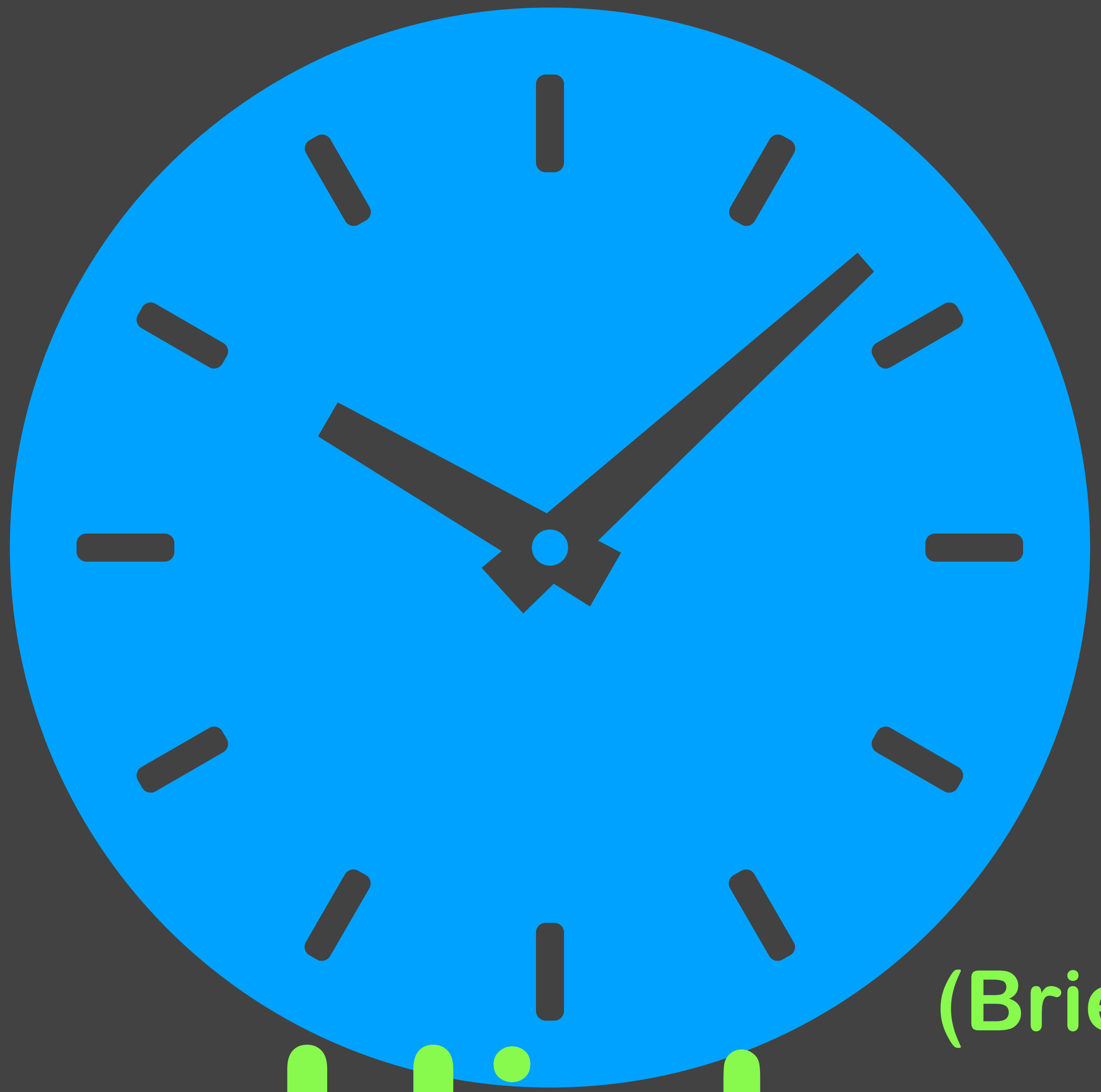
Awkward Packaging: building Scikit-HEP

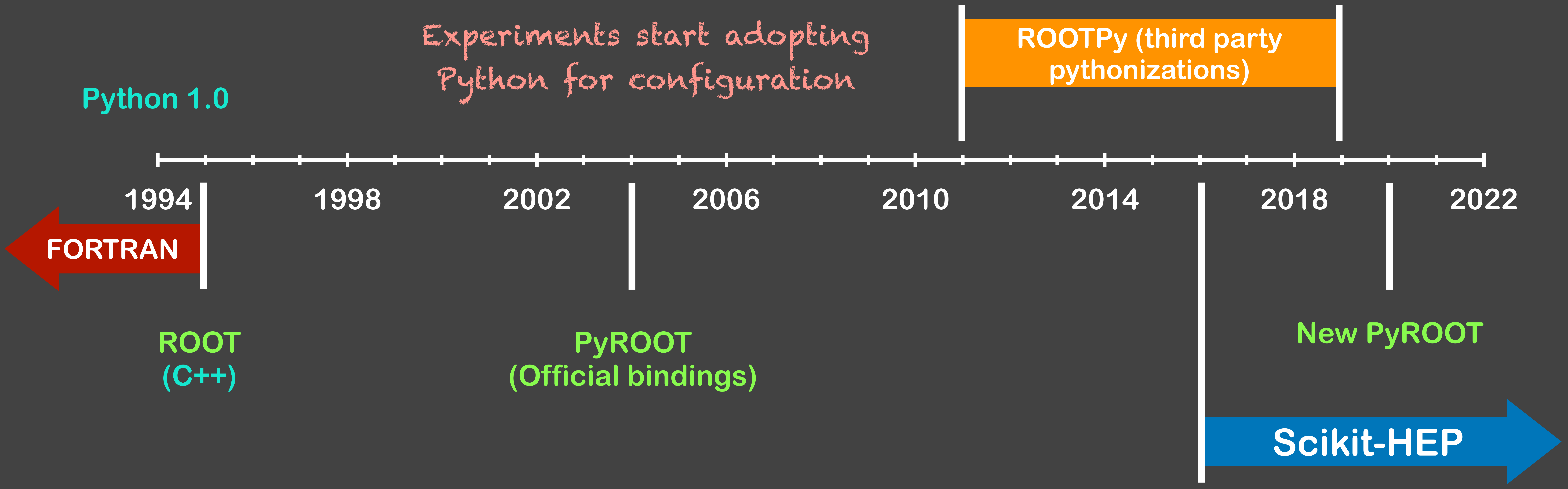
Jim Pivarski, Eduardo Rodrigues, Henry Schreiner

SciPy '22 July 14, 2022



(Brief) History of Python in HEP

New languages: C++ & Python



Python 1.0
Experiments start adopting Python for configuration
ROOTPy (third party pythonizations)
1994
1998
2002
2006
2010
2014
2018
2022
FORTRAN
ROOT (C++)
PyROOT (Official bindings)
New PyROOT
Scikit-HEP

(ROOT is both a toolkit and a file format)

ROOT: C++ toolkit (& interpreter)

Physicists only need to learn a single language
Python bindings introduced later as PyROOT

```
import ROOT
import numpy as np

file = ROOT.TFile("tree.root", "recreate")
tree = ROOT.TTree("name", "title")

px  = np.zeros(1, dtype=float)
phi = np.zeros(1, dtype=float)

tree.Branch("px",  px,  "normal/D")
tree.Branch("phi", phi, "uniform/D")

for i in range(10000):
    px[0]  = ROOT.gRandom.Gaus(20,2)
    phi[0] = ROOT.gRandom.Uniform(2*3.1416)
    tree.Fill()

file.Write()
file.Close()
```

Creating memory for pointer

Assigning pointers to fill from

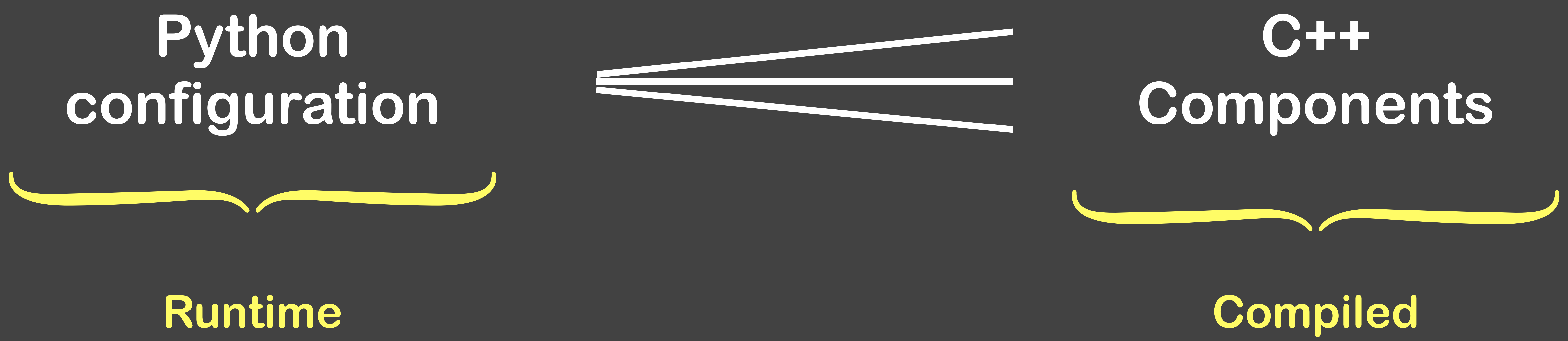
This is classic PyROOT!
More Pythonic methods exist now

