R

Programming

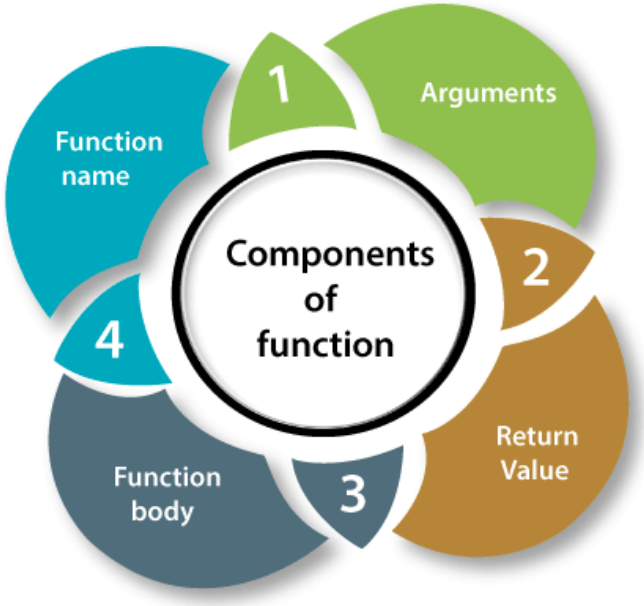
R Functions

A set of statements which are organized together to perform a specific task is known as a function.

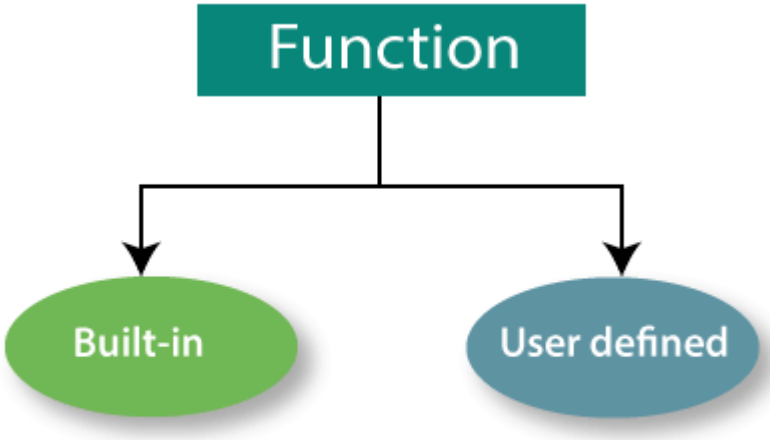
R provides a series of in-built functions, and it allows the user to create their own functions.

