# ObsPy: A powerful instrument for seismological data software development

## 38th Workshop of the International School of Geophysics
## 1st EPOS-ORFEUS Coordination Meeting
## Global challenges for seismological data analysis

Lion Krischer

Ludwig-Maximilians-University in Munich
Department of Earth and Environmental Sciences
Geophysics

May 27, 2012

# Python and ObsPy

**Goals**

- Showcase the simplicity and abilities of Python.
- Demonstrate why it, together with ObsPy, forms a solid foundation for seismological data software development.

A glimpse at Python

Introducing ObsPy

# Why use Python?

- Open Source
- Cross-platform
- General purpose language
- Dynamically typed
- Interactive shell
- Readability
- Speed
- "Batteries included"
- Mature third party libraries
- Language Interoperability

# Readability

- **Indentation**
  - Code blocks are defined by their indentation.
  - No explicit begin or end, and no curly braces to mark where a block starts and stops. The only delimiter is a colon (:) and the indentation of the code itself.

```
for i in [1, 2, 3, 4, 5]:
    if i<3:
        print i,
    else:
        print i*2,
```

Output:

```
1 2 6 8 10
```

- **Very minimalistic clean syntax & semantics**
  - Short code = Less errors!
  - But also faster development, quicker understanding, faster typing, faster finding errors, easier to modify ...

# Readability

- **Readable syntax**
  1. Does an element exist in a list/dict:

```
>>> 3 in [1, 2, 3, 4, 5]
True
```

  2. Does a substring exist in a string:

```
>>> 'sub' in 'string'
False
```

  3. Readable boolean values and logical operators

```
>>> a = True
>>> not a
False
>>> 'sub' not in ['string', 'hello', 'world']
True
>>> a or True and not 1==2
True
```

# Speed

"Python is extremely slow and wastes memory!"

```
xvec = range(2000000)
yvec = range(2000000)
zvec = [0.5*(x+y) for x,y in zip(xvec,yvec)]
```

- 104 MB memory usage!
- Runs almost 8 seconds!

# Speed

## Comparison with C

```
#include <stdlib.h>

main () {
    int *avec, *bvec;
    float *cvec;
    int i, n = 2000000;
    avec = (int*)calloc( n, sizeof(int) );
    bvec = (int*)calloc( n, sizeof(int) );
    for (i=0;i<n;i++) {
        avec[i] = bvec[i] = i;
    }
    cvec = (float*)calloc( n, sizeof(float) );
    for (i=0;i<n;i++) {
        cvec[i] = (avec[i]+bvec[i])*0.5;
    }
}
```

- 25 MB memory usage
- Runs about 0.1 s (almost a hundred times faster!)

# Speed

NumPy

```
import numpy as np

xvec = np.arange(2000000)
yvec = np.arange(2000000)
zvec = (xvec+yvec)*0.5
```

- 37 MB memory usage
- Runs about 0.3 s

$\Rightarrow$ **Choose the right tool for a job.**

Using Numpy yields satisfactory performance in many cases.

# Speed

**Implementation time vs. execution time**

- Python is designed for productivity
- No (separate) compilation step
  - No compiler problems
  - No makefiles
  - No linker problems
  - Faster development cycles
- When execution speed matters:
  - Use specialized modules
  - Implement time critical parts in C/C++/Fortran/Cython
  - Prototyping & profiling
  - Use specialized JIT compiler

# "Batteries included"

- Extensive standard libraries:
  - **Powerful string handling**
  - **Filesystem manipulation utitilties**
  - Data Persistence
  - Data Compression and Archiving
  - Cryptographic Services
  - **Internet Protocols and Data Handling**
  - Structured Markup Processing Tools
  - Development Tools
  - Multithreading & Multiprocessing
  - **Regular expressions**
  - Graphical User Interfaces with Tk
  - ...

# ”Batteries included”

- Well-documented
- Platform independent API, but optimized for each platform
- One place to look first for a proven solution
- Reuse instead of reinvent

# Must have third party libraries/tools

- **NumPy/SciPy** - array and matrix structures, linear algebra routines, numerical optimization, random number generation, statistics routines, differential equation modeling, Fourier transforms and signal processing, image processing, sparse and masked arrays, spatial computation, and numerous other mathematical routines
- **matplotlib** - 2D (and limited 3D) plotting library which produces publication quality figures via a set of functions familiar to MATLAB users
- **SQLAlchemy** - SQL toolkit and object-relational mapper (ORM - maps classes to the database in multiple ways), supports various databases
- **IPython** - Enhanced interactive shell, also provides a HTML notebook
- **lxml**, **PyQt**, ...

# IPython HTML Notebook

Untitled4

http://127.0.0.1:8888/0fadbca9-66c7-4cd6-a943-5a984c840784

Google

IP[y]: Notebook

Example | Rename | QuickHelp

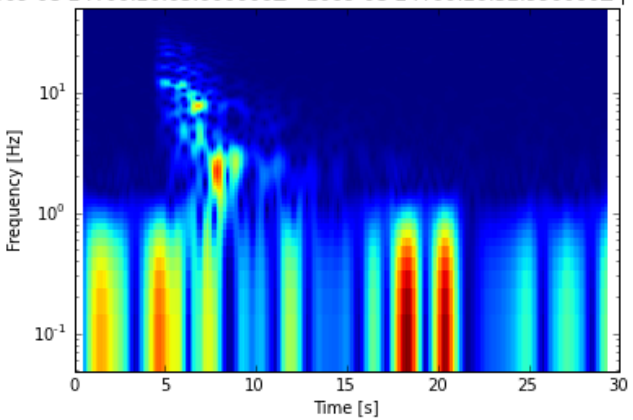
In [1]:
```
from obspy.core import read
st = read()[0]
print st
```

```
BW.RJOB..EHZ | 2009-08-24T00:20:03.000000Z - 2009-08-24T00:20:32.990000Z | 100.0 Hz, 3000 samples
```

In [2]:
```
st.plot()
```


In [3]:
```
st.spectrogram(log=True)
```

BW.RJOB..EHZ | 2009-08-24T00:20:03.000000Z - 2009-08-24T00:20:32.990000Z | 100.0 Hz, 3000 samples



# Language Interoperability

Python excels at gluing other languages together:

- **FORTRAN**: F2py - Fortran to Python interface generator (part of NumPy).
- General **C** or **C++** libraries: Ctypes, Cython, or SWIG are three ways to interface them.
- **R**: RPy - simple, robust Python interface to the R Programming Language. It can manage all kinds of R objects and can execute arbitrary R functions (including the graphic functions).

A glimpse at Python

Introducing ObsPy

# What is ObsPy and what can it do

A Python Toolbox for seismology/seismological observatories.

The goal of the ObsPy project is to **facilitate rapid application development for seismology**.

**http://www.obspy.org**

# What is ObsPy and what can it do

- In development since 2008
- 6 core developers
- Many more people contribute (http://docs.obspy.org/credits.html)
- Thoroughly unit tested
- Modularized architecture
- Written in Python (performance critical parts are written in C)
- Uses well established libraries (libmseed, GSE_UTI, . . . )
- Open source (LGPLv2/GPLv2) and cross platform

