Recap

Saturday, July 2, 2022 8:12 PM

1. Yarn Components
2. Hadoop Ecosystem
3. Hadoop Configuration File
4. Hadoop HDFS Commands
5. Big Data Analytics
6. Introduction to Scala
7. Scala Datatypes

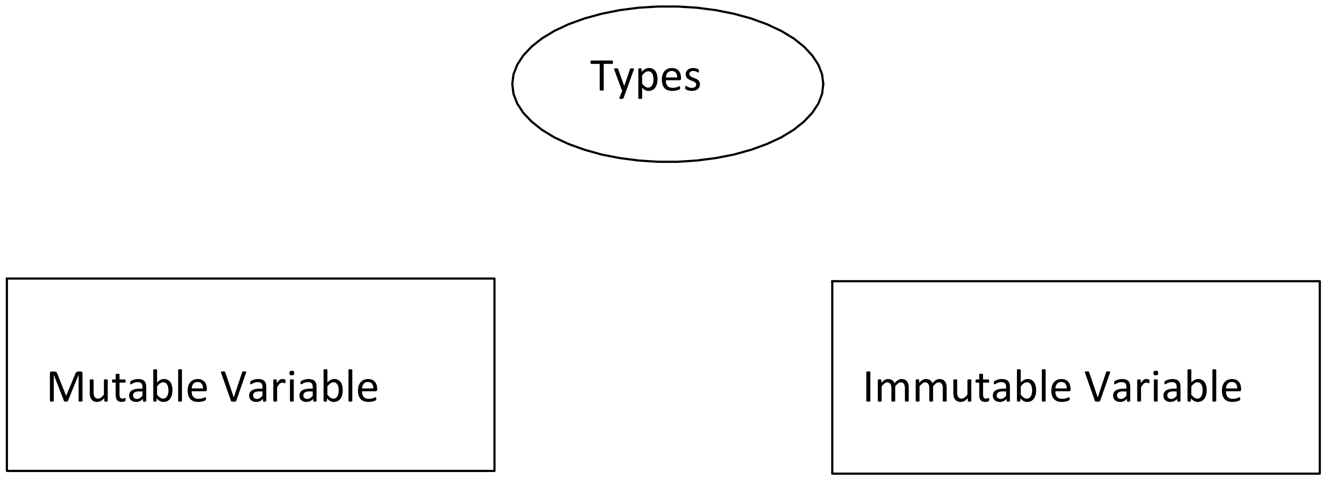
Topics for Today

Friday, May 27, 2022 7:50 PM

1. Scala Variables
2. Control Structures
3. Scala Functions
4. Array
5. Array Buffer
6. List
7. List Buffer
8. Tuple
9. Sets
10. Maps
11. Classes and Objects
12. Getters and Setters
13. Constructor

Scala Variables

28 June 2020 13:12



var - Mutable Variables
val - Immutable Variables

Type Inference -

Block Expression

```
val x = 12
val x  = { val a = 10; val b = 100; b -a}
```

Lazy Values

```
val file =
scala.io.Source.fromFile("/mnt/home/edureka_
967855/csvsample/csvfile1.txt")

lazy val m = { println("foo"); 1}
println("bar")
println(m)
```

Control Structures

28 June 2020 16:06

If else if else

Why Do we need loops ?
To iterate over data

For Loop

```
        for ( I <- 1 to 100) { //statements}

    for ( i <- 1 to 5 )
    {println(i)
    //multiple lines
    }
    val strvar = "Hello World"

    for( i <- 0 until strvar.length ) println(strvar(i))

    Range  = 0 to 11

    0 to 10
    0 to 11
```

```
//Multiloop Generator
for ( i <- 1 to 3 ; j <- 1 to 3){
println(10 * i + j)
}
//Condition
for ( i <- 1 to 3 ; j <- 1 to 3 ; if (i ==j)){
println(10 * i + j)
}
//Introduce a variables in loop
for ( i <- 1 to 3 ;x = 4 -i; j <- x to 3  ){
println(10 * i + j)
}
```

yield -> Return a collection of values

```
val x = for (i <- 1 to 20) yield i * 2.5
```

While Loop

```
while ( boolean expression) {//Statements}

var a = 10
while( a < 20) {
println("Value of a: " + a)
a = a + 1
}
```

Do While Loop

```
do { //Stamements}  while (boolean Expression)

var a = 20
do {
println("Value of a: " + a)
a = a + 1
}
while (a < 20)
```

Scala Functions

28 June 2020 17:38

Block of code that get executed when we call the function

Def - Key word to create functions

name - Name of function
Input Parameters - Datatype of Input Variables
OutPut - Datatype of out put variables
Statements - Processing of input to get output

```
def max( x: Int , y:Int)  : Int = {
if (x > y)
x
else
y
}
```

