PRINCETON UNIVERSITY

iris hep

Bridging Python and Julia

for Enhanced Data Analysis

Ianna Osborne

PRINCETON UNIVERSITY

iris hep

Why Python is so popular

for data analysis

- Simple and easy to understand syntax
- Rich ecosystem of libraries tailored for data analysis
- Versatile, integrates well with other languages and tools
- Vast and active community
- Various programming paradigms support, including procedural, object-oriented, and functional programming
- Jupyter notebooks as an interactive environment where users can combine code execution, visualization, and narrative text
- Scalability and performance

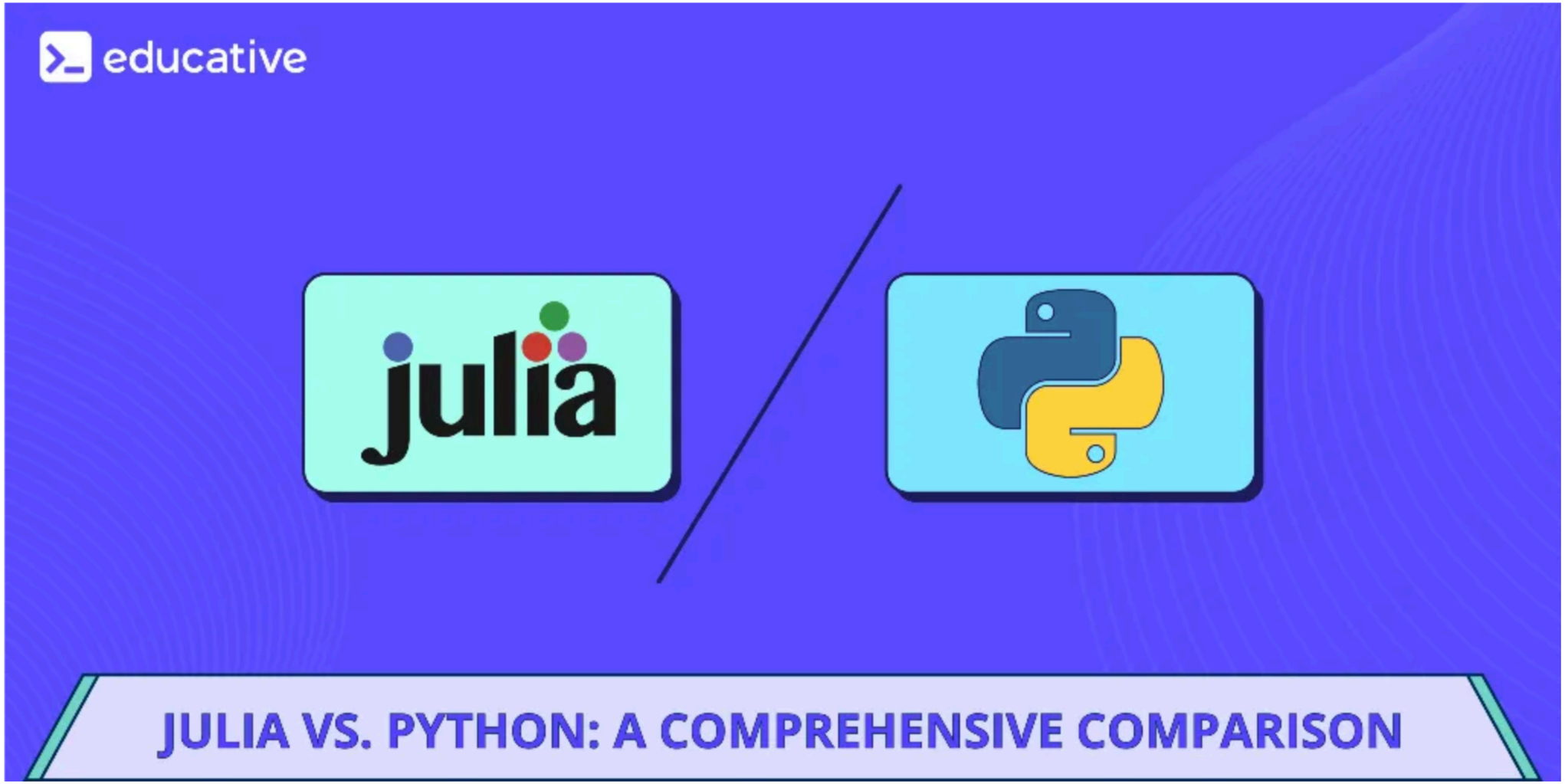
PRINCETON UNIVERSITY

iris hep

Key features of Julia

from a data scientist's perspective

- Speed of code execution
- Designed for interactive use
- Composability, leading to highly reusable code that is easy to maintain
- Package management
- Ease of integration with other languages

