Scala Programs

# Scala Basic Programs

1. Write a Scala program to create a new string which is n (non-negative integer) copies of a given string.

object scala\_basic {

def test(str1: String, n: Int): String = {

str1 \* n;

}

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

println("Result: " + test("Scala", 2));

println("Result: " + test("Python",1));

println("Result: " + test("JS",6));

}

}

1. Write a Scala program to check whether a given character presents in a string between 2 to 4 times.

object scala\_basic {

def test(str1: String): Boolean = {

val count\_char = str1.count{ n => n == 'z' }

count\_char >= 2 && count\_char <= 4

}

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

println("Result: " + test("frizz"));

println("Result: " + test("zane"));

println("Result: " + test("Zazz"));

println("Result: " + test("false"));

}

}

1. Write a Scala program to check whether two given positive integers have the same last digit.

object scala\_basic {

def test(x: Int, y: Int): Boolean = {

Math.abs(x % 10) == Math.abs(y % 10);

}

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

println("Result: " + test(123, 456));

println("Result: " + test(12, 512));

println("Result: " + test(7, 87));

println("Result: " + test(12, 45));

}

}

1. Write a Scala program to convert the last 4 characters of a given string in upper case. If the length of the string has less than 4 then uppercase all the characters.

object scala\_basic {

def test(str1: String): String = {

str1.take(str1.length() - 4) + str1.drop(str1.length() - 4).toUpperCase()

}

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

println("Result: " + test("Scala"));

println("Result: " + test("Python"));

println("Result: " + test("abc"));

}

}

1. Write a Scala program to check the largest number among three given integers.

object scala\_basic {

def test(x: Int, y: Int, z: Int): Int = {

List(x, y, z).max

}

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

println("Result: " + test(1,2,3));

println("Result: " + test(1,3,2));

println("Result: " + test(1,1,1));

println("Result: " + test(1,2,2));

}

}

# Scala String Exercises

1. Write a Scala program to add a string with specific number of times separated by a substring.   
   Sample Output:

The given strings are: try  and  best

Number to times to be repeat: 3

The new string is: trybesttrybesttry

object Scala\_String {

def test(main\_str: String, sep\_str: String, ctr: Int): String = {

var new\_word = "";

for (i <- 0 to ctr - 1) {

if (i < ctr - 1)

new\_word = new\_word + (main\_str + sep\_str)

else

new\_word += main\_str;

}

return new\_word;

}

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

val str1 = "try";

val str2 = "best";

var ctr = 3;

println("The given strings are: " + str1 + " and " + str2);

println("Number to times to be repeat: " + ctr);

println("The new string is: " + test(str1, str2, ctr));

}

}

1. Write a Scala program to repeat a specific number of characters for specific number of times from the last part of a given string   
   Sample Output:

The given string is: string

The new string after repetition: inginging

object Scala\_String {

def test(stng: String, no\_repeat: Int): String = {

val l = stng.length;

var new\_word = "";

for (i <- 0 to no\_repeat - 1) {

new\_word += stng.substring(l - no\_repeat, l);

}

new\_word;

}

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

val str1 = "string";

val no\_char = 3;

println("The given string is: " + str1);

println("The new string after repetition: " + test(str1, no\_char));

}

}

1. Write a Scala program to count the number of triples (characters appearing three times in a row) in a given string.   
   Sample Output:

The given string is: welllcommmmeee

The number of triples in the string is: 4

object Scala\_String {

def test(stng: String): Int = {

var l = stng.length();

var ctr = 0;

for (i <- 0 to l-3)

{

var tmp = stng.charAt(i);

if (tmp == stng.charAt(i+1) && tmp == stng.charAt(i+2))

ctr=ctr+1;

}

return ctr;

}

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

val str1 = "welllcommmmeee";

println("The given string is: "+str1);

println("The number of triples in the string is: "+test(str1));

