

















## rst-mode: reStructuredText Mode

Description	Keystroke	Function	Note
<b>reStructuredText</b> <ul style="list-style-type: none"> <li><a href="#">Emacs Support for reStructuredText</a></li> <li><a href="#">Basic Intro to rst</a></li> <li><a href="#">reStructuredText markup</a></li> <li><a href="#">reStructuredText Directives</a></li> <li><a href="#">Quick reference to rst</a></li> <li><a href="#">rst-cheatsheet (pdf)</a></li> <li><a href="#">Sphinx &amp; rst syntax guide</a></li> </ul> <p>See also: <a href="#">⌘ Speedbar</a></p>	<p>This page describes Emacs support for reStructuredText (abbreviated sometimes as ‘rst’ and sometimes as ‘reST’) .</p> <ul style="list-style-type: none"> <li>The reStructuredText files are supported by Emacs <code>rst-mode</code> from <code>rst.el</code> which is available in standard Emacs distribution.</li> <li><b>Supported file extensions:</b> <code>.rst</code>, <code>.rest</code>, <code>.stxt</code> and <code>.rst.txt</code>. The <code>.rst.txt</code> extension allows rendering by tools supporting <code>.txt</code> files.</li> </ul> <p> To activate it under PEL, you must set the PEL <b>pel-use-rst-mode</b> customization variable to <code>t</code>.</p> <p> <b>pel-rst-tab-width:</b> The width of a tab used for reStructuredText files. Defaults to 2.</p> <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> <p> <b>pel-rst-use-tabs:</b> whether hard tabs are used for indentation. Defaults to nil (use space characters for all indentation).</p> <p><a href="#">⌘ Speedbar Support:</a></p> <ul style="list-style-type: none"> <li>PEL activates <a href="#">⌘ Speedbar</a> support for reStructuredText when the <b>pel-use-speedbar</b> user-option is turned on (set to <code>t</code>). Use the Speedbar to see the sections of the reStructuredText document and navigate to them.</li> </ul>		
<b>Open this PDF file.</b> See also: <a href="#">⌘ Help/Info</a>	<div>&lt;f11&gt; SPC M-r &lt;f1&gt;</div> <div>&lt;f12&gt; &lt;f1&gt;</div>	<div>(pel-help-pdf &amp;optional OPEN-WEB-PAGE)</div>	Open the <code>M</code> <b>reStructuredText</b> local PDF. If the prefix argument (like <b>C-u</b> or <b>M--</b> ) is used, then it opens the remote GitHub hosted raw PDF instead. If the <b>pel-flip-help-pdf-arg</b> user-option is set it’s the other way around.
<a href="#">⌘ Customize</a> PEL reStructuredText support	<div>&lt;f11&gt; SPC M-r &lt;f2&gt;</div> <div>&lt;f12&gt; &lt;f2&gt;</div>	<div>(pel-customize-pel &amp;optional OTHER-WINDOW)</div>	Customize PEL reStructuredText support. <ul style="list-style-type: none"> <li>If OTHER-WINDOW is non-nil (use <b>C-u</b>), display in another window.</li> </ul>
<a href="#">⌘ Customize</a> Emacs reStructuredText support	<div>&lt;f11&gt; SPC M-r &lt;f3&gt;</div> <div>&lt;f12&gt; &lt;f3&gt;</div>	<div>(pel-customize-library &amp;optional OTHER-WINDOW)</div>	Customize Emacs reStructuredText support. <ul style="list-style-type: none"> <li>If OTHER-WINDOW is non-nil (use <b>C-u</b>), display in another window.</li> </ul>
<b><a href="#">rst-mode</a></b>	Emacs provides the <code>rst-mode</code> from the <code>rst.el</code> file. The following file extensions are associated with this major mode: <code>.rst</code> , <code>.rest</code> . PEL adds the <code>.stxt</code> extension.		
