








## rst-mode: reStructuredText Mode

Description	Keystroke	Function	Note
<div> <div>Editing reStructuredText files</div> <div>Manual: Emacs Support for reStructuredText</div> </div>	<div> <div> <div> <div>📖</div> <div>There's more information related to reStructuredText that need to be documented.</div> </div> <div> <div>📄</div> <div>The reStructuredText files are supported by the ret-mode which is available in standard Emacs distribution.</div> </div> <div> <div>🔗</div> <div>To activate it under PEL, you must set the PEL <b>pel-use-rst-mode</b> customization variable to <b>t</b>.</div> </div> <div> <div>🔧</div> <div> <div><b>pel-rst-tab-width</b>: The width of a tab used for reStructuredText files. Defaults to 2.</div> <ul style="list-style-type: none"> <li>This concept differs from indentation: you can have an indentation of 3 and tab width of 8: <b>M-i</b> will move point to columns that are multiple of 8 <b>&lt;tab&gt;</b> will indent to a column that is a multiple of 3. PEL stores this value inside the <b>tab-width</b> user option variable for rst-mode buffers. See <a href="#">🔗 Indentation</a>.</li> </ul> </div> </div> </div> </div>		
<div> <div>Open this PDF file.</div> <div>See also: <a href="#">🔗 Help/Info</a></div> </div>	<div> <div>&lt;f11&gt; SPC M-r &lt;f1&gt;</div> <div>&lt;f12&gt; &lt;f1&gt;</div> </div>	<div> <div>(pel-help-pdf &amp;optional OPEN-WEB-PAGE)</div> </div>	<div> <div>Open the <a href="#">🔗 reStructuredText</a> PDF using method specified by the <b>pel-open-pdf-method</b> user-option or the alternate one if a command prefix (like <b>C-u</b>) was used.</div> </div>
<div> <div><a href="#">🔗 Customize</a> PEL reStructuredText support</div> </div>	<div> <div>&lt;f11&gt; SPC M-r &lt;f2&gt;</div> <div>&lt;f12&gt; &lt;f2&gt;</div> </div>	<div> <div>(pel-customize-pel &amp;optional OTHER-WINDOW)</div> </div>	<div> <div>Customize PEL reStructuredText support.</div> <ul style="list-style-type: none"> <li>If OTHER-WINDOW is non-nil (use <b>C-u</b>), display in another window.</li> </ul> </div>
<div> <div><a href="#">🔗 Customize</a> Emacs reStructuredText support</div> </div>	<div> <div>&lt;f11&gt; SPC M-r &lt;f3&gt;</div> <div>&lt;f12&gt; &lt;f3&gt;</div> </div>	<div> <div>(pel-customize-library &amp;optional OTHER-WINDOW)</div> </div>	<div> <div>Customize Emacs reStructuredText support.</div> <ul style="list-style-type: none"> <li>If OTHER-WINDOW is non-nil (use <b>C-u</b>), display in another window.</li> </ul> </div>
<div> <div>Activate reStructuredText mode</div> </div>	<div> <div>M-x rst-mode</div> </div>	<div> <div>(rst-mode)</div> </div>	<div> <div>Toggle the rst-mode used to edit reStructuredText markup.</div> </div>
<div> <div>Get version of rst-mode</div> </div>	<div> <div>C-h v rst-version</div> </div>		<div> <div>Shows the content of the variable rst-version.</div> <ul style="list-style-type: none"> <li>Works once the rst-mode is loaded only.</li> </ul> </div>
<div> <div>Display table of content</div> </div>	<div> <div>C-c C-t C-t</div> </div>	<div> <div>(rst-doc)</div> </div>	<div> <div>Display a table of contents for current buffer inside another buffer.</div> <ul style="list-style-type: none"> <li>Displays all section titles found in the current buffer in a hierarchical list.</li> <li>The resulting buffer can be navigated, and selecting a section title moves the cursor to that section.</li> </ul> </div>
<div> <div>Indent list item</div> <div>(See <a href="#">🔗 Indentation</a>)</div> </div>	<div> <div>&lt;tab&gt;</div> </div>	<div> <div>(indent-for-tab-command &amp;optional ARG)</div> </div>	<div> <div>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.</div> </div>
<div> <div>Comment</div> <div>See also: <a href="#">🔗 Comments</a></div> </div>	<div> <div>M-;</div> </div>	<div> <div>(comment-dwim ARG)</div> </div>	<div> <div>Comment line or region.</div> <div><b>TODO:</b>🔧 the <b>uncommenting</b> does not work. According to the comment-dwim description it should. Need to investigate.</div> </div>
<div> <div>Move to previous section title</div> </div>	<div> <div> <ul style="list-style-type: none"> <li>C-M-a</li> <li>&lt;f12&gt; p</li> <li>&lt;f12&gt; &lt;up&gt;</li> </ul> </div> <div> <ul style="list-style-type: none"> <li>&lt;f11&gt; SPC M-r p</li> <li>&lt;f11&gt; SPC M-r &lt;up&gt;</li> </ul> </div> </div>	<div> <div>(rst-backward-section OFFSET)</div> </div>	<div> <div>Jump backward OFFSET section titles ending up at the start of the title line.</div> <ul style="list-style-type: none"> <li>OFFSET defaults to 1 and may be negative to move backward.</li> <li>An OFFSET of 0 does not move unless point is inside a title.</li> <li>Go to end or beginning of buffer if no more section titles in the desired direction.</li> </ul> </div>
<div> <div>Move to next section title</div> </div>	<div> <div> <ul style="list-style-type: none"> <li>C-M-e</li> <li>&lt;f12&gt; n</li> <li>&lt;f12&gt; &lt;down&gt;</li> </ul> </div> <div> <ul style="list-style-type: none"> <li>&lt;f11&gt; SPC M-r n</li> <li>&lt;f11&gt; SPC M-r &lt;down&gt;</li> </ul> </div> </div>	<div> <div>(rst-forward-section OFFSET)</div> </div>	<div> <div>Jump forward OFFSET section titles ending up at the start of the title line.</div> <ul style="list-style-type: none"> <li>OFFSET defaults to 1 and may be negative to move backward.</li> <li>An OFFSET of 0 does not move unless point is inside a title.</li> <li>Go to end or beginning of buffer if no more section titles in the desired direction.</li> </ul> </div>
<div> <div>Mark complete current section</div> </div>	<div> <div>C-M-h</div> </div>	<div> <div>(rst-mark-section &amp;optional COUNT ALLOW-EXTEND)</div> </div>	<div> <div>Select COUNT sections around point.</div> <ul style="list-style-type: none"> <li>Mark following sections for positive COUNT or preceding sections for negative COUNT.</li> </ul> </div>
<div> <div>Section level adornment</div> </div>	<div> <div>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.</div> <div>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.</div> <div>PEL provides 3 style of section adornments: default, Sphinx-Python and CRiSPer, which can be selected with commands.</div> <div>🔗 PEL remembers the preferred style inside the customizable variable: <b>pel-rst-adornment-style</b>.</div> <div>🔧 The rest.el provides the <b>rst-preferred-adornment</b> 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.</div> </div>		
<div> <div>Adjust section level</div> </div>	<div> <div> <ul style="list-style-type: none"> <li>C-=</li> <li>C-c C-=</li> <li>C-c C-a C-a</li> </ul> </div> </div>	<div> <div>(rst-adjust PFXARG)</div> </div>	<div> <div>Auto-adjust the adornment around point.</div> <ul style="list-style-type: none"> <li>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).</li> <li>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.</li> </ul> </div>
<div> <div>Adorn line at title level</div> </div>	<div> <div>&lt;f12&gt; t</div> <div>&lt;f11&gt; SPC M-r t</div> </div>	<div> <div>(pel-rst-adorn-title)</div> </div>	<div> <div>Adorn current line with level-0 (title) reStructuredText section adornment.</div> <ul style="list-style-type: none"> <li>If done at the top of the file, the first adorn line is placed on the first line of the file, a mark is left at the end of the title line and point is moved 2 lines below. <ul style="list-style-type: none"> <li>To return to the end of the title line, type <b>M-`</b>.</li> </ul> </li> </ul> </div>
<div> <div>Adorn to specific level</div> <div>From level 1 to level 10</div> </div>	<div> <div> <ul style="list-style-type: none"> <li>&lt;f12&gt; 1</li> <li>&lt;f12&gt; 2</li> <li>&lt;f12&gt; 3</li> <li>&lt;f12&gt; 4</li> <li>&lt;f12&gt; 5</li> <li>&lt;f12&gt; 6</li> <li>&lt;f12&gt; 7</li> <li>&lt;f12&gt; 8</li> <li>&lt;f12&gt; 9</li> <li>&lt;f12&gt; 0</li> </ul> </div> <div> <ul style="list-style-type: none"> <li>&lt;f11&gt; SPC M-r 1</li> <li>&lt;f11&gt; SPC M-r 2</li> <li>&lt;f11&gt; SPC M-r 3</li> <li>&lt;f11&gt; SPC M-r 4</li> <li>&lt;f11&gt; SPC M-r 5</li> <li>&lt;f11&gt; SPC M-r 6</li> <li>&lt;f11&gt; SPC M-r 7</li> <li>&lt;f11&gt; SPC M-r 8</li> <li>&lt;f11&gt; SPC M-r 9</li> <li>&lt;f11&gt; SPC M-r 0</li> </ul> </div> </div>	<div> <div> <ul style="list-style-type: none"> <li>(pel-rst-adorn-1)</li> <li>(pel-rst-adorn-2)</li> <li>(pel-rst-adorn-3)</li> <li>(pel-rst-adorn-4)</li> <li>(pel-rst-adorn-5)</li> <li>(pel-rst-adorn-6)</li> <li>(pel-rst-adorn-7)</li> <li>(pel-rst-adorn-8)</li> <li>(pel-rst-adorn-9)</li> <li>(pel-rst-adorn-0)</li> </ul> </div> </div>	<div> <div>Adorn current line with level [1 to 10] reStructuredText section adornment.</div> <div> <div>🔗</div> <div>The <b>&lt;f11&gt; SPC M-r 1</b> to <b>&lt;f11&gt; SPC M-r 0</b> key sequences can be used inside any buffer. The <b>&lt;f12&gt;</b> keys can only be used in inside the buffers in rst-mode.</div> </div> </div>



