Big Data Hadoop and Spark Developer

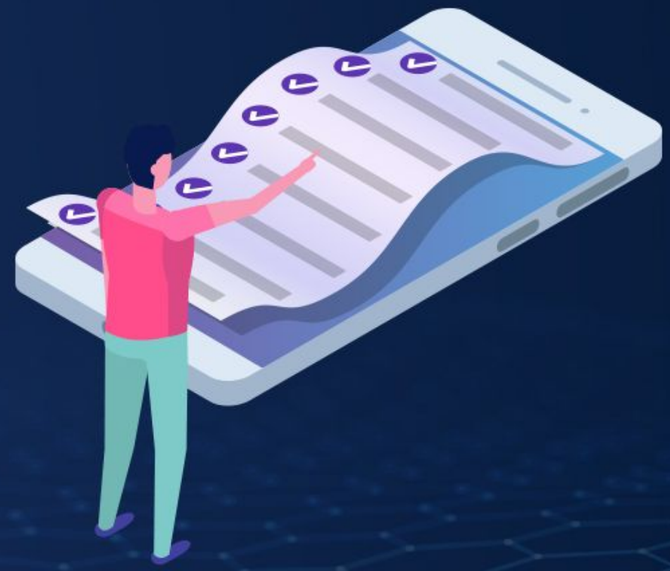
simplilearn

Basics of Functional Programming and Scala

Learning Objectives

By the end of this lesson, you will be able to:

- Describe and demonstrate programming with Scala
- Define functions, anonymous function, and class in Scala
- Use the different types of collections
- Describe and demonstrate Scala REPL (Read, Eval, Print, and Loop)