Write on file (tree is contained in file)

Based on https://wiki.physik.uzh.ch/cms/root:pyroot_ttree

Python: a configuration language

Experiments started driving their C++ applications with Python



Sneaking into analysis

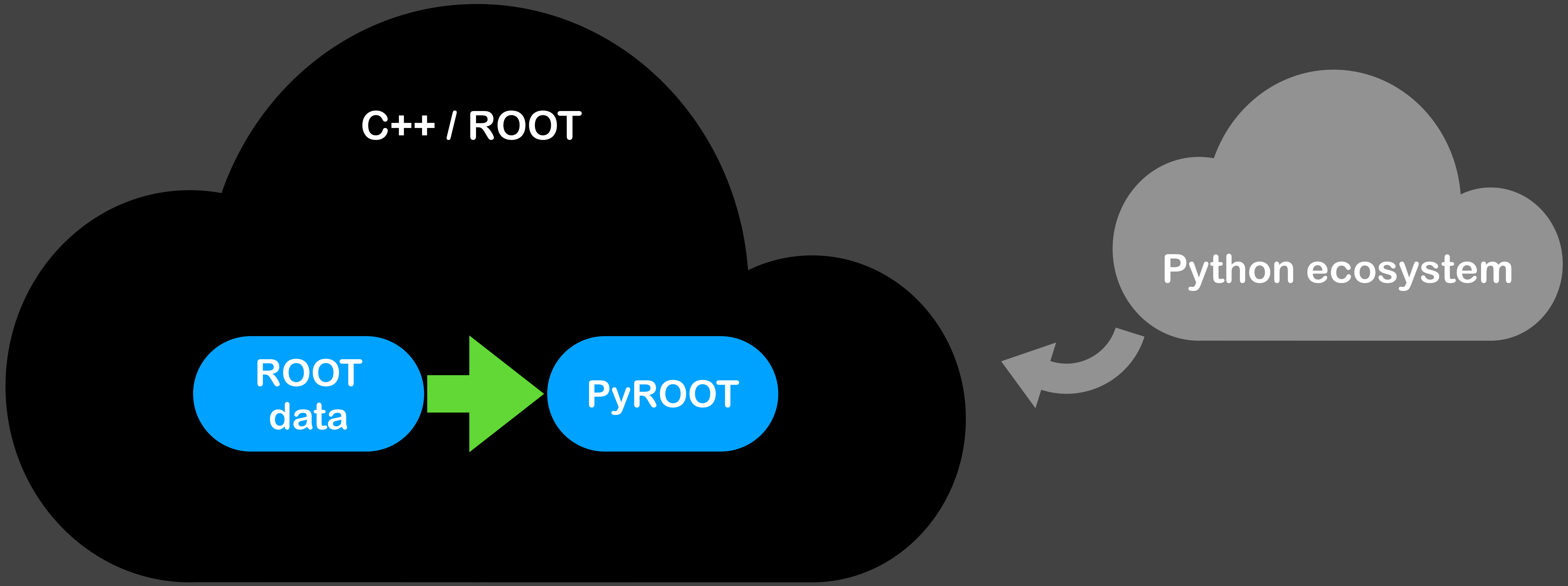
More students were entering with Python knowledge
The Python data-science stack was growing

root-numpy & root-pandas
bridged the gap between ROOT & NumPy/Pandas

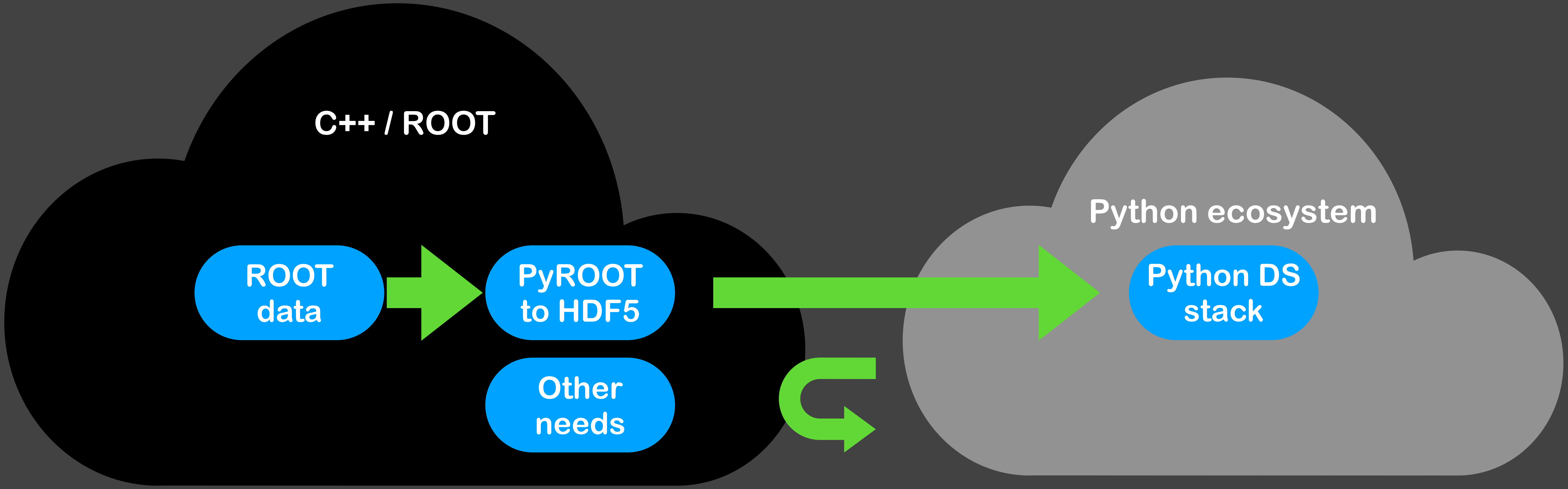
By 2015, most analysis work could be done in the Python data science stack!
A few domain specific things were "missing"

