Scala Viva Questions

Basic:

1. **What is Scala?** Scala is a hybrid functional and object-oriented programming language.
2. **How do you declare a variable in Scala?** Using val for immutable variables and var for mutable variables.
3. **What is the difference between val and var in Scala?** val creates an immutable variable, while var creates a mutable variable.
4. **How do you define a function in Scala?** Using the def keyword followed by the function name and parameters.
5. **What is a case class in Scala?** A special type of class used for immutable data structures with boilerplate code generated automatically.
6. **What is an object in Scala?** A singleton instance of a class, created using the object keyword.
7. **What is a trait in Scala?** A trait is a collection of methods and fields that can be mixed into classes.
8. **How do you extend a class and mix in a trait in Scala?** Using the extends keyword for the class and with keyword for the trait.
9. **What is the purpose of companion objects in Scala?** To hold static members and methods for a class and to provide a factory for creating instances of the class.
10. **What is a sealed trait in Scala?** A trait that can only be extended in the same file, ensuring all subclasses are known at compile time.
11. **How is Scala a programming language with a combination of both functional and object-oriented programming?**

    Scala programming language treats every single value as an object, which also includes Functions. This way, it is a combination of both functional and object-oriented programming

12. **What are the frameworks supported by Scala?**

There are various frameworks supported by Scala that include the following.

- Spark Framework
- Play Framework
- Akka Framework
- Neo4j Framework
- Bowler Framework
- Scalding Framework
- Lift Framework

13. **What are the different kinds of variables in Scala?** There are mainly two types of variables in Scala: Mutable variables and Immutable variables.
14. **Define the features of Mutable Variables.** Mutable variables can be declared using the var keyword. The values in these variables support changes.
15. **Define the features of Immutable Variables.** Immutable variables can be declared using the val keyword. The values in these variables do not support changes.
16. **Define Stream in Scala.** A stream is defined as a lazy list, which helps in the evaluation of the elements only when they are needed.
17. **What is the benefit of Streams in Scala?** The benefit of Streams in Scala is that it helps in enhancing the performance of the program.
18. **What are the advantages of Scala?** The advantages of Scala include being scalable, maintainable, productive, supporting concurrent programming, having native tuples and testable codes, concise code, no boilerplate code, and clearer singleton objects than static methods.
19. **What are the different operators in Scala?** The different operators in Scala include:
    - Assignment operators
    - Relational operators
    - Logical operators
    - Arithmetic operators
    - Bitwise operators
20. **What is Recursion in Scala?** Recursion is referred to as a function in Scala that calls itself.

21. **Give an example of Recursion in Scala.** When Function A calls Function B, which further calls Function C, it is called recursion in Scala and is mostly used in functional programming.
22. **What is Tail Recursive?** Tail recursive is a call back to the function that should be the end task function to be performed.
23. **What are Tuples in Scala?** Tuples in Scala combine a finite number of items together so that the programmer can pass the tuple around as a whole.
24. **What is the val keyword used for in Scala?** The val keyword is used to declare immutable variables.
25. **What is the var keyword used for in Scala?** The var keyword is used to declare mutable variables.
26. **What is an Option in Scala?** An Option is a container that may or may not hold a value, represented as Some(value) or None.
27. **How do you handle null values in Scala?** By using the Option type to avoid null references and safely handle optional values.
28. **What is a higher-order function in Scala?** A higher-order function is a function that takes other functions as parameters or returns a function as a result.
29. **How do you define a main method in Scala?** Using def main(args: Array[String]): Unit = { ... }.
30. **What is a case class in Scala?** A case class is a special type of class used for immutable data structures with boilerplate code automatically generated.
31. **What is a companion object in Scala?** A companion object is an object with the same name as a class that contains static members and methods.
32. **What is lazy evaluation in Scala?** Lazy evaluation is a technique where an expression is not evaluated until its value is needed, using the lazy keyword.

Experiment 1: Basic Arithmetic Operations

1. **How do you define a function in Scala?** Using the def keyword followed by the function name and parameters.
2. **What is the return type of your function for computing the sum of two integers?** Int.
3. **Explain the logic used to triple the sum if the two integers are the same.** Check if the integers are equal, then return 3 * (a + b).
4. **How do you handle integer inputs in Scala?** Use Int type for variables and parameters.
5. **How would you modify the program to handle floating-point numbers?** Change the parameter types to Double.

Experiment 2: Conditional Statements

6. **What Scala constructs are used to check conditions in your program?** if-else statements.
7. **How do you implement an if-else condition in Scala?** Use if (condition) { ... } else { ... }.
8. **What is the purpose of the program checking if one integer is 22 or their sum is 32?** To return true if either condition is met.
9. **How do you return a boolean value in Scala?** By using conditions that evaluate to true or false.
10. **Can you use pattern matching for this condition check?** Yes, but if-else is more straightforward for simple conditions.

