Gotta go fast

Overview

- High level, interactive like MatLab, Python (with Jupyter)
- User friendliness we expect from Python
  - Dynamically typed, sometimes
- (Often) as fast as C/FORTRAN
- Free / Open source
- Over 5,000 community packages and growing (including familiar faces)

Features

- Just-In-Time Compilation
  - Like Numba for python, but automatic and hassle-free
- Column-major array storage
- Parallel support
  - Distributed memory (including existing MPI libraries), shared memory, SIMD, and GPU support
- Easily call C, FORTRAN, and Python code from within Julia
- NOT object oriented >:)

Lots of familiar conveniences

- HDF5.jl - Essentially h5py. Just as user friendly
- Pandas.jl - Wrapper of Python Pandas library
  - DataFrames.jl - Essentially the same but pure Julia
- PyPlot.jl - Julia Matplotlib wrapper

Examples

```
"""
This is okay!
Returns Hilbert matrix of size n
"""
function Hilbert_dynamic( n )
  matrix = zeros( n, n )

  for i in 1:n
    for j in 1:n
      matrix[i,j] = 1.0 / ( i + j - 1.0 )
    end
  end

  return matrix
end
```

zeros(n,n) like np.zeros((n,n))

Indentation doesn't matter!

1 based indexing

Docstrings

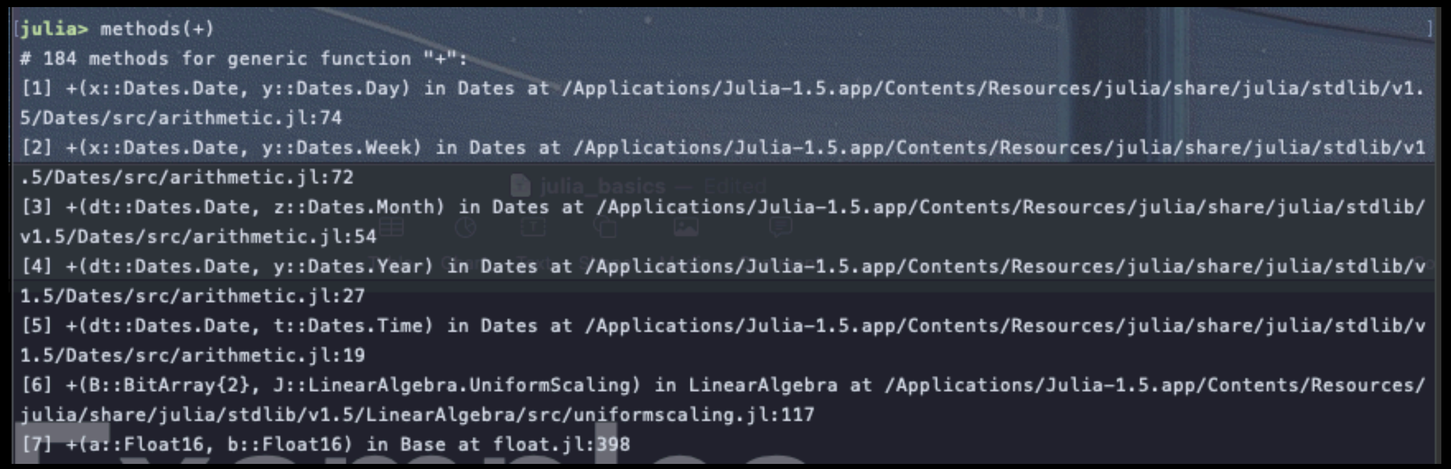
```
"""
This is better!
Returns Hilbert matrix of size n
"""
function Hilbert_static( n::Int64 )
  matrix::Array{Float64, 1} # or Vector{Float64}

  for i in 1:n
    for j in 1:n
      matrix[i,j] = 1.0 / (i + j - 1.0)
    end
  end

  return matrix
end
```

