Array : ArrayedCollection : SequenceableCollection : Collection : Object

fixed size collection

Source: Array.sc
Subclasses: Matrix, NFunc, PeerGroup, SystemNFunc

See also: Literals, List

Description

Arrays are ArrayedCollections whose slots may contain any object. Arrays have a fixed maximum size beyond which they cannot grow. For expandable arrays, use the List class.

Literal Arrays can be created at compile time, and are very efficient. See Literals for information.

For handling **multidimensional arrays**, there are specific methods which are covered in the helpfile J concepts in SC.

NOTE: For Arrays, the `add` method may or may not return the same Array object. It will add the argument to the receiver if there is space, otherwise it returns a new Array object with the argument added. Thus the proper usage of `add` with an Array is to always assign the result as follows:

```
z = z.add(obj);
```

This allows an efficient use of resources, only growing the array when it needs to. The List class manages the Array internally, and in many cases is more suitable.

Elements can be put into an existing slot with `a.put(2,obj)` and accessed with `a.at(2)` or `a[2]`

See ArrayedCollection for the principal methods: at, put, clipAt, wrapAt, etc...

Class Methods

Array.**new**(maxSize: 0)

From superclass: Object

Create a new array with size 0 that can grow up to the fixed size.

Arguments:

maxSize The maximum size of the array.

Array.**newClear**(indexedSize: 0)

From superclass: ArrayedCollection

Create a new array with all slots filled with nils.

Arguments:

indexedSize The size of the array.

Array.**with**(... args)

Create a new Array whose slots are filled with the given arguments. This is the same as the method in ArrayedCollection, but is reimplemented here to be more efficient.

```
Array.with(7, 'eight',  9).postln;
```

Creates a Collection of the given size, the elements of which are determined by evaluation the given function. The function is passed the index as an argument.

```
Array.fill(4, { arg i; i * 2 });
Bag.fill(14, { arg i; i.rand });
```

Arguments:

size The size of the collection which is returned. If nil, it returns an empty collection. If an array of sizes is given, the resulting collection has the appropriate dimensions (see: *fillND).

```
Array.fill([2, 2, 3], { arg i, j, k;  i * 100 + (j * 10) + k });
```

function The function which is called for each new element - the index is passed in as a first argument. The function be anything that responds to the message "value".

```
Array.fill(10, { arg i; 2 ** i });
Array.fill(10, Pxrand([0, 1, 2], inf).iter);
Array.fill(10, 7); // an object that doesn't respond with a new
value is just repeatedly added.
```

`Array.fill2D(rows, cols, function)`

From superclass: Collection

Creates a 2 dimensional Collection of the given sizes. The items are determined by evaluation of the supplied function. The function is passed row and column indexes as arguments. See J concepts in SC

```
Array.fill2D(2, 4, 0);
Array.fill2D(3, 4, { arg r, c; r*c+c; });
```

`Array.fillND(dimensions, function, args: [])`

From superclass: Collection

Creates a N dimensional Collection where N is the size of the array **dimensions**. The items are determined by evaluation of the supplied function. The function is passed N number of indexes as arguments. See J concepts in SC

```
Array.fillND([4, 4], { arg a, b; a+b; });            // 2D
Array.fillND([4, 4, 4], { arg a, b, c; a+b*c; });    // 3D
Array.fillND([1, 2, 3, 4], { arg a, b, c, d; b+d; });   // 4D
```

`Array.newFrom(aCollection)`

From superclass: Collection

Creates a new Collection from another collection. This supports the interface for the method "as".

```
Array.newFrom(Set[4, 2, 1]);
Set.newFrom(Array[4, 2, 1]);
[1, 2, 3, 4, 3, 2].as(Set); // as(someClass) calls someClass.newFrom(this)
```

`Array.geom(size, start, grow)`

From superclass: SequenceableCollection

Fill an ArrayedCollection with a geometric series.

```
Array.geom(5, 1, 3).postln;
```

Fill an ArrayedCollection with an arithmetic series.

```
Array.series(5, 10, 2).postln;
```

Array.**iota**(... sizes)

From superclass: ArrayedCollection

Fills an ArrayedCollection with a counter. See J concepts in SC for more examples.

```
Array.iota(2, 3);
Array.iota(2, 3, 4);
```

