## Data structures

## Lists

As in everyday language, in ROBO Pro Coding a list is an ordered collection of elements, such as a "to do" list or a shopping list. Elements in a list can be of any type, and the same value can appear in a list multiple times.

## Creating a list

## create list with

You can use the create list with block to enter the initial values in a new list. In this example, a list of words is created and saved in a variable named letters:

|  |  |
| :---: | :---: |
|  |  |
|  |  |

We designate this list as ["alpha," "beta," "gamma"].

This shows the block for creating a list of numbers:

|  |  |
| :---: | :---: |
|  |  |
|  |  |

Here is how to create a list of colors:


It is less common, but possible to create a list of values of different types:

## " alpha

## Change number of inputs

To change the number of inputs, click or touch the gear symbol. This will open a new window. You can drag element sub-blocks from the left side of the window to the list block on the right side to add a new input:

While the new element in this example is inserted at the bottom, it can be added anywhere. Similarly, element sub-blocks that are not desired can be dragged to the left and out of the list block.

## Create list with item

You can use the create list with item block to create a list containing the indicated number of copies of an item. The following blocks, for example, set the variable words on the list ["very," "very," "very"].

## set words " to create list with item C " ${ }^{66}$ very " repeated C 3 times

## Check the length of a list

## is empty

The value of an is empty block is true if its input is the empty list, and false if it is anything else. Is this input true ? The value of the following block would be false, because the variable color is not empty: It has three items.


Note how similar this is to the is empty block for text.

## Length of

The value of the length of block is the number of elements that are in the list used as the input. The value of the following block would be 3, for instance, since color has three elements:

## length of colors

The value of the length of block is the number of items in the list used as the input. The value of the following block would be 3, for example, although words consists of three copies of the same text:

## length of words

Note how similar this is to the block length of for the text.

## Searching for items in a list

These blocks find the position of an item in a list. The following example has a value of 1 , because the first occurrence of "very" is at the start of the list of words (["very," "very," "very"]).

## in list words . find first " occurrence of item " very"

The result of the following is 3 , because the last occurrence of "very" in the words is at position 3 .

```
in list C words * find last * occurrence of item C "6 very"
```

If the item is not in the list at all, then the result is a value of 0 , as in this example:
in list C words • find last * occurrence of item C unicorn "

These blocks behave the same way as the blocks for finding letters in text.

## Getting items from a list

## Getting a single element

Remember the definition of the list colors:


The following block contains the color blue, because it is the second item in the list (starting from the left):


This one contains green, because it is the second element (starting from the right end):
$\square$
in list colors .
get
\# from end
2

This contains the first item, red:


This contains the last item, yellow:

```
in list
colors
get
    last
```

This one chooses a random item from the list, with the same probability of returning one of the items red, blue, green or yellow.
$\square$

```
in list Colors * get * random
```


## Get and remove an item

You can use the drop down menu to change the block in list ... get to the block in list ... get and remove, which delivers the same output, but also changes the list:

this example sets the variable first letter to "alpha" and leaves the remaining letters (["beta," "gamma"]) in the list.


## Removing an entry

If you select remove from the drop down menu, the tab at the left of the block will be removed:


Then, the first item from letter will be removed.

## Get a sub-list

The block in list ... get sub-list is similar to the block in list ... get, with the difference that it extracts a sub-list and not an individual item. There are multiple options to enter the start and end of the sub-list:



In this example, a new list first letters is created. The new list has two items: ["alpha," "beta"].


Please note that this block does not change the original list.

## Adding items to a list

## Replacing items in a list

The block in list ... set replaces the item at a certain point in a list with another item.


The meanings of the individual drop down options are outlined in the previous section.

The following example does two things:

1. The list words is created with 3 items: ["very," "very," "very"].
2. The third item in the list is replaced with "good." The new value of words is ["very," "very," "good"]

| set words . to | create list with item C ${ }^{\text {c/ very }}$ " |
| :---: | :---: |
| in list $C$ words . | set * \# : 3 as C 66 well " |

## Insert items from a certain point into a list

The in list ... insert at block is accessed via the drop down menu for the in list ... set block:


It inserts a new item at the indicated point into the list, before the element that was previously located there. The following example (which builds on an earlier example) does three things:

1. The list words is created with 3 items: ["very," "very," "very"].
2. The third item in the list is replaced with "good." The new value of words is therefore ["very," "very," "good"].
3. The word "Be" is inserted at the start of the list. The final value of words is therefore ["Be," "very," "very," "good"].


## Divide character strings and merge lists

## Make list from text

The block make list from text uses a delimiter to divide the given text into parts:

| make list from text . C ${ }^{\text {c }}$ | with delimiter ${ }^{\text {c- }}$ |
| :---: | :---: |

In the example above, a new list will be returned containing three segments of text: "311," "555" and "2368".

## Make text from list

This block make text from list assembles a list into a single text using a delimiter:


## Related blocks

## Printing a list

The print block in the text category can output lists. The result of the following program is the console output shown:

| set words$\square$ to + create list with66 <br> 66$\square$ (alpha " <br> " beta " <br> ${ }^{6}$ gamma |  |
| :---: | :---: |
|  |  |
|  |  |

## print

 words .Console

Program starts ...
['alpha', 'beta', 'gamma']

Program finished.

## Complete something for each element in a list

The for each block in the controller category executes an operation for each element in a list. This block, for example, prints each item in the list individually:


The items in this case are not removed from the original list.

See also the examples for break out blocks.

## Map

## JSON

