

Programming

optimisation and operations research algorithms with Julia
for Business Tasks

Prof. Dr. Xavier Gandibleux

Université de Nantes
Département Informatique – Faculté des Sciences et Techniques
France

Lesson 1 – May-June 2022



version 1.7.x and later

Why Julia?

Discovered in 2016 (v0.4.5), it became my favourite language:

1. Runs like C, walks like Python, talks like Lisp

- Same speed as C or FORTRAN code
- Development dynamic like writing python code
- Less object-oriented more function-oriented (like Lisp)

⇒ High-level and high-performance!

⇒ Extremely good ratio between coding time and running time.

⇒ Familiar for the users of Matlab, Fortran, Python, C/C++, etc.

2. Additional value

- Open source: MIT license, source code available on GitHub
- Multi-platform (x86, ARM, Apple M-series -experimental-, etc.)
- Cross-language support: C, FORTRAN, python, R ... from Julia
- Rich set of packages: Optimisation, Plotting, Visualisation, Statistics, Data science, GPU, etc.

⇒ Ideal for the development of efficient open-source app

Why Julia?

Discovered in 2016 (v0.4.5), it became my favourite language:

1. Runs like C, walks like Python, talks like Lisp

- Same speed as C or FORTRAN code
- Development dynamic like writing python code
- Less object-oriented more function-oriented (like Lisp)

⇒ High-level and high-performance!

⇒ Extremely good ratio between coding time and running time.

⇒ Familiar for the users of Matlab, Fortran, Python, C/C++, etc.

2. Additional value

- Open source: MIT license, source code available on GitHub
- Multi-platform (x86, ARM, Apple M-series -experimental-, etc.)
- Cross-language support: C, FORTRAN, python, R ... from Julia
- Rich set of packages: Optimisation, Plotting, Visualisation, Statistics, Data science, GPU, etc.

⇒ Ideal for the development of efficient open-source app

Why Julia?

Discovered in 2016 (v0.4.5), it became my favourite language:

1. Runs like C, walks like Python, talks like Lisp

- Same speed as C or FORTRAN code
- Development dynamic like writing python code
- Less object-oriented more function-oriented (like Lisp)

⇒ High-level and high-performance!

⇒ Extremely good ratio between coding time and running time.

⇒ Familiar for the users of Matlab, Fortran, Python, C/C++, etc.

2. Additional value

- Open source: MIT license, source code available on GitHub
- Multi-platform (x86, ARM, Apple M-series -experimental-, etc.)
- Cross-language support: C, FORTRAN, python, R ... from Julia
- Rich set of packages: Optimisation, Plotting, Visualisation, Statistics, Data science, GPU, etc.

⇒ Ideal for the development of efficient open-source app

Why Julia?

Discovered in 2016 (v0.4.5), it became my favourite language:

1. Runs like C, walks like Python, talks like Lisp

- Same speed as C or FORTRAN code
- Development dynamic like writing python code
- Less object-oriented more function-oriented (like Lisp)

⇒ High-level and high-performance!

⇒ Extremely good ratio between coding time and running time.

⇒ Familiar for the users of Matlab, Fortran, Python, C/C++, etc.

2. Additional value

- Open source: MIT license, source code available on GitHub
- Multi-platform (x86, ARM, Apple M-series -experimental-, etc.)
- Cross-language support: C, FORTRAN, python, R ... from Julia
- Rich set of packages: Optimisation, Plotting, Visualisation, Statistics, Data science, GPU, etc.

⇒ Ideal for the development of efficient open-source app



A high-level, high-performance programming language for scientific computing <https://julialang.org/>

- ▶ Timeline:
 - 2009...2012 @ MIT, USA
 - version 1.0.0 in 2018; version 1.7.2 (February 6, 2022)
- ▶ Communities:
 - Over 7,400 Julia packages <https://juliahub.com/ui/Packages>
 - Optimization <https://jump.dev/>
- ▶ JuliaCon...ferences:
 - every year since 2014 <https://juliacon.org/>
 - 27th to 29th of July, 2022: free, online and everywhere
- ▶ Structures:
 - JuliaComputing <http://juliacomputing.com>
 - JuliaLab <http://julia.mit.edu>

Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B. Shah: Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59: 65–98. 2017.



A high-level, high-performance programming language for scientific computing <https://julialang.org/>

- ▶ Timeline:
 - 2009...2012 @ MIT, USA
 - version 1.0.0 in 2018; version 1.7.2 (February 6, 2022)
- ▶ Communities:
 - Over 7,400 Julia packages <https://juliahub.com/ui/Packages>
 - Optimization <https://jump.dev/>
- ▶ JuliaCon...ferences:
 - every year since 2014 <https://juliacon.org/>
 - 27th to 29th of July, 2022: free, online and everywhere
- ▶ Structures:
 - JuliaComputing <http://juliacomputing.com>
 - JuliaLab <http://julia.mit.edu>

Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B. Shah: Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59: 65–98. 2017.



A high-level, high-performance programming language for scientific computing <https://julialang.org/>

- ▶ Timeline:
 - 2009...2012 @ MIT, USA
 - version 1.0.0 in 2018; version 1.7.2 (February 6, 2022)
- ▶ Communities:
 - Over 7,400 Julia packages <https://juliahub.com/ui/Packages>
 - Optimization <https://jump.dev/>
- ▶ JuliaCon...ferences:
 - every year since 2014 <https://juliacon.org/>
 - 27th to 29th of July, 2022: free, online and everywhere
- ▶ Structures:
 - JuliaComputing <http://juliacomputing.com>
 - JuliaLab <http://julia.mit.edu>

Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B. Shah: Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59: 65–98. 2017.



A high-level, high-performance programming language
for scientific computing <https://julialang.org/>

- ▶ Timeline:
 - 2009...2012 @ MIT, USA
 - version 1.0.0 in 2018; version 1.7.2 (February 6, 2022)
- ▶ Communities:
 - Over 7,400 Julia packages <https://juliahub.com/ui/Packages>
 - Optimization <https://jump.dev/>
- ▶ JuliaCon...ferences:
 - every year since 2014 <https://juliacon.org/>
 - 27th to 29th of July, 2022: free, online and everywhere
- ▶ Structures:
 - JuliaComputing <http://juliacomputing.com>
 - JuliaLab <http://julia.mit.edu>

Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B. Shah: Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59: 65–98. 2017.

