Questions and Solutions

UTCDateTime Exercises

- Calculate the number of hours passed since your birth.
  - The current date and time can be obtained with **"UTCDateTime()"**
  - Optional: Include the correct time zone

```
from obspy.core import UTCDateTime
# Set the birth date assuming the timezone at date and place of birth was UTC+1.
birth = UTCDateTime("1980-01-02T03:04:05+01:00")
current_time = UTCDateTime()
print "Hours passed since birth:", (current_time - birth) / 3600.0
```

- Get a list of 10 UTCDateTime objects, starting yesterday at 10:00 with a spacing of 90 minutes.

```
from obspy.core import UTCDateTime

yesterday = UTCDateTime() - 86400
yesterday_ten_oclock = UTCDateTime(yesterday.year, yesterday.month, yesterday.day, 10)
time_list = [yesterday_ten_oclock + _i * 90 * 60 for _i in range(10)]
```

- The first session starts at 09:00 and lasts for 3 hours and 15 minutes. Assuming we want to have the coffee break 1234 seconds and 5 microseconds before it ends. At what time is the coffee break?

```
from obspy.core import UTCDateTime

session_start = UTCDateTime(2012, 9, 7, 9)
session_end = session_start + 3 * 3600 + 15 * 60
coffee_break = session_end - 1234.000005
print "Coffee at %s" % coffee_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?

```
from obspy.core import UTCDateTime

yesterday = UTCDateTime() - 86400
yesterday_breakfast = UTCDateTime(yesterday.year, yesterday.month, yesterday.day, 8)
now = UTCDateTime()
hours_without_coffee = (now - yesterday_breakfast) / 3600
print "Hours without coffee:", hours_without_coffee
```

Waveform Exercises

Trace Exercise 1

- 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 plot the trace
- Change the sampling rate to 20 Hz
- Change the *starttime* to the start time of this session
- Print the trace summary and plot the trace again

```
from obspy.core import Trace, UTCDateTime
import numpy as np

tr = Trace(data=np.zeros(200))
tr.data[100] = 1
tr.stats.network = "AB"
tr.stats.station = "CDE"
print tr
tr.plot()

tr.stats.sampling_rate = 20
tr.stats.starttime = UTCDateTime()
print tr
tr.plot()
```

Trace Exercise 2

- Use *tr.filter(...)* and apply a lowpass filter with a corner frequency of 1 second.
- 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 at the end.

```
tr.filter("lowpass", freq=1.0)
tr.plot()
tr.trim(tr.stats.starttime + 3, tr.stats.endtime - 3)
tr.plot()
```

Trace Exercise 3

- 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

```
tr.data *= 500.0
tr.plot()
tr2 = tr.copy()
tr2.data += np.random.randn(len(tr2)) * tr.data.mean()
tr2.stats.station = "ABC"
tr2.plot()
```

Stream Exercise

- 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

```
from obspy.core import read
st = read()
print st
st.plot()
tr = st[0]
st.remove(tr)
print tr
print st
```

obspy.xseed Exercise

- Read the **BW.FURT..EHZ.D.2010.005** waveform example file.
- Cut out some minutes of interest.
- Read the **dataless.seed.BW_FURT** SEED file.
- Correct the trimmed waveform file with the poles and zeros from the dataless SEED file using *st.simulate()*. This will, according to the SEED convention, correct to m/s.
- (Optional) Read the file again and convert to m by adding an extra zero. Choose a sensible waterlevel.
- (Optional) Convert the SEED file to XSEED, edit some values and convert it back to SEED again. This requires some knowledge of the general SEED file structure.

```
from obspy.core import read
from obspy.xseed import Parser

st = read("./BW.FURT..EHZ.D.2010.005")
st.trim(st[0].stats.starttime, st[0].stats.starttime + 90)
p = Parser("./dataless.seed.BW_FURT")
paz = p.getPAZ(st[0].id)

st.simulate(paz_remove=paz)

st = read("./BW.FURT..EHZ.D.2010.005")
st.trim(st[0].stats.starttime, st[0].stats.starttime + 90)
paz["zeros"].append(0 + 0j)
st.simulate(paz_remove=paz)
st.plot()
```

Event Exercise

- Read the **example_catalog.xml** file.
- Plot the events.
- Print the resulting Catalog object and filter it, so it only contains events with a magnitude larger then 7.
- Now assume you did a new magnitude estimation and want to add it to one event. Create a new magnitude object, fill it with some values and append it to magnitude list of the largest event.
- Write the Catalog as a QuakeML object.

```
from obspy.core.event import readEvents
cat = readEvents("example_catalog.xml")

cat.plot()
print cat
print cat.filter("magnitude > 7")

from obspy.core.event import Magnitude
mag = Magnitude()
mag.mag = 6.0
cat[0].magnitudes.append(mag)
cat.write("new_events.xml", format="quakeml")
```