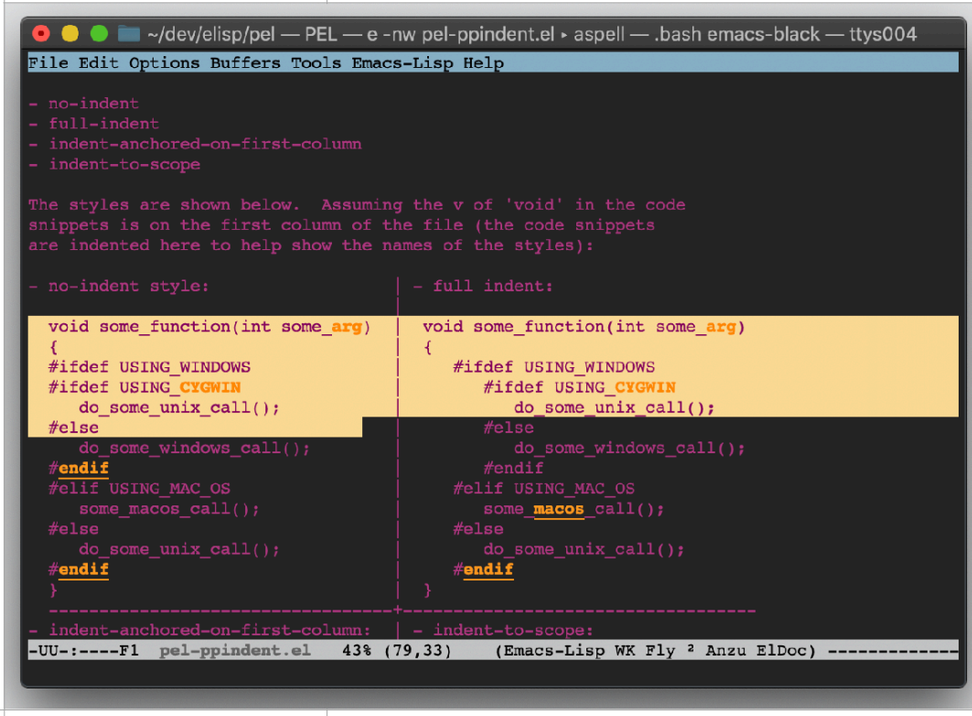
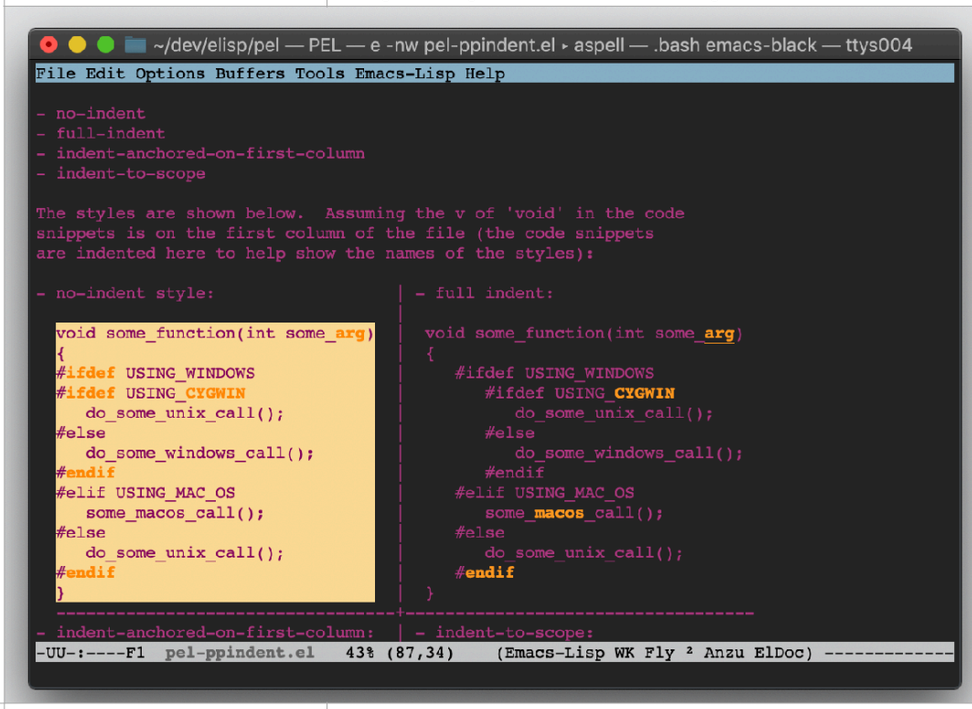
↖ Topic Index

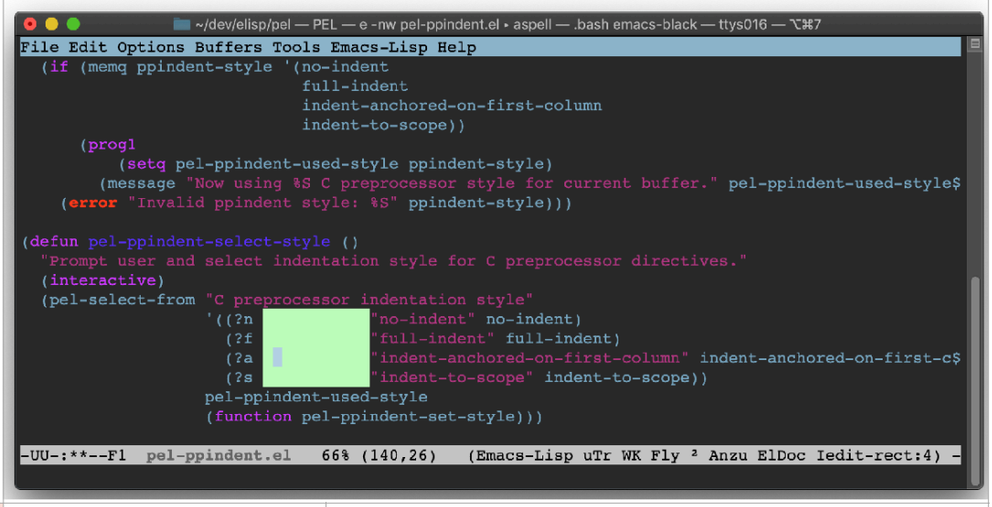
Rectangles

Operation	Keystroke	Function	Note
Rectangles See also: ∑ Drawing	The following commands operate on the rectangle area that is made of the area made of the opposite corners of the point and mark. • The rectangle can be defined by the normal set-mark-command as shown in the first screen shot below. However, when you use that, the highlighted area shows more than what constitutes the rectangle. You just have to remember that the rectangle is made of the opposite corners including the corner identifying the position when you issued the command. • The rectangle-mark-mode command provides a better visual feedback as it only highlights the area that constitutes the rectangle as shown in the second screenshot below. ☝ Remember that normally you cannot place your cursor in "void" space (positions that do not correspond to a character inside a buffer or file). That would prevent you from easily navigating vertically over ares where there is no text. To solve this problem simply activate a drawing mode while defining your rectangle: the **artist-mode** or **picture-mode** will both do. These modes allow you to place your cursor anywhere on window screen. • PEL binds **<f11> D a** to toggle artist-mode and **<f11> D p** to toggle picture-mode. You can also use M-x to issue the command. • See ∑ Drawing for more info on these modes.		
Set mark & activate/ deactivate it See also:∑ Marking	• **C-SPC** • **C-@** • **<f11> . s**	(**set-mark-command** ARG)	Set the mark where point is and toggle its activation. • If mark was not active it activates it: moving the cursor further will show the marked area (the region) if transient mode is enabled (the default in Emacs). • If the mark is active, de-activates it. ☛ Issuing the command twice (**C-SPC C-SPC**) sets the mark location and de-activates it. You can use this command to create a rectangle: the rectangle will not show explicitly, in the example below it is defined by the top-left and bottom-right corners of the marked area that is highlighted.
