Chapter 3: Using the Scala Shell

Exercises :

1. Create a variable of type Int, assign a value, and then see what methods are available for that type. Repeat the same process for the String variable.ls.
2. Explore other options that are available in Scala REPL and research how you can use them (e.g., :sh, :save, :load, etc.).
3. Research the spark-shell and see if the same commands and features are available there as well.
4. Try increasing the memory used by the Scala shell.

Answers :

1.

```
janot@janot-VivoBook-15-ASUS-Laptop-X540UAR:~$ scala
Welcome to Scala 2.11.12 (OpenJDK 64-Bit Server VM, Java 11.0.16).
Type in expressions for evaluation. Or try :help.

scala> val myIntVariable = 12
myIntVariable: Int = 12

scala> myIntVariable.
!=   +   <<   >=   abs        compareTo   getClass   isNaN          isValidChar   isWhole   round      to             toDegrees   toInt          toShort   underlying
%    -   <=   >>   byteValue   doubleValue intValue   isNegInfinity  isValidInt    longValue self       toBinaryString toDouble    toLong         unary_+   until
&    /   ==   >>>  ceil       floatValue  isInfinite  isPosInfinity  isValidLong   max       shortValue toByte         toFloat     toOctalString  unary_-   |
*    <   >    ^    compare    floor       isInfinity  isValidByte    isValidShort  min       signum     toChar         toHexString toRadians      unary_~

scala> myIntVariable.ceil
res0: Float = 12.0
```

```
scala> val myStringVariable ="janot"
myStringVariable: String = janot

scala> myStringVariable.
*              capitalize          contentEquals       flatten          indexOfSlice        lengthCompare       patch               reverse          split          takeWhile       toSet
+              charAt              copyToArray         fold             indexWhere          lift                permutations        reverseIterator  splitAt        to              toShort
++             chars               copyToBuffer        foldLeft         indices             lines               prefixLength        reverseMap       startsWith     toArray         toStream
++:            codePointAt         corresponds         foldRight        init                linesIterator       product             runWith          stringPrefix   toBoolean       toString
+:             codePointBefore     count               forall           inits               linesWithSeparators r                   sameElements     strip          toBuffer        toTraversable
/:             codePointCount      diff                foreach          intern              map                 reduce              scan             stripLeading   toByte          toUpperCase
:+             codePoints          distinct            format           intersect           matches             reduceLeft          scanLeft         stripLineEnd   toCharArray     toVector
:\             collect             drop                formatLocal      isBlank             max                 reduceLeftOption    scanRight        stripMargin    toDouble        transpose
<              collectFirst        dropRight           genericBuilder   isDefinedAt         maxBy               reduceOption        segmentLength    stripPrefix    toFloat         trim
<=             combinations        dropWhile           getBytes         isEmpty             min                 reduceRight         self             stripSuffix    toIndexedSeq    union
>              companion           endsWith            getChars         isTraversableAgain  minBy               reduceRightOption   seq              stripTrailing  toInt           unzip
>=             compare             equals              groupBy          iterator            mkString            regionMatches       size             subSequence    toIterable      unzip3
addString      compareTo           equalsIgnoreCase    grouped          last                nonEmpty            repeat              slice            substring      toIterator      updated
aggregate      compareToIgnoreCase exists              hasDefiniteSize  lastIndexOf         offsetByCodePoints  replace             sliding          sum            toList          view
andThen        compose             filter              hashCode         lastIndexOfSlice    orElse              replaceAll          sortBy           tail           toLong          withFilter
apply          concat              filterNot           head             lastIndexWhere      padTo               replaceAllLiterally sortWith         tails          toLowerCase     zip
applyOrElse    contains            find                headOption       lastOption          par                 replaceFirst        sorted           take           toMap           zipAll
canEqual       containsSlice       flatMap             indexOf          length              partition           repr                span             takeRight      toSeq           zipWithIndex

scala> myStringVariable.capitalize
res1: String = Janot
```

2.

```
scala> :help
All commands can be abbreviated, e.g., :he instead of :help.
:edit <id>|<line>        edit history
:help [command]          print this summary or command-specific help
:history [num]           show the history (optional num is commands to show)
:h? <string>             search the history
:imports [name name ...] show import history, identifying sources of names
:implicits [-v]          show the implicits in scope
:javap <path|class>      disassemble a file or class name
:line <id>|<line>        place line(s) at the end of history
:load <path>             interpret lines in a file
:paste [-raw] [path]     enter paste mode or paste a file
:power                   enable power user mode
:quit                    exit the interpreter
:replay [options]        reset the repl and replay all previous commands
:require <path>          add a jar to the classpath
:reset [options]         reset the repl to its initial state, forgetting all session entries
:save <path>             save replayable session to a file
:sh <command line>       run a shell command (result is implicitly => List[String])
:settings <options>      update compiler options, if possible; see reset
:silent                  disable/enable automatic printing of results
:type [-v] <expr>        display the type of an expression without evaluating it
:kind [-v] <expr>        display the kind of expression's type
:warnings                show the suppressed warnings from the most recent line which had any

scala>
```

3. Spark-shell reuses the latest Scala REPL to reduce code maintenance; the REPL made some changes to how snippets are wrapped to accommodate Spark.