Lobjects- Max objects.
by Peter Elsea
188 Music Center
University of California, Santa Cruz
Santa Cruz CA 95064 USA
elsea@cats.ucsc.edu

The Lobjects were born in a composition class. I was teaching the students how to use probability in Max algorithms. The table object works well for this, and I told my students to convert the list of possible notes to a table. Someone raised his hand and asked "how?" The resulting explanation took about 5 minutes and used up all of the rather small white board. I realized during this experience that it is difficult to use Max to teach general issues because we keep getting distracted by trivia.

The point of the Lobjects is to provide a set of high level abstractions that are powerful and obvious- obvious meaning that when you look at a patch and see something like Ladd, you understand it's something to do with adding lists. By looking at the rest of the patch, you probably can figure out how, even without using the help file.

There are some general rules I'm trying to follow here:

Lobjects should do something that takes more than three or four standard objects to do. (That's no longer true in all cases, partly because the standard set has grown with new versions of Max.) If it's covered by another third party object that is widely known, I won't copy it either.

Lobjects are mostly about lists, but they should do something useful with any input. Bangs usually just re-send the last output though. In order to recalculate with new right input, I'd have to store an extra 2 k of data. That adds up pretty fast. (Well, now that we all seem to have 128 Meg in our machines, I'll stop worrying about it and add recalculating to the next revision. Meantime, there's the Llist object.)

Arguments in Lobjects are initializers. There are no modes, except in a couple of cases where the presence or lack of arguments changes an object's behavior. (Unlist with no arguments unwinds a list once and bangs to ask for more. Unlist with arguments just cycles through the given list until something new comes in. Why? When I put arguments in unlist, it's because I want a specific series of numbers to repeat.) Lobjects that don't need arguments usually won't complain about extras-- you can stick in a comment if you like.

I hope the Lobjects are bulletproof. I test them as thoroughly as I know how (and I'm a professional software tester) and I ask others to try them before they are put into the main package. Nonetheless, bugs slip through- when they are reported to me, I'll find them, post new versions and publicize my goof on the Max-msp list.

Where did the name come from? Tom Mays suggested it, and I like it because it suggests the list oriented feel of most of them, and because my name, when pronounced properly, sounds like two letters: "L-Z" . Microsoft started using the term after I did.

A complete documentation is given here, and the help files are nearly exhaustive. Feel free to contact me via E-mail if something is unclear or appears to be broken.
pqe

Contents

!

An object to return the logical NOT of the input.

Input

Int 0 returns 1, any other int returns 0.

Float Converted to int (anything > 1.0 returns 1).

List Ints and floats treated as above, symbols return false, except "false" "false" "NULL" and "nil", which return the symbol "true".

Argument

None

Output 0 or 1.

Comment

I just got tired of using expr for this simple function. I use it with toggle boxes a lot.

banger

An object to bang outlets sequentially.

Input

Bang Bangs the next outlet.

Int Bangs an outlet. (The left outlet is 0.)

Set Sets the next outlet to be banged.

Argument

First argument sets the number of outlets. Default is 2.

Second argument sets the direction mode.
0 = left to right
1 = right to left
2 = back and forth

Output Bang.

If triggered by an int in the left inlet, each member of the stored list is rotated to the right by the number of steps set by the int and output. Note that the stored list itself is not changed.

Comment

I wrote this when I wanted to imitate a simple Moog-like sequencer.

bit

An object to separate an int into bits

Input

Int The bit value is shown at the designated outlet.

Argument

With no args, there are 8 outlets, which will show the 8 least significant bits of the input. (lsb at the right) Otherwise there will be one outlet per argument, and the output will be that bit of the input.

Output 1 or 0.

Comment

Useful when decoding sysex messages.

byte

An object to combine separate bits into a single int

Input

Int A 1 applied to an inlet sets the associated bit in the internal byte. A 0 clears the associate bit. Other values are or'ed to that bit position, so a 4 in the rightmost inlet sets bit 2. Input in the left inlet triggers output.

List A list of the form 1 0 0 0 0 is converted into a single int with the first member of the list as the most significant bit.

Clear sets the stored value to 0.

Set Changes the argument value.

Bang Outputs the stored value or'ed with the argument value.

Argument

Int The argument value is or'ed onto the internal value upon output. This allows you to permanently set bits.

Output int

Comment

Good for sysex and Midi Machine Control

L?

An object to perform the conditional expression from the C programming language.

Input

List Applied as if each member went to a successive inlet

Int or float

The $in and $fn tokens in the argument determines the use of ints or floats in the same manner as in expr. Data in the left inlet triggers the result, even if it is not used.

Arguments

The syntax of the conditional in the C programming language is

conditional_test ? evalute_if_true : evaluate_if_false
it's equivalent to
if(conditional_test) evalute_if_true else evaluate_if_false;

This is combined with the syntax of expr to give a very concise definition of a complex result. The main feature of this is the evaluation of expressions in the object itself, unlike Max's if which can only output canned messages.

The third expression is optional.

Output

If the first expression is true (any non zero value), the result of the second expression. If the first expression evaluates to 0, the result of the third expression, if there is one.

Comment

I've always wanted to make a T-shirt that says "Real C programmers use ? :"

L==

An object to compare two lists

Input

List In Left inlet: The input list is compared member by member with the stored list. If some values match the number of matches divided by the length of the input list is output. Symbols in either list are wild cards and always match.

In Right inlet: List is stored.

Int or float

In left inlet: The stored list is searched for the value. If found, a 1 is output, if not a 0 is output. Int or floats will not match wild cards.

In right inlet: value replaces all members of stored list.

clear Sets all members of stored list to 0.

Arguments

None: Stored list is initialized to all 0s.
One: All members of stored list are initialized to arg type and value.
More than one: Values initialize stored list (32 args maximum).

Output

If matches are found, the number of matches divided by the length of the input list will be output as a float. If the input was an int or float, a 1 will be output as an int if the value is anywhere in the stored list. If it is not found, a 0 is output.

Comment

This is a fuzzy concept- two lists can be sort of equal. Note that if you run the results through a number box, you will get the crisp logic function, 1 only if the lists are identical. To compare parts of lists, look at Lmatch and Like.

Label

The great prepender.

Args

Int or symbol. One int sets number of inlets (to 32) and the labels for each are 0 through the arg minus one. Two ints set a range of labels, and the appropriate number of inlets. Three or more ints or symbols each attach to an inlet.

Input

Bang The label associated with the inlet is output.

Anything else The label associated with the inlet is prepended to the input and the list output.

set x y z The labels for the first inlets are replaced with the arguments to the set command.

Output

List of label and input with anything other than bangs, which produce the naked label. Symbols of the type 00h (see Lhex) are converted to ints.

Comment

This works well with route. You can use a single line (or send) to move a lot of data around. It's also useful with Lsx.

Labs

An object to find the absolute values of all members of a list.

Input

List The list is scanned, any negative value is changed to positive.

Output

List The absolute values of all members of the input list.

Comment

Another simple function- actually I hardly ever do this with a list, but I need abs of ints a lot.

Laccum

An object to add a series of lists, member by member.

Input

List — In left inlet: Each member of list is added to corresponding member of stored list. Resulting list is output and stored.

In right inlet: Each member of list is added to corresponding member of stored list. Resulting list is stored without output.

Int — In left inlet: Value is added to each member of stored list and list is output.

In right inlet: Value is added to each member of stored list.

Float — In left inlet: Value is added to each member of stored list and stored list is output.

In right inlet: Value is added to each member of stored list, with output

bang — Outputs the stored list.

clear — Sets all members of stored list to 0.

flush — Outputs the stored list and sets stored list to 0.

Arguments

None: Stored list is initialized to all 0s
One: All members of stored list are initialized to arg type and value.
More than one: Values initialize stored list (maximum of 32 args).

Output

List — length of output list matches most recent input list.

Note that the type of each member of the output list is independent, and will be int only if all values added to that member have been int. A int added to a float is converted to float. Once a member has become a float, it will stay float until a clear is received.

Ladd

An object to add two lists, member by member.

Input

List — In Left inlet: Each member of list is added to corresponding member of stored list. Resulting list is output.

In Right inlet: List is stored.

Int — In left inlet: Value is added to each member of stored list, and resulting list is output.

In right inlet: value replaces all members of stored list.

float — In left inlet: Value is added to each member of stored list, and resulting list is output.

In right inlet: value replaces all members of stored list.

bang — Repeats last output.

clear — Sets all members of stored list to 0, sets length of stored list to 1.

Arguments

None: Stored list is initialized to all 0s

One: All members of stored list are initialized to arg type and value.

More than one: Values initialize stored list (maximum of 32 args).

Output

List — If triggered by a list, length of output list matches input list. If stored list is shorter than input list, extra values are passed unchanged. If triggered by a constant, length of output list matches length of stored list.

Larray

An object to store thousands of floats.

Input

Int — In left: If nothing has been received in the right inlet, the value stored at the location indicated will be output as a float. If an int or float has been received in the right inlet, that value will be stored at the location indicated. If a list has been received at the right inlet the members of the list will be stored sequentially, starting at the location indicated.

List — In left: The second value in the list is stored at the location indicated by the first value. All other members ignored.

Int, float, list — In right: Will be stored when an int is applied to the left inlet as described above.

get n x — n values, starting from location x, will be output as a list of floats. If x is omitted the values will be retrieved from the location following the last one accessed. If n is omitted, the number of values retrieved remains the same.

write name n — The first n values will be saved in a file called name. If n is omitted, the entire array is saved. If the name is omitted, the Save As... box will appear.

read name — The contents of the file name will be copied into the array. If name is omitted, the Open dialog will appear. Note that any Max file can be opened, but only those saved from an Larray object will be properly formatted.

Argument INT The capacity of the array. (required) This is the number of floats that may be stored. Note that the memory usage for longs or floats is four times this number.

I, f or b Set storage to ints, floats or bytes. Data will be converted to this type. Byte type takes up one quarter of the storage, but may only represent numbers from 0 to 255. Bytes are output as ints.

Output

Ints, Float, or list. Larray is designed to be bulk storage for tasks such as computing long strings of chaotic output ahead of need or doing computation from DNA codes. It is essentially a one dimensional array, but it may be used as a multi dimensional array by processing the indices with expr.
Larray[128][] would be produced by expr $i1 + $i2 * 128

Comment

This is a generic data to file routine. For something tailored to Sysex dumps, see Ldumpster.

Later

An object to hang on to an int and spit it out later. (Think of it as a big bucket with only one output.)

Input

Int — In left: the input will be held and output after specified number of successive inputs have been received. It only works with ints. If you want to delay lists, store them in a coll and delay the address.

In right: sets the delay interval. The limit is 1024.

Bang — produces output but enters a space in the data, so there will be no output when the delay interval is up.

Output

int — The input of several cycles ago.

Comment

I just needed this one day. It makes delay lines that are sensitive to changes in tempo.

Lave

An object to find the average of the non zero values in a list.

Input

List The list is scanned, the values are summed and divided by the number of non zero values.

Output

Float The average value of the list, disregarding 0's

Lbag

An object to maintain a list of integers.

