rst-mode: reStructuredText Mode

	130	-inode. restruc	tured lext Mode	
<u>Description</u>	<u>Keystroke</u>	Function	<u>Note</u>	
reStructuredText o rst-mode, syntax-control text-filling, text emphasis itemize list of lines Format reStucturedText table View/Navigate Table of Content Insert file header Adorn Section Level Creating/Using hyperlinks Using goto-url-mode Copy URL target in file & visit it Open File at point	This page describes Emacs support for reStructuredText (abbreviated sometimes as 'rst' and sometimes as 'reST'). • The reSructuredText files are supported by Emacs rst-mode from rst.el which is available in standard Emacs distribution. • Supported file extensions: .rst, .rest, .stxt and .rst.txt. The .rst.txt extension allows rendering by tools supporting .txt files. • To activate it under PEL, you must set the PEL pel-use-rst-mode customization variable to t. • pel-rst-tab-width: The width of a tab used for reStructuredText files. Defaults to 2. • This concept differs from indentation: you can have an indentation of 3 and tab width of 8: M-i will move point to columns that are multiple of 8 < tab> will indent to a column that is a multiple of 3. PEL stores this value inside the tab-width user option variable for rst-mode buffers. See Indentation. • pel-rst-use-tabs: whether hard tabs are used for indentation. Defaults to nil (use space characters for all indentation). • Speedbar Support: • PEL activates Speedbar support for reStructuredText when the pel-use-speedbar user-option is turned on (set to t). Use the Speedbar to see the			
Supports reStructured text hyperlinks	More Information about reStructuredText: Emacs Support for reStructuredText Basic Intro to rst reStructuredText markup		reStructuredText Directives Quick reference to rst rst-cheatsheet (pdf) Sphinx & rst syntax guide	
Last updated on:	2025-02-28			
Open this PDF file. See also: <u>▼ Help/Info</u>	<f11> SPC M-r <f1> <f12> <f1></f1></f12></f1></f11>	(pel-help-pdf &optional OPEN-WEB-PAGE)	Open the <u>M reStructuredText</u> 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 useroption is set it's the other way around.	
Customize PEL reStructuredText support	<f11> SPC M-r <f2> <f12> <f2></f2></f12></f2></f11>	(pel-customize-pel &optional OTHER-WINDOW)	Customize PEL reStructuredText support. • If OTHER-WINDOW is non-nil (use C-u), display in another window.	
© Customize Emacs reStructuredText support	<f11> SPC M-r <f3> <f12> <f3></f3></f12></f3></f11>	(pel-customize-library &optional OTHER-WINDOW)	Customize Emacs reStructuredText support. • If OTHER-WINDOW is non-nil (use C-u), display in another window.	
<u>rst-mode</u>		Emacs provides the <u>rst-mode</u> from the <u>rst.el</u> file. The following file extensions are associated with this major mode: .rst, .rest. PEL adds the .stxt extension.		
Activate reStructuredText mode	M-x rst-mode	(rst-mode)	Toggle the rst-mode used to edit reStructuredText markup. • Automatically invoked when visiting .rst, .rest files (and .stxt files with PEL).	
Get version of rst-mode	C-h v rst-version		Shows the content of the variable rst-version. Only works once the rst-mode is loaded.	
Syntax Control • superword-mode	reStructuredText files often benefit from the superword-mode and treating underscores as part of words. This helps searching for symbols and opening file that have names that include underscore characters. To force activation of various modes in reStructuredText: first open PEL customization buffer for reStructuredText with f12> f2> , then To force superword-mode to the pel-rst-activates-minor-mode user-option. To force treating underscore as symbol during superword-mode: add pel-rst-activates-minor-mode user-option list.			
Control syntax of the underscore character	<f12> _</f12>	(pel-rst-set-underscore- syntax &optional ACTION)	Control syntax of underscore to punctuation or symbol when the superword-mode is active.	
Requires superword-mode on. See: <u>▼ Text Modes</u>	If superword-mode is off the function issues a user error. Otherwise the function operates according to the value of the optional argument: If the argument is not specified or nil, the function toggles the syntax of the underscore character between punctuation (the default) and symbol. If the argument is positive, it sets the syntax of underscore to symbol. If the argument is 0 or negative it sets the syntax of the underscore back to punctuation.			
