Packages

Visual interface for exploring the packages:

Julia hub `https://juliahub.com/ui/Packages`



Alternative:

Julia Packages https://juliapackages.com/

Packages

A selection of useful packages:

Base library:

`Pkg`	primitives for managing packages
`Random`	primitives for random number
`LinearAlgebra`	primitives for linear algebra
`Printf`	primitives for C-like format to display on screen

External library:

`PyPlot`	Plotting for Julia based on matplotlib
`Plots`	a high-level plotting package
`JuMP`	Modeling language for Mathematical Optimization
`GLPK`	interface for GLPK solver
`Gurobi`	interface for Gurobi solver
`CPLEX`	interface for the CPLEX solver
`vOptGeneric`	Multiobjective linear optimization solver; generic problems
`vOptSpecific`	Multiobjective linear optimization solver; specific problems

Packages

A selection of useful packages:

Base library:

`Pkg`	primitives for managing packages
`Random`	primitives for random number
`LinearAlgebra`	primitives for linear algebra
`Printf`	primitives for C-like format to display on screen

External library:

`PyPlot`	Plotting for Julia based on matplotlib
`Plots`	a high-level plotting package
`JuMP`	Modeling language for Mathematical Optimization
`GLPK`	interface for GLPK solver
`Gurobi`	interface for Gurobi solver
`CPLEX`	interface for the CPLEX solver
`vOptGeneric`	Multiobjective linear optimization solver; generic problems
`vOptSpecific`	Multiobjective linear optimization solver; specific problems

Packages

A selection of useful packages:

Base library:

`Pkg`	primitives for managing packages
`Random`	primitives for random number
`LinearAlgebra`	primitives for linear algebra
`Printf`	primitives for C-like format to display on screen

External library:

`PyPlot`	Plotting for Julia based on matplotlib
`Plots`	a high-level plotting package
`JuMP`	Modeling language for Mathematical Optimization
`GLPK`	interface for GLPK solver
`Gurobi`	interface for Gurobi solver
`CPLEX`	interface for the CPLEX solver
`vOptGeneric`	Multiobjective linear optimization solver; generic problems
`vOptSpecific`	Multiobjective linear optimization solver; specific problems

Adding a package (from the external library)

Commands to invoque **once** before the first use of the given package:

Method 1:

```
(@v1.9) pkg> add PyPlot
(@v1.9) pkg> add JuMP,GLPK
```

Method 2:

```
julia> using Pkg
julia> Pkg.add("PyPlot")
julia> Pkg.add("JuMP")
julia> Pkg.add("GLPK")
```

Adding a package (from the external library)

Commands to invoque **once** before the first use of the given package:

Method 1:

```
(@v1.9) pkg> add PyPlot
(@v1.9) pkg> add JuMP,GLPK
```

Method 2:

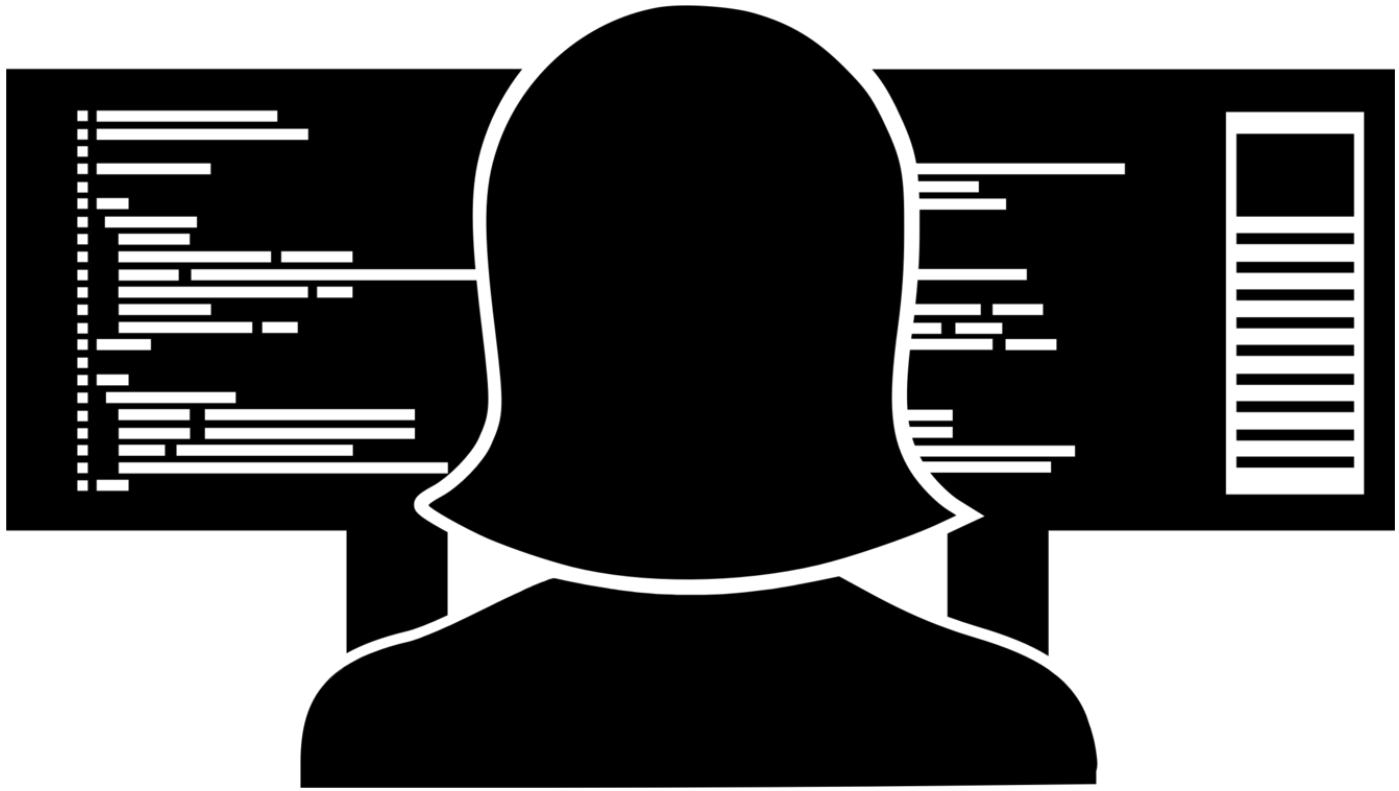
```
julia> using Pkg
julia> Pkg.add("PyPlot")
julia> Pkg.add("JuMP")
julia> Pkg.add("GLPK")
```

Using a package

Commands to invoque **each time** before the first use of the given package:

```
julia> using Pkg
julia> using JuMP,GLPK
julia> using Printf
julia> using LinearAlgebra
```

Let's try



Running instructions into the REPL

1) Click on the app icon...

... the REPL is ready!

