Python and Julia

Why do we need another language?

Dwight J. Browne
PyGotham 2014
Twitter: @dwightb2
GitHub: https://github.com/dbrowne/Presentations

History

The olden days: The Honeymooners - Oprah

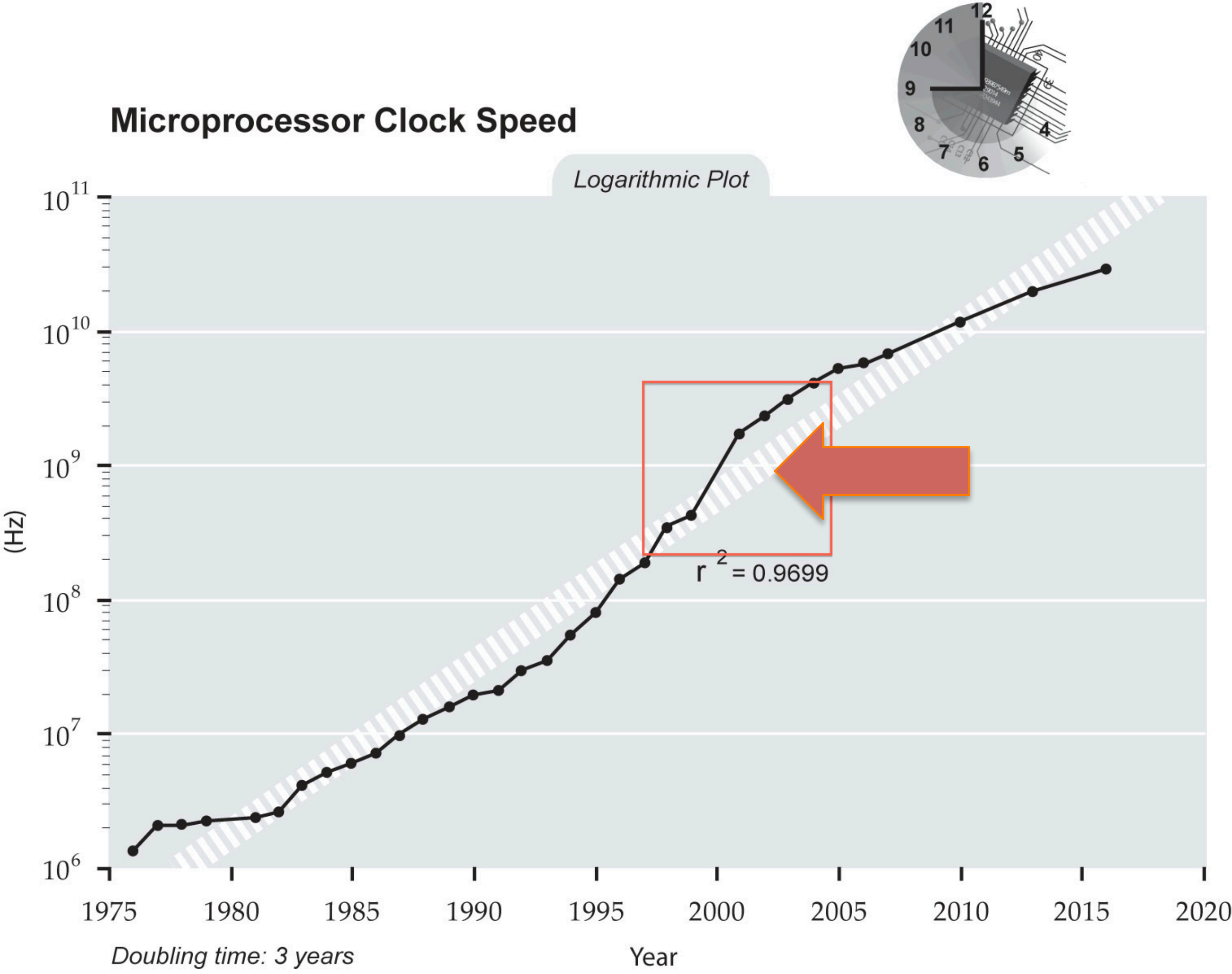
- Slow computers required efficient languages
  - FORTRAN, C
  - Required efficient and diligent coding
  - strncpy(tooSmall,tooBig,sizeof(tooBig) =