<b>Activate reStructuredText mode</b>	<code>M-x rst-mode</code>	<b>(rst-mode)</b>	Toggle the rst-mode used to edit reStructuredText markup. <ul style="list-style-type: none"> <li>Automatically invoked when visiting <code>.rst</code>, <code>.rest</code> files (and <code>.stxt</code> files with PEL).</li> </ul>
<b>Get version of rst-mode</b>	<code>C-h v rst-version</code>		Shows the content of the variable <code>rst-version</code> . <ul style="list-style-type: none"> <li>Only works once the rst-mode is loaded.</li> </ul>
<b>Syntax Control</b> <ul style="list-style-type: none"> <li><a href="#">superword-mode</a></li> </ul>	reStructuredText files often benefit from the <a href="#">superword-mode</a> and treating underscores as part of words. This helps searching for symbols and opening file that have names that include underscore characters. <p>  Force activation of <a href="#">superword-mode</a> for reStructuredText by adding the <a href="#">superword-mode</a> to the <b>pel-rst-activates-minor-modes</b> user-option.</p>		
<b>Control syntax of the underscore character</b> <ul style="list-style-type: none"> <li>Requires superword-mode on.</li> </ul>	<div>&lt;f12&gt; _</div>	<div>(pel-rst-set-underscore-syntax &amp;optional ACTION)</div>	Control syntax of underscore to punctuation or symbol when the superword-mode is active. <ul style="list-style-type: none"> <li>If superword-mode is off the function issues a user error. Otherwise the function operates according to the value of the optional argument:                             <ul style="list-style-type: none"> <li>If the argument is not specified or nil, the function toggles the syntax of the underscore character between punctuation (the default) and symbol.</li> <li>If the argument is positive, it sets the syntax of underscore to symbol.</li> <li>If the argument is 0 or negative it sets the syntax of the underscore back to punctuation.</li> </ul> </li> </ul> <p>  To force use of symbol syntax, add <b>pel-rst-set-underscore-syntax</b> to the <b>pel-rst-activates-minor-modes</b> user-option <b>after</b> superword-mode.</p>
<b>Editing Content</b>	The following generic commands are useful when editing reStructuredText content.		
<ul style="list-style-type: none"> <li><b>Text filling</b></li> </ul> See also: <ul style="list-style-type: none"> <li><a href="#">⌘ Filling/Justification</a></li> </ul>	Although text filling will be handled for the generated rendering, you may decide to fill the reStructuredText file itself, after all you’re using a markup that’s made to allow reading the original text. <ul style="list-style-type: none"> <li>You can turn the auto fill mode on and identify the fill column.</li> </ul> <p>  Force the auto-fill-mode when a reStructuredText file is visited by adding the auto-fill-mode to the <b>pel-rst-activates-minor-modes</b> user-option.</p>		
<b>Toggle auto-fill mode</b>	<ul style="list-style-type: none"> <li>&lt;f11&gt; t f a</li> <li>&lt;f11&gt; RET</li> </ul>	(auto-fill-mode &optional ARG)	Toggle automatic line breaking (Auto Fill mode). <ul style="list-style-type: none"> <li>With a prefix argument, enable Auto Fill mode if the prefix argument is positive, and disable it otherwise.</li> <li>When Auto Fill mode is enabled, inserting a space at a column beyond ‘current-fill-column’ automatically breaks the line at a previous space.</li> </ul>
<b>Set Fill Column</b>	<ul style="list-style-type: none"> <li>C-x f</li> <li>&lt;f11&gt; t f c</li> </ul>	(set-fill-column ARG)	When no prefix value: prompts for column unless a prefix argument was used. <ul style="list-style-type: none"> <li>If with <b>C-u</b> prefix: use current column.</li> <li>If with prefix value: use that value.</li> </ul>
<b>Fill current paragraph</b>	<ul style="list-style-type: none"> <li>M-q</li> <li>&lt;f11&gt; t f p</li> </ul>	(fill-paragraph &optional JUSTIFY REGION)	To justify as well: <b>C-u M-q</b> <ul style="list-style-type: none"> <li>In refill mode this is done automatically. In auto fill mode the filling is done at the end of the line.</li> </ul>
<b>Align a set of lines on some text</b>	<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.
	<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 <code>=</code> 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> <p> Use it to align hyperlink references URL: select all hyperlink lines and then issue the command, specifying <b>http</b> as the regexp to line them vertically.</p>		
<b><a href="#">Text Emphasis</a></b>	The PEL commands emphasize the current word or marked region, then move point to the character right after the emphasized text or inside if empty.		
<b>Bold</b>	<f12> 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> <li>Inserts required escaped spaces when the emphasized region is inside a word.</li> </ul>
	<f11> SPC M-r b		
<b>Italic</b>	<f12> i	(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> <li>Inserts required escaped spaces when the emphasized region is inside a word.</li> </ul>
	<f11> SPC M-r i		
<b>Literal</b>	<f12> l	(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> <li>Inserts required escaped spaces when the emphasized region is inside a word.</li> </ul>
	<f11> SPC M-r l		
<b>Interpreted</b>	<f12> `	(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> <li>Inserts required escaped spaces when the emphasized region is inside a word.</li> </ul>
	<f11> SPC M-r `		
<b>Indent list item</b> See also: <a href="#">⌘ Indentation</a>	<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.