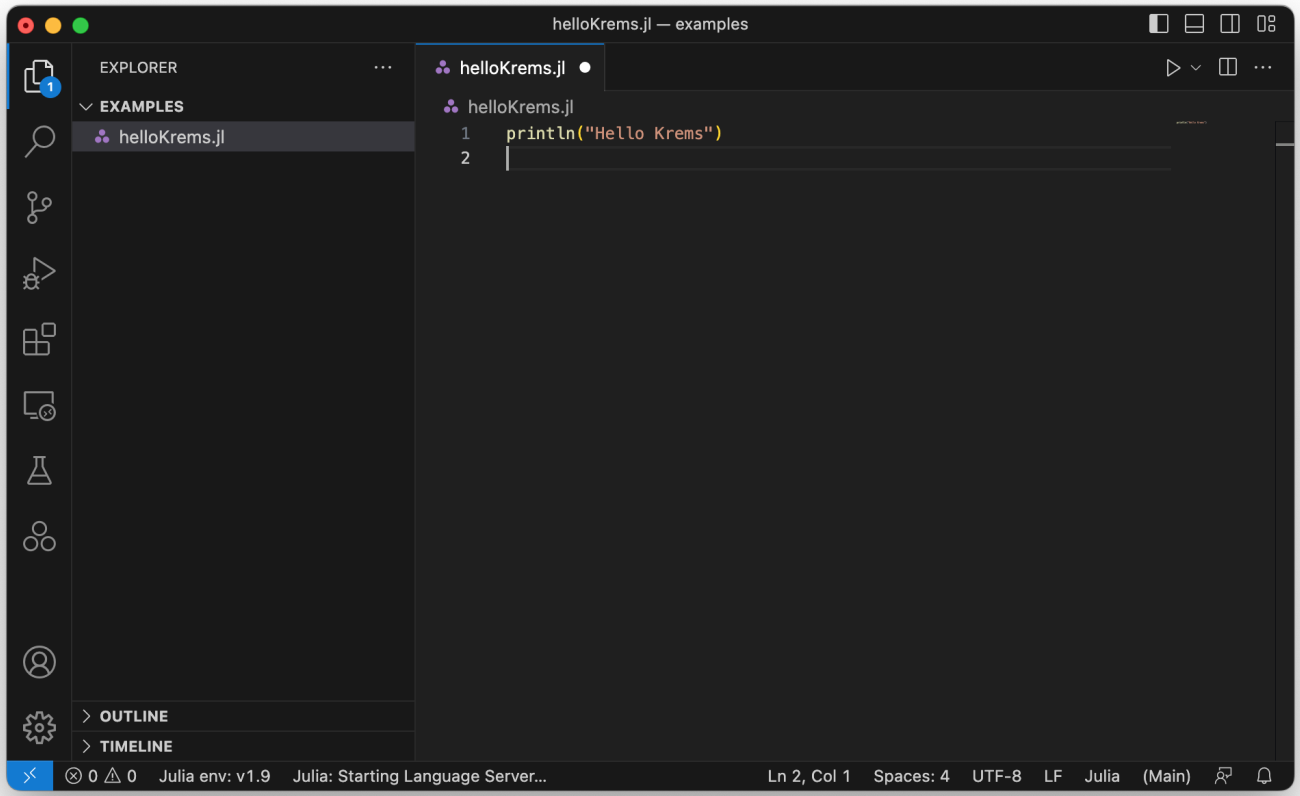
2) Type your instructions...

```
julia> println("Hello Krems")
```

... done!

Running a program into the REPL

Program prepared with a text editor (here VScode)...