Array.**interpolation**(size, start: 0.0, end: 1.0)

From superclass: SequenceableCollection

Fill a SequenceableCollection with the interpolated values between the **start** and **end** values.

```
Array.interpolation(5, 3.2, 20.5);
```

Array.**rand**(size, minVal, maxVal)

From superclass: SequenceableCollection

Fill a SequenceableCollection with random values in the range **minVal** to **maxVal**.

```
Array.rand(8, 1, 100);
```

Array.**rand2**(size, val)

From superclass: SequenceableCollection

Fill a SequenceableCollection with random values in the range -**val** to +**val**.

```
Array.rand2(8, 100);
```

Array.**linrand**(size, minVal, maxVal)

From superclass: SequenceableCollection

Fill a SequenceableCollection with random values in the range **minVal** to **maxVal** with a linear distribution.

```
Array.linrand(8, 1, 100);
```

Array.**exprand**(size, minVal, maxVal)

From superclass: SequenceableCollection

Fill a SequenceableCollection with random values in the range **minVal** to **maxVal** with exponential distribution.

```
Array.exprand(8, 1, 100);
```

Array.**fib**(size, a: 0.0, b: 1.0)

From superclass: SequenceableCollection

Fill a SequenceableCollection with a fibonacci series.

```
Array.fib(5);
```

Arguments:

size	the number of values in the collection
a	the starting step value
b	the starting value

`Array.zeroFill(size)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/SignalBox/Classes/extArray.sc

Create a new array with all slots filled with 0.0.

```
Array.zeroFill(8)
```

Inherited class methods

5 methods from SequenceableCollection ▶ show

1 methods from Collection ▶ show

13 methods from Object ▶ show

Undocumented class methods

`Array.fillNoteNames(startNote: "C3", endNote: "B3", step: 1)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Note/extVarious-midiname.sc

`Array.makeScale(groundNote: "C3", type: 'major', startNote, endNote)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Note/extVarious-midiname.sc

`Array.makeScaleCps(groundNote: 261.62556530114, type: 'major', startNote, endNote)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Note/extVarious-midiname.sc

`Array.makeScaleMidi(groundNote: 60, type: 'major', startNote, endNote)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Note/extVarious-midiname.sc

`Array.makeScaleName(groundNote: "C3", type: 'major', startNote, endNote)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Note/extVarious-midiname.sc

Instance Methods

`.at(index)`

From superclass: ArrayedCollection

Return the item at **index**.

The index can also be an Array of indices to extract specified elements. Example:

```
x = [10,20,30];
y = [0,0,2,2,1];
x[y]; // returns [ 10, 10, 30, 30, 20 ]
```

`.put(index, item)`

Put **item** at **index**, replacing what is there.

.insert(index, item)

From superclass: ArrayedCollection

Inserts the item into the contents of the receiver. This method may return a new ArrayedCollection. For this reason, you should always assign the result of `insert` to a variable - never depend on add changing the receiver.

```
(
// in this case a new object is returned
var y, z;
z = [1, 2, 3, 4];
y = z.insert(1, 999);
z.postln;
y.postln;
)
```

.clipAt(index)

From superclass: ArrayedCollection

Same as -at, but values for **index** greater than the size of the ArrayedCollection will be clipped to the last index.

```
y = [ 1, 2, 3 ];
y.clipAt(13).postln;
```

.wrapAt(index)

From superclass: ArrayedCollection

Same as -at, but values for **index** greater than the size of the ArrayedCollection will be wrapped around to 0.

```
y = [ 1, 2, 3 ];
y.wrapAt(3).postln; // this returns the value at index 0
y.wrapAt(4).postln; // this returns the value at index 1
y.wrapAt([-2, 1])   // index can also be a collection or negative numbers
```

.foldAt(index)

From superclass: ArrayedCollection

Same as -at, but values for **index** greater than the size of the ArrayedCollection will be folded back.

```
y = [ 1, 2, 3 ];
y.foldAt(3).postln; // this returns the value at index 1
y.foldAt(4).postln; // this returns the value at index 0
y.foldAt(5).postln; // this returns the value at index 1
```

.clipPut(index, item)

From superclass: ArrayedCollection

Same as -put, but values for **index** greater than the size of the ArrayedCollection will be clipped to the last index.