<b>Comment</b> See also: <a href="#">⌘ Comments</a>	<code>M-;</code>	(comment-dwim ARG)	Comment line or region. <p> <b>Uncommenting does not work.</b></p>
<b>Itemize all previous lines same indentation level</b>	<ul style="list-style-type: none"> <li>&lt;f12&gt; M--</li> <li>M-&lt;f12&gt; M--</li> </ul>	(pel-itemize-lines &optional ITEM-PREFIX-STRING)	Prepend each of the previous lines with a ITEM-PREFIX-STRING that is “- “ by default. <p> When writing a list of items, instead of manually typing the “- “ prefix on each line, type each line without them and then use this command to itemize each of the lines above the current one.</p> <ul style="list-style-type: none"> <li>It indents all lines above the current line that are at the same indentation level as the current position.</li> </ul>

Description	Keystroke	Function	Note
<b>File's Table of Content</b>	<ul style="list-style-type: none"><li>Use the <b>contents markup directive</b> to have reStructuredText tools automatically generate a table of contents for your file.</li><li>You can also insert an explicit table of content with the rst-toc-insert command.</li><li>There are several ways to view the files sections:<ul style="list-style-type: none"><li>with <b>C-c C-t C-t</b> to invoke the rst-doc command: it opens a "table of Content" buffer, moves point inside it, move to the section title, hit <b>RET</b> to select that section inside the original reStructuredText buffer.</li><li>using the Speedbar to open a buffer that lists the sections. See <a href="#">☞ Speedbar</a>.</li></ul></li></ul>		
See also: <a href="#">☞ Speedbar</a>			
Insert a table content at point	<b>C-c C-t TAB</b>	(rst-toc-insert &optional MAX-LEVEL)	Insert the table of contents of the current section at the current column. <ul style="list-style-type: none"><li>By default the top level is ignored if there is only one, because we assume that the document will have a single title.</li><li>A numeric prefix argument MAX-LEVEL overrides 'rst-toc-insert-max-level'.</li><li>Text in the line beyond column is deleted.</li></ul> 👉 You may want to use the <b>contents markup directive</b> instead.
Display table of content	<b>C-c C-t C-t</b>	(rst-doc)	Display a table of contents for current buffer inside the "Table of Contents" buffer. <ul style="list-style-type: none"><li>Displays all section titles found in the current buffer in a hierarchical list.</li></ul>
<ul style="list-style-type: none"><li>Navigate to specific section</li></ul>			<ul style="list-style-type: none"><li>Select the section of interest in the "Table of Contents" buffer by navigating to it, then hit <b>RET</b> on that section to move back to the section in the original reStructuredTex document and close the Table of Contents" buffer window.</li></ul>
Moving across sections	You can also use the following commands to move to the next or previous section.		
Move to previous section title	<ul style="list-style-type: none"><li><b>C-M-a</b></li><li><b>&lt;f12&gt; p</b></li><li><b>&lt;f12&gt; &lt;up&gt;</b></li></ul>	(rst-backward-section OFFSET)	Jump backward OFFSET section titles ending up at the start of the title line. <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>
	<ul style="list-style-type: none"><li><b>&lt;f11&gt; SPC M-r p</b></li><li><b>&lt;f11&gt; SPC M-r &lt;up&gt;</b></li></ul>		
Move to next section title	<ul style="list-style-type: none"><li><b>C-M-e</b></li><li><b>&lt;f12&gt; n</b></li><li><b>&lt;f12&gt; &lt;down&gt;</b></li></ul>	(rst-forward-section OFFSET)	Jump forward OFFSET section titles ending up at the start of the title line. <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>
	<ul style="list-style-type: none"><li><b>&lt;f11&gt; SPC M-r n</b></li><li><b>&lt;f11&gt; SPC M-r &lt;down&gt;</b></li></ul>		
Mark complete current section	<b>C-M-h</b>	(rst-mark-section &optional COUNT ALLOW-EXTEND)	Select COUNT sections around point. <ul style="list-style-type: none"><li>Mark following sections for positive COUNT or preceding sections for negative COUNT.</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>
Tempo Template Tag Insertion	<b>&lt;f12&gt; &lt;f12&gt; &lt;f12&gt;</b>	(tempo-complete-tag &optional SILENT)	Look for a tag and expand it.