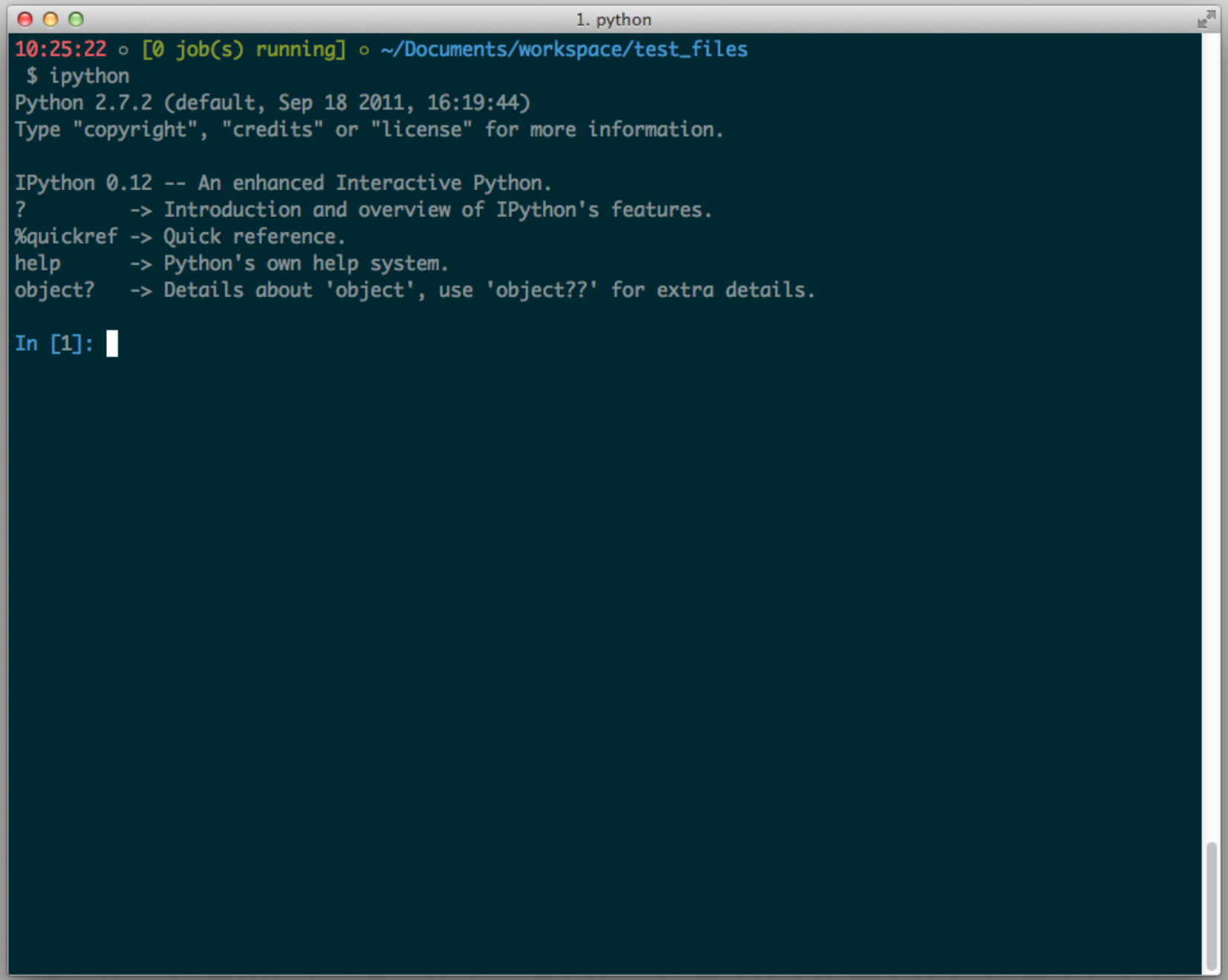
# What is ObsPy and what can it do

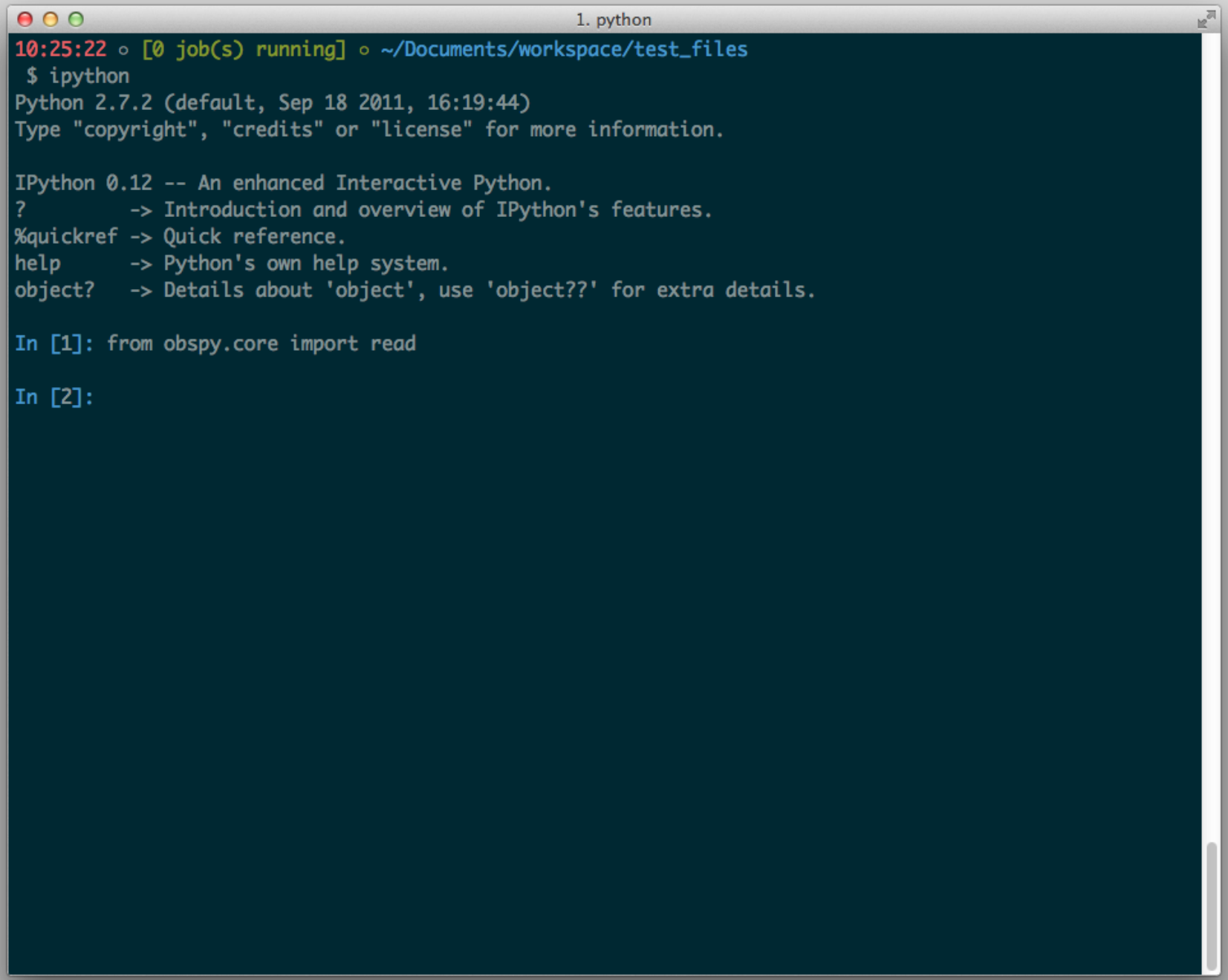
- **Read and write waveform data in various formats** (MiniSEED, SAC, GSE, SEG Y, ...) with a unified interface.
- **Database and webservice access clients** for NERIES, IRIS DMC, ArcLink, SeisHub and Earthworm.
- **Many seismological signal processing routines** like filters, trigger, instrument correction, array analysis, beamforming, ...
- **Support for inventory data** (SEED, XSEED, RESP and planned StationXML support)
- **Experimental event data handling** (currently only QuakeML)
- Waveform, spectrogram and beachball plotting, data availability (obspy-scan), ...

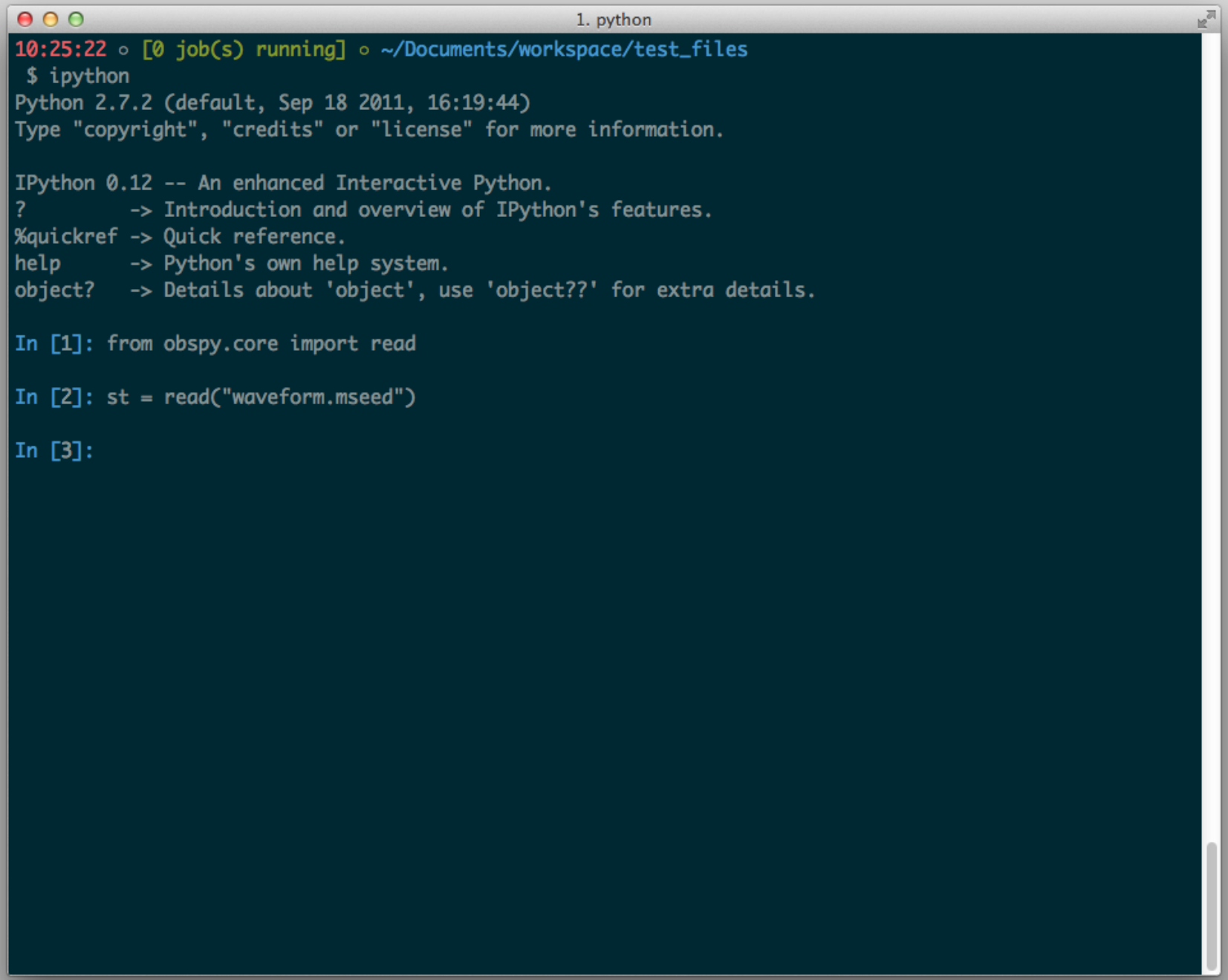
+

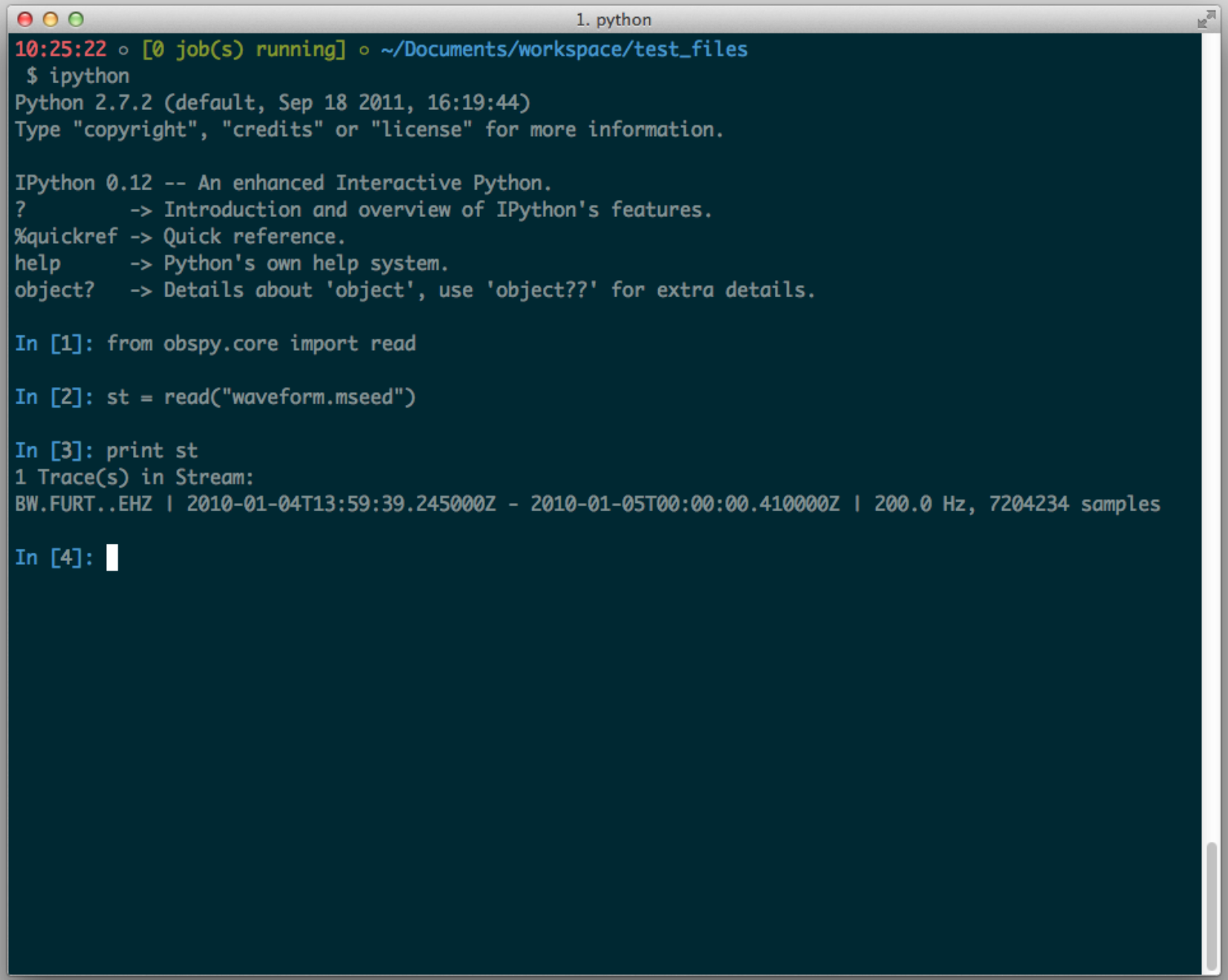
**The full power and flexibility of Python.**