Experiment 3: String Manipulation

11. **How do you remove a character from a string at a specific position?** Use string.take(pos) + string.drop(pos + 1).
12. **How do you access characters in a string in Scala?** Using string.charAt(index).
13. **What is the purpose of taking the first 5 characters of a string?** To create a new string by adding these characters at both the front and back.
14. **How do you concatenate strings in Scala?** Using the + operator or concat method.

15. **Can you modify a string directly in Scala?** No, strings are immutable in Scala.

Experiment 4: Control Structures

16. **How do you create a for loop to print a multiplication table?** Using for (i <- 1 to 10) { println(number * i) }.
17. **What is pattern matching in Scala?** A mechanism for checking a value against a pattern.
18. **How do you find the largest element in an array using pattern matching?** Use a recursive function with match cases.
19. **How do you declare an array in Scala?** Using Array(element1, element2, ...).
20. **What is the advantage of using pattern matching?** It provides a clear and concise way to handle different cases.

Experiment 5: Functions and Recursion

21. **How do you define a recursive function in Scala?** Using the def keyword and calling the function within its body.
22. **How do you calculate the product of digits in a number?** By recursively multiplying the digits.
23. **How do you check if a number is a perfect square in Scala?** By checking if the square root of the number is an integer.
24. **What library function can you use to find the square root?** math.sqrt.
25. **How do you handle large numbers in Scala?** Using BigInt for arbitrarily large integers.

Experiment 6: Classes and Inheritance

26. **How do you create a subclass in Scala?** Using the extends keyword.
27. **What is the purpose of the Student class extending Person?** To inherit properties and methods from the Person class.
28. **How do you define a class with properties in Scala?** Using the class keyword with constructor parameters.

29. **What is an enum in Scala and how do you create one?** An enum is a special class representing a group of constants, created using enum.
30. **How do you represent an object's color using an enum class?** Define a Color enum with color values and use it as a property in the class.

Experiment 7: Sets

1. **How do you create a set in Scala?** Using Set(element1, element2, ...).
2. **How do you find the difference between two sets?** Using the diff method: set1.diff(set2).
3. **How do you find the intersection between two sets?** Using the intersect method: set1.intersect(set2).
4. **How do you find the second largest element in a set?** Convert the set to a list, sort it, and access the second last element.
5. **Are sets mutable or immutable by default in Scala?** Immutable.

Experiment 8: Lists

6. **How do you create a list in Scala using Lisp style?** Using List(element1, element2, ...).
7. **How do you create a list in Scala using Java style?** Using new ListBuffer[elementType].
8. **What is a Range list in Scala and how do you create it?** A sequence of numbers; created using List.range(start, end).
9. **How do you create a uniform list in Scala?** Using List.fill(size)(value).
10. **How do you create a tabulate list in Scala?** Using List.tabulate(size)(function).
11. **How do you flatten a nested list structure?** Using flatten method: list.flatten.
12. **What is the benefit of using a tabulate list?** It generates a list based on a function, allowing for dynamic creation.

Experiment 9: Lists Manipulation

13. **How do you add each element n times to a given list of integers?** Use a for loop or flatMap with List.fill(n)(element).

14. **How do you split a list into two lists in Scala?** Using splitAt(index).
15. **What does List.fill do?** It creates a list with the given value repeated n times.
16. **How do you iterate over a list in Scala?** Using for loops or foreach method.
17. **How do you convert a list of lists into a single list?** Using flatten method.
18. **What is the purpose of splitting a list?** To divide the list into two parts for separate processing.

Experiment 10: Tuples

19. **How do you swap elements in a tuple in Scala?** Create a new tuple with the elements swapped: (tuple._2, tuple._1).
20. **What should the program do if the elements are the same?** Print "no swapping required".
21. **How do you define a tuple in Scala?** Using parentheses with elements separated by commas: (element1, element2).
22. **How do you access elements in a tuple?** Using _1, _2, etc.
23. **How do you find non-unique elements in a tuple?** Convert the tuple to a list, use groupBy and filter groups with size > 1.
24. **Can tuples hold elements of different types?** Yes, tuples can hold elements of different types.
25. **What is the use of tuples in Scala?** To group multiple values without creating a custom class.
26. **How do you convert a tuple to a list?** Using tuple.productIterator.toList.
27. **Can tuples be nested in Scala?** Yes, tuples can contain other tuples.
28. **How do you check if a tuple contains a specific element?** Using tuple.productIterator.contains(element).
29. **What is the maximum number of elements a tuple can hold in Scala?** 22 elements.
30. **What method would you use to iterate through tuple elements?** Using productIterator method.

Basic Scala Concepts

1. **What is Scala, and how is it different from Java?**

   - Scala is a hybrid functional and object-oriented programming language that runs on the JVM. Unlike Java, it supports concise syntax, higher-order functions, immutable collections, and a strong type inference system.

2. **Is Scala a statically or dynamically typed language?**

   - Scala is **statically typed**, meaning type checking is done at compile-time.