Input

List — The members of the list are stored or erased according to the setting of the right inlet.

Int — In left: stored or erased, according to the setting of the right inlet.

In right: A zero sets erase mode- anything received in the left will be erased if it is stored. A non zero value sets store mode.

clear — erases the entire stored list.

Output

List — A list of all currently stored values, sorted in ascending order. If Lbag is empty, the right outlet bangs.

Comment

It's the right outlet I'm interested in here. There's no way to know if a bag is empty- you just don't get anything.

Lbang

An object to tidy up windows a little

Input
bang Outputs the stored list.

Double-click also outputs the stored list

Arguments
One: All members of stored list are initialized to arg type and value.

Output
List

Comment

This just provides a defined output when its banged or the patcher is loaded. It saves a message box, which occasionally make a difference in a crowded window.

Lbits

An object to list the bits of an integer.

Input

List Only the first element of the list is processed.

Int In left inlet, the bits of the integer are output in a list
In right inlet, sets the number of bits displayed.

Float ignored.

bang Nada

Arguments

Length of list.

Output

List The input integer as a list of individual bits.

Comment

This was a request. It's the complement to Byte.

Lbondo

An object to send complete lists upon change of any data

Input

List — In any inlet: behaves as individual values in that and the following inlet. Note that a list in the last inlet will extend the list.

Int, Float or Symbol — In any inlet: triggers output of the list with the new value. There is a thresh effect, so closely spaced changes are accumulated before the list is sent out.

bang — Outputs the stored list.

clear — Sets all members of stored list to 0 without output.

Put A B places the value A at location B and outputs the list.

Arguments

First: Number of inlets.
Second: thresh time- a cluster of nearly simultaneous inputs will only cause one output.
Third: Length of list (can be longer than number of inlets)

Output

List

Comment

This was built long before pak appeared, Pak is faster when only one inlet changes, but pak gives a list out for each change, even if they are triggered on the same event.

Lbot

An object to find lowest valued members of a list

Input

List — In left inlet: List is searched for lowest N values. If list is shorter than N, 0s are removed and the list passed through.

Int — In right inlet: Sets N

bang — Repeats last output.

sort — Will cause list to be sorted by value.

Arguments

None: number of values to find
If arg is 0, "plateau finding" is enabled. This reports the center of a group of identical lowest values, or, if the group is at the beginning or end of the list, reports the end or start of the group respectively (as a float). It's a fuzzy logic thing.

Output

2 Lists — lowest values from right, positions of the values from left.

Comment

I used Ltop for years before someone asked "why not Lbot".

Lbuf

An object to store a list.

Input

List — In left inlet: The stored list is output and the new list is stored.

In right inlet: The list is stored without output.

Int — In left inlet: The value of the referenced member of the stored list is output.

In right inlet: Value replaces all members of stored list.

Float — In left inlet: Converted to int and the value of the referenced member of the stored list is output.

In right inlet: Value replaces all members of stored list.

bang — Outputs the stored list.

clear — Sets all members of stored list to 0.

restore — Sets the stored list to the initialization string.

Arguments

None: Stored list is initialized to all 0s
One: All members of stored list are initialized to arg type and value.
More than one: Values initialize stored list (maximum of 32 args).

Output

List — When a list is input at the left inlet, the previous list is output. This is really a Bucket for lists.

Value — When a constant is input at the left inlet, the value of the member referenced is output.

Comment

This is actually based on a piece of hardware called a register. It synchronizes data that may not come in the way you want it.

Lbuild

An object to gradually build and tear down lists

Input

Anything in either inlet is tacked onto the end of the list- if from left, output of the list so far is triggered.

bang Outputs the list from the left outlet. If the list is empty, the center outlet will bang.

clear empties the list.

delete n removes item in position n. Delete 0 will remove the first. Delete n, m will remove m items. The value(s) deleted will be sent out the right outlet.

drop removes the final item from the list. Drop m will remove m items. The values dropped will be sent out the right outlet.

get n reports the value at position n. get n m will report m values, if they exist. Requesting items past the end of the list will bang the center outlet.

next Reports the next value, cycles through the list. Next n reports n values. The next counter is rest by bangs or get commands.

Arguments

The initial list

Output

Right removed values.
Center bangs if empty list is requested.
Left the current list.

Comment

This was written with a specific piece in mind, but seems generally useful.

Lbuildset

Sets specified members of a list to stored value. In other words:
[0 2 5] is transformed to [1 0 1 0 0 1].
This is cumulative so
[1 4] would change the list to [1 1 1 0 1 1].
The list is not output until a bang is received.

Input

List Internal list members indicated by input list are set to stored value with no output. If input is out of range, it is wrapped around to fit into the list. In other words, a -1 sets the last member of the list.

Int In left inlet: Sets one member of the list as above.

In middle inlet: Determines value that will be placed in list. If zero, value currently in list will be incremented.

In right outlet: Determines length of output list.

float In left inlet is treated as int.

bang, dump Outputs list.

clear Sets all members of list to zero.

Arguments

first Determines length of output list. Default is 12.

second Determines value to be placed in set members. If 0 or no arg, the current value in a specified position will be incremented, keeping a count of the number of times a particular value is received.

Output

List Members indicated in input list are set to stored value or incremented. This process is cumulative until clear is received.

Comment

This converts lists of notes into pitch sets.

Lcatch

Collects data into lists, delimited by a pause.

Input

List Gathered into a larger list, which is output after a pause in the data stream or when specified length is reached.

Int In left inlet: Gathered into a list, which is output after a pause in the data stream or when specified length is reached.

In right outlet: Sets length of pause in milliseconds.

float In left inlet: Gathered into a list, which is output after a pause in the data stream or when specified length is reached.

Symbol x The symbol specified is added to the list. May be followed by a list of symbols.

bang Causes immediate output of the list so far.

Arguments

One Length in milliseconds of pause required to trigger output.

two The first is the pause length the second the maximum length of output lists.

Output

List Input data organized into constant or variable length lists.

This object is useful for gathering data that comes in clumps, like chords.

Comment

A long time ago, thresh didn't work—or it did work, but the documentation hadn’t caught up. I use thresh now, unless I need the size limiting feature.

Lcent

An object to find the weighted center of a list.

Input

List The list is processed and the weighted center is output.

Output

Float The position corresponding to the weighted center, or "center of gravity" of the list. Calculated by finding

$$\sum i*m[i] / \sum m[i].$$

Comment

This is commonly used to "defuzz" a complex fuzzy set, yielding a solution that best characterizes the set.

Lchange

An object to filter repeated data or lists.

Input

List The list is compared to the last input in the left inlet. If it is different in any way, it is passed out the left outlet. If it is a repeat, it is sent out the right outlet.
A list in the right inlet sets the comparison buffer. The next input will be compared with this.

Int or float Passed out the left if different, the right if a repeat.

Lock The message lock 1 traps the most recent input into the comparison buffer. All incoming lists will be compared to that.

Reset If the reset message is received, the next input will be passed out the left.

Output

List A new item will be sent out the left, a repeat out the right. Note that the first item of the day is always sent out the left.

Comment

Like the ordinary change object, but works with any kind of input.

Lchunk

Breaks a long list into a lot of little ones, or collects data into lists. Can break by length or after a specified marker.

Input

List — List is output as a series of lists no longer than the specified length or terminated by the specified marker. Although the maximum size for an output list is 256, an input list may be of any length.

Int — In left inlet: Gathered into a list, which is output when specified length is reached or after receipt of a marker.

In right outlet: Sets maximum length of output list.

float — In left inlet: Gathered into a list, which is output when specified length is reached or after receipt of a marker.

Symbol x — The symbol(s) specified is added to the list.

bang — Causes immediate output of the list so far.

markon n — Sets the end of list marker. May be any value or symbol. If no marker is indicated, the previous marker is reinstated.

markoff — Turns off breaking by mark.

Arguments

One — Length of output lists, no marker specified.

two — The first is the end of list marker, the second the maximum length of output lists.

Output

List — Input data organized into constant or variable length lists.

Comment

This object is useful for gathering data streams or large lists into manageable size lists. It really works well with text input—trigger the break on the return character.

Lclose

An object to find values from a template that are close to requested value.

Input

List — In left inlet: Each member is treated as an int and result is output as a pair of lists.

In right inlet: is stored as template.

Int — The template is searched for the member closest in value (a tie reports the first member). The value from the template is output from the right, and the position of the winning value in the template is output from the left.

Float — Same as int.

bang — Searches for last requested value in new template.

Arguments

All members of template are initialized to arg type and value.

Output

2 Lists — Value of nearest match from right, position in template from left.

Comment

Good for forcing notes into a scale.

Lcomp

An object to subtract two lists, member by member. The input list is subtracted from the stored list. (Complement)

Input

List — In Left inlet: Each member of list is subtracted from corresponding member of stored list. Resulting list is output.

In Right inlet: List is stored.

Int — In left inlet: Value is subtracted from each member of stored list, and resulting list is output.

In right inlet: Value replaces all members of stored list.

float — In left inlet: Value is subtracted from each member of stored list, and resulting list is output.

In right inlet: Value replaces all members of stored list.

bang — Repeats last output.

clear — Sets all members of stored list to 0, sets length to 1.

Arguments

None: Stored list is initialized to all 0s

One: All members of stored list are initialized to value.

More than one: Values initialize stored list (maximum of 32 args).

Output

List — If triggered by a list, length of output list matches input list. If stored list is shorter than input list, extra values are passed unchanged. If triggered by a constant, length of output list matches length of stored list.

Types of output list members are float unless the corresponding members of the input list were both int.

Comment

This is a lot handier than you might think- Lcomp 12 does pitch class inversion, for instance.

Lcond

This is the same as L?, provided for compatibility with windows machines, which will not accept ? in a file name.

Lcount

A simple counter that behaves rather like C code or some hardware.

Input

Bang — In Left inlet: Outputs count value and increments the count. If count becomes equal to or greater than end value, right outlet bangs, count value is output, and the count is reset to the start value. On the next bang, the start value is output and the left outlet bangs.

Int — In left inlet: sets value of the next count.
In other inlets: Sets increment, start value or end value.

float — Converted to ints unless there are float args.

reset — Makes start value the next count.

List — next count, increment, start, end.
increment, start, end.
start, end.

Arguments

None: Count will go to 2147483647
One: end value, start value will be 0
Two: start, end.
Three: increment, start, end.
If the end argument is a float, the count will be in float mode.

Output

Left — bang at start over
Middle — the count
Right — bang at count end

Counting proceeds from start value to end value -1. Bangs are provided both with the final count and the restart to simplify synchronization with other objects. For instance, if you want the counter to end at the top value, use the right bang to stop incoming bangs. If you are cascading counters for something like a clock display, use the left bangs.

Count is always calculated by adding the increment to the current count value. This can cause the count to proceed backwards with a negative increment. When underflow occurs while counting backwards, the count value is reset to end value minus increment. That way positive and negative increments give same count patterns. You can also count backwards by setting the end value lower than the start value. This will give slightly different results from negative increments.

