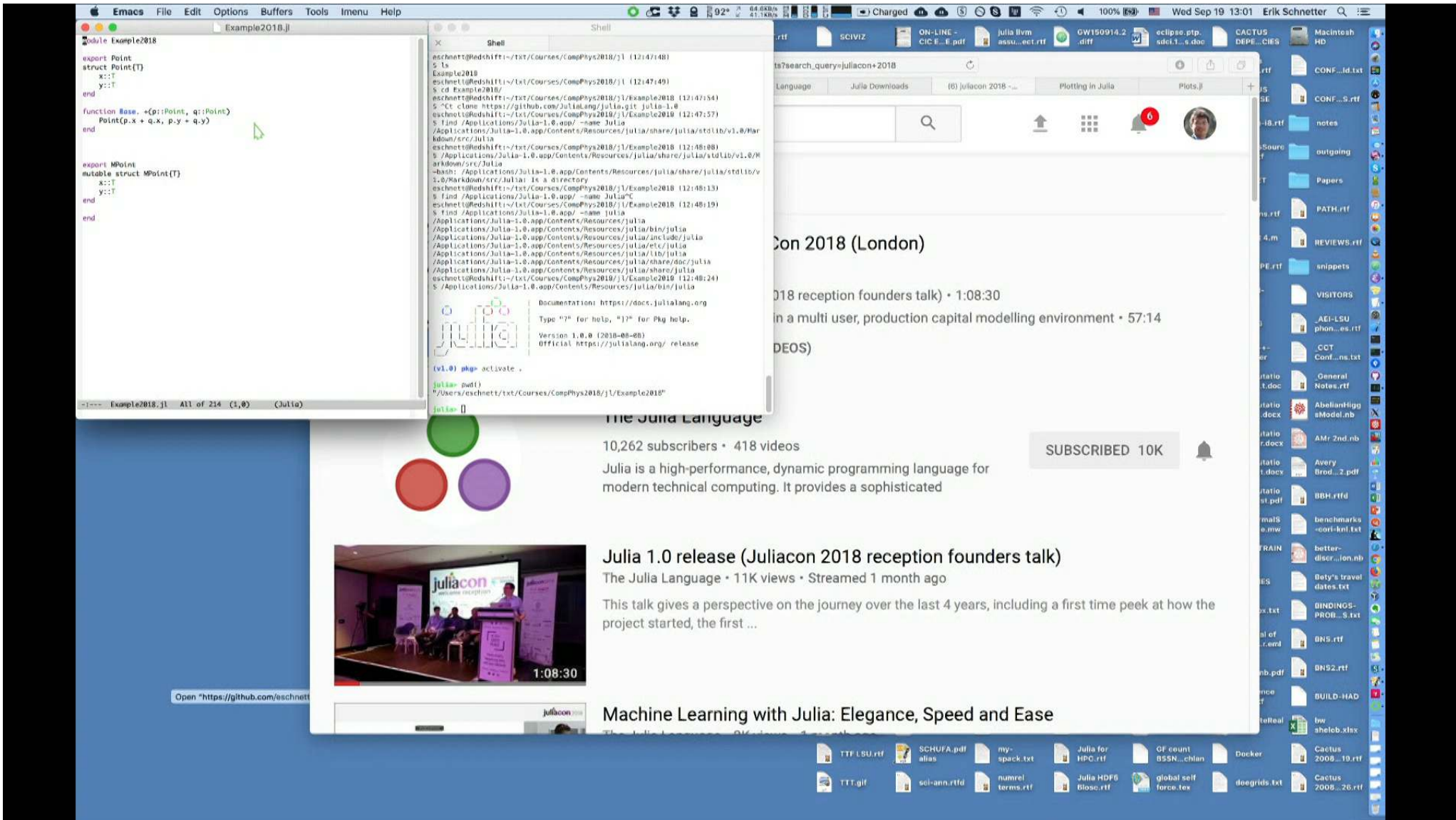


Title: Computational Physics - Lecture 3

Date: Sep 19, 2018 01:00 PM

URL: <http://pirsa.org/18090044>

Abstract:



```
File Edit Options Buffers Tools Help
```

```
Module Example2018
```

```
export Point
struct Point{T}
    x::T
    y::T
end

function Base. +(p::Point, q::Point)
    Point(p.x + q.x, p.y + q.y)
end
```

```
export MPoint
mutable struct MPoint{T}
    x::T
    y::T
end
```

```
--uu:--F1 Example2018.jl All L1 (Fundamental)
```

```
Mark set
```

```
eschnett@Redshift:~/txt/Courses/CompPhys2018/jl (12:47:48)
$ ls
Example2018
eschnett@Redshift:~/txt/Courses/CompPhys2018/jl (12:47:49)
$ cd Example2018/
eschnett@Redshift:~/txt/Courses/CompPhys2018/jl/Example2018 (12:47:54)
$ ^Ct clone https://github.com/JuliaLang/julia.git julia-1.0
eschnett@Redshift:~/txt/Courses/CompPhys2018/jl/Example2018 (12:47:57)
$ find /Applications/Julia-1.0.app/ -name Julia
/Applications/Julia-1.0.app/Contents/Resources/julia/share/julia/stdlib/v1.0/Markdown/src/Julia
eschnett@Redshift:~/txt/Courses/CompPhys2018/jl/Example2018 (12:48:08)
$ /Applications/Julia-1.0.app/Contents/Resources/julia/share/julia/stdlib/v1.0/Markdown/src/Julia
-bash: /Applications/Julia-1.0.app/Contents/Resources/julia/share/julia/stdlib/v1.0/Markdown/src/Julia: Is a directory
eschnett@Redshift:~/txt/Courses/CompPhys2018/jl/Example2018 (12:48:13)
$ find /Applications/Julia-1.0.app/ -name Julia^C
eschnett@Redshift:~/txt/Courses/CompPhys2018/jl/Example2018 (12:48:19)
$ find /Applications/Julia-1.0.app/ -name julia
/Applications/Julia-1.0.app/Contents/Resources/julia
/Applications/Julia-1.0.app/Contents/Resources/julia/bin/julia
/Applications/Julia-1.0.app/Contents/Resources/julia/include/julia
/Applications/Julia-1.0.app/Contents/Resources/julia/etc/julia
/Applications/Julia-1.0.app/Contents/Resources/julia/lib/julia
/Applications/Julia-1.0.app/Contents/Resources/julia/share/doc/julia
/Applications/Julia-1.0.app/Contents/Resources/julia/share/julia
eschnett@Redshift:~/txt/Courses/CompPhys2018/jl/Example2018 (12:48:24)
$ /Applications/Julia-1.0.app/Contents/Resources/julia/bin/julia
```

```
Documentation: https://docs.julialang.org
Type "?" for help, "}" for Pkg help.

Version 1.0.0 (2018-08-08)
Official https://julialang.org/ release
```

```
(v1.0) pkg> activate .
```

```
julia> pwd()
```

```
"/Users/eschnett/txt/Courses/CompPhys2018/jl/Example2018"
```

```
julia> []
```

```

File Edit Options Buffers Tools Help
y::T
end

#####

export Polynomial
struct Polynomial{T}
  coeffs::Vector{T}
end

function evaluate(p::Polynomial, x)
  sum(c * x^(i-1) for (i,c) in enumerate(p.coeffs))
end

end

--uu:--F1 Example2018.jl Bot L33 (Fundamental)-----
using Example2018

@show Polynomial([1,2,3])
@show evaluate(Polynomial([1,2,3]), 2)

