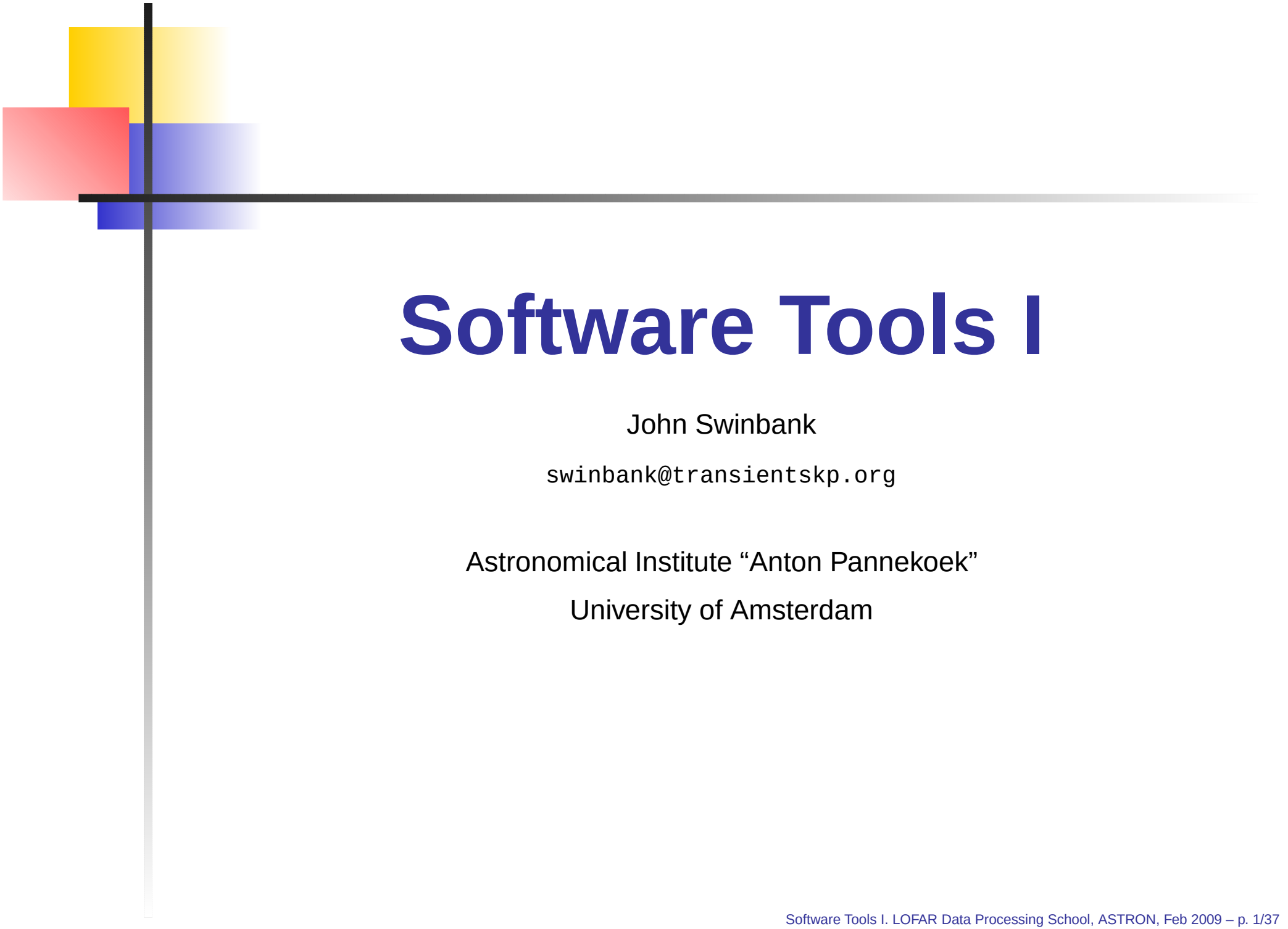


Software Tools I

John Swinbank

swinbank@transientskp.org

Astronomical Institute “Anton Pannekoek”

University of Amsterdam

Overview: tools for hands-on sessions

- Python
- Scientific & numerical libraries
- Data access
- Data inspection
- Plotting

