**The Groovy Programming Language**

****

Brandon Lingo, Hasani Mason

November 9th, 2017

CSCI 3370 Programming Languages

Dr. Vamsi Paruchuri

Department of Computer Science

University of Central Arkansas

# Table of Contents

Table of Contents

**Section 1 Introduction 3**

1.1 Goal of Project3

1.2 How does report achieve goal3

**Section 2 Basic Features4**

2.1 Introduction4

2.2 Paradigms4

2.3 Data Type10

2.4 Control Flow10

2.5 Subprograms12

2.6 Object Orientation18

**Section 4 History18**

4.1 Introduction18

4.2 Creation18

4.3 Lifetime18

**Section 5 Tutorials21**

5.1 Introduction22

5.2 Basic “Hello World”22

5.3 Major Features22

5.X Continued Features22

**Section 6 Medium Program Implementation25**

6.1 Introduction25

6.2 Basic Design25

6.3 Explanation25

6.X Conclusions25

**Section 7 Evaluation4**

7.1 Introduction26

7.2 Simplicity/orthogonality 26

7.3 Control Structures26

7.4 Data types26

7.5 Syntax Design26

7.6 Support for abstraction26

7.7 Expressivity26

7.8 Type checking26

7.9 Exception handling27

7.10 Restricted aliasing27

7.11 Conclusion 27

**Section 8 Reflection28**

8.1 Introduction28

# 

# 

# 

# 

# 

**Introduction:**

**1.1**

The goal of our project is to be able to learn and understand a different programming language outside of the UCA Computer Science required languages. We decided to choose Groovy because we wanted to learn a high level language that uses simplicity and allows developers to use it as very powerful tool. The goal of this report is not only to be able to learn a new language in 2-4 weeks so that we can use the Computer Science fundamentals to show that we have developed the skill to learn a new language in a reasonable time frame.

**1.2**

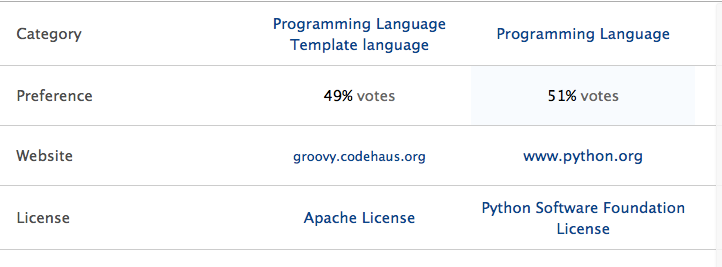
This report achieves this goal by analyzing the history, showing a strong tutorial, the basic features, and medium program implementation. Each section dissects and explains Groovy for people who are beginners with the framework. This report is concise and straight to the point for the courtesy of the reader.

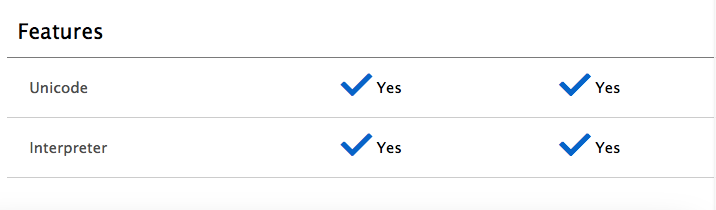
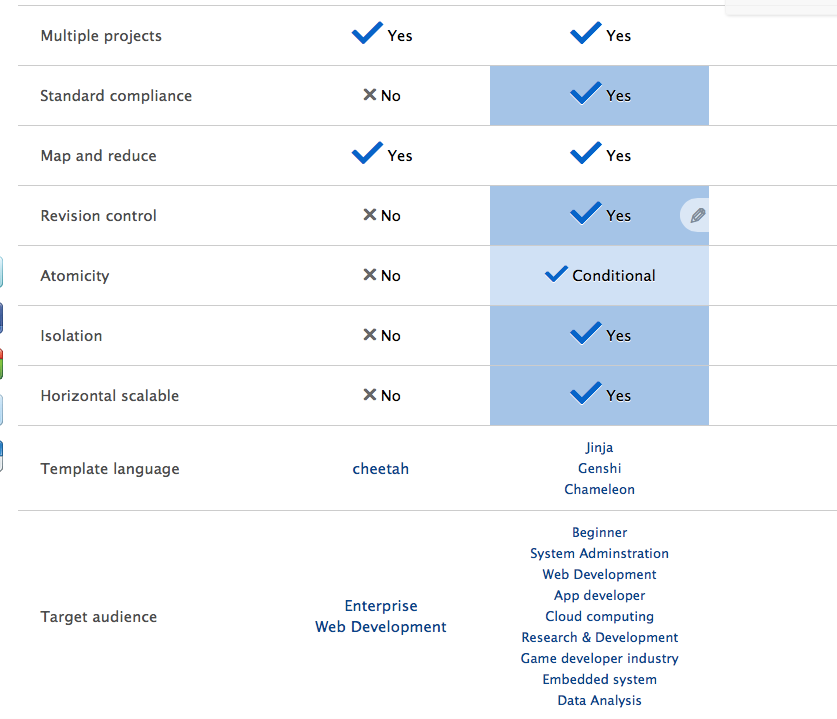
**Basic Features:**

