

# Emacs Lisp Types

Main Type Category	Sub-category	Sub-category	Sub-category	References
Symbols	Symbols			<ul style="list-style-type: none"> <li>Symbol Type</li> <li>Symbol components</li> </ul>
	booleans			
Numbers				
	Integers			
	Floats			
	<a href="#">complex-numbers</a>			Complex numbers are not explicitly supported by Emacs Lisp. <ul style="list-style-type: none"> <li>The <a href="#">cplx external library</a> supports the concept of complex numbers.</li> </ul>
Characters and Strings				
	Char			<ul style="list-style-type: none"> <li>Basic char literal syntax</li> </ul>
	Strings			<ul style="list-style-type: none"> <li>Creating Strings</li> <li>String Modifications @ Emacs Wiki</li> <li>Replace in String @ Emacs Wiki</li> <li>Split String @ Emacs Wiki</li> <li>String trim @ ΣXah</li> </ul>
<ul style="list-style-type: none"> <li>Encodings</li> </ul>				<ul style="list-style-type: none"> <li>Unicode Encoding @ Emacs Wiki</li> <li>Emacs Unicode Pitfalls, by Christopher Wellons, 2014-06-13</li> </ul>
Ordered Collection of elements				
See also: <ul style="list-style-type: none"> <li>Pure storage</li> <li>Elisp Cookbook @ Emacs Wiki</li> </ul>	Sequence			<ul style="list-style-type: none"> <li>Sequence Functions</li> </ul>
		List		<ul style="list-style-type: none"> <li>List Modifications @ Emacs Wiki</li> <li>List Destructive Operations @ Emacs Wiki</li> <li>Sorting Lists</li> </ul>
			cons cell	Note: a cons cell is not a sequence. The cons cell is placed here because of its strong relationship with lists.
			alist - association list	<ul style="list-style-type: none"> <li>Elisp: Association List @ ΣXah</li> </ul>
			plist - property list	<ul style="list-style-type: none"> <li>Elisp: Property List @ ΣXah</li> <li>Alist Vs Plist @ Emacs Wiki</li> </ul>
			set - (lists as sets)	
		Array		<ul style="list-style-type: none"> <li>Array Functions</li> </ul>
			Vector	<ul style="list-style-type: none"> <li>Vector Functions</li> </ul>
			Bool-vector	
			Char-table	
			Ring	
			String	See strings above. Strings are sequences of characters.
Creation of New Object Types				
	Record			Used as underlying representations of <i>cl-defstruct</i> and <i>defclass</i> instances.
	Structure			<ul style="list-style-type: none"> <li>Options for Structured Data in Emacs Lisp, by Christopher Wellons, 2018-02-14</li> <li>The Common Lisp Cookbook - Data Structures - Structures</li> </ul>
	Hash-table			
ieio - CLOS for Emacs Lisp	classes			
Emacs Specialized Types				
	Buffers			
	Markers			<ul style="list-style-type: none"> <li>Functions that create markers</li> </ul>
	Overlays			
Functions				<ul style="list-style-type: none"> <li>Emacs Lisp Code Guidelines - Functions, lambdas, macros</li> </ul>
	functions			Defining a function with: <ul style="list-style-type: none"> <li>defun macro                             <ul style="list-style-type: none"> <li>function argument list of functions defined with the defun macro</li> <li>inline functions</li> <li>Calling functions indirectly</li> </ul> </li> <li>cl-lib macros: cl-defun, cl-function, cl-defsubst, ...                             <ul style="list-style-type: none"> <li>See also Common Lisp references (which explain what Emacs Lisp cl-lib emulates):                                     <ul style="list-style-type: none"> <li>The Common Lisp Cookbook – Functions</li> <li>Practical Common Lisp - Common Lisp functions</li> </ul> </li> <li>Note that Emacs Lisp cl-defun support <i>&amp;aux auxiliary variables</i> in argument list, something not available in Common Lisp.</li> </ul> </li> </ul>
	lambda			<ul style="list-style-type: none"> <li>Anonymous functions</li> <li>lambda expressions</li> <li>What's in an Emacs lambda, by Christopher Wellons, 2017-12-14</li> <li>Emacs Lisp Lambda Expressions Are Not Self-Evaluating, by Christopher Wellons, 2018-02-22</li> </ul>
	closures (lambda)			<ul style="list-style-type: none"> <li>Emacs Lisp Readable Closures, by Christopher Wellons, 2013-12-30</li> <li>Lexical binding</li> </ul>
ieio - CLOS for Emacs Lisp	methods			<ul style="list-style-type: none"> <li>Generic functions</li> </ul>