

Jak zainstalować Julie?

Jak zainstalować Julię?

- Pobieramy odpowiedni instalator ze [strony języka](#):

Current stable release: v1.8.5 (January 8, 2023)

Checksums for this release are available in both [MD5](#) and [SHA256](#) formats.

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 (.dmg), 64-bit (.tar.gz)		
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)		
Generic Linux on PowerPC [help]	64-bit (little endian) (GPG)		
Generic FreeBSD on x86 [help]	64-bit (GPG)		
Source	Tarball (GPG)	Tarball with dependencies (GPG)	GitHub

Jak zainstalować Julię?

- Używać będziemy wersji **1.8.5**

Current stable release: v1.8.5 (January 8, 2023)

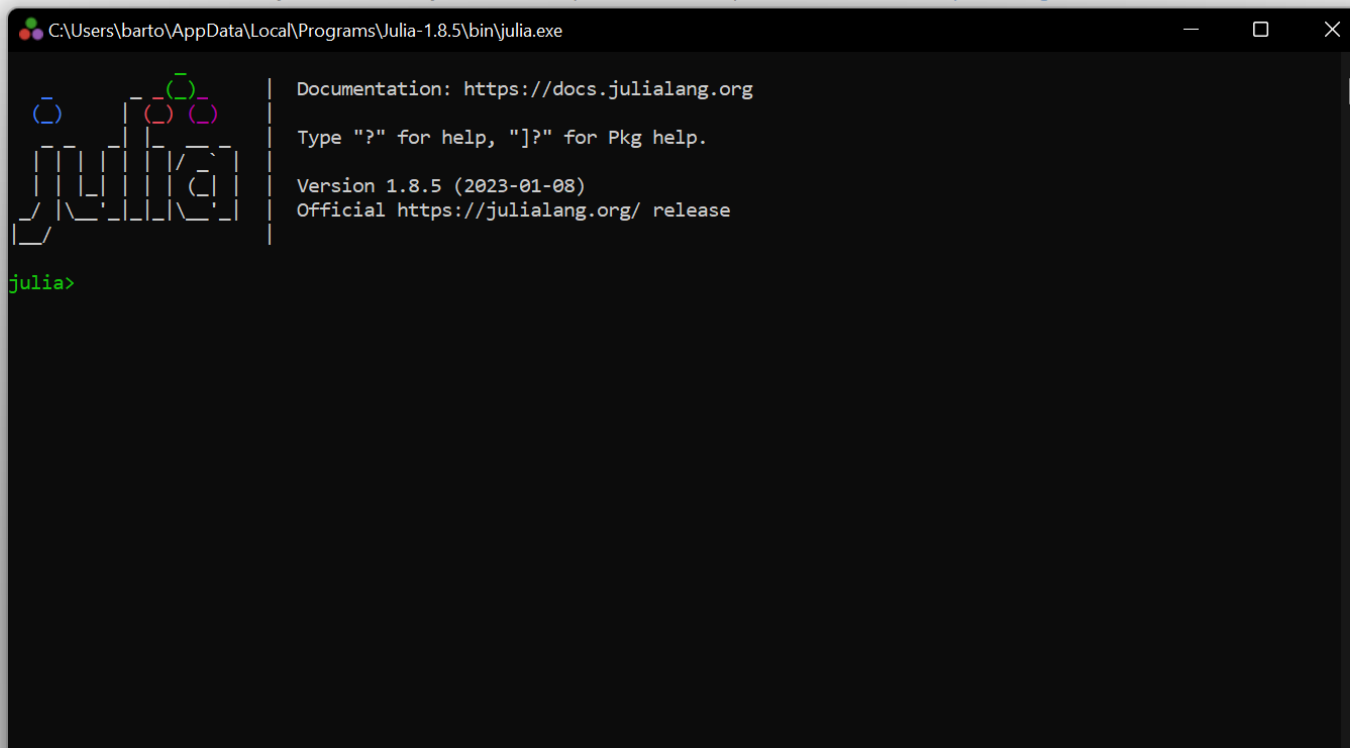
Checksums for this release are available in both [MD5](#) and [SHA256](#) formats.

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 (.dmg), 64-bit (.tar.gz)		
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)		
Generic Linux on PowerPC [help]	64-bit (little endian) (GPG)		
Generic FreeBSD on x86 [help]	64-bit (GPG)		
Source	Tarball (GPG)	Tarball with dependencies (GPG)	GitHub

Jak skonfigurować Julię?

