



































































Copy, Cut & Paste — Copy/Delete/Kill/Yank

Operation	Keystroke	Function	Note
Cut/paste/delete/copy <ul style="list-style-type: none"> Help & Customization OS Clipboard Commands showing copied/cut text duplicate line copy commands kill & delete manage kill ring browse kill-ring Delete 1 character (delete key) Kill/Delete elements(s) - word - symbol, line, sentence, paragraph - s-expression, list, function, filename, URL, rectangle, comment Kill/delete whitespace Hungry deletion of whitespace Yank / Paste popup-kill-ring 	<p>Emacs supports exceptionally powerful combinations of text cut/paste/delete/copy operations called kill, yank, delete, specialized delete and copy.</p> <p>👉 A kill operation stores the text inside a kill-ring buffer which can be retrieved through a yank operation. When text is <i>deleted</i>, no copy is retained.</p> <p>📅 Emacs pre-dates the IBM publication of the Common User Access (CUA) standard and uses different names for similar concepts.</p> <ul style="list-style-type: none"> In Emacs terminology: <ul style="list-style-type: none"> “<i>kill</i>” represents an operation similar to the CUA “cut”, “<i>yank</i>” represents an operation that is similar to the CUA “paste”. <p>👉 Emacs yank always insert, even when the buffer is in overwrite-mode, unless you activate delete-selection-mode and select the area to yank into.</p> <ul style="list-style-type: none"> PEL provides pel-overwrite-yank that can overwrite text in a buffer that is in overwrite-mode. See the yank section for details. <p>👉 See section Delete 1 Character to control the behaviour of the delete key. Emacs supports several behaviours for this key.</p> <p>PEL provides the following enhancements, installed and activated when the corresponding pel-use- user-option is turned on:</p>		
	📦 browse-kill-ring	🔗 pel-use-browse-kill-ring	Browse and manipulate the kill ring in a specialized buffer.
	📦 popup-kill-ring	🔗 pel-use-popup-kill-ring	Pop the kill-ring content in overlay and select entry for it. Graphics mode only. <ul style="list-style-type: none"> Requires pos-tip and popup external packages (installed by PEL).
Last updated on:	2025-12-18		
Open this PDF file. See also: 🔗 Help/Info	<ul style="list-style-type: none"> <f11> = <f1> <f11> - <f1> 	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the 🔗 Cut & Paste local PDF. If the prefix argument (like C-u or M--) is used, then it opens the remote GitHub hosted raw PDF instead. If the pel-flip-help-pdf-arg user-option is set it's the other way around.
Customize PEL support for cut & paste	<ul style="list-style-type: none"> <f11> = <f2> <f11> - <f2> 	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL support for cut and paste. <ul style="list-style-type: none"> If OTHER-WINDOW is non-nil (use C-u), display in other window.
Customize Emacs support for cut & paste	<f11> - <f3>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs cut and paste groups: editing-basics, browse-kill-ring, cua-mode, killing, popup-kill-ring. <ul style="list-style-type: none"> If OTHER-WINDOW is non-nil (use C-u), display in other window.
Show Delete/Cut/Copy/Paste behaviour information	<ul style="list-style-type: none"> <f11> = ? <f11> - ? <f11> DEL ? 	(pel-ccp-info &optional APPEND)	Display current buffer's delete, cut, copy & paste behaviour. <ul style="list-style-type: none"> Clear previous buffer content unless optional APPEND argument is non-nil, in which case it appends to the previous report. Displays information in a *pel-ccp-info* buffer. The variable names are buttons allowing quick access to their help and customization buffer. The information shown depends on the current major mode and the activated minor modes.
OS Clipboard Commands	<ul style="list-style-type: none"> When Emacs runs in graphical mode, the following commands can be used to copy and paste from the OS (system) clipboard. 📱 On macOS: <ul style="list-style-type: none"> with Emacs in graphical mode: this can also be done using the standard macOS keys: ⌘-c, ⌘-v and ⌘-x keystrokes. with emacs running under Terminal.app, you can use ⌘-v to paste from the OS-clipboard. <ul style="list-style-type: none"> When xterm-mouse-mode is off you can select text by marking it with the mouse, then use ⌘-c to copy to the OS clipboard. ⌘-x does not work in terminal mode. The PEL package binds <f11> <f12> to xterm-mouse-mode in terminal mode and change the way text selection works with the mouse. 		
Copy text to clipboard	<ul style="list-style-type: none"> <f11> C-c c 	(clipboard-kill-ring-save BEG END &optional REGION)	Copy region to kill ring, and save in the OS clipboard. <ul style="list-style-type: none"> 📱 In terminal mode, when the xtem-mouse-mode is off, the ⌘-c key copies text, but copies the terminal text: to copy multiple lines, ensure there is only one Emacs window horizontally.
	<ul style="list-style-type: none"> ⌘-c 	(ns-copy-ncluding-secondary)	📱 In graphics mode ⌘-c copies the text via the Emacs application and invokes the (ns-copy-including-secondary) function.
