About the Tutorial

Groovy is an object oriented language which is based on Java platform. Groovy 1.0 was released in January 2, 2007 with Groovy 2.4 as the current major release. Groovy is distributed via the Apache License v 2.0. In this tutorial, we would explain all the fundamentals of Groovy and how to put it into practice.

Audience

This tutorial is going to be extremely useful for all those software professionals who would like to learn the basics of Groovy programming.

Prerequisites

Before proceeding with this tutorial, you should have some hands-on experience of Java or any other object-oriented programming language. No Groovy experience is assumed.

Copyright & Disclaimer

© Copyright 2016 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com
# Table of Contents

About the Tutorial ................................................................................................................................. i

Audience .................................................................................................................................................... i

Prerequisites ............................................................................................................................................... i

Copyright & Disclaimer .......................................................................................................................... i

Table of Contents ................................................................................................................................... ii

1. GROOVY – OVERVIEW ........................................................................................................................ 1

2. GROOVY – ENVIRONMENT ................................................................................................................ 2

3. GROOVY – BASIC SYNTAX ................................................................................................................. 12

   Creating Your First Hello World Program ......................................................................................... 12

   Import Statement in Groovy ............................................................................................................... 12

   Tokens in Groovy ................................................................................................................................. 13

   Comments in Groovy ......................................................................................................................... 13

   Semicolons ........................................................................................................................................... 13

   Identifiers ............................................................................................................................................. 14

   Keywords ............................................................................................................................................. 14

   Whitspaces .......................................................................................................................................... 15

   Literals ................................................................................................................................................ 15

4. GROOVY – DATA TYPES ..................................................................................................................... 16

   Built-in Data Types .............................................................................................................................. 16

   Bound values ....................................................................................................................................... 16

   Class Numeric Types ......................................................................................................................... 17

5. GROOVY – VARIABLES ..................................................................................................................... 19

   Variable Declarations ........................................................................................................................ 19

   Naming Variables .............................................................................................................................. 20
6. **GROOVY – OPERATORS** .................................................................................................................. 22
   
   Arithmetic Operators ....................................................................................................................... 22
   
   Relational operators ...................................................................................................................... 24
   
   Logical Operators .......................................................................................................................... 26
   
   Bitwise Operators .......................................................................................................................... 27
   
   Assignment operators .................................................................................................................... 28
   
   Range Operators ........................................................................................................................... 29
   
   Operator Precedence ....................................................................................................................... 30
   
7. **GROOVY – LOOPS** ....................................................................................................................... 31
   
   while Statement ............................................................................................................................... 31
   
   for Statement .................................................................................................................................. 32
   
   for-in Statement .............................................................................................................................. 34
   
   Loop Control Statements ................................................................................................................ 36
   
   Continue Statement ....................................................................................................................... 37
   
8. **GROOVY – DECISION MAKING** ................................................................................................. 39
   
   if Statement ..................................................................................................................................... 39
   
   if / else Statement ........................................................................................................................... 40
   
   Nested If statements ....................................................................................................................... 42
   
   switch Statements ........................................................................................................................... 43
   
   Nested Switch Statements ............................................................................................................. 45
   
9. **GROOVY – METHODS** ................................................................................................................. 48
   
   Method Parameters ......................................................................................................................... 48
   
   Default Parameters ......................................................................................................................... 49
   
   Method Return Values ..................................................................................................................... 50
   
   Instance methods ............................................................................................................................ 51
   
   Local and External Parameter Names ............................................................................................. 52
10. GROOVY – FILE I/O ................................................................. 54
    Reading files ........................................................................... 54
    Reading the Contents of a File as an Entire String .................... 55
    Writing to Files ....................................................................... 55
    Getting the Size of a File .......................................................... 55
    Testing if a File is a Directory ..................................................... 56
    Creating a Directory .................................................................. 56
    Deleting a File ......................................................................... 57
    Copying files ............................................................................ 57
    Getting Directory Contents ....................................................... 57

11. GROOVY – OPTIONALS ............................................................. 59

12. GROOVY – NUMBERS ............................................................... 61
    Number Methods ..................................................................... 62

13. GROOVY – STRINGS ................................................................. 84
    String Indexing ....................................................................... 84
    Basic String Operations ........................................................... 85
    String Repetition ..................................................................... 86
    String Methods ....................................................................... 87