Description	Keystroke	Function	Note
<b>Activating URLs to browse and open files</b>  See also: • <a href="#">🔗 File mngt</a> • <a href="#">🔗 Navigation</a>	Emacs provides the <b>goto-url-mode</b> and the <b>goto-url-prog-mode</b> that turn URLs found in the current buffer into clickable buttons. • Once the mode is active the following key sequences are available wheel point is over a URL button: <ul style="list-style-type: none"> <li>• <b>C-c RET</b> or the mouse to click on the button.               <ul style="list-style-type: none"> <li>• If the URL is an email address a buffer to write an email to that address opens.</li> <li>• If the URL is a web or FTP address the system browser is invoked to open the address.</li> </ul> </li> <li>• <b>C-c C-n</b> : move point to the end of the next URL in the buffer.</li> <li>• <b>C-c C-p</b> : move point to to the previous URL in the buffer.</li> <li>• <b>C-c C-f</b> : download the file identified by the URL into a local temporary file and visit the file. See (pel-open-url-at-point) above.</li> </ul>  Customization group: <b>goto-address</b> . Mostly control the regex for URL and the face used.		
Toggle goto-address-mode	<b>&lt;f11&gt; f u</b>	(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-address-prog-mode	<b>&lt;f11&gt; f U</b>	(goto-address-prog-mode &optional ARG)	Like 'goto-address-mode', but only for comments and strings.
Open the URL (email or web page)	<b>C-c RET</b>	(goto-address-at-point &optional EVENT)	Open the URL at point: <ul style="list-style-type: none"> <li>• If URL is a web page: open it in a browser</li> <li>• If URL is a mail address:               <ul style="list-style-type: none"> <li>• Send mail to address at point:                   <ul style="list-style-type: none"> <li>Find e-mail address around or before point. Then search backwards to beginning of line for the start of an e-mail address.</li> </ul> </li> </ul> </li> <li>• If no email address is found there, then load the URL at or before point.</li> </ul>
Move to end of next URL in buffer See also: <a href="#">🔗 Navigation</a>	<b>C-c C-n</b> <b>&lt;f6&gt; C-n</b>	(pel-goto-next-url)	Move point forward to the end of the next URL located in the current buffer. <ul style="list-style-type: none"> <li>• The global <b>&lt;f6&gt; C-n</b> key binding activates the goto-address-mode if it is not already active.</li> </ul>
Move to beginning of previous URL in buffer See also: <a href="#">🔗 Navigation</a>	<b>C-c C-p</b> <b>&lt;f11&gt; C-p</b>	(pel-goto-previous-url)	Move point backward to the beginning of the previous URL located in the current buffer. <ul style="list-style-type: none"> <li>• The global <b>&lt;f6&gt; C-p</b> key binding activates the goto-address-mode if it is not already active.</li> </ul>
Copy URL at point in temporary file and visit the file  See also: <a href="#">🔗 File mngt</a>	<b>&lt;f11&gt; f M-u</b>  <b>C-c C-f</b>	(pel-open-url-at-point)	Copy the URL at point to a local temporary file and visit that file. <ul style="list-style-type: none"> <li>•  The download copy of the file does not have the same name and may not open with the proper mode because it won't have an extension. The HTML formatted files will be recognized by Emacs but most of the files won't be.</li> <li>• Save the file somewhere else using the <b>C-x C-w</b> key sequence and identify the proper extension to activate the required major mode.</li> </ul>  This binding is only available when point is over the URL and the <b>goto-address-mode</b> minor mode is active. Use <b>&lt;f11&gt; f u</b> or <b>&lt;f11&gt; f U</b> to activate this mode.
Editing Content	The following generic commands are useful when editing reStructuredText content.		