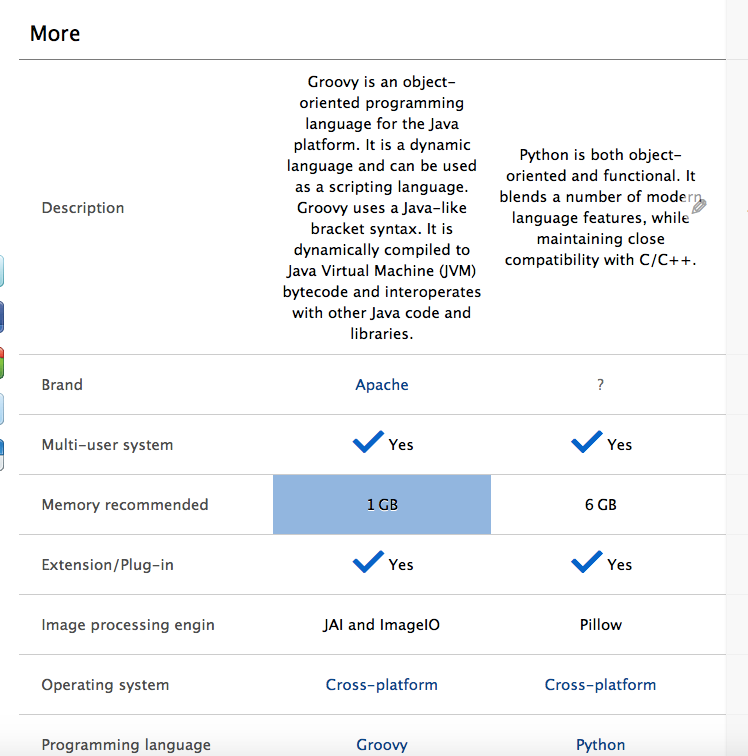
**2.1**

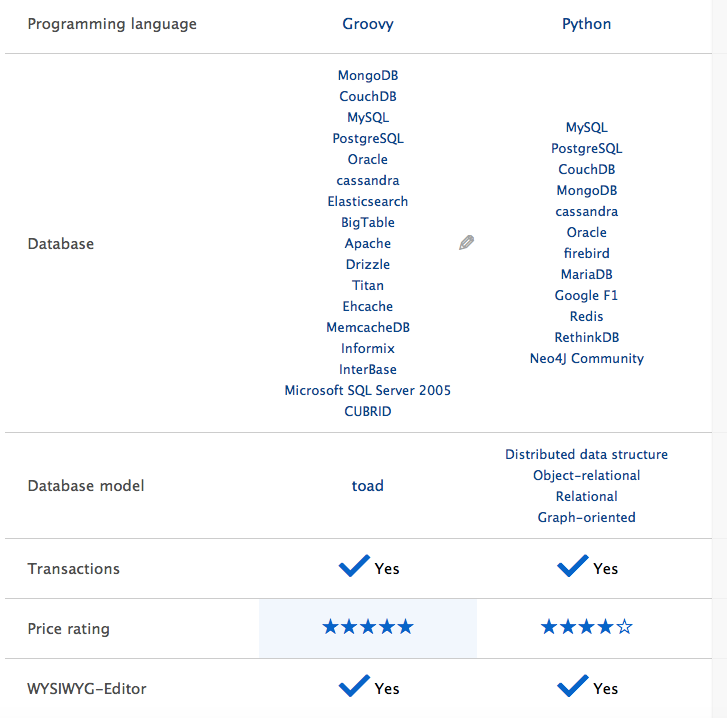
The goal of this section is to outline and explain the basic features of Groovy. We want to highlight the paradigm, data types, control flow, subprograms, and object-orientation.

