Indenting & Tab

<u>Description</u>	<u>Keystroke</u>	Function	Note
Indentation under Emacs Use hard tabs or spaces for indentation Set hard-tab visual width Adjust tab stop Show tab/indent settings Align on return Insert hard-tab To next tab stop tabify & untabify Behaviour of tab key Insert newline, split line Indent region Delete indentation Move to 1st non-blank Indenting /un-indenting rigidly Text alignment on newline Indent-tools Smart-shift	Furthermore the behaviour of the The standard behaviour may be Several major modes impler that uses indentation for detections of the Several major modes identified Some programming language. Most languages never identing number of positions for the Emacs can support anything. Emacs controls the display of the go-mode, for example entirely controlled by hard. The indentation width is often PEL supports various indentation implement other behaviours, supports the indentation implement other behaviours. The indent-tools external the smart-shift external the smart-tabs external the smart-tabs.	ment special indentation schemes, sument special indentation schemes, suffining scopes. fining scopes (such as Go) impose hard-tab for iffied any rule, which led in some case indentation level. g. It can tabify or untabify source concendering of hard tabs by the tab-wide, will move the first non-whitespace d tabs. It does not change the content in independent from the tab width but mechanisms and also provides some porting various major modes. This in I package PEL activates it when the package PEL activates it when the is described in the pages related to page scopes.	ajor mode; it may have surprising behaviour for people learning Emacs. inor modes. ch as Lisp where indentation is inferred by the code itself as opposed to Python level. Refer to the information on the programming language major mode. indentation, using tab for indentation and space for alignment. Works very nicely. to all sorts of conventions: use of both tabs and spaces, spaces only, with various le. Impose the use of hard tab or prevent it. Ith variable. character location inside the buffer as you modify the tab-width as indentation is t of the file, just the way the file looks on the screen. t not always. Again it depends on the major mode used. e of its own extensions. It also provides easy access to external packages that
Last updated on:	2025-10-29	See Showing Information Abou	tt Indentation and Hard Tab Control under PEL
Open this PDF file. See also: <u>∑ Help/Info</u>	<f11> <tab> <f1></f1></tab></f11>	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the <u>S</u> 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></f2></tab></f11>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL support for indentation management • If OTHER-WINDOW is non-nil (use C-u), display in other window.
∑ Customize Emacs indentation control	<f11> <tab> <f3></f3></tab></f11>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs indentation control groups: indent, <u>indent-tools</u> , <u>smart-shift</u> . • If OTHER-WINDOW is non-nil (use C-u), display in another window.
Use hard tabs or spaces for indentation		is variable has global impact, but this	acs (customizable) variable indent-tabs-mode . can be overridden by <u>directory local</u> value, <u>file local</u> value and <u>buffer local</u> value See also: <u>** Whitespace**</u>
Toggle use of hard tabs for indentation in the current buffer	• <f11> <tab> m • <f11> t w I</f11></tab></f11>	(pel-toggle-indent-tabs-mode &optional ARG)	Toggle whether indentation can insert hard tab characters in the current buffer. • 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-w	idth 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	<f11> <tab> w</tab></f11>	(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.
Does not change buffer/file content	 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 a 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 "ta	b-stop" columns that can be used as	reference points to align text. Something similar to typewriter tab-stop.
Change the tab stops • used by M-i	<f11> <tab> e</tab></f11>	(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. • The tab stop take effect at the top of the buffer, as used by M-i
Show Indentation settings For several major modes, a	• <f11> <tab> ? • <f11> ? <tab> The command opens a buffer</tab></f11></tab></f11>	(pel-show-indent &optional APPEND)	Print info about indentation control in a *pel-indent-info* help-mode buffer. 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.
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. Note however, that indentation for C buffer is controlled by the c-basic- offset user-option. Other major modes may	showing the indentation control for the current buffer. 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.	- pel-c-indent-width - pel-c-tab-width - pel-c-use-tabs The above major-mode specific us over the following global ones file variables): - c-basic-offset - tab-width -> Use pel-set-tab-width to column	Loc.c Wednesday, April 30, 2025 @ 08:21:59 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.</ret>
provide mode specific control variables and the printed information will differ.	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	<pre>- indent-tabs-mode - standard-indent - tab-always-indent - tab-stop-list -UUU:%*- F1 *pel-indent-info* f11 TAB ?</pre>	: nil : 4 : t : nil All (1,0) (Help WK Anzu) 08:22 2.34
provide mode specific control variables and the printed information will differ. Toggle text alignment on	has user-variables that take over. • All user-option variables are help buttons; click on them	- standard-indent - tab-always-indent - tab-stop-list -UUU:%*- F1 *pel-indent-info*	: 4 : t : nil