More History

Seinfeld, Friends

- Compressed delivery time
- Enterprise apps bloat
- C#, Java, Perl
- *nix or Win
- Computers kept getting faster

A Brief Diversion



The Dark Ages

Reality TV

- GOOD ENOUGH

Some Problems

500 Channels and Nothing to Watch

- Big applications
- Complex dependencies
- Cut and paste code
- Technical debt

The Big Problem

The Notorious E.C.L.E

- E.C.L.E. - Edit Compile Link Execute
- More data. Finite time
- Processing speeds are increasing slowly
- Need a better way

The Need For Speed

Fast and Furious

- High performance = C or FORTRAN
- Using BLAS, LAPACK = FORTRAN
- Still hindered by E.C.L.E.
- A use case for a dynamic language

Python

The Sopranos and Breaking Bad

- Multi Paradigm
- REPL and IPython save person years
- Less TIMTOWDI = Less wasted time
- Slowness will be forgiven for fast delivery
- But not always

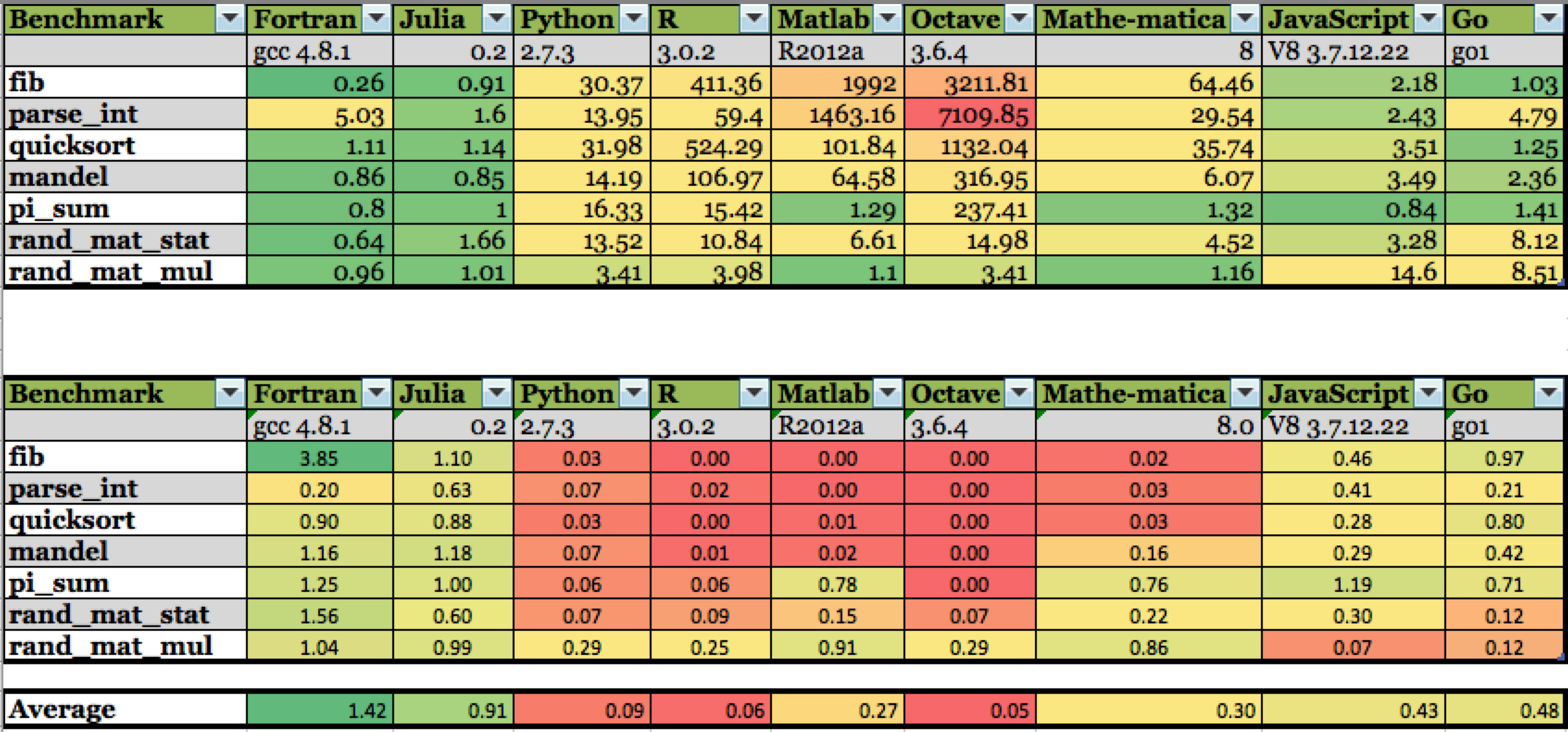