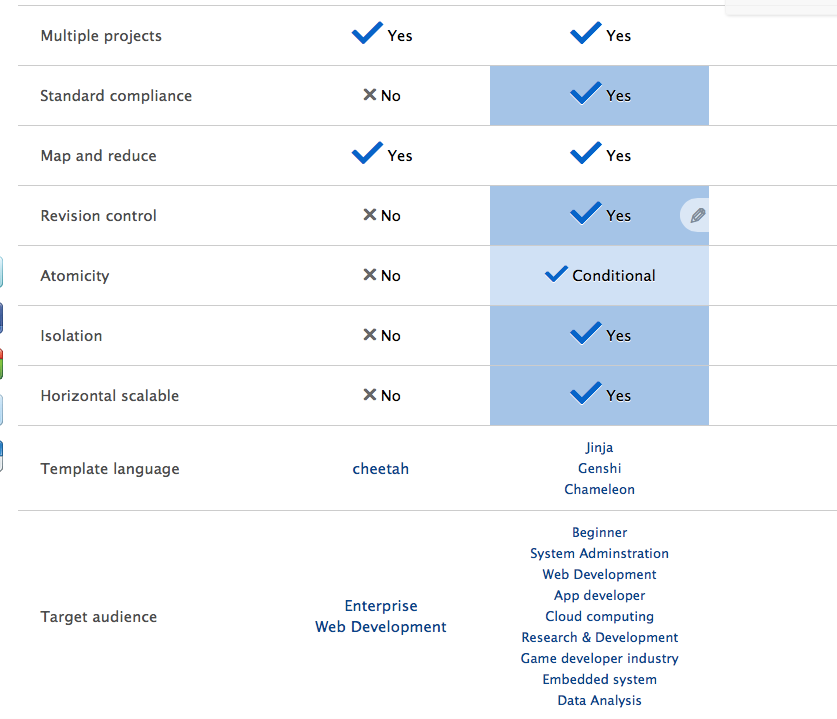
**2.2 Paradigm**

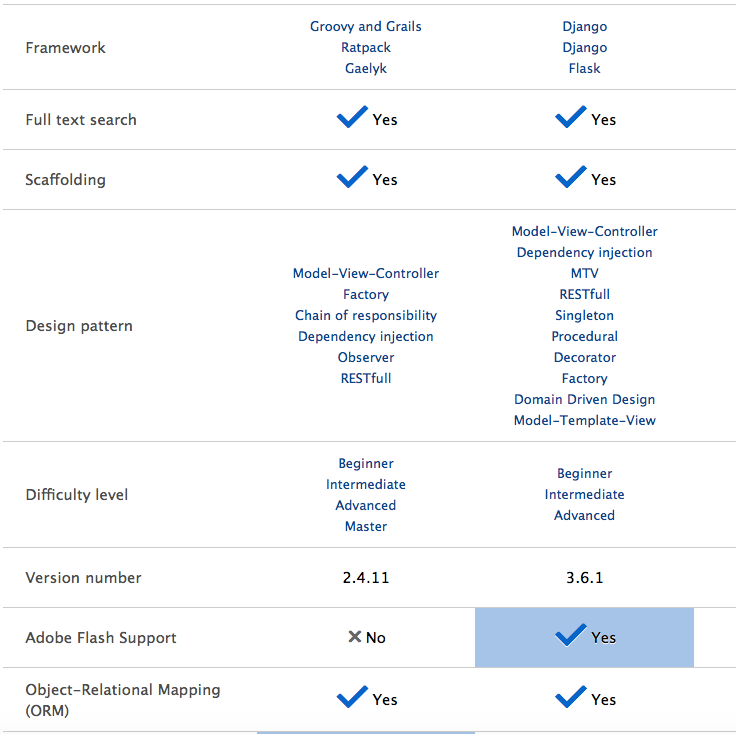
****

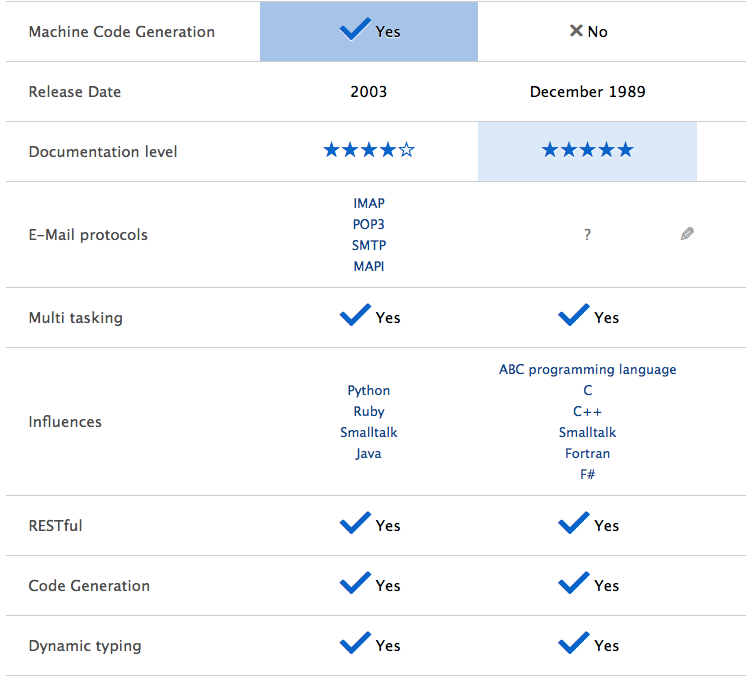
****

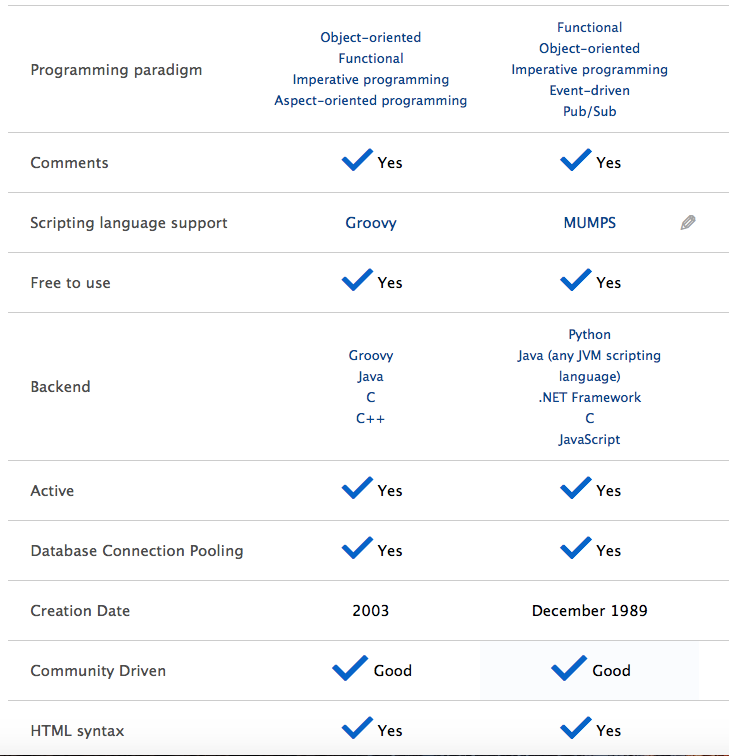
****

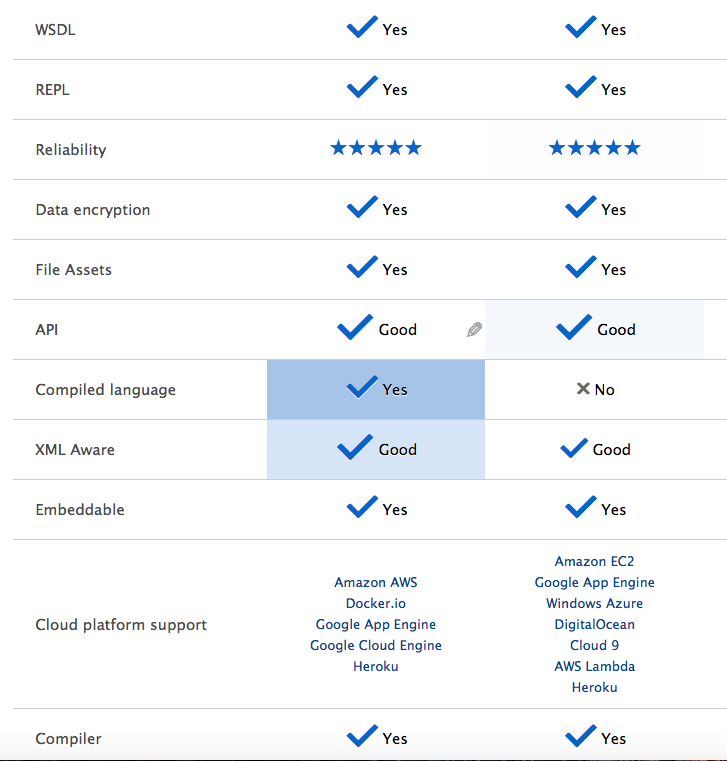
****

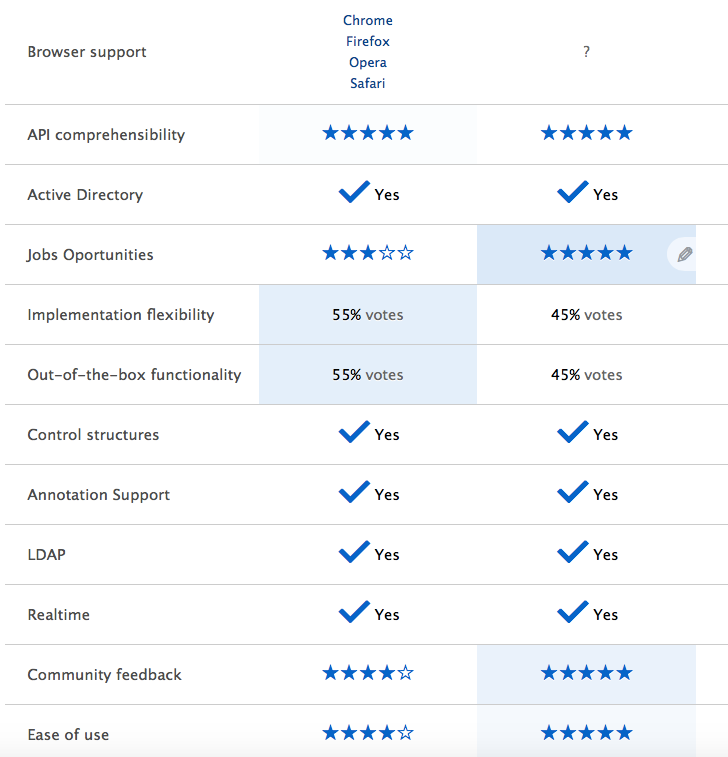
****

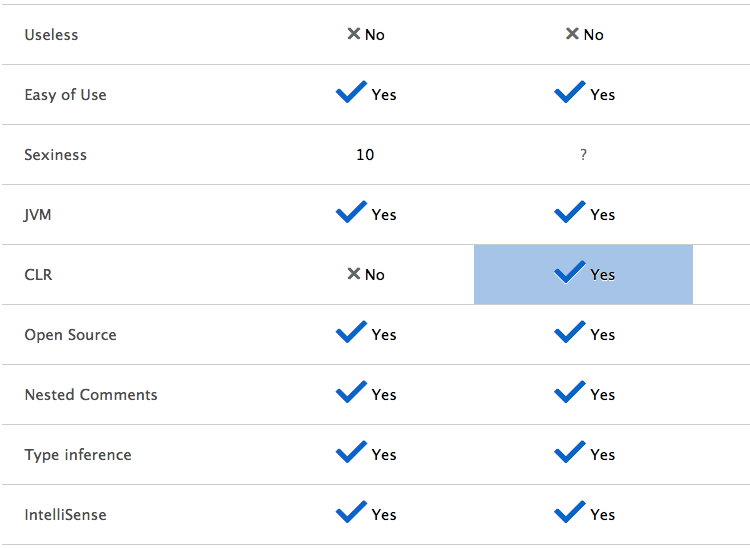
****

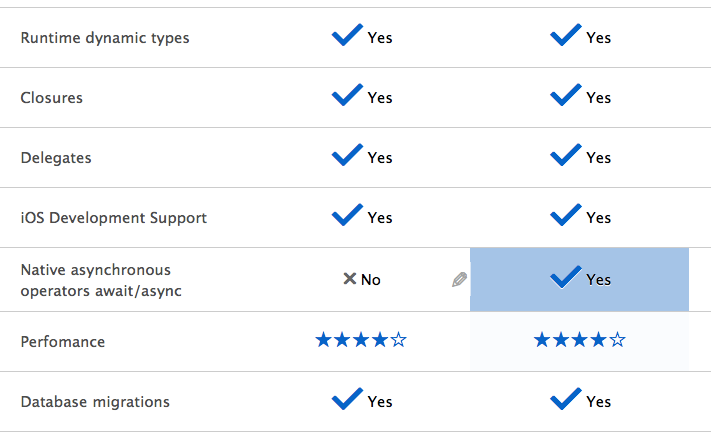
****

****

****

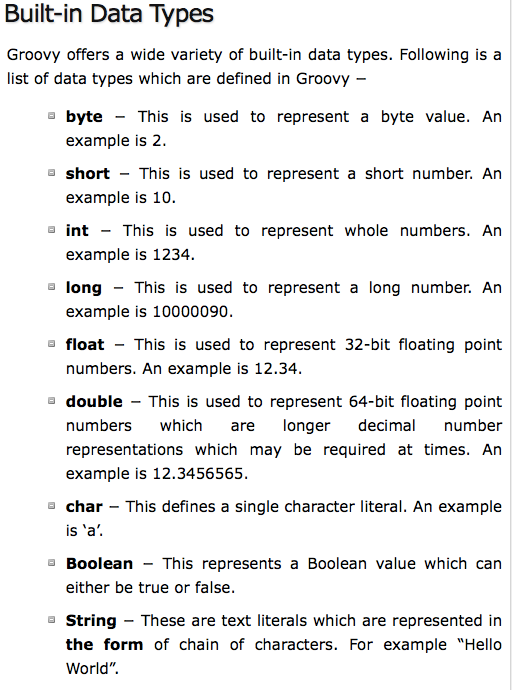
****

****

****

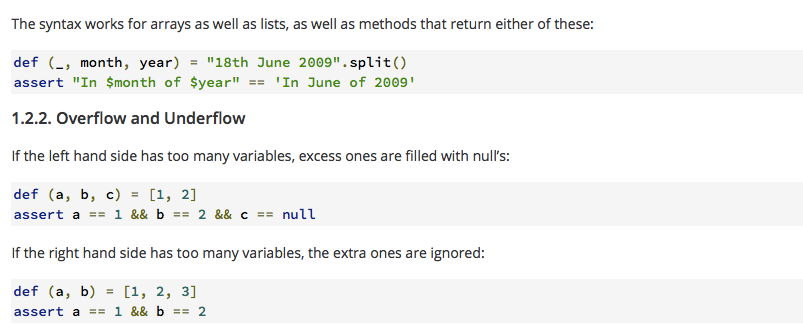
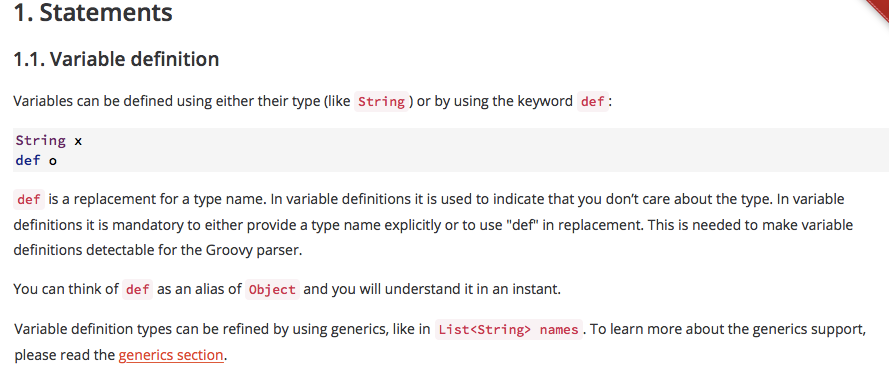
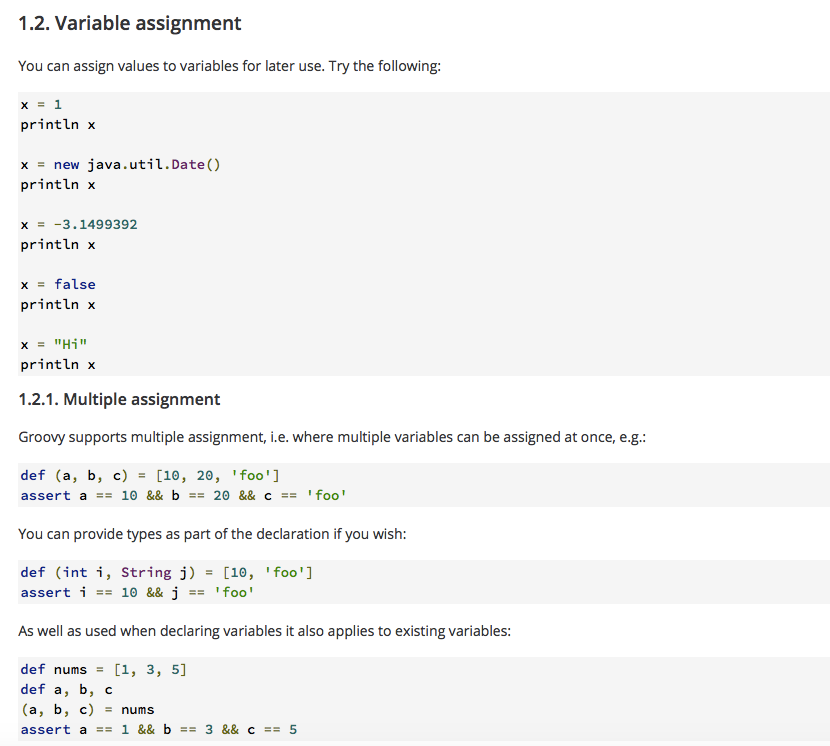
**2.3**

The list of Data Types for the groovy language is similar to higher level languages you might have seen like C++ or Java. The list of data types includes: byte, short, int, long, float, double, char, Boolean, and String.



