

Overview

Continuum Analytics Europe GmbH is a subsidiary of Continuum Analytics, Inc., based in Austin, Texas. Continuum focuses on Python for Data Exploration & Visualization. Around Python, Continuum is active mainly in four areas:

- **Open Source projects:** Continuum engages in a number of Open Source projects, ranging from a free Python distribution (Anaconda) to big data array solutions (Blaze)
- **products:** Continuum offers a number of performance libraries (e.g. NumbaPro for just-in-time CPU/GPU compiling) and packaged solutions (like Anaconda Accelerate for high performance analytics and Wakari for Web-based data analytics)
- **consulting & development:** we provide a whole range of consulting & development services around Python-based initiatives; our experts have domain expertise, among others, in industries like finance/banking, bio technology, defence/military, engineering
- **training:** Continuum has a large pool of Python talent and provides virtual as well as on-site trainings with topics ranging from general Python programming to finance and scientific applications

Python for Analytics

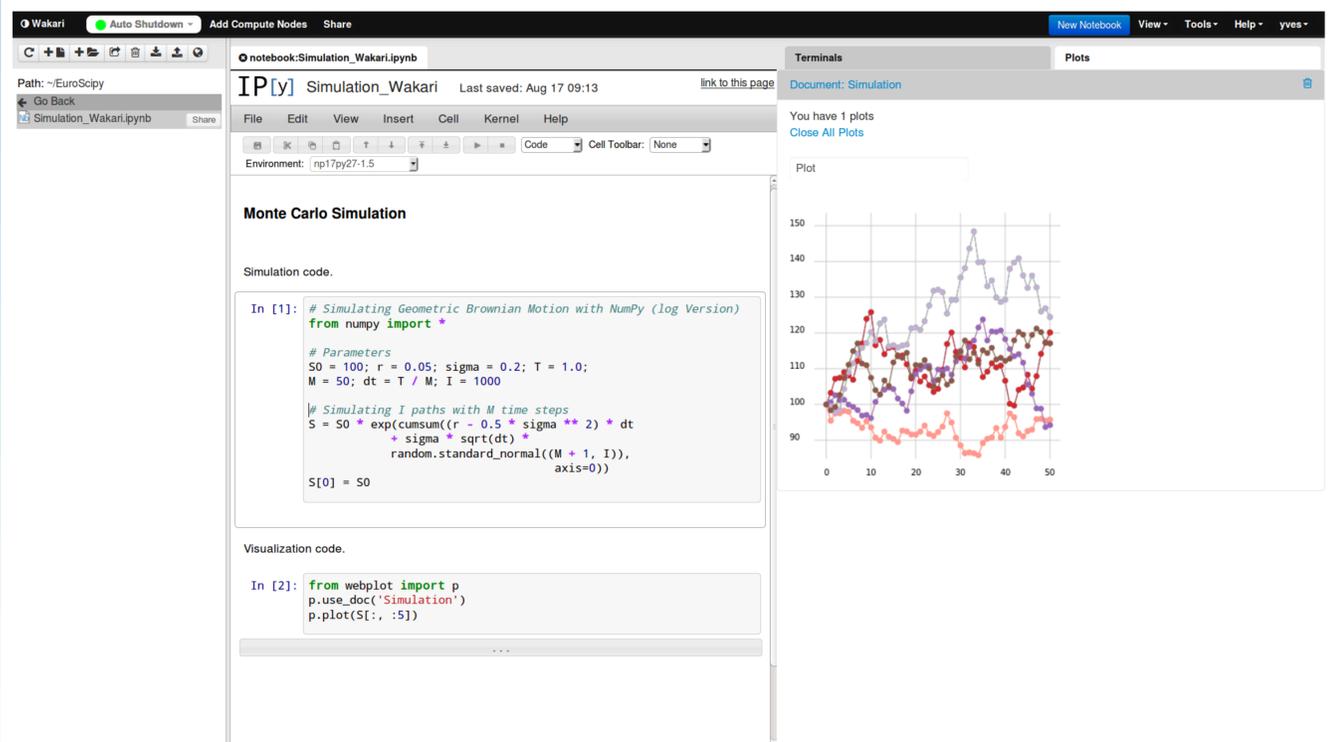
Today's analytics tasks demand for high performing implementations of sometimes complex algorithms. A fundamental analytics stack for Python should include at least the following:

- IPython: interactive development
- NumPy: array operations
- SciPy: scientific computing
- matplotlib: 2d and 3d plotting
- pandas: time series and panel data
- Cython: C extensions for Python
- Numba: just-in-time compiling
- PyTables: hierarchical database

Continuum products provide two easy routes to get a consistent Python-based data analytics infrastructure up and running:

- **Anaconda:** a free Python distribution providing easy and flexible install procedures on your local machine or your server
- **Wakari:** this is a browser-based data analytics & visualization environment (with free accounts) which offers, among others, Python shells, IPython Notebooks, complete Python environments and functions to collaborate and share

Wakari



The screenshot shows a Wakari IPython Notebook interface. The notebook title is "Simulation_Wakari.ipynb". The code in the first cell is for a Monte Carlo simulation of Geometric Brownian Motion. The code includes parameter definitions (S0=100, r=0.05, sigma=0.2, T=1.0, M=50, dt=T/M, I=1000) and a loop to simulate I paths with M time steps. The second cell shows visualization code using webplot to plot the simulated paths. The plot on the right shows multiple colored lines representing different simulated stock price paths over time (0 to 50).

Work Flow with Wakari

Suppose you have to implement a simulation algorithm for the evolution of a stock price over time. You also want to visualize some simulated stock price paths. Afterwards, you want to share your code and results with a colleague at your company.

An optimized work flow with Wakari would look like follows:

1. **login:** you log in to your Wakari account
2. **Python:** you have all libraries and tools available for interactive, collaborative data analytics
3. **IPython:** you open a new Notebook which will contain your code and your results
4. **code:** you interactively write your code, test, re-write and optimize it
5. **documentation:** IPython Notebook allows you to easily include markup-based documentation
6. **results:** you run your code to generate the numerical and graphical results
7. **publish:** you can share your complete Notebook via a simple click or convert it to a PDF or a HTML5 presentation

Wakari is a tool that minimizes the time to analytics insights through an optimized work flow. Precious time of data analysts is saved and decisions can be made more quickly.

Distributed Computing

With Wakari and IPython you can easily implement parallel execution of code. The solution is fully scalable and you can flexibly add single or multiple computing nodes to your environment. With Blaze you will then be able to have disk-based, distributed arrays over a whole cluster.

Performance Issues

For example, nested loops like the following are quite slow in pure Python:

```
def f_py(n):
    result = 0.0
    for i in range(n):
        for j in range(n * i):
            result += sin(pi / 2)
    return int(result)
```

One approach to speed-up code is to use highly optimized libraries, such as NumPy. In this case, memory issues would arise for large n . Another one is to use just-in-time compiling with Numba.

```
import numba
f_nb = numba.autojit(f_py)
```

Just-in-time compiling leads to a speed-up of 1,800x and preserves minimal memory usage:

```
n = 1000

%time result = f_py(n)
Wall time: 19min 21s

%time result = f_nb(n)
Wall time: 629 ms
```

Using NumbaPro, pure Python can also be compiled and optimized for parallel execution on multi-core CPUs as well as on GPUs.

Efficient I/O

To read and to write large sets of data is an important analytics task. Python provides with pandas and PyTables two libraries that are quite efficient when it comes to I/O operations as well as in-memory and out-of-memory analytics, respectively. IOPro is another solution that helps to improve the performance of typical I/O tasks.

Contact Information

If you have questions regarding our services or our products, please contact Dr. Yves J. Hilpisch under yves@continuum.io. You can also visit our Web site www.continuum.io for more information.