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

   Function body

}
```

Functions


Components of function
1
Arguments
2
Return Value
3
Function body
4
Function name

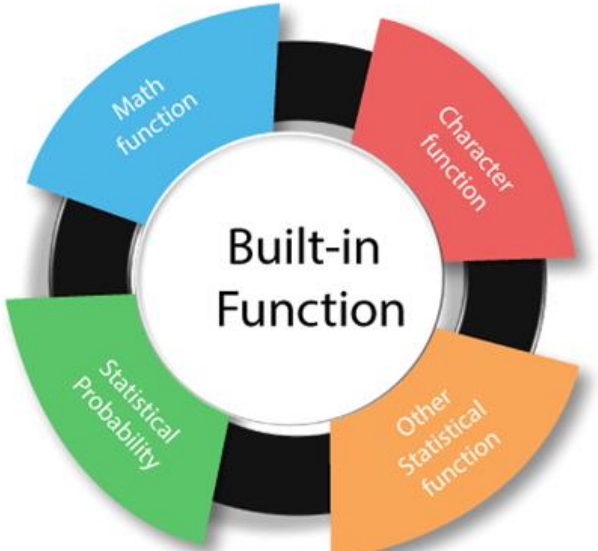
Function Types



Built-in functions

The functions which are already created or defined in the programming framework are known as built-in functions.

R have different types of built-in functions such as seq(), mean(), max(), and sum(x) etc.



Math Functions

abs(x), sqrt(x), ceiling(x)

floor(x), trunc(x), round(x, digits=n)

cos(x), sin(x), tan(x)

log(x), log10(x), exp(x)

String Function

R provides various string functions to perform tasks.

These string functions allow us to extract substring from string, search pattern etc.

substr(x, start=n1,stop=n2) -It is used to extract substrings in a character vector.

grep(pattern, x , ignore.case=FALSE, fixed=FALSE) It searches for pattern in x.

sub(pattern, replacement, x, ignore.case =FALSE, fixed=FALSE) It finds pattern in x and replaces it with replacement (new) text.

String Function

paste(..., sep="") -It concatenates strings after using sep string to separate them.

strsplit(x, split)- It splits the elements of character vector x at split point.

tolower(x) -It is used to convert the string into lower case.

toupper(x) -It is used to convert the string into upper case.

User defined functions

Simple function

Function with arguments

Function with default arguments

Function with return values

Nested functions

Nested Functions

There are two ways to create a nested function:

- Call a function within another function.
- Write a function within a function.

Recursion

R also accepts function recursion, which means a defined function can call itself.

Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.

R Global Variables

Variables that are created outside of a function are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.

Data Structures in R Programming

Vectors

- In R, a sequence of elements which share the same data type is known as vector.
- Vector is classified into two parts, i.e., Atomic vectors and Lists.

Creating vector

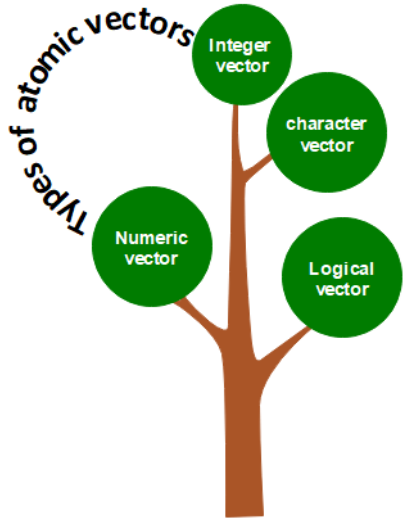
- Using the colon(:) operator
- z<-x:y
- a<-4:-10

Using the seq() function

1. seq_vec<-seq(1,4,by=0.5)
2. seq_vec
3. class(seq_vec)
4. seq_vec<-seq(1,4,length.out=6)
5. seq_vec
6. class(seq_vec)

Atomic vectors in R

In R, there are four types of atomic vectors. Atomic vectors are created with the help of **c()** function.



Numeric vector

The decimal values are known as numeric data types in R.

A vector which contains numeric elements is known as a numeric vector.

Eg: numeric-vector.R

Integer vector

A non-fraction numeric value is known as integer data.

This integer data is represented by "Int."

The Int size is 2 bytes and long Int size of 4 bytes.

There is two way to assign an integer value to a variable, i.e., by using as.integer() function and appending of L to the value.

Eg: integer-vector.R

Character vector

A character is held as a one-byte integer in memory.

In R, there are two different ways to create a character data type value, i.e., using as.character() function and by typing string between double quotes("") or single quotes('').

Logical vector

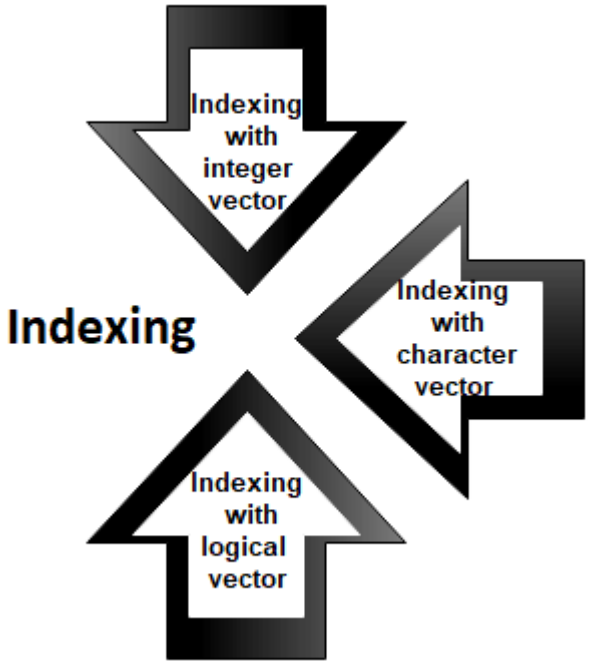
The logical data types have only two values i.e., True or False.

These values are based on which condition is satisfied.

A vector which contains Boolean values is known as the logical vector.

Eg: logical_vector.R

Accessing elements of vectors



Indexing with integer vector

C, C++, and java the indexing starts from 0, but in R, the indexing starts from 1.

we perform indexing by specifying an integer value in square braces [] next to our vector.

Indexing with a character vector

In character vector indexing, we assign a unique key to each element of the vector.

These keys are uniquely defined as each element and can be accessed very easily.

access-vector.R

Vector Operation

Combining vectors

The c() function is not only used to create a vector, but also it is also used to combine two vectors.

By combining one or more vectors, it forms a new vector which contains all the elements of each vector.

Arithmetic operations

We can perform all the arithmetic operation on vectors.

The arithmetic operations are performed member-by-member on vectors.

We can add, subtract, multiply, or divide two vectors.

Logical Index vector

This vector has the same length as the original vector.

The vector members are TRUE only when the corresponding members of the original vector are included in the slice; otherwise, it will be false.

Numeric Index

In R, we specify the index between square braces [ ] for indexing a numerical value.

If our index is negative, it will return us all the values except for the index which we have specified.

Duplicate Index

An index vector allows duplicate values which means we can access one element twice in one operation.

Range Indexes

Range index is used to slice our vector to form a new vector.

For slicing, we used colon(:) operator.

Range indexes are very helpful for the situation involving a large operator.

Out-of-order Indexes

In R, the index vector can be out-of-order.

