Operators and Functions

Kishore Hari

2022-08-08

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

Using R markdown is very simple. There is a nice cheatsheet (https://rmarkdown.rstudio.com/lesson-15.HTML) available for shortcuts. And you can embed R code and the results in these documents and get it all in a clean format.

Commenting

In the code blocks below, you will find text that follows a hash (#). Such text is called a **comment**. It is written for the convenience of the programmer/reader and is ignored by R. Essentially, R will automatically proceed to the next line if it encounters a #.

Operator

Operators are symbols that help you carry out some basic functionalities in R. Commonly used operators are:

- The assignment operator : '<-'
- The arithmetic operators : '+', '-', '*','/','^'
- Relational operators: '<', '>', '==', '>=', '<='

There are a wealth of other operators defined in R for different functions. A full list with details is given at https://www.tutorialspoint.com/r/r_operators.htm

Operators have a unique structure, in that there is always a right-hand-side expression (RHS) and a left-hand-side expression (LHS), as shown below.

```
a <- 3 # assignment. More precisely, Left Assignment. Note the LHS and RHS
b <- 4
a + b # addition
```

```
## [1] 7
```

```r
a * b # multiplication
```

```
## [1] 12
```

```r
a - b # subtraction
```

```
## [1] -1
```

```r
a / b # division
```

```
## [1] 0.75
```

```r
a ^ b # exponent
```

```
## [1] 81
```

The operators do not require spaces before and after, but giving spaces makes the code readable.

Functions

A function in the context of programming is defined as a set of commands/instructions that are aimed at performing a task. How is it different from the commands we write normally? Well, think of functions as placeholders for a block of code that you will be using multiple times in multiple places in the script.

The syntax for creating and using a function in R is as follows:

```r
## Function definition
myFunction <- function(args) { # arguments in the circular brackets
    # set of instructions here
    return()
}

## Function usage, referred to as calling a function.
myFunction(args = ___) # replace ___ with the appropriate arguments
```

There are three noteworthy aspects of the syntax above:

1. The circular brackets (parenthesis). The "arguments" required for the function go here. More details about arguments below.
2. The curly brackets. The set of instructions are written between the curly brackets, one per line.
3. The return command. This command defines the output of the function, such that when the function is called later, the value of the function becomes said output.

Arguments

Arguments are special variables used inside functions. As far as the definition of a function is concerned, as long as you put the argument in the parenthesis, it is considered to be defined by R and can be used freely in the code between the curly braces. While calling the function, however, you must assign a value to the argument.

An important aspect to note, is that you can assign **default values** to the arguments of a function. When such assignment is done, if no value is specifically supplied to the arguments during the fuction call, the default arguments are used.

Given below are a few examples of functions and their usage

```
## Given below is a function that takes a name and prints it,
####with a few additional statements
namePrinter <- function(nam) {
    print("Hello World!")
    print("My name is ")
    print(nam)
    return()
}
## Note that this function does not return anything.
### Therefore, the function call won't have a value. But the print command gets executed.

a <- namePrinter(nam = "Kishore")
```

```
## [1] "Hello World!"
## [1] "My name is "
## [1] "Kishore"
```

```
b <- namePrinter(nam = "Hadley")
```

```
## [1] "Hello World!"
## [1] "My name is "
## [1] "Hadley"
```

```
print(a)
```

```
## NULL
```

```
## Here is another function that performs some arithmetic operations on two numbers
### This is also an example of assigning default values to the arguments.

arithmetic <- function(num1, num2 = 3) {
    x <- num1 + num2
    y <- x*num2
    z <- x*y
    return(z)
}

k <- arithmetic(num1 = 3, num2 = 4)
l <- arithmetic(num1 = 5) # It is okay to not supply num2, but we must supply num1.
# Try calling the function without supplying num1. Note the error.
k
```

```
## [1] 196
```

```
l
```

```
## [1] 192
```

When the function is called, the value that is returned becomes the value of the function as well. If a number is returned, the value of the function is a number. If a word is returned, the value is a word.

Excercises:

1. Functions are the heart and soul of R. Majority of the utilities in R are functions. We have encountered quite a few of them ourselves. Make a list of them and note the return value of these functions. Hint: Use the help panel in R studio.
2. Read about functions from any 5 websites of your choice.
3. Write a function that takes 5 numbers as arguments and returns their arithmetic mean.
4. Write another function that takes 5 numbers and returns their geometric mean.
5. Do you think it is possible to return both arithmetic mean and geometric mean for the five numbers with the same function.
6. Read the tutorials point page about operators.