























Cut & Paste — Copy/Delete/Kill/Yank

Operation	Keystroke	Function	Note
Emacs Cut & Paste	<div>Emacs pre-dates the IBM publication of the Common User Access (CUA) standard and uses different names for similar concepts.</div> <ul style="list-style-type: none">In Emacs terminology, the word “kill” represents an operation similar to the CUA “cut”, and the word “yank” represents an operation that is similar to the CUA paste.A kill operation stores the text inside a kill-ring buffer which can be retrieved through a yank operation. However, when text is deleted, no copy is not retained. <div>This page describes the various commands available in Emacs to perform these operations.</div>		
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 local copy of the 🔗 Cut & Paste PDF file unless a command prefix (like C-u) was used. In that case it opens the Github-hosted file instead.
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 this can also be done using the standard This can also be done with the standard ⌘-c, ⌘-v and ⌘-x keystrokes.🍏 On macOS, with emacs running under Terminal.app, you can use ⌘-v to paste from the OS-clipboard. And 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.<ul style="list-style-type: none">The PEL package binds <f11> <f12> to xterm-mouse-mode when Emacs runs in terminal mode. Use this to change the way text selection works with the mouse.		
Copy text to clipboard	<ul style="list-style-type: none"><f11> C c⌘-c	<ul style="list-style-type: none">(clipboard-kill-ring-save BEG END & optional REGION)(ns-copy-including-secondary)	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, so if you want to copy multiple lines, ensure there is only one Emacs window horizontally. 🍏 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 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	<ul style="list-style-type: none"><f11> C x⌘-x	(clipboard-kill-region BEG END & optional REGION)	Kill the region, and save it in the OS clipboard.
Copy Commands	<div>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.</div> <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. <div>The commands are listed in order of the size of text potentially copied:</div> <ul style="list-style-type: none">character, whitespaceword, symbolfilename/urllinefunction, list/sexpsentence, paragraph		
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).Shows the text copied in the echo area.
Copy whitespaces at point	<f11> = SPC	(pel-copy-whitespace-at-point)	Kill all whitespace characters at/ around point on current line. <ul style="list-style-type: none">Shows the text copied in the echo area.
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. 👉 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 modifications 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 bsuperword-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. Syntax depends on the syntax table for the buffer. <ul style="list-style-type: none">Shows the text copied in the echo area. 👉 The syntax of the symbol depends on the major mode used by the current buffer.
Copy filename at point	<f11> = F	(pel-copy-filename-at-point)	Copy filename at point. <ul style="list-style-type: none">Shows the text copied in the echo area.
Copy URL at point	<f11> = u	(pel-copy-url-at-point)	Copy URL at point. <ul style="list-style-type: none">Shows the text copied in the echo area.
Copy line beginning	<f11> = a	(pel-copy-line-start)	Copy text from the beginning of the current line up to point. <ul style="list-style-type: none">Shows the text copied in the echo area.
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> +<kp-separator>	<div>(pel-copy-marked-or-whole-line)</div> <div>The copy operation is controlled by the (optional) argument:<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.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.</div>	Flexible copy to kill ring.: copy visible region if any, otherwise copy current line to kill ring. 🍏 In terminal (TTY) mode the keypad + key is interpreted as <kp-separator> on macOS so this key is available. 🍏 Replaces standard binding to kill-ring-save which only copies region 👉 See the 🔗 Marking table to mark (select) a text region to use with this command.
Copy line end	<f11> = e	(pel-copy-line-end)	Copy text from point up to the end of the line. <ul style="list-style-type: none">Shows the text copied in the echo area.
Copy function at point	<f11> = f	(pel-copy-function-at-point)	Copy complete body of function at point. <ul style="list-style-type: none">Shows the text copied in the echo area.
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.Shows the text copied in the echo area.
Copy S-expression at point	<f11> = x	(pel-copy-sexp-at-point)	Copy and show complete Lisp S-expression at point. <ul style="list-style-type: none">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.Shows the text copied in the echo area.
Copy complete sentence at point	<f11> = s	(pel-copy-sentence-at-point)	Copy entire sentence at point. <ul style="list-style-type: none">Shows the text copied in the echo area. 👉 Toggle the minimum number of spaces that end a sentence with: pel-toggle-sentence-end : <f11> t m s

