↖ Topic Index

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

Operation	Keystroke	Function	Note
Cut/paste/delete/copy • OS Clipboard Commands • showing copied/cut text • browse kill-ring • duplicate line ○ copy commands ○ kill & delete • manage kill ring • Delete character • Kill/Delete word ○ Kill/delete symbol, line, sentence, paragraph ○ Kill/delete S-expression • Kill/delete whitespace ○ Hungry deletion of whitespace • Yank / Paste • popup-kill-ring Last updated on:			Emacs supports exceptionally powerful combinations of text cut/paste/delete/copy operations called kill, yank, delete, specialized delete and copy. PEL enhances it further and provides the following external packages: 📦 The **browse-kill-ring** external package activated by **pel-use-browse-kill-ring** user-option. 📦 The **popup-kill-ring** external package and its pre-requisites **pos-tip** and **popup** activated by **pel-use-popup-kill-ring**. Graphics mode only. ☝ A kill operation stores the text inside a **kill-ring** buffer which can be retrieved through a yank operation. When text is ***deleted***, no copy is retained. Emacs pre-dates the IBM publication of the **Common User Access (CUA)** standard and uses different names for similar concepts. • In Emacs terminology: • ***"kill"*** represents an operation similar to the CUA "**cut**", • ***"yank"*** represents an operation that is similar to the CUA "**paste**". ☝ 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. • PEL provides **pel-overwrite-yank** that can overwrite text in a buffer that is in overwrite-mode. See the yank section for details. ☝ See section Delete 1 Character to control the behaviour of the delete key. Emacs supports several behaviours for this key. 2025-07-21
Open this PDF file. See also: **Help/Info**	• `<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	• `<f11> = <f2>` • `<f11> - <f2>`	(**pel-customize-pel** &optional OTHER-WINDOW)	Customize PEL support for cut and paste. • 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: browse-kill-ring, cua-mode, killing, popup-kill-ring. • If OTHER-WINDOW is non-nil (use `C-u`), display in other window.
OS Clipboard Commands			• When Emacs runs in graphical mode, the following commands can be used to copy and paste from the OS (system) clipboard. • On macOS: • 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. • 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	• `<f11> C c`	(**clipboard-kill-ring-save** BEG END &optional REGION)	Copy region to kill ring, and save in the OS clipboard. 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.
	• `⌘-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	• `<f11> C v` • `⌘-v`	(**clipboard-yank**)	Insert the OS clipboard contents, or the last stretch of killed text. 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	• `<f11> 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> M-=`) sets it to t. • 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> M-=`	(**pel-toggle-show-copy-cut-text** &optional GLOBALLY)	Toggle display of copied/cut text. • 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. • 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.
Browse kill ring 📦 Requires **browse-kill-ring** external package activated by **pel-use-browse-kill-ring**.	`C-c y`	(**browse-kill-ring**)	Display items in the 'kill-ring' in another buffer. • Several options are available. See **browse-kill-ring** customization buffer. • One option allow to highlight text being killed, another allows showing killed text in its original buffer & location when selected in the browse buffer. • With PEL `<f11> - <f3> 1` opens the **browse-kill-ring** customization buffer.
Inside *Kill Ring* buffer ====>			• Navigate in that buffer to see kill entry and select one with RET to insert it inside the edited buffer. ☝ Inside the *Kill Ring* buffer It's possible to edit and delete entries from the **kill ring**. There are several other available commands:. `C-g` browse-kill-ring-quit; `RET` browse-kill-ring-insert-and-quit.; `U` browse-kill-ring-undo-other-window; `a` browse-kill-ring-append-insert; `b` browse-kill-ring-prepend-insert; `d` browse-kill-ring-delete; `e` browse-kill-ring-edit; `g` browse-kill-ring-update; `i` browse-kill-ring-insert; `l` browse-kill-ring-occur; `n` browse-kill-ring-forward; `o` browse-kill-ring-insert-and-move; `p` browse-kill-ring-previous; `q` browse-kill-ring-quit; `r` browse-kill-ring-search-backward; `s` browse-kill-ring-search-forward; `u` browse-kill-ring-insert-move-and-quit.; `h` describe-mode; `x` browse-kill-ring-insert-and-delete; `y` browse-kill-ring-insert
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. • Use numeric argument to specify a different number: `M-5 <f6> d` inserts 5 duplicates.
			• 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. • 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. • 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 "*kill ring*" . Other commands are used to take text from the kill ing and insert it in the buffer. • 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: • 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. • Those are marked with: Activated by the **pel-show-copy-cut-text** by user-option. Toggle this display with **<f11> M-=**		
Copy region or line at point **★PEL Enhanced Key ★** See also: • **Marking** • **Numkeypad**	• **M-w** • **<f11> = l** • **<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. ☛ 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. The copy operation is controlled by the (optional) argument: • 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: • If no argument, (N=1) copy current line. • If N > 0: copy current line and N-1 following lines. • If I < 0: copy current line and N-1 previous lines. All copied lines are complete. The copied text is saved in the kill-ring. All copy operations are performed by `**kill-ring-save**' (the original binding for that key). ☛ In graphics mode: text is also copied to the OS clipboard.
Copy complete word at point See also: • **Numkeypad** • **Text Modes**	• **<f11> = w** • **C-<kp-add>**	(**pel-copy-word-at-point**)	Copy word at point. Shows the text copied in the echo area. ☝ See table **Text Modes** for information on text modes that affects this. • 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: • 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**	• **<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. • 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. • Copy from anywhere inside the list: copies the *entire* 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 **Pl- 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 *inside* 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**	• **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