DATA AND
ARTIFICIAL INTELLIGENCE

Introduction to Scala

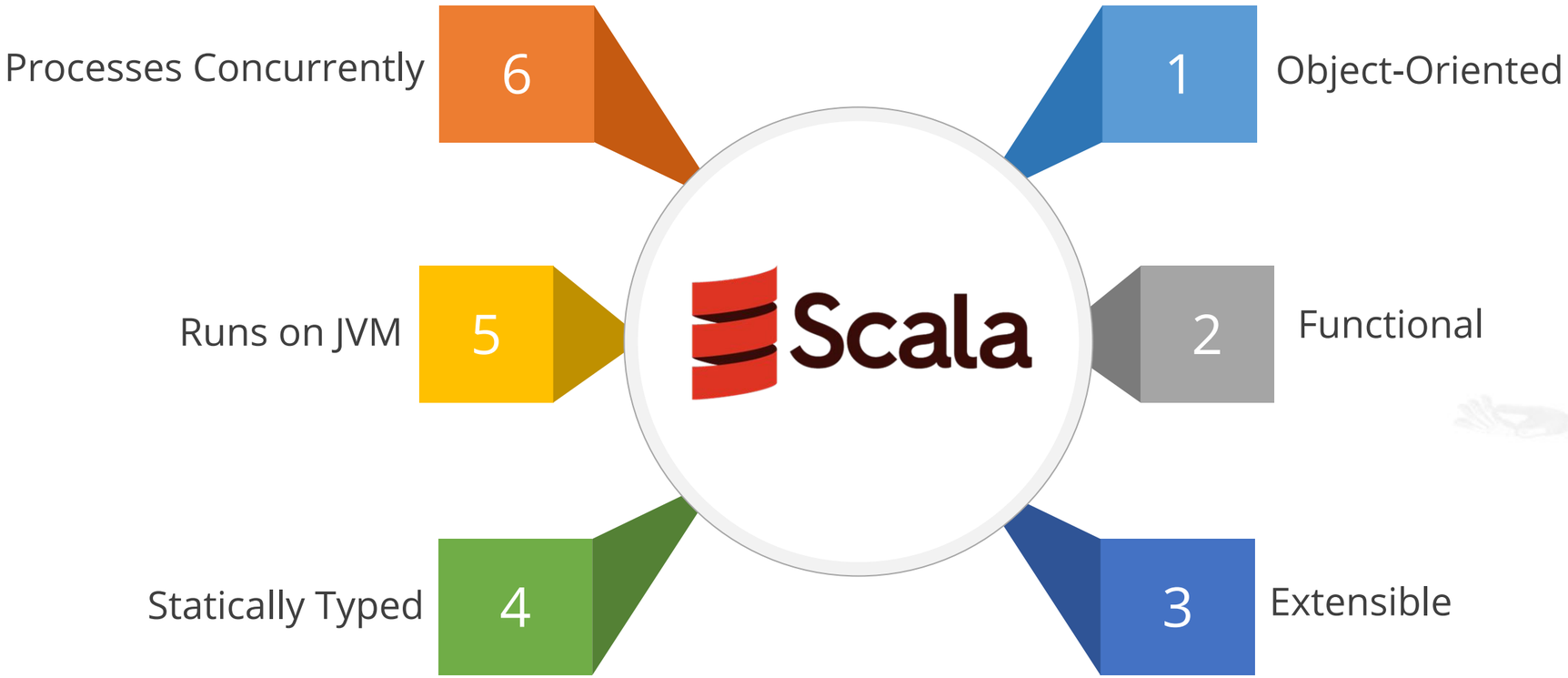
Scala

Scala

Scala

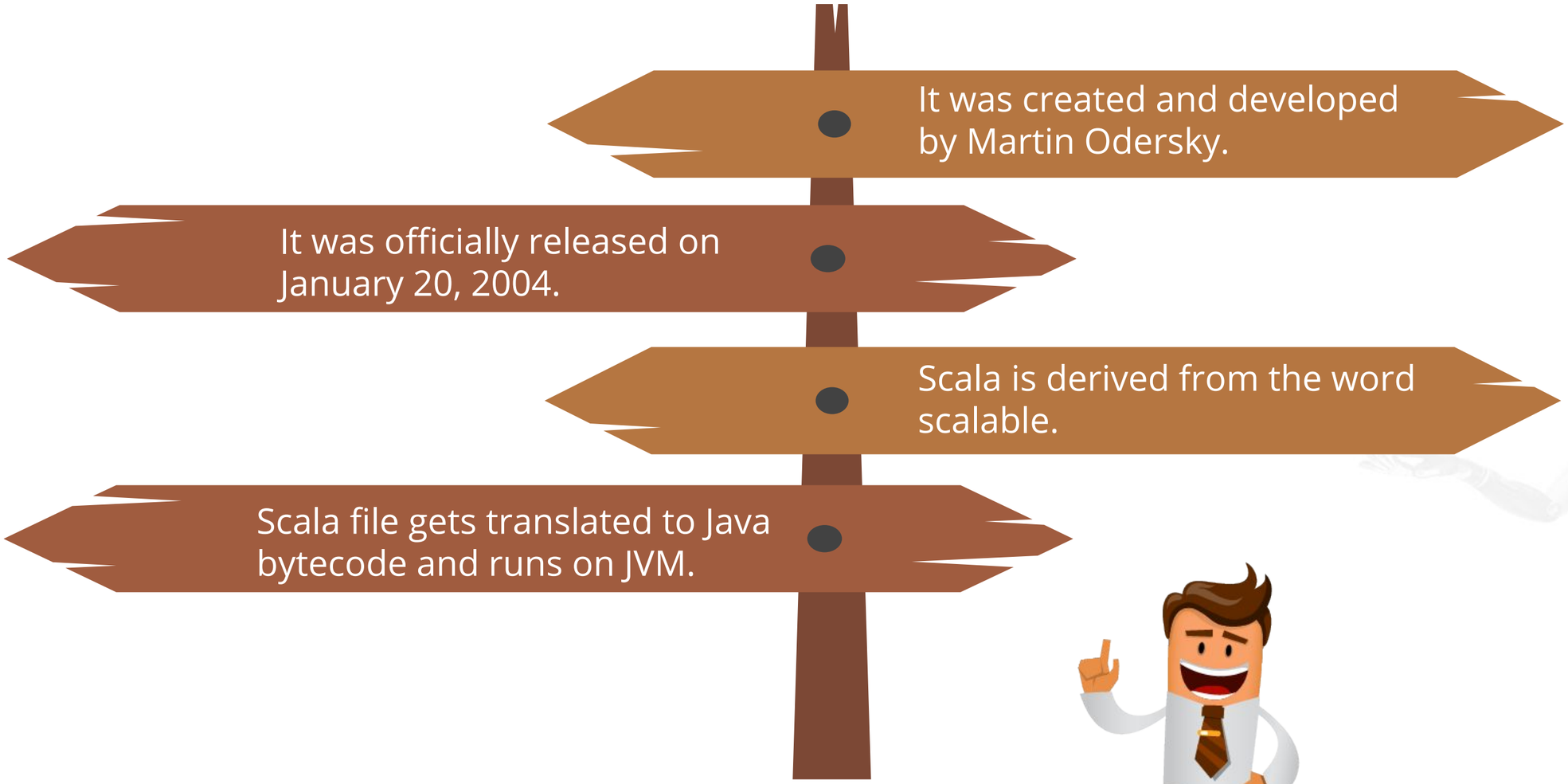
Scala is an object-oriented and functional programming language designed to demonstrate the common programming patterns in a concise, refined, and type-safe way.

