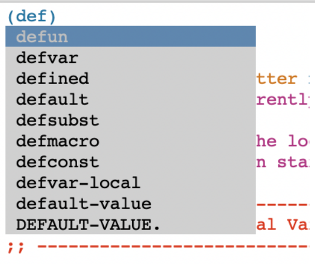
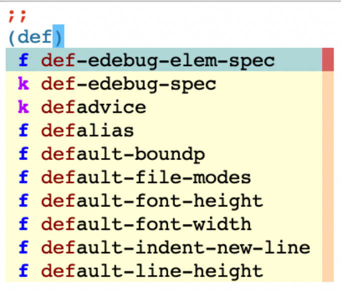
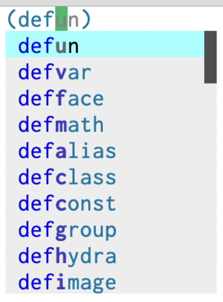
↖ Topic Index

Auto-Completion Support 🚧

Description	Keystroke	Function	Note
Auto Completion ○ Help @ Customization • **Built-in completion** • Show completion mode status • completion-at-point • toggle completion-preview-mode • PEL controlled completion ○ auto-complete • company-mode • corfu ○ Auto-completion Reference	When writing text or source code, Emacs provides support for completing what you type in the buffer: this is called **auto-completion**. • Emacs comes with a basic completion system, accessible via the (**completion-at-point**) command bound to **`C-M-i`** . • PEL supports the following external packages that provide alternative completion at point with pop-up menu:		
	📦 **auto-complete**	**pel-use-auto-complete**	The oldest completion engine.
	📦 **company-mode**	**pel-use-company**	A powerful completion engine.
	📦 **corfu**	**pel-use-corfu**	Use **CO**mpletion in **R**egion **FU**nction. A very powerful and popular auto-completion engine.
	📦 **corfu-terminal**	**pel-use-corfu-terminal**	Use corfu popup in terminal Emacs < 31. PEL activates it when pel-use-corfu is on, but Emacs < 31 and runs in terminal mode.
	• For these PEL supported modes, you can access the customization buffer quickly with the **`<f11> c <f2>`** key sequence. To activate an automatic completion mode for a major mode add one of the following special PEL functions to the list of minor modes listed in the major mode specific instance of:		
☝ Automatic activation of completion modes in specified major modes with PEL: ➠	**pel-MODE-activates-minor-modes**		Add one of the following special PEL functions that activate the corresponding engines: • pel-auto-complete-mode, pel-company-mode, pel-corfu-mode
	• For example, add **completion-preview-mode** and **pel-company-mode** to **pel-elisp-activates-minor-modes** to automatically activate these in all Emacs Lisp buffers.		
	• Emacs >= 30 provides the **completion-preview-mode** which provides an easy-to-use in-buffer completion help that complements the above. ☝ More on abbreviation completion is available in the **Abbreviations** table. Both abbreviation completion and one auto-completion mechanism can be used at the same time (using different keys if any), in some case the abbreviation choices can also be available via auto-completion. ☝ More dynamic completion modes based on **Eglot** or LSP are supported by Emacs but not yet documented in PEL.		
Last updated on:	2025-12-19		
Open this PDF file. See also: **Help/Info**	**`<f11> c <f1>`**	(**pel-help-pdf** &optional OPEN-WEB-PAGE)	Open the **Auto-Completion** 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** user-option is set it's the other way around.
Customize PEL auto-completion support. See also: **Customize**	**`<f11> c <f2>`**	(**pel-customize-pel** &optional OTHER-WINDOW)	Open the PEL customize group(s) for the current context: auto-completion support. Use this to open to change PEL user option variables the activate and control the various Apple script features such as the name of the narrator voice. • When a prefix argument (like **`C-u`**) opens the buffer inside another window.
Customize Emacs built-in auto-completion support See also: **Customize**	**`<f11> c <f3>`**	(**pel-customize-library** &optional OTHER-WINDOW)	Customize Emacs auto-completion group which includes: auto-complete, company, hippie-expand. When prefix arg. (like **`C-u`**) opens the buffer inside another window.
