Returning from a Function

In this lesson, we will learn how to return a value from a function in R.

We'll cover the following

- The return() Function
  - Syntax of return() Function
- Returning a Value Without the return() Function
- Processing Lists and Vectors
- Returning Multiple Values in R

The `return()` Function

Let's visit our previous example: finding the maximum number between two numbers.

```
maxNumber <- function(myNumber1, myNumber2)
{
  if(myNumber1 > myNumber2)
  {
    print(myNumber1)
  } else
  {
    print(myNumber2)
  }
}
```

Here, we are printing the **maximum number**. Now, suppose instead of printing the maximum number our program needs to perform some **task** with the maximum number. Therefore, instead of printing, we want the function `maxNumber` to give us the number, i.e., return the maximum number so that we can *use* it.

This is where the function `return()` comes in handy.

Syntax of `return()` Function

```
return(expression)
```

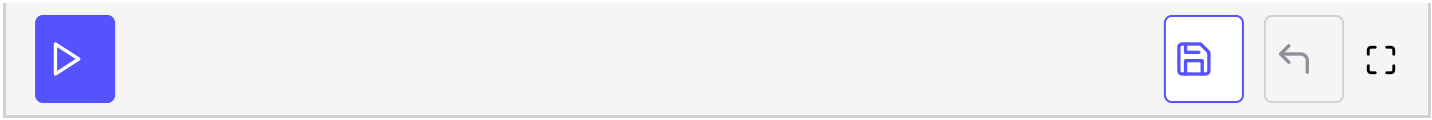
> The value returned from a `function` can be any valid object.

We can simply return the maximum number and the program calling the function can receive it.

```
maxNumber <- function(myNumber1, myNumber2)
{
  if(myNumber1 > myNumber2)
  {
    return(myNumber1)
  } else
  {
    return(myNumber2)
  }
}

# Driver Code
m <- 2
n <- 5
max <- maxNumber(m, n) # calling the maxNumber() function and
                       # getting the returned value in the variable max

# Now we can perform any task with this value for example
cat("The maximum value between ", m, " and ", n, " is: ", max)
```



Returning value from a function

In the above code snippet, the function `maxNumber()` uses `return()` to return the larger of the two numbers. The calling code, or driver code, receives this value and stores it in the `max` variable.

Returning a Value Without the `return()` Function

However, the interesting thing about the R language is; we need not specifically call the `return()` function. If there are no explicit *returns* from a function, the value of the **last evaluated expression** is returned automatically in R language.

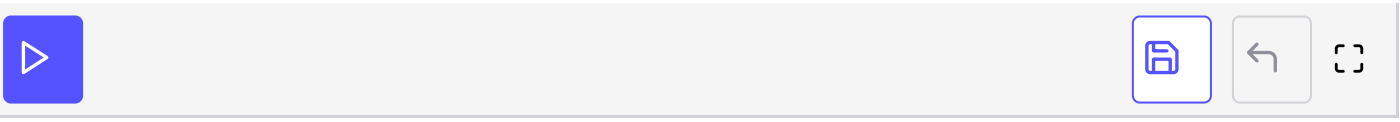
Let's try this out:

```
maxNumber <- function(myNumber1, myNumber2)
{
  if(myNumber1 > myNumber2)
  {
    myNumber1 # No return function; but this is the last expression
```

```
            # if conditional statement is satisfied
  } else
  {
    myNumber2 # No return function; but this is the last expression
              # if conditional statement is not satisfied
  }
}

# Driver Code
m <- 2
n <- 5
max <- maxNumber(m, n) # calling the maxNumber() function and
                       # getting the returned value in the variable max

cat("The maximum value between ", m, " and ", n, " is: ", max)
```



Last expression of a function is returned automatically in R language.

In the above code snippet, **Line 5** is the last expression to be evaluated if the conditional statement `myNumber1 > myNumber2` is satisfied. Therefore, this expression `myNumber1` is simply returned. Similarly, **Line 9** is the last line to be executed if the conditional statement `myNumber1 > myNumber2` is NOT satisfied. Therefore, this expression `myNumber2` is simply returned.