Paste text from clipboard	<ul style="list-style-type: none"> <f11> C-c v ⌘-v 	(clipboard-yank)	Insert the OS clipboard contents, or the last stretch of killed text. <ul style="list-style-type: none"> 📱 In graphics mode ⌘-v executes the standard (yank &optional ARG) which supports the clipboard. With Emacs running inside a macOS Terminal.app frame, the key will bring text from the clipboard but slowly and may fail to paste everything properly.
Cut region & place both in kill ring and on system clipboard	<ul style="list-style-type: none"> <f11> C-c x ⌘-x 	(clipboard-kill-region BEG END &optional REGION)	Kill the region, and save it in the OS clipboard.
Showing Copied/Cut Text	👁 Most PEL commands that copy and cut/kill text can also display that text in the echo area at the bottom of the screen if the pel-show-copy-cut-text user option is set to t or its buffer local value controlled by pel-toggle-show-copy-cut-text command (bound to <f11> - -) sets it to t. <ul style="list-style-type: none"> The commands that can display the copied/cut/kill text are identified by the special symbol 🏷 showing in the first column. 		
Toggle display of copied/cut/killed text.	<f11> - -	(pel-toggle-show-copy-cut-text &optional GLOBALLY)	Toggle display of copied/cut text. <ul style="list-style-type: none"> By default change behaviour in local buffer only. With optional GLOBALLY argument (use any prefix argument), change it for all buffers. Display new state. The change does not persist across Emacs sessions. <ul style="list-style-type: none"> To modify the global state permanently modify the customized value of the pel-show-copy-cut-text user option. You can use the <f11> - <f2> or the <f11> = <f2> key sequences to open the relevant customize buffer.
Duplicate Line	PEL provides text duplication commands with optional text replacement of marked text. Nothing is copied in the mark ring.		
Duplicate current line • replace any marked text	<f6> d	(pel-duplicate-line &optional N)	Duplicate the current line N times. N defaults to 1. Nothing is copied to the kill ring. <ul style="list-style-type: none"> Use numeric argument to specify a different number: M-5 <f6> d inserts 5 duplicates.
	<ul style="list-style-type: none"> Insert new line(s) below and move point to the last one entered, at the same relative position inside the line. If some text on the original line is marked, the function prompts for a replacement, and replace each instance of that text in the duplicated line. <ul style="list-style-type: none"> If N is negative the replacement is only done for the marked area. When (abs N) > 1 : insert that many duplicated lines, and prompts for a new replacement for each new line. <ul style="list-style-type: none"> The prompt maintains its history (accessible via M-p and M-n). With N = 0, the command behaves as if N=1 but it does not move point. Repeat it with <f5> to duplicate several lines without moving point. 		

Operation	Keystroke	Function	Note
Copy Commands	Emacs copy commands copy text into the “ <i>kill ring</i> ” . Other commands are used to take text from the kill ing and insert it in the buffer. <ul style="list-style-type: none"> By default, Emacs does not support the CUA compliant C-c for copy. To support that key you must enable the cua-mode. Some of the commands display the copied text inside the echo area. That can be useful to see what some commands copied, for example to distinguish what copying a word or symbol does and being able to see what a word or symbol is in the major mode of the current buffer. The echo area is cleared on the next key pressed. The commands are listed in order of the size/type of text copied: <ul style="list-style-type: none"> 1) character, whitespace 2) word, symbol 3) filename/url 4) line 5) function, list/sexp 6) sentence, paragraph All of the following commands, except the one for rectangle can show the copied text in the echo area. <ul style="list-style-type: none"> Those are marked with:  Activated by the pel-show-copy-cut-text by user-option. Toggle this display with <f11> - - 		
Copy region or line at point  ★PEL Enhanced Key ★ See also: <ul style="list-style-type: none"> 🔗 Marking 🔗 Numkeypad 	<ul style="list-style-type: none"> M-w <f11> = 1 <f11> = = <f11> + <f11> <kp-add> <kp-add> <kp-separator> 	(pel-copy-marked-or-whole-line)	Flexible copy to kill ring.: copy visible region if any, otherwise copy current line to kill ring. <ul style="list-style-type: none"> Replaces standard binding to kill-ring-save which only copies region On macOS terminal (TTY) mode the keypad+ key is interpreted as <kp-separator>. For environments where keypad+ maps to <kp-add> (as its the case in Terminals for some Linux distributions, set the pel-keypad+-is-kp-add user-option to t to activate the key.
		 See the 🔗 Marking table to mark (select) a text region to use with this command. <p>The copy operation is controlled by the (optional) argument:</p> <ul style="list-style-type: none"> If N = 0: copy region (regardless of whether it is visible or not. If a region is active/visible: copy the region's text. if no region is active/visible copy N lines: <ul style="list-style-type: none"> If no argument, (N=1) copy current line. If N > 0: copy current line and N-1 following lines. If l < 0: copy current line and N-1 previous lines. <p>All copied lines are complete. The copied text is saved in the kill-ring.</p> <p>All copy operations are performed by 'kill-ring-save' (the original binding for that key).</p> <ul style="list-style-type: none"> In graphics mode: text is also copied to the OS clipboard. 	
