**Academic Year 2024-25 Even**

**19CSE313 – Principles of Programming Language**

**B.Tech CSE 2022-26 F Section**

**Practice Set 8 – Array, List and Tuples in Scala**

**Scala Arrays**

Read a name from user and print a welcome message in the format “Hi \_\_\_\_\_\_,welcome to Scala” object Name{

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

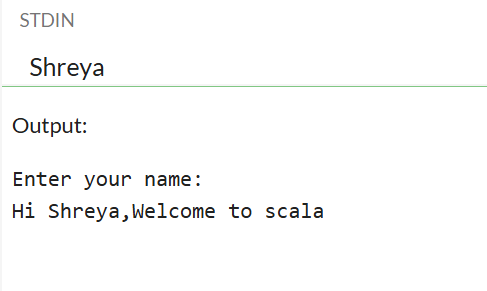
print("Enter your name: \n")

val name = scala.io.StdIn.readLine()

println(s"Hi $name,Welcome to scala")

}

}



Read an array of integers and find the maximum and minimum elements.

object MaxMinExample {

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

println("Enter integers separated by spaces:")

val numbers = scala.io.StdIn.readLine().split(" ").map(\_.toInt)

val maxElement = numbers.max

val minElement = numbers.min

println(s"Maximum element: $maxElement")

println(s"Minimum element: $minElement")

}

}

1.Read an array of N strings. Create another array of integers thatcontains the number of vowels in each string.

object VowelCount {

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

val n = scala.io.StdIn.readInt() // Read the number of inputs

val strings = Array.fill(n)(scala.io.StdIn.readLine()) // Read n strings

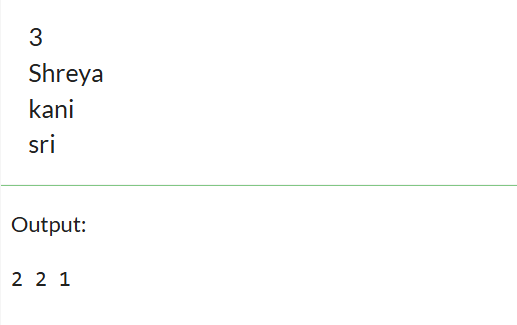
val vowels = Set('a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U')

val vowelCounts = strings.map(str => str.count(vowels.contains)) // Count vowels

println(vowelCounts.mkString(" "))

}

}



CHECK FOR THE PRESENCE OF VOWEL:

object VowelPresence {

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

val n = scala.io.StdIn.readInt() // Read number of strings

val strings = Array.fill(n)(scala.io.StdIn.readLine()) // Read n strings

val vowels = Set('a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U') // Vowel set

// Check if each string contains at least one vowel

val hasVowels = strings.map(str => str.exists(vowels.contains))

// Print results

hasVowels.foreach(result => println(if (result) "Yes" else "No"))

}

}

REPLACING THE VOWELS:

object VowelReplace {

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

val n = scala.io.StdIn.readInt() // Read number of strings

val strings = Array.fill(n)(scala.io.StdIn.readLine()) // Read n strings

val vowels = Set('a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U') // Vowel set

val processedStrings = strings.map { str =>

if (str.exists(vowels.contains))

str.map(ch => if (vowels.contains(ch)) '\*' else ch)

else

str

}

// Print the updated strings

processedStrings.foreach(println)

}

}

DELETING THE VOWELS:

object VowelDelete {

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

val n = scala.io.StdIn.readInt() // Number of strings

val strings = Array.fill(n)(scala.io.StdIn.readLine()) // Input strings

val vowels = Set('a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U') // Vowel set

val noVowelStrings = strings.map(str => str.filterNot(vowels.contains))

noVowelStrings.foreach(println)

}

}

1.Create an array for storing N integers. Split the numbers into two separate arrays such that the first array contains the prime numbers present in the original array and the second one contains the composite numbers. Write and use a separate Scala function to check whether a number is prime or composite.