	☝ Group belonging to files that have not yet been loaded are normally not accessible in Emacs and via the customize-group command. PEL, however, attempts to locate the file that defines a non-loaded customization group and will prompt you for loading the file if it finds it.		
Display Auto-completion status	**`<f11> c ?`**	(**pel-completion-info** &optional APPEND)	Print information about available auto-completion info in a *pel-autocomplete-info* **help-mode** buffer. Clear previous buffer content unless a prefix arg (like **`C-u`**) is used.
☝ Provides quick access buttons to change customizable user-options.	• Prints current state and values of relevant user-options as buttons you can use to get more info and change their customized values. • Shows which one is enabled via customization and their current activation state. ☝ Underlines links are buttons that open the customization buffer where you can change the customized value.		
Select auto-completion tool	**`<f11> c =`**	(**pel-select-auto-complete-tool** &GLOBALLY)	Prompt user to select auto-complete tool to use in current buffer. • With prefix arg (such as **`C-u`**) select for all buffers. Select: Emacs built-in bare auto-completion, **auto-complete**, **company-mode**, **corfu**.
Emacs Built-in Completion	Emacs built-in completion is provided by the completion-at-point command, described below.		
Symbol Completion at point See also: **Xref** ☝	• **`C-M-i`** • **`<Esc> <tab>`** • **`M-<tab>`**	(**completion-at-point**)	Perform completion on the text around point. • The completion method is determined by '**completion-at-point-functions**'. • The tags-completion-at-point-function is used for Emacs Lisp code by default. • It provides a list of possible values in the *Completions* buffer.
	`C-M-i` is also used for Flyspell, which can be used to spell check only moments and strings.		
Symbol Completion at point • **and language specific completion**	**`<f6> c`**	(**complete-symbol** ARG)	Perform completion of the text around point, using the method identified by the variable **completion-at-point-functions**, acting as the **completion-at-point** command above. • With prefix argument, such as **`C-u`**, this does completion within the collection of symbols listed in the index of the manual for the language you are using.
Complete language specific symbol at point	**`<f6> C`**	(**info-complete-symbol** &optional MODE)	Perform completion of symbol at point using mode-specific language items. • For example, inside a Emacs-Lisp buffer it finds the Emacs-Lisp functions, variables names.
Toggle Completion Preview mode Emacs >= 30 A minor mode that shows possible completion in a light face. Can be available while another completion major-mode is used.	**`<f11> c p`**	(**completion-preview-mode** &optional ARG)	Toggle completion-preview-mode in current buffer. • Activate it with positive prefix argument, disable it with negative prefix argument.
	`<f11> c P`	(**global-completion-preview-mode** &optional ARG)	Toggle Completion-Preview mode in all buffers. • Activate it with positive prefix argument, disable it with negative prefix argument.
	`<tab>`	Expand suggested completion in buffer. Type more characters to further refine the search.	
	`M-i`	Open a *Completion* buffer listing all possible completion candidates. • Select others with **`M-<up>`** and **`M-<down>`**. Then chose with **`M-RET`**.	
	`M-n`	Selects and show next completion candidate.	
	`M-p`	Selects and show previous completion candidate.	
Completion function - using tags Candidate for: completion-at-point-functions		(**tags-completion-at-point-function**)	Using tags, return a completion table for the text around point. If no tags table is loaded, do nothing and return nil. This uses the tag facility.

