

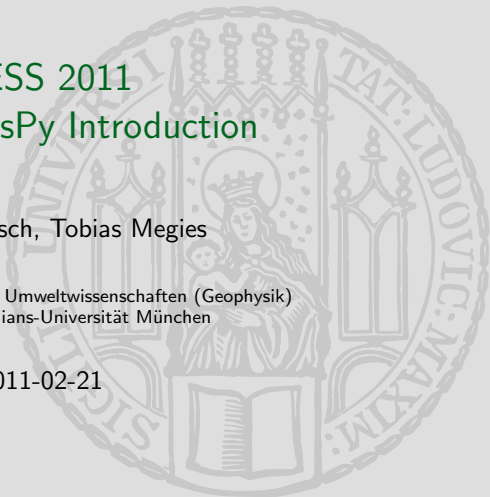
MESS 2011

Python/ObsPy Introduction

Robert Barsch, Tobias Megies

Department für Geo- und Umweltwissenschaften (Geophysik)
Ludwig-Maximilians-Universität München

2011-02-21



How to Work on the Practicals..

- Either..

- ▶ work line by line in IPython shell
- ▶ when it's working: save history and condense it

```
>>> %history [number_of_lines] [-n] [-f output_file]
```

- or..

- ▶ work on your program in a text editor
- ▶ in a second window, run program in an IPython shell and continue work at the end

```
$ ipython -i  
>>> run -i PROGRAM.PY
```

- ▶ (caution: best do this in a "fresh" IPython shell)
- ▶ extend program with appropriate lines of code and run it again in a new IPython shell

Getting Help..

IPython

- get help for a function: `>>> command?`
- have a look at the implementation: `>>> command??`
- search for variables/functions/modules starting with "ab": `>>> ab<Tab>`
- what's the value? `>>> variable`
- what's the type? `>>> type(variable)`
- which variables are assigned anyway?? `>>> whos`
- what attributes/methods are there? `>>> variable.<Tab>`
- get help for a variable's method: `>>> variable.command?`
- what functions are available in a module? `>>> module.<Tab>`

Getting Help..

- ObsPy web pages
 - ▶ Tutorial
 - ▶ <http://obspy.org/wiki/ObspyTutorial>
 - ▶ <file:///home/messuser/obspy/tutorial/ObspyTutorial.html>
 - ▶ API
 - ▶ <http://docs.obspy.org/>
 - ▶ <file:///home/messuser/obspy/docs/index.html>
- Python/Numpy/Scipy API
 - ▶ <http://docs.python.org/>
 - ▶ <file:///home/messuser/obspy/python/python-docs/index.html>
 - ▶ <http://docs.scipy.org/doc/numpy/reference/>
 - ▶ <file:///home/messuser/obspy/python/numpy-docs/index.html>
 - ▶ <http://docs.scipy.org/doc/scipy/reference/>
 - ▶ <file:///home/messuser/obspy/python/scipy-docs/index.html>

ObsPy Data Types – Overview

- `UTCDateTime`
 - ▶ extension of the Python `datetime` object
 - ▶ stores a time stamp
- `Stats`
 - ▶ extension of the Python `dict` object
 - ▶ stores header information of waveforms
- `Trace`
 - ▶ stores a single-channel, continuous piece of waveform data
 - ▶ consisting of waveform data and header information
- `Stream`
 - ▶ stores multiple traces (e.g. Z, N, E traces of one station)
- all of them defined in `obspy.core`
 - ▶ `from obspy.core import UTCDateTime`

ObsPy Data Types – UTCDateTime

- UTCDateTime
 - ▶ used to handle all time information in ObsPy
 - ▶ initialize via
 - ▶ `t = UTCDateTime("2011-02-21T08:00:00.00Z")`
 - ▶ `t = UTCDateTime(2011, 2, 21, 8)`
 - ▶ ...
 - ▶ several attributes/methods
(e.g. `t.microsecond`, `t.julday`, `t.weekday()`, ...)
 - ▶ important operations
 - ▶ subtracting two UTCDateTime objects gives time difference in seconds
 - ▶ adding/subtracting int/float returns new UTCDateTime object
 - ▶ see [ObsPy documentation](#)

Exercises – UTCDateTime

- A
 - ▶ the morning sessions start at 8 and are 3 hours..
assume we want to have the coffee break 1234 seconds and 5 microseconds before the session ends. What time is the break?
 - ▶ assume you had your last cup of coffee yesterday at breakfast. How many minutes do you have to survive with that cup of coffee?
- B
 - ▶ how many days from today is your birthday this year?
 - ▶ what day of week is it?
 - ▶ you want to have your birthday party at the first saturday after your birthday, what date is the party?
- C
 - ▶ some of your friends always seem to find an excuse not to come, for the next fifty years print the date of your party sticking to that scheme, we are superstitious and do not leave the house on Friday 13th.
 - ▶ print a list of dates and count the days we have to take off from now till the end of next year.

Exercises – UTCDateTime

- A
 - ▶ the morning sessions start at 8 and are 3 hours..
assume we want to have the coffee break 1234 seconds and 5 microseconds before the session ends. What time is the break?
 - ▶ assume you had your last cup of coffee yesterday at breakfast. How many minutes do you have to survive with that cup of coffee?
- B
 - ▶ how many days from today is your birthday this year?
 - ▶ what day of week is it?
 - ▶ you want to have your birthday party at the first saturday after your birthday, what date is the party?
- C
 - ▶ some of your friends always seem to find an excuse not to come..
for the next fifty years print the date of your party sticking to that scheme
 - ▶ we are superstitious and do not leave the house on friday 13th..
print a list of dates and count the days we have to take off from now till the end of next year..

