***Using lists, maps, and ranges***

Groovy makes collection handling simple, with added support for operators, literals, and extra methods beyond those provided by the Java standard libraries.

**LISTS**

Java supports indexing arrays with a square bracket syntax, which we’ll call the

*subscript operator*. In Groovy the same syntax can be used with *lists—*instances of

java.util.List—which allows adding and removing elements, changing the

size of the list at runtime, and storing items that aren’t necessarily of a uniform

type.

* Groovy allows lists to be indexed outside their current bounds which again can change the size of the list.
* Working with the list looks like you’re working with an array, but in Groovy, the manipulation is more expressive, and the restrictions that apply to arrays are gone:

def roman = ['','i','ii','iii','iv','v','vi','vii','viii','ix','x']

println(roman[3])

println("Size of list: $roman.size()")

roman[15]='xv'

println("Size of list: $roman.size()")

**SIMPLE MAPS**

* A *map* is a storage type that associates a key with a value. Maps store and retrieve values by key; lists retrieve them by numeric index.
* Unlike Java, Groovy supports maps at the language level, allowing them to be specified with literals and providing suitable operators to work with them. It does so with a clear and easy syntax.
* The syntax for maps looks like an array of key-value pairs, where a colon separates keys and values.

def http = [ 100: 'continue',

200: 'OK',

404: 'Resource Not Found',

500: 'Internal Error'

]

println(http[100])

println(http[200])

println('Size: '+http.size())

http[300] = 'Redirect'

println('Size: '+http.size())

println(http[500])

**RANGES**

* Although ranges don’t appear in the standard Java libraries, most programmers have an intuitive idea of what a range is—effectively a start point and an end point, with an operation to move between the two in discrete steps.
* Again, Groovy provides literals to support this useful concept, along with other language features such as the for statement, which understands ranges.

def x = 1..10

assert x.contains(5)

assert !x.contains(15)

assert x.size() == 10

assert x.from == 1

assert x.to == 10

assert x.reverse() == 10..1