Julia

Julia? Where did this come from?

- Fast C like performance
- Incredibly young language = Immature library base
- C interoperability = Python interoperability
- Multiple dispatch
  - Focus on methods not methods of classes

But How Fast is It?

Professional driver on closed course



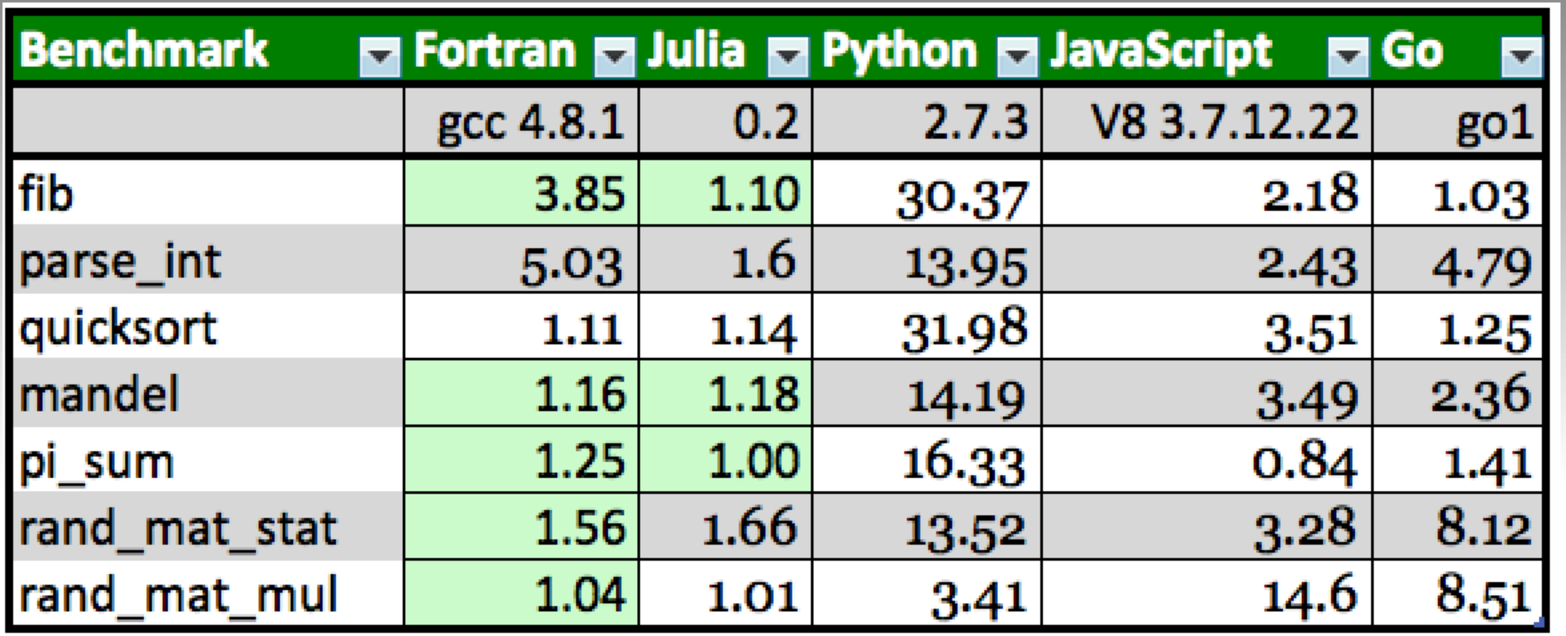
Benchmark	Fortran	Julia	Python	R	Matlab	Octave	Mathe-matica	JavaScript	Go
	gcc 4.8.1	0.2	2.7.3	3.0.2	R2012a	3.6.4	8	V8 3.7.12.22	go1
fib	0.26	0.91	30.37	411.36	1992	3211.81	64.46	2.18	1.03
parse_int	5.03	1.6	13.95	59.4	1463.16	7109.85	29.54	2.43	4.79
quicksort	1.11	1.14	31.98	524.29	101.84	1132.04	35.74	3.51	1.25
mandel	0.86	0.85	14.19	106.97	64.58	316.95	6.07	3.49	2.36
pi_sum	0.8	1	16.33	15.42	1.29	237.41	1.32	0.84	1.41
rand_mat_stat	0.64	1.66	13.52	10.84	6.61	14.98	4.52	3.28	8.12
rand_mat_mul	0.96	1.01	3.41	3.98	1.1	3.41	1.16	14.6	8.51