object PrimeCompositeSplit {

def isPrime(n: Int): Boolean = {

if (n < 2) return false // Numbers less than 2 are not prime

for (i <- 2 to n) {

if (n % i == 0) return false // If n is divisible by any number, it's not prime

}

true // If no divisors found, it's prime

}

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

print("Enter numbers (space-separated): ")

val numbers = scala.io.StdIn.readLine().split(" ").map(\_.toInt)

val primes = numbers.filter(isPrime)

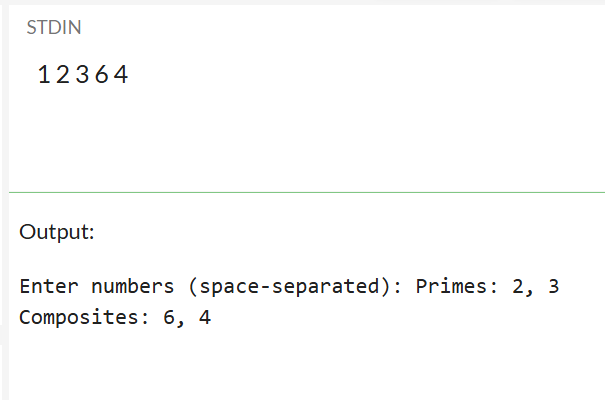
val composites = numbers.filter(n => n > 1 && !isPrime(n))

println(s"Primes: ${primes.mkString(", ")}")

println(s"Composites: ${composites.mkString(", ")}")

}

}



object PrimeCompositeSplit {

def isPrime(n: Int): Boolean = {

if (n < 2) return false // Numbers less than 2 are not prime

for (i <- 2 until n) {

if (n % i == 0) return false // If n is divisible by any number, it's not prime

}

true // If no divisors found, it's prime

}

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

print("Enter the number of elements: ")

val n = scala.io.StdIn.readInt() // Read the number of elements

println(s"Enter $n numbers:")

val numbers = Array.fill(n)(scala.io.StdIn.readInt()) // Read n integers into an array

val primes = numbers.filter(isPrime)

val composites = numbers.filter(n => n > 1 && !isPrime(n))

println(s"Primes: ${primes.mkString(", ")}")

println(s"Composites: ${composites.mkString(", ")}")

}

}

**Scala List**

1. Creating a list in Scala

object Scala\_List

{

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

{

println("Create a Scala List:")

println("Lisp style:")

val lisp\_list = 100 :: 200 :: 300 :: Nil :: 400 :: Nil

println(lisp\_list)

println("Java style:")

val nums = List(1,2,3,4,5,6,7)

println(nums)

println("Mixed type values in a list:")

val x = List[Number](100, 200, 110.20, 45d, 0x1)

println(x)

println("Range List:")

val y = List.range(1, 20)

println(y)

val z = List.range(0, 30, 3)

println(z)

println("Uniform List:")

val s = List.fill(5)("Scala")

println(s)

val n = List.fill(3)(4)

println(n)

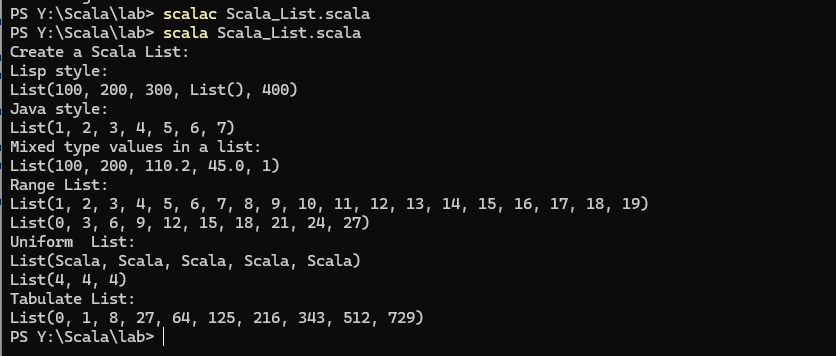
println("Tabulate List:")

val t = List.tabulate(10)(n => n \* n \* n)

println(t)