If the arguments to Lcount are floats, counting will be in float mode. All values will be interpreted as floats, and when the end value is exceeded, the count value

is wrapped into the count range instead of being reset to the start value. This can give complex count patterns.

Since the existing count value is output before it is incremented, the first number out of Lcount is the start value. An int in the left inlet resets the count value, so that will be output on the next bang, even if it is outside the count range. On the next count the count value will be reset if it is beyond the end value, but counting will continue toward the end value if the out of range number is below the start value.

Comment

This is based on experience with the good old 7493 counter chip. And a lot of experience with counter, too.

Lcut

Passes a list with all members below a threshold value set to 0.

Input

List — List is output with members set to zero if their value is not equal to or greater than stored value.

In right inlet: List is stored.

Int — In left inlet: Stored list is searched, and a list is output that has all members lower than input set to 0.

In right outlet: Is stored as threshold for processing list applied to left inlet.

float — is treated as int.

bang — Repeats last output.

Arguments

One — initializes threshold

List — Initializes list to be tested.

Output

List — A list in which all members are greater than a set threshold or 0.

Comment

This is the "alpha cut" function used in fuzzy set operations.

Ldiv

An object to divide one list by another, member by member.

Input

List In Left inlet: Each member of list is divided by the corresponding member of stored list. Resulting list is output.

In Right inlet: List is stored. If any member of the divisor list is zero, the corresponding member of the dividend list is passed through.

Int or float

In left inlet: Value is divided by each member of the stored list and resulting list is output.

In right inlet: value replaces all members of stored list.

bang Repeats last output.

clear Sets all members of stored list to 1.

Arguments

None: Stored list is initialized to all 1s.

One: All members of stored list are initialized to arg type and value.

More than one: Values initialize stored list (32 args maximum).

Output

List If triggered by a list, length of output list matches input list. If stored list is shorter than input list, extra values are passed unchanged. If triggered by a constant, length of output list matches length of stored list.

Types of output list members are float unless the corresponding members of the input list were both int.

Ldumpster

An object to manage system exclusive dumps.

Input

int If in ready mode, ints are placed in memory as they come in.

ready Clears Ldumpster and prepares it to receive data.

done Stops data reception and reports number of ints received at right outlet.

dump Sends out all data as individual numbers. If the value 247 is encountered, dump will halt after it is sent out.

continue After 247 has halted dump, continue picks up from where it started. If Ldumpster is out of data, continue will do nothing.

peek x n Reports n values starting at location x. This output is a list.

poke s a b c d Stores values a b c d starting at location x.

clear empties Ldumster.

write opens a file dialogue for saving data.

read opens a file dialog for loading data. If read <name>, the file is opened with no dialog.

Arguments

One Sets size of memory. Default is 1024.

b, f, l Sets data type as byte, float, long. If data type is byte, values are truncated to no larger than 255.

Output

The data in Ldumpster can be accessed number by number with peek or in bulk with dump. Peek gives a list of requested values, dump is a series of numbers suitable for sending straight to midiout. The dump stops when 247 (F7h) is encountered in the data stream- this is to make it work with synths that need a pause between blocks of data.

Comment

This is actually optimized to work with Emu products.

Ledge

An object to find transitions in a list.

Input

List positions of mark values preceded by a space value are reported as a list. It is assumed that a space precedes the list.

Int a mark following a space will cause a 1 to be out put.

Arguments

First is space value, second is mark value. Defaults are 0 for space and 1 for mark.

Output

List of positions of marks that follow spaces.

Value 1 if an int mark was received after a space in a list or as an int.

Comment

This is good for detecting the edges in a bit field. Note that values that are not marks or spaces are neither, and it is the space-mark transition that is found, so space-other-mark will not be reported.

Leftgate

An object to steer data around.

Input

Anything in left inlet is shunted out the active outlet.

Int in right selects the active outlet. There is always an active outlet. The left outlet is 0.

Arguments

One: sets number of outlets.

Second: sets inlet that is active at load time.

Output

Whatever.

Comment

If you really need an "off" just define an inlet and don't hook it up. The significant difference between this and gate is control from the right. That simplifies patches where the incoming data is examined to determine what to do with it.

Legato

An object to control articulation of notes.

Input

Int In inlets from right to left: sets overlap, duration, velocity, pitch. When pitch is received a note on pair (pitch, velocity) is sent from the outlets as in makenote. A number of milliseconds later (set by duration) a note off pair (pitch as before, velocity = 0) is sent. If a new pitch is received before duration has elapsed, the note off pair will be sent out overlap milliseconds after the new note was received. If overlap is 0, the note off for the existing note is sent first.

If a pitch is repeated while a note is sounding, a note off is output before the pitch is restarted.

If velocity is set to 0 and a pitch received, the note off is sent immediately.

list behaves as ints in individual inlets, except velocity (the second element of a list) is used for the note but not stored.

off stops most recent note.
flush stops all notes.

Outputs

Ints for velocity (from right) and pitch (from left).

Comment

This object is designed to model performance on a wind instrument or single string where only one pitch sounds at a time. Switching the overlap time from 0 to 1 will change the behavior of some synthesizers, which will skip the attack phase or switch on portamento only when notes overlap. Longer overlaps will give controlled transitions to sounds with slow attacks.

Lexpr

An object do series expansion on lists. The members of the list are processed sequentially and a result is computed that can include other members of the list or the results of previous computations.

Arguments

Syntax is very much like expr, except there are special symbols to indicate list members to operate on.

in, fn indicate the current member being evaluated. The symbol in means int, fn is float.
in-1, fn-1 are previous member of list; also in-2, fn-2, in-3 and fn-3.
iy-1, fy-1 are previous results (member of output) back to iy-4.
iC, fC are constants received in right inlet.

Input

list For each member of the list a result is computed according to the rules of expr. For the first member of the input list, predecessors are taken to be 0. If only iy or fy terms are present, the first members of the list are passed through (enough to satisfy the most distant y term) and used as seeds for the following terms.

int, float are treated like an iterated list, and the results are output individually.

bang A new in value is extrapolated from the difference between the previous inputs (or last 2 members of an input list) and a result calculated.

seed When individual numbers are input, there is no starting history to calculate y terms from. The seed message sets the values that are used for this history. The seed values are passed to the output.

Noseedout Prevents seed values from being passed to the output. Cancelled with seedout.

Output

The output will be a list or series of numbers that carry out the expansion indicated.

Comment

This is useful for setting up patterns or fractal sequences.

Lfilt

An object to remove selected values from a list.

Input

List — Members whose value did not match the stored values will be output in a list.

In right inlet: List is stored as values to filter out.

Int — In left inlet: Will be passed through if it does not match a stored value.

In right outlet: Is stored as value to reject.

float — is treated as int.

bang — Repeats last output.

Arguments

List — Sets values to filter out.

Output

List — Input list with matched values removed is passed out the left outlet. If all values are rejected, the input is passed out the right outlet.

int — If an input is an int, it will be passed out as an int.

float — Likewise.

Comment

One trick I like to do is feed the output into the right inlet. That makes it a change type object that gives you a right output to ask for another try. (see help file for example)

Lfind

An object to search a list for a specific value and give its position, treating the list as a continuous function.

Input

List In Left inlet: The list is searched for a member matching the stored value. If match is found, the position of the first matching member of the stored list is output from the left outlet. If not, an interpolated position is calculated from the member of the list nearest but below the stored value and the next or previous member higher than the value.

In Right inlet: List is stored.

Int or float

In left inlet: The stored list is searched for the first occurrence of the value. Output is as above.

In right inlet: value is stored.

clear Sets all members of stored list to 0.

Arguments

None: Stored list is initialized to all 0s.
One or more: Values initialize stored list.

Output

Float If match is found, the position of the first element in the input list within the stored list is output from the left outlet. If no match is found, an interpolated position is calculated from the member of the list nearest but below the stored value and the next or previous member higher than the value.

The list should be monotonic, either rising or falling, but not both. Values in a "valley" will be found if matched exactly, but interpolated results are calculated from the peaks. A list may contain zeros between members and appropriate results will be calculated. If a list contains leading or trailing zeros, the zero closest to a non zero member will be used as a base for calculations.

Comment

This is a fuzzy logic thing, the inverse of inference. If you want to find out if a number is contained in a list, use Lmatch.

Lformat

An object to replace members of lists

Input

List In Left inlet: The first member of the list is a index to the member of the stored list to change, the second is what to put there. Data cannot be placed beyond the end of the stored list. The first position in a list is position 0.

In Right inlet: List is stored.

Int or float

In right inlet: new value to put in the list

In left inlet: location in list to put value. Negative value and values beyond the list end will be ignored.

restore Sets all members of stored list to original values (the arguments).

Replace x y Substitutes all occurrences of x with y. (Replace by value.)

Bang Modified list is output.

Arguments

Values initialize stored list.

Output

List

Comment

This is a little like pack, except that it works with longer lists and isn't fussy about types. It's a bit like table, but works with floats and symbols.

Lhex

An object to display numbers in a list as hexadecimal.

Input

List — All ints in the list are converted to hexadecimal symbols of the type 00h. values greater than 255 are truncated. Floats are truncated to ints and symbols are passed through.

Int, float — Are converted to hex symbols.

Arguments

none

Output

List of symbols — Display with prepend set and a message box.

Comment

Why? Although hexadecimal numbers can be displayed by number boxes or entered in the 0x00 format, they are converted to decimal format when sent to a message box, so there is no way to display lists of hex numbers. This easily leads to mistakes when working with system exclusive codes, which are usually documented in hex.

See also Lsx, Llong, Label, and Like.

Lhigh

An object to report the highest valued members of a list as a set.

Input

List The list is scanned, and a list is output in which all but the highest n values are set to 0.

Int In left inlet: Tested as if it were a list.

In right inlet: Sets number of values to report. If this is larger that the length of the list, the list is passed through with all non zero values set to 1.

Arguments

first: Sets number of values to report.

Output

Left List with all but the highest n values set to 0. If there are no non zero values, there is no output.

Comment

This is useful in fuzzy logic. I use Laccum to sum up the result of a lot of tests and Lhigh to show the winner.

Lifo

An object to store items in a list, last in first out.

Input

Any In Right: The input is stored in front of any stored items.

In Left: the input is prepended to anything stored, and the stored list is output and cleared.

Clear The stored list is cleared without output.

Output

List Everything that has come in, in reverse order. The internal list is cleared.

Comment

It's really a bit like pack, except it will combine lists and other things, and you don't have to rewire when you change the order or number of items you expect.

Like

An object to sort lists by their beginnings.

Input

List In left inlet: The list is compared with the stored list; if the stored list matches the beginning of the input list, the input list is passed out the left outlet. If not, the input list is passed out the right outlet. Symbols in the input list are wild cards.

In right inlet: the list is stored.

Int In left inlet: Tested as if it were a list.

In right inlet: Stored as a one item list.

Arguments

Initialize stored list. Arguments may be hex in the form 00h. (See Lhex, Lsx). Other symbols are wild cards.

Output

List The input list is passed out the left if the header matches the stored list or out the right for further testing.

Comment

Sysex messages always have a defined header followed by variable data. Like makes it easy to build parsing trees, where different messages are routed to Lswaps, which extract the relevant values.