Data IO | Fitting | Jagged Data | Vectors | Particle info

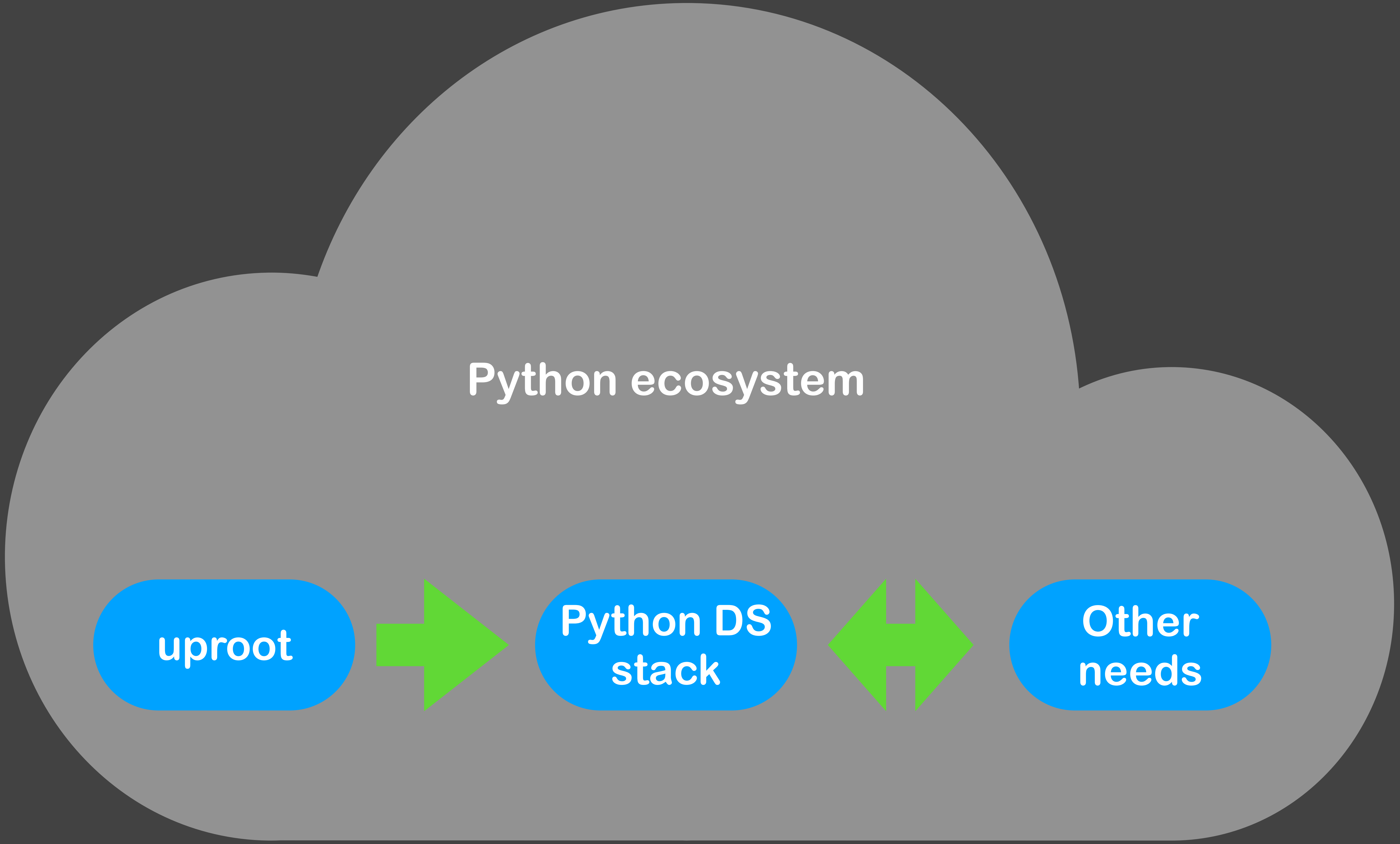
Python analysis circa 2015



Python analysis circa 2015



Python analysis with Scikit-HEP

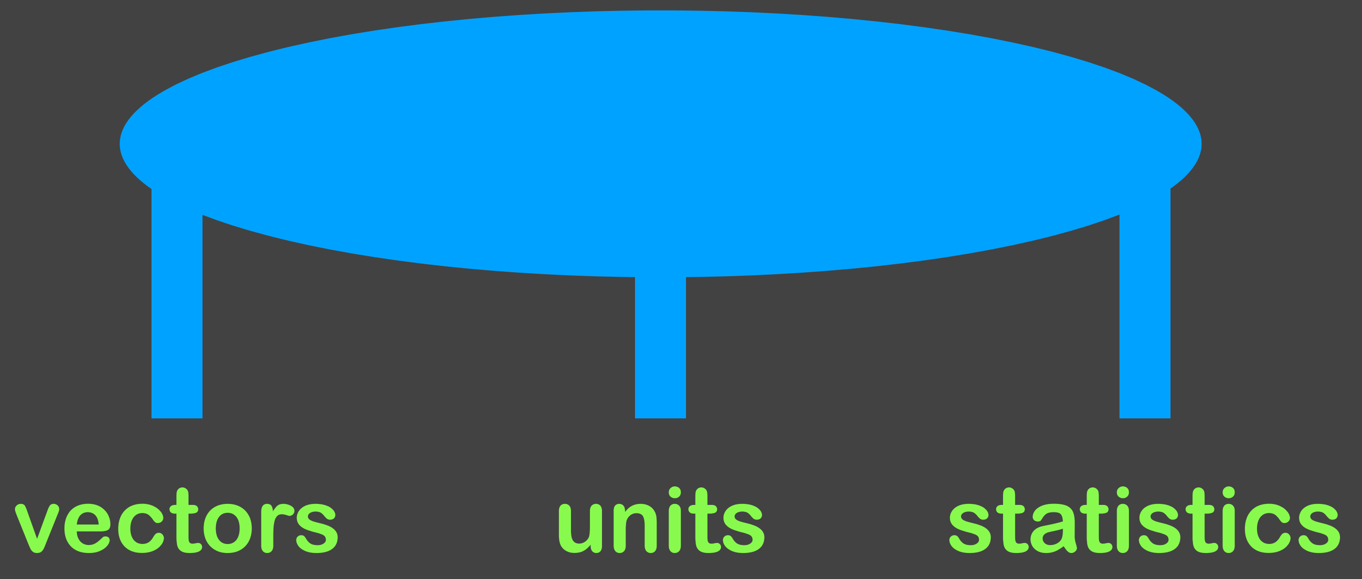


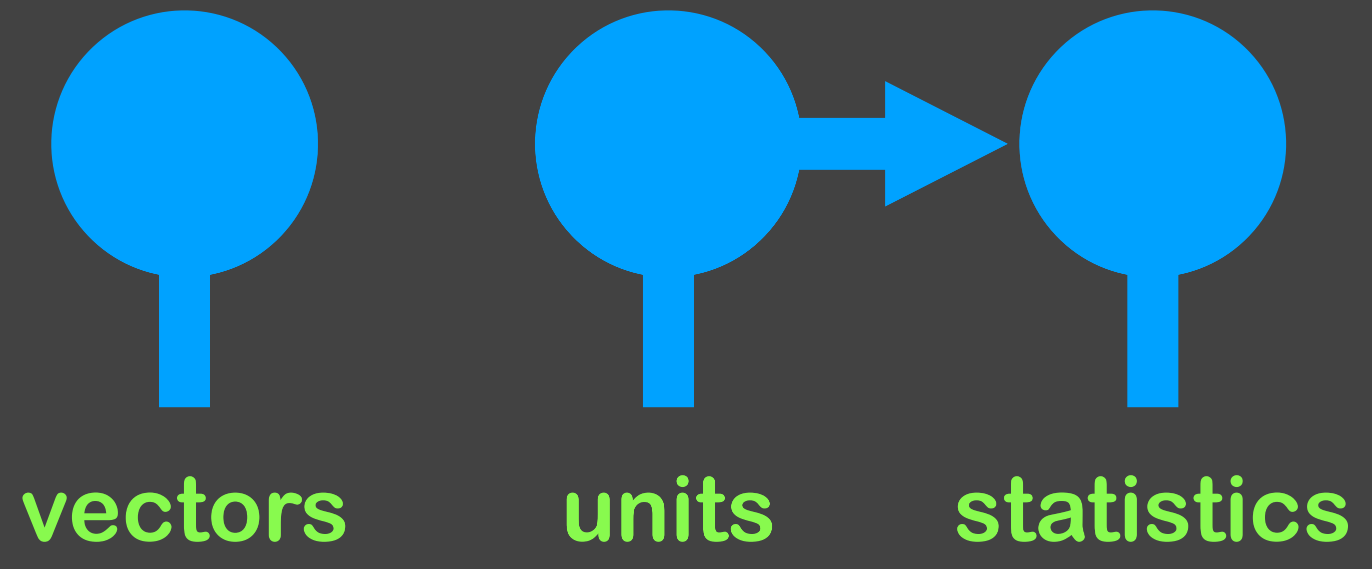
Beginnings of a Scikit

A new package

Eduardo Rodrigues created the Scikit-HEP org with the scikit-hep package in 2016

Toolkit vs. Toolset





Initially all-in-one toolkit approach with topical modules

But this would change…

Joining forces

He also created an organization and invited some other popular packages

ROOTPy was winding down

We got several of the related packages

root-numpy

root-pandas

Several simulation bindings

pyjet

numpythia

And the standalone MINUIT binding

The most popular package to join

iminuit

First major new package

uproot

```python
import uproot
import numpy as np

rng = np.random.default_rng(12345)

px = rng.normal(20, 2, 10_000)
phi = rng.uniform(0, 2*np.pi, 10_000)

with uproot.create("tree.root") as f:
    f["tree_name"] = {"px": px, "phi": phi}
```

ROOT file reader (and then writer)

Just* IO

Pure Python (pip install uproot)