```

```

--uu:--F1 runtests.jl All L5 (Fundamental)-----
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Example2018/test/runtests.jl

```

```

1
2
3

julia> [c for c in enumerate(cs)]
3-element Array{Tuple{Int64,Int64},1}:
(1, 1)
(2, 2)
(3, 3)

julia> cs = ["a", "bc", "def"]
3-element Array{String,1}:
"a"
"bc"
"def"

julia> [c for c in enumerate(cs)]
3-element Array{Tuple{Int64,String},1}:
(1, "a")
(2, "bc")
(3, "def")

(Example2018) pkg> test
Updating registry at `~/.julia/registries/General`
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Updating `Project.toml`
[no changes]
Testing Example2018
Resolving package versions...
Polynomial{Int64}([1, 2, 3])
ERROR: LoadError: UndefVarError: evaluate not defined
Stacktrace:
 [1] top-level scope at show.jl:555
 [2] include at ./boot.jl:317 [inlined]
 [3] include_relative(::Module, ::String) at ./loading.jl:1038
 [4] include(::Module, ::String) at ./sysimg.jl:29
 [5] include(::String) at ./client.jl:388
 [6] top-level scope at none:0
in expression starting at /Users/eschnett/txt/Courses/CompPhys2018/jl/Example2018/test/runtests.jl:4
ERROR: Package Example2018 errored during testing

(Example2018) pkg>

```

```

File Edit Options Buffers Tools Help
export evaluate
function evaluate(p::Polynomial, x)
    sum(c * x^(i-1) for (i,c) in enumerate(p.coeffs))
end

function Base. +(p::Polynomial, q::Polynomial)
    l = max(length(p.coeffs), length(q.coeffs))
    rcoeffs = zeros(l)
    for (i,c) in enumerate(p.coeffs)
        rcoeffs[i] += c
    end
    for (i,c) in enumerate(q.coeffs)
        rcoeffs[i] += c
    end
    Polynomial(rcoeffs)
end

end

--uu:--F1 Example2018.jl Bot L42 (Fundamental)
using Example2018

@show Polynomial([1,2,3])
@show evaluate(Polynomial([1,2,3]), 2)
@show Polynomial([1,2,3]) + Polynomial([4,5])

```

```

--uu:--F1 runtests.jl All L6 (Fundamental)
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Example2018/test/runtests.jl

```

```

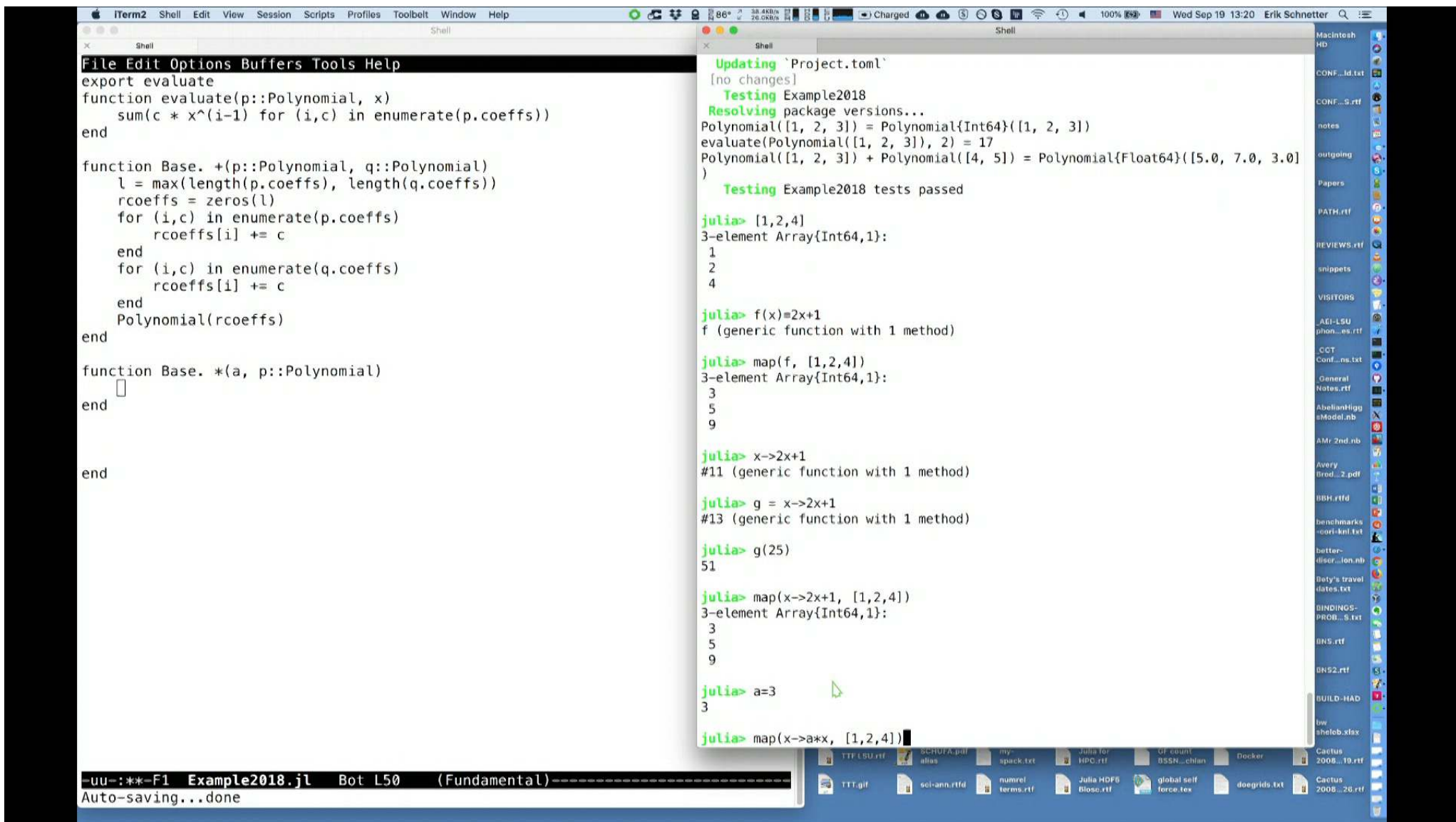
Updating `Project.toml`
[no changes]
Testing Example2018
Resolving package versions...
Polynomial{Int64}([1, 2, 3]) = Polynomial{Int64}([1, 2, 3])
ERROR: LoadError: UndefVarError: evaluate not defined
Stacktrace:
 [1] top-level scope at show.jl:555
 [2] include at ./boot.jl:317 [inlined]
 [3] include_relative(::Module, ::String) at ./loading.jl:1038
 [4] include(::Module, ::String) at ./sysimg.jl:29
 [5] include(::String) at ./client.jl:388
 [6] top-level scope at none:0
in expression starting at /Users/eschnett/txt/Courses/CompPhys2018/jl/Example2018/test/runtests.jl:4
ERROR: Package Example2018 errored during testing

(Example2018) pkg> test
Updating registry at `~/julia/registries/General`
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Updating `Project.toml`
[no changes]
Testing Example2018
Resolving package versions...
Polynomial{Int64}([1, 2, 3]) = Polynomial{Int64}([1, 2, 3])
evaluate(Polynomial{Int64}([1, 2, 3]), 2) = 17
Testing Example2018 tests passed

(Example2018) pkg> test
Updating registry at `~/julia/registries/General`
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Updating `Project.toml`
[no changes]
Testing Example2018
Resolving package versions...
Polynomial{Int64}([1, 2, 3]) = Polynomial{Int64}([1, 2, 3])
evaluate(Polynomial{Int64}([1, 2, 3]), 2) = 17
Polynomial{Float64}([5.0, 7.0, 3.0]) = Polynomial{Float64}([5.0, 7.0, 3.0])
Testing Example2018 tests passed

(Example2018) pkg>

