

Inserting Text

Description	Keystroke	Function	Note
Inserting Text	The commands described in this table insert specialized text at point (cursor) location. <ul style="list-style-type: none"> The first sections of the table show commands that are not template-based. The bottom section describe the flexible template system supported by PEL: tempo skeletons and yasnipppet. 		
Open this PDF file. See also: 🔗 Help/Info	<ul style="list-style-type: none"> <f11> i <f1> <f11> y <f1> <f11> _ <f1> 	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the 🔗 Inserting Text 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 Text Insertions control	<f11> i <f2>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL text insertion support: lice, smart-dash, tempo, time-stamp, yasnipppet. <ul style="list-style-type: none"> If OTHER-WINDOW is non-nil (use C-u), display in other window.
🔗 Customize Emacs Text Insertions control	<f11> i <f3>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs text insertion support: lice, smart-dash, tempo, time-stamp, yasnipppet
Insert time & file info	The following commands insert time stamps of specific formats and name of the current file.		
Insert current date	<f11> i d	(pel-insert-current-date &optional UTC)	Insert current date (only, no time) at point. <ul style="list-style-type: none"> Local by default, UTC if C-u prefix used.
Insert current date & time	<f11> i D	(pel-insert-current-date-time &optional UTC)	Insert current date and time at point. <ul style="list-style-type: none"> Local by default, UTC if C-u prefix used.
Insert current filename	<ul style="list-style-type: none"> <f11> i f <f6> f 	(pel-insert-filename &optional N)	Insert the file name of the currently edited file at point. <ul style="list-style-type: none"> By default: insert filename of current buffer with complete absolute path. With a numeric argument you can select the file name of the current buffer or the buffers in the 4 surrounding windows. 8: up, 2: down, 4: left, 6: right. Any other number identifies the current window. <ul style="list-style-type: none"> When the numeric argument is positive the file with complete absolute path is inserted, With negative numeric argument the path is omitted.
Insert time stamp	<f11> i t	(pel-insert-iso8601-timestamp &optional UTC)	Insert ISO 8601 conforming abbreviated YYYY-MM-DD hh:mm:ss format timestamp. <ul style="list-style-type: none"> Local by default, UTC if C-u prefix is used.
Insert software license text	<ul style="list-style-type: none"> <f11> i L <f6> L 	(lice NAME)	Insert license and headers at point. Prompts for license NAME, which is a license template name like “mit”, “gpl-3.0”, etc... The list is available with TAB completion: hit TAB on prompt to get the complete list of templates. <div>📦 Requires the lice external package. 🔗 PEL activates it if pel-use-lice user option is t.</div>
Automatic File Time Stamp on file save References: <ul style="list-style-type: none"> TimeStamps @ EmacsWiki Change time stamp format in: <ul style="list-style-type: none"> markdown file reStructuredText file See also: 🔗 File mngt	Emacs has a built-in automatic time-stamping of files . It must be activated by adding the time-stamp function to the before-save-hook variable. This can either be done via Emacs customization system or explicitly inside your init file with the following code: <pre>(add-hook 'before-save-hook 'time-stamp)</pre> <ul style="list-style-type: none"> The time stamp will be added to files that contain, inside their first 8 lines, a line that looks like one of the following: <ul style="list-style-type: none"> Time-stamp: <> Time-stamp: “ ” <div>👉 You can, however change these defaults and get Emacs to update all sorts of time stamp formats, even inside source code statements:</div> <div>🔧 Emacs controls automatic insertion of timestamp with the following variables: <ul style="list-style-type: none"> time-stamp-pattern consists of 4 parts, each one controlled by a variable: <ul style="list-style-type: none"> time-stamp-line-limit : identifies where in the file the time stamp can be located. Defaults to 8: the first 8 lines. time-stamp-start: identifies the text pattern that precedes the time stamp. time-stamp-end: identifies the end of the time stamp. time-stamp-format specifies the format of the time stamp. <ul style="list-style-type: none"> Something like “%:y-%02m-%02d %02H:%02M:%02S %u” to specify the date and time in ISO format, with the user login’s name. time-stamp-time-zone specifies the time zone selection: <ul style="list-style-type: none"> nil: Emacs local time t: Universal time wall : system wall clock time TZ : controlled by a TZ environment