A Remarkably Groovy Hands on Lab

Dave Schleis & Joe Aultman

of

Insum Solutions & interRel Consulting

(respectively)

follow us at

@dschleis & @joeaultman


WELCOME

KSCOPE16 EPM CARNIVAL

JOIN US!

GRAB A DRINK, WHILE NETWORKING WITH SOME FRIENDS, OLD AND NEW!

MAKE SOME MEMORIES IN THE SELFIE BOOTH OR GRAB YOUR PHONE AND TAKE A SELFIE WITH YOUR FAVORITE EPM PRO!

GIVE YOUR MIND A REST AND TEST OUT YOUR MUSCLES ON THE 11 FOOT HIGH STRIKER.

SHOOT SOME HOOPS OR JUST RELAX AND ENJOY THE 8 FOOT TALL MAGICIAN. NO, HE'S NOT AN EPM MAGICIAN ;-)

TEST YOUR MIND AND HYPERION SKILLS IN THE EPM RUBIK'S CUBE QUEST!

MONDAY JUNE 27, 2016 | 8 - 10PM
IN ROOM CHICAGO X.

DON'T MISS IT!!!

Join Your Community!

Tonight from 8:00 – 10:00 in Sheraton II

D
Database

Networking Hopportunity

Unwind and network with the Database Community. Get there early to enjoy a relaxing evening with your Database peers. Participants will sample some specially chosen microbrews and walk away with a great gift and lots of new connections. Don’t miss out!

insum

- North America’s largest APEX consulting firm
- 12 years working with Oracle APEX
- 80 employees and growing
- Committed to APEX innovation
- Our services: application development, coaching, consulting, EBS extensions, and Forms migration
- Visit insum.ca for more information!

ORACLE Gold Partner

ODTUG

OAUG oracle applications users group

interRel: Oracle's BI/EPM Partner of the Year

- **3-Time Winner of Oracle's BI & EPM *Partner of the Year***
- ***Six*** Oracle ACE Directors & Associates
- **10+** Best selling books on Hyperion, Essbase, & Oracle EPM Cloud
- **100+** EPM Cloud specializations
- **10+** Active Cloud implementations including
  - **1st ARCS** Implementation
  - **1st EPBCS** Implementation
  - **1st FCCS** Implementation
- Oracle Platinum & Oracle Cloud Partner
- 8-Time Inc. 5000 Fastest Growing company in USA

Founded in 1997,
we are the longest-standing, Oracle EPM/BI-dedicated partner *in the world*

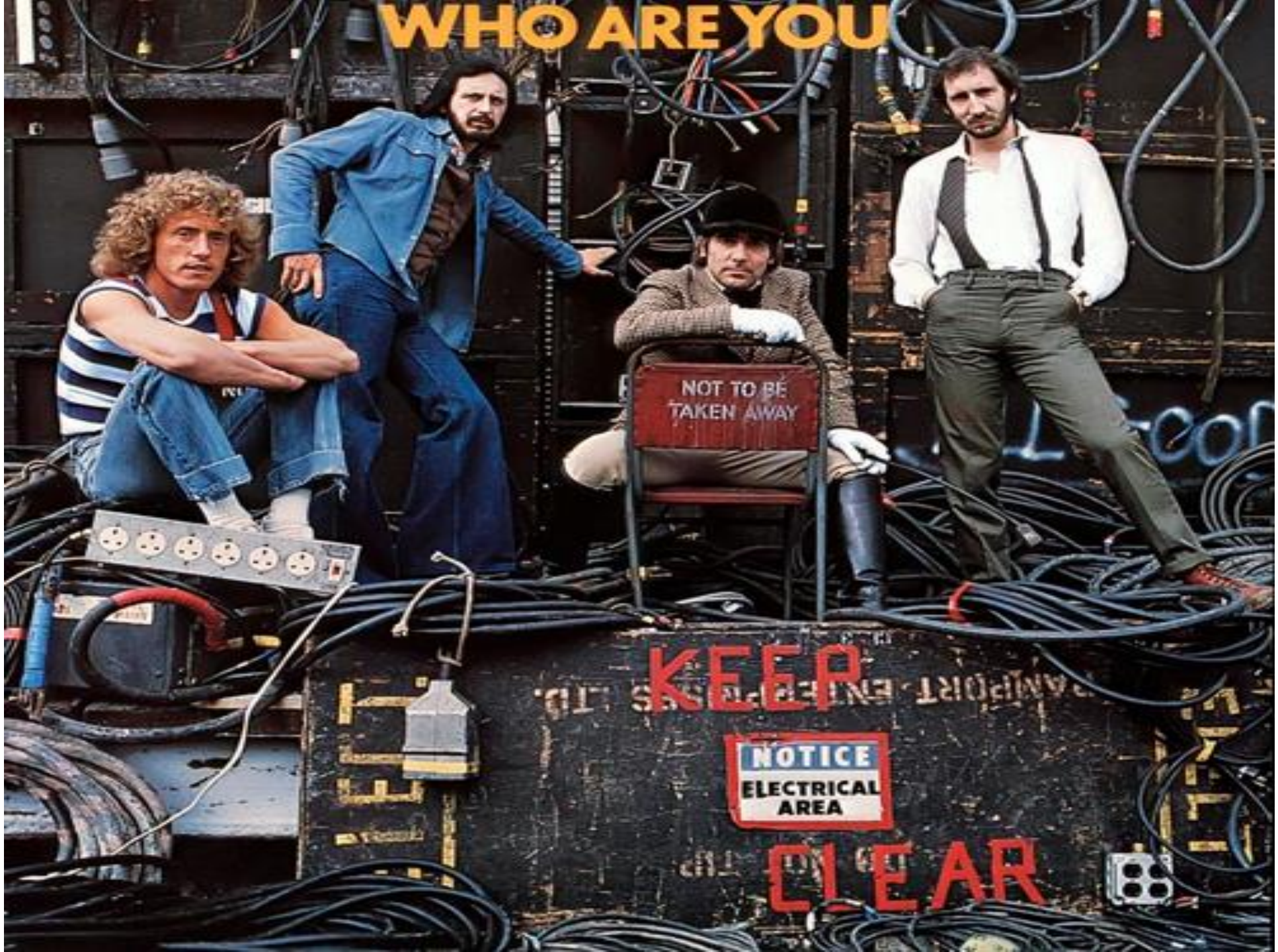
About the Presenter

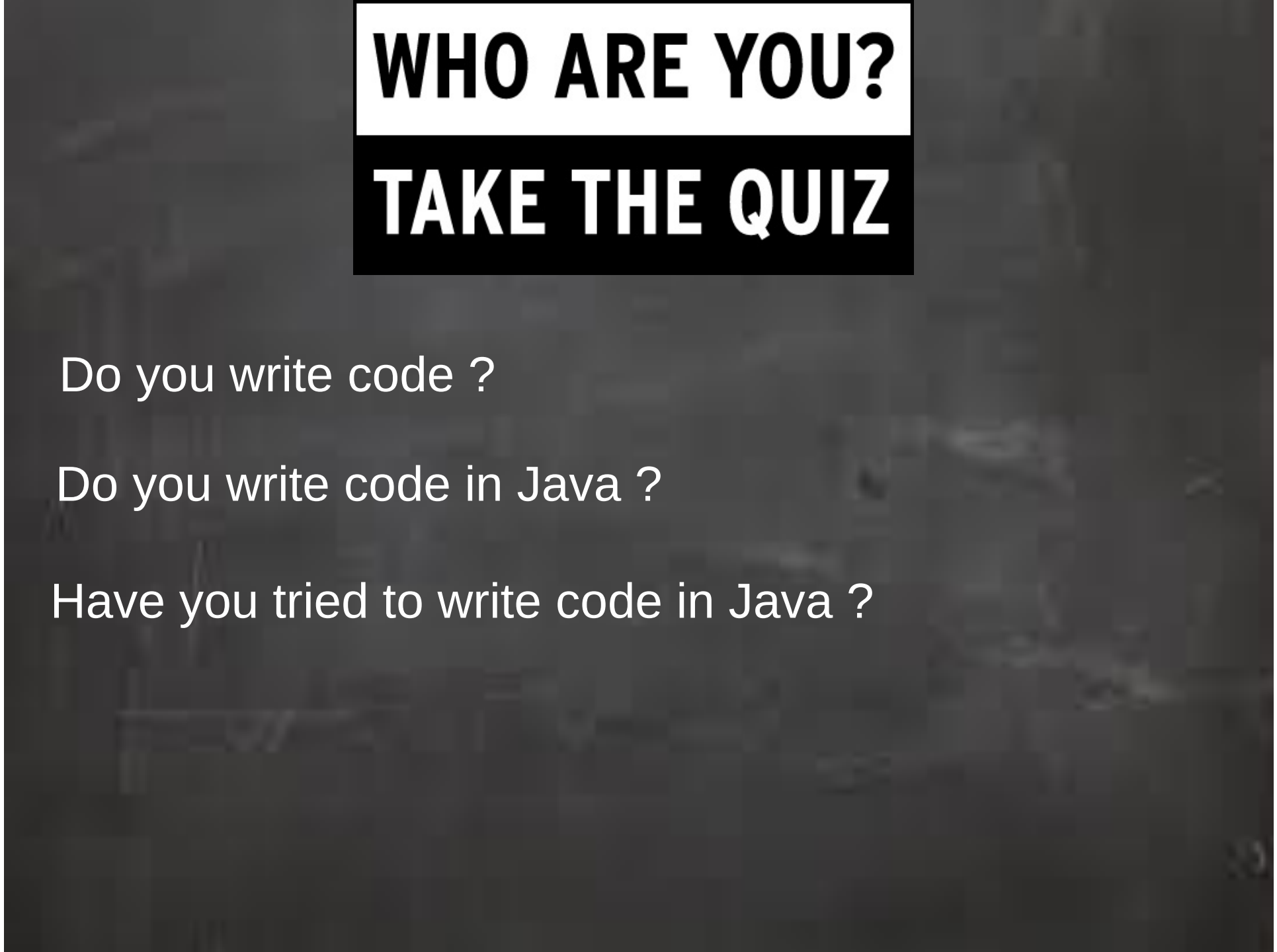
Joe Aultman

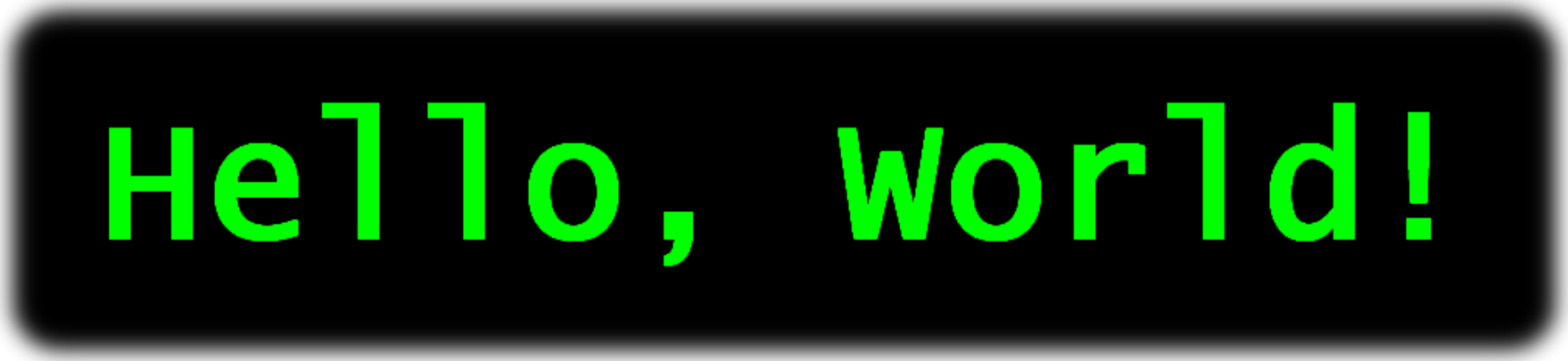
Director of Strategic Services, interRel Consulting

15+ years EPM development & management experience

- Strategic Services = EPM / BI Assessments
- Assessments = an in-depth review of a company's systems, processes, products, vision, strategy, direction, goals, trials, tribulations, and triumphs
- PLUS a team of experienced product experts
- PLUS an independent, actionable findings document and multi-year roadmap created just for you

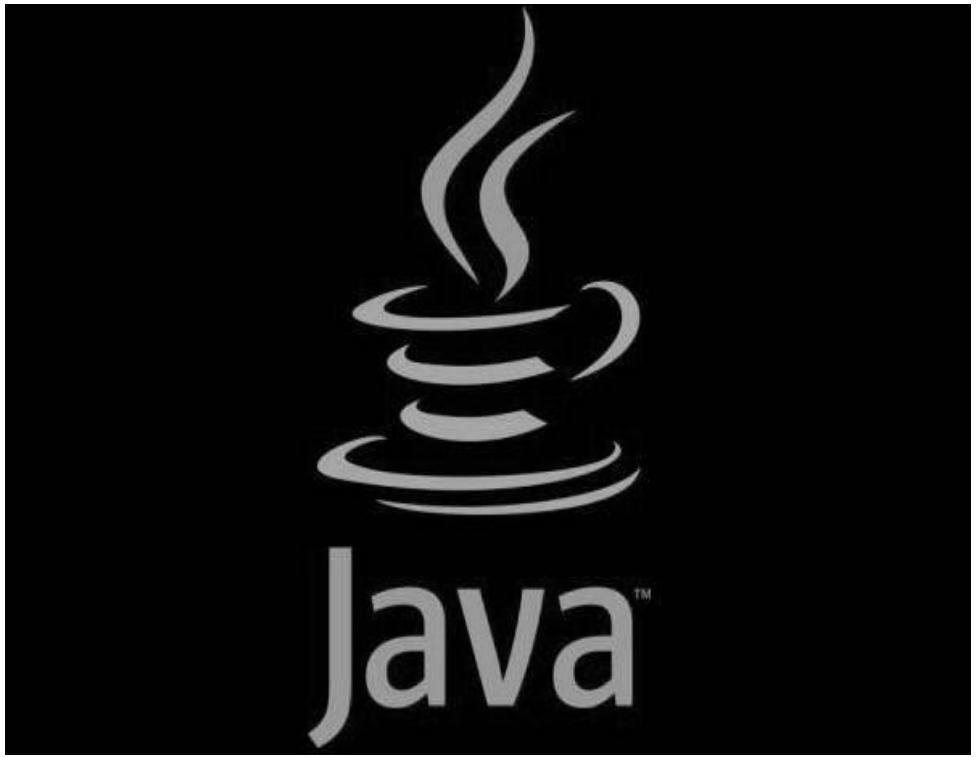

WHO ARE YOU
NOT TO BE
TAKEN AWAY
KEEP
NOTICE
ELECTRICAL
AREA
CLEAR


WHO ARE YOU?
TAKE THE QUIZ
Do you write code ?
Do you write code in Java ?
Have you tried to write code in Java ?




```
print “Hello, World!”
```

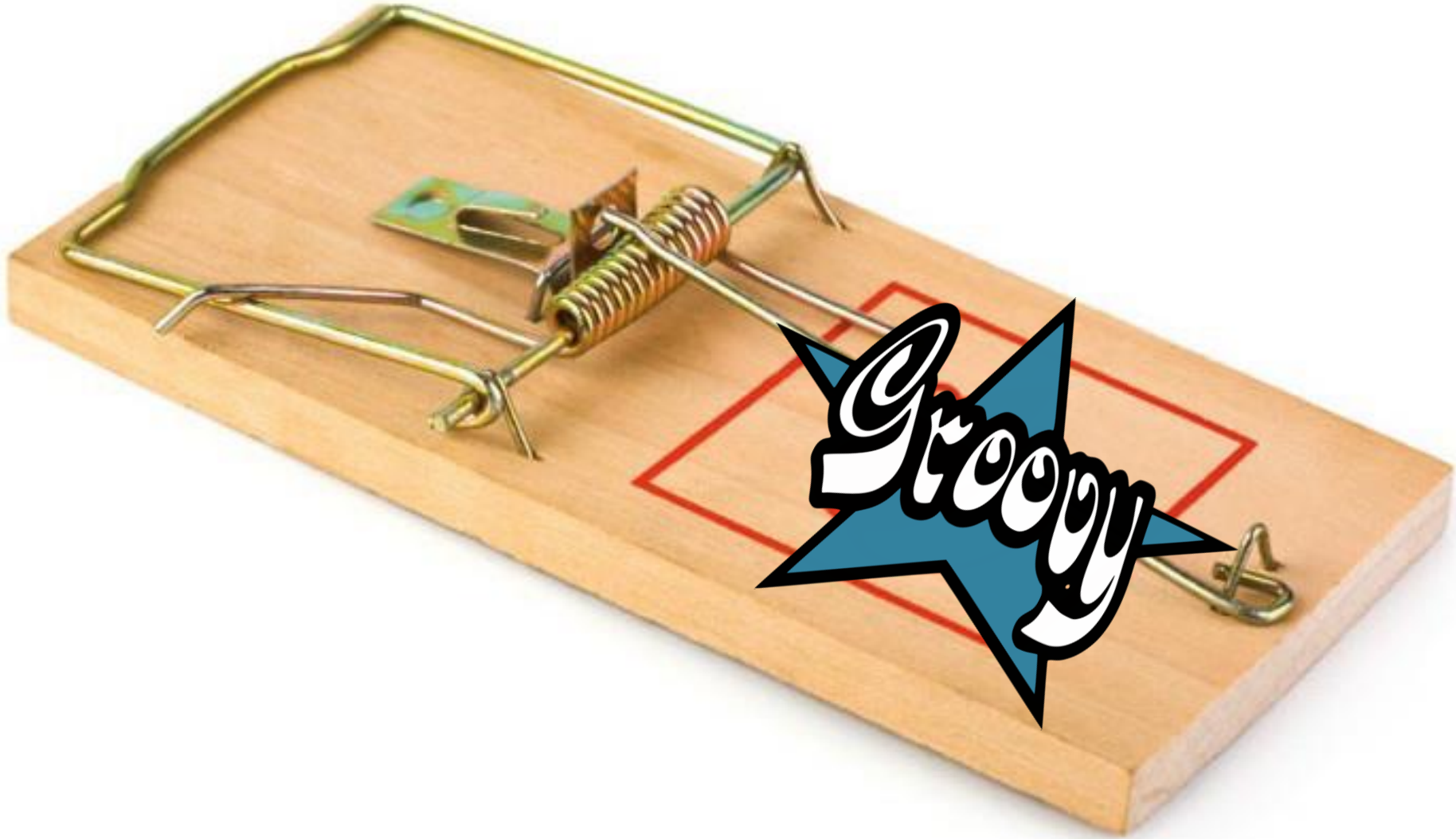
Hello, World!



