Lispy — A specialized modal editing for Lisp code🚧

Description	Key	Function	Note
Lispy : **Context-based modal editing of Lisp code**	The lispy minor mode provides modal editing to Emacs for Lisp-like languages. • Lisp is a very structured programming language, made of succession and combinations of S-expressions ("sexp"): lists that start with **(** and end with **)** "***paren***". Lispy takes advantage of the structure of Lisp code. • As long as point (the cursor) is before the left, opening, paren or after the right, closing paren, **the keys are interpreted as lispy commands**. • Keys in other locations are interpreted as usual. This table lists the lispy command keys, with links to the **Lispy function Reference** for each one. 📦 This requires the **lispy** external package. ⤵️ PEL downloads, installs and activates lispy when the **pel-use-lispy** user option is set to **t**. 🎛️ ☝️To get lispy mode run when Emacs visits a file of a specified mode, include the major mode in the PEL user-option **pel-modes-activating-lispy .** • 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.		
∑ Customize PEL use of Lispy and Lispy itself.	`<f11> <f2> SPC M-L`	(**pel-cfg-pkg-lisp** &optional OTHER-WINDOW)	Prompt to customize: 1. PEL lispy support for Emacs Lisp and Common Lisp 2. lispy itself. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
Toggle Lispy mode See also: • **𝔓𝔏 - Common Lisp** • **𝔈𝔓𝔏 - Emacs Lisp**	• `<f12> M-L` • `<M-f12> M-L` `<f11> SPC l M-L`	(**pel-lispy-mode** &optional ARG)	Toggle lispy-mode on/off. Lispy is a minor mode for navigating and editing LISP dialects. 📦 Requires lispy external package. ⤵️ PEL downloads, installs and configure it when **pel-use-lispy** user option is set to **t**. Please read the information on lispy web site. 💻 **pel-lispy-mode** calls **lispy-mode** but also prepares hydra, loaded dynamically with PEL. ☝️Set the **pel-modes-activating-lispy** user-option to activate lispy automatically for major modes.
Getting Code Help			
Describe function at point **See Also: ∑ Help/Info**	• `C-1`	(**lispy-describe-inline**)	Display documentation of current Lisp function: 'lispy--current-function' inline. • If docstring is small enough it is displayed in a pop-up box above point. Otherwise it is displayed inside a *lispy-help* buffer. 📦This requires the **lispy** external package. ⤵️ PEL downloads, installs and activates lispy when the **pel-use-lispy** user option is set to **t**.
	• `<f12> 1`		The **<f12> 1** key can be used even when lispy mode is not active.
Describe function arguments	• `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.
Describe function/ variable	`xh`		A shorthand for describe-function or describe-variable. • If you want to call describe-variable, you should mark the symbol first.
Inserting in code			
Insert a space	`Space`	(**lispy-space** ARG)	Insert one space, with position depending on ARG. • If ARG is 2, amend the current list with a space from current side. • If ARG is 3, switch to the different side beforehand. • If jammed between parens, "(|(" unjam: "(| (". • 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)".
Inserting comment	`;`	(**lispy-comment** &optional ARG)	Comment ARG sexps.
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. • When the region is active, wrap it in quotes instead. • When inside string, if ARG is nil quotes are quoted, otherwise the whole string is unquoted.
Move using avy	The following commands use avy-style highlighting to identify a word target to move to. Avy is similar to Ace. Lispy uses Avy internally. • By default the scope is the current list. Use a command numerical prefix to select a larger outer scope. • After hitting the command key, type the letter(s) identifying the target to move to that word and select it.		
ace symbol move • **Uses avy navigation** • **ARG sets target scope** • **ace highlight targets** • **move to selected one** • **mark selected word**	`a`	(**special-lispy-ace-symbol** ARG)	Jump to a symbol within the **current S-exp** and mark it. • Use ace method: each symbol in S-exp is shown with highlight letter: type that letter to move to the symbol. • 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: • Example: **M-3 a** selects 3 layers of enclosing S-exp to select ace targets.
ace sub-word • **ARG sets target scope** • **ace highlight targets** • **move to selected one** • **mark selected sub-word**	`-`	(**special-lispy-ace-subword** ARG)	Similar to lispy-ace-symbol, but selects a subword instead. • 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: • Example: **M-3 a** selects 3 layers of enclosing S-exp to select ace targets.
Move to Ace paren target	`q`	(**special-lispy-ace-paren** &optional ARG)	Highlights each **symbol** in current sexp as ace target and jump to the selected one. • Updates lispy-back history. • S-exp scope is obtained by exiting the list ARG times: default is 1
Move to Ace target char	`Q`	(**special-lispy-ace-char**)	Prompts for character, highlights each one in current sexp as ace target and jump to the selected one.
Navigate in code	The following commands move point inside code when point is before left paren or after right paren. Use **d** to switch side to control direction.		
Move to beginning of current defun	`A`		Forward to beginning-of-defun. When called twice in a row, restore the previous point and mark positions.
Move to different (other) side of sexp	`d`	(**special-lispy-different**)	Switch to the different side of current sexp. • If before '**(**' move after '**)**' and vice-versa.
Flow: move in the direction of current paren • **(➜ down** • **; ➜ down** • **) ➜ up**	`f`	(**special-lispy-flow** ARG)	Move in the direction of current paren inside current list and then to the next/previous list: • At left : move to next left paren (move going down the file). • At right: move to previous right parent (move going up the file). • Don't enter strings or comments. • Updates lispy-back history.
Move down current list • **never exit current list**	`j`	(**special-lispy-down** ARG)	Move down ARG times inside current list. • Guaranteed to never exit the list: **99j** moves to the last element of the current list. • Updates lispy-back history. • Moves downward to next to comment if issued from point at start of comment line (on the **;;**).

Description	Key	Function	Note
Move forward to end of list	`]`	(**lispy-forward** ARG)	Move forward list ARG times or until error.
Move backward to beginning of list	`[`	(**lispy-backward** ARG)	Move backward list ARG times or until error.
Move up current list • **never exit current lis**	`k`	(**special-lispy-up** ARG)	Move up ARG times inside current list. • Guaranteed to never exit the list: `99k` moves to the first element of the current list. • Updates lispy-back history. • Moves upward to previous to comment if issued from point at start of comment line (on the `;;`).
Move left outward	`h`	(**special-lispy-left** ARG)	Move outside list backwards ARG times. Return nil on failure, t otherwise.
Move outside list forward	`l`	(**special-lispy-right** ARG)	Move outside list forwards ARG times. • Parens in strings and comments are ignored. • Updates lispy-back history.
Start knight hydra	The hydra starts with `z` and stops with any key except `j` and `k`. Useful to navigate disregarding syntax since it can escape a list (which normal `j` and `k` won't do).		
	`z`	(**special-lh-knight/body**)	Start/Terminate the knight hydra
Move down left-most parens on each line	• `zj` • `j`	(**lispy-knight-down**)	Move down left-most paren to the next line (can exit list)
Move up left-most parens on each line	• `zk` • `k`	(**lispy-knight-up**)	Move up left-most paren to the previous line (can exit list)
Navigation History			
Move back	`b`	(**special-lispy-back** ARG)	Move point to ARGth previous position in lisps-back history • If position isn't special, move to previous or error. • Lispy back history updated by: `l`, `h`, `f`, `j`, `k`, `m`, `q`, and `i`.
View: center current sexp	`v`	(**special-lispy-view**)	Recenter current sexp to be on the first line of the window. When called twice in a row, recenter back to the original position.
Visit another file See: **∑ Projectile**	`V`	(**special-lispy-visit** ARG)	Visit another file within this project using projectile or find-file-in-project. • Customize **lispy-visit-method** to select what function to use. • 📦🛃 PEL supports both of these external packages, and use the **pel-use-projectile** and **pel-use-find-file-in-project** user-options to download and activate each one. Unless you are familiar with find-file-in-project you may find projectile more useful and faster. • Use `V` to open the file in the current window. • Use `2V` to open the file in another window.
Search			
Occur search inside the current top-level sexp	`y`	(**special-lispy-occur**)	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.
Goto Definition			
goto definition using directory tabgs	`g`	(**special-lispy-goto** &optional ARG)	Jump to symbol within files **in current directory**. Prompt for symbol and jump to it. • When ARG isn't nil, call 'lispy-goto-projectile' instead. • See **lispy goto wiki page**.
goto definition using projectile base directory	• `0g` • `ogp`	(**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	• `F`	(**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.
	• `M-.`	(**lispy-goto-symbol** SYMBOL)	
Pop tag	• `D`	(**special-pop-tag-mark**)	Go back from where it came with Follow
	• `M-,`	(**pop-tag-mark**)	
Narrow/Widening See also: **∑ Narrowing**	• Narrowing hides everything in the buffer except the selected region, allowing work on that region alone. • Widen it back to see the complete buffer again.		
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`	• With no active region: (**special-lispy-tab**)	• If inside outline: hide/show outline, • otherwise indent and prettify all code of current paren
mark car: select car of marked list	`i`	• With active region: (**lispy-mark-car**)	Mark the car of currently active region. Moves point after the first symbol in the list. • Updates lispy-back history.
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. • Updates lispy-back history.
Paste	`P`	(**special-lispy-paste** ARG)	When region is active, replace it with current kill. Forward to yank otherwise. • When ARG is given, paste at that place in the current list.
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. • When the sexp is top level, insert an additional newline.

Description	Key	Function	Note
Toggle to the other mode	`o`	**(special-lispy-other-mode** &optional ARG)	Prefix to the following other verbs: `key` `binding` `---` `-------` `SPC` special-lispy-other-space `g` special-lispy-goto-mode `h` special-lispy-move-left `j` special-lispy-down-slurp `k` special-lispy-up-slurp `l` special-lispy-move-right
Teleport: move current sexp to Ace target	`t`	**(special-lispy-teleport** ARG)	Move current sexp to Ace target inside current function. • Use numerical argument to move that many sexp
	`tt`		Move current sexp to Ace target to any sexp inside current window.
Move current sexp to the left	`oh`		
Move current sexp inside first element of list below	`oj`		
Move current sexp to become last element of list above	`ok`		
Move current sexp to the right, outside current list	`ol`		
Raise: use current sexp as replacement for its parent	`r`	**(special-lispy-raise** ARG)	Use current sexp or region as replacement for its parent. • Do so ARG times.
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.
Bind var: current sexp to let bound variable	`xb`		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.
Unbind a let bound variable	`xu`		Unbind a let-bound variable. Also works for Clojure.
turn current lambda into a defun	`xd`		
turn current defun into a lambda	`xl`		
turn nested if into cond	`xc`		
turn cond into nested if expressions	`xi`		
Inline current function or macro call	`xf`		Inline current function or macro call, i.e. replace it with function body. The function should be interned and its body find-able.
Convolute: Exchange the order of application of 2 closest outer forms	`C`		Exchange the order of application of two closest outer forms, relative to current expression or region.
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. • With an arg of 0, grow as far as possible. • 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.
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.
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.
Move to Ace target symbol & erase to replace	`H`	**(special-lispy-ace-symbol-replace** ARG)	Jump to a symbol within the current sexp and delete it, leaving point at location to type the new symbol. • Sexp is obtained by exiting the list ARG times. • Calls lispy-ace-symbol and deletes the selected symbol.
Convert current sexp into multi-line	`M`	**(special-lispy-alt-multiline** &optional SILENT)	Spread current sexp over multiple lines. • When SILENT is non-nil, don't issue messages. • Especially useful on results of macroexpand.
Turn current sexp into one line	`O`	**(special-lispy-oneline)**	Turn current sexp into one line. • Move comments ahead of sexp.
Stringify current sexp	`S`	**(special-lispy-stringify** &optional ARG)	Transform current sexp into a string. • Quote newlines if arg isn't 1.
Outline operations			
Toggles on|off org-mode-like outline	`I`		Toggles on/off an org-mode-like outline. • To make this work, lispy-mode will modify outline-regexp and outline-level-function for the current buffer while it's on.
Indent / hide/show outline	`i`	• With no active region: **(special-lispy-tab)**	If in outline: hide/show outline, otherwise indent all code of current paren • When region is active, call 'lispy-mark-car'.
Next outline level	`J`		Takes a numeric prefix arg and calls outline-next-visible-heading arg times or until past the last outline-regexp.
Previous outline level	`K`		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. • When ARG is 2, insert the result as a comment.
Eval current region|sexp. Insert result.	`E`		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`		
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`		• Evaluate the arguments at the current function's call • Jump to the function's definition • Set the result of evaluation to the function's arguments
EDebug stop	`Z`		Does the same as q in edebug, except current function's arguments will be saved to their current values. • This allows to continue debugging with lispy-eval (e) from edebug's current context. • The advantage is that you can edit the code as you debug, as edebug puts your code in read-only mode.
Execute Tests: run ert	`xT`		
Buffer/Region operations			
Store current buffer and region for further operation	`xB`		
Ediff regions	`B`		