.wrapPut(index, item)

From superclass: ArrayedCollection

Same as -put, but values for **index** greater than the size of the ArrayedCollection will be wrapped around to 0.

.foldPut(index, item)

Same as -put, but values for **index** greater than the size of the ArrayedCollection will be folded back.

.**swap**(i, j)

From superclass: ArrayedCollection

Swap the values at indices i and j.

```
[ 1, 2, 3 ].swap(0, 2).postln;
```

.**replace**(find, replace)

From superclass: ArrayedCollection

Return a new array in which a number of elements have been replaced by another. Elements are checked for equality (not for identity).

```
a = (0..10) ++ (0..10);
a.replace([4, 5, 6], 100);
a.replace([4, 5, 6], [1734, 1985, 1860]);
```

this method is inherited by String :

```
a = "hello world";
a.replace("world", "word");
```

++(anArray)

From superclass: ArrayedCollection

Concatenate the contents of the two collections into a new ArrayedCollection.

```
(
var y, z;
z = [1, 2, 3, 4];
y = z ++ [7, 8, 9];
z.postln;
y.postln;
)
```

.**add**(item)

From superclass: ArrayedCollection

Adds an item to an ArrayedCollection if there is space. This method may return a new ArrayedCollection. For this reason, you should always assign the result of add to a variable - never depend on add changing the receiver.

```
(
// z and y are the same object
var y, z;
z = [1, 2, 3];
y = z.add(4);
z.postln;
y.postln;
)

(
// in this case a new object is returned
var y, z;
z = [1, 2, 3, 4];
y = z.add(5);
z.postln;
y.postln;
```

`.addAll(aCollection)`

From superclass: ArrayedCollection

Adds all the elements of aCollection to the contents of the receiver. This method may return a new ArrayedCollection. For this reason, you should always assign the result of `addAll` to a variable - never depend on add changing the receiver.

```
(
// in this case a new object is returned
var y, z;
z = [1, 2, 3, 4];
y = z.addAll([7, 8, 9]);
z.postln;
y.postln;
)
```

`.addFirst(item)`

From superclass: ArrayedCollection

Inserts the item before the contents of the receiver, possibly returning a new collection.

```
(
// in this case a new object is returned
var y, z;
z = [1, 2, 3, 4];
y = z.addFirst(999);
z.postln;
y.postln;
)
```

`.removeAt(index)`

From superclass: ArrayedCollection

Remove and return the element at **index**, shrinking the size of the ArrayedCollection.

```
y = [ 1, 2, 3 ];
y.removeAt(1);
y.postln;
```

`.collect(function)`

From superclass: Collection

Answer a new collection which consists of the results of function evaluated for each item in the collection. The function is passed two arguments, the item and an integer index. See Collection helpfile for examples.

`.do(function)`

From superclass: ArrayedCollection

Iterate over the elements in order, calling the function for each element. The function is passed two arguments, the element and an index.

```
['a', 'b', 'c'].do({ arg item, i; [i, item].postln; });
```

`.reverseDo(function)`

From superclass: ArrayedCollection

Iterate over the elements in reverse order, calling the function for each element. The function is passed two arguments, the element and an index.

.deepCollect(depth: 1, function, index: 0, rank: 0)

From superclass: ArrayedCollection

The same as -collect, but can look inside sub-arrays up to the specified **depth**.

```
a = [99, [4,6,5], [[32]]];
a.deepCollect(1, {|item| item.isArray}).postln;
a.deepCollect(2, {|item| item.isArray}).postln;
a.deepCollect(3, {|item| item.isArray}).postln;
```

.reshape(... shape)

From superclass: ArrayedCollection

For a multidimensional array, rearranges the data using the desired number of elements along each dimension. The data may be extended using wrapExtend if needed.

```
a = [4,7,6,8];
a.reshape(2,2);
a.reshape(2,3);
```

.windex

From superclass: ArrayedCollection

Interprets the array as a list of probabilities which should sum to 1.0 and returns a random index value based on those probabilities.

```
(
Array.fill(10, {
    [0.1, 0.6, 0.3].windex;
}).postln;
)
```

.size

From superclass: ArrayedCollection

Return the number of elements the ArrayedCollection.