Lin

An object to gather data into a list and show progress.

Input

Ints, floats, or lists in left inlet are gathered into the stored list and the list is output from the right outlet. When the list reaches a preset length or the preset mark value is received, the list is output from the left outlet, the stored list is cleared, and an empty list is sent out the right outlet.

Int in right inlet sets break length of stored list.

back deletes last item from stored list and outputs list from the right.

Markon n Sets n to be the mark value or symbol that causes list to be output from the left. This will be included in the list.

Markoff turns off mark feature. Lists will only be output when they are full length.

Arguments

One: Sets break length of stored list.
Two: first is mark value, second length of stored list.

Output

List The accumulated data from the right, and the final version from the left.

Comment

Lin is designed to simplify data input from a keyboard. As data is received, it is appended to a stored list, and that list is output from the right, presumably for display. The back command gives a backspace function. When the list is full length or the mark value is received, an empty list is sent out the right to clear the display, the stored list is cleared, and the complete list (including mark value) is output from the left. You could easily implement a command line interface from Lin.

Linc

A simple increment/ decrement object

Input

inc	the stored value is incremented and sent out.
dec	The stored value is decremented and sent out.
Int	In left inlet: is stored and sent out. In right inlet: sets step size.
Float	in left is converted to int. Float in right is ignored.
Set n	Stores n as next value to output.
clear	returns Linc to initial value.

Arguments

one:	Sets step size. Initial value defaults to 0.
Two:	Initial value, stepsize.
Third:	low limit.
Fourth	high limit.

Output

Int	The first inc or dec after the patcher is opened, or after clear or set, will output the initial value. After that, inc will add the step to the value and output it. Dec will subtract the step size from the internal value and output it. If high and low limits are set, the output will not go past them.

Comment

I like Chris Muir's IncDec object, but I always want something like it that can be triggered by other process and gives an initial value.

Why does it give 0 on the first click? Incrementing and decrementing imply there is some value to be incremented, but until there's some output, that value is indeterminate (Dr Schrodinger, I've found your cat). In practical terms, there are objects downstream that need to know. They can be individually initialized (by a lot of loadbangs probably), or the control object can provide an initial value. If you don't like making the user click on something once to prime the pump, provide a loadbang to Linc.

Linfer

An object to report the interpolated value implied by a fractional position. The value is calculated from the nearest non zero values above and below the position.

Input

List — In Left inlet: An output is calculated from the list using a stored value as the position.

In Right inlet: List is stored.

Int or float

In left inlet: An output is calculated for that position using the stored list

In right inlet: value is stored.

clear — Sets all members of stored list to 0.

Arguments

One: Sets stored position.
More than one: Values initialize stored list.

Output

Float — If the position queried has no fractional part, and the member has a non zero value, that value is output. If the position has a fractional part or contains a zero, the list is searched for non zero members before and after the position. The output is then interpolated between these values.

If the position falls in a region of leading or trailing zeros, a zero is output. If the position is beyond the end of the list, the end value is output.

If the list is all zeros, there will be no output.

Comment

This is a method of extracting the membership value from a fuzzy set (inference). Ignoring intermediate zeros keeps you from having to calculate values for those positions yourself (that's the computer's job, right?) If your set includes intermediate zeros, use Linterp.

Link

An object to create lists by appending them to a header.

Input

Anything In right is appended to internal list without output.
In left is appended to internal list and the concatenated lists are output.
The internal list is cleared.

Bang in left causes output of list and clears internal list
In right causes output of list without clearing internal list.

Set n n .. changes the list header.

Arguments

Set header of list. All lists output will begin with this.

Output

The output is a list starting with the args and including each subsequent item in the order they arrived.

Comment

I use link instead of join when I don't know anything about a list except how it will start and end. The middle stuff usually comes in front to back rather than back to front as expected by join. It's also useful to be able to get a list in progress, especially if you want to display it.

Linterp

An object to report the interpolated value implied by a fractional position. The value is calculated from the nearest values above and below the position, even if those values are zero.

Input

List In Left inlet: An output is calculated from the list using a stored value as the position.

In Right inlet: List is stored.

Int or float

In left inlet: An output is calculated for that position using the stored list

In right inlet: value is stored.

rotate n Rotates the stored list n steps to the right.

clear Sets all members of stored list to 0.

Arguments

One: Sets stored position.
More than one: Values initialize stored list.

Output

Float If the position queried has no fractional part the value is output. If the position has a fractional part, the output is interpolated between the values before and after the position.

Comment

This is a fuzzy thing, but it will make complex envelopes if you drive it with Lcount. It's really two types of object rolled into one- since it needs a list and a constant to work, I made it so you can store the list and apply the constant, or store the constant and apply the list. The two functions are independent of each other.

Linvert

An object to divide one list by another, member by member.

Input

List In Left inlet: Each member of list is divided into the corresponding member of stored list. Resulting list is output. If any member of the divisor list is zero, the corresponding member of the dividend list is passed through.

In Right inlet: List is stored.

Int or float

In left inlet: Value is divided by each member of the stored list and resulting list is output.

In right inlet: value replaces all members of stored list.

bang Repeats last output.

clear Sets all members of stored list to 1.

Arguments

None: Stored list is initialized to all 1s.

One: All members of stored list are initialized to arg type and value.

More than one: Values initialize stored list.

Output

List If triggered by a list, length of output list matches input list. If stored list is shorter than input list, extra values are passed unchanged. If triggered by a constant, length of output list matches length of stored list.

Types of output list members are float unless the corresponding members of the input list were both int. In this case, floats are truncated.

Comment

With no arguments, all members of a list will be inverted in the mathematical way. This isn't pitch class inversion; for that use Lcomp.

Llast

Produces a list of the last n items received.

Input

Any	In left inlet- There is no output until n items have been received. Then data is output as a series of lists of length n, each containing the last n items. Members of lists are treated as individual items.
Int	In right outlet: Sets length of output list.
bang	Causes immediate output of the list so far, or repeats the last output.

Arguments

One	Length of output lists.

Output

List	Input data organized into list of last n items.

Comment

This object is useful with either Lmatch or L== for detecting patterns in incoming data streams.

Llatch

An object to synchronize external and on screen UI controls. An ordinary slider attached to a control in will cause drastic jumps in a value if that parameter is set by some internal mechanism such as a UI object or preset. Llatch contains the code to require that the slider be matched to the internal value before a change occurs. There is also a line like feature to produce smooth transitions when presets change.

Input

Int — An int in the right inlet clears an internal latch bit and 0 is sent out the right outlet. The value is saved and passed out the left outlet. (See the effect of time below.)

An int in the left inlet is compared with the saved value. If they are the same, or there has been a change of above to below or below to above, the latch is set (with a 1 output from the right outlet) and the value is passed. As long as the latch remains set, data received in the left inlet will be passed through.

List — The first item in the list is considered, and the remaining items tag along.

Time — A time message sets the transition time for data changed by the right inlet. If the left input is matched to the latch value while the data is changing, the transition will stop and output will follow the left input.

Argument

An int or float argument sets the initial match value and determines the format of the output.

Output

Right -- the state of the latch
Left – current data

Comment

Look at the help file to see how to set this up.

Llength

An object to report the length of lists

Input
List the length is sent out the outlet.
Int or float a 1 is sent out the outlet.
Anything else: the number of arguments following symbols are reported.

Arguments
None

Output
The length of input, or the number of arguments to a command.

Comment

Originally, there was third party external called len that did this, but it stopped working at version 4. I added llength at that time. Now zl len is built-in, so llength is provided only to keep old patches running.

Llimit

An object to accumulate lists clipped to limits.

Input

List In left inlet: is added to internal list (member by member) and result is output. If result of addition exceeds upper or lower bounds, member is set to the bound.

In right inlet: processed without output.

Int or float are ignored

Addto n x adds x to member n with out triggering output.

bang Outputs the stored list.

clear Sets all members of stored list to 0.

flush outputs the list, then clears it.

Arguments

Set lower and upper bounds.

Output

List

Comment

This makes a list behave like a row of sliders, with defined tops and bottoms.

Llist

An object to store a list. Rather like an int object.

Input

List — In left: the list is stored and passed through. In right, it is simply stored.

Set n x — Sets member n to value x, if there are n or more members of the list. Set n alone sets member n to 0.

Bang — reports the stored list.

Clear — sets all members of stored list to 0.

Arguments

Initialize stored list.

Output

List: if there is only one member, type is appropriate, int or float.

Comment

Why not use a message box? For one thing it takes up less space. For another, you can have more than 128 items in overdrive.

Thanks to Trond Lossus for suggesting this.

Llogic

An object to emulate Boolean logic gates.
nand

Input	Input2	Output
0	0	1
0	1	1
1	0	1
1	1	0

and

Input	Input2	Output
0	0	0
0	1	0
1	0	0
1	1	1

nor

Input	Input2	Output
0	0	1
0	1	0
1	0	0
1	1	0

nor

Input	Input2	Output
0	0	0
0	1	1
1	0	1
1	1	1

xor

Input	Input2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Not inverts the input.
Lists are processed as if they filled all inputs.

Arguments

Symbol required: determines function: nand, and, nor, or, xor.
Second argument sets number of gates.

Comment

I used to do a lot of engineering with 7400 ttl and 4000 series CMOS integrated circuits. Sometimes I just enjoy thinking that way.

Llong

An object to convert hex symbols to ints

Input

List — Any members of the list of the format 00h will be converted to the equivalent int. Floats in the list will be converted to ints. Ints and other symbols will pass through unchanged.

Output

List of ints.

Comment

See also Lhex. The 00h format symbols are useful for working with hex strings, since Max converts ints of 0x00 format to ints when it saves patches.

Although Lhex only works on 8 bit values 00h to FFh, Llong will convert 32 bit items like FFFFFFFFh.

Lmask

An object to predefine certain members of a list

Arguments

Arguments define the mask. Any int in the mask will occupy that space in the output. A symbol in the mask will be replaced by something from the input.

Input

List In left: Unmasked elements will be passed through, masked ones replaced with the mask values. Members beyond the mask will be passed through.

In right: defines a new mask.

Int In left: replaces all symbols in the mask.

In Right: sets the mask to a single number.

Anything Any command with arguments will have the mask applied to the arguments, and will be passed through.

Output

List

Comment

This is useful for creating versions of a list that vary somewhat.

Lmatch

An object to search a list for the first occurrence of another list

Input

List In Left inlet: The stored list is searched for a series of members matching the numbers in the input list. If match is found, the position of the first matching member of the stored list is output from the left outlet. If no match is found, the input list is sent out the right output. Any symbol in a list is a wild card, but a list may not start with one.

In Right inlet: List is stored.

Int or float

In left inlet: The stored list is searched for the first occurrence of the value and the position is output from the left outlet. If not found, the value is output from the right outlet. Constants will not match wild cards in the stored list.

In right inlet: value replaces all members of stored list.

bang Repeats last output from right outlet.

clear Sets all members of stored list to 0.

wrapon If wrap is on, a list that wraps around the end of the stored list will be found.

wrapoff If wrap is off, a list that wraps around the end of the stored list will not be found.

