**Lambda expressions** use the operator symbol =, which reads as "goes to." Input parameters are specified on the operator's left side, and statement/expressions are specified on the right. Generally, lambda expressions are not directly used in query syntax but are often used in method calls. Query expressions may contain method calls.  
  
Lambda expression syntax features are as follows:

* It is a function without a name.
* There are no modifiers, such as overloads and overrides.
* The body of the function should contain an expression, rather than a statement.
* May contain a call to a function procedure but cannot contain a call to a subprocedure.
* The return statement does not exist.
* The value returned by the function is only the value of the expression contained in the function body.
* The End function statement does not exist.
* The parameters must have specified data types or be inferred.
* Does not allow generic parameters.
* Does not allow optional and ParamArray parameters.

**Q) Why do we need functional interfaces?**

The main reason why we need functional interfaces is that**we can use them in a lambda expression and method references**. This way we reduce boilerplate code. You don’t have to include the abstract keyword, because by default a method declared inside a functional interface is abstract.

**Q) Difference between lambda expression and method reference**

**🡪 Lambda expression** is an anonymous method (method without a name) that has used to provide the **inline implementation**of a method defined by the**functional interface** while a**method reference**is similar to a **lambda expression** that refers a method without executing it. The arrow **(->)** operator can be used to connect the argument and functionality in a lambda expression while the **(::)** operator separates the method name from the name of an object or class in a method reference.

Syntax for Lambda Expression

([comma seperated argument-list]) -> {body}

Syntax for Method Reference

<classname> :: <methodname>

Example

import java.util.\*;

public class LambdaMethodReferenceTest {

   public static void main(String args[]) {

      List<String> myList = Arrays.asList("INDIA", "AUSTRALIA", "ENGLAND", "NEWZEALAND", "SCOTLAND");

      System.out.println("------- Lambda Expression --------");

      // Using Lambda function to call system.out.println()

      myList.stream().map(s -> s.toUpperCase())

                     .forEach(s -> System.out.println(s));

      System.out.println("------- Method Reference ---------");

      // Using Method reference to call system.out.println()

      myList.stream().map(String :: toUpperCase).sorted()

                     .forEach(System.out :: println);

   }

}

Java provides a new feature called **method reference** in Java 8. Method reference is used to refer method of functional interface. It is compact and easy form of lambda expression. Each time when you are using lambda expression to just referring a method, you can replace your lambda expression with method reference

Types of method references in java:

1. Reference to a static method.
2. Reference to an instance method.
3. Reference to a constructor.