provide mode specific control variables and the printed information will differ.	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 <f11> M-RET It affects the way function 'pel-ne • If pel-newline-does-align is t, i • \$\mathbb{c}\$ set modes that automatical</f11>	- standard-indent - tab-always-indent - tab-stop-list -UUU:**- F1 *pel-indent-info* f11 TAB ? (pel-toggle-newline-indent-align) ewline-and-indent-below' operates. it aligns several syntactic element in telly activates pel-newline-does-align be following commands: pel-cc-new pel-newline	: 4 : t : nil All (1,0) (Help WK Anzu) 08:22 2.34
provide mode specific control variables and the printed information will differ. Toggle text alignment on pel-newline-and-indent-below	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 <f11> M-RET It affects the way function 'pel-ne • If pel-newline-does-align is t, i • \$\mathbb{z}\$ set modes that automatical • This affects the behaviour of the</f11>	- standard-indent - tab-always-indent - tab-stop-list -UUU:**- F1 *pel-indent-info* f11 TAB ? (pel-toggle-newline-indent-align) ewline-and-indent-below' operates. it aligns several syntactic element in telly activates pel-newline-does-align be following commands: pel-cc-new pel-newline	: 4 : t : nil All (1,0) (Help WK Anzu) 08:22 2.34 Toggle variable pel-newline-does-align for the local buffer. The current block: the comments, the assignments. By adding the major mode to pel-modes-activating-align-on-return user option. Iline (assigned to RET in CC modes like c-mode, c++-mode and d-mode).
provide mode specific control variables and the printed information will differ. Toggle text alignment on pel-newline-and-indent-below See also: Align	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 • f11> M-RET It affects the way function 'pel-ne • If pel-newline-does-align is t, i • \$\mathbb{z}\$ set modes that automatical • This affects the behaviour of th • See the list with <f11> t a <f11> t m ? • Quickly show several settings • When indent-tabs-mode is ac • The number of hard tabs ins • If indent-tabs-mode is not active</f11></f11>	- standard-indent - tab-always-indent - tab-stop-list -UUU:%*- F1 *pel-indent-info* f11 TAB ? (pel-toggle-newline-indent-align) ewline-and-indent-below' operates. it aligns several syntactic element in tilly activates pel-newline-does-align to the following commands: pel-cc-new pel-newline? (pel-show-text-modes) inside the mode line: the tab settings titve, Emacs inserts a number of hard tead depends on the amount of charve, then Emacs inserts only space characteristics.	: 4 : t : nil All (1,0) (Help WK Anzu) 08:22 2.34 Toggle variable pel-newline-does-align for the local buffer. The current block: the comments, the assignments. By adding the major mode to pel-modes-activating-align-on-return user option. Itine (assigned to RET in CC modes like c-mode, c++-mode and d-mode). By adding the major mode to pel-modes-activating-align-on-return user option. Itine (assigned to RET in CC modes like c-mode, c++-mode and d-mode). By adding the major mode to pel-modes-activating-align-on-return user option. It can be a sign of the mode and d-mode and
provide mode specific control variables and the printed information will differ. Toggle text alignment on pel-newline-and-indent-below See also: SAlign Show state of text modes • whether hard tabs are used for indentation, tabwidth • whether newline aligns text • electric-quote-mode • delete-selection-mode • enriched-mode • overwrite-mode • case folding • subword, superword, glass modes	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 • f11> M-RET It affects the way function 'pel-ne • If pel-newline-does-align is t, i • \$\frac{1}{2}\$ set modes that automatical • This affects the behaviour of th • See the list with <f11> t a • f11> t m ? • Quickly show several settings • When indent-tabs-mode is ac • The number of hard tabs ins • If indent-tabs-mode is not active • \$\frac{1}{2}\$ PEL provides user-options Text Modes Status: - Local indent-tabs-mode - Local newline does alight of the period of the peri</f11>	- standard-indent - tab-always-indent - tab-always-indent - tab-stop-list -UUU:%*- F1 *pel-indent-info* f11 TAB ? (pel-toggle-newline-indent-align) ewline-and-indent-below' operates. it aligns several syntactic element in t lly activates pel-newline-does-align to the following commands: pel-cc-new pel-newline ? (pel-show-text-modes) inside the mode line: the tab settings stive, Emacs inserts a number of hard stead depends on the amount of characteristic element in t to the form pel- <mode>-use-tabs will the control of the form pel-<mode>-use-tabs will the control of the control of</mode></mode>	: 4 : t : nil All (1,0) (Help WK Anzu) 08:22 2.34
provide mode specific control variables and the printed information will differ. Toggle text alignment on pel-newline-and-indent-below See also: Salign Show state of text modes whether hard tabs are used for indentation, tab-width whether newline aligns text electric-quote-mode delete-selection-mode enriched-mode overwrite-mode case folding subword, superword,	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 • f11> M-RET It affects the way function 'pel-ne • If pel-newline-does-align is t, i • \$\frac{1}{2}\$ set modes that automatical • This affects the behaviour of th • See the list with <f11> t a • f11> t m ? • Quickly show several settings • When indent-tabs-mode is ac • The number of hard tabs ins • If indent-tabs-mode is not active • \$\frac{1}{2}\$ PEL provides user-options • Text Modes Status: • Local indent-tabs-mode • Local newline does alig • Local electric-quote-mode • whitespace-mode</f11>	- standard-indent - tab-always-indent - tab-always-indent - tab-stop-list	: 4 : t : nil All (1,0) (Help WK Anzu) 08:22 2.34