variable The time-stamp-format and time-stamp-time-zone variables can be set in your init file or via the Emacs customization system. <ul style="list-style-type: none"> They are defined in the time-stamp customization group. <ul style="list-style-type: none"> 👉 To change the format or the pattern preceding or after the automatically updated time stamp, it is best to use file local variables: this will allow automatic time stamp updates in files with various formats. As an example, see the top and end of the PEL manual raw format file. <div>⚠ By default, the time-stamp string must be placed within the first 8 lines of the file, otherwise it will not be updated automatically. <ul style="list-style-type: none"> If you want it located somewhere else in your file set the time-stamp-line-limit file local variable. </div> <div>🔧 PEL provides the extra user-option to control the automatic generation of time-stamps: <ul style="list-style-type: none"> pel-update-time-stamp user-option controls whether time-stamps are automatically update time stamps in all files where a valid time-stamp corresponding to Emacs settings as described above. Set it to t (the default) to allow automatic time stamp updates. Set it to nil to prevent them. You can also toggle it globally for the current editing session by using the <f11> f M-t key sequence. </div> <div>👉 To insert a non-updatable time stamp, the PEL package provides a set of text insert commands which include inserting a time stamp . <ul style="list-style-type: none"> See the 🔗 Inserting Text table for the appropriate commands. </div> </div>		
Update file time stamp See also: 🔗 File mngt	<f11> f t	(time-stamp)	Force update the time stamp string(s) in the current buffer. <ul style="list-style-type: none"> Updates a time stamp of format recognized by <i>Emacs current settings</i> even when automatic time-stamp update is off. More information about the “<i>Emacs current settings</i>” in the description block above.
Toggle time stamp automatic update	<f11> f M-t	(time-stamp-toggle-active &optional ARG)	Toggle ‘time-stamp-active’, setting whether <f11> f t updates a buffer. <ul style="list-style-type: none"> With ARG, turn time stamping on if and only if arg is positive.
Inserting & Automatically Updating Copyrights	Emacs has built-in support for insertion and update of copyright notices inside files. <ul style="list-style-type: none"> Two commands, shown below, are provided to manually insert or update the file’s copyright notice. The copyright notice can be automatically updated by adding the copyright-update function to the list of before-save-hook variable with the following code: <pre>(add-hook 'before-save-hook 'copyright-update)</pre> <div>⚠ To be automatically updated, the copyright notice must be placed within an area at the beginning of the file specified by the value of the copyright-limit variable, normally defined as the first 2000 characters. This variable is customizable.</div>		
Insert copyright notice at point See also: 🔗 File mngt	<f11> i c	(copyright &optional STR ARG)	Insert a copyright by \$ORGANIZATION notice at cursor. <ul style="list-style-type: none"> If the ORGANIZATION environment variable is not available, Emacs prompts for it.
Update file’s copyright notice	<f11> i M-c	(copyright-update &optional ARG INTERACTIVEP)	Update copyright notice to indicate the current year. <ul style="list-style-type: none"> With prefix ARG, replace the years in the notice rather than adding the current year after them. If necessary, and ‘copyright-current-gpl-version’ is set, any copying permissions following the copyright are updated as well. <div>⚠ Even when used interactively copyright-update does not warn if there is no copyright in the current buffer to update. It does not create a missing notice.</div> <div>👉🔗 If you want automatic copyright notice updates when a modified buffer is saved, set the pel-update-copyright user option to t. <ul style="list-style-type: none"> Without PEL add the following inside your init.el file: <pre>(add-hook 'before-save-hook 'copyright-update)</pre> </div>

Description	Keystroke	Function	Note										
Toggle pel-tempo-mode	<f6> SPC	(pel-tempo-mode &optional ARG)	Toggle PEL tempo mode on/off. PEL tempo mode activates C-c . and C-c , as well as to C-c C-. and C-c C-, key bindings to navigate across tempo mark hot-spots. When pel-tempo-mode is active the pel-tempo-mode lighter (‡) is shown on the status bar. The second set of keys are only available when Emacs runs in graphics mode. 👉 The pel-generic-file-header command inserts the text using a tempo skeleton: the PEL tempo mode is automatically activated by typing <f6> h.										