.normalize(min: 0.0, max: 1.0)

From superclass: ArrayedCollection

Returns a new Array with the receiver items normalized between **min** and **max**.

```
[1, 2, 3].normalize;           //default min=0, max= 1
[1, 2, 3].normalize(-20, 10);
```

.normalizeSum

From superclass: ArrayedCollection

Returns the Array resulting from :

```
(this / this.sum)
```

so that the array will sum to 1.0.

This is useful for using with windex or wchoose.

.plot(name, bounds, discrete: false, numChannels, minval, maxval, separately: true, parent)

From superclass: ArrayedCollection

Plot values in a GUI window. See plot for more details. When the receiver contains `nil` items, the plot fails with an error.

.reverse

Returns a new Array whose elements are reversed. The receiver is unchanged.

```
x = [1, 2, 3];
z = x.reverse;
x.postln;
z.postln;
```

.scramble

Returns a new Array whose elements have been scrambled. The receiver is unchanged.

```
[1, 2, 3, 4, 5, 6].scramble.postln;
```

.mirror

Return a new Array which is the receiver made into a palindrome. The receiver is unchanged.

```
[1, 2, 3, 4].mirror.postln;
```

.mirror1

Return a new Array which is the receiver made into a palindrome with the last element removed. This is useful if the list will be repeated cyclically, the first element will not get played twice. The receiver is unchanged. If the receiver is a single-element array, a copy is returned.

```
[1, 2, 3, 4].mirror1.postln;
```

.mirror2

Return a new Array which is the receiver concatenated with a reversal of itself. The center element is duplicated. The receiver is unchanged.

```
[1, 2, 3, 4].mirror2.postln;
```

.stutter(n: 2)

NOTE: It is recommended to use `dupEach` instead. This method is retained for backwards compatibility.

Return a new Array whose elements are repeated n times. The receiver is unchanged.

```
[1, 2, 3].stutter(2).postln;
```

Arguments:

n Number of repeats.

Return a new Array whose elements are repeated n times. The receiver is unchanged.

```
[1, 2, 3].dupEach(2).postln;
```

Arguments:

n Number of repeats.

.rotate(n: 1)

Return a new Array whose elements are in rotated order. The receiver is unchanged.

```
[1, 2, 3, 4, 5].rotate(1).postln;
[1, 2, 3, 4, 5].rotate(-1).postln;
[1, 2, 3, 4, 5].rotate(3).postln;
```

Arguments:

n Number of elements to rotate. Negative n values rotate left, positive n values rotate right.

.pyramid(patternType: 1)

Return a new Array whose elements have been reordered via one of 10 "counting" algorithms. Run the examples to see the algorithms.

```
10.do({ arg i;
    [1, 2, 3, 4].pyramid(i + 1).postcs;
});
```

Arguments:

patternType Choose counting algorithm. The algorithms are numbered 1 through 10.

.pyramidg(patternType: 1)

Like pyramid, but keep the resulting values grouped in subarrays.

```
// compare:
[1, 2, 3, 4].pyramid(1).postln;
[1, 2, 3, 4].pyramidg(1).postln;
```

.sputter(probability: 0.25, maxlen: 100)

Return a new Array of length maxlen with the items partly repeated (random choice of given probability).

```
// compare:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].sputter(0.5, 16).postln;
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].sputter(0.8, 8).postln;
```

Arguments:

probability	Probability of repeat.
maxlen	The length of the new Array.

.lace(length)

Returns a new Array whose elements are interlaced sequences of the elements of the receiver's subcollections, up to size length. The receiver is unchanged.

```
x.postln;
y.postln;
```

.permute(nthPermutation)

Returns a new Array whose elements are the nthPermutation of the elements of the receiver. The receiver is unchanged.

```
x = [ 1, 2, 3];
6.do({|i| x.permute(i).postln;});
```

.allTuples(maxTuples: 16384)

Returns a new Array whose elements contain all possible combinations of the receiver's subcollections.

```
[[1, 2, 3, 4, 5], [10, 20, 30]].allTuples;
[[1, 2, 3, 4, 5], [10, 20, 30], [5, 6]].allTuples;
```

.wrapExtend(length)