```
q[c(2,1,3,4,5,6)]
```

Named vectors members

We first create our vector of characters as:

```
z=c("TensorFlow","PyTorch")
```

Once our vector of characters is created, we name the first vector member as "Start" and the second member as "End"

Vector Functions

length()

sort()

Change an Item-use indexing

rep()- repeat_each <- rep(c(1,2,3), each = 3)

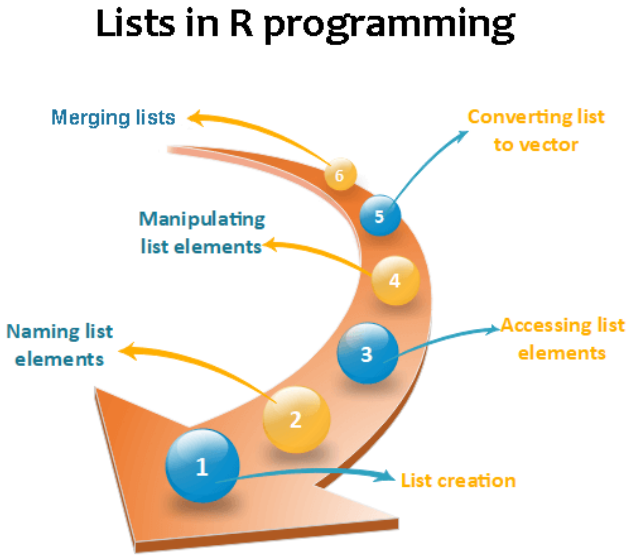
repeat_each

Applications of vectors

1. In machine learning for principal component analysis vectors are used. They are extended to eigenvalues and eigenvector and then used for performing decomposition in vector spaces.
2. The inputs which are provided to the deep learning model are in the form of vectors. These vectors consist of standardized data which is supplied to the input layer of the neural network.
3. In the development of support vector machine algorithms, vectors are used.
4. Vector operations are utilized in neural networks for various operations like image recognition and text processing.

Lists

Lists are the objects of R which contain elements of different types such as number, vectors, string and another list inside it.



Lists creation

he process of creating a list is the same as a vector.

list() which is used to create a list in R.

access the list items by referring to its index number, inside brackets.