Features of Scala



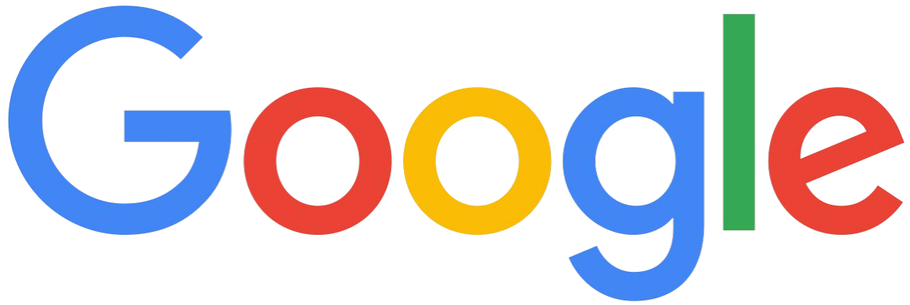
Scala

History of Scala

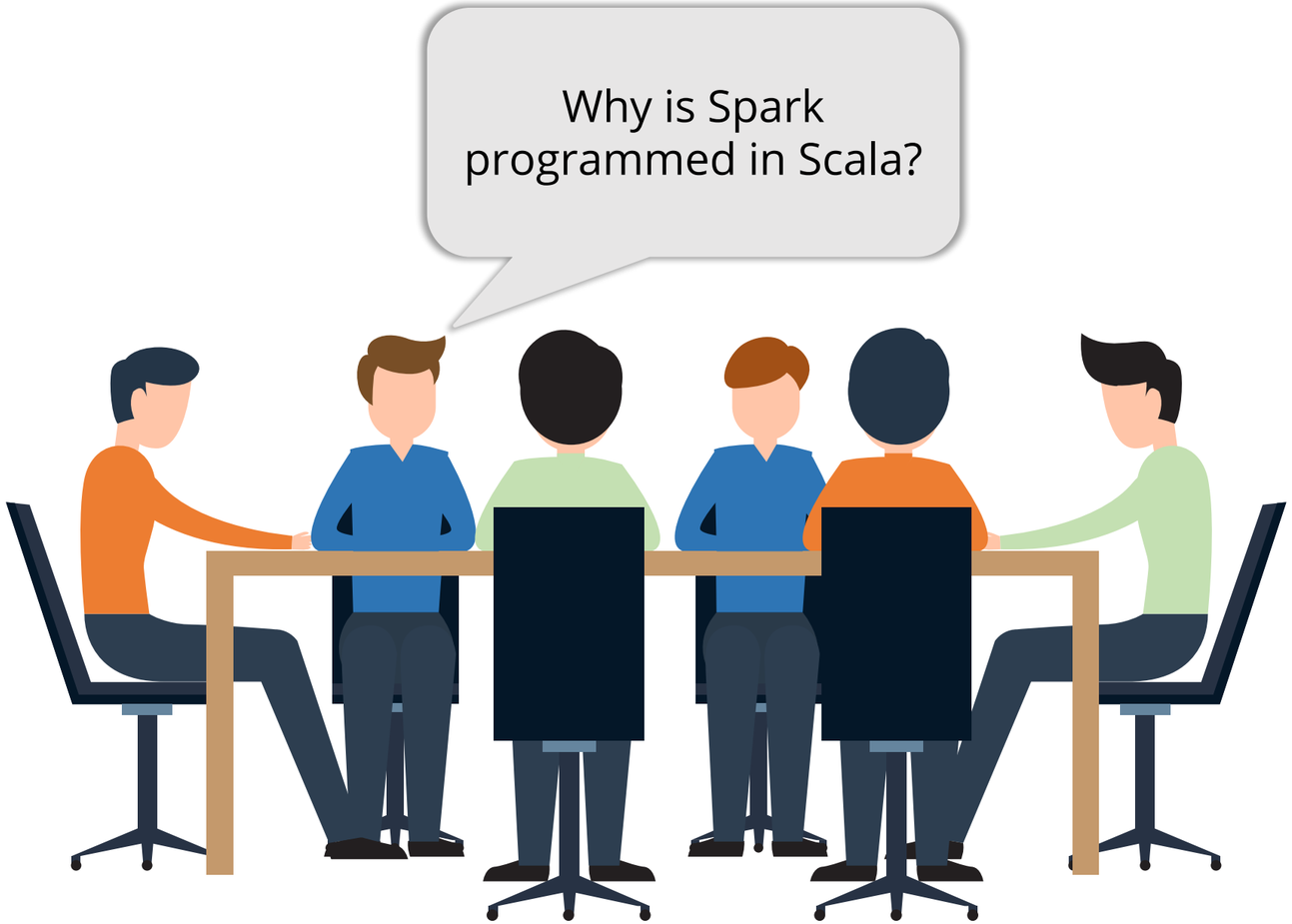


Popularity of Scala

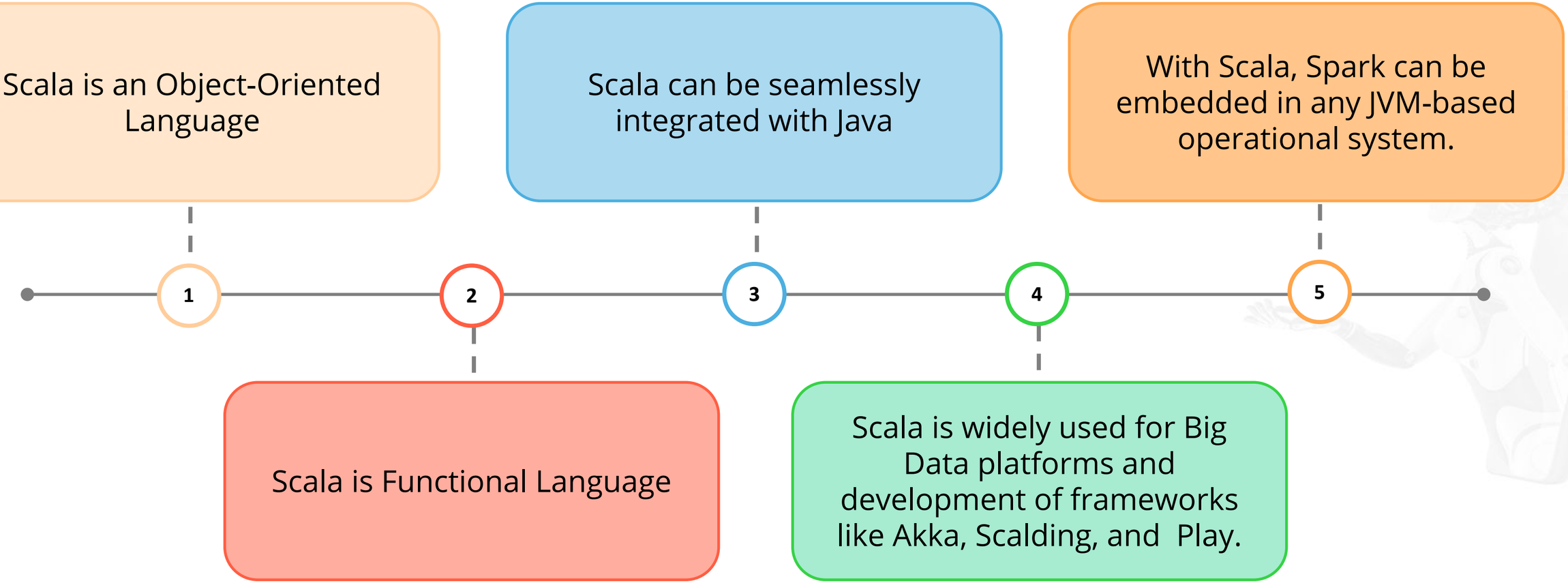
The following companies use Scala, which makes it popular:



Why Spark Programmed in Scala?



