# Inserting Text

| Description | Keystroke | Function | Note |
|---|---|---|---|
| **Inserting Text** | The commands described in this table insert specialized text at point (cursor) location. These are not template-based, and are mostly PEL specific commands. | | |
| **Text Insertions**<br>(See Also: ∑ Customize) | **`<f11> <f1> I`** | (**pel-cfg-insertions** &optional OTHER-WINDOW) | Customize PEL text insertion support.<br>• If OTHER-WINDOW is non-nil (use **C–u**), display in another window and open insertion related and supported groups: lice, smart-dash. |
| **Time-stamps** | | | |
| **Insert current date** | **`<f11> i d`** | (**pel-insert-current-date** &optional UTC) | Insert current date (only, no time) at point.<br>• 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.<br>• Local by default, UTC if C-u prefix used. |
| **Insert current filename** | • **`<f11> i F`**<br>• **`<f6> F`** | (**pel-insert-filename** &optional N) | Insert the file name of the currently edited file at point.<br>• By default: insert filename of current buffer with complete absolute path.<br>• 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.<br>  • When the numeric argument is positive the file with complete absolute path is inserted, if the numeric argument is negative the path is omitted.<br>☛ When creating keyboard macros that must insert text that includes a file name, you can open 2 windows and use this facility to enter the name of the file that is located in the other window, making the keyboard macro more general. |
| **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.<br>• Local by default, UTC if C-u prefix is used. |
| **Insert software license text** | • **`<f11> i L`**<br>• **`<f6> L`** | (**lice** NAME) | Insert license and headers at point.<br>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.<br>📦 Requires the **lice** external package. 🔼 PEL activates it if **pel-use-lice** user option is t. |
| **Updating file time stamp** | | | |
| **Update file time stamp**<br><br>(See Also: ∑ File mngt) | **`<f11> f t`** | (**time-stamp**) | Force update the time stamp string(s) in the current buffer.<br>The time stamp is updated if the one of the following strings is found in the first 8 lines of the file:<br>• Time-stamp: <><br>• Time-stamp: " "<br>☝ If you want time stamps updated automatically, write the following inside your init.el file:<br>    `(add-hook 'before-save-hook 'time-stamp)` |
| **Toggle time stamp automatic update** | | (**time-stamp-toggle-active** &optional ARG) | Toggle 'time-stamp-active', setting whether **`<f11> f t`** updates a buffer.<br>• 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.<br>• Two commands, shown below, are provided to manually insert or update the file's copyright notice.<br>• 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:<br>    `(add-hook 'before-save-hook 'copyright-update)`<br>⚠️ 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. | | |
| **Insert copyright notice at point**<br><br>(See Also: ∑ File mngt) | **`<f11> i C`** | (**copyright** &optional STR ARG) | Insert a copyright by $ORGANIZATION notice at cursor.<br>• If the ORGANIZATION environment variable is not available, Emacs prompts for it. |
| **Update file's copyright notice** | | (**copyright-update** &optional ARG INTERACTIVEP) | Update copyright notice to indicate the current year.<br>• 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.<br>• If non-nil, INTERACTIVEP tells the function to behave as when it's called interactively.<br>⚠️ 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.<br>☝ If you want to be prompted automatically to update an existing but out-of-date copyright notice, write the following inside your init.el file:<br>    `(add-hook 'before-save-hook 'copyright-update)` |
| **Insert Commented Lines** | The following commands help insert commented lines or just underlines the current line of text using the character corresponding to one of the adornment level used for reStructuredText sections. The strings are commented according to the major mode of the current buffer. If the buffer has no identified comment strings, the command prompts for them the first time it is used in that type of buffer.<br>The following commands are also listed in the ∑ Comments table. | | |
| **Insert commented line**<br><br>(See also: ∑ Comments) | • **`<f11> i l`**<br>• **`<f6> l`** | (**pel-insert-line** &optional LINELEN) | Insert a (commented) line before/at current line.<br>• If point is at the beginning of the line insert it there.<br>• If point is in the middle of a line, move point at beginning of line before inserting it.<br>• The number of dash characters of the line is specified by LINELEN.<br>  • If LINELEN is not specified the buffer's fill-column value is used.<br>☛ fill-column is customizable and can be used as a file or directory variable. |
| **Comment-underline current line with level 1 adornment** | **`<f11> _ 1`** | (**pel-commented-adorn-1**) | Insert a commented level-1 reST line adornment at point. |
| **Comment-underline current line with level 2 adornment** | **`<f11> _ 2`** | (**pel-commented-adorn-2**) | Insert a commented level-2 reST line adornment at point. |
| **Comment-underline current line with level 3 adornment** | **`<f11> _ 3`** | (**pel-commented-adorn-3**) | Insert a commented level-3 reST line adornment at point. |
| **Comment-underline current line with level 4 adornment** | **`<f11> _ 4`** | (**pel-commented-adorn-4**) | Insert a commented level-4 reST line adornment at point. |
| **Comment-underline current line with level 5 adornment** | **`<f11> _ 5`** | (**pel-commented-adorn-5**) | Insert a commented level-5 reST line adornment at point. |
| **Comment-underline current line with level 6 adornment** | **`<f11> _ 6`** | (**pel-commented-adorn-6**) | Insert a commented level-6 reST line adornment at point. |
| **Comment-underline current line with level 7 adornment** | **`<f11> _ 7`** | (**pel-commented-adorn-7**) | Insert a commented level-7 reST line adornment at point. |
| **Comment-underline current line with level 8 adornment** | **`<f11> _ 8`** | (**pel-commented-adorn-8**) | Insert a commented level-8 reST line adornment at point. |
| **Comment-underline current line with level 9 adornment** | **`<f11> _ 9`** | (**pel-commented-adorn-9**) | Insert a commented level-9 reST line adornment at point. |
| **Comment-underline current line with level 10 adornment** | **`<f11> _ 0`** | (**pel-commented-adorn-10**) | Insert a commented level-10 reST line adornment at point. |

