Day 2: Basics of Scala

A] Comments

- Single line comment example:

```
scala> var a = 5
a: Int = 5

scala> var a = 5         Assigning the value 5 to variable a
<console>:7: error: value Assigning is not a member of Int
       var a = 5         Assigning the value 5 to variable a
                         ^

scala> var a = 5         // Assigning the value 5 to variable a
a: Int = 5
```

B] Keywords in Scala (Ref: https://www.geeksforgeeks.org/scala-keywords/)

abstract, case, catch, class, def, do, else, extends, false, final, finally, for, forSome, if, implicit, import, lazy, match, new, null, object, override, package, private, protected, return, sealed, super, this, throw, trait, true, try, type, val, var, while, with, yield, >:, ⇒, =>, =, <%, <:, ←, <-, #, @, :, _

```
scala> var a = 5
a: Int = 5

scala> var if = 5
<console>:1: error: illegal start of simple pattern
       var if = 5
           ^

scala> var def = 5
<console>:1: error: illegal start of simple pattern
       var def = 5
           ^

scala> var false = 5
<console>:7: error: type mismatch;
 found   : Boolean(false)
 required: Int
       var false = 5
```

C] Escape characters - println statement

```
scala> println ("Hello World!")
Hello World!

scala> println ("Hello \n World!") // New Line
Hello
 World!
```

```
scala> println ("Hello\tWorld!")  // Tab
Hello   World!

scala> println ("Hello\bWorld!")   // Back Space
HellWorld!

scala> println ("Hello\b\bWorld!")
HelWorld!

scala> println ("Hello\b\b\bWorld!")
HeWorld!

scala> println ("Hello\b\b\b\bWorld!")
HWorld!

scala> println ("Hello\b\b\b\b\bWorld!")
World!

scala> println ("Hello\fWorld!") // Form Feed
Hello
```

```
World!

scala> println ("Hello\rWorld!")  // carriage return
World!

scala> println ("Hello \r World!")
 World!

scala> println ("Hello \"India\" World!")  // using "
Hello "India" World!

scala> println ("Hello \'India\' World!")  // using '
Hello 'India' World!

scala> println ("https:\\\\www.google.com") // using \
https:\\www.google.com
```

