NeXpy: A Python Toolbox for Interactive Data Analysis

http://www.nexusformat.org/NeXpy

Ray Osborn, Stephan Rosenkranz, John-Paul Castellan
Materials Science Division, Argonne National Laboratory, Argonne, IL

Boyana Norris, Jason Sarich, Dmitry Karpeev
Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL

with special thanks to
Paul Kienzle
NIST Center for Neutron Research, NIST, Gaithersburg, MD

NeXpy - a Python-based approach

- What NeXpy is not:
  - a comprehensive solution to all the issues so far raised
  - particularly sophisticated or novel
- What NeXpy is:
  - a toolbox for manipulating and visualizing arbitrary NeXus data
  - a possible scripting engine for GUI applications
  - a demonstration of the value of combining:
    - a flexible data model
    - a powerful scripting language

NeXus + NumPy

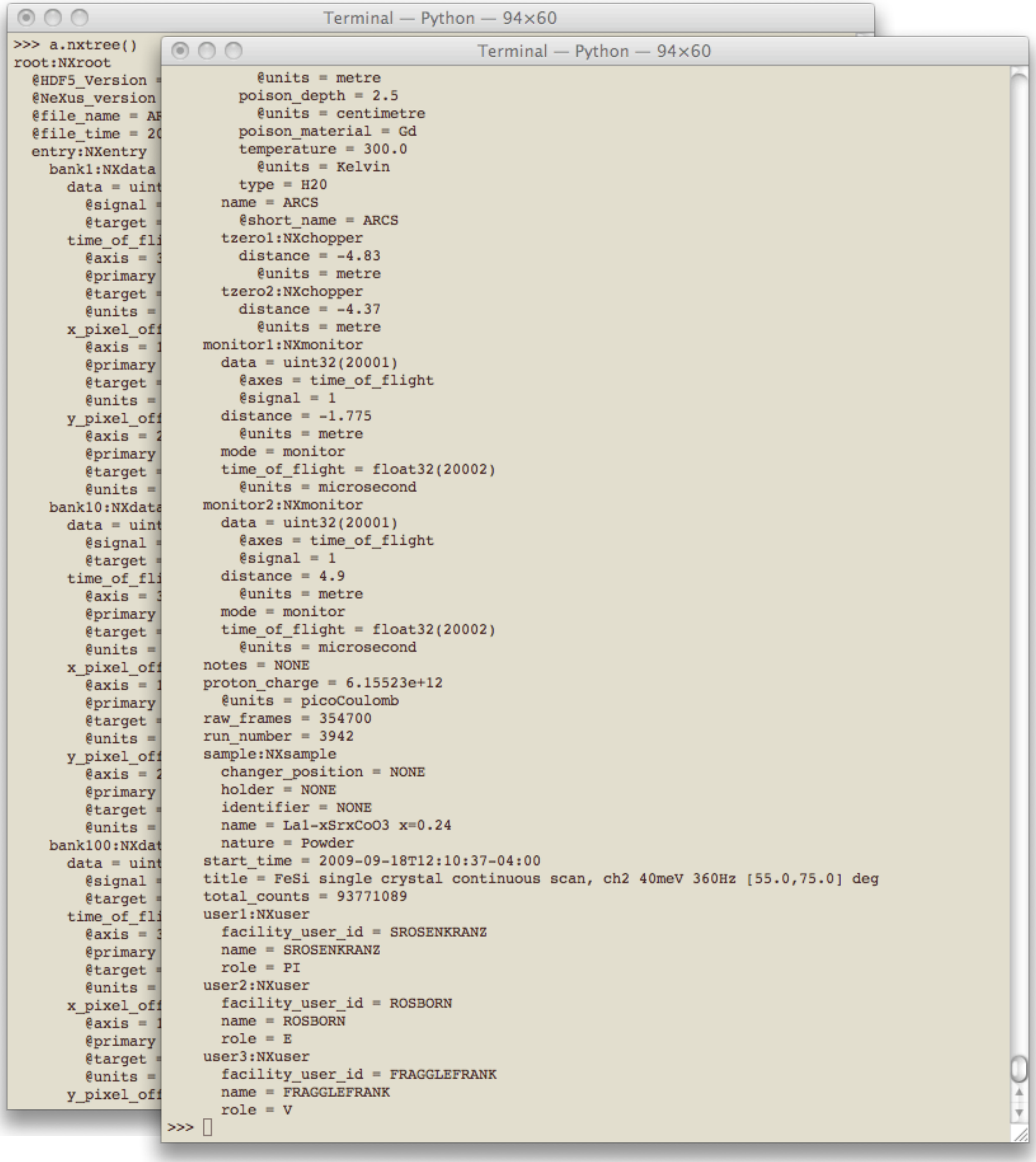
Python API

- There are two levels to the Python API
  - A one-to-one mapping of the C-API returning Numpy arrays (napi.py)
  - A one-to-one mapping of the NeXus data objects into Python classes (tree.py)

```
>>> f=nexus.open("data/chopper.nxs")
>>> f.opengroup("entry","NXentry")
>>> f.opendata("title")
>>> f.getdata()
'MgB2 PDOS 43.37g 8K 120meV E
>>> f.closedata()
>>> f.closegroup()
>>> f.close()
```

```
>>> a=nexus.load("data/chopper.nxs")
>>> a.entry.data.nxtree()
data:NXdata
  data = int32(148x750)
    @axes = polar_angle:time_of_flight
    @long_name = Neutron Counts
    @signal = 1
    @units = counts
  polar_angle = float32(148)
    @long_name = Polar Angle [degrees]
    @units = degrees
  time_of_flight = [ 1900.  1902.  1904. ...,  3396.  3398.  3400.]
    @long_name = Time-of-Flight [microseconds]
    @units = microseconds
  title = MgB2 PDOS 43.37g 8K 120meV E0@240Hz T0@120Hz
>>> print a.entry.data.title
MgB2 PDOS 43.37g 8K 120meV E0@240Hz T0@120Hz
```

ARCS/SNS Data



```
>>> a.nxtree()
root:NXroot
 @HDF5_Version =
 @NeXus_version
 @file_name = AR
 @file_time = 20
 entry:NXentry
  bank1:NXdata
   data = uint
    @signal =
    @target =
   time_of_fli