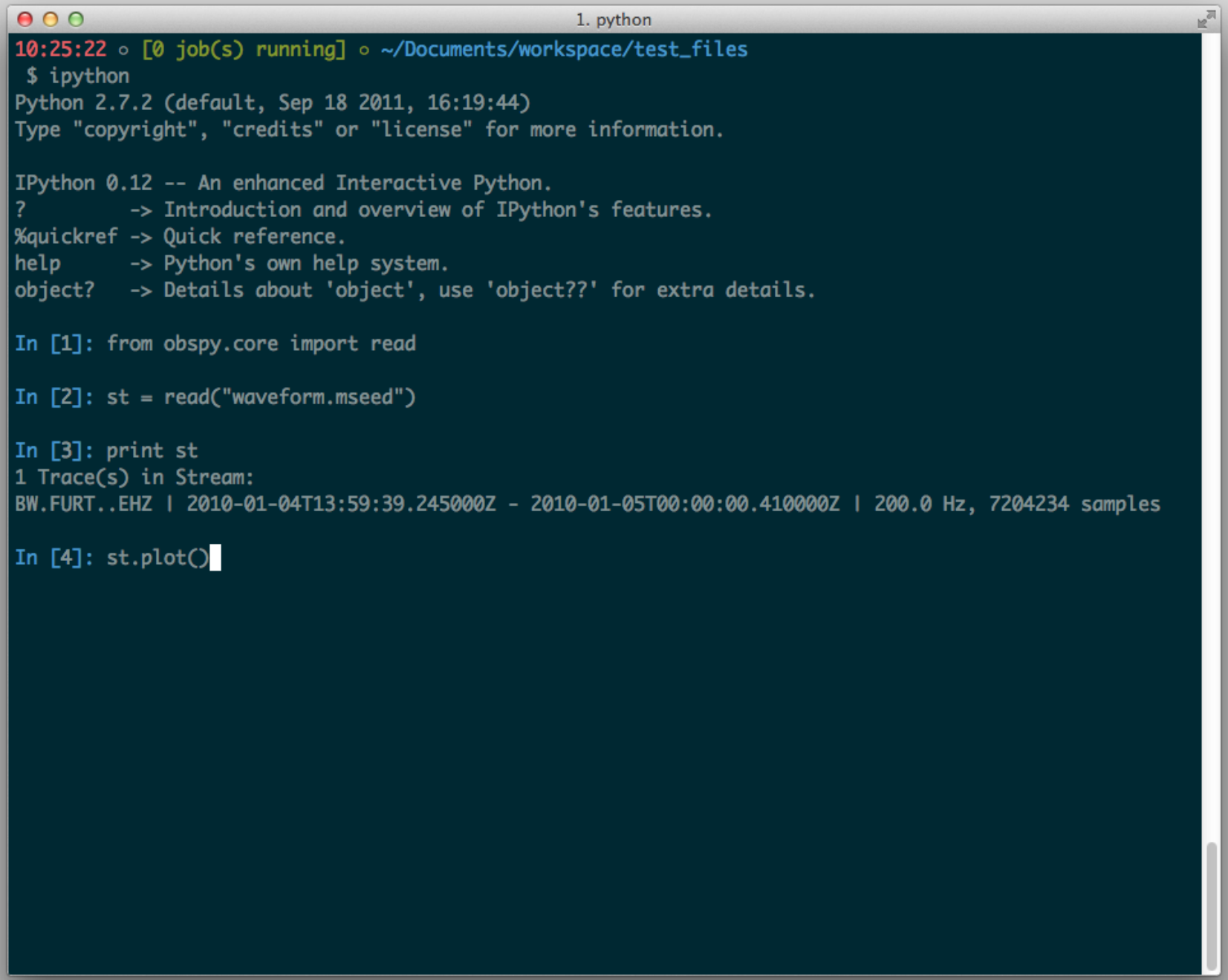
“Live” demonstration


1. python
10:25:22 ○ [0 job(s) running] ○ ~/Documents/workspace/test_files
$ ipython
Python 2.7.2 (default, Sep 18 2011, 16:19:44)
Type "copyright", "credits" or "license" for more information.
IPython 0.12 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
In [1]:


1. python
10:25:22 ○ [0 job(s) running] ○ ~/Documents/workspace/test_files
$ ipython
Python 2.7.2 (default, Sep 18 2011, 16:19:44)
Type "copyright", "credits" or "license" for more information.
IPython 0.12 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
In [1]: from obspy.core import read
In [2]:


1. python
10:25:22 ○ [0 job(s) running] ○ ~/Documents/workspace/test_files
$ ipython
Python 2.7.2 (default, Sep 18 2011, 16:19:44)
Type "copyright", "credits" or "license" for more information.
IPython 0.12 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
In [1]: from obspy.core import read
In [2]: st = read("waveform.mseed")
In [3]:


1. python
10:25:22 ○ [0 job(s) running] ○ ~/Documents/workspace/test_files
$ ipython
Python 2.7.2 (default, Sep 18 2011, 16:19:44)
Type "copyright", "credits" or "license" for more information.
IPython 0.12 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
In [1]: from obspy.core import read
In [2]: st = read("waveform.mseed")
In [3]: print st
1 Trace(s) in Stream:
BW.FURT..EHZ | 2010-01-04T13:59:39.245000Z - 2010-01-05T00:00:00.410000Z | 200.0 Hz, 7204234 samples
In [4]:


1. python
10:25:22 ○ [0 job(s) running] ○ ~/Documents/workspace/test_files
$ ipython
Python 2.7.2 (default, Sep 18 2011, 16:19:44)
Type "copyright", "credits" or "license" for more information.
IPython 0.12 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
In [1]: from obspy.core import read
In [2]: st = read("waveform.mseed")
In [3]: print st
1 Trace(s) in Stream:
BW.FURT..EHZ | 2010-01-04T13:59:39.245000Z - 2010-01-05T00:00:00.410000Z | 200.0 Hz, 7204234 samples
In [4]: st.plot()