14. GROOVY – RANGES ................................................................. 105
    contains() ........................................................................... 105
    get() .................................................................................... 106
    getFrom() ............................................................................ 107
    getTo() .................................................................................. 107
    isReverse() ........................................................................... 108
    size() .................................................................................... 109
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>subList()</td>
<td>109</td>
</tr>
<tr>
<td>add()</td>
<td>111</td>
</tr>
<tr>
<td>contains()</td>
<td>112</td>
</tr>
<tr>
<td>get()</td>
<td>113</td>
</tr>
<tr>
<td>isEmpty()</td>
<td>113</td>
</tr>
<tr>
<td>minus()</td>
<td>114</td>
</tr>
<tr>
<td>plus()</td>
<td>115</td>
</tr>
<tr>
<td>pop()</td>
<td>116</td>
</tr>
<tr>
<td>remove()</td>
<td>116</td>
</tr>
<tr>
<td>reverse()</td>
<td>117</td>
</tr>
<tr>
<td>size()</td>
<td>118</td>
</tr>
<tr>
<td>sort()</td>
<td>118</td>
</tr>
<tr>
<td>containsKey()</td>
<td>120</td>
</tr>
<tr>
<td>get()</td>
<td>120</td>
</tr>
<tr>
<td>keySet()</td>
<td>121</td>
</tr>
<tr>
<td>put()</td>
<td>121</td>
</tr>
<tr>
<td>size()</td>
<td>122</td>
</tr>
<tr>
<td>values()</td>
<td>123</td>
</tr>
<tr>
<td>Date()</td>
<td>125</td>
</tr>
<tr>
<td>Date(long millsec)</td>
<td>125</td>
</tr>
<tr>
<td>after()</td>
<td>126</td>
</tr>
<tr>
<td>equals()</td>
<td>127</td>
</tr>
<tr>
<td>compareTo()</td>
<td>128</td>
</tr>
<tr>
<td>toString()</td>
<td>129</td>
</tr>
</tbody>
</table>
1. **GROOVY – OVERVIEW**

Groovy is an object oriented language which is based on Java platform. Groovy 1.0 was released in January 2, 2007 with Groovy 2.4 as the current major release. Groovy is distributed via the Apache License v 2.0.

**Features of Groovy**

Groovy has the following features:

- Support for both static and dynamic typing
- Support for operator overloading
- Native syntax for lists and associative arrays
- Native support for regular expressions
- Native support for various markup languages such as XML and HTML
- Groovy is simple for Java developers since the syntax for Java and Groovy are very similar
- You can use existing Java libraries
- Groovy extends the java.lang.Object

The official website for Groovy is [http://www.groovy-lang.org/](http://www.groovy-lang.org/)
There are a variety of ways to get the Groovy environment setup.

**Binary download and installation** – Go to the link [www.groovy-lang.org/download.html](http://www.groovy-lang.org/download.html) to get the Windows Installer section. Click on this option to start the download of the Groovy installer.

Once you launch the installer, follow the steps given below to complete the installation.

**Step 1:** Select the language installer
Step 2: Click the Next button in the next screen.
**Step 3:** Click the ‘I Agree’ button.
Step 4: Accept the default components and click the Next button.
Step 5: Choose the appropriate destination folder and then click the Next button.
**Step 6:** Click the Install button to start the installation.
Step 7: Once the installation is complete, click the Next button to start the configuration.
Step 8: Choose the default options and click the Next button.
Step 9: Accept the default file associations and click the Next button.
**Step 10:** Click the Finish button to complete the installation.
Once the above steps are followed, you can then start the groovy shell which is part of the Groovy installation that helps in testing our different aspects of the Groovy language without the need of having a full-fledged integrated development environment for Groovy. This can be done by running the command groovysh from the command prompt.

![Groovy Shell](image)

If you want to include the groovy binaries as part of you maven or gradle build, you can add the following lines

**Gradle**

'org.codehaus.groovy:groovy:2.4.5'

**Maven**

<groupId>org.codehaus.groovy</groupId>

<artifactId>groovy</artifactId>

.getVersion>2.4.5</version>
In order to understand the basic syntax of Groovy, let’s first look at a simple Hello World program.

**Creating Your First Hello World Program**

Creating your first hello world program is as simple as just entering the following code line:

```groovy
class Example
{
    static void main(String[] args)
    {
        // Using a simple println statement to print output to the console
        println('Hello World');
    }
}
```

When we run the above program, we will get the following result:

```
Hello World
```

**Import Statement in Groovy**

The import statement can be used to import the functionality of other libraries which can be used in your code. This is done by using the `import` keyword.

