



Lispy — A specialized modal editing for Lisp code🔧🚧

Description	Key	Function	Note
<b>Lispy :</b> <b>Context-based modal editing of Lisp code</b>  <b>Ref: Lispy function Reference</b>	<p>The lispy minor mode provides modal-like editing to Emacs for Lisp-like languages with very few keys.</p> <ul style="list-style-type: none"><li>Lisp is a very structured programming language, made of succession and combinations of S-expressions (“sexp”): lists that start with ( and end with ) <b>“paren”</b>. Lispy takes advantage of the structure of Lisp code.</li><li>As long as point (the cursor) is before the left, opening, paren or after the right, closing paren, <b>the keys are interpreted as lispy commands</b>.</li><li>Keys in other locations are interpreted as usual.</li></ul> <p>This table lists the lispy command keys, with links to the <a href="#">Lispy function Reference</a> for each one.</p> <p>📦 This requires the <a href="#">lispy</a> external package. 📖 PEL downloads, installs and activates lispy when the <b>pel-use-lispy</b> user option is set to t.</p> <p>👤👉 To get lispy mode run when Emacs visits a file of a specified mode, include the major mode in the PEL user-option <b>pel-modes-activating-lispy</b> .</p> <ul style="list-style-type: none"><li>PEL does not activate lispy for any major mode by default. That’s OK to learn lispy by activating it for testing. But once you learn and are comfortable with it you will want to activate when the file is opened automatically by adding the major mode in that list.</li></ul>		
🔗 <b>Customize</b> PEL use of Lispy and Lispy itself.	<f11> <f2> SPC M-L	(pel-cfg-pkg-lisp &optional OTHER-WINDOW)	Prompt to customize: <ul style="list-style-type: none"><li>PEL lispy support for Emacs Lisp and Common Lisp</li><li>lispy itself.</li></ul> <ul style="list-style-type: none"><li>If OTHER-WINDOW is non-nil (use C-u), display in another window.</li></ul>
<b>Toggle Lispy mode</b> See also: <ul style="list-style-type: none"><li>🔗 I - Common Lisp</li><li>🔗🔗 I - Emacs Lisp</li></ul>	<ul style="list-style-type: none"><li>&lt;f12&gt; M-L</li><li>&lt;M-f12&gt; M-L</li></ul> <f11> SPC 1 M-L	(pel-lispy-mode &optional ARG)	Toggle lispy-mode on/off. Lispy is a minor mode for navigating and editing LISP dialects. 📦 <a href="#">Requires lispy external package.</a> 📖 PEL <a href="#">downloads, installs and configure it when pel-use-lispy user option is set to t.</a> Please read the information on <a href="#">lispy web site</a> . 🖥️ <b>pel-lispy-mode</b> calls <b>lispy-mode</b> but also prepares hydra, loaded dynamically with PEL. 👉 Set the <b>pel-modes-activating-lispy</b> user-option to activate lispy automatically for major modes.
<b>Getting Code Help</b> See also: 🔗 <a href="#">Help/Info</a>	Use the following keys to pop information inside the current window (if small enough) or into a help buffer. <ul style="list-style-type: none"><li>The &lt;f12&gt; 1 and &lt;f12&gt; 2 PEL keys are available even when lispy mode is off.</li><li>See the 🔗 <a href="#">Help/Info</a> table for more commands you can use to get help about Emacs Lisp code and Emacs in general.</li></ul>		
<b>Describe function at point</b>  See Also: 🔗 <a href="#">Help/Info</a>	C-1	(lispy-describe-inline)	Display documentation of current Lisp function: ‘lispy--current-function’ inline. <ul style="list-style-type: none"><li>If docstring is small enough it is displayed in a pop-up box above point. Otherwise it is displayed inside a *lispy-help* buffer.</li></ul> 📦 This requires the <a href="#">lispy</a> external package. 📖 PEL <a href="#">downloads, installs and activates lispy when the pel-use-lispy user option is set to t.</a>
	<f12> 1		The <f12> 1 key can be used even when lispy mode is not active.