Why Is Spark Programmed in Scala?



Install Scala

Duration: 5 mins

Problem Statement: In this demonstration, you will install Scala.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

DATA AND
ARTIFICIAL INTELLIGENCE

Object Oriented Programming

Object Oriented Programming

Object-oriented programming (OOP) is a computer programming model that organizes software design rather than functions and logic around the data or objects. An object can be defined as a data field that has unique attributes and behavior.

OOP focuses on the objects that developers want to manipulate rather than the logic required to manipulate them. This approach to programming is well-suited for programs that are large, complex, and actively updated or maintained.

DATA AND
ARTIFICIAL INTELLIGENCE

Functional Programming

Functional Programming

Functional Programming

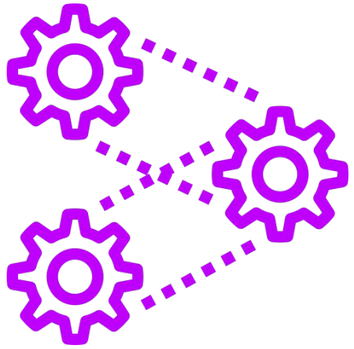
Functional programming languages are designed on the concept of mathematical functions that use conditional expressions and recursion to perform computation.

Pure Functional Language

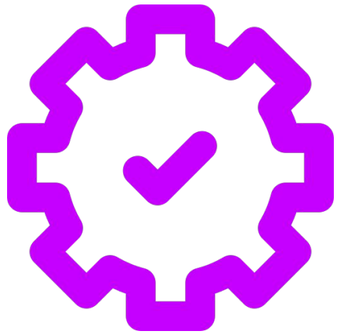
Impure Functional Language

Advantages of Functional Programming

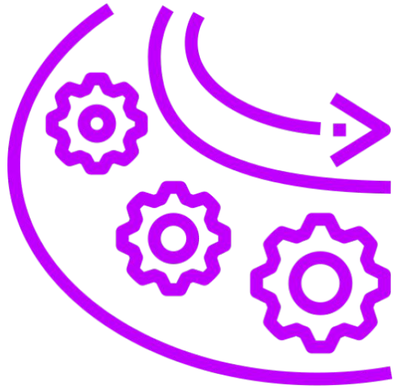
Bug Free



Efficient Parallel Processing



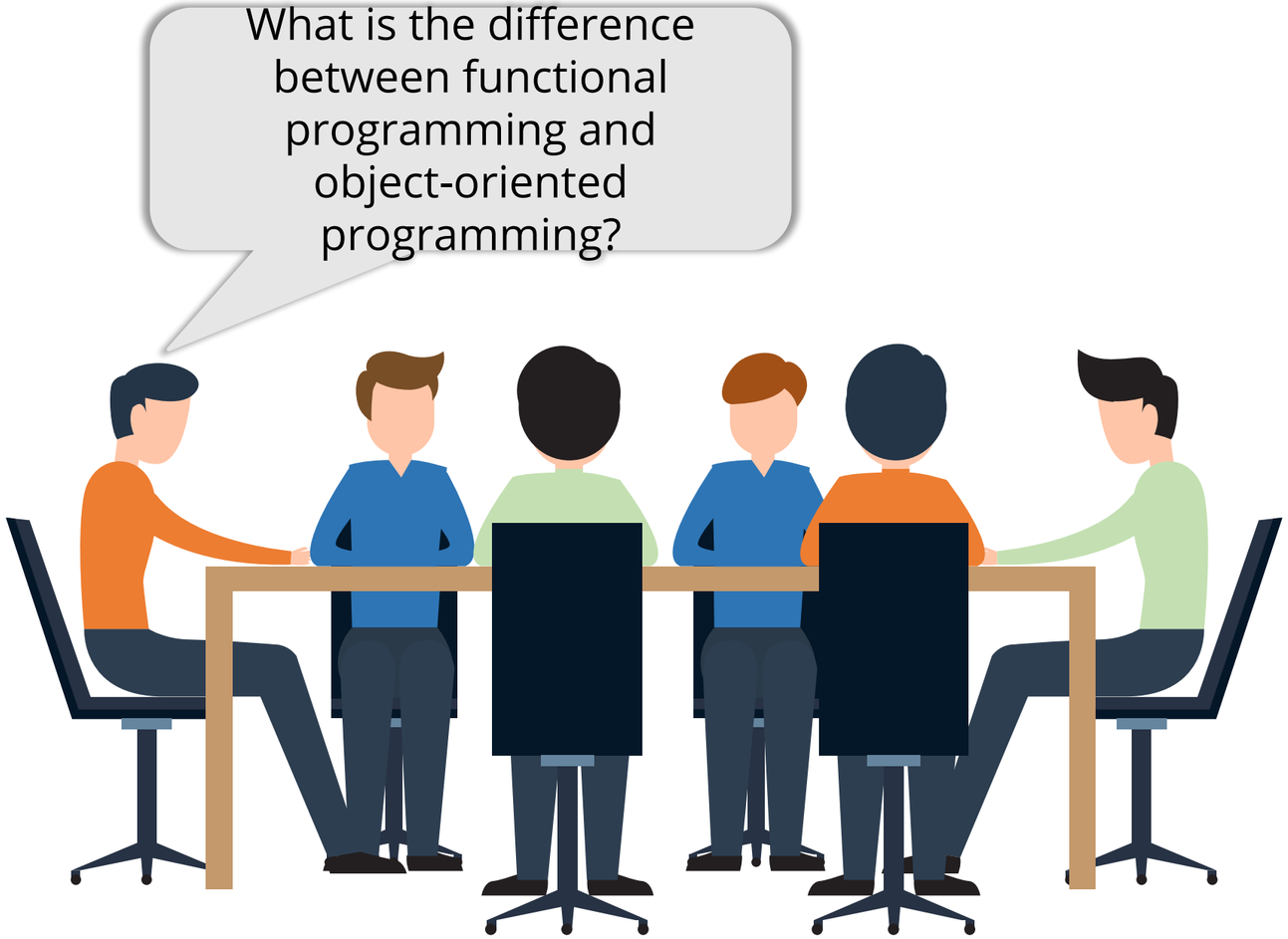
Efficiency



Nested Processing

Lazy Evaluation

Functional Programming



Functional Programming vs. Object-Oriented Programming

Functional Programming	Object-Oriented Programming
Uses immutable data	Uses mutable data
Supports parallel processing	Does not support parallel processing
Flow control is done using function calls and function calls with recursion	Flow control is done using loops and conditional statements
Supports both "Abstraction over Data" and "Abstraction over Behavior"	Supports only "Abstraction over Data"
Follows Declarative Programming Model	Follows Imperative Programming Model

DATA AND
ARTIFICIAL INTELLIGENCE

Programming with Scala

Data Types

Scala supports the same data types as Java does. The table below lists and explains the Scala data types:

Data Type	Explanation
Byte	8-bit signed value; Range: -128 to 127
Short	16-bit signed value; Range: -128 to 127
Int	32-bit signed value; Range: -128 to 127
Long	64-bit signed value; Range: -128 to 127
Float	Single-precision 32-bit IEEE 754 value
Double	Double-precision 64-bit IEEE 754 value
Char	16-bit unsigned Unicode character; Range: U+0000 to U+FFFF
String	Chars sequence
Unit	No value
Null	Empty or null reference
Boolean	Literal true or false
Any	Super type of any type
AnyRef	Super type of any reference type
Nothing	Sub type of every other type; No value contained

Basic Literals

The basic literals used in Scala are listed below with examples:

Integer Literals

Example:
scala> val hex = 0x6; output - hex: Int = 6

Floating Type Literals

Examples:
scala> val big = 1.2345; output - big: Double = 1.2345
scala> val little = 1.2345F; output - little: Float = 1.2345

Character Literals