Jump to next tempo mark	<ul style="list-style-type: none">C-c M-fC-c .C-c C-.	(tempo-forward-mark)	Jump to the next mark in ‘tempo-back-mark-list’: the location where code must be updated inside the inserted skeleton. <ul style="list-style-type: none">These key key bindings are only available when pel-tempo-mode is active.										
Jump to previous tempo mark	<ul style="list-style-type: none">C-c M-bC-c ,C-c C-,	(tempo-backward-mark)	Jump to the previous mark in ‘tempo-back-mark-list’: the location where code must be updated inside the inserted skeleton. <ul style="list-style-type: none">These key binding are only available when pel-tempo-mode is active.										
Store PEL code template settings in .dir-locals.el to fine-tune layout of files in a directory tree	👉 Emacs user options by default take effect globally. But by using file and directory variables (see 🔗 File/Directory Variables) they can also be used to take effect on a single file or all files inside a directory tree. So by default, the user options that control the PEL tempo template take effect globally. If you want to change the behaviour for only one file, write the user option control block at the end of that file. If you want to control the behaviour of the PEL tempo templates for all files inside a directory tree create a .dir-locals file and store the values of the relevant options variables inside that file. This allows you to control the user options affecting the format of the tempo templates precisely.												
Example:	Although the default settings of pel-generic-skel-module-section-titles identifies the 3 sections “Module Description”, “Dependencies” and “Code” you can keep this for other files but inside a directory you can force all shell-mode files to use 2 sections: “Description” and “Script” and ensure that all files have a 1-line copyright notice with the .dir-locals.el file containing the following code: <pre>;;; Directory Local Variables ;;; For more information see (info "(emacs) Directory Variables") ((nil . ((pel-generic-skel-with-copyright . t) (pel-generic-skel-with-license . "MIT")))) (sh-mode . ((pel-generic-skel-module-section-titles . ("Description" "Script")))))</pre>												
Entering Templated Text with Tempo Skeletons	Emacs built-in support includes the tempo skeletons . PEL implements extension to the tempo skeleton Emacs built-in package under two prefix keys: <ul style="list-style-type: none">The commands under the <f6> prefix keys insert template text that are adapted to each major mode. They are generic in nature, and dynamically adapt to the major mode and the comment style supported by the major mode. The layout of the templates is the same for every major mode, they differ only by the comment strings.The commands under the <f12> <f12> prefix key insert templates specialized for the programming or markup language of the major mode that support this key prefix. PEL attempts to use the same key bindings for equivalent concepts (such as file header block) inside each mode specific instance of the <f12> <f12> key maps as much as possible. The tempo skeletons provided by PEL can be quite complex and their formats are controlled by user options. PEL currently only support this key prefix with for the following major modes (more are planned):<ul style="list-style-type: none">C, Emacs Lisp, ErlangreStructuredText												
Major-mode specific Tempo Templates Prefix	<f12> <f12>		Key prefix sequence to the list of tempo skeleton commands. <ul style="list-style-type: none">This command prefix is available only for some major modes (see the list in the first column) of the section row above.The commands under this prefix insert text specialized for their specific major mode, as opposed to the commands bound to the <f6> prefix key. For more information see the language specialized reference table.										