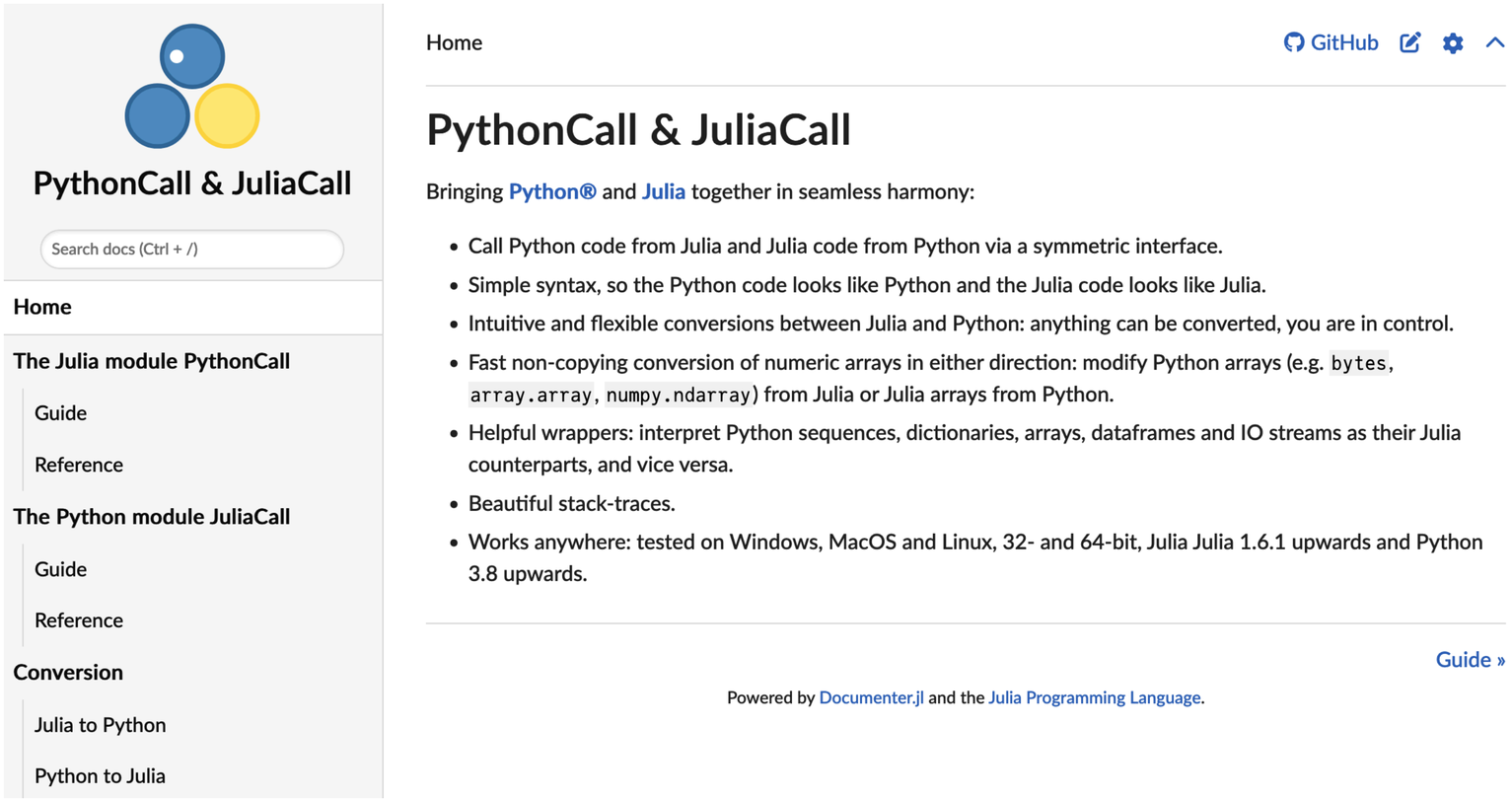

educative
julia
JULIA VS. PYTHON: A COMPREHENSIVE COMPARISON

Factor	Julia	Python
Maturity	Recent, created in 2012	Established, created in 1991
Scope	General purpose, but data-oriented	General purpose and used for almost anything
Language type	High level	High level
Typing	Dynamic and static	Dynamic
Open-source	Yes	Yes
Implementation	Compiled	Interpreted
Usage	**Data science and machine learning**, expanding to other areas	Mobile/web development, AI, data science, web scripting, desktop GUI development, game development
Data science	**Math functions are easy to write and understand** without external libraries	Requires Numoy or other libraries for advanced math
Commuity	Small but strong in the science community, continuously growing	Large, with more libraries and support
Performance	High-speed runtime, ideal for handling millions of data threads	Slower for production compare to Julia
Libraries	Growing library sources, but not as extensive as Python	Extensive library sources for code and application development
Code conversion	Easier to convert code from other language codes	More difficult to convert code from other languages
Popularity and community	Smaller, rapidly growing community and a majority support from developers themselves	Large, established community offering extensive support and solutions
Speed	Compiled language, often as fast as C, excellent for data analysis and statistical computing	Interpreted language, slower than Julia, relies on libraries for speed
Libraries	Limited collection, some packages not well maintained, ut expected to grow	Extensive range, supported by numerous third-party libraries
Code conversion	Straightforward process for converting code from other languages	More difficult than Julia, but possible through modules like PyCall
Linear algebra	Built for statistics and machine learning with methods easy to implement, syntax similar to math expressions	Requires libraries like Numpy, which is not as straightforward as Julia

PRINCETON UNIVERSITY

iris hep

PythonCall and JuliaCall



- PythonCall & JuliaCall allow to call Python code from Julia and Julia code from Python via a symmetric interface.
- Using PythonCall to bring Python functions and libraries into Julia
- Embed Julia code right into our Python scripts using JuliaCall

conda install conda-forge::pyjuliacall

PRINCETON UNIVERSITY

iris hep

JuliaPkg

What if Julia is not installed?

- JuliaCall relies on JuliaPkg to discover and install Julia
- JuliaCall uses JuliaUp (which uses the default depot) IF it is already installed - you need to install it manually. Otherwise JuliaCall will download and install Julia directly into your Python environment, but still using the default depot

```
In [2]: import awkward as ak
        from juliacall import Main as jl

        jl.seval("using AwkwardArray")
```

```
[juliapkg] Found dependencies: /usr/local/lib/python3.10/site-packages/awkward/juliapkg.jso
n
[juliapkg] Found dependencies: /usr/local/lib/python3.10/site-packages/juliacall/juliapkg.j
son
[juliapkg] Found dependencies: /usr/local/lib/python3.10/site-packages/juliapkg/juliapkg.js
on
[juliapkg] Locating Julia ~1.9, =1.10.0, ^1.10.3
[juliapkg] Querying Julia versions from https://julialang-s3.julialang.org/bin/versions.jso
n
[juliapkg] WARNING: About to install Julia 1.10.4 to /home/cms-jovyan/.julia/environments/p
yjuliapkg/pyjuliapkg/install.
[juliapkg]   If you use juliapkg in more than one environment, you are likely to
[juliapkg]   have Julia installed in multiple locations. It is recommended to
[juliapkg]   install JuliaUp (https://github.com/JuliaLang/juliaup) or Julia
[juliapkg]   (https://julialang.org/downloads) yourself.
[juliapkg] Downloading Julia from https://julialang-s3.julialang.org/bin/linux/x64/1.10/jul
ia-1.10.4-linux-x86_64.tar.gz
              download complete
[juliapkg] Verifying download
```

PRINCETON UNIVERSITY

iris hep

Julia Package Manager

a built-in state-of-the-art package manager

```
name = "AwkwardArray"
uuid = "7d259134-7f60-4bf1-aa00-7452e11bde56"
authors = ["Jim Pivarski <pivarski@princeton.edu>", "Jerry Ling <jerry.ling@cern.ch>", "and contributors"]
version = "0.1.5"

[deps]
Conda = "8f4d0f93-b110-5947-807f-2305c1781a2d"
JSON = "682c06a0-de6a-54ab-a142-c8b1cf79cde6"
PythonCall = "6099a3de-0909-46bc-b1f4-468b9a2dfc0d"
Tables = "bd369af6-aec1-5ad0-b16a-f7cc5008161c"

[compat]
Conda = "1.10.0"
JSON = "0.21.4"
PythonCall = "0.9"
Tables = "1.11.1"
julia = "1.9"

[extras]
Test = "8dfed614-e22c-5e08-85e1-65c5234f0b40"

[targets]
test = ["Test"]
```

- Package manager manages the computational environment state
- To fully specify the state of your Julia environment, it is enough to share **Project.toml** that uniquely identifies the package versions
- Automatically re-create the entire configuration of the runtime environment: ensures the reproducibility
- Resolves the common problem of managing dependencies of code written in other languages