The following example shows how to use a simple import of the MarkupBuilder class which is probably one of the most used classes for creating HTML or XML markup.

```groovy
import groovy.xml.MarkupBuilder

def xml=new MarkupBuilder()
```

By default, Groovy includes the following libraries in your code, so you don’t need to explicitly import them.

```groovy
import java.lang.*
import java.util.*
```
import java.io.*
import java.net.*
import groovy.lang.*
import groovy.util.*
import java.math.BigInteger
import java.math.BigDecimal

**Tokens in Groovy**

A token is either a keyword, an identifier, a constant, a string literal, or a symbol.

```groovy
def printn("Hello World");
```

In the above code line, there are two tokens, the first is the keyword `println` and the next is the string literal of "Hello World".

**Comments in Groovy**

Comments are used to document your code. Comments in Groovy can be single line or multiline.

Single line comments are identified by using the // at any position in the line. An example is shown below:

```groovy
class Example
{
    static void main(String[] args)
    {
        // Using a simple println statement to print output to the console
        println('Hello World');
    }
}
```

Multiline comments are identified with /* in the beginning and */ to identify the end of the multiline comment.

```groovy
class Example
{
```
static void main(String[] args) {
    /* This program is the first program
    This program shows how to display hello world */
    println('Hello World');
}

Semicolons

Just like the Java programming language, it is required to have semicolons to distinguish between multiple statements defined in Groovy.

class Example {
    static void main(String[] args) {
        // One can see the use of a semi-colon after each statement
        def x=5;
        println('Hello World');
    }
}

The above example shows semicolons are used to distinguish between different lines of code statements.

Identifiers

Identifiers are used to define variables, functions or other user defined variables. Identifiers start with a letter, a dollar or an underscore. They cannot start with a number. Here are some examples of valid identifiers:

def employeename
def student1
def student_name

where def is a keyword used in Groovy to define an identifier.
Here is a code example of how an identifier can be used in our Hello World program.

```java
class Example {
    static void main(String[] args) {
        // One can see the use of a semi-colon after each statement
        def x=5;
        println('Hello World');
    }
}
```

In the above example, the variable `x` is used as an identifier.

## Keywords

Keywords as the name suggest are special words which are reserved in the Groovy Programming language. The following table lists the keywords which are defined in Groovy.

<table>
<thead>
<tr>
<th>as</th>
<th>assert</th>
<th>break</th>
<th>case</th>
</tr>
</thead>
<tbody>
<tr>
<td>catch</td>
<td>class</td>
<td>const</td>
<td>continue</td>
</tr>
<tr>
<td>def</td>
<td>default</td>
<td>do</td>
<td>else</td>
</tr>
<tr>
<td>enum</td>
<td>extends</td>
<td>false</td>
<td>Finally</td>
</tr>
<tr>
<td>for</td>
<td>goto</td>
<td>if</td>
<td>implements</td>
</tr>
<tr>
<td>import</td>
<td>in</td>
<td>instanceof</td>
<td>interface</td>
</tr>
<tr>
<td>new</td>
<td>pull</td>
<td>package</td>
<td>return</td>
</tr>
<tr>
<td>super</td>
<td>switch</td>
<td>this</td>
<td>throw</td>
</tr>
<tr>
<td>throws</td>
<td>trait</td>
<td>true</td>
<td>try</td>
</tr>
<tr>
<td>while</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Whitespaces

Whitespace is the term used in a programming language such as Java and Groovy to describe blanks, tabs, newline characters and comments. Whitespace separates one part of a statement from another and enables the compiler to identify where one element in a statement.
For example, in the following code example, there is a white space between the keyword **def** and the variable `x`. This is so that the compiler knows that **def** is the keyword which needs to be used and that `x` should be the variable name that needs to be defined.

```groovy
def x=5;
```

### Literals

A literal is a notation for representing a fixed value in groovy. The groovy language has notations for integers, floating-point numbers, characters and strings. Here are some of the examples of literals in the Groovy programming language:

```
12
1.45
'a'
"aa"
```
In any programming language, you need to use various variables to store various types of information. Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory to store the value associated with the variable.

You may like to store information of various data types like string, character, wide character, integer, floating point, Boolean, etc. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.

### Built-in Data Types

Groovy offers a wide variety of built-in data types. Following is a list of data types which are defined in Groovy:

- **byte** – This is used to represent a byte value. An example is 2.
- **short** - This is used to represent a short number. An example is 10.
- **int** – This is used to represent whole numbers. An example is 1234.
- **long** – This is used to represent a long number. An example is 10000090.
- **float** – This is used to represent 32-bit floating point numbers. An example is 12.34.
- **double** - This is used to represent 64-bit floating point numbers which are longer decimal number representations which may be required at times. An example is 12.3456565.
- **char** – This defines a single character literal. An example is ‘a’.
- **Boolean** – This represents a Boolean value which can either be true or false.
- **String** – These are text literals which are represented in the form of chain of characters. For example “Hello World”.

