

# User-Defined Functions

**PRESENTED BY**

*Vinod Raju*  
*Data Science Practitioner*

## TOC

[Functions](#)

[Functions and Data Structures](#)

[Functions with Variable Number of Arguments](#)

[Summary](#)

Topic Number	Topic Name
1	<a href="#"><u>Functions</u></a>
2	<a href="#"><u>Functions and Data Structures</u></a>
3	<a href="#"><u>Functions with Variable Number of Arguments</u></a>
4	<a href="#"><u>Summary</u></a>

## TOC

[Functions](#)

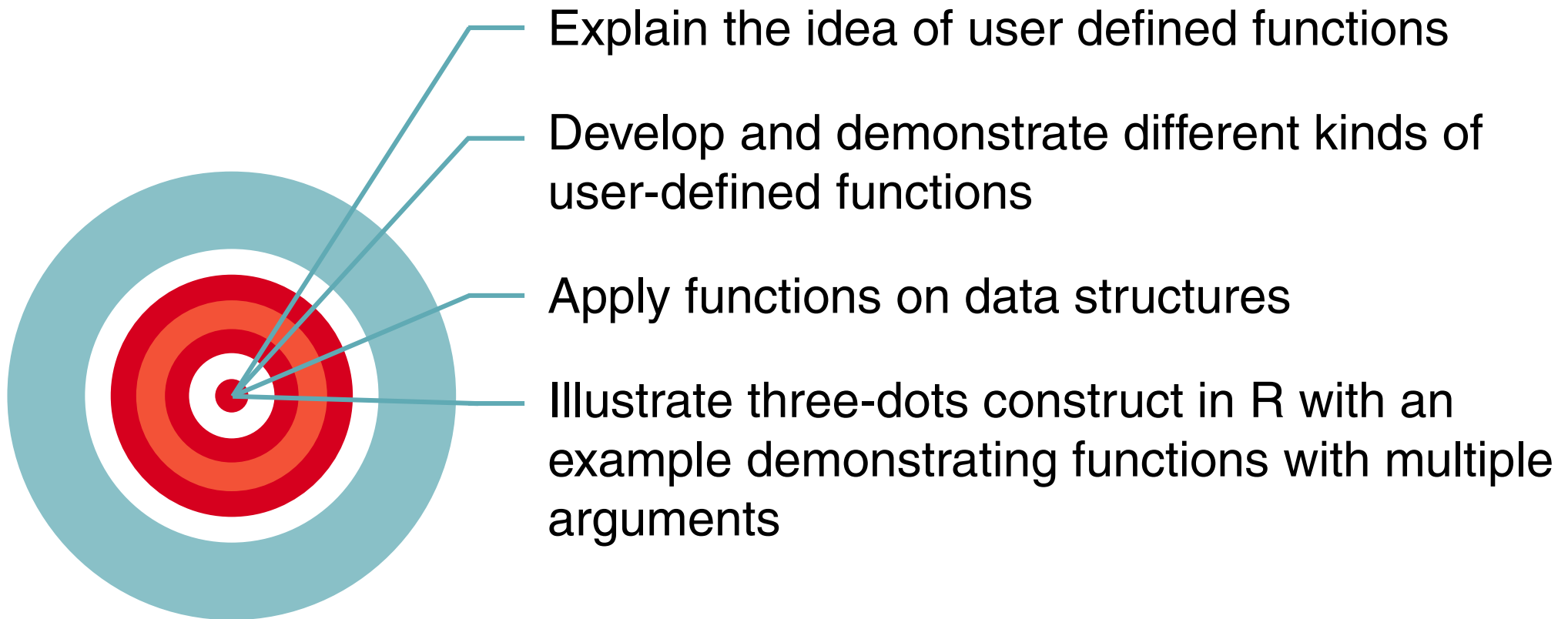
[Functions and Data Structures](#)

[Functions with Variable Number of Arguments](#)

[Summary](#)

## Learning Objectives

By the end of this unit, you will be able to:



## TOC

[Functions](#)

[Functions and Data Structures](#)

[Functions with Variable Number of Arguments](#)

[Summary](#)

# Functions

## TOC

# Functions

[Functions](#)[Functions and Data Structures](#)[Functions with Variable Number of Arguments](#)[Summary](#)

Function can be defined as a set of statements organised together to perform a specific task facilitating code reuse. They are also known as user-defined functions.

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

**Example: This below function takes a value “y” and checks if it is greater than 1**

```
isGTOne <- function(y)  
{  
    if (y > 1)  
        return("y is greater than 1")  
    else  
        return("y is less than 1")  
}
```

## TOC

# Detailed Explanation of Functions

[Functions](#)[Functions and Data Structures](#)[Functions with Variable Number of Arguments](#)[Summary](#)

Below is an example of a function taking arguments. The value which the argument “a” in the function is received from the point where the function is called.

