R Functions

A set of statements which are organized together to perform a specific task is known as a function. R provides a series of in-built functions, and it allows the user to create their own functions. Functions are used to perform tasks in the modular approach.

Functions are used to avoid repeating the same task and to reduce complexity. To understand and maintain our code, we logically break it into smaller parts using the function. A function should be

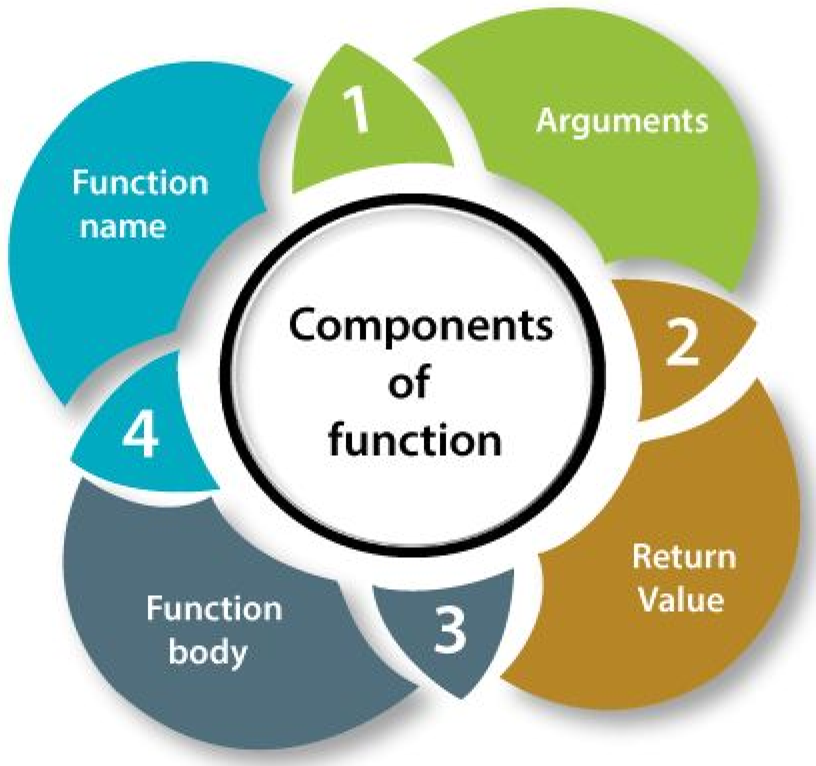
1. Written to carry out a specified task.
2. May or may not have arguments
3. Contain a body in which our code is written.
4. May or may not return one or more output values.

"An R function is created by using the keyword function." There is the following syntax of R function:

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

Components of Functions

There are four components of function, which are as follows:



Function Name

The function name is the actual name of the function. In R, the function is stored as an object with its name.

Arguments

In R, an argument is a placeholder. In function, arguments are optional means a function may or may not contain arguments, and these arguments can have default values also. We pass a value to the argument when a function is invoked.

Function Body

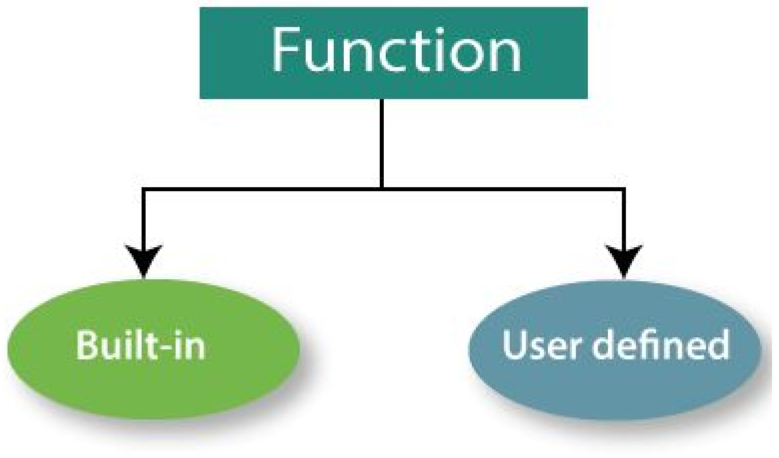
The function body contains a set of statements which defines what the function does.

Return value

It is the last expression in the function body which is to be evaluated.

Function Types

Similar to the other languages, R also has two types of function, i.e. **Built-in Function** and **User-defined Function**. In R, there are lots of built-in functions which we can directly call in the program without defining them. R also allows us to create our own functions.



