Compound Statements

```
 z= begin     #3begin.....end
    x=1
    y=2
    x+y
end
```

```
3
```

```
z = (x = 1; y = 2; x+y)
```

```
3
```

```
z =(x = 1;
    y = 2;
    x + y)
```

```
3
```

Conditional Execution

```
if 1
    println("I am here")
end
```

```
TypeError: non-boolean (Int64) used in boolean context

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
```

```
if Bool(1)
    println("I am here")
end
```

```
if Bool(1)
    println("I am here")
end
```

```
I am here
I am here
```

```
str = i > 1 ? "Greater" : "Less"
```

```
str = i > 1 ? "Greater" : "Less"
```

```
str = i > 1 ? "Greater" : "Less"
```

```
UndefVarError: i not defined

Stacktrace:
 [1] top-level scope
   @ In[6]:1
 [2] eval
   @ .\boot.jl:360 [inlined]
 [3] include_string(mapexpr::typeof(REPL.softscope), mod::Module,
code::String, filename::String)
   @ Base .\loading.jl:1116
```

```
val = 3
if val == 1
    "one"
elseif val == 2
    "two"
elseif val == 3
    "three"
elseif val == 4
    "four"
else
    "unknown"
end
```

```
"three"
```

```
val = 3                 #else if vs elseif
In [1]:
if val == 1
    "one"
elseif val == 2
    "two"
elseif val == 3
    "three"
elseif val == 4
    "four"
else if val >= 5

if val == 1
    "one"
elseif val == 2
    "two"
elseif val == 3
    "three"
elseif val == 4
    "four"
else
```

```
    "unknown"
end
```

```
syntax: space before "[" not allowed in "In [" at In[8]:2

Stacktrace:
 [1] top-level scope
   @ In[8]:2
 [2] eval
   @ .\boot.jl:360 [inlined]
 [3] include_string(mapexpr::typeof(REPL.softscope), mod::Module,
code::String, filename::String)
   @ Base .\loading.jl:1116
```

```
if val == 1
    "one"
elseif val == 2
    "two"
elseif val == 3
    "three"
elseif val == 4
    "four"
else
    if val >= 5
        "above five"
    else
        "below five"
    end
end
```

```
"three"
```

```
begin                          #unstructured branching
    s = 0
    n = 10
@label loop
    s = s + n
    n = n - 1
    if n > 0
        @goto loop
    end
    s
end
```

```
55
```

```
s = 0
for i = 1:10
    s = s + i
end
s
```

```
55
```

Iterative Execution

```
s = 0
for i = 1:10
    s = s + i
end
s
```

```
55
```

```
s = 0;
for i = 1:2:10
    println(i)
    s = s + i
end
s
```

```
1
3
5
7
9

25
```

```
s = 0;                      #continue and break
for i = 1:10
    if i % 3 == 0
        continue
    end
    println(i)
    s = s + i
end
s
```

```
1
2
4
5
7
8
10

37
```

```
s = 0;                 #continue and break
for i = 1:10
    if i % 3 == 0
        break
    end
```

```
    println(i)
    s = s + i
end
s
```

```
1
2

3
```

```
for i in [5 10 15]  #for...in
    println(i)
end
```

```
5
10
15
```

```
for i=1:3, j=1:2  #multiple range objects
    println((i, j))
end
```

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

```
for i=1:3, j=1:i
    println((i, j))
end
```

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

```
for i=1:3, j=1:2
    println((i, j))
    if i == j
        break
    end
end
```

```
(1, 1)
```

```
for i=1:3
    for j=1:2
```

```
        println((i, j))
        if i == j
            break
        end
    end
end
```

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

```
s, n = 0, 10;   #while
while n > 0
    s = s + n
    n = n - 1
end
s
```

```
55
```