Fill current paragraph See also: <a href="#">🔗 Filling/Justification</a>	<ul style="list-style-type: none"> <li>• <b>M-q</b></li> <li>• <b>&lt;f11&gt; t f p</b></li> </ul>	(fill-paragraph &optional JUSTIFY REGION)	To justify as well: <b>C-u M-q</b> <ul style="list-style-type: none"> <li>• Notes: in refill mode this is done automatically. In auto fill mode the filling is done at the end of the line.</li> <li>•  Refill also properly refill a multi-line comment.</li> </ul>
Align a set of lines on some text	<b>&lt;f11&gt; t w a</b>	(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. <ul style="list-style-type: none"> <li>• First select a region, then issue the command. For example, to align assignment of variables over the equal sign use = as the <i>regexp</i>.</li> <li>• The PEL package creates the <b>ar</b> alias for <b>align-regexp</b>, so it's also possible to invoke it with <b>M-x ar RET</b></li> </ul>  Useful command to align the hyperlink references on their URL: select all hyperlink lines and then issue the command, specifying <b>http</b> 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	<b>&lt;f12&gt; b</b> <b>&lt;f11&gt; SPC M-r b</b>	(pel-rst-bold)	Mark current word or marked region bold. <ul style="list-style-type: none"> <li>• Leave point after to the next character.</li> </ul>
Italic	<b>&lt;f12&gt; i</b> <b>&lt;f11&gt; SPC M-r i</b>	(pel-rst-italic)	Mark current word or marked region italic. <ul style="list-style-type: none"> <li>• Leave point after to the next character.</li> </ul>
Literal	<b>&lt;f12&gt; l</b> <b>&lt;f11&gt; SPC M-r l</b>	(pel-rst-literal)	Mark current word or marked region with the literal markup. <ul style="list-style-type: none"> <li>• Leave point after to the next character.</li> </ul>
Interpreted	<b>&lt;f12&gt; `</b> <b>&lt;f11&gt; SPC M-r `</b>	(pel-rst-interpreted)	Mark current word or marked region with the interpreted markup. <ul style="list-style-type: none"> <li>• Leave point after to the next character.</li> </ul>
<b>Tempo skeletons for reStructuredText</b> See also: <a href="#">🔗 Inserting Text</a>	PEL provides support for flexible text template insertion through the Emacs built-in <b>tempo skeleton</b> mechanism. <ul style="list-style-type: none"> <li>• PEL creates key bindings to invoke the skeletons in the supported major modes, using the same key prefix sequence for each mode: <b>&lt;f12&gt; &lt;f12&gt;</b>, with the same key bindings for equivalent concepts (such as file header block) as much as possible.</li> </ul>  See also: <a href="#">🔗 Inserting Text</a> for more info and information about tempo skeleton and yasnippet template-based text insertion).		
Insert a file header	<b>&lt;f12&gt; &lt;f12&gt; h</b>	(pel-rst-large-header)	Insert a large header includes all normal header fields plus separators. <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> </ul>
Toggle pel-tempo-mode	<b>&lt;f12&gt; &lt;f12&gt; SPC</b>	(pel-tempo-mode &optional ARG)	Toggle PEL tempo mode on/off. PEL tempo mode activates <b>C-c .</b> and <b>C-c ,</b> , as well as to <b>C-c C-.</b> and <b>C-c C-,</b> 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	<ul style="list-style-type: none"> <li>• <b>C-c M-f</b></li> <li>• <b>C-c .</b></li> <li>• <b>C-c C-.</b></li> </ul>	(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"> <li>• These key key bindings are only available when pel-tempo-mode is active.</li> </ul>
Jump to previous tempo mark	<ul style="list-style-type: none"> <li>• <b>C-c M-b</b></li> <li>• <b>C-c ,</b></li> <li>• <b>C-c C-,</b></li> </ul>	(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"> <li>• These key binding are only available when pel-tempo-mode is active.</li> </ul>

Description	Keystroke	Function	Note
Tempo Template Tag Insertion	<f12> <f12> <f12>	(tempo-complete-tag &optional SILENT)	<p>Look for a tag and expand it.</p> <p>👉 Instead of using the &lt;f12&gt; &lt;f12&gt; key bindings above, you can type the template name (shown in the title column like “if”, “case”, etc) completely or partially and then hit &lt;f12&gt; &lt;f12&gt; &lt;f12&gt;. A completion buffer opens up if the template name is incomplete (or empty in which case the buffer lists <b>all</b> available template names). Select the template name and hit RET. Emacs expands the template.</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• If a single match is found, the corresponding template is expanded in place of the matching string.</li> <li>• If a partial completion or no match at all is found, and SILENT is non-nil, the function will give a signal.</li> <li>• If a partial completion is found and ‘tempo-show-completion-buffer’ is non-nil, a buffer containing possible completions is displayed.</li> </ul> <p>🚩 Since only one template is available in rst-mode, the usefulness of this command is limited for reStructuredText.</p>

### 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	