```



```

File Edit Options Buffers Tools Help
export evaluate
function evaluate(p::Polynomial, x)
    sum(c * x^(i-1) for (i,c) in enumerate(p.coeffs))
end

function Base. +(p::Polynomial, q::Polynomial)
    l = max(length(p.coeffs), length(q.coeffs))
    rcoeffs = zeros(l)
    for (i,c) in enumerate(p.coeffs)
        rcoeffs[i] += c
    end
    for (i,c) in enumerate(q.coeffs)
        rcoeffs[i] += c
    end
    Polynomial(rcoeffs)
end

function Base. *(a, p::Polynomial)
    Polynomial(map(x->a*x, p.coeffs))
end
uu-:*-F1 Example2018.jl 43% L49 (Fundamental)-----
using Example2018

@show Polynomial([1,2,3])
@show evaluate(Polynomial([1,2,3]), 2)
@show Polynomial([1,2,3]) + Polynomial([4,5])
@show 3 * Polynomial([1,2,4,42])

```

```

3
5
9

julia> x->2x+1
#11 (generic function with 1 method)

julia> g = x->2x+1
#13 (generic function with 1 method)

julia> g(25)
51

julia> map(x->2x+1, [1,2,4])
3-element Array{Int64,1}:
 3
 5
 9

julia> a=3
3

julia> map(x->a*x, [1,2,4])
3-element Array{Int64,1}:
 3
 6
12

(Example2018) pkg> test
Updating registry at `~/julia/registries/General`
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Updating `Project.toml`
[no changes]
Testing Example2018
Resolving package versions...
Polynomial([1, 2, 3]) = Polynomial{Int64}([1, 2, 3])
evaluate(Polynomial([1, 2, 3]), 2) = 17
Polynomial([1, 2, 3]) + Polynomial([4, 5]) = Polynomial{Float64}([5.0, 7.0, 3.0])
3 * Polynomial([1, 2, 4, 42]) = Polynomial{Int64}([3, 6, 12, 126])
Testing Example2018 tests passed

(Example2018) pkg>

```

```

uu-:---F1 runtests.jl All L7 (Fundamental)-----

```

```

File Edit Options Buffers Tools Help
export evaluate
function evaluate(p::Polynomial, x)
    sum(c * x^(i-1) for (i,c) in enumerate(p.coeffs))
end

function Base. +(p::Polynomial, q::Polynomial)
    l = max(length(p.coeffs), length(q.coeffs))
    rcoeffs = zeros(l)
    for (i,c) in enumerate(p.coeffs)
        rcoeffs[i] += c
    end
    for (i,c) in enumerate(q.coeffs)
        rcoeffs[i] += c
    end
    Polynomial(rcoeffs)
end

function Base. *(a, p::Polynomial)
    Polynomial(map(x->a*x, p.coeffs))
end

export deriv
function deriv(p::Polynomial)
end

end

-uu-:**-F1 Example2018.jl Bot L55 (Fundamental)

```

```

function with 1 method)
2x+1
function with 1 method)

2x+1, [1,2,4])
y{Int64,1}:

a*x, [1,2,4])
y{Int64,1}:

pkg> test
istry at `~/julia/registries/General`
-repo `https://github.com/JuliaRegistries/General.git`
kage versions...
object.toml`

mple2018
kage versions...
2, 3]) = Polynomial{Int64}([1, 2, 3])
omial([1, 2, 3]), 2) = 17
2, 3]) + Polynomial{Int64}([4, 5]) = Polynomial{Float64}([5.0, 7.0, 3.0])

([1, 2, 4, 42]) = Polynomial{Int64}([3, 6, 12, 126])
mple2018 tests passed

pkg>

```

```
File Edit Options Buffers Tools Help
end
Polynomial(rcoeffs)
end
function Base.*(a, p::Polynomial)
    Polynomial(map(x->a*x, p.coeffs))
end
export deriv
function deriv(p::Polynomial)
    Polynomial([i * c for (i,c) in enumerate(p.coeffs)][2:end])
end
end

--uu-:---F1 Example2018.jl Bot L55 (Fundamental)-----
using Example2018

@show Polynomial([1,2,3])
@show evaluate(Polynomial([1,2,3]), 2)
@show Polynomial([1,2,3]) + Polynomial([4,5])
@show 3 * Polynomial([1,2,4,42])
@show deriv(Polynomial([1,1,1]))
[]

julia> map(x->a*x, [1,2,4])
3-element Array{Int64,1}:
 3
 6
12

(Example2018) pkg> test
Updating registry at `~/julia/registries/General`
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Updating `Project.toml`
[no changes]
Testing Example2018
Resolving package versions...
Polynomial([1, 2, 3]) = Polynomial{Int64}([1, 2, 3])
evaluate(Polynomial([1, 2, 3]), 2) = 17
Polynomial([1, 2, 3]) + Polynomial([4, 5]) = Polynomial{Float64}([5.0, 7.0, 3.0])
3 * Polynomial([1, 2, 4, 42]) = Polynomial{Int64}([3, 6, 12, 126])
Testing Example2018 tests passed

julia> [2,4,6,8]
4-element Array{Int64,1}:
 2
 4
 6
 8

julia> [2,4,6,8][2]
4

julia> [2,4,6,8][2:3]
2-element Array{Int64,1}:
 4
 6

julia> [2,4,6,8][2:end]
3-element Array{Int64,1}:
 4
 6
 8

(Example2018) pkg> tes

--uu-:---F1 runtests.jl All L8 (Fundamental)-----
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Example2018/test/runtests.jl
```

```

File Edit Options Buffers Tools Help
end
Polynomial(rcoeffs)
end

function Base.*(a, p::Polynomial)
    Polynomial(map(x->a*x, p.coeffs))
end

export deriv
function deriv(p::Polynomial)
    Polynomial([i * c for (i,c) in enumerate(p.coeffs)][2:end])
end

end

--uu-:---F1 Example2018.jl Bot L55 (Fundamental)-----