Function Without parameters without equal to

Function With parameters without equal to

Function WithParameter With equal To

Function With equal to

def - Keyword to create scala function
maxnumber = name of function
 : Datatype = Input Parameters

```
def maxnumber  ( x : Int, y : Int) : Int = {
 if (x > y)
 x
 else
 y}
```

```
//Without Parameters
def sayHello ()  { println("Hello") }
```

// With Parameter that does not return anything

```
def sum (a : Int , b :Int) { println(a+b)}

// Return but no input paratmeters

def func() : Int = { 7}

// Return  and input Parameters
def sum(a:Int, b:Int) : String = { (a + b).toString}
def sumWR (a : Int , b :Int) { println(a+b)}
```

Recursion Functions
Recursion means a function that calls itself repeatedly

Arguments to Functions - Initialization

```
def func1 (a:Int = 0 ,b :Int = 0) : Int  = a+b
```

Arguments to Functions - Changing orders of

Parameters

Nested Function
A function defined inside a function

```
def factorial (i:Int) : Int = {

def fact (i:Int,accumulator:Int) : Int = {
if (i <= 1)
accumulator
else
fact(i-1,i * accumulator)
}
fact(i,1)
}
```

Array

02 July 2020 22:14

What is collections?
Group of Values

Collection of elements of same type.

Array is mutable object

Values in Array can be changed

Array With Values

```
val arr1 = Array('a','b','c')
arr1: Array[Char] = Array(a, b, c)
```

```
arr1(0) = 'z'
arr1(2) = "e"
```

In Arrays i am not able to increase the size of array

My requirement is to create a growing Array

```
scala> val arr1 = Array ('a',10,11.2,"Ram")
arr1: Array[Any] = Array(a, 10, 11.2, Ram)
```

```
val b = Array("jan","feb","mar","apr")
for ( i <- b) println(i)
for ( i <- 0 until b.length) println(b(i))

b.foreach(println)
```

Array Without Value

Can we change the datatype?

Can we change array Elements ?

Can we Change array Size ?

Iterate over Array

ArrayBuffer

02 July 2020 22:47

Variable Length Array

Import Statement

import scala.collection.mutable.ArrayBuffer
declare Array Buffer

Can we change data type?

```
import scala.collection.mutable.ArrayBuffer

val a = ArrayBuffer[Int]()
```

```
scala> val arr1 = Array ('a',10,11.2)
arr1: Array[Double] = Array(97.0, 10.0, 11.2)
scala> val arr1 = Array ('a',10,11.2,"Ram")
arr1: Array[Any] = Array(a, 10, 11.2, Ram)
scala> arr1(1) + 1
<console>:9: error: type mismatch;
 found   : Int(1)
 required: String
              arr1(1)+1
```

List

02 July 2020 22:14

Declaration

Lists are immutable, which means elements of list cannot be changed by assignment

Different Ways of Creating List

```
val lst = List(1,2,3,4,5)
scala> lst(4) = 4
<console>:10: error: value update is not a member
of List[Int]
            lst(4) = 4

lst.foreach(println)

val lst2 = 1 :: 2 :: 3 :: Nil

val lst3 = List(1,2,3)

val lst4 = List.fill(3)("foo")
lst4: List[String] = List(foo, foo, foo)

val x = List.tabulate(5)( n => n *n)
```

Adding Elements to List(Only Prepend)

Var - Reassigning Variable
list8.foreach(println)

Deleting Elements in List

Merging Lists

Iterator To iterate over the list

```
def sum(l:List[Int]) :  Int = {
if (l ==Nil) 0
else l.head + sum(l.tail)
}
```

List Buffer

04 July 2020 14:01

Declaration

import scala.collection.mutable.ListBuffer

Tuple

04 July 2020 14:06

Why Tuples?
Used to stores different data types

Declaration

```
val lst = List(1,true,"str")

lst.foreach(println)

lst(0)  -  First element in list

a:Any
```

Tuples

```
val a = (1,4,"Bob",true, 'a')

a: (Int, Int, String, Boolean, Char) = (1,4,Bob,true,a)

a.productIterator.foreach(println)
```

Access Elements of Tuples

```
a._1   - First Element of Tuple
```

Index starts with position 1

offsets
Offset starts with 1 and not from 0

Iteration
productIterator

```
a.productIterator.foreach(println)
```

Swap Elements

Sets

04 July 2020 14:08

Why Sets?