Deleting Text **Kill Commands**	Emacs supports "deleting" and "killing" text. Deleted text is **not** retained. Killed text is retained in the "**kill ring**". Emacs kill commands erase text and copy it into the kill ring. Several commands below can show the killed text in the echo area. • Those are marked with: 👁 Activated by the **pel-show-copy-cut-text** by user-option. Toggle this display with **<f11> M-=**		
Manage Kill Ring	The following are examples of commands that can be used to show the kill ring and the various variables that control it. The kill-ring is an Emacs variable. It can be manipulated by Emacs Lisp code and its content can be shown using the help variable command. The maximum number of elements inside the kill ring is also controllable. ☝ See the (**browse-kill-ring**) command above. It provides ability to edit the content of the kill ring through a *Kill Ring* buffer.		
Display content of kill ring	**<f1> v kill-ring RET**		Display the content of the kill ring in the *Help* buffer
Display kill ring size	**<f1> v kill-ring-max RET**		Display the maximum number of kill ring entries in the *Help* buffer.
Set kill ring size	M-x set-variable RET kill-ring-max RET		The variable **kill-ring-max** is the number of entries in the kill ring. Defaults to 60.
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.
Toggles delete selection mode See also: Σ **Marking** Σ **Text Modes**	**<f11> t m d**	(**delete-selection-mode**)	Toggles delete selection-mode on/off. • In delete-selection-mode typing a character while a region is active replaces the entire region with what is typed. By default delete selection-mode is off.
Kill/Delete marked region/ line(s) 👁 ★PEL Enhanced Key ★ Available in PEL non-numlock mode. See: Σ⌨Numkeypad See also: • Σ **Marking** • Σ⌨ **Numkeypad**	• **C-w** • **<f11> - l** • **<kp-subtract>** • **⌘-x**	(**pel-kill-or-delete-marked-or-whole-line** &optional N)	Flexible region/whole-line kill/delete. Argument controls behaviour (see next cell below). ☝ In graphics mode this also copies text to the OS clipboard. ☝ With PEL in non-numlock mode, the **<keypad-subtract>** (the keypad **-** key) is bound to this command. On macOS in graphics mode only: PEL rebinds **⌘-x** from (kill-region) to this command, making this easy to use key able to perform more. ☝ See the Σ **Marking** table to mark (select) a text region to use with this command.
	• N=0 := kill region (active/visible or not) • Sign of N selects operation: positive := kill (default) negative := delete • ***Select text to delete/kill based on presence of region***: • if a region is marked: kill/delete region's text, • if no region: kill/delete abs(N) lines, start at point. • If operation is to kill 1 line and the line is empty, then **delete line** instead of killing it. • **Scenarios:** • With no arg: • with no active/visible region: **kill current line**, but if line is empty delete it. • with an active/visible region: **kill region's text**. • With arg **0**: (**M-0 C-w**) : **kill region's text**, whether region is active/visible or not. • With a non zero arg: • With no region active/visible: • With arg **-** : (**M-- C-w**) or (**C-- C-w**) : delete current line • With arg **- 1** : (**M-- 1 C-w**) or (**C-- 1 C-w**) : delete current line • With arg **4**: (**M-4 C-w**) : kill 4 lines including current one. • With arg **-3**: (**M-- 3 C-w**) : delete 3 lines including current one. • With a region active/visible: • With any negative mark argument: delete the region's text. • With no argument or any positive argument: kill the region's text. ✂ This replaces the standard Emacs binding to kill-region which always kill text between mark and point, even when the region is not marked. When text is killed it is killed with kill-region, so it retains the filtering and kill ring text appending capabilities.		
Append to Kill Ring	• **C-M-w** • **C-[C-w** • **Esc C-w**	(**append-next-kill** &optional INTERACTIVE)	Preparation command. Next kill command issued after this will add to the top of the kill ring item (the previous kill): • If the next command kills forward from point, the kill is appended to the previous killed text. • If the command kills backward, the kill is prepended. If the next command is not a kill command, this has no effect.
Kill text between point and mark	**S-⌦**	(**kill-region** BEG END &optional REGION)	Kill text between mark and point, even if region is not marked. See also: **C-w** above.
Delete 1 Character ☝➡	**Delete Keys:** Emacs recognizes 2 delete keys: 1) a **delete forward** and 2) a **delete backward** (backspace). Some keyboards have both, others have only one of them (e.g. macOS laptop keyboards). On those the forward delete key is composed with the **Fn** key and the backspace key. The behaviour of the delete keys is controlled by the **normal-erase-is-backspace** variable, which can be customized and controlled by executing the command normal-erase-is-backspace-mode. See: Emacs Manual -- If <DEL> Fails to Delete. ◉ This table uses the ⌦ and ⌫ symbols to represent these 2 keys: 1. ⌦ := "*forward delete*" := **<deletechar>** := **Fn ⌫** 2. ⌫ := "*backward delete*" := **<backspace>** Often labelled "delete" on keyboards. ⚠ **C-⌦** and **C-⌫** are not accessible in terminal mode.		
Kill character at point 👁	**<f11> - c**	(**pel-kill-char-at-point** &optional N)	Kill single character at point. With argument N, kill N consecutive characters; a negative N kills characters backwards.
Delete character - forward	**C-d**	(**delete-char** N &optional KILLFLAG)	Delete following N characters (previous if N is negative). N defaults to 1. When region is marked: region is only deleted if delete-selection-mode is on.
	⌦	(**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	• **DEL** • ⌫	(**backward-delete-char-untabify** ARG &optional KILLP)	Deletes character before cursor (deletes backward), replaces hard tab with spaces as required. With arguments: • 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) • **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. • 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. • Those are marked with: 👁 Activated by the **pel-show-copy-cut-text** by user-option. Toggle this display with **<f11> M-=**		
Delete complete word at point 👁	• **<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 👁	• **<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. • With any prefix argument delete the beginning of word up to current point.
Kill word backward	• **M-⌫** • **C-⌫**	(**backward-kill-word** ARG)	Kill characters backward until beginning of word.