Arguments

None: Stored list is initialized to all 0s.

One: All members of stored list are initialized to arg type and value.

More than one: Values initialize stored list.

Output

int If match is found, the position of the first element in the input list within the stored list is output from the left outlet.

Comment

Input from users prompted me to add Lsearch, which is probably better than this or Lpos in most applications.

Lmatrix

An object to stack lists and extract lists of specified members.

Input

List The first member of a list indicates which row to store it in. The rest of the list gets stored. If the list is shorter than the specified row size, the end members are set to 0.

put stores list in a column. First member of the list selects the column.

Int retrieves a column as a list (i.e. 4 returns the 4h member of each list).

getRow n returns the list in slot n.

getColumn n returns column n as a list.

set r c v sets member c of row r to v.

get r c returns the value of row r member c.

clear Sets all members of stored lists to 0.

Arguments

Set number of rows (down) and number of columns (across).

Output

List or int

Comment

Lmatrix only works with ints. It is nothing like a jit matrix.

Lmax

An object to compare two lists member by member and produce a list of the high values.

Input

List In Left inlet: Each member of list is compared to the corresponding member of stored list. A list of the high value of each pair is output.

In Right inlet: List is stored.

Int or float

In left inlet: The value is compared to the members of the stored list, and a list of the higher value of each pair is output.

In right inlet: value replaces all members of stored list.

bang Repeats last output.

clear Sets all members of stored list to 0

Arguments

None: Stored list is initialized to all 0s.
One: All members of stored list are initialized to arg type and value.
More than one: Values initialize stored list.

Output

List If triggered by a list, length of output list matches input list. If stored list is shorter than input list, extra values are compared to 0. If triggered by a constant, length of output list matches length of stored list.

Types of output list members are the same as the high member of each pair. If an int and float are equal, type follows input list.

Lmean

An object to calculate the mean values between two lists, member by member. The process is (x[i] +y[i])/2

Input

List In Left inlet: Each member of list is processed.

In Right inlet: List is stored.

Int In left inlet: Stored list is processed with constant.

In right inlet: value replaces all members of stored list.

float Same behavior as int.

bang Repeats last output.

clear Sets all members of stored list to 0, sets length of stored list to 1.

Arguments

None: Stored list is initialized to all 0s

One: All members of stored list are initialized to arg value.

More than one: Values initialize stored list.

Output

List All members of the output list are float.

Comment

If you need to compute the average across several lists, use Laccum and Ldiv.

Lmerge

An object to mix lists

Input

List In Left inlet: A new list is created, with members taken in alternation from the input and stored lists.

In Right inlet: List is stored.

Int In left inlet: A new list is created, with the number firsts, then inserted between each member of the stored list.

In right inlet: value replaces all members of stored list.

float Same behavior as int.

interleaf n m enables interleaving, where n items are taken from the input list, then m from the stored list, and so on.

clear removes stored list son inputs pass unchanged.

Arguments

None: Stored list is initialized to all 0s

One: All members of stored list are initialized to arg value.

More than one: Values initialize stored list.

Symbol a symbol of the type 2:3 (with no spaces) will set initial interleaving. Arguments after this symbol initialize the stored list.

Output

List The newly combined list. Note that the length of the output list is controlled by the length of the input.

Comment

There were originally aliases to lmerge titled Lmerge3, lmerge 4, on to lmerge9. When these are used, every third member, every fourth member…every ninth will be taken from the stored list. To make this mechanism work, lmerge should be installed in the Max-Startup folder. This is still supported, but may go away soon.

Lmin

An object to compare two lists member by member and produce a list of the low value of each pair.

Input

List In Left inlet: Each member of list is compared to the corresponding member of stored list. A list of the low value of each pair is output.

In Right inlet: List is stored.

Int or float

In left inlet: The value is compared to the members of the stored list, and a list of the low value of each pair is output.

In right inlet: value replaces all members of stored list.

bang Repeats last output.

rotate n Rotates the stored list n steps to the right.

clear Sets all members of stored list to 0

Arguments

None: Stored list is initialized to all 0s.
One: All members of stored list are initialized to arg type and value.
More than one: Values initialize stored list.

Output

List If triggered by a list, length of output list matches input list. If stored list is shorter than input list, extra values are compared to 0. If triggered by a constant, length of output list matches length of stored list.

Types of output list members are the same as the high member of each pair. If an int and float are equal, type follows input list.

Comment

This is a fuzzy intersection.

Lmost

An object to report the most common item in a list.

Input

List The list is examined, and the most common member value found. The number of time this occurs is output from the right and the item is output from the left.

Int or float
Just passed through.

Arguments

None:

Output

Right Number of repetitions of the most common member.
Left The item, symbol, float or int.

Comment

This was a requested object.

Lmult

An object to multiply two lists, member by member.

Input

List In Left inlet: Each member of list is multiplied by the corresponding member of stored list. Resulting list is output.

In Right inlet: List is stored.

Int or float

In left inlet: Each member of stored list is multiplied by value, and resulting list is output.

In right inlet: value replaces all members of stored list.

bang Repeats last output.

clear Sets all members of stored list to 1

Arguments

None: Stored list is initialized to all 1s.
One: All members of stored list are initialized to arg type and value.
More than one: Values initialize stored list.

Output

List If triggered by a list, length of output list matches input list. If stored list is shorter than input list, extra values are passed unchanged. If triggered by a constant, length of output list matches length of stored list.

Types of output list members are float unless the corresponding members of the input list were both int.

Comment

If your lists are restricted to 0 or 1 as values, this will give you the logical AND function or the traditional (crisp) intersection.

Lnot

This is the same as !, included for compatibility with Windows systems, which do not allow punctuation in filenames.

Lnote

A multichannel version of make note.

Input

Int — In inlets from right to left: sets default channel, duration, velocity, pitch. When pitch is received a note on (pitch, velocity, channel) is sent from the outlets as in makenote. A number of milliseconds later (set by duration) a note off (pitch and channel as before, velocity = 0) is sent.

If a pitch is repeated while a note is sounding, a note off is output before the pitch is restarted.

If velocity is set to 0 and a pitch received, the note off is sent immediately.

list — interpreted as pitch, velocity, duration, channel. The list may be shorter than 4 items, or a symbol may be substituted for any item except pitch. In that case the default (most recently received int) is used.

flush — stops all notes.

Arguments set initial defaults for velocity, duration, and channel.

Outputs

Ints for velocity, pitch and channel. There is a limit of 512 pending note offs. If this number is exceeded, the oldest notes are cut off.

Comment

I wrote this because of the annoying way makenote turns repeating notes into long ones and short ones if you get going too fast—at least with Emu products. This also reduces hung notes caused by changing channels at noteout.

Lnth

An object to let every nth item through.

Argument
Sets n.

Input
In left Every nth item is let through.

Int in right: sets n. 0 stops everything, 1 passes everything.

Output
Some of what came in.

Comment

This a simple switch that I like to use for generating rhythm patterns. (it could be used as a gate that defaults to open).

Loop

An object to safely control iterated processes.

Input

Int In left inlet: Value sets start value and starts loop.

In middle inlet: sets start value without starting the loop.

In right inlet: Value sets end point of loop.

Float treated as int.

bang In left inlet: starts the loop. The starting value will appear at the left outlet.

In right inlet: steps through the loop, a new value appearing at the left outlet with each bang. When the ending value is reached, the right outlet will bang. (The end value is not output.)

list sets start and end values.

start Starts loop from beginning.

stop Stops the loop, bangs the right outlet, and resets pointers.

pause Stops the loop without resetting pointers.

continue Picks up the count where it left off.

Arguments

None: Start and end value will be 0.
One: Sets end value. Start will be 0.
Two Set start and end values.

Output

Ints from start value to one less than end value from left outlet, then bang from right. These actions are deferred to a low priority so the right inlet bangs may be fed back from the end of the process chain with no danger of stack overflow. If feedback is used, the loop will execute as fast as possible given other demands on the system.

Lpack

(Obsolete) A slightly modified version of pack. The number of inlets are determined by the number of arguments or by the value of a single argument. Data received in each inlet replaces the corresponding argument in the stored list. Data in the left inlet or a bang will trigger list output. **After output, all values are reset to the arguments.**

Input

List	In any inlet: members of the list replace corresponding members of the stored list, as if received in inlets counting from the input. The length of the output will be no longer that the arguments allow, however.
Int	In any input, replaces corresponding member of stored list.
float	Same behavior as int. Note that types in output list match new input.
Symbols	Same behavior. A symbol in the first position creates a command message.
bang	Outputs list and resets the stored list to initial arguments.
Set	changes the remembered values that list will revert to. Note that this can extend stored list beyond number of inlets.
revert 0/1	disables/enables reversion to initial arguments.
autobang 0/1	disables/enables output on receipt of left input.

Arguments

None:	Stored list is initialized to two 0s.
One:	sets number of inlets. Internal list is all 0s.
More than one:	Values initialize stored list.

Output

List

Comment

This was deleted in 2004, with no complaints.

Lpad

An object to generate lists of predetermined length.

Input

List — In Left inlet: A list is generated of specified length, with extra places filled with padding if necessary.

Int

In left inlet: Triggers output, a list starting with the int and padded to the length.

In second inlet. Sets Place in output to insert input. (In other words the list will start with this number of padding values.)

In third inlet: Padding value. Default is 0.

Right inlet: length of output list. Default is 12

bang — Repeats last output.

clear — Sets all members of stored list to 0

reset — Sets all members of stored list to Padding value

Arguments

One: Length.
Two: Padding, length.
Three: Place, padding, length.

Output

List of specified length, with extra 0s or other padding value to fill out if required. If a place value is specified, that many padding values will fall at the beginning of the list.

Lpair

Iterates and synchronizes two lists.

Input

List — List is output one member at a time from Left outlet. Stored list is output one member at a time from right outlet. If input list is longer than the stored list, the last member of the stored list is repeated as necessary.

In right inlet: List is stored.

Int — In left inlet: Value is output from left outlet. A member of the stored list is output from the right outlet. When all members of the stored list have been used, the list starts over.

In right outlet: Is stored as a list of one member.

float — is treated as int.

Arguments

List — Initializes stored list.

Output

int or float — The members of the stored and input list are output from the right and left outlet respectively.

Comment

In theory, this will replace about half of the Lobjects. Actually, I only use it in conjunction with expr for complicated processes. The behavior when the lists are not the same length may seem bizarre, but it gives you control over what happens in such cases. If you want the short list to cycle, put it in the left. If you want it extended with the last value, put the short one in the right.

Lpast

A threshold detector with hysterisis. There is a target and a reset point. The first number that is equal to or higher than the target will produce a bang. No further bangs will be produced until a number equal to or lower than the reset point has been received. If the target is lower than the reset point, all senses are reversed.

Input

List Treated as individual numbers.

Int
float In left are tested
center sets target
right sets reset point

Reset same as receiving reset value

Arguments

Target, reset point. A single arg sets both points the same. Default is both to 0.

Output

Bang

Comment

This is very useful for detecting values in a noisy data stream.