Built-in function

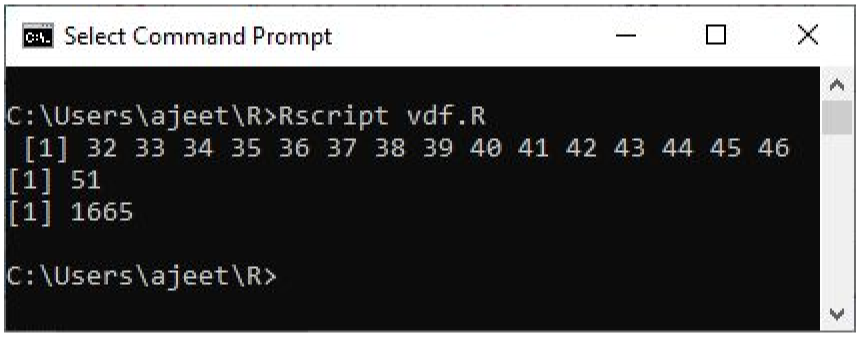
The functions which are already created or defined in the programming framework are known as built-in functions. User doesn't need to create these types of functions, and these functions are built into an application. End-users can access these functions by simply calling it. R have different types of built-in functions such as seq(), mean(), max(), and sum(x) etc.

```
# Creating sequence of numbers from 32 to 46.
print(seq(32,46))

# Finding the mean of numbers from 22 to 80.
print(mean(22:80))

# Finding the sum of numbers from 41 to 70.
print(sum(41:70))
```

Output:


Select Command Prompt

```
C:\Users\ajeet\R>Rscript vdf.R
 [1] 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
[1] 51
[1] 1665

C:\Users\ajeet\R>
```

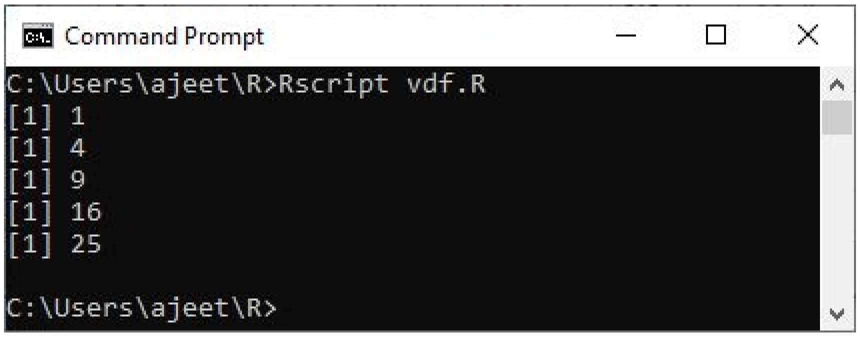
User-defined function

R allows us to create our own function in our program. A user defines a user-define function to fulfill the requirement of user. Once these functions are created, we can use these functions like in-built function.

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

new.function()
```

Output:


Command Prompt

```
C:\Users\ajeet\R>Rscript vdf.R
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25

C:\Users\ajeet\R>
```

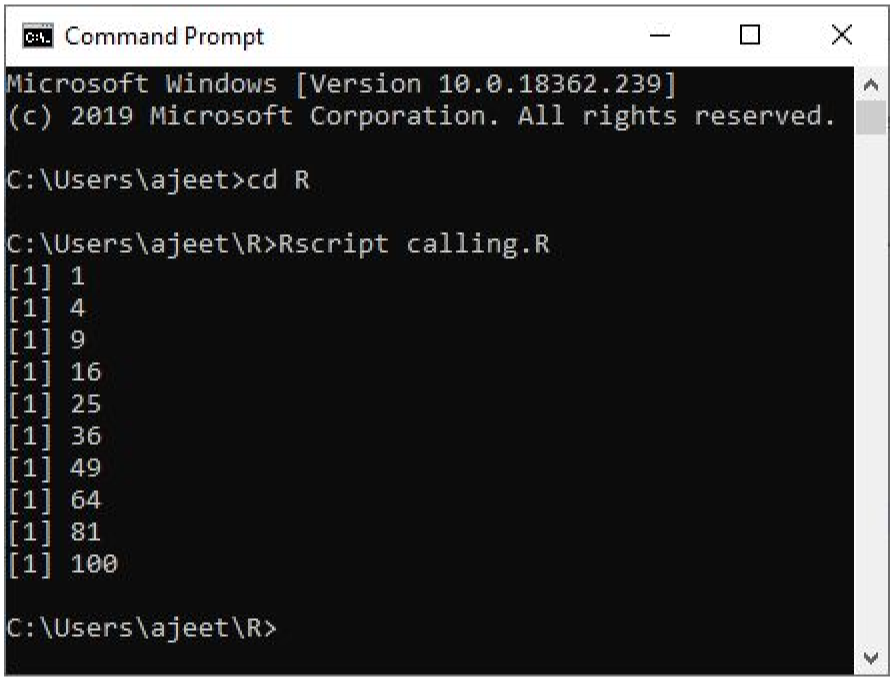
Function calling with an argument

We can easily call a function by passing an appropriate argument in the function. Let see an example to see how a function is called.

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

# Calling the function new.function supplying 10 as an argument.
new.function(10)
```

Output:



```
Microsoft Windows [Version 10.0.18362.239]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\ajeet>cd R

C:\Users\ajeet\R>Rscript calling.R
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
[1] 49
[1] 64
[1] 81
[1] 100

C:\Users\ajeet\R>
```