Example:
scala> val a = 'A'

Basic Literals

Boolean Literals

Examples:
scala> val bool = true; output - bool: Boolean = true
scala> val fool = false; output - fool: Boolean = false

Symbol Literals

Example:
scala> updateRecordByName('favoriteBook, “Spark in Action")

String Literals

Examples:
scala> val hello = "hello“; output - hello: java.lang.String = hello

println("""Welcome to Ultamix 3000 Type "HELP" for help.""")

Welcome to Ultamix 3000
Type "HELP" for help

Introduction to Operators

An operator is a symbol that allows the compiler to perform specific logical or mathematical manipulations. It provides a syntax for general method calls.

Example: 2 + 1 actually implies (2)+(1)

The class Int includes a method called + that takes an Int and provides an Int as a result. To invoke the + method, you need to add two Ints as follows:
scala> val sum = 2 + 1 //Scala invokes (2)+(1)

Types of Operators

Arithmetic Operators

Object Equality Operators

Relational and Logical Operators

Bitwise Operators

Basic Literals and Arithmetic Operators

Duration: 5 mins

Problem Statement: In this demonstration, you will use basic literals and arithmetic operators in Scala.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Logical Operators

Duration: 5 mins

Problem Statement: In this demonstration, you will use the logical operators in Scala.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

DATA AND
ARTIFICIAL INTELLIGENCE

Type Inference, Classes, Objects, and Functions in Scala

Type Inference

Type Inference is a built-in mechanism that allows you to omit certain types of annotations and return types of methods.

Example:

```
object InferenceTest1 extends Application {
val x = 1 + 2 * 3 // the type of x is Int
val y = x.toString() // the type of y is String
def succ(x: Int) = x + 1 // method succ returns Int values
}
```

Objects

An object is a named instance with members like methods and fields.

Scala has singleton objects instead of static members, which is a class with only one instance and can be created using the keyword object.

```
package logging

object Logger {
    def info(message: String): Unit = println(s”INFO: $message”)
}
```

Classes

Classes in Scala are blueprints for creating objects. They can contain methods, values, variables, types, objects, and traits which are collectively called members.

```
Class User

Val user1 = new User
```

Functions

Scala provides a rich set of built-in functions and allows you to create user defined functions also.

In Scala, functions are first class values. These can be returned as a result or passed as a parameter.

```
def sumInts(a: Int, b: Int) : Int = {if (a > b) 0
else
a + sumInts(a + 1, b)}
```

Anonymous Functions

An anonymous function is an alternative of named function definitions that gets created by parameterization by functions.

Example:

```
(y: Int) => y * y
```

Higher-Order Functions

Higher-order functions take other functions as parameters or return a function as a result.

