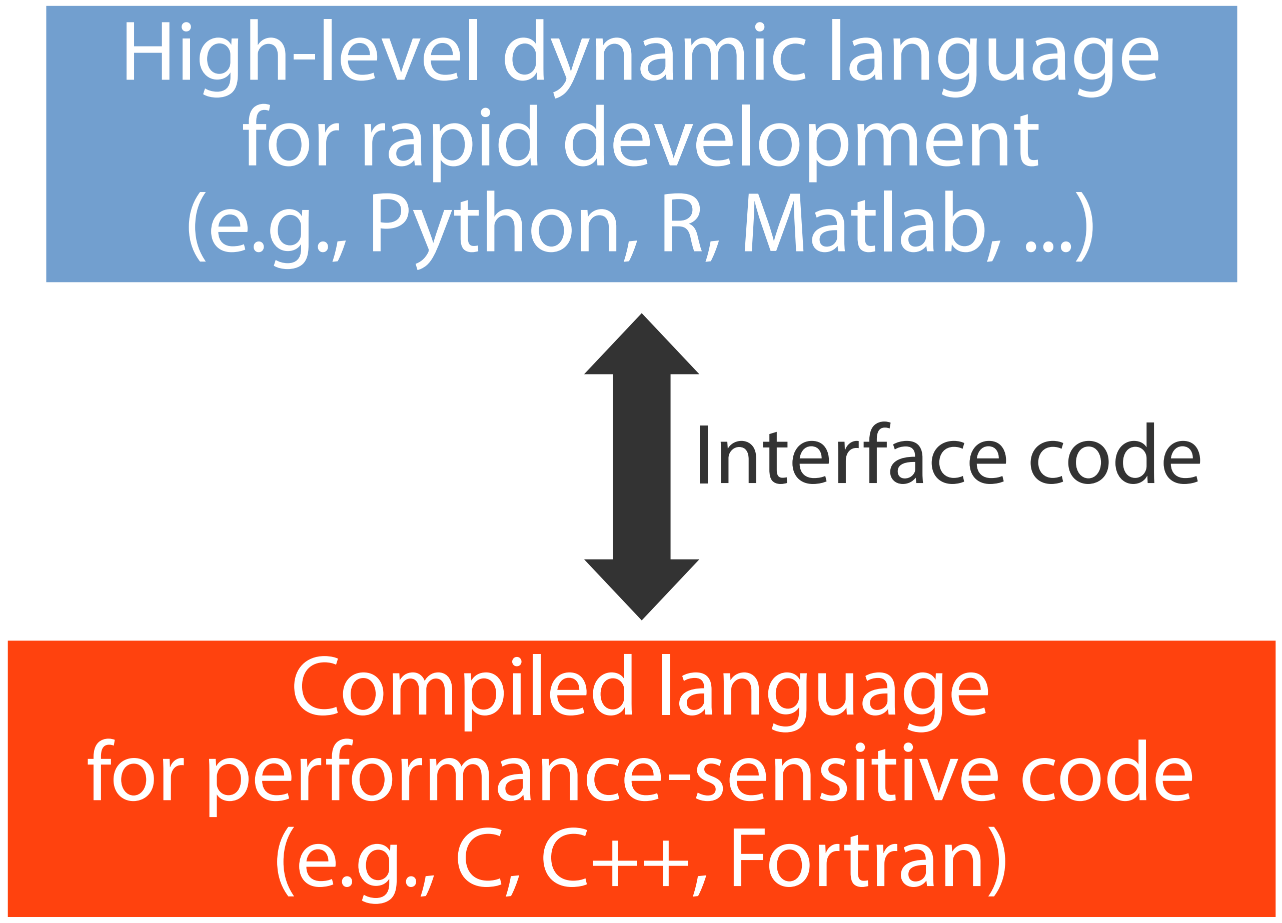
julia

# “Do I really have to learn yet another language?”

Kyle Barbary

*Large parts of this talk “borrowed” from Steven G Johnson's EuroSciPy 2014 Keynote. http://julialang.org/learning/*

## The “two language” problem


High-level dynamic language for rapid development (e.g., Python, R, Matlab, ...)
Interface code
Compiled language for performance-sensitive code (e.g., C, C++, Fortran)

*In contrast, most of Julia's standard library is written in Julia.*
*“Users are developers.”*

## "Just vectorize your code"?

= rely on mature external libraries
operating on large blocks of data
for performance-critical code

Good Advice! But...

- Someone has to write those libraries
- Eventually that person might be you
  (some problems impossible to vectorize or just very awkward)

## dynamic ≠ slow

*We want a language that's* ***open source****, with a liberal license. We want the* ***speed of C*** *with the* ***dynamism of Ruby****. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as* ***usable for general programming as Python****, as* ***easy for statistics as R****, as natural for string processing as Perl, as* ***powerful for linear algebra as Matlab****, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it* ***interactive*** *and we want it* ***compiled****.*

*(Did we mention it should be* ***as fast as C****?)*

– http://julialang.org/blog/2012/02/why-we-created-julia/

# Is it fast?

| | Fortran | Julia | Python | R | Matlab | Octave | Mathe-matica | JavaScript | G |
|---|---|---|---|---|---|---|---|---|---|
| | gcc 4.8.2 | 0.3.7 | 2.7.9 | 3.1.3 | R2014a | 3.8.1 | 10.0 | V8 3.14.5.9 | go1. |
| fib | 0.57 | 2.14 | 95.45 | 528.85 | 4258.12 | 9211.59 | 166.64 | 3.68 | 2.2 |
| parse_int | 4.67 | 1.57 | 20.48 | 54.30 | 1525.88 | 7568.38 | 17.70 | 2.29 | 3.7 |
| quicksort | 1.10 | 1.21 | 46.70 | 248.28 | 55.87 | 1532.54 | 48.47 | 2.91 | 1.0 |
| mandel | 0.87 | 0.87 | 18.83 | 58.97 | 60.09 | 393.91 | 6.12 | 1.86 | 1.1 |
| pi_sum | 0.83 | 1.00 | 21.07 | 14.45 | 1.28 | 260.28 | 1.27 | 2.15 | 1.2 |
| rand_mat_stat | 0.99 | 1.74 | 22.29 | 16.88 | 9.82 | 30.44 | 6.20 | 2.81 | 8.2 |
| rand_mat_mul | 4.05 | 1.09 | 1.08 | 1.63 | 1.12 | 1.06 | 1.13 | 14.58 | 8.4 |

*times relative to C (lower is better)*

```
# approximate π^2/6 with infinite series
function pi_sum()
    sum = 0.0
    for k = 1:10000
        sum += 1.0/(k*k)
    end
    sum
end
```

## Using Julia

- Command-line REPL (Read-Eval-Print Loop):

```
julia
```

- Running scripts:

```
julia filename.jl
```

- Ipython/Jupyter notebook:

```
ipython notebook --profile julia
```

## Can Python/R/etc be fast?

- Maybe, but it's difficult.

  *goto Steven G Johnson's slides...*

## Is it for you?

- Do you hate hate hate backwards compatibility issues?
  *The language is evolving.*

- Do you rely on tons of domain-specific libraries in other languages?
  *Native Julia package ecosystem is still young*

- Is speed never ever an issue for you?
  *There's other cool stuff, but main selling point is speed*