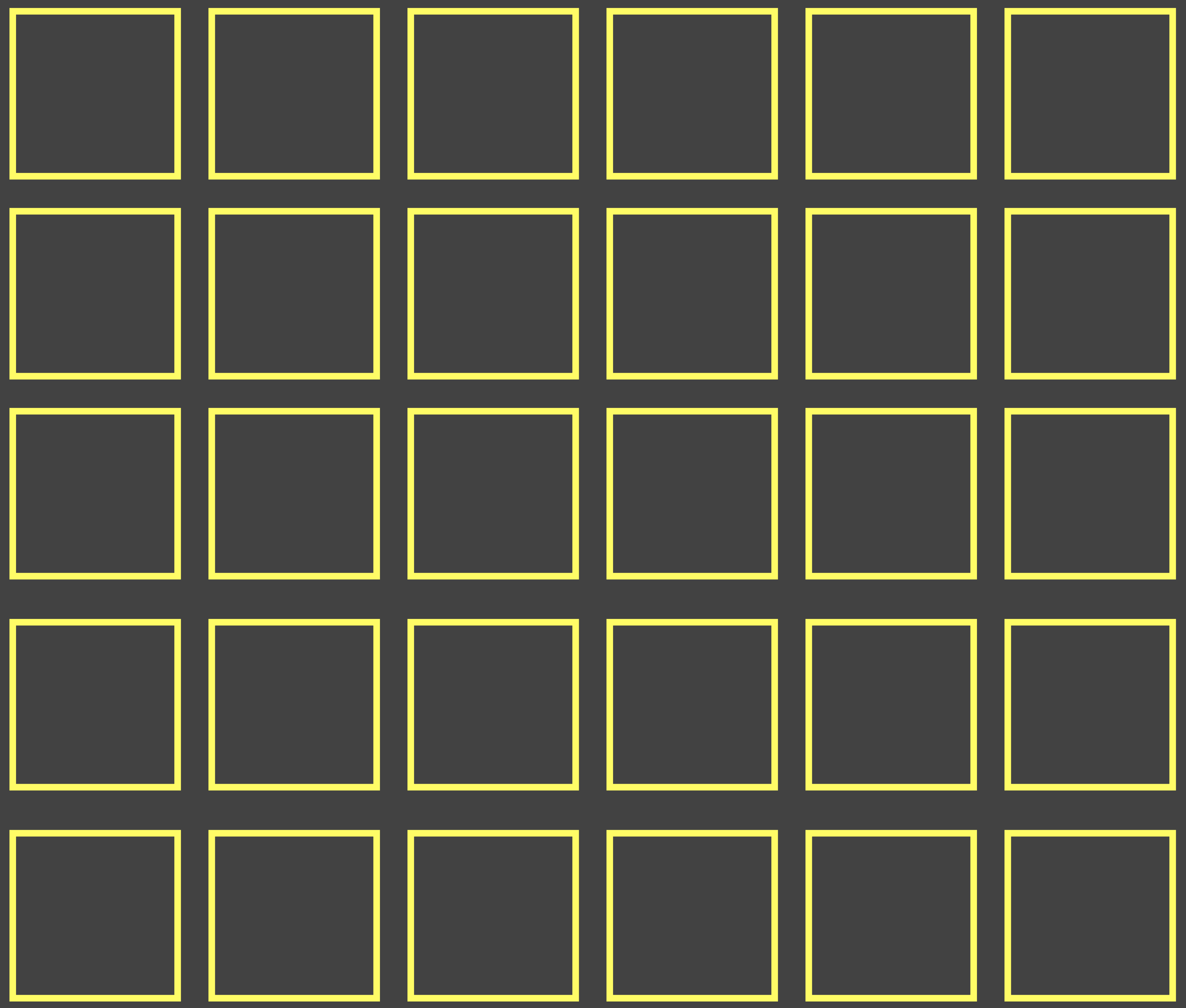
Answer to a key need!

*: Missing functionality in the Python ecosystem became new packages (Awkward, Vector)

First new general package

Awkward Array

Regular array



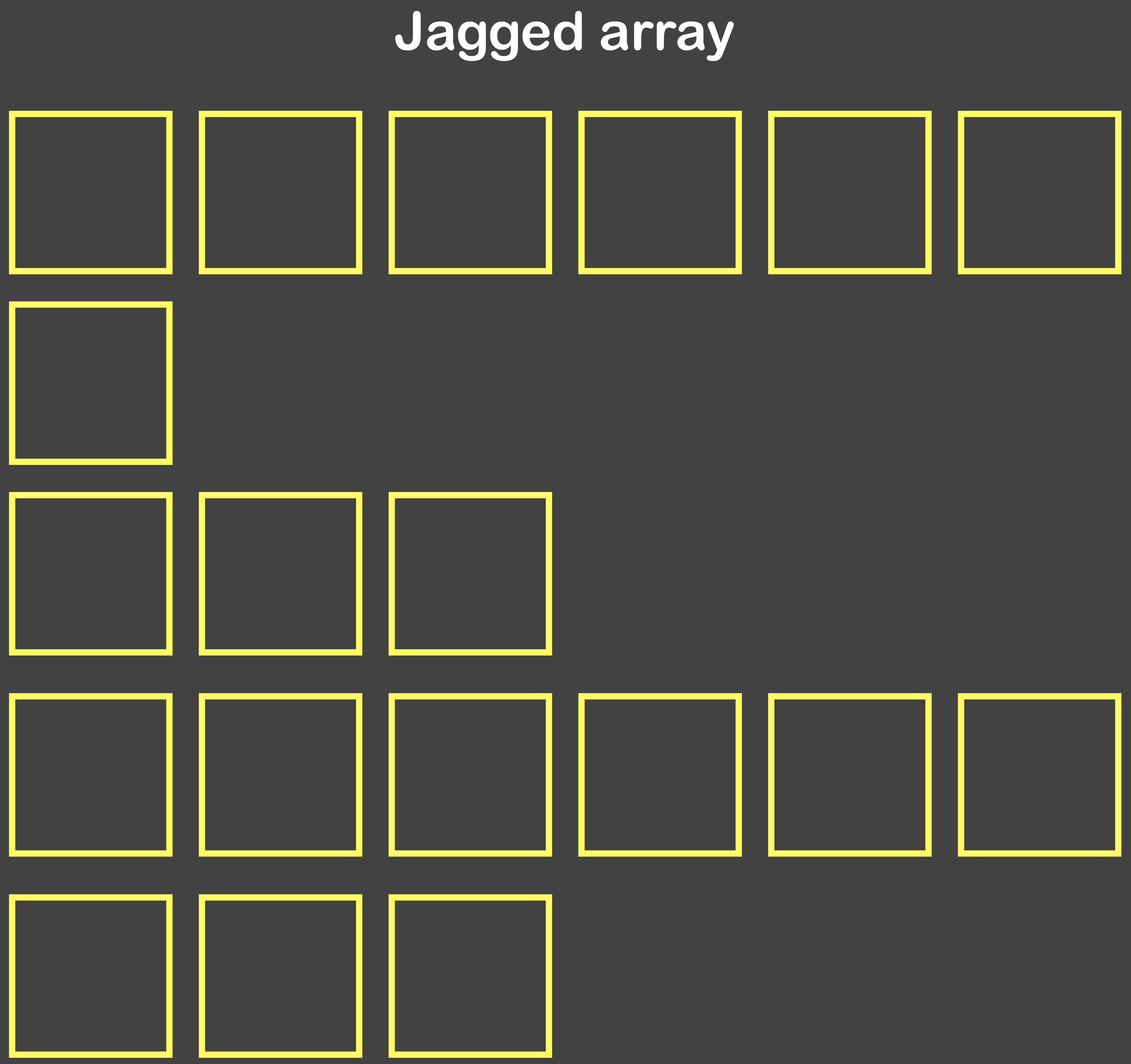
Jagged structured data format
NumPy like manipulation
Assignable “behaviors”
Custom Numba support