}

}



1. Write a scala program to count the number of elements in the given list.

object countele{

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

val n1=scala.io.StdIn.readLine().split(" ").map(\_.toInt).toList

val c=n1.length

println(s"the length is $c")

}

}

1. ListBuffer and List – Adding Items

object Scala\_List

{

import scala.collection.mutable.ListBuffer

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

{

//As a List is immutable we use ListBuffer

var colors = new ListBuffer[String]()

println("Add Single element in the said list:")

colors += "Red"

colors += "Green"

colors += "Black"

println(colors)

println("Add multiple elements in the said list:")

colors ++= List("Orange", "Pink", "Black")

println(colors)

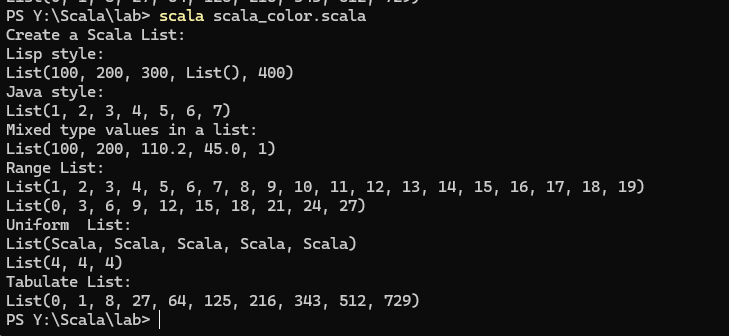
println("Convert the ListBuffer to a List:")

val colors\_list = colors.toList

println(colors\_list)

}

}



1. Removing Items

object Scala\_List

{

import scala.collection.mutable.ListBuffer

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

{

//As a List is immutable we use ListBuffer

//and finally convert the ListBuffer to list.

var colors = new ListBuffer[String]()

colors += "Red"

colors += "Green"

colors += "Black"

colors += "Orange"

colors += "Pink"

println("Original ListBuffer:")

println(colors)

println("Remove one element:")

colors -= "Red"

println(colors)

println("Remove multiple elements:")

println(colors)

colors --= Seq("Black", "Pink")

println("After removing two elements, final ListBuffer:")

println(colors)

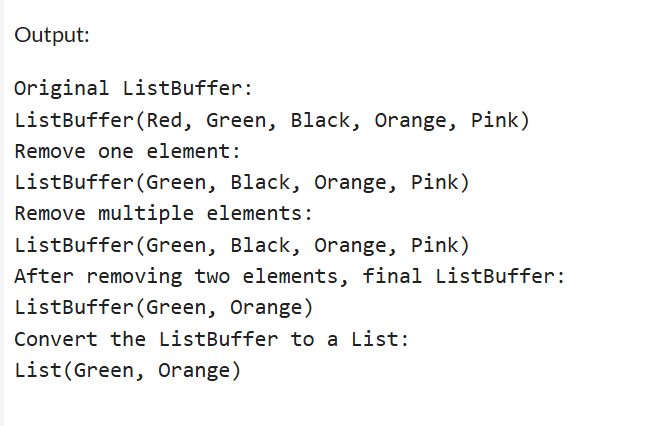
println("Convert the ListBuffer to a List:")

val colors\_list = colors.toList

println(colors\_list)

}

}



1. Edit the program in 3 such that you already have a list and must remove an item from the list [ list is immutable. You need to convert the list to ListBuffer if you need to do any add/delete]

object Scala\_List {

import scala.collection.mutable.ListBuffer

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

// Step 1: Define an immutable List

val colors\_list = List("Red", "Green", "Black", "Orange", "Pink", "Black")

println("Original List:")

println(colors\_list)

// Step 2: Convert immutable List to mutable ListBuffer

var colorsBuffer = ListBuffer(colors\_list: \_\*) // Converts List to ListBuffer

println("List converted to ListBuffer:")

