

# KeRF Reference Card

## Lists

Name		Behavior
<code>ascend(x)</code>	<	Ascending indices / grade up.
<code>bucketed(x,y)</code>		Bucket y Values by x.
<code>car(x)</code>		First element of x.
<code>cdr(x)</code>		All except first element of x.
<code>combinations(x,n,repeats)</code>		N-long combinations.
<code>count(x)</code>		Number of elements in x.
<code>cross(x,y)</code>		Cartesian product of lists.
<code>deal(x,y)</code>		x unique random items of y.
<code>descend(x)</code>	>	Descending indices / grade dn.
<code>distinct(x)</code>	%	First instance of elements of x.
<code>drop(x,y)</code>	-	Remove first x elements from y.
<code>enlist(x)</code>		Wrap x in a list.
<code>enumerate(x)</code>	~	Range [0,x) / map keys.
<code>except(x,y)</code>		Remove all y from x.
<code>explode(x,y)</code>		Split y at instances of x.
<code>filter(f,x)</code>		Items where f(x) is truthy.
<code>first(x,y)</code>	~	Take first x elements of y.
<code>flatten(x)</code>		Concatenate elements of x.
<code>implode(x,y)</code>		Join elements of y with x.
<code>in(x,y)</code>		Is the item x in y?
<code>intersect(x,y)</code>		Collect items in x and y.
<code>join(x,y)</code>	#	Create a list from x and y.
<code>last(x,y)</code>		Last x elements of y.
<code>order(x)</code>		Equivalent to <<x.
<code>part(x)</code>	&	Map unique items to indices.
<code>permutations(x,repeats)</code>		Optional repeat elements.
<code>powerset(x)</code>		All subsets of x.
<code>range(x)</code>		Integer vector [0, x).
<code>range(x,y)</code>		Numeric vector [x, y), step 1.
<code>range(x,y,z)</code>		Numeric vector [x, y), step z.
<code>repeat(x,y)</code>		List of x copies of y.
<code>reverse(x)</code>	/	Reverse the order of items in x.
<code>search(x,y)</code>		Find index of x in y or null.
<code>shift(x,y,z)</code>		Shift y by x, fill with z.
<code>shuffle(x)</code>		Randomly permute x.
<code>sort(x)</code>		Sort the items in x.
<code>split(x,y)</code>		Split x by indices in y.
<code>trim(x)</code>		Remove leading/trailing spaces.
<code>union(x)</code>		Set union of x and y.
<code>which(x)</code>	?	Gather nonzero indices.

## Tables And Maps

Name	Behavior
<code>asof_join(x, y, k1, k2)</code>	Left join x and y on k1 as of k2.
<code>has_column(x, name)</code>	Does table have a given column?
<code>has_key(x, key)</code>	Check for valid index.
<code>left_join(x, y, keys)</code>	Left join x and y on k.
<code>map(keys, values)</code>	Build a map from two vectors.
<code>xkeys(x)</code>	Keys of map/table, indices of list.
<code>xvals(x)</code>	Values of map/table/list.
<code>tables()</code>	List names of loaded tables.

## Aggregates

Name	Behavior
<code>rsum(list)</code>	Running sum. + unfold list.
<code>count_nonnull(list)</code>	How many items are not null?
<code>count_null(list)</code>	How many items are null?
<code>xbar(x,y)</code>	$x * \text{floor } y/x$ .
List Statistics	
avg max median min std sum var	
Sliding Windows (width, list)	
mavg mcount mmax mmin msum	

## Math

Name	Behavior
<code>between(x,y)</code>	Is x between y[0] and y[1]?
<code>dotp(x,y)</code>	Dot/Scalar product of x and y.
<code>log(x)</code>	$\log_{10}(x)$ .
<code>log(x,y)</code>	$\log_x(y)$ .
<code>lsq(A,B)</code>	\ Solve $Ax = B$ for x.
<code>minv(A)</code>	Inverse of matrix A.
<code>mmul(A,B)</code>	Multiply matrices A and B.
<code>rand()</code>	1 random float [0, 1).
<code>rand(x)</code>	1 random integer [0, x).
<code>rand(x,y)</code>	? x random integers [0, y).
<code>stamp.diff(x,y)</code>	Stamp difference in ns.
<code>transpose(x)</code>	. Matrix transpose of x.
+ - / * % ** ! &   ~ : = == < > <= >= != <> abs acos add asin atan ceil cos cosh divide erf erfc exp floor lg ln mod negative pow sin sinh sqrt subtract tan tanh times and equal greater greatereq less lesseq match not noteq or	