Operation	Keystroke	Function	Note
Kill word (forward) • stop at punctuation, whitespace	• C-S-⌦ • M-d	(**kill-word** ARG)	By default kill forward from point up to the end of the current word. • Numeric argument specify number of consecutive words. Negative argument reverses the direction.
Kill word forward and delete whitespace after it. • deletes actuation and whitespace after last work deleted.	M-D	(**pel-kill-word-and-whitespace** ARG)	Kill word forward and *delete* the whitespace following it. • 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 • kill complete word	• <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 • 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. • With any prefix argument kill the beginning of word up to current point.
• **symbol**	**symbol**		
Kill symbol at point Customize via: <f11> - <f2>	• <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 '-'. • 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.
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. • With any prefix argument kill the beginning of symbol up to current point.
Delete complete symbol at point	• <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	• <f11> DEL , • <f11> ⌫ ,	(**pel-delete-symbol-part** &optional BEGINNING)	Delete the end of symbol at point: from point to end of current symbol. • With any prefix argument delete the beginning of symbol up to current point.
• **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	• <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	• 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! • M-<⌫> binds to C-<backspace> executing backward-kill-word. • M-S-<⌫> binds to (mark-defun &optional ARG) instead (which is bound to C-M-h). The binding works properly in graphics mode.
Delete to end of line	• C-K • <f11> DEL e • <f11> ⌫ e	(**pel-delete-line**)	Delete text from cursor to end of line.
Kill to end of line	• 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. • 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	• <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). • 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.
• **Sentence**	**sentence**		
Delete sentence at point	• <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.
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	• <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	• <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	• <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.