Python

- (Becoming) the standard “glue” in astronomical applications
  - PyRAF, CASA, stsci_python...
  - & now LOFAR
- Quick scripts to full-blown applications
  - LOFAR transients pipeline is (almost) pure Python
- Various implementations(.net, Java, ...)
  - Safe option: CPython 2.4/2.5/2.6

Scripting Python

```
$ cat > hello.py
#!/usr/bin/python
print "Hello world"
$ chmod u+x hello.py
$ ./hello.py
Hello world
$
```

- No need to compile
- Note .pyc bytecode file

Interactive Python

```
$ python
Python 2.4.2 (#1, Jan 10.17008, 17:45:02)
[GCC 4.1.2 20070115 (prerelease) (SUSE Linux)] on linux2
Type "help", "copyright", "credits" or "license" for more...
>>> 1 + 1
2
>>> print "hello world"
hello world
>>> type(1 + 1)
<type 'int'>
```

IPython: `http://ipython.scipy.org/`

Types

```
>>> 'a string ' + 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
>>> 'a string ' + str(1)
'a string 1'
```

Types

```
>>> 'a string ' + 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
>>> 'a string ' + str(1)
'a string 1'

>>> i = 1
>>> type(i)
<type 'int'>
>>> i = "string"
>>> type(i)
<type 'str'>
```

Objects

```
>>> i = -1
>>> type(i)
<type 'int'>
>>> type(int)
<type 'type'>
>>> i.<TAB> # In IPython
i.__abs__           i.__doc__           i.__int__
...
>>> i.__abs__()
1
```

Whitespace

```
>>> i = 1
>>>     j = 2
  File "<stdin>", line 1
    j = 2
    ^
IndentationError: unexpected indent
>>> def function():
... print "hello"
  File "<stdin>", line 2
    print "hello"
        ^
IndentationError: expected an indented block
```

Lists and tuples

```
>>> my_list = [1, 2.0, "3"]
>>> type(my_list)
<type 'list'>
>>> my_list.append(17)
>>> my_list[0] = 6
>>> print my_list
[6, 2.0, '3', 17]
```

Lists and tuples

```
>>> my_list = [1, 2.0, "3"]
>>> type(my_list)
<type 'list'>
>>> my_list.append(17)
>>> my_list[0] = 6
>>> print my_list
[6, 2.0, '3', 17]


>>> my_tuple = 1, 2.0, "3"
>>> type(my_tuple)
<type 'tuple'>
>>> my_tuple.append(17)
AttributeError: 'tuple' object has no attribute 'append'
>>> my_tuple[0] = 6
TypeError: 'tuple' object does not support item assignment
```

Containers and iteration

```
>>> my_list = [1, 2, 3]
>>> for x in my_list: print x,
...
1 2 3
```

Containers and iteration

```
>>> my_list = [1, 2, 3]
>>> for x in my_list: print x,
...
1 2 3

>>> new_list = [2 * x for x in my_list]
>>> print new_list
[2, 4, 6]
```

Containers and iteration

```
>>> my_list = [1, 2, 3]
>>> for x in my_list: print x,
...
1 2 3

>>> new_list = [2 * x for x in my_list]
>>> print new_list
[2, 4, 6]

>>> my_gen = (2 * x for x in my_list)
>>> print my_gen
<generator object at 0xb2f878>
>>> for x in my_gen: print x,
...
2 4 6
```

Libraries and modules

```
>>> import random
>>> random.randint(1, 10)
8
>>> from random import randint
>>> randint(1, 10)
1
>>> from sys import platform
>>> platform
'linux2'
```

Default location + $PYTHONPATH

Numerical tools

```
>>> import numpy
>>> ar = numpy.array(((1,2,3), (4,5,6)))
>>> ar
array([[1, 2, 3],
       [4, 5, 6]])
>>> numpy.vdot(ar[0], ar[1])
32
```