### Example:

```
myfunc = function(a) { # A function to print hello 5 times
i = 1
while( i <= a)
{      print("Hello")
      i=i+1          }
```

### Call the function myfunc:

`myfunc(5)` # Calling a function, 5 is the value that gets passed to “a”.

## TOC

# Different Kinds of Functions

[Functions](#)

[Functions and Data Structures](#)

[Functions with Variable Number of Arguments](#)

[Summary](#)

Different kinds of functions are:

Functions without arguments

Functions with default arguments

Functions returning values

## TOC

# Functions Without Arguments

[Functions](#)

[Functions and Data Structures](#)

[Functions with Variable Number of Arguments](#)

[Summary](#)

Different kinds of functions are:

Functions without arguments

Functions with default arguments

Functions returning values

### Example:

```
greetings = function() {  
    print("Hello..good morning")  
}
```

### Calling the function:

`greetings()`

### Output:

Hello..good morning



## Toc

# Functions With Default Arguments

[Functions](#)

[Functions and Data Structures](#)

[Functions with Variable Number of Arguments](#)

[Summary](#)

Different kinds of functions are:

Functions without arguments

Functions with default arguments

Functions returning values

### Example:

```
myfunc = function(a=6,b=4)
{
    s=a+b
    print(s) }
```

### Calling the function:

`myfunc()`

### Output:

10

**Note:** The function was called without passing any values to the arguments. However, the result was computed based on the default arguments present during function definition.

## TOC

# Functions Returning Values

[Functions](#)

[Functions and Data Structures](#)

[Functions with Variable Number of Arguments](#)

[Summary](#)

Different kinds of functions are:

Functions without arguments

Functions with default arguments

Functions returning values

### Example:

```
mysum = function(a,b)
{
    s = a+b
    return(s)
}
```

### Calling the function:

```
print(mysum(10,20))
```

**Note:** The functions mysum is called by passing the values 10 and 20.

## TOC

[Functions](#)

[Functions and Data Structures](#)

[Functions with Variable Number of Arguments](#)

[Summary](#)

# Functions and Data Structures

## TOC

[Functions](#)[Functions and Data Structures](#)[Functions with Variable Number of Arguments](#)[Summary](#)

# Using functions for manipulating data structures

Passing a data frame as an argument to a function:

**Example:**

Create a simple data frame called “mydata” and pass it to the function my.func

```
a = 1:5
```

```
b = 6:10
```

```
c = 11:15
```

```
mydata=data.frame(a,b,c)
```

```
my.func <-function(mydata)
```

```
{      row15andcol3 = mydata[c(1,5),3]
```

```
      return(row15andcol3)      }
```

```
print(my.func(mydata)) # function call
```

The third element of row 1 and row 5 would be returned.

## TOC

[Functions](#)

[Functions and Data Structures](#)

[Functions with Variable Number of Arguments](#)

[Summary](#)

# Functions with Variable Number of Arguments

## TOC

[Functions](#)[Functions and Data Structures](#)[Functions with Variable Number of Arguments](#)[Summary](#)

# The Three-dots Construct in R

The three dots construct is a mechanism which allows a function to accept variable number of arguments.

## Example:

```
print(x, ...)
```

This means that, the function can take any number of named or unnamed arguments.

## TOC

[Functions](#)

[Functions and Data Structures](#)

[Functions with Variable Number of Arguments](#)

[Summary](#)

# The Three-dots Construct in R

## Example:

```
Greetings <- function(...)  
  {  
    arguments <- list(...)  
    print(arguments)  
  }
```

```
Greetings("Good", "Morning", "!")
```

## TOC

[Functions](#)

[Functions and Data Structures](#)

[Functions with Variable Number of Arguments](#)

[Summary](#)

# Summary



## TOC

[Functions](#)

[Functions and Data Structures](#)

[Functions with Variable Number of Arguments](#)

[Summary](#)

## Summary

In this unit, you learnt:

- Function can be defined as a set of statements organised together to perform a specific task facilitating code reuse. They are also known as user-defined functions.
- Different kinds of functions are:
  - Functions without arguments
  - Functions with default arguments
  - Functions returning values
- The three dots construct is a mechanism which allows a function to accept variable number of arguments.

**THANK  
YOU!**

Copyright Manipal Global Education Services Pvt. Ltd. All Rights Reserved.

*All product and company names used or referred to in this work are trademarks or registered trademarks of their respective holders. Use of them in this work does not imply any affiliation with or endorsement by them.*

*This work contains a variety of copyrighted material. Some of this is the intellectual property of Manipal Global Education, some material is owned by others which is clearly indicated, and other material may be in the public domain. Except for material which is unambiguously and unarguably in the public domain, permission is not given for any commercial use or sale of this work or any portion or component hereof. No part of this work (except as legally allowed for private use and study) may be reproduced, adapted, or further disseminated without the express and written permission of Manipal Global Education or the legal holder of copyright, as the case may be.*