Copy complete word at point  See also: <ul style="list-style-type: none"> 🔗 Numkeypad 🔗 Text Modes 	<ul style="list-style-type: none"> <f11> = w C-<kp-add> 	(pel-copy-word-at-point)	Copy word at point. Shows the text copied in the echo area. <ul style="list-style-type: none"> See table 🔗 Text Modes for information on text modes that affects this.
		<ul style="list-style-type: none"> The <f11> t m ? command displays the mode and the <f11> t m prefix allows modification of the mode. See changing the word mode to include or exclude some characters as word delimiters: <ul style="list-style-type: none"> subword-mode . To toggle that mode: <f11> t m b superword-mode . To toggle that mode: <f11> t m p 	
Copy complete symbol at point  See also: 🔗 Numkeypad	<ul style="list-style-type: none"> <f11> = . M-+ M-<kp-add> 	(pel-copy-symbol-at-point)	Copy symbol at point.  The syntax of what constitutes a symbol depends on the syntax table for the buffer and therefore on the major mode of the current buffer.
		 In terminal mode of some Linux distribution, the M- <kp-add> is not recognized.  PEL tries to identify these systems and use another key binding identified by the pel-keypad-meta+-special-sequence user-option (it identifies M-O 3 k for Linux for instance). If the key sequence for your environment running in terminal mode is different set pel-keypad-meta+-special-sequence to another value: enter a string: it will be passed to the kbd function.	
Copy character at point 	<f11> = c	(pel-copy-char-at-point &optional N)	Copy single character at point. <ul style="list-style-type: none"> With argument N, copy N consecutive characters; a negative N copies the character backwards (before point).
Copy whitespaces at point 	<f11> = SPC	(pel-copy-whitespace-at-point)	Kill all whitespace characters at/ around point on current line.
Copy filename at point 	<f11> = F	(pel-copy-filename-at-point)	Copy filename at point.
Copy URL at point 	<f11> = u	(pel-copy-url-at-point)	Copy URL at point.
Copy line beginning 	<f11> = a	(pel-copy-line-start)	Copy text from the beginning of the current line up to point.
Copy line end 	<f11> = e	(pel-copy-line-end)	Copy text from point up to the end of the line.
Copy function at point 	<f11> = f	(pel-copy-function-at-point)	Copy complete body of function at point.
Copy list at point 	<f11> = ((pel-copy-list-at-point)	Copy and show complete Lisp-syntax list at point. <ul style="list-style-type: none"> Copy from anywhere inside the list: copies the <i>entire</i> list.
Copy S-expression at point 	<f11> = x	(pel-copy-sexp-at-point)	Copy and show complete Lisp S-expression at point. For Lisp code see also 🔗 L- Lispy . Point must be at the start parenthesis or right after the closing parenthesis otherwise it does not copy. In particular it will not copy if point is <i>inside</i> the list.
Copy complete sentence at point 	<f11> = s	(pel-copy-sentence-at-point)	Copy entire sentence at point.  Toggle the minimum number of spaces that end a sentence with: pel-toggle-sentence-end : <f11> t m s
Copy paragraph beginning 	<f11> = b	(pel-copy-paragraph-start)	beginning of paragraph to point.
Copy paragraph 	<f11> = H	(pel-copy-paragraph-at-point)	Copy entire paragraph at point.
Copy paragraph end 	<f11> = h	(pel-copy-paragraph-end)	Copy from point to end of paragraph.
Save rectangle text See also: 🔗 Rectangles	<ul style="list-style-type: none"> C-x r M-w <f11> = r 	(copy-rectangle-as-kill START END)	Copy the region-rectangle and save it as the last killed one.

Operation	Keystroke	Function	Note
		(delete-forward-char N &optional KILLFLAG)	Delete following N characters (previous if N is negative). N defaults to 1. When region is marked: region is deleted, regardless of argument and state of delete-selection-mode.
Delete character - backward	<ul style="list-style-type: none"> DEL 	(backward-delete-char-untabify ARG &optional KILLP)	Deletes character before cursor (deletes backward), replaces hard tab with spaces as required. With arguments: <ul style="list-style-type: none"> positive numeric argument: kill that many characters backward negative numeric argument: kill that many characters forward When region is marked: the region is deleted, regardless of argument.
Kill trough next occurrence of char	M-z char	(zap-to-char ARG CHAR)	Kill up to and including ARGth occurrence of CHAR. Case is ignored if 'case-fold-search' is non-nil in the current buffer. Goes backward if ARG is negative; error if CHAR not found.
Delete & Kill element(s) <ul style="list-style-type: none"> word 	The following PEL commands delete words symbols, paragraphs, S-expressions (sexp), functions, etc... They do not retain information in the kill ring. None of these commands operate on read-only buffers. <ul style="list-style-type: none"> PEL provide similar commands to kill the same entities, see them in the kill section below. All of the following commands, except the one for rectangle can show the deleted text in the echo area. <ul style="list-style-type: none"> Those are marked with: Activated by the pel-show-copy-cut-text by user-option. Toggle this display with <f11> - - 		
Delete complete word at point	<ul style="list-style-type: none"> <f11> DEL w <f11> w 	(pel-delete-word-at-point)	Delete the complete word at point, regardless of point's position inside the word.