```

```

using Example2018

@show Polynomial([1,2,3])
@show evaluate(Polynomial([1,2,3]), 2)
@show Polynomial([1,2,3]) + Polynomial([4,5])
@show 3 * Polynomial([1,2,4,42])
@show deriv(Polynomial([1,1,1]))

```

```

--uu-:---F1 runtests.jl All L8 (Fundamental)-----
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Example2018/test/runtests.jl

```

```

2
4
6
8

julia> [2,4,6,8][2]
4

julia> [2,4,6,8][2:3]
2-element Array{Int64,1}:
 4
 6

julia> [2,4,6,8][2:end]
3-element Array{Int64,1}:
 4
 6
 8

(Example2018) pkg> test
Updating registry at ~/.julia/registries/General
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Updating `Project.toml`
[no changes]
Testing Example2018
Resolving package versions...
Polynomial([1, 2, 3]) = Polynomial{Int64}([1, 2, 3])
evaluate(Polynomial([1, 2, 3]), 2) = 17
Polynomial([1, 2, 3]) + Polynomial([4, 5]) = Polynomial{Float64}([5.0, 7.0, 3.0])
3 * Polynomial([1, 2, 4, 42]) = Polynomial{Int64}([3, 6, 12, 126])
ERROR: LoadError: syntax: incomplete: premature end of input
Stacktrace:
 [1] include at ./boot.jl:317 [inlined]
 [2] include_relative(::Module, ::String) at ./loading.jl:1038
 [3] include(::Module, ::String) at ./sysimg.jl:29
 [4] include(::String) at ./client.jl:388
 [5] top-level scope at none:0
in expression starting at /Users/eschnett/txt/Courses/CompPhys2018/jl/Example2018/test/runtests.jl:7
ERROR: Package Example2018 errored during testing

(Example2018) pkg>

```

```

File Edit Options Buffers Tools Help
end
Polynomial(rcoeffs)
end
function Base.*(a, p::Polynomial)
    Polynomial(map(x->a*x, p.coeffs))
end
export deriv
function deriv(p::Polynomial)
    Polynomial([i * c for (i,c) in enumerate(p.coeffs)][2:end])
end
end

-uu-:---F1 Example2018.jl Bot L55 (Fundamental)-----
using Example2018

```

```

@show Polynomial([1,2,3])
@show evaluate(Polynomial([1,2,3]), 2)
@show Polynomial([1,2,3]) + Polynomial([4,5])
@show 3 * Polynomial([1,2,4,42])
@show deriv(Polynomial([1,1,1]))

-uu-:---F1 runtests.jl All L7 (Fundamental)-----
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Example2018/test/runtests.jl

```

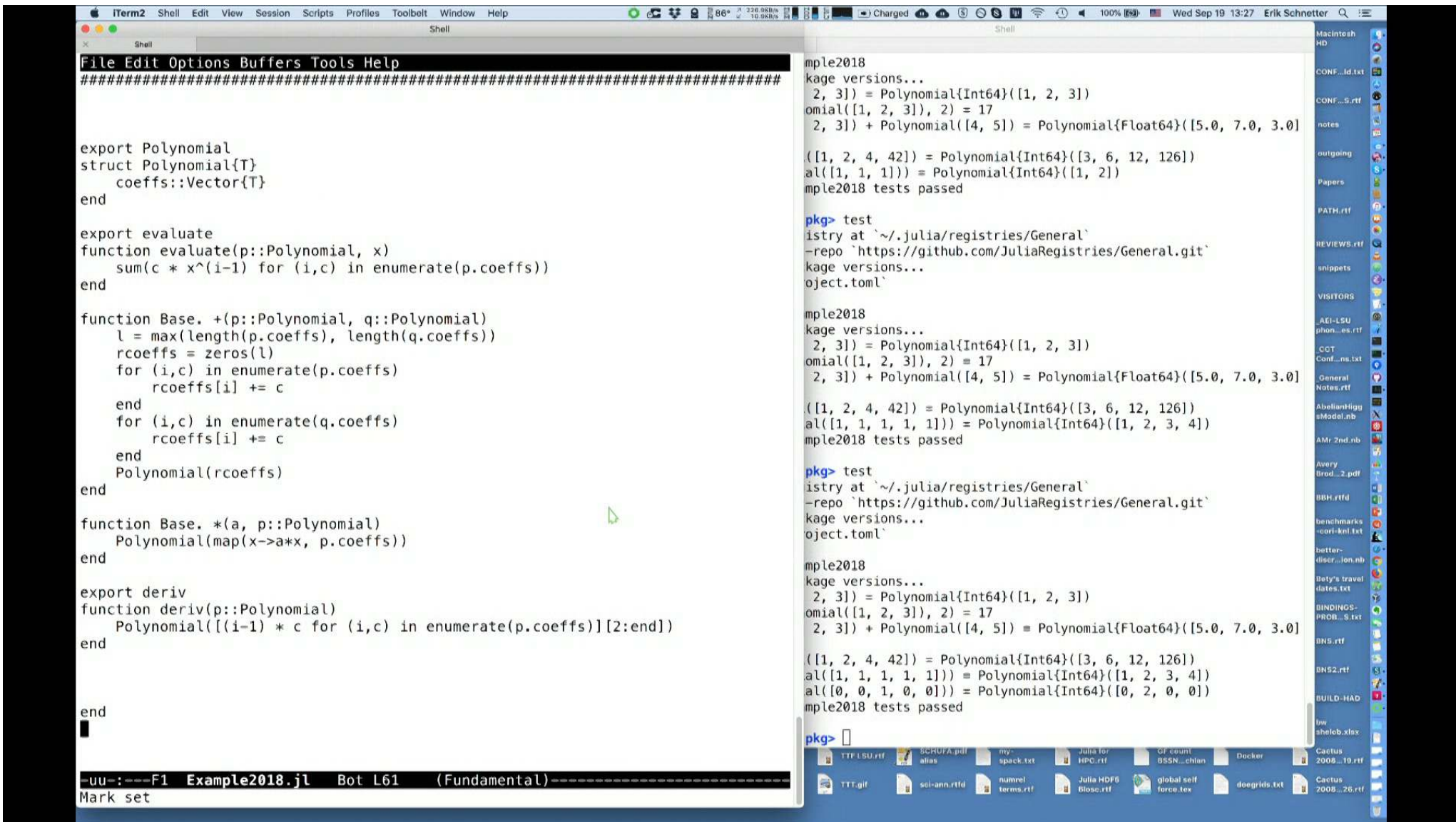
```

6
8
(Example2018) pkg> test
Updating registry at ~/.julia/registries/General
Updating git-repo https://github.com/JuliaRegistries/General.git
Resolving package versions...
Updating Project.toml
[no changes]
Testing Example2018
Resolving package versions...
Polynomial{Int64}([1, 2, 3]) = Polynomial{Int64}([1, 2, 3])
evaluate(Polynomial{Int64}([1, 2, 3]), 2) = 17
Polynomial{Float64}([1, 2, 3]) + Polynomial{Float64}([4, 5]) = Polynomial{Float64}([5.0, 7.0, 3.0])
3 * Polynomial{Int64}([1, 2, 4, 42]) = Polynomial{Int64}([3, 6, 12, 126])
ERROR: LoadError: syntax: incomplete: premature end of input
Stacktrace:
 [1] include at ./boot.jl:317 [inlined]
 [2] include_relative(::Module, ::String) at ./loading.jl:1038
 [3] include(::Module, ::String) at ./sysimg.jl:29
 [4] include(::String) at ./client.jl:388
 [5] top-level scope at none:0
in expression starting at /Users/eschnett/txt/Courses/CompPhys2018/jl/Example2018/test/runtests.jl:7
ERROR: Package Example2018 errored during testing

(Example2018) pkg> test
Updating registry at ~/.julia/registries/General
Updating git-repo https://github.com/JuliaRegistries/General.git
Resolving package versions...
Updating Project.toml
[no changes]
Testing Example2018
Resolving package versions...
Polynomial{Int64}([1, 2, 3]) = Polynomial{Int64}([1, 2, 3])
evaluate(Polynomial{Int64}([1, 2, 3]), 2) = 17
Polynomial{Float64}([1, 2, 3]) + Polynomial{Float64}([4, 5]) = Polynomial{Float64}([5.0, 7.0, 3.0])
3 * Polynomial{Int64}([1, 2, 4, 42]) = Polynomial{Int64}([3, 6, 12, 126])
deriv(Polynomial{Int64}([1, 1, 1])) = Polynomial{Int64}([2, 3])
Testing Example2018 tests passed

(Example2018) pkg>