Numerical tools

```
>>> import numpy
>>> ar = numpy.array(((1,2,3), (4,5,6)))
>>> ar
array([[1, 2, 3],
       [4, 5, 6]])
>>> numpy.vdot(ar[0], ar[1])
32

>>> ar * 2
array([[ 2, 4, 6],
       [ 8, 10, 12]])
>>> from math import sqrt
>>> vector_sqrt = numpy.vectorize(sqrt)
>>> vector_sqrt(ar)
array([[ 1.        , 1.41421356, 1.73205081],
       [ 2.        , 2.23606798, 2.44948974]])
```

Scientific tools

```
>>> import scipy.xxx
```

ndimage	n-dimensional image package
stats	Statistical Functions
signal	Signal Processing Tools
lib	Python wrappers to external libraries
linalg	Linear algebra routines
linsolve.umfpack	Interface to the UMFPACK library
odr	Orthogonal Distance Regression
misc	Various utilities that don't have another home
sparse	Sparse matrix
interpolate	Interpolation Tools
optimize	Optimization Tools
cluster	Vector Quantization / Kmeans
linsolve	Linear Solvers
fftpack	Discrete Fourier Transform algorithms
io	Data input and output
maxentropy	Routines for fitting maximum entropy models
integrate	Integration routines
lib.lapack	Wrappers to LAPACK library
special	Airy Functions
lib.blas	Wrappers to BLAS library

Data access

- PyFITS
  - `http://www.stsci.edu/resources/software_hardware/pyfits`
- LOFAR Data Access Library (DAL)
  - `http://usg.lofar.org/`
- pyrap
  - AIPS++ tables
  - Talk by van Diepen
  - `http://code.google.com/p/pyrap/`

PyFITS: headers

```
>>> import pyfits
>>> hdulist = pyfits.open('filename.fits') # list of HDUs
>>> hdulist.info()
Filename: /home/jds/test-data/fits/1986-01-15.FITS
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU     241  (1024, 1024, 1, 1)  float32
1    AIPS CC     BinTableHDU     19  200R x 3C     [1E, 1E, 1E]
```

PyFITS: headers

```
>>> import pyfits
>>> hdulist = pyfits.open('filename.fits') # list of HDUs
>>> hdulist.info()
Filename: /home/jds/test-data/fits/1986-01-15.FITS
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU     241  (1024, 1024, 1, 1)  float32
1    AIPS CC     BinTableHDU     19  200R x 3C     [1E, 1E, 1E]
>>> header = hdulist[0].header
>>> header.ascardlist().keys()[:3]
['SIMPLE', 'BITPIX', 'NAXIS']
>>> header.get('NAXIS')
4
```

PyFITS: headers

```
>>> import pyfits
>>> hdulist = pyfits.open('filename.fits') # list of HDUs
>>> hdulist.info()
Filename: /home/jds/test-data/fits/1986-01-15.FITS
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU     241  (1024, 1024, 1, 1)  float32
1    AIPS CC     BinTableHDU     19  200R x 3C     [1E, 1E, 1E]
>>> header = hdulist[0].header
>>> header.ascardlist().keys()[:3]
['SIMPLE', 'BITPIX', 'NAXIS']
>>> header.get('NAXIS')
4
>>> header.update('example', 'header')
>>> header.get('example')
'header'
>>> hdulist.close()
```

PyFITS: data

```
>>> import pyfits
>>> hdulist = pyfits.open('filename.fits')
>>> image_data = hdulist[0].data
```

PyFITS: data

```
>>> import pyfits
>>> hdulist = pyfits.open('filename.fits')
>>> image_data = hdulist[0].data
>>> type(image_data)
<type 'numpy.ndarray'>
>>> image_data.shape
(1, 1, 1024, 1024)
>>> image_data = image_data.squeeze().transpose()
>>> image_data.shape
(1024, 1024)
>>> image_data[0, 0:3]
[ -7.14534908e-05  -2.09340811e-04  -2.94430123e-04]
```

PyFITS: saving files

```
>>> import pyfits
>>> hdulist = pyfits.open('filename.fits')
>>> hdulist[0].header.update('example', 'header')
>>> hdulist.writeto('newfile.fits') # Write new file
>>> hdlist.close()
```

