R Lists

In R, lists are the second type of vector. Lists are the objects of R which contain elements of different types such as number, vectors, string and another list inside it. It can also contain a function or a matrix as its elements. A list is a data structure which has components of mixed data types. We can say, a list is a generic vector which contains other objects.

Example

1. vec **<-** c(3,4,5,6)
2. char\_vec**<-c**("shubham","nishka","gunjan","sumit")
3. logic\_vec**<-c**(TRUE,FALSE,FALSE,TRUE)
4. out\_list**<-list**(vec,char\_vec,logic\_vec)
5. out\_list

**Output:**

[[1]]

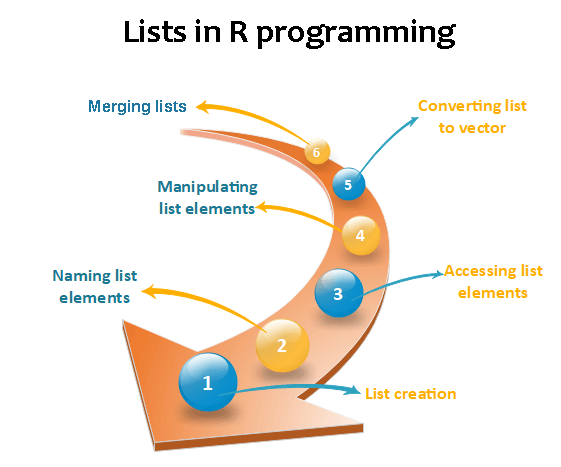
[1] 3 4 5 6

[[2]]

[1] "shubham" "nishka" "gunjan" "sumit"

[[3]]

[1] TRUE FALSE FALSE TRUE



Lists creation

The process of creating a list is the same as a vector. In R, the vector is created with the help of c() function. Like c() function, there is another function, i.e., list() which is used to create a list in R. A list avoid the drawback of the vector which is data type. We can add the elements in the list of different data types.

**Syntax**

1. list()

**Example 1:** Creating list with same data type

1. list\_1**<-list**(1,2,3)
2. list\_2**<-list**("Shubham","Arpita","Vaishali")
3. list\_3**<-list**(c(1,2,3))
4. list\_4**<-list**(TRUE,FALSE,TRUE)
5. list\_1
6. list\_2
7. list\_3
8. list\_4

**Output:**

[[1]]

[1] 1

[[2]]

[1] 2

[[3]]

[1] 3

[[1]]

[1] "Shubham"

[[2]]

[1] "Arpita"

[[3]]

[1] "Vaishali"

[[1]]

[1] 1 2 3

[[1]]

[1] TRUE

[[2]]

[1] FALSE

[[3]]

[1] TRUE

**Example 2:** Creating the list with different data type

1. list\_data**<-list**("Shubham","Arpita",c(1,2,3,4,5),TRUE,FALSE,22.5,12L)
2. print(list\_data)

In the above example, the list function will create a list with character, logical, numeric, and vector element. It will give the following output

**Output:**

[[1]]

[1] "Shubham"

[[2]]

[1] "Arpita"

[[3]]

[1] 1 2 3 4 5

[[4]]

[1] TRUE

[[5]]

[1] FALSE

[[6]]

[1] 22.5

[[7]]

[1] 12

Giving a name to list elements

R provides a very easy way for accessing elements, i.e., by giving the name to each element of a list. By assigning names to the elements, we can access the element easily. There are only three steps to print the list data corresponding to the name:

1. Creating a list.
2. Assign a name to the list elements with the help of names() function.
3. Print the list data.

Let see an example to understand how we can give the names to the list elements.

**Example**

1. # Creating a list containing a vector, a matrix and a list.
2. list\_data **<-** list(c("Shubham","Nishka","Gunjan"), matrix(c(40,80,60,70,90,80), nrow = 2),
3. list("BCA","MCA","B.tech"))
5. # Giving names to the elements in the list.
6. names(list\_data) **<-** c("Students", "Marks", "Course")
8. # Show the list.
9. print(list\_data)

**Output:**

$Students

[1] "Shubham" "Nishka" "Gunjan"

$Marks

[,1] [,2] [,3]

[1,] 40 60 90

[2,] 80 70 80

$Course

$Course[[1]]

[1] "BCA"

$Course[[2]]

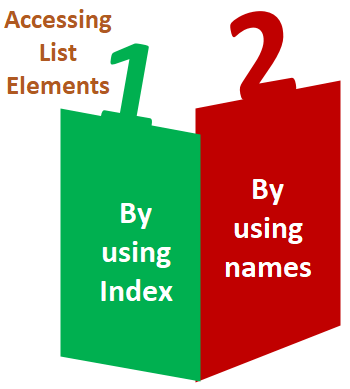
[1] "MCA"

$Course[[3]]

[1] "B. tech."

Accessing List Elements

R provides two ways through which we can access the elements of a list. First one is the indexing method performed in the same way as a vector. In the second one, we can access the elements of a list with the help of names. It will be possible only with the named list.; we cannot access the elements of a list using names if the list is normal.