Hello, World!

```
class HelloWorld
{
    public static void main(String args[])
    {
        System.out.println("Hello, World!");
    }
}
```


turn crank

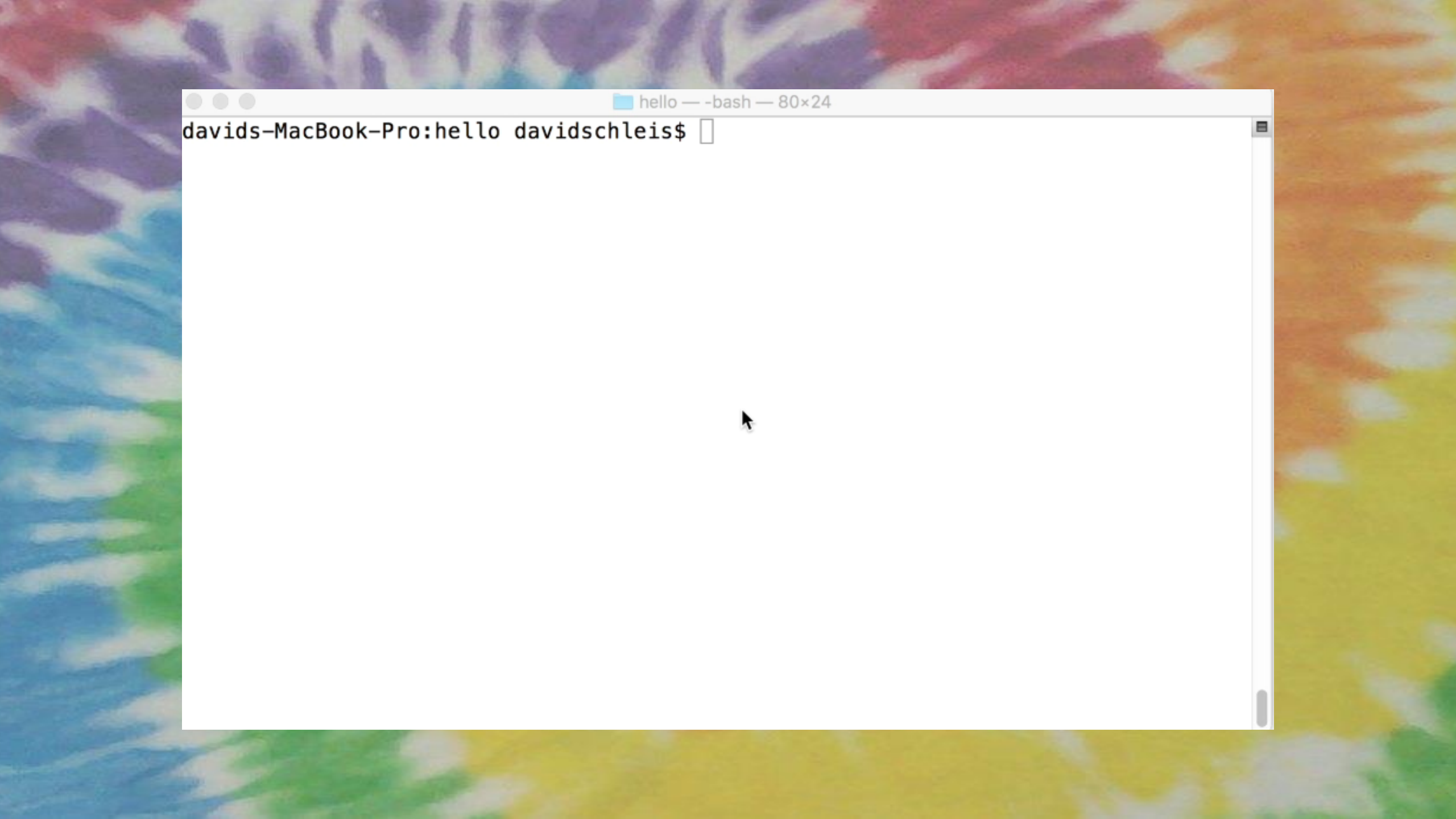

Groovy

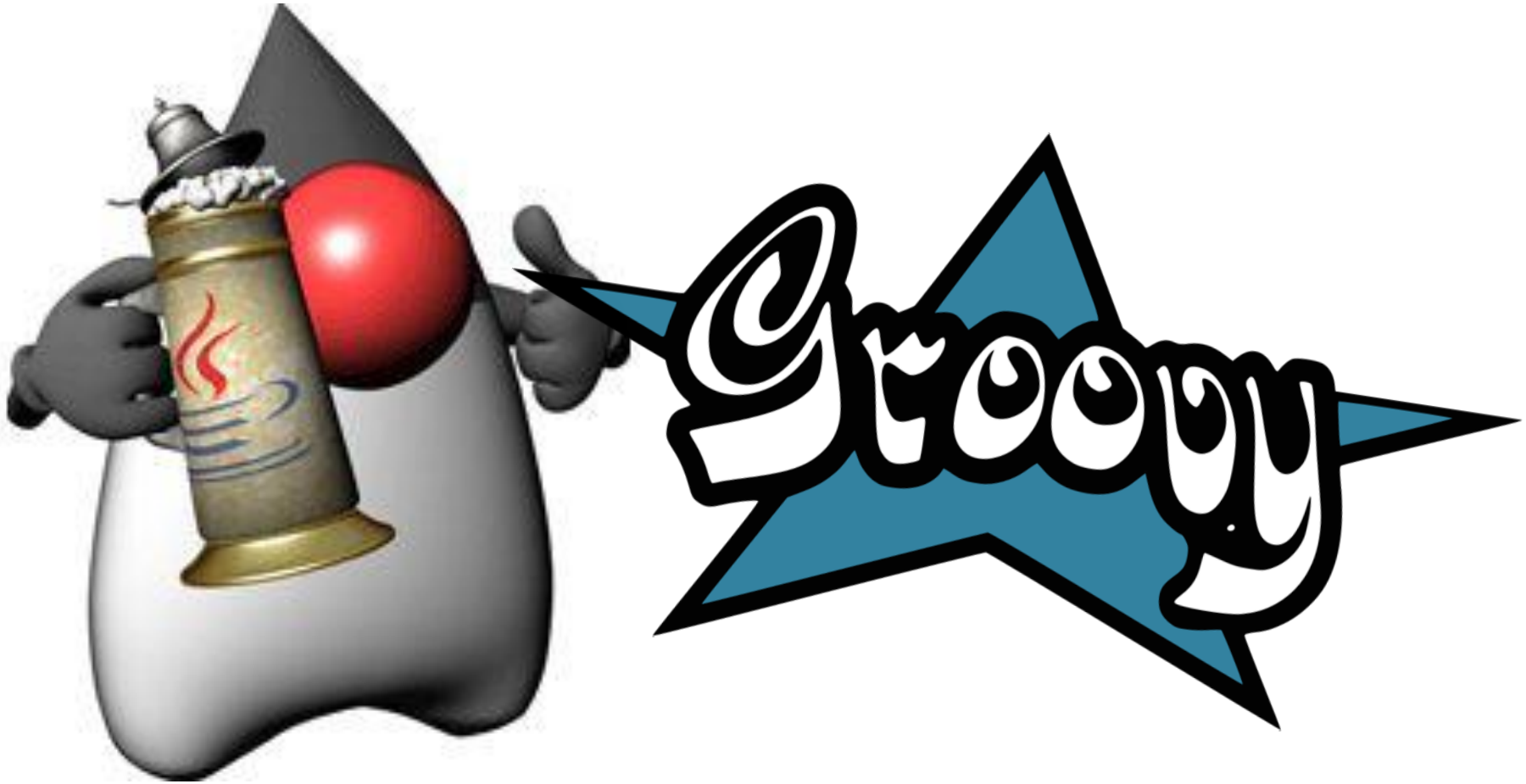
Feature rich

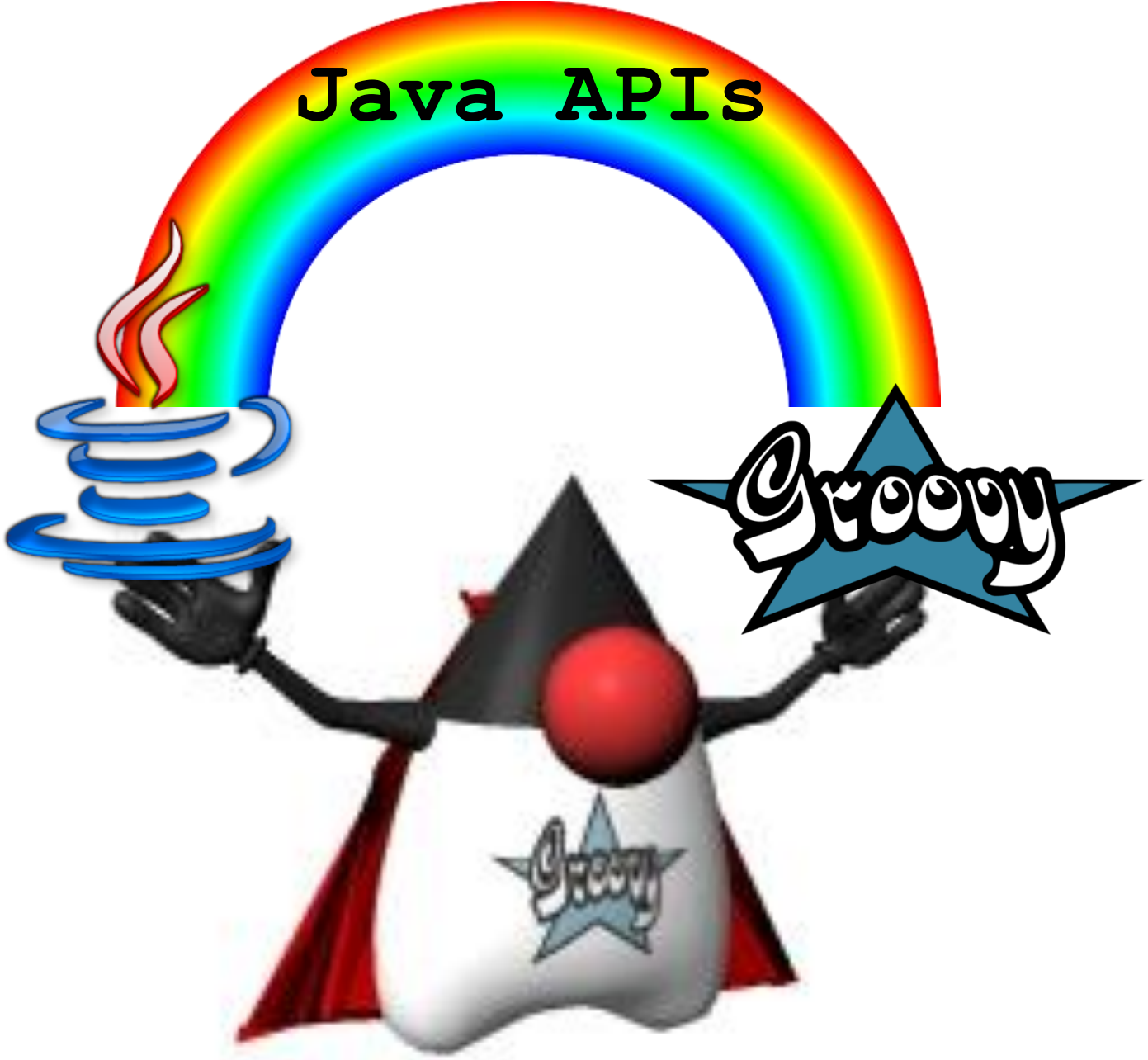
Optionally-typed

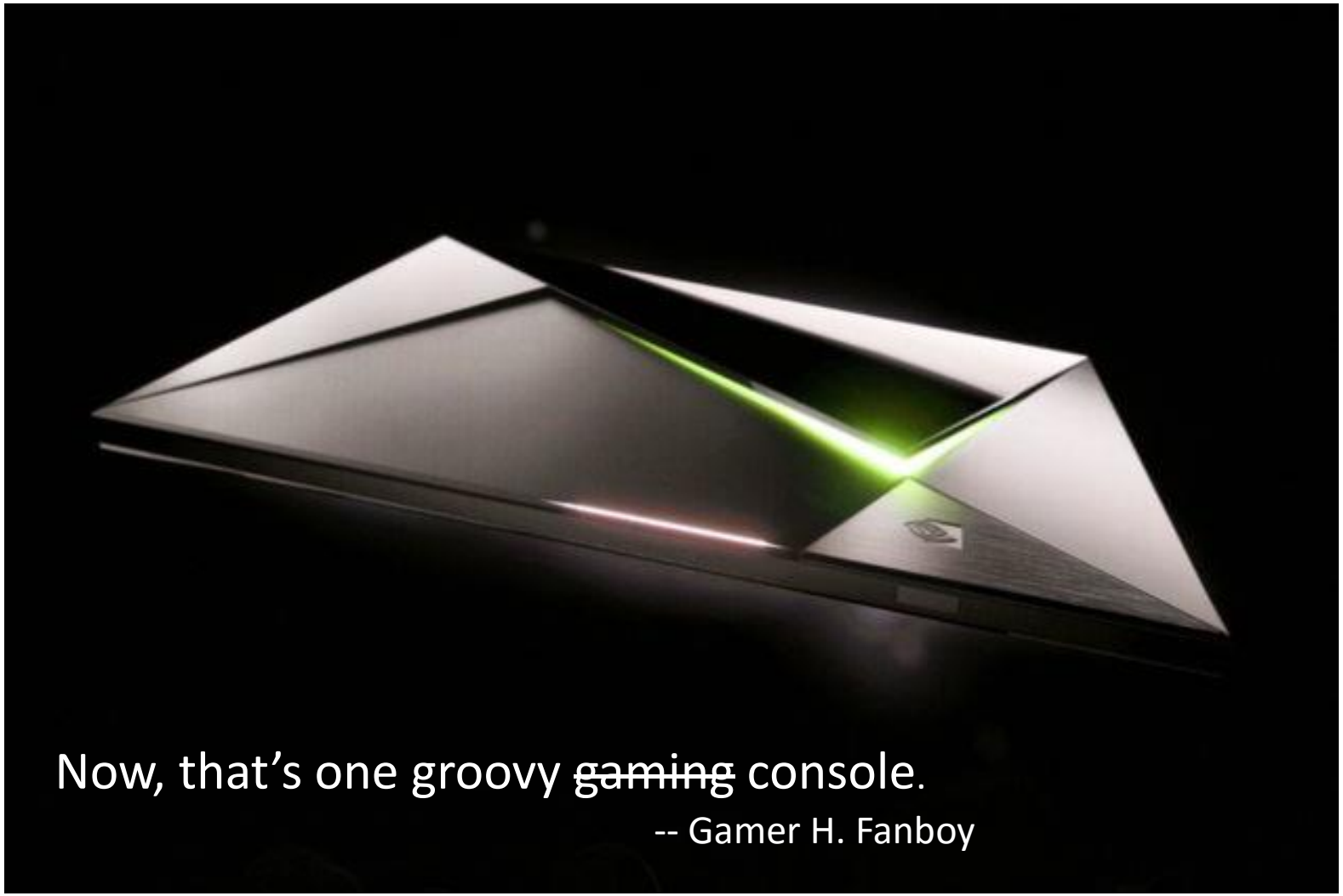
Dynamic

Java-like


hello — -bash — 80×24
davids-MacBook-Pro:hello davidschleis$


Groovy

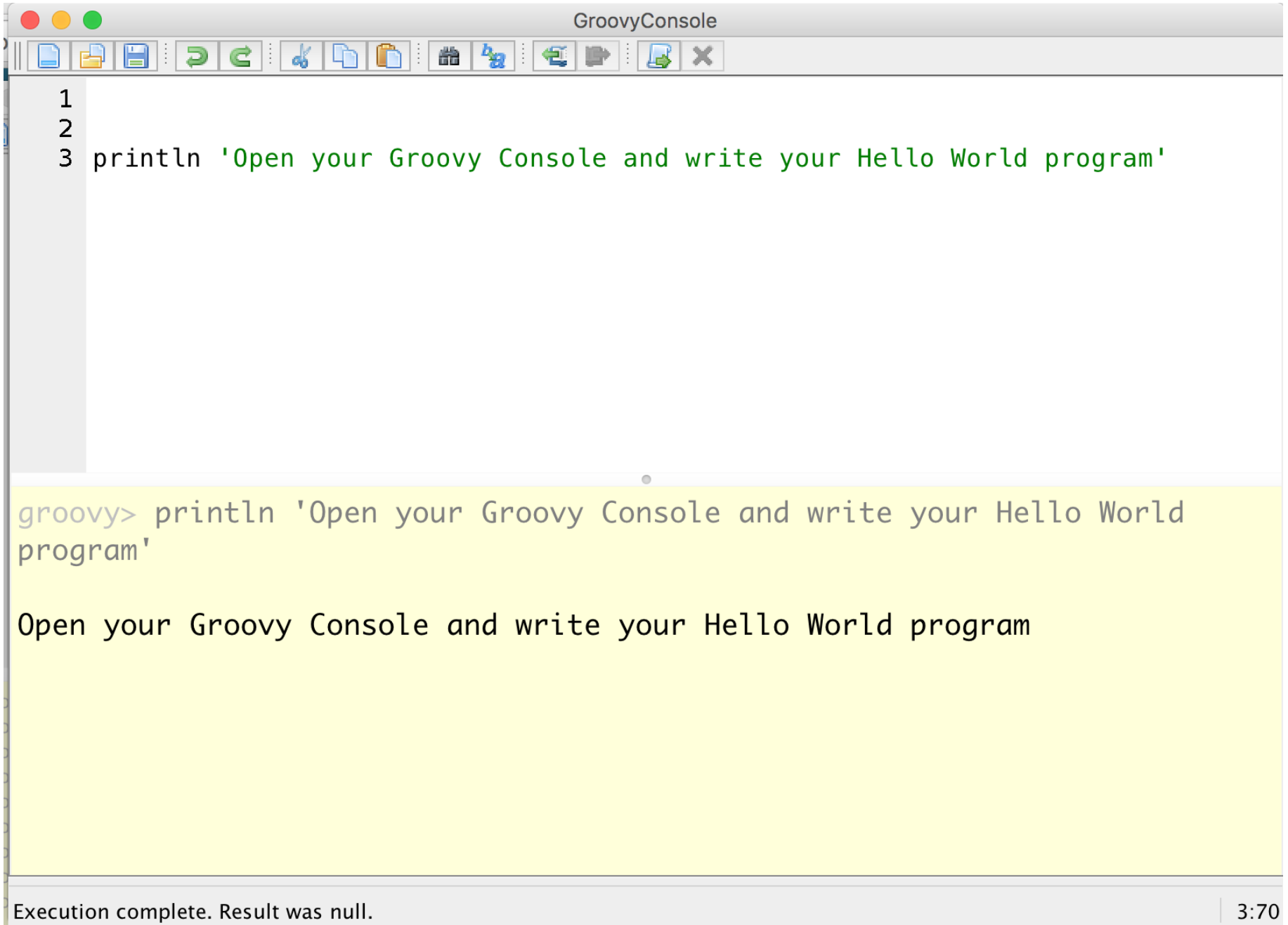

Java APIs
Groovy
Groovy



Launch your Groovy console from the shortcut on your desktop
(points to %GROOVY_HOME%/bin/GroovyConsole)

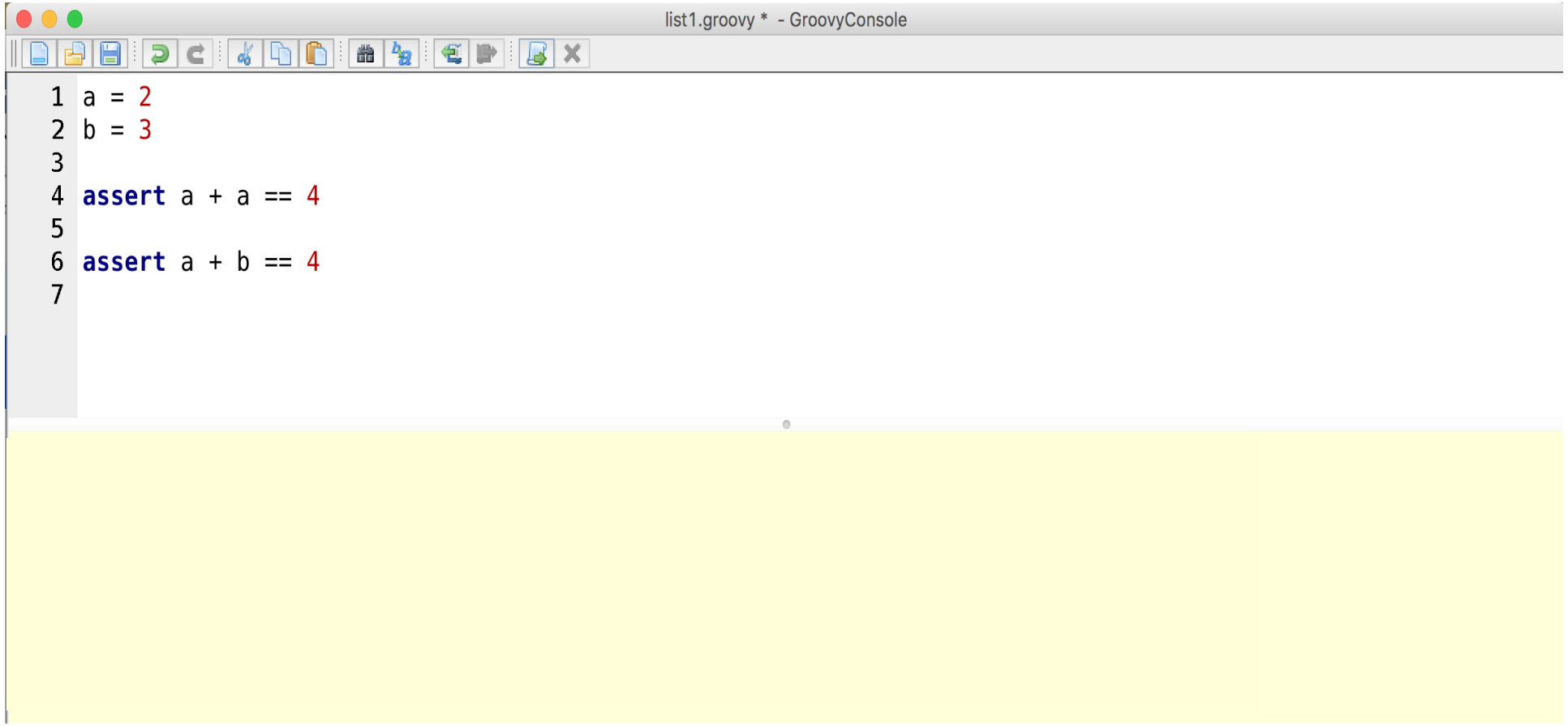
Minimize (do not close) the Groovy Console command window.

