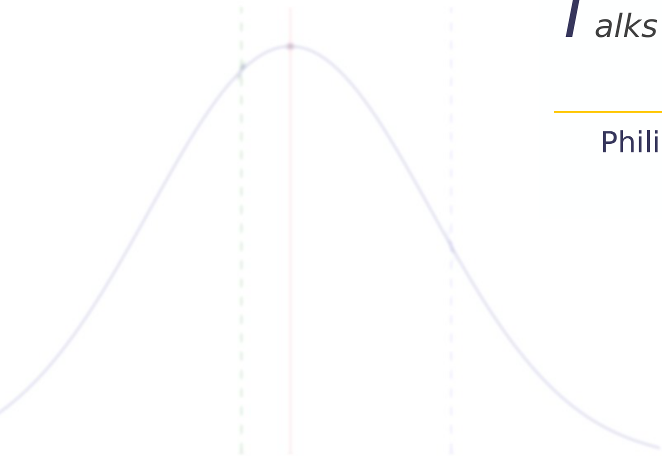


Talks

Philipp Eisenhauer

Material available on

Visit us!

UNIVERSITÄT BONN

High-performance computing using Python

Philipp Eisenhauer

I draw on the material presented in:

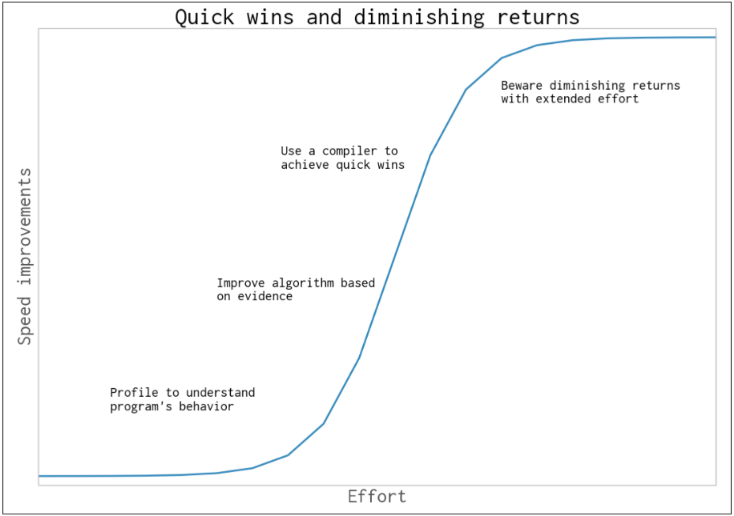
- Gorelick, M., & Ozsvald, I. (2014). *High performance Python*.
- Lanaro, G. (2017). *Python high performance*.

Profiling

- *Premature optimization is the root of all evil.*
- focus on readability
- set up development environment
- flesh out testing harness
- tackle performance bottlenecks

Points of attack

- pure Python
- high-performance libraries, SciPy Stack
- compilation to faster language
- parallel processing
- distributed computing

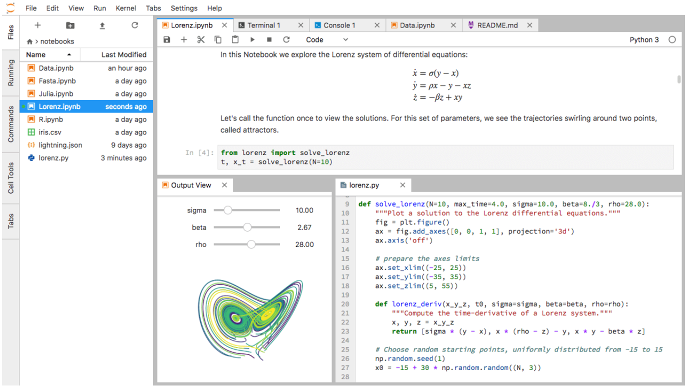

Quick wins and diminishing returns
Beware diminishing returns
with extended effort
Use a compiler to
achieve quick wins
Improve algorithm based
on evidence
Profile to understand
program's behavior
Speed improvements
Effort

High-performance libraries

Vectorization

- Vectorization is when a CPU is provided with multiple pieces of data at a time and is able to operate on all of them at once. This sort of CPU instruction is known as SIMD (Single Instruction, Multiple Data).

Practical illustration

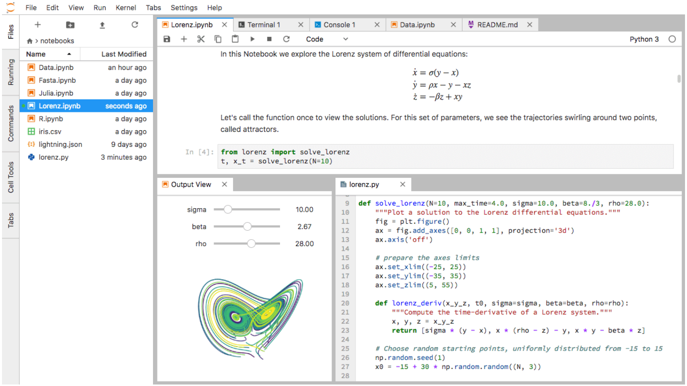


Compilation to faster language

Alternatives

- ahead-of-time, e.g. f2py, Cython
- just-in-time, e.g. numba

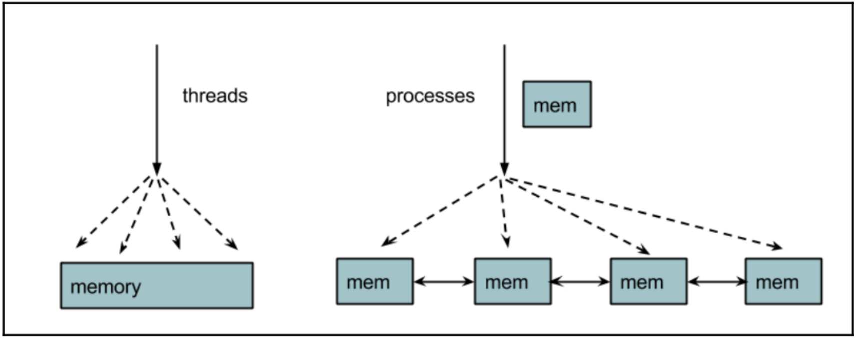
Practical illustration



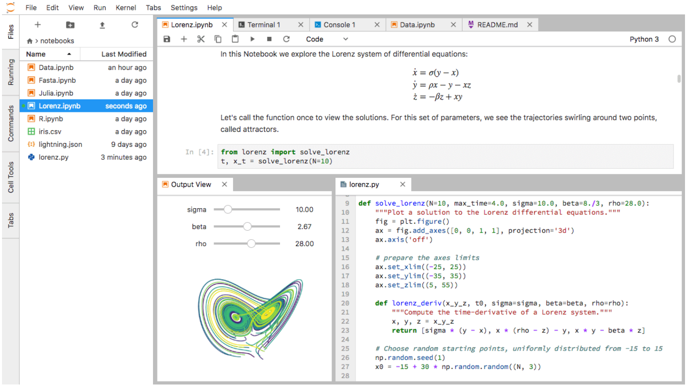
Parallel processing

Memory access

- shared memory
- distributed memory


threads
processes
mem
memory
mem
mem
mem
mem

Practical illustration



Resources

Textbooks

- Lanaro, G. (2017). *Python high performance*.
- Gorelick, M., & Ozsvald, I. (2014). *High performance Python*.

Appendix

References

Gorelick, M., & Ozsvald, I. (2014). *High performance Python*.
Lanaro, G. (2017). *Python high performance*.