// Step 3: Remove an element from ListBuffer

colorsBuffer -= "Black" // Removes the first occurrence of "Black"

println("ListBuffer after removing 'Black':")

println(colorsBuffer)

// Step 4: Convert back to immutable List

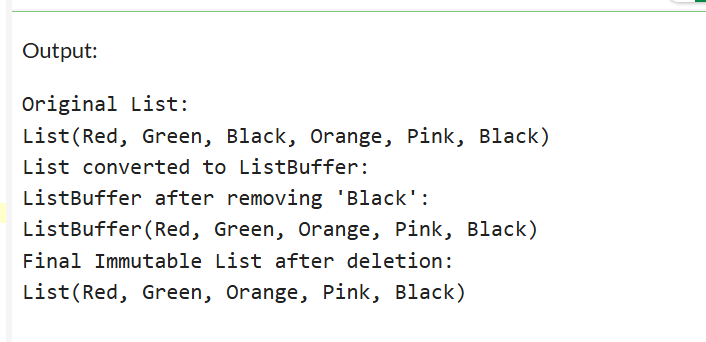
val updated\_colors\_list = colorsBuffer.toList

println("Final Immutable List after deletion:")

println(updated\_colors\_list)

}

}



1. Write a Scala program to iterate over a list to print the elements and calculate the sum and product of all elements of this list.

object Sumandprod{

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

val n=scala.io.StdIn.readInt()

val num=scala.io.StdIn.readLine().split(" ").map(\_.toInt).toList

var sum=0

println(num)

for(ele<-num)

{

sum+=ele

}

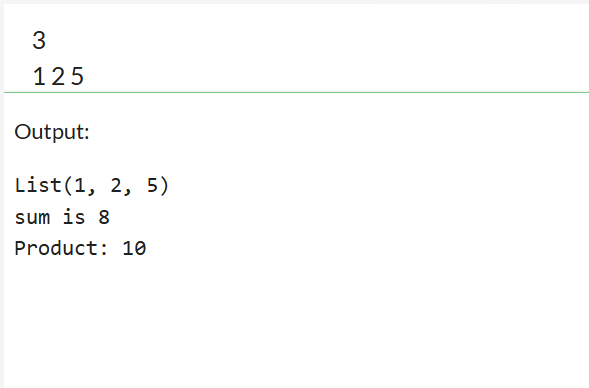
println(s"sum is $sum")

val product = num.product

println(s"Product: $product")

}

}



1. Write a Scala program to find the largest and smallest number from a given list.

object minmax{

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

val n=scala.io.StdIn.readInt()

val num=scala.io.StdIn.readLine().split(" ").map(\_.toInt).toList

var maxele=num(0)

var minele=num(0)

for(ele<-num)

{

if(ele>maxele) maxele=ele

if(ele<minele) minele=ele

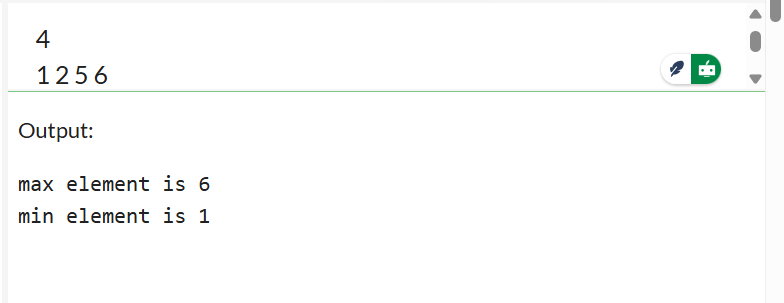
}

println(s"max element is $maxele")

println(s"min element is $minele")

}

}



1. Write a scala program to delete the duplicate elements in a list.

**BUILT-IN**

object uniquelist{

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

val n=scala.io.StdIn.readInt()

val num=scala.io.StdIn.readLine().split(" ").map(\_.toInt).toList

val un=num.distinct

println(s"List after removing duplicates: ${un.mkString(", ")}")

