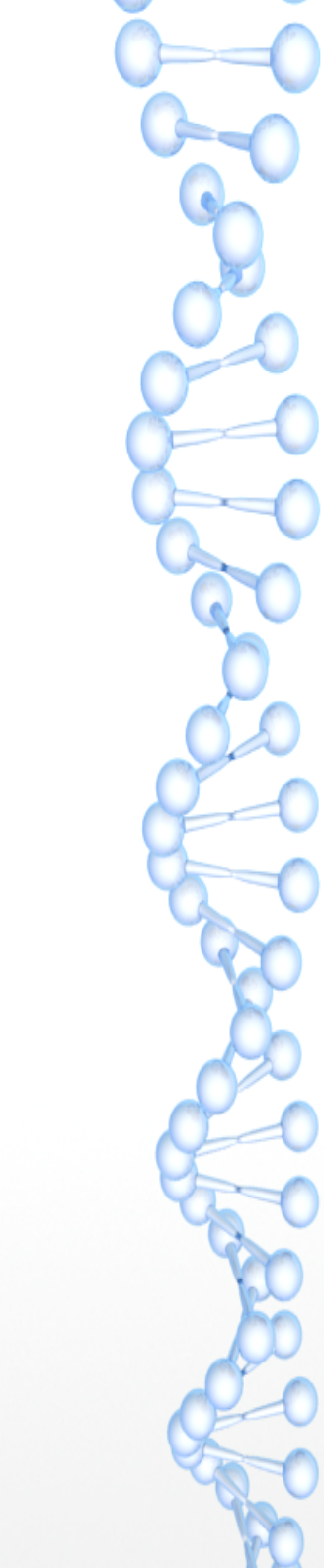


The Julia programming language

Bolyai Kollégium Informatika Szeminárium

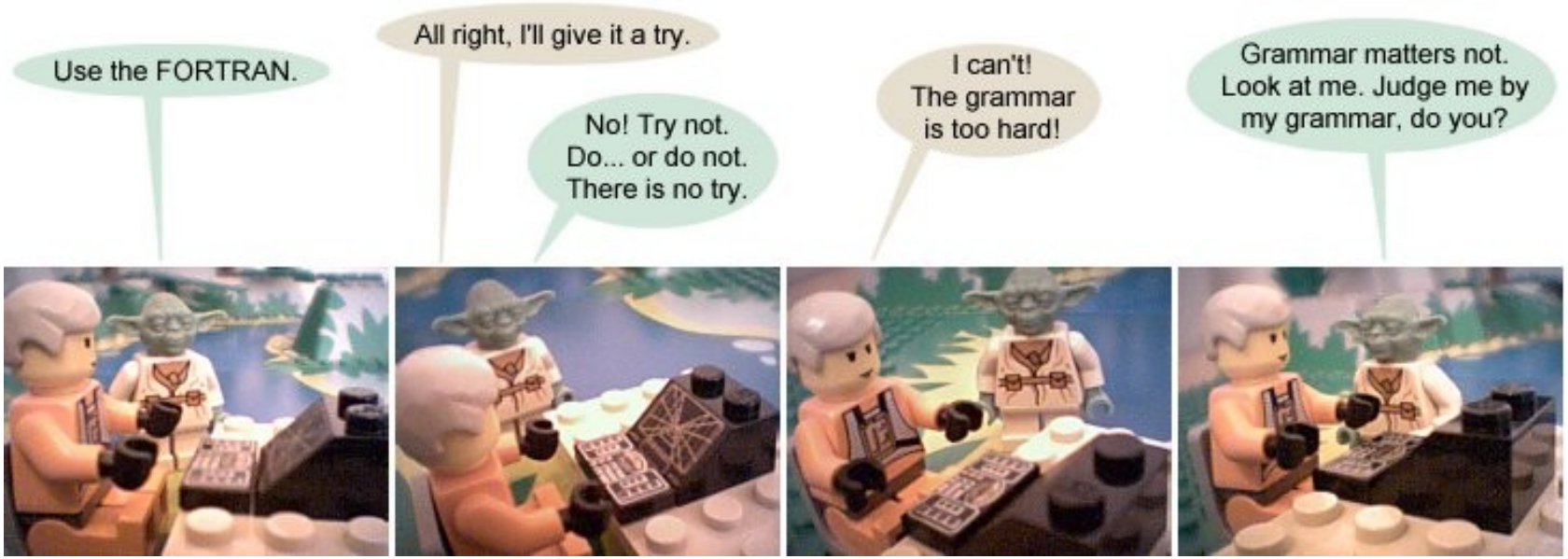
2018. 11. 06.