    @axis = 3
    @primary
    @target =
    @units =
   x_pixel_off
    @axis = 1
    @primary
    @target =
    @units =
   y_pixel_off
    @axis = 2
    @primary
    @target =
    @units =
  bank10:NXdata
   data = uint
    @signal =
    @target =
   time_of_fli
    @axis = 3
    @primary
    @target =
    @units =
   x_pixel_off
    @axis = 1
    @primary
    @target =
    @units =
   y_pixel_off
    @axis = 2
    @primary
    @target =
    @units =
  bank100:NXdat
   data = uint
    @signal =
    @target =
   time_of_fli
    @axis = 3
    @primary
    @target =
    @units =
   x_pixel_off
    @axis = 1
    @primary
    @target =
    @units =
   y_pixel_off
```

```
     @units = metre
    poison_depth = 2.5
     @units = centimetre
    poison_material = Gd
    temperature = 300.0
     @units = Kelvin
    type = H2O
   name = ARCS
    @short_name = ARCS
   tzero1:NXchopper
    distance = -4.83
     @units = metre
   tzero2:NXchopper
    distance = -4.37
     @units = metre
  monitor1:NXmonitor
   data = uint32(20001)
    @axes = time_of_flight
    @signal = 1
   distance = -1.775
    @units = metre
   mode = monitor
   time_of_flight = float32(20002)
    @units = microsecond
  monitor2:NXmonitor
   data = uint32(20001)
    @axes = time_of_flight
    @signal = 1
   distance = 4.9
    @units = metre
   mode = monitor
   time_of_flight = float32(20002)
    @units = microsecond
  notes = NONE
  proton_charge = 6.15523e+12
   @units = picoCoulomb
  raw_frames = 354700
  run_number = 3942
  sample:NXsample
   changer_position = NONE
   holder = NONE
   identifier = NONE
   name = La1-xSrxCoO3 x=0.24
   nature = Powder
  start_time = 2009-09-18T12:10:37-04:00
  title = FeSi single crystal continuous scan, ch2 40meV 360Hz [55.0,75.0] deg
  total_counts = 93771089
  user1:NXuser
   facility_user_id = SROSENKRANZ
   name = SROSENKRANZ
   role = PI
  user2:NXuser
   facility_user_id = ROSBORN
   name = ROSBORN
   role = E
  user3:NXuser
   facility_user_id = FRAGGLEFRANK
   name = FRAGGLEFRANK
   role = V
>>> 
```