Delete part of word at point	<ul style="list-style-type: none"> <f11> DEL q <f11> q 	(pel-delete-word-part &optional BEGINNING)	Delete the end of word at point: from point to end of current word. <ul style="list-style-type: none"> With any prefix argument delete the beginning of word up to current point.
Kill word backward	<ul style="list-style-type: none"> M- C- 	(backward-kill-word ARG)	Kill characters backward until beginning of word.
Kill word (forward) <ul style="list-style-type: none"> stop at punctuation, whitespace 	<ul style="list-style-type: none"> C-S- M-d 	(kill-word ARG)	By default kill forward from point up to the end of the current word. <ul style="list-style-type: none"> Numeric argument specify number of consecutive words. Negative argument reverses the direction.
Kill word forward and delete whitespace after it. <ul style="list-style-type: none"> deletes actuation and whitespace after last work deleted. 	M-D	(pel-kill-word-and-whitespace ARG)	Kill word forward and <i>delete</i> the whitespace following it. <ul style="list-style-type: none"> Numeric argument specify number of consecutive words. Negative argument reverses the direction. Whitespace is deleted only after the last of the words killed. If punctuation follows the last deleted word it is also deleted, like whitespace. Consecutive execution save the consecutive words in kill ring, but with only 1 space between each word (even newlines are replaced by a single space)
Kill word at point <ul style="list-style-type: none"> kill complete word 	<ul style="list-style-type: none"> <f11> - w C-<kp-subtract> 	(pel-kill-word-at-point)	Kill the complete word at point, regardless of point's position inside the word.
Kill part of word at point <ul style="list-style-type: none"> kill part after point (- before) 	<f11> - q	(pel-kill-word-part &optional BEGINNING)	Kill the end of word at point: from point to end of current word. <ul style="list-style-type: none"> With any prefix argument kill the beginning of word up to current point.
<ul style="list-style-type: none"> symbol 	symbol		
Kill symbol at point	<ul style="list-style-type: none"> <f11> - . M-<kp-subtract> 	(pel-kill-symbol-at-point)	Kill the complete word at point as identified by word and symbol syntactic unit, regardless of point's position inside the word. This is useful in source code files when the subword-mode and superword-mode are not activated; it kills all consecutive characters that include symbol characters such as '-'. <ul style="list-style-type: none"> The keypad key binding can sometimes be made available on some terminals, but not all. Customize pel-kill-symbol-at-point-terminal-binding to bind something else.
Customize via: <f11> - <f2>			
Kill part of current symbol at point	<f11> - ,	(pel-kill-symbol-part &optional BEGINNING)	Kill the end of symbol at point: from point to end of current symbol. <ul style="list-style-type: none"> With any prefix argument kill the beginning of symbol up to current point.
Delete complete symbol at point	<ul style="list-style-type: none"> <f11> DEL . <f11> . 	(pel-delete-symbol-at-point)	Delete the complete word at point as identified by word and symbol syntactic unit, regardless of point's position inside the word. This is useful in source code files when the subword-mode and superword-mode are not activated; it deletes all consecutive characters that include symbol characters such as '-'.
Delete part of current symbol at point	<ul style="list-style-type: none"> <f11> DEL , <f11> , 	(pel-delete-symbol-part &optional BEGINNING)	Delete the end of symbol at point: from point to end of current symbol. <ul style="list-style-type: none"> With any prefix argument delete the beginning of symbol up to current point.
<ul style="list-style-type: none"> Line 	line		
Kill whole line	C-S-	(kill-whole-line &optional ARG)	Deletes current line (in graphics mode). Use C-w instead, it is more flexible, see above.
Delete beginning of line	<ul style="list-style-type: none"> <f11> DEL a <f11> a 	(pel-delete-from-beginning-of-line)	Deletes the beginning of the line up to the cursor.
Kill beginning of line	<ul style="list-style-type: none"> M-0 C-k C-\ <f11> - a 	(pel-kill-from-beginning-of-line)	Kills the beginning of the line up to the cursor. In terminal the M binding does not work properly, and they do different things! <ul style="list-style-type: none"> M-< M-S-< The binding works properly in graphics mode.
Delete to end of line	<ul style="list-style-type: none"> C-K <f11> DEL e <f11> e 	(pel-delete-line)	Delete text from cursor to end of line.
Kill to end of line	<ul style="list-style-type: none"> M- C-k <f11> - e 	(kill-line &optional ARG)	Kills from current position to end of line. If no visible characters on it kill through newline. <ul style="list-style-type: none"> With prefix argument ARG, kill that many lines from point. Negative arguments kill lines backward. With zero argument, kills the text before point on the current line. If you want to append the killed line to the last killed text, use C-M-w before C-k. If the buffer is read-only, Emacs will beep and refrain from deleting the line, but put the line in the kill ring anyway essentially performing a copy to kill ring. M- is bound to (insert-parentheses &optional ARG) as in M-(in terminal mode. The M- binding works properly in graphics mode. With PEL, instead of killing (and copying the text to the kill ring), you can delete to end of line with pel-delete-to-eol, bound to C-K (in graphics mode only) and <f11> - E .