Returns a new Array whose elements are repeated sequences of the receiver, up to size length. The receiver is unchanged.

```
x = [ 1, 2, 3, "foo", 'bar' ];
y = x.wrapExtend(9);
x.postln;
y.postln;
```

.foldExtend(length)

Same as wrapExtend but the sequences fold back on the list elements.

```
x = [ 1, 2, "foo"];
y = x.foldExtend(9);
x.postln;
y.postln;
```

.clipExtend(length)

Same as wrapExtend but the sequences "clip" (return their last element) rather than wrapping.

```
x = [ 1, 2, "foo"];
y = x.clipExtend(9);
x.postln;
y.postln;
```

.slide(windowLength: 3, stepSize: 1)

Return a new Array whose elements are repeated subsequences from the receiver. Easier to demonstrate than explain.

```
[1, 2, 3, 4, 5, 6].slide(3, 1).postcs;
[1, 2, 3, 4, 5, 6].slide(3, 2).postcs;
[1, 2, 3, 4, 5, 6].slide(4, 1).postcs;
```

Shift the values of the array n steps to the right (n positive) or to the left(n negative), dropping the excess and filling empty space with zero.

```
[1, 2, 3, 4, 5, 6].shift(3).postln;
[1, 2, 3, 4, 5, 6].shift(-3).postln;
```

.containsSeqColl

Returns true if the receiver Array contains any instance of SequenceableCollection

```
[1, 2, 3, 4].containsSeqColl.postln
[1, 2, [3], 4].containsSeqColl.postln
```

.powerset

Returns all possible combinations of the array's elements.

```
[1, 2, 3].powerset.postln
[1, 2, 3].powerset.sort({ |a, b| a.size > b.size }); // sort by size, big
first
[1, 2, 3].powerset.sort({ |a, b| a.size > b.size }).reverse; // by size, small
first
```

powerset is also supported in Collection:

```
Set[1, 2, 3].powerset;
List[1, 2, 3].powerset
(a: 1, b: 2, c: 3).powerset;
```

.envirPairs

Given an array of symbols, this returns an array of pairs of (symbol, value) from the current environment. This can then be used as arguments for a Synth, or in an OSC message.

```
e = (freq: 340, amp: 0.001, strangeness: 0.85);
e.use {
    [\amp, \taste, \strangeness].envirPairs;
}
```

.flop

Invert rows and columns in a two dimensional Array (turn inside out). See also: Function, SequenceableCollection.

```
[[1, 2, 3], [4, 5, 6]].flop;
[[1, 2, 3], [4, 5, 6], [7, 8]].flop; // shorter array wraps
[].flop; // result is always 2-d.
```

.multiChannelExpand

Used by UGens to perform multi channel expansion. Same as flop.

.source

Some UGens return Arrays of OutputProxy when instantiated. This method allows you to get at the source UGen.

```
z.source.postln;
```

.fork(join, clock, quant: 0.0, stackSize: 64)

Used within Routines and assumes an array of functions, from which subroutines are created. The subroutines are played while the outer Routine carries on. The join parameter expresses after how many subroutines complete the outer Routine is allowed to go on. By default this happens after all subroutines have completed.

```
// an array of routine functions:
(
a = [
    { 1.wait; \done_one.postln },
    { 0.5.wait; \done_two.postln },
    { 0.2.wait; \done_three.postln }
];
)
// join after 0
(
Routine {
    "join = 0.".postcln;
    a.fork(0); \doneAll.postln;
}.play;
)
// join after 1
(
Routine {
    "join = 1.".postcln;
    a.fork(1); \doneAll.postln;
}.play;
)
// join after all
(
Routine {
    "join = a.size (default).".postcln;
    a.fork; \doneAll.postln;
}.play;
)
```

.poll(trig: 10, label, trigid: -1)

apply an array of Poll units to an array of UGens (see those helpfiles for more details).

```
(
x = {
    SinOsc.ar([0.1, 0.2], 0).poll * 0.1
}.play;
)
x.trace; // By tracing the Synth you can see the two Poll units we created
x.free
```

.dpoll(label, run: 1, trigid: -1)

apply an array of Dpoll units to an array of UGens (see those helpfiles for more details).

.atIdentityHash(argKey)

This method is used by IdentitySet to search for a key among its members.