**2.4**

Next we want to explain the control flow of Groovy. In computer science, **control flow** (or **flow** of **control**) is the order in which individual statements, instructions or function calls of an imperative program are executed or evaluated (Wiki). In Groovy there are many semantics: statements, conditional structures, GPath expressions, number promotion, optionality, Boolean expressions, optional typing, type checking exceptions, and extensions. One example is:



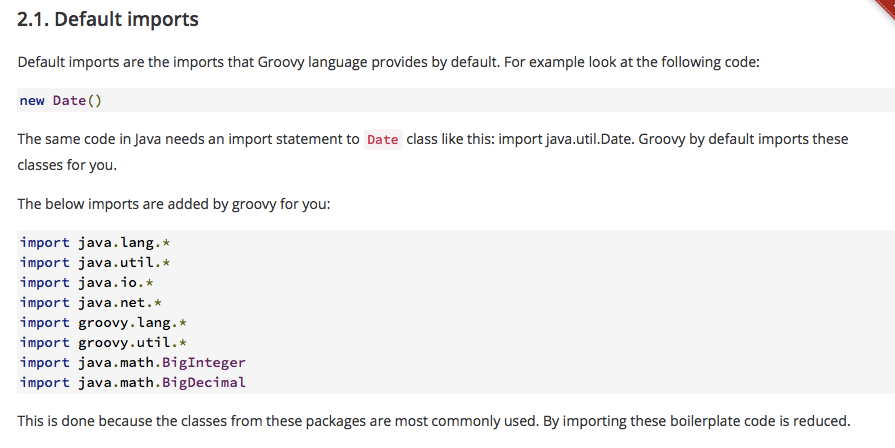
* For sake of time please view this link for full analysis: <http://groovy-lang.org/semantics.html#_statements>