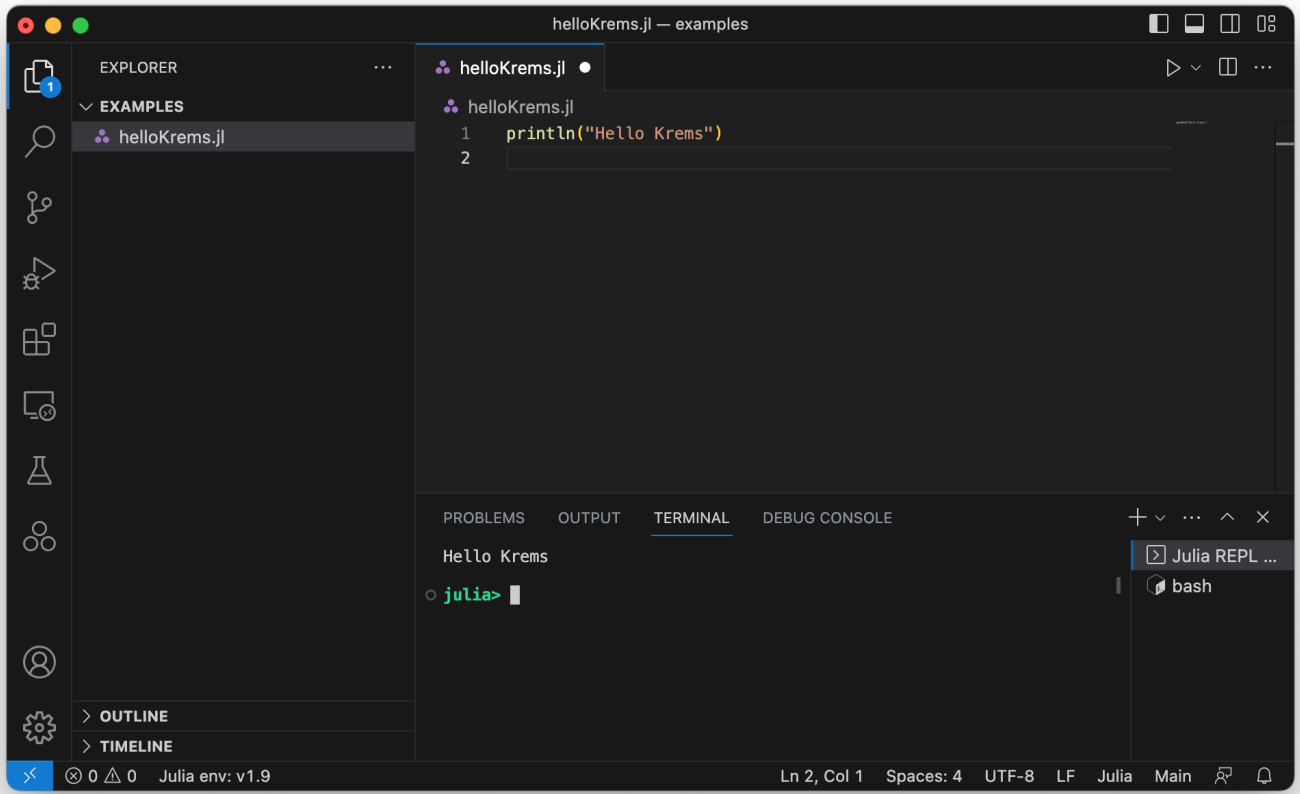
...to load and execute into the REPL :

```
julia> include("hello.jl")
```

Running a program into an IDE (VScode)

Program prepared with a text editor (here VScode)...

...to load and execute into the VScode terminal:



Running Julia code into a notebook (jupyter)

Jupyter `https://jupyter.org/`

`IJulia` has to be installed
(`https://github.com/JuliaLang/IJulia.jl`)

Steps to follow:

```
(@v1.9) pkg> add IJulia
```

```
julia> using IJulia
julia> notebook()
```

answer `yes` to the questions

Done!

Running Julia code into a notebook (jupyter)

Jupyter `https://jupyter.org/`

`IJulia` has to be installed
(`https://github.com/JuliaLang/IJulia.jl`)

Steps to follow:

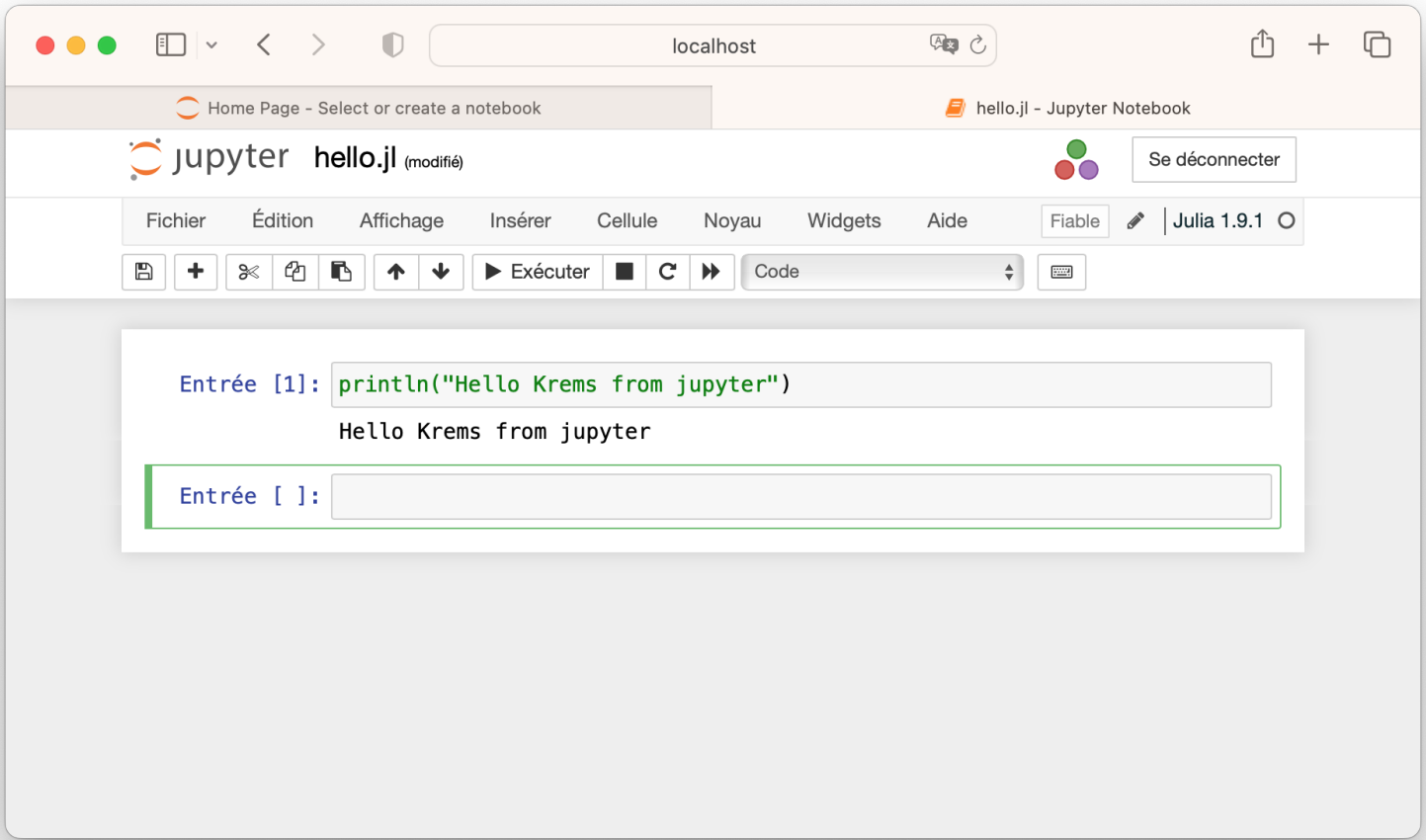
```
(@v1.9) pkg> add IJulia
```

```
julia> using IJulia
julia> notebook()
```

answer `yes` to the questions

Done!

Running Julia code into a notebook (jupyter)



Running Julia code into a notebook (Pluto)

Pluto `https://github.com/fonsp/Pluto.jl`

Install: `https://www.youtube.com/watch?v=C4QhZcX34mI`

Steps to follow:

```
(@v1.9) pkg> add Pluto
```

```
julia> using Pluto
julia> Pluto.run()
```

Done!

Running Julia code into a notebook (Pluto)

Pluto `https://github.com/fonsp/Pluto.jl`

Install: `https://www.youtube.com/watch?v=C4QhZcX34mI`

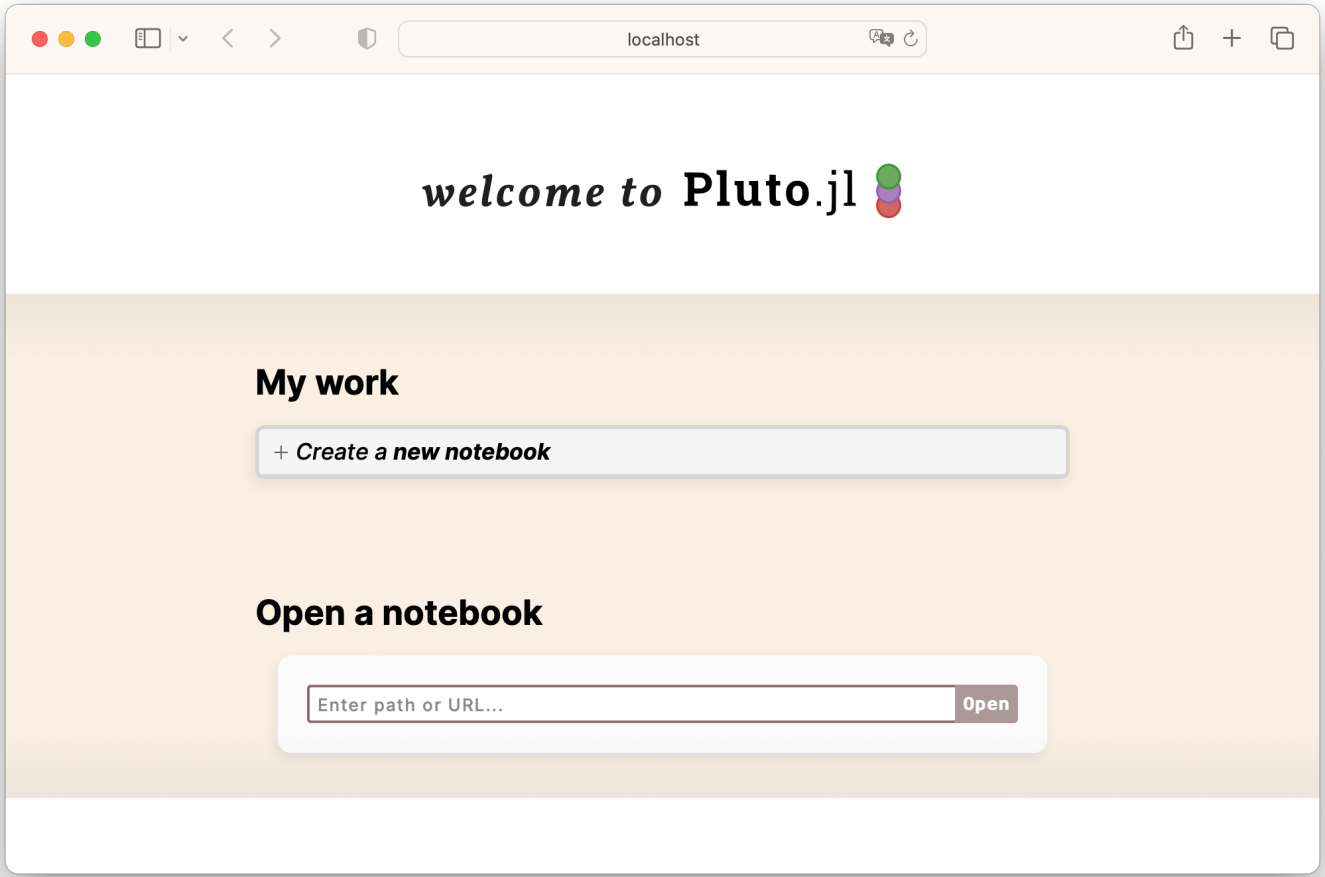
Steps to follow:

```
(@v1.9) pkg> add Pluto
```

