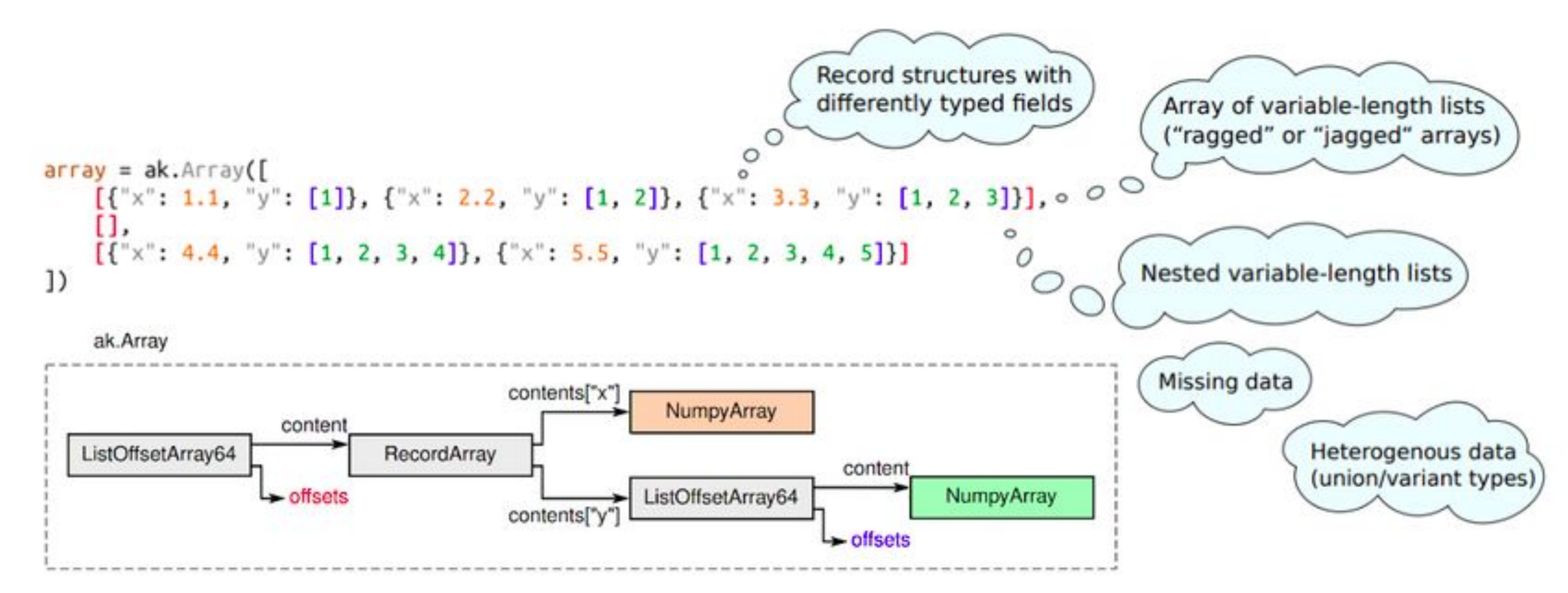
Angus Hollands

•••

Princeton University

About Me

- Based in the UK
- Background in LEP (nuclear) @ University of Birmingham
- Working @ Princeton as an SRA → postdoc
  - Mainly on `awkward` array
- Interest in Software Design



```
array = ak.Array([
    [{"x": 1.1, "y": [1]}, {"x": 2.2, "y": [1, 2]}, {"x": 3.3, "y": [1, 2, 3]}],
    [],
    [{"x": 4.4, "y": [1, 2, 3, 4]}, {"x": 5.5, "y": [1, 2, 3, 4, 5]}]
])
```

Record structures with differently typed fields

Array of variable-length lists (“ragged” or “jagged“ arrays)

Nested variable-length lists

Missing data

Heterogenous data (union/variant types)

Awkward Array Structure

Awkward Goals

- Further improvements to `dask-awkward`
- Vectorised string operations
- Serialisable & transient metadata
- Unit conversion