Originally HEP focused,
now being developed for 6+ fields

V0: pure Python
V1+: Compiled

First new general package

Awkward Array

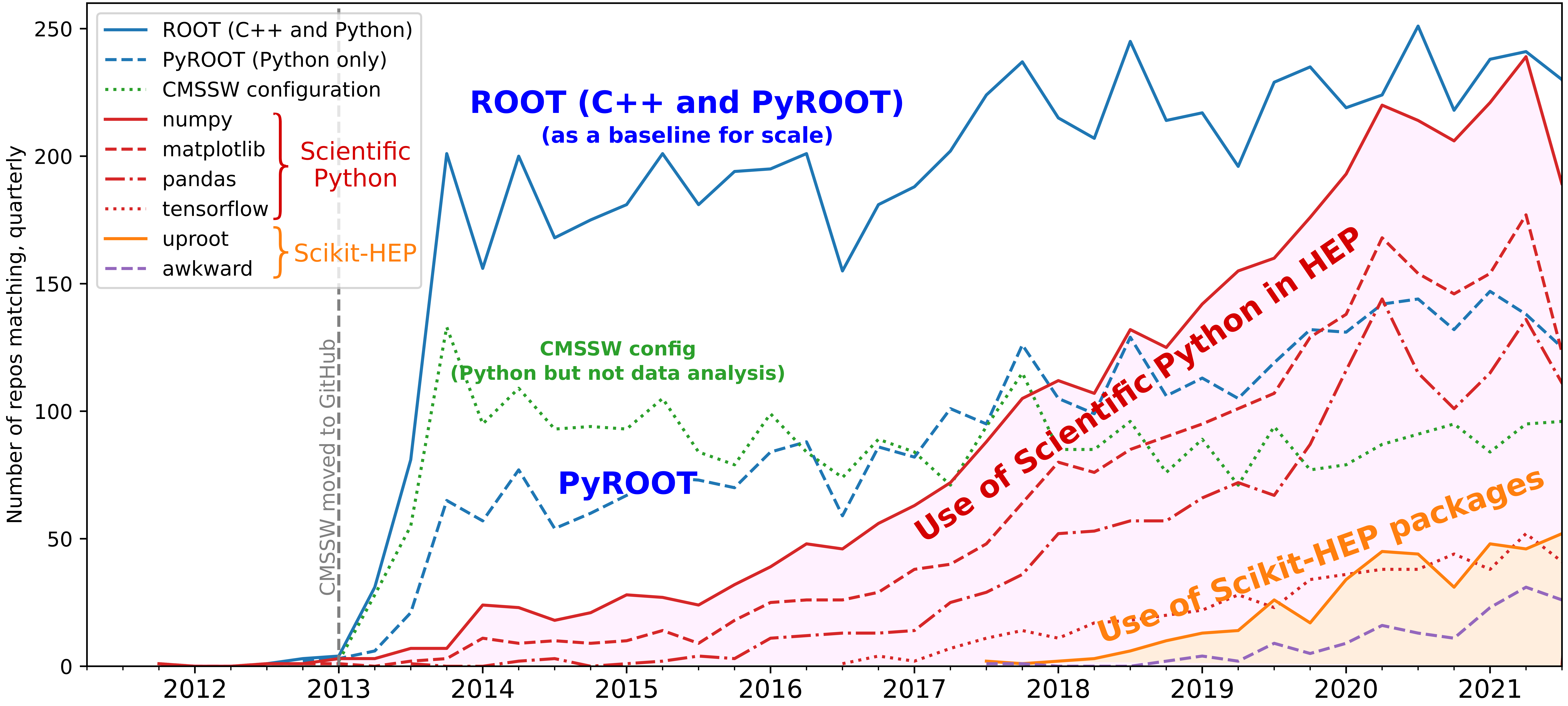


Jagged structured data format
NumPy like manipulation
Assignable “behaviors”
Custom Numba support

Originally HEP focused,
now being developed for 6+ fields

V0: pure Python
V1+: Compiled

Uptake (one expirement - CMS)



Building Scikit-HEP

Case study: Histograms

Work developed here adapted to other packages

Development of reliable bindings

Development of reliable binary building tools

Development of a Protocol based ecosystem

Bindings tools

pybind11

Chose pybind11 over Cython / SWIG

Simpler build (no preprocessing)
Simpler distribution - no NumPy build dependence
Powerful generation capabilities via template meta programming

Integrated improvements upstream

Awkward 1.0 and iMinuit 2 rewrite followed

Building tools

Building redistributable wheels is not straight forward

	Linux	macOS	Windows
Python source	Manylinux docker image	Official download	Anything
Post-process step	Auditwheel	Delocate	Develwheel
Architectures	64/32/ARM/PPC/…	Intel/AS/Universal	32/64/ARM

+ Testing, PyPy, Musllinux, …

First attempt: azure-wheel-helpers

Early attempt at shared infrastructure

Built using git subtree (before Azure templates & remote config support)

Covered by blog-post series
https://iscinumpy.dev/categories/azure-devops/

Great learning experience - but some problems:
Tied to one CI system
Mostly YAML files - poor testability / Code QA
Small number of users

Adopted by:
Boost-histogram
awkward-array
pyjet
numpythia
iminuit

Using cibuildwheel

cibuildwheel

We merged our improvements to cibuildwheel & joined that project!
All projects moved (and now pyhepmc added too)

Great positives:

Not tied to a CI system (easy move to GHA)
Python package - great testability / code QA
Large number of users

From azure-wheel-helpers:

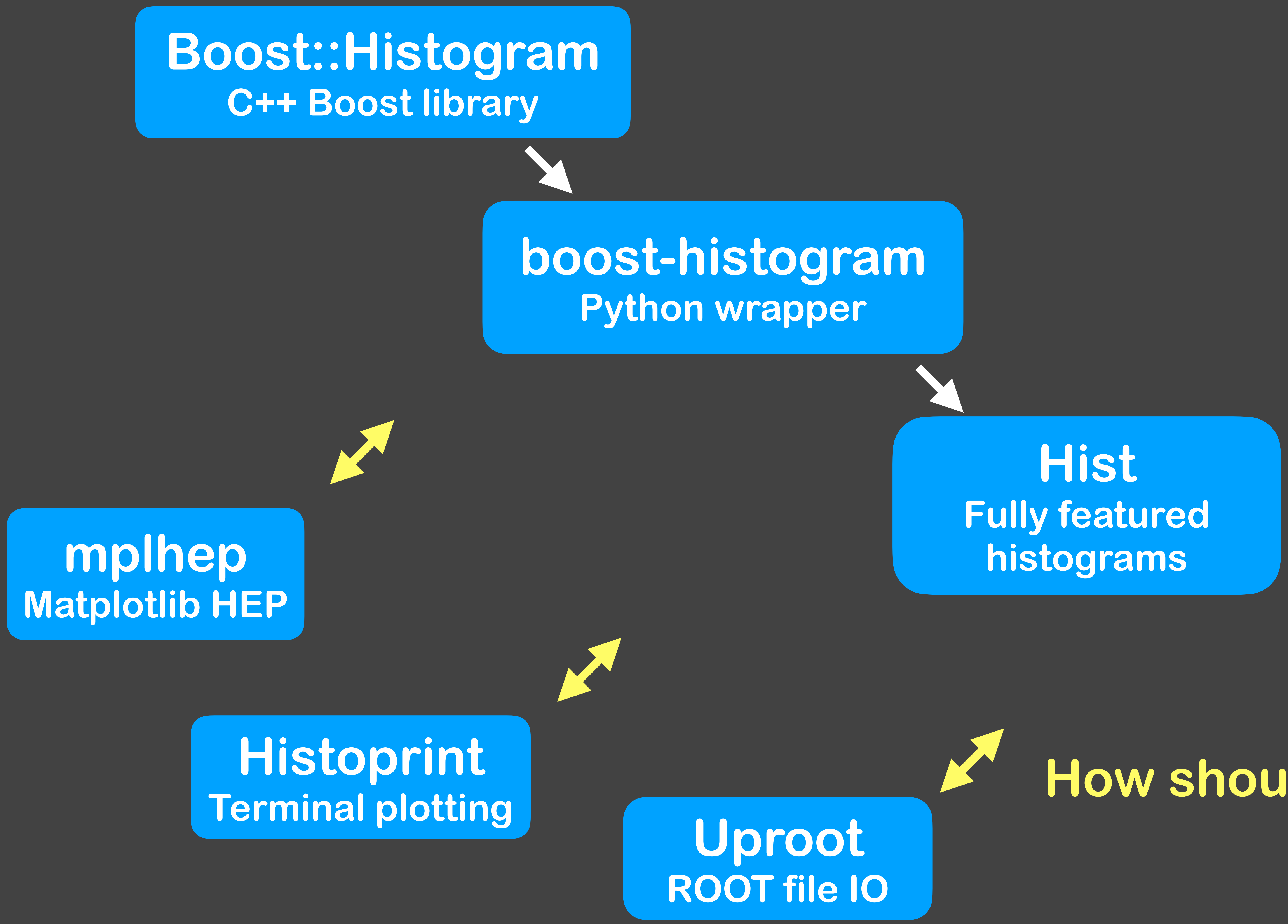
Better Windows support
VSC versioning support
Better PEP 518 support

Contributions from Scikit-HEP:

Static configuration in TOML
Static overrides
Direct SDist support
build (the tool) support
Automatic Python version limit detection
Better globbing support
Full static typing & more Code QA