Map is an example of higher-order functions in Scala.

```
val salaries = Seq(20000, 70000, 40000)
val doubleSalary = (x: Int) => x * 2
val newSalaries = salaries.map(doubleSalary) // List(40000, 140000, 80000)
```

Define Type Inference, Functions, Anonymous Functions, and Class

Duration: 10 mins

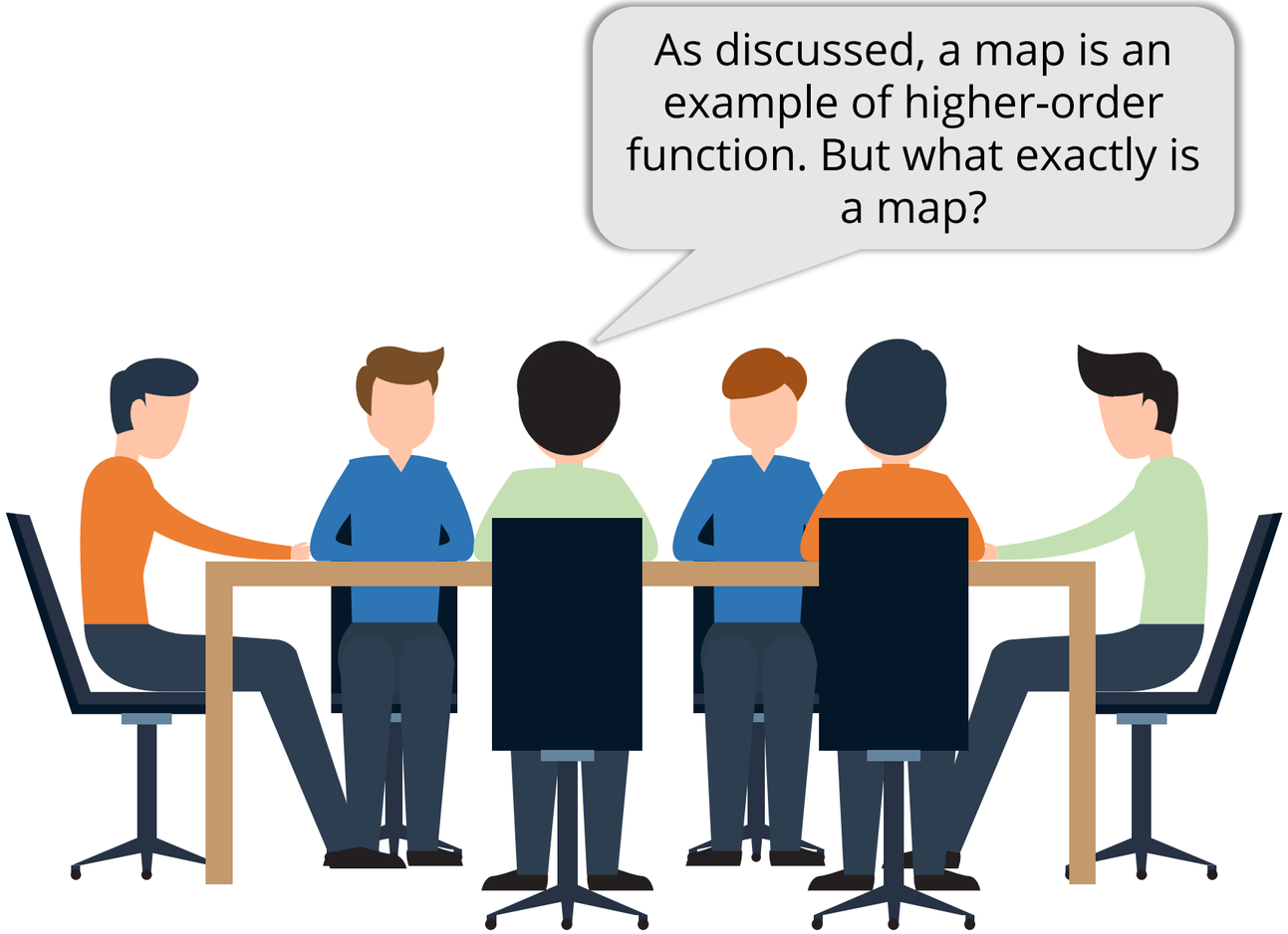
Problem Statement: In this demonstration, you will learn how to define type inference, functions, anonymous functions, and class in Scala.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