PRINCETON UNIVERSITY

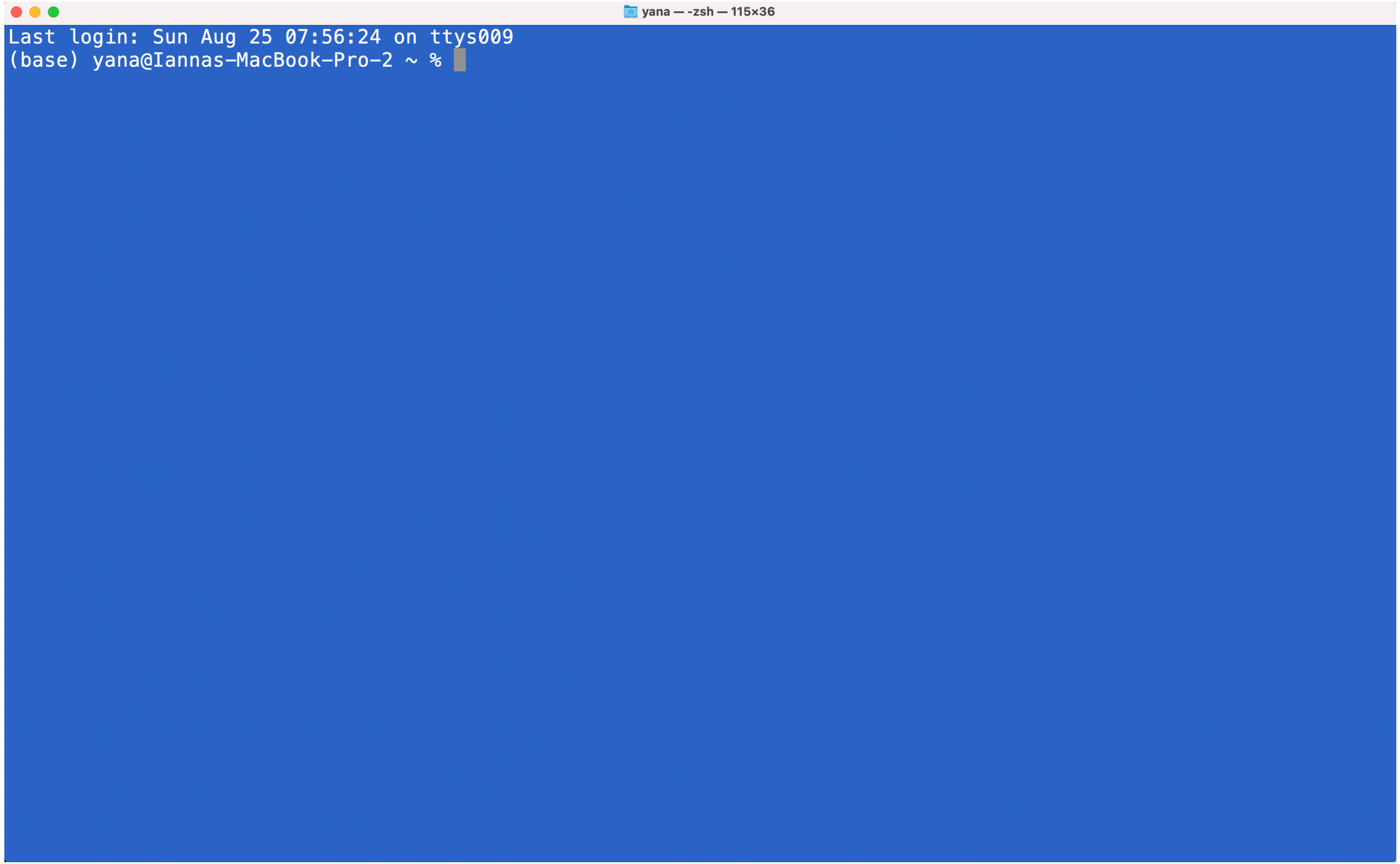
iris hep

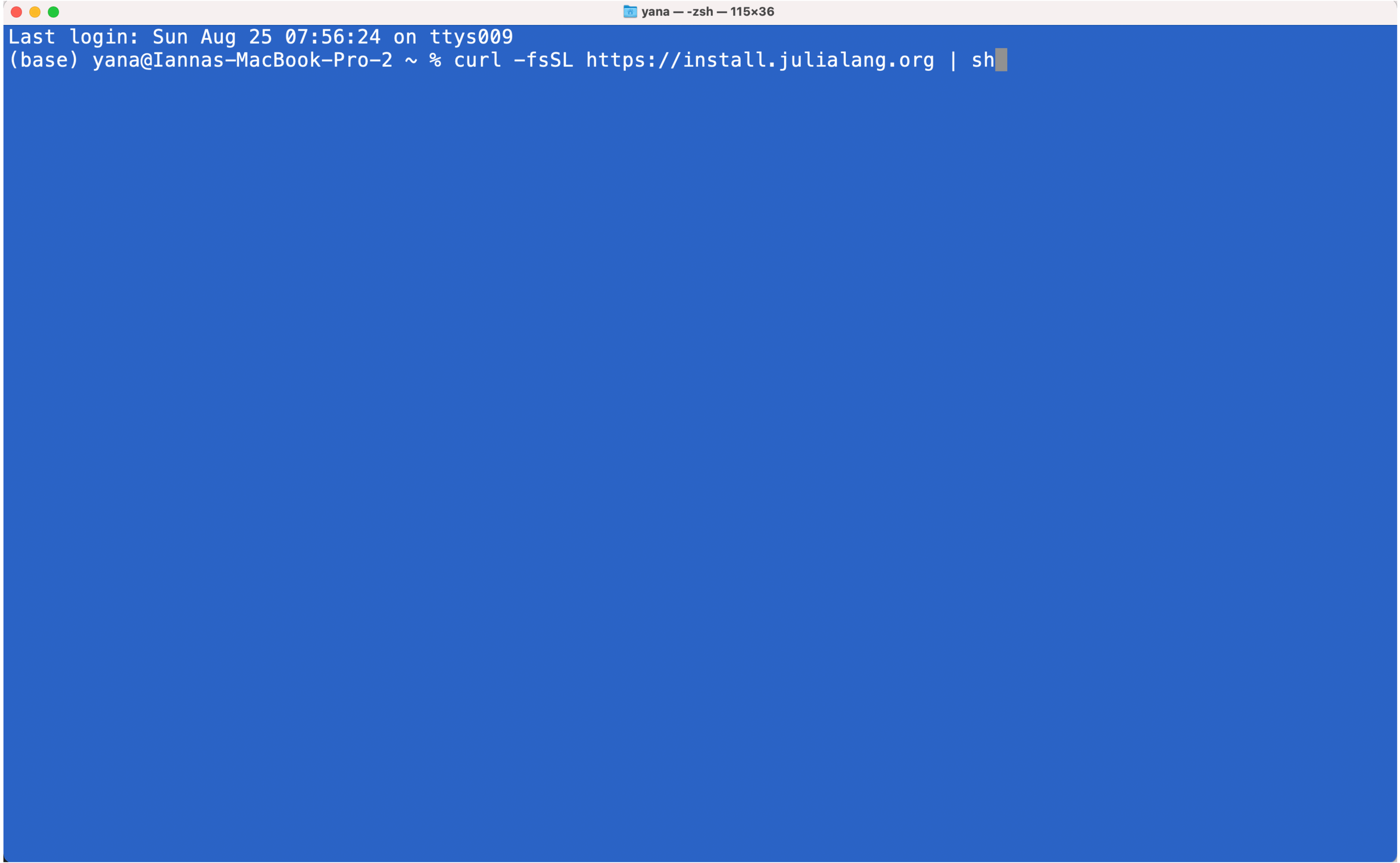
Best Practices for Managing Dependencies