Function calling with no argument

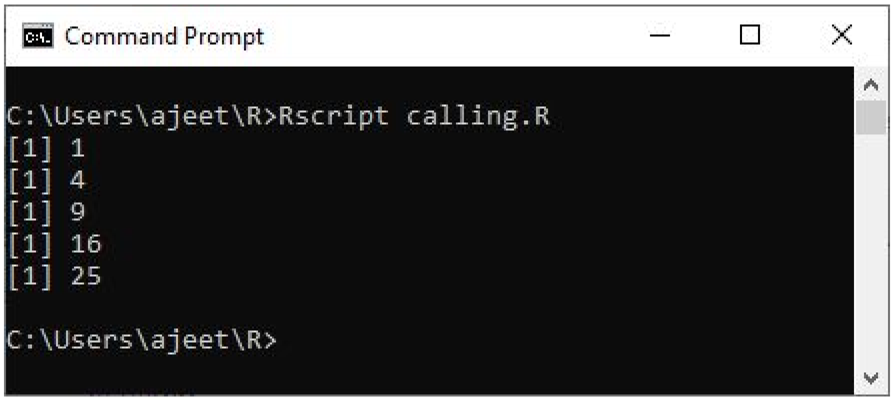
In R, we can call a function without an argument in the following way

```
# Creating a function to print squares of numbers in sequence.
new.function <- function() {
  for(i in 1:5) {
    a <- i^2
    print(a)
```

```r
    }
}

# Calling the function new.function with no argument.
new.function()
```

Output:



```
C:\Users\ajeet\R>Rscript calling.R
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25

C:\Users\ajeet\R>
```

Function calling with Argument Values

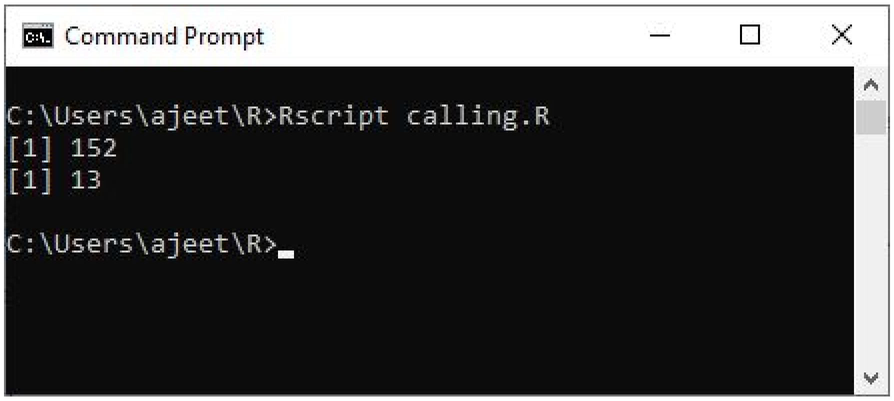
We can supply the arguments to a function call in the same sequence as defined in the function or can supply in a different sequence but assigned them to the names of the arguments.

```r
# Creating a function with arguments.
new.function <- function(x,y,z) {
  result <- x * y + z
  print(result)
}

# Calling the function by position of arguments.
new.function(11,13,9)

# Calling the function by names of the arguments.
new.function(x = 2, y = 5, z = 3)
```

Output:



```
C:\Users\ajeet\R>Rscript calling.R
[1] 152
[1] 13

C:\Users\ajeet\R>
```

Function calling with default arguments

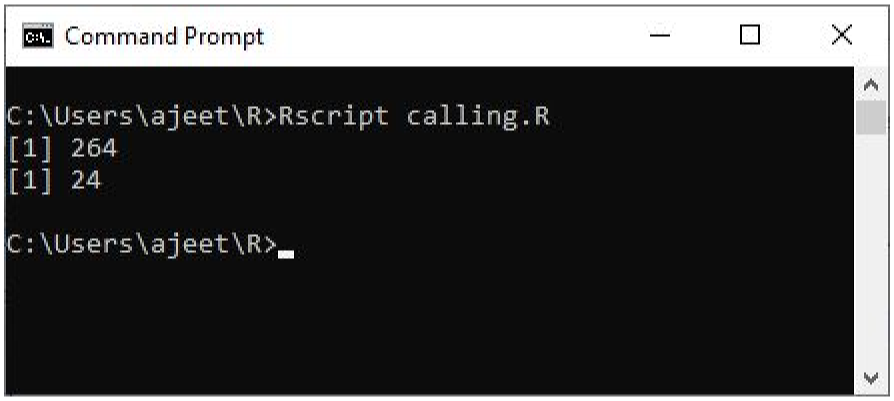
To get the default result, we assign the value to the arguments in the function definition, and then we call the function without supplying argument. If we pass any argument in the function call, then it will get replaced with the default value of the argument in the function definition.

```
# Creating a function with arguments.
new.function <- function(x = 11, y = 24) {
  result <- x * y
  print(result)
}

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

# Calling the function with giving new values of the argument.
new.function(4,6)
```

Output:



```
C:\Users\ajeet\R>Rscript calling.R
[1] 264
[1] 24

C:\Users\ajeet\R>
```