Variables

```
a=2
```

```
2
```

```
c=8
```

```
8
```

```
abc=1
```

```
1
```

```
1abc=2
```

```
syntax: "1" is not a valid function argument name around In[4]:1

Stacktrace:
 [1] top-level scope
   @ In[4]:1
 [2] eval
   @ .\boot.jl:360 [inlined]
 [3] include_string(mapexpr::typeof(REPL.softscope), mod::Module,
code::String, filename::String)
   @ Base .\loading.jl:1116
```

```
typeof(a)
```

```
Int64
```

```
π
```

```
π = 3.1415926535897...
```

```
π_2=π/2
```

```
1.5707963267948966
```

```
c=8
```

```
8
```

Definition Vs Initialization

```
i::Int=1
```

```
syntax: type declarations on global variables are not yet supported

Stacktrace:
 [1] top-level scope
   @ In[9]:1
 [2] eval
```

```
   @ .\boot.jl:360 [inlined]
 [3] include_string(mapexpr::typeof(REPL.softscope), mod::Module, 
code::String, filename::String)
   @ Base .\loading.jl:1116
```

```julia
function f()
    i
    return i
end
```

```
f (generic function with 1 method)
```

```julia
f()
```

```
UndefVarError: i not defined

Stacktrace:
 [1] f()
   @ Main .\In[10]:2
 [2] top-level scope
   @ In[11]:1
 [3] eval
   @ .\boot.jl:360 [inlined]
 [4] include_string(mapexpr::typeof(REPL.softscope), mod::Module, 
code::String, filename::String)
   @ Base .\loading.jl:1116
```

Constants

```julia
const ICONSTANT=1
```

```
1
```

```julia
ICONSTANT=5
```

```
WARNING: redefinition of constant ICONSTANT. This may fail, cause 
incorrect answers, or produce other errors.

5
```

```julia
ICONSTANT
```

```
5
```

Liberals

```julia
#Integer
2
```

```
2
```

```julia
2.0 #double precision float(float64)
```

```
2.0
```

```
2f0 #double precision float(float32)
```

```
2.0f0
```

```
0.2f0
```

```
0.2f0
```

```
2f00
```

```
2.0f0
```

```
4.5
```

```
4.5
```

```
"shalom"
```

```
"shalom"
```

```
2=4
```

```
syntax: invalid assignment location "2" around In[22]:1

Stacktrace:
 [1] top-level scope
   @ In[22]:1
 [2] eval
   @ .\boot.jl:360 [inlined]
 [3] include_string(mapexpr::typeof(REPL.softscope), mod::Module, 
code::String, filename::String)
   @ Base .\loading.jl:1116
```

```
2π
```

```
6.283185307179586
```

```
"string" #string
```

```
"string"
```

```
'a' #char
```

```
'a': ASCII/Unicode U+0061 (category Ll: Letter, lowercase)
```

Tuples

```
i=1
j=2
i,j=j,1
```

```
(2, 1)
```

```
j
```

```
1

i

2

1,2

(1, 2)

a= (2,3)

(2, 3)

typeof(a)

Tuple{Int64, Int64}

a[1]

2

a[1]=5

MethodError: no method matching setindex!(::Tuple{Int64,
Int64}, ::Int64, ::Int64)

Stacktrace:
 [1] top-level scope
   @ In[38]:1
 [2] eval
   @ .\boot.jl:360 [inlined]
 [3] include_string(mapexpr::typeof(REPL.softscope), mod::Module,
code::String, filename::String)
   @ Base .\loading.jl:1116

i,j=1,2

(1, 2)

i

1

j

2

i=1,2 #i is a tuple

(1, 2)

i,=1,2 #i is an Int

(1, 2)
```

```
i
```

```
1
```

```
i,j=1,2,3 #i and j get assigned 3 is ignored
```

```
(1, 2, 3)
```

```
i
```

```
1
```

```
j
```

```
2
```

Built in types

Nothing

```
typeof(nothing)
```

```
Nothing
```

```
a=nothing
i=5
if i<5
    a=5
end
5
typeof(a)
```

```
Nothing
```

Numeric Types

Bool-True & False

Integral types-Int8,Int16,...

```
typeof(1)
```

```
Int64
```

```
typeof(0b1),typeof(0o7),typeof(0xff)
```

```
(UInt8, UInt8, UInt8)
```

```
typeof(0xf),typeof(0xfff),typeof(0xffff),typeof(0xfffff),typeof(0xffff
ff)
```

```
(UInt8, UInt16, UInt16, UInt32, UInt32)
```

Floating point numbers

```
typeof(1.0), typeof(1e0), typeof(1.e4)
```

```
(Float64, Float64, Float64)
```

```
typeof(1.0f0),typeof(1f-6),typeof(1.f4)
```

```
(Float32, Float32, Float32)
```

Abstract types

```
abstract type MyAbstractType end
struct MyConcreteType <:MyAbstractType
    member
end
```

```
a=MyConcreteType(5)
```

```
MyConcreteType(5)
```

```
a isa MyAbstractType
```

```
true
```