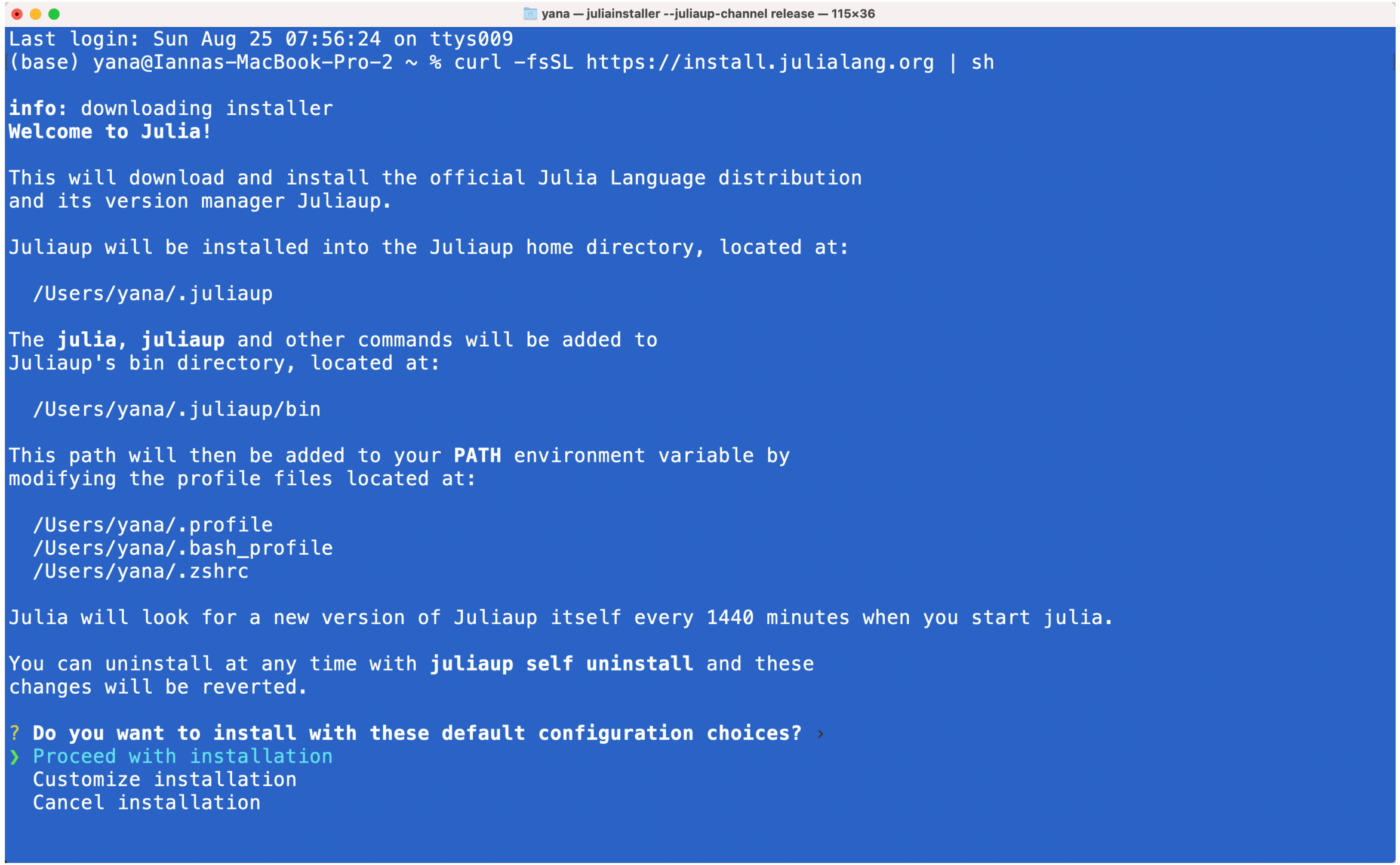
and keeping everything running smoothly

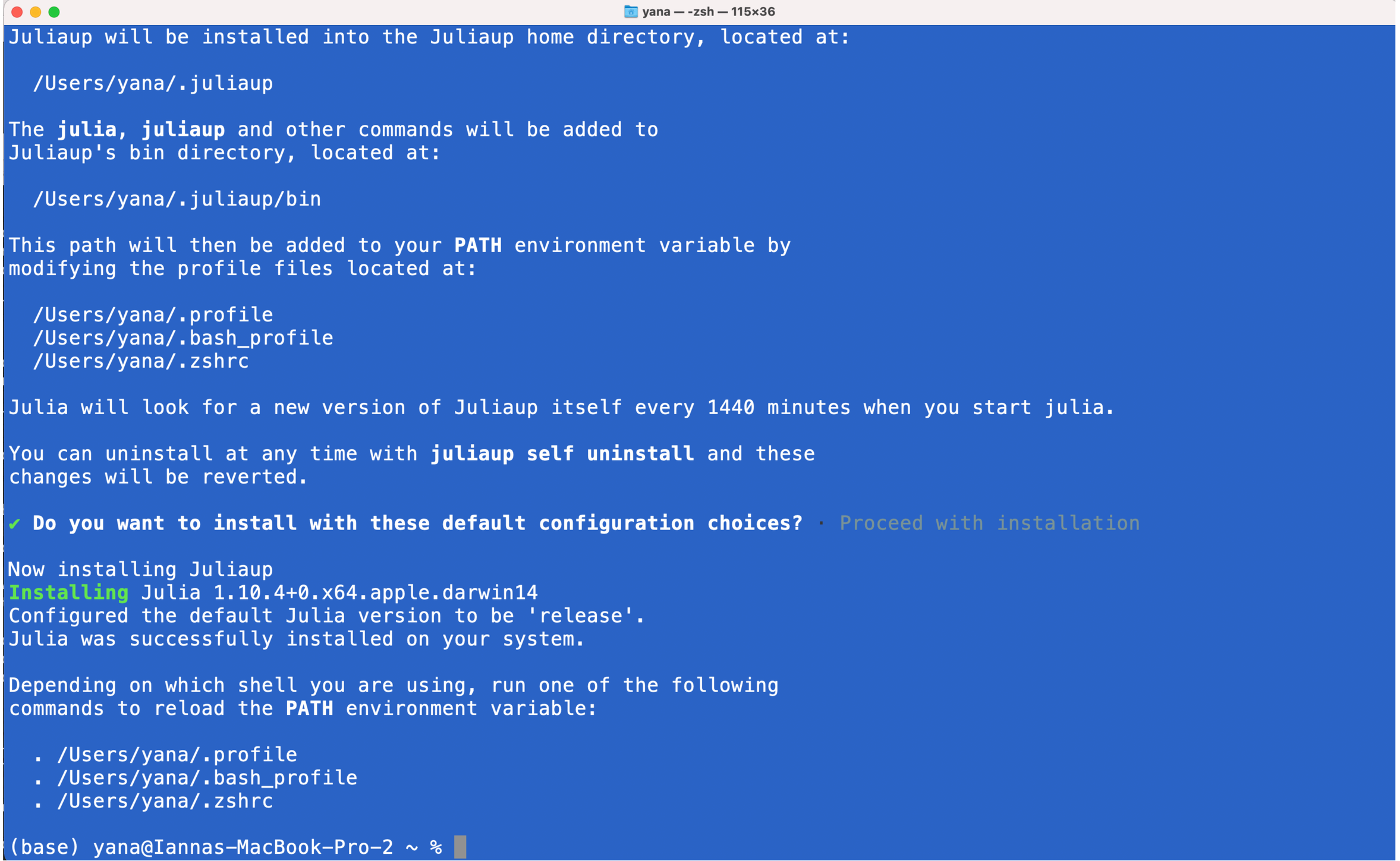
- JuliaUp is a recommended way to install Julia

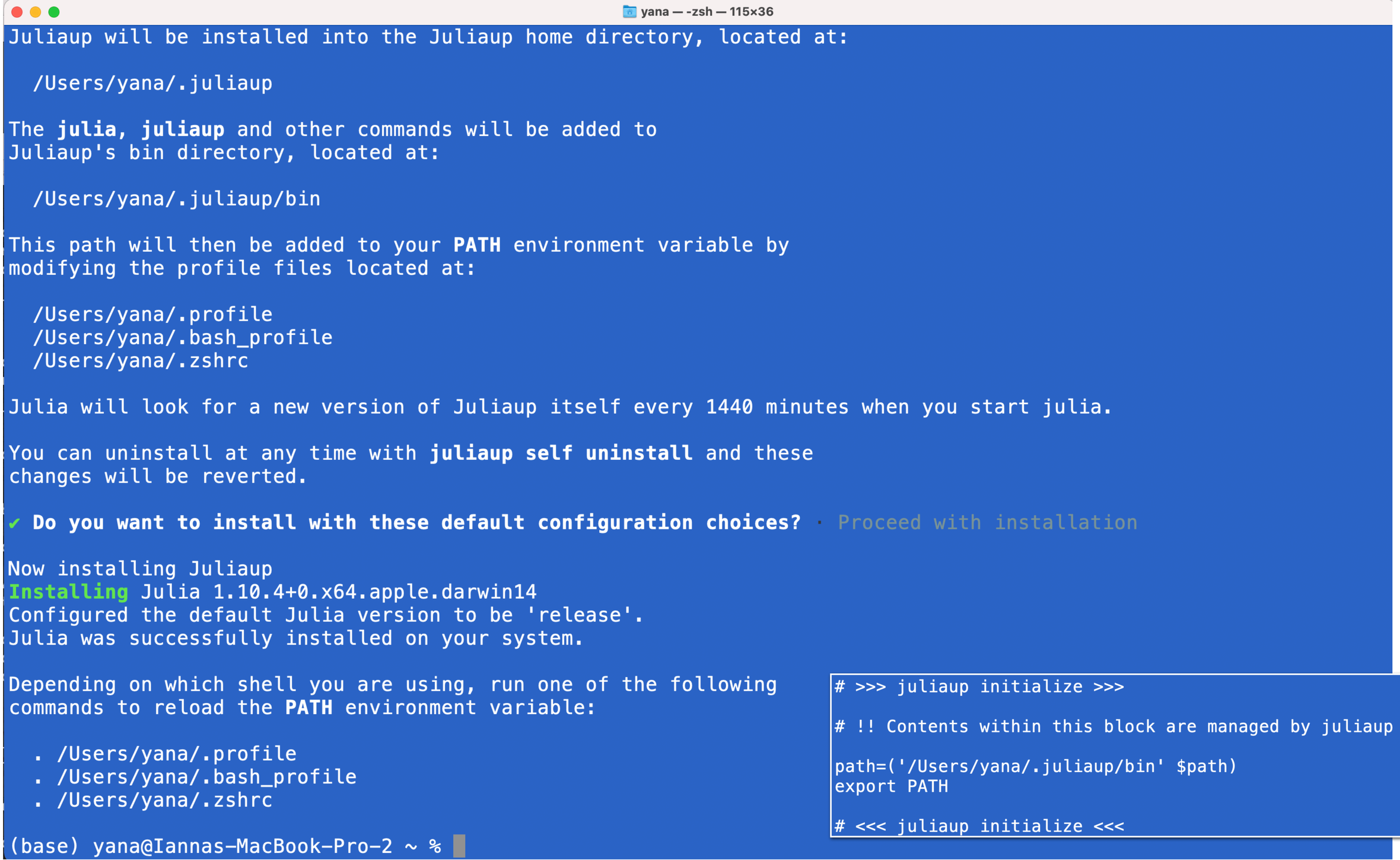
```
curl -fsSL https://install.julialang.org | sh
```


yana — -zsh — 115×36
Last login: Sun Aug 25 07:56:24 on ttys009
(base) yana@Iannas-MacBook-Pro-2 ~ %


yana — -zsh — 115×36
Last login: Sun Aug 25 07:56:24 on ttys009
(base) yana@Iannas-MacBook-Pro-2 ~ % curl -fsSL https://install.julialang.org | sh


