Lists

In this lesson, we will learn all about lists and how to create them.

We'll cover the following

In the previous lesson, we learned about **vectors**, that are containers where each element has the same type. However, R language provides programmers with the ability to put elements of **different types** into a **single container**.

How cool is that! And this is exactly where **lists** come in the picture.

> A list is basically a vector in which all the elements can be of a *different* type.

Creating Lists

Lists can be created using the `list()` function.

> It works similar to the `c()` function we saw in the previous lesson.

In this function, list the contents you want to add as arguments to the `list()` function.

```
myList <- list(1, 1+1i, "a", TRUE)
print(myList)
```

Lists have the capability of containing other lists within themselves. Therefore, lists are *recursive*. We can test this using the `is.recursive()` function which returns `true` if the object is recursive and `false` otherwise.

```
myList <- list(1, 1+1i, "a", TRUE)
is.recursive(myList)
```

Inserting Elements in a List

We can insert an element in the list using the `c()` method that we used for inserting elements in a vector.

```
myList <- list(1, 1+1i, "a", TRUE)

myList <- c("s", myList)
cat("Appending 's' at the start of the vector: \n")
print(myList)
```

Accessing and Modifying Lists

We can fetch an element at a specific index by using the square brackets `[]` around the specified index.

> Indexing starts at 1, which means that the first element of the list is at index 1.

```
myList <- list(1, 1+1i, "a", TRUE)
print(myList[1])
```

Using single square brackets.

In the above code, we access the **first list element** of a list.

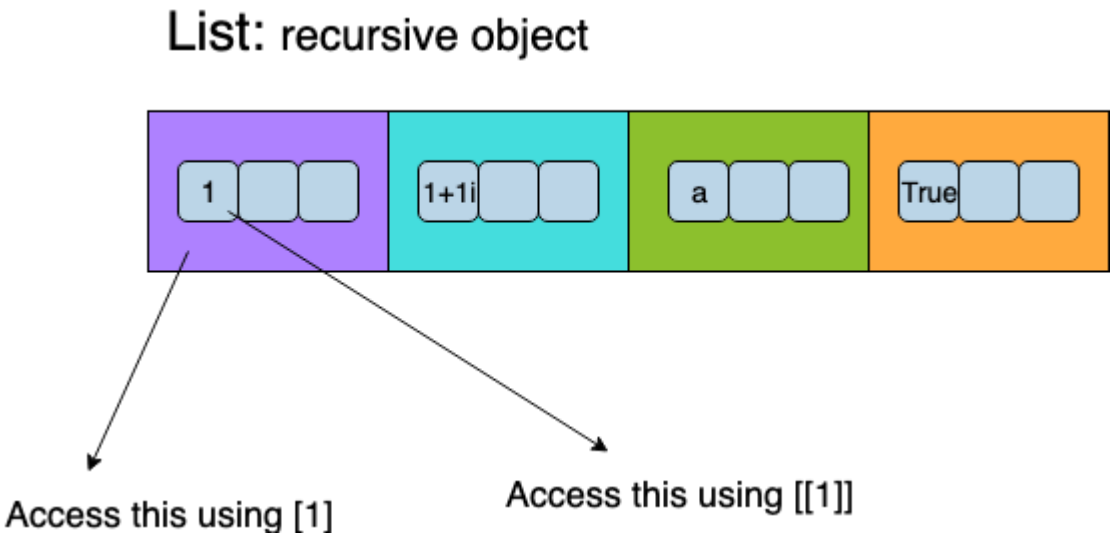

Illustration of list object

As we discussed previously, and looking at the illustration above, each element in a list can be another list, so to obtain a single element use double square brackets `[[]]` instead.

```
myList <- list(1, 1+1i, "a", TRUE)
print(myList[[1]])
```

Using double square brackets

We can *modify* a single element also using the double square brackets.

```
myList <- list(1, 1+1i, "a", TRUE)
myList[[1]] = 100
print(myList[[1]])
```

Using double square brackets for modification

Named Entries in Lists

List entries can be named:

For example:

```
myList <- list(
          integerVar = 1:3,

          numericVar = 0.5,
          characterVar = c('a', 'b') )
```

The entries in named lists can be accessed by their name instead of their indexes as well.

```
myList <- list(
  integerVar = 1:3,
  numericVar = 0.5,
  characterVar = c('a', 'b') )

print(myList['integerVar']) # prints the name as well as the value

print(myList[['integerVar']]) # prints only the value
```

As we have seen in this lesson, lists can be particularly useful because they can store objects of different lengths and various classes.

In the next lesson, we have a short exercise for you.