Scala Basics cont'

Types, Loops, Strings, Reading Files

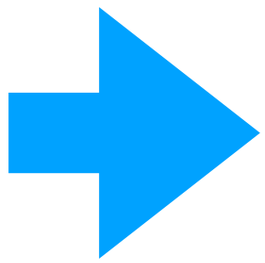
Scala Types

- All values in Scala are objects
  - Contain methods
  - No primitive values in Scala
- We'll start with
  - Int
  - Double
  - Boolean
  - Unit
  - String
- Many more types to come

Int

- A whole number
- 32 bit representation
- -2147483648 to 2147483647
- Values outside this range will overflow
  - Wrap around

```
val a: Int = 2147483647
println(a + 1)
```

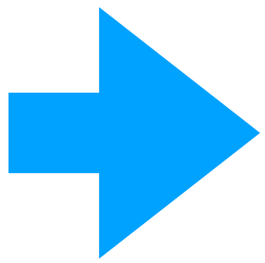


-2147483648

Double

- Number with a decimal
- 64 bit representation
- Truncated to fit in 64 bits
    - Rounding errors
- Check for equality with
    - Math.abs(x - y) < small_value

```
val b: Double = 0.1
val c: Double = b * 3
println(c)
```



0.30000000000000004

Boolean and Unit

- Boolean
  - true or false
- Unit
  - Nothing
  - Used to indicates a function that doesn't return a value
  - Ex: main and println return Unit

String

- A sequence of characters (type Char)
- Declared with double quotes
  - val s:String = "valid string literal"
- Many useful methods. Examples:
  - startsWith(String) - check if this String starts with the given String
  - length() - number of characters in this String
  - .split(String) - Separates this String by the given String

Scala Type Conversions

```scala
package example

object Types {

  def main(args: Array[String]): Unit = {

    // Declaring variable
    var anInt: Int = 10
    var aDouble: Double = 5.8
    var aBoolean: Boolean = true
    var aString: String = "6.3"

    // Converting variable types
    var anotherDouble: Double = aString.toDouble
    var anotherString: String = anInt.toString

    // Truncates the decimal. anotherInt == 5
    var anotherInt: Int = aDouble.toInt
  }

}
```

Use the to<Type> methods to convert between type

For Loop

For Loop

```
for(<variable_name> <- <data_structure>){
  <loop_body>
}
```

Reads:

"for variable_name in data_structure execute loop_body"

For Loop

```scala
package example

object Loop {

  def printOneTo(n: Int): Unit = {
    for(i <- 1 to n){
      println("i == " + i)
    }
  }

  def printOneToAlternate(n: Int): Unit = {
    val numbers: Range = 1 to n
    for (i <- numbers) {
      println("i == " + i)
    }
  }

  def main(args: Array[String]): Unit = {
    printOneTo(10)
  }

}
```

"1 to n" creates a Range of integers that can be iterated over with a for loop
-Similar to range(n) in Python

For Loop + String Example

```
package example

object StringSplitter {

  def computePercentTrue(line: String): Double = {
    val splits: Array[String] = line.split(";")
    var totalCount: Double = 0
    var trueCount: Double = 0
    for (value <- splits) {
      val valueAsBoolean: Boolean = value.toBoolean
      if (valueAsBoolean) {
        trueCount += 1
      }
      totalCount += 1
    }
    trueCount / totalCount
  }

  def main(args: Array[String]): Unit = {
    val testInput = "true;false;true;true;true"
    val percentTrue = computePercentTrue(testInput) // expecting 0.8
    println("Percentage true == " + percentTrue)
  }

}
```

Given a String containing boolean values separated by semicolons, return the percentage of values that are true

For Loop + String Example

```
package example

object StringSplitter {

  def computePercentTrue(line: String): Double = {
    val splits: Array[String] = line.split(";")
    var totalCount: Double = 0
    var trueCount: Double = 0
    for (value <- splits) {
      val valueAsBoolean: Boolean = value.toBoolean
      if (valueAsBoolean) {
        trueCount += 1
      }
      totalCount += 1
    }
    trueCount / totalCount
  }

  def main(args: Array[String]): Unit = {
    val testInput = "true;false;true;true;true"
    val percentTrue = computePercentTrue(testInput) // expecting 0.8
    println("Percentage true == " + percentTrue)
  }

}
```