yana — juliainstaller --juliaup-channel release — 115×36
Last login: Sun Aug 25 07:56:24 on ttys009
(base) yana@Iannas-MacBook-Pro-2 ~ % curl -fsSL https://install.julialang.org | sh
info: downloading installer
Welcome to Julia!
This will download and install the official Julia Language distribution
and its version manager Juliaup.
Juliaup will be installed into the Juliaup home directory, located at:
/Users/yana/.juliaup
The julia, juliaup and other commands will be added to
Juliaup's bin directory, located at:
/Users/yana/.juliaup/bin
This path will then be added to your PATH environment variable by
modifying the profile files located at:
/Users/yana/.profile
/Users/yana/.bash_profile
/Users/yana/.zshrc
Julia will look for a new version of Juliaup itself every 1440 minutes when you start julia.
You can uninstall at any time with juliaup self uninstall and these
changes will be reverted.
? Do you want to install with these default configuration choices? ›
❯ Proceed with installation
Customize installation
Cancel installation


yana — -zsh — 115×36
Juliaup will be installed into the Juliaup home directory, located at:
/Users/yana/.juliaup
The julia, juliaup and other commands will be added to
Juliaup's bin directory, located at:
/Users/yana/.juliaup/bin
This path will then be added to your PATH environment variable by
modifying the profile files located at:
/Users/yana/.profile
/Users/yana/.bash_profile
/Users/yana/.zshrc
Julia will look for a new version of Juliaup itself every 1440 minutes when you start julia.
You can uninstall at any time with juliaup self uninstall and these
changes will be reverted.
✔ Do you want to install with these default configuration choices? · Proceed with installation
Now installing Juliaup
Installing Julia 1.10.4+0.x64.apple.darwin14
Configured the default Julia version to be 'release'.
Julia was successfully installed on your system.
Depending on which shell you are using, run one of the following
commands to reload the PATH environment variable:
. /Users/yana/.profile
. /Users/yana/.bash_profile
. /Users/yana/.zshrc
(base) yana@Iannas-MacBook-Pro-2 ~ %


yana — -zsh — 115×36
Juliaup will be installed into the Juliaup home directory, located at:
/Users/yana/.juliaup
The julia, juliaup and other commands will be added to
Juliaup's bin directory, located at:
/Users/yana/.juliaup/bin
This path will then be added to your PATH environment variable by
modifying the profile files located at:
/Users/yana/.profile
/Users/yana/.bash_profile
/Users/yana/.zshrc
Julia will look for a new version of Juliaup itself every 1440 minutes when you start julia.
You can uninstall at any time with juliaup self uninstall and these
changes will be reverted.
✔ Do you want to install with these default configuration choices? · Proceed with installation
Now installing Juliaup
Installing Julia 1.10.4+0.x64.apple.darwin14
Configured the default Julia version to be 'release'.
Julia was successfully installed on your system.
Depending on which shell you are using, run one of the following
commands to reload the PATH environment variable:
. /Users/yana/.profile
. /Users/yana/.bash_profile
. /Users/yana/.zshrc
(base) yana@Iannas-MacBook-Pro-2 ~ %
# >>> juliaup initialize >>>
# !! Contents within this block are managed by juliaup !!
path=('/Users/yana/.juliaup/bin' $path)
export PATH
<<< juliaup initialize <<<

```
(base) yana@Iannas-MacBook-Pro-2 ~ % . /Users/yana/.zshrc
(base) yana@Iannas-MacBook-Pro-2 ~ % juliaup list
 Channel             Version
-------------------------------------------------
 0                   0.7.0+0.x64.apple.darwin14
 0.7                 0.7.0+0.x64.apple.darwin14
 0.7.0               0.7.0+0.x64.apple.darwin14
 0.7.0~x64           0.7.0+0.x64.apple.darwin14
 0.7~x64             0.7.0+0.x64.apple.darwin14
 0~x64               0.7.0+0.x64.apple.darwin14
 1                   1.10.4+0.x64.apple.darwin14
 1.0                 1.0.5+0.x64.apple.darwin14
 1.0.0               1.0.0+0.x64.apple.darwin14
 1.0.0~x64           1.0.0+0.x64.apple.darwin14
 1.0.1               1.0.1+0.x64.apple.darwin14
 1.0.1~x64           1.0.1+0.x64.apple.darwin14
 1.0.2               1.0.2+0.x64.apple.darwin14
 1.0.2~x64           1.0.2+0.x64.apple.darwin14
 1.0.3               1.0.3+0.x64.apple.darwin14
 1.0.3~x64           1.0.3+0.x64.apple.darwin14
 1.0.4               1.0.4+0.x64.apple.darwin14
 1.0.4~x64           1.0.4+0.x64.apple.darwin14
 1.0.5               1.0.5+0.x64.apple.darwin14
 1.0.5~x64           1.0.5+0.x64.apple.darwin14
 1.0~x64             1.0.5+0.x64.apple.darwin14
 1.1                 1.1.1+0.x64.apple.darwin14
 1.1.0               1.1.0+0.x64.apple.darwin14
 1.1.0~x64           1.1.0+0.x64.apple.darwin14
 1.1.1               1.1.1+0.x64.apple.darwin14
 1.1.1~x64           1.1.1+0.x64.apple.darwin14
 1.3.1~x64           1.3.1+0.x64.apple.darwin14
 1.4.1~x64           1.4.1+0.x64.apple.darwin14
 1.4.2~x64           1.4.2+0.x64.apple.darwin14
 1.6.0               1.6.0+0.x64.apple.darwin14
 1.6.0-beta1~x64     1.6.0-beta1+0.x64.apple.darwin14
 1.6.0-rc2           1.6.0-rc2+0.x64.apple.darwin14
```

```
(base) yana@Iannas-MacBook-Pro-2 ~ % . /Users/yana/.zshrc
(base) yana@Iannas-MacBook-Pro-2 ~ % juliaup list
 Channel             Version
------------------------------------------------
 0                   0.7.0+0.x64.apple.darwin14
 0.7                 0.7.0+0.x64.apple.darwin14
 0.7.0               0.7.0+0.x64.apple.darwin14
 0.7.0~x64           0.7.0+0.x64.apple.darwin14
 0.7~x64             0.7.0+0.x64.apple.darwin14
 0~x64               0.7.0+0.x64.apple.darwin14
 1                   1.10.4+0.x64.apple.darwin14
 1.0                 1.0.5+0.x64.apple.darwin14
 1.0.0               1.0.0+0.x64.apple.darwin14
 1.0.0~x64           1.0.0+0.x64.apple.darwin14
 1.0.1               1.0.1+0.x64.apple.darwin14
 1.0.1~x64           1.0.1+0.x64.apple.darwin14
 1.0.2               1.0.2+0.x64.apple.darwin14
 1.0.2~x64           1.0.2+0.x64.apple.darwin14
 1.0.3               1.0.3+0.x64.apple.darwin14
 1.0.3~x64           1.0.3+0.x64.apple.darwin14
 1.0.4               1.0.4+0.x64.apple.darwin14
 1.0.4~x64           1.0.4+0.x64.apple.darwin14
 1.0.5               1.0.5+0.x64.apple.darwin14
 1.0.5~x64           1.0.5+0.x64.apple.darwin14
 1.0~x64             1.0.5+0.x64.apple.darwin14
 1.1                 1.1.1+0.x64.apple.darwin14
 1.1.0               1.1.0+0.x64.apple.darwin14
 1.1.0~x64           1.1.0+0.x64.apple.darwin14
 1.1.1               1.1.1+0.x64.apple.darwin14
 1.1.1~x64           1.1.1+0.x64.apple.darwin14
 1.3.1~x64           1.3.1+0.x64.apple.darwin14
 1.4.1~x64           1.4.1+0.x64.apple.darwin14
 1.4.2~x64           1.4.2+0.x64.apple.darwin14
 1.6.0               1.6.0+0.x64.apple.darwin14
 1.6.0-beta1~x64     1.6.0-beta1+0.x64.apple.darwin14
 1.6.0-rc2           1.6.0-rc2+0.x64.apple.darwin14
```

```
(base) yana@Iannas-MacBook-Pro-2 ~ % juliaup update
(base) yana@Iannas-MacBook-Pro-2 ~ % julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.10.4 (2024-06-04)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia>
```