Processing Lists and Vectors

Let's begin by modifying our example: Instead of two arguments, our function will now take 1 argument which is a `vector` containing n numbers and finds the maximum number amongst all of them.

> Now, since we have a `vector`, and we are going to iterate over each of its elements, we can use a `loop`.

```
maxNumber <- function(myVector)
{
  max <- -Inf # Here -Inf is a reserved word for negative infinity (the minimum number possible)
            # we need to set this to perform the comparison with the first element

  # looping over the entire vector
  for(v in myVector)
  {
    if(max < v) # compared the current element with max variable
    {
      max = v # if a local maximum is found
               # i.e current value is maximum uptil now, update the max variable
    }
```

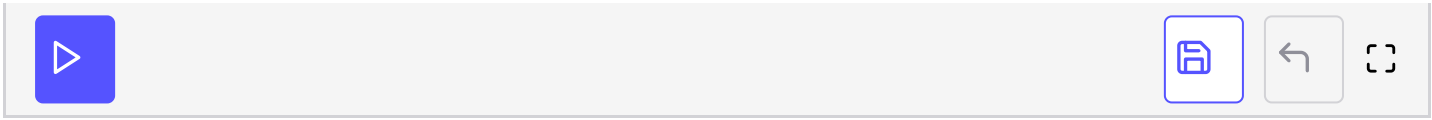
```
  }

  max # returning the max variable. This is the last expression being evlauted in the function.

}

# Driver Code
inputs <- c(2, 5, 4, 10)
output <- maxNumber(inputs) # calling the maxNumber() function and
                      # getting the returned value in the variable max

print(paste("The maximum value in the vector is: ", output), quote = FALSE)
```



Find the maximum number in a vector

In the above code, we used a reserved word:

```
-Inf
```

> Reserved words are the set of words that have special **meaning** and **purpose**. These keywords **cannot** be used as *identifiers*.

Here, `-Inf` is used so that initially the variable `max` has the least value possible. Now, when the loop from **Line** 7 will start, the conditional statement `max < v` will evaluate to `TRUE` for the first element in any vector, unless, of course, if the first element is `-Inf` itself. This value will be stored in `max`.

From the second iteration onwards, the `max` variable is compared with the current element `v`. If the conditional statement `max < v` is satisfied, variable `max` is updated, otherwise, there is no change to `max`.

Returning Multiple Values in R

How about we introduce another modification. Now, we want our `function` to return the **maximum** as well as the **minimum** value.

For this, we can store both the **maximum** and **minimum** value in a list and then return it. Let's see how this is implemented:

```
maxMinNumber <- function(myVector)
{
  max <- -Inf # initially the max number is set to -Inf
           # we need to set this to perform the comparison with the first element
```

```
  min <- Inf # Here Inf is a reserved word for infinity (the maximum number possible)
             # we need to set this to perform the comparison with the first element

  # looping over the entire vector
  for(v in myVector)
  {
    if(max < v) # check the current element with the max variable
    {
      max = v # if a local maximum is found
               # i.e current value is maximum uptil now, update the max variable
    }
    if(min > v) # check the current element with the min variable
    {
      min = v # if a local minimum is found
               # i.e current value is minimum uptil now, update the min variable
    }
  }

  list(max, min) # returning both the max and min in the form of a list. This is the last expressi
}

# Driver Code
input <- c(2, 5, 4, 10)
output <- maxMinNumber(input) # calling the maxMinNumber() function and

# Here the variable output is a list where first element is the maximum value
# and the second element is the minimum value
print(paste("The maximum value in the vector is: ", output[1]), quote = FALSE)
print(paste("The minimum value in the vector is: ", output[2]), quote = FALSE)
```

Finding maximum and minimum value in a vector

In this code on **Line 24** we return a `list` and therefore when we receive the output, we have to treat it like a `list`.

> The variable `output` in this code snippet is a list with the first element the **maximum** value and the second element the **minimum** value.

How about a quick exercise on returning data from a function in the next lesson?