D] Variables

```
scala> //Variables - This is commented line

scala> var isPresent = true   // Boolean variable
isPresent: Boolean = true

scala> var isPresent:Boolean = true
isPresent: Boolean = true

scala> var isPresent:Int = true  // Error: Trying to assign Boolean to an Int
<console>:7: error: type mismatch;
 found   : Boolean(true)
 required: Int
       var isPresent:Int = true
                           ^
scala> var isPresent:Boolean = 122  // Error: Trying to assign Int to Boolean
<console>:7: error: type mismatch;
 found   : Int(122)
 required: Boolean
       var isPresent:Boolean = 122
                               ^
scala> var a = 123  // Integer Variable
a: Int = 123

scala> var a:Int = 123  // Integer variable
a: Int = 123

scala> var a:Float = 123  // Float Variable
a: Float = 123.0

scala> var a = 123f  // Type Inference to Float
a: Float = 123.0

scala> var a = 12.3   // Default Double for decimal value & Int for whole no.
a: Double = 12.3
```

```
scala> var a = 12.3f
a: Float = 12.3

scala> var a = 123
a: Int = 123

scala> var a:Byte = 123
a: Byte = 123

scala> var a = 123.toByte()  // Error: toByte method does not accept argument
<console>:7: error: Byte does not take parameters
       var a = 123.toByte()
                         ^

scala> var a = 123.toByte  // Convert to byte
a: Byte = 123

scala> var a = 129.toByte  // Byte from -128 to 127
a: Byte = -127

scala> var a:Byte = 5
a: Byte = 5

scala> var a:Byte = 7
a: Byte = 7

scala> var a:Byte = 5
a: Byte = 5

scala> var b:Byte = 7
b: Byte = 7

scala> var c = a + b   // Note: + Method for Byte input returns Int as per
scala opensource code/documentation in scala-lang.org
c: Int = 12

scala> var c:Byte = a + b  // Error: Can not assign Int to a Byte
<console>:9: error: type mismatch;
 found   : Int
 required: Byte
       var c:Byte = a + b
                      ^

scala> var c = (a + b).toByte
c: Byte = 12

scala> var a = 12.3  // Default data type for decimal value is double
a: Double = 12.3

scala> var a:Float = 12.3  // Error:Trying to assign double to Float variable
<console>:7: error: type mismatch;
 found   : Double(12.3)
 required: Float
       var a:Float = 12.3
                     ^

scala> var a:Float = 12.3f // Specify f at end of no. to specify float
a: Float = 12.3

scala> var b:Float = 12.5f
b: Float = 12.5

scala> var c = a + b //Note: + Method for Float input returns Float as per
scala opensource code/documentation in scala-lang.org
c: Float = 24.8

scala> var a = 5  // Integer variable
a: Int = 5

scala> var c = a/0  // Divide by 0
java.lang.ArithmeticException: / by zero
        at .<init>(<console>:8)
        at .<clinit>(<console>)
        at .<init>(<console>:7)
        at .<clinit>(<console>)
        at $print(<console>)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
        at java.lang.reflect.Method.invoke(Unknown Source)
        at scala.tools.nsc.interpreter.IMain$ReadEvalPrint.call(IMain.scala:734)
        at scala.tools.nsc.interpreter.IMain$Request.loadAndRun(IMain.scala:983)
        at scala.tools.nsc.interpreter.IMain.loadAndRunReq$1(IMain.scala:573)
        at scala.tools.nsc.interpreter.IMain.interpret(IMain.scala:604)
        at scala.tools.nsc.interpreter.IMain.interpret(IMain.scala:568)
        at scala.tools.nsc.interpreter.ILoop.reallyInterpret$1(ILoop.scala:760)
        at scala.tools.nsc.interpreter.ILoop.interpretStartingWith(ILoop.scala:805)
        at scala.tools.nsc.interpreter.ILoop.command(ILoop.scala:717)
        at scala.tools.nsc.interpreter.ILoop.processLine$1(ILoop.scala:581)
        at scala.tools.nsc.interpreter.ILoop.innerLoop$1(ILoop.scala:588)
        at scala.tools.nsc.interpreter.ILoop.loop(ILoop.scala:591)
        at scala.tools.nsc.interpreter.ILoop$$anonfun$process$1.apply$mcZ$sp(ILoop.scala:882)
        at scala.tools.nsc.interpreter.ILoop$$anonfun$process$1.apply(ILoop.scala:837)
        at scala.tools.nsc.interpreter.ILoop$$anonfun$process$1.apply(ILoop.scala:837)
        at scala.tools.nsc.util.ScalaClassLoader$.savingContextLoader(ScalaClassLoader.scala:135)
        at scala.tools.nsc.interpreter.ILoop.process(ILoop.scala:837)
        at scala.tools.nsc.MainGenericRunner.runTarget$1(MainGenericRunner.scala:83)
        at scala.tools.nsc.MainGenericRunner.process(MainGenericRunner.scala:96)
        at scala.tools.nsc.MainGenericRunner$.main(MainGenericRunner.scala:105)
        at scala.tools.nsc.MainGenericRunner.main(MainGenericRunner.scala)



scala> var c = a*0  // Multiply by 0
c: Int = 0

scala> var c = a.*(0)  // * operator refers to * method(function)
c: Int = 0

scala> var c = a * 0
c: Int = 0
```

```
scala> var a = 5.toByte
a: Byte = 5

scala> var a = 5.asInstanceOf[Byte]  // Same as toByte, but asInstanceOf is
part of Any class and hence it can be used on any Object of Any class
a: Byte = 5

scala>

scala> var a = 5.toInt
a: Int = 5

scala> var a = 5.asInstanceOf[Int]
a: Int = 5

scala>

scala> var a:Any = 5
a: Any = 5

scala> var c = a.toInt
<console>:8: error: value toInt is not a member of Any
       var c = a.toInt

scala> var c = a.asInstanceOf[Int] // Same as toInt, but asInstanceOf is part
of Any class and hence it can be used on any Object of Any class
c: Int = 5
```

