

## Common Lisp Cheat Sheet

by ehaliewicz via cheatography.com/1470/cs/487/

Common Lisp Sec	uences	
Creating Collections		
(vector elements)	Generic vector of args.	
(make-array size [opts])	Returns empty n-dimensional array.	
(list elements)	Creates a singly-linked list of args.	
Operating on Sequences		
General		
(length sequence)	Returns length of sequence.	
(elt sequence index)	Returns the corresponding element of sequence.	
Vectors		
(vector-pop array)	Destructively removes the last element of array.	
(vector-push new-el array)	Destructively appends new-el to array.	
	(If the array fill-pointer indicates full, nothing occurs)	
(vector-push- extend new-el array [opts])	Destructively appends new-el to array.	
	(will extend an array with a fill-pointer if necessary)	
Search and Sort		
(count item sequence [opts])	Returns number of times item appears.	
(position item sequence [opts])	Returns index of item or nil.	
(sort sequence predicate [opts])	Sorts sequence by predicate function. (i.e.	

Common Lisp	Sequences (cont)
i.e. (sort '(1 4 3 2) #'<)	=> '(1 2 3 4)
Filters	
(count-if predicate sequence)	Returns number of elements that satisfy predicate
(remove-if predicate sequence)	Removes all elements that satisfy predicate. (non-d-estructive)
	These functions have 'if- not' variants.
Iteration	
(map result- type func sequences )	Returns the result of applying an n-arg function to the current element of n sequences.
(mapcar function sequences)	Same as above, but for lists.
(map-into result-se- quence func sequences )	Places results into 'result- sequence' rather than creating a new sequence.
(reduce func sequence [opts])	Applies func to elements in twos to return a single value.
i.e. (reduce #'+ '(1 2 3))	=> (+ 1 2) => (+ (+ 1 2) 3) => 6
(loop [opts])	http://cl-cookbook.sourc- eforge.net/loop.html

Assignment and Binding
(setf place val &rest args)
Sets value of 'place' to val. Works in pairs.
i.e. (setf 'a 1
'b 2)
Sets 'a to 1 and 'b to 2.
(let ((var-a value)
(var-b value))
Creates a context where var-a and var-b are
bound to their respective values.
(let* ((var-a value)
(var-b value))
Same as above, but bound sequentially.
i.e. Bindings can refer to previous bindings.
(defparameter var-name value) Creates a
global variable.



By **ehaliewicz** cheatography.com/ehaliewicz/

#'<)

Not published yet. Last updated 12th May, 2016. Page 1 of 1. Sponsored by **CrosswordCheats.com**Learn to solve cryptic crosswords!
http://crosswordcheats.com