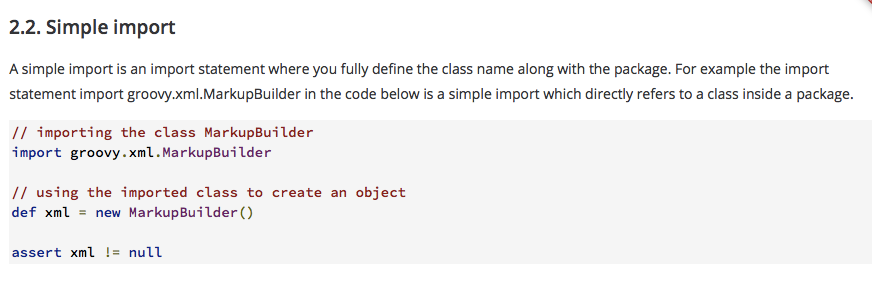
**2.5**

Here is the complete breakdown of Groovy’s Subroutine:

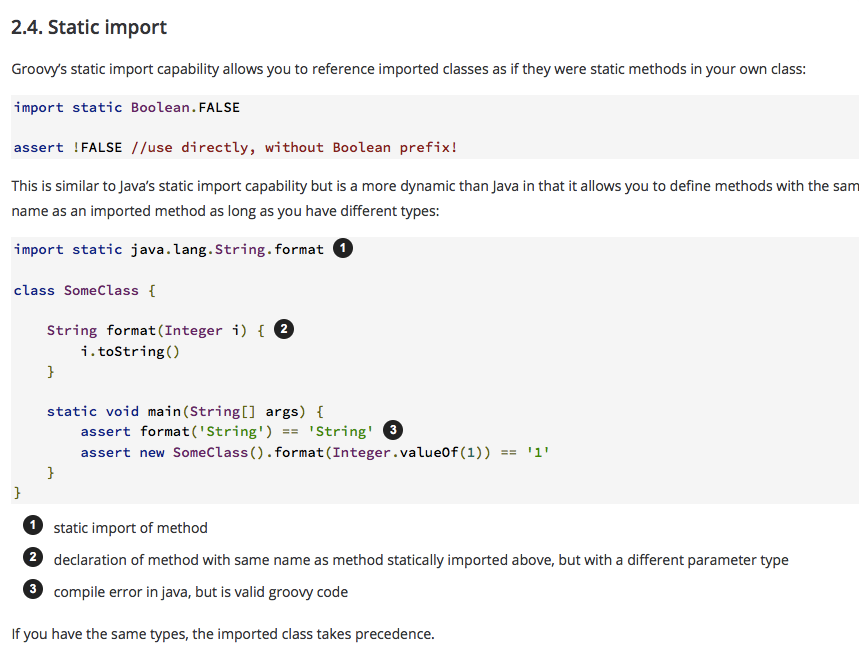


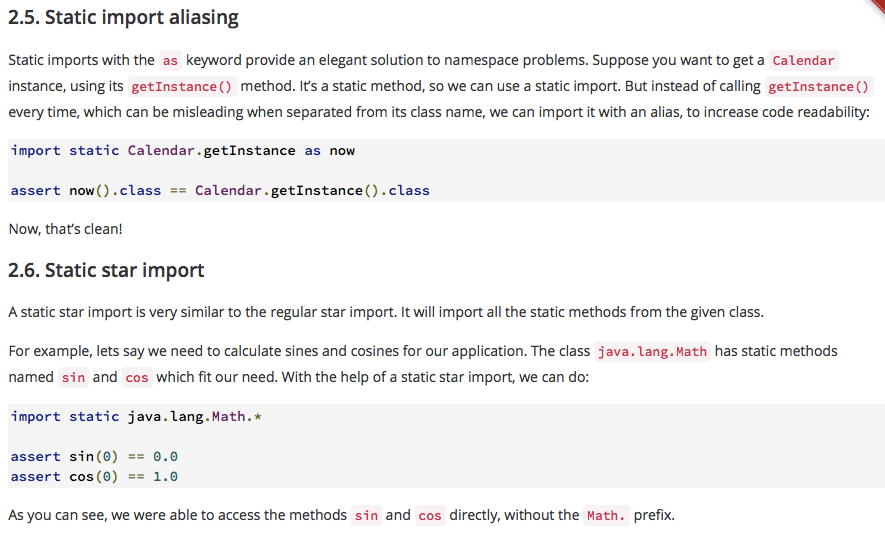


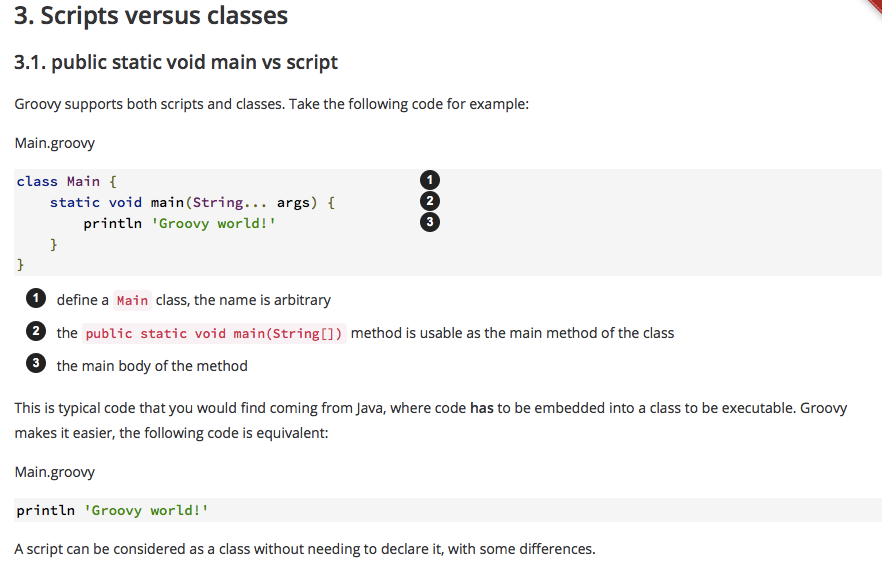


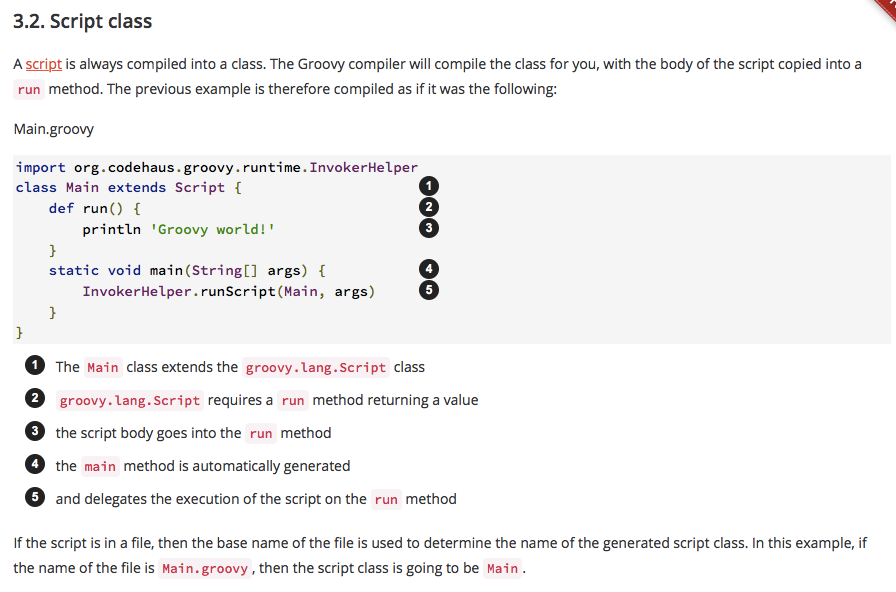


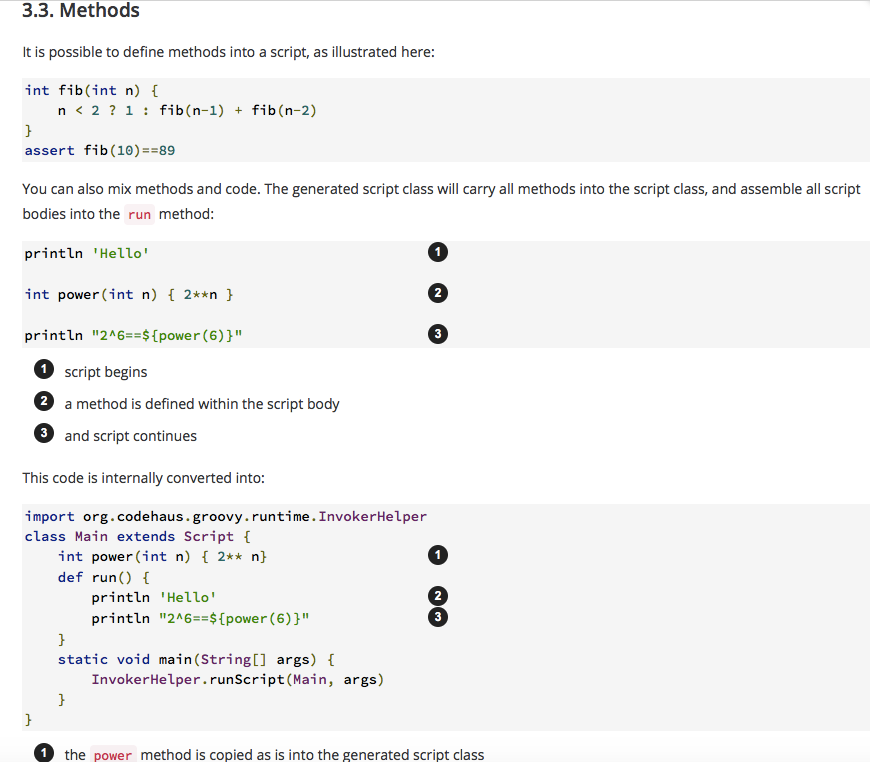


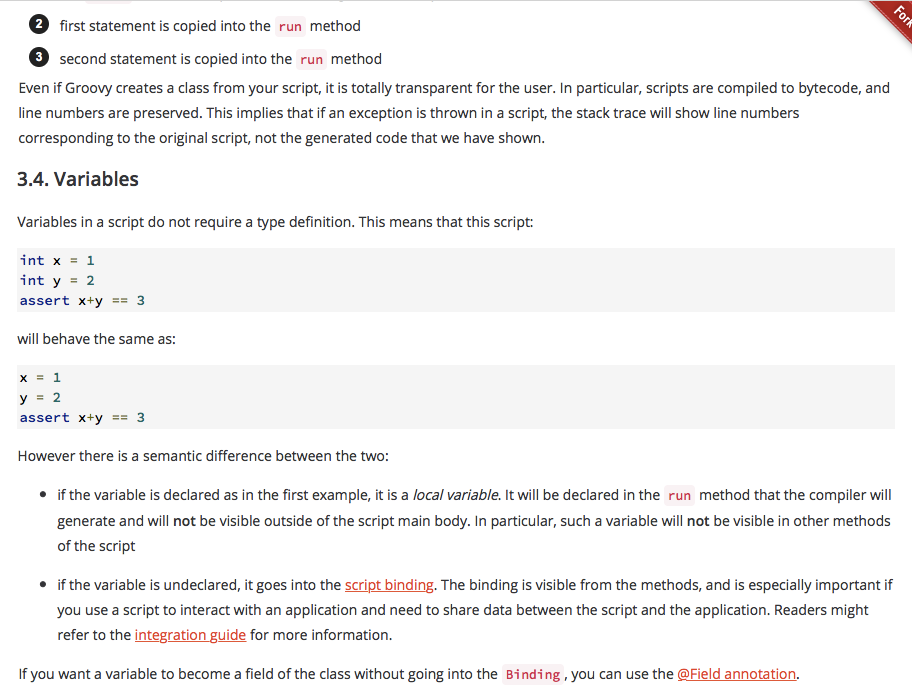












**2.6**

Groovy is an object oriented language based of the language Java. Groovy is a fully fledged object-oriented (OO) language supporting all of the OO programming concepts that are familiar to Java developers: classes, objects, interfaces, inheritance, polymorphism, and others.

Groovy is a pure OO language in which *everything* is an object. Don’t let the fact that you can write scripts in Groovy fool you; the Groovy compiler will convert such scripts into Java classes of type groovy. Lang. Script (Springer).

**History:**

**4.1**

The goal of this section is to develop a strong understanding of the history of Groovy: who created the language, why was it created, what applications is it good for, how has it evolved, and the related popularity to the language.

**4.2**

James Strachan is a software engineer who created the Apache Groovy programming language in 2003. Apache Groovy is an object-oriented dynamic programming language for Java virtual machine (JVM). Groovy was developed to use syntax is simple and easy. It saves a lot of code and effort thus increasing the productivity of developer if he had to do the same thing in Java. Its ease of use where, things can be developed with little effort but ultimately have the same result of Java, because the groovy code is compiled to bytecode which JVM understands.

This can make a developer more productive and it can give them more agility. Groovy also, accepts almost all Java programs, it can use the Java libraries and it even extends some core Java classes as well. It is a framework (xebia). Although the two languages are similar, Groovy code can be more compact, because it does not need all the elements that Java needs, which is a great bonus. This makes it possible for Java programmers to learn Groovy gradually, by starting with familiar Java syntax, before acquiring more Groovy programming styles.

