










Operation	Keystroke	Function	Note
Smartparens Mode • Smartparens manual See also: ☞ Smartparens	Simplify insertion of matching pairs with the smartparens minor mode. PEL binds a set of keys, described below, to toggle activation of that mode.  This uses the smartparens external package.  PEL activates it when pel-use-smartparens is set to t . <ul style="list-style-type: none"> Mode line lighter: <ul style="list-style-type: none"> smartparens-mode: SP smartparens-strict-mode: SP/s 		
Help on smartparens	<f11> (?	(sp-cheat-sheet &optional ARG)	Generate a cheat sheet of all the smartparens interactive functions. Shows inside Emacs buffer. <ul style="list-style-type: none"> Without a prefix argument, print only the short documentation and examples. With non-nil prefix argument ARG, show the full documentation for each function. You can follow the links to the function or variable help page. <ul style="list-style-type: none"> To get back to the full list, use M-x help-go-back. You can use ‘beginning-of-defun’ and ‘end-of-defun’ to jump to the previous/next entry. Examples are fontified using the ‘font-lock-string-face’ for better orientation.
Describe user system	<f11> (M-?	(sp-describe-system STARTERKIT)	Describe user’s system. Prompt for starter kit: Evil, Spacemacs, Vanilla. <ul style="list-style-type: none"> The output of this function can be used in bug reports.
Toggle smartparens mode	<f11> (((smartparens-mode &optional ARG)	Toggle smartparens mode.
Toggle smartparens-strict mode	<f11> ()	(smartparens-strict-mode &optional ARG)	Toggle the strict smartparens mode. <ul style="list-style-type: none"> When strict mode is active, ‘delete-char’, ‘kill-word’ and their backward variants will skip over the pair delimiters in order to keep the structure always valid (the same way as ‘paredit-mode’ does). This is accomplished by remapping them to ‘sp-delete-char’ and ‘sp-kill-word’. There is also function ‘sp-kill-symbol’ that deletes symbols instead of words, otherwise working exactly the same (it is not bound to any key by default). When strict mode is active, this is indicated with “/s” after the smartparens indicator in the mode list
Toggle smartparens mode	<f11> (M-((smartparens-global-mode &optional ARG)	Toggle Smartparens mode in all buffers. <ul style="list-style-type: none"> With prefix ARG, enable Smartparens-Global mode if ARG is positive; otherwise, disable it. Smartparens mode is enabled in all buffers where ‘turn-on-smartparens-mode’ would do it.
Toggle smartparens-strict mode	<f11> (M-)	(smartparens-global-strict-mode &optional ARG)	Toggle Smartparens-Strict mode in all buffers. <ul style="list-style-type: none"> With prefix ARG, enable Smartparens-Global-Strict mode if ARG is positive; otherwise, disable it. Smartparens-Strict mode is enabled in all buffers where ‘turn-on-smartparens-strict-mode’ would do it.
Smart-shift See also: ☞ Indentation	The smart-shift external package simplifies shifting a complete line or region of lines right or left but also up or down. <ul style="list-style-type: none"> It is implemented as a minor or global minor mode that must be enabled first. You can identify the smart-shift-mode inside one of the pel-<mode>-activates-minor-modes user-options to activate it automatically. You can also use the commands manually or through the key bindings provided by PEL to activate the smart-shift-mode in the current buffer or globally for all buffers. PEL controls it through customization user-options: <ul style="list-style-type: none">  The smart-shift external package  PEL activates it when the pel-use-smart-shift user-option is turned on (set to t).  PEL also provides the pel-smart-shift-keybinding user-option that allows you to select additional alternative key bindings for the smart-shift commands that shift line(s). By default the key bindings are using C-c as a key prefix. With PEL you can also use a control key for the cursor or change the prefix key to use the <f9> key. The 3 possible key bindings are shown below but only one of them will be available at any given time. The one available is the one selected by the user-option value. 		
Toggle smart-shift mode in current buffer	<f11> <tab> s	(smart-shift-mode &optional ARG)	Activate/de-activate the smart-shift mode in the current buffer. <ul style="list-style-type: none"> Activate the line-shift key bindings listed below, in the current buffer. <ul style="list-style-type: none"> With PEL, the actual key binding selected for the line shift commands depend on the value of the pel-smart-shift-keybinding user-option.
Toggle smart-shift mode globally	<f11> <tab> S	(global-smart-shift-mode &optional ARG)	<ul style="list-style-type: none"> Toggle Smart-Shift mode in all buffers. With prefix ARG, enable Global Smart-Shift mode if ARG is positive; otherwise, disable it. Smart-Shift mode is enabled in all buffers where ‘smart-shift-mode-on’ would do it.
Shift line or region right	<ul style="list-style-type: none"> C-c <right> C-c C-<right> <f9> <right> 	(smart-shift-right &optional ARG)	Shift the line or region to the ARG times to the right. <ul style="list-style-type: none">  With PEL one of the extra key bindings can be enabled via the pel-smart-shift-keybinding user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.
Shift line or region left	<ul style="list-style-type: none"> C-c <left> C-c C-<left> <f9> <left> 	(smart-shift-left &optional ARG)	Shift the line or region to the ARG times to the left. <ul style="list-style-type: none">  With PEL one of the extra key bindings can be enabled via the pel-smart-shift-keybinding user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.
Shift line or region up	<ul style="list-style-type: none"> C-c <up> C-c C-<up> <f9> <up> 	(smart-shift-up &optional ARG)	Shift the line or region to the ARG times to the upwards. <ul style="list-style-type: none">  With PEL one of the extra key bindings can be enabled via the pel-smart-shift-keybinding user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.
Shift line or region down	<ul style="list-style-type: none"> C-c <down> C-c C-<down> <f9> <down> 	(smart-shift-down &optional ARG)	Shift the line or region to the ARG times to the downwards <ul style="list-style-type: none">  With PEL one of the extra key bindings can be enabled via the pel-smart-shift-keybinding user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.