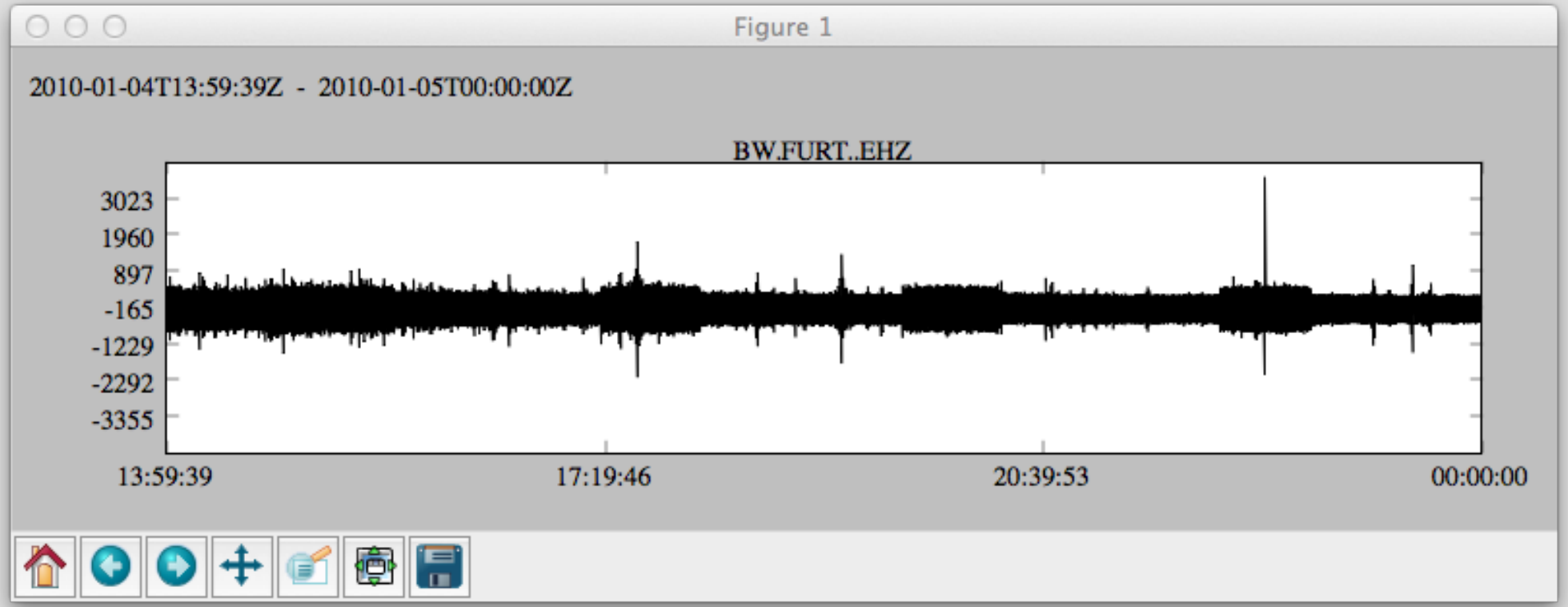
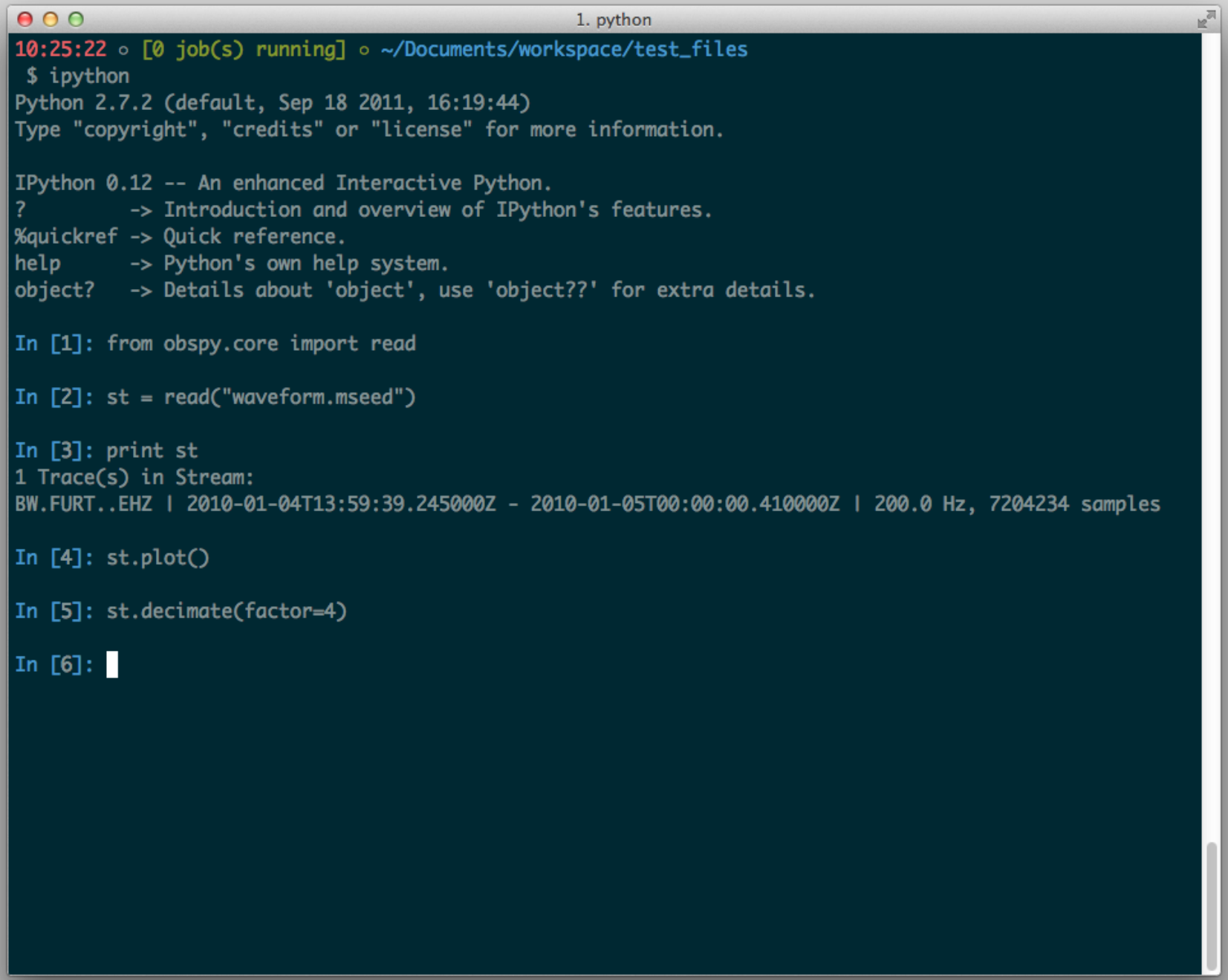
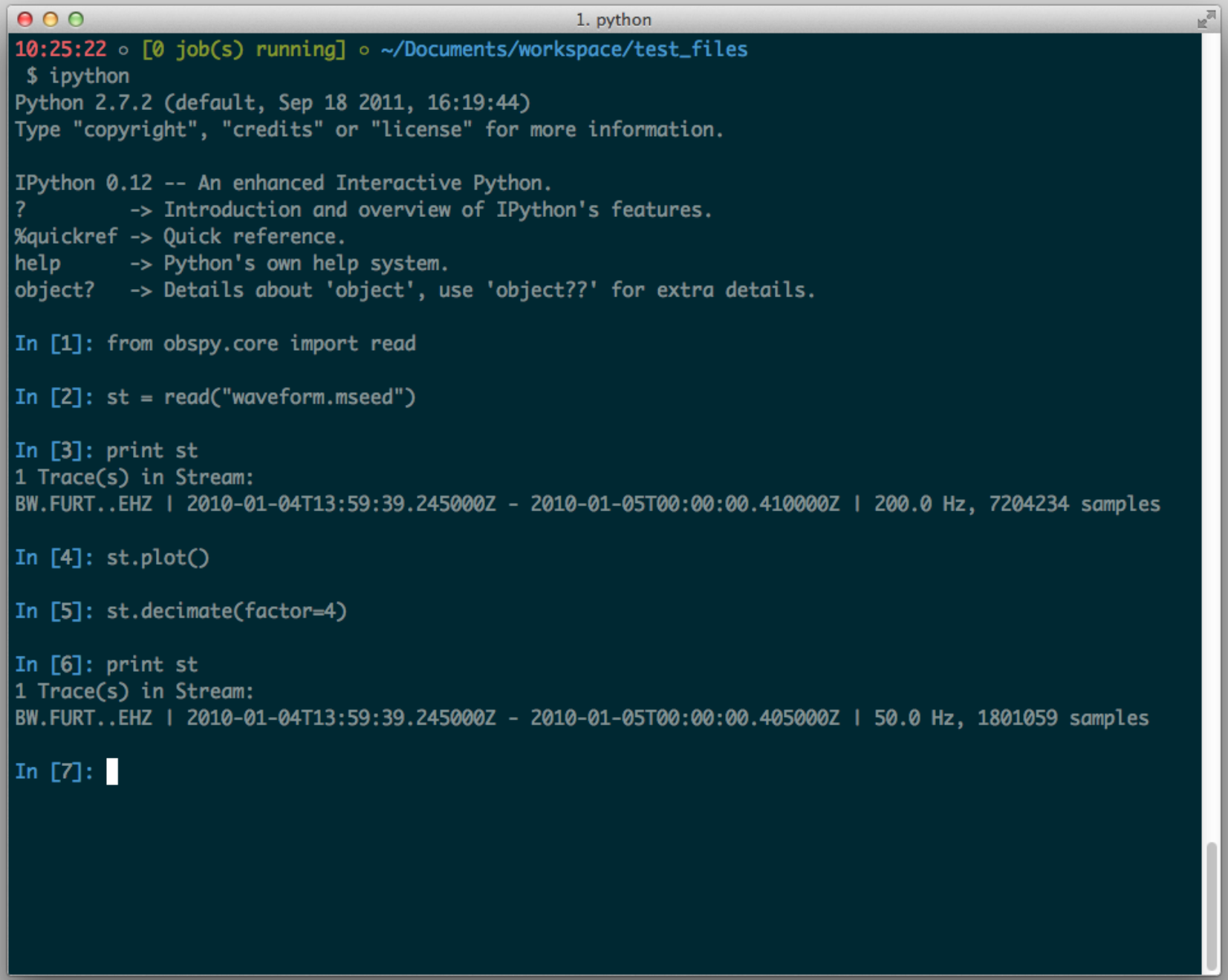
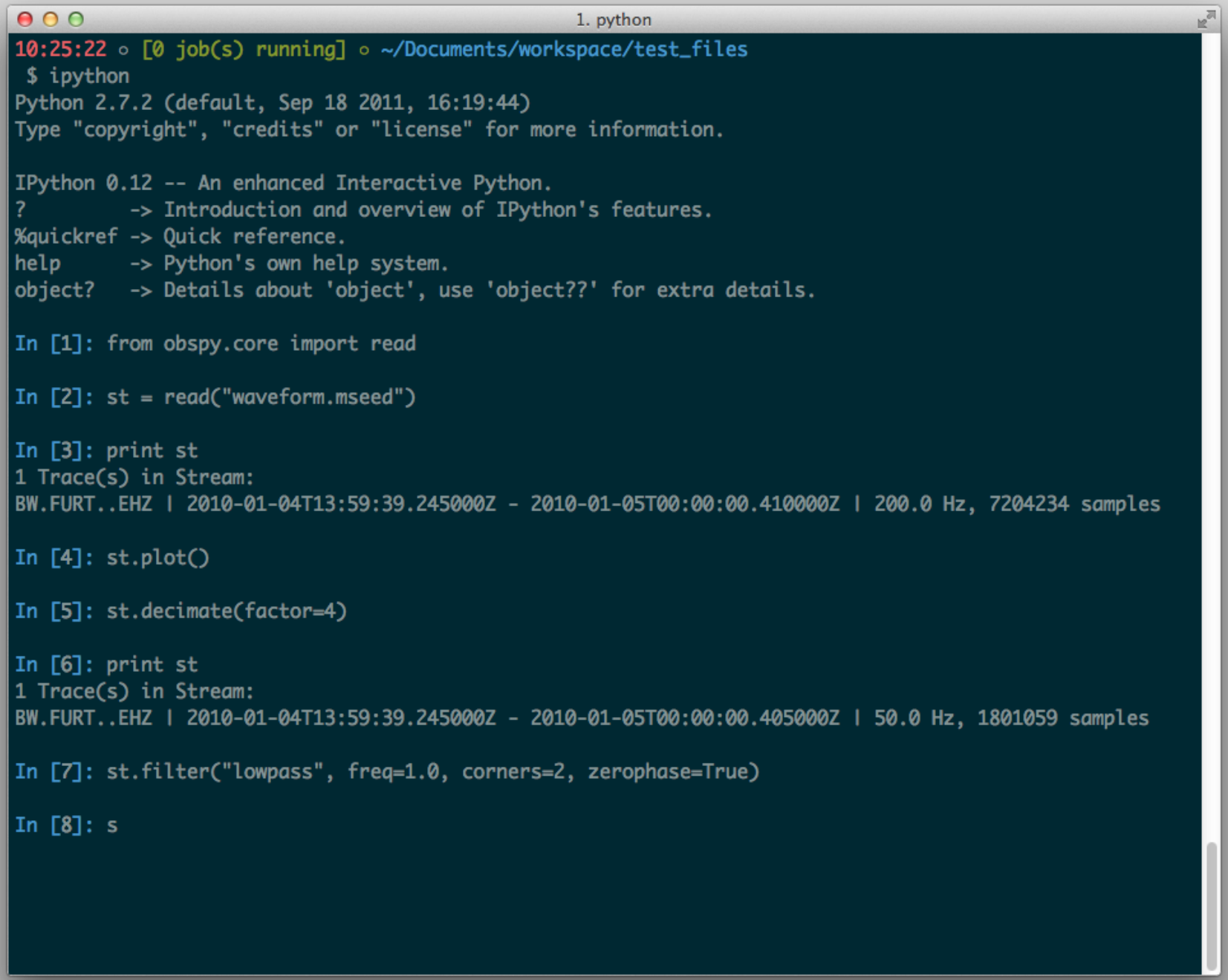
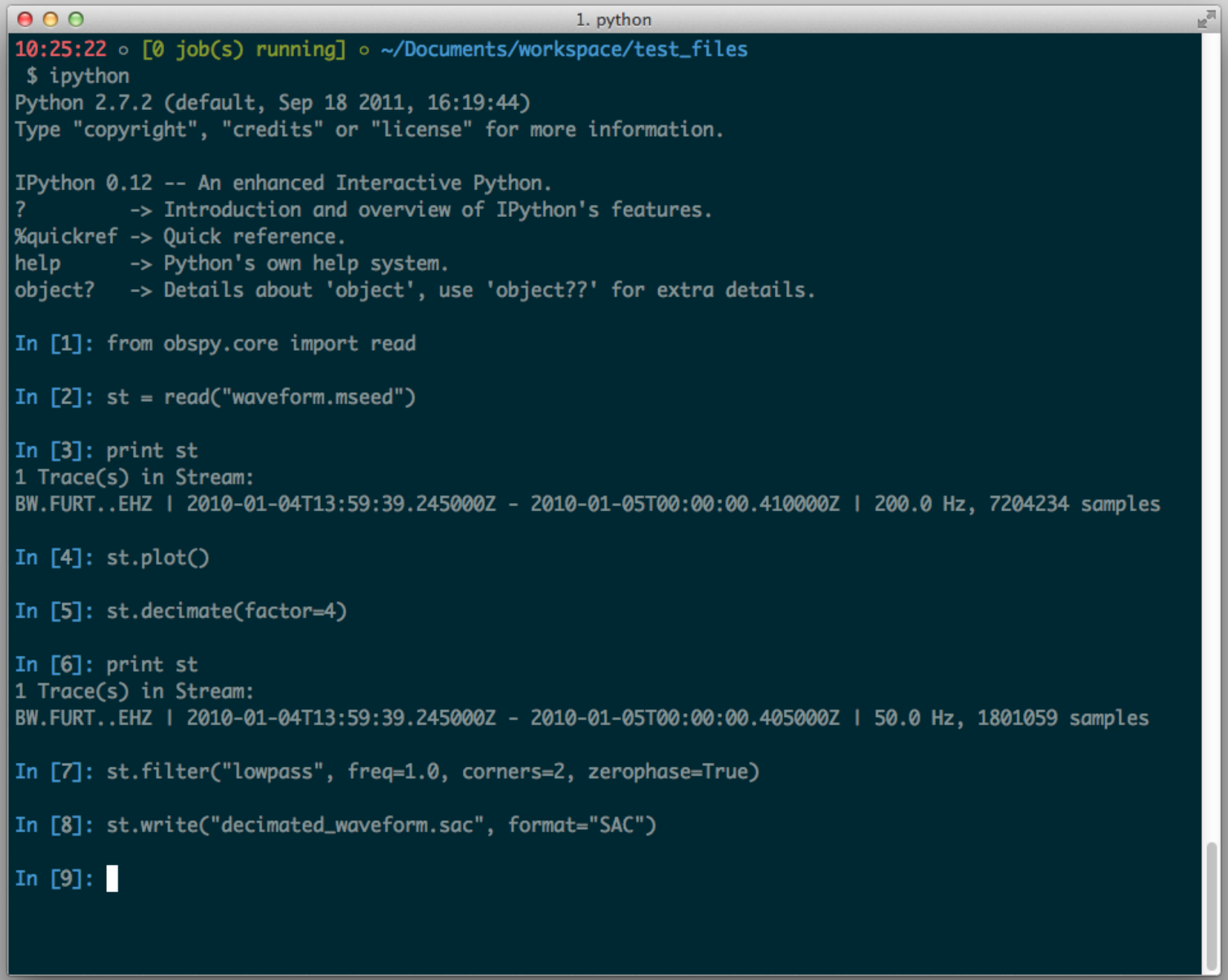


