|  |
| --- |
| Circle Language Spec: Parameters |

## Miscellaneous Parameter Topics

The following articles will explain some miscellaneous parameter topics, that are not part of the main idea of parameters, but are important to explain anyway:

- *Parameters & Arguments*

- *Parameters For The Add Command*

- *Parameters For Objects*

The last concept, *Parameters For Objects*, is not covered here, but is covered in the *System Objects* documentation section.

### Parameters & Arguments

#### Concept

A command contains parameters. *Arguments* are what is filled in into the parameters. Basically arguments are the sources or targets of assignments, that have a parameter as a source or target. When the other side of the assignment, that is not the parameter, is outside the command, then that is the argument.

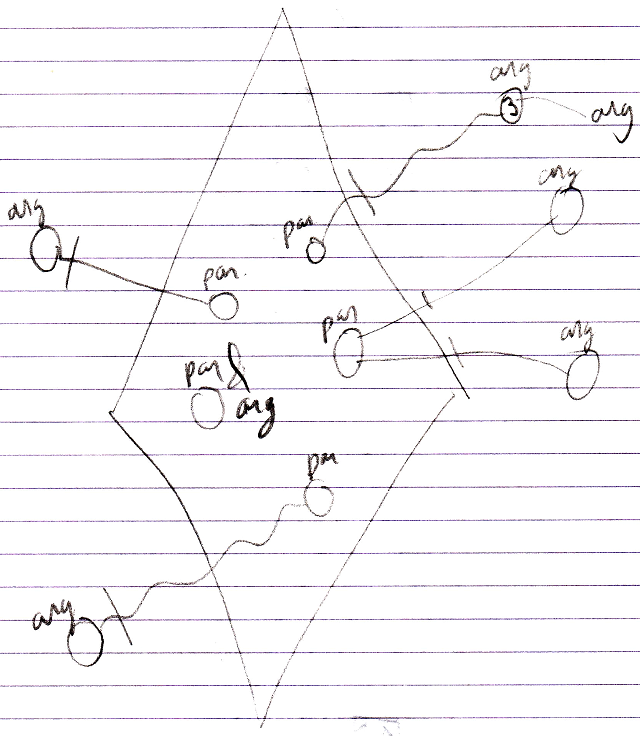
A parameter can be assigned a value. The source object of the value is then called the argument. Also, the value itself could be callled the argument, in contrast to the *source* of the value. A parameter can also be pointing to an external object. The object pointed to is the argument. Sometimes a parameter is not filled in at all, but is a created object by default anyway. In that case, the parameter is its own argument. When a parameter is pointed *at* from the outside, the pointer to the parameter is the argument. When a value assignment sources a parameter and yields over the value to another object, then the other object is the argument.

So an argument is what is filled in as the parameter, what is pointing to the parameter, or the object on an external end of an assignment, the other end of which is the parameter.

Since an argument can be a pointer to a parameter, and a parameter can be pointed to multiple times, a single parameter can have multiple arguments.

#### In a Diagram

The article *Parameters & Arguments* conceptually explained the usage of the term argument in relation to parameters. The current article gives examples of the diagrammatic expression of parameters and their arguments.



### Parameters For The Add Command

The Add command is a system command of a list. It can be called without any parameters. This will add a new item to the list.

But there are multiple ways to provide parameters to the Add command, that may come in handy.

These will simply produce different overloaded variations of the Add command.

First of all, the Add command could return the object, that was added, when the list automatically creates objects when adding a position to the list.

Secondly, when objects are not automatically created, then the *item position* in the list can be returned.

But those are just the basic Add overloads.

Some lists will automatically create an object when you call the Add command. But to occasionally prevent that, you can pass an existing object to the Add command, that will be put in the new position in the list. This will keep the list filled with created objects, without having unnecessary creation of new objects, that are immediately destroyed when replaced by an existing object.

Another application of parameters for the Add command, is providing *attribute values* as parameters in the call to the Add command. A standard Add command could be generated, that includes any attribute as an optional parameter. Then the caller of the Add command can choose which ones to fill in.

It is all up to the one that programs the behavior of the list.

Perhaps the designer of the list decides, it is required that you fill in attributes when adding an object to the list. He can then make those attribute values required parameters of the Add command, and make the other Add overloads inaccessible.

A set of default Add overloads could be generated as such by supporting the Add concept.

Perhaps object initialization on Add could be realized already by a neat, inline object initialization syntax in text code, and you would not need to make each attribute a parameter of the Add command. You would just need the Add overload, that takes a new object as a parameter, and use inline object initialization syntax.