Key concepts

- ▶ Julia has an LLVM-based Just-in-time compilation (JIT)
- ▶ Compilation down to native code (running on CPU and GPU, ...)
- ▶ Julia is dynamically typed, provides multiple dispatch
- ▶ Julia has many built-in mathematical functions
- ▶ A built-in package manager
- ▶ Designed for parallelism and distributed computation

Resources

Some URL to know:

- ▶ Official website

<https://julialang.org/>

- ▶ Official documentation (online and pdf)

<https://docs.julialang.org/>

- ▶ Julia channel on *youtube*

https://www.youtube.com/channel/UC9IuUwwE2xdjQUT_LMLONoA/featured

- ▶ Questions and discussion on *discourse*

<https://discourse.julialang.org/>

- ▶ Book online « Think Julia: How to Think Like a Computer Scientist »

<https://benlauwens.github.io/ThinkJulia.jl/latest/book.html>

- ▶ Julia (v1.6.3) on *replit*

<https://replit.com/languages/julia>

Resources

Some URL to know:

- ▶ Official website

<https://julialang.org/>

- ▶ Official documentation (online and pdf)

<https://docs.julialang.org/>

- ▶ Julia channel on *youtube*

https://www.youtube.com/channel/UC9IuUwwE2xdjQUT_LMLONoA/featured

- ▶ Questions and discussion on *discourse*

<https://discourse.julialang.org/>

- ▶ Book online « Think Julia: How to Think Like a Computer Scientist »

<https://benlauwens.github.io/ThinkJulia.jl/latest/book.html>

- ▶ Julia (v1.6.3) on *replit*

<https://replit.com/languages/julia>



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 1.7.2**
- ▶ Editor: **Visual Studio Code (VSCode)**, Emacs, etc.

Apps to install for optimisation needs:

- ▶ Open-source MILP solver: **GLPK**, COIN-OR, etc.
- ▶ Commercial MILP solver: **Gurobi**, Cplex, etc.

Apps used (installation not mandatory but recommended):

- ▶ 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 1.7.2**
- ▶ Editor: **Visual Studio Code (VSCode)**, Emacs, etc.

Apps to install for optimisation needs:

- ▶ Open-source MILP solver: **GLPK**, COIN-OR, etc.
- ▶ Commercial MILP solver: **Gurobi**, Cplex, etc.

Apps used (installation not mandatory but recommended):

- ▶ 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 1.7.2**
- ▶ Editor: **Visual Studio Code (VSCode)**, Emacs, etc.

Apps to install for optimisation needs:

- ▶ Open-source MILP solver: **GLPK**, COIN-OR, etc.
- ▶ Commercial MILP solver: **Gurobi**, Cplex, etc.

Apps used (installation not mandatory but recommended):

- ▶ 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 1.7.2**
- ▶ Editor: **Visual Studio Code (VSCode)**, Emacs, etc.

Apps to install for optimisation needs:

- ▶ Open-source MILP solver: **GLPK**, COIN-OR, etc.
- ▶ Commercial MILP solver: **Gurobi**, Cplex, etc.

Apps used (installation not mandatory but recommended):

- ▶ interacting with repository: **GitHub Desktop**

Language: Julia 1.7.2

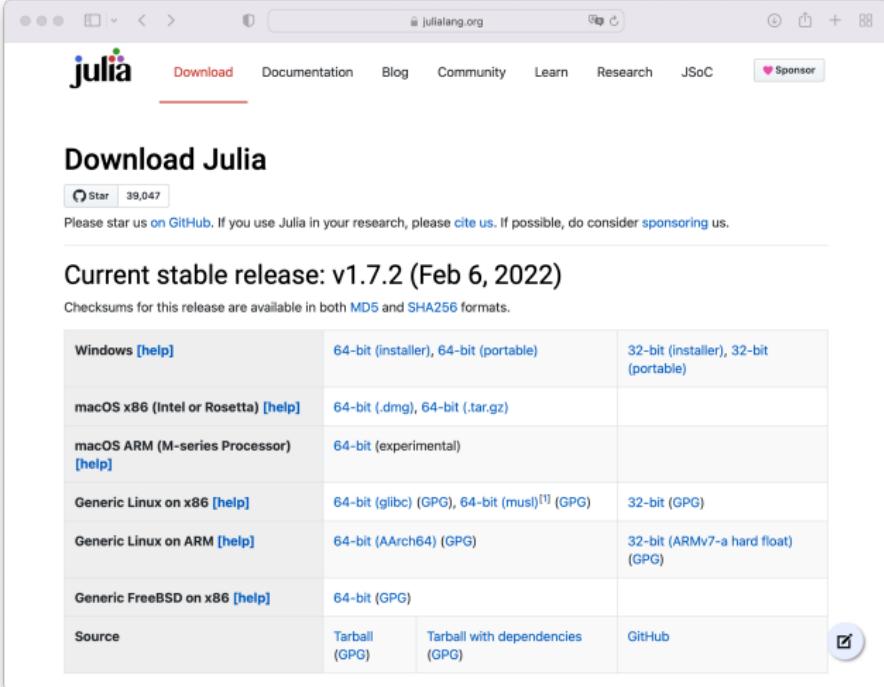
<https://julialang.org/>

The screenshot shows the official website for Julia 1.7.2. The header features the Julia logo and navigation links for Download, Documentation, Blog, Community, Learn, Research, JSOC, and a Sponsor button. Below the header is a large banner with a colorful geometric pattern and the text "The Julia Programming Language". Underneath the banner are two buttons: "Download" (green) and "Documentation" (blue). To the right of the "Download" button is a star icon and the number "39,048". The main content area is titled "Julia in a Nutshell" and contains six sections arranged in a 2x3 grid:

- Fast**: Julia was designed from the beginning for [high performance](#). Julia programs compile to efficient native code for [multiple platforms](#) via LLVM.
- Dynamic**: Julia is [dynamically typed](#), feels like a scripting language, and has good support for [interactive use](#).
- Reproducible**: [Reproducible environments](#) make it possible to recreate the same Julia environment every time, across platforms, with [pre-built binaries](#).
- Composable**: Julia uses [multiple dispatch](#) as a paradigm, making it easy to express many object-oriented and functional programming patterns. The talk on the [Unreasonable Effectiveness of Multiple Dispatch](#) explains why it works so well.
- General**: Julia provides [asynchronous I/O](#), [metaprogramming](#), [debugging](#), [logging](#), [profiling](#), a [package manager](#), and more. One can build entire [Applications](#) and [Microservices](#) in Julia.
- Open source**: Julia is an open source project with over 1,000 contributors. It is made available under the [MIT license](#). The [source code](#) is available on GitHub.