}

}

1. Write a Scala program to check whether a specified character is happy or not. A character is happy when the same character appears to its left or right in a string.   
   Sample Output:

The given string is: azzlea

Is z happy in the said string: true

The given string is: abcfdkefg

Is f happy in the said string: false

object Scala\_String {

def test(stng: String, spc: Char): Boolean = {

var l = stng.length();

var char\_happy = true;

for (i <- 0 to l - 1) {

if (stng.charAt(i) == spc) {

if (i > 0 && stng.charAt(i - 1) == spc)

char\_happy = true;

else if (i < l - 1 && stng.charAt(i + 1) == spc)

char\_happy = true;

else

char\_happy = false;

}

}

char\_happy;

}

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

var str1 = "azzlea";

var spc = 'z'

println("The given string is: " + str1);

println("Is " + spc + " happy in the said string: " + test(str1, spc));

str1 = "abcfdkefg";

spc = 'f'

println("The given string is: " + str1);

println("Is " + spc + " happy in the said string: " + test(str1, spc));

}

}

1. Write a Scala program to calculate the sum of the numbers appear in a given string.   
   Sample Output:

The given string is: it 15 is25 a 20string

The sum of the numbers in the said string is: 60

object Scala\_String {

def test(stng: String): Int = {

val l = stng.length;

var sum = 0;

var temp = "";

for (i <- 0 to l-1)

{

if (Character.isDigit(stng.charAt(i)))

{

if (i < l-1 && Character.isDigit(stng.charAt(i+1)))

{

temp += stng.charAt(i);

}

else

{

temp += stng.charAt(i);

sum += Integer.parseInt(temp);

temp = "";

}

}

}

sum;

}

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

val str1 = "it 15 is25 a 20string";

println("The given string is: "+str1);

println("The sum of the numbers in the said string is: "+test(str1));

}

}

## Scala List Programs

1. Write a Scala program to delete element(s) from a given List.

object Scala\_List

{

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

{

val nums = List(1, 3, 5, 7, 9, 11, 14, 12)

println("Original list:")

println(nums)

//As scala List is immutable, so we can’t delete elements from it, but

//filter out element(s) as per requirement.

println("Filter out 3 from the above list:")

val nums1 = nums.filter(\_ != 3)

println(nums1)

println("Filter out numbers which are greater than 10:")

val nums2 = nums.filter(\_ > 10)

println(nums2)

}

}

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 Scala\_List

{

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

{

//Iterate over a list

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

println("Iterate over a list:")

for( i <- nums)

{

println(i)

}

println("Sum all the items of the said list:")

//Applying sum method

val result = nums.sum

println(result)

println("Multiplies all the items of the said list:")

val result1 = nums.product

println(result1)

}

}

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

object Scala\_List

{

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

{

//Iterate over a list

val nums = List(1, 3, 5, 7, 9, 11, 14, 12)

println("Original list:")

println(nums)

println("Largest number of the said list:")

println(nums.max)

println("Smallest number from the said list:")

println(nums.min)

}

}

1. Write a Scala program to remove duplicates from a given list.

object Scala\_List

{

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

{

val nums = List(1, 3, 5, 2, 7, 9, 11, 5, 2, 14, 12, 3)

println("Original list:")

println(nums)

val result1 = nums.distinct

println("Unique elements of the said list:")

println(result1)

val chars = List("a", "a", "b", "c", "d", "c", "e", "f")

println("Original list:")

println(chars)

val result2 = chars.distinct

println("Unique elements of the said list:")

println(result2)

}

}

1. Write a Scala program to check a given list is empty or not.

object Scala\_List

{

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

{

val nums = List(1, 3, 5, 2, 7, 9, 11, 5, 2, 14, 12, 3)

println("Original list:")

println(nums)

val result = nums.isEmpty

println("Test the said list is empty or not?")

println(result)

val nums1 = List()

val result1 = nums1.isEmpty

println("Test the said list is empty or not?")

