








Indenting & Tab

Description	Keystroke	Function	Note
<div>Indentation under Emacs</div> <ul style="list-style-type: none">Use hard tabs or spaces for indentationSet hard-tab visual widthAdjust tab stopShow tab/indent settingsAlign on returnInsert hard-tabTo next tab stoptabify & untabifyBehaviour of tab keyInsert newline, split lineIndent regionDelete indentationMove to 1st non-blankIndenting /un-indenting rigidlyText alignment on newlineIndent-toolsSmart-shift <div>Last updated on: 2025-05-16</div>			<div>Emacs controls indentation according to various rules controlled by the buffer major mode.</div> <ul style="list-style-type: none">Furthermore the behaviour of the tab key is also controlled by the major mode; it may have surprising behaviour for people learning Emacs.The standard behaviour may be modified by the use of major and minor modes.<ul style="list-style-type: none">Several major modes implement special indentation schemes, such as Lisp where indentation is inferred by the code itself as opposed to Python that uses indentation for defining scopes.Several major modes identify a variable that sets the indentation level. Refer to the information on the programming language major mode.Some programming languages (such as Go) impose hard-tab for indentation, using tab for indentation and space for alignment. Works very nicely.Most languages never identified any rule, which led in some case to all sorts of conventions: use of both tabs and spaces, spaces only, with various number of positions for the indentation level.Emacs can support anything. It can tabify or untabify source code. Impose the use of hard tab or prevent it.Emacs controls the <i>display rendering</i> of hard tabs by the tab-width variable.<ul style="list-style-type: none">The go-mode, for example, will move the first non-whitespace character location inside the buffer as you modify the tab-width as indentation is entirely controlled by hard tabs. It does not change the content of the file, just the way the file looks on the screen.The indentation width is often independent from the tab width but not always. Again it depends on the major mode used. <div>PEL supports various indentation mechanisms and also provides some of its own extensions. It also provides easy access to external packages that implement other behaviours, supporting various major modes. This includes the following:</div> <ul style="list-style-type: none"> The indent-tools external package  PEL activates it when the pel-use-indent-tools user-option is turned on (set to t). The smart-shift external package  PEL activates it when the pel-use-smart-shift user-option is turned on (set to t). <div>Information related to indentation is described in the pages related to programming major modes. The information in this page is generic and complements the mode specific information.</div>
Open this PDF file. See also: ↗ Help/Info	<f11> <tab> <f1>	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the ↗ Indentation 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 highlighting control	<f11> <tab> <f2>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL support for indentation management <ul style="list-style-type: none">If OTHER-WINDOW is non-nil (use C-u), display in other window.
↗ Customize Emacs indentation control	<f11> <tab> <f3>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs indentation control groups: indent, indent-tools , smart-shift . <ul style="list-style-type: none">If OTHER-WINDOW is non-nil (use C-u), display in another window.
Use hard tabs or spaces for indentation	The use of hard tabs or spaces for indentation is controlled by the Emacs (customizable) variable indent-tabs-mode . <ul style="list-style-type: none">Like several Emacs variable this variable has global impact, but this can be overridden by directory local value, file local value and buffer local value <div>See also: ↗ Whitespace</div>		
Toggle use of hard tabs for indentation in the current buffer	<ul style="list-style-type: none"><f11> <tab> m<f11> t w I	(pel-toggle-indent-tabs-mode &optional ARG)	Toggle whether indentation can insert hard tab characters in the current buffer. <ul style="list-style-type: none">Beep on each change to warn user of the change and display new value.If ARG is positive set to use hard tabs, otherwise force use of spaces only.This uses Emacs indent-tabs-mode function and provides more feedback.
Hard-tab “width”	The current-buffer value of tab-width affects the <i>visual rendering</i> of the hard-tab character. It does not affect the content of the buffer or file.		
Set visual rendering of hard tabs for the current buffer <ul style="list-style-type: none">Does not change buffer/file content	<f11> M-t	(pel-set-tab-width N)	Change the tab-width of the current buffer, only affecting the display rendering of hard tabs inserted in the buffer text. Prompts for a new value in the [2, 8] range. <ul style="list-style-type: none">This modifies a buffer local value of the the tab-width user-option. The change is temporary and affects the current buffer only.PEL provides a specialized user-option to set the default value of tab-width for several major modes. For example, to change the tab width used for all Go source code files, change the 'pel-go-tab-width' user-option variable instead. See the documentation of each major mode for more information.
Tab stops	Emacs keeps track of a set of “ <i>tab-stop</i> ” columns that can be used as reference points to align text. Something similar to typewriter tab-stop .		
Change the tab stops <ul style="list-style-type: none">used by M-i	<f11> <tab> e	(edit-tab-stops)	Opens a *Tab Stops* buffer . Identify the tab stops in the first line with colons. Use C-c C-c to activate and exit the buffer. <ul style="list-style-type: none">The tab stop take effect at the top of the buffer, as used by M-i
