Python for Scientific Computing 3

Konrad Hinsen
Centre de Biophysique Moléculaire (CNRS)
Orléans, France

1 More information

The complete documentation of Numerical Python (array operations etc.) can be found at `http://www.pfdubois.com/numpy/html2/numpy.html`.
A very complete introduction to Tk programming in Python can be found at `http://www.pythonware.com/library/tkinter/introduction/`. Also look at the Tk chapter of the Python documentation at `http://www.python.org/doc/current/lib/tkinte`
ScientificPython contains two useful Tk widgets, one for data plots (2D) and one for visualizing 3D wireframe models. The documentation is at `http://dirac.cnrs-orleans.fr/man`

2 Exercises

2.1 Tensorial product

Write a function that takes two 1-d array arguments a and b and returns their tensorial product, a 2-d array whose elements are defined by

$$t_{ij} = a_i \times b_j$$

2.2 Positive elements

Write a function that takes a 1-d array argument and returns another 1-d array which contains only the positive elements of the argument.

2.3 Numerical derivative

Write a function that takes a 2-d $N \times 2$ array f that represents a function $y(x)$ evaluated on a grid. $f[:, 0]$ contains the x-values in increasing order (but not necessarily equidistant). $f[:, 1]$ contains the y-value. The return value is an array representing the numerical derivative $y'(x)$ using the same arrangement.

2.4 GUI

Write a GUI version of the "Text file processing 3" exercise (first part). Let the user choose an input file and provide buttons for doing the calculation (and showing the results in a separate window) and for quitting.

2.5 Molecular Dynamics

[This is a longer exercise, not meant to be completed during the course.]

Write a Molecular Dynamics simulation program for atomic systems with Lennard-Jones interactions. Store the resulting trajectories in netCDF files.

Start with a very simple program: only one kind of atom, initial positions on a lattice, all simulation parameters are hard-coded into the program. Consider carefully how to represent the data. For example, should the atom positions be stored in lists, arrays, or vectors?

Don't try to optimize anything at this stage. For example, calculate all N^2 interactions. Make sure this program works properly before continuing.

Then add features such as random initial configurations or support for several kinds of atoms. After each modification, make sure the program works properly. Also, reconsider your choice of data representation. Don't hesitate to rewrite your code if it seems appropriate.

Think about analysis programs that read the trajectory files. How do they affect the design of the trajectory layout? Try to put all information that an analysis program might need into a single netCDF file.