R - Functions

A function is a set of statements organized together to perform a specific task. R has a large number of in-built functions and the user can create their own functions.

In R, a function is an object so the R interpreter is able to pass control to the function, along with arguments that may be necessary for the function to accomplish the actions.

The function in turn performs its task and returns control to the interpreter as well as any result which may be stored in other objects.

Function Definition

An R function is created by using the keyword **function**. The basic syntax of an R function definition is as follows −

```
function_name <- function(arg_1, arg_2, ...) {
   Function body
}
```

Function Components

The different parts of a function are −

- **Function Name** − This is the actual name of the function. It is stored in R environment as an object with this name.
- **Arguments** − An argument is a placeholder. When a function is invoked, you pass a value to the argument. Arguments are optional; that is, a function may contain no arguments. Also arguments can have default values.
- **Function Body** − The function body contains a collection of statements that defines what the function does.
- **Return Value** − The return value of a function is the last expression in the function body to be evaluated.

R has many **in-built** functions which can be directly called in the program without defining them first. We can also create and use our own functions referred as **user defined** functions.

Built-in Function

Simple examples of in-built functions are **seq()**, **mean()**, **max()**, **sum(x)** and **paste(...)** etc. They are directly called by user written programs. You can refer most widely used R functions.

Live Demo

```
# Create a sequence of numbers from 32 to 44.
print(seq(32,44))

# Find mean of numbers from 25 to 82.
print(mean(25:82))

# Find sum of numbers frm 41 to 68.
print(sum(41:68))
```

When we execute the above code, it produces the following result –

```
[1] 32 33 34 35 36 37 38 39 40 41 42 43 44
[1] 53.5
[1] 1526
```

User-defined Function

We can create user-defined functions in R. They are specific to what a user wants and once created they can be used like the built-in functions. Below is an example of how a function is created and used.

```
# Create a function to print squares of numbers in sequence.
new.function <- function(a) {
   for(i in 1:a) {
      b <- i^2
      print(b)
```

```
   }
}
```

Calling a Function

Live Demo

```
# Create a function to print squares of numbers in sequence.
new.function <- function(a) {
   for(i in 1:a) {
      b <- i^2
      print(b)
   }
}

# Call the function new.function supplying 6 as an argument.
new.function(6)
```

When we execute the above code, it produces the following result –

```
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
```

Calling a Function without an Argument

Live Demo

```
# Create a function without an argument.
new.function <- function() {
   for(i in 1:5) {
      print(i^2)
   }
}
```

```
# Call the function without supplying an argument.
new.function()
```

When we execute the above code, it produces the following result –

```
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
```

Calling a Function with Argument Values (by position and by name)

The arguments to a function call can be supplied in the same sequence as defined in the function or they can be supplied in a different sequence but assigned to the names of the arguments.

Live Demo

```
# Create a function with arguments.
new.function <- function(a,b,c) {
   result <- a * b + c
   print(result)
}

# Call the function by position of arguments.
new.function(5,3,11)

# Call the function by names of the arguments.
new.function(a = 11, b = 5, c = 3)
```

When we execute the above code, it produces the following result –

```
[1] 26
[1] 58
```

Calling a Function with Default Argument

We can define the value of the arguments in the function definition and call the function without supplying any argument to get the default result. But we can also call such functions by supplying new values of the argument and get non default result.

Live Demo

```
# Create a function with arguments.
new.function <- function(a = 3, b = 6) {
   result <- a * b
   print(result)
}

# Call the function without giving any argument.
new.function()

# Call the function with giving new values of the argument.
new.function(9,5)
```

When we execute the above code, it produces the following result –

```
[1] 18
[1] 45
```

Lazy Evaluation of Function

Arguments to functions are evaluated lazily, which means so they are evaluated only when needed by the function body.

Live Demo

```
# Create a function with arguments.
new.function <- function(a, b) {
   print(a^2)
   print(a)
   print(b)
}

# Evaluate the function without supplying one of the arguments.
new.function(6)
```

When we execute the above code, it produces the following result –

```
[1] 36
[1] 6
Error in print(b) : argument "b" is missing, with no default
```