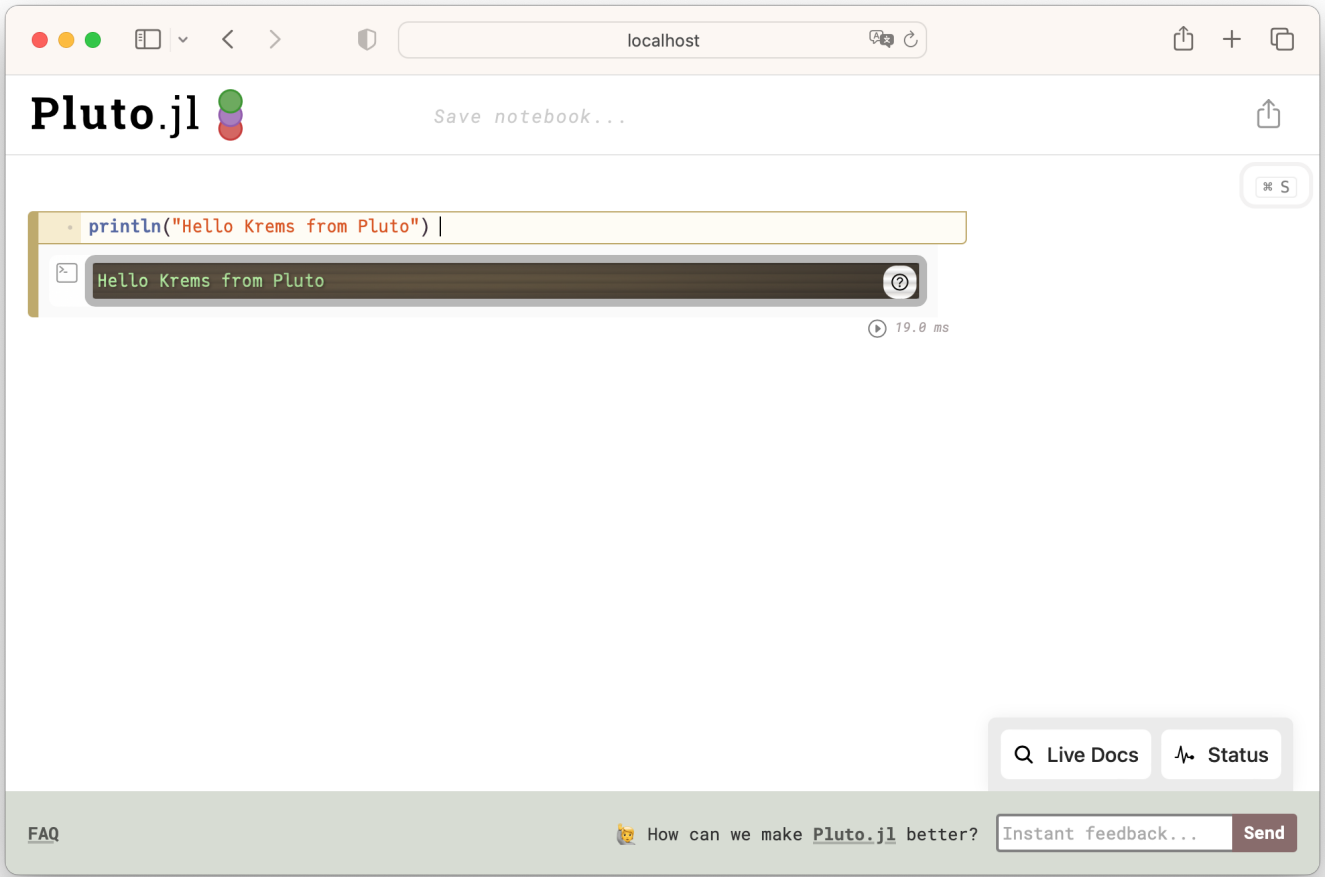
```
julia> using Pluto
julia> Pluto.run()
```

Done!

Running Julia code into a notebook (Pluto)



Running Julia code into a notebook (2)



Let’s go for installing the apps and do your first runs