Language: Julia 1.7.2

<https://julialang.org/downloads/>

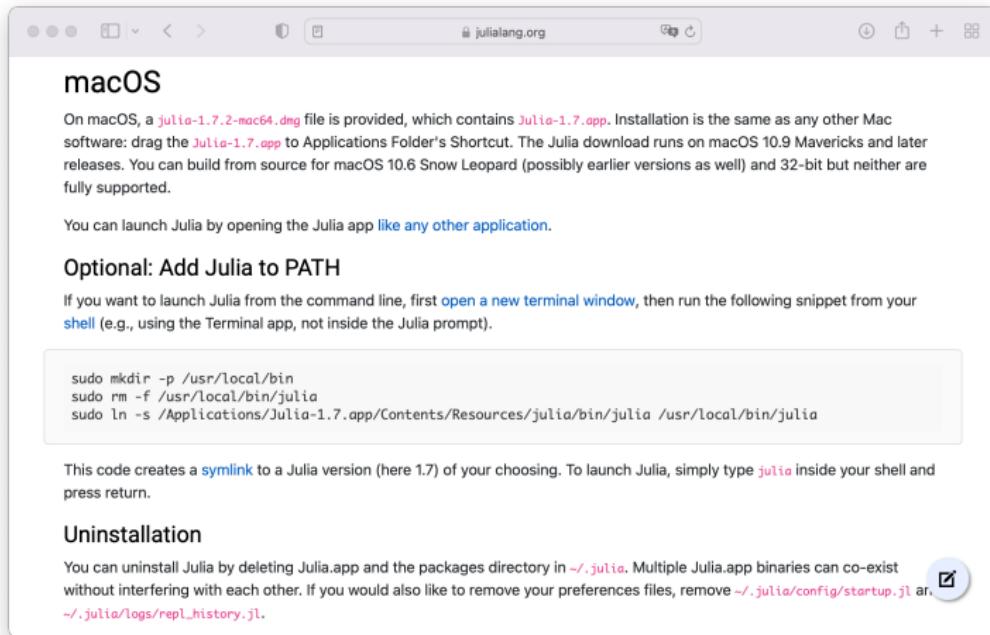


The screenshot shows a web browser window displaying the Julia website at julialang.org. The page title is "Download Julia". A "Star" button with the number 39,047 is visible. Below it, a message encourages users to star the project on GitHub, cite it, and consider sponsoring. The main heading is "Current stable release: v1.7.2 (Feb 6, 2022)". A table provides download links for various platforms:

Platform	Architectures	Notes	
Windows [help]	64-bit (installer), 64-bit (portable)	32-bit (installer), 32-bit (portable)	
macOS x86 (Intel or Rosetta) [help]	64-bit (.dmg), 64-bit (.tar.gz)		
macOS ARM (M-series Processor) [help]	64-bit (experimental)		
Generic Linux on x86 [help]	64-bit (glibc) (GPG), 64-bit (musl) ^[1] (GPG)	32-bit (GPG)	
Generic Linux on ARM [help]	64-bit (AArch64) (GPG)	32-bit (ARMv7-a hard float) (GPG)	
Generic FreeBSD on x86 [help]	64-bit (GPG)		
Source	Tarball (GPG)	Tarball with dependencies (GPG)	GitHub

Language: Julia 1.7.2

<https://julialang.org/downloads/>



The screenshot shows a web browser window with the URL julialang.org in the address bar. The main content is titled "macOS". It explains that a `julia-1.7.2-mac64.dmg` file is provided, which contains `Julia-1.7.app`. Installation is the same as any other Mac software: drag the `Julia-1.7.app` to Applications Folder's Shortcut. The Julia download runs on macOS 10.9 Mavericks and later releases. You can build from source for macOS 10.6 Snow Leopard (possibly earlier versions as well) and 32-bit but neither are fully supported.

You can launch Julia by opening the Julia app like any other application.

Optional: Add Julia to PATH

If you want to launch Julia from the command line, first open a new terminal window, then run the following snippet from your shell (e.g., using the Terminal app, not inside the Julia prompt).

```
sudo mkdir -p /usr/local/bin  
sudo rm -f /usr/local/bin/julia  
sudo ln -s /Applications/Julia-1.7.app/Contents/Resources/julia/bin/julia /usr/local/bin/julia
```

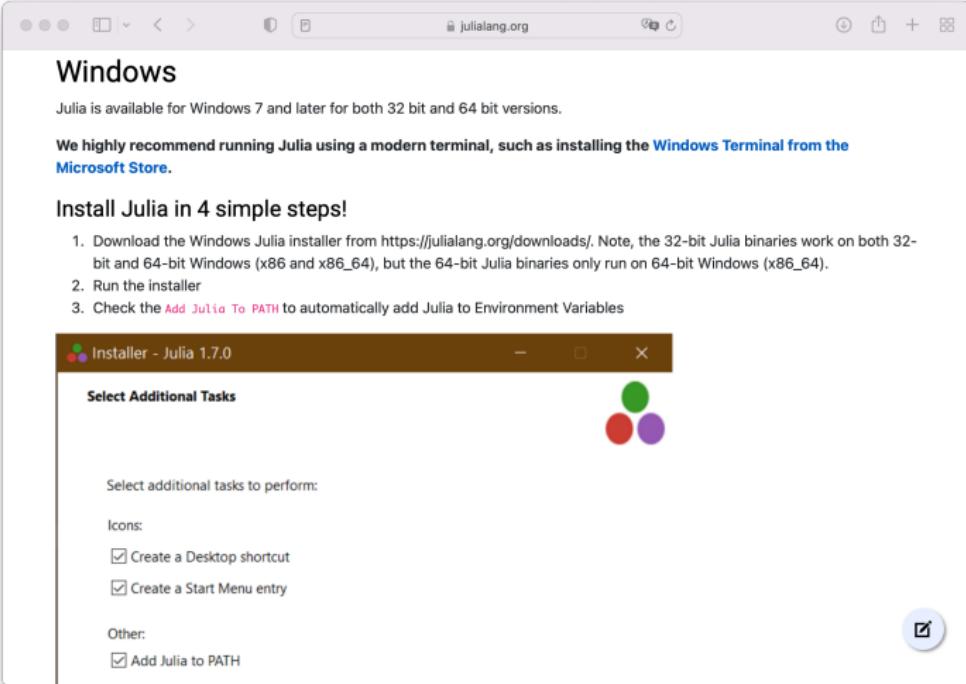
This code creates a symlink to a Julia version (here 1.7) of your choosing. To launch Julia, simply type `julia` inside your shell and press return.

Uninstallation

You can uninstall Julia by deleting Julia.app and the packages directory in `~/.julia`. Multiple Julia.app binaries can co-exist without interfering with each other. If you would also like to remove your preferences files, remove `~/.julia/config/startup.jl` and `~/.julia/logs/repl_history.jl`.

Language: Julia 1.7.2

<https://julialang.org/downloads/>



The screenshot shows a window titled "Installer - Julia 1.7.0". The main title bar has three colored dots (green, blue, red) on the left. Below the title bar, there's a dark header with the text "Select Additional Tasks" and a small decorative graphic of three overlapping circles (green, red, purple) centered below it. The main content area contains the following text: "Select additional tasks to perform:" followed by two sections: "Icons:" and "Other:". Under "Icons:", there are two checked checkboxes: "Create a Desktop shortcut" and "Create a Start Menu entry". Under "Other:", there is one checked checkbox: "Add Julia to PATH". In the bottom right corner of the content area, there is a circular button with a checkmark icon.

Windows

Julia is available for Windows 7 and later for both 32 bit and 64 bit versions.

