1. What are scala sets

Scala set is a pairwise collection of elements of same type. It does not contain any duplicate. They can be mutable or immutable.

1. What are scala maps?

Scala maps are like python dictionaries that store data in key value pairs, key has to be unique, value might not be.

1. What are some advantages of scala?

Scala lets ‘if blocks’, ‘for-yield loops’, and ‘code’ in braces to return a value. It is more preferable, and eliminates the need for a separate ternary operator

Singleton has singleton objects rather than C++/Java/ C# classic static. It is a cleaner solution

Persistent immutable collections are the default and built into the standard library.

1. What are the Scala variables?

Values and variables are two shapes that come in Scala. A value variable is constant and cannot be changed once assigned. It is immutable, while a regular variable, on the other hand, is mutable, and you can change the value.

The two types of variables are

var myVar : Int=0;

val myVal: Int=1;

1. What are options in scala?

It is a collection that has only two type Some[T] or None.

It is used to wrap the missing values.

It has several methods like get, isEmpty, getOrElse which can be used when the return value is none.

object Demo {

def main(args: Array[String]) {

val capitals = Map("France" -> "Paris", "Japan" -> "Tokyo")

println("show(capitals.get( \"Japan\")) : " + show(capitals.get( "Japan")) )

println("show(capitals.get( \"India\")) : " + show(capitals.get( "India")) )

}

def show(x: Option[String]) = x match {

case Some(s) => s

case None => "?"

}

}

1. Tuples in scala?

Scala tuples combine a fixed number of items together so that they can be passed around as whole.

A tuple is immutable and can hold objects with different types, unlike an array or list.

1. What is a recursion tail?

A recursive function is tail recursive when a recursive call is the last thing executed by the function.

1. What is the use of tail recursive function?

The tail recursive functions considered better than non-tail recursive functions as tail-recursion can be optimized by the compiler. Compilers usually execute recursive procedures by using a stack. This stack consists of all the pertinent information, including the parameter values, for each recursive call. When a procedure is called, its information is pushed onto a stack, and when the function terminates the information is popped out of the stack. Thus for the non-tail-recursive functions, the stack depth (maximum amount of stack space used at any time during compilation) is more. The idea used by compilers to optimize tail-recursive functions is simple, since the recursive call is the last statement, there is nothing left to do in the current function, so saving the current function’s stack frame is of no use

1. What is ‘scala trait’ in scala?

It is like a class and encapsulates methods and field definitions, but unlike classes, a class can inherit multiple traits.

Traits can be partially implemented, but cannot have constructor parameters.

1. What are universal traits?

Universal traits are traits that extend Any. These traits are the only ones that can be extended by value classes.

Value classes are used in scala to avoid allocating runtime objects. It contains primary constructor with exactly 1 Val parameter.

Value classes cannot be extended by any other classes.

Though, you can extend your value class with AnyVal as below

trait Printable extends Any {

def print(): Unit = println(this)

}

class Wrapper(val underlying: Int) extends AnyVal with Printable

object Demo {

def main(args: Array[String]) {

val w = new Wrapper(3)

w.print() // actually requires instantiating a Wrapper instance

}

}

1. When can you use traits?

There is no specific rule when you can use traits, but there is a guideline which you can consider.

If the behaviour will not be reused, then make it a concrete class. Anyhow it is not a reusable behaviour.

In order to inherit from it in Java code, an abstract class can be used.

If efficiency is a priority then lean towards using a class

Make it a trait if it might be reused in multiple and unrelated classes. In different parts of the class hierarchy only traits can be mixed into different parts.

You can use abstract class, if you want to distribute it in compiled form and expects outside groups to write classes inheriting from it.

1. What is trait mixins?

Traits and classes when extended, follow an order of extension. When any class is extended alongwith trait, extend class first and all traits will be extended after this class like below:

class A6 extends PrintA4 with Print{ // PrintA4 is abstract class Print is trait

1. What is a case class?

Case class is a constructor less definition of an immutable class, that can be used for pattern matching.

A parameter-less case class becomes a case object.

1. What is closure?

A closure is a function whose return value depends on the value of the variables declared outside the function.

val multiplier = (i:Int) => i \* factor

Here, factor is an outside variable and the return value is determined by the type of this variable.

1. What is Monad in Scala?

Monad is a concept in scala, where one object covers another object.