Exercise 1

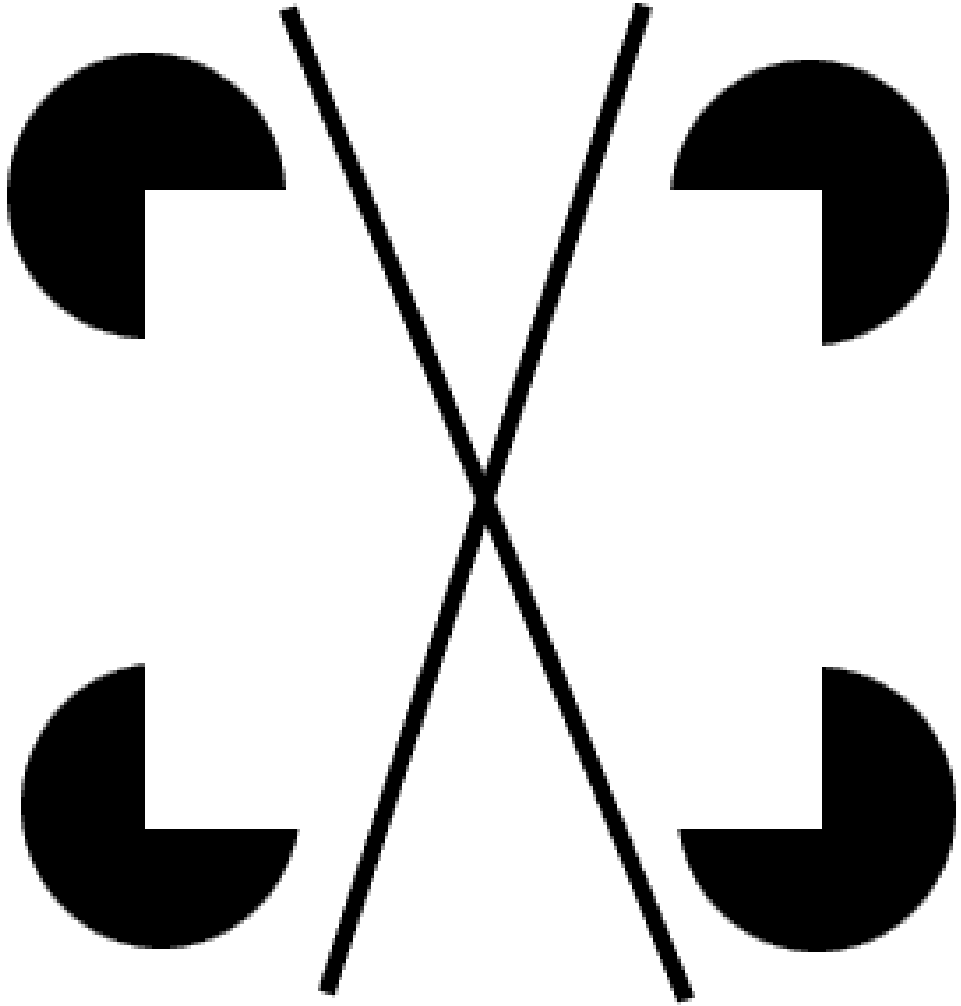

GroovyConsole
1
2
3
println 'Open your Groovy Console and write your Hello World program'
groovy> println 'Open your Groovy Console and write your Hello World program'
Open your Groovy Console and write your Hello World program
Execution complete. Result was null.
3:70


KEEP
CALM
AND
ASSERT

Exercise 1.5 (3 min.)



Finding closure



Closures

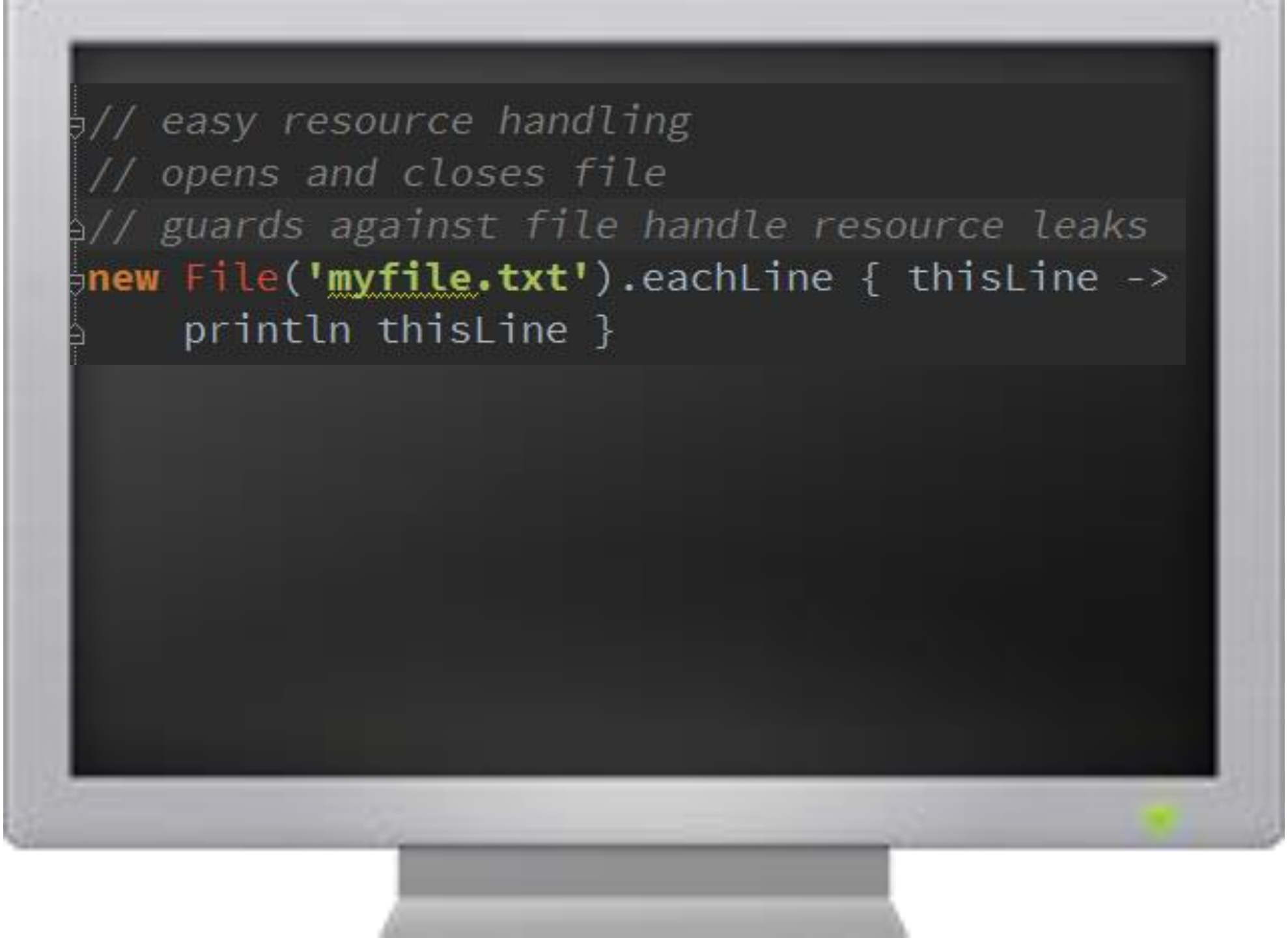
Code wrapped up as an object

Kind of like a method

But it’s an object

Statements in { }


import java.io.BufferedInputStream;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
public class FileInput {
public static void main(String[] args) {
File file = new File("myFile.txt");
FileInputStream fis = null;
BufferedInputStream bis = null;
DataInputStream dis = null;
try {
fis = new FileInputStream(file);
bis = new BufferedInputStream(fis);
dis = new DataInputStream(bis);
System.out.println(dis.readLine());
fis.close();
bis.close();
dis.close();
} catch (FileNotFoundException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
}
}
This is where the action is


// easy resource handling
// opens and closes file
// guards against file handle resource leaks
new File('myfile.txt').eachLine { thisLine ->
println thisLine }

GroovyConsole

```
1  a = [1,2,3]
2  b = [2,4,6]
3  a.each {
4    println it * 2
5  }
6
7  g = {println it * 3}
8  a.each(g)
9
10 c = []
11 a.each { aNum ->
12   c << aNum * 2
13 }
14
15 assert c == b
16 assert b == a * 2
17
```

Download course materials

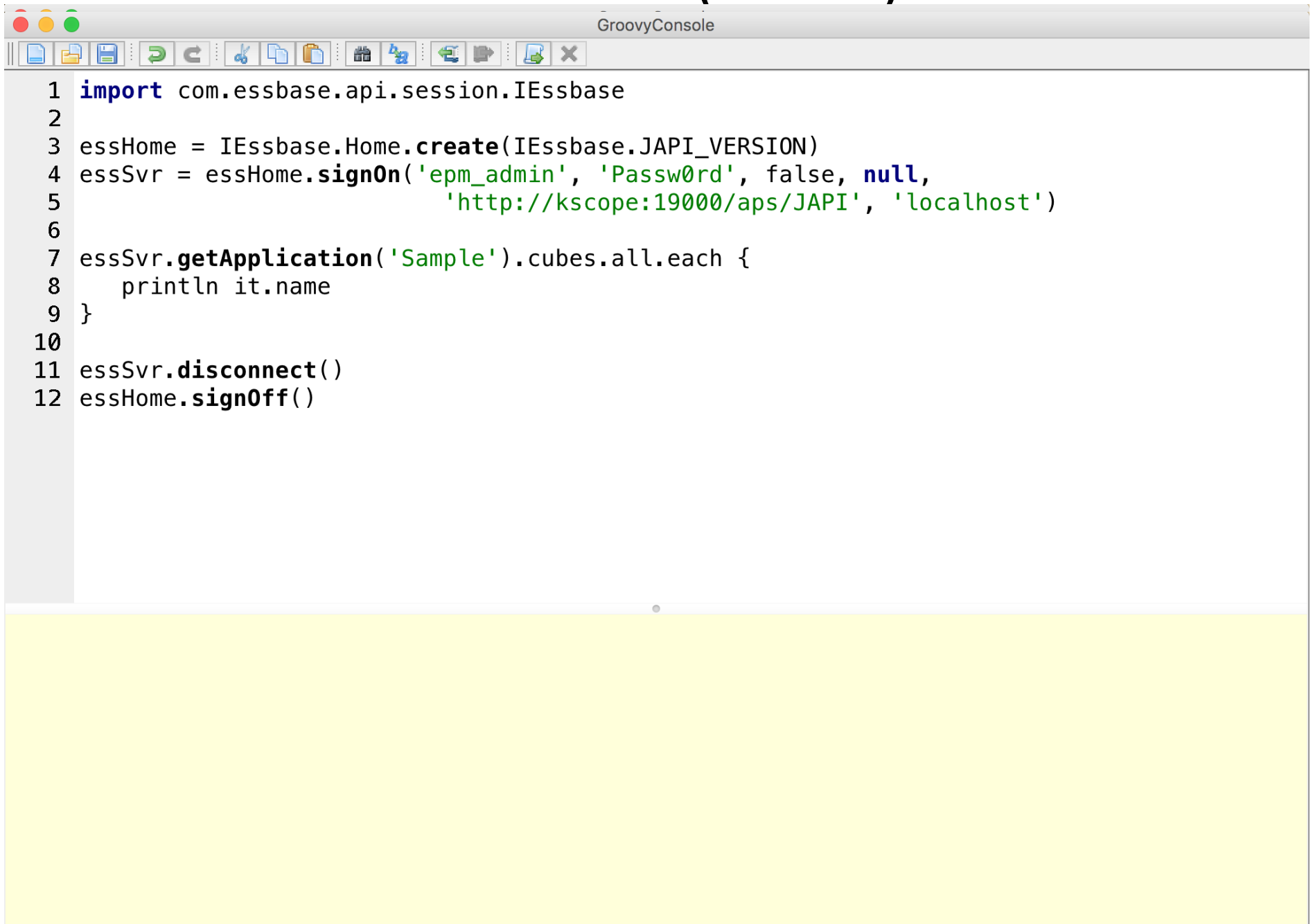
1. Open Internet Explorer
2. Open Favorites (Click on the star in the upper right)
3. Select **Totally Groovy Essbase Lab**
4. Once the page loads, click on the beautiful tye-dye image
5. Click **Save**
6. Click **View downloads**
7. Click on **Open** next to ***GroovyCode.zip***
8. This opens a File Explorer window
9. Drag the **GroovyCode** folder in the file explorer onto the **Kscope16** shortcut on the desktop
10. Select **Replace the files in the destination** in the **Replace or Skip Files** popup
11. Close the File Explorer
12. Close View Downloads
13. Close the IE tab containing the beautiful tye-dye image
14. ***Kscope16\GroovyCode\A totally Groovy Essbase Hands on Lab***

Exercise 2 (5 min.)

GroovyConsole

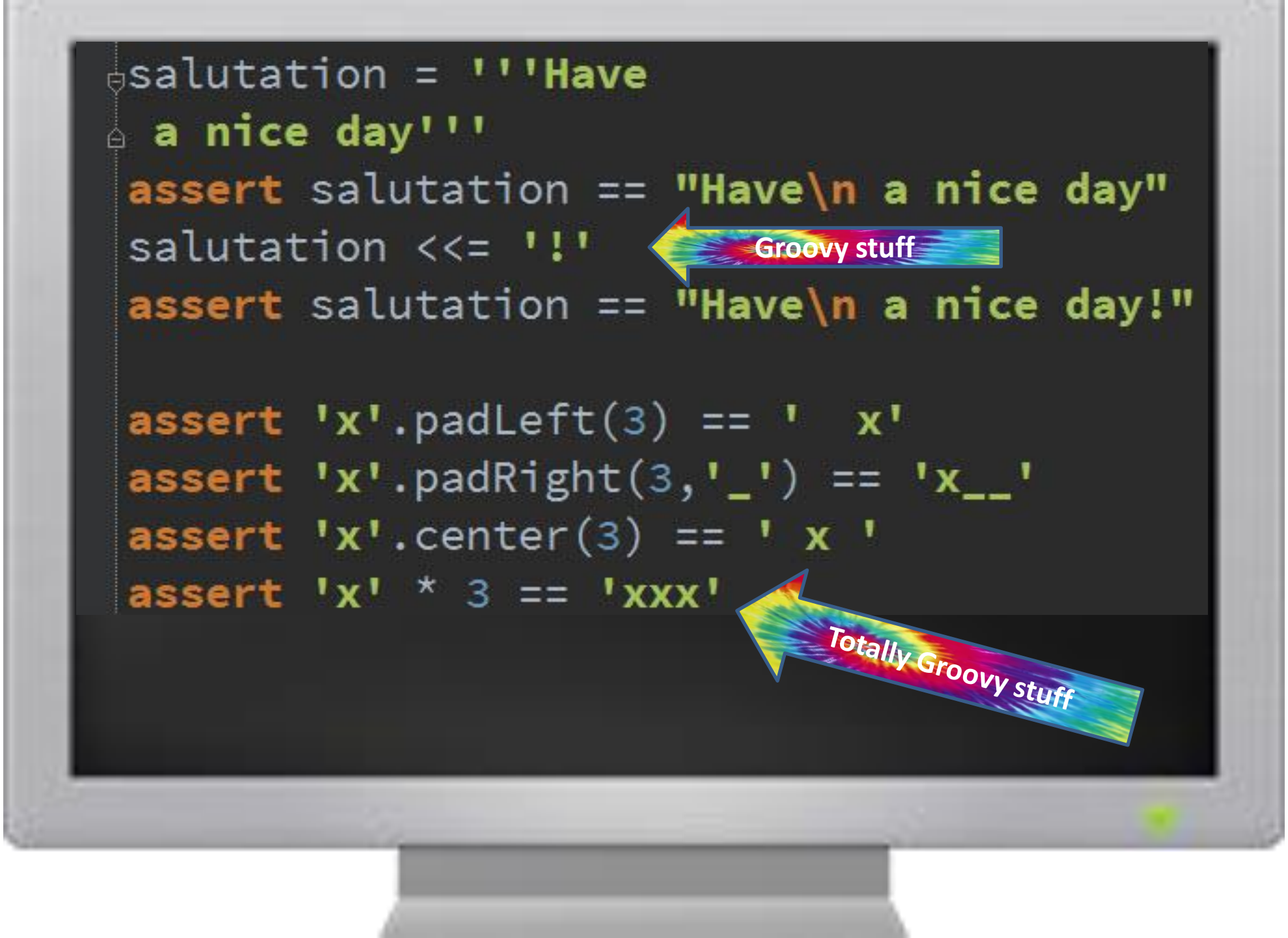
```
1  a = [1,2,3]
2  b = [2,4,6]
3  a.each {
4    println it * 2
5  }
6
7  g = {println it * 3}
8  a.each(g)
9
10 c = []
11 a.each { aNum ->
12   c << aNum * 2
13 }
14
15 assert c == b
16 assert b == a * 2
```

Exercise 3 (5 min.)

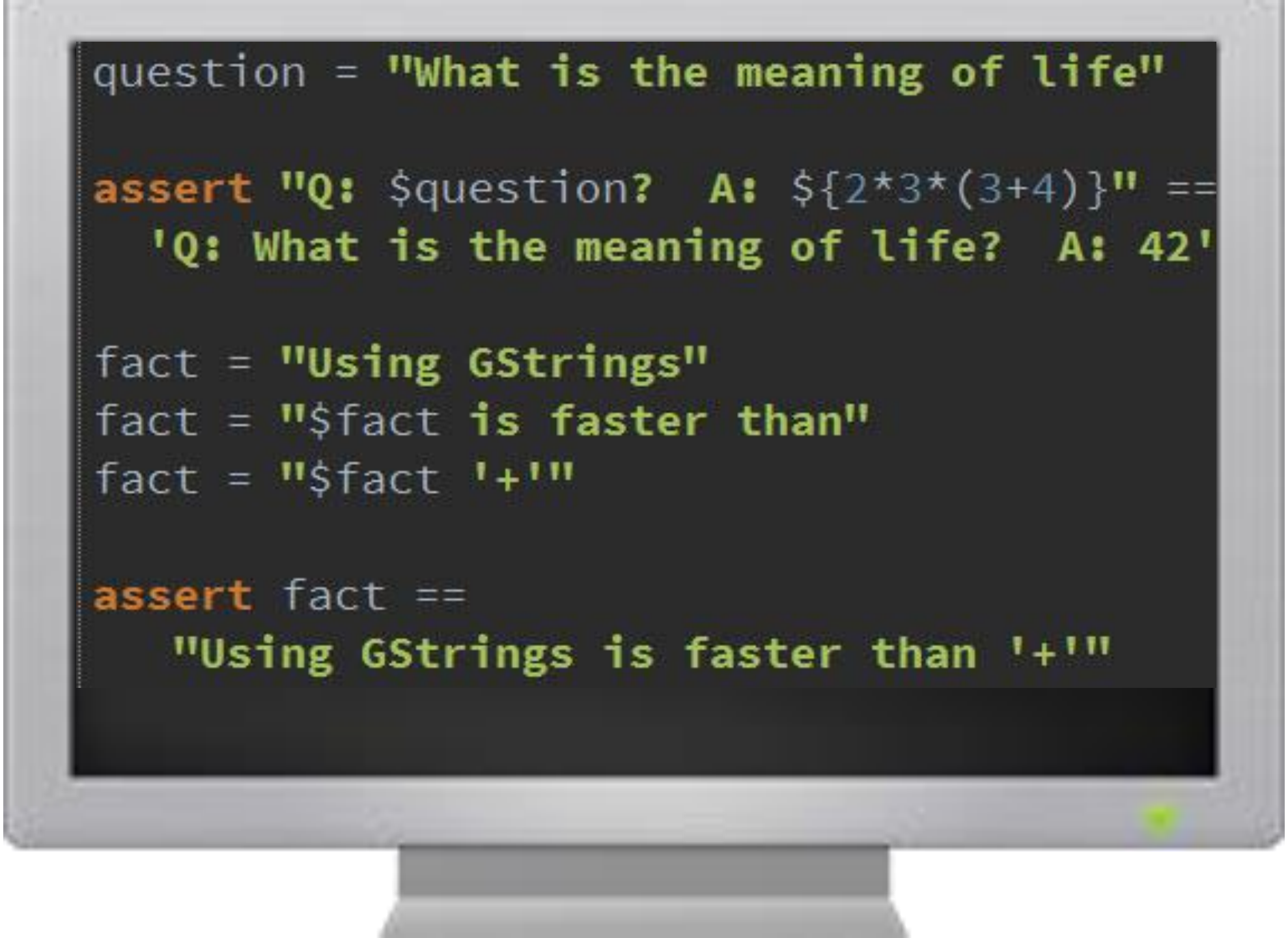


‘This is a string’

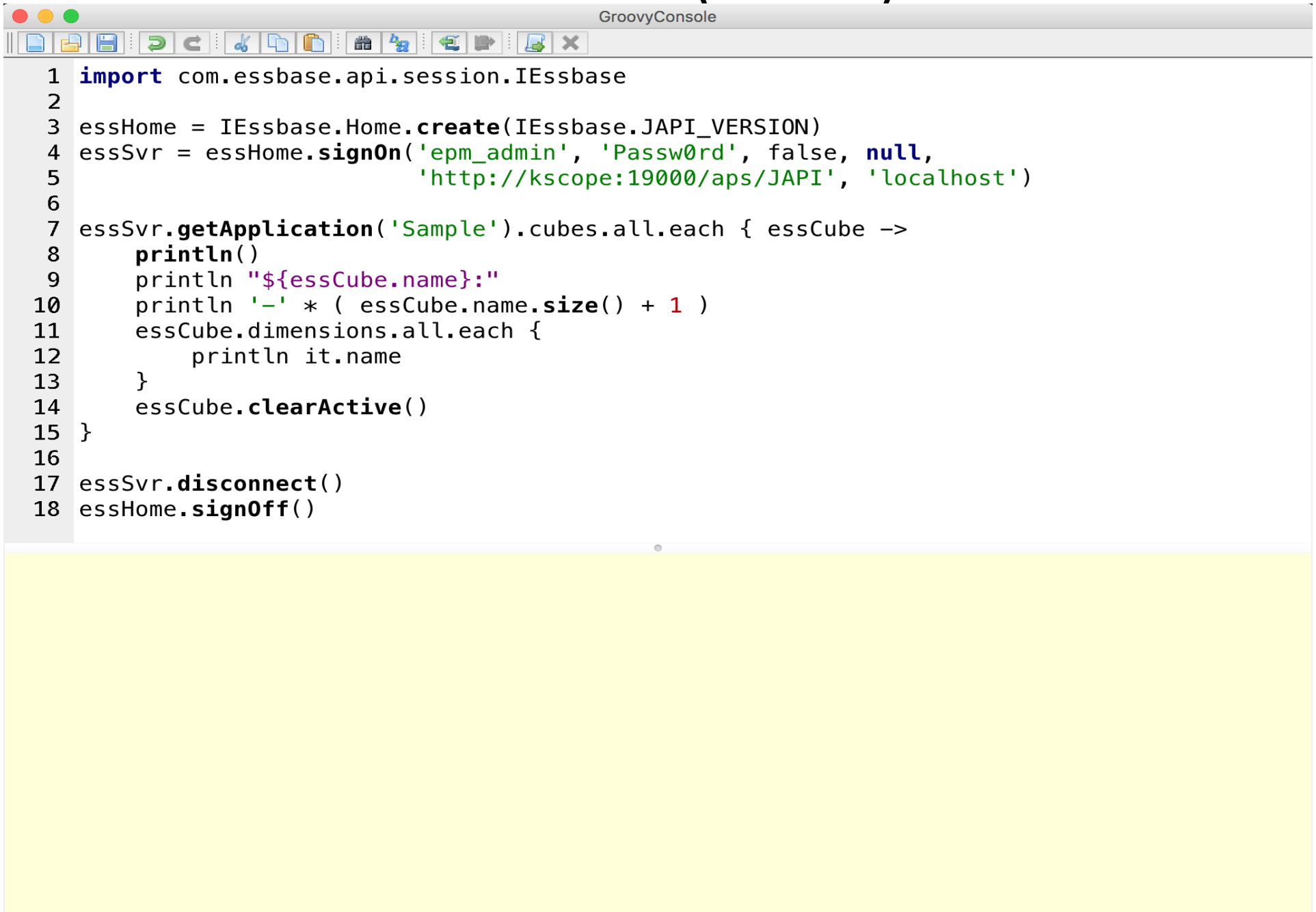

greeting = 'Hello Groovy!'
assert greeting.startsWith('Hello')
assert greeting[0] == 'H'   // 0-based
assert greeting.contains('Groovy')
assert greeting.indexOf('Groovy') == 6
assert greeting.count('o') == 3
assert greeting - ' Groovy' == 'Hello!'
assert greeting.reverse() == '!yvoorG olleH'
Groovy stuff


salutation = '''Have
a nice day'''
assert salutation == "Have\n a nice day"
salutation <<= '!'
Groovy stuff
assert salutation == "Have\n a nice day!"
assert 'x'.padLeft(3) == ' x'
assert 'x'.padRight(3,'_') == 'x__'
assert 'x'.center(3) == ' x '
assert 'x' * 3 == 'xxx'
Totally Groovy stuff

“This is a ${GString}”


question = "What is the meaning of life"
assert "Q: $question? A: ${2*3*(3+4)}" ==
'Q: What is the meaning of life? A: 42'
fact = "Using GStrings"
fact = "$fact is faster than"
fact = "$fact '+'"
assert fact ==
"Using GStrings is faster than '+'"

Exercise 4 (5 min.)



Things to Know About the JAPI

Things to Know About the Essbase JAPI…

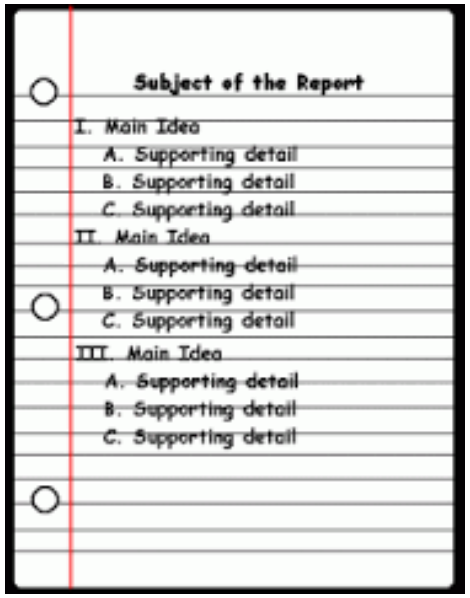
- Set of JARs available for import
- Installed with Essbase client (or copy JARs)
- Runs in two modes
  - Client ⇔ APS Server ⇔ Essbase Server
  - Client ⇔ Essbase Server (“Embedded” mode)
    - Requires extra JARs (varies with Essbase version)
- Interfaces, interfaces, interfaces
  - Throughout model and docs
  - Return objects
  - Sometimes not all methods/properties available

Main JAPI Objects / Interfaces

- IEssbase
  - .Home.create() – gets you started
  - .signOn
- IEssOlapServer
  - .getApplication
- IEssOlapApplication
  - .getCube
- IEssCube
  - .getOutline
  - .getMember
  - .openMemberSelection

Main JAPI Objects / Interfaces

- IEssCubeOutline
  - .associateAttributeMember / .disassociateAttributeMember
  - .findMember(s) / .findAlias
  - .moveMember
- IEssMember
  - .createChildMember
  - .delete
  - .getChildMembers
- IEssMemberSelection
  - .executeQuery
  - .getMembers



Groovy / JAPI “Do”s & “Don’t”s

- DO…
- refer to the JAPI samples provided with the Essbase client
- close it if you open it
- unlock it if you lock it
- consider static typing in classes & compiled code
- reuse code by putting classes in self-named .groovy files
  - Ex: class EssLoggerOnner in file EssLoggerOnner.groovy
  - Group files in a directory
  - Add directory to GROOVY_HOME\conf\groovy-starter.conf
    OR add it to the CLASSPATH
  - Wildcard (directory*) finds all .jar files
  - No wildcard (directory\) finds all .class and .groovy files
- write self-documenting code, and where you can not, comment generously
  - Explain design decisions and purpose
  - Say why it is being done this way, if another way might seem obvious, but is known to have hidden pitfalls

Groovy / JAPI “Do”s & “Don’t”s

- DON’T…
- panic
- copy and paste from JAPI samples too much
  - Miss opportunities to Groovy-cize the code
  - Stripping static typing and re-doing loops are great ways to make the code “just do stuff”
- write code that will be hard to read
  - Groovy makes this harder than some languages
  - If it must happen, comment like crazy
- overuse the “think it → code it → run it” paradigm
  - Early exits (unfinished code, errors, deliberate bailouts) may cause problems on the Essbase server
  - Resources may remain locked or processes hung
  - JAPI code should always clean up after itself
- let an evangelist go to waste
  - Reach out for help
  - You can find me on LinkedIn

~~SIN TAXES~~

syntax

Everything is an object

No primitives

Optional semicolons

Optional parenthesis

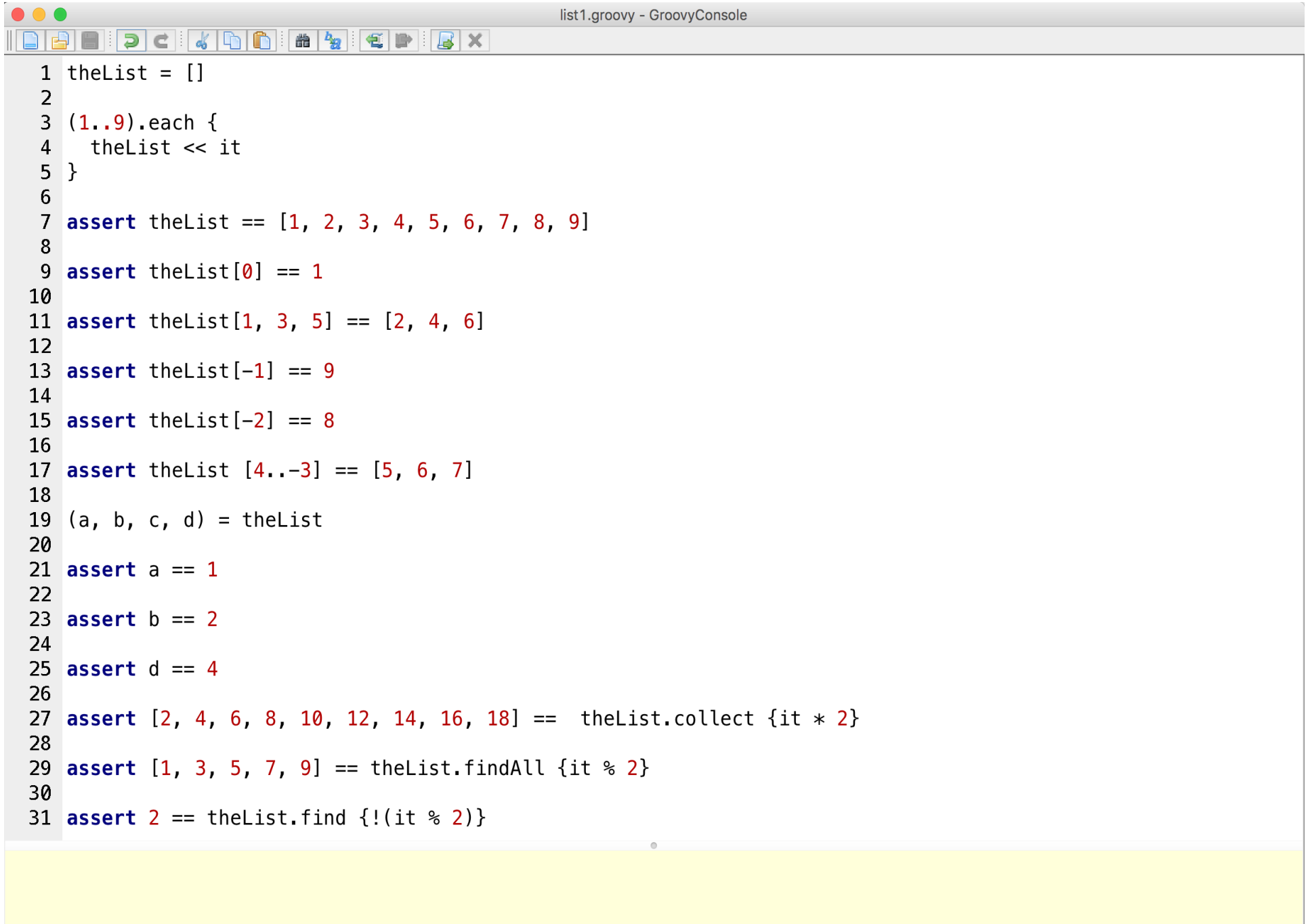
Clear, expressive, compact

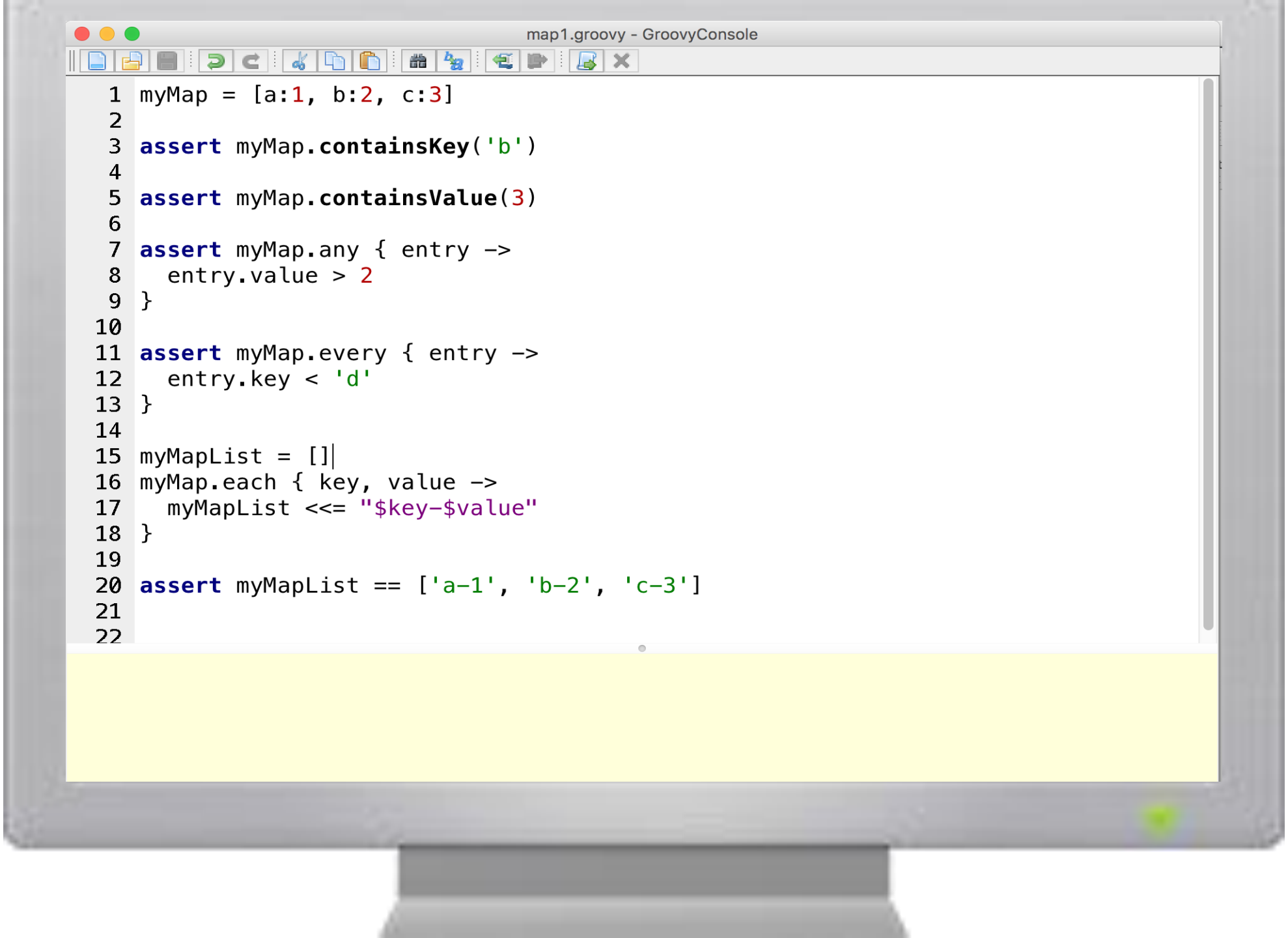
Development Kit

- Imports common packages
- Adds features to JRE objects
  - Turns getters & setters into properties:
    mbrList.getAll() becomes mbrlist.all
- Every Object is enhanced
  - Even common Java classes like File, String, List, Map
    Iterators automatically added

size() matters


Collections


list1.groovy - GroovyConsole
1 theList = []
2
3 (1..9).each {
4 theList << it
5 }
6
7 assert theList == [1, 2, 3, 4, 5, 6, 7, 8, 9]
8
9 assert theList[0] == 1
10
11 assert theList[1, 3, 5] == [2, 4, 6]
12
13 assert theList[-1] == 9
14
15 assert theList[-2] == 8
16
17 assert theList [4..-3] == [5, 6, 7]
18
19 (a, b, c, d) = theList
20
21 assert a == 1
22
23 assert b == 2
24
25 assert d == 4
26
27 assert [2, 4, 6, 8, 10, 12, 14, 16, 18] == theList.collect {it * 2}
28
29 assert [1, 3, 5, 7, 9] == theList.findAll {it % 2}
30
31 assert 2 == theList.find {!(it % 2)}


map1.groovy - GroovyConsole
1 myMap = [a:1, b:2, c:3]
2
3 assert myMap.containsKey('b')
4
5 assert myMap.containsValue(3)
6
7 assert myMap.any { entry ->
8 entry.value > 2
9 }
10
11 assert myMap.every { entry ->
12 entry.key < 'd'
13 }
14
15 myMapList = []
16 myMap.each { key, value ->
17 myMapList <<= "$key-$value"
18 }
19
20 assert myMapList == ['a-1', 'b-2', 'c-3']
21
22

Joe introduces Egress

Groovy Iterators (So Cool!)

