

# Performance Comparison

→ipython

js?

- julia / R / python / c++ / perl / fortran / rust
- performance : runtime? memory? devtime? readability? storage? reuse?
- problems for comp: skill level, packages, compile flags, machine dependant OS
- language specific packages exist
- related: profiling
- when do you need good performance?
  - huge datasets
  - long running calc.
  - interactivity
- not
  - small datasets
  - one-off calculations
- there are theoretical limits: optimize algo before implement.
- related techniques: caching memoization
- interpreted vs compiled (jit?)

## TODOs:

- Slides zwischen drin (Andreas)
- Alle Notebooks nochmal laufen lassen (Markus)
- Plot mit Zeiten von Sprachen (Markus)
- Bullet Points für letzte Folie (Markus)
- Probe halten (remote control?) → Mo 10:00 Uhr

1. Hier ist das Problem (DGL)
2. Hier ist die/eine Lösung in python Simon
3. Python ist schön lesbar
4. Aber nur mit Schleife  $\Rightarrow$  das könnte langsam sein
5. Wirkt nicht langsam also: messen wir es
6. Wirkt immer noch schnell 1ms
7. Probleme für Vergleichbarkeit erwähnen  
 $\rightarrow$  Maschine / OS / Last

(py)

(1m)

8. Julia behauptet schneller zu sein  $\rightarrow$  stimmt das?
  9. Übersetzen von python klappt problemlos
  10. Und tatsächlich ist es deutlich schneller >100x
  11. Woher kommt das?  $\rightarrow$  interpreted vs compiled
  12. Kurz Vor- und Nachteile davon
- 
13. Hier hätten wir aufhören können aber was ist mit anderen "beliebten" Sprachen (am CTF) und mit anderen schnellen oder interessanten Spr.  
 $\rightarrow$  Hackathon

14. R Notebook → wir erwarten noch schlimmeres ®
  15. Code auch verständlich
  16. Ähnlich langsam wie python → keine julia Konkurrenz
- 
17. Super hässlicher Code (rot)
  18. Aber auch nur halbwissen
  19. z.B. auch kein Plot → auch zukünftig nicht
  20. Ähnlich langsam wie python und R

- 
21. Zwischenfazit: Julia gewinnt deutlich
  22. Aber was ist mit anderen compilierten Sprachen?

- 
23. Unsere C++ Implementierung → umständlich !  
kompliziert  
pointer (C++)
  24. Trotzdem langsamer als julia
  25. Optimierung: Code Faktor 2 schneller  
↳ manches weggelassen
  26. Raus aus Notebook rein in CLI → compiler Optimierung
  27. Immer trade-offs compile-time, Speicherverbrauch, Portabilität, Genauigkeit
  28. Jetzt doch schneller als Julia

29. Fortran kurz zeigen (implicit none erklären)  
30. Rust kurz zeigen
- 

31. Auch hier hätten wir aufhören können aber zurück zu Julia  
32. Optimiertes Julia Notebook mit C++ Tricks  
33. Julia kann doch mithalten  
34. Vermutlich nah an theoretischer Grenze
- 

35. Ein Graph zum Abschluss  
36. Bullet Points mit wichtigsten Messages
  - It depends language, skill, machine, flags, ...
  - Use Julia