Operation	Keystroke	Function	Note
• **S-Expression**	**S-expression**		
Delete Lisp S-Expression at point	• **<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	• **<f11> DEL [** • **<f11> ⌫ [**	(**pel-backward-delete-sexp** &optional ARG)	Delete the sexp (balanced expression) preceding point. • 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	• **C-M-⌦** • **<f11> - [** • **C-[C-⌦** • **Esc C-⌦**	(**backward-kill-sexp** &optional ARG)	Kill the sexp (balanced expression) preceding point. • 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. ⚠ 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. 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.
Delete next Lisp S-expression	• **<f11> DEL]** • **<f11> ⌫]**	(**pel-delete-sexp** &optional ARG)	• No argument: delete the next sexp (or the current from the point forward). • With negative sign: delete the previous sexp (the sexp backward). • 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	• **C-M-k** • **<f11> -]** • **C-[C-k** • **Esc C-k**	(**kill-sexp** &optional ARG)	• No argument: kill the next sexp (or the current from the point forward). • With negative sign: kill the previous sexp (the sexp backward). • 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	• **<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 **𝔐- 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	• **<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. • 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. • Kills the complete body of the function.
• **Filename**	**filename**		
Delete filename at point	• **<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	• **<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**	• **<f11> DEL r** • **<f11> ⌫ r**	(**pel-delete-rectangle** START END &optional FILL)	Delete the rectangle region.
Kill text in rectangle See also: **Rectangles**	• **C-x r k** • **<f11> - r**	(**kill-rectangle** START END &optional FILL)	Delete the region-rectangle and save it as the last killed one. • 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**	• **<f11> ; DEL** • **<f11> ; ⌫**	(**pel-delete-all-comments**)	Delete all comments in current (possibly narrowed) buffer or marked region. ☝ 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.
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. ☝ 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.
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	• **C-⌦** • **Fn C-⌫**	(**pel-delete-to-next-visible** &optional n)	Delete all whitespace between point and next non-whitespace character (forwards). • 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	• **<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	• **<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.

Operation	Keystroke	Function	Note
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. • 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**)	• 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: Σ **Indentation** Σ **Whitespace**	• **M-^** • **<f11> ⌫ 6** • **<f6> 6**	(**delete-indentation** &optional ARG)	Join this line to previous and fix up whitespace at join. • 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	• **<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. • 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. • 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. • 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**.
	• ☝ 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 **<DEL>** and **C-d**, to perform hungry deletions. That's not currently supported in other modes.
	  • When the Hungry Delete Mode is on, the mode-line displays a 'h' to the right of the '//l' 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.	• **C-c DEL** • **C-c ⌫** • **C-c C-⌫** • **C-c C-<backspace>** • **C-c C-DEL** • **<f11> ⌫ ⌫** • **<f11> DEL 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.
Delete next char or all following whitespace.	• **C-c C-d** • **C-c ⌦** • **C-c C-⌦** • **C-c C-<delete>** • **<f11> ⌦**	(**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.

Yank / Paste	Emacs calls "**yanking**" the action of inserting previously killed or copied text, retrieved it from the "*kill ring*". Other editors call this "*pasting text*".		

Yank last killed into buffer
See also: Σ⌨ **Numkeypad**
 	
Special cases:
• Using **C-v**:
  • **pel-with-cua-paste**
• Using **<kp-0>** on some situations might require:
  • **pel-keypad-0-is-kp-yank**
	• **C-y**
	• **⌘-v**
	• **<insert>**
	• **<kp-0>**
	• **C-v** (see note)
		(**yank** &optional ARG)
		
		(**pel-overwrite-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**.
			• **⌘-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.
			  • ⚠ 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 implements **pel-overwrite-yank** and binds it to **C-y** effectively replacing Emacs' standard yank (while the yank command is still available an un-modified). **pel-overwrite-yank** can overwrite text, instead of inserting, when the buffer is in overwrite-mode. This behaviour is controlled
			• 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. • Access the customization buffer with **<f11> = <f2>**
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. • 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. • 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. • 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**. • 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	