Benchmark	Fortran	Julia	Python	R	Matlab	Octave	Mathe-matica	JavaScript	Go
	gcc 4.8.1	0.2	2.7.3	3.0.2	R2012a	3.6.4	8.0	V8 3.7.12.22	go1
fib	3.85	1.10	0.03	0.00	0.00	0.00	0.02	0.46	0.97
parse_int	0.20	0.63	0.07	0.02	0.00	0.00	0.03	0.41	0.21
quicksort	0.90	0.88	0.03	0.00	0.01	0.00	0.03	0.28	0.80
mandel	1.16	1.18	0.07	0.01	0.02	0.00	0.16	0.29	0.42
pi_sum	1.25	1.00	0.06	0.06	0.78	0.00	0.76	1.19	0.71
rand_mat_stat	1.56	0.60	0.07	0.09	0.15	0.07	0.22	0.30	0.12
rand_mat_mul	1.04	0.99	0.29	0.25	0.91	0.29	0.86	0.07	0.12

Average	1.42	0.91	0.09	0.06	0.27	0.05	0.30	0.43	0.48

It’s Really That Fast?

Your mileage may vary



Benchmark	Fortran	Julia	Python	JavaScript	Go
	gcc 4.8.1	0.2	2.7.3	V8 3.7.12.22	go1
fib	3.85	1.10	30.37	2.18	1.03
parse_int	5.03	1.6	13.95	2.43	4.79
quicksort	1.11	1.14	31.98	3.51	1.25
mandel	1.16	1.18	14.19	3.49	2.36
pi_sum	1.25	1.00	16.33	0.84	1.41
rand_mat_stat	1.56	1.66	13.52	3.28	8.12
rand_mat_mul	1.04	1.01	3.41	14.6	8.51

