rst-mode: reStructuredText Mode

Description	Keystroke	Function	Note
Editing reStructuredText files	There's more information related to reStructuredText that need to be documented. The reSructuredText files are supported by the ret-mode which is available in standard Emacs distribution. To activate it under PEL, you must set the PEL **pel-use-rst-mode** customization variable to **t**.		
Customize PEL reStructuredText Support (See also: ∑ Customize)	• **<f11> <f1> SPC r** • **<f12> <f1>**	(**pel-cfg-pkg-reST** &optional OTHER-WINDOW)	Customize PEL reStructuredText support. • If OTHER-WINDOW is non-nil (use **C-u**), display in another window. • The **<f12> <f1>** binding is available when point is in a buffer visiting a reStructuredText file.
Activate reStructuredText mode	**M-x rst-mode**	(**rst-mode**)	Toggle the rst-mode used to edit reStructuredText markup.
Get version of rst-mode	**C-h v rst-version**		Shows the content of the variable rst-version. • Works once the rst-mode is loaded only.
Display table of content	**C-c C-t C-t**	(**rst-doc**)	Display a table of contents for current buffer inside another buffer. • Displays all section titles found in the current buffer in a hierarchical list. • The resulting buffer can be navigated, and selecting a section title moves the cursor to that section.
Indent list item (See ∑ Indentation)	**<tab>**	(**indent-for-tab-command** &optional ARG)	When point is anywhere on a list item line (a line that starts with one if the supported bullet characters), this cycles the indentation through the possible indentations of the item.
Comment (See ∑ Comments)	**M-;**	(**comment-dwim** ARG)	Comment line or region. **TODO:** the uncommenting does not work. According to the comment-dwim description it should. Need to investigate.
Move to previous section title	• **C-M-a** • **<f12> p** • **<f12> <up>**	(**rst-backward-section** OFFSET)	Jump backward OFFSET section titles ending up at the start of the title line. • OFFSET defaults to 1 and may be negative to move backward. • An OFFSET of 0 does not move unless point is inside a title. • Go to end or beginning of buffer if no more section titles in the desired • direction.
Move to next section title	• **C-M-e** • **<f12> n** • **<f12> <down>**	(**rst-forward-section** OFFSET)	Jump forward OFFSET section titles ending up at the start of the title line. • OFFSET defaults to 1 and may be negative to move backward. • An OFFSET of 0 does not move unless point is inside a title. • Go to end or beginning of buffer if no more section titles in the desired • direction.
Mark complete current section	**C-M-h**	(**rst-mark-section** &optional COUNT ALLOW-EXTEND)	Select COUNT sections around point. • Mark following sections for positive COUNT or preceding sections for negative COUNT.
Section level adornment	The rst.el library provides the rst-adjust command to create section adornment of the current line. This command tries to infer the level required and unfortunately sometimes fails when market is used and not expected by its code. PEL provides a set of very simple commands that use multiple key bindings to adorn the current line to a fixed section level: title level and up to 10 other levels, from 1 to 9 and then 0 for 10. It also provides commands to adorn a line to the same level as the previous section or a lower or higher level. And then to increase or decrease the section level of the adornment of the current line. PEL provides 3 style of section adornments: default, Sphinx-Python and CRiSPer, which can be selected with commands. PEL remembers the preferred style inside the customizable variable: **pel-rst-adornment-style**. The rest.el provides the **rst-preferred-adornment** user option to select the adornment characters for the various sections. PEL code selects the value according to the adornment style you select. See section "Select Adornment Styles" below.		
Adjust section level	• **C-=** • **C-c C-=** • **C-c C-a C-a**	(**rst-adjust** PFXARG)	Auto-adjust the adornment around point. • Adjust/rotate the section adornment for the section title around point or promote/ demote the adornments inside the region, depending on whether the region is active. This function is meant to be invoked possibly multiple times, and can vary its behavior with a positive PFXARG (toggle style), or with a negative PFXARG (alternate behavior). • This function is a bit of a swiss knife. It is meant to adjust the adornments of a section title in reStructuredText. It tries to deal with all the possible cases gracefully and to do "the right thing" in all cases.
Adorn line at title level	• **<f12> t** • **<f11> SPC r t**	(**pel-rst-adorn-title**)	Adorn current line with level-0 (title) reStructuredText section adornment. • If done at the top of the file, the first adorn line is placed on the first line of the file.
Adorn to specific level **From level 1 to level 10**	• **<f12> 1** • **<f11> SPC r 1** • **<f12> 2** • **<f11> SPC r 2** • **<f12> 3** • **<f11> SPC r 3** • **<f12> 4** • **<f11> SPC r 4** • **<f12> 5** • **<f11> SPC r 5** • **<f12> 6** • **<f11> SPC r 6** • **<f12> 7** • **<f11> SPC r 7** • **<f12> 8** • **<f11> SPC r 8** • **<f12> 9** • **<f11> SPC r 9** • **<f12> 0** • **<f11> SPC r 0**	• (**pel-rst-adorn-1**) • (**pel-rst-adorn-2**) • (**pel-rst-adorn-3**) • (**pel-rst-adorn-4**) • (**pel-rst-adorn-5**) • (**pel-rst-adorn-6**) • (**pel-rst-adorn-7**) • (**pel-rst-adorn-8**) • (**pel-rst-adorn-9**) • (**pel-rst-adorn-0**)	Adorn current line with level [1 to 10] reStructuredText section adornment. ☛The **<f11> SPC 1** to **<f11> SPC r 0** key sequences can be used inside any buffer. The **<f12>** keys can only be used in inside the buffers in rst-mode.
Adorn current line: same section level as previous section	• **<f12> =** • **<f11> SPC r =**	(**pel-rst-adorn-same-level**)	Adorn current line with the same level as the previous section. • If the line is already adorned, update the adornment: adjust to previous section level. •
Adorn to higher section level	• **<f12> +** • **<f11> SPC r +**	(**pel-rst-adorn-increase-level**)	Adorn current line at a higher-level that current if already adorned. • If the line is not already adorned, adorn it with a level higher than previous section.
Adorn to lower section level	• **<f12> -** • **<f11> SPC r -**	(**pel-rst-adorn-decrease-level**)	Adorn current line at a lower-level than current if already adorned. • If the line not already adorned, adorn it with a level lower than previous section.
Refresh current line adornment	• **<f12> r** • **<f11> SPC r r**	(**pel-rst-adorn-refresh**)	Refresh the adornment of the current line, adjusting the underlining to the current length of the line. This can be useful when changing the text on the line.
Select Adornment Styles	The underlying character used for section line adornment is customizable. The number of available levels and whether the line is indented, has a line over and under the title line is selected by the adornment style. PEL supports 3 styles. The following commands can be used to select a style.		
Select default adornment style	• **<f12> A d** • **<f11> SPC r A d**	(**pel-rst-adorn-default**)	Set the default section adornment style. This is Emacs rst-mode default: a title with 7 levels.

Description	Keystroke	Function	Note
Select Sphinx-Python adornment style	• **<f12> A S** • **<f11> SPC r A S**	(**pel-rst-adorn-Sphinx-Python**)	Set the Sphinx-Python section adornment style. This is what Sphinx supports: 6 levels: • parts, • chapters, • sections, • subsections, • subsubsections, • paragraphs.
Select CRiSPer adornment style	• **<f12> A C** • **<f11> SPC r A C**	(**pel-rst-adorn-CRiSPer**)	Set the CRiSPer section adornment style. A title level with another 12 levels. Use **<f12> +** to create those levels.
Creating and Using Hyperlinks	The following 3 PEL commands help write hyperlink of various forms: • the embedded form where the URL is stored inside the text between angle brackets and • the full named format where the link is located elsewhere in the file on its own line. When editing a buffer using the rat-mode , the **<f12> .** keystroke runs the command that creates a hyperlink, the long named format by default: it uses the region (if one is highlighted) or the word at point otherwise as the title for the link and creates the link entry on a line identified by a dedicated bookmark: that bookmark is created by the **<f12> s** keystroke. That helps identify an area inside the file where the next (or several) hyperlinks will be located. In PEL, the **<f12>** key prefix is mode sensitive. If you want to use the same commands inside another mode, you can use the longer key chord that uses the **<f11> SPC r** prefix. To activate it under PEL, you must set the PEL **pel-use-rst-mode** customization variable to **t**.		
Set location of hyperlinks	• **<f12> s** • **<f11> SPC r s**	(**pel-rst-set-ref-bookmark**)	• Set the reference bookmark for the currently edited file at point. • Used to identify the location where the next invocation of M-x pel-rst-mekelink inserts fully expanded links. • Ensures the bookmark is at the beginning of an empty line which is followed by another empty line, by inserting 2 lines and placing the point at the beginning of the first of the 2 lines.
Go to hyperlink location	• **<f12> g** • **<f11> SPC r g**	(**pel-rst-goto-ref-bookmark**)	Move point to the reference bookmark. • Useful to see where the bookmark for storing the hyperlink are currently located or add empty lines for future references. • Command pushes the mark on mark ring, type **M-`** to move back to previous location.