DATA AND
ARTIFICIAL INTELLIGENCE

Collections

Collections



Collections

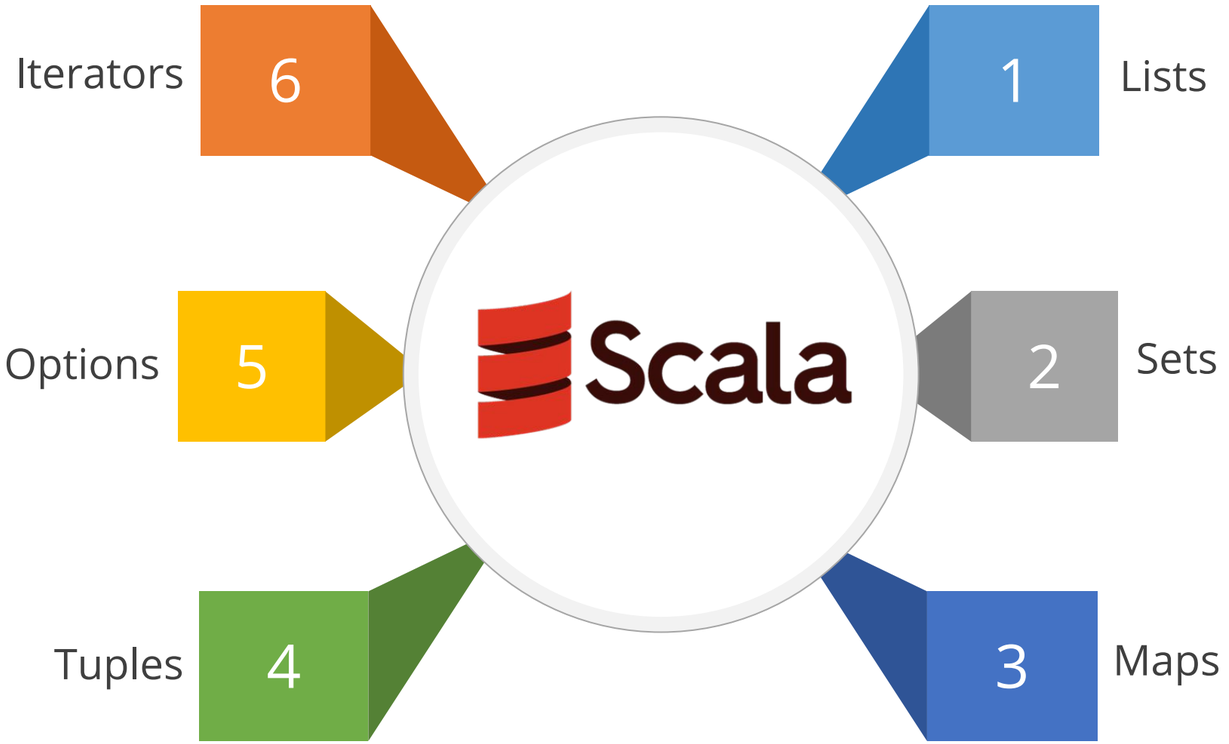
Collections

Collections are containers of things, that can be sequenced as linear sets of items.

Mutable Collections

Immutable Collections

Types of Collections



Types of Collections

DATA AND
ARTIFICIAL INTELLIGENCE

Types of Collections

Lists

In Scala lists, all the elements have the same data type. They are immutable and represent a linked list.

The type of a list that has elements of type T is written as List[T].

```
//List of strings
val fruit: List[String] = List("apples", "oranges", "pears")

//List of integers
val nums: List[Int] = List(1, 2, 3, 4)
```

Sets

Sets are iterables that contain no duplicate elements.

```
//Empty set of integer type
var s : Set[Int] = Set()

//Set of integer type
var s : Set[Int] = Set(1,3,5,7)
```

Sets

Sets have the following fundamental operations:

01 Tests

02 Additions

03 Removals

04 Set Operations

Maps

A map is an iterable consisting of pairs of keys and values.

```
//Empty hash table whose keys are strings and values are integers:
var A:Map[Char,Int] = Map()

//A map with keys and values.
val colors = Map("red" -> "#FF0000", "azure" -> "#F0FFFF")
```

Maps

Maps have the following fundamental operations:

01 Lookup

02 Additions and Updates

03 Removals

04 Subcollection Producers

05 Transformations

Tuples

In Scala, a tuple is a value that contains a fixed number of elements, each with a distinct type.

```
//Tuple of integers
val t = (1,2,3,4)

//Tuple containing integer, string, and a console
val t = (1, “Welcome", Console)
```

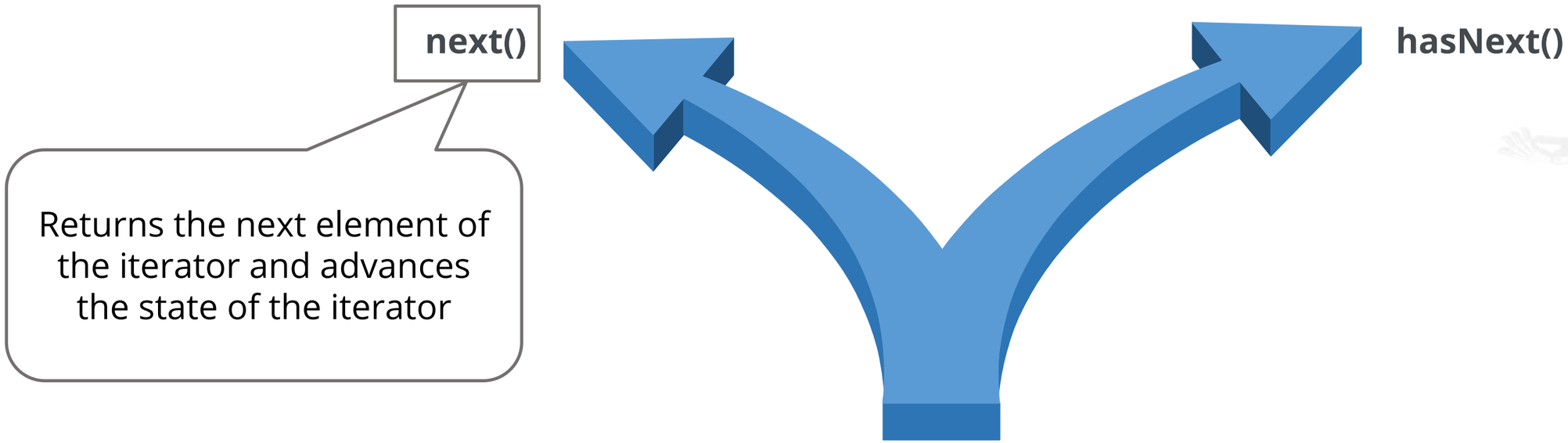
Options

An Option[T] is a container for one element or zero, which represents a missing value.

```
//Define an option
val x:Option[Int] = Some(5)
```

Iterators

An iterator is a way to access the elements of a collection one by one.



