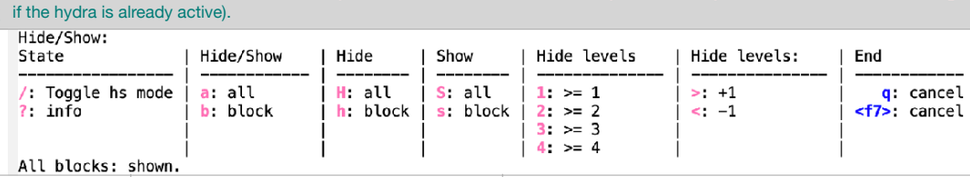
Hide/Show Code Blocks & Selective Display

Operation	Keystroke	Function	Note
HideShow Minor Mode	When working with source code files, you can use the Hide/Show minor mode to collapse and expand blocks of code, where the concept of "*block of code*" depends on the specific programming language. For example C-like programming language use braces to delimit blocks, while in Lisp languages use parentheses for all blocking. • When a block is hidden (collapsed) it is replaced by "…" surrounded by the block delimiter of the specific programming language. • ☝📦🤖 PEL provides an easy to use Hydra invoked via the **`<f7> /`** key prefix to control the Hide/Show mode and quickly issue commands to hide or show portions of the text when the **pel-use-hydra** is set to **t** to activate the **external hydra package**. The next row provides more information. • Without PEL you must activate the Hide/Show mode with M-x hs-minor-mode and then use the Emacs commands bound to **`C-c @`** key prefix. • Once the Hide/Show minor mode is active the modeline will show "hs" . • When you disable the Hide/Show mode, hidden text is restored. ☝You may want to hide comments as well when collapsing code: you can use **`<f11> ; ~`** to toggle visibility of comments, see ∑ Comments table for more info on this.)		
HideShow Hydra	**Using the PEL HideShow Hydra** 📦🤖PEL provides a Hide/Show Hydra when **pel-use-hydra** user option is set to **t**. • Activate this hydra with the **`<f7> /`** key prefix. • Once this hydra is active, you can then type any of the keys in the hydra (see the menu below) without having to type the **`<f7> /`** prefix again and so until you terminate the hydra by typing the **`q`** key or the **`<f7>`** key. • While active the hydra displays the menu shown below and operation results below the menu. • You can issue any other command (not bound to the keys listed in the menu) while the hydra is active. • You do not have to activate the hs-minor-mode for any of the commands in the menu: they automatically activate it. Use **`<f7> / / q`** to de-activate it (or **`/ q`** if the hydra is already active).		
			

Toggle Hide/Show Minor Mode	**`<f7> / /`**	(**hs-minor-mode** &optional ARG)	Toggle Hide/Show minor mode to selectively hide/show code and comment blocks.
			• With a prefix argument ARG, enable the mode if ARG is positive, and disable it otherwise.
			• When hideshow minor mode is on:
			  • The menu bar is augmented with hideshow commands and the hideshow commands are enabled.
			  • The line-mode shows 'hs'.