- Następnie możemy zacząć konfigurować język do pracy na zajęciach.
- Uruchommy **REPLa**:

Please star us [on GitHub](#). If you use Julia in your research, please [cite us](#). If possible, do consider [sponsoring us](#).

A screenshot of the Julia REPL (Read-Eval-Print Loop) window. The window title is "C:\Users\barto\AppData\Local\Programs\Julia-1.8.5\bin\julia.exe". The window content shows the Julia logo on the left and the following text on the right: "Documentation: https://docs.julialang.org", "Type '?' for help, ']' for Pkg help.", "Version 1.8.5 (2023-01-08)", and "Official https://julialang.org/ release". At the bottom left, the prompt "julia>" is visible.

```
C:\Users\barto\AppData\Local\Programs\Julia-1.8.5\bin\julia.exe

Documentation: https://docs.julialang.org
Type "?" for help, "]" for Pkg help.
Version 1.8.5 (2023-01-08)
Official https://julialang.org/ release

julia>
```

portable)

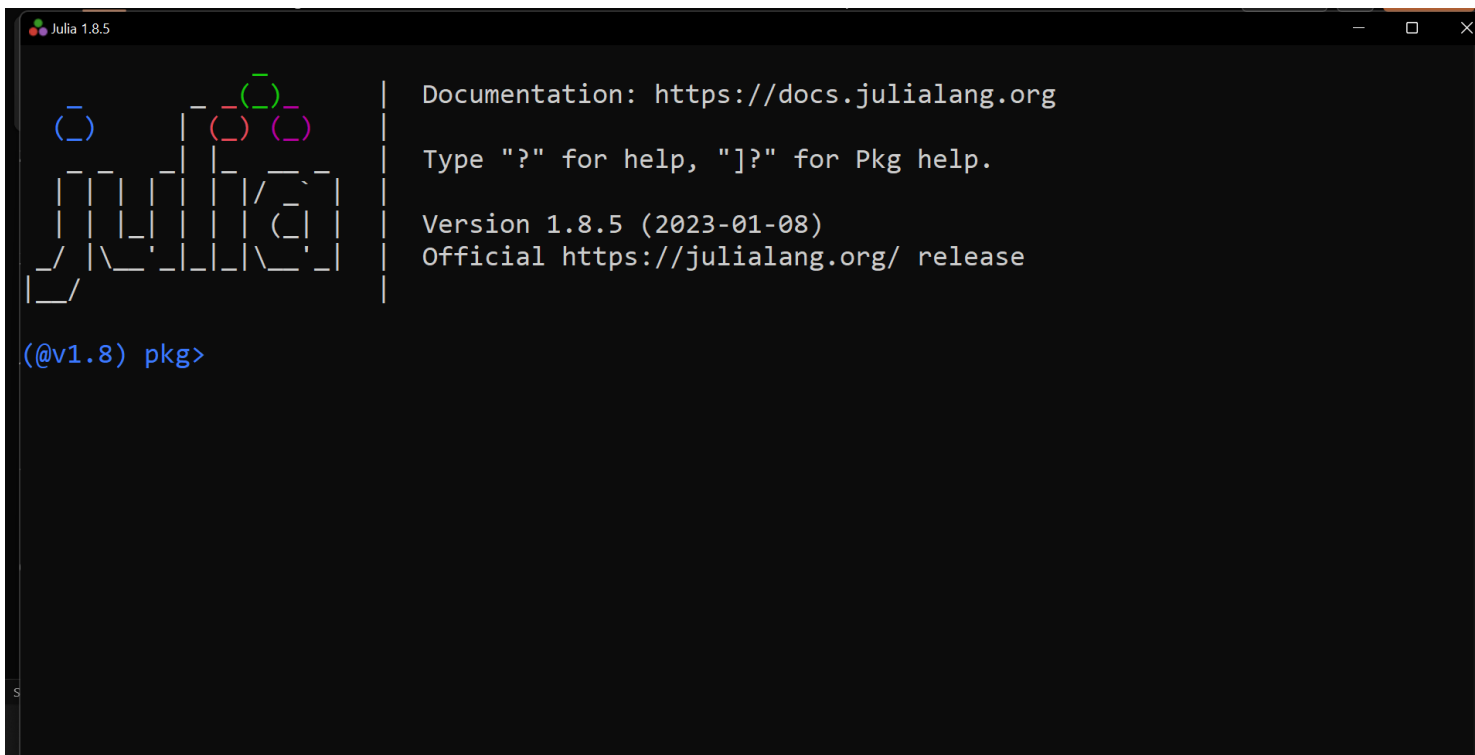
GitHub

Julia versions, so



Jak skonfigurować Julię?