**4.3**

Since Groovy was developed it has evolved to be used in several businesses, IDE packages, systems, appliances, and more. Some of the adoptions are:

* Apache OFBiz, the open-source enterprise resource planning (ERP) system, uses Groovy.
* DataMelt integrates Groovy into a numeric and statistical data-analysis framework
* Eucalyptus, a cloud management system, uses a significant amount of Groovy.
* eXo Platform, an Open-Source Enterprise Social Collaboration Platform uses Groovy.
* Freeplane (freemind), free mind mapping and knowledge building software uses Groovy as scripting language.
* LinkedIn uses Groovy and Grails for some of their subsystems.
* Jenkins, a platform for continuous integration. With version 2, Jenkins includes a *Pipeline* plug-in that allows for build instructions to be written in Groovy.
* Oracle Sales Cloud uses Groovy as the main scripting language to enhance the application.
* Sky.com uses Groovy and Grails to serve massive online media content.
* SmartThings, an open platform for smart homes and the consumer Internet of Things, uses a security-oriented subset of Groovy
* SoapUI provides Groovy as a language for web service tests development.
* Survata, a market research startup, uses Groovy and Grails.
* The European Patent Office (EPO) developed a dataflow programming language in Groovy "to leverage similarities in the processes for communicating with each individual country’s patent office, and transform them into a single, universal process."
* Though Groovy can be integrated into any JVM environment, the JBoss Seam framework provides Groovy, besides Java, as a development language, out of the box.
* vCalc.com uses Groovy for all of the user defined mathematics in its math crowd-sourcing engine.
* Wired.com uses Groovy and Grails for the Product Reviews standalone section of the website.
* XWiki SAS uses Groovy as scripting language in their collaborative open-source product.

