Editorial

Gael Varoquaux (gael.varoquaux@normalesup.org) – *INRIA, Saclay* France
Stéfan van der Walt (stefan@sun.ac.za) – *University of Stellenbosch, Stellenbosch* South Africa
Jarrod Millman (millman@berkeley.edu) – *UC Berkeley, Berkeley, CA* USA

SciPy 2009 marks our eighth annual Python in Science conference and the second edition of the conference proceedings. The conference and these proceedings highlight the ongoing focus of the community on providing practical software tools, created to address real scientific problems.

As in previous years, topics at the conference ranged from the presentation of tools and techniques for scientific work with the Python language, to reports on scientific achievement using Python. Interestingly, several people noticed that something important happened in the Scientific Python world during the last year: we are no longer constantly comparing our software with commercial packages, nor justifying the need for Python in Science. Python has now reached the level of adoption where this sort of justification is no longer necessary. The exact moment when this shift in focus occurred is difficult to identify, but that it happened was apparent during the conference.

Recurring scientific themes

This year the conference spanned two days, and each day commenced with a keynote address. The first keynote was delivered by Peter Norvig, the Director of Research at Google; the second by Jonathan Guyer, a materials scientist in the Thermodynamics and Kinetics Group at the National Institute of Standards and Technology (NIST).

Peter Norvig's talk was titled "What to demand from a Scientific Computing Language—even if you don't care about computing or languages", where he discussed a number of desired characteristics in a scientific computing environment. Such a platform should have the ability to share code and data with other researchers easily, provide extremely fast computations and state-of-the-art algorithms to researchers in the field, and be as easy as possible to use in a time-efficient manner. He also stressed the importance of having code that read like the mathematical ideas it expressed.

Jonathan Guyer's keynote centred around "Modeling of Materials with Python". He expanded on several of the above-mentioned characteristics as he discussed the development of FiPy, a framework for solving partial differential equations based on a finite volume approach. Jonathan explained how FiPy was created to provide the most advanced numerical techniques to scientists, so that they could focus on the scientific questions at hand, while having a standard platform for sharing codes with colleagues. Importantly, FiPy has become a critical tool in Jonathan's research group and has been adopted by many of their colleagues for both research as well as teaching.

Both keynote addresses served to outline prominent themes that were repeated throughout the conference, as witnessed by the proceedings. These themes include: the need for software tools that allow scientists to focus on their research, while taking advantage of best-of-class algorithms and utilizing the full power of their computational resources; the need for a high-level computing environment with an easy—to-write and read syntax; the usefulness of high-quality software tools for teaching and education; and the importance of sharing code and data in scientific research.

The first several articles address high-level approaches aimed at improving the performance of numerical code written in Python. While making better use of increased computation resources, such as parallel processors or graphical processing units, many of these approaches also focus on reducing code complexity and verbosity. Again, simpler software allows scientists to focus on the details of their computations, rather than on administrating their computing resources.

The remaining articles focus on work done to solve problems in specific research domains, ranging from numerical methods to biology and astronomy. For the last several years, using Python to wrap existing libraries has been a popular way to provide a scripting frontend to computational code written primarily in a more low-level language like C or Fortran. However, as these proceedings show, Python is increasingly used as the primary language for large scientific applications. Python and its stack of scientific tools appears to be well suited for application areas ranging from database applications to user interfaces and numerical computation.

Review and selection process

This year we received 30 abstracts from five different countries. The submissions covered a number of research fields, including bioinformatics, computer vision, nanomaterials, neutron scattering, neuroscience, applied mathematics, astronomy, and X-ray fluorescence. Moreover, the articles discussed involve a number of computational tools: these include statistical modeling, data mining, visualization, performance optimization, parallel computing, code wrapping, instrument control, time series analysis, geographic information science, spatial data analysis, adaptive interpolation, spectral analysis, symbolic mathematics, finite element, and virtual reality. Several abstracts also addressed the role of scientific Python in teaching and education.

Each abstract was reviewed by both the program chairs, as well as two members of the program committee (PC). The PC consisted of 11 members from five countries, and represented both industry and academia. Abstracts were evaluated according to the following criteria:

- Relevance of the contribution, with regard to the topics and goals of the conference.
- Scientific or technical quality of the work presented.
- Originality and soundness.

We accepted 23 (76%) submission for oral presentation at the conference. At the closure of the conference, we invited the presenters to submit their work for publication in the conference proceedings. These submissions were reviewed by 11 proceeding reviewers from seven countries, according to the following criteria:

- Does the paper describe a well-formulated scientific or technical achievement?
- Is the content of the paper accessible to a computational scientist with no specific knowledge in the given field?
- Are the technical and scientific decisions well-motivated?
- Does the paper reference scientific sources and material used?
- Are the code examples (if any) sound, clear, and well-written?
- Is the paper fit for publication in the SciPy proceedings? Improvements may be suggested, with or without a second review.

From the 30 original abstracts, 12(40%) have been accepted for publication in these proceedings.

Prior to commencing the conference, we had two days of tutorials with both an introductory and advanced track. In addition to publishing a selection of the presented work, we also selected one of this year's tutorial presentations for publication.

The proceedings conclude with a short progress report on the two-year long NumPy and SciPy documentation project.

The SciPy Conference has been supported since its inception by the Center for Advanced Computing Research (CACR) at Caltech and Enthought Inc. In addition, we were delighted to receive funding this year from the Python Software Foundation to cover the travel, registration, and accommodation expenses of 10 students. Finally, we are very grateful to Leah Jones of Enthought and Julie Ponce of the CACR for their invaluable help in organizing the conference.