```



```

File Edit Options Buffers Tools Help
#####

export Polynomial
struct Polynomial{T}
  coeffs::Vector{T}
end

export evaluate
function evaluate(p::Polynomial, x)
  sum(c * x^(i-1) for (i,c) in enumerate(p.coeffs))
end

function Base. +(p::Polynomial, q::Polynomial)
  l = max(length(p.coeffs), length(q.coeffs))
  rcoeffs = zeros(l)
  for (i,c) in enumerate(p.coeffs)
    rcoeffs[i] += c
  end
  for (i,c) in enumerate(q.coeffs)
    rcoeffs[i] += c
  end
  Polynomial(rcoeffs)
end

function Base. *(a, p::Polynomial)
  Polynomial(map(x->a*x, p.coeffs))
end
Base .*(p::Polynomial, a) = a*p

export deriv
function deriv(p::Polynomial)
  Polynomial([(i-1) * c for (i,c) in enumerate(p.coeffs)][2:end])
end

end

-uu-:---F1 Example2018.jl Bot L52 (Fundamental)-----
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Example2018/src/Example2018.jl

```

```

mple2018
kage versions...
  2, 3]) = Polynomial{Int64}([1, 2, 3])
omial([1, 2, 3]), 2) = 17
  2, 3]) + Polynomial{Float64}([5.0, 7.0, 3.0]

([1, 2, 4, 42]) = Polynomial{Int64}([3, 6, 12, 126])
al([1, 1, 1]) = Polynomial{Int64}([1, 2])
mple2018 tests passed

pkg> test
istry at ~/.julia/registries/General`
-repo `https://github.com/JuliaRegistries/General.git`
kage versions...
object.toml`

mple2018
kage versions...
  2, 3]) = Polynomial{Int64}([1, 2, 3])
omial([1, 2, 3]), 2) = 17
  2, 3]) + Polynomial{Float64}([5.0, 7.0, 3.0]

([1, 2, 4, 42]) = Polynomial{Int64}([3, 6, 12, 126])
al([1, 1, 1, 1, 1]) = Polynomial{Int64}([1, 2, 3, 4])
mple2018 tests passed

pkg> test
istry at ~/.julia/registries/General`
-repo `https://github.com/JuliaRegistries/General.git`
kage versions...
object.toml`

mple2018
kage versions...
  2, 3]) = Polynomial{Int64}([1, 2, 3])
omial([1, 2, 3]), 2) = 17
  2, 3]) + Polynomial{Float64}([5.0, 7.0, 3.0]

([1, 2, 4, 42]) = Polynomial{Int64}([3, 6, 12, 126])
al([1, 1, 1, 1, 1]) = Polynomial{Int64}([1, 2, 3, 4])
al([0, 0, 1, 0, 0]) = Polynomial{Int64}([0, 2, 0, 0])
mple2018 tests passed

pkg> 

```

```
File Edit Options Buffers Tools Help
#####

export Polynomial
struct Polynomial{T}
    coeffs::Vector{T}
end

export evaluate
function evaluate(p::Polynomial, x)
    sum(c * x^(i-1) for (i,c) in enumerate(p.coeffs))
end

function Base. +(p::Polynomial, q::Polynomial)
    l = max(length(p.coeffs), length(q.coeffs))
    rcoeffs = zeros{T}(l)
    for (i,c) in enumerate(p.coeffs)
        rcoeffs[i] += c
    end
    for (i,c) in enumerate(q.coeffs)
        rcoeffs[i] += c
    end
    Polynomial(rcoeffs)
end

function Base. *(a, p::Polynomial)
    Polynomial(map(x->a*x, p.coeffs))
end
Base .*(p::Polynomial, a) = a*p

export deriv
function deriv(p::Polynomial)
    Polynomial([(i-1) * c for (i,c) in enumerate(p.coeffs)][2:end])
end

end

Updating `Project.toml`
[no changes]
Testing Example2018
Resolving package versions...
Polynomial{Int64}([1, 2, 3]) = Polynomial{Int64}([1, 2, 3])
evaluate(Polynomial{Int64}([1, 2, 3]), 2) = 17
Polynomial{Float64}([5.0, 7.0, 3.0]) + Polynomial{Float64}([4.0, 5.0, 1.0]) = Polynomial{Float64}([9.0, 12.0, 4.0])
3 * Polynomial{Int64}([1, 2, 4, 42]) = Polynomial{Int64}([3, 6, 12, 126])
deriv(Polynomial{Int64}([1, 1, 1, 1, 1])) = Polynomial{Int64}([1, 2, 3, 4])
deriv(Polynomial{Int64}([0, 0, 1, 0, 0])) = Polynomial{Int64}([0, 2, 0, 0])
Testing Example2018 tests passed

julia> 2 + 3
5

julia> x
ERROR: UndefVarError: x not defined

julia> x=2
2

julia> :x
:x

julia> typeof(:x)
Symbol

julia> expr = :(2 + 3)
:(2 + 3)

julia> typeof(expr)
Expr

julia> expr.head
:call

julia> expr.args
3-element Array{Any,1}:
 :+
 2
 3

julia>

-uu-:---F1 Example2018.jl Bot L52 (Fundamental)-----
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Example2018/src/Example2018.jl
l
```

```
File Edit Options Buffers Tools Help
```

```
module Deriv2018
```

```
export derivative
```

```
derivative(x::Number, var::Symbol) = 0
```

```
end
```

```
-uuu-:---F1 Deriv2018.jl All L6 (Fundamental)-----
```

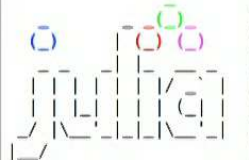
```
using Deriv2018
```

```
@show derivative(3, :x)
```

```
-uuu-:---F1 runtests.jl<2> All L3 (Fundamental)-----
```

```
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Deriv2018/test/runtests.jl
```

```
$ /Applications/Julia-1.0.app/Contents/Resources/julia/bin/julia
```



Documentation: <https://docs.julialang.org>

Type "?" for help, "j?" for Pkg help.