Features of NeXus tree interface

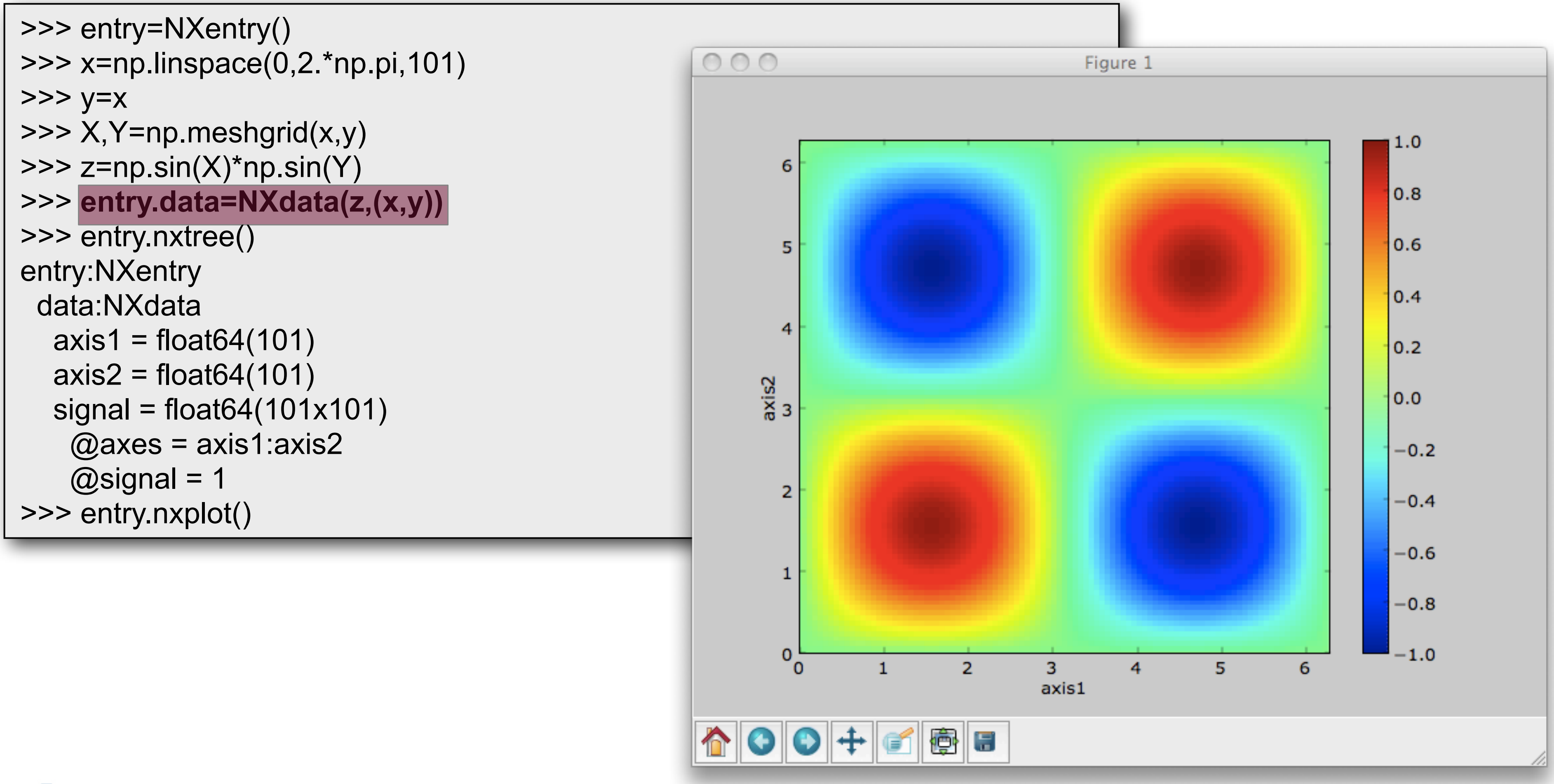
- The entire tree structure of a NeXus file can be loaded with a single command
  - The data values are not read until directly referenced
- NeXus objects can be created by simple assignments

```
>>> sample = NXsample()
>>> sample.temperature=40.0
>>> sample.temperature.units='K'
>>> entry = NXentry(sample)
>>> entry.nxtree()
entry:NXentry
  sample:NXsample
    temperature = 40.0
      @units = K
```

- Note: Command-line assignments automatically convert Numpy data into NeXus objects
  - *e.g.*, typing "entry.sample.temperature=40.0" converts the temperature into valid NeXus data
- All objects of the same class can easily be listed
  - *e.g.*, entry.NXdata[0], entry.NXdata[1], ...