- each()
- any()
- every()
- collect()
- find()
- findAll()
- min()/max()

max / min

returns the max/min values of the collection - for Comparable objects

```
value = [9, 4, 2, 10, 5].max()
assert value == 10
```

```
value = [9, 4, 2, 10, 5].min()
assert value == 2
```

```
value = ['x', 'y', 'a',
'z'].min()
assert value == 'a'
```

What I Don’t Like About the JAPI

Broken Item #1: IEssIterator

- It's ***Everywhere***
- Not really an "iterator"
- More of a wrapper around an array
- Three methods:
  - getAll()
  - getAt()
  - getCount()
- Can't index into it like an array
- (At least we can know we will get IEssbaseObjects out of it – but whoopty-do! We still have to cast them to the objects we want.)

Broken Item #1: IEssIterator

- I don’t want to do this:

```
for (int i = 0; i < mbrs.getCount(); i++) {
    IEssMember mbr = (IEssMember)mbrs.getAt(i);
    displayMemberProperties(mbr);
}
```

- I want to do this:

```
mbrs.each {
    displayMemberProperties(it)
}
```

- …and use all the other iterators, too!

Broken Item #1: IEssIterator

- I can almost get there with property-style access
  - Access IEssIterator.getAll() as IEssIterator.all

```
mbrs.all.each {
    displayMemberProperties(it)
}
```

- Works, but it's not good enough
  - It's not consistent with how I write other code
  - I have to remember to use .all every time

Broken Item #2: Resource Management

So many objects to clean up after

- IEssCube.clearActive()
- IEssOlapServer.disconnect()
- IEssCubeView.getGridView().setSize(0, 0)
- IEssCubeView.close()
- IEssbase.signOff()
- IEssCubeOutline.close()
- IEssMemberSelection.close()

Some of these leave resources open on the server if not called when done with the objects

Broken Item #2: Resource Management

So many exceptions that can be thrown

- try / catch / finally blocks required all over
- It’s not so much that this is wrong, per se…
- but it’s more than I want to mess with.
- Not all the time!
- Not just to ensure proper resource disposal!
- But I don’t want to clutter up the server with hanging resources!

Broken Item #3: Property Consistency

- IEssMember has a pile of getXXXX() methods
- And a pile of setXXXX(val) methods
- But not for aliases
  - getAlias(table) – the only one that takes a parameter
  - setAlias(table, val) – the only one that takes > 1 params
- I want them all to work the same
- And I want property-style access

Broken Item #4: Unnecessary Exceptions

- Methods that fail inelegantly
  - IEssOlapServer.getApplication()
  - IEssOlapApplication.getCube()
  - IEssCubeOutline.findDimension()
  - IEssCubeOutline.findMember()
- Three of them raise exceptions if not found
- getCube() returns a bogus cube (!)
- I call both results bad design

What I Did About It

Broken Item #1: IEssIterator – Fixed!

- I wrote this one line:

```
IEssIterator.metaClass.iterator = {-> delegate.all.iterator() }
```

- And that's all it took!
  - Adds the required method to the interface
  - Every IEssIterator I get from the JAPI is now loaded with Groovy goodness

```
essApps = essSvr.applications
assert essApps.any { it.name == 'Sample' }
assert essApps.findAll { it.name =~ /(?i)Samp/ }.size() == 5

assert essApps.find { it.name == 'Sample' } instanceof
    IEssOlapApplication
```

You May Have Noticed…

- I also wrote this:

```
IEssIterator.metaClass.size = {-> delegate.count }
```

- And now IEssIterator also has a size() method
  - size() is the standard across Groovy
  - Has been added to native Java objects
  - Replaces
    - count
    - length
    - capacity
    - etc.
  - Note that it's a method, not a property (needs parens)

```
assert essApps.findAll { it.name =~ /(?i)Samp/ }.size() == 5
```

And Another IEssIterator Enhancement

- I also wrote this:

```
/* Replace the getAt() method with one that supports negative indexes.*/
IEssIterator.metaClass.getAt = { int index -> delegate.all.getAt(index) }
/*Support the range subscript operator in IEssIterator*/
IEssIterator.metaClass.getAt = { Range range -> delegate.all.getAt(range) }
/*Select a List of items from an IEssIterator using a Collection of indices.*/
IEssIterator.metaClass.getAt = { Collection indices -> delegate.all.getAt(indices) }
```

- Now IEssIterator has indexing, because Groovy treats obj[index] as obj.getAt(index)
- Groovy has more indexing options than Java

```
assert essCube.dimensions[0].name == 'Year'
assert essCube.dimensions[-2].toString() == 'Intro Date'
assert essCube.dimensions[0..2]*.name == ['Year', 'Measures', 'Product']
assert essCube.dimensions[[0, 2, 4]]*.name == ['Year', 'Product', 'Scenario']
```

Broken Item #2: Resource Management – Fixed!

- Automatic Resource Management
  - Enabled by closures
  - Closure passed to a method
  - Methods are often named withSomething()
  - The method
    - Sets up all the required resources
    - Runs the closure
    - Cleans up the resources
    - Implements try / catch / finally inside

The user is freed from the hassles, yet gets resources that are always cleaned up

Broken Item #2: Resource Management – Fixed!

- IEssbase
  - withHome()
  - withServer()
- IEssOlapServer
  - withApplication()
  - withCube()
  - withOutline()
- IEssOlapApplication
  - withCube()
  - withOutline()
- IEssCube
  - withOutline()
  - withMemberSelection()

Now Instead of This:

```
try {
    ess = IEssbase.Home.create(IEssbase.JAPI_VERSION);
    IEssDomain dom = ess.signOn(user, pass, false, null, provider);
    svr = (IEssOlapServer)dom.getOlapServer(server);
    svr.connect();
    IEssCube cube = svr.getApplication("Demo").getCube("Basic");
    doStuffWithCube(cube);
} catch (EssException x) {
    System.err.println("ERROR: " + x.getMessage());
} finally {
    if (svr != null && svr.isConnected() == true)
        svr.disconnect();
    }
    if (ess != null && ess.isSignedOn() == true)
        ess.signOff();
    }
}
```

I Can Do This:

```
withServer(user, pass, server) { svr ->

    svr.withCube('Demo', 'Basic') { cube ->
        doStuffWithCube(cube);
    }

)
```

But Are the Iterators Resource Safe?

Well… No. You got me there.

- This does not iterate safely:

```
svr.applications.each { app ->
    app.cubes.each { cube ->
        /* do stuff with cube */
    }
}
```

- No opportunity to set up resources first, or tear them down afterward
- No background try / catch / finally

You'd Have to Do This

```
svr.applications*.name.each { appName ->
   svr.withApplication(appName) { safeApp ->
      safeApp.cubes*.name.each { cubeName ->
         safeApp.withCube(cubeName) { safeCube ->
            /* do stuff with safeCube */
         }
      }
   }
}
```

- I don't want to do that.

Also, I Want More ‘Each’es

- Several Common Use Cases
  - Dimensions
  - Members
  - Relationships to Members
    - Descendants
    - Members at Level
    - Members at Generation
    - Ancestors
- The API doesn’t have methods like:
  - getDescendants(), getAncestors()
- So I can’t get my enhanced IEssIterators

  (i.e. I can’t go mbr.ancestors.each {})

So I Made Some Specialized Iterators

- IEssOlapServer
  - eachApplication()
- IEssOlapApplication
  - eachCube()
- IEssCubeOutline
  - eachDimension()
  - eachMember()
- IEssMember
  - eachDescendant()
  - eachDescendantAtLevel()
  - eachDescendantAtGeneration()
  - eachChild()
  - eachAncestor()

Broken Item #3: Property Consistency – Fixed!

- IEssMember.getAlias()
  - Adds a no-parameter version
  - All other getXXXX() methods take no parameters
  - Allows member.alias property-style access
- IEssMember.setAlias(val)
  - Adds a one-parameter version
  - All other setXXXX() methods take one parameter
  - Allows member.alias = val property-style access
- Both of these hit the default alias table

Why These Alias Methods Matter

- Property-style access is very powerful
- I can now get or set *any* property
  - At runtime
  - Without having to know which ones ahead of time
  - And without alias messing me up

```
['dimensionName', 'alias', 'formula'].each {
    println essMbr[it]  /* This is awesome */
}
['solveOrder':5, 'alias':'Cola', 'expenseMember':false] { k,v ->
    essMbr[k] = v       /* And so is this */
}
```

- Data-driven build / modify scripts, anyone?

Broken Item #4: Unnecessary Exceptions – Fixed!

- I have added these methods
  - IEssOlapServer.getApplicationOrNull()
  - IEssOlapApplication.getCubeOrNull ()
  - IEssCubeOutline.findDimensionOrNull ()
  - IEssCubeOutline.findMemberOrNull ()
- They do what I want
- Are more appropriate to what should be treated as normal, not as warranting an exception

(Oh, BTW, I Added This…)

- IEssAttributeQuery works a little oddly
- Piggybacks on top of IEssMemberSelection
- .withMemberSelection() doesn't quite cut it
- So I made IEssCube.executeAttributeQuery()

```
results = essCube.executeAttributeQuery { essQry ->
   essQry.set(IEssAttributeQuery.MBR_TYPE_ATTRIBUTE_DIMENSION,
IEssAttributeQuery.MBR_TYPE_BASE_DIMENSION)
    essQry.setInputMember('Population') // Any member of the attribute dimension will
do.
}
assert results*.name == ['Market']
```

And Now: A Real World Example!

- Print all members with a certain attribute to a file

```
def results = []
IEssbase.withServer(user, pass, server) { essSvr ->
  essSvr.withOutline('App', 'Db') { essOtl ->
    essOtl.findMember('BaseDim').eachDescendantAtLevel(0) { essMbr ->
      if (essMbr.associatedAttributes*.name.contains('AttribMbr')) {
        results << essMbr.alias
} } } }

new File('./Members_With_Attib.txt').withWriter {
  it << results.sort(true).join('\r\n')
}
```

- This is the whole program!

How Can I Get Me Some of That?

- https://github.com/HyperionAddict/Egress

Egress

A Groovy API for Essbase

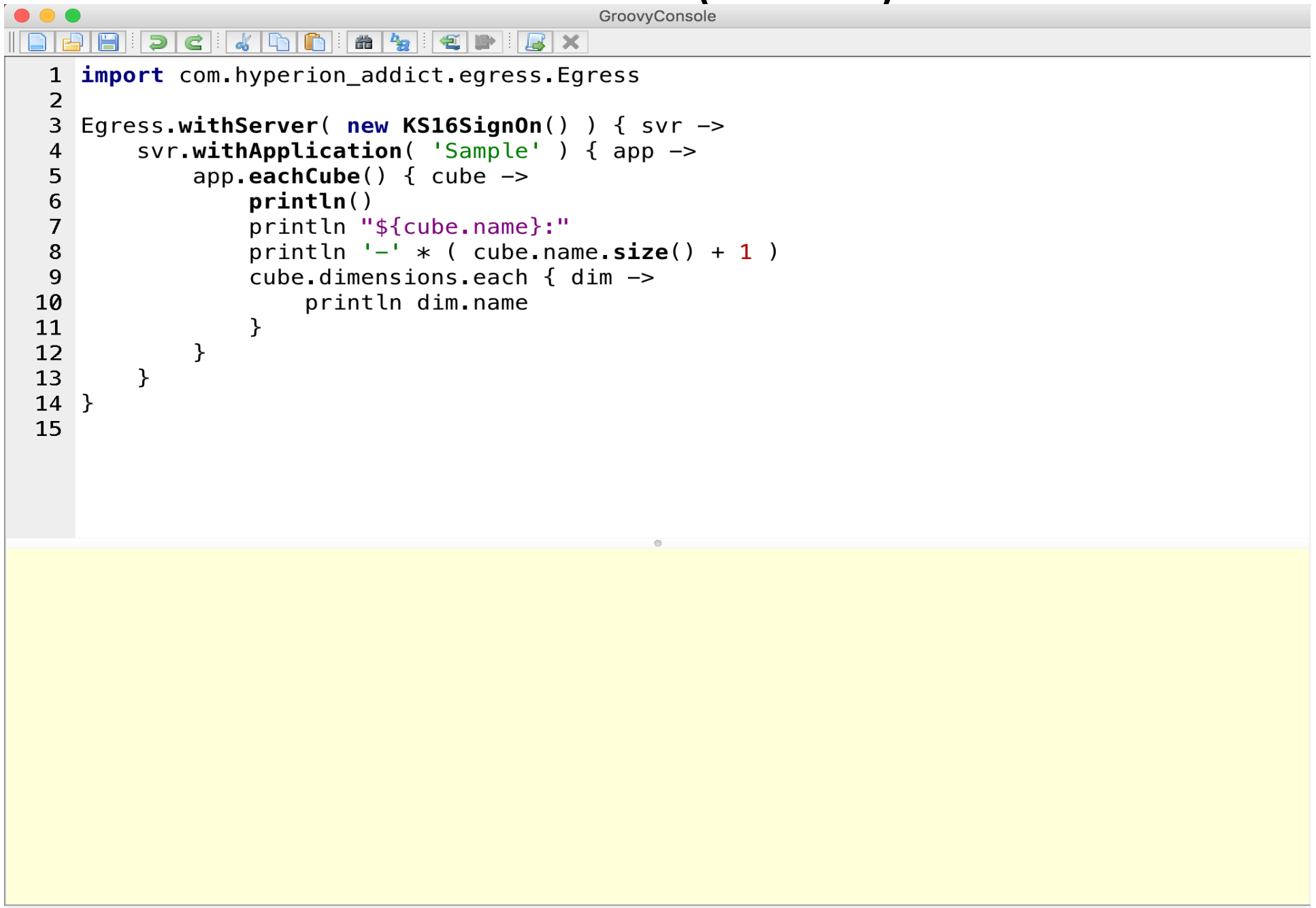
Looking to make your life as an Essbase admin easier? Frustrated with MaxL's limitations? Confused by Windows CMD, PowerShell, bash, Perl, etc? Stuck trying bridge the gap between scripting options and the power of an API? Trapped in a world with no really good options for automating Essbase tasks?

Egress is the way out.

What Does Egress Do?

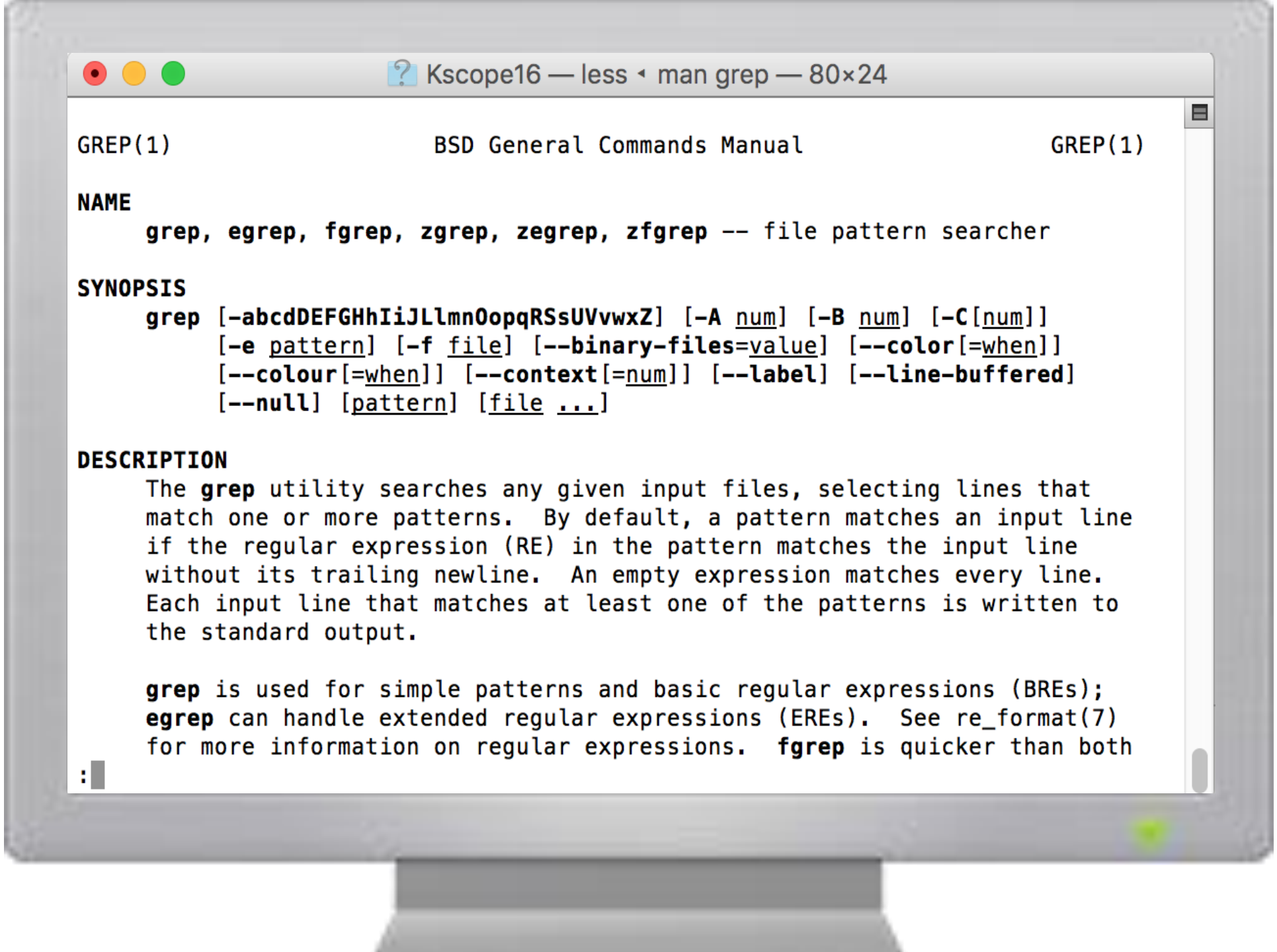
- Different kinds of changes / enhancements to the JAPI
  1. Encapsulates logon methods & parameters:
     Keeps id's & passwords out of script code
  2. Static methods on Egress class:
     Make access to starting points in Essbase easy to get to
  3. Injecting Grooviness:
     Changes to the JAPI that give it a more Groovy flavor
  4. Automatic Resource Management:
     Adds withXXXX() methods in various places
  5. Specialized Iterators:
     Adds eachXXXX() methods in various places
  6. Tweaks and small enhancements:
     Methods that make the API a little easier to work with

