`0`	`zH`	`^`	`B`	`b`	`h`	`l`	`e`	`E`	`w`	`W`	`zL`	`$`
beginning of line	shift view 1/2 screen left	first non-blank character	previous WORD	previous word	previous character	next character	end of word	end of WORD	beginning of next word	beginning of next WORD	shift view 1/2 screen right	end of line

`:e` path	`:e!`	`:view` path	`:r` file	`:q`	`:q!`
load and edit given file	return to unmodified file	load file in read-only mode	insert contents of file	quit	quit, discard changes

`:w`	`:saveas` *path*	`:browse old`	`:cd`	`:pwd`
write file	save with new name	List recent files, allow to select one by number.	change dir	echo current working dir

`^o`	previous jump location
`gg`	beginning of file
`^b`	back 1 page
`^u`	up 1/2 page
`[[`	previous section / f()
`[{`	opening brace of this function
`{`	previous blank line
`k`	up 1 line
`j`	down 1 line
`}`	next blank line
`]}`	closing brace of this function
`]]`	next section / f()
`^d`	down 1/2 page
`^f`	forward 1 page
`G`	end of file
`8G`	to first non-blank char of line 8
`:8`	start of line 8

`%`	`f`*c*	`F`*c*
jump to matching bracket	jump to next occur of *c*	jump to prev occur of *c*

`zz`	scroll current line to middle of the screen

Buffers

`:ls`	`:b` n	`:b` file
list all buffers	jump to buffer nr n	jump to buffer with containing file
`:bp`	**`:bn`**	**`:bd`**
jump to prev buffer	jump to next buffer	delete current buffer
`:e` path	**`:enew`**	**`:bufdo` c**
load and edit given file	new blank buffer	apply command c to all buffers

Inserting

`I`	`i`	`a`	`A`
beginning of this line	before cursor	after cursor	end of this line
`O`	**`o`**	**`s`**	**`S`**
beginning of previous line	beginning of next line	substitute character	substitute line
`cc`	**`cw`**	**`r`**	**`R`**
change this line	change rest of word	replace 1 character	replace from cursor

Deleting

`x`	`diw`	`dw`	`d/`*foo*
delete character	delete all of word	delete rest of word	delete until first 'foo'
`dd`	**`d0`**	**`d^`**	**`d$`**
delete line	del from BOL to cursor	delete 1st non-blank char to cursor	del from cursor to EOL
`d}`	**`:1,.d`**	**`:.,$d`**	**`dt`*c***
del to end of paragr.	del from line 1 to cursor	del from curs to EOF	delete up to char *c*

Substitute

`:s/foo/bar/g`
In current line replace all (`/g`) 'foo' with 'bar'

`:%s/foo/bar/g`
In all lines replace all 'foo' with 'bar'

`:%s/foo/bar/gic`
In all lines replace all 'foo' with 'bar', case-insensitive, and ask for confirmation

`:3,9s/foo/bar/g`
In lines 3 to 9 (incl) replace all 'foo' with 'bar'

`:.,.+5/^/   /g`
From current line to 5 line further, prepend 3 spaces at the beginning of each line

`:s/\<foo\>/bar/g`
In current line replace all occurrences of 'foo' that are a separate word, with 'bar'

`:5,$s/foo//g`
From current line to last line, replace all 'foo' with nothing (= delete 'foo')

`:%s/\t/    /g`
In all lines, replace tab with 4 spaces

`:%s/red\|pink/blue/g`
In all lines, replace all instances of "red" and "pink" with "blue"

`:'<,'>s/foo/bar/g`
In the visually pre-selected lines, replace foo with bar.

Selecting (visual)

`v`	`V`	`^v`	`gv`
select char	select line	select block	reselect
`d`	**`y`**	**`P`**	**`p`**
delete selection	yank (copy) selection	paste before cursor	paste after cursor
`o`	**`U`**	**`aw`**	**`iw`**
toggle end of selection	turn to uppercase	select word & whitespace	select inner word
`f`*a*	**`/`*foo***	**`25G`**	**`ii`**
select until 1st 'a'	select until 'foo'	select until line 25	select inner indentation

`a", a', a(, a[, a{, at`
`i", i', i(, i[, i{, it`
Version with a: include “” ‘’ () [] {} <\>
Version with i: exclude “” ‘’ () [] {} <\>

Copying

`yy`	`yiw`	`yw`
yank this line	yank inner word	yank rest of word
`y0`	**`y^`**	**`y$`**
yank from BOL to cursor	yank from 1st non-blank char to cursor	yank from cursor to EOL

Undo/Redo

`u` undo	`:earlier 4m`
`^r` redo	`:later 50s`

`:UndotreeToggle`

Search

`*`	`#`
Go to next match of exact current word	Go to prev match of exact current word
`g*`	**`g#`**
Idem as * but not exact	Idem as # but not exact
`/foo`	**`?foo`**
Search forward for foo, not exact	Search backward for foo, not exact
`n`	**`N`**
Go to next match	Go to prev match
`/\<foo\>`	**`:noh`**
Search forward for foo, exact	no search highlights

Move

`:m` *n*	`:m.+2`	`:2,4m $`
mv this line to after line *n*	mv this line 2 lines below	mv lines 2,3,4 to after last line
`:2,4m 9`	**`:'<,'>m'>-5`**	**`:.,.+2m 9`**
mv lines 2,3,4 to after line 9	Move selection to 5 lines before end of selection	mv 3 lines from this line to after line 9

Search & Replace

`/foo`	Search & highlight all 'foo', go to 1st match
`cgn`	Change current search pattern (`gn`) - no ':'
`<esc>`	Get out of edit mode
`.`	Repeat for next match
`n` / `N`	Go to next/prev match if need to skip one

Search/Replace in multiple files

`:vimgrep /foo/g *.py` Search in all files. Use `:copen` to get a list of matches.

`:bufdo %s/foo/bar/ge | update`
Apply to all buffers. Ignore pattern not found errors (`e`). Save file if changes were made (`update`).

`:arg *.cpp` Create arglist with C++ files and load them into buffers
`:argadd *.h` Add header files to arglist, and load into buffers
`:argdo %s/foo/bar/ge | update` Apply to every file in arglist

netrw & Ctrl-p

`:edit|read|write sftp://user@hostname/path`

`:edit` folder	(folder can be .)
`:Vexplore`	split left-right and show file browser
`i`	toggle between file browser view options
/path	move cursor to nearest match in file browser
`^p` *filename*	Fuzzy search file. Use ^f and ^b to toggle between files, buffers, and most recently used. Can be used for "**Open Recent…**"

Global command

`:[range]g/pattern/command`

`:g/foo/d`	Delete all lines that contain 'foo'
`:.,$g!/foo/m$`	Move all lines from cursor to EOF, that do not contain 'foo', to EOF.
`:g/^/m0`	For all lines move them to the beginning of the file = reverse file.
`:g/^foo/s/$/bar`	For all lines starting with 'foo', substitute/append 'bar' to EOL.
`:g/foo/#`	Show with line nrs. Then go to a line.

Case switching

`~`	Toggle case of character under cursor
`g~iw`	Toggle case of the inner word
`g~$`	Toggle case from cursor to EOL
`gUw`	Convert rest of word to large caps
`gUU`	Convert current line to upper case
`gu0`	Convert from BOL to cursor to small caps.

Vimtex

`\ll`	Start/stop continuous compiling mode
`\lv`	Jump to PDF viewer at the same point of the cursor
`\lc`	Clean auxiliary files
`[[`	Jump to the previous (sub)section
`]]`	Jump to the next (sub)section
`%`	Move between matching delimiters

Tags and YCM

`:tag` *name* `:stag`	Jump to (1st) match of the tag. :stag shows it in a new window. *Name* can be a regex (e.g. */Detect**).
`:ts` *name*	List all tag matches, and lets you select one. If name is absent, use list of last :tag command.

`^]`	`g]`	`^o`	`^Space Tab`
Jump to tag under cursor	List all matches of tag under cursor	Go back one jump	In insert mode: show window with f() def.

`:Tagbar` Toggle show/hide tagbar

- Double-click or press enter on a tag to jump
- Press space on a function to display arg list.

`:!ctags -R --extra=+f` Create tags file, recursively, in *current* dir

Windows

`:split`	`:vsplit`	`:only`	`:close`
split top-bottom	split left-right	keep only current window	close current

Indenting

`>>`	`<<`	`>`	`<`	`>ap`	`>aB`
ident current line	un-indent current line	indent selected lines	un-indent selected lines	indent current parag.	indent current block

CSV

`:WhatColumn`	Show current column nr
`:nrColumns`	Show total numbers of columns
`:DeleteColumn 2`	Delete column nr 2
`:MoveColumn 2 4`	Move col 2 to right next to col 4
`:DupColumn`	Duplicate current column
`:Header 2`	Open separate fixed window with first 2 header lines. Close with :Header!
`:%ArrangeColumn!`	Prettify the columns by aligning them
`:NewDelimiter ;`	Change the delimiter to ;
`:1,10Sort 3`	Sort lines 1-10 of column 3

`:SearchInColumn 1 /foo/`
Search for patter foo in column 1

`:%Substitute 1,4/foo/bar/gc`
Subst in cols 1-4 foo with bar, globally + ask for confirmation

Fugitive

`:Gblame`	Show git blame info for an indexed file.
`:Gedit SHA`	Show info for blob, tree, commits, and tags
`:Gedit master:python/transits.py`	Read-only!
`:Gdiff`	Shows differences between working copy of the current file and the index, in a split window. **:q** to exit.
`[c`	Previous hunk
`]c`	Next hunk
`:Gdiffsplit!`	In case of merge conflicts: open 3-way diffs

E.g. if you have conflicts after:

```
$ git checkout master
$ git merge feature
```

Left/center/right window: master / current working copy / feature

Keep your cursor in the center window (working copy).