Where one computation acts as an input to another computation.

All datatypes that implement map and flatmap are monads.

1. What are anonymous functions in scala?

Anonymous functions in source code are called function literals and at run time, function literals are instantiated into objects called function values.

Example : var mul = (x: Int, y: Int) => x\*y

It does not require def or return keywords.

1. What are higher order functions in scala?

Functions that use other functions as parameters or whose result is a function are called higher order functions.

object Test {

def main(args: Array[String]) {

println( apply( layout, 10) )

}

def apply(f: Int => String, v: Int) = f(v)

def layout[A](x: A) = "[" + x.toString() + "]"

1. What is functions call-by-name?

When you need a function that accepts as a parameter an expression that we don’t want to be evaluated until its called within our function, you use functions call by name as follows:

object Demo {

def main(args: Array[String]) {

delayed(time());

}

def time() = {

println("Getting time in nano seconds")

System.nanoTime

}

def delayed( t: => Long ) = {

println("In delayed method")

println("Param: " + t)

}

}

Output:

In delayed method

Getting time in nano seconds

Param: 2027245119786400

1. Functions with named arguments?

Much like functions with keyword arguments, where order of arguments doesn’t matter, you pass the parameters by their names as follows:

object Demo {

def main(args: Array[String]) {

printInt(b = 5, a = 7);

}

def printInt( a:Int, b:Int ) = {

println("Value of a : " + a );

println("Value of b : " + b );

}

}

1. Partially applied functions?

When you have a requirement to call a function with some fixed parameters and some changing, you can make use of these functions.

We bind the variables that are fixed with the variable value and the changing variables with an underscore as shown below.

import java.util.Date

object Demo {

def main(args: Array[String]) {

val date = new Date

val logWithDateBound = log(date, \_ : String)

logWithDateBound("message1" )

Thread.sleep(1000)

logWithDateBound("message2" )

Thread.sleep(1000)

logWithDateBound("message3" )

}

def log(date: Date, message: String) = {

println(date + "----" + message)

}

}

1. What is function currying in scala?

Currying transforms a function that takes multiple parameters into a chain of functions, each taking a single parameter. Curried functions are defined with multiple parameter lists, as follows

def strcat(s1: String)(s2: String) = s1 + s2

or

def strcat(s1: String) = (s2: String) => s1 + s2

It is called as below:

strcat("foo")("bar")

1. How do you append to a list in scala?

var myList = List.empty[String]

myList :+= "a"

myList :+= "b"

myList :+= "c"

use++ for appending a list

var myList = List.empty[String]

myList ++= List("a", "b", "c")

1. How can you format a string?

To format a string, use the .format () method, in scala you can use

Val formatted= “%s %i”.format (mystring.myInt)

1. What is pattern matching in scala?

Use of case just like a switch statement is called pattern matching. You can use Any datatype as input and output to match with multiple datatypes in single case block.

object Demo {

def main(args: Array[String]) {

println(matchTest("two"))

println(matchTest("test"))

println(matchTest(1))

}

def matchTest(x: Any): Any = x match {

case 1 => "one"

case "two" => 2

case y: Int => "scala.Int"

case \_ => "many"

}

}

1. Exceptions in scala.

Handled with try->catch->finally with case in catch for matching exception type.

1. Why do we need App in Scala?

App is a helper class that holds the main method and its Members together. The App trait can be used to quickly turn Objects into Executable programs. We can have our classes extend App to render the executable code.

object Edureka extends App{

println("Hello World")

}

1. Explain the difference between Null, None, Nil and Nothing

Null: Null represents the absence of a value.

None: None is the value of an Option with no value.

Nil: Nil denotes the end a List

Nothing: Nothing is lowest type in type System.

1. Any 5 string methods

Split, concat, length, to\_chararray, trim, toUpperCase

1. Different ways to create an array

Use ofDim to create multi dimensional array

var myMatrix = ofDim[Int](3,3)

and range method:

range (10, 20, 2)

1. What is an Auxiliary constructor?

We use Auxiliary constructor in Scala for Constructor Overloading. The Auxiliary Constructor must call either previously defined auxiliary constructors or primary constructor in the first line of its body.

1. Explain Que with example

Queue is a Data Structure similar to Stack except, it follows First In First Out procedure for data processing. In Scala, to work with Queues, you need to import a library called,

import scala.collection.mutable.Queue