Entering Templated Test with Yasnippet	PEL also supports the popular yasnippet external package which provides another way to insert templated text, and yasnippet-snippets external package which provides a large set of code snippets for a large set of major modes. <ul style="list-style-type: none">To use yasnippets, you must type the snippet abbreviation and then hit the TAB key to expand the text. <div>📦 Requires yasnippet 🔗 activated when pel-use-yasnippet is set to t or to use-from-start.</div> <div>📦 Requires yasnippet-snippets 🔗 activated when pel-use-yasnippet-snippets is set to t.</div> <ul style="list-style-type: none">Use the key <f11> i <f2> to access the PEL Insertion customization buffer to customize these user options (see above, first row).The list of snippets available in the current buffer is listed in the menu bar (see 🔗 Menus) and can also be listed using the yas-describe-tables command (which PEL binds to <f11> y t). <p>PEL bins the following yasnippet commands to keys in the pel: key prefix, shown below.</p>												
🔗 Customize PEL yasnippet use	<f11> y <f2>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL Yasnippet text insertion support. <ul style="list-style-type: none">If OTHER-WINDOW is non-nil (use C-u), display in other window.										
🔗 Customize Emacs yasnippet control	<f11> y <f3>	(pel-customize-library &optional OTHER-WINDOW)	Customize Yasnippet groups: yasnippet, yasnippet-snippets, yas-minor										
Toggle YASnippet minor mode on/off	<f11> y y	(yas-minor-mode &optional ARG)	Toggle YaSnippet mode. <div><ul style="list-style-type: none">When YASnippet mode is enabled, ‘yas-expand’, normally bound to the TAB key, expands snippets of code depending on the major mode.With no argument, this command toggles the mode. Positive prefix argument turns on the mode. Negative prefix argument turns off the mode.YASnippet mode key bindings:<table><tr><td>key</td><td>binding</td></tr><tr><td>-----</td><td>-----</td></tr><tr><td>C-c & C-n</td><td>yas-new-snippet</td></tr><tr><td>C-c & C-s</td><td>yas-insert-snippet</td></tr><tr><td>C-c & C-v</td><td>yas-visit-snippet-file</td></tr></table></div>	key	binding	-----	-----	C-c & C-n	yas-new-snippet	C-c & C-s	yas-insert-snippet	C-c & C-v	yas-visit-snippet-file
key	binding												
-----	-----												
C-c & C-n	yas-new-snippet												
C-c & C-s	yas-insert-snippet												
C-c & C-v	yas-visit-snippet-file												
Toggle YASnippet global mode on/off	<f11> y Y	(yas-global-mode &optional ARG)	Toggle Yas minor mode in all buffers. <ul style="list-style-type: none">With prefix ARG, enable Yas-Global mode if ARG is positive; otherwise, disable it.										
Expand snippet whose name is just before point	TAB	(yas-expand &optional FIELD)	Expand a snippet before point. If no snippet expansion is possible, do nothing. <ul style="list-style-type: none">This key binding is only active when the YASnippet mode is active. Once the snippet was expanded the TAB key normal behaviour is restored.										
Write a new snippet	<ul style="list-style-type: none"><f11> y nC-c & C-n	(yas-new-snippet &optional NO-TEMPLATE)	Pops a new buffer for writing a snippet. <ul style="list-style-type: none">Expands a snippet-writing snippet, unless the optional prefix arg NO-TEMPLATE is non-nil.										
Prompt for snippet and insert it	<ul style="list-style-type: none"><f11> y sC-c & C-s	(yas-insert-snippet &optional NO-CONDITION)	Choose a snippet to expand, pop-up a list of choices according to ‘yas-prompt-functions’. <ul style="list-style-type: none">With prefix argument NO-CONDITION, bypass filtering of snippets by condition.										
Visit a snippet file	<ul style="list-style-type: none"><f11> y vC-c & C-v	(yas-visit-snippet-file)	Choose a snippet to edit, selection like ‘yas-insert-snippet’. <ul style="list-style-type: none">Only success if selected snippet was loaded from a file. Put the visited file in ‘snippet-mode’.										
Display all snippets for current major mode	<f11> y t	(yas-describe-tables &optional WITH-NONACTIVE)	Display snippets for each table.										
Prints Yasnippet version info	<f11> y ?	(yas-about)	Prints version information in the mini buffer.										

Inserting Text — References

Topic & link	Description
GNU Emacs Manual: Time Stamps	
Smart-Dash Mode homepage	A description of this extremely useful mode and why it was created.