.atIdentityHashInPairs(argKey)

.asString(limit: 512)

From superclass: Object

Returns a string representing the Array. May not be compilable due to elision (...) of excessive arguments.

.asCompileString

From superclass: Object

Returns a string that will compile to return an Array equal to the receiver.

.isValidUGenInput

Returns true. Arrays are valid UGen inputs.

.asRawOSC

Returns the OSC message as an Int8Array. Receiver must be a bundle.

```
[0.1, [\s_new, \default, -1, 1, 1, \freq, 1961]].asRawOSC;
```

Bela

.belaScope(scopeChannel, server)

From extension in /Users/prko/Dropbox/prko/__myDocs/Writings/Making Sound using Open Sources/mixed/dev - Bleeding edge/SuperCollider.app/Contents/Resources/SCClassLibrary/Common/Audio/Bela/BelaScope.sc

Send this Array's content to Bela's Oscilloscope (see BelaScope for required setup)

Arguments:

scopeChannel	Bela's oscilloscope channel to start scoping on. This has to be a non-negative number, and can't be changed after scoping starts.
server	The server on which BelaScope is running. If not specified, it looks for the first server for which BelaScope was already initialized. If none is found, it attempts to initialize a BelaScope instance on Server: *default.

Returns:

This Array.

Cycle

Browse Math/Cycle

Cycle includes the methods Array: -campanology, Array: -circPerm, Array: -interlace, Array: -kaprekar, Array: -kreuzspiel, Array: -pea, Array: -sieve, Array: -symGroup, Array: -symPerm, Integer: -campanology,Integer: -circPerm, Integer: -collatz, Integer: -euclidean, Integer: -kaprekar, Integer: -pea, Number: -lorenz, Env: *collatz.

.campanology(mode: 0, rev: false)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/cycle/SystemOverwrites/cycle.sc

As a musical study of bells, campanology focuses on the ringing methods according to the two following rules:

1. Each bell sounds exactly once in each row.
2. In any change, each bell can move at most one position.

Reference: White, A., & Wilson, R. (1995). The Hunting Group. *The Mathematical Gazette*, 79(484), 5–16. https://doi.org/10.2307/3619985

Arguments:

mode	assigns the first switched couple as indices either 0 or 1.
rev	Boolean, stops the algorithm when its input is reversed.

Returns:

the cycle itself.

Discussion:

```
[1, 2, 3, 4].campanology;
[1, 2, 3, 4].campanology(1);
[1, 2, 3, 4].campanology(rev:true);
```

.circPerm(iBase: 10, cBase: 2)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/cycle/SystemOverwrites/cycle.sc

This algorithm consists of moving the first element of a list to the tail of this list. This is done n times with n equal to the length of the list according to a circular permutation cycle from a given radix to another.

Arguments:

iBase	initial base
cBase	circular base

Returns:

the cycle as a circular permutation according to the radix as arguments.

Discussion:

```
[1, 2, 3, 4].circPerm(10, 2);
```

.interlace

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/cycle/SystemOverwrites/cycle.sc

Based on the principle of ornamental interlace, each array as a 'pattern' is repeated some times according to the least common multiple to generate a complete cycle.

Returns:

an array where each element groups the circular permutations of each array as a 'pattern' such as the number of voices equal to the number of arrays.

Discussion:

```
[(1..4), (1..8), (1..3)].interlace;
```

.kaprekar(base: 10)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/cycle/SystemOverwrites/cycle.sc

Apply the method Integer: -kaprekar to an array according to a given radix (10 by default).

Arguments:

base

Returns:

Discussion:

```
[1, 2, 3, 4].kaprekar;
[1, 2, 3, 4].kaprekar.seq;
[1, 2, 3, 4].kaprekar.path;
[1, 2, 3, 4].kaprekar.cycle;
```

.kreuzspiel(ind)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/cycle/SystemOverwrites/cycle.sc

In 1951, Karlheinz Stockhausen wrotes *Kreuzspiel* (*« Jeux croisés »*) which uses the crossing technique inspired by Olivier Messiaen called *« des extrêmes au centre »* as a kind of retrogradation en éventail. This consists to take the first and the last values of a sequence and places them in the middle (or optionally at a given indexed place) by opposite crossing.