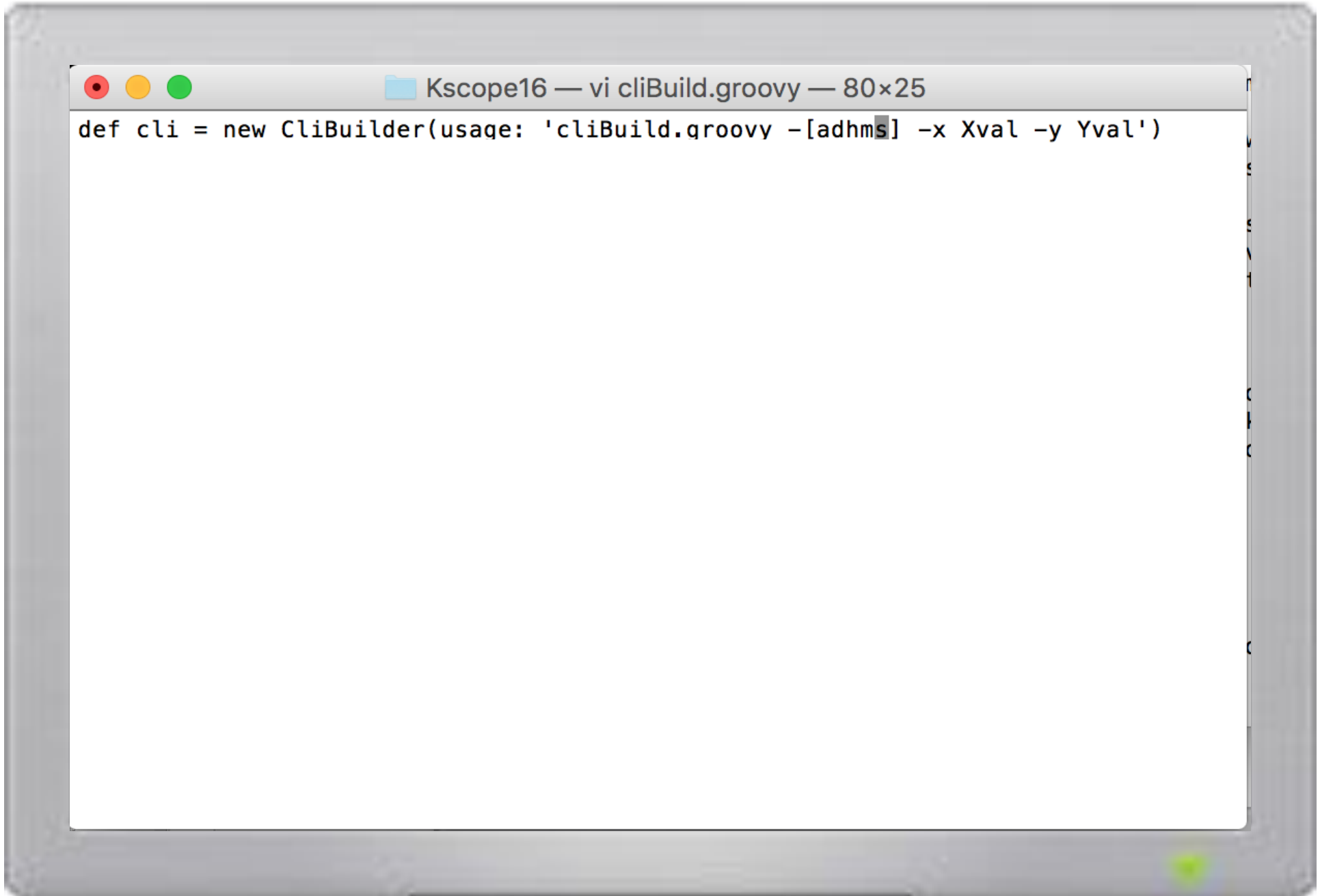
Exercise 5 (5 min.)



COMMAND

line interface


Kscope16 — less ◂ man grep — 80×24
GREP(1) BSD General Commands Manual GREP(1)
NAME
grep, egrep, fgrep, zgrep, zegrep, zfgrep -- file pattern searcher
SYNOPSIS
grep [-abcdDEFGHhIiJLlmnOopqRSsUVvwxZ] [-A num] [-B num] [-C[num]]
[-e pattern] [-f file] [--binary-files=value] [--color[=when]]
[--colour[=when]] [--context[=num]] [--label] [--line-buffered]
[--null] [pattern] [file ...]
DESCRIPTION
The grep utility searches any given input files, selecting lines that
match one or more patterns. By default, a pattern matches an input line
if the regular expression (RE) in the pattern matches the input line
without its trailing newline. An empty expression matches every line.
Each input line that matches at least one of the patterns is written to
the standard output.
grep is used for simple patterns and basic regular expressions (BREs);
egrep can handle extended regular expressions (EREs). See re_format(7)
for more information on regular expressions. fgrep is quicker than both
:

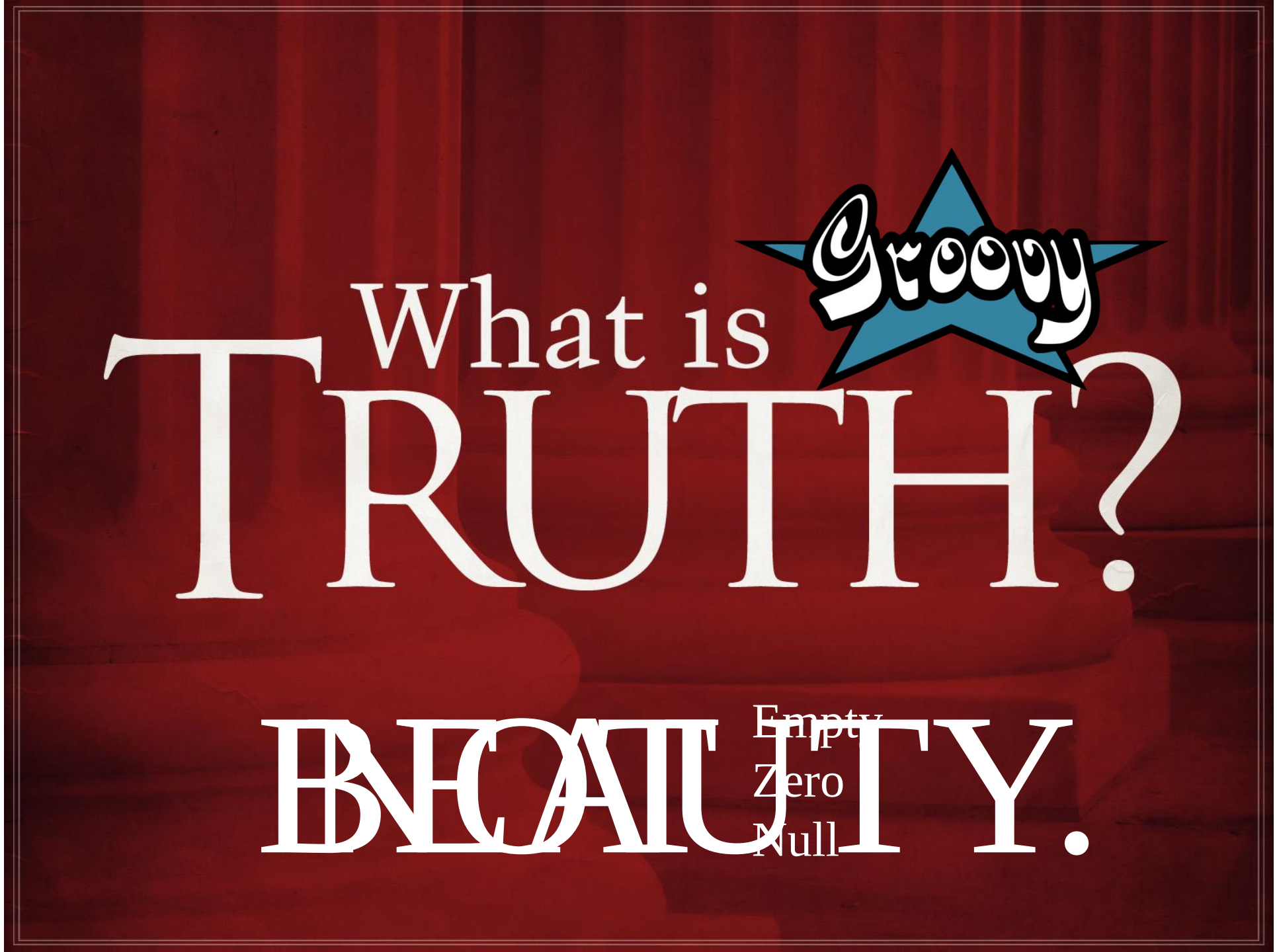

Kscope16 — vi cliBuild.groovy — 80×25
def cli = new CliBuilder(usage: 'cliBuild.groovy -[adhms] -x Xval -y Yval')

Exercise 6 (8 min.)

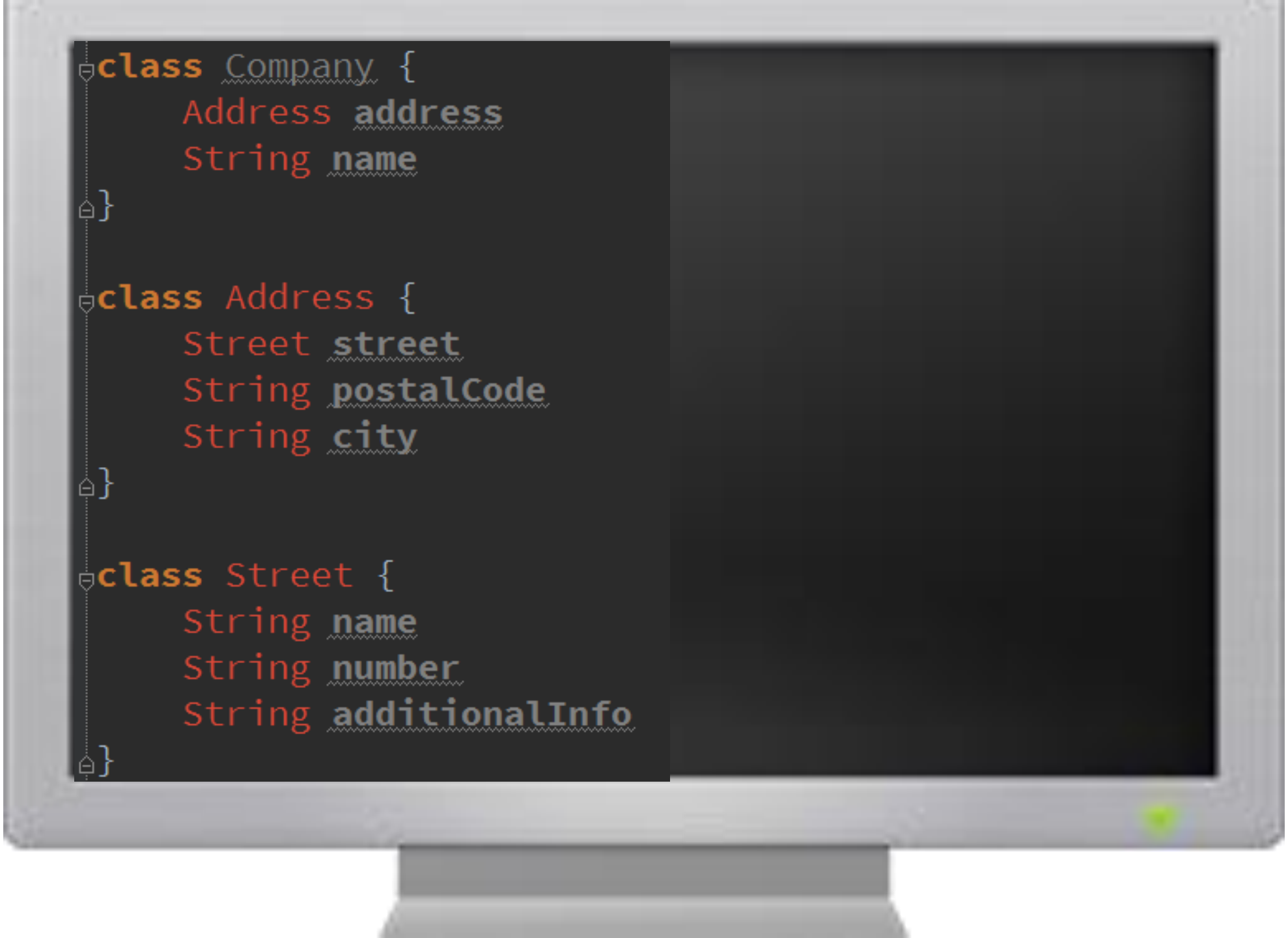
Kscope16 — vi cliBuild.groovy — 80×25

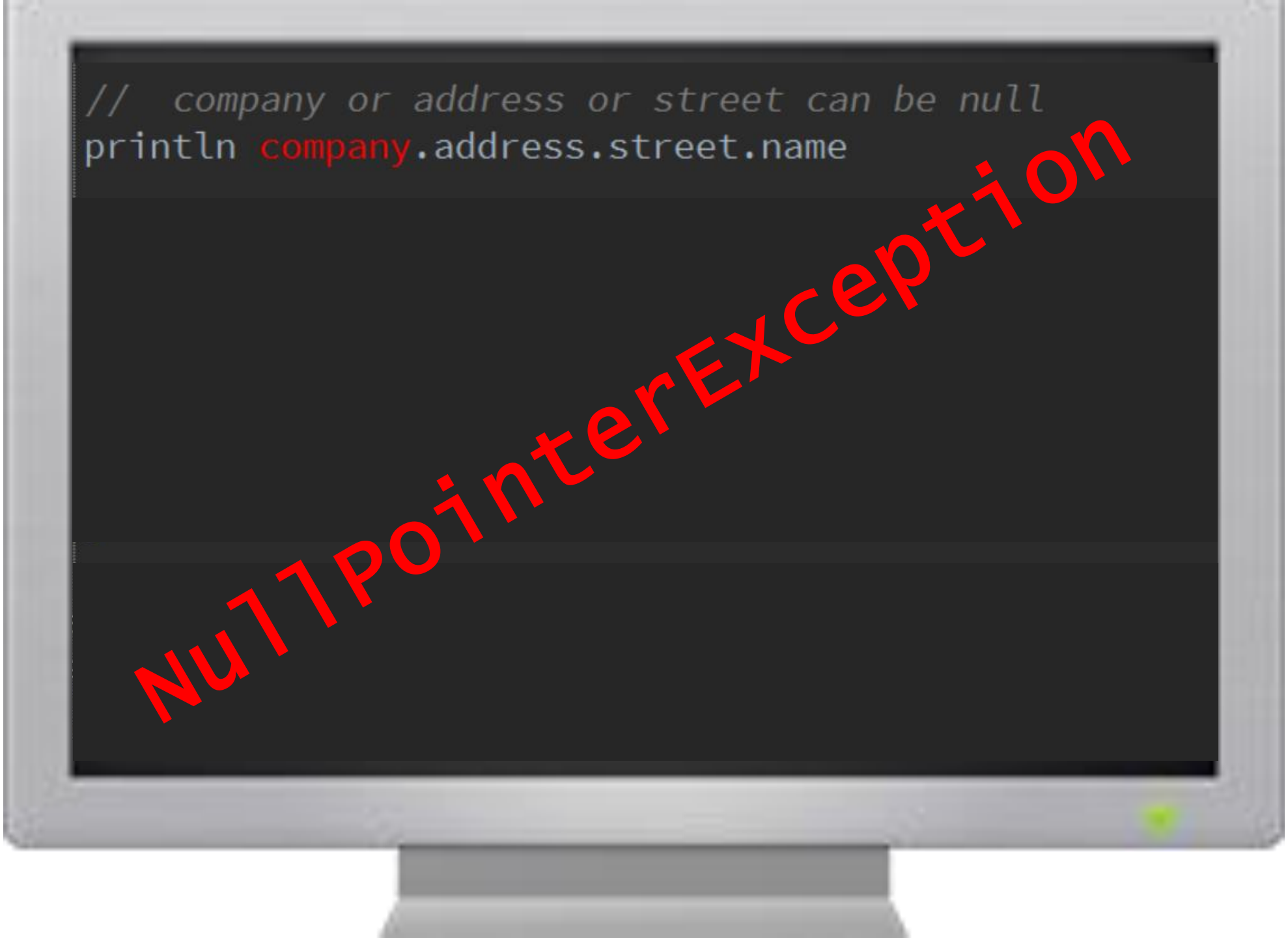
```
def cli = new CliBuilder(usage: 'cliBuild.groovy -[adhms] -x Xval -y Yval')

cli.with {
        h longOpt: 'help', 'Show usage information'
        a longOpt: 'add', 'Xval + Yval (default)'
        s longOpt: 'subtract', 'Xval - Yval'
        m longOpt: 'multiply', 'Xval * Yval'
        d longOpt: 'divide', 'Xval / Yval'
        x longOpt: 'X', required: true, args:1, 'Xval'
        y longOpt: 'Y',  required: true, args:1, 'Yval'
}

def options = cli.parse(args)
if (options) {
if (options?.h) {
  cli.usage()
}
if (options.x && options.y) {
x = options.x.toInteger()
y = options.y.toInteger()

if (options.s) println x - y
else if (options.m) println x * y
else if (options.d) println x / y
else println x + y
} }
```

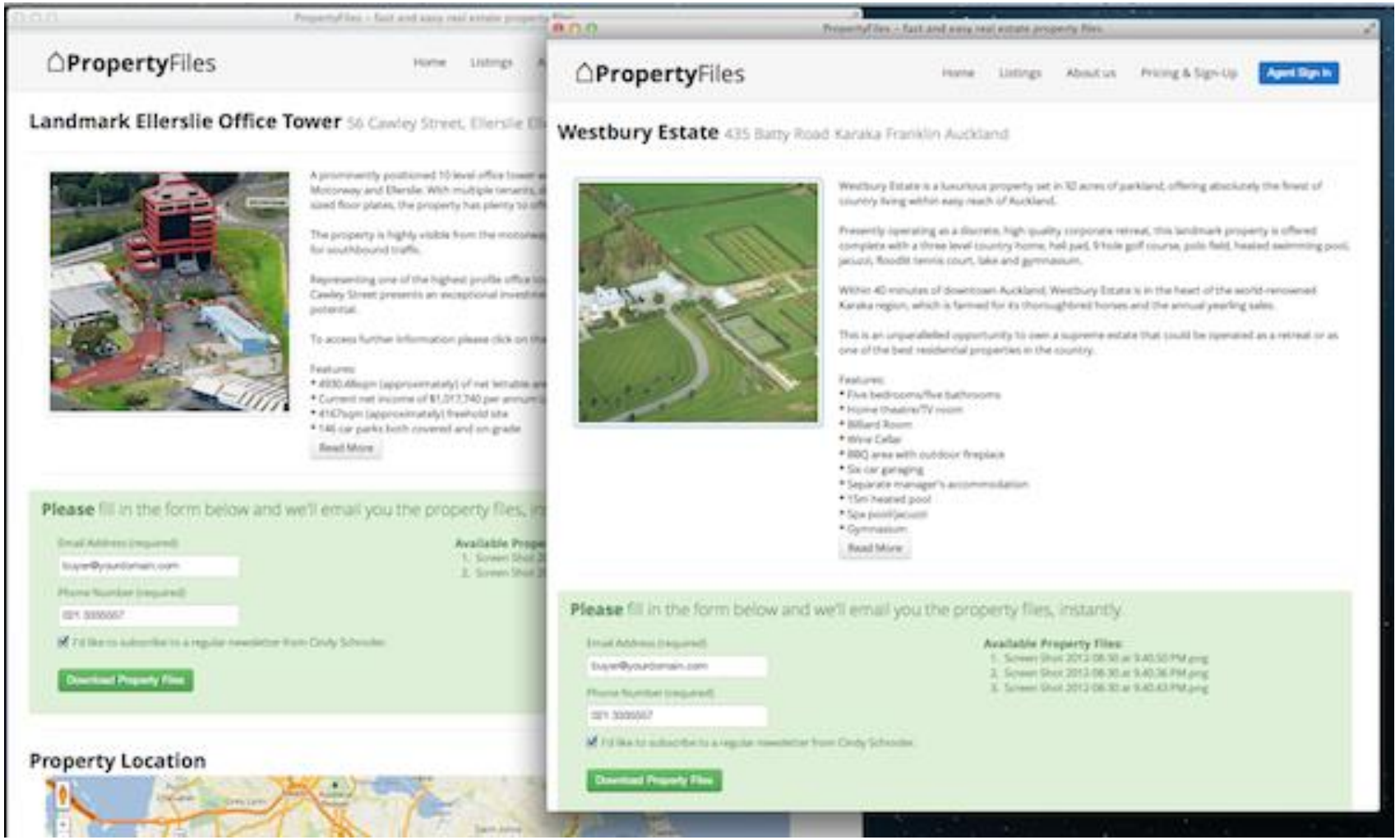
Extra credit?