`:diffget //2`	include left version in working copy
`:diffget //3`	include right version in working copy
`:diffupdate`	update the diff highlighting

fzf

`:Files`	Fuzzy search files
`:Lines`	Fuzzy grep in all open buffers
`:BLines`	Fuzzy grep in current buffer
`:Tags`	Fuzzy search all tags
`:BTags`	Fuzzy search tags in current buffer
`:GFiles?`	Git status
`:Commits`	Fuzzy search in all commits
`:BCommits`	Fuzzy search in commits of this buffer

Special Text Objects

`gn`	Current search pattern	`ii`	Inner indentation level, no line above
`aa`	Function arg under cursor, with comma	`ai`	Indentation level + one line above (*Python*)
`ia`	Function arg under cursor, without comma (*inner*)	`aI`	Indentation level + line above + line below (*C++*)

Example in C++ : `caI` : change current indentation level incl. { } above/below

Tabularize

`:Tab/=`	Align selection on the '=' char

`:Tab/,/r1c1l0`

First part right aligned, then 1 space, then comma center aligned, then 1 space, then part left aligned, no extra space, then rest

Miscellaneous

`^a / ^x`	Incr/decr the nr under cursor.
`J`	Join this line with the next one
`:set scrollbind`	Scroll synchronously. Set in all windows that need to be in-sync. Add ! to to undo.
`:ab fe for example`	Replace abbreviation on the fly
:set list	Show all the invisible characters
:set nolist	Stop showing all the invisible characters
^oo	After startup: load last edited file
^n	In insert mode: word completion
:Matrix	Follow the white rabbit
:ToggleWhitespace	(Un)highlight trailing whitespace
:StripWhitespace	Strip trailing whitespace
:set wrap linebreak	Wrap text around
:%s/ ctrl v ctrl m/\r/g	Convert dos ^M to unix carriage return
gv	Reselect last selection
Vr-	Select line, replace all chars with `-`

Folds

`zo/zc`	`zO/zC`	`zm/zr`	`zj/zk`
Open/close 1 fold under cursor	Open/close all folds under cursor	Increase /reduce fold level	Go to start of next/previous fold

Markers

`m a`	Set marker named 'a' under cursor
`m a!`	Same but override previous marker 'a'
`'a`	Go to start of line containing marker 'a'
`` `a ``	Jump to marker 'a' (back quote)
`:marks`	Show a list of markers
`:delm ab / !`	Delete marker 'a' and 'b' / all markers
`` :c `a ``	Change from cursor to marker 'a'
`:]' / ['`	Jump to next/prev line with marker
`` `. ``	Special marker: last change in buffer

Macros

`qa`	Start recording to register 'a'	`@a`	Execute macro
…	Fancy commands	`5@a`	Execute 5 times
`q`	Stop recording	`:reg`	List all macros

Repeating

`n`	Repeat search - forward	`;`	Rpt last `f c` - forward
`N`	Repeat search - backward	`,`	Rpt last `f c` - backward
`@:`	Rpt last ':' command	`&`	Rpt last :s/x/y/ subst
`.`	Rpt last change in Normal mode		

Swapping

`cx{motion}`	Select the first part of text to be swapped…
`.`	… go to the 2nd part of text. Repeat cmd: swaps
`cxx`	Like cx for the current line
`cxc`	Clear selection and memory

(Un)commenting

Old School:

`^v`	Block select
select	Select the block
`I`	Insert mode at beginning
//	Write comment symbols
<esc>	Exit & apply to all lines

`^v`	Block select
select	Select comment symbols
`d`	delete

With tcomment:

`gc{motion}`	Toggle comment. E.g `gc4j` or `gcaB`
`gcc`	Toggle comment for this line.

Surround

ys *{motion}{delimiter}*	Surround motion by given delimiter
yss *{delimiter}*	Surround current line by delimiter
ds *{delimiter}*	Delete surrounding delimiter
cs *{old delim}{new delim}*	Change surrounding delimiters
cst *{new delim}*	Change surrounding tags to delimiters

Easymotion

`\\w`	Word motion, then select appropriate letter. Forward.
`\\b`	Idem but backward
`\\s` *char*	Trigger character motion, then select letter.

MacVim

`⌘= / ⌘-`	Increase/decrease font size

Insert Mode

^o	In INSERT mode: go back to normal mode for 1 command
^r=*<expr>*	In Insert mode: compute and insert result of *expression*

Clipboard

`:w !pbcopy`	Write selected lines to clipboard (visual mode)
`:r !pbpaste`	Insert contents of clipboard at current position