3. **What are the advantages of using Scala?**

   - Concise and expressive syntax
   - Supports functional and object-oriented programming
   - Type inference reduces boilerplate code
   - Immutable collections for safer concurrency
   - Interoperability with Java

4. **Explain the difference between `var`, `val`, and `lazy val`.**

   - `var` → Mutable variable (can be reassigned).
   - `val` → Immutable variable (cannot be reassigned).
   - `lazy val` → Evaluated only when accessed for the first time.

5. **What is the difference between mutable and immutable collections in Scala?**

   - Mutable collections allow modification after creation.
   - Immutable collections do not allow modifications, requiring new instances for changes.

6. **What is the default visibility of a class in Scala?**

   - By default, classes and members in Scala have **public** access.

7. **What is the use of the `apply` method in Scala?**

   - The `apply` method enables calling an object like a function.

```
class Demo { def apply() = "Hello from apply" }
val obj = new Demo()
println(obj()) // Calls apply method
```

8. **What is a companion object in Scala?**
   - A singleton object with the **same name as a class** that allows defining static-like members.

```
class MyClass(val x: Int)
object MyClass { def apply(x: Int) = new MyClass(x) }
val obj = MyClass(10) // Calls the apply method
```

9. **What is the difference between `object` and `class` in Scala?**
   - `object` → Singleton instance (one per JVM).
   - `class` → Blueprint for creating multiple instances.
10. **What are case classes in Scala? Why are they useful?**
    - Case classes automatically provide:
      - `equals`, `hashCode`, `toString`
      - Immutable by default
      - Pattern matching support

```
case class Person(name: String, age: Int)
val p1 = Person("Alice", 25)
```

Functional Programming Concepts

11. **What is a higher-order function in Scala?**
    - A function that takes another function as an argument or returns a function.

```
def operate(x: Int, y: Int, f: (Int, Int) => Int): Int
= f(x, y)
val sum = operate(4, 5, _ + _)
```

    -

12. **What is the difference between `map`, `flatMap`, and `foreach`?**
    - `map` → Transforms each element and returns a collection.
    - `flatMap` → Flattens nested collections.
    - `foreach` → Executes code for each element but returns `Unit`.
13. **What are anonymous functions (lambda expressions) in Scala?**
    - Functions without names.

```
val square = (x: Int) => x * x
println(square(5)) // 25
```

    -

14. **What is a partially applied function in Scala?**
    - A function where some arguments are fixed.

```
def multiply(a: Int, b: Int) = a * b
val double = multiply(2, _: Int)
```

    -

15. **What is the difference between a function and a method in Scala?**
    - Method: Defined inside a class.
    - Function: A value assigned to a variable.
16. **What is a closure in Scala?**
    - A function that captures variables from its surrounding scope.
17. **What is currying in Scala?**
    - A function that takes multiple parameters one at a time.

```
def add(a: Int)(b: Int) = a + b
val addFive = add(5) _
println(addFive(10)) // 15
```

    -

Object-Oriented Features

18. **How does Scala support both Object-Oriented and Functional programming?**
    - It has classes and objects (OOP) while supporting higher-order functions and immutability (FP).
19. **How is inheritance implemented in Scala?**
    - Using `extends` for classes and `with` for traits.
20. **What is the significance of the `override` keyword in Scala?**
    - Required when overriding methods in subclasses.
21. **What is the difference between `extends` and `with` in Scala?**

    - `extends` → Extends a class.
    - `with` → Mixes in a trait.

22. **How do traits work in Scala?**
    - Traits are like interfaces but can have concrete methods.

23. **Can a Scala class extend multiple traits?**
    - Yes, using multiple `with` keywords.

Collections and Data Handling

24. **What is the difference between `List`, `Array`, and `Vector` in Scala?**
    - `List` → Immutable, linked-list.
    - `Array` → Mutable, fixed-size.
    - `Vector` → Immutable, fast indexing.

25. **What are the main types of Scala collections?**
    - `List`, `Array`, `Vector`, `Set`, `Map`, `Queue`.

26. **What is the difference between an immutable and a mutable `Map`?**
    - Immutable `Map` cannot be changed after creation, whereas mutable `Map` allows modifications.

Error Handling

29. **How does Scala handle exceptions?**
    - Using `try`, `catch`, and `finally`.

30. **What is the difference between `Try`, `Success`, `Failure`, and `Option`?**

    - `Try` is for handling exceptions.
    - `Success(value)` stores a successful computation.
    - `Failure(exception)` stores an error.
    - `Option` is for handling missing values (`Some` or `None`).

Pattern Matching & Advanced Topics

32. **What is pattern matching in Scala?**

    - A powerful switch-case alternative.

```
val num = 2
num match {
  case 1 => println("One")
  case 2 => println("Two")
  case _ => println("Other")
}
```

    -

33. **What are extractors in Scala?**

    - Used in pattern matching to extract values from objects.

Concurrency in Scala

39. **What is Akka in Scala?**

    - A toolkit for building distributed and concurrent applications.

40. **What is the difference between `Future` and `Promise` in Scala?**

- `Future` represents a computation that may complete in the future.
- `Promise` is a writable, single-assignment container for a future.