Editing Content	The following generic comma	ands are useful when editing reS	tructuredText content.	
Text filling See also: ∑ Filling/Justification ∑ Filling/Justification	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. You can turn the auto fill mode on and identify the fill column. Force the auto-fill-mode when a reStructuredText file is visited by adding the auto-fill-mode to the pel-rst-activates-minor-modes user-option.			
Toggle auto-fill mode	• <f11> t f a • <f11> RET</f11></f11>	(auto-fill-mode &optional ARG)	Toggle automatic line breaking (Auto Fill mode). With a prefix argument, enable Auto Fill mode if the prefix argument is positive, and disable it otherwise. When Auto Fill mode is enabled, inserting a space at a column beyond 'current-fill-column' automatically breaks the line at a previous space.	
Set Fill Column	• C-x f • <f11> t f c</f11>	(set-fill-column ARG)	When no prefix value: prompts for column unless a prefix argument was used. If with C-u prefix: use current column. If with prefix value: use that value.	
Fill current paragraph	• M-q • <f11> t f p</f11>	(fill-paragraph &optional JUSTIFY REGION)	To justify as well: C-u M-q • In refill mode this is done automatically. In auto fill mode the filling is done at the end of the line.	
Align a set of lines on some text	<f11> t w a</f11>	(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 Use it to align hyperlink references URL: select all hyperlink lines and then issue the command, specifying http as the regexp to line them vertically. 			
Text Emphasis	The PEL commands emphas	ize the current word or marked r	egion, then move point to the character right after the emphasized text or inside if empty.	
Bold	<f12> b <f11> SPC M-r b</f11></f12>	(pel-rst-bold)	Mark current word or marked region bold. • Leave point after to the next character. • Inserts required escaped spaces when the emphasized region is inside a word.	
Italic	<f12> i</f12>	(pel-rst-italic)	Inserts required escaped spaces when the emphasized region is inside a word. Mark current word or marked region italic.	
	<f11> SPC M-r i</f11>	,	Leave point after to the next character. Inserts required escaped spaces when the emphasized region is inside a word.	
Literal	<f12> 1</f12>	(pel-rst-literal)	Mark current word or marked region with the literal markup.	
	<f11> SPC M-r 1</f11>		 Leave point after to the next character. Inserts required escaped spaces when the emphasized region is inside a word. 	
Interpreted	<f12> ` <f11> SPC M-r `</f11></f12>	(pel-rst-interpreted)	Mark current word or marked region with the interpreted markup. Leave point after to the next character. Inserts required escaped spaces when the emphasized region is inside a word.	
Indent list item See also: ∑ Indentation	<tab></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 also: ∑ Comments	М-;	(comment-dwim ARG)	Comment line or region. ! Uncommenting does not work.	
Itemize all previous lines same indention level	• <f12> M • M-<f12> M</f12></f12>	(pel-itemize-lines &optional ITEM-PREFIX-STRING)	Prepend each of the previous lines with a ITEM-PREFIX-STRING that is "- " by default. • Indents all lines above current line that are at the same indentation level as the current position.	
4	Use it to put a "- " prefix on e mark.	each line instead of typing manua	ally. Put point at empty line after list. Type the command to itemize all lines above. No need to	

<u>Description</u>	<u>Keystroke</u>	Function	Note
Duplicate current table	• <f12> M-t</f12>	(pel-rst-table-dup-	Duplicate the current table underlining separator, adding it to the top and the bottom of the
underlining This is not a perfect helper for	• M- <f12> M-t</f12>	separator-lines &optional UPDATE)	table. This command helps you creating a <u>reStructuredText simple table layout</u> , assuming you have only 1 title row and point is at the written separator line. • With the C-u option updates the top and bottom line to the line at point.