println(result1)

}

}

# Scala Array Exercises

1. Write a Scala program to sum values of a given array.

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(s"${x}, ")

}

println("\nUsing sum():")

val result = nums.sum

println(s"Result: ${result}");

println("\nUsing for loop:")

var total = 0.0;

for ( i <- 0 to (nums.length - 1)) {

total += nums(i);

}

println(s"Result: ${total}");

}

}

1. Write a Scala program to remove a specific element from a given array.   
   Note: The size of an Array can't be changed, so we can't directly delete elements from an array but replace them with "" / null etc.

object Scala\_Array

{

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

{

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

println("Original Array elements:")

// Print all the array elements

for ( x <- colors ) {

print(s"${x}, ")

}

println("\nReplace some elements with ''/null etc.:")

colors(0) = ""

colors(3) = null

println("Now the Original Array becomes:")

// Print all the array elements

for ( x <- colors ) {

print(s"${x}, ")

}

}

}

1. Write a Scala program to find the maximum and minimum value of an array of integers.

object Scala\_Array {

def min\_max(a: Array[Int]) : (Int, Int) = {

if (a.isEmpty) throw new IllegalArgumentException("Blank Array...!")

a.foldLeft((a(0), a(0)))

{ case ((min, max), e) => (math.min(min, e), math.max(max, e))}

}

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

val nums1 = Array(1, 2, 3, 4, 5, 7, 9, 11, 14, 12, 16)

println("Original Array elements:")

// Print all the array elements

for ( x <- nums1 ) {

print(s"${x}, ")

}

println("\nMaximum - Minimum value of the said array: " +min\_max(nums1));

val nums2 = Array(-111,-124)

println("Original Array elements:")

// Print all the array elements

for ( x <- nums2 ) {

print(s"${x}, ")

}

println("\nMaximum - Minimum value of the said array: " +min\_max(nums2));

val nums3 = Array(10)

println("Original Array elements:")

// Print all the array elements

for ( x <- nums3 ) {

print(s"${x}, ")

}

println("\nMaximum - Minimum value of the said array: " +min\_max(nums3));

var nums4 : Array[Int] = Array()

println("Result: " +min\_max(nums4));

}

}

1. Write a Scala program to create a new array taking the middle element from three arrays of length 5.

object Scala\_Array {

def test(x: Array[Int], y: Array[Int], z: Array[Int]): Array[Int] = {

if (x.length != 5 || y.length != 5 || z.length != 5) throw new IllegalArgumentException("Array length not matched!")

else Array(x(2), y(2), z(2))

}

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

var result1 = test(Array(1,2,3,4,5),Array(2,3,4,5,6),Array(3,4,5,6,7))

// Print all the array elements

println("New array:")

for ( x <- result1 ) {

print(s"${x}, ")

}

}

}

1. Write a Scala program to reverse an array of integer values.

object Scala\_Array {

def test(nums: Array[Int]): Array[Int] = {

var temp1 = 0

var temp2 = 0

var index\_position = 0

var index\_last\_pos = nums.length - 1

while (index\_position < index\_last\_pos) {

temp1 = nums(index\_position)

temp2 = nums(index\_last\_pos)

nums(index\_position) = temp2

nums(index\_last\_pos) = temp1

index\_position += 1

index\_last\_pos -= 1

}

nums

}

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

var nums1 = Array(1789, 2035, 1899, 1456, 2013)

println("Orginal array:")

for ( x <- nums1) {

print(s"${x}, ")

}

var result1= test(nums1)

println("\nReversed array:")

for ( x <- result1) {

print(s"${x}, ")

}

var nums2 = Array(1789, 2035, 1899, 1456)

println("\nOrginal array:")

for ( x <- nums2) {

print(s"${x}, ")

}

var result2= test(nums2)

println("\nnReversed array:")

for ( x <- result2) {

print(s"${x}, ")

}

}

}