Add an hyperlink for text at point	• **<f12> .**
	• **<f11> SPC r .**
		(**pel-rst-makelink** &optional ARG)	Create a reStructuredText hyperlink prefix for the word at point or region's text.
			• If a region is active, use the text of the region to make the link, otherwise use the word at point.
			• If an argument (ARG, which can be a **C-u)** is specified, use the embedded URI format.
			• If no argument is specified, use the named hyperlink format:
			  • if the region is a single word, just append an underscore to make the link
			  • if the region is several words, surround the region with the "‘" start string and the "’_" end string.
			• The named link is placed in the location of bookmark named "RST" if it exists and points to same file, otherwise the link is placed at the beginning of the next empty line.
			• The cursor is placed where the URL is to be written.
			• Command pushes the mark on mark ring, type **M-`** to move back to previous location.

Open file or web-page whose name or markup link is at point
 	
★★
 	
(See also ⅀ Key-Chords,File-Mngt)
	• **C-^**
	• **<f11> f .**
	• **6y**
		(**pel-open-at-point** &optional N)	Open the file, library or the URL, named at point, with potential line & column #s.
			• If point is ***on a reStructuredText link in a rst-mode buffer***, open the link target (that might be a local file or a URL on remote web site. In the latter case the page is opened in the systems' browser).
			• If embedded space(s) are allowed in the filename, then point must be located at the first of the 2 delimiter characters. These delimiter character can be any of the following: **“`’|()[]{}<>’’“” 「」 〔〕 〈〉 《》 【】 〖〗 «»‹›❮❯〘〙 ·。**
			  • In the above list, the first 12 characters are ASCII characters, the remainders are Unicode characters: **’’“” 「」 〔〕 〈〉 《》 【】 〖〗 «»‹›❮❯〘〙 ·。**
			  • Tab and newline are also delimiter characters.
			• If embedded space in the file name is not allowed, then the file name must also be enclosed in the above delimiters, the space acts as an extra delimiter, and point can be positioned anywhere between the delimiters.
			• If the string identifies a URL, the function opens the page in the default browser.
			• Prompts for incomplete file names, allowing editing the find file (with completion), search for libraries files (type **l**) according to current file type.
			  Currently only supports Emacs Lisp files. Planning to support other programming languages with and without project management packages.
			• Without argument:
			  • If file is already opened in a window, move point to that window and to the line column coordinates if specified following the file name at point.
			  • If no window holds that file, select the target window based on the number of editable windows in frame: if 1, split that window and use the new window, if 2: use the other window, if 3 or more, use the current window.
			• With numeric argument N:
			  • N < 0 : create a new window and use that
			  • N = 0: use the '*other*' (the next) window
			  • N = 1,3,7or above (excluding 9):
			  select the target window based on the number of editable windows in frame: if 1, split that window and use the new window, if 2: use the other window, if 3 or more, use the current window.
			  • N is: 8: up, 2: down, 4:left, 5:current, 6:right.
			  • N is 9: open the file in the system's browser, and for a directory name at point open the application associated with directory browsing (eg. macOS Finder, Windows Explorer).
			• Selecting Minibuffer, inexistent or dedicated window is not allowed.
			• If the file name is followed by line and column numbers the point is moved to that position.
			More information available in the command's help docstring.
			With PEL, the **6y** key-chord is available if pel-use-key-chord is non-nil.
			Command prefixes are supported with the key-chord. See ⅀ Key-Chords.

Activating URLs (See also: ⅀ File Mngt)	Emacs provides the goto-url-mode and the goto-url-prog-mode that turn URLs found in the current buffer into clickable buttons. Once the mode is active use C-c RET or the mouse to click on the button. If the URL is an email address a buffer to write an email to that address opens. If the URL is an web or FTP address the system browser is invoked to open the address.		
Toggle goto-addr-mode	**<f11> f u**	(**goto-address-mode** &optional ARG)	Minor mode to buttonize URLs and e-mail addresses in the current buffer. With a prefix argument ARG, enable the mode if ARG is positive, and disable it otherwise.
Toggle goto-addr-prog-mode	**<f11> f U**	(**goto-address-prog-mode** &optional ARG)	Like 'goto-address-mode', but only for comments and strings.
Open the URL (email or web page)	**C-c RET**	(**goto-address-at-point** &optional EVENT)	Send to the e-mail address or load the URL at point. • Send mail to address at point: Find e-mail address around or before point. Then search backwards to beginning of line for the start of an e-mail address. • If no email address is found there, then load the URL at or before point.