Interactive manipulation of NeXus data

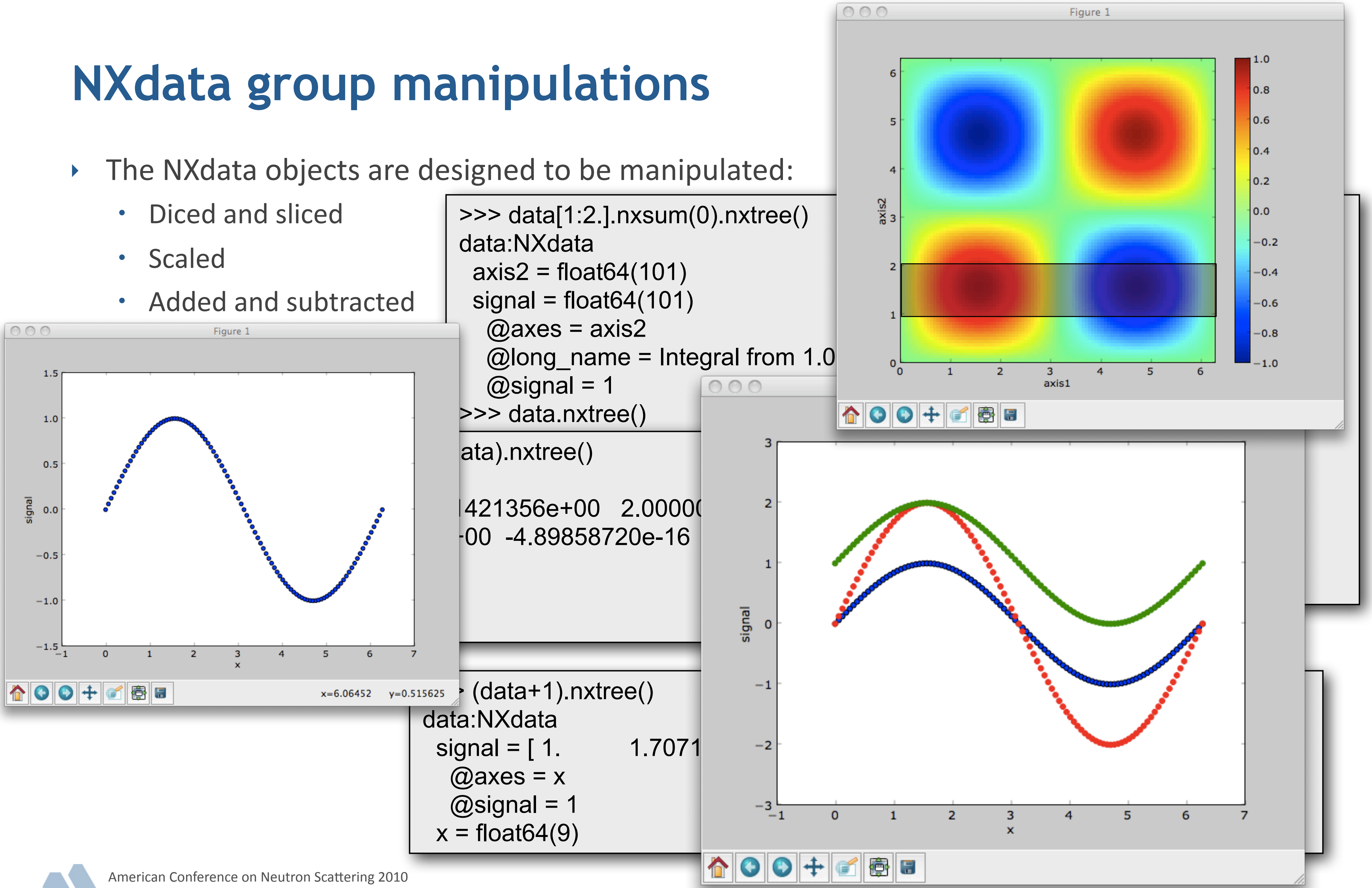
- The syntax makes the creation of standard-conforming NeXus data structures simple



```
>>> entry=NXentry()
>>> x=np.linspace(0,2.*np.pi,101)
>>> y=x
>>> X,Y=np.meshgrid(x,y)
>>> z=np.sin(X)*np.sin(Y)
>>> entry.data=NXdata(z,(x,y))
>>> entry.nxtree()
entry:NXentry
  data:NXdata
    axis1 = float64(101)
    axis2 = float64(101)
    signal = float64(101x101)
      @axes = axis1:axis2
      @signal = 1
>>> entry.nxplot()
```