<b>Describe function arguments</b>	C-2	(lispy-arglist-inline)	Show the argument list of current function.
	<f12> 2		The <f12> 2 key can be used even when lispy mode is not active.
<b>Describe function/variable</b>	xh		A shorthand for describe-function or describe-variable. <ul style="list-style-type: none"><li>If you want to call describe-variable, you should mark the symbol first.</li></ul>
<b>Insert space &amp; comments</b>			
<b>Insert a space</b>	Space	(lispy-space ARG)	Insert one space, with position depending on ARG. <ul style="list-style-type: none"><li>If ARG is 2, amend the current list with a space from current side.</li><li>If ARG is 3, switch to the different side beforehand.</li><li>If jammed between parens, "((" unjam: "((" (".</li><li>If after an opening delimiter and before a space (after wrapping a sexp, for example), do the opposite and delete the extra space, "(  foo)" to "( foo)".</li></ul>
<b>Inserting comment</b>	;	(lispy-comment &optional ARG)	Comment ARG sexps.
<b>Insert pairs</b>	The following commands insert pairs of delimiters or quotes.		
insert a paren pair	(	(lispy-parens ARG)	lispy-pair’ with ().
Insert [ ]	}	(lispy-brackets ARG)	‘lispy-pair’ with [].
Insert { }	{	(lispy-braces ARG)	‘lispy-pair’ with {}.
Insert “ ”	“	(lispy-quotes ARG)	Insert a pair of quotes around the point. <ul style="list-style-type: none"><li>When the region is active, wrap it in quotes instead.</li><li>When inside string, if ARG is nil quotes are quoted, otherwise the whole string is unquoted.</li></ul>
<b>Delete</b>			
Delete sexp forward	C-d	(lispy-delete ARG)	Delete ARG sexps.
Delete sexp backward	DEL	(lispy-delete-backward ARG)	From ") ", delete ARG sexps backwards. <ul style="list-style-type: none"><li>Otherwise (‘backward-delete-char-untabify’ ARG).</li></ul>
		(lispy-delete-or-splice-or-slurp ARG)	Call ‘lispy-delete’ unless before a delimiter. <ul style="list-style-type: none"><li>Before an opening delimiter, splice. Before a closing delimiter, slurp to the end of the line without moving the point. When in a position where slurping will not have an effect such as at the final delimiters before the end of a line, move forward. In comments and strings, call ‘lispy-delete’.</li><li>When before the opening quote of a string, delete the entire string. When before the closing quote of a string, move forward.</li></ul>
		(lispy-delete-backward-or-splice-or-slurp ARG)	Call ‘lispy-delete-backward’ unless after a delimiter. <ul style="list-style-type: none"><li>After an opening delimiter, splice. After a closing delimiter, slurp to the end of the line without moving the point. When in a position where slurping will not have an effect such as after the final delimiters before the end of a line, move backward. In comments and strings, call ‘lispy-delete-backward’. When after the opening quote of a string, delete the entire string. When after the closing quote of a string, move backward.</li></ul>
<b>Navigate with avy commands</b>	The following commands use avy-style highlighting to identify a word target to move to. Avy is similar to Ace. Lispy uses Avy internally. <ul style="list-style-type: none"><li>By default the scope is the current list. Use a command numerical prefix to select a larger outer scope.</li><li>After hitting the command key, type the letter(s) identifying the target to move to that word and select it.</li></ul>		
<b>ace symbol move</b> <ul style="list-style-type: none"><li>Uses <a href="#">avy navigation</a></li><li>ARG sets target scope</li><li>ace highlight targets</li><li>move to selected one</li><li>mark selected word</li></ul>	a	(special-lispy-ace-symbol ARG)	Jump to a symbol within the <b>current S-exp</b> and mark it. <ul style="list-style-type: none"><li>Use ace method: each symbol in S-exp is shown with highlight letter: type that letter to move to the symbol.</li><li>S-exp scope is obtained by exiting the list ARG times: default is 1: current S-exp. to select a larger scope S-exp, use a numeric argument:<ul style="list-style-type: none"><li>Example: <b>M-3 a</b> selects 3 layers of enclosing S-exp to select ace targets.</li></ul></li></ul>