Split the String on semicolons
-Returns a data structure of Strings

For Loop + String Example

```
package example

object StringSplitter {

  def computePercentTrue(line: String): Double = {
    val splits: Array[String] = line.split(";")
    var totalCount: Double = 0
    var trueCount: Double = 0
    for (value <- splits) {
      val valueAsBoolean: Boolean = value.toBoolean
      if (valueAsBoolean) {
        trueCount += 1
      }
      totalCount += 1
    }
    trueCount / totalCount
  }

  def main(args: Array[String]): Unit = {
    val testInput = "true;false;true;true;true"
    val percentTrue = computePercentTrue(testInput) // expecting 0.8
    println("Percentage true == " + percentTrue)
  }

}
```

Iterate over each value

For Loop + String Example

```
package example

object StringSplitter {

  def computePercentTrue(line: String): Double = {
    val splits: Array[String] = line.split(";")
    var totalCount: Double = 0
    var trueCount: Double = 0
    for (value <- splits) {
      val valueAsBoolean: Boolean = value.toBoolean
      if (valueAsBoolean) {
        trueCount += 1
      }
      totalCount += 1
    }
    trueCount / totalCount
  }

  def main(args: Array[String]): Unit = {
    val testInput = "true;false;true;true;true"
    val percentTrue = computePercentTrue(testInput) // expecting 0.8
    println("Percentage true == " + percentTrue)
  }

}
```

Convert to Boolean

For Loop + String Example

```
package example

object StringSplitter {

  def computePercentTrue(line: String): Double = {
    val splits: Array[String] = line.split(";")
    var totalCount: Double = 0
    var trueCount: Double = 0
    for (value <- splits) {
      val valueAsBoolean: Boolean = value.toBoolean
      if (valueAsBoolean) {
        trueCount += 1
      }
      totalCount += 1
    }
    trueCount / totalCount
  }

  def main(args: Array[String]): Unit = {
    val testInput = "true;false;true;true;true"
    val percentTrue = computePercentTrue(testInput) // expecting 0.8
    println("Percentage true == " + percentTrue)
  }

}
```

Count the total number of values and the number that are true

For Loop + String Example

```
package example

object StringSplitter {

  def computePercentTrue(line: String): Double = {
    val splits: Array[String] = line.split(";")
    var totalCount: Double = 0
    var trueCount: Double = 0
    for (value <- splits) {
      val valueAsBoolean: Boolean = value.toBoolean
      if (valueAsBoolean) {
        trueCount += 1
      }
      totalCount += 1
    }
    trueCount / totalCount
  }

  def main(args: Array[String]): Unit = {
    val testInput = "true;false;true;true;true"
    val percentTrue = computePercentTrue(testInput) // expecting 0.8
    println("Percentage true == " + percentTrue)
  }

}
```

Compute the average

-Note: If these values were Ints this would be integer division

Reading Files

Reading Files

```
package example

import scala.io.Source

object FileReader {

  def convertFileToString(filename: String): String = {
    var contents: String = ""
    val file: BufferedSource = Source.fromFile(filename)
    for (line <- file.getLines()){
      contents += line + "\n"
    }
    contents
  }

  def main(args: Array[String]): Unit = {
    val filename = "data/testFile.txt"
    val contents = convertFileToString(filename)
    println(contents)
  }

}
```

Read the contents into a String line-by-line
-Assumes "data/testFile.txt" exists in the project

Reading Files

```
package example

import scala.io.Source

object FileReader {

  def convertFileToString(filename: String): String = {
    var contents: String = ""
    val file: BufferedSource = Source.fromFile(filename)
    for (line <- file.getLines()){
      contents += line + "\n"
    }
    contents
  }

  def main(args: Array[String]): Unit = {
    val filename = "data/testFile.txt"
    val contents = convertFileToString(filename)
    println(contents)
  }

}
```

Use scala.io.Source.fromFile(filename: String): BufferedSource

Reading Files

```
package example

import scala.io.Source

object FileReader {

  def convertFileToString(filename: String): String = {
    var contents: String = ""
    val file: BufferedSource = Source.fromFile(filename)
    for (line <- file.getLines()){
      contents += line + "\n"
    }
    contents
  }

  def main(args: Array[String]): Unit = {
    val filename = "data/testFile.txt"
    val contents = convertFileToString(filename)
    println(contents)
  }

}
```

Call BufferedSource.getLines() to get the lines in a data structure of Strings

Reading Files

Example in IntelliJ