Reference at http://hdl.handle.net/1773/23571

Arguments:

ind as indice, `(this.size/2).floor` by default.

Returns:

The cycle itself.

Discussion:

```
[1, 2, 3, 4].kreuzspiel;
[1, 2, 3, 4].kreuzspiel(1);
```

.pea

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/cycle/SystemOverwrites/cycle.sc

Primarily developed by John Conway, the pea pattern is a variation of a 'look-and-say' sequence by counting the elements of a seed from the lowest number to the highest iteratively until the effectiveness of a cycle.

Returns:

The cycle itself prepended by its 'path' if it exists.

Discussion:

```
[1, 2, 3, 4].pea;
[1, 2, 3, 4].pea.seq;
[1, 2, 3, 4].pea.path;
[1, 2, 3, 4].pea.cycle;
```

.sieve(field, i, j, optimize: 'no')

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/cycle/SystemOverwrites/cycle.sc

Kind of *Metabole* according to Iannis Xenakis, rhythmic sieves rely on the construction of a matrix to generate two cycles: one by horizontal shift and one by vertical shift.

For instance, with `i = 4` and `j = 5`, the shifting cycle is done according to the following matrix:

	0	*1*	*2*	*3*
0	0	5	10	15
1	16	1	6	11
2	12	17	2	7
3	8	13	18	3

NOTE: by convention, `i` should be strictly inferior to `j` . Also, this kind of matrix requires `i.gcd(j) == 1` .

Arguments:

field	set the maximum value of the matrix, `this.maxItem` by default.
i	as the x-axis of the matrix.
j	as the y-axis of the matrix.
optimize	requires one of the following symbols:

`\no`	`\n`	nearest solution of `this.maxItem` or `field` if set
`\yes`	`\y`	look for the next solution in order to minimize `(i-j).abs`
`\field`	`\f`	look for the minimal `(i-j).abs` between `this.maxItem` and `field`

Returns:

an array of two cycles.

Discussion:

```
[2, 11, 7, 12, 8, 14].sieve;
[2, 11, 7, 12, 8, 14].sieve(field: 52, optimize: 'field');
```

`.symGroup(ref)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/cycle/SystemOverwrites/cycle.sc

The symmetric group allows to enumerate all cycles from a list of *n* elements according to their respective position of the same ordered list (i.e. { 1, 2, ..., *n* }) such as each set is a group of permutations as a bijective maps.

Arguments:

ref allows to apply the algorithm on any kind of elements provided that `(this.asBag == ref.asBag)` .

Returns:

a list of cycles.

Discussion:

```
[3, 10, 2, 4, 7, 9, 8, 5, 6, 1].symGroup;
```

`.symPerm(arrayCode)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/cycle/SystemOverwrites/cycle.sc

Highlighted by Olivier Messiaen, symmetric permutations constitute a system of numbered durations that consists in changing the order of its durations in order to obtain a limited number of rhythmic cells.

Arguments:

arrayCode allows to re-order durations

Returns:

the cycle as a symmetric permutation according to an re-ordered array as arguments.

Discussion:

```
[3, 1, 2, 2].symPerm([3, 1, 4, 2]);
```

59 methods from ArrayedCollection ▶ show

605 methods from SequenceableCollection ▶ show

159 methods from Collection ▶ show

394 methods from Object ▶ show

Undocumented instance methods

`.allSplineIntControls(amt, clipMode: 'wrap')`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

`.asActionFunc`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Extensions/Various/ActionFunc.sc

`.asAudioRateInput(for)`

`.asControlInput`

`.asNote(cents)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Note/extVarious-asNote.sc

`.asRewritingRule`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/MathLib/classes/LazyLindenmayer/extStringRewrite.sc

`.asSequenceNote`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/ddwCommon/SequenceNotes/SequenceNoteMathExtensions.sc

`.asSpec`

`.asUGenInput(for)`

`.atB(index, loop: true, extra)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

`.atH(index, loop: true)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

`.atL(index, loop: true)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

`.atQ(index, loop: true)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

`.atS(index, loop: true, extra)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

`.atSin(index, loop: true)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

`.bSplineInt(i, amt, loop: true)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