GREEN CELLS	= X FASTER THAN C
NORMAL CELLS	= X SLOWER THAN C

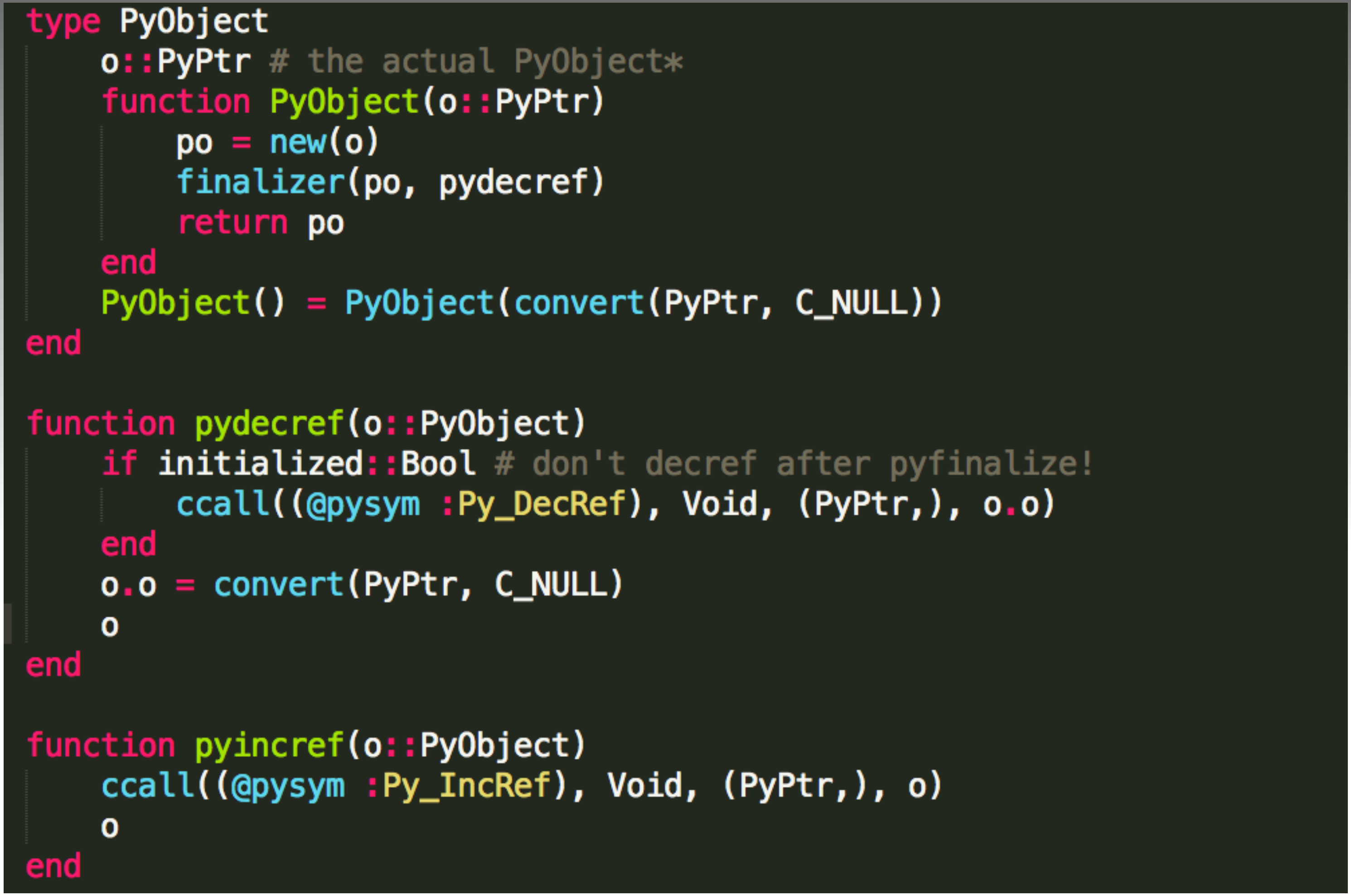
What is this Witchcraft?

How does it work?