FunctionName(var::Type)

declare similar to FORTRAN

Multiple Dispatch

- Function and operator overloading, but not
-



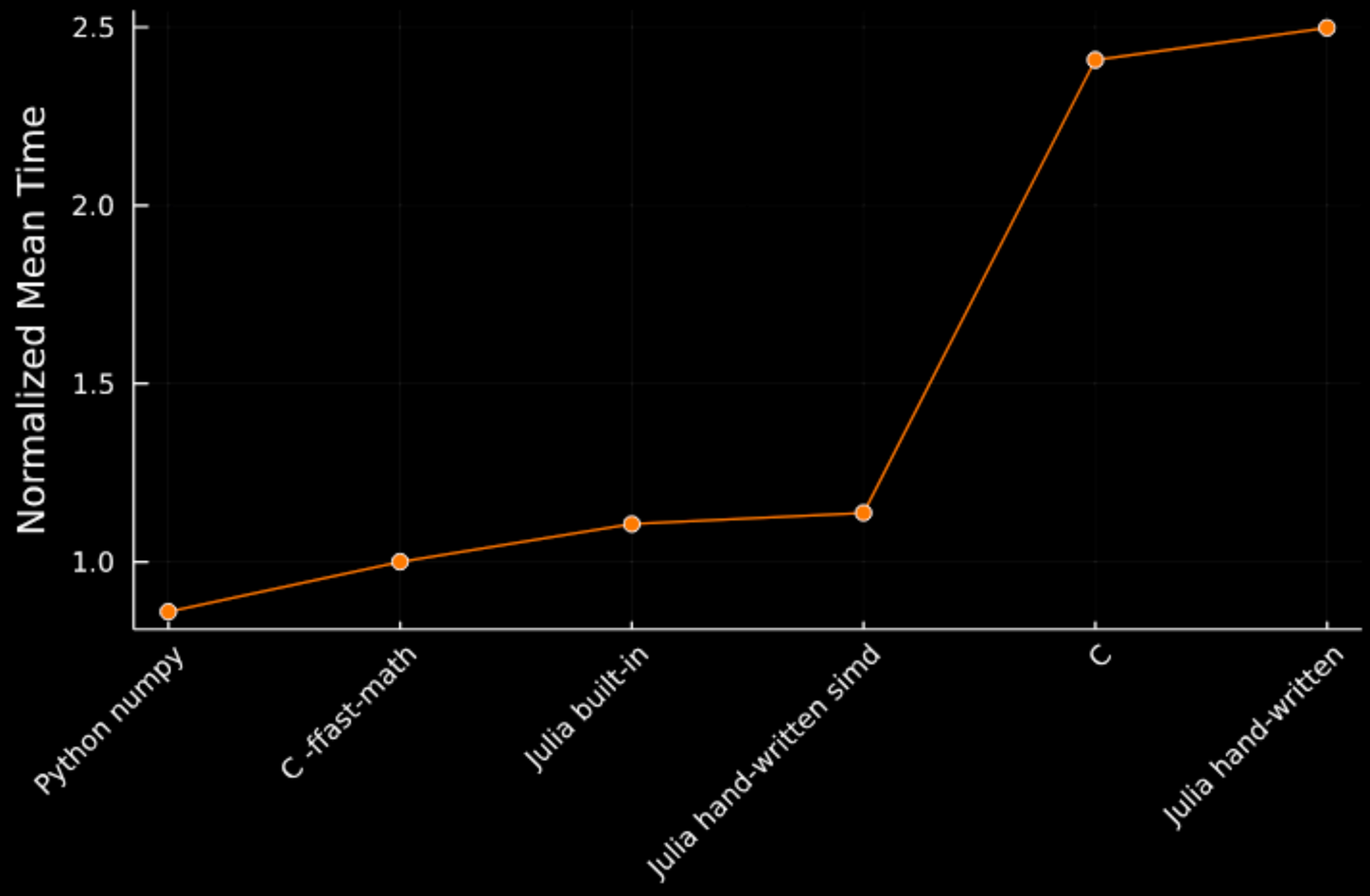
Free to define new methods for existing functions

Using and Running Julia

- Download Julia (might be in HomeBrew? I just get from website), alias executable, possibly add to path
- Like running “python” at command line, can run Julia to open REPL
- “julia code.jl” executes code
- Supported in Jupyter notebooks by default!
- Pluto.jl
  - Like Jupyter, but real-time updates cell dependencies

