Python in High-Performance Computing

Are M. Bruaset, Xing Cai,
Hans P. Langtangen, Kent-Andre Mardal,
Halvard Moe, Ola Skavhaug, Åsmund Ødegård

Abstract

The virtue of Python is flexibility, making it an ideal tool for the scientist who wants to experiment with different models, do rapid prototyping, and even do large scale simulations. The syntax and functionality seem much like Matlab, but Python is a full-fledged advanced programming language with excellent support for user-defined data types, operator overloading, object orientation, and generic ("template") programming. Moreover, it is convenient to migrate Python code to Fortran, C, or C++, or equip legacy libraries in those languages with modern and user-friendly interfaces. Python also has cross-platform support for most operating system and file management tasks. This tutorial presents concepts and software that will give the scientist adaptable, powerful, and efficient tools.

Detailed description

Goals

This tutorial will give an example-oriented introduction to using Python as a programming platform for high-performance computing. The participants will learn how their daily work can benefit from the flexibility of the Python language, and from several powerful add-on tools. Also those already familiar with the Python programming environment can expect to improve their skills and pick up useful techniques and procedures.

Targeted audience

We address both researchers doing a lot of "Matlab-style" numerical computing and plotting, and researchers doing scientific application development or adaption in low-level languages like Fortran, C, or C++. Interactive computing and plotting can easily make use of applications and libraries written in compiled languages, while program development in Fortran, C, or C++ can benefit from a mixed language approach where Python is used for data management tasks while the most time-critical number crunching takes place in compiled code.

No previous knowledge about Python is required, but in such a short course it will be an advantage to have some familiarity with basic Python syntax. Newcomers to Python should, however, be able to get an overview of what the language can be used for and how the technical solutions to various problems look like.

Course contents

The tutorial starts with a crash course on basic Python syntax and programming. Then we cover simple ("Matlab-style") numerical computing with matrices and vectors using the Numerical Python package. Combination of Python and Fortran, C, or C++ for increased performance is also discussed. In particular we show how conveniently Python integrates with Fortran using the F2PY tool. An overview of plotting and visualization packages callable from Python is also given. Furthermore, we discuss high-level parallel computing using the PyPar module. Computational efficiency and techniques to boost the performance will be presented throughout the short course.

The teachers at PARA'06 will be Professors Xing Cai, Hans Petter Langtangen, and Kent-Andre Mardal from Simula Research Laboratory (Oslo, Norway) and the University of Oslo. The course material will be made available as handouts. Parts of the material are also covered by the book H. P. Langtangen: *Python Scripting for Computational Science*, Springer, 2nd edition, 2006.

After the lectures, participants may try out some exercises on their own laptop with guidance from the course instructors. Necessary software will be provided, both at `http://www.simula.no/~hpl/PyTutorialPARA06/` and on a CD.

Why is this topic relevant to PARA'06 attendees?

Python has become a popular language over recent years, and is already in widespread use within the scientific community. Still, we think that the capabilities of Python for high-performance computing are under-utilized, especially as a tool for driving numerical simulations. The language is ideal for gluing together existing codes and programs in various ways. This topic is relevant to PARA'06 attendees who are running numerical simulations as part of their scientific work.