}

}

**MANUAL**

object RemoveDuplicates {

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

val n = scala.io.StdIn.readInt()

val num = scala.io.StdIn.readLine().split(" ").take(n).map(\_.toInt).toList

var seen = Set[Int]() // Track seen elements

var uniqueList = List[Int]() // Store unique elements

for (element <- num) {

if (!seen.contains(element)) { // If not seen before, add to list

uniqueList = uniqueList :+ element

seen += element

}

}

println(s"List after removing duplicates: ${uniqueList.mkString(", ")}")

}

}

1. Write a scala program to find the difference between two lists [L1-L2 is the set of elements in L1 but not in L2]

object ListDifference {

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

val n1 = scala.io.StdIn.readInt() // Read size of L1

val L1 = scala.io.StdIn.readLine().split(" ").take(n1).map(\_.toInt).toList

val n2 = scala.io.StdIn.readInt() // Read size of L2

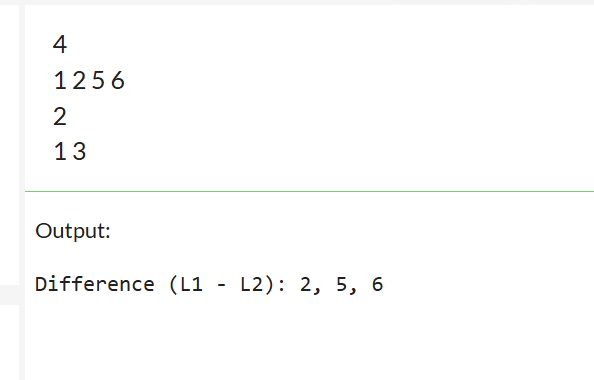
val L2 = scala.io.StdIn.readLine().split(" ").take(n2).map(\_.toInt).toList

val difference = L1.diff(L2) // Finds L1 - L2

println(s"Difference (L1 - L2): ${difference.mkString(", ")}")

}

}



object ListDifference {

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

val n1 = scala.io.StdIn.readInt()

val L1 = scala.io.StdIn.readLine().split(" ").take(n1).map(\_.toInt).toList

val n2 = scala.io.StdIn.readInt()

val L2 = scala.io.StdIn.readLine().split(" ").take(n2).map(\_.toInt).toList

val difference = L1.filterNot(L2.contains) // Keep only elements not in L2

println(s"Difference (L1 - L2): ${difference.mkString(", ")}")

}

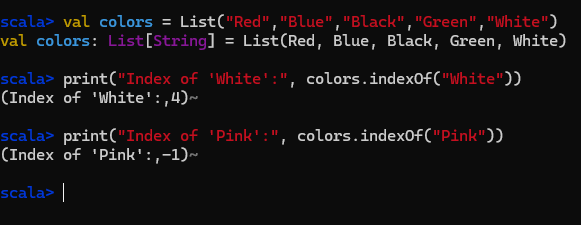
}

1. Try the following:

val colors = List("Red","Blue","Black","Green","White")

println("Index of 'White':", colors.indexOf("White"))

println("Index of 'Pink':", colors.indexOf("Pink"))



1. Merging two lists – different ways

val nums1 = List(1,3,5,7,9)

val nums2 = List(2,4,6,8,10)

val nums\_1 = nums1 ++ nums2

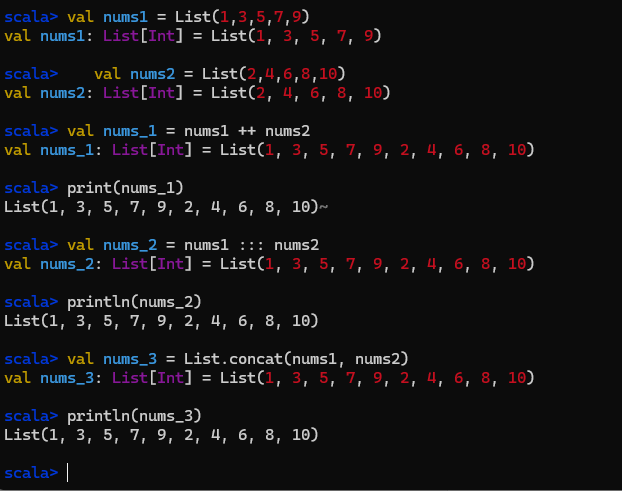
println(nums\_1)

