Indenting & Tab

See alone Telephorho Contaminate PEL Ministry Contaminate PEL See alone Contamina	indenting & lab			
** Uniformition to the behance of th	<u>Description</u>	<u>Keystroke</u>	Function	<u>Note</u>
Productions **Productions*	Indentation with PEL Behaviour of tab key Text alignment Indent region Indenting /un-indenting rigidly Controlling use of tabs and spaces for indentation Replacing tab with	Furthermore the behaviour of the translation of the translation for defended by the translation of the display of the display of the translation o	he tab key is also controlled by the may a modified by the use of major and moment special indentation schemes, surfaining scopes. By a variable that sets the indentation larges (such as Go) impose hard-tab for iffed any rule, which led in some case indentation level. G. It can tabify or untabify source coderendering of hard tabs by the tab-widdle, will move the first non-whitespace of tabs. It does not change the content an independent from the tab width but an independent from the tab width but	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 evel. 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 e. Impose the use of hard tab or prevent it. th variable. character location inside the buffer as you modify the tab-width as indentation is of the file, just the way the file looks on the screen. Inot always. Again it depends on the major mode used.
Sea above Carterina PEL Contamina PEL		implement other behaviours, supplement of the indent-tools external The smart-shift external Information related to indentation	porting various major modes. This inc package PEL activates it when the package PEL activates it when the is described in the pages related to p	cludes the following: the pel-use-indent-tools user-option is turned on (set to t). the pel-use-smart-shift user-option is turned on (set to t).
Contenting Employers	•	<f11> <tab> <f1></f1></tab></f11>		Open the <u>Name Indentation</u> 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.
Indication control Indication control For several major modes, the value of the tab-width variable is controlled by a mode specific user options variable, like pell-clab-width to form the control of the control		<f11> <tab> <f2></f2></tab></f11>		
mode, it those buffers he value of also videth will be set by PEI, to the mode specific value when the buffer is operated. See vitin <110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 110 < <120 > 1		<f11> <tab> <f3></f3></tab></f11>		Customize Emacs indentation control groups: indent, indent-tools, smart-shift. • If OTHER-WINDOW is non-nil (use C-u), display in another window.
Show Indentation settings	Indentation with PEL			
This modifies a buffer load value of the the lab-width use-option	Show Indentation settings		(pel-show-indent &optional	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.
This modifies a buffer local value of the the tab-width user-option. The provides a specialized user-option or at the default value of the default value of bashwidth to several major modes. For example, to change the tab width used to several major modes. Show state of text modes *Whether hard table are used for indentation, tab-width in several major modes. *Elize to a 2		<f11> M-t</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.
• Whether hard tabs are used for indentation. • Window in indentation, tab width with the second of the property of the prope	buffer	PEL provides a specialized use Go source code files, change to	er-option to set the default value of tak he ' pel-go-tab-width ' user-option var	p-width for several major modes. For example, to change the tab width used for all
In Emacs the behaviour of Tab Key	used for indentation, tab- width electric-quote-mode delete-selection-mode enriched-mode overwrite-mode case folding subword, superword, glass modes visible-mode, smart- dash-mode	This command is specially useful to see if hard-tabs or spaces are inserted for indentation: • When indent-tabs-mode is active, Emacs inserts a number of hard tabs and spaces. • 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 for a buffer in the</mode>		-UU-:F1 keymaps.c Top (1,0) Git-master (C/*la WK Anzi Modes status: - Local indent-tabs-mode : off: use spaces. Tab width = 8 - Local electric-quote-mode: off. - delete-selection-mode : off. - enriched-mode : not loaded. - overwrite 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: on. - Sentences end with 2 space characters. - paragraph-start : "[]*\\('/+**\\)[]*\$\\\^^L"
Psy default, in text modes, tabs are set to 8 spaces, inserting hard tabs. However, if there is text in the above lines, the tab moves to the spot under two 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. Indent current line (or region) Cab	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 .
Soptional ARG) **In This function either inserts a tab, or indents the current line, or performs symbol completion, depending on 'tab-always-indent'. The function called tractually indent the line or insert a tab is given by the variable 'indent-line-function'. **In 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 stat at the beginning of the current line, to reflect the current line is 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 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. **At 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 controlled		By default, in text modes, tabs word above. Note that if a line	are set to 8 spaces, inserting hard takis full of text (without any space), then	os. However, if there is text in the above lines, the tab moves to the spot under the the tab stops controlled by the ruler take effect again.