Iterators

An iterator is a way to access the elements of a collection one by one.

Demonstrate the Types of Collections

Duration: 10 mins

Problem Statement: In this demonstration, you will learn how to use different types of collections such as List, Set, Map, Tuple, and Option in Scala.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Perform Different Operations on List

Duration: 10 mins

Problem Statement: In this demonstration, you will learn how to perform different operations on list.

Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Scala REPL

Scala REPL

Scala REPL is a tool for evaluating expressions in Scala. The REPL reads expressions promptly, wraps them in an executable template, then compiles and executes the result.

The following are the features of REPL:

- **$intp**
- **//print<tab>**
- **:load**
- **lastException**
- **:help**
- **:paste**

Scala REPL

Few more features of REPL are:

- :paste -raw
- -Yrepl-outdir
- :settings
- :javap
- :power
- :replay

Implementation Notes

1. User code can be wrapped in either an object or a class.
   The switch is -Yrepl-class-based.
2. Every line is compiled individually.
3. The automatically generated imports include the dependencies on previous lines.
4. Implicit import of scala.Predef can be controlled by inputting an explicit import.

Example of Scala REPL

```
scala> import Predef.{any2stringadd => _, _}
import Predef.{any2stringadd=>_, _}
scala> new Object + "a string"
<console>:13: error: value + is not a member of Object
       new Object + "a string"
                  ^
scala> import Predef._ import Predef._
scala> new Object + "a string"
res1: String = java.lang.Object@787a0fd6a string
```

Demonstrate the Features of Scala REPL

Duration: 10 mins

Problem Statement: In this demonstration, you will demonstrate the features of Scala REPL.

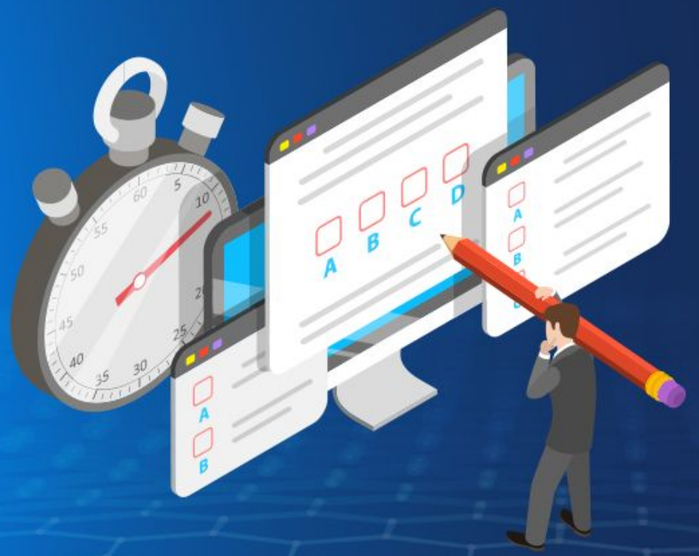
Access: Click on the **Practice Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Key Takeaways

You are now able to:

- Describe and demonstrate programming with Scala
- Define functions, anonymous function, and class in Scala
- Use the different types of collections
- Describe and demonstrate Scala REPL

DATA AND
ARTIFICIAL INTELLIGENCE



Knowledge Check

Knowledge Check 1

scala> val hex = 0x6; output - hex: Int = 6 is an example of which of the following literals?

a. Character literal

b. Integer literal

c. Floating literal

d. None of the above

Knowledge Check 1	**scala> val hex = 0x6; output - hex: Int = 6 is an example of which of the following literals?**

a. Character literal

b. Integer literal

c. Floating literal

d. None of the above

The correct answer is **b**

scala> val hex = 0x6; output - hex: Int = 6 is an example of integer literal.

Knowledge Check 2	_____ is a value that contains a fixed number of elements, each with a distinct type.

a. Map

b. Tuple

c. List

d. Set

Knowledge Check 2	_____ is a value that contains a fixed number of elements, each with a distinct type.

a. Map

b. Tuple

c. List

d. Set

The correct answer is **b**

Tuple is a value that contains a fixed number of elements, each with a distinct type.

Knowledge Check 3

_____ is a way to access the elements of a collection one by one.

a. Option

b. Iterator

c. Map

d. Set

Knowledge Check 3

_____ is a way to access the elements of a collection one by one.

a. Option

b. Iterator

c. Map

d. Set

The correct answer is **b**

Iterator is a way to access the elements of a collection one by one.

Lesson-End Project

Problem Statement: Scala is one of the most popular languages to work with Spark. One of the advantages of Scala is that it runs on JVM. It supports both object-oriented and functional programming. Mostly Scala is used for functional programming. It provides java-like features such as collections, file handling & exception handling, etc.
“People data Labs” is one of the biggest data organizations that collects and provides data for the companies. It has open-sourced data about companies that operate in different countries. You must find some important facts about companies based on the year they were founded in, their employee count and country, etc.

Lesson-End Project

The following details are given for the list of companies in a CSV file.

1. Name
2. Domain
3. Year founded
4. Industry
5. Size range
6. Country
7. LinkedIn URL
8. Current employee estimate
9. Total employee estimate

Filename: companies.csv

You must read the file and use Scala collections to solve the following problems:

1. Companies which were founded before 1980.
2. Top 5 companies which have the maximum number of employees.
3. Top Industry in which the maximum number of companies were founded.

DATA AND
ARTIFICIAL INTELLIGENCE