Description	Keystroke	Function	Note
Editing Content	The following generic commands are useful when editing reStructuredText content.		
Fill current paragraph (See also: ⨜ Filling/ Justification)	• **M-q** • **<f11> t f p**	(**fill-paragraph** &optional JUSTIFY REGION)	To justify as well: **C-u M-q** • Notes: in refill mode this is done automatically. In auto fill mode the filling is done at the end of the line. • ☝Refill also properly refill a multi-line comment.
Align a set of lines on some text	**<f11> t w a**	(**align-regexp** BEG END REGEXP &optional GROUP SPACING REPEAT)	Align the current region using an ad-hoc rule read from the minibuffer. BEG and END mark the limits of the region. Interactively, this function prompts for the regular expression REGEXP to align with. • First select a region, then issue the command. For example, to align assignment of variables over the equal sign use **=** as the *regexp*. • The PEL package creates the **ar** alias for **align-regexp**, so it's also possible to invoke it with **M-x ar <RET>** ☛ Useful command to align the hyperlink references on their URL: select all hyperlink lines and then issue the command, specifying **http** as the regexp to line them all vertically.
Text Emphasis	The PEL commands emphasize the current word or marked region, then move point to the character right after the emphasized text.		
Bold	• **<f12> b** • **<f11> SPC r b**	(**pel-rst-bold**)	Mark current word or marked region bold. • Leave point after to the next character.
Italic	• **<f12> i** • **<f11> SPC r i**	(**pel-rst-italic**)	Mark current word or marked region italic. • Leave point after to the next character.
Literal	• **<f12> l** • **<f11> SPC r l**	(**pel-rst-literal**)	Mark current word or marked region with the literal markup. • Leave point after to the next character.
Interpreted	• **<f12> `** • **<f11> SPC r `**	(**pel-rst-interpreted**)	Mark current word or marked region with the interpreted markup. • Leave point after to the next character.
Tempo skeletons for reStructuredText (See also: ⨜ Inserting Text)	PEL provides support for flexible text template insertion through the Emacs built-in **tempo skeleton** mechanism. • PEL creates key bindings to invoke the skeletons in the supported major modes, using the same key prefix sequence for each mode: **<f12> <f12>**, with the same key bindings for equivalent concepts (such as file header block) as much as possible. ☝ (See also: ⨜ Inserting Text for more info and information about tempo skeleton and yasnippet template-based text insertion).		
Insert a file header	**<f12> <f12> C-h**	(**pel-rst-large-header**)	Insert a large header includes all normal header fields plus separators. • Prompts for title and insert title, automatically updated timestamp, attributes for home page and license, markup for table of contents using the tempo skeleton mechanism. • Automatically activates the PEL tempo skeleton mode so you can move to the target points where extra text must be entered to complete the template.
Toggle pel-tempo-mode	**<f12> <f12> 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 are only available when Emacs runs in graphics mode. ☝When a skeleton is inserted via the execution of one of the pel-rst-… commands, the pel-tempo-mode is automatically activated.
Jump to next tempo mark	• **C-c M-f** • **C-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. • These key key bindings are only available when pel-tempo-mode is active.
Jump to previous tempo mark	• **C-c M-b** • **C-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. • These key binding are only available when pel-tempo-mode is active.
Tempo Template Tag Insertion	**<f12> <f12> <f12>**	(**tempo-complete-tag** &optional SILENT)	Look for a tag and expand it. ☝Instead of using the **<f12><f12>** key bindings above, you can type the template name (shown in the title column like "if", "case", etc) completely or partially and then hit **<f12><f12><f12>**. A completion buffer opens up if the template name is incomplete (or empty in which case the buffer lists **all** available template names). Select the template name and hit RET. Emacs expands the template. • All the tags in the tag lists in 'tempo-local-tags' (this includes 'tempo-tags') are searched for a match for the text before the point. The way the string to match for is determined can be altered with the variable 'tempo-match-finder'. If 'tempo-match-finder' returns nil, then the results are the same as no match at all. • If a single match is found, the corresponding template is expanded in place of the matching string. • If a partial completion or no match at all is found, and SILENT is non-nil, the function will give a signal. • If a partial completion is found and 'tempo-show-completion-buffer' is non-nil, a buffer containing possible completions is displayed. ☛ Since only one template is available in rst-mode, the usefulness of this command is limited for reStructuredText.

rst-mode — References

Description & URL	Notes
Emacs Support for reStructuredText	
How to get the table of content with section numbers?	
reStructuredText	Main page for all reStructuredText documents.
reStructuredText markup Specifications	Formal markup specifications.
Sphinx Python Documentation Generator	
Sphinx — Documentation Contents	
Sphinx — Documentation —Sections	