Features/Interpolation/extArray-interpolation.sc

.bSplineIntControls(amt: 4)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

.bSplineIntDeltaControls(amt: 4)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

.buildForProxy(proxy, channelOffset: 0)

From extension in /Users/prko/Dropbox/prko/__myDocs/Writings/Making Sound using Open Sources/mixed/dev - Bleeding edge/SuperCollider.app/Contents/Resources/SCClassLibrary/JITLib/ProxySpace/wrapForNodeProxy.sc

.cpsname(sign)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Note/extVarious-midiname.sc

.cycle

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/cycle/SystemOverwrites/cycle.sc

.deinterlace(clumpSize: 2, numChan: 1)

.eliminateMatrix

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/MathLib/classes/various/matrix_elimination.sc

.envAt(time)

.fastAtL(index)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

.fillEnds(nStart: 1, nEnd: 2)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

.findReplace(findString, replaceString: "", ignoreCase: false)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Extensions/String/extString-findReplace.sc

.ghostAt(i)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

.hermiteInt(i)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

.intAt(index, type: 'linear', loop: true, extra)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

.interpolate(division: 10, type: 'linear', loop: true, extra, close: false)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

.linearInt(i, step: 0)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

.madd(mul: 1.0, add: 0.0)

.matchForRewriting(string, value, args)

.midiname(sign)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Note/extVarious-midiname.sc

.modeAt(index, mode: 'wrap')

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

.namecps(cents)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Note/extVarious-midiname.sc

.namemidi(cents)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Note/extVarious-midiname.sc

.namename(cents, sign)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Note/extVarious-midiname.sc

.notate(musicXMLfilePath, app: "MuseScore 4")

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/Notator/Classes/Notator.sc

.numChannels

.path

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/cycle/SystemOverwrites/cycle.sc

.prUnarchive(slotArray)

.prUnlace(clumpSize: 2, numChan: 1)

.proxyControlClass

From extension in /Users/prko/Dropbox/prko/__myDocs/Writings/Making Sound using Open Sources/mixed/dev - Bleeding edge/SuperCollider.app/Contents/Resources/SCClassLibrary/JITLib/ProxySpace/wrapForNodeProxy.sc

.quadInt(i)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

.quadIntControl

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

.quadIntFunction(i, x1)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

.resize(newSize: 10, type: 'linear', loop: false, extra)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

.rewritingContextFree

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/MathLib/classes/LazyLindenmayer/extStringRewrite.sc

.rewritingRuleSize

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/MathLib/classes/LazyLindenmayer/extStringRewrite.sc

.rotateL(n: 0)

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Extensions/UGens/extArray-Rotation.sc classes/Extensions/UGens/extArray-Rotation.sc

`.rotateS(n: 0)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Extensions/UGens/extArray-Rotation.sc

`.scope(name: "UGen Scope", bufsize: 4096, zoom: 1.0)`

From extension in /Users/prko/Dropbox/prko/__myDocs/Writings/Making Sound using Open Sources/mixed/dev - Bleeding edge/SuperCollider.app/Contents/Resources/SCClassLibrary/Common/GUI/PlusGUI/Control/UGen-scope.sc

`.selectFindString(stringToFind, ignoreCase: false)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Extensions/String/extString-findReplace.sc

`.seq`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/cycle/SystemOverwrites/cycle.sc

`.sineInt(i)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

`.splineInt(i, amt)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

`.splineIntControls(amt)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

`.splineIntFunction(i, x1, x2)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

`.splineIntFunctionArray(i, x1array, x2array)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

`.splineIntPart1(x1, x2)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

`.splineIntPart2(i)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/wslib/wslib-classes/Main Features/Interpolation/extArray-interpolation.sc

`.storeEditOn(stream, cacheKey)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/ddwCommon/Misc/Object-asEditString.sc

`.unlace(clumpSize: 2, numChan: 1, clip: false)`

`.weight(weights)`

From extension in /Users/prko/Library/Application Support/SuperCollider/downloaded-quarks/ddwCommon/Misc/Collection-extensions.sc

helpfile source: /Users/prko/Dropbox/prko/__myDocs/Writings/Making Sound using Open Sources/mixed/dev - Bleeding edge/SuperCollider.app/Contents/Resources/HelpSource/Classes/Array.schelp
link::Classes/Array::