<u>Description</u>	<u>Keystroke</u>	Function	<u>Note</u>	
Align on return	The following commands control	indentation on return. The behaviour	can be modified and PEL provides a quick command to list relevant user options.	
Display current buffer's vertical alignment behaviour See also: Align	<f11> t a ?</f11>	(pel-align-info &optional APPEND)	Print information about text aligning info in a *pel-align-info* help-mode buffer. • Prints current state and values of relevant user-options as buttons to help: • M-RET behaviour in current buffer: show if that command aligns text or not. • whether the modes are activating vertical alignment on return. • Clear previous buffer content. Use prefix arg (like C-u) to append instead.	
Insert an indented line below current line See also: Align	• M-RET • <f11> <tab> RET</tab></f11>	(pel-newline-and-indent-below)	Insert an indented line just below current line. • Also align vertically if mode was activated for the buffer with <f11> M-RET. • See current behaviour with <f11> t a ?</f11></f11>	
Inserting hard tab	Regardless of <tab> key behavior</tab>	our, it is possible to insert a <u>hard tab o</u>	character with C-q <tab></tab>	
Insert Literal Tab	C-q <tab></tab>	(quoted-insert ARG) <tab></tab>	Inserts a <u>hard tab</u> inside buffer. Render text according to value of tab-width .	
Insert whitespace to next tab-stop		be modified with (edit-tab-stop), boun	combinations of hard tab and spaces) to put point to the next "tab-stop" with M-i nd to <f11> <tab> e .</tab></f11>	
Insert spaces or tabs to next defined tab-stop column	M-i	(tab-to-tab-stop)	Insert spaces or tabs to next defined tab-stop column. • Use <f11> <tab> e to modify position of the tab stops.</tab></f11>	
Tabify & untabify	• The exact location of the next tab stop is identified by the value of the tab-stop-list and tab-width for the current buffer. The following two commands can be used to replace hard tabs in a file with the corresponding number of space characters while retaining the same			
labily α untabily	indentation and vice-versa.			
Replace tabs with spaces in a region See also: Whitespace	<f11> t w SPC</f11>	(untabify START END &optional ARG)	Convert all tabs in region to multiple spaces, preserving columns. If called interactively with prefix ARG, convert for the entire buffer. First select a region (Use C-x h for selecting the whole file). Then use the untabify function to replace all tabs by spaces in that region.	
Replace multiple spaces with tabs in a region	<f11> t w <tab></tab></f11>	(tabify START END &optional ARG)	Convert multiple spaces in region to tabs when possible. • A group of spaces is partially replaced by tabs when this can be done without	
See also: <u>∑ Whitespace</u>			changing the column they end at. If called interactively with prefix ARG, convert for the entire buffer.	
Behaviour of Tab Key	In Emacs the behaviour of the <tab></tab> key depends on the major mode of the current buffer. This key is rebound by several major mode. • In text modes, tabs default to inserting hard tabs corresponding to a alignment at every 8 columns. However, if there is text in the above lines, the tab moves to the spot under the word above. Note that if a line is full of text (without any space), then the tab stops controlled by the ruler take effect again. • In several programming modes the tab key does not insert anything, it adjusts the line indentation, according to surrounding code			
Indent current line (or region)	<tab></tab>	(indent-for-tab-command &optional ARG)	Indent the current line or region, or insert a tab, as appropriate.	
	 This function either inserts a tab, or indents the current line, or performs symbol completion, depending on 'tab-always-indent'. The function called to actually indent the line or insert a tab is given by the variable 'indent-line-function'. If a prefix argument is given, after this function indents the current line or inserts a tab, it also rigidly indents the entire balanced expression which starts at the beginning of the current line, to reflect the current line's indentation. In most major modes, if point was in the current line's indentation, it is moved to the first non-whitespace character after indenting; otherwise it stays at the same position relative to the text. If 'transient-mark-mode' is turned on and the region is active, this function instead calls 'indent-region'. In this case, any prefix argument is ignored. 			