Version 1.0.0 (2018-08-08)

Official <https://julialang.org/> release

```
(v1.0) pkg> activate .
```

```
(Deriv2018) pkg> test
```

```
Updating registry at `~/julia/registries/General`
```

```
Updating git-repo `https://github.com/JuliaRegistries/General.git`
```

```
Resolving package versions...
```

```
Updating `Project.toml`
```

```
[no changes]
```

```
Testing Deriv2018
```

```
Resolving package versions...
```

```
ERROR: LoadError: UndefVarError: deriv not defined
```

```
Stacktrace:
```

```
[1] top-level scope at show.jl:555
```

```
[2] include at ./boot.jl:317 [inlined]
```

```
[3] include_relative(::Module, ::String) at ./loading.jl:1038
```

```
[4] include(::Module, ::String) at ./sysimg.jl:29
```

```
[5] include(::String) at ./client.jl:388
```

```
[6] top-level scope at none:0
```

```
in expression starting at /Users/eschnett/txt/Courses/CompPhys2018/jl/Deriv2018/
```

```
test/runtests.jl:3
```

```
ERROR: Package Deriv2018 errored during testing
```

```
(Deriv2018) pkg> test
```

```
Updating registry at `~/julia/registries/General`
```

```
Updating git-repo `https://github.com/JuliaRegistries/General.git`
```

```
Resolving package versions...
```

```
Updating `Project.toml`
```

```
[no changes]
```

```
Testing Deriv2018
```

```
Resolving package versions...
```

```
derivative(3, :x) = 0
```

```
Testing Deriv2018 tests passed
```

```
(Deriv2018) pkg>
```

```
File Edit Options Buffers Tools Help
module Deriv2018

export derivative

derivative(x::Number, var::Symbol) = 0

derivative(sym::Symbol, var::Symbol) = sym == var ? 1 : 0

function derivative(expr::Expr, var::Symbol)
    if expr.head == :call && expr.args[1] == :+
        return Expr(:call, :+, derivative(expr.args[2:end], var)...)
    end
    @error "not implemented"[]
end

end

-uuu:---F1 Deriv2018.jl All L13 (Fundamental)-----
using Deriv2018

@show derivative(3, :x)
@show derivative(:x, :x)
@show derivative(:y, :x)

[2] include at ./boot.jl:317 [inlined]
[3] include_relative(::Module, ::String) at ./loading.jl:1038
[4] include(::Module, ::String) at ./sysimg.jl:29
[5] include(::String) at ./client.jl:388
[6] top-level scope at none:0
in expression starting at /Users/eschnett/txt/Courses/CompPhys2018/jl/Deriv2018/test/runtests.jl:3
ERROR: Package Deriv2018 errored during testing

(Deriv2018) pkg> test
Updating registry at ~/.julia/registries/General
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Updating `Project.toml`
[no changes]
Testing Deriv2018
Resolving package versions...
derivative(3, :x) = 0
Testing Deriv2018 tests passed

(Deriv2018) pkg> test
Updating registry at ~/.julia/registries/General
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Updating `Project.toml`
[no changes]
Testing Deriv2018
Resolving package versions...
derivative(3, :x) = 0
derivative(:x, :x) = 1
derivative(:y, :x) = 0
Testing Deriv2018 tests passed

julia> @error "oops"
[ Error: oops
  @ Main REPL[3]:1

julia> [1, 2, 4]
3-element Array{Int64,1}:
 1
 2
 4

julia> [1, [2, 4]]
```

```
-uuu:---F1 runtests.jl<2> All L6 (Fundamental)-----
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Deriv2018/src/Deriv2018.jl
```

```

File Edit Options Buffers Tools Help
module Deriv2018

export derivative

derivative(x::Number, var::Symbol) = 0

derivative(sym::Symbol, var::Symbol) = sym == var ? 1 : 0

function derivative(expr::Expr, var::Symbol)
    if expr.head == :call && expr.args[1] == :+
        subexprs = expr.args[2:end]
        res = map(expr -> derivative(expr, var), subexprs)
        Expr(:call, :+, res...)
    end
    @error "not implemented"
end

end

-uu-:---F1 Deriv2018.jl All L13 (Fundamental)-----
using Deriv2018

@show derivative(3, :x)
@show derivative(:x, :x)
@show derivative(:y, :x)
@show derivative(:(2 + 3), :x)
[]

-uu-:---F1 runtests.jl<2> All L7 (Fundamental)-----
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Deriv2018/test/runtests.jl

```

```

julia> expr.head
:call

julia> expr.args
3-element Array{Any,1}:
 :+
  2
 :x

julia> expr.args[2:end]
2-element Array{Any,1}:
  2
 :x

julia> map(expr -> derivative(expr, var), [2,:x])
ERROR: UndefVarError: derivative not defined
Stacktrace:
 [1] (::getfield(Main, Symbol{"##3#4"}))(::Int64) at ./REPL[15]:1
 [2] iterate at ./generator.jl:47 [inlined]
 [3] _collect(::Array{Any,1}, ::Base.Generator{Array{Any,1},getfield(Main, Symbol{"##3#4"}), ::Base.EltypeUnknown, ::Base.HasShape{1}}) at ./array.jl:632
 [4] collect_similar(::Array{Any,1}, ::Base.Generator{Array{Any,1},getfield(Main, Symbol{"##3#4"})}) at ./array.jl:561
 [5] map(::Function, ::Array{Any,1}) at ./abstractarray.jl:1987
 [6] top-level scope at none:0

(Deriv2018) pkg> test
Updating registry at `~/julia/registries/General`
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Updating `Project.toml`
[no changes]
Testing Deriv2018
Resolving package versions...
derivative(3, :x) = 0
derivative(:x, :x) = 1
derivative(:y, :x) = 0
[ Error: not implemented
 @ Deriv2018 ~/txt/Courses/CompPhys2018/jl/Deriv2018/src/Deriv2018.jl:15
derivative($(Expr(:quote, :(2 + 3))), :x) = nothing
Testing Deriv2018 tests passed

(Deriv2018) pkg>