creating reStructuredText table but it helps creating simple ones.	Assuming you have the following text and point is	Header 1 Header 2	Header 3
For example, to create the top and	on the second line (in blue)	abcdef. ghijkl.	12345
bottom line from the line under the title:	After executing the command, the top and	Header 1 Header 2	
	bottom lines are inserted. The point does not move.	Header 1 Header 2 abcdef. ghijkl.	12345
File's Table of Content	 Use the <u>contents markup directive</u> to generate a table of contents for your reStructuredText file based on its sections. Insert the table of content text inside the file with the <u>rst-toc-insert</u> command (although you may want to use the <u>contents markup directive</u> instead). Generate a table of content in a buffer to view the file sections and navigate inside them: with C-c C-t to invoke the <u>rst-doc</u> command: it opens a *table of Content* buffer, moves point inside it, move to the section title, hit <u>RET</u> to 		
See also: <u>Speedbar</u>	select that section inside the original reStructuredText buffer. • using the Speedbar to open a buffer that lists the sections. See Speedbar.		
Insert a table content at point	C-c C-t TAB	(rst-toc-insert &optional MAX-LEVEL)	Insert the table of contents of the current section at the current column. • By default the top level is ignored if there is only one. • A numeric prefix argument MAX-LEVEL overrides 'rst-toc-insert-max-level'. • Text in the line beyond column is deleted. • You may want to use the contents markup directive instead.
Display table of content	C-c C-t C-t	(rst-doc)	Display a table of contents for current buffer inside the *Table of Contents* buffer. • Displays all section titles found in the current buffer in a hierarchical list.
Navigate to specific section			Select the section of interest in the *Table of Contents* buffer by navigating to it, then hit RET on that section to move back to the section in the original reStructuredTex document and close the Table of Contents* buffer window.
Moving across sections	You can also use the following	g commands to move to the nex	t or previous section.
Move to previous section title	• C-M-a • <f12> p • <f12> <up> • <f11> SPC M-r p</f11></up></f12></f12>	(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.
	• <f11> SPC M-r <up></up></f11>		
Move to next section title	• C-M-e • <f12> n</f12>	(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.
	• <f12> <down> • <f11> SPC M-r n</f11></down></f12>		
	• <f11> SPC M-r • <down></down></f11>		
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.
Insert elements based on 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).</f12></f12>		
Insert a file header	<f12> <f12> h</f12></f12>	(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</f12></f12>	(pel-tempo-mode &optional ARG)	Toggle pel-tempo-mode on/off. That mode activates key bindings to move to pre-defined hot-spots where template text must be added.
See also: <u>▼ Mode Line</u>	The keys are: C-c . and. C-c , as well as (in graphics mode only): C-c C and C-c C-, When pel-tempo-mode is active the pel-tempo-mode lighter (‡) is shown on the status bar. Solution with the company of the pel-tempo-mode is automatically activated.		
Jump to next tempo mark	• C-c M-f	(tempo-forward-mark)	Jump to the next mark in 'tempo-back-mark-list': the location where code must be updated
	• C-c . • C-c C		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></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 contained and then hit <f12> <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, at the moment, only one template is available in rst-mode, the usefulness of this command is limited for reStructuredText.</f12></f12></f12></f12></f12></f12>		
Select Section Title 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</f12>	(pel-rst-adorn-default)	Set the default section adornment style. This is Emacs rst-mode default: a title with 7 levels.
Salast Sphing Dallers	<f11> SPC M-r A d</f11>	(not not adams Only)	
Select Sphinx-Python adornment style	<f12> A S</f12>	(pel-rst-adorn-Sphinx- Python)	Set the Sphinx-Python section adornment style. This is what Sphinx supports: 6 levels:
	<f11> SPC M-r A S</f11>		 parts, chapters, sections, subsections, subsubsections, paragraphs.
Select CRiSPer adornment style	<f12> A C</f12>	(pel-rst-adorn-CRiSPer)	Set the CRiSPer section adornment style.
	<f11> SPC M-r A C</f11>	,	A title level with another 12 levels. Use <f12> + to create those levels.</f12>

Description	<u>Keystroke</u>	Function	<u>Note</u>
Section Title level adornment commands that insert section titles	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" above.		
