# The Basics

* + **Scala Interpreter / REPL (read-eval-print loop)**

**What is REPL in scala?**

Tool to evaluate expressions in scala. Can be started by typing Scala.

* + **Declaring Values and Variable:**

There are two keywords in scala to declare variables

Val : It is constant we cannot change its value. (Read Only)

Var : This variable can be reassigned. Its content changes as program runs

Note: In scala semicolon is optional it is mandatory when you have multiple statements in same line.

Scala type inference(guess):

Scala can identify type of the variable by itself

* + **Commonly used scala data types**:

**Numeric Datatype**:

Byte

Char

Short

Int

Long

Float

Double

Boolean

String

All these data types are Classes in scala. There is no distinction between primitive datatypes and Classes in Scala. As all types are classes we can invoke methods on them.

* + **Arithmetic and operator overloading**

Scala has all arithmetic operators like java and other programming languages.

One surprising thing here is all operators are methods in Scala !!

Scala has no increment operators

* + **More about calling method**

If the method has only one parameter then we do not need to use parenthesis.

This is the method with one paramenter

"Hello".intersect("World")

sorted method takes no parameter it just sorts characters in string so that method can be called as

"Bonjour".sorted // Yields the string "Bjnooru"

// In Scala, the \_ character is a “wildcard,” like \* in Java

## Apply Method:

This method is like toString of Java. Every Scala object can implement apply method as per their need. For example

val s = "Hello"

s(4) // Yields 'o'

In this case s(4) is like

s.apply(4)

This is the method signature in StringOps class def apply(n: Int): Char

Note: We know that scala runs on JVM that is true for standard Scala distribution. However there is scala.js project [**https://www.scala-js.org/**](https://www.scala-js.org/)this project provides tools to translate scala to javascript . If we take advantage of this tool, we can write both server side and client side code using scala

* + **Scala Doc**

<https://www.scala-lang.org/api/current/>

Excersise:

https://gist.github.com/parambirs/9932268

1. In the Scala REPL, type 3. followed by the Tab key. What methods can be applied? This is something related to apply

There are many methods including %, &, \*, +, toByte, toChar etc

1. In the Scala REPL, compute the square root of 3, and then square that value. By how much does the result differ from 3? (Hint: The res variables are your friend.)

scala> math.sqrt(3)

res27: Double = 1.7320508075688772

scala> math.pow(res27,2)

res28: Double = 2.9999999999999996

scala> 3 - res28;

res29: Double = 4.440892098500626E-16

1. Are the res variables val or var?

Val

1. Scala lets you multiply a string with a number—try out "crazy" \* 3 in the REPL. What does this operation do? Where can you find it in Scaladoc?

scala> "crazy" \* 3;

res32: String = crazycrazycrazy

scala> "crazy".\*(3);

res33: String = crazycrazycrazy

The above two styles of coding is same here we are calling \* method on string

We can find in scala.collection.immutable.StringOps class in java doc

def \*(n: Int): String

Return the current string concatenated n times.

1. What does 10 max 2 mean? In which class is the max method defined?

scala> 10 max 100

res35: Int = 100

scala> 10.max(100)

res36: Int = 100

def max(that: Int): Int

Returns this if this > that or that otherwise.

Max method defined in RichInt method

1. Using BigInt, compute 21024?

scala> BigInt(2).pow(1024)

res39: scala.math.BigInt = 17976931348623159077293051907890247336179769789423065

72734300811577326758055009631327084773224075360211201138798713933576587897688144

16622492847430639474124377767893424865485276302219601246094119453082952085005768

83815068234246288147391311054082723716335051068458629823994724593847971630483535

6329624224137216

scala> BigInt(2) pow 1024

res43: scala.math.BigInt = 17976931348623159077293051907890247336179769789423065

72734300811577326758055009631327084773224075360211201138798713933576587897688144

16622492847430639474124377767893424865485276302219601246094119453082952085005768

83815068234246288147391311054082723716335051068458629823994724593847971630483535

6329624224137216

1. What do you need to import so that you can get a random prime as probablePrime(100, Random), without any qualifiers before probablePrime and Random?

// import BigInt.probablePrime

// import util.Random

probablePrime(100, Random) //> res3: scala.math.BigInt = 912013777957722342425952556723

1. One way to create random file or directory names is to produce a random BigInt and convert it to base 36, yielding a string such as "qsnvbevtomcj38o06kul". Poke around Scaladoc to find a way of doing this in Scala.

//TODO Need to understand more

1. How do you get the first character of a string in Scala? The last character?

scala> val sampleString = "Hello"

sampleString: String = Hello

scala> sampleString.head

res7: Char = H

scala> sampleString(0) // This implicitly call apply method

res8: Char = H

scala> sampleString.last

res9: Char = o

scala> sampleString(sampleString.length - 1)

res10: Char = o

def apply(index: Int): Char

Returns element at index n

1. What do the take, drop, takeRight, and dropRight string functions do? What advantage or disadvantage do they have over using substring?

scala> val name = "chandra"

name: String = Chandra

// take: Selects the first n elements

scala> name take 2

res11: String = ch

scala> name.take(2)

res12: String = ch

// drop: Selects all elements except first n ones

scala> name drop 2

res13: String = andra

scala> name.drop(2)

res14: String = andra

// takeRight: Selects the last n elements

scala> name takeRight 2

res15: String = ra

scala> name.takeRight(2)

res16: String = ra

// dropRight: Selects all elements except last n ones

scala> name dropRight 2

res17: String = chand

scala> name.dropRight(2)

res18: String = chand

## Declaring values and Variables

### Val

val makes a variable immutable — like final in Java

### Var

var makes a variable mutable.