Show Indentation settings <div>For several major modes, a PEL mode-specific user-option controls the the value of the tab-width variable for the mode. For example, pel-c-tab-width is used for c-mode buffers.</div> <div>Note however, that indentation for C buffer is controlled by the c-basic-offset user-option.</div>	<ul style="list-style-type: none"><f11> <tab> ?<f11> ? <tab> <div>The command opens a buffer showing the indentation control for the current buffer.</div> <ul style="list-style-type: none">The first line shows the buffer name and date.The first group of user options are buffer specific and take precedence over the global user-options.The next group are global user-options that are used when they are not overridden by major-mode specific ones.<ul style="list-style-type: none">inside c-mode buffers, PEL has user-variables that take over.All user-option variables are help buttons; click on them (in graphics mode) or tab and return to open help on the	(pel-show-indent &optional APPEND)	<div>Print info about indentation control in a *pel-indent-info* help-mode buffer.</div> <ul style="list-style-type: none">Buffer-specific values of relevant user-options as buttons to use to get more info and change their customized values. Includes major-mode specific ones.Clear previous buffer content. Use prefix arg (like C-u) to append instead. <div>----Indentation Control from alloc.c --- Wednesday, April 30, 2025 @ 08:21:59 ---- - pel-c-indent-width : 4 - pel-c-tab-width : 4 - pel-c-use-tabs : nil ---- The above major-mode specific user options take precedence over the following global ones (unless they are set by file variables): - c-basic-offset : 4 - tab-width : 4 -> Use pel-set-tab-width to change locally and have tabs rendered with a different width. - indent-tabs-mode : nil - standard-indent : 4 - tab-always-indent : t - tab-stop-list : nil -UUU:%*- F1 *pel-indent-info* All (1,0) (Help WK Anzu) 08:22 2.34 ----- f11 TAB ?</div> <div>These are help buttons! Click on them in graphic mode or text with mouse enabled. In text-mode without mouse support, <tab> to navigate and <ret> to open help.</div>
Toggle text alignment on pel-newline-and-indent-below See also: ↗ Align	<f11> M-RET	(pel-toggle-newline-indent-align)	Toggle variable <i>pel-newline-does-align</i> for the local buffer. <div>It affects the way function 'pel-newline-and-indent-below' operates.</div> <ul style="list-style-type: none">If <i>pel-newline-does-align</i> is t, it aligns several syntactic element in the current block: the comments, the assignments. set modes that automatically activates <i>pel-newline-does-align</i> by adding the major mode to pel-modes-activating-align-on-return user option.This affects the behaviour of the following commands: pel-cc-newline (assigned to RET in CC modes like c-mode, c++-mode and d-mode). pel-newline-and-indent-below (assigned the M-RET) <ul style="list-style-type: none">See the list with <f11> t a ?
Show state of text modes <ul style="list-style-type: none">whether hard tabs are used for indentation, tab-widthwhether newline aligns textelectric-quote-modedelete-selection-modeenriched-modeoverwrite-modecase foldingsubword, superword, glass modesvisible-mode, smart-dash-modeparagraph definition <div>Output example ➡</div>	<f11> t m ?	(pel-show-text-modes)	Display the state of the various text modes in the mini buffer. <div> Quickly show several settings inside the mode line: the tab settings, text alignment on newline, whitespace mode, etc...</div> <ul style="list-style-type: none">When indent-tabs-mode is active, Emacs inserts a number of hard tabs and spaces.<ul style="list-style-type: none">The number of hard tabs instead depends on the amount of characters required for indentation and the tab-width.If indent-tabs-mode is not active, then Emacs inserts only space characters. PEL provides user-options of the form pel-<mode>-use-tabs which is used to initialize the indent-tabs-mode supported major modes buffers. <div>Text Modes Status: - Local indent-tabs-mode : off: use spaces, Tab width = 8 - Local newline does align : off . Automatically activated by modes (<f11> t a <f2>): (c-mode c++-mode sh-mode) - Local electric-quote-mode : off , electric-quote-local-mode: not loaded. - whitespace-mode : not loaded, show-trailing-whitespace : off , indicate-empty-lines: off. - enriched-mode : not loaded. - overwrite mode : off , delete-selection-mode : off. - case-fold-search : on , sort-fold-case : not loaded. - subword mode : off , superword mode : on , glass-mode: not loaded. - visible-mode : off , smart-dash-mode : not loaded. - Sentences end with 2 space characters. - paragraph-start : "^L\\ []*\$" - paragraph-separate: "[^L]*\$"</div>

Description	Keystroke	Function	Note
Indenting and un-indenting rigidly	The following commands provide non-semantic indentation of the current line or marked region. <ul style="list-style-type: none">The first command allows you to use further keystrokes to fine-tune the indentation back and forth using cursor keys. That's probably all you ever need to use.Currently, PEL also provides the last 2 commands that indent or un-indent the current line or marked region. Once used, the region remains marked to allow further use of the command.		
Indent/Unindent rigidly See also: 🔗 Key-Chords See also: <ul style="list-style-type: none">🔗 I - C🔗 I - C++🔗 I - D🔗 reStructuredText	<ul style="list-style-type: none">C-x <tab><f11> <tab> <tab><tab>g <div><div></div><div>✂ PEL uses the above instead of the standard: (indent-rigidly START END ARG &optional INTERACTIVE)</div></div>	(pel-indent-rigidly &optional N) <	