PyFITS: saving files

```
>>> import pyfits
>>> hdulist = pyfits.open('filename.fits')
>>> hdulist[0].header.update('example', 'header')
>>> hdulist.writeto('newfile.fits') # Write new file
>>> hdlist.close()

>>> hdulist = pyfits.open('filename.fits', mode='update')
>>> hdulist[0].header.update('example', 'header')
>>> hdulist.flush()                  # Update original file
>>> hdulist.close()
```

Data Access Library

- ”Abstract data product implementation details from the user”.
- Uniform access system for
  - General: FITS, AIPS++ Tables, HDF5
  - LOFAR: beam-formed, TBB, ...
- C++, **Python** support
- Still a work-in-progress
  - Note: currently lacking FITS support.

DAL (2)

```
$ export PYTHONPATH=/app/usg/release/lib/python
$ ipython
[...]
>>> import pydal
>>> dataset = pydal.dalDataset()
>>> dataset.open('filename.MS')
True
>>> dataset.listTables()
['OBSERVATION', 'DATA_DESCRIPTION', 'SOURCE'...]
>>> obs_table = dataset.openTable('OBSERVATION')
>>> obs_table.listColumns()
['TIME_RANGE', 'TELESCOPE_NAME', 'OBSERVER'...]
>>> print obs_table.getColumn('TELESCOPE_NAME').data()
['LOFAR']
```

DAL (3)

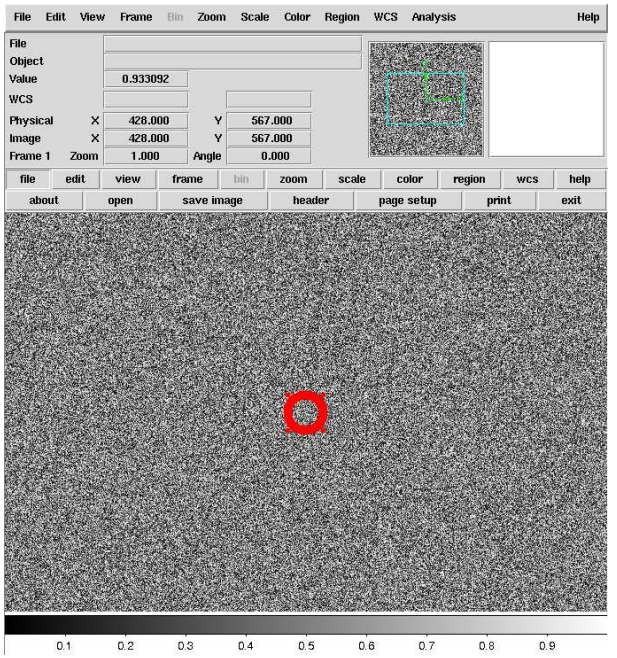
```
>>> import pydal
>>> dataset = pydal.dalDataset()
>>> dataset.open('filename.MS')
True
>>> data = dataset.openTable('MAIN').getColumn('UVW').data()
```

DAL (3)

```
>>> import pydal
>>> dataset = pydal.dalDataset()
>>> dataset.open('filename.MS')
True
>>> data = dataset.openTable('MAIN').getColumn('UVW').data()
>>> print data
[[  0.           0.           0.        ]
 [ 24.84548655  21.44633043  11.60285309]
 ...
>>> print type(data)
<type 'numpy.ndarray'
>>> print data.shape
(389096, 3)
>>> print data.dtype
float64
```

Examine your data

```
$ export PYTHONPATH=/app/RO-2.2.10/python
>>> import RO.DS9, numpy
>>> data = numpy.random.random((1000,1000))
>>> ds9win = RO.DS9.DS9Win()
>>> ds9win.showArray(data)
>>> ds9win.xpaset('regions', 'circle 500 500 20')
>>> ds9win.xpaset('regions select all')
>>> ds9win.xpaset('regions color red')
>>> ds9win.xpaset('regions width 10')
```