## Miscellaneous

Name	Behavior
<code>atom(x)</code>	Is x an atom?
<code>eval(x)</code>	Evaluate the string x.
<code>ifnull(x)</code>	Is x null?
<code>kerf_type(x)</code>	Numeric typecode of x.
<code>kerf_type_name(x)</code>	Human-readable type name.
<code>now()</code>	Current date-time.
<code>now_date()</code>	Current date.
<code>now_time()</code>	Current time.
<code>rep(x)</code>	String representation of x.
<code>shell(x)</code>	Execute x as a shell command.
<code>seed_prng(x)</code>	Seed the RNG used by <code>rand</code> .
<code>sleep(x)</code>	Wait at least x milliseconds.
<code>timing(x)</code>	Enable or disable time logging.
Casting/Conversion	
atlas char float hashed indexed int string json_from_kerf kerf_from_json type_null	

## IPC

Name	Behavior
<code>open_socket (host, port)</code>	Host and port must be strings.
<code>close_socket(handle)</code>	Close an IPC socket handle.
<code>send_async(handle, expr)</code>	Doesn't block, Returns 1.
<code>send_sync(handle, expr)</code>	Returns <code>eval(y)</code> on remote host.
Launch with <code>-p &lt;portnumber&gt;</code> to start IPC server	

Combinators	
Name	Behavior
u converge x	Apply u to x until the value repeats or stops changing.
x u converge y	Apply u to x, y times.
u deconverge x	As <b>converge</b> , but gather intermediate results.
x u deconverge x	
b fold x	Apply b between elements of x.
x b fold y	
b mapback x	Apply b between elements of x and their predecessor.
x b mapback y	
u mapdown x	Apply u/b to each element of x (and each of y if present).
x b mapdown y	
x b mapleft y	Apply b to y and each of x.
x b mapright y	Apply b to x and each of y.
b unfold x	As <b>fold</b> , but gather intermediate results.
x b unfold y	

Kerf types			
Example	type_null	kerf_type_name	kerf_type
[2000.01.01]	00:00:00.000	stamp vector	-4
[0.1]	nan	float vector	-3
[1]	NAN	integer vector	-2
"A"	` " "	character vector	-1
{[x] 1+x}	null	function	0
`A	` " "	character	1
1	NAN	integer	2
0.1	nan	float	3
2000.01.01	00:00:00.000	stamp	4
()	null	null	5
[]	null	list	6
{a:1}	null	map	7
#["a"]	null	enum	8
= [1]	null	sort	9
{{a:1}}	null	table	10

IO and Scripting	
Name	Behavior
load(filename)	Read and execute Kerf source.
exit(statusCode)	Exit, status code is optional.
out(string)	Print a raw string x to stdout.
display(x)	Prettyprint x to stdout.
dir_ls(path, showFullPaths)	List the files in a directory path.
lines(filename, limit)	Read lines of a text file.
open_table(filename)	<b>mmap</b> serialized Kerf table.
read_from_path(filename)	Load a serialized Kerf object.
read_striped_from_path(dirname)	Load a striped Kerf object.
read_table_from_csv (filename,fields,hrows)	Load a CSV file into a table.
read_table_from_delimited_file (delim,filename,fields,hrows)	Load a delimited file into a table.
read_table_from_fixed_file (filename,attr)	Load a fixed-width file. <b>attr</b> contains field types, widths, etc.
read_table_from_tsv (filename,fields,hrows)	Load a TSV file into a table with optional header rows.
write_csv_from_table (filename,table)	Write a table to a CSV file.
write_delimited_file_from_table (delim,filename,table)	Write a table to a delimited file.
write_striped_to_path (dirname,x)	Write a striped Kerf object.
write_text(filename, x)	Serialize x to a file as JSON.
write_to_path(filename, x)	Serialize x to a file as binary.
build_table_from_csv (tablepath,csvpath,fields,hrows)	Append to disk-backed table.
build_table_from_fixed_file (tablepath,fixedpath,attr)	Append to disk-backed table.
build_table_from_psv (tablepath,psvpath,fields,hrows)	Append to disk-backed table.
build_table_from_tsv (tablepath,tsvpath,fields,hrows)	Append to disk-backed table.
dllload(filename,funcname,argc)	Load a dynamic library.

fixed_file Attributes	
Name	Behavior
<b>fields</b>	Field specifier string.
<b>widths</b>	List of column widths in chars.
<b>line_limit</b>	Max row count.
<b>titles</b>	List of column names.
<b>header_rows</b>	How many rows are headers?
<b>line_separated</b>	Are rows separated by newlines?

Delimited/Fixed-Width Field Specifiers	
Symbol	Datatype
I	Integer
F	Float
S	String
E	Enumerated String
G	IETF RFC-4122 UUID
N	IP address as parsed by <b>inet_pton()</b>
Z/Y	Custom. See <b>.Parse.strptime_format</b>
*	Skipped field
R	NYSE TAQ symbol (Fixed-width only)
Q	NYSE TAQ time format (Fixed-width only)

SQL Syntax
INSERT INTO table VALUES data
DELETE FROM table [ WHERE condition ]
SELECT field1, field2 [ AS name ] ... FROM table [ WHERE condition ] [ GROUP BY aggregate ]
UPDATE table SET field1:val1, field2:val2 ... [ WHERE condition ] [ GROUP BY aggregate ]