Functional Programming

- Functional programming is a style of writing computer programs that treat computations as evaluating mathematical functions**.**
- In other words, it is a declarative style of programming rather than imperative. The basic objective of this style of programming is to make code more concise, less complex, more predictable, and easier to test compared to the legacy style of coding.
- Functional programming deals with certain key concepts such as pure function, immutable state, assignment-less programming etc.
- It is designed to handle the symbolic computation and application processing list, and it is based on mathematical work. The most popular functional programming languages are Python, Lisp, Haskell, Clojure, Erlang etc.
- In simple terms, functional programming is all about building functions for immutable variables. In contrast, object-oriented programming is about having a relatively fixed set of functions. Because of its nature, functional programming is great for in-demand tasks such as data analysis and machine learning.

- Functional programming does not have any state, so all the time, there is a need to create new objects to perform the actions. That's why functional programming needs a large memory space. This programming is used to perform the different operations on the same data set.

Let's understand a few concepts in **functional programming**:

- **Higher-order functions:** In functional programming, functions are to be considered as first-class citizens. That is, so far in the legacy style of coding, we can do below stuff with objects.
  1. We can pass **objects** to a function.
  2. We can create **objects** within function.
  3. We can return **objects** from a function.
  4. We can pass a **function** to a function.
  5. We can create a **function** within function.

6. We can return a **function** from a function.

- **Pure functions:** A function is called pure function if it always returns the same result for same argument values and it has no side effects like modifying an argument (or global variable) or outputting something.
- **Lambda expressions:** A Lambda expression is an anonymous method that has mutability at very minimum and it has only a parameter list and a body. The return type is always inferred based on the context. Also, make a note, Lambda expressions work in parallel with the functional interface. The syntax of a lambda expression is:

  (parameter) -> body

- In its simple form, a lambda could be represented as a comma-separated list of parameters, the –> symbol and the body.

Functional Programming has two types; those are shown as below:

- **Pure Functional Languages:** Pure functional language supports only the functional pattern. An example of the pure functional language is Haskell.
- **Impure Functional Language:** Impure Functional language supports the prototype of functions and the programming's imperative style. An example of an impure functional language is LISP.

Characteristics of the Functional Programming Language

Characteristics of the functional programming languages are like as shown below:

- Functional programming languages are designed to perform the functions of mathematical functions. These functions use conditional expressions and recursion to perform the computation.
- Functional programming supports functions in **higher-order** and features of **lazy evaluation**.

- Functional Programming language directly uses the functions and function calls. It does not support the flow of the controls like statements of the loop, and statements are like the conditional statements such as If-Else and Switch Statements.
- Object-Oriented Programming supports the Abstraction, Encapsulation, and Polymorphism, just like functional programming languages support OOPS concepts.

Advantages of the Functional Programming

Advantages of the functional programming languages are as shown below:

- **Bugs-Free code:** Functional Programming language does not support **state**, so there is no side effect of the functional programming; hence we can write the error-free code.
- **Efficient Programming Language:** Functional Programming language has no mutable state, so there is no state change issue. We can do the program "Functions" to work parallel to "Instruction". This type of code supports reusability and testability easily.
- **Efficiency**- Functional Program contains the Independent Units. Independent units run concurrently. Hence these functional programs are more efficient.
- **Supports Nested Functions**- Nested functions are supported by functional programming.
- **Lazy Evaluation**- Lazy Functional Constructions are also supported by functional programming such as Lazy Lists, Lazy Maps, etc.

Efficiency of Program

The program's code is directly proportional to the efficiency of the algorithm and the execution speed of the program. If the efficiency is good, that means the performance will be high in the program.

The efficiency of the program is affected by the below factors:

- The machine's speed affects the efficiency of the program.
- The compiler's speed also affects the efficiency of the program.

- The operating system also plays a crucial role in the efficiency of the programming code.
- The choice of the right Programming language affects the efficiency of the programming.
- Data organization is also affecting the efficiency of the program.
- The use of the algorithm in the program affects the efficiency of the programs. An algorithm in the Functional Programming solves the problem.

We can increase the efficiency of the programming language with the help of the below tasks -

- To increase the program's efficiency, we have to remove the program's unusable code or the code that is having redundant processing.
- The use of optimal memory and non-volatile storage helps to improve the efficiency of the programming language.
- We can reuse the components. This will also help to increase the efficiency of the program.
- By using the error & exception handling on all the layers of the program.
- During the program's coding, the program should have to be ensured about the data's integrity and consistency.
- By using the programming code, we can do design logic and flow.

Efficient programming code can reduce the consumption of the resources and time taken by the completion programs.

How to Implement Functional Programming in Java?

```
// Java program to demonstrate
// anonymous method
import java.util.Arrays;
import java.util.List;
public class GFG {
    public static void main(String[] args)
    {
```

```
        // Defining an anonymous method
        Runnable r = new Runnable() {
            public void run()
            {
                System.out.println(
                    "Running in Runnable thread");
            }
        };

        r.run();
        System.out.println(
            "Running in main thread");
    }
}
```

Output:

Running in Runnable thread

Running in main thread