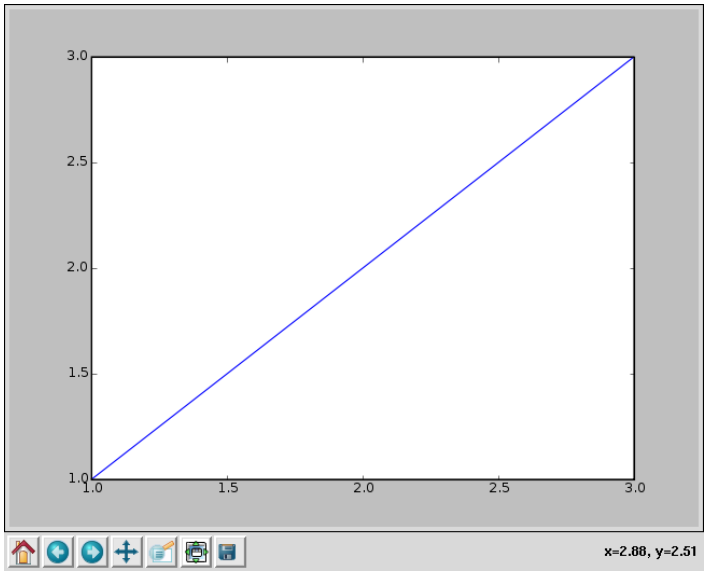
Handling coordinates

```
$ export PYTHONPATH=/app/usg/release/lib/python
>>> from wcslib import wcs
>>> my_wcs = wcs()
>>> my_wcs.crval = ( 3.508500000000E+02, 5.881500000000E+01)
>>> my_wcs.cdelt = (-3.333333333333E-02, 3.333333333333E-02)
>>> my_wcs.crota = ( 0.000000000000E+00, 0.000000000000E+00)
>>> my_wcs.crpix = ( 1.441000000000E+03, 1.441000000000E+03)
>>> my_wcs.ctype = ('RA---SIN', 'DEC--SIN')
>>> my_wcs.cunit = ('deg', 'deg')
>>> my_wcs.p2s((1000,1000))
[10.892749099930882, 41.676643862188534]
>>> my_wcs.s2p((289.88484466996505, 68.122432582567072))
[1858.629640823601, 2852.9223838113721]
```

Calabretta & Greisen, A&A, 295, 1077-1122, 2002.

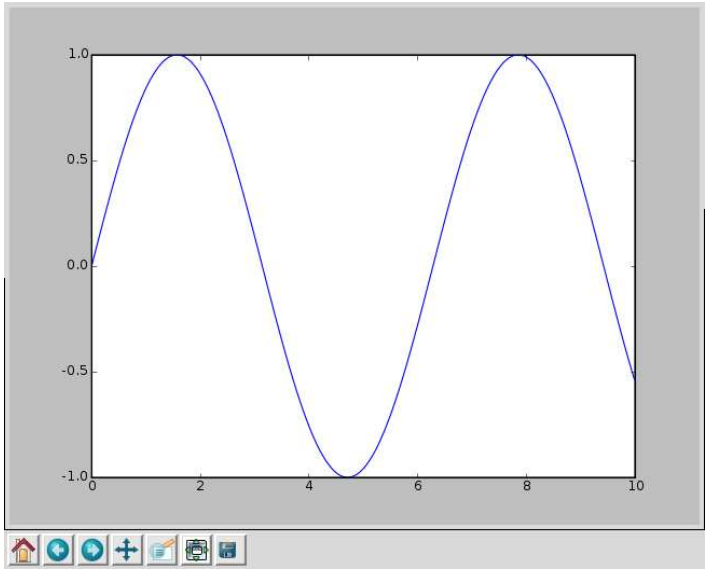
Plotting: matplotlib

```
$ ipython -pylab
[...]
  Welcome to pylab, a matplotlib-based Python environment.
  For more information, type 'help(pylab)'.
>>> plot([1,2,3], [1,2,3])
[<matplotlib.lines.Line2D instance at 0x1dcfdd0>]
```



matplotlib (2)