* tab-always-indent = t Several major modes adjust the behaviour of the tab key to perform semantically aware indentation, such as what is being done in Lisp. (c-indent-line-or-region & optional ARG REGION) In C related modes: Indent active region, current line, or block starting on the I 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. C-M-q Indent lines of list after point Example: CLBC s3.lisp C-M-q Indent-exp & optional ENDPOS) (c-indent-exp & optional SHUTUP-P) Insert spaces or tabs to next defined tab-stop column BI-C, BI-C++, BI-D The exact location of the next tab stop is identified by the value of the tab-width variable is controlled by a mode specific user options variable, like pel-c-tab-width for buffers in c-mode. In those buffers the value of tab-width is set by PEL to the mode specific value when opening the buffer. See with <fi1> < tab> N-RET (pel-newline-and-indent-below) Insert an indented line just below current line. * The command can also align text vertically if this special mode was activate for the buffer with the <fi1> M-RET.</fi1></fi1>	region)	actually indent the line or insert If a prefix argument is given, af at the beginning of the current In most major modes, if point v the same position relative to th If 'transient-mark-mode' is turn The behaviour of the tab k correct for the current context, to documented on the behaviour of would be to place a description in	b, or indents the current line, or perfort a tab is given by the variable 'indent ter this function indents the current line, to reflect the current line's indent was in the current line's indentation, it e text. The dot and the region is active, this further vastly differ between major modes cycling through various potential post that key and how it can be controlled aside the table of each major mode.	rms symbol completion, depending on 'tab-always-indent'. The function called to -line-function'. The or inserts a tab, it also rigidly indents the entire balanced expression which starts ation. The is moved to the first non-whitespace character after indenting; otherwise it stays at notion instead calls 'indent-region'. In this case, any prefix argument is ignored. This ranges from not moving the cursor at all if the indentation is identified as itions to just what someone new to Emacs would expect. Much more has to be and customized. It's quite possible that the best way to document its behaviour
(c-indent-line-or-region & optional ARG REGION) In C related modes: Indent active region, current line, or block starting on the I 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 ENDPOS Example: CLBC s3.lisp		<tab></tab>		·
ARG REGION) • 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 C-M-q • (indent-sexp & optional ENDPOS) • (c-indent-exp & optional SHUTUP-P) Insert spaces or tabs to next defined tab-stop column Part - C , Part - C++, Part - D Insert spaces or tabs to next defined tab-stop column. • 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. With PEL, for several major modes, the value of the tab-width variable is controlled by a mode specific user options variable, like pel-c-tab-width for buffers in c-mode. In those buffers the value of tab-width is set by PEL to the mode specific value when opening the buffer. See with <f11> <tab> (pel-newline-and-indent-below) Insert an indented line below current line. • The command can also align text vertically if this special mode was activated for the buffer with the <f11> M-RET • The command can also align text vertically if this special mode was activated for the buffer with the <f11> M-RET • The command can also align text vertically if this special mode was activated for the buffer with the <f11> M-RET</f11></f11></f11></tab></f11>				
The command used depends on the major mode of the current buffer. Insert spaces or tabs to next defined tab-stop column		<tab></tab>		In Transient Mark mode, when the region is active, reindent the region. With prefix argument, rigidly reindent the expression starting on current line.
• 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. With PEL, for several major modes, the value of the tab-width variable is controlled by a mode specific user options variable, like pel-c-tab-width for buffers in c-mode. In those buffers the value of tab-width is set by PEL to the mode specific value when opening the buffer. See with <f11> <tab> Insert an indented line below current line • M-RET • <f11> <tab> RET (pel-newline-and-indent-below) • The command can also align text vertically if this special mode was activate for the buffer with the <f11> M-RET • The command can also align text vertically if this special mode was activate for the buffer with the <f11> M-RET • The command can also align text vertically if this special mode was activate for the buffer with the <f11> M-RET • The command can also align text vertically if this special mode was activate for the buffer with the <f11> M-RET • The command can also align text vertically if this special mode was activate for the buffer with the <f11> M-RET • The command can also align text vertically if this special mode was activate for the buffer with the <f11> M-RET • The command can also align text vertically if this special mode was activate for the buffer with the <f11> M-RET • The command can also align text vertically if this special mode was activate for the buffer with the <f11> M-RET • The command can also align text vertically if this special mode was activate for the buffer with the <f11> M-RET</f11></f11></f11></f11></f11></f11></f11></f11></f11></tab></f11></tab></f11>	point	С-м-q	ENDPOS) • (c-indent-exp & optional	
• <f11> <tab> RET • The command can also align text vertically if this special mode was activate for the buffer with the <f11> M-RET.</f11></tab></f11>	next defined tab-stop column	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.		
• To see the current behaviour use <f11> t a ?: it displays whether the N RET command aligns text or not.</f11>			(pel-newline-and-indent-below)	The command can also align text vertically if this special mode was activated for the buffer with the <f11> M-RET. To see the current behaviour use <f11> t a ?: it displays whether the M-</f11></f11>