- Zaczniemy instalować potrzebne paczki.
- Można to zrobić na dwa sposoby:
 - Albo poprzez wpisanie “**]**” i przejście do menagera paczek (znacznik zmieni się wtedy na niebieski):

A screenshot of the Julia REPL (Read-Eval-Print Loop) window. The window title is "Julia 1.8.5". On the left side, there is a stylized ASCII art logo of the word "julia" where the letters are formed by dashed lines and some characters are highlighted with colored boxes (blue, red, green, purple). To the right of the logo, the following text is displayed: "Documentation: https://docs.julialang.org", "Type '?' for help, ']?' for Pkg help.", "Version 1.8.5 (2023-01-08)", and "Official https://julialang.org/ release". At the bottom left, the prompt "(@v1.8) pkg>" is shown in blue text, indicating that the package manager is active.

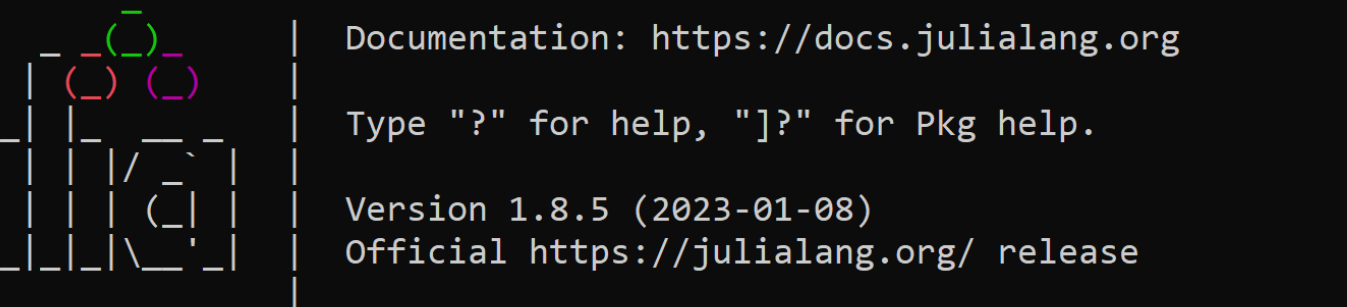
```
Julia 1.8.5

Documentation: https://docs.julialang.org
Type "?" for help, "]? " for Pkg help.
Version 1.8.5 (2023-01-08)
Official https://julialang.org/ release

(@v1.8) pkg>
```

Jak skonfigurować Julię?

- Zaczniemy instalować potrzebne paczki.
- Można to zrobić na dwa sposoby:
 - Albo poprzez wywołanie biblioteki “Pkg”:



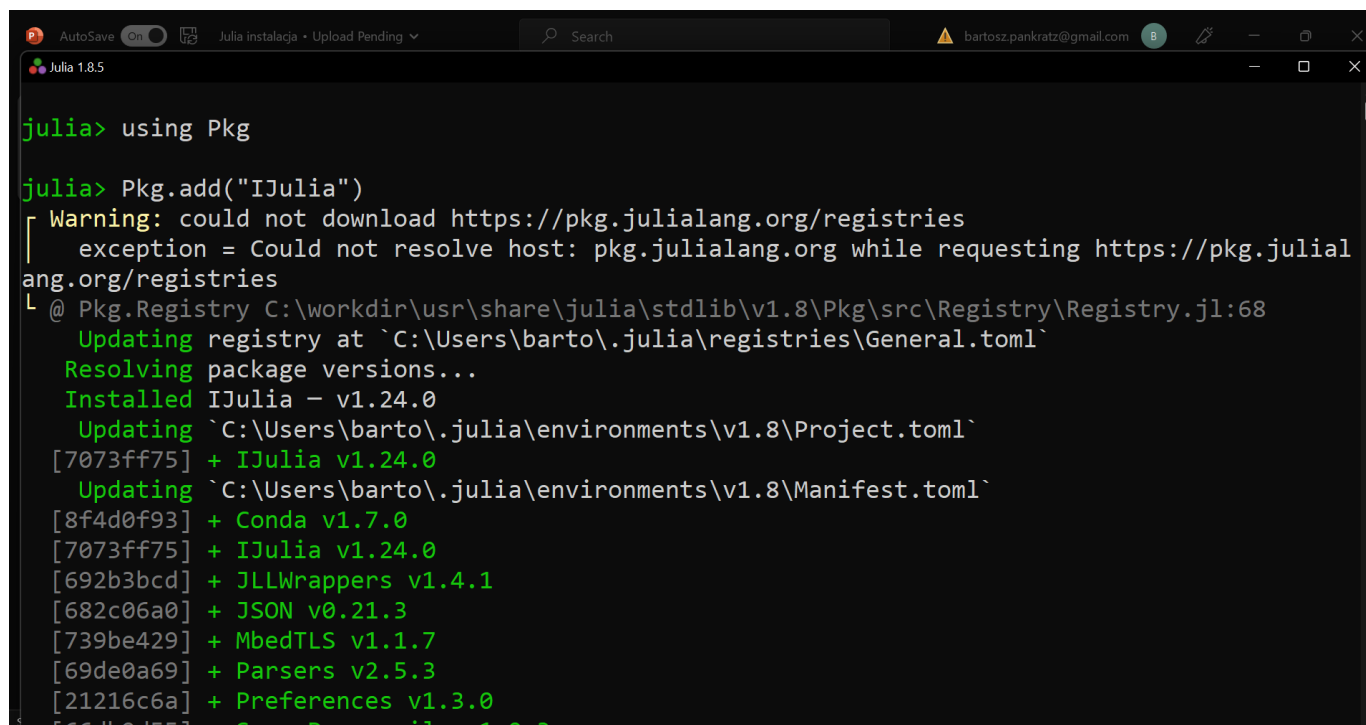
```
C:\Users\barto\AppData\Local\Programs\Julia-1.8.5\bin\julia.exe

julia> using Pkg

julia>
```

Jak skonfigurować Julię?

- Zaczniemy od skonfigurowania biblioteki "Julia".
- Pozwoli ona uruchamiać kod Julii w Jupyter Notebookach. Aby móc uruchamiać taki kod, konieczne jest poprawne skonfigurowanie środowiska Jupyter. Dodajmy bibliotekę:



```
julia> using Pkg

julia> Pkg.add("IJulia")
[ Warning: could not download https://pkg.julialang.org/registries
  exception = Could not resolve host: pkg.julialang.org while requesting https://pkg.julialang.org/registries
]
@ Pkg.Registry C:\workdir\usr\share\julia\stdlib\v1.8\Pkg\src\Registry\Registry.jl:68
  Updating registry at `C:\Users\barto\.julia\registries\General.toml`
  Resolving package versions...
  Installed IJulia - v1.24.0
  Updating `C:\Users\barto\.julia\environments\v1.8\Project.toml`
[7073ff75] + IJulia v1.24.0
  Updating `C:\Users\barto\.julia\environments\v1.8\Manifest.toml`
[8f4d0f93] + Conda v1.7.0
[7073ff75] + IJulia v1.24.0
[692b3bcd] + JLLWrappers v1.4.1
[682c06a0] + JSON v0.21.3
[739be429] + MbedTLS v1.1.7
[69de0a69] + Parsers v2.5.3
[21216c6a] + Preferences v1.3.0
[66db0d55] + ProgressMeter v1.0.0
```

Jak skonfigurować Julię?

- Dodajmy bibliotekę "PyCall".
- Pozwoli ona uruchamiać kod Pythona na poziomie Julii:

```
julia> Pkg.add("PyCall")
  Resolving package versions...
  Installed PyCall - v1.95.1
  Updating `C:\Users\barto\.julia\environments\v1.8\Project.toml`
[438e738f] + PyCall v1.95.1
  Updating `C:\Users\barto\.julia\environments\v1.8\Manifest.toml`
[1914dd2f] + MacroTools v0.5.10
[438e738f] + PyCall v1.95.1
[37e2e46d] + LinearAlgebra
[e66e0078] + CompilerSupportLibraries_jll v1.0.1+0
[4536629a] + OpenBLAS_jll v0.3.20+0
[8e850b90] + libblastrampoline_jll v5.1.1+0
  Building PyCall → `C:\Users\barto\.julia\scratchspaces\44cfe95a-1eb2-52ea-b672-e2afdf69b78f\62f417f6ad727987c755549e9cd88c46578da562\build.log`
  Precompiling project...
  5 dependencies successfully precompiled in 10 seconds. 16 already precompiled.

julia> _
```


Jak skonfigurować Julię?