We highly recommend running Julia using a modern terminal, such as installing the [Windows Terminal from the Microsoft Store](#).

Install Julia in 4 simple steps!

1. Download the Windows Julia installer from <https://julialang.org/downloads/>. Note, the 32-bit Julia binaries work on both 32-bit and 64-bit Windows (x86 and x86_64), but the 64-bit Julia binaries only run on 64-bit Windows (x86_64).
2. Run the installer
3. Check the [Add Julia To PATH](#) to automatically add Julia to Environment Variables

Language: Julia 1.7.2

<https://julialang.org/downloads/>

The screenshot shows a web browser window with the URL "julialang.org" in the address bar. The main content is titled "Linux and FreeBSD". It includes a note about using official generic binaries, a command box containing the wget and tar commands to download the binary, a section on running Julia, and instructions for adding Julia's bin folder to the PATH environment variable. A "Copy" button is visible in the bottom right corner of the code block.

Linux and FreeBSD

It is strongly recommended that the official generic binaries from the downloads page be used to install Julia on Linux and FreeBSD. The following set of commands downloads the latest version of Julia into a directory named `julia-1.7.2`.

```
wget https://julialang-s3.julialang.org/bin/linux/x64/1.7/julia-1.7.2-linux-x86_64.tar.gz  
tar zxvf julia-1.7.2-linux-x86_64.tar.gz
```

Running Julia

The generic Linux and FreeBSD binaries do not require any special installation steps, but you will need to ensure that your system can find the `julia` executable. The directory where Julia is installed is referred to as <Julia directory>.

To run Julia, you can do any of the following:

- Invoke the `julia` executable by using its full path: <Julia directory>/bin/julia
- Create a symbolic link to `julia` inside a folder which is on your system PATH
- Add Julia's `bin` folder (with full path) to your system PATH environment variable

To add Julia's `bin` folder (with full path) to PATH environment variable, you can edit the `~/.bashrc` (or `~/.bash_profile`) file. Open the file in your favourite editor and add a new line as follows:

```
export PATH="$PATH:/path/to/<Julia directory>/bin"
```

Apart from this, there are several ways through which you can change environment variable. You can follow [this guide](#) to find out a way convenient for you.

Julia installs all its files in a single directory. Deleting the directory where Julia was installed is sufficient. If you would also like to remove your packages, remove `~/.julia`. The startup file is at `~/.julia/config/startup.jl` and the history at `~/.julia/logs/repl_history.jl`.

Language: Julia 1.7.2

Just click on the app icon



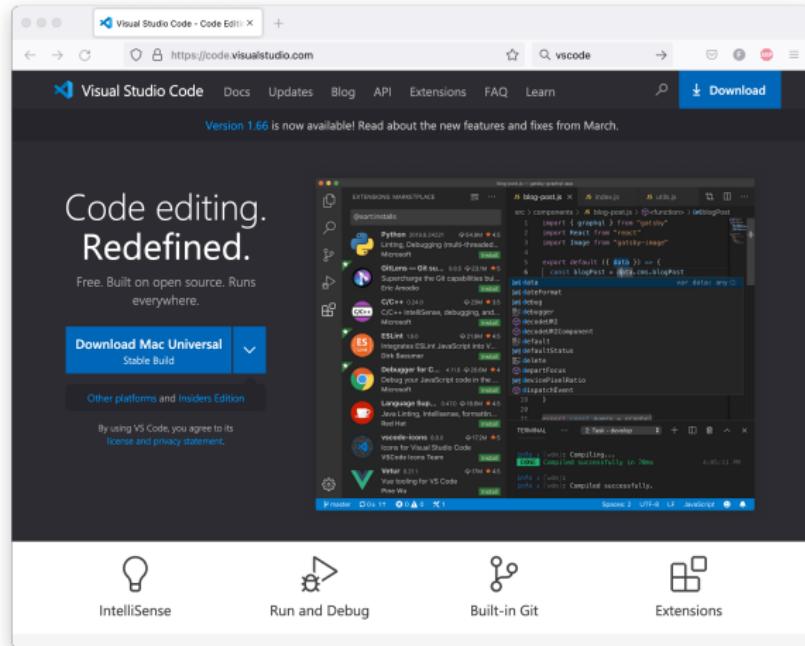
```
xgandibleux — julia — 80x24
Last login: Tue May  3 11:21:30 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Pro-de-Xavier:~ xgandibleux$ exec '/Applications/Julia-1.7.app/Contents/Resources/julia/bin/julia'

julia> ████ ████ ████ | Documentation: https://docs.julialang.org
      ████ ████ ████ | Type "?" for help, "]?" for Pkg help.
      ████ ████ ████ | Version 1.7.2 (2022-02-06)
      ████ ████ ████ | Official https://julialang.org/ release
      ████ ████ ████ |
```

Editor: Visual Studio Code

<https://code.visualstudio.com/>

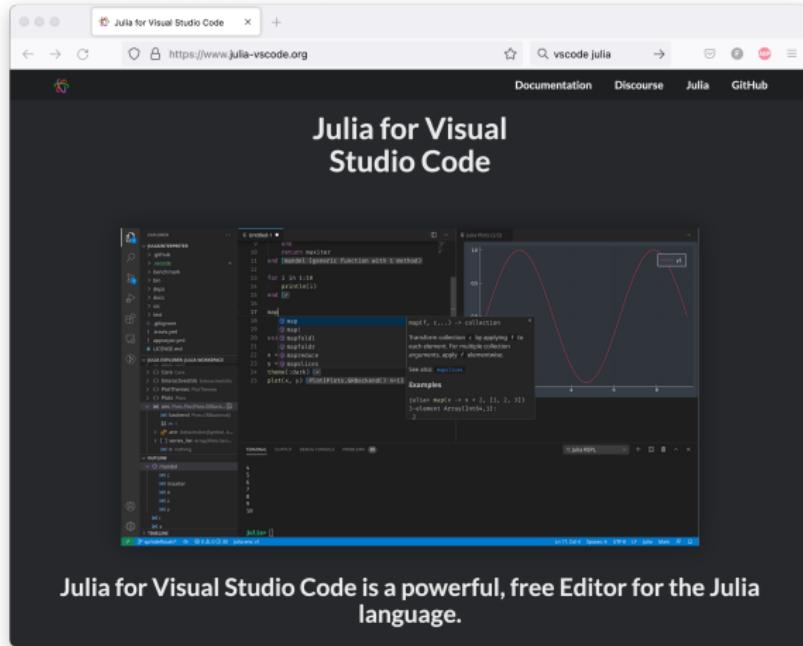


extended for Julia

<https://www.julia-vscode.org/>

Editor: Visual Studio Code

<https://code.visualstudio.com/>



Julia for Visual Studio Code is a powerful, free Editor for the Julia language.