Description	<u>Keystroke</u>	Function	<u>Note</u>	
Toggle text alignment on	<f11> M-RET</f11>	(pel-toggle-newline-indent-align)	Toggle variable <i>pel-newline-does-align</i> for the local buffer.	
pel-newline-and-indent- below		I-newline-and-indent-below' operates.		
See also: <u>∑ Align</u>	 If pel-newline-does-align is t, it aligns several syntactic element in the current block: the comments, the assignments. Identify modes where pel-newline-does-align is automatically activated (set to t) by adding the major mode to the list in the pel-modes-activating- 			
	align-on-return user option.This affects the behaviour of th			
	 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) 			
Display current buffer's	<f11> t a ?</f11>	(pel-align-info &optional APPEND)	Print information about text aligning info in a *pel-align-info* help-mode buffer.	
vertical alignment behaviour			Prints current state and values of relevant user-options as buttons you can use to get more info and change their customized values:	
See also: <u>∑ Indentation</u>			 behaviour of M-RET in the 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. 	
Change the tab stops	M-x 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. Again, the tab stop take effect at the top of the buffer,	
Change the tab width	M-: (setq tab-width N)		ults to 8 in emacs. It can be set locally inside a buffer with (setq tab-width N) or N). The M-: keystroke allows evaluating a lisp expression interactively (as the	
		above two).	er impact the location of the column where the next character shows. When	
			shown in the window that contains literal tabs will be modified according to the	
		• With PEL, remember that the lo	cal value of tab-width is controlled by PEL and set to the value of a mode specific	
Make <tab> insert space/</tab>	M	user option, such as pel-c-tab-wic	th in c-mode buffers. By default, pressing <tab> insert literal (hard) tabs inside the file. The indent-</tab>	
tab	M-: (setq indent-tabs- mode nil/t)		tabs-mode variable controls that: set to t it inserts tabs, set to nil it inserts spaces.	
Next line, indented	c-j	(electric-newline-and-maybe-indent)	Add new line and indent next line. Indentation is controlled by the variable left-margin.	
Indoné Desile		,	Pressing Tab anywhere on the line also indents the line properly.	
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.	
			Indent each line via 'indent-according-to-mode'.	
Move to fist nonbank character on the line	М-т	(back-to-indentation)	Move point to the first non-whitespace character on this line.	
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.	
Delete Indentation, join line to the previous one	• M-^ • <f11> ⊠ 6</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.	
See also: <u>See also: Whitespace</u>	• <f6> 6</f6>		With argument, join this line to following line.	
Indent relative to line above	<f11> <tab> r</tab></f11>	(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.	
	The following line shows the indentation points in this line.			
	 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'.			
			ned with the first non-whitespace character on the previous line (actually, the last stop instead—unless called with a numeric argument, in which case do nothing.	
Indenting and un-	The following commands provide non-semantic indentation of the current line or marked region. 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			
indenting rigidly				
Indont/Unindont rigidly	allow further use of the comma		Indont rigidly the marked region or current line N times	
Indent/Unindent rigidly	• C-x <tab> • <f11> <tab> <tab></tab></tab></f11></tab>	(pel-indent-rigidly &optional N)	Indent rigidly the marked region or current line N times. • If a region is marked, it uses 'indent-rigidly' and provides the same prompts • a captal indentities have a	
See also: <u>Xey-Chords</u>	• <tab>q</tab>		to control indentation changes. • If no region is marked, it operates on current line(s) identified by the numeric	
			argument N (or if not specified N=1): • N = [-1, 0, 1] : operate on current line	
			 N > 1 : operate on the current line and N-1 lines below. N < -1 : operate on the current line and (abs N) -1 lines above. 	
			with PEL, the <tab>q key-chord is also available when pel-use-key-chord is non-nil. See ∑ Key-Chords.</tab>	
See also:			Command numeric prefix is available with the key-chord binding.	
• <u>\$1 - C</u> • \$1 - C++			FEL rebinds this key, but it extends the functionality: pel-indent-rigidly uses indent-rigidly, described below the dashed line.	
• <u>\$\$t - D</u>		PEL uses the above instead of the standard:	Indent all lines starting in the region.	
• <u>M reStructuredText</u>		(indent-rigidly START END ARG	 If called interactively with no prefix argument, activate a transient mode in which the indentation can be adjusted interactively by typing <left>,</left> 	
		&optional INTERACTIVE)	<right>, <s-left>, or <s-right>.</s-right></s-left></right>	
	Both of these commands activate a transient mode where Emacs prompts for extra keys to control how to indent. Indenting and un-indenting is possible. The capabilities are controlled by the variable <i>indent-rigidly-map</i> with by default provides:			
	S- <right> indent-rigidly-right-to-tab-stop S-<left> indent-rigidly-left-to-tab-stop</left></right>			
	<pre>• <right> indent-rigidly-right • <left> indent-rigidly-left</left></right></pre>			
	Typing any other key deactivates the transient mode. • The S- <right> and S-<left> keys indent/de-indent to the next tab-stop position, which is controlled by the tab-width user option.</left></right>			
	With PEL, for several major modes, the indentation is controlled by a mode-specific user option variable. For example, for buffers in c-mode, the value of pel-c-tab-width is automatically stored into tab-width when the buffer is opened.			
	1 If you use the cua-mode: the	⚠ If you use the cua-mode: the cua-mode uses C-x, to invoke this command when cua-mode is active, type it really fast or type C-x C-x <tab> (or</tab>		
	use the PEL binding <f11> <tab< td=""><td>ab> <tab>).</tab></td><td></td></tab<></f11>	ab> <tab>).</tab>		