ObsPy Data Types – Stats

- Stats – header information for waveform data
 - ▶ contains at least the following keys
 - ▶ `stats.network` – network code (str)
 - ▶ `stats.station` – station code (str)
 - ▶ `stats.location` – location code (str)
 - ▶ `stats.channel` – channel code (str)
 - ▶ `stats.starttime` – time of first sample (UTCDateTime)
 - ▶ `stats.sampling_rate` – sampling rate in Hz (float)
 - ▶ `stats.npts` – number of samples (int)
 - ▶ derived keys
 - ▶ `stats.endtime` – time of last sample (UTCDateTime)
 - ▶ `stats.delta` – sampling interval (float)
 - ▶ optional keys
 - ▶ `stats._format` – format of original data file (str, e.g. "MSEED")
 - ▶ `stats.paz` – poles, zeros, sensitivity and gain of instrument (dict)
 - ▶ `stats.coordinates` – longitude, latitude and elevation of station (dict)
 - ▶ ...
 - ▶ see [ObsPy documentation](#)

ObsPy Data Types – Trace

- Trace – continuous waveform data
 - ▶ usually constructed internally during `read(...)` or `getWaveform(...)`
 - ▶ consists of
 - ▶ `tr.data` – waveform data as a `numpy.ndarray` instance
 - ▶ `tr.stats` – header information as a `Stats` instance
 - ▶ built-in methods
 - ▶ `tr.id` – complete channel id in SEED standard (e.g. "BW.RJOB..BHZ")
 - ▶ `tr.plot()` – shows preview plot of trace
 - ▶ `tr.copy()` – returns copy of trace (most operations work in-place)
 - ▶ `tr.trim(starttime, endtime)` – cut trace to specified time span
 - ▶ `tr.filter("type", **kwargs)` – filter waveform data
 - ▶ `tr.simulate(paz_remove, paz_simulate, **kwargs)`
– apply instrument correction/simulation
 - ▶ `tr.write("filename", "format")` – write waveform to local file
 - ▶ ...
 - ▶ many built-in methods on `tr.data` (`numpy.ndarray`)!
 - ▶ see [ObsPy documentation](#)
 - ▶ see [Numpy documentation – ndarray](#)

Exercises – Trace

- A
 - ▶ make a trace with all zeros (e.g. `numpy.zeros(200)`) and an ideal pulse at the center
 - ▶ fill in some station information (`network, station`)
 - ▶ print trace summary and display the preview plot of the trace
 - ▶ change the sampling rate to 20Hz
 - ▶ change the `starttime` to the start time of this sessions
 - ▶ print trace summary and display the preview plot of the trace again

Exercises – Trace

- B
 - ▶ use `tr.filter(...)` and apply a lowpass with 1s corner frequency
 - ▶ display the preview plot, there are a few seconds of zeros that we can cut off
 - ▶ use `tr.trim(...)` to remove some of the zeros at start and end
- C
 - ▶ scale up the amplitudes of the trace by a factor of 500
 - ▶ make a copy of the original trace
 - ▶ add standard normal gaussian noise to the copied trace (use `numpy.random.randn(...)`)
 - ▶ change the station name of the copied trace
 - ▶ display the preview plot of the new trace

ObsPy Data Types – Stream

- Stream – collection of Trace objects in a list-like container
 - ▶ usually returned by a `read(...)` or `getWaveform(...)` call
 - ▶ `print st` – prints summary of all traces
 - ▶ `print len(st)` – prints number of traces in stream
 - ▶ list-like operations
 - ▶ `st[i]` – return trace at index `i`
 - ▶ `st.append(tr)` – add a single trace
 - ▶ `st.extend(st)` – add a list of traces
 - ▶ `st.remove(tr)` – remove specified trace from stream
 - ▶ `st.pop(i)` – remove trace at specified index and return it
 - ▶ `st.sort(...)` – sort traces in stream according to specified criteria
 - ▶ other built-in methods
 - ▶ `st.select(**kwargs)`
– return new stream with matching traces (e.g. `component="Z"`)
 - ▶ `st.merge(method)` – merge traces with identical id
 - ▶ `st.printGaps()` – prints summary of gaps in the stream
 - ▶ many built-in methods of Trace (`trim`, `filter`, `simulate`,...)
 - ▶ see [ObsPy documentation](#)

Exercises – Stream

- A
 - ▶ read the example earthquake data into a stream object (`read()` without arguments)
 - ▶ print the stream summary and display the preview plot
 - ▶ assign the first trace to a new variable and then remove that trace from the original stream
 - ▶ print the summary for the single trace and for the stream

Exercises – Stream

- B
 - ▶ read the example earthquake data again
 - ▶ make a dictionary with `paz` information, assign poles at `[-0.037+0.037j, -0.037-0.037j]`, zeros at `[0j, 0j]`, the sensitivity of `2.517e9` and unity gain
 - ▶ remove the instrument response using this `paz` dictionary
 - ▶ print the data maximum and minimum of the first trace (now in m/s)
 - ▶ save the data to a local file in `MSEED` format
- C
 - ▶ read the example earthquake data again
 - ▶ change the station name for all traces in the stream
 - ▶ read the example earthquake data yet again and add the traces to the first stream
 - ▶ print the summary for the resulting stream and display the preview plot
 - ▶ select the `Z` traces and assign this stream to another variable
 - ▶ filter the `Z` components with a highpass at 5Hz
 - ▶ display the preview plot of the `Z` component stream
 - ▶ display the preview plot of the original stream