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 M-r t</f11></f12>	(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, a mark is left at the end of the title line and point is moved 2 lines below. To return to the end of the title line, type M-` or <f6><f6>.</f6></f6>
Adorn to specific level From level 1 to level 10	• <f12> 1 • <f12> 9 • <f12> 0 • <f11> SPC M-r 1 • <f11> SPC M-r 0</f11></f11></f12></f12></f12>	(pel-rst-adorn-1) (pel-rst-adorn-2) (pel-rst-adorn-3) (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 M-r 1 to <f11> SPC M-r 0 key sequences can be used inside any buffer. The <f12> keys can only be used in inside the buffers in rst-mode.</f12></f11></f11>
Adorn current line: same section level as previous section	• <f12> = <f11> SPC M-r =</f11></f12>	(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 M-r +</f11></f12>	(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 M-r -</f11></f12>	(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 M-r r</f11></f12>	(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.
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 rst-mode, type the <f12> • keystroke to create a hyperlink. • 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 <f12> s keystroke. That helps identify an area inside the file where the next (or several) hyperlinks will be located. • With 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</f12></f12></f12>		
Set location of hyperlinks	<pre></pre>	r prefix (assuming that pel-use (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.
Add an hyperlink for text at point	Ensures the bookmark is a beginning of the first of the <f12> . SPC M-r .</f12>		which is followed by another empty line, by inserting 2 lines and placing the point at the Create a reStructuredText hyperlink prefix for the word at point or region's text. If region active, use text of the region for the link, otherwise use the word at point. If an argument (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 it with the "' and the "' strings. 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- or <f6><f6> to move back to previous location.</f6></f6> 		
Go to hyperlink location	<f12> g <f11> SPC M-r g</f11></f12>	(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- or <f6><f6> to move back to previous location.</f6></f6>
Activating URLs to browse and open files See also: •	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 the following key sequences are available wheel point is over a URL button: 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 a web or FTP address the system browser is invoked to open the address. C-c C-n: move point to the end of the next URL in the buffer. C-c C-g: move point to to the previous URL in the buffer. C-c C-f: download the file identified by the URL into a local temporary file and visit the file. See (pel-open-url-at-point) above. Customization group: goto-address. Mostly control the regex for URL and the face used.		
Toggle goto-address-mode	<f11> f u</f11>	(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-addrress-prog- mode	<f11> f U</f11>	(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)	Open the URL at point: If URL is a web page: open it in a browser If URL is a mail address: 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.
Move to end of next URL in buffer	C-c C-n	(pel-goto-next-url)	Move point forward to the end of the next URL located in the current buffer. • The global <f6> C-n key binding activates the goto-address-mode if it is not already</f6>
See also: <u>Navigation</u> Move to beginning of previous	<f6> C-n C-c C-p</f6>	(pel-goto-previous-url)	active. Move point backward to the beginning of the previous URL located in the current buffer.
URL in buffer See also: <u>▼ Navigation</u>	<f11> C-p</f11>		The global <f6> C-p key binding activates the goto-address-mode if it is not already active.</f6>

<u>Description</u>	<u>Keystroke</u>	Function	<u>Note</u>		
Copy URL at point in temporary ille and visit the file See also: ∑ File mngt (pe	(pel-open-url-at-point)	Copy the URL at point to a local temporary file and visit that file. • 1 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. • Save the file somewhere else using the C-x C-w key sequence and identify the proper extension to activate the required major mode.			
	C-c C-f		This binding is only available when point is over the URL and the <u>goto-address-mode</u> minor mode is active. Use <f11> f u or <f11> f U to activate this mode.</f11></f11>		
Open file or web-page whose name is at point	• M- <f6> • <f11> f . • 6y This command works ger</f11></f6>	(pel-open-at-point &optional N) nerically but is also specialized for	Open the file, library or the URL, named at point, with potential line & column #s. Supports glob characters, partial directory path. When multiple files are found it prompts using the method selected by pel-prompt-read-method user-option. The <u>6y</u> key-chord is available if pel-use-key-chord is non-nil. See <u>E Key-Chords</u> . The reStructuredText: Inside a rst-mode buffer, when the point is over a reStructuredText		
• BI - C++	 external hyperlink target reference, the command locates the reference and opens the file identified by the reference (unless N >= 100): If the reference is a web URL, it opens the identified web using the system browser. If the reference is a HTML file name that corresponds to the rendering of a local restructuredText file, it opens that reStructuredText file. If the reference is a HTML file that does not correspond to a reStructuredText source, it opens that HTML file. If the reference is another type of file it opens that file. 				