Description	Keystroke	Function	Note
Other completion at point engines	PEL dynamically activate either auto-complete-mode, company-mode and corfu for one or all buffers through commands accessible via the **`<f11> c`** prefix. • You must first select witch one should be available via PEL customization: • Set **pel-use-auto-complete** to t to enable the ability to use auto-complete-mode. • Set **pel-use-company** user option to t to enable the ability to use company-mode. • When **pel-init** is executed, the commands made available depend on the section made by customization of PEL but also of the two modes. Then the corresponding commands listed in the sections below are available. PEL's auto-completion support implementation is not yet completed. For now just standard hooks are setup, but not for all programming languages. Support for each programming language is described in the language specific page.		
Explicitly List Completion Candidates with the currently active auto completion system • Use auto-complete or company-mode, whichever that is currently active globally or in the buffer.	• **`<f11> c c`** • **`M-l`**	(pel-complete)	List completion candidates. There must be at least 1 character preceding point. • Force auto-completion of text at point, don't wait for timeout, using the currently active auto-completion system (either auto-complete-mode or company-mode).
	• If no auto completion system is active in the current buffer, the command issues an error. • ✂ **`M-l`** default binding is to downcase-word. PEL rebinds this key company-mode is activated via customization and company-mode is active. • ☝ With PEL, the **`M-l`** key is close to the **`M-/`** key, bound to the command used for abbreviation expansions. It becomes easy to use either.		
Completion Menu keys • **Auto-completion Menu Operations** • **Company-Mode Menu Operations** See also: **Scrolling**	When an completion pop-up menu generated **either** by auto-complete or company-mode is shown, you can use the following keys for operating on that menu: • **`M-n`** : next candidate (or <down> cursor) • **`M-p`** : previous candidate (or <up> cursor) • **`M-1, M-2, M-3,`** etc…: select candidate by line number • **`<tab>`** : complete using 1 candidate (if 1 choice), using the prefix part among many candidates, or cycle through all candidates. • **`<DEL>`** : Delete 1 char of the current candidate prefix • **`<RET>`** : Select current candidate, execute action for candidate if any (eg. when template selection used) • **`C-?`** : Show candidate help in separate buffer • **`<f1>`** : Show candidate help in separate buffer. ☝This is **very** handy to quickly review documentation of several symbols! • **`C-M-v`** : Scroll help buffer forward (note: see the **Scrolling** table for more info on scrolling) • **`Esc <PgDown>`** : Scroll help buffer forward • **`C-M-S-v`** : Scroll help buffer backward • **`Esc <Pg-up>`** : Scroll help buffer backward • **`C-g`** : Stop completion		
auto-complete	Auto-Complete is one of the auto completion package for Emacs supported by PEL. 📦 Requires the **auto-complete** package that PEL supports if the **pel-use-auto-complete** customization variable is set to **t.** Once activated by customization with PEL you can then activate it globally and/or control whether it is available for the current buffer, using the following commands. • ☝The **pel-init** function will install Auto-Complete from the MELPA archive if it is not already present. You may want to use another version (such as the one from MELPA stable). Just install it before customizing to use it and executing **pel-init**. • This is an early version of PEL. Future versions of PEL will integrate logic to support use of Auto-Complete for more programming languages and systems (like templating package). For now PEL only incorporates the basic configuration of Auto-Complete provided by its **ac-config-default** function. • Auto-complete provides the following customizable variables (and several others): • **ac-use-quick-help** : set to **t** to activate a quick pop-up help display that shows right beside the menu choice. • **ac-quick-help-delay** : delay before the quick help pops up. Default is 1.5 seconds. PEL provides access to the auto-complete commands via the commands below.		
	When invoked through the **`<f11> c c`** or **`M-l`** key bindings in a buffer that is set for it, **auto-complete** pops a drop-down menu similar to the one shown here.		 (def) defun defvar defined default defsubst defmacro defconst defvar-local default-value DEFAULT-VALUE.
company-mode	Company-Mode is the other auto completion package supported by PEL. 📦 Requires the **company-mode** external package that PEL activates when the **pel-use-company** customization variable is set to **t.**		
	When invoked through the **`<f11> c c`** or **`M-l`** key bindings in a buffer that is set for it, **company-mode** pops a drop-down menu similar to the one shown here. The menu identifies the type of target.		 ;; (def) f def-edebug-elem-spec k def-edebug-spec k defadvice f defalias f default-boundp f default-file-modes f default-font-height f default-font-width f default-indent-new-line f default-line-height
corfu			
			 (defun) defun defvar defface defmath defalias defclass defconst defgroup defhydra defimage

Auto-completion — References

Document	Note
Basic Auto Completion	**GNU Emacs Manual - Completion for Symbol Names**
Auto Completion with Auto-Complete	
Auto Complete @ MELPA	You can get auto-complete from MELPA. An interesting point of this page lists the other packages that need auto-complete. There's over 45 packages that use it for various programming languages and environments.
Auto Complete @ GitHub	Auto complete source code
Auto Complete Manual @ Github	Covers installation, check, features, concepts, configuration, advanced usage. Reading required for users.
Using Emacs: 8 - Auto-complete @ Youtube	Mike Zamansky video that covers abbreviation and auto-complete. Duration: 5 minutes.
Using Emacs: 45- Company or Autocomplete @ Youtube	Another video from Mike Zamansky that covers both auto-complete and company-mode. Duration: 13 minutes.
Auto Completion with Company-mode	
company-mode ; Modular in-buffer completion framework for Emacs	Text completion framework for emacs
Using digits to select company-mode candidates @ (or emacs irrelevant)	