Figure 1
2010-01-04T13:59:39Z - 2010-01-05T00:00:00Z
BW.FURT..EHZ
3023
1960
897
-165
-1229
-2292
-3355
13:59:39
17:19:46
20:39:53
00:00:00

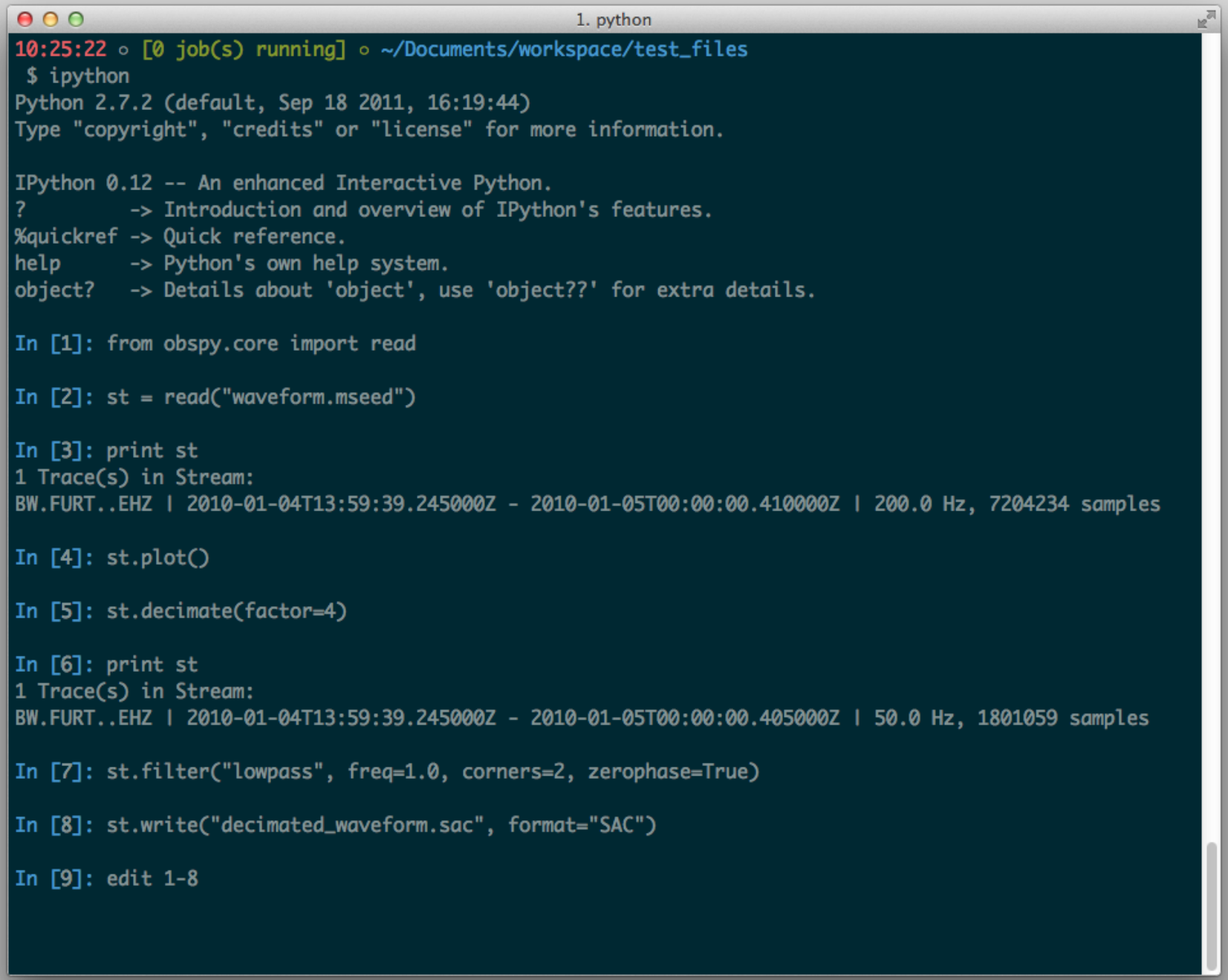

1. python
10:25:22 ○ [0 job(s) running] ○ ~/Documents/workspace/test_files
$ ipython
Python 2.7.2 (default, Sep 18 2011, 16:19:44)
Type "copyright", "credits" or "license" for more information.
IPython 0.12 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
In [1]: from obspy.core import read
In [2]: st = read("waveform.mseed")
In [3]: print st
1 Trace(s) in Stream:
BW.FURT..EHZ | 2010-01-04T13:59:39.245000Z - 2010-01-05T00:00:00.410000Z | 200.0 Hz, 7204234 samples
In [4]: st.plot()
In [5]: st.decimate(factor=4)
In [6]:

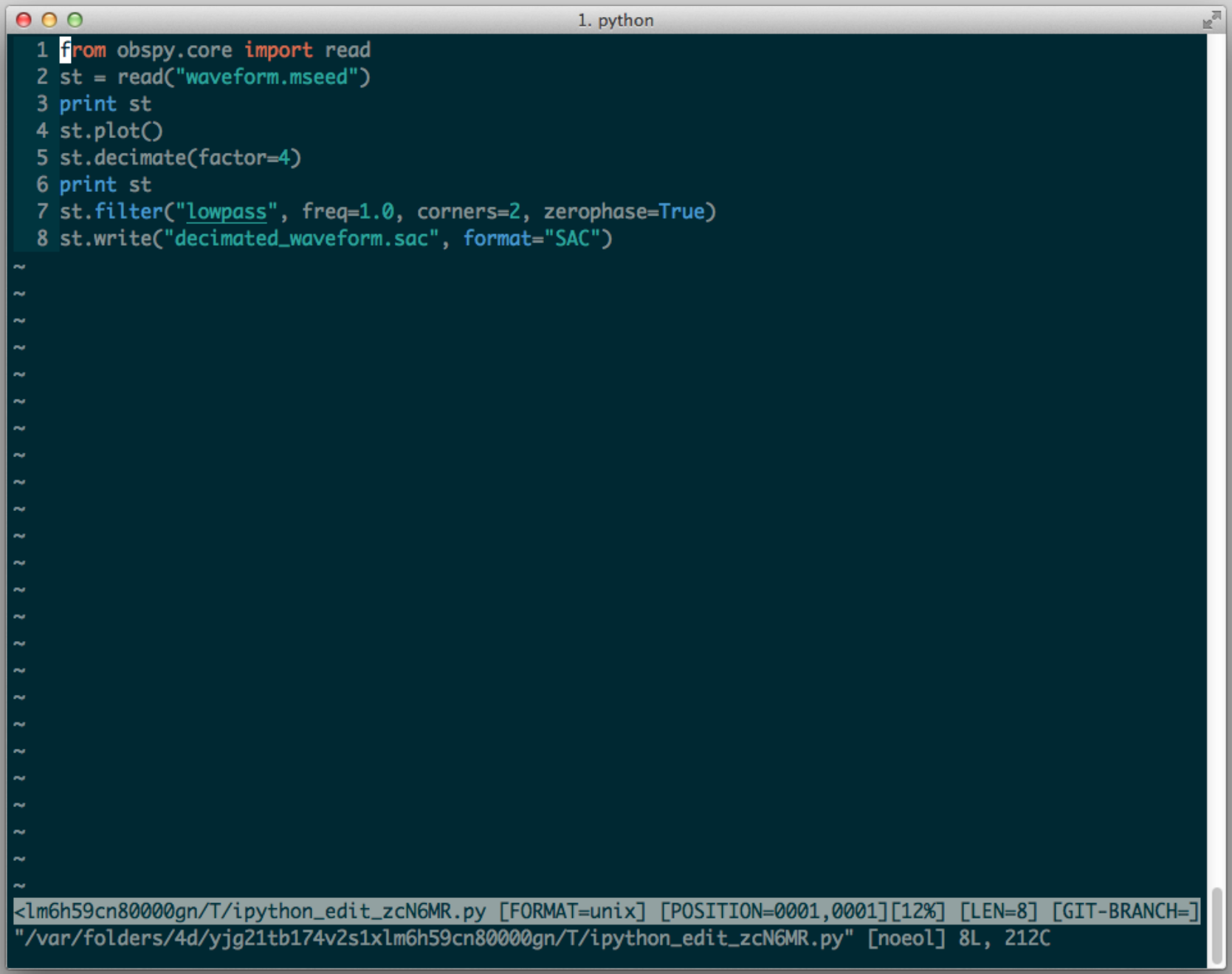

1. python
10:25:22 ○ [0 job(s) running] ○ ~/Documents/workspace/test_files
$ ipython
Python 2.7.2 (default, Sep 18 2011, 16:19:44)
Type "copyright", "credits" or "license" for more information.
IPython 0.12 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
In [1]: from obspy.core import read
In [2]: st = read("waveform.mseed")
In [3]: print st
1 Trace(s) in Stream:
BW.FURT..EHZ | 2010-01-04T13:59:39.245000Z - 2010-01-05T00:00:00.410000Z | 200.0 Hz, 7204234 samples
In [4]: st.plot()
In [5]: st.decimate(factor=4)
In [6]: print st
1 Trace(s) in Stream:
BW.FURT..EHZ | 2010-01-04T13:59:39.245000Z - 2010-01-05T00:00:00.405000Z | 50.0 Hz, 1801059 samples
In [7]:

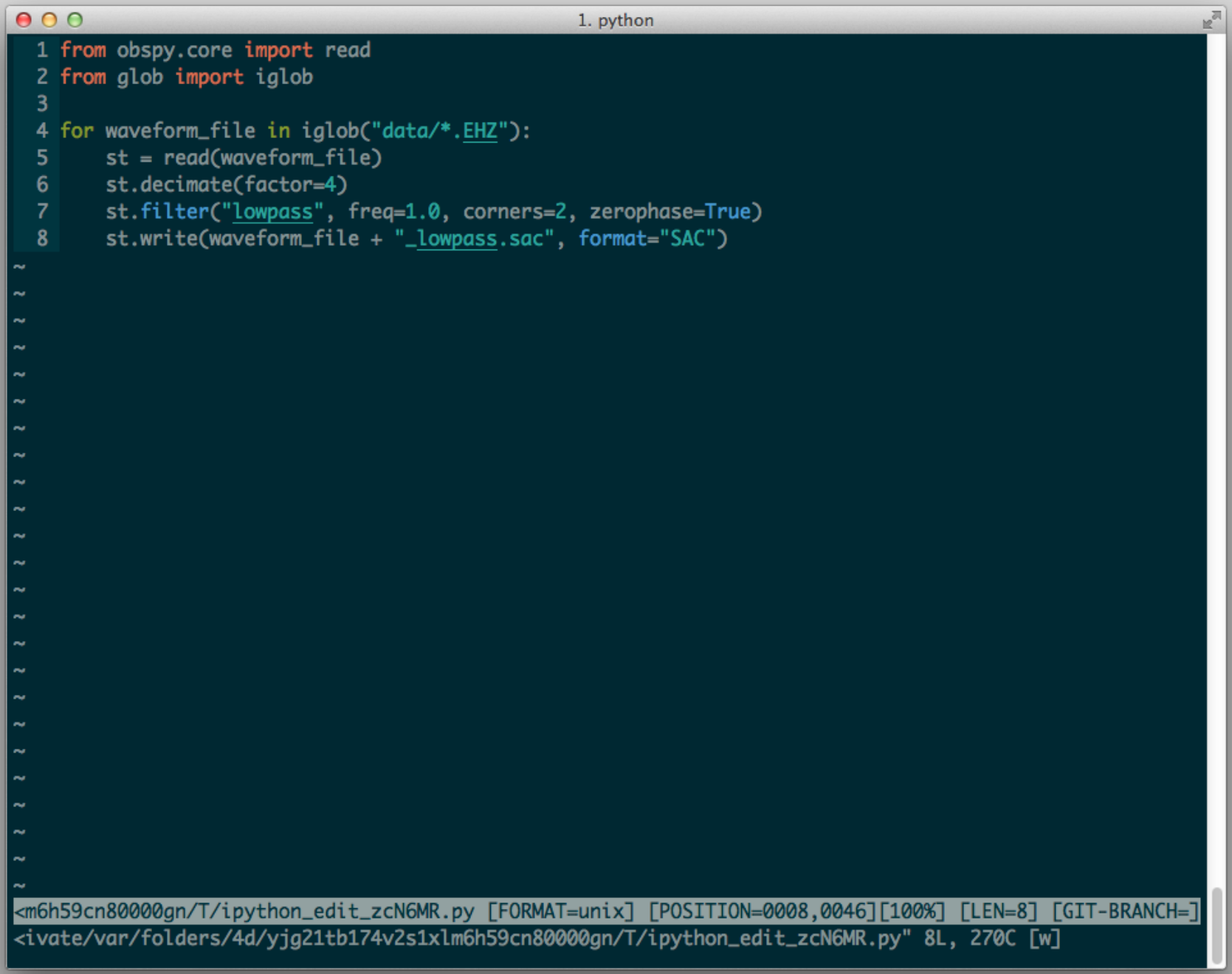

1. python
10:25:22 ○ [0 job(s) running] ○ ~/Documents/workspace/test_files
$ ipython
Python 2.7.2 (default, Sep 18 2011, 16:19:44)
Type "copyright", "credits" or "license" for more information.
IPython 0.12 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
In [1]: from obspy.core import read
In [2]: st = read("waveform.mseed")
In [3]: print st
1 Trace(s) in Stream:
BW.FURT..EHZ | 2010-01-04T13:59:39.245000Z - 2010-01-05T00:00:00.410000Z | 200.0 Hz, 7204234 samples
In [4]: st.plot()
In [5]: st.decimate(factor=4)
In [6]: print st
1 Trace(s) in Stream:
BW.FURT..EHZ | 2010-01-04T13:59:39.245000Z - 2010-01-05T00:00:00.405000Z | 50.0 Hz, 1801059 samples
In [7]: st.filter("lowpass", freq=1.0, corners=2, zerophase=True)
In [8]: s


1. python
10:25:22 ○ [0 job(s) running] ○ ~/Documents/workspace/test_files
$ ipython
Python 2.7.2 (default, Sep 18 2011, 16:19:44)
Type "copyright", "credits" or "license" for more information.
IPython 0.12 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
In [1]: from obspy.core import read
In [2]: st = read("waveform.mseed")
In [3]: print st
1 Trace(s) in Stream:
BW.FURT..EHZ | 2010-01-04T13:59:39.245000Z - 2010-01-05T00:00:00.410000Z | 200.0 Hz, 7204234 samples
In [4]: st.plot()
In [5]: st.decimate(factor=4)
In [6]: print st
1 Trace(s) in Stream:
BW.FURT..EHZ | 2010-01-04T13:59:39.245000Z - 2010-01-05T00:00:00.405000Z | 50.0 Hz, 1801059 samples
In [7]: st.filter("lowpass", freq=1.0, corners=2, zerophase=True)
In [8]: st.write("decimated_waveform.sac", format="SAC")
In [9]:

Now do the same for 1000 files


1. python
10:25:22 ○ [0 job(s) running] ○ ~/Documents/workspace/test_files
$ ipython
Python 2.7.2 (default, Sep 18 2011, 16:19:44)
Type "copyright", "credits" or "license" for more information.
IPython 0.12 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
In [1]: from obspy.core import read
In [2]: st = read("waveform.mseed")
In [3]: print st
1 Trace(s) in Stream:
BW.FURT..EHZ | 2010-01-04T13:59:39.245000Z - 2010-01-05T00:00:00.410000Z | 200.0 Hz, 7204234 samples
In [4]: st.plot()
In [5]: st.decimate(factor=4)
In [6]: print st
1 Trace(s) in Stream:
BW.FURT..EHZ | 2010-01-04T13:59:39.245000Z - 2010-01-05T00:00:00.405000Z | 50.0 Hz, 1801059 samples
In [7]: st.filter("lowpass", freq=1.0, corners=2, zerophase=True)
In [8]: st.write("decimated_waveform.sac", format="SAC")
In [9]: edit 1-8


1. python
1 from obspy.core import read
2 st = read("waveform.mseed")
3 print st
4 st.plot()
5 st.decimate(factor=4)
6 print st
7 st.filter("lowpass", freq=1.0, corners=2, zerophase=True)
8 st.write("decimated_waveform.sac", format="SAC")
~
<lm6h59cn80000gn/T/ipython_edit_zcN6MR.py [FORMAT=unix] [POSITION=0001,0001][12%] [LEN=8] [GIT-BRANCH=]
"/var/folders/4d/yjg21tb174v2s1xlm6h59cn80000gn/T/ipython_edit_zcN6MR.py" [noeol] 8L, 212C


1. python
1 from obspy.core import read
2 from glob import iglob
3
4 for waveform_file in iglob("data/*.EHZ"):
5 st = read(waveform_file)
6 st.decimate(factor=4)
7 st.filter("lowpass", freq=1.0, corners=2, zerophase=True)
8 st.write(waveform_file + "_lowpass.sac", format="SAC")
~
<m6h59cn80000gn/T/ipython_edit_zcN6MR.py [FORMAT=unix] [POSITION=0008,0046][100%] [LEN=8] [GIT-BRANCH=]
<ivate/var/folders/4d/yjg21tb174v2s1xlm6h59cn80000gn/T/ipython_edit_zcN6MR.py" 8L, 270C [w]

# Simple example - Reading a waveform file

```
>>> from obspy.core import read
>>> st = read("waveform.mseed")
>>> print st
1 Trace(s) in Stream:
BW.FURT..EHZ | 2010-01-04...| 200.0 Hz, 7204234 samples
```

- Automatic file format detection.
- Always results in a Stream object.
- Raw data available as a numpy.ndarray.

```
>>> st[0].data
array([-426, -400, ..., -489, -339], dtype=int32)
```

# Simple example - The resulting Stream object

- A **Stream** object is a collection of **Trace** objects
- A **Trace** object is a single, continuous waveform data block
- Working with them is easy:
  - **st.filter()** - Filter all attached traces.
  - **st.trim()** - Cut all traces.
  - **st.resample() / st.decimate()** - Change the sampling rate.
  - **st.trigger()** - Run triggering algorithms.
  - **st.plot() / st.spectrogram()** - Visualize the data.
  - **st.simulate(), st.merge(), st.normalize(), st.detrend(), ...**
- A **Stream** object can also be exported to many formats making ObsPy a good tool for converting between different file formats.

```
>>> st.write("output_file.sac", format="SAC")
```

# Handling time - The UTCDateTime class

- All absolute time values are consistently handled with this class.
- No ambiguities, e.g. timezones, leap seconds, ...
- Based on a high precision POSIX timestamp
- Easy usage:

```
>>> from obspy.core import UTCDateTime
>>> time = UTCDateTime(2011, 11, 11)
>>> one_hour_later = time + 3600
```

# Clients - Getting waveform data from the web

**ObsPy** has clients for **NERIES**, **IRIS**, **ArcLink**, **SeisHub** and **Earthworm**.

```
>>> from obspy.neries import Client
>>> from obspy.core import UTCDateTime
>>> client = Client(user="test@obspy.org")
>>> dt = UTCDateTime("2009-08-20 04:03:12")
>>> st = client.getWaveform("BW", "RJOB", "", "EH*",
                            dt - 3, dt + 15)
```

- Similar interfaces for the other clients.
- The returned Stream object is already known.
- Therefore it does not matter if the source of the data is the internet or a local file.

# Clients - Retrieving other data

The webservices are not limited to retrieving waveform data. Depending on the client module used, the available data includes:

- Event data (soon to be integrated in the new Event class).
- Inventory and response data.
- Availability information.
- ...

# Inventory Data

- Can currently read/write/convert between SEED and XML-SEED.
- RESP file support.
- StationXML support is planned.

```
000001V 010009402.3121970,001,00:00:00.0000~2038,001,00:00:00.0000~
2009,037,04:32:41.0000~BayernNetz~~0110032002RJOB 000003RJOB 000008
...
```

```
<?xml version='1.0' encoding='utf-8'?>
<xseed version="1.0">
  <volume_index_control_header>
    <volume_identifier blockette="010">
      <version_of_format>2.4</version_of_format>
      <logical_record_length>12</logical_record_length>
      <beginning_time>1970-01-01T00:00:00</beginning_time>
      <end_time>2038-01-01T00:00:00</end_time>
...
```

# Events - Work in progress

- Aims to get a unique interface independent of the data source.
- Currently has limited QuakeML support.
- In development. If you have any good ideas let us know.

```
>>> from obspy.core.events import readEvents
>>> url = "http://www.seismicportal.eu/services/..."
>>> catalog = readEvents(url)
>>> print catalog
99 Event(s) in Catalog:
2012-04-11T10:43:09.400000Z |  ... | 8.2 Mw | ...
2012-04-11T08:38:33.000000Z |  ... | 8.4 M  | ...
...
```

# Advanced example - Instrument correction

```
from obspy.core import read

paz_sts2 = {
        "poles": [-0.037004 + 0.037016j, ...
        "zeros": [0j, 0j],
        "gain": 60077000.0,
        "sensitivity": 2516778400.0}

st = read()
st.simulate(paz_remove=paz_sts2)
```

In a real world case the response information would be read from a SEED or RESP file or retrieved from one of the webservices.

# What's next?

- Getting more developers and external contributions
- Events
- StationXML
- More powerful correction module
- Suggestions? Let us know.