```
thislist <- list("apple", "banana", "cherry")
thislist[1]
```

Change the values:

```
thislist <- list("apple", "banana", "cherry")
thislist[1] <- "blackcurrant"
```

List Length

```
thislist <- list("apple", "banana", "cherry")
length(thislist)
```

Check if Item Exists

To find out if a specified item is present in a list, use the %in% operator:

```
thislist <- list("apple", "banana", "cherry")
"apple" %in% thislist
```

Add List Items

To add an item to the end of the list, use the append() function:

R Arrays

R, arrays are the data objects which allow us to store data in more than two dimensions.

In R, an array is created with the help of the **array()** function.

This array() function takes a vector as an input and to create an array it uses vectors values in the **dim** parameter.

```
array_name <- array(data, dim= (row_size, column_size, matrices, dim_names))
```

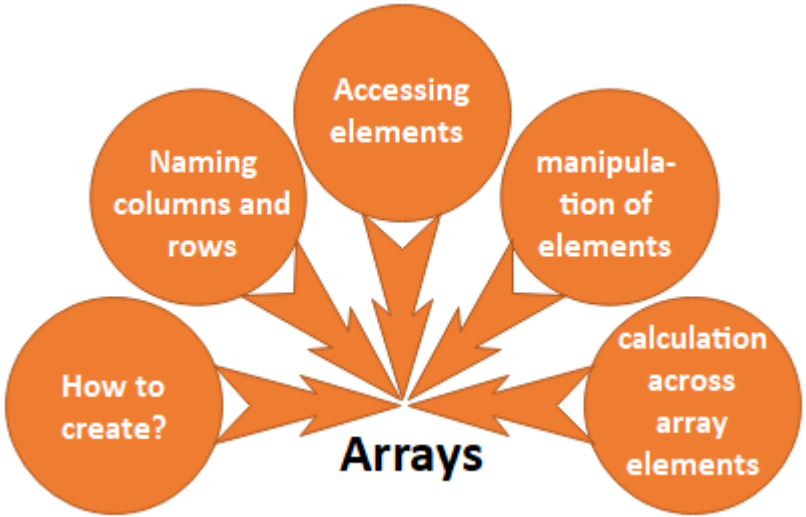
In R, the array consists of multi-dimensional matrices.

Row_size : This parameter defines the number of row elements which an array can store.

Column_size: This parameter defines the number of columns elements which an array can store.

Dim_names: This parameter is used to change the default names of rows and columns.

Arrays


Accessing elements
Naming columns and rows
manipula-tion of elements
How to create?
calculation across array elements
Arrays

Creating Array

can use the array() function to create an array, and the dim parameter to specify the dimensions.

Arrays can only have one data type.

can access the array elements by referring to the index position.

can use the [] brackets to access the desired elements from an array.

```
thisarray <- c(1:24)
multiarray <- array(thisarray, dim = c(4, 3, 2))
multiarray[2, 3, 2]
```

Check if an Item Exists

To find out if a specified item is present in an array, use the %in% operator

Amount of Rows and Columns

Use the dim() function to find the amount of rows and columns in an array.

```
thisarray <- c(1:24)

multiarray <- array(thisarray, dim = c(4, 3, 2))

dim(multiarray)
```

Array Length

Use the length() function to find the dimension of an array

Loop Through an Array

```
for(x in multiarray){
  print(x)
}
```

Naming rows and columns

In R, we can give the names to the rows, columns, and matrices of the array. This is done with the help of the dim name parameter of the array() function.

```
col_names<-c("Col1","Col2","Col3")

row_names<-
c("Row1","Row2","Row3","Row4","Row5","Row6","Row7","Row8")

res<-
array(multiarray,dim=c(8,3,1),dimnames=list(row_names,col_names))
```