Features

- **Website: https://julialang.org/.**
- **Open-source: https://github.com/JuliaLang/julia**
- **High-level general-purpose dynamic programming language**
- **High-performance**
- **Interpreted script language (JIT)**
- **Cross-platform (even FreeBSD)**

History

- **Work began in 2009**
- **v1.0 in August 2018**
- **Tries to solve the “Fortran-Matlab problem”**



Languages for computing

C, Fortran, Assembly (?)

- Low-level
- Compiled
- Fast
- Procedural, no modern programming concepts

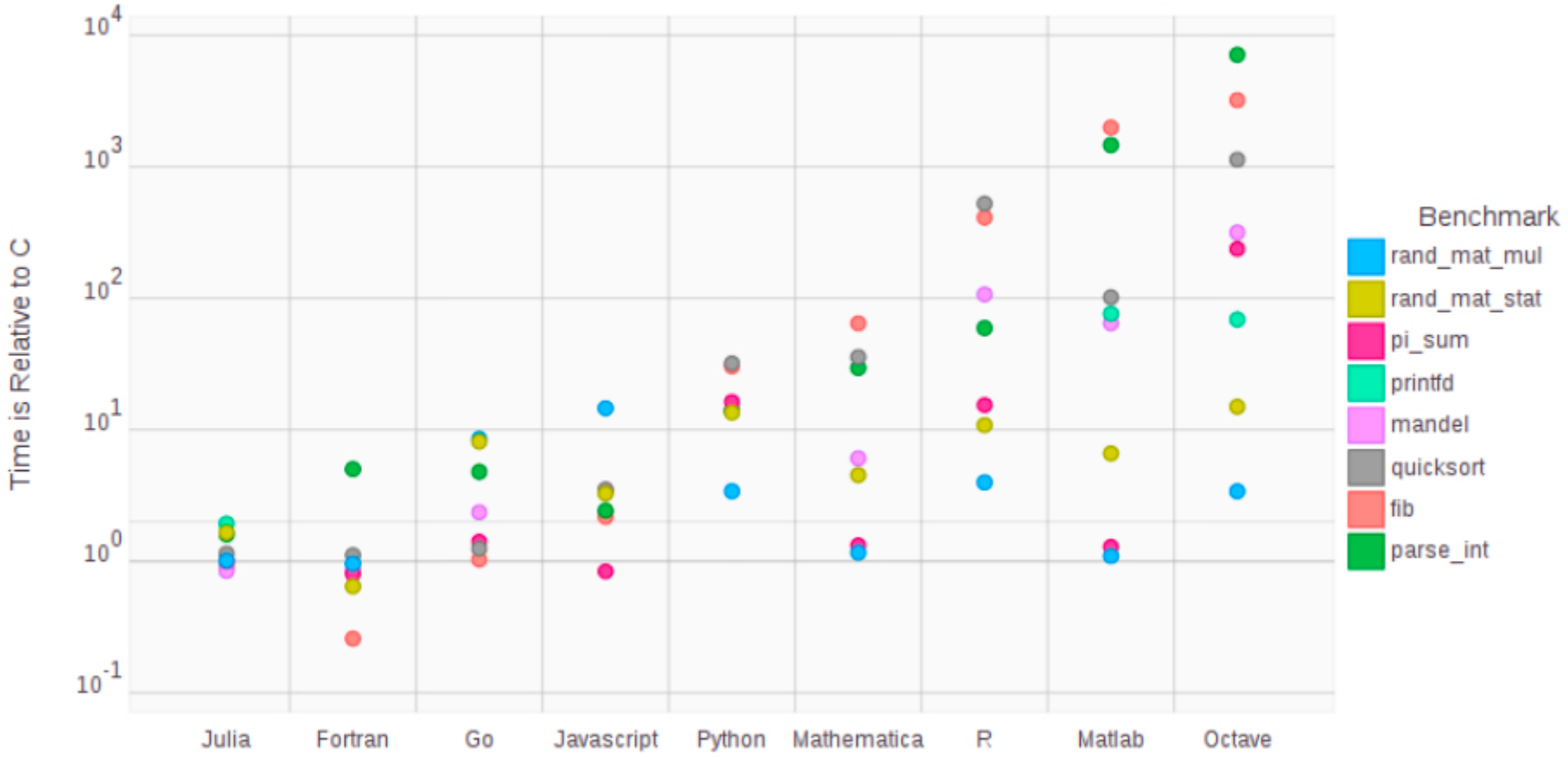
Python, Matlab, Mathematica

- High-level
- Interpreted
- Much slower
- Built-in support for many math operations (M*)
- Modern programming language concepts (Python)

Speed

- **Less than 2 times slower than C**
  - while "about 10 times faster" than Matlab (according to the FED [1])