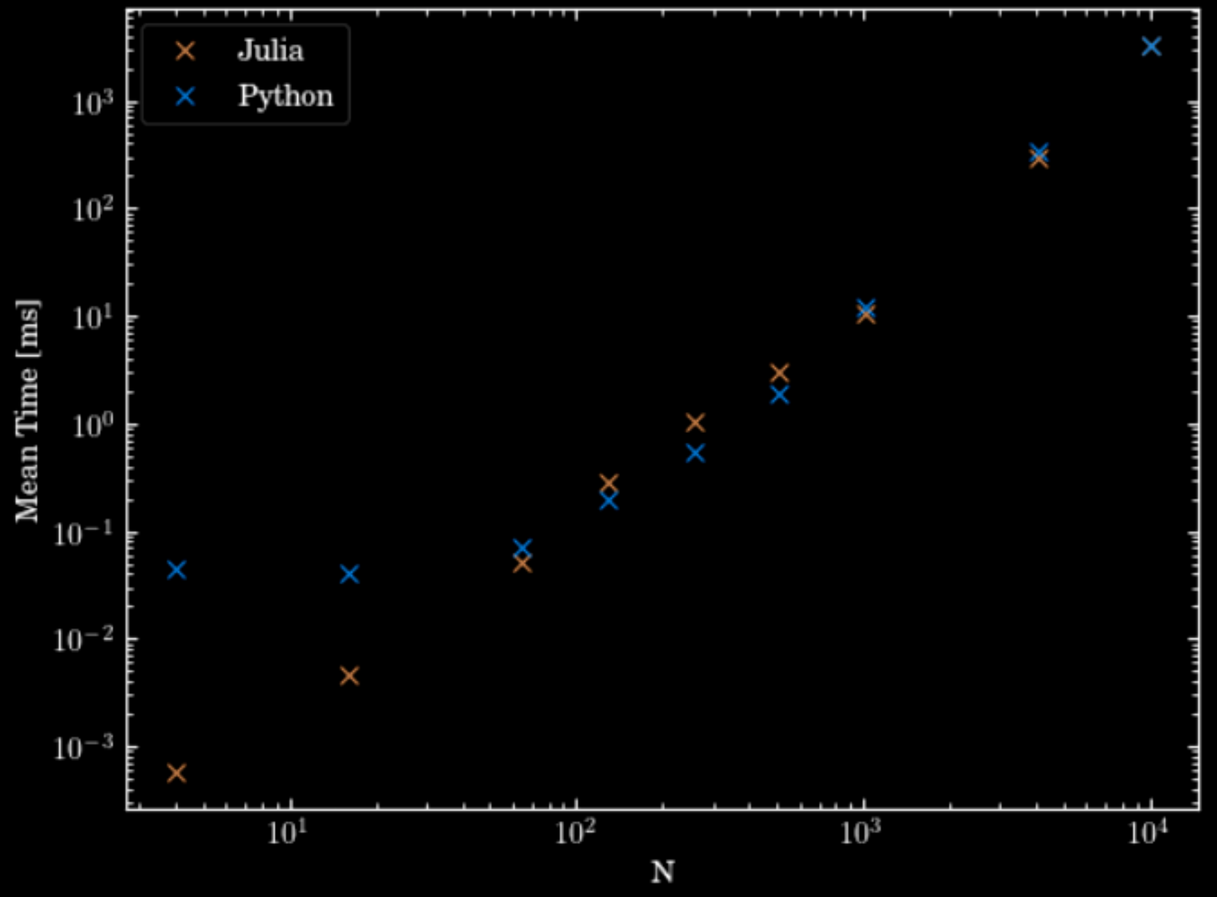
Some Benchmarks

Sum 1e7 rands

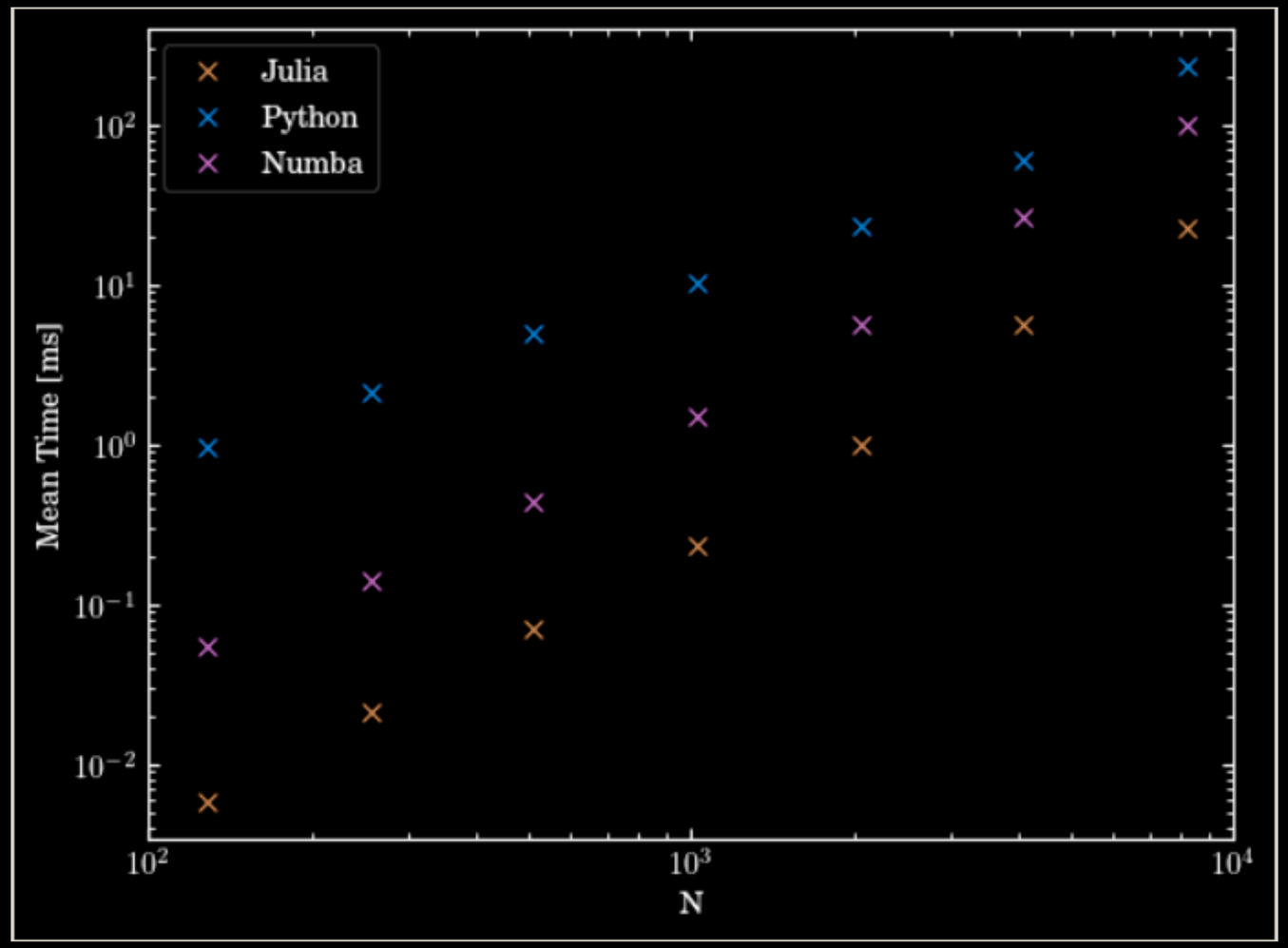


Native Python at ~150

Linear solvers



1D Advection



Cons?

- Fewer mature libraries / ports
  - Exs: built in (not PyPlot) plotting still lacking some controls, no yt port
- Short / “simple” scripts may be slower than python eqv. from compile
  - “np.loadtxt and plot script” equivalent will just be slower
- Harder to share code — for now!
  - Used to be that people were hesitant to switch Perl -> Python!

Pros

- It’s just fast, for free
- Even writing “bad” Julia code, you can compete with “good” Python/C
  - No need to fuss with Numba - pain in the ***
  - No need to spend time planning how to avoid Python’s optimization issues
- Growing quickly
- Just as “easy” to write as Python, some syntactical obstacles
- Integrates into analysis workflow (Jupyter support)