	👉 Instead of using the <b>&lt;f12&gt; &lt;f12&gt;</b> key bindings above, you can type the template name (shown in the title column like "if", "case", etc) completely or partially and then hit <b>&lt;f12&gt; &lt;f12&gt; &lt;f12&gt;</b> . 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. <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> ➡ Since only one template is available in rst-mode, the usefulness of this command is limited for reStructuredText.		
<b>Select Section Title Adornment Styles</b>	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	<b>&lt;f12&gt; A d</b>	(pel-rst-adorn-default)	Set the default section adornment style. This is Emacs rst-mode default: a title with 7 levels.
	<b>&lt;f11&gt; SPC M-r A d</b>		
Select Sphinx-Python adornment style	<b>&lt;f12&gt; A S</b>	(pel-rst-adorn-Sphinx-Python)	Set the Sphinx-Python section adornment style. This is what Sphinx supports: 6 levels: <ul style="list-style-type: none"><li>parts,</li><li>chapters,</li><li>sections,</li><li>subsections,</li><li>subsubsections,</li><li>paragraphs.</li></ul>
	<b>&lt;f11&gt; SPC M-r A S</b>		
Select CRISPer adornment style	<b>&lt;f12&gt; A C</b>	(pel-rst-adorn-CRISPer)	Set the CRISPer section adornment style. A title level with another 12 levels. Use <b>&lt;f12&gt; +</b> to create those levels.
	<b>&lt;f11&gt; SPC M-r A C</b>		

Description	Keystroke	Function	Note
<b>Section Title level adornment</b> <ul style="list-style-type: none"> <li>commands that insert section titles</li> </ul>	<p>The rst.el library provides the <b>rst-adjust</b> command to create section adornment of the current line.</p> <ul style="list-style-type: none"> <li>This command tries to infer the level required and unfortunately sometimes fails when market is used and not expected by its code.</li> <li>PEL provides a set of very simple commands that use multiple key bindings to adorn the current line to a fixed section level: <ul style="list-style-type: none"> <li>title level and up to 10 other levels, from 1 to 9 and then 0 for 10.</li> <li>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.</li> </ul> </li> <li>PEL provides 3 style of section adornments: default, Sphinx-Python and CRiSPer, which can be selected with commands.</li> <li>PEL remembers the preferred style inside the customizable variable: <b>pel-rst-adornment-style</b>.</li> <li> The rest.el provides the <b>rst-preferred-adornment</b> user option to select the adornment characters for the various sections. <ul style="list-style-type: none"> <li>PEL code selects the value according to the adornment style you select.</li> <li>See section “Select Adornment Styles” above.</li> </ul> </li> </ul>		
Adjust section level	<ul style="list-style-type: none"> <li><b>C--</b></li> <li><b>C-c C--</b></li> <li><b>C-c C-a C-a</b></li> </ul>	(rst-adjust PFXARG)	<p>Auto-adjust the adornment around point.</p> <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>
Adorn line at title level	<f12> t <f11> SPC M-r t	(pel-rst-adorn-title)	<p>Adorn current line with level-0 (title) reStructuredText section adornment.</p> <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> or <b>&lt;f6&gt;&lt;f6&gt;</b>.</li> </ul> </li> </ul>
Adorn to specific level From level 1 to level 10	<ul style="list-style-type: none"> <li>&lt;f12&gt; 1 ...</li> <li>&lt;f12&gt; 9</li> <li>&lt;f12&gt; 0</li> <li>&lt;f11&gt; SPC M-r 1 ...</li> <li>&lt;f11&gt; SPC M-r 9</li> <li>&lt;f11&gt; SPC M-r 0</li> </ul>	<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>	<p>Adorn current line with level [1 to 10] reStructuredText section adornment.</p> <p>➡The &lt;f11&gt; SPC M-r 1 to &lt;f11&gt; SPC M-r 0 key sequences can be used inside any buffer. The &lt;f12&gt; keys can only be used in inside the buffers in rst-mode.</p>
Adorn current line: same section level as previous section	<ul style="list-style-type: none"> <li>&lt;f12&gt; =</li> <li>&lt;f11&gt; SPC M-r =</li> </ul>	(pel-rst-adorn-same-level)	<p>Adorn current line with the same level as the previous section.</p> <ul style="list-style-type: none"> <li>If the line is already adorned, update the adornment: adjust to previous section level.</li> </ul>
