Commands – black

Terminal – green

Comments – red

Output – blue

Notes – Orange

**07-04-2022**

scala (open the scala shell)

scala>

scala> :q (close the shell)

scala> val a = 100 (No need to specify datatype) (val says that a is constant and cannot be reassigned)

a: Int = 100

scala> a = 200

<console> :12: error: reassignment to val

a = 200

^

scala> var b = 100 (if you want to reassign, use “var”)

b: Int = 100

scala> b = 200

b: Int = 200

scala> var str = “Amrita”

str: String = Amrita

scala> var f = 3.14

f: Double = 3.14

scala> var ff = 3.14f

ff: Float = 3.14

scala> car ch = ‘A’

ch: Char = A

scala> var arr = Array(1, 2, 3, 4, 5)

arr : Array[Int] = Array(1, 2, 3, 4, 5)

scala> var arr1 = Array[Int](5)

arr1: Array[Int] = Array(0, 0, 0, 0, 0)

scala> arr1(0) = 100 (use () and not [] for the array)

scala> arr1(1) = 200

scala> arr1(2) = 300

scala> arr1(3) = 400

scala> arr1(4) = 500

scala> for(i<-0 to 4)

| {

| println(arr1)

| }

100

200

300

400

500

scala> arr1(2) = 350

scala> for(i<-0 to 4)

| {

| println(arr1)

| }

100

200

350

400

500

scala> var list1 = List(100, 200, 300, 400, 500)

list1: List[Int] = List(100, 200, 300, 400, 500)

scala> list1(2) = 350

<console>:13: error: value update is not a member of List[Int]

list1(2) = 350

^

scala> gedit FILENAME.scala (create and write the program)

Notes opens, type program, close editor and go back to terminal

scala> scala FILENAME.scala (to execute the program)

output comes

Data types in Scala

* Byte
* Short
* Int
* Long
* Char
* String
* Float
* Double
* Boolean

In Scala, array is mutable (change should take place in the same place itself)

Whereas, a list is immutable (in the same place, you cannot change i.e., a new list would be created)

To install scala in laptop:

sudo apt-get update

sudo apt-get install scala

In new terminal

gedit arraydemo.scala

CODE:

object Scala\_Array

{

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

{

var nums = Array(1.2, 1.7, 1.12, 1.16, 1.81, 1.99)

println(“Original array elements: “)

// print all the array elements

for(x<-nums)

{

//print(x+”,”+” “)

print(s”$x, “) //string interpolation

}

println(“\nUsing sum():”)

val result = nums.sum

println(s”Result: $result”);

println(”Result: “+result);

println(“\nUsing for loop:”)

var total = 0.0;

for(i<- 0 to (nums.length – 1))

{

total += nums(i);

}

println(s”Results: ${total}”);

}

}

Format:

defNameOfYourMethod(list of parameters) : return type =

{

//method body here

}

(a:Int, b:Float)

(name of param: datatype)

Method 1

scala

(paste the main function of the program)

Method 2

Sudo apt remove scala-library scala

Wget <https://downloads.lightbend.com/scala/2.13.4/scala-2.13.4.deb>

Sudo dpkg -i scala-2.13.4.deb

Method 3

scala -nobootcp -nc arraydemo.scala

1. Write a Scala program to reverse an array of integer values without using an built-in methods
2. Write a Scala program to insert an integer element into the given position of an integer array

Ex: Array of size 6 but you have only 5 integer elements. Elements -> 100, 200, 30, 400, 500. Read the position, at 2nd index you have 300, there you need to insert a new element and 300 will shift back by one

After insertion, array will look like 100, 200, 250, 300, 400, 500

**12-04-2022**

Tuples

* Collection of data of different datatypes
* Immutable
* Contains different types of elements
* Contain both an integer and a string at the same time
* Instantiate a new tuple:

val/var pair = (99, “Luftballons”)

* To access its elements:

println(pair.\_1)

println(pair.\_2)

* The actual type of a tuple depends on the number of elements it contains and the types of those elements.
* The type of (99, “Luftballons”) is Tuple2[Int, String]. The type of (‘u’, ‘r’, “the”, 1, 4, “me”) is Tuple6[Char, Char, String, Int, Int, String]

Array/ List/ Tuple

Var A = Array(1, 2, 3, 4, 5)

Var B = List(“Arun”, “Kavya”, “Vivek”)

Var C = (99, “Amruth”)

Var 7 = List((99, “Amruth”), (88, “Hello”))

Calling the Array/ List/ Tuple

A(2)

B(2)

Pair.\_2

Concatenation (:::)

* Combining two lists and the operator is :::

Cons (::)

* Prepends a new element to the beginning of an existing list and returns the resulting list
* We put “Nil” in the end if we have only integers (Eg: 1::2::3::Nil)

The syntax of a function literal in Scala

(x : Int, y : Int) => x + y



Function Right arrow Function body

parameters

in

parentheses

**19-04-2022**

* Scala is a functional programming language
* thrill is immutable (list is immutable in scala)
* thrill has 3 elements, after dropping two, thrill is expected to have any one element but if that happens, it can’t be called as thrill as it is immutable so a new list is created but not modified in thrill itself

scala> println(thrill)