Primitive Types

```
UInt32(3f-1)
```

```
InexactError: UInt32(0.3)

Stacktrace:
 [1] UInt32(x::Float32)
   @ Base .\float.jl:702
 [2] top-level scope
   @ In[63]:1
 [3] eval
   @ .\boot.jl:360 [inlined]
 [4] include_string(mapexpr::typeof(REPL.softscope), mod::Module,
code::String, filename::String)
   @ Base .\loading.jl:1116
```

```
primitive type MyType1 40 end
```

```
primitive type MyType2 4 end
```

```
invalid number of bits in primitive type MyType2

Stacktrace:
 [1] top-level scope
   @ In[65]:1
 [2] eval
   @ .\boot.jl:360 [inlined]
 [3] include_string(mapexpr::typeof(REPL.softscope), mod::Module,
code::String, filename::String)
   @ Base .\loading.jl:1116
```

Bit Types

```
a=10
```

```
10
```

```
isbits(a)
```

```
true
```

```
isbitstype(Int)
```

```
true
```

Rational and complex

Char:one unicode,

string:'',""....

User Defined Types

Struct

```
struct Rectangle
    h::Float64
    w::Float64
end
```

```
r=Rectangle(10.0,20.0)
```

```
Rectangle(10.0, 20.0)
```

Mutable struct

```
mutable struct MRectangle
    h::Float64
    w::Float64
end
```

```
mr= MRectangle(10.0,20.0)
```

```
MRectangle(10.0, 20.0)
```

```
mr.h+15.0
```

```
25.0
```

```
mr
```

```
MRectangle(10.0, 20.0)
```

```julia
abstract type Shape end
struct Rectangle1 <: Shape
    w::Float64
    h::Float64
end
struct Square <: Shape
    l::Float64
end
```

Members

```julia
mutable struct A
    member
end
```

```julia
a=A(5)
```

```
A(5)
```

```julia
typeof(a.member)
```

```
Int64
```

```julia
a1=A("string")
```

```
A("string")
```

```julia
typeof(a1.member)
```

```
String
```

Any

```julia
a.member="string"
```

```
"string"
```

```julia
typeof(a.member)
```

```
String
```

Parametric data types

```julia
Rational{Any}
```

```
TypeError: in Rational, in T, expected T<:Integer, got Type{Any}

Stacktrace:
 [1] top-level scope
   @ In[86]:1
 [2] eval
   @ .\boot.jl:360 [inlined]
 [3] include_string(mapexpr::typeof(REPL.softscope), mod::Module,
code::String, filename::String)
   @ Base .\loading.jl:1116
```

```
Rational{Int32} <: Rational
```

```
true
```

```
Rational{Int32} <:Rational{Integer}
```

```
false
```

Abstract types can be parametric as well.

```
abstract type ShapeParametric{T<:AbstractFloat} end
```

```
struct RectangleParametric{T<:AbstractFloat} <: ShapeParametric{T}
     w::T
     h::T
end
struct SquareParametric{T<:AbstractFloat} <: ShapeParametric{T}
    s::T
end
```

```
struct Point{T<:AbstractFloat, N}
    x::Vector{T}
end
p = Point{Float32, 2}([1f0, 2f0])
```

```
Point{Float32, 2}(Float32[1.0, 2.0])
```

Operations of type

```
a=1//2 #typeof
typeof(a)
```

```
Rational{Int64}
```

```
typeof(Int) #aliases
```

```
DataType
```

typesof(Int

```
typeof(Rational{Int})
```

```
DataType
```

```
isa(1,Number) #isa
```

```
true
```

```
isa(1,Matrix)
```

```
false
```

```
isa(1,Int)
```

```
true
```

```
1 isa Number
```

```
true
```

```
supertype(Int32) #supertype
```

```
Signed
```

```
Int32 <: Integer
```

```
true
```

```
Int32 <: AbstractFloat
```

```
false
```

```
Int32 <: Real
```

```
true
```

```
Int32 <: Signed
```

```
true
```

Printing Data Types

```
struct AA
    a1::Int32
    a2::Float64
end
a = AA(1, 2)
```

```
AA(1, 2.0)
```

```
a;
```

```
a
```

```
AA(1, 2.0)
```

```
b=2.0
```

```
2.0
```

```
a;b
```

```
2.0
```

```
a
```

```
AA(1, 2.0)
```

```
a;
```

```
a;nothing
```

```
struct AAA       #show
    a1::Int32
    a2::Float64
end
a = AAA(1, 2)
```

```
AAA(1, 2.0)
```

```
function Base.show(io::IO, a::AAA)
    println(io, "a1: ", a.a1, " a2: ", a.a2)
end
```

```
a
```

```
a1: 1 a2: 2.0
```

```
print(a) #print
```

```
a1: 1 a2: 2.0
```

```
string(a)  #string
```

```
"a1: 1 a2: 2.0\n"
```