extended for Julia

<https://www.julia-vscode.org/>

Editor: Visual Studio Code

<https://code.visualstudio.com/>

The screenshot shows the Visual Studio Code extension marketplace interface. On the left, there's a sidebar with various extension categories like 'Extensions: PLACE ...' and 'julia'. The main area is titled 'Extension : Julia' and shows the 'Julia' extension by 'julialang' version v1.6.17. It has a rating of 5 stars (70 reviews) and is described as 'Julia Language Support'. There are buttons for 'Désactiver' and 'Désinstaller'. A note says 'Cette extension est activée globalement.' Below this, there are tabs for 'Détails', 'Contributions', 'Journal des modifications', and 'État du runtime'. The 'Détails' tab shows the extension provides support for the Julia programming language. It also lists 'Build and Test' status as 'passing' and 'docs' as 'latest'. To the right, there are sections for 'Catégories' (Programming Languages, Snippets, Linters, Debuggers, Data Science, Notebooks, Machine Learning) and 'Ressources' (Place de marché, Dépôt, Licence, julia-vscode.org). At the bottom, there's information about the extension's publication date (21/06/2016), last update (07/04/2022), and a note about identifying the Julia language.

EXTENSIONS: PLACE ... Y U E ...

Extension : Julia

julia

Julia v1.6.17

julialang | 336 750 | ★★★★★ (70)

Julia Language Support

Désactiver Désinstaller

Cette extension est activée globalement.

Détails Contributions Journal des modifications État du runtime

Julia

Build and Test passing docs latest

This VS Code extension provides support for the [Julia](#) programming language.

Getting started

Installing Julia/VS Code/VS Code Julia extension

1. Install Julia for your platform: <https://julialang.org/downloads/>
2. Install VS Code for your platform: <https://code.visualstudio.com/download> At the end of this step you should be able to start VS Code.
3. Choose Install in the [VS Code Marketplace](#); or manually install with:
 1. Start VS Code.
 2. Inside VS Code, go to the extensions view either by executing the View: Show Extensions command (click View->Command Palette...) or by clicking on the extension icon on the left side of the VS Code window.
 3. In the extensions view, simply search for the term **julia** in the marketplace search box, then select the extension named **Julia** and click the install button. You might have to restart VS Code after this step.

Configure the Julia extension

If you have installed Julia into a standard location on Mac or Windows, or if the Julia binary is on your PATH, the Julia VS Code extension should automatically find your Julia installation.

Catégories

Programming Languages
Snippets Linters
Debuggers
Data Science
Notebooks
Machine Learning

Ressources

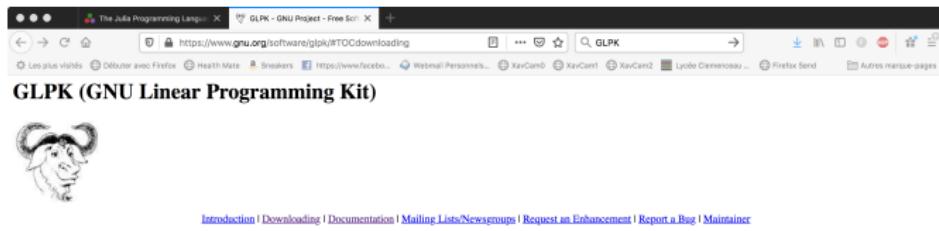
Place de marché
Dépôt
Licence
julia-vscode.org

Informations sur la Place de marché

Publié le 21/06/2016, 23:56:01
Dernière 07/04/2022, mise à jour 19:16:44
Identifier [julialang.language-julia](#)

Open Source MILP solver: GLPK

<https://www.gnu.org/software/glpk/>



The screenshot shows a Firefox browser window with the URL <https://www.gnu.org/software/glpk/> in the address bar. The page content is the GLPK (GNU Linear Programming Kit) homepage. At the top left is a cartoon illustration of a goat's head with a large, curved horn. Below the illustration is a horizontal menu bar with the following links: Introduction | Downloading | Documentation | Mailing Lists/Newsgroups | Request an Enhancement | Report a Bug | Maintainer.

Introduction to GLPK

The GLPK (GNU Linear Programming Kit) package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library.

GLPK supports the *GNU MathProg modeling language*, which is a subset of the AMPL language.

The GLPK package includes the following main components:

- primal and dual simplex methods
- primal-dual interior-point method
- branch-and-cut method
- translator for GNU MathProg
- application program interface (API)
- stand-alone LP/MIP solver

Downloading GLPK

The GLPK distribution tarball can be found on <http://ftp.gnu.org/gnu/glpk/> [via http] and <ftp://ftp.gnu.org/gnu/glpk/> [via FTP]. It can also be found on one of [our PTP mirrors](#); please use a mirror if possible.

To make sure that the GLPK distribution tarball you have downloaded is intact you need to download the corresponding .sig file and run a command like this:

```
gpg --verify glpk-4.32.tar.gz.sig
```

If that command fails because you do not have the required public key, run the following command to import it:

```
gpg --keyserver keys.gnupg.net --recv-keys 59818818
```

and then re-run the previous command.

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/>

The screenshot shows the Gurobi website homepage. At the top, there is a navigation bar with links for Documentation, Downloads & Licenses, Support, Register, Login, and a Free Trial button. Below the navigation bar, there is a main heading "Optimization for the Entire Business" and a sub-section for "Data Scientist". On the left side of the main content area, there is a portrait of a woman identified as a Data Scientist. To the right of the portrait, the heading "Data Scientist" is displayed in red. Below this heading, there is a paragraph of text explaining the role of mathematical optimization for data scientists. At the bottom of the main content area, there are four rectangular boxes labeled "Level 1 – Introduction for Data Scientists", "Level 2 – Resources for Beginners", "Level 3 – Resources for Intermediate Users", and "Level 4 – Resources for Advanced Users".

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:

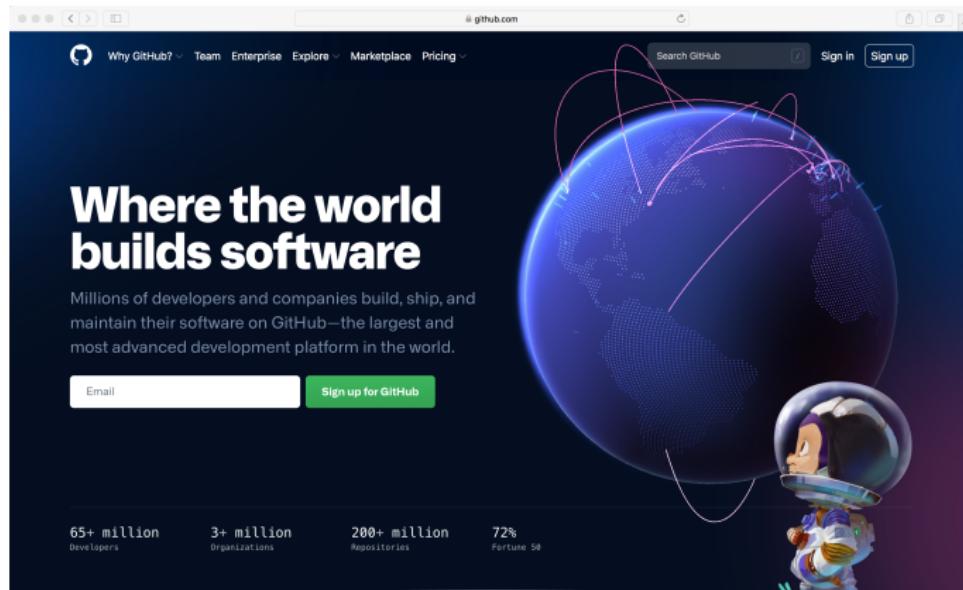
<https://www.youtube.com/watch?v=fQVxuW0iPpI&t=0s>