Let see an example of both methods to understand how they are used in the list to access elements.

**Example 1:** Accessing elements using index

1. # Creating a list containing a vector, a matrix and a list.
2. list\_data **<-** list(c("Shubham","Arpita","Nishka"), matrix(c(40,80,60,70,90,80), nrow = 2),
3. list("BCA","MCA","B.tech"))
4. # Accessing the first element of the list.
5. print(list\_data[1])
7. # Accessing the third element. The third element is also a list, so all its elements will be printed.
8. print(list\_data[3])

**Output:**

[[1]]

[1] "Shubham" "Arpita" "Nishka"

[[1]]

[[1]][[1]]

[1] "BCA"

[[1]][[2]]

[1] "MCA"

[[1]][[3]]

[1] "B.tech"

**Example 2:** Accessing elements using names

1. # Creating a list containing a vector, a matrix and a list.
2. list\_data **<-** list(c("Shubham","Arpita","Nishka"), matrix(c(40,80,60,70,90,80), nrow = 2),list("BCA","MCA","B.tech"))
3. # Giving names to the elements in the list.
4. names(list\_data) **<-** c("Student", "Marks", "Course")
5. # Accessing the first element of the list.
6. print(list\_data["Student"])
7. print(list\_data$Marks)
8. print(list\_data)

**Output:**

$Student

[1] "Shubham" "Arpita" "Nishka"

[,1] [,2] [,3]

[1,] 40 60 90

[2,] 80 70 80

$Student

[1] "Shubham" "Arpita" "Nishka"

$Marks

[,1] [,2] [,3]

[1,] 40 60 90

[2,] 80 70 80

$Course

$Course[[1]]

[1] "BCA"

$Course[[2]]

[1] "MCA"

$Course[[3]]

[1] "B. tech."

Manipulation of list elements

R allows us to add, delete, or update elements in the list. We can update an element of a list from anywhere, but elements can add or delete only at the end of the list. To remove an element from a specified index, we will assign it a null value. We can update the element of a list by overriding it from the new value. Let see an example to understand how we can add, delete, or update the elements in the list.

**Example**

1. # Creating a list containing a vector, a matrix and a list.
2. list\_data **<-** list(c("Shubham","Arpita","Nishka"), matrix(c(40,80,60,70,90,80), nrow = 2),
3. list("BCA","MCA","B.tech"))
5. # Giving names to the elements in the list.
6. names(list\_data) **<-** c("Student", "Marks", "Course")
8. # Adding element at the end of the list.
9. list\_data[4] **<-** "Moradabad"
10. print(list\_data[4])
12. # Removing the last element.
13. list\_data[4] **<-** NULL
15. # Printing the 4th Element.
16. print(list\_data[4])
18. # Updating the 3rd Element.
19. list\_data[3] **<-** "Masters of computer applications"
20. print(list\_data[3])

**Output:**

[[1]]

[1] "Moradabad"

$<NA>

NULL

$Course

[1] "Masters of computer applications"

Converting list to vector

There is a drawback with the list, i.e., we cannot perform all the arithmetic operations on list elements. To remove this, drawback R provides unlist() function. This function converts the list into vectors. In some cases, it is required to convert a list into a vector so that we can use the elements of the vector for further manipulation.

The unlist() function takes the list as a parameter and change into a vector. Let see an example to understand how to unlist() function is used in R.

**Example**

1. # Creating lists.
2. list1 **<-** list(10:20)
3. print(list1)
5. list2 **<-list**(5:14)
6. print(list2)
8. # Converting the lists to vectors.
9. v1 **<-** unlist(list1)
10. v2 **<-** unlist(list2)
12. print(v1)
13. print(v2)
15. adding the vectors
16. result **<-** v1+v2
17. print(result)

**Output:**

[[1]]

[1] 1 2 3 4 5

[[1]]

[1] 10 11 12 13 14

[1] 1 2 3 4 5

[1] 10 11 12 13 14

[1] 11 13 15 17 19

Merging Lists

R allows us to merge one or more lists into one list. Merging is done with the help of the list() function also. To merge the lists, we have to pass all the lists into list function as a parameter, and it returns a list which contains all the elements which are present in the lists. Let see an example to understand how the merging process is done.

**Example**

1. # Creating two lists.
2. Even\_list **<-** list(2,4,6,8,10)
3. Odd\_list **<-** list(1,3,5,7,9)
5. # Merging the two lists.
6. merged.list **<-** list(Even\_list,Odd\_list)
8. # Printing the merged list.
9. print(merged.list)

**Output:**

[[1]]

[[1]][[1]]

[1] 2

[[1]][[2]]

[1] 4

[[1]][[3]]

[1] 6

[[1]][[4]]

[1] 8

[[1]][[5]]

[1] 10

[[2]]

[[2]][[1]]

[1] 1

[[2]][[2]]

[1] 3

[[2]][[3]]

[1] 5

[[2]][[4]]

[1] 7

[[2]][[5]]

[1] 9