	The behaviour of the tab key vastly differ between major modes. This ranges from not moving the cursor at all if the indentation is identified as correct for the current context, to cycling through various potential positions to just what someone new to Emacs would expect. Much more has to be documented on the behaviour of that key and how it can be controlled and customized. It's quite possible that the best way to document its behaviour would be to place a description inside the table of each major mode.			
	<tab></tab>	indent-for-tab-command &optional ARG)	In Lisp related modes. indent-line-function = indent-relative. • tab-always-indent = t	
	Several major modes adjust the	ne behaviour of the tab key to perform	semantically aware indentation, such as what is being done in Lisp.	
	<tab></tab>	(c-indent-line-or-region &optional ARG REGION)	In C related modes: Indent active region, current line, or block starting on the line. In Transient Mark mode, when the region is active, reindent the region. With prefix argument, rigidly reindent the expression starting on current line. Otherwise reindent just the current line.	
Indent lines of list after point Example: CLBC s3.lisp	С-М-ф	(indent-sexp & optional ENDPOS) (c-indent-exp & optional SHUTUP-P)	Indent each line of the list starting just after point. • The command used depends on the major mode of the current buffer.	
Newline & indent				
Insert new-line and maybe indent	C-j	(electric-newline-and-maybe-indent)	Add new line and indent next line. Indentation is controlled by the variable left-margin. Pressing Tab anywhere on the line also indents the line properly.	
Insert new-line and maybe indent	RET	(pel-cc-newline &optional N)	Insert a newline and perhaps align. With argument N repeat N times. • For newline insertion, operate according to the value of the variable 'pel-cc-newline-mode'.	
Split current line & indent	С-М-о	(split-line &optional ARG)	Split current line, moving portion beyond point vertically down. If the current line starts with 'fill-prefix', insert it on the new line as well. With prefix ARG, don't insert 'fill-prefix' on new line.	
Indent Region	C-M-\	(indent-region START END &optional COLUMN)	Indent each nonblank line in the region. • A numeric prefix argument specifies a column: indent each line to that column. • With no prefix argument, the command chooses one of these methods and indents all the lines with it: 1. If 'fill-prefix' is non-nil, insert 'fill-prefix' at the beginning of each line in the region that does not already begin with it. 2. If 'indent-region-function' is non-nil, call that function to indent the region. 3. Indent each line via 'indent-according-to-mode'.	
Indent relative to line above	<f11> <tab> r (indent-relative & optional FIRST-ONLY UNINDENTED-OK) Space out to under next indent point in previous nonblank line. An indent point is a non-whitespace character following whitespace. • If FIRST-ONLY is non-nil (ie. using C-u prefix) then only the first indent point is considered. • If the previous nonblank line has no indent points beyond the column point starts at, then 'tab-to-tab-stop' is done, if both FIRST-ONLY and UNINDENTED-OK are nil, otherwise nothing is done. • If there isn't a previous nonblank line and UNINDENTED-OK is nil, call 'tab-to-tab-stop'. Essentially, this command inserts whitespace at point, until point is aligned with the first non-whitespace character on the previous line (actually, the last non-blank line). If point is already farther right than that, run tab-to-tab-stop instead—unless called with a numeric argument, in which case do nothing.</tab></f11>			
Delete Indentation, join line to the previous one See also: ∑ Cut & Paste ∑ Whitespace	• M-^ • <f11> © 6 • <f6> 6</f6></f11>	(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.	
Move to fist nonbank character on the line	М-т	(back-to-indentation)	Move point to the first non-whitespace character on this line.	