- ▶ linux (ubuntu):

<https://www.youtube.com/watch?v=yNmeG6Wom1o&t=0s>

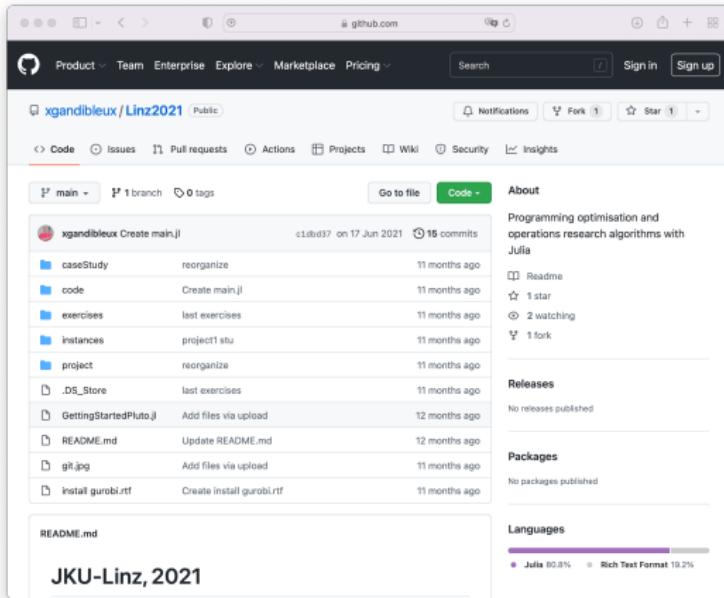
Repository: GitHub and the app GitHub Desktop

<https://github.com> and <https://desktop.github.com>



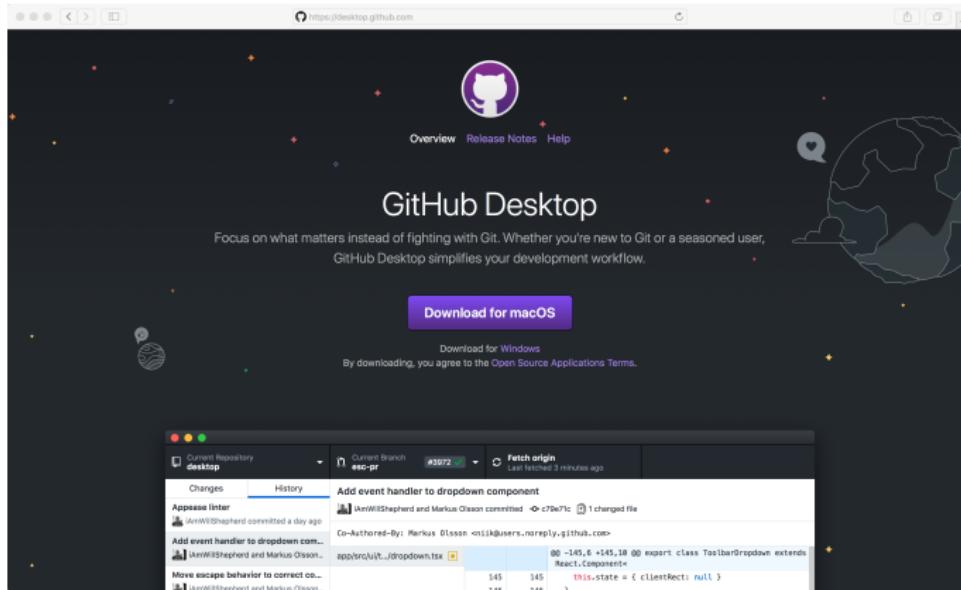
Repository: GitHub and the app GitHub Desktop

<https://github.com> and <https://desktop.github.com>



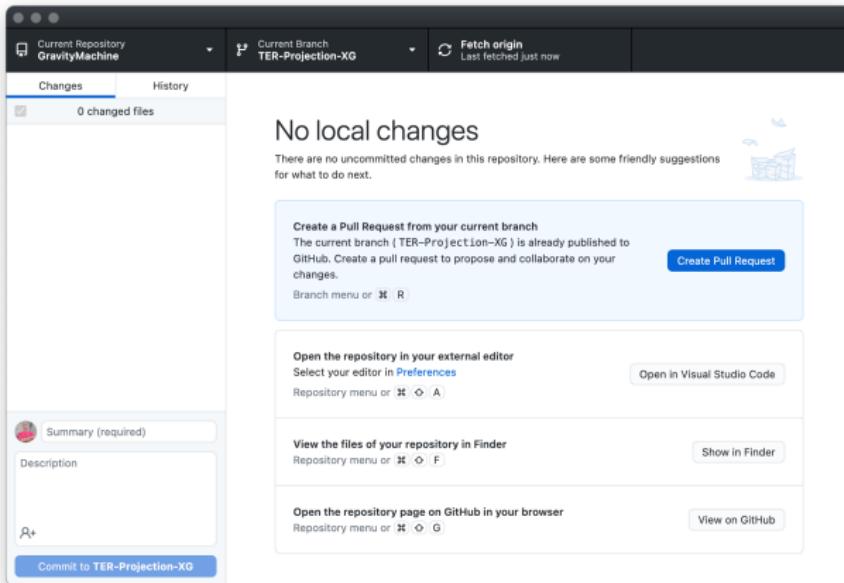
Repository: GitHub and the app GitHub Desktop

<https://github.com> and <https://desktop.github.com>



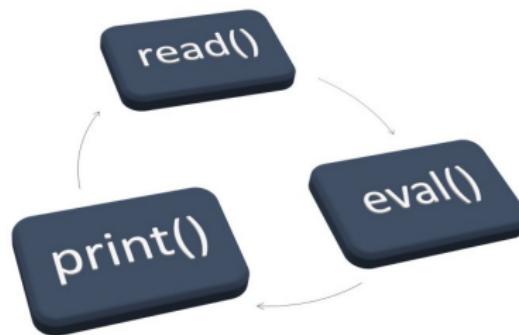
Repository: GitHub and the app GitHub Desktop

