h5pydantic, from a Synchrotron through Python to HDF5

Clinton Roy

ANSTO
Australian Synchrotron

PyCon AU 2023

Outline

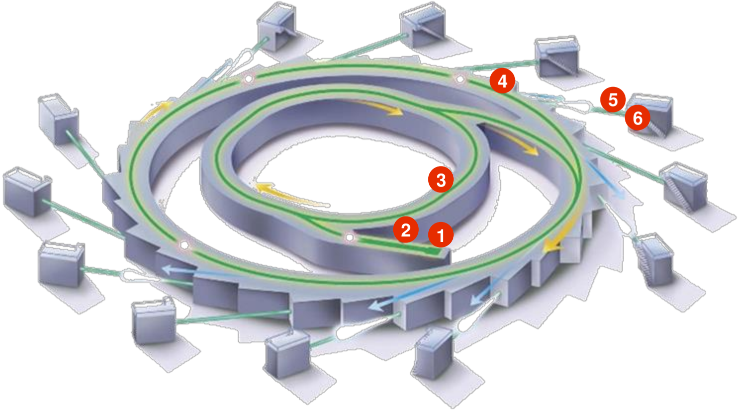
Synchrotron

HDF5

`h5pydantic`

Australian Synchrotron

Accelerating and bending electrons



1 Electron Gun

2 Linear Accelerator

3 Booster Ring

4 Storage Ring

5 Beamline

6 End Station

Lots of [meta]data

- sample information
  - chemistry, volume, concentration, thickness
  - strain, temperature, shearing, magnetic, UV light, chemical, pressure

Lots of [meta]data

- sample information
  - chemistry, volume, concentration, thickness
  - strain, temperature, shearing, magnetic, UV light, chemical, pressure
- x-ray beam setup
  - energy
  - distance between sample and detector
  - xray detector images
  - detector exposure time

Lots of [meta]data

- sample information
  - chemistry, volume, concentration, thickness
  - strain, temperature, shearing, magnetic, UV light, chemical, pressure
- x-ray beam setup
  - energy
  - distance between sample and detector
  - xray detector images
  - detector exposure time
- calibration
  - detector images with no beam
  - detector images with beam, but no sample
  - position of beam on detector

ANSTO

Lots of [meta]data

- sample information
  - chemistry, volume, concentration, thickness
  - strain, temperature, shearing, magnetic, UV light, chemical, pressure
- x-ray beam setup
  - energy
  - distance between sample and detector
  - xray detector images
  - detector exposure time
- calibration
  - detector images with no beam
  - detector images with beam, but no sample
  - position of beam on detector
- processing arguments
  - different statistical methods and arguments
  - different assumptions

What Users and Scientists want

- All the data and metadata together
- Ideally in one file

HDF5 Files

- One file, with internal structure
- Archive file, think tar/zip

HDF5 Core Concepts

HDF5 Core Concepts

1. Groups
   - which are basically directories/folders

HDF5 Core Concepts

1. Groups
   - which are basically directories/folders
2. Datasets
   - multi dimensional arrays
   - 2D arrays commonly used to store images

HDF5 Core Concepts

1. Groups
   - which are basically directories/folders
2. Datasets
   - multi dimensional arrays
   - 2D arrays commonly used to store images
3. Attributes
   - key, value metadata
   - can attach to Groups and Datasets

Hand written, hand wavy hdf5 contents

```
/sample
/sample/thickness = 2.5
/sample/concentration = 3.0
/beam
/beam/energy = 1.8
/beam/distance = 7.8
/calibration
/calibration/no_beam = [0 0 0 0 1 0 0 1 0 ...]
/calibration/position/x = 247
/calibration/position/y = 253
/processing
/processing/threshold = 25
```

Hand written calibration

```
import h5py

h5file = h5py.File("experiment.hdf5", "w")

calibration_group = h5file.create_group("calibration")

dataset = calibration_group.create_dataset("no_beam",
                                           [0, 0, 0, 1, 0, 0])

position = calibration_group.create_group("position")
position.attrs["x"] = 247
position.attrs["y"] = 253
```

Introducing `h5pydantic`

- Groups - Model
- Dataset - Model
- Attributes - fields of Groups/Datasets

Defining the model with h5pydantic

```
from h5pydantic import H5Group, H5Dataset, H5Integer32

class Position(H5Group):
    x: H5Integer32
    y: H5Integer32

class CalibImage(H5Dataset, dtype=H5Integer32, shape=(3,2)):
    pass

class Calibration(H5Group):
  no_beam: CalibImage = CalibImage()
  position: Position

class Experiment(H5Group):
    calibration: Calibration
```

Using the Model

```
from model import Experiment

exp = Experiment(calibration={"position": {"x": 247, "y": 253}})
exp.calibration.no_beam.data([[0, 0, 1], [0, 0, 0]])
exp.dump("experiment.hdf5")
```

And later on...

```
from model import Experiment

exp = Experiment.load("experiment.hdf5")
(x, y) = exp.calibration.position.x, exp.calibration.position.y
```

Pydantic Fields

- Can extend pydantic fields with all sorts of information
  - Documentation
  - Units
  - Relationships between fields

Pydantic Fields example

```
from h5pydantic import H5Group, H5Integer32
from pydantic import Field

import pint
ureg = pint.UnitRegistry()

class Beam(H5Group):
    energy: float = Field(ge=0, doc="X-Ray beam energy", unit=ureg.joule)
    distance: H5Integer32 = Field(ge=0,
                                  doc="Distance from sample to detector",
                                  unit=ureg.millimeter)

beam = Beam(energy=3.4, distance=7500)
```

Outputs

- Can output a HDF5 description in PDF, using Sphinx
- Can start to handle versioning

Design

The overall design of `h5pydantic` has three main aims:

1. All instantiated models can be saved to HDF5 files
2. All HDF5 features can be used.
3. All Pydantic features can be used.

Questions?

Thank Yous

- Thank you to the scientists that have test driven the library:
  Dr Lester Barnsley
- Thank you to all the members of the Scientific Computing team and other Scientists for feedback on the library and this presentation