



Learning Objectives

After the completion of this chapter, the student will be able to:

- Understand Function Specification.
- Parameters (and arguments).
- Interface Vs Implementation.
- Pure functions.
- Side - effects (impure functions).

1.1 Introduction

The most important criteria in writing and evaluating the algorithm is the time it takes to complete a task. The duration of computation time must be independent of the programming language, compiler, and computer used. As you aware that algorithms are expressed using statements of a programming language. If a bulk of statements to be repeated for many numbers of times then subroutines are used to finish the task.

Subroutines are the basic building blocks of computer programs. Subroutines are small sections of code that are used to perform a particular task that can be used repeatedly. In Programming languages these subroutines are called as Functions.

1.2 Function with respect to Programming language

A function is a unit of code that is often defined within a greater code structure. Specifically, a function contains a set of code that works on many kinds of inputs, like variables, expressions and produces a concrete output.

1.2.1 Function Specification

Let us consider the example $a := (24)$. $a := (24)$ has an expression in it but (24) is not itself an expression. Rather, it is a function definition. Definitions bind values to names, in this case the value 24 being bound to the name 'a'. Definitions are not expressions, at the same time expressions are also not treated as definitions. Definitions are distinct syntactic blocks. Definitions can have expressions nested inside them, and vice-versa.

1.2.2 Parameters and arguments

Parameters are the variables in a function definition and arguments are the values which are passed to a function definition.

1. Parameter without Type

Let us see an example of a function definition: