# **Functional Programming**

Functional programming (also called FP) is a way of thinking about software development by creating pure functions. It avoid concepts of shared state, mutable data observed in Object Oriented Programming.

Functional languages emphasizes on expressions and declarations rather than execution of statements. Therefore, unlike other procedures which depend on a local or global state, value output in functional programming depends only on the arguments passed to the function.

### **Characteristics of Functional Programming**

* The focus point of Functional programming method is on results, not the process
* Emphasis is on what is to be computed
* Data is immutable
* Functional programming breaks down problems into functions
* It is built on the concept of mathematical functions which uses conditional expressions and recursion to do perform the calculation
* It does not support iteration like loop statements and conditional statements like If-Else

**Functional Programming Terminology and Concepts**

#### **Immutable Data**

Immutable Data means that you should easily able to create data structures instead of modifying ones which is already exist.

#### **Referential transparency**

Functional programs performs operations as if it is for the first time. So, you will know what may or may not have happened during the program's execution, and its side effects. In Functional Programming term it is called Referential transparency.

#### **Modularity**

Modular design increases and improves productivity. Small modules can be coded quickly and have a greater chance of re-use which leads to faster development of programs. Apart from it, the modules can be tested separately which helps you to reduce the time spent on unit testing and debugging as you can easily fish out errors or bugs.

#### **Maintainability**

Maintainability is a simple term which means Functional Programming is easier to maintain as you don't need to worry about accidentally changing anything outside the specified function.

#### **First-class function**

'First-class function' is a definition, attributed to programming language entities that have no restriction on they are used. First-class functions can appear anywhere in a program.

#### **Closure**

The closure is an inner function which can access variables of parent function's, even after the parent function has executed.

#### **Higher-order functions**

Higher-order functions either take other functions as arguments or return them as results.

Higher-order functions allow partial applications or currying. This technique applies a function to its arguments one at a time, as each application returning a new function which accepts the next argument.

#### **Pure function**

A **Pure function** is a function whose inputs are declared as inputs and none of them should be hidden. The outputs are also declared as outputs.

Pure functions act on their parameters. It is not efficient if you will not be returning anything.

##### **Example:**

Function Pure(x,y)

{

return x+y;

}

#### **Impure functions**

Impure functions have hidden inputs or output; it is called impure. Impure functions cannot be used or tested in isolation as they have dependencies(they are dependent on other resources outside them)

##### **Example**

int num = 12;

function notPure(){

num += 14;

}

#### **Function Composition**

Function composition is combining 2 or more functions to make a new function.

## The Benefits of Functional Programming

* Pure functions are easier to understand
* Testing is easier and faster
* Debugging is easier
* Programs are more bulletproof
* Programs are written at a higher level, and are therefore easier to comprehend
* Function signatures are more meaningful
* Parallel/concurrent programming is easier

## Functional Programming vs. Object-oriented Programming

|  |  |
| --- | --- |
| **Functional Programming** | **OOP** |
| Makes use of Immutable data. | Makes use of Mutable data. |
| Follows Declarative Programming based Model. | Follows Imperative Programming Model. |
| Focuses on "What you are doing in the programme." | Focuses on "How you are doing your programming." |
| Supports Parallel Programming. | Doesn’t support Parallel Programming. |
| Flow Control is performed using function calls & function calls with recursion. | Flow control process is conducted using loops and conditional statements. |
| Execution order of statements is not very important. | Execution order of statements is important. |
| Supports both "Abstraction over Data" and "Abstraction over Behavior." | Supports only "Abstract” |

**Conclusion**

* Functional programming concepts focuses on results, not the process
* The objective of any FP language is to mimic mathematical functions
* A 'Pure function' is a function whose inputs are declared as inputs and none of them should be hidden. The outputs are also declared as outputs.
* Immutable Data means that you should easily able to create data structures instead of modifying ones which is already exist
* Allows you to avoid confusing problems and errors in the code
* Functional code is not easy, so it is a bit difficult to understand for the beginner
* FP uses Immutable data while OOP uses Mutable data