What is
Groovy
TRUTH?
BEAUTY.
Empty
Zero
Null


Safe Navigation


class Company {
Address address
String name
}
class Address {
Street street
String postalCode
String city
}
class Street {
String name
String number
String additionalInfo
}


// company or address or street can be null
println company.address.street.name
NullPointerException



Property files are typically simple text or XML

my.properties:

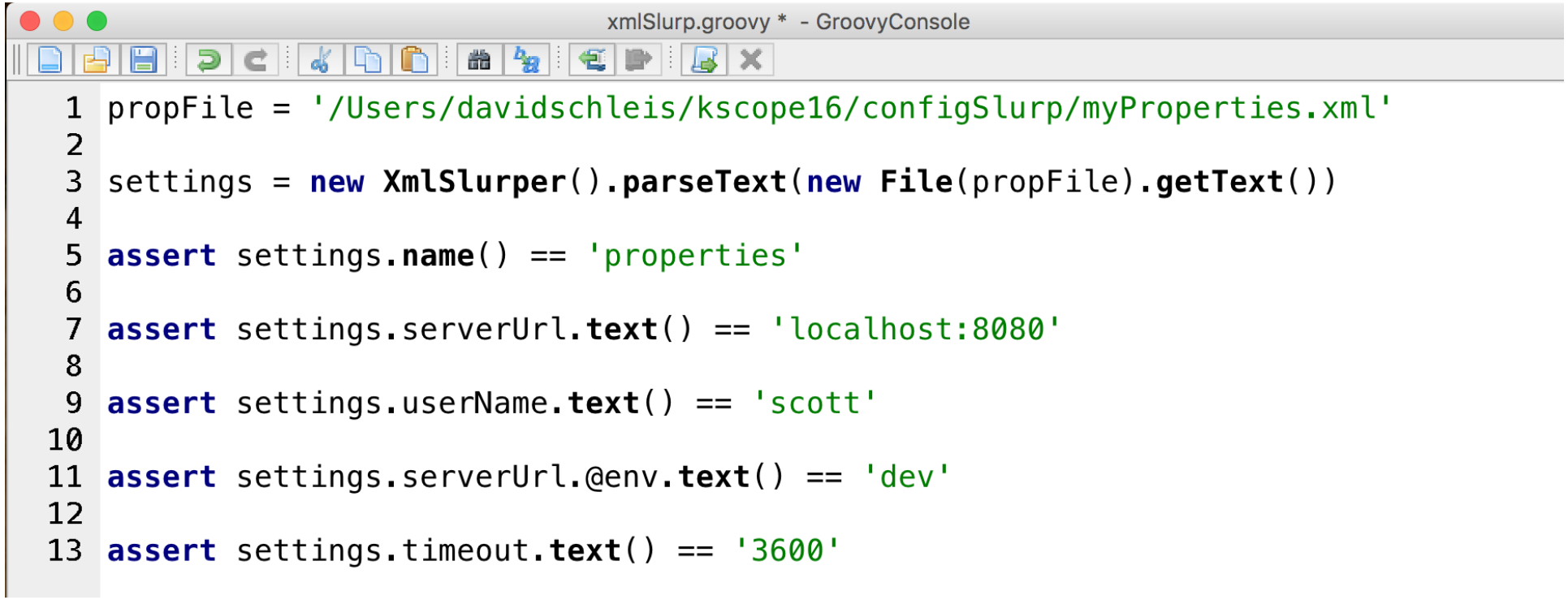
```
serverUrl=localhost:8080
userName=scott
Password=tiger
timeout=3600
```

myProperties.xml:

```
<properties>
  <serverUrl env='dev'>
    localhost:8080</serverUrl>
  <userName>scott</userName>
  <password>tiger</password>
  <timeout>3600</timeout>
</properties>
```

<?xml?>

Slurper



myProperties.xml:

```
<properties>
  <serverUrl env='dev'>localhost:8080</serverUrl>
  <userName>scott</userName>
  <password>tiger</password>
  <timeout>3600</timeout>
</properties>
```

xmlSlurp.groovy - GroovyConsole

```groovy
propFile = '/Users/davidschleis/kscope16/configSlurp/myProperties.xml'

settings = new XmlSlurper().parseText(new File(propFile).getText())

assert settings.name() == 'properties'

timeout1 = settings.timeout.text()

assert timeout1 + 1000 == '36001000'

timeout2 = settings.timeout.text().toInteger()

assert timeout2 + 1000 == 4600
```


Config
Config
Config
200mL
FRUIT DRINK
25% FruitJuice
ORANGE SUPER
SLURPER
NO PRESERVATIVES
NO ADDED COLOURS

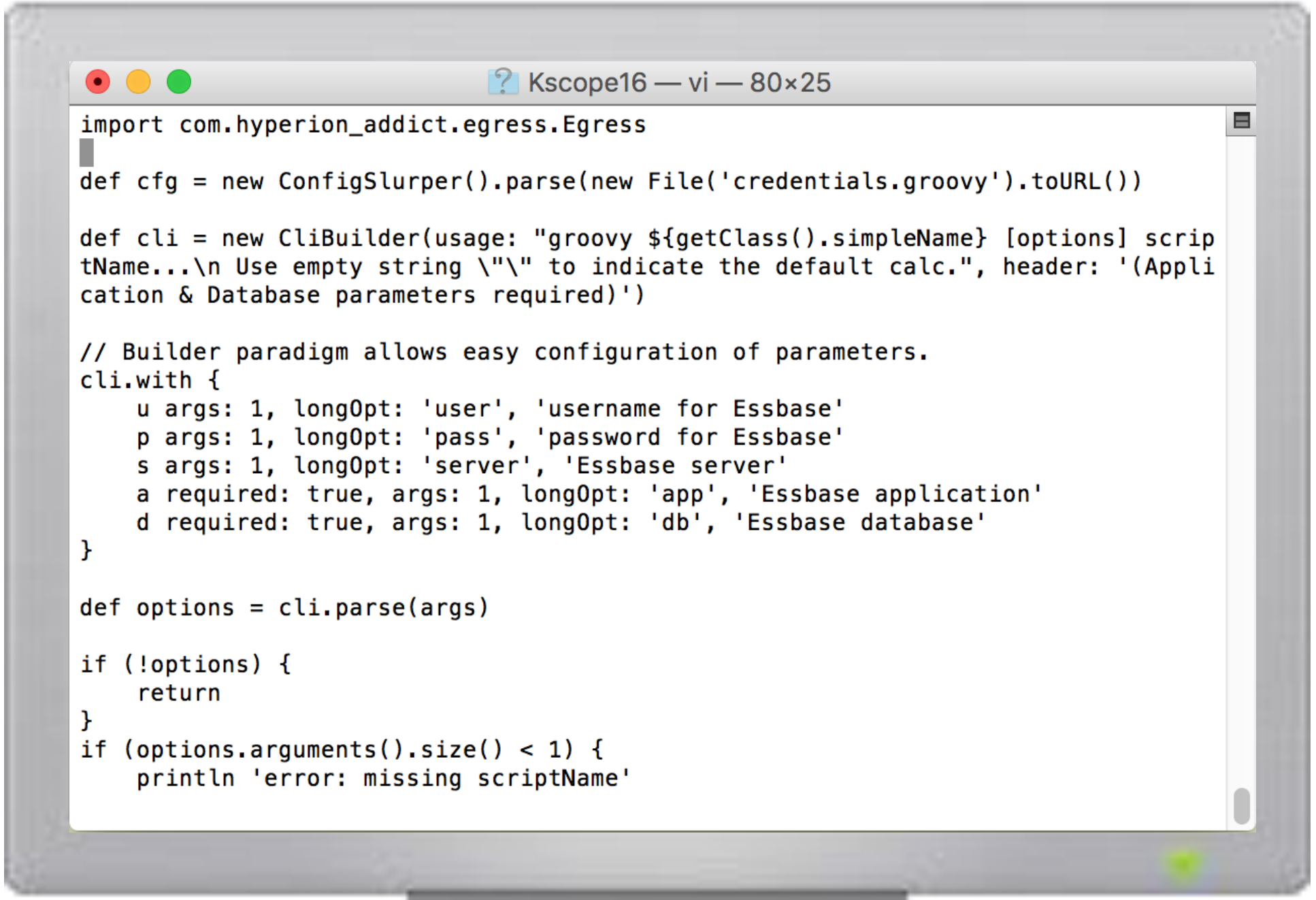
Kscope16 — vi configFile.groovy — 101×25

```
general {
  theList = ['the', 'list', 1, 2, 3, 4]
  username = 'ess_admin'
}
environments {
  dev {
    server = 'localhost'
    password = 'passw0rd'
  }
  test {
    server = 'otherhost'
    password = 'pa$$w0rd'
  }
}
~
```

configTest.groovy * - GroovyConsole

```
 1 configFile = "/Users/davidschleis/Dropbox/kscope16/configFile.groovy"
 2
 3 config = new ConfigSlurper('test').parse(new File(configFile).toURL())
 4
 5 assert config.general.theList.class == ArrayList
 6 assert config.general.theList[2] == 1
 7 assert config.general.theList[2].class == java.lang.Integer
 8
 9 assert config.general.username == 'ess_admin'
10 assert config.password == 'pa$$w0rd'
11 assert config.server == 'otherhost'
```

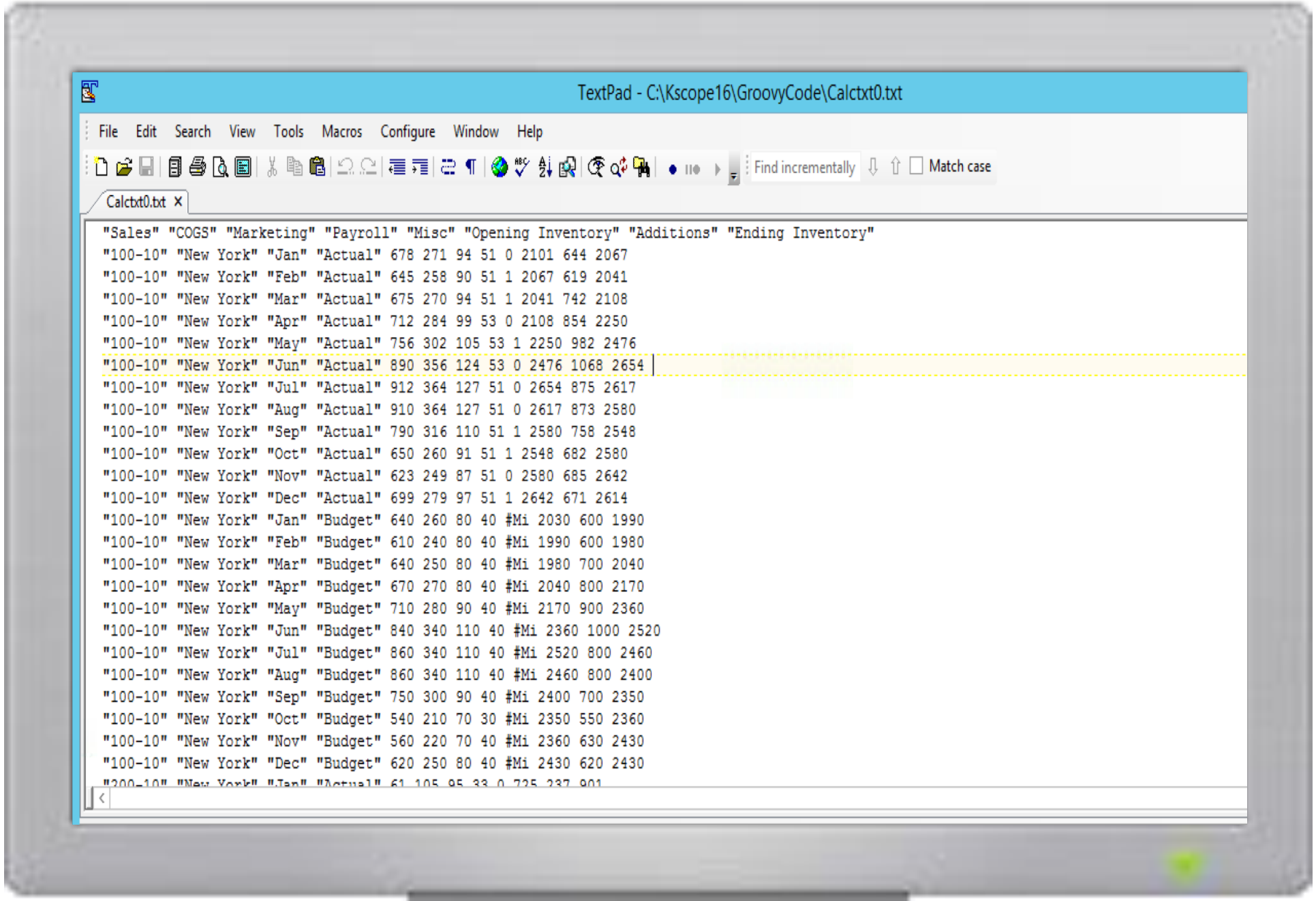
Exercise 7 (8 min.)