Description	<u>Keystroke</u>	Function		<u>Note</u>	
Indenting and un- indenting rigidly	The first command allows you to use.	e non-semantic indentation of the curro to use further keystrokes to fine-tune ne last 2 commands that indent or un-	the indentation back an	d forth using cursor keys. T	
Indent/Unindent rigidly See also: Key-Chords	• C-x <tab> • <f11> <tab> <tab> • <tab>q</tab></tab></tab></f11></tab>	(pel-indent-rigidly &optional N)	If a region is mark to control indentation If no region is mark argument N (or if no N = [-1, 0, 1] : 0 N > 1 : operate N < -1 : operate	on changes. rked, it operates on current of specified N=1): operate on current line on the current line and N-1 on the current line and (abs is issued use the keys listed)	nd provides the same prompts line(s) identified by the numeric lines below.
FEL rebinds this key, but extends the functionality: pel-indent-rigidly uses indent-rigidly, described below the dashed line.		➤ PEL uses the above instead of the standard: (indent-rigidly START END ARG &optional INTERACTIVE)	which the indentati	g in the region. ly with no prefix argument, a on can be adjusted interacti ft>, or S- <right>.</right>	
See also: • PL - C • PL - C++ • PL - D • M reStructuredText	Both of these commands activate a transient mode where Emacs prompts for extra keys to control how to indent or un-indent. The capabilities are controlled by the variable indent-rigidly-map with by default provides: • S- <right> indent-rigidly-right-to-tab-stop</right>			width user option. ble, for buffers in c-mode, the	
		y-chord is available when pel-use-key is available with the key-chord bindin		∑ Key-Chords.	
Indent line(s) rigidly	<f6> <tab> <f11> <tab> c Works with point anywhere on All lines touched by the region</tab></f11></tab></f6>		Indent current or mark	ked lines by N indentation le	vels
	A special argument N can specific If a negative number is specific If a region is marked, the funct characters in all affected lines. Use C-g to de-activate the regional Handles presence of hard tabs If indent-tabs-mode is non-tabs-mode is non-tabs-mode.	cify more than one indentation level. Ited, 'pel-unindent-lines' is used. ion does not deactivate it to allow repigion. it is: inil the indentation is created with a miny hard tab in the indentation of the mindentation of th	eated execution of the o	ce characters.	·
Un-indent line(s) rigidly	If a region was marked, the fur characters in all affected lines Use C-g to de-activate the reg Handles presence of hard tabs If indent-tabs-mode is non-indepth.	are un-indented. tion does not deactivate it to allow repond to a low repond	peated execution of the epeated execution of the x of hard-tabs and space	e command. It also modifie ce characters.	s the region to include all
Indent-tools	The indent-tools external package provides several commands to indent, un-indent and navigate across indented text levels. It provides a minor mode and a key hydra that provides all of these commands. The indent-tools external package PEL activates it when the pel-use-indent-tools user-option is turned on (set to t). This also automatically activates the hydra external package. PEL provide a global key binding to its key hydra and provides the ability to activate the proposed key binding globally and for python mode: pel-indent-tools-key-bound: activates the C-c > key binding either globally or for python-mode only.		o t).		
Open the indent-tools	• <f11> <tab> <f7></f7></tab></f11>	(indent-tools-hydra/body)	Activate the "indent-to	ools-hydra" hydra.	
hydra See also: <u>ಭැ - Python</u>	• <f7> <tab> • C-c ></tab></f7>		 globally, when pel- in python-mode on	•	set to globally ,
	The heads for the associated hyders: 'indent-tools-indent'		Indent	Navigation	Actions
	 'indent-tools-indent-tools-indent-tools-indent-tools-indent-tools-comme 'indent-tools-comme 'indent-tools-uncomme 'indent-tools-indent-tools-indent-tools-indent-tools-indent-tools-kill-tree 'indent-tools-copy-h 'indent-tools-select', 	s', end-of-defun', ent', ment', paragraph', end-of-level', ', ydra/body',	> indent < de-indent l end of level end of fn p paragraph SPC space undo	j v k A n next sibling p previous sibling u up parent d down child e end of tree	<pre>K kill i imenu C Copy c comment U uncomment (paragraph) f fold q quit</pre>
	e: 'indent-tools-goto-er u: 'indent-tools-goto-pr d: 'indent-tools-goto-cl S: 'indent-tools-select-r n: 'indent-tools-goto-nr p: 'indent-tools-goto-pr i: 'helm-imenu', j: 'forward-line', k: 'previous-line', SPC: 'indent-tools-indent-	arent', nild', end-of-tree', ext-sibling', revious-sibling',			
See also: <u>National Hide/Show</u>	_: 'undo-tree-undo', L: 'recenter-top-bottom f: 'yafolding-toggle-ele q: exit	, ,	√ The f key toggles t again to display it bac	_	nce to hide the sub-tree, press-