Interest in

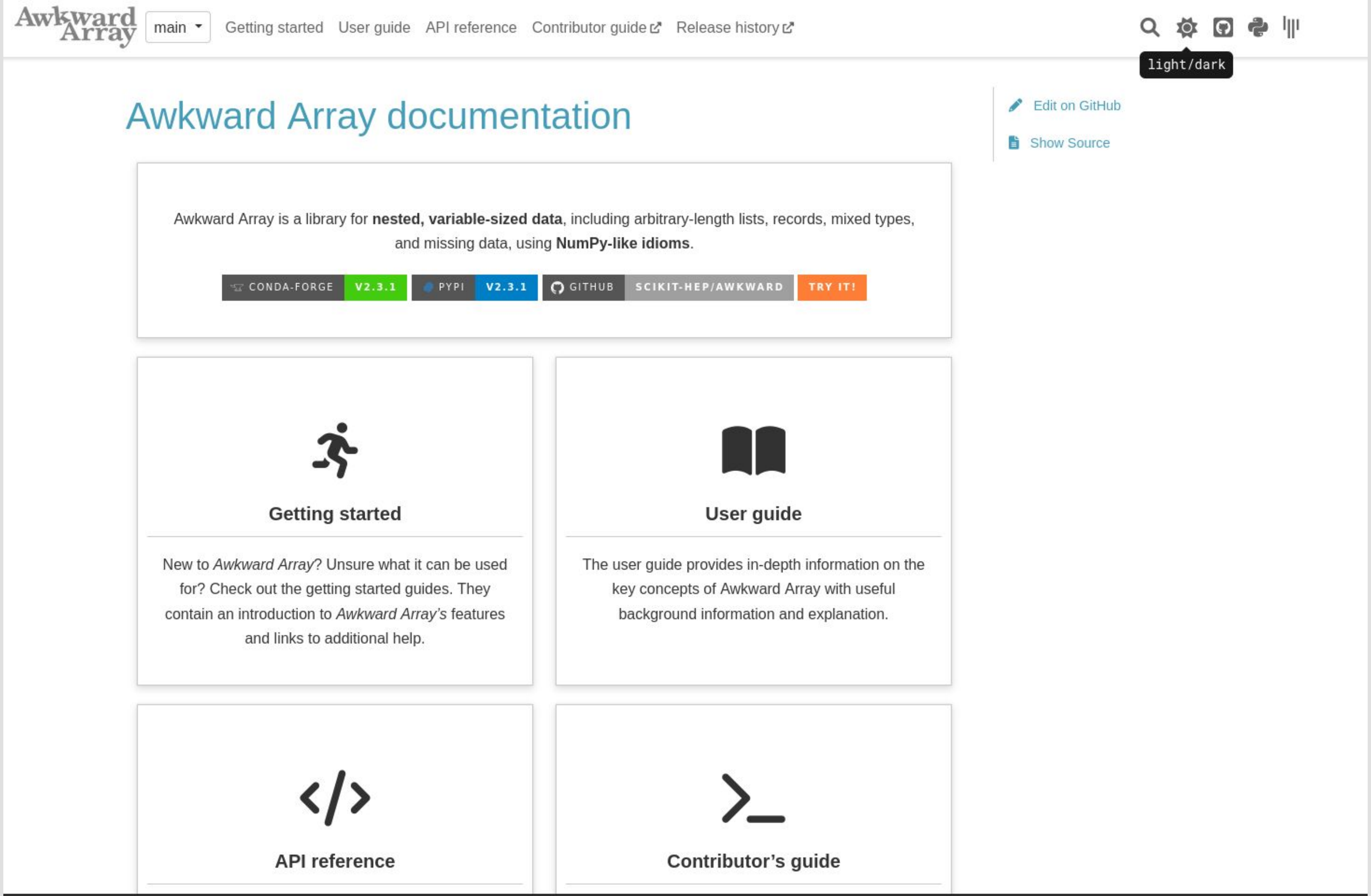
Packaging

- Helping developers upgrade to PEP 517/621
- Supporting transition to modern Python build chains
  - (see Awkward Array)
- Maintaining `conda-forge` recipes

Interest in

Documentation

- Benefits of Jupyter Book for user-guides & tutorials
- Replacement of ReST with MyST
  - Future work around cross-project documentation (e.g. papyri)



Awkward Array documentation

Awkward Array is a library for **nested, variable-sized data**, including arbitrary-length lists, records, mixed types, and missing data, using **NumPy-like idioms**.

Getting started

New to *Awkward Array*? Unsure what it can be used for? Check out the getting started guides. They contain an introduction to *Awkward Array’s* features and links to additional help.

User guide

The user guide provides in-depth information on the key concepts of Awkward Array with useful background information and explanation.