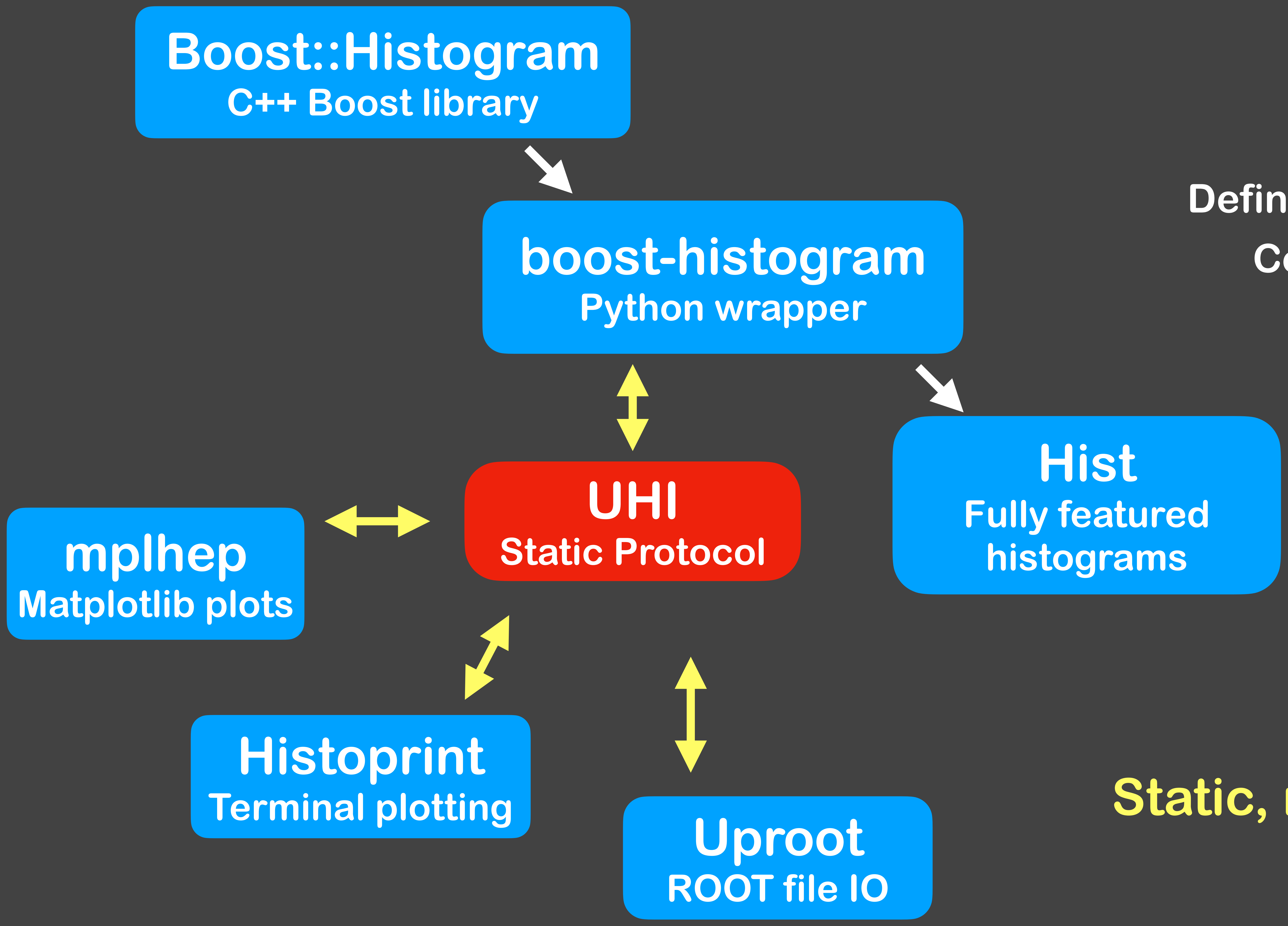
And we helped get cibuildwheel into the PyPA!

Protocol ecosystem



How should these be related?

Protocol ecosystem



`PlottableHistogram`

Defines expected behaviors for producers

Consumers expect these behaviors

Static, not runtime dependency!

Example of a Protocol

Definition

```python
from typing import Protocol

class Duck(Protocol):
    def quack() -> str: ...
```

Producer

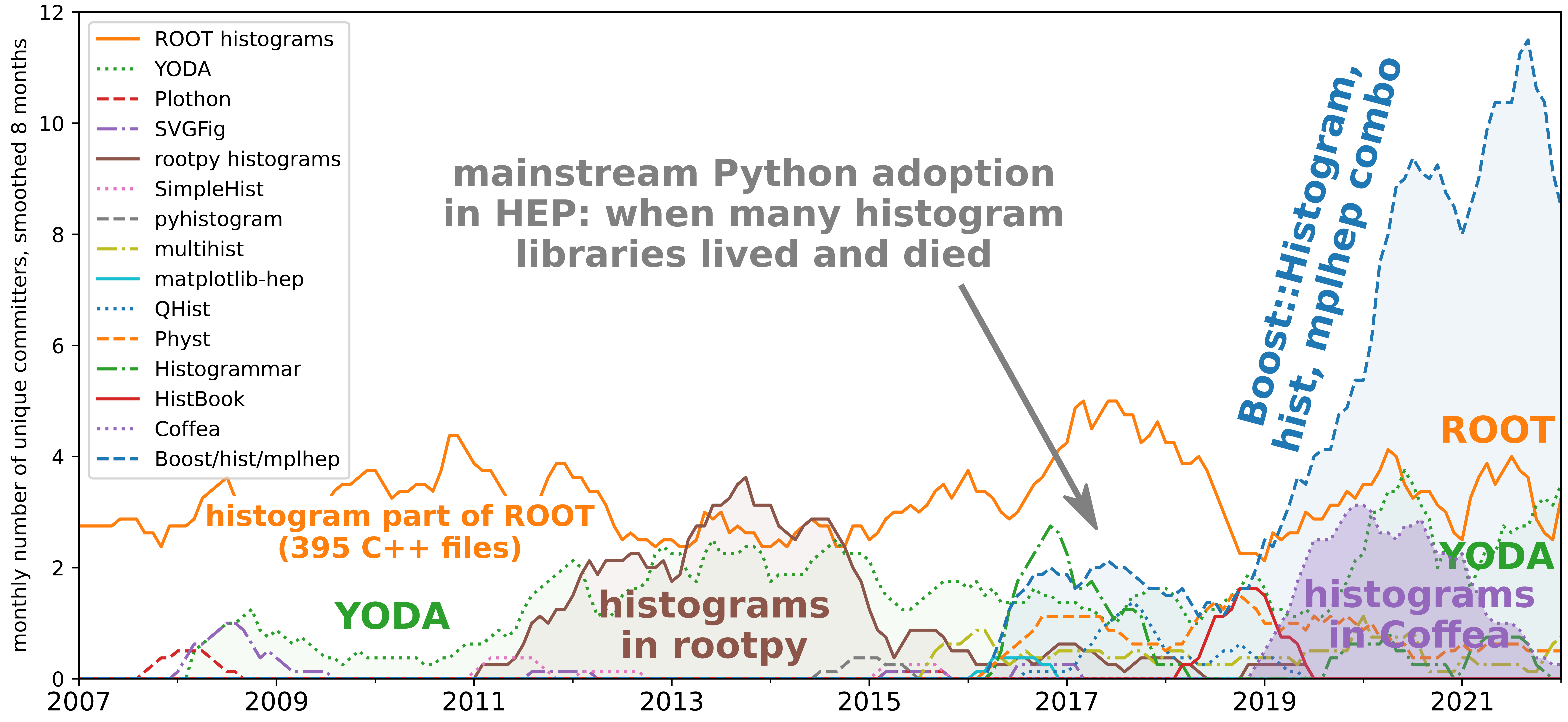
```python
class MyDuck:
    def quack() -> str:
        return "quack"

if typing.TYPE_CHECKING:
    _: Duck = typing.cast(MyDuck, None)
```

Consumer

```python
def need_a_duck(duck: Duck) -> None:
    print(duck.quack())
```

No runtime dependence, unlike ABC!
All static type annotations
Validation by checker