Exception handling

```
try             #try....catch
    sqrt(-1)
catch e
    println(e)
end
```

```
DomainError(-1.0, "sqrt will only return a complex result if called
with a complex argument. Try sqrt(Complex(x)).")
```

```
sqrt(-1)
```

```
DomainError with -1.0:
sqrt will only return a complex result if called with a complex
argument. Try sqrt(Complex(x)).

Stacktrace:
 [1] throw_complex_domainerror(f::Symbol, x::Float64)
   @ Base.Math .\math.jl:33
 [2] sqrt
   @ .\math.jl:582 [inlined]
 [3] sqrt(x::Int64)
   @ Base.Math .\math.jl:608
 [4] top-level scope
   @ In[23]:1
 [5] eval
   @ .\boot.jl:360 [inlined]
 [6] include_string(mapexpr::typeof(REPL.softscope), mod::Module,
```

```
code::String, filename::String)
   @ Base .\loading.jl:1116
```

```
try
    sqrt(-1)
catch e
    rethrow()
end
```

```
DomainError with -1.0:
sqrt will only return a complex result if called with a complex
argument. Try sqrt(Complex(x)).

Stacktrace:
 [1] throw_complex_domainerror(f::Symbol, x::Float64)
   @ Base.Math .\math.jl:33
 [2] sqrt
   @ .\math.jl:582 [inlined]
 [3] sqrt(x::Int64)
   @ Base.Math .\math.jl:608
 [4] top-level scope
   @ In[24]:2
 [5] eval
   @ .\boot.jl:360 [inlined]
 [6] include_string(mapexpr::typeof(REPL.softscope), mod::Module,
code::String, filename::String)
   @ Base .\loading.jl:1116
```

```
try
    throw(1)
catch e
    println(typeof(e))
end
```

```
Int64
```

```
f = open("file")   #finally
try
    b = read(f)
finally
    close(f)
end
```

```
SystemError: opening file "file": No such file or directory

Stacktrace:
 [1] systemerror(p::String, errno::Int32; extrainfo::Nothing)
   @ Base .\error.jl:168
 [2] #systemerror#62
   @ .\error.jl:167 [inlined]
 [3] systemerror
```

```
   @ .\error.jl:167 [inlined]
 [4] open(fname::String; lock::Bool, read::Nothing, write::Nothing,
create::Nothing, truncate::Nothing, append::Nothing)
   @ Base .\iostream.jl:293
 [5] open(fname::String)
   @ Base .\iostream.jl:282
 [6] top-level scope
   @ In[26]:1
 [7] eval
   @ .\boot.jl:360 [inlined]
 [8] include_string(mapexpr::typeof(REPL.softscope), mod::Module,
code::String, filename::String)
   @ Base .\loading.jl:1116
```

```
sqrt(-1)   #information from exception
```

```
DomainError with -1.0:
sqrt will only return a complex result if called with a complex
argument. Try sqrt(Complex(x)).

Stacktrace:
 [1] throw_complex_domainerror(f::Symbol, x::Float64)
   @ Base.Math .\math.jl:33
 [2] sqrt
   @ .\math.jl:582 [inlined]
 [3] sqrt(x::Int64)
   @ Base.Math .\math.jl:608
 [4] top-level scope
   @ In[27]:1
 [5] eval
   @ .\boot.jl:360 [inlined]
 [6] include_string(mapexpr::typeof(REPL.softscope), mod::Module,
code::String, filename::String)
   @ Base .\loading.jl:1116
```

```
try            #stacktraces
    sqrt(-1)
catch e
    stacktrace(catch_backtrace())
end
```

```
12-element Vector{Base.StackTraces.StackFrame}:
 throw_complex_domainerror(f::Symbol, x::Float64) at math.jl:33
 sqrt at math.jl:582 [inlined]
 sqrt(x::Int64) at math.jl:608
 top-level scope at In[28]:2
 eval at boot.jl:360 [inlined]
 include_string(mapexpr::typeof(REPL.softscope), mod::Module,
code::String, filename::String) at loading.jl:1116
 softscope_include_string(m::Module, code::String, filename::String)
at SoftGlobalScope.jl:65
```

```
 execute_request(socket::ZMQ.Socket, msg::IJulia.Msg) at
execute_request.jl:67
 #invokelatest#2 at essentials.jl:708 [inlined]
 invokelatest at essentials.jl:706 [inlined]
 eventloop(socket::ZMQ.Socket) at eventloop.jl:8
 (::IJulia.var"#15#18")() at task.jl:417
```