<https://github.com> and <https://desktop.github.com>





Getting started
R ead E valuate P rint L oop



REPL: normal mode

Just click on the app icon



```
xgandibleux — julia — 80x24
Last login: Tue May  3 11:21:30 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Pro-de-Xavier:~ xgandibleux$ exec '/Applications/Julia-1.7.app/Contents/Resources/julia/bin/julia'

julia> □
```

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> ?
```

```
xgandibleux — julia — 80x24

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Pro-de-Xavier:~ xgandibleux$ exec '/Applications/Julia-1.7.app/Contents/Resources/julia/bin/julia'

Documentation: https://docs.julialang.org
Type "?" for help, "]?" for Pkg help.
Version 1.7.2 (2022-02-06)
Official https://julialang.org/ release

[help?> VERSION
search: VERSION VersionNumber versioninfo reverseind

VERSION
A VersionNumber object describing which version of Julia is in use. For
details see Version Number Literals.

julia> ]
```

REPL: shell mode

```
julia> ;
```

```
xgandibleux — julia — 80x24
Last login: Wed May  4 14:07:17 on ttys003

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Pro-de-Xavier:~ xgandibleux$ exec '/Applications/Julia-1.7.app/Contents/Resources/julia/bin/julia'

          _ _ _ _ _ | Documentation: https://docs.julialang.org
  _ _ _ _ _ | Type "?" for help, "]??" for Pkg help.
  _ _ _ _ _ | Version 1.7.2 (2022-02-06)
  _ _ _ _ _ | Official https://julialang.org/ release
|_ _/_ _ _ _ | 

[shell> pwd
/Users/xgandibleux

shell> 
```

Return to normal mode: [Backspace] on empty line

REPL: package manager mode

julia>]

The terminal window shows the following text:

```
Last login: Wed May 4 14:08:17 on ttys003

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Pro-de-Xavier:~ xgandibleux$ exec '/Applications/Julia-1.7.app/Contents/Resources/julia/bin/julia'

(@v1.7) pkg>
```

Documentation: <https://docs.julialang.org>
Type "?" for help, "]?" for Pkg help.
Version 1.7.2 (2022-02-06)
Official <https://julialang.org/> release

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



Getting started

Packages



Packages

Visual interface for exploring the packages:

Julia hub

<https://juliahub.com/ui/Packages>

The screenshot shows the JuliaHub Packages interface. At the top, there's a search bar labeled "Search..." and dropdown menus for "Topics..." and "Licenses...". Below the search bar, it says "7559/7559 Packages" and shows a navigation bar with pages 1 through 5. The main content area displays three package cards:

- JuliaLang / julia**: The Julia Programming Language. Description: "The Julia Programming Language". Tags: machine-learning, programming-language, julia-language, julia, scientific, hpc, numerical, science. Stats: 39070 stars, 1.6.1 version, MIT license. Links: Documentation, Source.
- fonsp / Pluto**: Simple reactive notebooks for Julia. Description: "Simple reactive notebooks for Julia". Tags: pluto-notebooks, julia, reactive, notebook, interactive, exploration, visualization, education, designed-for-teachers. Stats: 3843 stars, 0.19.3 version, MIT license. Links: Documentation, Source.
- FluxML / Flux**: Relax! Flux is the ML library that doesn't make you tensor. Description: "Relax! Flux is the ML library that doesn't make you tensor". Tags: hacktoberfest2021, data-science, deep-learning, neural-networks, machine-learning, hacktoberfest, the-human-brain, flux. Stats: 6.7 new users.

Alternative:

Julia Packages

<https://juliapackages.com/>

Packages

A selection of useful packages:

Standard 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:

Standard 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:

Standard 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 invoke **once** before the first use of the given package:

Method 1:

```
(@v1.7) pkg> add PyPlot  
(@v1.7) 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 invoke **once** before the first use of the given package:

Method 1:

```
(@v1.7) pkg> add PyPlot  
(@v1.7) pkg> add JuMP,GLPK
```

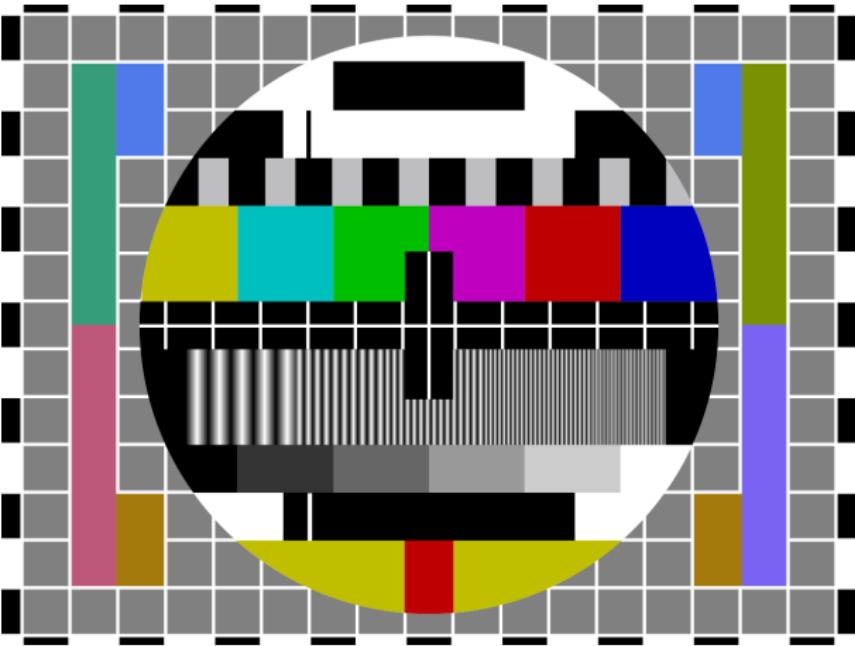
Method 2:

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

Using a package

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

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



Getting started

Let's try



Running instructions into the REPL

1) Click on the app icon...



... the REPL is ready!

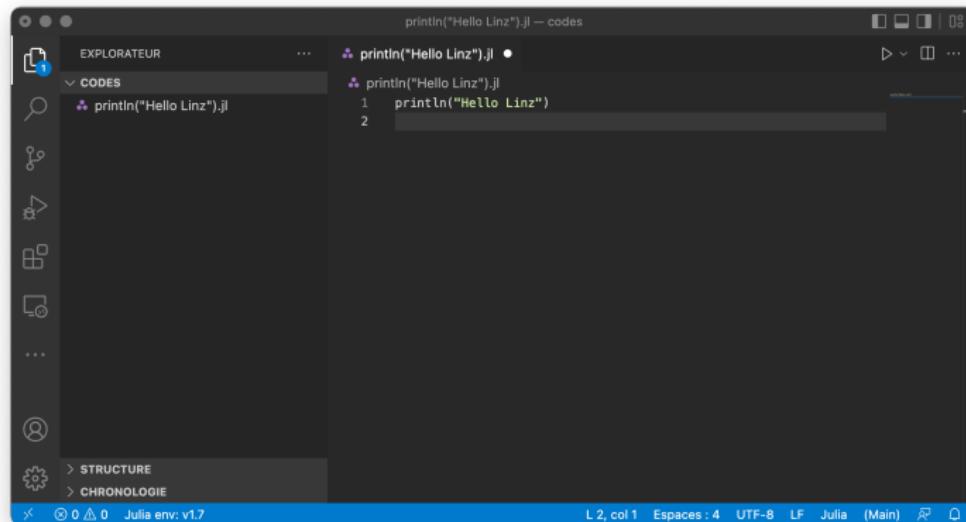
2) Type your instructions...