NXdata group manipulations

- The NXdata objects are designed to be manipulated:
  - Diced and sliced
  - Scaled
  - Added and subtracted



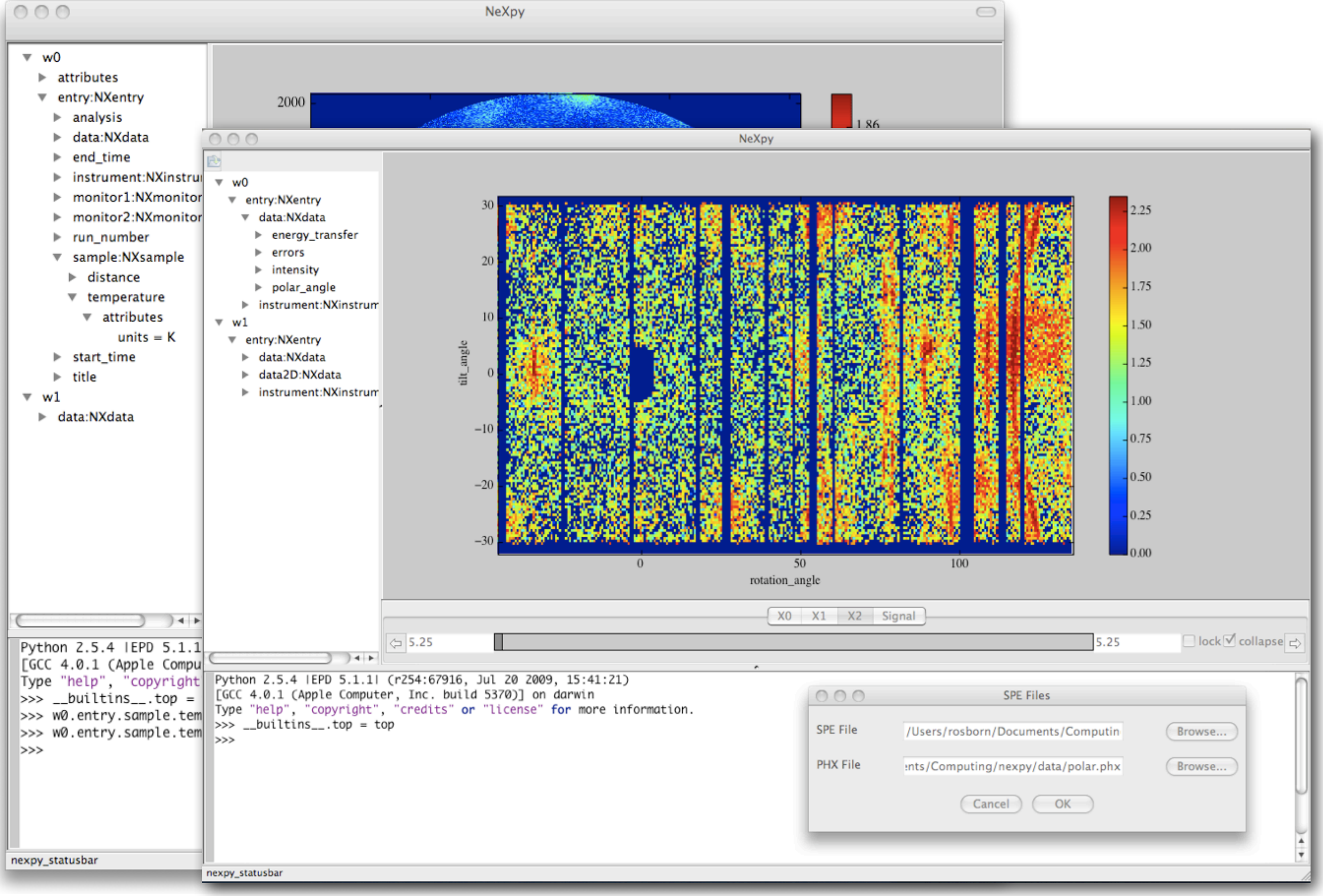
```
>>> data[1:2.].nxsum(0).nxtree()
data:NXdata
  axis2 = float64(101)
  signal = float64(101)
    @axes = axis2
    @long_name = Integral from 1.0
    @signal = 1
>>> data.nxtree()
```

```
ata).nxtree()

1421356e+00   2.0000(
-00  -4.89858720e-16
```

```
> (data+1).nxtree()
data:NXdata
  signal = [ 1.          1.7071
    @axes = x
    @signal = 1
  x = float64(9)
```