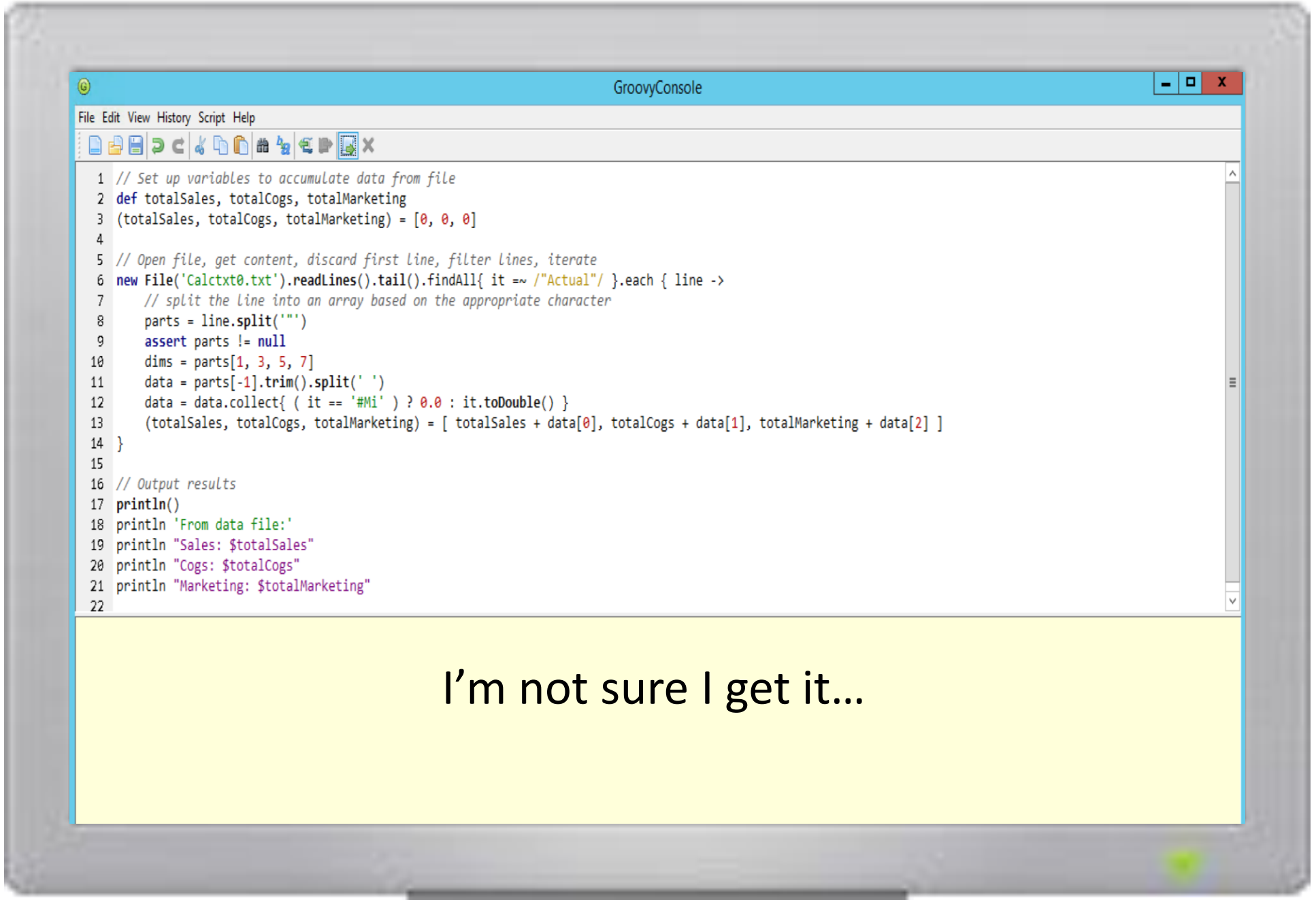
Exercise 8

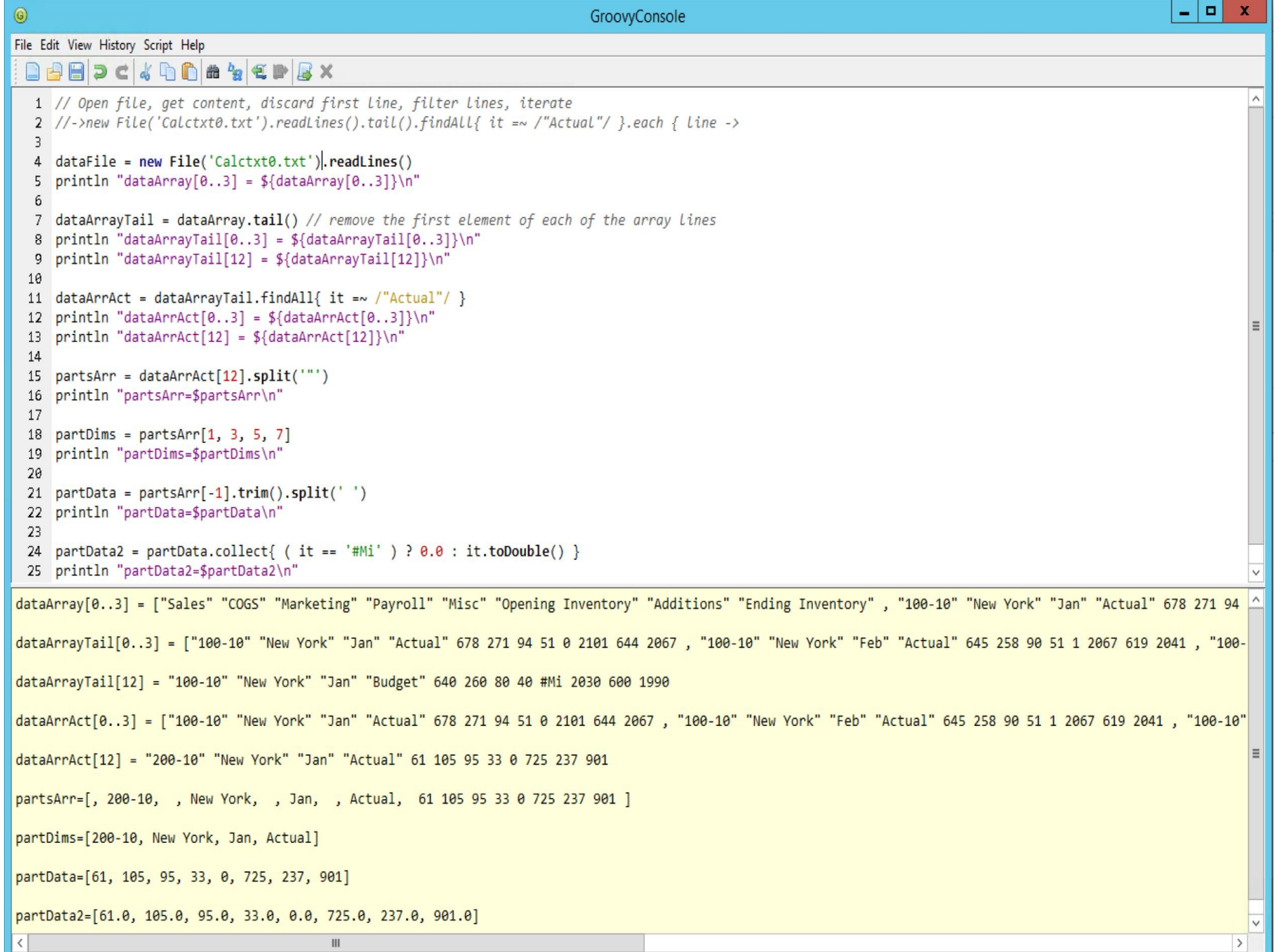
Load and Verify

Exercise 8

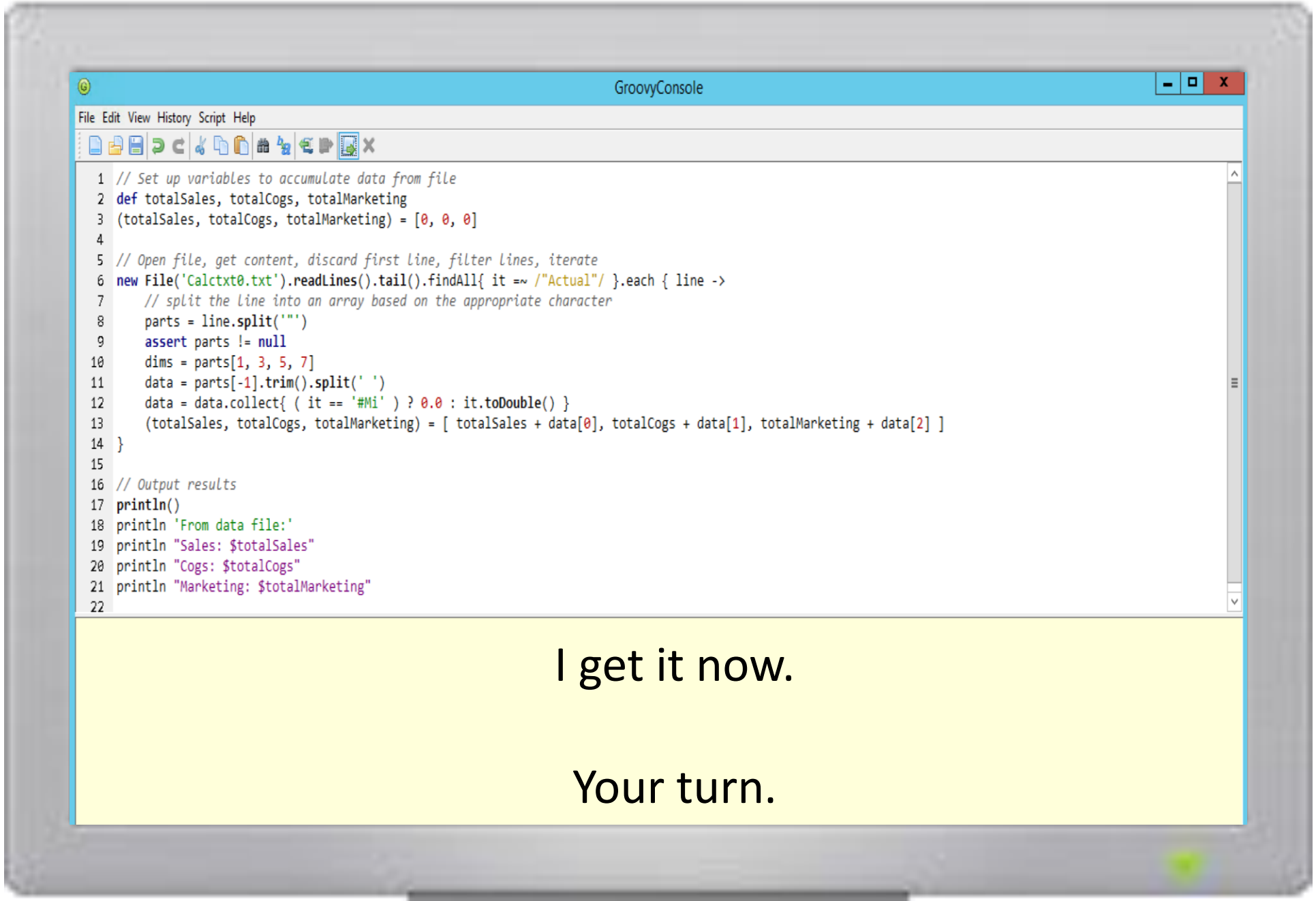

TextPad - C:\Kscope16\GroovyCode\Calctxt0.txt
File Edit Search View Tools Macros Configure Window Help
Find incrementally
Match case
Calctxt0.txt
"Sales" "COGS" "Marketing" "Payroll" "Misc" "Opening Inventory" "Additions" "Ending Inventory"
"100-10" "New York" "Jan" "Actual" 678 271 94 51 0 2101 644 2067
"100-10" "New York" "Feb" "Actual" 645 258 90 51 1 2067 619 2041
"100-10" "New York" "Mar" "Actual" 675 270 94 51 1 2041 742 2108
"100-10" "New York" "Apr" "Actual" 712 284 99 53 0 2108 854 2250
"100-10" "New York" "May" "Actual" 756 302 105 53 1 2250 982 2476
"100-10" "New York" "Jun" "Actual" 890 356 124 53 0 2476 1068 2654
"100-10" "New York" "Jul" "Actual" 912 364 127 51 0 2654 875 2617
"100-10" "New York" "Aug" "Actual" 910 364 127 51 0 2617 873 2580
"100-10" "New York" "Sep" "Actual" 790 316 110 51 1 2580 758 2548
"100-10" "New York" "Oct" "Actual" 650 260 91 51 1 2548 682 2580
"100-10" "New York" "Nov" "Actual" 623 249 87 51 0 2580 685 2642
"100-10" "New York" "Dec" "Actual" 699 279 97 51 1 2642 671 2614
"100-10" "New York" "Jan" "Budget" 640 260 80 40 #Mi 2030 600 1990
"100-10" "New York" "Feb" "Budget" 610 240 80 40 #Mi 1990 600 1980
"100-10" "New York" "Mar" "Budget" 640 250 80 40 #Mi 1980 700 2040
"100-10" "New York" "Apr" "Budget" 670 270 80 40 #Mi 2040 800 2170
"100-10" "New York" "May" "Budget" 710 280 90 40 #Mi 2170 900 2360
"100-10" "New York" "Jun" "Budget" 840 340 110 40 #Mi 2360 1000 2520
"100-10" "New York" "Jul" "Budget" 860 340 110 40 #Mi 2520 800 2460
"100-10" "New York" "Aug" "Budget" 860 340 110 40 #Mi 2460 800 2400
"100-10" "New York" "Sep" "Budget" 750 300 90 40 #Mi 2400 700 2350
"100-10" "New York" "Oct" "Budget" 540 210 70 30 #Mi 2350 550 2360
"100-10" "New York" "Nov" "Budget" 560 220 70 40 #Mi 2360 630 2430
"100-10" "New York" "Dec" "Budget" 620 250 80 40 #Mi 2430 620 2430

Exercise 8

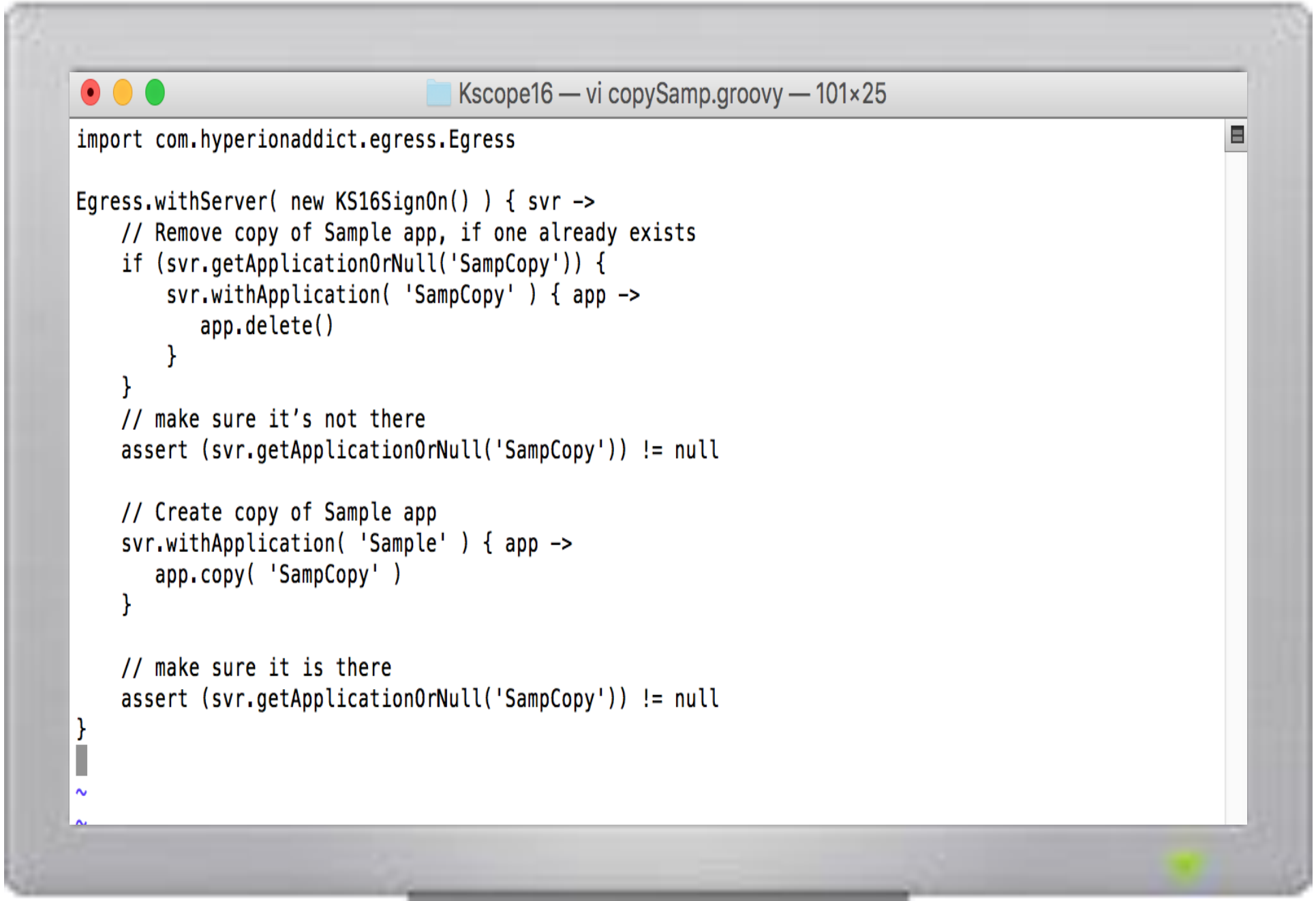

GroovyConsole
File Edit View History Script Help
1 // Set up variables to accumulate data from file
2 def totalSales, totalCogs, totalMarketing
3 (totalSales, totalCogs, totalMarketing) = [0, 0, 0]
4
5 // Open file, get content, discard first line, filter lines, iterate
6 new File('Calctxt0.txt').readLines().tail().findAll{ it =~ /"Actual"/ }.each { line ->
7 // split the line into an array based on the appropriate character
8 parts = line.split('"')
9 assert parts != null
10 dims = parts[1, 3, 5, 7]
11 data = parts[-1].trim().split(' ')
12 data = data.collect{ (it == '#Mi') ? 0.0 : it.toDouble() }
13 (totalSales, totalCogs, totalMarketing) = [totalSales + data[0], totalCogs + data[1], totalMarketing + data[2]]
14 }
15
16 // Output results
17 println()
18 println 'From data file:'
19 println "Sales: $totalSales"
20 println "Cogs: $totalCogs"
21 println "Marketing: $totalMarketing"
22
I’m not sure I get it…


GroovyConsole
File Edit View History Script Help
// Open file, get content, discard first line, filter lines, iterate
//->new File('Calctxt0.txt').readLines().tail().findAll{ it =~ /"Actual"/ }.each { line ->
dataFile = new File('Calctxt0.txt').readLines()
println "dataArray[0..3] = ${dataArray[0..3]}\n"
dataArrayTail = dataArray.tail() // remove the first element of each of the array lines
println "dataArrayTail[0..3] = ${dataArrayTail[0..3]}\n"
println "dataArrayTail[12] = ${dataArrayTail[12]}\n"
dataArrAct = dataArrayTail.findAll{ it =~ /"Actual"/ }
println "dataArrAct[0..3] = ${dataArrAct[0..3]}\n"
println "dataArrAct[12] = ${dataArrAct[12]}\n"
partsArr = dataArrAct[12].split('"')
println "partsArr=$partsArr\n"
partDims = partsArr[1, 3, 5, 7]
println "partDims=$partDims\n"
partData = partsArr[-1].trim().split(' ')
println "partData=$partData\n"
partData2 = partData.collect{ (it == '#Mi') ? 0.0 : it.toDouble() }
println "partData2=$partData2\n"
dataArray[0..3] = ["Sales" "COGS" "Marketing" "Payroll" "Misc" "Opening Inventory" "Additions" "Ending Inventory" , "100-10" "New York" "Jan" "Actual" 678 271 94
dataArrayTail[0..3] = ["100-10" "New York" "Jan" "Actual" 678 271 94 51 0 2101 644 2067 , "100-10" "New York" "Feb" "Actual" 645 258 90 51 1 2067 619 2041 , "100-
dataArrayTail[12] = "100-10" "New York" "Jan" "Budget" 640 260 80 40 #Mi 2030 600 1990
dataArrAct[0..3] = ["100-10" "New York" "Jan" "Actual" 678 271 94 51 0 2101 644 2067 , "100-10" "New York" "Feb" "Actual" 645 258 90 51 1 2067 619 2041 , "100-10"
dataArrAct[12] = "200-10" "New York" "Jan" "Actual" 61 105 95 33 0 725 237 901
partsArr=[, 200-10, , New York, , Jan, , Actual, 61 105 95 33 0 725 237 901]
partDims=[200-10, New York, Jan, Actual]
partData=[61, 105, 95, 33, 0, 725, 237, 901]
partData2=[61.0, 105.0, 95.0, 33.0, 0.0, 725.0, 237.0, 901.0]

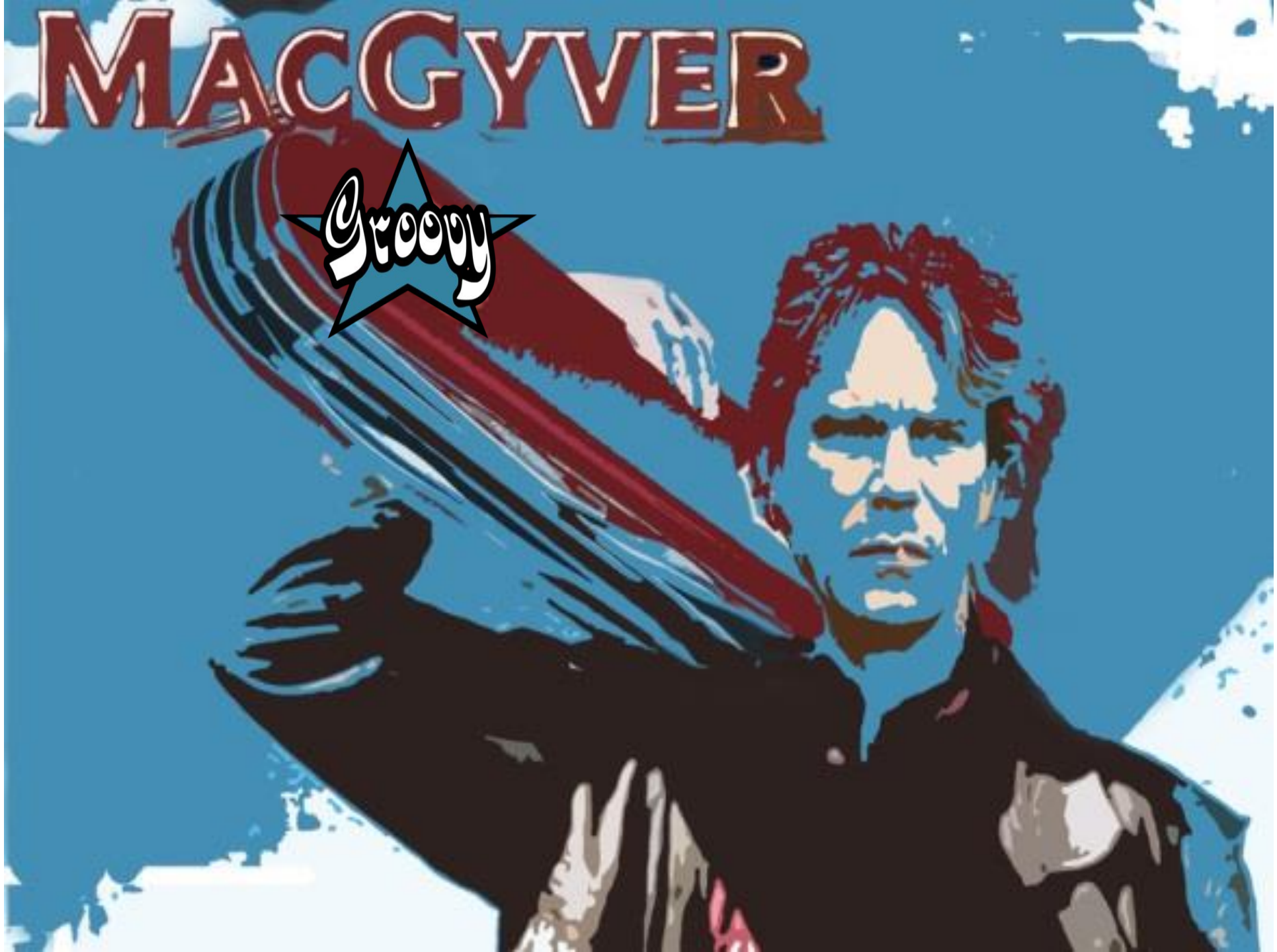
Exercise 8 (15 min)



Exercise 9 (15 min.)



Exercise 10 (15 min)


MacGyver
Groovy