Adorn to higher section level	<f12> + <f11> SPC M-r +	(pel-rst-adorn-increase-level)	<p>Adorn current line at a higher-level that current if already adorned.</p> <ul style="list-style-type: none"> <li>If the line is not already adorned, adorn it with a level higher than previous section.</li> </ul>
Adorn to lower section level	<f12> - <f11> SPC M-r -	(pel-rst-adorn-decrease-level)	<p>Adorn current line at a lower-level than current if already adorned.</p> <ul style="list-style-type: none"> <li>If the line not already adorned, adorn it with a level lower than previous section.</li> </ul>
Refresh current line adornment	<f12> r <f11> SPC M-r r	(pel-rst-adorn-refresh)	<p>Refresh the adornment of the current line, adjusting the underlining to the current length of the line.</p> <ul style="list-style-type: none"> <li>This can be useful when changing the text on the line.</li> </ul>
<b>Creating and Using Hyperlinks</b>	<p>The following 3 PEL commands help write hyperlink of various forms:</p> <ul style="list-style-type: none"> <li>the embedded form where the URL is stored inside the text between angle brackets and</li> <li>the full named format where the link is located elsewhere in the file on its own line.</li> </ul> <p>When editing a buffer using the rst-mode, type the &lt;f12&gt; . keystroke to create a hyperlink.</p> <ul style="list-style-type: none"> <li>It uses the selected 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 &lt;f12&gt; s keystroke. That helps identify an area inside the file where the next (or several) hyperlinks will be located.</li> <li>With PEL, the &lt;f12&gt; 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 &lt;f11&gt; SPC M-r prefix (assuming that pel-use-rst-mode user-option is set).</li> </ul>		
Set location of hyperlinks	<f12> s <f11> SPC M-r s	(pel-rst-set-ref-bookmark)	<ul style="list-style-type: none"> <li>Set the reference bookmark for the currently edited file at point.</li> <li>Used to identify the location where the next invocation of M-x pel-rst-mekelink inserts fully expanded links.</li> <li>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.</li> </ul>
Add an hyperlink for text at point	<f12> . <f11> SPC M-r .	(pel-rst-makelink &optional ARG)	<p>Create a reStructuredText hyperlink prefix for the word at point or region’s text.</p> <ul style="list-style-type: none"> <li>If region active, use text of the region for the link, otherwise use the word at point.</li> <li>If an argument (which can be a <b>C-u</b>) is specified, use the embedded URI format.</li> <li>If no argument is specified, use the named hyperlink format: <ul style="list-style-type: none"> <li>if the region is a single word, just append an underscore to make the link</li> <li>if the region is several words, surround it with the "" and the ""_ " strings.</li> </ul> </li> <li>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.</li> <li>The cursor is placed where the URL is to be written.</li> <li>Command pushes the mark on mark ring, type <b>M-`</b> or <b>&lt;f6&gt;&lt;f6&gt;</b> to move back to previous location.</li> </ul>
Go to hyperlink location	<f12> g <f11> SPC M-r g	(pel-rst-goto-ref-bookmark)	<p>Move point to the reference bookmark.</p> <ul style="list-style-type: none"> <li>Useful to see where the bookmark for storing the hyperlink are currently located or add empty lines for future references.</li> <li>Command pushes the mark on mark ring, type <b>M-`</b> or <b>&lt;f6&gt;&lt;f6&gt;</b> to move back to previous location.</li> </ul>
<b>Activating URLs to browse and open files</b> <p>See also:</p> <ul style="list-style-type: none"> <li> <b>File mngrt</b></li> <li> <b>Navigation</b></li> </ul>	<p>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.</p> <ul style="list-style-type: none"> <li>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> </li> <li> Customization group: <b>goto-address</b> . Mostly control the regex for URL and the face used.</li> </ul>		
Toggle goto-address-mode	<f11> f u	(goto-address-mode &optional ARG)	<p>Minor mode to buttonize URLs and e-mail addresses in the current buffer.</p> <p>With a prefix argument ARG, enable the mode if ARG is positive, and disable it otherwise.</p>
Toggle goto-address-prog-mode	<f11> f U	(goto-address-prog-mode &optional ARG)	<p>Like ‘goto-address-mode’, but only for comments and strings.</p>



rst-mode — References

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