# www.obspy.org

- Documentation and extensive tutorial.
- Gallery to showcase some features.
- **mailing list** - subscribe for updates and discussions about the project.
- Source code repository and bug tracker.
- Automatic, daily running tests bots (http://tests.obspy.org)
- Get in touch!


- Moritz Beyreuther et al. (2010) **ObsPy: A Python Toolbox for Seismology**, SRL, 81(3), 530-533
- Tobias Megies et al. (2011) **ObsPy – What can it do for data centers and observatories?** Annals Of Geophysics, 54(1), 47-58. doi:10.4401/ag-4838

# Installing ObsPy

- **Debian/Ubuntu** - via package management (http://deb.obspy.org)
- **Windows** - Windows installer (automatically installs all dependencies)
- **OSX $\geq$ 10.6** - one-click-install application (contains all dependencies)

For the most recent additions and bug fixes install the developer version.

Detailed instructions for all platforms can be found on **www.obspy.org**.

# Projects using ObsPy

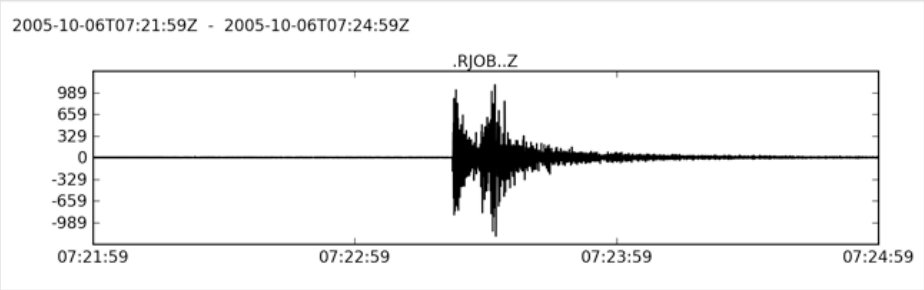
- NERA wavesdownloader (http://webservices.rm.ingv.it/wavesdownloader)
- The ADMIRE project (http://www.admire-project.eu)
- Antelope Python moment code (http://eqinfo.ucsd.edu/ rnewman/howtos/antelope/moment_tensor)
- The new Python based Seismic Handler (http://www.seismic-handler.org)
- The Whisper project (http://whisper.obs.ujf-grenoble.fr)
- ...

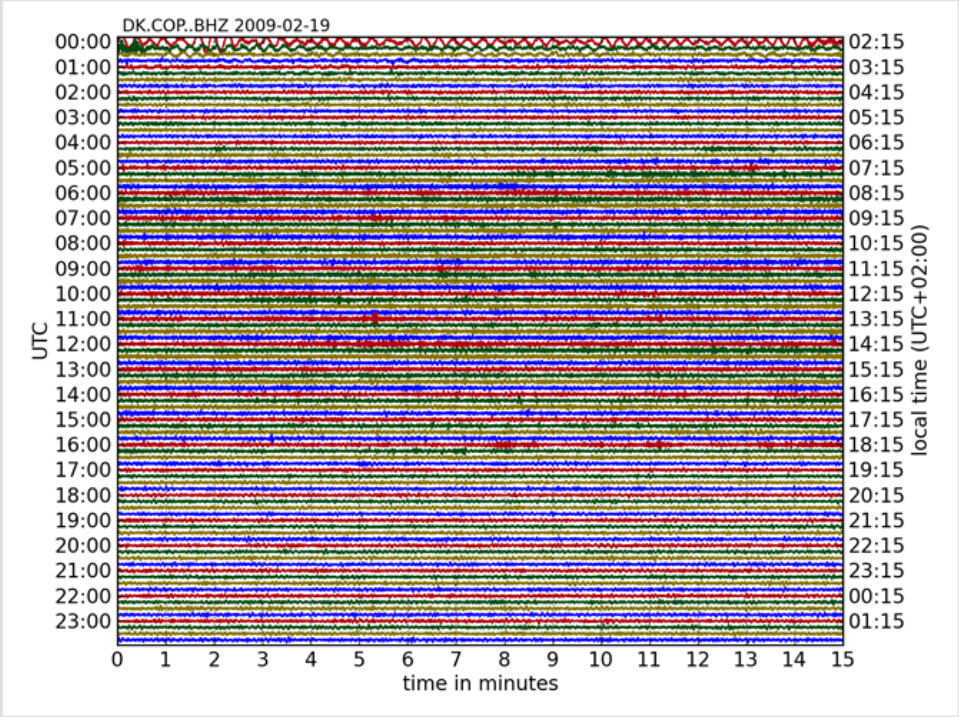
## Thanks for your attention.

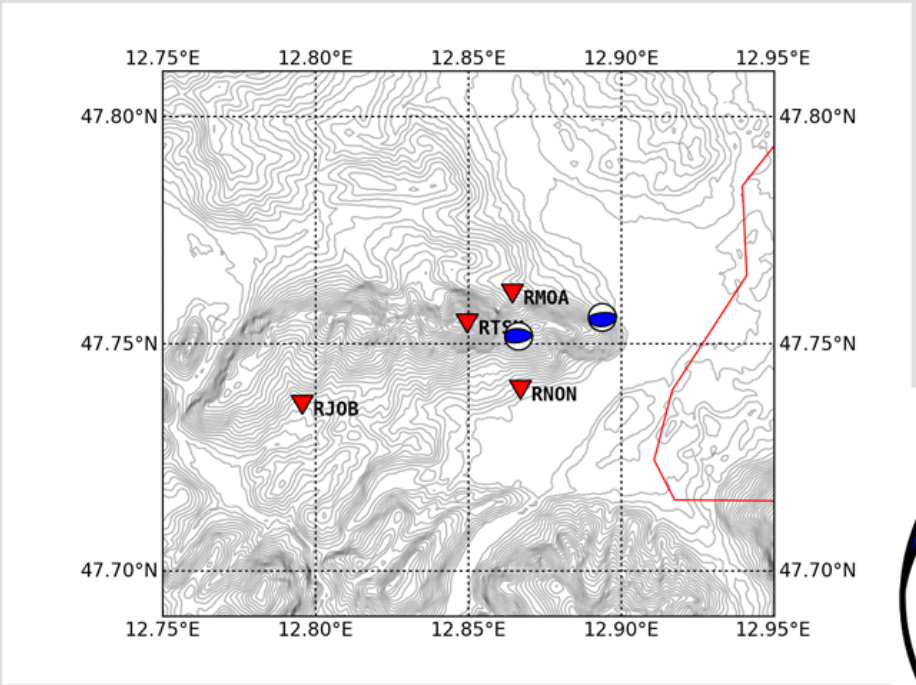
There are currently plans to host an ObsPy workshop in September at the ETH in Zurich. Subscribe to the **mailing list at www.obspy.org** for upcoming details.

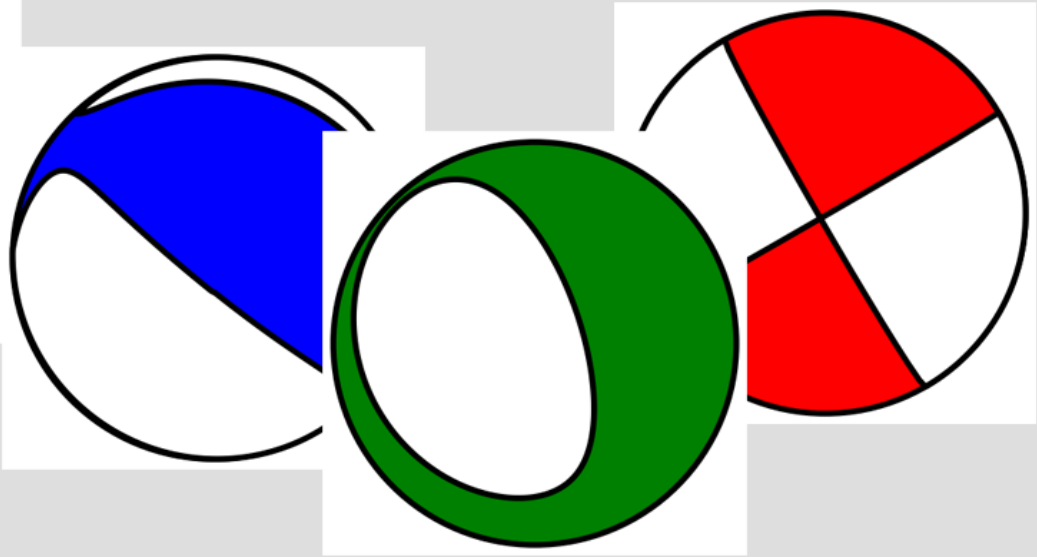
### Credits and further reading

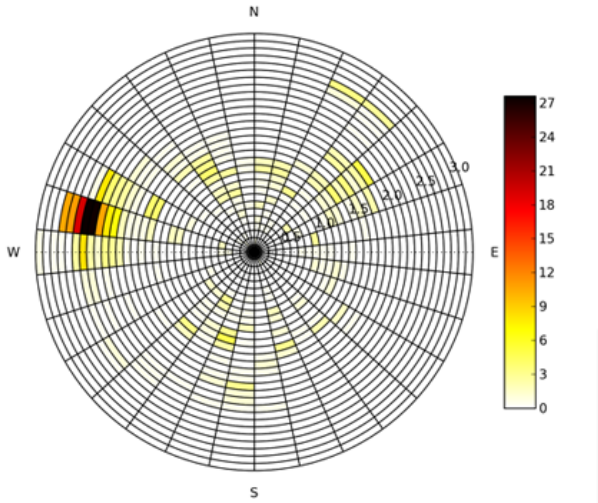
- The Python Tutorial (http://docs.python.org/tutorial/)
- Sebastian Heimann - The Informal Python Boot Camp (http://emolch.org/pythonbootcamp.html)
- Hoyt Koepke - 10 Reasons Python Rocks for Research (http://www.stat.washington.edu/ hoytak/blog/whypython.html)
- Software Carpentry (http://software-carpentry.org/4_0/python/)
- Kent S Johnson - Python Rocks! and other rants (http://kentsjohnson.com/blog/arch_Python.html)