Description	<u>Keystroke</u>	Function	<u>Note</u>
Smart-shift	The smart-shift external package simplifies shifting a complete line or region of lines right or left but also up or down. It is implemented as a minor or global minor mode that must be enabled first.		
© Customize with: <f11> <tab> s <f2></f2></tab></f11>	 Automatically activate the smart-shift-mode in specified major mode by customizing the pel-<mode>-activates-minor-modes user-options.</mode> You can also use the commands manually or through the key bindings provided by PEL to activate the smart-shift-mode in the current buffer or globally for all buffers. PEL controls it through customization user-options: The smart-shift external package PEL activates it when the pel-use-smart-shift user-option is turned on (set to t). PEL also provides the pel-smart-shift-keybinding user-option that allows you to select whether the shift keys used by smart-shift mode is the default provided keys only or whether you also want to activate another set. The default are always available when smart-shift mode is active: C-c <right>, C-c <left>, C-c <up> and C-c <down>.</down></up></left></right> PEL can also activate one of the following extra key binding sets: Using the control cursor key: C-c C-<right>, C-c C-<left>, C-c C-<up> and C-c C-<down>.</down></up></left></right> Using the <f9> key as prefix: <f9> <right> , <f9> <left> , <f9> <up> and <f9> <down></down></f9></up></f9></left></f9></right></f9></f9> 		
Toggle smart-shift mode in current buffer	<f11> <tab> s</tab></f11>	(smart-shift-mode &optional ARG)	Activate/de-activate the smart-shift mode in the current buffer. Activate the line-shift key bindings listed below, in the current buffer. With PEL, the actual key binding selected for the line shift commands depend on the value of the pel-smart-shift-keybinding user-option.
Toggle smart-shift mode globally	<f11> <tab> S</tab></f11>	(global-smart-shift-mode &optional ARG)	Toggle Smart-Shift mode in all buffers. With prefix ARG, enable Global Smart-Shift mode if ARG is positive; otherwise, disable it. Smart-Shift mode is enabled in all buffers where 'smart-shift-mode-on' would do it.
When smart-shift mode is active:	As described above, with PEL only one of the extra key bindings provided by PEL can be enabled via the pel-smart-shift-keybinding user-option. So unlike other key binding description cells in this and other tables, only one of the last 2 key bindings is available in the smart-shift minor mode.		
Shift line or region right	• C-c <right> • C-c C-<right> • <f9> <right></right></f9></right></right>	(smart-shift-right &optional ARG)	Shift the line or region to the ARG times to the right.
Shift line or region left	• C-c <left> • C-c C-<left> • <f9> <left></left></f9></left></left>	(smart-shift-left &optional ARG)	Shift the line or region to the ARG times to the left.
Shift line or region up	• C-c <up> • C-c C-<up> • <f9> <up></up></f9></up></up>	(smart-shift-up &optional ARG)	Shift the line or region to the ARG times to the upwards.
Shift line or region down	• C-c <down> • C-c C-<down> • <f9> <down></down></f9></down></down>	(smart-shift-down &optional ARG)	Shift the line or region to the ARG times to the downwards