Success of histograms



All together: uproot-browser

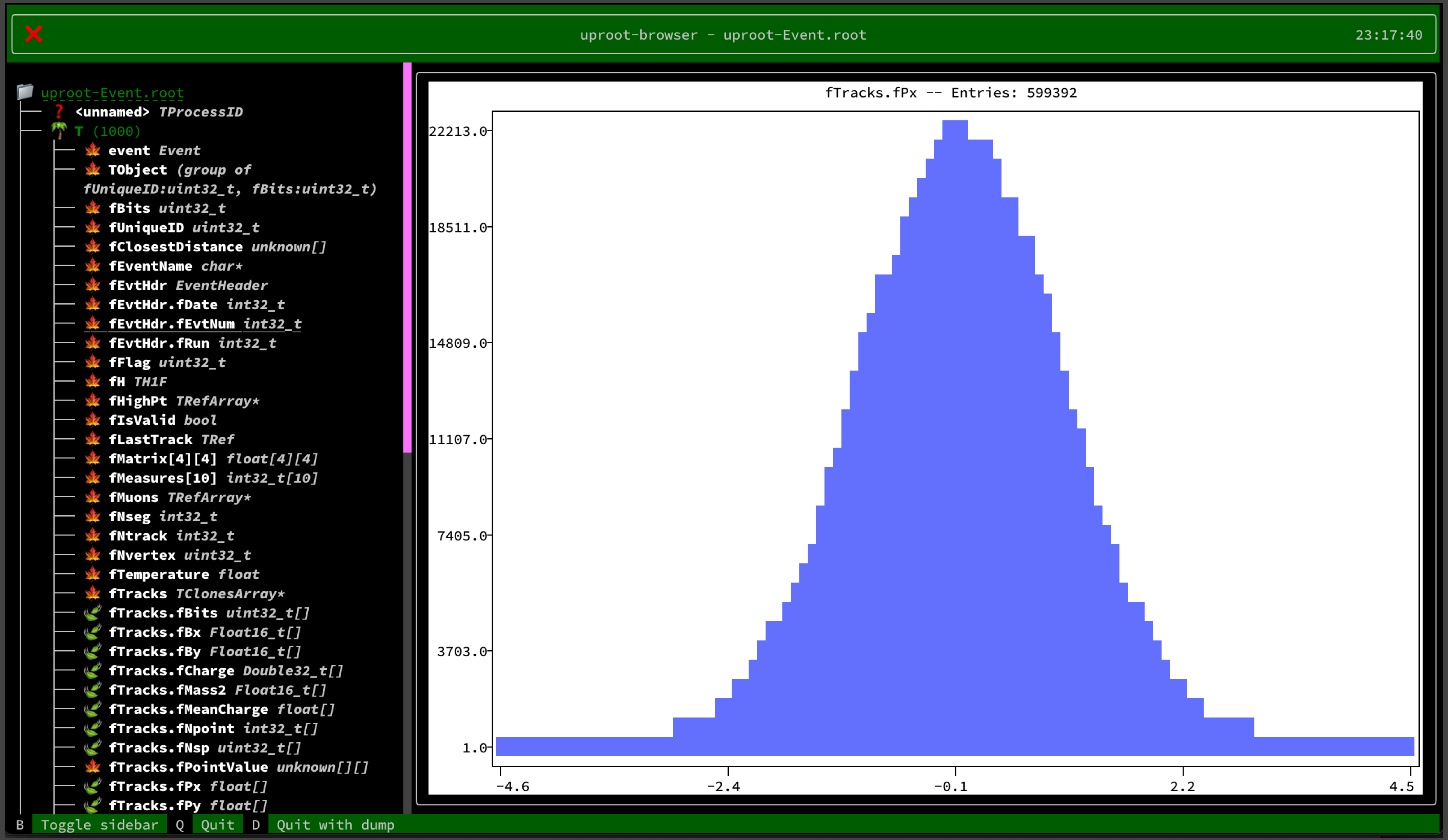
Textual + Rich UI

uproot file

Awkward data

Hist computation

plotext rendering



```
✕                     uproot-browser – uproot-Event.root                     23:17:40

📁 uproot-Event.root
├── ❓ <unnamed> TProcessID
└── 🌴 T (1000)
    ├── 🍁 event Event
    ├── 🍁 TObject (group of
    │   fUniqueID:uint32_t, fBits:uint32_t)
    ├── 🍁 fBits uint32_t
    ├── 🍁 fUniqueID uint32_t
    ├── 🍁 fClosestDistance unknown[]
    ├── 🍁 fEventName char*
    ├── 🍁 fEvtHdr EventHeader
    ├── 🍁 fEvtHdr.fDate int32_t
    ├── 🍁 fEvtHdr.fEvtNum int32_t
    ├── 🍁 fEvtHdr.fRun int32_t
    ├── 🍁 fFlag uint32_t
    ├── 🍁 fH TH1F
    ├── 🍁 fHighPt TRefArray*
    ├── 🍁 fIsValid bool
    ├── 🍁 fLastTrack TRef
    ├── 🍁 fMatrix[4][4] float[4][4]
    ├── 🍁 fMeasures[10] int32_t[10]
    ├── 🍁 fMuons TRefArray*
    ├── 🍁 fNseg int32_t
    ├── 🍁 fNtrack int32_t
    ├── 🍁 fNvertex uint32_t
    ├── 🍁 fTemperature float
    ├── 🍁 fTracks TClonesArray*
    ├── 🌿 fTracks.fBits uint32_t[]
    ├── 🌿 fTracks.fBx Float16_t[]
    ├── 🌿 fTracks.fBy Float16_t[]
    ├── 🌿 fTracks.fCharge Double32_t[]
    ├── 🌿 fTracks.fMass2 Float16_t[]
    ├── 🌿 fTracks.fMeanCharge float[]
    ├── 🌿 fTracks.fNpoint int32_t[]
    ├── 🌿 fTracks.fNsp uint32_t[]
    ├── 🍁 fTracks.fPointValue unknown[][]
    ├── 🌿 fTracks.fPx float[]
    ├── 🌿 fTracks.fPy float[]

B  Toggle sidebar  Q  Quit  D  Quit with dump
```

Broader Ecosystem

Scikit-HEP today

Domain
Specifics
Physics
Core
Affiliated package
Decay Language
numpythia
pyhepmc
Scikit HEP
UPROOT-BROWSER
FASTJET
nndrone
pyhf
∂ifferentiable
ℒikelihoods
cabinetry
hepstats
mplhep
pylhe
VECTOR
uproot
hepunits
Coffea
(/ˈkɔ.fi/)
Awkward Array
Particle
hist
uhi
Vega Scope
GooFit
CUDA/OpenMP Fitting Framework for C++ & Python
histoprint
Boost Histogram
jupyter
NumPy
matplotlib
iminuit
zfit
DASK
python
Numba

Many contributions upstream

Helping with maintaining several projects

Affiliated status helped

Upstreaming features and fixes

We can all benefit!

Scikit-HEP Developer Pages

Scikit-HEP developer pages

Scikit-HEP was growing quickly
How to keep quality high across all packages?
scikit-hep.org/developer!

Topics
- Packaging (classic & simple)
- Style guide (pre-commit)
- Development setup
- pytest
- Static typing (MyPy)
- GitHub Actions (3 pages)
- Nox

Utilities
- Cookie-cutter
- Repo-review

Maintained by nox
& GitHub Actions

Guided our other packages, like Awkward

Universally useful advice for maintaining packages!

Search Scikit-HEP

Scikit-HEP

Home

Project news

Packages

User information

Developer information

Who uses Scikit-HEP?

About

Scikit-HEP project - welcome!

The Scikit-HEP project is a community-driven and community-oriented project with the aim o providing Particle Physics at large with an ecosystem for data analysis in Python. Read more

New users can start with our user pages. See our developer pages for information on develop Python packages.

NEWS • TUTORIAL • RESOURCES • CITE US • GET IN TOUCH

Basics:

Awkward Array	Manipulate JSON-like data with NumPy-like idioms.
hepunits	Units and constants in the HEP system of units.
VECTOR	Manipulate Lorentz, 3D, and 2D vectors in NumPy, Numba, or Aw

Style

Pre-commit hooks
Black
Check-Manifest (setuptools only)
Type checking
PyCln
Flake8
YesQA
isort
PyUpgrade
Setup.cfg format (setuptools only)
Spelling
PyGrep hooks
Prettier
Clang-format (C++ only)
Shellcheck (shell scripts only)
PyLint (noisy)

Modern (simple) packaging

pyproject.toml

```toml
[build-system]
requires = ["hatchling"]
build-backend = "hatchling.build"

[project]
name = "package"
version = "0.1.2"
```

No setup.py, setup.cfg, or MANIFEST.in required!

Cookiecutter

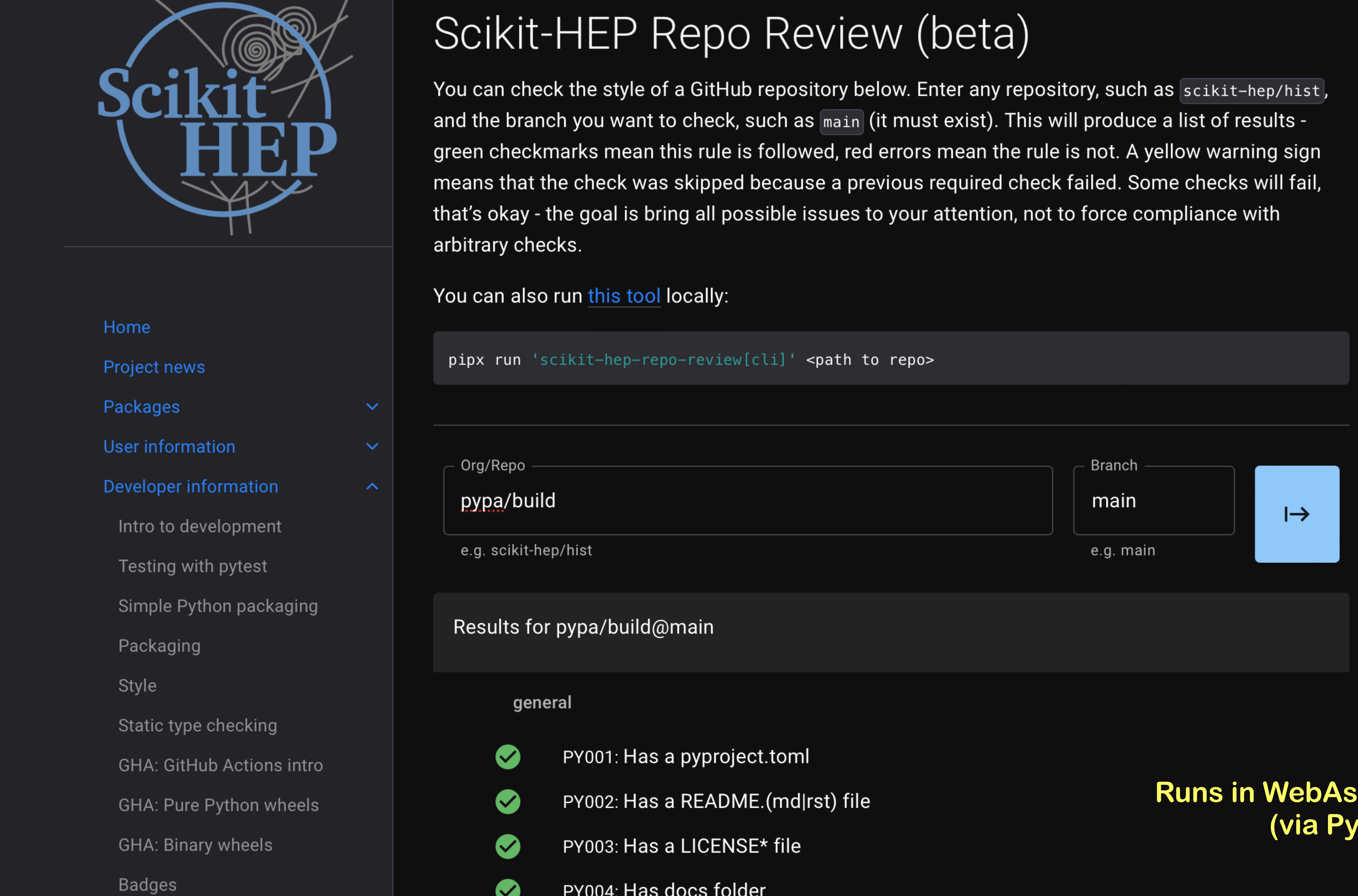
```
pipx run cookiecutter gh:scikit-hep/cookie
```