Delete duplicate lines	<ul style="list-style-type: none"> <f11> DEL * <f11> * 	(delete-duplicate-lines BEG END &optional REVERSE ADJACENT KEEP-BLANKS INTERACTIVE)	Delete all but one copy of any identical lines in the region (or entire buffer if nothing marked). <ul style="list-style-type: none"> If REVERSE is non-nil (interactively, with a C-u prefix), it searches backwards and keeps the last instance of each repeated line. Identical lines need not be adjacent, unless the argument ADJACENT is non-nil (interactively, with a C-u C-u prefix). This is a more efficient mode of operation, and may be useful on large regions that have already been sorted. If the argument KEEP-BLANKS is non-nil (interactively, with a C-u C-u C-u prefix), it retains repeated blank lines. Prints a message describing the number of deletions.
<ul style="list-style-type: none"> Sentence 	sentence		
Delete sentence at point	<ul style="list-style-type: none"> <f11> DEL s <f11> s 	(pel-delete-sentence-at-point)	Delete complete sentence at point.
Kill sentence at point	<f11> - s	(pel-kill-sentence-at-point)	Kill complete sentence at point.
Kill sentence - backward	C-x 	(backward-kill-sentence &optional ARG)	Kill back from point to start of sentence. With arg, repeat, or kill forward to Nth end of sentence if negative arg -N.

Operation	Keystroke	Function	Note
Kill sentence - forward	M-k	(kill-sentence &optional ARG)	Kill from point to end of sentence. With arg, repeat; negative arg -N means kill back to Nth start of sentence.
• Paragraph	paragraph		
Delete complete paragraph at point 	<ul style="list-style-type: none"> • <f11> DEL H • <f11> ☒ H 	(pel-delete-paragraph-at-point &optional N)	Delete complete paragraph at point. With argument N, delete N consecutive paragraphs; a negative N deletes the current one and N-1 previous paragraphs.
Kill complete paragraph at point 	<f11> - H	(pel-kill-paragraph-at-point &optional N)	Kill complete paragraph at point. With argument N, kill N consecutive paragraphs; a negative N kills the current one and N-1 previous paragraphs.
Kill back to start of paragraph 	<ul style="list-style-type: none"> • <f11> DEL b • <f11> ☒ b 	(pel-backward-delete-paragraph ARG)	Delete back to start of paragraph. With arg N, delete back to Nth start of paragraph; negative arg -N means delete forward to Nth end of paragraph.
Kill back to start of paragraph  	<f11> - b	(backward-kill-paragraph ARG)	Kill back to start of paragraph. With arg N, kill back to Nth start of paragraph; negative arg -N means kill forward to Nth end of paragraph.
Delete forward to end of paragraph 	<ul style="list-style-type: none"> • <f11> DEL h • <f11> ☒ h 	(pel-delete-paragraph ARG)	Delete forward to end of paragraph. With arg N, delete forward to Nth end of paragraph; negative arg -N means delete backward to Nth start of paragraph.
Kill forward to end of paragraph  	<f11> - h	(kill-paragraph ARG)	Kill forward to end of paragraph. With arg N, kill forward to Nth end of paragraph; negative arg -N means kill backward to Nth start of paragraph.
• S-Expression	S-expression		
Delete Lisp S-Expression at point 	<ul style="list-style-type: none"> • <f11> DEL x • <f11> ☒ x 	(pel-delete-sexp-at-point)	Delete the S-Expression at point. The point must be at the opening parenthesis or just after the closing parenthesis.
Kill Lisp S-Expression at point 	<f11> - x	(pel-kill-sexp-at-point)	Kill the S-Expression at point. The point must be at the opening parenthesis or just after the closing parenthesis.
Delete previous Lisp S-expr 	<ul style="list-style-type: none"> • <f11> DEL [• <f11> ⌵ [(pel-backward-delete-sexp &optional ARG)	Delete the sexp (balanced expression) preceding point. <ul style="list-style-type: none"> • With ARG, delete that many sexps before point. • Negative arg -N means delete N sexps after point. • This command assumes point is not in a string or comment.
Kill previous Lisp S-expression	<ul style="list-style-type: none"> • C-M-⤴ • <f11> - [• C-[C-⤴ • Esc C-⤴ 	(backward-kill-sexp &optional ARG)	Kill the sexp (balanced expression) preceding point. <ul style="list-style-type: none"> • With ARG, kill that many sexps before point. • Negative arg -N means kill N sexps after point. • This command assumes point is not in a string or comment. <p>⚠ Note: In some text (like The Common Lisp Cookbook - Using Emacs as a Lisp IDE), the C-M-⤴ - <backspace> keystroke is being described to kill the previous sexp. This key does not seem to be used anymore. This key sequence is normally not accessible in terminal mode as it would map to C-M-h instead.</p> <p>The C-M-⤴ binding only works in terminal mode. Since this key-sequence is not the best match for the operation, use any of the alternatives or M- - C-M-k instead.</p>
Delete next Lisp S-expression 	<ul style="list-style-type: none"> • <f11> DEL] • <f11> ☒] 	(pel-delete-sexp &optional ARG)	<ul style="list-style-type: none"> • No argument: delete the next sexp (or the current from the point forward). • With negative sign: delete the previous sexp (the sexp backward). <ul style="list-style-type: none"> • For example: M- - <f11> DEL] deletes the sexp backward. • With numeric argument: delete that many sexp in the direction identified by the sign of the argument.