Indentation - References

Title & URL	Description
Understanding GNU Emacs and Tabs	Overview description of how Emacs handle the Tab key, often used for strict indentation in many editors. In Emacs it can do much more.
GNU Emacs Manual - Indentation	
GNU Emacs Manual - Indentation for Programs	
Indentation Basic Concepts Tutorial @ XEmacs	A tutorial on indentation written by KaiGrossjohann

Tabs or space for indentation??

There are several views on the use of hard-tab and space characters for indenting source code. They are:

- Use only hard-tab for indentation. Uncontrolled use of tabs or spaces for alignment.
 Use only space characters for indentation. Popular in C like languages. Also popular in Python.
- 3. Use hard tabs for indentation, and space character for alignment.
- Method 1 was popular originally since it reduces file size when hard tab size was always the same. But soon it became possible to identify a different number of character positions to render a hard tab. And then it became impossible to guarantee the rendering of code indentation and alignment when the number of hard-tabs did not match the indentation level of a line of source code.
- A reaction to this problem is to use Method 2 where hard-tabs are banned. The rendering is therefore always the same no matter what the size of a hard tab is since you don't use
- any. This however increases the size of files. Not a problem for storage today you'd say, but perhaps a problem for data transfer and/or power consumption.

 Method 3 is used by some programming environments. The Go programming language imposes the use of hard-tabs for indentation. And if you want to align text at the right of the
 - indentation level, you use spaces.

 To use this method in other programming languages, you can use the smart-tabs-mode explained in the **Smart-Tabs Emacs Wiki page**.

Emacs support all modes. It has 2 different buffer local variables that are important and control the rendering of hard-tabs and the indentation:

- tab-width: How many columns a hard-tab occupies, the distance between tab-stops.
 indentation offset variable: a variable for each major mode, like **c-basic-offset** for CC modes (C, C++, Java, etc...), that identifies the number of columns per indentation level.

PEL provides access to the smarttabs package but it's not yet fully configured nor tested. 🚧 For CC modes it provides PEL user-options that control the indentation using method 2. Using method 3 requires a better understanding from all developers working on the source code with all their editors being able to handle the mix of hard tab and space characters

Smarttabs @ GitHub	Starttabs source code repository.
Indentation Styles for Curly Bracket Languages	
Indentation Styles @ Wikipedia	
StackOverflow - Emacs BSD/Allman Style with 4 Space Tabs?	
GNU Emacs Manual - Styles	
Emacs BSD/Allman Style with 4 Space Tabs?	
Emacs: Linux Kernel Style but with Allman/BSD Style Braces?	
Emacs Wiki - Indenting C	
Indent preprocessor directives as C code in emacs	Does not fully address the way I want to have multi-indentations for pre-processor
elisp code - ppindent.el	Implements pre-processor indentation with the # always in the first column. Not yet exactly what I want.
Demystify C++ Metaprograms using Emacs	
Programming in C++, Rules and Recommendations	ellemtel style