Lpeak

Reports the highest member of a list

Input

List in left inlet: List is scanned and position and value of highest value in list are reported.

Int in right inlet, sets a reporting threshold. If a threshold has been set, lists will not trigger output unless the peak value is above this threshold.

Arguments

None no thresholding
One sets threshold

Output

Int Out right: value of highest member
Out left: position of highest member

Comment

This is simplified Ltop, and works reliably with negative numbers.

Lperc

An object to play notes with velocity and duration preset according to pitch. This object is designed to control synthesizer patches that feature different sounds on each pitch.

Input

Int — In inlets from left to right: sets pitch, default velocity, duration, and rate. When pitch is received a note on pair (pitch, velocity) is sent from the outlets .Duration milliseconds later a note off pair is sent. The default velocity and duration may be replaced by individual values for each pitch.

List — A list in the left inlet will play a note. First element sets pitch, second sets velocity, third sets duration. If velocity or duration are missing or 0, the individual or default values are used. (To shut off a note, use mute.)

roll — with arguments for duration, pitch, velocity, note duration, and rate. Plays the pitch repeatedly every rate milliseconds, for duration milliseconds. If velocity, note duration, and rate are missing or 0, the individual or default values are used. Any changes to individual or default values during the roll will take effect with the next note of the roll. Only one roll may play at a time.

tremolo — with arguments for duration, and pitches. The list of pitches is played in rotation at the default rate. Velocities and durations are taken from individual or default values. Only one tremolo may play at a time.

up, down, updown — messages set the direction of the tremolo. Up reads the list forwards. These messages may take an argument list for a tremolo to play.

stop — stops roll or tremolo, letting notes ring.

bang — replays the last pitch, roll or tremolo.

mute — with no argument stops all sounding notes. May take an argument list to shut off selected notes. If a selected note is part of an ongoing tremolo, the tremolo will stop, but only the selected note is muted.

Individual Settings for Lperc
Double click on the object box to open a window to edit velocity, duration and roll rate for each pitch. Enter one command per line, followed by a semicolon. You will be prompted to save your settings when you close the window. You may also set values with messages.

vel n x sets individual velocity for pitch n to value x.

dur n x sets individual duration for pitch n to value x.

rate n x sets individual roll rate for each n to value x. This is the time in milliseconds between repetitions. If this is shorter than the individual duration, all notes but the last will be shortened. Tremolos always use the default rate or the rate argument to the tremolo command.

link n1 n2 sets a link so that playing n1 will mute n2. The classic example of this is to have a hi-hat closed sound mute a hit–hat open sound.

These messages affect individual settings:

linkoff n clears a link for pitch n.

clear removes all individual settings and links.

read loads a file containing individual settings. May take a filename as argument. All existing individual settings are cleared when a new file is read. The defaults are changed only if the file contains values for defVel, defDur, or defRate.

Arguments
Ints set default velocity, duration, and roll/tremolo rate.

Filename is name of individual settings file to be loaded when patcher is opened.

Output
Ints for velocity (from right) and pitch (from left). If a pitch is repeated before it is finished playing, a note off is sent before restarting.

Lpoly

An object to control polyphony

Input

Int Left inlet: Pitch of note.

Center inlet: velocity of next note.

Right inlet: number of voices.

bang Repeats last output.

clear Sets all members of stored list to 1

Arguments

One: Number of voices.

Output

When a pitch is received, a list is output in the format voice number, pitch, velocity. If velocity is 0, the voice number is the same as last used for this pitch. When all voices are playing and another note is received, the oldest note playing is turned off, unless it is the lowest note playing.

Comment

This is a common approach to the "rip-off" problem in commercial products. A low note cut off is pretty noticeable but inner ones are less likely to be missed. This is most useful with msp instrument type patches.

Lpos

An object to report an item's position in a list. It complements Lmatch.

Input

Left inlet: Item (list, int or float) to search for

Right inlet List to search

Wildcard x sets any value or symbol as a wild card. If * is a wild card, both 1 0 3 and 1 2 3 will match 1 * 3.

bang Repeats last output.

clear Sets all members of stored list to 1

Arguments

List Initialize search target.

Output

When item is received in the left inlet, the stored list is searched for matching. If a match is found, the position of the item in the searched list is output from the left. If it is not, the input is passed out the right.

Comment

Input from users prompted me to add Lsearch, which is probably better than this or Lmatch in most applications.

Lpow

An object to raise the members of a list to the powers set in another list.

Input

List In Left inlet: Each member of list is raised to the power of the corresponding member of stored list. Resulting list is output.

In Right inlet: List is stored.

Int or float

In left inlet: The value is raised to the power of each member of stored list and resulting list is output.

In right inlet: value replaces all members of stored list.

bang Repeats last output.

clear Sets all members of stored list to 1

Arguments

None: Stored list is initialized to all 1s.
One: All members of stored list are initialized to arg type and value.
More than one: Values initialize stored list (32 args maximum).

Output

List If triggered by a list, length of output list matches input list. If stored list is shorter than input list, extra values are passed unchanged. If triggered by a constant, length of output list matches length of stored list.

Types of output list members are float unless the corresponding members of the input list were both int.

If the result is undefined, such as 0^0 or the root of a negative number, the left value is passed through, and a warning is posted to the MAX window.

Comment

This is used in fuzzy logic as a hedge.

Lpref

An object to manage preference files for patchers. It is a table of 256 ints.

Input

List	in left: Second element is stored at location indicated by the first In right: will be stored sequentially starting at location specified by next int in left.
Int	In left inlet: returns value at that location, or if int has been received in right, will store that value. In right inlet: data to store
get n	returns next n items as a list
get n x	returns next n items starting at location x
read	will attempt to load the named file from the system preferences folder. If the file does not exist, it will be created with the contents of lpref.
write	will attempt to write the data to the named file. If the file does not exist, -43 will be output from the right and the file will not be created.
clear	Sets all data to 0.

Arguments

Required: a symbol to name the file. A non zero second argument will make the file invisible.

Output

Int	from left: data requested From right: results of file operations. 0 means no error.

Comment

This is most useful with collectives and stand alone applications. You can store some values here and let the end user set them. The help file shows how to use this for mild copy protection.

LQueue

Slows down a stream of ints

Input

List List is output one member at a time from left outlet. The rate of output is determined by the argument or rate setting. If more lists are received, their data is output in turn.

Int In left inlet: Value is output from left outlet. Successive ints are output at a steady rate. Up to 1024 ints may be accumulated for output. When ints have been output, the right outlet bangs.

In right outlet: sets the output rate.

float Is treated as int.

Arguments

Int Rate-- time between outputs in microseconds.

Output

Ints- This is different from speedlim, in that no data is lost. Queuing is a standard programming technique that lets you convert data that comes in bursts into a manageable flow.

Tip – if you want to queue lists, store them in a coll and queue the addresses.

Comment

I use this in situations where a lot of MIDI output is being generated and I'm concerned with overloading instruments.

Lreg

A register type object, useful for keeping track of sounding notes.

Input

Int — In left inlet: if value is from 0 to 127, it is stored or forgotten as controlled by the right inlet. The contents of the object are then output as a list. If there are no contents (i.e. this was the last forget command) the right outlet bangs.

In right outlet: 0 means forget the next number, non zero means store it.

clear — Deletes all values without output.

Arguments

Int — Sets mode- if there is no argument, a note is deleted on the first note off, if the argument is 1, a value has to be deleted as many times as it has been received.

Output

List of current values from the left outlet or bang from the right outlet.

Comment

This is such a common problem it deserves its own object. You can do this with Lbag (or a lot of other ways) but this is optimized for speed.

Lrem

An object to divide one list by another, member by member and output a list of the remainders.

Input

List — In Left inlet: Each member of list is divided by the corresponding member of stored list. A list of remainders is output. A float is processed as an int, but the fractional part is added onto the result.

In Right inlet: List is stored. Floats are converted to ints. Zeros are converted to 1.

Int — In left inlet: Value is divided by each member of the stored list and resulting list is output.

In right inlet: value replaces all members of stored list.

float — In left inlet: processed as an int, but the fractional part is added onto the result.

In right inlet: Converted to int by truncation.

bang — Repeats last output.

clear — Sets all members of stored list to 1.

Arguments

None: Stored list is initialized to all 1s.

One: All members of stored list are initialized as ints.

More than one: Values initialize stored list (32 args maximum).

Output

List — If triggered by a list, length of output list matches input list. If stored list is shorter than input list, extra values are passed unchanged. If triggered by a constant, length of output list matches length of stored list.

Lrepeat

Generates lists of repeated patterns.

Input

List A list is output that consists of the input list repeated the specified number of times.

Int In left inlet: A list consisting of the int repeated is output.

In right outlet: sets number of repeats. Numbers less than 1 have no effect.

Arguments

Int Sets number of repeats. Default is 2.

Output

List The input is repeated as specified.

Lreplace

An object to modify values in a list

Input

List — In left inlet: Any member of the stored list that matches the first member of the input list is replaced by the rest of the input list.

In right inlet: List is stored. Maximum size is 256.

Int n — triggers the output of the nth member of the stored list.

In right outlet: Is stored as a list of one member.

Bang — outputs list as modified.

Delete n — deletes all occurrences of n in the stored list.

Restore — returns stored list to arguments.

Arguments

List — Initializes stored list.

Output

List — The stored list as modified. This object is good for expanding patterns, since you can replace a single member with a list.

Lror

An object to rotate all members of a list. Members falling off the end of the list are placed at the beginning.

Input

List — In left inlet: Each member of list is rotated to the right by the set number of steps.
In right inlet: List is stored.

Int — In left inlet: triggers output of stored list rotated n steps to the right. A negative number rotates list to the left. n is stored as number steps for next rotation.
In right inlet: Sets number of steps to rotate any list applied to left inlet.

clear — Sets stored list to a singleton of length 12. (A singleton is a list of the type 1 0 0 0 ....) Sets rotation to 0.

length n — Sets the length of the stored list to n.

bang — Stored list is rotated again.

Arguments

None: — Rotation (number of steps to rotate a list received in left inlet) is set to 0. Stored list is set to singleton of length 12.

One: — Sets rotation for any list received in the left inlet. A negative number rotates the list to the left. Stored list is set to singleton of length 12.

List: — Sets stored list. Rotation is set to 0.

Output

List — If triggered by a list in the left inlet, each member of the list is rotated to the right by the number of steps set by a single argument or an int in the right inlet.

Lround

Rounds off members of a list to desired precision.

Input

List Each member of list is rounded off to precision specified by argument.

Int, float Value is rounded off to precision specified by argument.

bang Repeats last output.

Arguments

None: Input values are rounded off to int.

One: Sets number of digits to right of decimal place. Negative number sets rounding to powers of ten. If any arguments are present all output values will be floats

Output

List List of values rounded off to desired precision. The rule of rounding values ending in 5 to even digit is followed.

Lrun

Generates lists of consecutive numbers

Input

List A list of at least two numbers will produce an output list where the values run from one number to the other.

Int In left inlet: A list is generated that runs from the starting point to the int.

In center: sets starting value, default is 0.