<u>Description</u>	<u>Keystroke</u>	Function	<u>Note</u>
Indent line(s) rigidly	• <f6> <tab> • <f11> <tab> c</tab></f11></tab></f6>	(pel-indent-lines &optional N)	Indent current or marked lines by N indentation levels
	Works with point anywhere on a All lines touched by the region. A special argument N can specifie If a negative number is specifie If a region is marked, the function characters in all affected lines. Use C-g to de-activate the reg Handles presence of hard tabs. If indent-tabs-mode is non-reserved.	are indented. ify more than one indentation level. It d, 'pel-unindent-lines' is used. on does not deactivate it to allow repe ion. in the indentation is created with a min by hard tab in the indentation of the ma	eated execution of the command. It also modifies the region to include all
Un-indent line(s) rigidly	• <backtab> • <f6> <backtab> • <f11> <tab> C</tab></f11></backtab></f6></backtab>	(pel-unindent-lines &optional N)	Un-indent current line or marked lines by N indentation levels.
	If a region was marked, the fun characters in all affected lines Use C-g to de-activate the reg Handles presence of hard tabs: If indent-tabs-mode is non-re	are un-indented. ion does not deactivate it to allow rep ction does not deactivate it to allow re ion. : iil the indentation is created with a mix by hard tab in the indentation of the ma	epeated execution of the command. It also modifies the region to include all
Controlling use of 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. 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 allowing fine control over set of files and buffers. PEL provides the following related commands. See also: Whitespace		
Toggle use of hard tabs and only spaces for indentation in the current buffer	<f11> t w I</f11>	(pel-toggle-indent-tabs-mode &optional ARG)	Toggle use of hard tabs or spaces for indentation in 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.
Replacing Tabs with spaces or spaces with tabs	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 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 See also: Whitespace	<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 changing the column they end at. If called interactively with prefix ARG, convert for the entire buffer.
<u>Indent-tools</u>	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.		
Open the indent-tools	<f11> <tab> <f7></f7></tab></f11>	(indent-tools-hydra/body)	Activate the "indent-tools-hydra" hydra.
hydra See also: <u>乳(- Python</u>	C-c >		 With PEL, this key binding is only available when: globally, when pel-indent-tools-key-bound is set to globally, in python-mode only when pel-indent-tools-key-bound is set to python. The actual key is selected by indent-tools indent-tools-keymap-prefix user-option, the default is C-c >
See also: <u>∑ Hide/Show</u>	The heads for the associated hydra are: >: 'indent-tools-indent', <: 'indent-tools-demote', E: 'indent-tools-indent-end-of-defun', c: 'indent-tools-comment', P: 'indent-tools-uncomment', I: 'indent-tools-indent-paragraph', I: 'indent-tools-indent-end-of-level', K: 'indent-tools-kill-tree', C: 'indent-tools-solor-goto-parent', e: 'indent-tools-goto-parent', d: 'indent-t		