PRINCETON UNIVERSITY

iris hep

Practical Examples

where mixing Python and Julia really shines

[1]:
```
import awkward as ak
import numpy as np
import matplotlib.pyplot as plt
```

PRINCETON UNIVERSITY

iris hep

Practical Examples

where mixing Python and Julia really shines

[1]:
```
import awkward as ak
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
from juliacall import Main as jl
from juliacall import Pkg
```

```
Detected IPython. Loading juliacall extension. See https://juliapy.github.io/PythonCall.jl/stable/compat/#IPython
```

PRINCETON UNIVERSITY

iris hep

Practical Examples

where mixing Python and Julia really shines

```
[3]: Pkg.add('AwkwardArray')
```

```
  Updating registry at `~/.julia/registries/General.toml`
 Resolving package versions...
  Updating `~/anaconda3/julia_env/Project.toml`
[7d259134] + AwkwardArray v0.1.5
  Updating `~/anaconda3/julia_env/Manifest.toml`
[7d259134] + AwkwardArray v0.1.5
[8f4d0f93] + Conda v1.10.2
[682c06a0] + JSON v0.21.4
[81def892] + VersionParsing v1.3.0
```

PRINCETON UNIVERSITY

iris hep

Practical Examples

where mixing Python and Julia really shines

[3]:
```
Pkg.add('AwkwardArray')
```

```
  Updating registry at `~/.julia/registries/General.toml`
 Resolving package versions...
  Updating `~/anaconda3/julia_env/Project.toml`
[7d259134] + AwkwardArray v0.1.5
  Updating `~/anaconda3/julia_env/Manifest.toml`
[7d259134] + AwkwardArray v0.1.5
[8f4d0f93] + Conda v1.10.2
[682c06a0] + JSON v0.21.4
[81def892] + VersionParsing v1.3.0
```

[4]:
```
jl.seval("using AwkwardArray")
```

PRINCETON UNIVERSITY

iris hep

Practical Examples

where mixing Python and Julia really shines

[5]:
```
Pkg.add('ForwardDiff')
```

```
  Resolving package versions...
 No Changes to `~/anaconda3/julia_env/Project.toml`
 No Changes to `~/anaconda3/julia_env/Manifest.toml`
```

[6]:
```
jl.seval("using ForwardDiff")
```

[7]:
```
jl.seval('g(x) = x * exp(-x) / (x^2 + 1) * sin(pi * x)')
jl.seval('g1(x) = ForwardDiff.derivative(g, x)')
jl.seval('g2(x) = ForwardDiff.derivative(g1, x)')
jl.seval('g3(x) = ForwardDiff.derivative(g2, x)')
```

[7]:
```
g3 (generic function with 1 method)
```

PRINCETON UNIVERSITY

iris hep

Practical Examples

where mixing Python and Julia really shines

[8]:
```
def taylor(x, a):
    return jl.g(a) + \
    jl.g1(a) * (x - a) + \
    jl.g2(a) * (x - a)**2 / 2 + \
    jl.g3(a) * (x - a)**3 / 6
```

PRINCETON UNIVERSITY

iris hep

Practical Examples

where mixing Python and Julia really shines

[8]:
```
def taylor(x, a):
    return jl.g(a) + \
    jl.g1(a) * (x - a) + \
    jl.g2(a) * (x - a)**2 / 2 + \
    jl.g3(a) * (x - a)**3 / 6
```

[9]:
```
domain = ak.Array(np.arange(-2, 2, step=0.05))
image = jl.map(jl.g, domain)
a = -0.4
```

PRINCETON UNIVERSITY

iris hep

Practical Examples

where mixing Python and Julia really shines

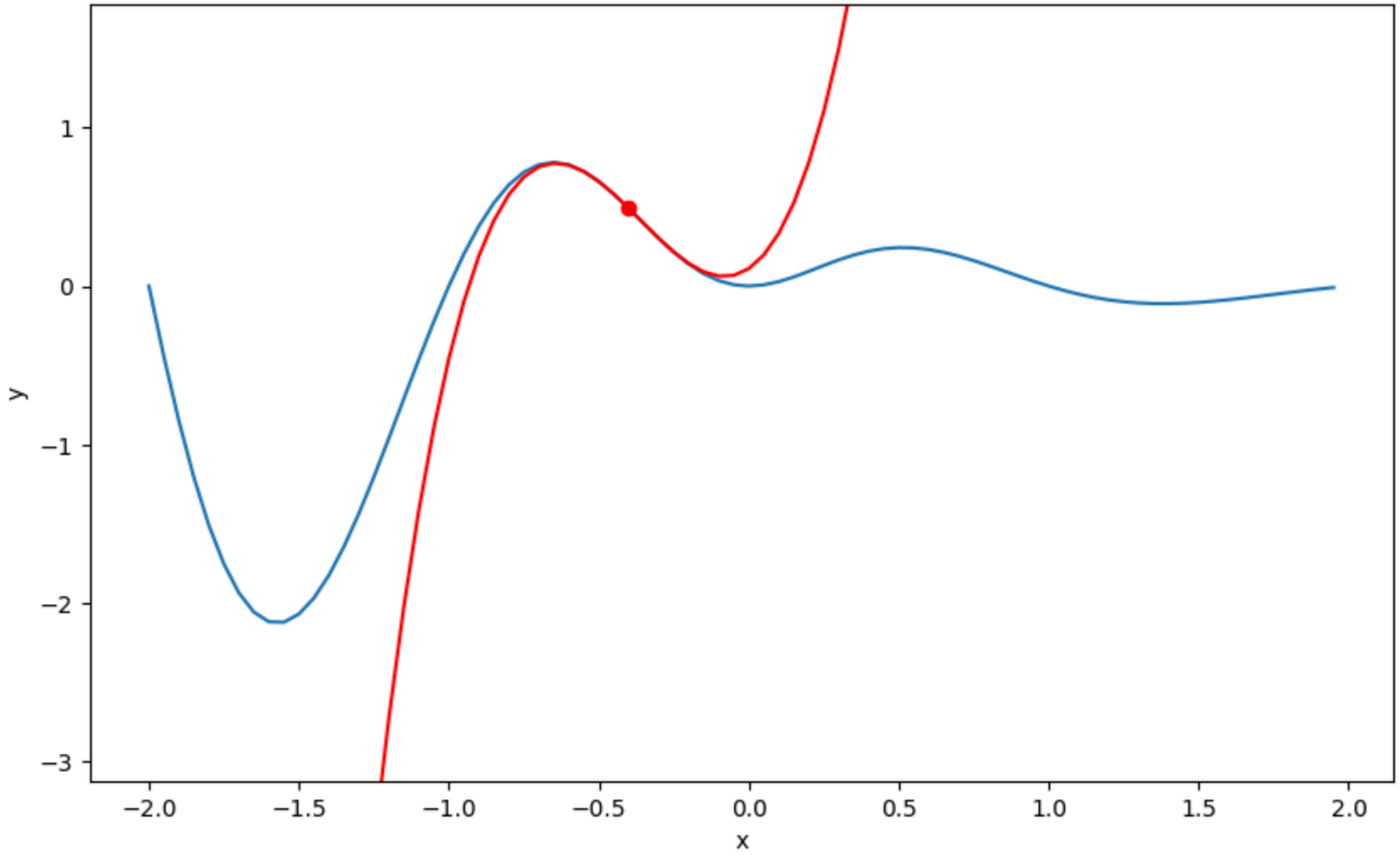
[10]:
```
plt.figure(figsize=(10,6))
plt.plot(domain, image, label='f(x)')
plt.scatter([a], [jl.g(a)], color='red', label='a')
plt.plot(domain, taylor(domain, a), color='red', label='3rd order Taylor poly arou
plt.ylim(ak.min(image) - 1, ak.max(image) + 1)
plt.ylabel('y')
plt.xlabel('x');
```

PRINCETON UNIVERSITY

iris hep

Practical Examples

where mixing Python and Julia really shines



PRINCETON UNIVERSITY

iris hep

Production-grade Examples

Python package using Julia as backend

- diffeqpy is a package for solving differential equations in Python. It utilizes DifferentialEquations.jl for its core routines to give high performance solving of many different types of differential equations, including:
  - Discrete equations (function maps, discrete stochastic (Gillespie/Markov) simulations)
    Ordinary differential equations (ODEs)
    Split and Partitioned ODEs (Symplectic integrators, IMEX Methods)
    Stochastic ordinary differential equations (SODEs or SDEs)
    Random differential equations (RODEs or RDEs)
    Differential algebraic equations (DAEs)
    Delay differential equations (DDEs)
    Mixed discrete and continuous equations (Hybrid Equations, Jump Diffusions) directly in Python.
- PySR is an open-source tool for Symbolic Regression: a machine learning task where the goal is to find an interpretable symbolic expression that optimizes some objective.
  - Over a period of several years, PySR has been engineered from the ground up to be (1) as high-performance as possible, (2) as configurable as possible, and (3) easy to use. PySR is developed alongside the Julia library SymbolicRegression.jl, which forms the powerful search engine of PySR. The details of these algorithms are described in the PySR paper.
  - Symbolic regression works best on low-dimensional datasets, but one can also extend these approaches to higher-dimensional spaces by using "Symbolic Distillation" of Neural Networks, as explained in 2006.11287, where we apply it to N-body problems. Here, one essentially uses symbolic regression to convert a neural net to an analytic equation. Thus, these tools simultaneously present an explicit and powerful way to interpret deep neural networks.