In right: sets increment, default is 1. With increments other than 1, end value may not fit series- it is included in the output anyway. The absolute value of the increment is used-- the counting direction is determined by the start and end points.

float is treated as int.

bang produces ints that continue the series.

Arguments

Ints Start, increment.

Output

Lists of ints that run from one value to another.

Lsame

An object to test two messages to see if they are identical

Input
Anything: In left inlet: test this

In right inlet: compare with this

Arguments
Sets initial message.

Output
Int 0 means no match, 1 is a match.

Comment
Tests not only value, but type and length of message.

Lscale

An object to adjust range of input values in a linear fashion.

Input

List In left inlet: All members of list are re-scaled

Int In left inlet: value is re-scaled.
Float truncated to ints unless any arguments are float

Other inlets set: low end of input range
High end of input range
Low end of output range
High end of output range

set With four arguments changes the ranges

limit Limit 1 turns limiting on, restricting output to specified range.
Limit 0 turns limiting off (default)

Arguments

low end of input range: default 0
High end of input range: default 127
Low end of output range: default 0.0
High end of output range: default 1.0

Output

The outputs are scaled and offset to fit proportionally into the output range as they fit the input range. In other words, 25% of input gives 25% output. If output range arguments are float, the output will be float. If the sense of input or output range is reversed, output is inverted.

Comment

This is similar to the scale object I wrote some years ago for Karl Essl's Real Time Composition Lab.

Lscaler

An object to scale values in a list to individual ranges.

Input

List In left inlet: All members of list are re-scaled.
In right, is stored as input maxima for each position in the a list.
The default value is 127.

Int or float In left inlet: value is re-scaled for each argument and output in a list.
In right inlet, all input maxima are set to value.

set Sets the output maxima, replacing the arguments.

Clear Sets input and output maxima to 127.

Arguments

Set the output maxima, and output data types. Default is 127, which will be applied to the final members of input lists that are longer than the number of arguments.

Output

Each position in a list is scaled so that a certain percentage of input maximum produces the same percent of output maximum. So if input maximum is 127 and output maximum is 1.0, an input value of 64 is scaled to 0.5039.

Comment

This was requested for processing the output of multislider when the sliders need different ranges.

Lsearch

An object to search a list for occurrences of input data

Input

List In left inlet: The stored list is searched for all occurrences of the input
In right: replaces stored list to search.

Int or float In left inlet: list is searched for occurrences of this.

wrap With an argument of 1, searching will wrap around the end of the stored list, so 5 1 2 would be found in the target 1 2 3 4 5.

Arguments

Initialize the stored list.

Output

If the search item is not found, a 0 is output from the left outlet. If the search item is found, a list of the location(s) of the item will be output from the right and a one will be output from the left.

Comment

This is mostly useful as a simple membership test.

Lshiftr

An object to shift all members of a list to the right. The value found at the first position is extended back to the beginning of the list.

Input

List — In left inlet: Each member of list is shifted to the right by the set number of steps.
In right inlet: List is stored.

Int — In left inlet: triggers output of stored list shifted n steps to the right. A negative number rotates list to the left. n is stored as the value for the next shift operation.
In right inlet: Sets number of steps to shift any list applied to left inlet.

clear — Sets stored list to a singleton of length 12. (A singleton is a list of the type 1 0 0 0 ....) Sets rotation to 0.

length n — Sets the length of the stored list to n.

bang — Shifts stored list again.

Arguments

None: — Rotation (number of steps to rotate a list received in left inlet) is set to 0. Stored list is set to singleton of length 12.

One: — Sets rotation for any list received in the left inlet. A negative number rotates the list to the left. Stored list is set to singleton of length 12.

List: — Sets stored list. Rotation is set to 0.

Output

List — If triggered by a list in the left inlet, each member of the list is rotated to the right by the number of steps set by a single argument or an int in the right inlet.

If triggered by an int in the left inlet, each member of the stored list is rotated to the right by the number of steps set by the int and output. Note that the stored list itself is not changed.

Lshiftreg

An object to insert items at the beginning of a list. As numbers come in, the stored list is shifted right and output. The "lost" member is output from the right outlet.

Input

List — In left inlet: is itered, and members are inserted at beginning of the base list as if a series of ints had been received. Note that this will turn the list around. There will be distinct output for each member of the input list.

In right inlet: replaces the stored list and changes length.

Int — In left inlet: Is inserted at the beginning of the stored list, which is shifted right and output. The number cut off the end of the list is output from the right outlet.

In right inlet: changes the length of the stored list. If the list is lengthened, the added locations are initially 0.

Float — In left inlet: same effect as int.

bang — Outputs the stored list.

clear — Sets all members of stored list to 0.

Arguments

One: — Set length of shift register. All members of stored list are initialized to 0.

More — initialize stored list

Output

Right — value dropped from end of list

Left — newly shifted list.

Comment

It a lot like a bucket, but you see the contents.

Lsieve

Passes members of a list that match desired values.

Input

List Members that match any of the stored values are output as a list.

In right inlet: List is stored as values to match.

Int In left inlet: Will be passed through if it matches a stored value.

In right outlet: Is stored as value to match.

float is treated as int.

bang Repeats last output.

Arguments

List Values to match.

Output

List Inputs that match stored values are passed out the left outlet. If all values are rejected, the input is passed out the right outlet.

Int If the input is an int, the output will be an int.

Float Likewise.

Lsign

An object to return the sign of the input value.

Input

List In left inlet: signs of members reported individually.

Int In left inlet: sign reported

Float In left inlet: sign reported

bang Repeats last output.

Arguments

None

Output

An output of 1 indicates above 0, a 0 indicates equals 0, and a -1 indicates less than 0.

Comment

This actually replaces a surprising number of little boxes.

Lsort

An object to sort lists of numbers into ascending order.

Input

List gets sorted by value. Duplicate values are kept. If any member of the list is a float the entire output list will be floats. Symbols are thrown away.

Arguments

None

Output

List sorted in ascending order.

Comment

It's just your basic bubble sort. Works best with random data, worst with reversed lists.

Lspace

An object to move drawings in an LCD. A drawing moves through the defined space with the specified velocity vector as bangs are applied. The drawing will be removed after a predefined number of bangs. What you are getting is actually the position and color for your own drawing routine. [paintrect x,y, x+1, y+1, color]will draw dots.

Input

List	defines a new position for a dot. Values indicate; Lifetime in generations Color (indexed) X velocity-- number of pixels to move right each generation Y velocity-- number of pixels to move down each generation Optional starting x Optional starting y
Int	does nothing.
Float	Scales all velocity values. Acts as master speed control.
bang	creates a new generation.
erase	erases all dots, but they will reappear next generation.
clear	kills all dots.
bcolor	is the (indexed) color that will be used for erase. If it does not match the background of the LCD, the dots will leave streaks.
bounce	(1 0r 0) enables bounce mode, when items will be reflected from edges of space.
gravity	(y x) introduces an acceleration factor for y and x. y and x are floats. Putting y first suggests gravity works downwards. Negative values make gravity work up.
Friction	(float) scales velocities on each generation, causing old dots to slow down. Negative friction will give acceleration.
Origin	(x y) sets the point where dots with 0 starting coordinates will appear.
read	gives a list of x, y, color, for each active dot. Useful for sonification of the display.

Arguments Two required: set boundaries of the space.

Output

Lists	[x y color] there are two for each living dot- the first has previous x y and the background color to erase the screen. The second has new x y and drawing color for the new dot. These can feed drawing instructions and be sent to the LCD.

Comment

Relationships to laws of physics in any actual universe are purely accidental. What is drawn in the LCD is up to you. To draw a simple dot, construct from the above; [paintrect x y x+1 y+1 color]

Lspeak

Speaks lists aloud, using the Mac speech manager and internal sound.

Input

List — List is spoken aloud. Numbers are pronounced as numbers. If another list is received while one is speaking, it is held until the first is finished (up to 16 stacked lists).

Acsii x x — Numbers appended to the ascii command are converted to characters and the text spoken.

Bang — repeats the last speech.

Ptch n. — Sets speech pitch to n. The effect of a given number varies with voice, but it approximates the midi notes expressed as a float.

Rte n. — Sets rate of speech. Normal rate is around 150.00. Rate affects the next speech, not the current one. Also, the pause after a speech is controlled by the rate at the start of speech.

xvoice Fred — sets the voice. You can get a list of valid voices from the speech manager control panel. You can call voices by number, but names are more dependable.

xstop — Stops speech.

xresume — resumes stopped speech.

xreset — restores variables to default for voice (and forgets stopped speech.)

[[embeds]] embedded commands give detailed control over each word (or phoneme) of the speech. The format is [[command arg]] and stays in effect until the next command of the same type occurs. These are documented in Inside Macintosh: Sound (downloadable from appledev.com).
The args are floating point (FIX really) and must be written with a decimal point.

[[pbas nn.000]] — sets basic pitch. Nn = MIDI note. Not all voices follow this, especially not the “singing” ones.

[[pmod nn.00]] — sets the amount of prosody or inflection of the voice.

[[rate nn.00]] — sets the rate of speech.

[[volm nn.00]] — sets the volume. 1.00 is full on. 0.05 is inaudible.

[[char LTRL]] — sets spelling mode [[char NORM]] turns it back.

[[nmbr LTRL]] — reads numbers as digits [[nmbr NORM]] turns it back.

[[inpt PHON]] sets phonetic mode [[inpt TEXT]] turns it back. The phonetic symbols are given in Inside Macintosh: Sound

Arguments

List Initializes stored list. This will be spoken when the object is banged. Remember that punctuation has to get through the Max parser, so precede commas with \.

Output

Speech Each instance of Lspeak creates an independent voice channel, which can give you several voices speaking at once.

Bangs The left outlet bangs more or less with each word. The right outlet bangs when all speech is finished.

Lstring

An object to convert ASCII into messages.

Input

List A list of ints is interpreted as an ASCII string and converted to a message of symbols and numbers.

Int Gives ASCII equivalent.

Float Ignored

breakatspace 0 Normally the ASCII 32 will delimit symbols. With breakatspace off, the output will be one huge symbol, suitable for file operations.

numeralsasints 0 Normally, combinations of the ASCII values from 48 - 57 are converted to ints and floats to produce the mixed messages that Max uses. If numeralsasints or breakatspace is off, these values will come out as symbols.(If the resulting symbols are applied to math operators, you can get some perplexing error messages.)

Arguments

None:

Output

Messages of mixed symbols and numbers, or just symbols

Comment

This is an attempt to bridge the gap between Max and the outside world, which mostly speaks ACSII. Some popular if confusingly named third party externals used to do this, but are now hard to find. Lstring works well with lin.

Lstrum

An object to arpeggiate and mute chords. This object is designed to model performance on a multi-string instrument such as a guitar.

Input

Int — In inlets from left to right: sets pitch, velocity, duration, rate. When pitch is received a note on pair (pitch, velocity) is sent from the outlets as in makenote. A number of milliseconds later (set by duration) a note off pair (pitch as before, velocity = 0) is sent. If a new pitch is received before duration has expired, the old note is muted before the new one is played. The muting is canceled by the "harp" message and reinstated by the "pick" message. In all cases, a repeated pitch will be cut off and reiterated.