Only Unique values are stored in sets

Declaration

```
val s  = Set(1,2,3,4,4,5)
val t = Set(4,5,6,7,8)
```

Intersection

```
val u = s.intersect(t)
```

Union

Classes and Objects

04 July 2020 16:08

```
classlearnClass{
private var value=0//fieldsmustbeinitialized
def incr(){value+=1}
def curr()=value
}
```

Keyword

Class counter

```
classlearnClass{

privatevarvar1=10
//defincr(){value+=1}//Null
//defcurr()=value//Returningthevalue
defcustGetter()=var1//customisedGetter
defcustSetter(x:Int){var1=x}//customisedsetter

}

valobj1=newlearnClass

obj1.custGetter()//CallingtheCustomisedgetter
obj1.custSetter(13)//CallingtheCustomisedsetter
obj1.custGetter()//CallingtheCustomisedgetter

valobj2=newlearnClass
```

Example

Object is instance of the class

```
val ctr1 = new cntr

ctr1.incr           cntr1 = 1
ctr1.incr           cntr1 = 2
ctr1.incr           cntr1 = 3
ctr1.incr           cntr1 = 4
ctr1.curr           4

class birds {
var color = "green"
def changecolor (newColor : String)  { color =
newColor }
def findColor = { color}
}

val brd1 = new birds
brd1.findColor
brd1.changecolor("pink")
brd1.findColor
```

Getters and Setter

05 July 2020 11:14

Why Getter and Setters?

Used to expose class properties/variables

Getter Example

```
class learnGetter {
  val size = 1
}

val f = new learnGetter
val a =  f.size
println("Printing after geting value: " + f.size)
```

Getter and Setter Example

```
class learnGetterSetter {
  var size = 1
}

val f = new learnGetterSetter
val a =  f.size
println("Printing before setting value: " + f.size)
f.size
```

```
f.size_=(10)
println("Printing after setting value: " + f.size)
```

Another Getter and Setter Example

```
class learnGetterSetter2 {

  private var privateAge = 0
  def age = privateAge //getter
  def age_=(newAge: Int) { if (newAge > privateAge)
privateAge = newAge } //setter
}
val a = new learnGetterSetter2

a.privateAge
a.privateAge_
a.age
a.age_=(10)
a.age
```

Primary Constructor

05 July 2020 11:50

To construct our objects

note

Example

```
class learnPrimaryConstructor(firstname: String,
                              lastName: String,
                              middleName: String) {
  println(firstname +' '+ lastName +' '+ middleName)
  def first() { println(firstname) }
  def middle() { println(middleName) }
first()
middle()
 }

val p1 = new
learnPrimaryConstructor("Ram" ,"","Singh")
```

Auxiliary Constructor

05 July 2020 12:33

Used for Constructor Overloading

KeyWord This - Auxiliary Constructor
First line of Auxiliary - We must call Primary Constructor or Previously Defined Auxiliary Constructor this keyword

Example 1

```
class learnAuxiliaryConstructor(firstname: String,
                                lastName: String,
                                middleName: String) {

  /**def this - Define an Auxiliary Constructor
     * Each constructor must call one of the previously defined constructors
  */
  println("Complete Name is " + firstname + lastName + middleName)

def this(firstname: String) {
    this(firstname, "", "")
    println("First Name is " + firstname)
  }
}

val p1 = new learnAuxiliaryConstructor("Ram" ,"Singh","")
val p2 = new learnAuxiliaryConstructor("Ram")
```

Example 2

```
class learnMultipleAuxuliaryConstructor(firstName: String,
                                    lastName: String,
                                    middleName: String) {

  /**def this - Define an Auxiliary Constructor
    *Rule - First Line of Auxiliary Constructor ,you have to call primary
constructor
    * While calling primary constructor , you need to pass all the arguments
    */
  println("This is primary constructor")
  println("Complete Name is " + firstName + lastName + middleName)
//First Auxiliary Constructor
def this(firstname: String) {
    this(firstname, "", "")
    println("This is Auxiliary constructor with firstname")
    println("First Name is " + firstName)
  }

//Can this be allowed
 // def this(lastname: String) {
  //  this("", lastname, "")
//    println("This is Auxiliary constructor with lastname")
//    println("lastname  is " + lastname)
//  }
//Another Auxiliary Constructor
  def this(lastname: String,middlename: String) {
   // this("")
 this("",lastname,middlename)
    println("This is Auxiliary constructor with Lastname and MiddleName")
    println("Last Name is " + lastName)
    println("Middle Name is " + middleName)
  }
}
val p1 = new learnMultipleAuxuliaryConstructor("Ram","Sharma","Pawan")
val p2 = new learnMultipleAuxuliaryConstructor("Ram")
val p3 = new learnMultipleAuxuliaryConstructor("Ram","Sharma")
```