List(will, fill, until)

scala> thrill.drop(2)

res8: List[String] = List(until)

scala> println(thrill)

List(will, fill, until)

scala> thrill.mkstring(“,”)

res10: String = will,fill,until

* Set: Immutable set (it never changes. But you have still operations that simulate additions, removes or updates, but those operations will in each case return a new collection and leave the old collection unchanged) and Mutable set (It can be updated / extended in place)
* Immutable set:

var jetSet = Set(“Boeing”, “Airbus”)

jetSet == “Lear”

println(“jetSet.contains(“Cessna”)

* Mutable set:

import scala.collection.mutable

var movieSet = Set(“Hitch”, “Poltergeist”)

movieSet == “Shrek”

println(movieSet)

* Map

Mutable Map:-

Import scala.collection.mutable

Val treasureMap = mutable.Map[Int, String]()

treasureMap += (1->”Go to island”)

treasureMap += (2->”Find big X on ground.”)

treasureMap += (3->”Dig.”)

println(treasureMap(2))

Immutable Map

Command line arguments in scala

Cd scala\_programs/

Scala whiledemo.scala Scala is fun

Whiledemo Var i=0

While (i<args.length)

{

Println(args(i))

i+=1

}

For

For (arg <- args)

**21-04-2022**

Class:

It is a blueprint for objects. Once you define a class, you can create objects from the class blueprint with the keyword new

Example ->

class ChecksumAccumulator {

// class definition

}

Members:

Fields and methods which are kept inside a class definition

Fields / Instance variables:

They can be defined with either val or car. They are variables

Methods:

We define methods with def and it contains executable code. Fields hold the state or data of an object, whereas the methods use that data to do the computational work of the object

def max(x:Int, y:Int): Int = {

if(x>y)

x

else

y

}

Public is scala’s default access level

Method parameters in Scala is that they are vals, not vars

Semicolon is required if you write multiple statements on a single line



Singleton Object:

* Scala has singleton object instead of Static in Java
* A singleton object definition looks like a class definition except instead of the keyword class we use the keyword object
* When a singleton object shares the same name with q class, it is called that class’s companion object
* The must define both the class and its companion in the same source file
* The class is called the companion class of the singleton object
* A class and its companion object can access each other’s private members

~/scala\_programs$ scala class\_checksum\_demo.scala

-248 (String -> “Every value is an object.”)

~/scala\_programs$ scala class\_checksum\_demo.scala

-248

~/scala\_programs$ scala class\_checksum\_demo.scala

-248

~/scala\_programs$ scala class\_checksum\_demo.scala

-248

~/scala\_programs$ scala class\_checksum\_demo.scala

-248

**21-04-2022 (LAB)**

**03-05-2022**

Class is a user defined datatype.

Axillary constructor helps us write a simpler code and omits the denominator 1.

**12-05-2022**

A controlled statement is called an expression.

Built in control structures : if, while, for, try, match and function calls

if expression:

var filename = “default.txt”

if(!args.isEmpty)

filename = args(0)

val filename = if(!args.isEmpty) args(0) else “default.txt”

for expression:

val filesHere = (new java.io.File(“.”)).listFiles

for(file <- filesHere)

println(file)

match is scala’s equivalent for switch

val firstArg = if (args.length > 0) args(0) else “ ”

firstArg match {

case “salt” => println(“pepper”)

case “chips” => println(“salsa”)

case “eggs” => println(“bacon”)

case \_ => println(“huh?”)

}

**12-05-2022 (Lab)**

Question 1

Create a class salesman with data fields name, company name, sales amount (array of sales amount for 12 months). This class should have the following methods:

1. Store - To store values to the data fields
2. Calculate – To calculate total incentives (criteria for calculating incentive -> if the salesman has achieved above 5 lakhs sales for a month, then 5% of sales amount to be considered as incentive)

Calculate total incentives for the whole year

1. Count – To count how many months the salesman has done sales over two lakhs

{Create a class salesman, it will have 3 to 4 fields, and 3 methods. The man method, you need to write inside standalone object (name of standalone object and scala file name (class name) is same). Inside the main method, you’ll create an object}

Question 2

Same as above but for a student class. It will have 3 to 4 fields like student name, reg no, array of marks and average. Store the values to these fields and apart from store method, you will have avg method for calculating avg of marks and another method is maximum which is used to find maximum marks stored by a particular student in a list of marks

PROJECT

DEVELOP SPARK APPLICATION.

USE SCALA OR PYTHON (PY SPARK FOR WORKING WITH PYTHON AND SPARK)

CAN BE A MACHINE LEARNING LANHUAGE

IN SPARK THERE IS A MACHINE LEARNING LIBRARY (ML LIB)

WORK WITH 10 GB DATA SET

USE GOOGLE COLLAB

NO TITLE WILL BE GIVEN

DATA ANALYTICS

WHEN YOU RUN THE APPLICATION, YOU SHOULD BE ABLE TO DEMONSTRATE THAT ALL THE CORES ARE BEING UTILISED

DEMONSTRATE REAL CLUSTER (INSTEAD OF PSEUDO DISTRIBUTED MODE)