PRINCETON UNIVERSITY

iris hep

Real-world scenarios

how this combination can speed up our data analysis and make our work more efficient

julia

[11]:
```
jl.seval("""
function invariant_mass(tree)
  layout = AwkwardArray.PrimitiveArray{Float64}()
  for event in tree
    if event.nMuon == 2
      if event.Muon_charge[1] != event.Muon_charge[2]
        result = sqrt(2 * event.Muon_pt[1] * event.Muon_pt[2] *
                  (cosh(event.Muon_eta[1] - event.Muon_eta[2]) -
                   cos(event.Muon_phi[1] - event.Muon_phi[2])))
        if result > 70
          push!(layout, result)
        end
      end
    end
  end
  layout
end
""")
```

[11]:
```
invariant_mass (generic function with 1 method)
```

PRINCETON UNIVERSITY

iris hep

Real-world scenarios

how this combination can speed up our data analysis and make our work more efficient

julia

[12]:
```
Pkg.add('UnROOT')
```

```
   Resolving package versions...
  No Changes to `~/anaconda3/julia_env/Project.toml`
  No Changes to `~/anaconda3/julia_env/Manifest.toml`
```

[13]:
```
jl.seval("using UnROOT")
```

[14]:
```
file = jl.Main.ROOTFile("data/SMHiggsToZZTo4L.root")
```

[15]:
```
file
```

[15]:
```
ROOTFile with 1 entry and 18 streamers.
data/SMHiggsToZZTo4L.root
└─ Events (TTree)
   ├─ "run"
   ├─ "luminosityBlock"
   ├─ "event"
   ├─ ":"
   ├─ "Electron_dzErr"
   ├─ "MET_pt"
   └─ "MET_phi"
```

PRINCETON UNIVERSITY

iris hep

Real-world scenarios

how this combination can speed up our data analysis and make our work more efficient

[16]:
```
events = jl.Main.LazyTree(file, "Events")
```

[17]:
```
events
```

[17]:

Row	Electron_mass SubArray{Float3	nElectron UInt32	luminosityBlock UInt32	nMuon UInt32	Electron_phi SubArray{Float3	M … F …
1	[]	0	156	3	[]	– …
2	[0.00544,	4	156	0	[0.134, -1	2 …
3	[-0.00609,	2	156	0	[2.18, 1.6	– …
4	[-0.00123]	1	156	7	[-0.643]	– …
5	[0.0117, 0	4	156	0	[1.01, -1.	1 …
6	[-0.00183]	1	156	2	[-0.497]	– …
7	[-0.00183]	1	156	1	[-1.47]	– …
8	[-0.00216]	1	156	0	[-0.633]	– …
9	[-0.0128,	4	156	0	[2.79, -2.	– …
10	[]	0	156	0	[]	– …
11	[-0.00119,	2	156	1	[2.28, -2.	0 …
12	[0.00608,	3	156	2	[-1.31, -1	2 …
13	[-0.00765,	4	156	0	[2.52, -3.	– …
14	[]	0	156	2	[]	– …
15	[]	0	156	0	[]	– …
⋮	⋮	⋮	⋮	⋮	⋮	⋱

27 columns and 299958 rows omitted

PRINCETON UNIVERSITY

iris hep

Real-world scenarios

how this combination can speed up our data analysis and make our work more efficient

julia

[18]:
```
jl.seval("""
using AwkwardArray

function make_record_array(events)
    array = AwkwardArray.RecordArray(
        NamedTuple{(:pt, :eta, :phi, :mass, :charge, :isolation)}((
            AwkwardArray.from_iter(events.Muon_pt),
            AwkwardArray.from_iter(events.Muon_eta),
            AwkwardArray.from_iter(events.Muon_phi),
            AwkwardArray.from_iter(events.Muon_mass),
            AwkwardArray.from_iter(events.Muon_charge),
            AwkwardArray.from_iter(events.Muon_pfRelIso03_all),
        )
    ))
    return AwkwardArray.convert(array)
end
""")
```

[18]:
```
make_record_array (generic function with 1 method)
```

PRINCETON UNIVERSITY

iris hep

Real-world scenarios

how this combination can speed up our data analysis and make our work more efficient

[19]:
```
%%time
muons = jl.make_record_array(events[1])
```

```
CPU times: user 1.29 s, sys: 14.9 ms, total: 1.31 s
Wall time: 1.31 s
```

[20]:
```
%%time
muons = jl.make_record_array(events)
```

```
CPU times: user 581 ms, sys: 24.2 ms, total: 605 ms
Wall time: 603 ms
```

PRINCETON UNIVERSITY

iris hep

Real-world scenarios

how this combination can speed up our data analysis and make our work more efficient

[21]:
```
muons = ak.zip({
                "pt": muons.pt,
                "eta": muons.eta,
                "phi": muons.phi,
                "mass": muons.mass,
                "charge": muons.charge,
                "isolation": muons.isolation,
            },
            with_name="PtEtaPhiMCandidate",)
```

[22]:
```
cutflow = dict()

# Sort muons by transverse momentum
muons = muons[ak.argsort(muons.pt, axis=1)]

cutflow["all events"] = ak.num(muons, axis=0)

# Quality and minimum pt cuts
muons = muons[(muons.pt > 5) & (muons.isolation < 0.2)]
cutflow["at least 4 good muons"] = ak.sum(ak.num(muons) >= 4)
```

[23]:
```
# reduce first axis: skip events without enough muons
muons = muons[ak.num(muons) >= 4]
```

PRINCETON UNIVERSITY

iris hep

Real-world scenarios

how this combination can speed up our data analysis and make our work more efficient

julia

[24]:
```
jl.seval("""
using AwkwardArray

four_muons = AwkwardArray.ListArray{
    AwkwardArray.Index64,
    AwkwardArray.ListArray{
        AwkwardArray.Index64,
        AwkwardArray.PrimitiveArray{Int64},
    },
}()

function find_4lep(events_leptons)

    for leptons in events_leptons
        nlep = length(leptons)
        for i0 in 1:nlep
            for i1 in (i0 + 1):nlep
                if leptons[i0][:charge] + leptons[i1][:charge] != 0
                    continue
                end
                for i2 in 1:nlep
                    for i3 in (i2 + 1):nlep
                        if length(Set([i0, i1, i2, i3])) < 4
                            continue
                        end
                        if leptons[i2][:charge] + leptons[i3][:charge] != 0
                            continue
                        end

                        push!(four_muons.content.content, (i0 - 1)) # Julia is 1-based, subtract 1 for 0-based indexing
                        push!(four_muons.content.content, (i1 - 1))
                        push!(four_muons.content.content, (i2 - 1))
                        push!(four_muons.content.content, (i3 - 1))
                        AwkwardArray.end_list!(four_muons.content)
                    end
                end
            end
        end
        AwkwardArray.end_list!(four_muons)
    end
    return four_muons
end""")
```

[24]:
```
find_4lep (generic function with 1 method)
```

PRINCETON UNIVERSITY

iris hep

Real-world scenarios

how this combination can speed up our data analysis and make our work more efficient

