Beginner's Guide to Lists and List Operations in START Programming

Lists are versatile data structures used to store collections of items in programming. Understanding how to work with lists is essential for manipulating data effectively. In this guide, we'll explore various list operations using the syntax and functions provided START.

What do Lists look like?

```
l1 is [1,2,3,4,5]
l2 is ["Hello", "world"]
l3 is [true, false]
l4 is [1, "Hello", true]
```

The above are all examples of valid lists in START. They are a collection of data stored together under a single variable name (as discussed before).

Accessing List Elements

You can access individual elements in a list using index notation. Lets look at the following code:

```
l1 is [1,2,3]
firstNumber is l1[0]
write(firstNumber) nl
```

This code outputs the following result:

```
1
```

Lets examine the second line of the code, notice we use `l1[0]` to extract the first number, with 0 being used instead of 1. Almost all programming languages use **Zero-based numbering**. As such so does START, as we are following the popular convention to make learning and transitioning smooth and easy. So remember, LIST INDEX STARTS AT 0!

Modifying List Elements

You can modify a given index of a list in most languages including START, lets look at the following code as an example:

```
list is [1,2,3]
list[1] is "now a string"
write(list) nl
```

This code outputs the following result:

```
[1, now a string, 3]
```

As we can see the middle element of the list is now a string. We can modify all elements of a list to be any data type.

Removing List Elements

We can remove either the first instance or all instances of a data point from a list, lets see how can do this. Lets examine the following code:

```
list is [1,2,1,3,4,5]
remove 1 from list
write(list) nl
```

This code produces the following:

```
[2,1,3,4,5]
```

As we can see the first 1 is now removed from the list. Now lets modify the code:

```
list is [1,2,1,3,4,5]
remove all 1 from list
write(list) nl
```

This code produces the following:

```
[2,3,4,5]
```

Now no instances of 1 are left in the list.

Appending to a list

We can add data to the end of a given list. Lets see an example of this:

```
list is [1,2,3]
list is list add 4
write(list) nl
list is list add 5
write(list) nl
```

This code produces the following:

```
[1,2,3,4]
[1,2,3,4,5]
```

As we can see, firstly 4 was added to the end of the array, and then from that new version of the list, 5 is added to the end, leading to a list from 1 up to 5.

Adding 2 List together

We can take 2 separate lists and add them together to create one list with all elements of both lists in it.

```
l1 is [1,2,3]
l2 is [4,5]
```

```
l3 is l1 concat l2
write(l3) nl
```

This code produces the following:

```
[1,2,3,4,5]
```

Finding the Length of a List

We can find the length of a list using the built-in `length of` function. This will be useful later on when we look at loops. Lets see how to find the length of a list:

```
list is [1,2,3,true,false,"hello","world"]
listLen is length of list
write(listLen) nl
```

This code produces the following:

```
7
```

Conclusion

Lists are powerful data structures that allow you to store and manipulate collections of items in programming. By mastering list operations, you'll be able to organize and process data efficiently in your programs. Practice using these operations with different lists to gain a solid understanding of how to work with lists effectively.