Lispy minor mode 🚧

Description	Key	Function	Note
Lispy - Single letter commands to manipulate Lisp-like code	The lispy 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***". As long as point (the cursor) is before the left, opening, paren or the right, closing paren, the keys are interpreted as lispy commands. Keys in other locations are interpreted as usual. The nature of the Lips programming languages enables this type of modal editing that is even more powerful than Vi-type modal editing. 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**.		
∑ **Customize** PEL and Emacs Lispy support	**<f11> <f2> SPC M-L**	(**pel-cfg-pkg-lisp** &optional OTHER-WINDOW)	Customize support for Lisp programming languages - A group that also contains the groups for Emacs Lisp and Common Lisp: lispy. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window.
Getting Help on code			
Describe function at point **See Also:** ∑ **Help/Info**	• **C-1** • **<f12> 1**	(**lispy-describe-inline**)	Display documentation for '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**.
Describe function arguments	• **C-2** • **<f12> 2**	(**lispy-arglist-inline**)	Show the argument list of current function.
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.
Navigate inside code	The following commands move point inside code when point is before left paren or after right paren.		
ace symbol move	**a**	(**special-lispy-ace-symbol** ARG)	Jump to a symbol within the current sexp and mark it. • Use ace method: each symbol in sexp is shown with highlight letter: type that letter to move to the symbol. • Sexp is obtained by exiting the list ARG times.
ace sub-word	**-**		Similar to lispy-ace-symbol, but selects a subword instead.
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.**
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	**f**	(**special-lispy-flow** ARG)	Move in the direction of current paren: • 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 left outward	**h**	(**special-lispy-left** ARG)	Move outside list backwards ARG times. Return nil on failure, t otherwise.
Move down 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.
Move up current list	**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.
Start knight hydra	**z**		Start/Terminate the knight hydra
Move down left-most parens on each line	• **zj** • **j**		Move down left-most paren to the next line (can exit list)
Move up left-most parens on each line	• **zk** • **k**		Move up left-most paren to the previous line (can exit list)
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.
Move to Ace target	**q**		Highlights each **symbol** in current sexp as ace target and jump to the selected one. • Updates lispy-back history.
Move to Ace target char	**Q**		Prompts for character, highlights each one in current sexp as ace target and jump to the selected one.
View: center current sexp	**v**		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	**V**		Visit another file within this project using projectile or find-file-in-project (customize lispy-visit-method to choose). • Use V to call projectile-find-file. Use 2V to call projectile-find-file-other-window.
Search			
Occur search inside the current top-level sexp	**y**		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 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**		Similar to lispy-goto, but only current file's tags are used instead of whole directory's tags.
Follow: jump to definition	• **F** • **M-.**		When region is active jump to the definition of marked symbol. Otherwise jump to the definition of the first symbol in current sexp.
Pop tag	• **D** • **M-,**		Go back from where it came with Follow
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.		

Description	Key	Function	Note
Narrow current sexp| region	`N`		Narrow current sexp or region.
Widen	`W`		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 in outline: hide/show outline, otherwise indent all code of current paren • When region is active, call 'lispy-mark-car'.
mark car	`i`	• With active region: (**lispy-mark-car**)	Mark the car of current thing. • 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`		When region is active, replace it with current kill. Forward to yank otherwise.
Edit code	Transform code using the following commands		
clone	`c`	(**special-lispy-clone** ARG)	Clone sexp ARG times. • When the sexp is top level, insert an additional newline.
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`		
Raise: current and next| previous sexp as replacement for their parent	`R`		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 sexp down in list	`s`		Move current sexp or region down arg times. Don't exit the parent list. Also works for outlines.
undo	`u`		
Move current sexp up	`w`		Move current sexp or region up 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	`>`		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	`<`		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	`/`		Splice the current list into the 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 beginning of current defun	`A`		Forward to beginning-of-defun. When called twice in a row, restore the previous point and mark positions.
Teleport: move current sexp to Ace target	`t`		Move current sexp to Ace target inside current function
	`tt`		Move current sexp to Ace target to any sexp inside current window
Move to Ace target symbol & erase to replace	`H`		Calls lispy-ace-symbol and deletes the selected symbol.
Convert current sexp into multi-line	`M`		Extend current sexp into multiple lines. Especially useful on results of macroexpand.
Turn current sexp into one line	`O`		Turn current sexp into one line.
Stringify current sexp	`S`		Transform current sexp into a string. Quote newlines if arg isn't 1.
Insert a space	`Space`		
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.

Description	Key	Function	Note
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`		
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`		