[25]:
```
fourmuon = jl.find_4lep(muons[1:10])
```

[26]:
```
%%time
fourmuon = jl.find_4lep(muons)
```

```
CPU times: user 417 ms, sys: 10.4 ms, total: 427 ms
Wall time: 426 ms
```

[27]:
```
fourmuon
```

[27]:
```
14963-element AwkwardArray.ListArray{Vector{Int64}, AwkwardArray.ListArray{Vector{Int64}, AwkwardArray.PrimitiveArray{Int64, Vector{Int64}, :default}, :default},
:default}:
 [[0, 1, 2, 3], [0, 3, 1, 2], [1, 2, 0, 3], [2, 3, 0, 1]]
 [[0, 1, 2, 3], [0, 3, 1, 2], [1, 2, 0, 3], [2, 3, 0, 1]]
 [[0, 1, 2, 3], [0, 3, 1, 2], [1, 2, 0, 3], [2, 3, 0, 1]]
 [[0, 1, 2, 3], [0, 2, 1, 3], [1, 3, 0, 2], [2, 3, 0, 1]]
 [[0, 2, 1, 3], [0, 3, 1, 2], [1, 2, 0, 3], [1, 3, 0, 2]]
 [[0, 1, 2, 3], [0, 3, 1, 2], [1, 2, 0, 3], [2, 3, 0, 1]]
 [[0, 2, 1, 3], [0, 3, 1, 2], [1, 2, 0, 3], [1, 3, 0, 2]]
 [[0, 1, 2, 3], [0, 3, 1, 2], [1, 2, 0, 3], [2, 3, 0, 1]]
 [[0, 2, 1, 4], [0, 2, 3, 4], [0, 4, 1, 2], ..., [3, 4, 0, 2], [3, 4, 1, 2]]
 0-element AwkwardArray.ListArray{Vector{Int64}, AwkwardArray.PrimitiveArray{Int64, Vector{Int64}, :default}, :default}
 ⋮
 [[0, 1, 2, 3], [0, 3, 1, 2], [1, 2, 0, 3], [2, 3, 0, 1]]
 [[0, 1, 2, 3], [0, 2, 1, 3], [1, 3, 0, 2], [2, 3, 0, 1]]
 [[0, 2, 1, 3], [0, 3, 1, 2], [1, 2, 0, 3], [1, 3, 0, 2]]
 [[0, 2, 1, 3], [0, 3, 1, 2], [1, 2, 0, 3], [1, 3, 0, 2]]
 [[0, 1, 2, 3], [0, 3, 1, 2], [1, 2, 0, 3], [2, 3, 0, 1]]
 0-element AwkwardArray.ListArray{Vector{Int64}, AwkwardArray.PrimitiveArray{Int64, Vector{Int64}, :default}, :default}
 [[0, 1, 2, 3], [0, 2, 1, 3], [1, 3, 0, 2], [2, 3, 0, 1]]
 [[0, 1, 2, 3], [0, 2, 1, 3], [1, 3, 0, 2], [2, 3, 0, 1]]
 [[0, 1, 2, 3], [0, 3, 1, 2], [1, 2, 0, 3], [2, 3, 0, 1]]
```

PRINCETON UNIVERSITY

iris hep

Compatibility Tools: IPython

- The line magic **%julia** code executes the given Julia code in-line
- The cell magic **%%julia** executes a cell of Julia code
- Julia's **stdout** and **stderr** are redirected to IPython
- Calling **display(x)** from Julia will display x in IPython

```
In [1]: %load_ext juliacall

        Detected IPython. Loading juliacall extension. See https://juliapy.github.io/PythonCall.jl/stable/compat/#IPython

        WARNING: replacing module _ipython.

In [2]: x = 2

In [3]: y = 8
```

The `%%julia` cell magic can synchronise variables between Julia and Python by listing them on the first line:

```
In [4]: %%julia x y z
            z = "$x^$y = $(x^y)";

In [5]: z

Out[5]: "2^8 = 256"
```

PRINCETON UNIVERSITY

iris hep

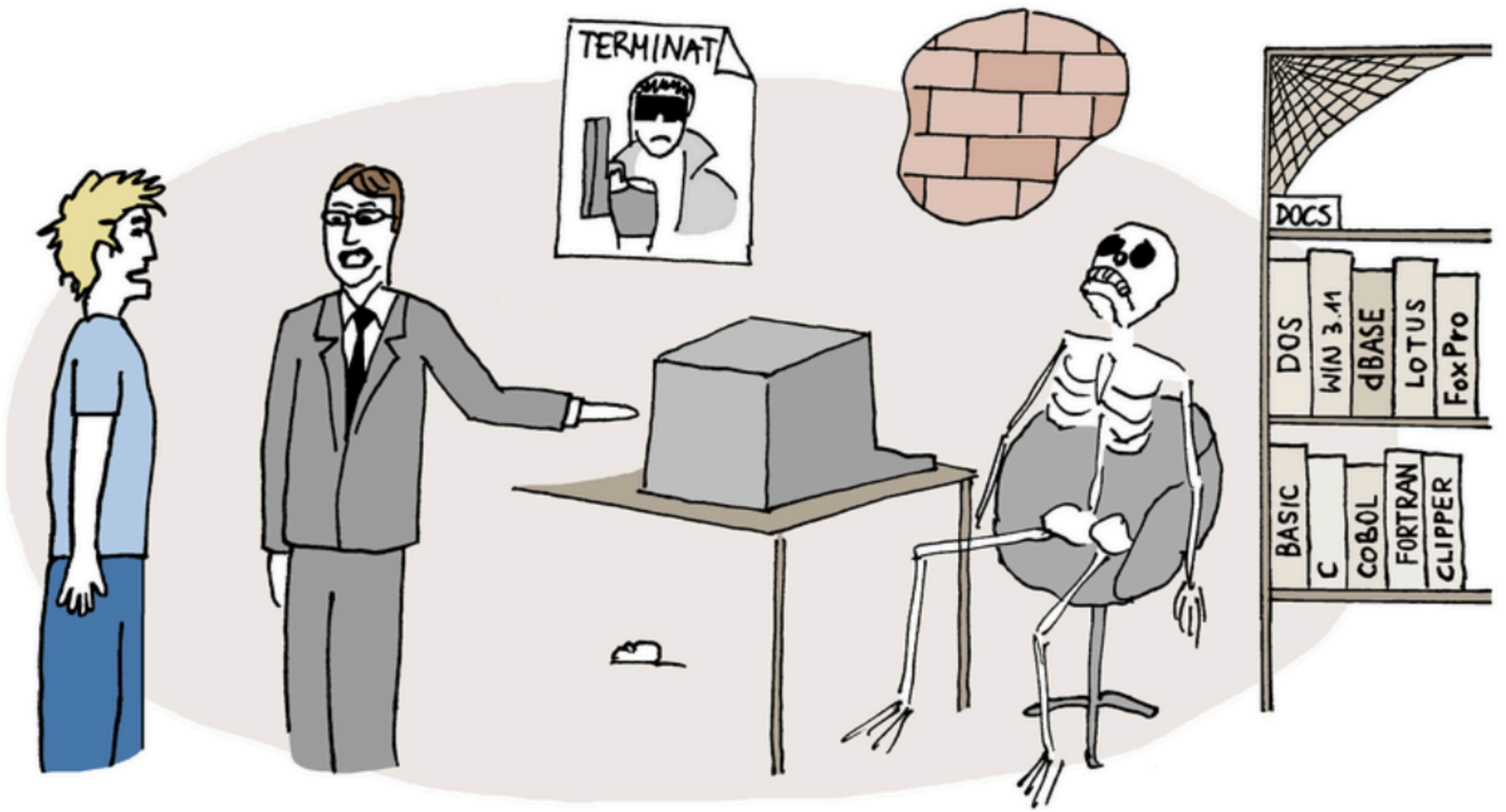
Summary and Outlook

- Some common challenges you might face and how to overcome them
- Look ahead to the future
- Potential for deeper integration and community-driven innovation

Pluto.jl

```
julia> using Pluto

julia> Pluto.run()
[ Info: Loading...
┌ Info:
└ Opening http://localhost:1234/?secret=3EMEw90i in your default browser... ~ have fun!
┌ Info:
│ Press Ctrl+C in this terminal to stop Pluto
└
```



AND YOU WILL TAKE OVER PROJECT FROM JERRY

Dataedo /cartoon

Piotr@Dataedo