Kill next Lisp S-expression	<ul style="list-style-type: none"> • C-M-k • <f11> -] • C-[C-k • Esc C-k 	(kill-sexp &optional ARG)	<ul style="list-style-type: none"> • No argument: kill the next sexp (or the current from the point forward). • With negative sign: kill the previous sexp (the sexp backward). <ul style="list-style-type: none"> • For example: M- - C-M-k kills the sexp backward. • With numeric argument: kill that many sexp in the direction identified by the sign of the argument.
• Lisp List	lisp list		
Delete Lisp list at point 	<ul style="list-style-type: none"> • <f11> DEL (• <f11> ☒ ((pel-delete-list-at-point)	Delete the balanced expression at point: a block of text between parentheses, braces, squared or angled bracket, single or double quotes. Point must be located at the opening block character. For Lisp code see also ⌘- Lispy .
Kill Lisp list at point 	<f11> - ((pel-kill-list-at-point)	Kill the balanced expression at point: a block of text between parentheses, braces, squared or angled bracket, single or double quotes. Point must be located at the opening block character.
• Function	function		
Delete function at point 	<ul style="list-style-type: none"> • <f11> DEL f • <f11> ☒ f 	(pel-delete-function-at-point)	Delete the function at point. Point can be anywhere, or just past the function code. <ul style="list-style-type: none"> • Deletes the complete body of the function.
Kill function at point 	<f11> - f	(pel-kill-function-at-point)	Kill the function at point. Point can be anywhere, or just past the function code. <ul style="list-style-type: none"> • Kills the complete body of the function.
• Filename	filename		
Delete filename at point 	<ul style="list-style-type: none"> • <f11> DEL F • <f11> ☒ F 	(pel-delete-filename-at-point)	Delete the filename at point. Point can be located anywhere inside the file name or right after.
Kill filename at point 	<f11> - F	(pel-kill-filename-at-point)	Kill the filename at point. Point can be located anywhere inside the file name or right after.
• URL	url		
Delete URL at point 	<ul style="list-style-type: none"> • <f11> DEL u • <f11> ☒ u 	(pel-delete-url-at-point)	Delete the URL at point. Point can be located anywhere inside the file name or right after.
Kill URL at point 	<f11> - u	(pel-kill-url-at-point)	Kill the URL at point. Point can be located anywhere inside the file name or right after.
• Rectangle area	rectangle area		
Delete text in rectangle See also: ⌘ Rectangles	<ul style="list-style-type: none"> • <f11> DEL r • <f11> ☒ r 	(pel-delete-rectangle START END &optional FILL)	Delete the rectangle region.
Kill text in rectangle See also: ⌘ Rectangles	<ul style="list-style-type: none"> • C-x r k • <f11> - r 	(kill-rectangle START END &optional FILL)	Delete the region-rectangle and save it as the last killed one. <ul style="list-style-type: none"> • If the buffer is read-only, Emacs will beep and refrain from deleting the rectangle, but put it in ‘killed-rectangle’ anyway. This means that you can use this command to copy text from a read-only buffer. (If the variable ‘kill-read-only-ok’ is non-nil, then this won’t even beep.)
• Comments in area	comments in area		
Delete all comments in buffer or marked region See also: ⌘ Comments	<ul style="list-style-type: none"> • <f11> ; DEL • <f11> ; ☒ 	(pel-delete-all-comments)	Delete all comments in current (possibly narrowed) buffer or marked region. <p>👉 To delete all comments inside a region mark the region first. You can also narrow a region and then use this command to remove all comments from that narrowed region, without affecting anything else. See the ⌘ Narrowing table for information on narrowing.</p>
Kill all comments in buffer or marked region (& retain them in kill ring) See also: ⌘ Comments	<f11> - ;	(pel-kill-all-comments)	Kill all comments in current (possibly narrowed) buffer or marked region and retain them in kill ring. <p>👉 To kill all comments inside a region mark the region first. You can also narrow a region and then use this command to remove all comments from that narrowed region, without affecting anything else. See the ⌘ Narrowing table for information on narrowing.</p>

Operation	Keystroke	Function	Note
Delete whitespace See also: ↗ Whitespace	The following Emacs commands delete whitespaces. The deleted characters are not copied in the kill ring . These commands are also described in the Text Whitespace table.		
Delete all spaces between point and next/previous non-white	<ul style="list-style-type: none"> C- Fn C- 	(pel-delete-to-next-visible &optional n)	Delete all whitespace between point and next non-whitespace character (forwards). <ul style="list-style-type: none"> Repeat N times with numeric argument. Delete backwards if N is negative.  Useful to delete the current word when point is at the beginning of the word. Note:  on macOS laptop, type: “Fn C-delete”.
Delete all whitespace at point	<ul style="list-style-type: none"> <f11> DEL SPC <f11>  SPC 	(pel-delete-whitespace-at-point)	Delete all whitespace at and around point on a single line.
Kill whitespace at point 	<f11> - SPC	(pel-kill-whitespace-at-point)	Kill all whitespace characters at/around point on current line. Copy them to kill ring.