- Bardzo często w przypadku biblioteki "**PyCall**" konieczne jest ręczne określenie wersji Pythona do której ma odwoływać się Julia. Aby to zrobić należy wpisać ścieżkę, w której znajduje się preferowana dystrybucja Pythona i przebudować paczkę "**PyCall**":

```
ENV["PYTHON"] = "... path of the python executable ..."  
# ENV["PYTHON"] = raw"C:\Python310-x64\python.exe" # example for Windows, "raw" to not have to escape: "C:\\P  
  
# ENV["PYTHON"] = "/usr/bin/python3.10" # example for *nix  
Pkg.build("PyCall")
```

Jak skonfigurować Julię?

- W przypadku bazowej Anacondy na Windowsie:

```
julia> ENV["PYTHON"] = "C:\\Users\\barto\\Anaconda3\\python.exe"
"C:\\Users\\barto\\Anaconda3\\python.exe"

julia> Pkg.build("PyCall")
Building Conda → `C:\\Users\\barto\\.julia\\scratchspaces\\44cfe95a-1eb2-52ea-b672-e2afdf69b78f\\6e47d11ea2776bc5627421d59cdcc1296c058071\\build.log`
Building PyCall → `C:\\Users\\barto\\.julia\\scratchspaces\\44cfe95a-1eb2-52ea-b672-e2afdf69b78f\\62f417f6ad727987c755549e9cd88c46578da562\\build.log`
Precompiling project...
1 dependency successfully precompiled in 8 seconds. 20 already precompiled.
```

Jak skonfigurować Julię?

- Biblioteki można też instalować wymuszając ich specyficzną wersję:

```
julia> Pkg.add(name = "DataFrames", version = "1.3.6")
  Resolving package versions...
No Changes to `C:\Users\barto\.julia\environments\v1.8\Project.toml`
No Changes to `C:\Users\barto\.julia\environments\v1.8\Manifest.toml`
```

Jak skonfigurować Julię?

- Można też podejrzeć jakie biblioteki i jakie ich wersję są zainstalowane korzystając z polecenia “**Pkg.status()**”:

```
julia> Pkg.status()
Status `C:\Users\barto\.julia\environments\v1.8\Project.toml`
 [b7f77d8d] ArcadeLearningEnvironment v0.2.4
 [fbb218c0] BSON v0.3.6
 [a93c6f00] DataFrames v1.3.6
 [587475ba] Flux v0.13.4
 [e15a9946] GridWorlds v0.5.0
 [7073ff75] IJulia v1.24.0
 [a09fc81d] ImageCore v0.9.4
 [916415d5] Images v0.25.2
 [91a5bcd] Plots v1.38.2
 [438e738f] PyCall v1.95.1
 [d330b81b] PyPlot v2.11.0
 [158674fc] ReinforcementLearning v0.10.1
 [e575027e] ReinforcementLearningBase v0.9.7
 [25e41dd2] ReinforcementLearningEnvironments v0.6.12
Info Packages marked with [?] and [?] have new versions available, but those with [?] are restricted
by compatibility constraints from upgrading. To see why use `status --outdated`
```

Jak skonfigurować Julię?

- W trakcie semestru będziemy potrzebowali następujących bibliotek:
 - **ArcadeLearningEnvironment** v0.2.4
 - **BSON** v0.3.6
 - **DataFrames** v1.3.6
 - **Flux** v0.13.4
 - **GridWorlds** v0.5.0
 - **IJulia** v1.24.0
 - **ImageCore** v0.9.4
 - **Images** v0.25.2
 - **Plots** v1.38.2
 - **PyCall** v1.95.1
 - **PyPlot** v2.11.0
 - **ReinforcementLearning** v0.10.1
 - **ReinforcementLearningBase** v0.9.7
 - **ReinforcementLearningEnvironments** v0.6.12