E} Operators in Scala [https://www.geeksforgeeks.org/operators-in-scala/]

- **Arithmetic Operators**
- **Relational Operators**
- **Logical Operators**
- **Assignment Operators**
- **Bitwise Operators**

Arithmatic Operators

```
    scala> var a = 50
    a: Int = 50

    scala> var b = 30
    b: Int = 3
0

    // Addition
    scala> println("Addition of a + b = " + (a + b))
    Addition of a + b = 80

    // Subtraction
    scala> println("Subtraction of a - b = " + (a - b))
    Subtraction of a - b = 20
    // Multiplication
    scala> println("Multiplication of a * b = " + (a * b))
    Multiplication of a * b = 1500


    // Division
    scala> println("Division of a / b = " + (a / b))
    Division of a / b = 1

    // Modulus
    scala> println("Modulus of a % b = " + (a % b))
    Modulus of a % b = 20
```

Relational Operators

```
    // variables
scala> var a = 50;
a: Int = 50

scala> var b = 50;
b: Int = 50

scala> a == b;
res1: Boolean = true

scala> println("Equality of a == b is : " + ( a == b).toString)
Equality of a == b is : true

scala> println("Equality of a == b is : " + ( a == b))
Equality of a == b is : true

scala> println("Equality of a != b is : " + ( a != b))
Equality of a != b is : false

scala> println("Equality of a > b is : " + ( a > b))
Equality of a > b is : false

scala> println("Equality of a < b is : " + ( a < b))
Equality of a < b is : false

scala> println("Equality of a >= b is : " + ( a >= b))
Equality of a >= b is : true

scala> println("Equality of a <= b is : " + ( a <= b))
Equality of a <= b is : true
Logical Operator:



scala> var a = false
a: Boolean = false

scala> var b = true
```

```
b: Boolean = true

scala> println ("Logical Not of (a && b) = " + (a&&b))
Logical Not of (a && b) = false

scala> println ("Logical And of (a && b) = " + (a&&b))
Logical And of (a && b) = false

scala> println ("Logical Not of And of !(a && b) = " + !(a&&b))
Logical Not of And of !(a && b) = true

scala> println ("Logical Or of (a || b) = " + (a||b))
Logical Or of (a || b) = true
```

Assignment Operator

```
scala> var a = 50;
a: Int = 50

scala> var b = 40;
b: Int = 40

scala> var c = 0
c: Int = 0

scala>

scala> //Simple Addition

scala> c = a + b;
c: Int = 90

scala> println("simple addition: c= a + b = " + c);
simple addition: c= a + b = 90

scala> c += a  // c = c + a

scala> c
res15: Int = 140

scala> c -= a // c = c - a

scala> c
res17: Int = 90

scala> c *= a // It means c = c * a

scala> c
res19: Int = 4500

scala> c /=a   // It means c = c/a = 4500/50

scala> c
```

```
res21: Int = 90

scala> c %=a // it means c = c%a = 90%50 = 40

scala> c
res23: Int = 40

scala> c <<= 3

scala> c
res25: Int = 320

scala> c >>=3

scala> c
res27: Int = 40

scala> c &= a

scala> c
res29: Int = 32

scala> c ^= a

scala> c
res31: Int = 18


scala> c |= a

scala> c
res34: Int = 50
```

Bitwise Operator

```
scala> var a = 20;
a: Int = 20

scala> var b = 18;
b: Int = 18

scala> var c = 0;
c: Int = 0

scala>

scala> c = a & b;
c: Int = 16

scala> c = a | b;
c: Int = 22

scala> c = a ^ b;
c: Int = 6

scala> c = ~a
c: Int = -21
```

```
scala> c = a << 3
c: Int = 160

scala> c = a >> 3
c: Int = 2

scala> c = a >>> 3
c: Int = 2    c = a >> 3;
```