val nums\_2 = nums1 ::: nums2

println(nums\_2)

val nums\_3 = List.concat(nums1, nums2)

println(nums\_3)



1. Write a Scala Program to find the scalar product of two lists.

object ScalarProduct {

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

val n = scala.io.StdIn.readInt()

val L1 = scala.io.StdIn.readLine().split(" ").take(n).map(\_.toInt).toList

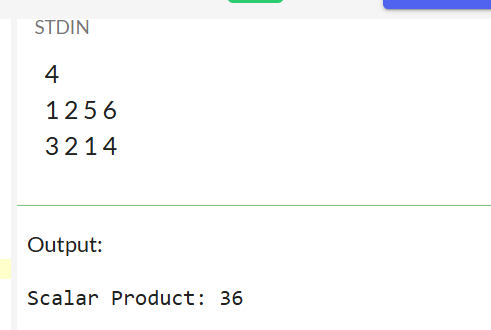
val L2 = scala.io.StdIn.readLine().split(" ").take(n).map(\_.toInt).toList

val dotProduct = L1.zip(L2).map { case (x, y) => x \* y }.sum

println(s"Scalar Product: $dotProduct")

}

}



1. Write a Scala Program to find the Kth smallest element in a given list.

object KthSmallest {

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

val n = scala.io.StdIn.readInt() // Read number of elements

val L = scala.io.StdIn.readLine().split(" ").take(n).map(\_.toInt).toList

val k = scala.io.StdIn.readInt() // Read K value

if (k > 0 && k <= L.length) {

val kthSmallest = L.sorted.apply(k - 1) // Sort and get (K-1)th element

println(s"$k-th smallest element: $kthSmallest")

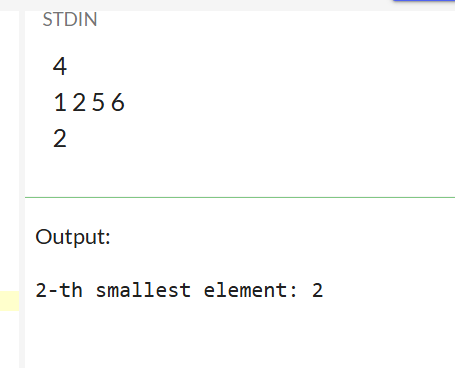
} else {

println("Invalid K value!")

}

}

}



1. Write a Scala Program to find the Kth largest element in a given list.

object KthLargest {

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

val n = scala.io.StdIn.readInt() // Read number of elements

val L = scala.io.StdIn.readLine().split(" ").take(n).map(\_.toInt).toList

val k = scala.io.StdIn.readInt() // Read K value

if (k > 0 && k <= L.length) {

val kthLargest = L.sorted(Ordering[Int].reverse).apply(k - 1) // Sort Descending

println(s"$k-th largest element: $kthLargest")

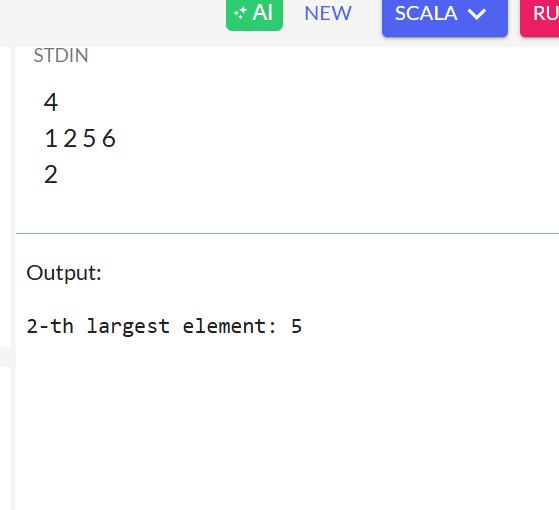
} else {

println("Invalid K value!")