<b>ace sub-word</b> <ul style="list-style-type: none"><li>ARG sets target scope</li><li>ace highlight targets</li><li>move to selected one</li><li>mark selected sub-word</li></ul>	-	(special-lispy-ace-subword ARG)	Similar to lispy-ace-symbol, but selects a subword instead. <ul style="list-style-type: none"><li>S-exp scope is obtained by exiting the list ARG times: default is 1: current S-exp. to select a larger scope S-exp, use a numeric argument:<ul style="list-style-type: none"><li>Example: <b>M-3 a</b> selects 3 layers of enclosing S-exp to select ace targets.</li></ul></li></ul>

Description	Key	Function	Note
<u>Move to Ace target symbol &amp; erase to replace</u>	<b>H</b>	( <a href="#">special-lispy-ace-symbol-replace</a> ARG)	Jump to a symbol within the current sexp and delete it, leaving point at location to type the new symbol. <ul style="list-style-type: none"> <li>Sexp is obtained by exiting the list ARG times.</li> <li>Calls lispy-ace-symbol and deletes the selected symbol.</li> </ul>
<u>Move to Ace paren target</u>	<b>q</b>	( <a href="#">special-lispy-ace-paren</a> &optional ARG)	Highlights each <b>symbol</b> in current sexp as ace target and jump to the selected one. <ul style="list-style-type: none"> <li>Updates lispy-back history.</li> <li>S-exp scope is obtained by exiting the list ARG times: default is 1</li> </ul>
<u>Move to Ace target char</u>	<b>Q</b>	( <a href="#">special-lispy-ace-char</a> )	Prompts for character, highlights each one in current sexp as ace target and jump to the selected one.
<b>Navigate in code</b>	The following commands move point inside code when point is before left paren or after right paren. Use <b>d</b> to switch side to control direction.		
<u>Move to beginning of current defun</u>	<b>A</b>	( <a href="#">special-lispy-beginning-of-defun</a> &optional ARG)	Forward to beginning-of-defun. When called twice in a row, restore the previous point and mark positions.
<u>Move to different (other) side of sexp</u>	<b>d</b>	( <a href="#">special-lispy-different</a> )	Switch to the different side of current sexp. <ul style="list-style-type: none"> <li>If before '(' move after ')' and vice-versa.</li> </ul>
<u>Flow: move in the direction of current paren</u> <ul style="list-style-type: none"> <li>( → down</li> <li>; → down</li> <li>) → up</li> </ul>	<b>f</b>	( <a href="#">special-lispy-flow</a> ARG)	Move in the direction of current paren inside current list and then to the next/previous list: <ul style="list-style-type: none"> <li>At left : move to next left paren (move going down the file).</li> <li>At right: move to previous right parent (move going up the file).</li> <li>Don't enter strings or comments.</li> <li>Updates lispy-back history.</li> </ul>
<u>Move down current list</u> <ul style="list-style-type: none"> <li>never exit current list</li> </ul>	<b>j</b>	( <a href="#">special-lispy-down</a> ARG)	Move down ARG times inside current list. <ul style="list-style-type: none"> <li>Guaranteed to never exit the list: <b>99j</b> moves to the last element of the current list.</li> <li>Updates lispy-back history.</li> <li>Moves downward to next to comment if issued from point at start of comment line (on the <b>; ;</b>).</li> </ul>
<u>Move forward to end of list</u>	<b>J</b>	( <a href="#">lispy-forward</a> ARG)	Move forward list ARG times or until error.
<u>Move backward to beginning of list</u>	<b>[</b>	( <a href="#">lispy-backward</a> ARG)	Move backward list ARG times or until error.
<u>Move up current list</u> <ul style="list-style-type: none"> <li>never exit current lis</li> </ul>	<b>k</b>	( <a href="#">special-lispy-up</a> ARG)	Move up ARG times inside current list. <ul style="list-style-type: none"> <li>Guaranteed to never exit the list: <b>99k</b> moves to the first element of the current list.</li> <li>Updates lispy-back history.</li> <li>Moves upward to previous to comment if issued from point at start of comment line (on the <b>; ;</b>).</li> </ul>
<u>Move left outward</u>	<b>h</b>	( <a href="#">special-lispy-left</a> ARG)	Move outside list backwards ARG times. Return nil on failure, t otherwise.
<u>Move outside list forward</u>	<b>l</b>	( <a href="#">special-lispy-right</a> ARG)	Move outside list forwards ARG times. <ul style="list-style-type: none"> <li>Parens in strings and comments are ignored.</li> <li>Updates lispy-back history.</li> </ul>
<u>Start knight hydra</u>	The <a href="#">hydra</a> starts with <b>z</b> and stops with any key except <b>j</b> and <b>k</b> . Useful to navigate disregarding syntax since it can escape a list (which normal <b>j</b> and <b>k</b> won't do).		
	<b>z</b>	( <a href="#">special-lh-knight/body</a> )	Start/Terminate the knight hydra
<u>Move down left-most parens on each line</u>	<ul style="list-style-type: none"> <li><b>z j</b></li> <li><b>j</b></li> </ul>	( <a href="#">lispy-knight-down</a> )	Move down left-most paren to the next line (can exit list)
<u>Move up left-most parens on each line</u>	<ul style="list-style-type: none"> <li><b>z k</b></li> <li><b>k</b></li> </ul>	( <a href="#">lispy-knight-up</a> )	Move up left-most paren to the previous line (can exit list)
	<b>oga</b>	( <a href="#">special-lispy-goto-def-ace</a> )	Jump to definition of selected element of current sexp. <ul style="list-style-type: none"> <li>Sexp is obtained by exiting list ARG times.</li> </ul>
	<b>ogb</b>	( <a href="#">special-pop-tag-mark</a> )	Pop back to where M-. was last invoked.
	<b>ogd</b>	( <a href="#">special-lispy-goto</a> )	Jump to symbol within files in current directory. <ul style="list-style-type: none"> <li>When ARG isn't nil, call 'lispy-goto-projectile' instead.</li> </ul>
	<b>oge</b>	( <a href="#">special-lispy-goto-elisp-commands</a> )	Jump to Elisp commands within current file. <ul style="list-style-type: none"> <li>When ARG is non-nil, force a reparse</li> </ul>
	<b>ogf</b>	( <a href="#">special-lispy-follow</a> )	Follow to 'lispy--current-function'.
	<b>ogj</b>	( <a href="#">special-lispy-goto-def-down</a> )	Jump to definition of ARGth element of current list.
	<b>ogl</b>	( <a href="#">special-lispy-goto-local</a> )	Jump to symbol within current file. <ul style="list-style-type: none"> <li>When ARG is non-nil, force a reparse.</li> </ul>
	<b>ogp</b>	( <a href="#">special-lispy-goto-projectile</a> )	Jump to symbol within files in ('projectile-project-root').
	<b>ogq</b>	( <a href="#">special-lispy-quit</a> )	Remove modifiers.
	<b>ogr</b>	( <a href="#">special-lispy-goto-recursive</a> )	Jump to symbol within files in current directory and its subdirectories.
<b>Navigation History</b>			
<u>Move back</u>	<b>b</b>	( <a href="#">special-lispy-back</a> ARG)	Move point to ARGth previous position in lisps-back history <ul style="list-style-type: none"> <li>If position isn't special, move to previous or error.</li> <li>Lispy back history updated by: <b>l</b>, <b>h</b>, <b>f</b>, <b>j</b>, <b>k</b>, <b>m</b>, <b>q</b>, and <b>i</b>.</li> </ul>
<u>View: center current sexp</u>	<b>v</b>	( <a href="#">special-lispy-view</a> )	Recenter current sexp to be on the first line of the window. When called twice in a row, recenter back to the original position.
<u>Visit another file</u>  See: <a href="#">☞ Projectile</a>	<b>V</b>	( <a href="#">special-lispy-visit</a> ARG)	Visit another file within this project using <a href="#">projectile</a> or <a href="#">find-file-in-project</a> . <ul style="list-style-type: none"> <li>Customize <b>lispy-visit-method</b> to select what function to use. <ul style="list-style-type: none"> <li>  PEL supports both of these external packages, and use the <b>pel-use-projectile</b> and <b>pel-use-find-file-in-project</b> user-options to download and activate each one. Unless you are familiar with <a href="#">find-file-in-project</a> you may find <a href="#">projectile</a> more useful and faster.</li> </ul> </li> <li>Use <b>V</b> to open the file in the current window.</li> <li>Use <b>2V</b> to open the file in another window.</li> </ul>
<b>Search</b>			
<u>Occur search inside the current top-level sexp</u>	<b>y</b>	( <a href="#">special-lispy-occur</a> )	Do an occur for the current top-level sexp. Go back-to-paren afterwards. This is useful e.g. to see where a particular variable is used within the current defun.
<b>Goto Definition</b>			

Description	Key	Function	Note
goto definition using directory tabgs	g	(special-lispy-goto &optional ARG)	Jump to symbol within files in <b>current directory</b> . Prompt for symbol and jump to it. <ul style="list-style-type: none"><li>When ARG isn't nil, call 'lispy-goto-projectile' instead.</li><li>See <b><a href="#">lispy goto wiki page</a></b>.</li></ul>
goto definition using projectile base directory	<ul style="list-style-type: none"><li>0g</li><li>ogp</li></ul>	(lispy-goto-projectile)	Jump to symbol within files in ('projectile-project-root').
goto definition in local file	G	(special-lispy-goto-local &optional ARG)	Similar to lispy-goto, but only current file's tags are used instead of whole directory's tags.
Follow: jump to definition	<ul style="list-style-type: none"><li>F</li></ul>	(special-lispy-follow)	When region is active jump to the definition of marked symbol. Otherwise jump to the definition of the first symbol in current sexp.
	<ul style="list-style-type: none"><li>M- .</li></ul>	(lispy-goto-symbol SYMBOL)	
Pop tag	<ul style="list-style-type: none"><li>D</li></ul>	(special-pop-tag-mark)	Go back from where it came with Follow
	<ul style="list-style-type: none"><li>M- ,</li></ul>	(pop-tag-mark)	
Narrow/Widening See also: <a href="#">☞ Narrowing</a>	<ul style="list-style-type: none"><li>Narrowing hides everything in the buffer except the selected region, allowing work on that region alone.</li><li>Widen it back to see the complete buffer again.</li></ul>		
Narrow current sexp   region	N	(special-lispy-narrow ARG)	Narrow current sexp or region.
Widen	W	(special-lispy-widen)	Widen back to see the complete buffer.
Cut/Paste/Mark/Hide/Indent			
Indent / hide/show outline	i	<ul style="list-style-type: none"><li>With no active region: (special-lispy-tab)</li></ul>	<ul style="list-style-type: none"><li>If inside outline: hide/show outline,</li><li>otherwise indent and prettify all code of current paren</li></ul>
mark car: select car of marked list	i	<ul style="list-style-type: none"><li>With active region: (lispy-mark-car)</li></ul>	Mark the car of currently active region. Moves point after the first symbol in the list. <ul style="list-style-type: none"><li>Updates lispy-back history.</li></ul>
Copy region or sexp to kill ring	n	(special-lispy-new-copy)	Copy marked region or sexp to kill ring.
Mark list	m	(special-lispy-mark-list ARG)	Mark the current sexp. If mark is already active, deactivate it instead. When ARG is more than 1, mark ARGth element. <ul style="list-style-type: none"><li>Updates lispy-back history.</li></ul>
Paste	P	(special-lispy-paste ARG)	When region is active, replace it with current kill. Forward to yank otherwise. <ul style="list-style-type: none"><li>When ARG is given, paste at that place in the current list.</li></ul>
Edit code	Transform code using the following commands		
undo	u	(special-lispy-undo)	Deactivate region and 'undo'.
clone	c	(special-lispy-clone ARG)	Clone sexp ARG times. <ul style="list-style-type: none"><li>When the sexp is top level, insert an additional newline.</li></ul>
Teleport: move current sexp to Ace target	t	(special-lispy-teleport ARG)	Move current sexp to Ace target inside current function. <ul style="list-style-type: none"><li>Use numerical argument to move that many sexp</li></ul>
	tt		Move current sexp to Ace target to any sexp inside current window.
	o<SPACE>	(special-lispy-other-space)	Alternative to 'lispy-space'.
Move current sexp to the left	oh	(special-lispy-move-left)	Move region left ARG times.
Move current sexp inside first element of list below	oj	(special-lispy-down-slurp)	Move current sexp or region into the next sexp.
Move current sexp to become last element of list above	ok	(special-lispy-up-slurp)	Move current sexp or region into the previous sexp. <ul style="list-style-type: none"><li>If the point is by itself on a line or followed only by right delimiters, slurp the point into the previous list.</li><li>This can be of thought as indenting the code to the next level and adjusting the parentheses accordingly.</li></ul>
Move current sexp to the right, outside current list	ol	(special-lispy-move-right)	Move region right ARG times.
Raise: use current sexp as replacement for its parent	r	(special-lispy-raise ARG)	Use current sexp or region as replacement for its parent. <ul style="list-style-type: none"><li>Do so ARG times.</li></ul>
Raise: current and next previous sexp as replacement for their parent	R	(special-lispy-raise-some)	Use current sexp and the following (if called from the left), or the preceeding (if called from the right) sexps, or the active region as replacement for their parent.
Move current sexp up	w	(special-lispy-move-up ARG)	Move current sexp or region up arg times. Don't exit the parent list. Also works for outlines.
Move sexp down in list	s	(special-lispy-move-down ARG)	Move current sexp or region down arg times. Don't exit the parent list. Also works for outlines.
Convolute: Exchange the order of application of 2 closest outer forms  Example animation	C	(special-lispy-convolute ARG)	Exchange the order of application of two closest outer forms, relative to current expression or region. <ul style="list-style-type: none"><li>Replace <code>(... (,,,  (</code> with <code>(,,, (...  (</code> where <code>...</code> and <code>,,,</code> is arbitrary code.</li><li>When ARG is more than 1, pull ARGth expression to enclose current sexp.</li><li>When ARG is nil, convolute only the part above sexp.</li></ul>
Slurp: grow either current sexp or region	>	(special-lispy-slurp ARG)	Grow either current sexp or region (if it's active) in appropriate direction. Opposite of lispy-barf. <ul style="list-style-type: none"><li>With an arg of 0, grow as far as possible.</li><li>With an arg of -1, grow until the end of the line where the current sexp ends or as far as possible before that position.</li></ul> <pre>(progn (foo) <u>f</u>(bar))  → &gt; → (progn <u>f</u>((foo) bar)) (progn (foo)_ (bar))  → &gt; → (progn (foo (bar))<u>f</u>)</pre>
Barf: shrink either current sexp or region	<	(special-lispy-barf ARG)	Shrink either current sexp or region (if it's active) in appropriate direction. Opposite of lispy-slurp. <pre>(progn (foo) (bar))_ → &lt; → (progn (foo))_(bar) (progn (foo) (bar))<u>f</u> → &lt; → (progn (foo) ( )<u>f</u>bar)</pre>
Splice the current list into the parent list	/	(special-lispy-splice ARG)	Splice ARG sexp into the containing (parent) list. Move the point to the next list to splice in appropriate direction. If there are none within the parent list, move to the parent list in appropriate direction. <pre>((<u>f</u>) (a) (b) (c))  → / → (a (<u>f</u>(b) (c)))</pre>

Description	Key	Function	Note
Convert current sexp into multi-line	M	(special-lispy-alt-multiline &optional SILENT)	Spread current sexp over multiple lines. <ul style="list-style-type: none"> <li>When SILENT is non-nil, don't issue messages.</li> <li>Especially useful on results of macroexpand.</li> </ul>
Turn current sexp into one line	O	(special-lispy-online)	Turn current sexp into one line. <ul style="list-style-type: none"> <li>Move comments ahead of sexp.</li> </ul>
Stringify current sexp	S	(special-lispy-stringify &optional ARG)	Transform current sexp into a string. <ul style="list-style-type: none"> <li>Quote newlines if arg isn't 1.</li> </ul>
Bind var: current sexp to let bound variable	xb	(lispy-bind-variable)	Transform the current list expression into a let-bound variable; iedit-mode is used to name the new variable. Use M-m to finish naming the variable. <p>Bind current expression as variable.</p> <p>'lispy-map-done' is used to finish entering the variable name. The bindings of 'lispy-backward' or 'lispy-mark-symbol' can also be used.</p>
turn nested if into cond	xc	(lispy-to-cond)	Transform current 'if' expressions to equivalent 'cond' expression.
	xC	(lispy-cleanup)	
turn current lambda into a defun	xd	(lispy-to-defun)	Turn the current lambda or toplevel sexp or block into a defun.
	xD	(lispy-extract-defun)	Extract the marked block as a defun. <ul style="list-style-type: none"> <li>For the defun to have arguments, capture them with 'lispy-bind-variable'</li> </ul>
	xe	(lispy-edebug ARG)	Start/stop edebug of current thing depending on ARG. <ul style="list-style-type: none"> <li>ARG is 1: 'edebug-defun' on this function.</li> <li>ARG is 2: 'eval-defun' on this function.</li> <li>ARG is 3: 'edebug-defun' on the function from this sexp.</li> <li>ARG is 4: 'eval-defun' on the function from this sexp.</li> </ul>
Inline current function or macro call	xf	(lispy-flatten ARG)	Inline current function or macro call, i.e. replace it with function body. The function should be interned and its body find-able. <ul style="list-style-type: none"> <li>Pass the ARG along.</li> </ul>
	xF	(lispy-let-flatten)	Inline a function at the point of its call using 'let'.
	xh	(lispy-describe)	Display documentation for 'lispy--current-function'
turn cond into nested if expressions	xi	(lispy-to-ifs)	Transform current 'cond' expression to equivalent 'if' expressions.
	xj	(lispy-debug-step-in)	Eval current function arguments and jump to definition
	xk	(lispy-extract-block)	Transform the current sexp or region into a function call. <ul style="list-style-type: none"> <li>The newly generated function will be placed above the current function.</li> <li>Starts the input for the new function name and arguments.</li> <li>To finalize this input, press "[".</li> </ul>
turn current defun into a lambda	xl	(lispy-to-lambda)	Turn the current function definition into a lambda.
	xm	(lispy-cursor-ace)	Add a cursor at a visually selected paren. <ul style="list-style-type: none"> <li>Currently, only one cursor can be added with local binding.</li> <li>Any amount can be added with a global binding.</li> </ul>
	xn	(lispy-cd)	Change the current REPL working directory.
	xp	(lispy-set-python-process)	
	xr	(lispy-eval-and-replace)	Eval last sexp and replace it with the result.
	xs	(save-buffer &optional ARG)	Save current buffer in visited file if modified. Same as <b>C-x C-s</b>
	xt	(lispy-view-test)	View better the test at point.
Unbind a let bound variable	xu	(lispy-unbind-variable)	Substitute let-bound variable <ul style="list-style-type: none"> <li>Unbind a let-bound variable. Also works for Clojure.</li> </ul>
	xv	(lispy-eval-expression)	Like 'eval-expression', but for current language (Emacs Lisp, Common Lisp, Clojure, etc..)
	xw	(lispy-show-top-level)	Show first line of top-level form containing point.
	xB	(lispy-store-region-and-buffer)	Store current buffer and 'lispy--bounds-dwim'.
	xR	(lispy-reverse)	Reverse the current list or region selection.
	XT	(lispy-ert)	Call ('ert' t) : run all ERT tests.
	x>	(lispy-toggle-thread-last)	Toggle current expression between last-threaded/unthreaded forms. <ul style="list-style-type: none"> <li>Macro used may be customized in 'lispy-thread-last-macro', which see.</li> </ul>
	<ul style="list-style-type: none"> <li>xC-h</li> <li>x?</li> </ul>	(lispy-x-more-verbosity)	
Outline operations			
Toggles on/off org-mode-like outline	I	(special-lispy-shifftab ARG)	Toggles on/off an org-mode-like outline. <ul style="list-style-type: none"> <li>To make this work, lispy-mode will modify outline-regexp and outline-level-function for the current buffer while it's on.</li> </ul>
Indent / hide/show outline	i	<ul style="list-style-type: none"> <li>With no active region: (special-lispy-tab)</li> </ul>	If in outline: hide/show outline, otherwise indent all code of current paren <ul style="list-style-type: none"> <li>When region is active, call 'lispy-mark-car'.</li> </ul>
Next outline level	J	(special-lispy-outline-next ARG)	Takes a numeric prefix arg and calls outline-next-visible-heading arg times or until past the last outline-regexp.
Previous outline level	K	(special-lispy-outline-prev ARG)	Takes a numeric prefix arg and calls outline-previous-visible-heading arg times or until past the first outline-regexp.
Evaluate Code			
Eval last sexp	e	(special-lispy-eval ARG)	Eval last sexp. Display result in echo area. <ul style="list-style-type: none"> <li>When ARG is 2, insert the result as a comment.</li> </ul>
Eval current region sexp. Insert result.	E	(special-lispy-eval-and-insert)	Eval current region or sexp. The result will be inserted in the current buffer after the evaluated expression.
Eval current sext & replace it at point	xr		

Description	Key	Function	Note
Eval current sexp in the content of the of the other window	p	(special-lispy-eval-other-window &optional ARG)	Eval current expression in the context of other window. <ul style="list-style-type: none"> <li>In case the point is on a let-bound variable, add a 'setq'.</li> <li>When ARG is non-nil, force select the window.</li> </ul>
EDebug current defun	xe		edebug current defun. Or cider-debug-defun-at-point for Clojure.
	2xe		2xe will eval current defun instead.
Debug - step in	xj		<ul style="list-style-type: none"> <li>Evaluate the arguments at the current function's call</li> <li>Jump to the function's definition</li> <li>Set the result of evaluation to the function's arguments</li> </ul>
EDebug stop	z	(special-lispy-edebug-stop)	Does the same as q in edebug, except current function's arguments will be saved to their current values. <ul style="list-style-type: none"> <li>This allows to continue debugging with lispy-eval (e) from edebug's current context.</li> <li>The advantage is that you can edit the code as you debug, as edebug puts your code in read-only mode.</li> </ul>
Execute Tests: run ert	xT		
Buffer/Region operations			
Store current buffer and region for further operation	xB		
Ediff regions	B	(special-lispy-ediff-regions)	Comparable to 'ediff-regions-linewise'. <ul style="list-style-type: none"> <li>First region and buffer come from 'lispy-store-region-and-buffer'</li> <li>Second region and buffer are the current ones.</li> </ul>