ObsPy Workshop 2012 – Python Practical

The recommended way to do the exercises is to use a text editor (for example gedit, emacs or vim) to work on the program and run the program in an IPython shell:

```
$ ipython -i
>>> run -i myfile.py
```

That way you can continue to work interactively after the program is executed. (It is best to quit and start a new IPython shell often to avoid possible confusion with "old" variables from prior program executions.)

1. Python data types, flow control and input/output
   (a) Using the modules `random`, `sys` and `math` write a script that reads mean and standard deviation of a normal distribution from the command line and writes 100 normally distributed random numbers to a file. Read those numbers back in and write their mean and standard deviation to stdout.
   (b) Use the text of this exercise and count the number of occurences for every word using `collections.defaultdict`.
2. Functions, modules, classes and exceptions
   (a) Rewrite your script from exercise 1 (a) into a class that requires the standard deviation as a variable in its constructor and which has a method that adds Gaussian noise to an input trace. Generate a sine wave, add noise to it and plot both traces using the `matplotlib` package.
   (b) Write a function that takes a list of program names as input and returns a dictionary with the programs' complete path on the current computer system as values. Programs that are not found should have the value None. For this exercise you will need the `os.environ` dictionary and `os.path` module.
3. Numpy and scipy
   (a) Define a 3 x 3 matrix, e.g. `A = np.matrix('1 2 3; 4 5 6; 7 8 9')`. Extract the 2 x 2 matrix in the lower right corner of the matrix A as a slice. Add this slice to another 2 x 2 matrix, multiply the result by a 2 x 2 matrix and insert this final result in the upper left corner of the original matrix A. Control the result by hand calculation.
   (b) Take the original matrix A from the previous exercise and replace all values greater than 1 but smaller than 5 with 0 using the numpy function `where`.

(c) Redo exercise 1 (a) using only numpy functions.

(d) If $x_i = x_1, \ldots, x_n$ are uniformly distributed numbers between a and b, then $\frac{b-a}{n} \sum_{i=0}^{n} f(x_i)$ is an approximation to the integral $\int_a^b f(x)dx$ (Monte Carlo integration). Implement this menthod using the `numpy.random` module for $f(x) = sin(x)$ and compare the result with any of the integration methods from `scipy.integrate` and the analytical result.

4. Basemap and pyproj

(a) Plot the following two stations as red triangles on a map using the `Basemap` module and calculate their distance, azimuth and back-azimuth for a spherical earth with the `pyproj` module.

```
SULZ: latitude = 47.52748, longitude = 8.11153
SALO: latitude = 45.6183,  longitude = 10.5243
```

(b) Plot the real component of the spherical harmonics of order 2 and degree 5 on a sphere using `Basemap` and the function `sph_harm` from the `scipy.special` module. Note: Spherical harmonics are computed in colatitude and basemap uses latitude.

5. Quakepy

(a) Read the EQ catalog from Geonet web service (QuakeML). If there is no internet access, use the provided XML file. Print the number of events. Filter with area polygon, magnitude, depth. Print the number of events after filtering. Export the catalog to file in ZMAP-ASCII format.

(b) Read the EQ catalog from the provided gnuzipped GSE2.0 bulletin file. Create a CompactCatalog object and sort by magnitude in descending order. Export the catalog to a file in ZMAP-ASCII format.

(c) Read the EQ catalog from the provided gnuzipped QuakeML file. Fit the Gutenberg-Richter law to the data and create a cumulative frequency-magnitude distribution and plot this as an EPS file using matplotlib.