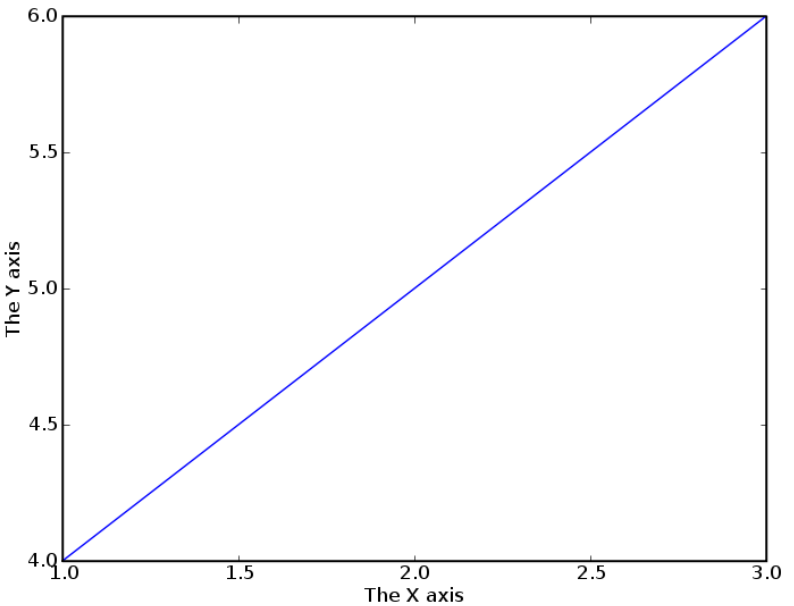
```
$ ipython -pylab
[...]
>>> import numpy
>>> x_data = numpy.arange(0, 10, 0.01)
>>> print x_data[:4]
[ 0.    0.01  0.02  0.03]
>>> print numpy.sin(x_data)[:3]
[ 0.    0.00999983  0.01999867 ...]
>>> plot(x_data, numpy.sin(x_data))
```



matplotlib (3)

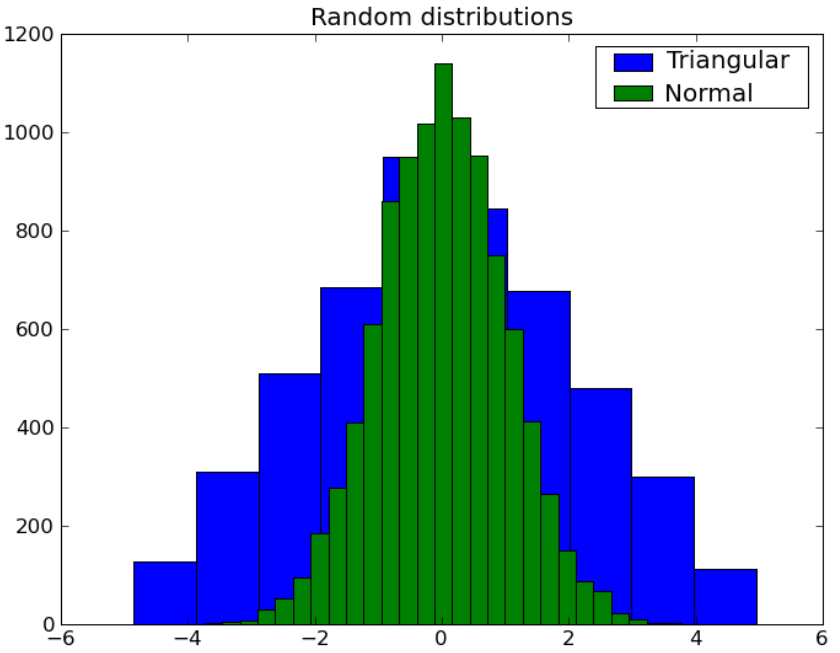
From a script:

```
from pylab import *

data = [1, 2, 3]
result = [4, 5, 6]

xlabel("The X axis")
ylabel("The Y axis")
plot(data, result)
savefig('figure.png')
```



matplotlib (4)

```
from pylab import *
from numpy import random
title("Random distributions")
hist(random.triangular(-5, 0, 5, size=5000), label="Triangular")
hist(random.normal(size=10000), bins=30, label="Normal")
legend()
savefig('figure.png')
```



The End

Now go and play with some data!
...after Ger's talk.

- `http://docs.python.org/`
- `http://www.scipy.org/`
- `http://www.stsci.edu/resources/software_hardware/pyfits`
- `http://matplotlib.sourceforge.net/`