### Bound values

The following table shows the maximum allowed values for the numerical and decimal literals.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>-128 to 127</td>
<td></td>
</tr>
<tr>
<td>short</td>
<td>-32,768 to 32,767</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>-2,147,483,648 to 2,147,483,647</td>
<td></td>
</tr>
</tbody>
</table>
Groovy Programming

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>long</code></td>
<td>-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807</td>
</tr>
<tr>
<td><code>float</code></td>
<td>1.40129846432481707e-45 to 3.40282346638528860e+38</td>
</tr>
<tr>
<td><code>double</code></td>
<td>4.94065645841246544e-324d to 1.79769313486231570e+308d</td>
</tr>
</tbody>
</table>

### Class Numeric Types

In addition to the primitive types, the following object types (sometimes referred to as wrapper types) are allowed:

- `java.lang.Byte`
- `java.lang.Short`
- `java.lang.Integer`
- `java.lang.Long`
- `java.lang.Float`
- `java.lang.Double`

In addition, the following classes can be used for supporting arbitrary precision arithmetic:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>java.math.BigInteger</code></td>
<td>Immutable arbitrary-precision signed integral numbers</td>
<td>30g</td>
</tr>
<tr>
<td><code>java.math.BigDecimal</code></td>
<td>Immutable arbitrary-precision signed decimal numbers</td>
<td>3.5g</td>
</tr>
</tbody>
</table>

The following code example showcases how the different built-in data types can be used:

```java
class Example
{
    static void main(String[] args)
    {
        //Example of a int datatype
        int x=5;
        //Example of a long datatype
        long y=100L;
        //Example of a floating point datatype
        float a=10.56f;
        //Example of a double datatype
        double b=10.5e40;
        //Example of a BigInteger datatype
```
BigInteger bi=30g;
//Example of a BigDecimal datatype
BigDecimal bd=3.5g;
println(x);
println(y);
println(a);
println(b);
println(bi);
println(bd);

When we run the above program, we will get the following result:

5
100
10.56
1.05E41
30
3.5
Variables in Groovy can be defined in two ways – using the native syntax for the data type or the next is by using the def keyword. For variable definitions it is mandatory to either provide a type name explicitly or to use "def" in replacement. This is required by the Groovy parser.

There are following basic types of variable in Groovy as explained in the previous chapter:

- **byte** – This is used to represent a byte value. An example is 2.
- **short** - This is used to represent a short number. An example is 10.
- **int** – This is used to represent whole numbers. An example is 1234.
- **long** – This is used to represent a long number. An example is 10000090.
- **float** – This is used to represent 32-bit floating point numbers. An example is 12.34.
- **double** - This is used to represent 64-bit floating point numbers which are longer decimal number representations which may be required at times. An example is 12.3456565.
- **char** – This defines a single character literal. An example is ‘a’.
- **Boolean** – This represents a Boolean value which can either be true or false.
- **String** – These are text literals which are represented in the form of chain of characters. For example “Hello World”.

Groovy also allows for additional types of variables such as arrays, structures and classes which we will see in the subsequent chapters.

**Variable Declarations**

A variable declaration tells the compiler where and how much to create the storage for the variable.

Following is an example of variable declaration:

```java
class Example
{
    static void main(String[] args)
    {
        // x is defined as a variable
```
```java
String x="Hello";
// The value of the variable is printed to the console
println(x);

}
}
```

When we run the above program, we will get the following result:

Hello

### Naming Variables

The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because Groovy, just like Java is a case-sensitive programming language.

```java
class Example
{
    static void main(String[] args)
    {
        // Defining a variable in lowercase
        int x=5;
        // Defining a variable in uppercase
        int X=6;
        // Defining a variable with the underscore in it's name
        def _Name="Joe";
        println(x);
        println(X);
        println(_Name);
    }
}
```

When we run the above program, we will get the following result:

5
6
We can see that \texttt{x} and \texttt{X} are two different variables because of case sensitivity and in the third case, we can see that \_\texttt{Name} begins with an underscore.

\begin{verbatim}
Printing Variables
You can print the current value of a variable with the println function. The following example shows how this can be achieved.

\begin{verbatim}
class Example
{
    static void main(String[] args)
    {
        //Initializing 2 variables
        int x=5;
        int X=6;
        //Printing the value of the variables to the console
        println("The value of x is "+x+"The value of X is "+X);
    }
}
\end{verbatim}

When we run the above program, we will get the following result:

\begin{verbatim}
The value of x is 5 The value of X is 6
\end{verbatim}
End of ebook preview
If you liked what you saw...
Buy it from our store @ https://store.tutorialspoint.com