For developers several IDE packages have acquired Groovy as well:

* Android Studio, IDE used for making Android apps
* Atom IDE
* DataMelt Java IDE
* Eclipse, through Groovy-Eclipse
* Emacs, using the groovy-emacs-mode project's groovy-mode.
* IntelliJ IDEA, Community Edition, Grails/Griffon in the Ultimate Edition only
* JDeveloper, for use with Oracle ADF
* jEdit, an advanced text editor for the Java platform
* Kate, an advanced text editor for KDE supports Groovy and over 200 other file formats
* NetBeans, since version 6.5
* Notepad++, an advanced text editor for Microsoft Windows
* Sublime Text 2, a cross platform text editor
* TextMate
* Visual Studio Code
* UltraEdit, general purpose program editor

After looking at the two bullet list, we can see a strong connection for Groovy to be a desired language of the future (Wiki). There is something for every developer that wants to use Groovy as an effective tool. When it comes to Groovy’s popularity, it is popular for Java developers, because it has became the perfect complement for general purpose syntax. It introduces interesting feature that Java cannot as well. When it comes to popularity it we give Groovy a 9.25/10. Here are eight reasons why we believe so:

* Support for static and dynamic typing
* Concise, brief, and direct syntax
* Relative short, learning curve similar to Python
* Support for unit testing
* Native support for regular expressions
* Native syntax for list and associative arrays
* Native support for markup languages like XML and HTML
* Support for domain specific languages

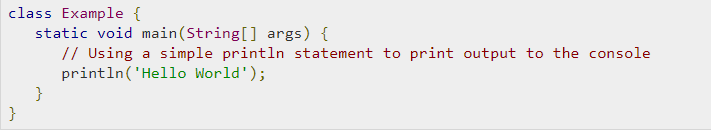
It began as a moderately popular framework until Java Developers discovered its true potential. Currently it is one of the most used frameworks, there have even been frameworks designed based off its technology. One example is Grails, it is an open source web application framework that uses the Apache Groovy programming language. Experts believe in the near future it will be widely used between all Java Developers. Currently many jobs prefer Groovy skills and pay $70,000+ a year for them!

**Tutorial:**

**5.1**

This section is designed to help get you along in the process of getting into groovy as a language. We will show you the basics and how to set it up as well as some harder code later.

**5.2**

****

The code above is the simple hello world function designed to have the computer display “Hello World”. The initial setup is much like java making a class that contains all the functions in this case just main. Println is much like cout from C++ and is the same as System.out.println from Java.

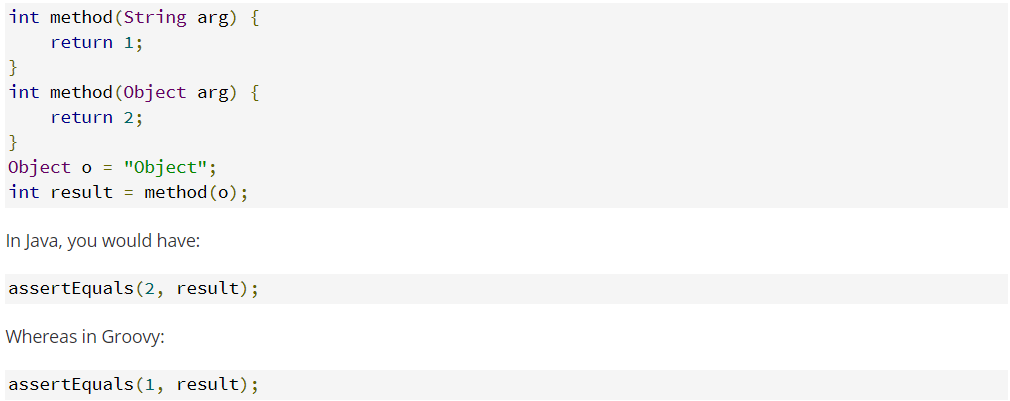
**5.3**

The download is very simple all you need to do is go to <http://groovy-lang.org/download.html> scroll down a little bit and you will see the windows installer. It is under 2.6 version once it is downloaded you select a language. Then a sequence of popups will appear just hit next if you don't want to read it all. Make sure you agree with the terms and then you will choose components. We selected choose full components you need 200 mbs of space to do this. Then it will ask an install location you can use default or select your own. You will next hit then install button and watch it go. Once it finishes hit complete then it will pick your path and path extension. It will add the system to the environment. Continue to hit next until you hit finish and you are done. Open the GroovyConsole and you are ready to get coding (at least the simple ones). Just code in the hello world from above select the script button at the top and hit run script, then bam you have made your first groovy code.

**5.4**

A major feature is the multi-methods or runtime dispatch. The methods which will be invoked are chosen at runtime in Java this is the opposite. This means that the method will be chosen based on the types of the arguments at runtime.

An example of this is provided on <http://groovy-lang.org/differences.html>



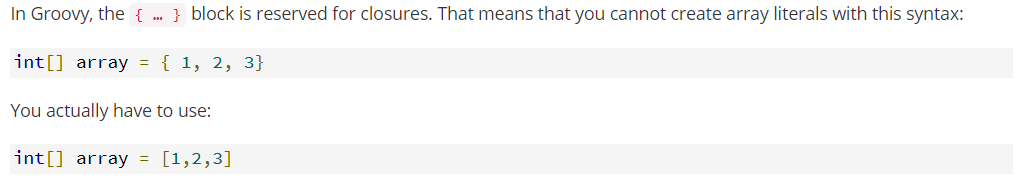
As you can see because the object contains a string groovy sees this and runs the first method instead of just going by the variable type.

**5.5**

You can actually set the variables you create to package scope so you won't need setters and getters. When you create the variable in the method just add @pakagescope in front of the variable type and you are all set.

**5.6**

Arrays are declared a little differently in Groovy. You can’t make an array with space in the declaration.



**5.7**

Even the == is different in java and groovy



The differences range from game changers to just minor tweaks between these two languages, so just like in all coding languages it will come down to preference.

In the interest of time and paper please look at the following source for even more features

<http://groovy-lang.org/differences.html>.

**5.8**

The places to go and find more information to further research the language are:

<http://groovy-lang.org> this is the official website,

<https://www.youtube.com/watch?v=B98jc8hdu9g> a YouTube video to explain some things,

<https://eds.b.ebscohost.com/eds/detail/detail?vid=1&sid=5ab529f0-55bc-4ca7-b603-6d86e94e2874%40sessionmgr120&bdata=JnNpdGU9ZWRzLWxpdmUmc2NvcGU9c2l0ZQ%3d%3d#AN=187394&db=nlebk> this will be the E-Book, and

<https://www.tutorialspoint.com/groovy/groovy_data_types.htm> this has in-depth tutorials on everything in groovy.