Describe current state of PEL show/hide	• **`<f7> / ?`**	(**pel-show-hide-state**)	Display state of pel-hideshow in current buffer.
Show (expand) all blocks in buffer	• **`<f7> / S`** • **`C-c @ C-M-s`** • **`C-c @ C-a`**	(**pel-show-all**) (**hs-show-all**)	Show all blocks.
Hide (collapse) all blocks in buffer	• **`<f7> / H`** • **`C-c @ C-M-h`** • **`C-c @ C-t`**	(**pel-hide-all**) (**hs-hide-all**)	Hide all top level blocks, displaying only first and last lines.
Hide (collapse) current block	• **`<f7> / h`** • **`C-c @ C-h`** • **`C-c @ C-d`**	• (**pel-hide-block** &optional END) • (**hs-hide-block** &optional END)	Select a block and hide it. • With prefix arg, reposition at END.
Show (expand) current block	• **`<f7> / s`** • **`C-c @ C-s`**	• (**pel-show-block** &optional END) • (**hs-show-block** &optional END)	Select a block and show it. • With prefix arg, reposition at END.
Toggle visibility of all blocks in buffer	• **`<f7> / a`**	(**pel-toggle-hide-all**)	Toggle hide/show of all blocks. • Activates the Hide/Show mode if not already active (and hide all blocks)
Toggle visibility of current block	• **`<f7> / b`** • **`C-c @ C-c`** • **`C-c @ C-e`**	• (**pel-toggle-hide-block**) • (**hs-toggle-hiding**)	Toggle hide/show of current block.
Hide all blocks 1 level below current block	• **`<f7> / 1`**	(**pel-hide-level-1**)	Hide all blocks 1 level below the current block. ☝ Useful in language like Python to show the methods of a class.
Hide all blocks 2 level below current block	• **`<f7> / 2`**	(**pel-hide-level-2**)	Hide all blocks 2 level below the current block.
Hide all blocks 3 level below current block	• **`<f7> / 3`**	(**pel-hide-level-3**)	Hide all blocks 3 level below the current block.
Hide all blocks 4 level below current block	• **`<f7> / 4`**	(**pel-hide-level-4**)	Hide all blocks 4 level below the current block.
Hide all blocks N levels below this block.	**`C-u n C-c @ C-l`**	(**hs-hide-level** ARG)	Hide all blocks ARG levels below this block. • Like all other commands that take a numeric argument, the numeric argument (shown in the keystroke column as **`C-u n`**, can also be typed with the **`M-`***number*)
Hide one more extra level below current level	• **`<f7> / >`**	(**pel-hs-hide-block-below-inc**)	Hide all blocks of 1 more level deep below this block level of point. • Warns/stops upon reaching the limit of +10 levels (a hard-coded limit) of blocks.
Hide one less level below current level	• **`<f7> / <`**	(**pel-hs-hide-block-below-dec**)	Hide all blocks of 1 less level deep below the block level of point. • Warns/stops upon going down to +0 levels.
Selective Display	As stated in the Emacs manual: "Emacs has the ability to hide lines indented more than a given number of columns. You can use this to get an overview of a part of a program. • To hide lines in the current buffer, type C-x $ (set-selective-display) with a numeric argument n. Then lines with at least n columns of indentation disappear from the screen. The only indication of their presence is that three dots ('...') appear at the end of each visible line that is followed by one or more hidden ones. • The commands C-n and C-p move across the hidden lines as if they were not there. • The hidden lines are still present in the buffer, and most editing commands see them as usual, so you may find point in the middle of the hidden text. When this happens, the cursor appears at the end of the previous line, after the three dots. If point is at the end of the visible line, before the newline that ends it, the cursor appears before the three dots. • To make all lines visible again, type C-x $ with no argument."		
Set/clear selective display of lines with indentation >= n	**`C-x $`**	(**set-selective-display** ARG)	Set 'selective-display' to ARG; clear it if no arg. • When the value of 'selective-display' is a number > 0, lines whose indentation is >= that value are not displayed. • The variable 'selective-display' has a separate value for each buffer.

Operation	Keystroke	Function	Note
Start selective display Hydra	`<f7> C-x $`	pel-∑hide-indent	📦🔄 With PEL, when **pel-use-hydra** is set to **t**, this key starts a hydra to manage selective display and easily move the selective display column left or right with the cursors by column or indentation level, to stop it (**0**), to hide most lines (**1**) and to highlight the right-most visible column (|). The hydra menu that appears in the minibuffer when the hydra is active is shown below. • You can execute other commands while this hydra is active. • Terminate the Hydra by typing the **<f7>** key again.

```
Selective Display:
By Column          | By Indent             | Show                       | End
-------------------|-----------------------|----------------------------|-----------
<right>: +1        | S-<right>: +indent    | |: rightmost visible limit | <f7>: cancel
 <left>: -1        |  S-<left>: -indent    |                            |
       0: unhide   |                       |                            |
       1: hide at 1|                       |                            |
```