```

```

File Edit Options Buffers Tools Help
module Deriv2018

export derivative

derivative(x::Number, var::Symbol) = 0

derivative(sym::Symbol, var::Symbol) = sym == var ? 1 : 0

function derivative(expr::Expr, var::Symbol)
    if expr.head == :call && expr.args[1] == :+
        subexprs = expr.args[2:end]
        res = map(expr -> derivative(expr, var), subexprs)
        return Expr(:call, :+, res...)
    end
    @error "not implemented"
end

end

---uu-:---F1 Deriv2018.jl All L13 (Fundamental)-----
using Deriv2018

@show derivative(3, :x)
@show derivative(:x, :x)
@show derivative(:y, :x)
@show derivative(:(2 + 3), :x)
@show derivative(:(2 + x + a), :x)

```

```

---uuu-:---F1 runtests.jl<2> All L8 (Fundamental)-----
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Deriv2018/test/runtests.jl

```

```

-repo `https://github.com/JuliaRegistries/General.git`
kage versions...
object.toml`

iv2018
kage versions...
:x) = 0
:x) = 1
:x) = 0
mplemented
~/txt/Courses/CompPhys2018/jl/Deriv2018/src/Deriv2018.jl:15
xpr(:quote, :(2 + 3)), :x) = nothing
iv2018 tests passed

g> test
istry at `~/julia/registries/General`
-repo `https://github.com/JuliaRegistries/General.git`
kage versions...
object.toml`

iv2018
kage versions...
:x) = 0
:x) = 1
:x) = 0
xpr(:quote, :(2 + 3)), :x) = :(0 + 0)
iv2018 tests passed

g> test
istry at `~/julia/registries/General`
-repo `https://github.com/JuliaRegistries/General.git`
kage versions...
object.toml`

iv2018
kage versions...
:x) = 0
:x) = 1
:x) = 0
xpr(:quote, :(2 + 3)), :x) = :(0 + 0)
xpr(:quote, :(2 + x + a)), :x) = :(0 + 1 + 0)
iv2018 tests passed

g>

```

```

File Edit Options Buffers Tools Help
module Deriv2018

export derivative

derivative(x::Number, var::Symbol) = 0

derivative(sym::Symbol, var::Symbol) = sym == var ? 1 : 0

function derivative(expr::Expr, var::Symbol)
    if expr.head == :call && expr.args[1] == :+
        subexprs = expr.args[2:end]
        res = map(expr -> derivative(expr, var), subexprs)
        return Expr(:call, :+, res...)
    end
    @error "not implemented"
end

end

-uuu:---F1 Deriv2018.jl All L13 (Fundamental)-----
using Deriv2018

@show derivative(3, :x)
@show derivative(:x, :x)
@show derivative(:y, :x)
@show derivative(:(2 + 3), :x)
@show derivative(:(2 + x + a), :x)
[]

-uuu:---F1 runtests.jl<2> All L8 (Fundamental)-----
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Deriv2018/test/runtests.jl

```

```

Resolving package versions...
Updating `Project.toml`
[no changes]
Testing Deriv2018
Resolving package versions...
derivative(3, :x) = 0
derivative(:x, :x) = 1
derivative(:y, :x) = 0
[Error: not implemented
 @ Deriv2018 ~/txt/Courses/CompPhys2018/jl/Deriv2018/src/Deriv2018.jl:15
derivative($(Expr(:quote, :(2 + 3))), :x) = nothing
Testing Deriv2018 tests passed

(Deriv2018) pkg> test
Updating registry at `~/julia/registries/General`
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Updating `Project.toml`
[no changes]
Testing Deriv2018
Resolving package versions...
derivative(3, :x) = 0
derivative(:x, :x) = 1
derivative(:y, :x) = 0
derivative($(Expr(:quote, :(2 + 3))), :x) = :(0 + 0)
Testing Deriv2018 tests passed

(Deriv2018) pkg> test
Updating registry at `~/julia/registries/General`
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Updating `Project.toml`
[no changes]
Testing Deriv2018
Resolving package versions...
derivative(3, :x) = 0
derivative(:x, :x) = 1
derivative(:y, :x) = 0
derivative($(Expr(:quote, :(2 + 3))), :x) = :(0 + 0)
derivative($(Expr(:quote, :(2 + x + a))), :x) = :(0 + 1 + 0)
Testing Deriv2018 tests passed

julia> eval(:(0 + 1 + 0))
0

```

```

File Edit Options Buffers Tools Help
module Deriv2018

export derivative

derivative(x::Number, var::Symbol) = 0

derivative(sym::Symbol, var::Symbol) = sym == var ? 1 : 0

function derivative(expr::Expr, var::Symbol)
    if expr.head == :call && expr.args[1] == :+
        subexprs = expr.args[2:end]
        res = map(expr -> derivative(expr, var), subexprs)
        return Expr(:call, :+, res...)
    end
    @error "not implemented"
end

end

---uu:---F1 Deriv2018.jl All L13 (Fundamental)-----
using Deriv2018

@show derivative(3, :x)
@show derivative(:x, :x)
@show derivative(:y, :x)
@show derivative(:(2 + 3), :x)
@show derivative(:(2 + x + a), :x)
[]

---uu:---F1 runtests.jl<2> All L8 (Fundamental)-----
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Deriv2018/test/runtests.jl

Resolving package versions...
derivative(3, :x) = 0
derivative(:x, :x) = 1
derivative(:y, :x) = 0
derivative($(Expr(:quote, :(2 + 3))), :x) = :(0 + 0)
Testing Deriv2018 tests passed

(Deriv2018) pkg> test
Updating registry at `~/julia/registries/General`
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Updating `Project.toml`
[no changes]
Testing Deriv2018
Resolving package versions...
derivative(3, :x) = 0
derivative(:x, :x) = 1
derivative(:y, :x) = 0
derivative($(Expr(:quote, :(2 + 3))), :x) = :(0 + 0)
derivative($(Expr(:quote, :(2 + x + a))), :x) = :(0 + 1 + 0)
Testing Deriv2018 tests passed

julia> eval(:(0 + 1 + 0))
1

julia> xs = [1,2,3,4]
4-element Array{Int64,1}:
 1
 2
 3
 4

julia> xs = [1,2]
2-element Array{Int64,1}:
 1
 2

julia> A = [1 2;
            3 4]
2x2 Array{Int64,2}:
 1 2
 3 4

julia>