- Python PyObject:
  - object.h, methodobject.h
  - descrobject.h

- Julia PyObject:
  - PyCall.jl, pytype.jl
  - PyPlot.jl for Matplotlib
  - ccall to call external C libraries

Julia PyObject



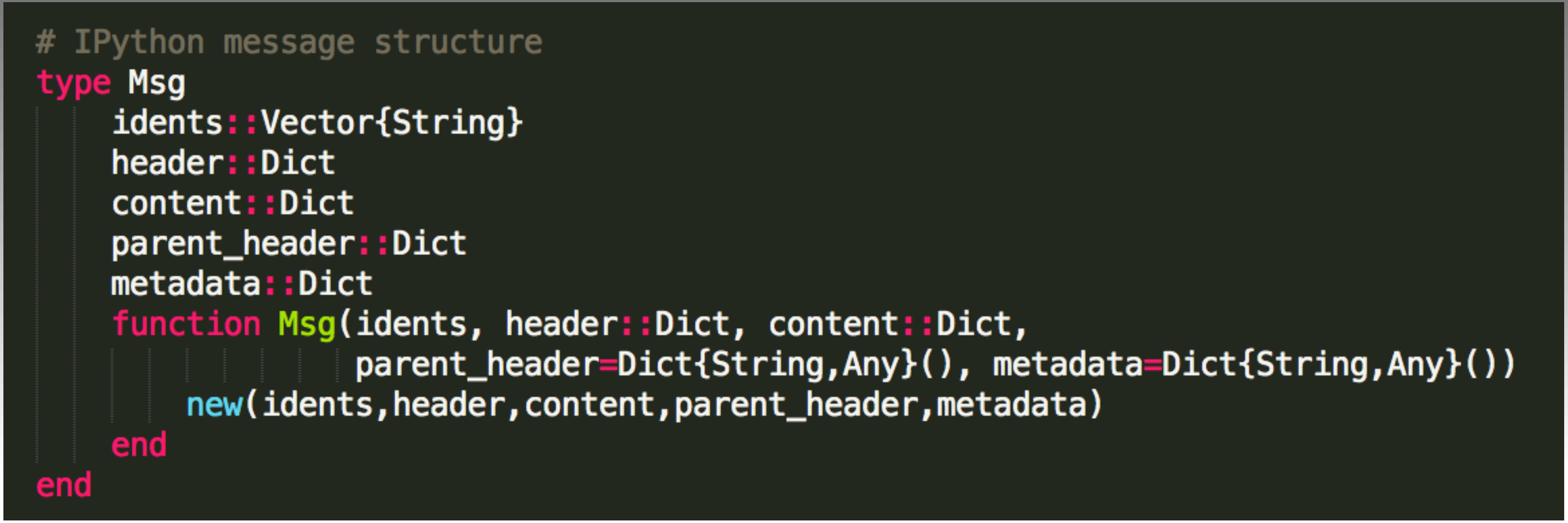
```
type PyObject
    o::PyPtr # the actual PyObject*
    function PyObject(o::PyPtr)
        po = new(o)
        finalizer(po, pydecref)
        return po
    end
    PyObject() = PyObject(convert(PyPtr, C_NULL))
end

function pydecref(o::PyObject)
    if initialized::Bool # don't decref after pyfinalize!
        ccall((@pysym :Py_DecRef), Void, (PyPtr,), o.o)
    end
    o.o = convert(PyPtr, C_NULL)
    o
end

function pyincref(o::PyObject)
    ccall((@pysym :Py_IncRef), Void, (PyPtr,), o)
    o
end
```

IPython and Julia

Yes. You can have your cake and eat it too!



```julia
# IPython message structure
type Msg
    idents::Vector{String}
    header::Dict
    content::Dict
    parent_header::Dict
    metadata::Dict
    function Msg(idents, header::Dict, content::Dict,
                 parent_header=Dict{String,Any}(), metadata=Dict{String,Any}())
        new(idents,header,content,parent_header,metadata)
    end
end
```

- Ijulia/src/msg.jl
- IPython/kernel/zmqIPython/kernel/zmq/session.py

That’s All Folks!

Is that it?

Demonstration