| Description | Keystroke | Function | Note |
|---|---|---|---|
| **Smart Dash Mode** | 📦 This uses the **smart-dash** external package.  📘 PEL activates it when **pel-use-smart-dash** is set to **t**.  📘 The **pel-modes-activating-smart-dash-mode** user option identifies the major modes where PEL automatically activates the smart-dash-mode.  Anyone that has been writing Lisp code for a while knows that using dash as word separator instead of underscore is more natural and faster to type. Unfortunately most programming languages (all non-Lisp?) have restrictions on the characters available in identifiers and underscore is often used. Typing underscore requires hitting the Shift key and it annoys some people that enjoyed writing Lisp code. This is where the smart-dash-mode helps. You can insert underscore in text by typing the dash key without hitting the Shift key! A **very** useful mode. More information is available in the **author's page**. | | |
| Toggle smart dash mode | `<f11> M---` | (**smart-dash-mode** &optional ARG) | Toggle the smart-dash-mode on/off.  • When smart-dash-mode is active, it redefines the dash key to insert an underscore within C-style identifiers and a dash otherwise. This allows you to type all_lowercase_c_identifiers as comfortably as you would lisp-style-identifiers.  • While Smart-Dash mode is active, you can type **C−q  −** or use the minus key on the numeric keypad to override it and insert a dash after a C-style identifier character. You might need to do this if you want to type a cramped-looking expression like x-5.  • If Smart-Dash mode is activated while in a C-like mode (c-mode, c++-mode, and objc-mode by default, customizable with '**smart-dash-c-modes**') it will also activate Smart-Dash-C mode, which translates "**_>**" into "**->**" and "**__**" into "**--**" automatically so that struct pointer member access and postfix-decrement aren't made more difficult by Smart-Dash mode's tendency to insert underscores at the tail ends of identifiers whether you want it to or not. Note that this will necessitate that you type literal underscores if you want more than one underscore in a row. |

## 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. |