}

}

}



**Scala Tuples**

1. Checking whether a tuple is empty or not

object CheckTupleEmptyExample {

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

val tuple1 = ()

val tuple2 = ("Red", 10, true)

val isEmpty1 = tuple1 == ()

val isEmpty2 = tuple2 == ()

println("Is tuple1 is empty? " + isEmpty1)

println("Is tuple2 empty? " + isEmpty2)

}

}

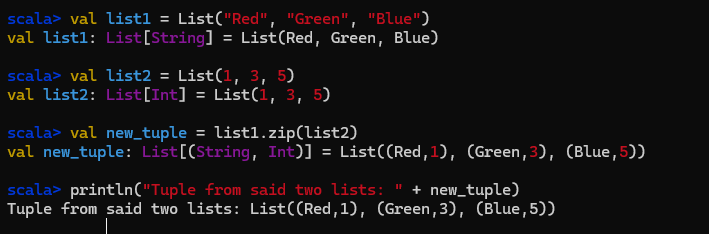
1. Creating a tuple from a list

val list1 = List("Red", "Green", "Blue")

val list2 = List(1, 3, 5)

val new\_tuple = list1.zip(list2)

println("Tuple from said two lists: " + new\_tuple)



1. What does the code snippet do?

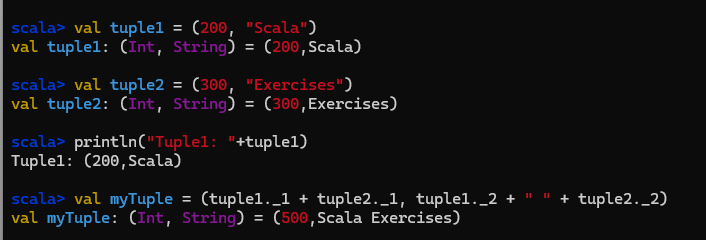
val tuple1 = (200, "Scala")

val tuple2 = (300, "Exercises")

println("Tuple1: "+tuple1)

val myTuple = (tuple1.\_1 + tuple2.\_1, tuple1.\_2 + " " + tuple2.\_2)

println("Concatenated tuple: " + concatenatedTuple)





1. Write a Scala program to find distinct elements of a tuple.

object DistinctTupleElements {

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

val myTuple = (1, 2, 3, 2, 4, 1, 5, 6, 3) // Tuple with duplicate elements

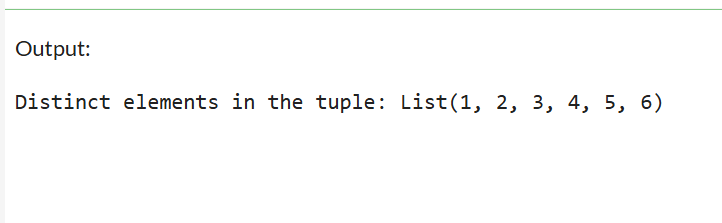
// Convert tuple to List and get distinct elements

val distinctElements = myTuple.productIterator.toList.distinct

println("Distinct elements in the tuple: " + distinctElements)

}

}



1. Write a Scala program to check whether two tuples are equal.

object TupleEqualityCheck {

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

val tuple1 = (1, "Scala", true)

val tuple2 = (1, "Scala", true)

val tuple3 = (1, "Java", false)

// Checking if tuples are equal

println("Is tuple1 equal to tuple2? " + (tuple1 == tuple2)) // true

println("Is tuple1 equal to tuple3? " + (tuple1 == tuple3)) // false

}

}