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

... 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:

The screenshot shows the Visual Studio Code (VScode) interface. On the left is the Explorer sidebar with a tree view showing a folder named 'CODES' containing a file named 'println("Hello Linz").jl'. The main workspace shows the code content:

```
println("Hello Linz").jl — codes
1 println("Hello Linz")
2
```

Below the workspace are several tabs: PROBLÈMES, SORTIE, CONSOLE DE DÉBOGAGE, and TERMINAL. The TERMINAL tab is active, displaying the output of the code execution:

```
Hello Linz
julia>
```

To the right of the terminal is a sidebar with two entries: 'bash' and 'Julia REPL'. The status bar at the bottom of the screen shows the following information:

Y 0 ▲ 0 Julia env: v1.7 L 2, col 1 Espaces : 4 UTF-8 LF Julia Main ⌂ ⌂

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:

```
(@v1.7) 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:

```
(@v1.7) pkg> add IJulia
```

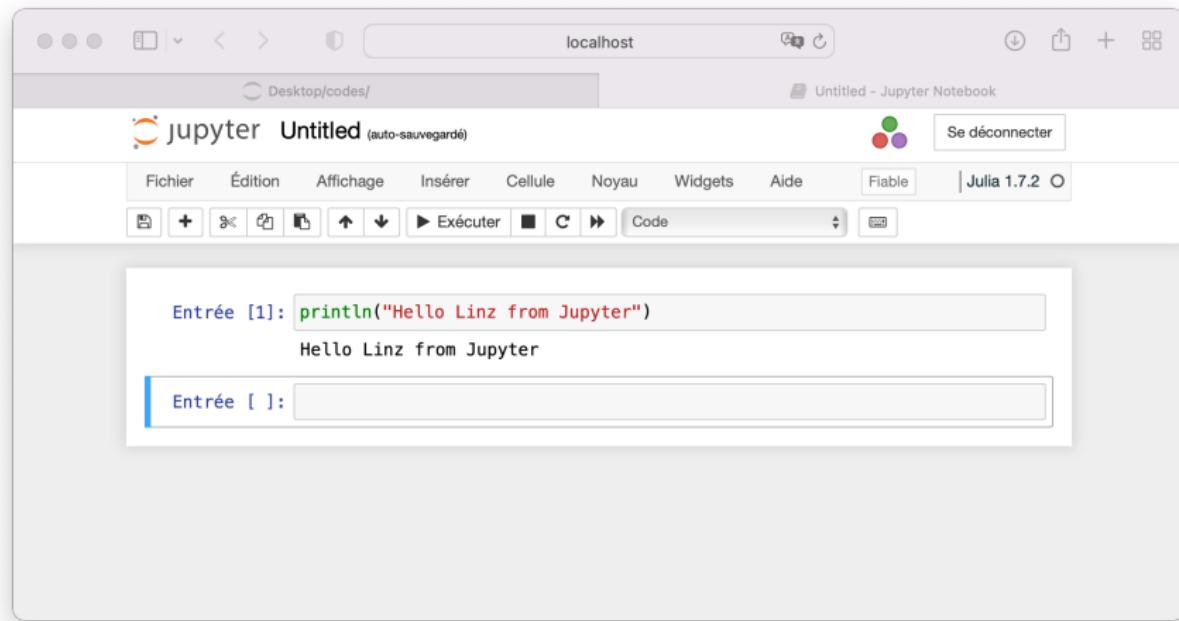
```
julia> using IJulia
julia> notebook()
```

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:

```
(@v1.7) 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>

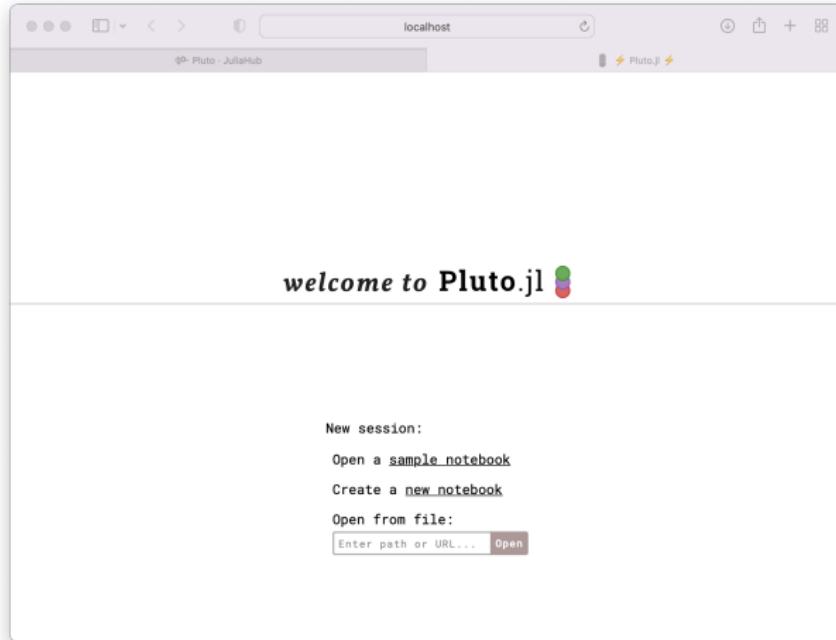
Steps to follow:

```
(@v1.7) pkg> add Pluto
```

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

Done!

Running Julia code into a notebook (Pluto)



Running Julia code into a notebook (2)

The screenshot shows a web-based Julia notebook interface. At the top, there's a toolbar with standard browser controls (back, forward, search, etc.) and a tab labeled "localhost". Below the toolbar, the title "Pluto.jl" is displayed next to a logo consisting of three colored circles (green, blue, red). To the right of the title is a "Save notebook..." button and a "New notebook" icon. The main area contains a code cell with the following content:

```
- println("Hello Linz from Pluto")
```

Below the code cell, the output is shown in a grey box:

```
Hello Linz from Pluto [1]
```

At the bottom of the interface, there's a "Live docs" button with a small icon. In the footer, there are links for "FAQ", "How can we make Pluto.jl better?", "Instant feedback...", and a "Send" button.



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