Toggle rectangle Mark Mode See also:∑ Marking	**C-x SPC**	(**rectangle-mark-mode** &optional ARG)	Toggle the region as rectangular. • Activates the region if needed. Only lasts until the region is deactivated. • When this mode is active, the region-rectangle is highlighted and can be shrunk/ grown, and the standard kill and yank commands operate on it. See the screenshot below, where the mark was activated with **C-x SPC** on the first letter of the word "void" and then the cursor moved down right to highlight a rectangle. Nothing "*bleeds*" outside of the rectangle.
Copy/Save rectangle text See also: ∑ Cut & Paste	• **C-x r M-w** • **<f11> = r**	(**copy-rectangle-as-kill** START END)	Copy the region-rectangle and save it as the last killed one.
Kill text in rectangle See also: • ∑ Cut & Paste	• **C-x r k** • **<f11> - r**	(**kill-rectangle** START END &optional FILL)	Delete the region-rectangle and save it as the last killed one. • If the buffer is read-only, Emacs will beep and refrain from deleting the rectangle, but put it in 'killed-rectangle' anyway. This means that ou can use this command to copy text from a read-only buffer. (If the variable 'kill-read-only-ok' is non-nil, then this won't even beep.)





Operation	Keystroke	Function	Note
Delete rectangle text	**C-x r d**	(**delete-rectangle** START END &optional FILL)	Delete (don't save) text in the region-rectangle. • The same range of columns is deleted in each line starting with the line where the region begins and ending with the line where the region ends. • With a prefix (or a FILL) argument, also fill lines where nothing has to be deleted.
Yank last killed rectangle	**C-x r y**	(**yank-rectangle**)	Yank the last killed rectangle with upper left corner at point.
Fill rectangle with space	**C-x r o**	(**open-rectangle** START END &optional FILL)	Blank out the region-rectangle, shifting text right. • The text previously in the region is not overwritten by the blanks, but instead winds up to the right of the rectangle. • With a prefix (or a FILL) argument, fill with blanks even if there is no text on the right side of the rectangle.
Insert line numbers to left or rectangle	**C-x r N**	(**rectangle-number-lines** START END START-AT &optional FORMAT)	Insert numbers in front of the region-rectangle. • With a prefix argument, prompt for START-AT and FORMAT.
Clear rectangle - replace text with space	**C-x r c**	(**clear-rectangle** START END &optional FILL)	Blank out the region-rectangle. • The text previously in the region is overwritten with blanks. • With a prefix (or a FILL) argument, also fill with blanks the parts of the rectangle which were empty.
Replace rectangle content with specified string on each line	**C-x r t**	(**string-rectangle** START END STRING)	Replace rectangle contents with STRING on each line. • The length of STRING need not be the same as the rectangle width. • When called interactively and option 'rectangle-preview' is non-nil, display the result as the user enters the string into the minibuffer. ☝ This command can be used to draw vertical lines in tables, using | as the character with a rectangle over the specific column. See this example.
Delete whitespace in rectangle lines		(**delete-whitespace-rectangle** START END &optional FILL)	Delete all whitespace following a specified column in each line. • The left edge of the rectangle specifies the position in each line at which whitespace deletion should begin. • On each line in the rectangle, all contiguous whitespace starting at that column is deleted. • With a prefix (or a FILL) argument, also fill too short lines.
Insert string on each rectangle line		(**string-insert-rectangle** START END STRING)	Insert STRING on each line of region-rectangle, shifting text right. • This command does not delete or overwrite any existing text.
iedit-rectangle-mode	The iedit-rectangle-mode also allow editing rectangle areas of text. • Like other rectangle commands, you must first mark a rectangle area using the set-mark-command (**C-SPC**). See above. • In this mode the following commands are available. ⚠ However the other standard rectangle commands above do not always work. 📦 Requires the **iedit** external package. PEL downloads, installs and activates it when either **pel-use-iedit** or **pel-use-lispy** user options is set to **t**.		