Delete empty/whitespace lines in region or all buffer	<ul style="list-style-type: none"> <f11> DEL M-SPC <f11>  M-SPC 	(pel-delete-all-empty-lines &optional BEGIN END)	Delete all empty lines from marked area or the entire buffer if nothing is marked.
Delete all spaces between 2 words	M-\	(delete-horizontal-space &optional BACKWARD-ONLY)	Delete all spaces and tabs around point. Only works when cursor is on the spaces between the words or on the first character of the second word.
Delete all spaces but one beween words	M-SPC	(just-one-space &optional N)	Delete all spaces and tabs around point, leaving one space (or N spaces). If N is negative, delete newlines as well, leaving -N spaces. <ul style="list-style-type: none"> This command ensures that words are separated by just one space character. The cursor may be between the words but can also be on the fist character of the word. At the end of the word it inserts a space.
Delete all contiguous blank lines after point	C-x C-o	(delete-blank-lines)	<ul style="list-style-type: none"> On blank line, delete all surrounding blank lines, leaving just one. On isolated blank line, delete that one. On nonblank line, delete any immediately following blank lines.
Delete Indentation, join this line to the previous one See also: ↗ Indentation ↗ Whitespace	<ul style="list-style-type: none"> M-^ <f11>  6 <f6> 6 	(delete-indentation &optional ARG)	Join this line to previous and fix up whitespace at join. <ul style="list-style-type: none"> If there is a fill prefix, delete it from the beginning of this line. With argument, join this line to following line.
Join this line with next line	<ul style="list-style-type: none"> <f11>  7 <f6> 7 	(pel-join-next-line)	Join this line to following line.
Cycle spacing around point	<f11> t w .	(cycle-spacing &optional N PRESERVE-NL-BACK MODE)	Manipulate whitespace around point in a smart way. <ul style="list-style-type: none"> The first call in a sequence acts like ‘just-one-space’. It deletes all spaces and tabs around point, leaving one space (or N spaces). N is the prefix argument. If N is negative, it deletes newlines as well, leaving -N spaces. (If PRESERVE-NL-BACK is non-nil, it does not delete newlines before point.) The second call in a sequence deletes all spaces. The third call in a sequence restores the original whitespace (and point). The easiest way to use this command for the second or third call (or further) is to issue it once and then use the repeat command (C-x z or <f5>).
Delete all trailing whitespaces	<f11> t w t	(delete-trailing-whitespace &optional START END)	Delete trailing whitespace in the entire (or narrowed part of the) buffer or in the marked region. <ul style="list-style-type: none"> This command deletes whitespace characters after the last non-whitespace character in each line between START and END. It does not consider formfeed characters to be whitespace. If this command acts on the entire buffer, it also deletes all trailing lines at the end of the buffer if the variable ‘delete-trailing-lines’ is non-nil.
Cleanup whitespace Removes excess whitespaces: trailing whitespace, unnecessary or excessive indentation.	<f11> t w c	(whitespace-cleanup)	Cleanup some blank problems (non-required whitespace) in all buffer or at region. <ul style="list-style-type: none"> It usually applies to the whole buffer, but in transient mark mode when the mark is active, it applies to the region. It also applies to the region when it is not in transient mark mode, the mark is active and C-u was pressed just before calling ‘whitespace-cleanup’ interactively.
Hungry Deletion of Whitespace	The CC mode provides two commands that can perform “hungry whitespace deletion” that can also be used in every mode . <ul style="list-style-type: none">  PEL provides the convenient keys with the <f11> prefix keys for those 2 commands, available in all modes. In modes compatible with the CC Mode (e.g. for C, C++, D, Java, Pike, etc..) it is also possible to activate the Hungry Delete Mode to modify the behaviour of the simple and C-d, to perform hungry deletions. That’s not currently supported in other modes. <ul style="list-style-type: none"> When the Hungry Delete Mode is on, the mode-line displays a ‘h’ to the right of the ‘/I’ indication of electric mode. The Hungry Mode also activates the key prefixes below that start with C-c. They are listed but remember they are only available once the Hungry state mode is activated (and that can only be done in modes that are CC Mode compatible). In modes derived from CC Mode you can also activate the hungry state to make standard delete commands delete hungrily, but that does not work for other modes. PEL provides the <f12> M-DEL key for those modes. See the specific modes for more info. 		
Delete preceding char or all preceding whitespace.	<ul style="list-style-type: none"> C-c DEL C-c  C-c C- C-c C-<backspace> C-c C-DEL 	(c-hungry-delete-backwards)	Delete the preceding character or all preceding whitespace back to the previous non-whitespace character.  In terminal mode, even though C-␣ , C-<backspace> and C-DEL are not available, they are mapped to the non-control key so attempting to type them end up invoking the command anyway because the first key bindings are recognized.  With PEL, the <f11>   binding is available in all modes . The other keys are only available in modes derived from the CC Mode.
	<ul style="list-style-type: none"> <f11>   <f11> DEL DEL 		
Delete next char or all following whitespace.	<ul style="list-style-type: none"> C-c C-d C-c  C-c C- C-c C-<delete> 	(c-hungry-delete-forward)	Delete the following character or all following whitespace up to the next non-whitespace one.  In terminal mode, even though C-␣ and C-<delete> are not available, they are mapped to the non-control key so attempting to type them end up invoking the command anyway because the first key bindings are recognized.  With PEL, the <f11>  binding is available in all modes . The other keys are only available in modes derived from the CC Mode.