<u>Description</u>	<u>Reystroke</u>	runction	isote
Smart-shift	It is implemented as a minor or You can identify the smart-shift use the commands manually o buffers. PEL controls it through custom The smart-shift external PEL also provides the pel commands that shift line(s). I change the prefix key to use the smart shift line is	global minor mode that must be enable-mode inside one of the pel- <mode> r through the key bindings provided by ization user-options: package PEL activates it when the -smart-shift-keybinding user-option is By default the key bindings are using C</mode>	region of lines right or left but also up or down. led first. -activates-minor-modes user-options to activate it automatically. You can also pel to activate the smart-shift-mode in the current buffer or globally for all pel-use-smart-shift user-option is turned on (set to t). that allows you to select additional alternative key bindings for the smart-shift C-c as a key prefix. With PEL you can also use a control key for the cursor or dings are shown below but only one of them will be available at any given time.
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.
Shift line or region right	• C-c <right> • C-c <c-right> • <f9> <right></right></f9></c-right></right>	(smart-shift-right &optional ARG)	Shift the line or region to the ARG times to the right. With PEL one of the extra key bindings can be enabled via the pel-smart-shift-keybinding user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.
Shift line or region left	• C-c <left> • C-c <c-left> • <f9> <left></left></f9></c-left></left>	(smart-shift-left &optional ARG)	Shift the line or region to the ARG times to the left. With PEL one of the extra key bindings can be enabled via the pel-smart-shift-keybinding user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.
Shift line or region up	• C-c <up> • C-c <c-up> • <f9> <up></up></f9></c-up></up>	(smart-shift-up &optional ARG)	Shift the line or region to the ARG times to the upwards. With PEL one of the extra key bindings can be enabled via the pel-smart-shift-keybinding user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.
Shift line or region down	• C-c <down> • C-c <c-down> • <f9> <down></down></f9></c-down></down>	(smart-shift-down &optional ARG)	Shift the line or region to the ARG times to the downwards With PEL one of the extra key bindings can be enabled via the pel-smart-shift-keybinding user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.

Function

Note

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??

Description

Kevstroke

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 does not yet integrate the smarttabs package. 🚧 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 correctly

GOTTOON).		
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	