Toggle the iedit-rectangle-mode	**C-x r RET**	(**iedit-rectangle-mode** &optional BEG END)	When Iedit-rect mode is on, a rectangle is started with visible rectangle highlighting. • Rectangle editing support is based on Iedit mechanism. • First select a region using marking. Use set-mark-command (bound to **C-SPC**) and then move point. • ☝ Useful to insert or remove vertical spacing or editing tabular data. • PEL activates this key binding on startup when either **pel-use-iedit** or **pel-use-lispy** user options is set to **t**
		In the following example the iedit-rectangle-mode is used to delete a rectangle of whitespace inside the text to un-indent inside code. Once the rectangle is highlighted (using a color different from other rectangle commands), any key typed inside the rectangle is applied on all row of the rectangle on specific column. Use the DEL key to delete 1 rectangle column at a time. You can move the cursor inside the rectangle.	
Kill text in rectangle	**M-K**	(**iedit-kill-rectangle** &optional FILL)	Kill the rectangle: delete the region-rectangle and save it as the last killed one. • The behavior is the same as 'kill-rectangle' in rect mode.
Replace all rectangle text with space characters	**M-SPACE**	(**iedit-blank-occurrences**)	Replace occurrences with blank spaces.
Insert line numbers in each line of rectangle	**M-N**	(**iedit-number-occurrences** START-AT &optional FORMAT-STRING)	Insert numbers in front of each line of the rectangle • START-AT, if non-nil, should be a number from which to begin counting. FORMAT, if non-nil, should be a format string to pass to 'format-string' along with the line count. • When called interactively with a prefix argument, prompt for START-AT and FORMAT.
Quit edit-mode	**C-g**	(**iedit-quit**)	Quit edit-rectangle-mode. Must be typed inside the rectangle to take effect.
Picture Mode Rectangle Commands	• The following commands allow drawing rectangles in the buffer as well as copy and remove them. • They also allow storing the rectangles in registers and restore them from rectangles. • To use them you must activate Picture mode first. With PEL use **<f11> D p**		
Draw rectangle around region	**C-c C-r**	(**picture-draw-rectangle** START END)	Draw a rectangle around region.
Clear & save rectangle	**C-c C-k**	(**picture-clear-rectangle** START END &optional KILLP)	Clear and save rectangle delineated by point and mark. • The rectangle is saved for yanking by C-c C-y and replaced with whitespace. The previously saved rectangle, if any, is lost. With prefix argument, the rectangle is actually killed, shifting remaining text.
Clear reactangle	**C-c C-w**	(**picture-clear-rectangle-to-register** START END REGISTER &optional KILLP)	Clear rectangle delineated by point and mark into REGISTER. • The rectangle is saved in REGISTER and replaced with whitespace. • With prefix argument, the rectangle is actually killed, shifting remaining text.



Operation	Keystroke	Function	Note
Yank and overlay saved rectangle	`C-c C-y`	(**picture-yank-rectangle** &optional INSERTP)	Overlay rectangle saved by C-c C-k • The rectangle is positioned with upper left corner at point, overwriting existing text. • With prefix argument, the rectangle is inserted instead, shifting existing text. • Leaves mark at one corner of rectangle and point at the other (diagonally opposed) corner.
Overlay rectangle saved in register	`C-c C-x`	(**picture-yank-rectangle-from-register** REGISTER &optional INSERTP)	Overlay rectangle saved in REGISTER. • The rectangle is positioned with upper left corner at point, overwriting existing text. • With prefix argument, the rectangle is inserted instead, shifting existing text. • Leaves mark at one corner of rectangle and point at the other (diagonally opposed) corner.

Rectangle — References

Topic & Link	Notes
GNU Emacs Manual — Rectangles	