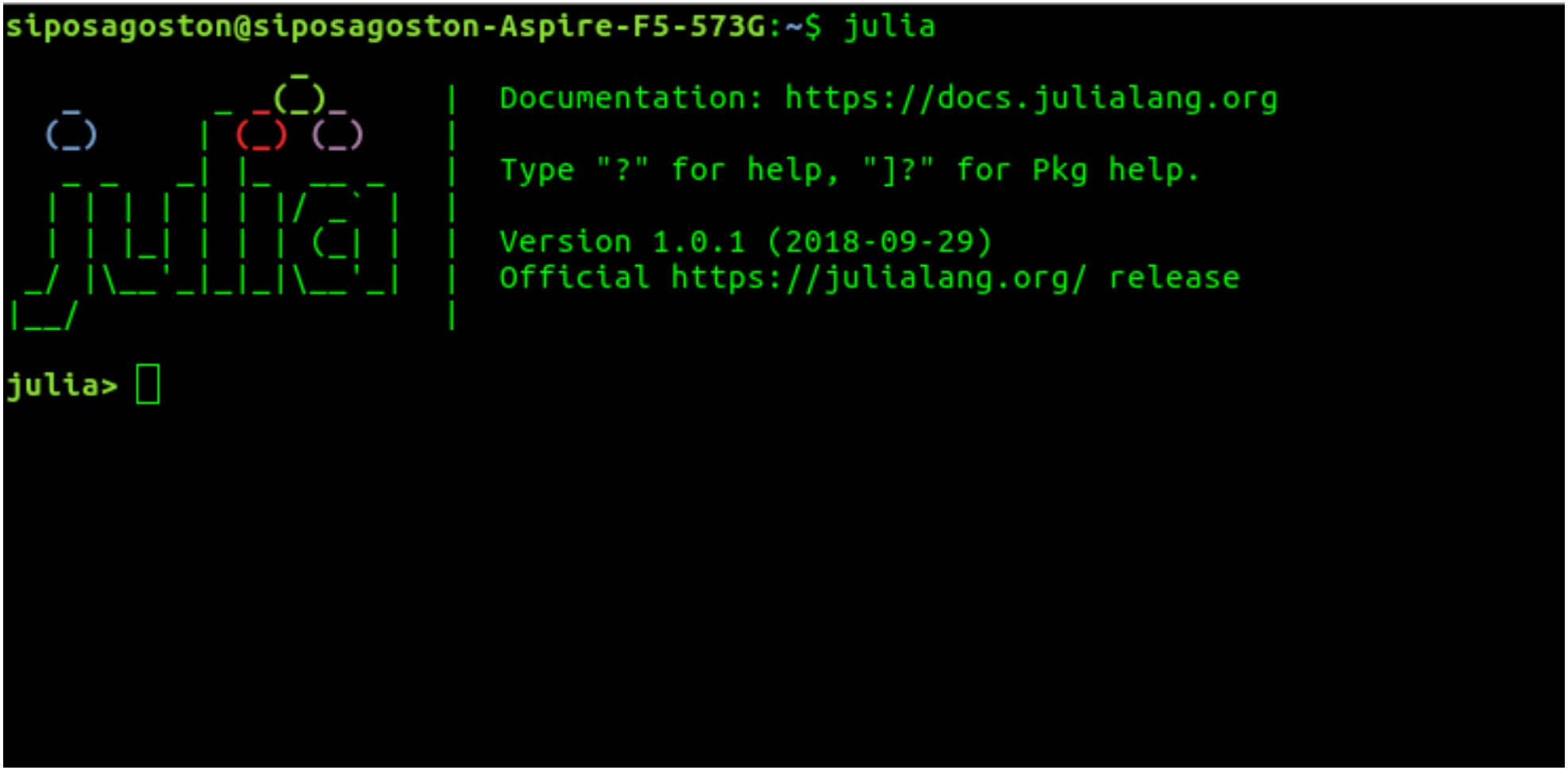
Speed


Time is Relative to C
10^4
10^3
10^2
10^1
10^0
10^{-1}
Julia
Fortran
Go
Javascript
Python
Mathematica
R
Matlab
Octave
Benchmark
rand_mat_mul
rand_mat_stat
pi_sum
printfd
mandel
quicksort
fib
parse_int

Usability

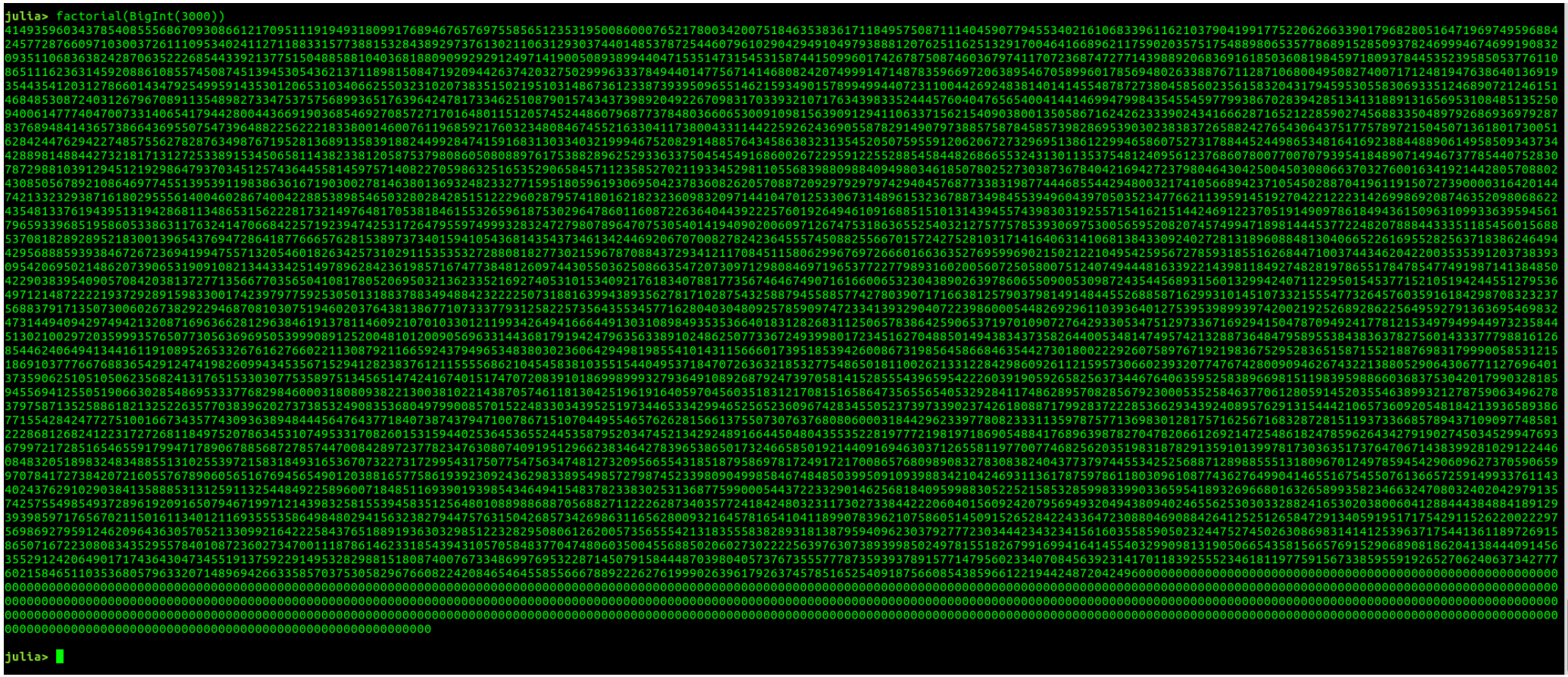
- **Multi-paradigm (procedural, functional, and object-oriented)**
- **Built-in support for many things**
  - Standard library is written in Julia
  - Package manager integrated with Github (1900+ packages)
- **Usable interactive shell**
- **IDE as Atom/VS code plugin**