**6.1**

The goal of this section is to be able to understand Groovy and to be able to write, implement, and run a Groovy program. The goal of the program we wrote is to be able to sort data using a common algorithm in Computer Science called Merge Sort.

**6.2**

We have a class Merge sort and a structure Merge List. The goal of the class is to sort the data from beginning to end using our desired algorithm using a good runtime. In the structure, we copy the left and right array and merge the list according to biggest values.

**6.3**

The basic design of the program was we had the main that started the domino effect. Main printed the unsorted array and then called sort sort then separated the array and sorted 2 halves of it then it called merge lists. Merge lists did what it sounds like it merged the 2 lists back together while also sorting to make sure the entire thing was in the correct order. The sort function returned the new sorted list back to main then finally main printed it back out.

**6.4**

We used the different style for arrays to accomplish this, a minor change but it wouldn't work without it.

**6.5**

The great thing about this in groovy was how similar it was to java we could easily use what we already knew about how to make the code without setting up the extra packages and other tedious things with java.

**6.6**

The bad things about using groovy was it wasn't as clean as for example papyrus is the ability to just press the dot operator and have a list of functions and methods you can use. It was an annoying thing to not have after becoming so used to it. The output showed the code again and made it seem more cluttered than it was. The error messages were nowhere near as helpful as they are in java.

**6.7**

In conclusion we enjoyed playing around in a new language but the reasons Java and C++ are so popular are made clear however with the simplicity of groovy at times we could see it could become one of those big names in the future.

**Section 7: Evaluation**

**7.1**

The goal of this section is to evaluate groovy based of 9 qualities: simplicity/orthogonality, control structures, data types, syntax design, support for abstraction, expressivity, type checking, exception handling, and restricted aliasing.

**7.2**

When it comes to simplicity Groovy is very similar to Python and Ruby. It is a very powerful language and runs of the platform of Java. It also rivals Ruby with its framework Grails. Developers can get much more done in fewer lines of code.

**7.3**

Groovy’s control structures are built very similar to python, it has a wide variety of built in libraries and functions that allows users to be very creative in a time efficient way.

**7.4**

Groovy has several built in data types that Java and C++ developer can acquire and start building with right away. Even though they are simple they hold big potential.

**7.5**

The syntax and design is a favorite for developers who love simplicity, it really does complete 100 lines of Java in 20 Groovy lines. Well done Groovy team!

**7.6**

When it comes to programming abstraction Groovy has an abundance of open source libraries and built in Java libraries. This allows developers to solve problems with built in tools that Java would not have. Also, users can write Java code with Groovy code and it will run smoothly when it compiles.

**7.7**

Groovy **expressions** are a combination of explicit values, constants, variables, operators, and functions that the **programming** language interprets. They are built in and very simple to use and developer can write code in a timely manner.

**7.8**

Groovy runtime is very fast and simple, when it comes to type checking it meets expectations.

**7.9**

Groovy’s Exception Handling is just like any other high level language, **it** processes the responding to the occurrence, during computation, of **exceptions** – anomalous or exceptional conditions requiring special processing – often changing the normal flow of program execution.

**7.10**

Theability to have the same data accessed with multiple names is similar to C++ or Java. The name doesn't matter as long as you have the address.

**7.11**

My overall feeling about the language is that it is very useful and worth learning if you are someone who wants to develop great software in an efficient way. A way in which you get more results with less code. It would have definitely been more challenging to implement our program in C++ do to longer code segments, semicolons, and prototypes. Groovy eliminates those problems and makes life much easier.

**Section 8: Reflection**

The goal of this section is to look back on what we learned and experienced with Groovy. Studying a new language was an interesting task. We got to see some things that would make someone choose to pick this up instead of another language as well as the opposite. If someone was starting to learn groovy it would definitely help to have at least a little knowledge or experience with Java. If someone is working on this project, the advice would be to set a meet time once a week or so, as soon as it is assigned to get a little done each week so it doesn't seem like so much to do.

One thing one shouldn’t do, is wait till round finals week to start is a horrible idea not only because this project will take some time, but because you will have all your other classes add projects at the end and you will end up rushing and doing bad on some of them. If I was making the medium level code again I guess we would have done something a little more fun than the merge sort. The language didn't agitate us in the coding aspect just in the fact that to see the errors you had to try to run it each time it didn't just underline something in red for you even if it was obviously wrong...

**References:**

(n.d.). Retrieved December 04, 2017, from http://docs.groovy-lang.org/next/html/documentation/core-syntax.html

(2017, November 19). Retrieved December 02, 2017, from http://docs.groovy-lang.org/latest/html/api/org/codehaus/groovy/GroovyException.html

A multi-faceted language for the Java platform. (n.d.). Retrieved November 22, 2017, from http://groovy-lang.org/

Groovy (programming language). (2017, December 06). Retrieved November 16, 2017, from https://en.wikipedia.org/wiki/Groovy\_(programming\_language)

Groovy vs. Python. (n.d.). Retrieved December 05, 2017, from http://vschart.com/compare/groovy-programming-language/vs/python-programming-language

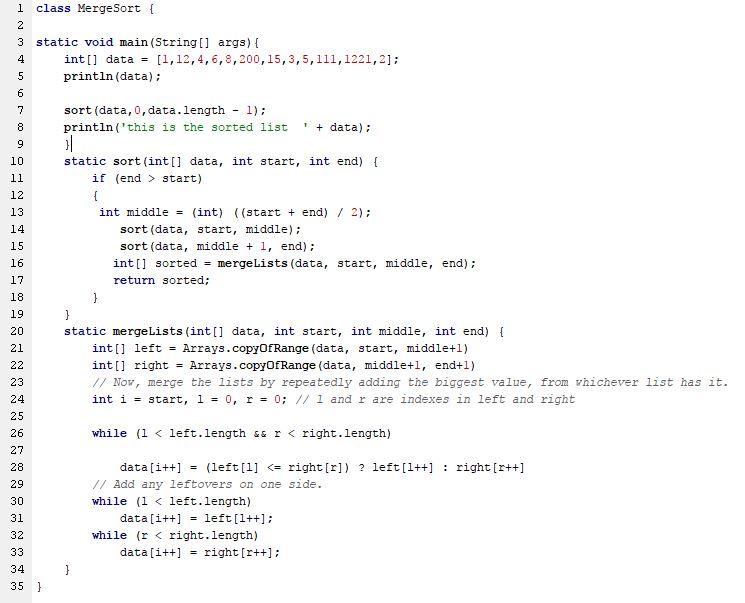
Object-Oriented Groovy. (1970, January 01). Retrieved November 23, 2017, from https://link.springer.com/chapter/10.1007/978-1-4302-1601-8\_4

Point, T. (2017, August 15). Groovy Data Types. Retrieved November 16, 2017, from https://www.tutorialspoint.com/groovy/groovy\_data\_types.htm

Semantics. (n.d.). Retrieved November 18, 2017, from http://groovy-lang.org/semantics.html

Why and When to use Groovy. (n.d.). Retrieved December 02, 2017, from http://blog.xebia.com/why-and-when-to-use-groovy/

Why is Groovy gaining popularity among Java developers? (n.d.). Retrieved December 07, 2017, from https://bbvaopen4u.com/en/actualidad/why-groovy-gaining-popularity-among-java-developers

**Appendix: A Medium Implementations Source Code:**

****

**Acknowledgements**

Created by Brandon Lingo and Hasani Mason at the University of Central Arkansas, Fall 2017.