API reference

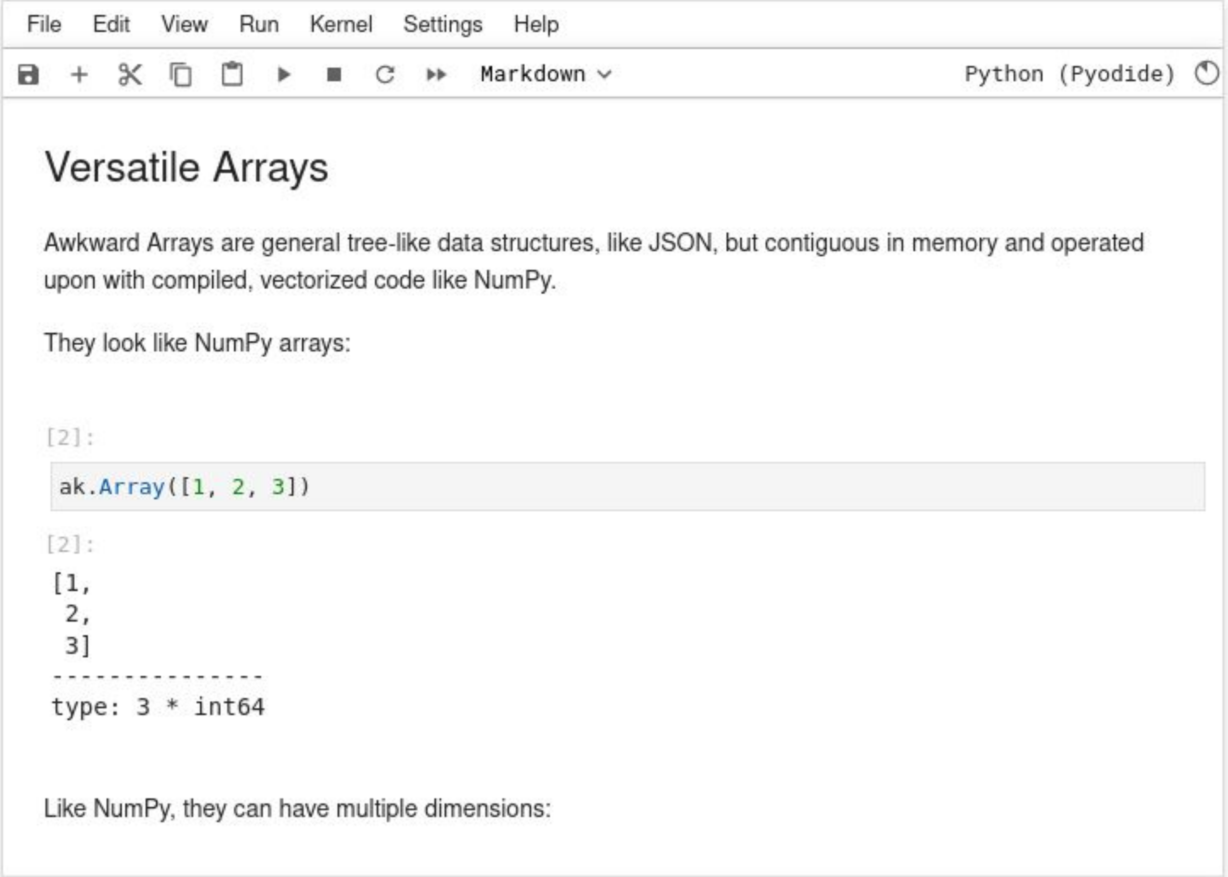
Contributor’s guide

Interest in

Deployment

- Novel applications of JupyterLite
  - Replace Binder given loss of capacity
  - Support web-first workflows & experiences (e.g. docs)
- Emscripten-forge for build-together WASM ecosystem

Try it



Versatile Arrays

Awkward Arrays are general tree-like data structures, like JSON, but contiguous in memory and operated upon with compiled, vectorized code like NumPy.

They look like NumPy arrays:

```
[2]:
ak.Array([1, 2, 3])
```

```
[2]:
[1,
 2,
 3]
---------------
type: 3 * int64
```

Like NumPy, they can have multiple dimensions:

Interest in

Open Source Development

- How to engage the “silent majority”?
- Best practices for fostering community contributions?
- Public-facing development for small teams?

Interest in

- Differential Programming
- Combinatorics
- Array Programming
- Two-language Problem
- Jupyter