Hide/Show Comments (See also: ∑Comments)	The hide-cmnt file, written by Drew Adams, provides the following commands to quickly hide/show the comments in a buffer. • PEL provides binding for these 2 commands, but the file must be installed manually copied in a directory the is in your Emacs load path. • You can download the file from the hide-comnt.el EmacsWiki link or from the hide-count EmacsMirror. They should contain the exact same code. ☝ These are very useful to see a list of methods without all comments when you also use the Hide/Show Mode commands (see ∑ Hide/Show Code table). 📦 Both of these commands require the hide-comnt.el package (see above). 🔄 PEL activates it when the pel-use-hide-comnt user option is **t**.		
Toggle display of comments in buffer or active region	`<f11> ; ;`	(**hide/show-comments-toggle** &optional START END)	Toggle hiding/showing of comments in the active region or whole buffer. • If the region is active toggle in the region, otherwise, in the whole buffer.

Show (or hide) comments in buffer	`<f11> ; :`	(**hide/show-comments** &optional HIDE/SHOW START END)	Hide or show comments in buffer or active region.
			• Hide if no argument. To show, use any prefix argument (any of the **C-u**, **M- -**, **M-0** to **M-9** will do).
			• If a region is active the command applies to the active region, otherwise it applies to the entire or narrowed buffer.
			• Uses 'save-excursion', restoring point.
			• Option 'show-invisible-comments-shows-all':
			  • If non-nil then using this command to show invisible text shows *ALL* such text, regardless of how it was hidden. IOW, it does not just show invisible text that you previously hid using this command.
			  • If nil (the default value) then using this command to show invisible text makes visible only such text that was previously hidden by this command. (More precisely, it makes visible only text whose 'invisible' property has value 'hide-comment'.)

Hide/Show docstrings 🚧 Work in Progress 🚧	Some programming languages support the concept of **docstrings**. These are no comments and are highlighted differently than comments by Emacs. In some programming languages (Lisp, Python, Clojure) the docstring for functions appears between the argument list and the function body. In some cases the docstring is long and being able to quickly hide it helps when editing or reviewing source code. Docstrings can also be present in other places, at the beginning of a Python module or class definition for example. PEL provides its own facility to selectively hide and show the docstring of definitions for languages that support the concept of docstrings. With the commands below you can hide and show back the docstring of the current, previous or next Emacs Lisp definition (function, macro, defsubst, etc…) 🚧 This is work in progress: • It currently only supports Lisp type languages and Python. • Python support is incomplete: support for hiding module docstring is not yet implemented. • Does not properly handle the ability to hide several docstrings, then show back some of them. That will work as long as you do not use another command that uses the visible property, such as comment hiding. Docstring and comment hiding co-exists without problem when only hiding one docstring at a time. Elixir, Haskell and Julia also support docstrings, but they are located before the function definition, just like documentation comments used by conventions in languages that do not support docstring. The following commands do not support those languages for the moment. There's also Erlang's Typer specifications. I might want to add some support there. My plan is to first complete robust support for Emacs Lisp, then for Python and then check if the Emacs Lisp works well with Clojure and Common Lisp. For now just be careful when using these commands.		
Toggle visibility of the docstring	`<f11> ; '`	(**pel-hide/show-docstring-toggle** &optional NEXT)	Toggle the visibility of the docstring in the current or previous definition. • With any prefix argument (like **C-u**, **C--** or **M--**) toggles the docstring visibility of the next definition.
Hide/show current/ previous docstring	`<f11> ; "`	(**pel-hide/show-docstring** &optional SHOW)	Hide docstring of current or previous definition. • With any prefix argument (like **C-u**, **C--** or **M--**) show the docstring instead.

Hide/Show Code Blocks — References

Topic & Link	Description
GNU Emacs Manual - Hideshow minor mode	Emacs section that describes the Hide/Show minor mode.
GNU Emacs Manual - Selective Display	Description of the selective display feature.