	<ul style="list-style-type: none"> <f11>  		

Operation	Keystroke	Function	Note
Yank / Paste	Emacs calls “ yanking ” the action of inserting previously killed or copied text, retrieved it from the “ <i>kill ring</i> ”. Other editors call this “ <i>pasting text</i> ”.		
Select text stored in kill ring	<f10> → Edit → Select and Paste		Use the Select and Paste menu entry to list each entry of the kill ring and insert it at point.
Yank last killed into buffer See also: 📖 Numkeypad <i>Special cases:</i> <ul style="list-style-type: none"> Using C-v: <ul style="list-style-type: none"> pel-with-cua-paste Using <kp-0> on some situations might require: <ul style="list-style-type: none"> pel-keypad-0-is-kp-yank 	<ul style="list-style-type: none"> C-y %-v <insert> <kp-0> C-v (see note) 	(yank &optional ARG)	Reinsert ("paste") the last stretch of killed text. More precisely, reinsert the most recent kill , which is the stretch of killed text most recently killed OR yanked. Put point at the end, and set mark at the beginning without activating it. With just C-u as argument, put point at beginning, and mark at end. With argument N, reinsert the Nth most recent kill . <ul style="list-style-type: none"> %-v In graphical mode: supports OS clipboard. With PEL, <kp-0> which is also the location of the <insert> key on some keyboard, performs the same yank operation when the keypad numlock is off. <ul style="list-style-type: none"> ⚠️ On some situations, like when when using Emacs on a Linux host accessed through ssh, this may not work. Try setting pel-keypad-0-is-kp-yank to t. With PEL, if pel-with-cua-paste user option is set to t, C-v is bound to yank, otherwise it uses Emacs default (used for scrolling).
		(pel-overwrite-yank &optional ARG)	PEL pel-overwrite-yank is bound to C-y effectively replacing Emacs’ standard yank (while the original Emacs yank command is still available). pel-overwrite-yank can overwrite text, instead of inserting, when the buffer is in overwrite-mode. <ul style="list-style-type: none"> This behaviour is controlled <ul style="list-style-type: none"> globally by the pel-activate-overwrite-yank user-option, and that can be overridden in each buffer by executing the pel-toggle-overwrite-yank command, bound to <f11> <f4> C-y
Toggle yank overwriting. 📑	<f11> <f4> C-y	(pel-toggle-overwrite-yank)	Toggle pel-overwrite-yank’s ability to overwrite text when the current buffer is in overwrite-mode. The original (global) settings is determined by he pel-activate-overwrite-yank user-option. <ul style="list-style-type: none"> Access the customization buffer with <f11> = <f2>
Replace line with last kill	<f11> C-y	(pel-replace-line-with-kill)	Replace the current line with the content of last kill. <ul style="list-style-type: none"> Change what was just yanked right away with M-y 💡 Useful, when overwrite mode is off, to quickly replace the content of a line with another.
Paste from OS clipboard	%-y	(ns-paste-secondary)	🍏 On macOS in graphics mode only: paste from OS clipboard (not from kill ring).
Replace last yank with previous kill	M-y	(yank-pop &optional ARG)	Replace just-yanked stretch of killed text with a different stretch. <ul style="list-style-type: none"> This command is allowed only immediately after a ‘yank’ or a ‘yank-pop’. At such a time, the region contains a stretch of reinserted previously-killed text. ‘yank-pop’ deletes that text and inserts in its place a different stretch of killed text. With no argument, the previous kill is inserted. With argument N, insert the Nth previous kill. If N is negative, this is a more recent kill. The sequence of kills wraps around, so that after the oldest one comes the newest one. 🗨️ Also referred to as: “yank next”.
Pop-up menu with kill ring content, to select entry to insert at point. <ul style="list-style-type: none"> Available in Graphics Mode only. 	<f11> M-y	(popup-kill-ring)	Pop-up a menu that shows all entries in kill ring, allowing insertion of a specified kill ring entry at point. <ul style="list-style-type: none"> While the pop-up menu is available, it’s also possible to perform interactive search in kill ring text: only matching entries will now show in the pop-up men 📦 Requires the popup-kill-ring package and its pre-requisites pos-tip and popup 📦 PEL activates this when the pel-use-popup-kill-ring user option is set to t . <ul style="list-style-type: none"> Use <f11> - <f2> to access its customization group.

Cut & Paste — References

Topic & Link	Notes
GNU Emacs Manual: Killing and Moving Text	
GNU Emacs Manual: Killing - Yanking	
Copy & Paste	
Emacs Wiki - Copy and Paste	
simpleclip	
Emacs Wiki - Deleting Whitespace	
Delete without storing to Kill Ring	
Emacs: how to delete text without kill ring? @ StackOverflow	
Emacs: Deleting a line without sending it to the kill ring @ StackExchange	
Backspace without adding to kill ring @ Stack Exchange	
Kill or copy current line with minimal keystrokes	
show-marks.el @ Emacs Wiki	