```

```

File Edit Options Buffers Tools Help
module Deriv2018

export derivative

derivative(x::Number, var::Symbol) = 0

derivative(sym::Symbol, var::Symbol) = sym == var ? 1 : 0

function derivative(expr::Expr, var::Symbol)
    if expr.head == :call && expr.args[1] == :+
        subexprs = expr.args[2:end]
        res = map(expr -> derivative(expr, var), subexprs)
        return Expr(:call, :+, res...)
    end
    @error "not implemented"
end

end

-uu-:---F1 Deriv2018.jl All L13 (Fundamental)-----
using Deriv2018

@show derivative(3, :x)
@show derivative(:x, :x)
@show derivative(:y, :x)
@show derivative(:(2 + 3), :x)
@show derivative(:(2 + x + a), :x)
[]

-uu-:---F1 runtests.jl<2> All L8 (Fundamental)-----
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Deriv2018/test/runtests.jl

julia> xs = [1,2]
2-element Array{Int64,1}:
 1
 2

julia> A = [1 2;
           3 4]
2x2 Array{Int64,2}:
 1 2
 3 4

julia> A * xs
2-element Array{Int64,1}:
 5
11

julia> A'
2x2 LinearAlgebra.Adjoint{Int64,Array{Int64,2}}:
 1 3
 2 4

julia> xs * A
ERROR: DimensionMismatch("matrix A has dimensions (2,1), matrix B has dimensions (2,2)")
Stacktrace:
 [1] _generic_matmatmul!(::Array{Int64,2}, ::Char, ::Char, ::Array{Int64,2}, ::Array{Int64,2}) at /Users/osx/buildbot/slave/package_osx64/build/usr/share/julia/stdlib/v1.0/LinearAlgebra/src/matmul.jl:588
 [2] generic_matmatmul!(::Array{Int64,2}, ::Char, ::Char, ::Array{Int64,2}, ::Array{Int64,2}) at /Users/osx/buildbot/slave/package_osx64/build/usr/share/julia/stdlib/v1.0/LinearAlgebra/src/matmul.jl:578
 [3] mul! at /Users/osx/buildbot/slave/package_osx64/build/usr/share/julia/stdlib/v1.0/LinearAlgebra/src/matmul.jl:172 [inlined]
 [4] * at /Users/osx/buildbot/slave/package_osx64/build/usr/share/julia/stdlib/v1.0/LinearAlgebra/src/matmul.jl:141 [inlined]
 [5] *(::Array{Int64,1}, ::Array{Int64,2}) at /Users/osx/buildbot/slave/package_osx64/build/usr/share/julia/stdlib/v1.0/LinearAlgebra/src/matmul.jl:62
 [6] top-level scope at none:0

julia> xs' * A
1x2 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 7 10

julia>

```

```

File Edit Options Buffers Tools Help
module Deriv2018

export derivative

derivative(x::Number, var::Symbol) = 0

derivative(sym::Symbol, var::Symbol) = sym == var ? 1 : 0

function derivative(expr::Expr, var::Symbol)
    if expr.head == :call && expr.args[1] == :+
        subexprs = expr.args[2:end]
        res = map(expr -> derivative(expr, var), subexprs)
        return Expr(:call, :+, res...)
    end
    @error "not implemented"
end

end

-uu-:---F1 Deriv2018.jl All L13 (Fundamental)-----
using Deriv2018

@show derivative(3, :x)
@show derivative(:x, :x)
@show derivative(:y, :x)
@show derivative(:(2 + 3), :x)
@show derivative(:(2 + x + a), :x)
[]

-uu-:---F1 runtests.jl<2> All L8 (Fundamental)-----
Wrote /Users/eschnett/txt/Courses/CompPhys2018/jl/Deriv2018/test/runtests.jl

arg{Any,N} where N} where A<:DenseArray where N where T, DenseArray}, SubArray{T
<:Union{Complex{Float32}, Complex{Float64}, Float32, Float64},2,A,I,L} where L w
here I<:Tuple{Vararg{Union{Int64, AbstractRange{Int64}, AbstractCartesianIndex},
N} where N} where A<:Union{ReinterpretArray{T,N,S,A} where S where A<:Union{SubA
rray{T,N,A,I,true} where I<:Tuple{AbstractUnitRange,Vararg{Any,N} where N} where
A<:DenseArray where N where T, DenseArray} where N where T, ReshapedArray{T,N,A
,MI} where MI<:Tuple{Vararg{SignedMultiplicativeInverse{Int64},N} where N} where
A<:Union{ReinterpretArray{T,N,S,A} where S where A<:Union{SubArray{T,N,A,I,true
} where I<:Tuple{AbstractUnitRange,Vararg{Any,N} where N} where A<:DenseArray wh
ere N where T, DenseArray} where N where T, SubArray{T,N,A,I,true} where I<:Tupl
e{AbstractUnitRange,Vararg{Any,N} where N} where A<:DenseArray where N where T,
DenseArray} where N where T, DenseArray}} where #s549, ::Union{DenseArray{S,1},
ReinterpretArray{S,1,S,A} where S where A<:Union{SubArray{T,N,A,I,true} where I<
:Tuple{AbstractUnitRange,Vararg{Any,N} where N} where A<:DenseArray where N wher
e T, DenseArray}, ReshapedArray{S,1,A,MI} where MI<:Tuple{Vararg{SignedMultiplic
ativeInverse{Int64},N} where N} where A<:Union{ReinterpretArray{T,N,S,A} where S
where A<:Union{SubArray{T,N,A,I,true} where I<:Tuple{AbstractUnitRange,Vararg{A
ny,N} where A<:DenseArray where N where T, DenseArray} where N where T, DenseArr
ay{T,N,A,I,true} where I<:Tuple{AbstractUnitRange,Vararg{Any,N} where N} where A<:DenseArray where N where T, DenseArray}, SubArray{S,1,A,I,L} where L wh
ere I<:Tuple{Vararg{Union{Int64, AbstractRange{Int64}, AbstractCartesianIndex},N
} where N} where A<:Union{ReinterpretArray{T,N,S,A} where S where A<:Union{SubAr
ray{T,N,A,I,true} where I<:Tuple{AbstractUnitRange,Vararg{Any,N} where N} where
A<:DenseArray where N where T, DenseArray} where N where T, ReshapedArray{T,N,A,
MI} where MI<:Tuple{Vararg{SignedMultiplicativeInverse{Int64},N} where N} where
A<:Union{ReinterpretArray{T,N,S,A} where S where A<:Union{SubArray{T,N,A,I,true}
where I<:Tuple{AbstractUnitRange,Vararg{Any,N} where N} where A<:DenseArray wher
e N where T, DenseArray} where N where T, SubArray{T,N,A,I,true} where I<:Tupl
e{AbstractUnitRange,Vararg{Any,N} where N} where A<:DenseArray where N where T, D
enseArray} where N where T, DenseArray}} where {T<:Union{Complex{Float32}, Comp
lex{Float64}, Float32, Float64}, S} at /Users/osx/buildbot/slave/package_osx64/b
uild/usr/share/julia/stdlib/v1.0/LinearAlgebra/src/matmul.jl:97
*(:LinearAlgebra.Adjoint{#s549,#s548} where #s548<:LinearAlgebra.AbstractTria
ngular where #s549, ::AbstractArray{T,1} where T) at /Users/osx/buildbot/slave/p
ackage_osx64/build/usr/share/julia/stdlib/v1.0/LinearAlgebra/src/triangular.jl:1
805
...
Stacktrace:
 [1] top-level scope at none:0

julia> xs' * xs
5

julia>

```