List — A list in the left inlet is interpreted as a chord. The pitches in the list are played with a delay between note ons as set by rate. The direction the list is scanned is modified by the up and down messages. A chord that is sounding when a new chord is received will be shut off before the new one is played. A list in any other inlet is stored but not played.

bang — Plays the stored chord or replays the last one.

up, down, updown — messages set the direction of the arpeggio. Up reads the list forwards, down backwards. In updown mode the directions will alternate. These messages may take an argument list for a chord to play.

harp — turns off muting for notes played by single ints.

pick — turns individual muting on again. These messages may take a single argument as a note to play in the new mode.

mute — with no argument stops all sounding notes. May take an argument list to shut off selected notes.

Output

Ints for velocity (from right) and pitch (from left).

Lsub

An object to subtract two lists, member by member.

Input

List In Left inlet: Each member of stored list is subtracted from corresponding member of list. Resulting list is output.

In Right inlet: List is stored.

Int In left inlet: Each member of stored list is subtracted from value, and resulting list is output.

In right inlet: value replaces all members of stored list.

float In left inlet: Each member of stored list is subtracted from value, and resulting list is output.

In right inlet: value replaces all members of stored list.

bang Repeats last output.

clear Sets all members of stored list to 0, sets length of stored list to 1.

Arguments

None: Stored list is initialized to all 0s

One: All members of stored list are initialized to arg type and value.

More than one: Values initialize stored list (maximum of 32 args).

Output

List If triggered by a list, length of output list matches input list. If stored list is shorter than input list, extra values are passed unchanged. If triggered by a constant, length of output list matches length of stored list.

Output list members are float unless the corresponding members of the lists were both int.

Lsum

An object to find the total values of all members of a list.

Input

List The list is scanned and added up.

Output

Float The total value of all members of the input list.

Lsustain

An object to manage note off messages and provide detailed sustain and sostenuto functions.

Input

Int In right and left inlets set velocity and pitch respectively. A pitch received while velocity is non zero is treated as a note on: the velocity and pitch are sent straight to the outlets. If the same pitch is currently sounding, a note off is sent out before the note on. A pitch received while velocity is zero is a note off. A note off will be sent out if the note is currently playing unless sustain is in effect.

List First two items are treated as pitch and velocity.

bang Shuts off all notes, sustaining or not.

sust With no arguments sets all pitches to sustain. Arguments set individual pitches to sustain. When a pitch is set to sustain, any note offs for that pitch are held until the release command.

sost With no arguments, sets all currently sounding pitches to sustain. Arguments set additional pitches to sustain.

release With no arguments, clears all sustains and sends all pending note offs. With arguments, clears sustain on the listed pitches and sends note offs if appropriate.

Output

Ints for velocity (right) and pitch (left). Note offs will only be passed for notes that are currently on. If sustain is on for that note, the note off will be delayed until sustain is released. If a note that is on is repeated, a note off will be set before the note on.

Lswap

An object to rearrange lists.

Input

List In Left inlet: The items in the list are reordered according to the stored template and output as a list.

In Right inlet: List is stored as a template for processing lists in the left inlet. The template is the new order of the indices from the input list. In other words, a template of [0 2 1] would define the output list as first, third, second items of the input list. If values are repeated in the template, the list items will be repeated. If the input list is shorter than any of the indices in the template, there is no output.

Negative numbers in the template mean count back from the end. A template of [0 -1] will produce the first and last items from a list.

Any symbol in the template means "fill in with missing values". A template of [0 * 7] will produce the first 8 items. A template of [-1 * 0] will reverse the list. Symbols may be used more than once. A template of [0 * -1 * 0] will convert a list into a palindrome. Other common operations:

[0]	first
[1 * -1]	butfirst
[-1]	last
[0 * -2]	butlast
[0 * 2 4 * -1]	leave out the fourth
[0 * -1 0 * -1]	repeat the list

If the template ends with a symbol, the implied series is continued. Examples:

[0 2 *] every other item.
[1 3 *] the odd items.
[1 0 3 2 *] exchange every other item.

Arguments specify the initial template.

Output

List The reordered list.

Lswitch

An object to choose messages from various inlets.

Input

Anything in the active inlet is passed on. An int in the rightmost inlet chooses the active inlet.

Arguments

First	sets the number of message inlets
Second	sets the inlet that is initially active.

Output

Whatever.

Comment

It's switch, but there is always an active inlet, and the right inlet is the selector.

Lsx

Creates sysex strings using persistent hex format.

Arguments

The foundation for the sysex string is typed in as the argument list. Three types of entry are possible;

Ints will be part of the output as is

00h symbols of the type 00h are interpreted as hexadecimal values and will be part of the output.

Symbols other symbols are taken as placeholders.

Input

Symbol n the argument appended to a symbol will replace that symbol in the output. If the symbol occurs twice, the argument is nibblized across all occurrences of the symbol (up to four times). The default nibblizing mode is 7 bit Low High.

For instance, with the string F0h 3 xx xx, the message xx 255 would create the string 240 3 127 1.

Nibble n sets the nibble size to n. (n bits per placeholder)

hiLo sets nibilization so the msb occurs first.

loHi sets nibbilization so the lsb occurs first.

clear sets all placeholders to 0.

Bang sends the message as a series of ints.

Output

Ints itered and ready for midiout.

Comment

I wrote this so I wouldn't make so many mistakes in dealing with sysex strings. The documentation for sysex is almost always in hex, and I found I was messing up a lot when Max insisted on showing me decimal. (I also get confused with number boxes, because there's no way to tell if they are showing hex.)

Ltocoll

Formats data for insertion into a coll. Lists are kept intact.

Input

Any In left inlet- data is placed in a message that begins with an address. The addresses increment with each new input.

Int In right inlet- sets next address to use. If address is within range set by arguments, address sequence will continue from that point. If address is below of range, automatic addressing will count up into the range. If address is beyond range, the next item will be placed there, but the following item will go to the beginning of the range.

reset sets next address to beginning of range.

Arguments

one Sets end of address range.

two Set beginning and end of address range.

Output

Left List beginning with address

Right Address.

If the left outlet is connected to the inlet of a coll, data input to Ltocoll will be placed in sequential address in the coll.

Ltop

An object to report the highest valued members of a list.

Input

List — The list is scanned, the highest values are output as a list at the right outlet. The positions these members occupied in the initial list are output in a list at the left outlet. Members of 0 or less are never reported.

Int — In left inlet: Tested as if it were a list.

In right inlet: Sets number of values to report. If this is larger that the length of the list, the list is passed through with all 0s excised, and no sorting is performed.

sort — If this message is received, the list of highest values will be sorted from high to low. If not, the values appear in their original order.

sortoff — Turns sorting off.

Arguments

first: — Sets number of values to report. If missing or 0, and the list has several successive members at the highest value, the midpoint of these members is reported at the right outlet.

second: — If 1, sorting is turned on.

Output

Left — List of the original positions of the n highest positive non-zero values in the input list.

Right — List of values of the reported members from list. If sorting is on, list will be highest to lowest. If sorting is off, values will keep their original order.

Comment

This is essential for evaluating fuzzy operations.

Ltoset

Creates a list with specified members set to stored value. In other words:
[0 2 5] is transformed to [1 0 1 0 0 1]

Input

List — List is output with members set to stored value if their position was indicated in the input list. Values out of range are ignored.

Int — In left inlet: Sets one member of the output list. If value is out of range, first or last member of output list will be set.

In middle inlet: Determines value that will be placed in list.

In right outlet: Determines length of output list.

float — In left inlet is treated as int.

bang — Repeats last output.

Arguments

first — Determines length of output list. Default is 12.

second — Determines value to be placed in set members. Default is 1.

Output

List — Members indicated in input list are set to stored value.

Comment

This transforms a list of pitch classes such as 2 5 9 into a pitch set like 0 0 1 0 0 0 1 0 0 1 0 0.

LtoTab

Convert list to a series of ints or floats, with accompanying indices. Designed to fill a table with the values in a list.

Input

List List is disassembled and sent, item by item, out right output, followed immediately by its position from the left outlet.

Int sent out the right outlet, followed by a zero from the left outlet.

Output

Right Items from list.

Left Ints indicating position in list of item.

If the left outlet is connected to the left inlet of a table or funbuff, and the right outlet is connected to the right inlet of same, a list input to the LtoTab will be stored as the first n elements of the table. An int input will clear that member of the table.

If your lists vary in length, be sure to clear the table before sending the list.

Ltrue

An object to report if a particular position in the list is non zero.

Input

List In Left inlet: The list member at the stored position is tested. If non zero, the left outlet bangs. If zero, the list is passed out the right outlet.

In Right inlet: List is stored.

Int or float

In left inlet: The member referenced in the stored list is tested. If non zero, the left outlet bangs. If zero, the int is passed out the right outlet.

In right inlet: value is stored as position to test in incoming lists.

clear Sets all members of stored list to 0.

Arguments

One: Sets stored position.
More than one: Values initialize stored list.

Output

Bang at left, list or int at right. This object behaves rather like a sel. If the tested position is non zero, the left output bangs, If the tested position is zero, the test stimulus is sent out the right outlet.

Comment

This is simpler than the explanation would make it seem. You would probably either set the position to look at in the object and feed in a list or set a list in the object and query positions. This code will do either independently.

Lunique

An object to remove duplicates from a list.

Input

List The list is examined, and if any items appear more than once, the latter occurrences are removed.

Int In right inlet sets number of repetitions allowed.

Arguments

Sets number of allow repetitions.

Output

List, somewhat simpler than the input. Floats and ints are distinct, even if they have the same value.

Lxor

An object to calculate the exclusive OR of items in a list.

Input

List An exclusive OR of all numbers in the list is calculated and output.

Arguments

None:

Output

Int the exclusive OR of the list. The rules for XOR are:

0 xor 0 = 0
1 xor 0 = 1
0 xor 1 = 1
1 xor 1 = 0
These rules are applied bit by bit, so 10010100 xor 11111110 is 01101010

Comment

Xor is used to calculate checksums in some system exclusive messages.

unlist

output members of list individually.

Input

Int Value is sent out left input and stored.

List List is stored. If unlist has no arguments, first member is sent out left outlet immediately. List may contain ints, floats, and symbols; but may not start with a symbol.

Bang Next member of list is sent out left outlet. If all members of list have been sent, bang is output at right outlet. Next bang received will start over at beginning of list.

access n The nth member of the list is sent out. (The first member is member 0).

set n Sets the internal pointer so that the nth member of the list will be sent on the next bang.

Arguments

Initialize list. The presence of arguments will change the behavior of unlist. An initialized list is assumed to be a set of constants to cycle through repeatedly. An empty unlist object implies that a new list will be supplied for each cycle.

Output

Left Members of stored list.

Right Bang after end of list. If there are no arguments, there will be no output from the left out during the bang cycle unless a new list is received. If there are args, the first element of the stored list will be output from the left outlet immediately after the bang.

Use the right outlet to fetch new lists.