Operation	Keystroke	Function	Note
Copy paragraph beginning	<f11> = b	(pel-copy-paragraph-start)	beginning of paragraph to point. • Shows the text copied in the echo area.
Copy paragraph	<f11> = H	(pel-copy-paragraph-at-point)	Copy entire paragraph at point. • Shows the text copied in the echo area.
Copy paragraph end	<f11> = h	(pel-copy-paragraph-end)	Copy from point to end of paragraph. • Shows the text copied in the echo area.
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.
Deleting Text	Emacs supports “deleting” and “killing” text. Deleted text is not retained. Killed text is retained in the “kill ring”.		
Toggles delete selection mode See also: ↗ Marking ↗ 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.
Delete 1 Character	<p>Delete Keys:</p> <p>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.</p> <p>➡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 Fails to Delete.</p> <p>🕒 This table uses the ☒ and ☒ symbols to represent these 2 keys:</p> <ol style="list-style-type: none"> ☒ := “forward delete” := <deletechar> := Fn ☒ ☒ := “backward delete” := <backspace> Often labelled “delete” on keyboards. <p>⚠ C-☒ and C-☒ are not accessible in terminal mode.</p>		
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 deleted if delete-selection-mode is on, otherwise the region is not deleted.
character - forward	☒	(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	☒	(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.
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 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 between 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: ↗ Indentation ↗ Whitespace	M-^	(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.
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.
Remove non required whitespaces	<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.
Delete all spaces between point and next non-white on same line.	C-☒	(pel-delete-to-next-visible)	Delete all whitespace between point and next non-whitespace character (stops at end of line). 🍏 on macOS laptop, use: Fn C-delete
Hungry Deletion of Whitespace	<p>The CC mode provides two commands that can perform “hungry whitespace deletion” that can also be used in every mode.</p> <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.	• C-c DEL • C-c ☒ • C-c C-☒ • C-c C-☒ • C-c C-DEL • <f11> ☒	(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. This prevents conflicts with other modes that may use the popular C-c bindings.

Operation	Keystroke	Function	Note
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> <f11>  	(c-hungry-delete-forward)	<p>Delete the following character or all following whitespace up to the next non-whitespace character.</p> <p> 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.</p> <p> With PEL, the <f11>  binding is available in all modes. The other keys are only available in modes derived from the CC Mode. This prevents conflicts with other modes that may use the popular C-c bindings.</p>
Other Delete/Kill Commands	There's also a set of deletion commands to delete duplicate lines and delete or kill comments.		
Delete duplicate lines	<f11> - *	(delete-duplicate-lines BEG END &optional REVERSE ADJACENT KEEP-BLANKS INTERACTIVE)	<p>Delete all but one copy of any identical lines in the region (or entire buffer if nothing marked).</p> <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.
Delete all comments in buffer or marked region See also: ☞ Comments	<f11> ; 	(pel-delete-all-comments)	<p>Delete all comments in current (possibly narrowed) buffer or marked region.</p> <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)	<p>Kill all comments in current (possibly narrowed) buffer or marked region and retain them in kill ring.</p> <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>
Kill Commands	Emacs kill commands erase text and copy it into the kill ring.		
Kill/Delete marked region/line(s) ★PEL Enhanced Key ★ Available in PEL non numlock mode See also: <ul style="list-style-type: none"> ☞ Marking ☞ Num keypad 	<ul style="list-style-type: none"> C-w <f11> - 1 <kp-subtract> ⌘-x 	(pel-kill-or-delete-marked-or-whole-line &optional N)	<p>Flexible region/whole-line kill/delete. Argument controls behaviour (see next cell below).</p> <p> In graphics mode this also copies text to the OS clipboard.</p> <p> With PEL in non-numlock mode, the “-” key on the number keypad is bound to this command.</p> <p> 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.</p> <p> See the ☞ Marking table to mark (select) a text region to use with this command.</p>
	<ul style="list-style-type: none"> N=0 := kill region (active/visible or not) Sign of N selects operation: <ul style="list-style-type: none"> positive := kill (default) negative := delete Select text to delete/kill based on presence of region: <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> With no arg: <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> With no region active/visible: <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> With any negative mark argument: delete the region's text. With no argument or any positive argument: kill the region's text. <p> 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.</p>		
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.
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.
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 .
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.
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"> 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 at point	<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 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 ‘-’.
Kill beginning of line	<ul style="list-style-type: none"> M-0 C-k C-\ <f11> - a 	(pel-kill-from-beginning-of-line)	<p>Kills the beginning of the line up to the cursor.</p> <p>In terminal the M binding  does not work properly, and they do different things!</p> <ul style="list-style-type: none"> <M-<C-backspace> executing backward-kill-word. <M-S- <p>The binding works properly in graphics mode.</p> <p>This used to be mapped to (backward-kill-word ARG) to implement delete word backward. But remapped it.</p>