• \$\mathbb{g}(\cdot\) - Erlang • \$\mathbb{g}(\cdot\) - UNIX Shell					
Jump to referenced link (unless N >= 100) ►		rence URL identifies a # <u>URI fragment</u> that identifies the name of a target file section, the command moves point to the section erlink refers to a title inside the document move point to that title instead of trying to open a file.			
Delimiting characters 🖛	In general the command extracts the file or directory name, and possibly line and column numbers, from text at point and tries to open the file or directory. • The generic mode extraction works by identifying the beginning & end of the file/directory/library/URL name string by delimiter characters, one of: tab, newline and: "`' ()[]{}<> ''" 「」 () 〈〉 《》 [] 《》 、 () 〈》 [] 《》 。				
	If embedded space(s) are a The name may include glo		be located at the first of the 2 delimiter chars. Otherwise point can be anywhere in the name.		
File identification heuristic <f11> f <f2> F <f11> f ; F</f11></f2></f11>	The command uses a URL unchanged but uses the following heuristic to identify the exact location of the file/directory: • In the file/dir name is an absolute path it uses that. Otherwise • it builds a absolute path using the extracted relative path name inside the directory identified by the pel-open-file-at-point-dir user-option, which can be used to be user-specified directory. It uses the found file/dir name if it exists. Otherwise • it searches for the relative file/dir name in directory tree under the root marker file identified by the pel-project-root-identifiers user-option which is something like .git, .hg, .project, .pel-project (the default). If it can find such a file in the above directories it searches the tree under the found root. • If it finds several files it prompts using the current completion mode to allow selection of the appropriate name (see below) and opens the selected one. • If it finds only one it opens that file. Otherwise, • it prompts showing the name searched and provide the following choices: 1) create the file with specified name, 2) edit the name to search again, 3) use the name found and search for an Emacs library file with that name, or 4) quit.				
	The command opens the extracted name according to this heuristic: • If the string is a properly formatted URL , it opens it using the OS default browser (even if a optional nume if the string is a file or directory name it opens it. • If the file name is followed by line and column numbers the point is moved to that position in the buffer. When finding several file names, the command lists them and prompts using the method selected by p . • The default is a very primitive function implemented by PEL. You can select a more powerful <u>ivy</u> prom		he OS default browser (even if a optional numeric argument specified otherwise), otherwise		
	With ivy selected, PE Note that the command	L will automatically set 🚧 pel-u shows all files found by the spe	ise-ivy to t and lvy mode will be installed automatically when you restart Emacs. cified search method, it does not only use the first one found.		
Select multi-file selection	• Subsetthis to detect	potential duplication in header t	file names in large include paths.		
method =	The command opens the file in the window selected by the following logic controlled by presence or absence of typed numerical prefix arguments: • Select target window: • Without argument:				
Select target window -	If no window holds th	nat file, select the target window other window, if 3 or more, use t	e point to that window and to the line column coordinates if specified. according to the number of editable windows in frame: if 1, split that window and use the new the current window.		
·	 N = 0: use the 'other' N = 1, 3, 7or above (e if 1 window: if 2 windows: 	en the directory instead of the fi (the next) window. excluding 8, 9 and 10): select the split that window and use the other window,	ile. Interpret the window position from the N value adjusted: N-20 (or N+20 if N is negative) target window based on the number of editable windows in frame: the new window,		
N>20 : open the directory	 N is: 8: up, 2: down, N is 9: force openin (eg. macOS Finder, W If N >= 100, restruction 	Vindows Explorer). If this is a UF	application (with N=29 or N=-29, open the file's directory with the OS associated application RL, open it in the OS default web browser. gnored, text is used as the file target and N-100 is used to determine the window selection. not allowed.		

rst-mode - References

See function docstring for more info.

Tat-mode — Neierences				
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				