REPL


siposagoston@siposagoston-Aspire-F5-573G:~$ julia
Documentation: https://docs.julialang.org
Type "?" for help, "]?" for Pkg help.
Version 1.0.1 (2018-09-29)
Official https://julialang.org/ release
julia>

REPL

```
> factorial(BigInt(3000))
```



Functions

```
function hello()
    println("你好,世界")
end
```

Functions

```
function ∑(x...)
    if(length(x) == 1)
        return x[1]
    end
    x[1]+∑(x[2:length(x)]...)
end
```

Functions

```
funs = [sin, cos, tan]
vals = [1, 2, 3]
apply(f, x) = map((f,x)->f(x), f, x)

apply(funs, vals)
```

Numbers

```
# Division
7 / 5
7 ÷ 5
7 % 5
[1,2,3,4,5,6] ./ 3
# Complex
(1+2im)*(4-3im)
cos(5+3im)
sqrt(-1) # error
sqrt(-1+0im)
# Rationals
r = 6//5 * 7//10
float(r)
# BigInt
2^BigInt(1200)
# BigFloat
BigFloat(2.0^66) / 3
```

Numbers

```
A = [1 2 3; 4 1 6; 7 8 1]
B = [1.5 2 -4; 3 -1 -6; -10 2.3 4]

using LinearAlgebra

function linalgtest(M)
    println(LinearAlgebra.det(M))
    vals, vecs = LinearAlgebra.eigen(M)
    x = M\[1;1;1]
    l, u = factorize(M)
    for y in (l, u, vals, vecs, x)
        Base.show(stdout, "text/plain", y)
        println();
    end
end
```

Types

```
primitive type UInt128 <: Unsigned 128 end

struct A
    x :: Int64
    y
end

b = A( 2, "alma" )
c = A( 3, 6.2 )
d = A( 2.1, 6 ) # error

typeof(b.y) # String
typeof(c.y) # Float64
typeof(b.y) <: Number # false
typeof(c.y) <: Number # true
```

Types

```
struct Polynomial{R}
    coeffs::Vector{R}
end

function (p::Polynomial)(x)
    v = p.coeffs[end]
    for i = (length(p.coeffs)-1):-1:1
        v = v*x + p.coeffs[i]
    end
    return v
end
```

Function dispatch

```
f(x, y) = x + y
f(x :: Number, y :: Number) = x + y
f(x :: AbstractFloat, y :: Number) = x * y
f(x :: Number, y :: Integer) = x - y

f(1, 1) # 0
f(1.0, 1) # error
f(1, 1.0) # 2.0
f(1.0, 1.0) # 1.0
f("a", "b") # "ab"
```

Functions

```
function syntax(str)
    expr = Meta.parse(str)
    dump(expr)
    function (x)
      eval(Base.Cartesian.lreplace(expr,:x,x))
    end
end

syntax("(x + 4) / 2")
```

Calling C

```
function libctime()
    t = ccall((:time), Int64, ())
end

function sdltime()
    t = Int64(ccall((:SDL_GetTicks, :libSDL2), UInt32, ()))
end
```

Calling C

```
function mycompare(a, b)::Cint
    return (a < b) ? -1 : ((a > b) ? +1 : 0)
end

function csort()
    mycompare_c = @cfunction(mycompare, Cint,
(Ref{Cdouble}, Ref{Cdouble}))
    A = [1.3, -2.7, 4.4, 3.1]
    ccall(:qsort, Cvoid, (Ptr{Cdouble}, Csize_t,
Csize_t, Ptr{Cvoid}), A, length(A),
sizeof(eltype(A)), mycompare_c)
    A
end
```

Syntax things

```
x, y = "q", "w"
"∀ $x ∃ $y"

alma[1](x, y)[3]
```

Questions?

- **Example codes available at**
  http://cg.elte.hu/~agostons/presentations/julia/examples.jl