Designing Python Libraries for Rock Physics

Rosa Filgueira

Malcolm Atkinson

Joshua Einserberg

Python and Scientific Communities

- Python is a high-level programming language, interpreted with capabilities for object-oriented programming
- A language very easy to learn for non-programmers
- Well structured: Easy to read and understand code
- Well-known language with a big community
- Free and open source

Python and Scientific Communities

- With very rich scientific computing libraries:
  - Obspy: Simplifies the usage of Python programming for *seismologists*.
  - SymPy: Python library for *symbolic mathematics*
  - Geopy: Python *Geocoding* Toolbox
  - scikits.learn: A set of python modules for machine *learning and data mining*
  - Numpy: Array processing for numbers, strings, records, and objects
  - Scipy: Scientific Library for Python
- Most Popular libraries: http://www.s-anand.net/blog/the-most-popular-scientific-python-modules/

Overview of Numpy

- NumPy is the fundamental package for scientific computing with Python. It contains among other things:
  - a powerful N-dimensional array object
  - sophisticated (broadcasting) functions
  - tools for integrating C/C++ and Fortran code
  - useful linear algebra, Fourier transform, and random number capabilities

Overview of SciPy

- SciPy is an open source library of algorithms and mathematical tools for Python.
- SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy

- Available subpackages:
  - **constants**: physical constants and conversion factors (since version 0.7.0[4])
  - **cluster**: hierarchical clustering, vector quantization, K-means
  - **fftpack**: Discrete Fourier Transform algorithms
  - **integrate**: numerical integration routines
  - **interpolate**: interpolation tools
  - **io**: data input and output
  - **lib**: Python wrappers to external libraries
  - **linalg**: linear algebra routines
  - **misc**: miscellaneous utilities (e.g. image reading/writing)
  - **ndimage**: various functions for multi-dimensional image processing
  - **optimize**: optimization algorithms including linear programming
  - **signal**: signal processing tools
  - **sparse**: sparse matrix and related algorithms
  - **spatial**: KD-trees, nearest neighbors, distance functions
  - **special**: special functions
  - **stats**: statistical functions
  - **weave**: tool for writing C/C++ code as Python multiline strings

Overview of Obspy

- It is an open-source project dedicated to provide a Python framework for processing **seismological data**.

- It provides parsers for common file formats and seismological signal processing routines which allow the manipulation of seismological time series, allowing the use of powerful numerical array-programming **modules like NumPy and SciPy**

Obspy

- General packages:
  - obspy.core - ObsPy core package, glues the single obspy packages together
  - obspy.imaging - Imaging spectograms, beachballs and waveforms
  - obspy.signal - Filters, triggers, instrument correction, rotation, array analysis, beamforming
  - obspy.xseed - Converter for Dataless SEED, XML-SEED and SEED RESP files
- Waveform import/export plug-ins:
  - obspy.gse2 - GSE2 and GSE1 read and write support
  - obspy.sac - SAC read and write support
  - obspy.mseed - MiniSEED read and write support
  - obspy.wav - WAV (audio) read and write support
  - obspy.sh - Q and ASC read and write support (file formats of SeismicHandler)
  - obspy.seisan - SEISAN read support
  - obspy.segy - SEGY read and write support
- Database or Web service access clients:
  - obspy.seishub - SeisHub database client
  - obspy.arclink - ArcLink/WebDC request client
  - obspy.fissures - IRIS DMC DHI/Fissures request client
  - obspy.iris - IRIS DMC Core Web services request client
  - obspy.neries - NERIES Seismic Data Portal request client

Python and Scientific Communities

- Although there are many software tools available for interactive data analysis and development, there is not any library designed for work with Rock Physics data.

A Python Toolbox for Rock Physics

- Our objective is to help scientists who write methods, or components to be used in methods, that will then be used in (and with data downloaded from) the EFFORT gateway.

- We propose a Python toolbox called "EffortPy" to facilitate **rapid application development for rock physicists**.

EffortPy aims

- Provide a full repertoire of commonly required actions to facilitate rock physicists to write their methods, analyses and visualisations in Python.

- Wrap operations to enable the EFFORT gateway to maintain the integrity of its data, and to implement the rules developed for governance of users' actions.

EFFORT gateway and EffortPy

- This library will help to run the user's models in the gateway **automatically**.
- What is happening right now every time that we have a new model:
  - Modification of input parameters
  - Modification of the output parameters
  - Modify how the data is selected, read and written
  - Modify how the results are displayed and stored.

  - Once all those modifications are made, the model is ready to execute in the gateway as many times as the users want.

EFFORT gateway and EffortPy

- In EFFORT project we want to encourage the **model sharing** through the EFFORT gateway
- EffortPy will facilitate us to achieve this objective:
  - Giving a description of the user's models
  - Giving an explanation for executing the models
- A lot of models have similar code inside them for doing the same tasks:
  - Plot the data in an specific format :
    - plot of depth vs time
    - plot of magnitude vs .time
    - plot of frequency vs. magnitude
- EffortPy will have a big set of "most-common" functions that the users can call in their codes, saving them time and effort.
- There will be two variants of the EffortPy with many of the functions completely identical

EffortPy versions: Gateway

- This variant will be used when programs are being run under the gateway:
  - Validate authorisation
  - Generate a record metadata
  - Manipulate the catalogues
  - Perform a specified function.
- EffortPy will enforce the governance rules and they make the metadata complete and consistent.

EffortPy versions: Developers

- The developers' version will be used when programs are used on data that have been downloaded onto a researcher's own computer.
  - Allow developers of algorithms to explore, tune and experiment, in order to develop their models, analyses and visualisations.

EffortPy Gateway and Developers versions

- The effect of all of the functions used and the representations of internal data structures should be identical in **both variants** from the viewpoint of the researchers.
- This is critical to ensure that code developed and tested on a personal machine still executes correctly when uploaded.

EffortPy format

- We take the library style from the ObsPy library.
- EffortPy should not attempt to replace the functions provided by other python libraries, such as NumPy and SciPy. It should be complementary of them.

EffortPy Sketch

- Experiment-Data Management:
- Sharing Management
- Metadata Management
- Shared Code Management
- Continuous Stream Analysis
- Continuous Stream Generation
- Event Stream Analysis
- Result Visualisation Utilities