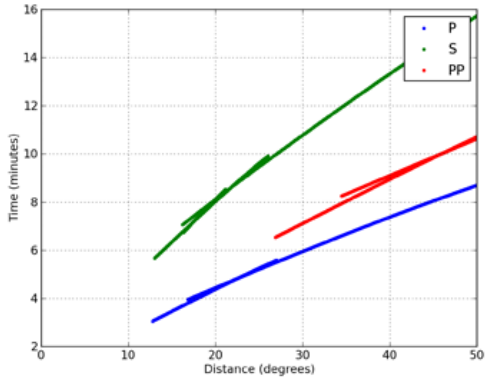

2005-10-06T07:21:59Z - 2005-10-06T07:24:59Z
.RJOB..Z
989
659
329
0
-329
-659
-989
07:21:59
07:22:59
07:23:59
07:24:59

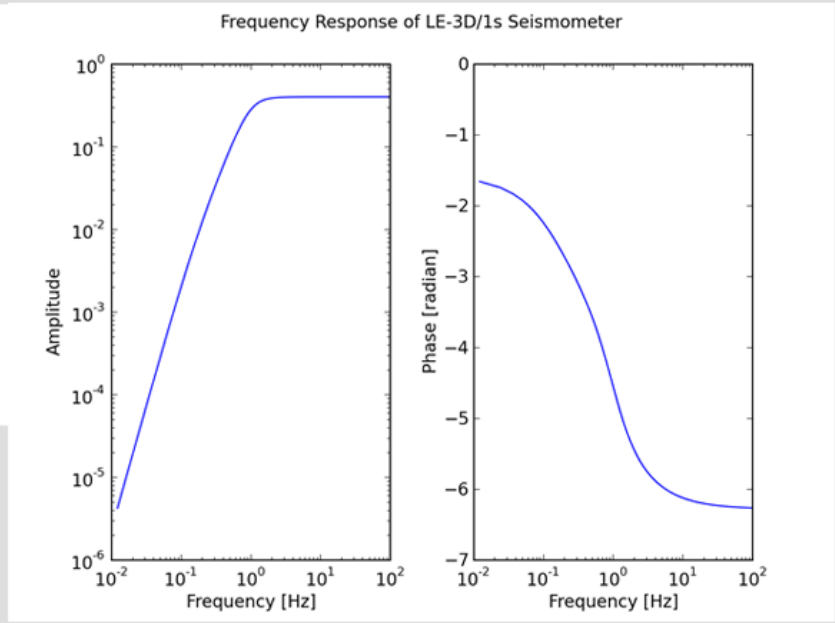

DK.COP..BHZ 2009-02-19
UTC
local time (UTC+02:00)
time in minutes

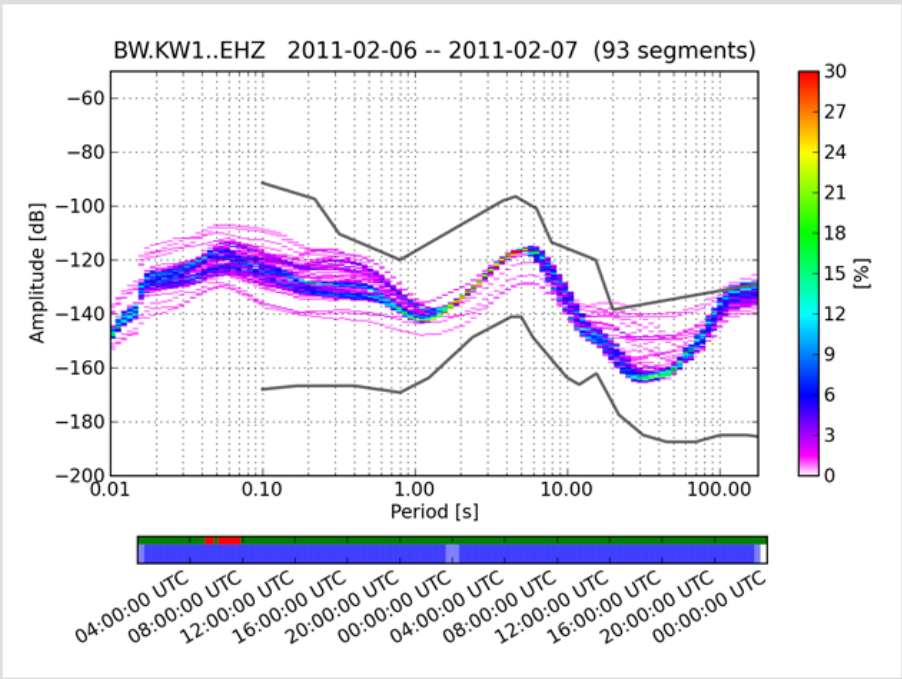

12.75°E
12.80°E
12.85°E
12.90°E
12.95°E
47.80°N
47.75°N
47.70°N
RMOA
RTS
RNON
RJOB

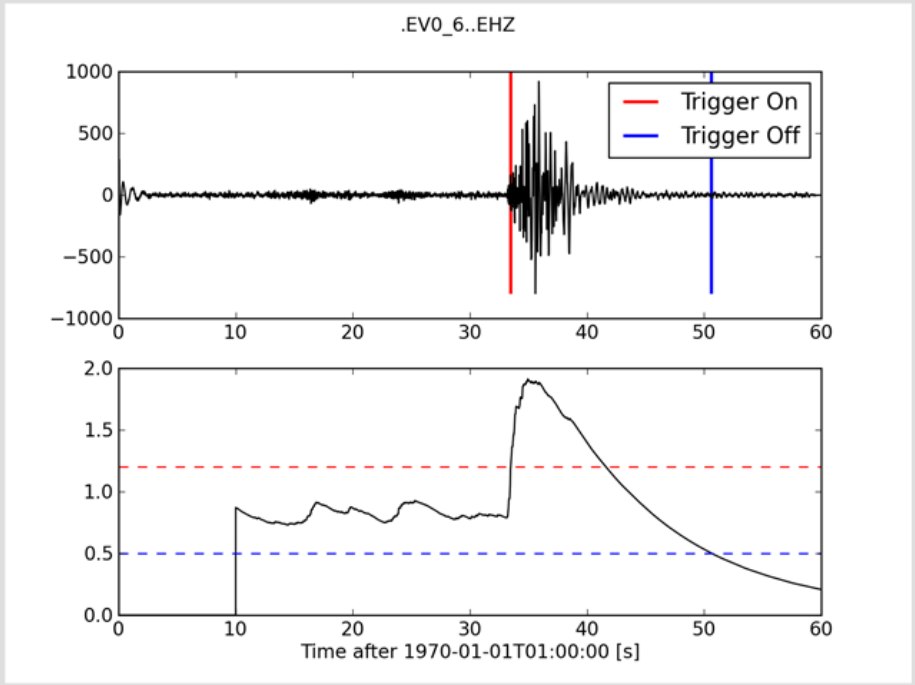


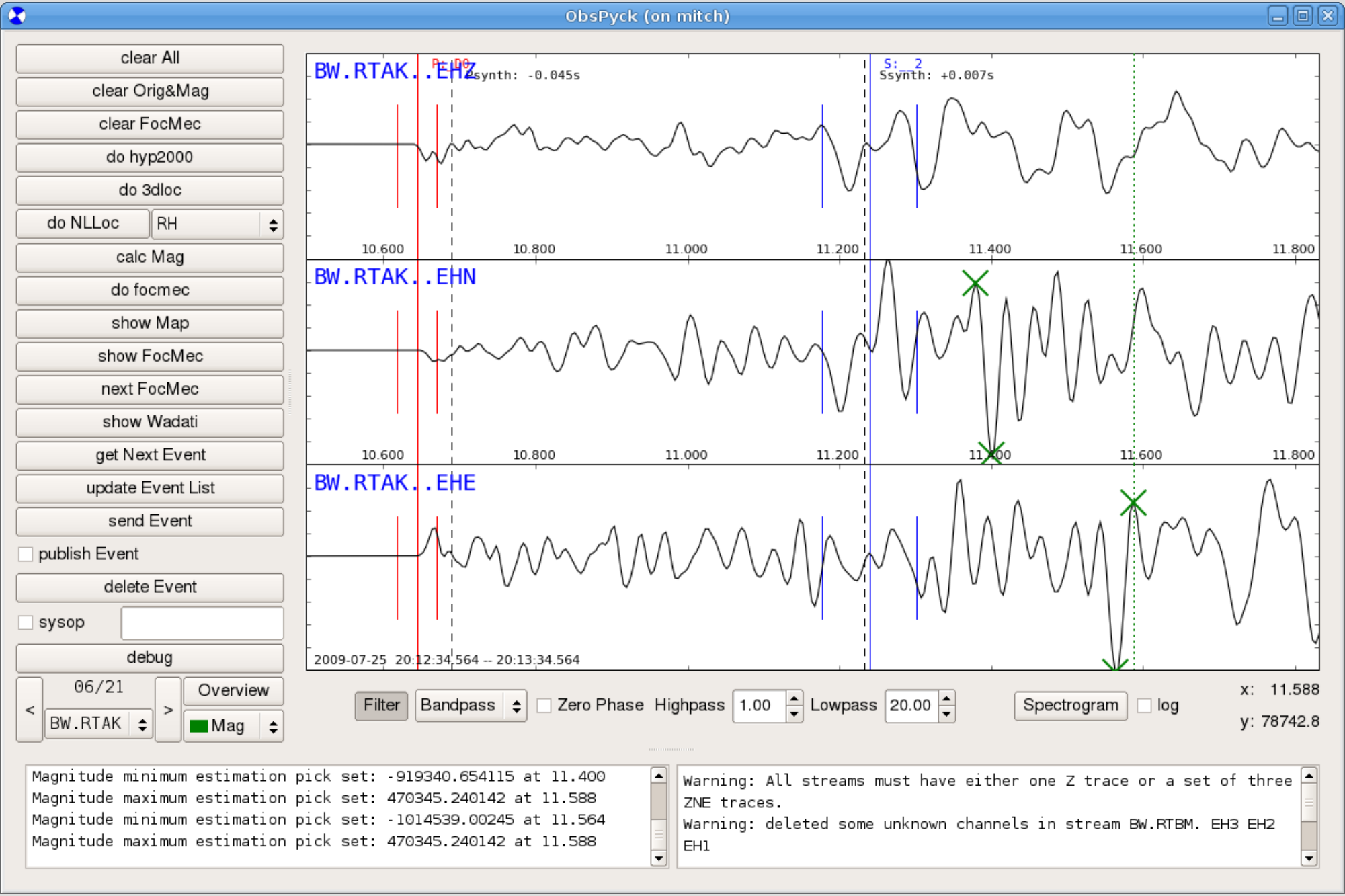

N
W
E
S
0.5
1.0
1.5
2.0
2.5
3.0
27
24
21
18
15
12
9
6
3
0


P
S
PP
Time (minutes)
Distance (degrees)


Frequency Response of LE-3D/1s Seismometer
Amplitude
Frequency [Hz]
Phase [radian]
Frequency [Hz]


BW.KW1..EHZ 2011-02-06 -- 2011-02-07 (93 segments)
Amplitude [dB]
Period [s]
[%]
04:00:00 UTC
08:00:00 UTC
12:00:00 UTC
16:00:00 UTC
20:00:00 UTC
00:00:00 UTC


.EV0_6..EHZ
Trigger On
Trigger Off
Time after 1970-01-01T01:00:00 [s]


ObsPyck (on mitch)
clear All
clear Orig&Mag
clear FocMec
do hyp2000
do 3dloc
do NLLoc
RH
calc Mag
do focmec
show Map
show FocMec
next FocMec
show Wadati
get Next Event
update Event List
send Event
publish Event
delete Event
sysop
debug
06/21
<
>
BW.RTAK
Overview
Mag
BW.RTAK..EHZ
P: 0.0
Ssynth: -0.045s
S: 2
Ssynth: +0.007s
10.600
10.800
11.000
11.200
11.400
11.600
11.800
BW.RTAK..EHN
BW.RTAK..EHE
2009-07-25 20:12:34.564 -- 20:13:34.564
Filter
Bandpass
Zero Phase
Highpass
1.00
Lowpass
20.00
Spectrogram
log
x: 11.588
y: 78742.8
Magnitude minimum estimation pick set: -919340.654115 at 11.400
Magnitude maximum estimation pick set: 470345.240142 at 11.588
Magnitude minimum estimation pick set: -1014539.00245 at 11.564
Magnitude maximum estimation pick set: 470345.240142 at 11.588
Warning: All streams must have either one Z trace or a set of three ZNE traces.
Warning: deleted some unknown channels in stream BW.RTBM. EH3 EH2 EH1