Operation	Keystroke	Function	Note
<u>Kill to end of line</u>	<ul style="list-style-type: none"> M-⌘ C-k <f11> - e 	(kill-line &optional ARG)	<p>Kills from current position to end of line. If no visible characters on it kill through newline.</p> <p>M-⌘ is bound to (insert-parentheses &optional ARG) as in M-(in terminal mode. The M-⌘ binding works properly in graphics mode.</p>
<u>Kill whole line</u>	C-S-⌘	(kill-whole-line &optional ARG)	Deletes current line (in graphics mode). Use C-w : it's more flexible.
<u>Kill sentence - backward</u>	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.
<u>Kill sentence - forward</u>	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.
<u>Kill sentence at point</u>	<f11> - s	(pel-kill-sentence-at-point)	Kill complete sentence at point.
<u>Kill forward to end of paragraph</u>	<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.
<u>Kill back to start of paragraph</u>	<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.
<u>Kill complete paragraph at point</u>	<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.
<u>Kill next Lisp S-expression</u>	<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.
<u>Kill previous Lisp S-expression</u>	<ul style="list-style-type: none"> C-M-⌘ <f11> - [C-[C-⌘ Esc C-⌘ 	(backward-kill-sexp &optional ARG)	<p>Kill the sexp (balanced expression) preceding point.</p> <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-⌘ binding 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>
<u>Kill Lisp S-Expression at point</u>	<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.
<u>Kill Lisp list at point</u>	<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.
<u>Kill function at point</u>	<f11> - f	(pel-kill-function-at-point)	Kill the function at point. Point can be anywhere, or just past the function code.
<u>Kill filename at point</u>	<f11> - F	(pel-kill-filename-at-point)	Kill the filename at point. Point can be located anywhere inside the file name or right after.
<u>Kill URL at point</u>	<f11> - u	(pel-kill-url-at-point)	Kill the URL at point. Point can be located anywhere inside the file name or right after.
<u>Kill text in rectangle</u> See also: ☞ Rectangles	<ul style="list-style-type: none"> C-x r k <f11> - r 	(kill-rectangle START END &optional FILL)	<p>Delete the region-rectangle and save it as the last killed one.</p> <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.)
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> ”.		
<u>Yank last killed into buffer</u> See also: ☞🖱 Numkeypad	<ul style="list-style-type: none"> C-y ⌘-v <kp-0> 	(yank &optional ARG)	<p>Reinsert ("paste") the last stretch of killed text.</p> <ul style="list-style-type: none"> 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. <p>🖱 ⌘-v In graphical mode: supports OS clipboard.</p> <p>🖱 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. See ☞🖱 Numkeypad</p>
<u>Paste from OS clipboard</u>	⌘-y	(ns-paste-secondary)	🍏 On macOS in graphics mode only: paste from OS clipboard (not from kill ring).
<u>Replace last yank with previous kill</u>	M-y	(yank-pop &optional ARG)	<p>Replace just-yanked stretch of killed text with a different stretch.</p> <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. <p>☞ Also referred to as: “yank next”.</p>
<u>Pop-up menu with kill ring content, to select entry to insert at point.</u>	<f11> M-y	(popup-kill-ring)	<p>Pop-up a menu that shows all entries in kill ring, allowing insertion of a specified kill ring entry at point.</p> <p>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 menu.</p> <p>🖱📦 Available only in graphics mode when popup-kill-ring package and its pre-requisites pos-tip and popup are installed.</p> <p>🖱🖱 PEL activates this when the pel-use-popup-kill-ring user option is set to t.</p>
<u>Append to Kill Ring</u>	<ul style="list-style-type: none"> C-M-w C-[C-w Esc C-w 	(append-next-kill &optional INTERACTIVE)	<p>Preparation command. Next kill command issued after this will add to the top of the kill ring item (the previous kill):</p> <ul style="list-style-type: none"> If the next command kills forward from point, the <u>kill is appended</u> to the previous killed text. If the command kills backward, the kill is prepended. <p>If the next command is not a kill command, this has no effect.</p>
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.		
<u>Display content of kill ring</u>	<f1> v kill-ring RET		Display the content of the kill ring in the “Help” buffer
<u>Display kill ring size</u>	<f1> v kill-ring-max RET		Display the maximum number of kill ring entries in the “Help” buffer.
<u>Set kill ring size</u>	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.
<u>Select text stored in kill ring</u>	<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.

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	