NeXpy GUI



```
Python 2.5.4 |EPD 5.1.1
[GCC 4.0.1 (Apple Compu
Type "help", "copyright
>>> __builtins__.top =
>>> w0.entry.sample.tem
>>> w0.entry.sample.tem
>>>
```

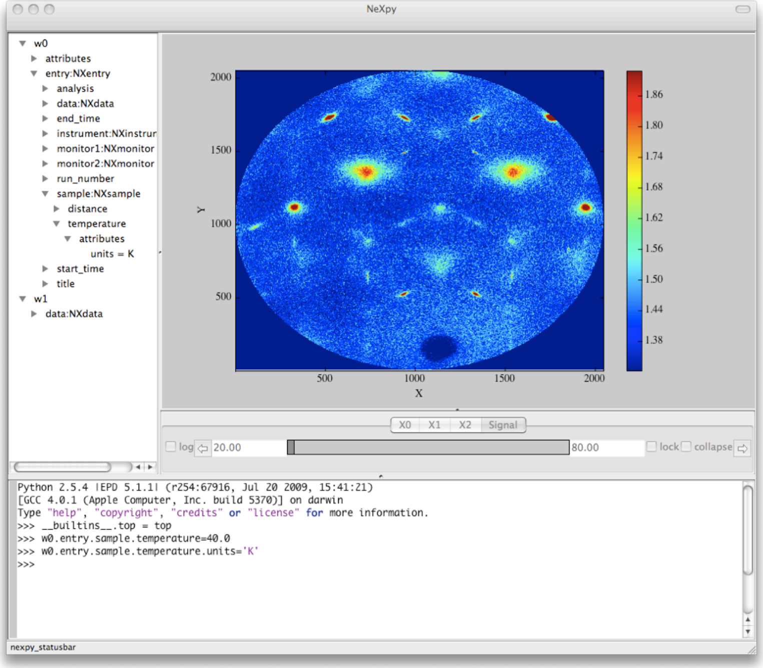
```
Python 2.5.4 |EPD 5.1.1| (r254:67916, Jul 20 2009, 15:41:21)
[GCC 4.0.1 (Apple Computer, Inc. build 5370)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> __builtins__.top = top
>>>
```

SPE File /Users/rosborn/Documents/Computin

PHX File nts/Computing/nexpy/data/polar.phx

Features of NeXpy GUI

- Persistent data
- Comprehensive access to metadata
- File-based memory management
  - Each data object maps directly to a valid NeXus file
- Non-proprietary language (*i.e.*, Python)
- Flexibility to do whatever you want to the data
  - No well-defined algorithms



```
Python 2.5.4 |EPD 5.1.1| (r254:67916, Jul 20 2009, 15:41:21)
[GCC 4.0.1 (Apple Computer, Inc. build 5370)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> __builtins__.top = top
>>> w0.entry.sample.temperature=40.0
>>> w0.entry.sample.temperature.units='K'
>>>
```

On the To-Do List (near-term)

- Adding a data editor
- Incorporating generalized coordinate transformations
- Incorporating Open GL modules to improve current Matplotlib speeds
- Expanded use of parallelization, including GPUs
- Adding full 3D plotting
  - Using Mayavi
- Adding a fitting pane
  - Using the DANSE *MYSTIC/PARK* framework
- Installing NeXpy as a standard part of the NeXus distribution
- Create a version for the iPad

http://www.nexusformat.org/NeXpy