YAML & Emacs — References

Description & URL	Notes
YAML	
YAML @ Wikipedia	Overview, syntax, criticisms
YAML official home page	Links to YAML specification, links to various resources and projects. <ul style="list-style-type: none">YAML 1.2 SpecsYAML 1.1 SpecsYAML 1.0 Specs
YAML Resource sites	<ul style="list-style-type: none">Learn YAML in Y MinutesOnline YAML validator (runs yamlint.py) ⚠️ No link as the site is not using https. Instead install yamlint.py locally and use it on the command line or via Emacs.
StrictYAML	A stricter, type-safe YAML
StrictYAML @ Github	
StrictYAML @ hitchdev (Python libraries)	
RAML	RESTful API Modeling Language : RAML files have the .raml file extension.
	<ul style="list-style-type: none">RAML @ WikipediaRAML.orgRAML Spec @ GitHub
Common Workflow Language	Common Workflow Language (CWL) uses a <u>subset of YAML</u> and provides YAML supporting tools.
See also: M CWL	<ul style="list-style-type: none">CWL home page<ul style="list-style-type: none">CWL User GuideCWL YAML Guide
Emacs support for YAML	
yaml-mode (major mode for YAML)	<ul style="list-style-type: none">yaml-mode @ GitHubYaml Mode @ Emacs Wiki
indent-tools	<ul style="list-style-type: none">indent-tools @ GitLabindent-tools @ Melpa
smartparens	The smartparens mode can help deal with data that is within matching pair of characters. <ul style="list-style-type: none">smartparens @ GitHubsmartparens documentation
Emacs/YAML Support Articles	
Blogs about YAML editing on Emacs	<ul style="list-style-type: none">The best ways to work with yaml files in Emacs, from Chmouel Boudjnah's blog, 2016-09-07Editing ansible files in Emacs, from Enis Özgen, 2017-12-29
General blogs about YAML	<ul style="list-style-type: none">10 YAML tips for people who hate YAML<ul style="list-style-type: none">BTW, the last tip is: use something else... well... S-expressions are very flexible and powerful.
Using YAML on Github	YAML is used to control the actions of several Continuous Integration/Development systems. Github is one of them. <ul style="list-style-type: none">Workflow syntax for GitHub Actions