11 backends to pick from
Generation tested by nox
In sync with the developer pages

Setuptools
Setuptools PEP 621
Flit Hatch PDM
Poetry

Scikit-build
Setuptools C++
Maturin (Rust)

+ more!



Home

Project news

Packages

User information

Developer information

Intro to development

Testing with pytest

Simple Python packaging

Packaging

Style

Static type checking

GHA: GitHub Actions intro

GHA: Pure Python wheels

GHA: Binary wheels

Badges

Scikit-HEP Repo Review (beta)

You can check the style of a GitHub repository below. Enter any repository, such as `scikit-hep/hist`, and the branch you want to check, such as `main` (it must exist). This will produce a list of results - green checkmarks mean this rule is followed, red errors mean the rule is not. A yellow warning sign means that the check was skipped because a previous required check failed. Some checks will fail, that's okay - the goal is bring all possible issues to your attention, not to force compliance with arbitrary checks.

You can also run this tool locally:

```
pipx run 'scikit-hep-repo-review[cli]' <path to repo>
```

Org/Repo: pypa/build
e.g. scikit-hep/hist

Branch: main
e.g. main

Results for pypa/build@main

general

✓ PY001: Has a pyproject.toml

✓ PY002: Has a README.(md|rst) file

✓ PY003: Has a LICENSE* file

✓ PY004: Has docs folder

**Runs in WebAssembly locally!
(via Pyodide)**

The Future

Scikit-build

CMake bridge for Python packaging

Year 1 plans:

Introduce scikit-build-core: modern PEP 517 backend
Support PEP 621 configuration
Support use as plugin (possibly via extesionlib)
Tighter CMake integration (config from PyPI packages)
Distutils-free code ready for Python 3.12

Year 2 plans:

Convert selected projects to Scikit-build

Year 3 plans:

Website, tutorials, outreach

https://iscinumpy.dev/post/scikit-build-proposal/

Scikit-build

CMake bridge for Python packaging

Year 1 plans:

Introduce scikit-build-core: modern PEP 517 backend
Support PEP 621 configuration
Support use as plugin (possibly via extesionlib)
Tighter CMake integration (config from PyPI packages)
Distutils-free code ready for Python 3.12

Year 2 plans:

Convert selected projects to Scikit-build

Year 3 plans:

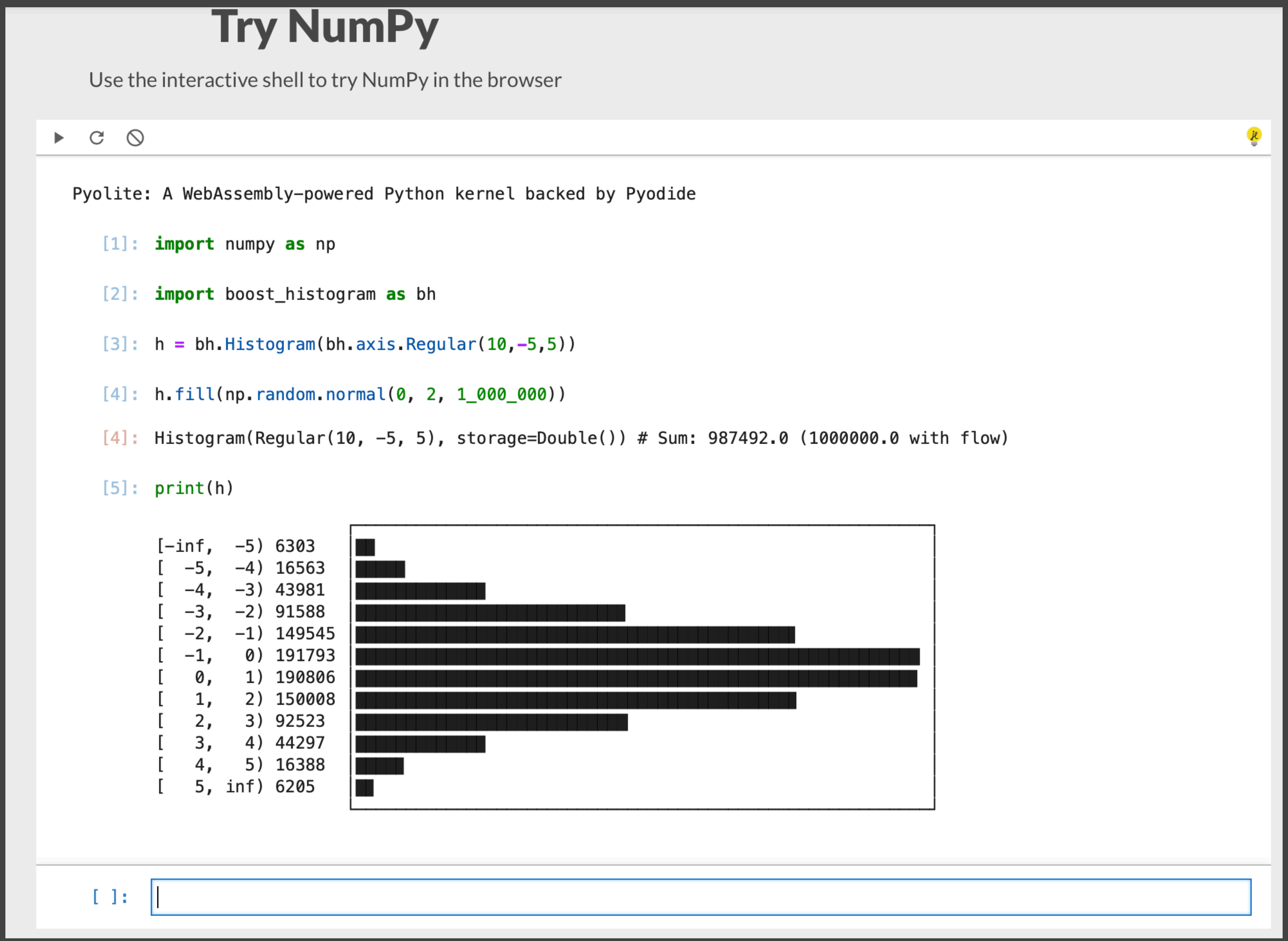
Website, tutorials, outreach

https://iscinumpy.dev/post/scikit-build-proposal/

Yesterday, NSF 2209877 was awarded!

Web Assembly

Already have boost-histogram (pybind11 project) added to pyodide!
iminuit (better CMake support) next, then work on Awkward Array!



Try NumPy

Use the interactive shell to try NumPy in the browser

```
Pyolite: A WebAssembly-powered Python kernel backed by Pyodide

[1]: import numpy as np

[2]: import boost_histogram as bh

[3]: h = bh.Histogram(bh.axis.Regular(10,-5,5))

[4]: h.fill(np.random.normal(0, 2, 1_000_000))

[4]: Histogram(Regular(10, -5, 5), storage=Double()) # Sum: 987492.0 (1000000.0 with flow)

[5]: print(h)

     [-inf,  -5) 6303
     [  -5,  -4) 16563
     [  -4,  -3) 43981
     [  -3,  -2) 91588
     [  -2,  -1) 149545
     [  -1,   0) 191793
     [   0,   1) 190806
     [   1,   2) 150008
     [   2,   3) 92523
     [   3,   4) 44297
     [   4,   5) 16388
     [   5, inf) 6205

[ ]:
```

← numpy.org

Questions?

- History of Python in HEP
  - ROOT, PyROOT, Analysis
- Beginnings of a Scikit
  - New package, joining forces
  - Uproot, Awkward Array
- Building Scikit-HEP
  - Histograms case study
  - All together: uproot-browser
- Broader ecosystem
- Scikit-HEP Developer Pages
- The Future
  - Scikit-build
  - Web Assembly

https://scikit-hep.org

https://iscinumpy.dev