:h operator :h navigation

[operator][count][motion]

d delete/cut

y yank/copy

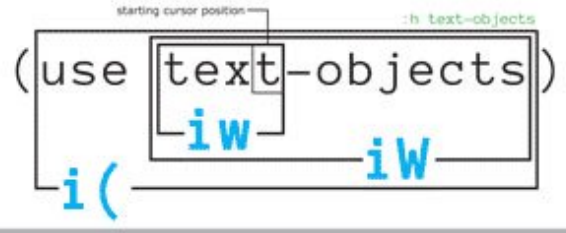
c change

gU make uppercase · ~ swap case · < shift left · = indent

Any motion can follow an operator. Marks and searches count as motions, too! `d/foo` will delete from the cursor to the next instance of "foo". `y3fi` will yank from the cursor to the 3rd "i" on the line after it. Counts can also come before operators: `5dd` will delete five lines.

w	word
W	WORD
s	sentence
[,]	[ ] block
(,)	( ) block
< , >	< > block
t	XML/HTML tag
{ , }	{ } block
" , '	quoted string

:h text-objects



:h up-down-motions

gg first line

^b up 1 page

^u up ½ page

k up 1 line

j down 1 line

^d down ½ page

^f down 1 page

beginning of line **0** · first non-blank character **^** · previous WORD **B** · previous word **b** · previous character **h**

next character **l** · end of word **e** · beginning of next word **w** · end of WORD **E** · beginning of next WORD **W** · end of line

	ts	sw	sts	et
use spaces only	*n*	*n*	*n*	on
use tabs only	*n*	*n*	0	off

Set *n* to desired tab width (default 8)

tabstop	ts	Columns per tabstop
shiftwidth	sw	Columns per <<
softtabstop	sts	Spaces per tab
expandtab	et	<Tab> inserts spaces

MIXING TABS AND SPACES IS RIGHT OUT
(that means don't do it.

:retab Replace all tabs with spaces according to current tabstop setting

fileformat	ff	Try changing this if your line-endings are messed up
list		Display whitespace visibly according to listchars

:h left-right-m

:h pattern-searches

SEARCHING

Prev	Next	Forward	Backward	Matches
N	n	/*foo*	?*foo*	*foo*
		*	#	word under cursor
;	,	t*x*	T*x*	upto *x*
		f*x*	F*x*	find *x*

:h mark-motions

m*m*	set mark *m* (a-z) in file	m*M*	set mark *M* (A-Z) across files	'[	jump to first char of just-changed text

p	paste after cursor	P	paste before cursor	^[	return to Normal mode
u	undo	^r	redo	.	repeat
gf	find file under cursor in path and jump to it	dd	delete current line	yy	yank current line
x	delete character after cursor	%	jump to matching paren	r	replace char under cursor
*n*G	jump to line *n*	^o	jump back	^i	jump forward
zz	center screen on cursor	zt	align top of screen with cursor	zb	align bottom of screen with cursor

'm jump to first char of line containing m

`m jump to exact character of m

'' jump back to last jump

G last line

== auto-indent current line

<< shift current line left by shiftwidth

>> shift current line right by shiftwidth

Pass a directory to the :edit command to open a directory explorer.
Instructions for usage are at the top of the screen.

Using ^[to return to Normal mode lets you keep your fingers on the home row. It's even easier if you map Caps Lock to Control!

:h insert.txt

ENTERING INSERT MODE

beginning of line I

before cursor i

after cursor a

end of line A

previous line O

next line o

substitute character s

substitute line S

line from cursor C

COOL INSERT MODE STUFF

^w delete word before cursor

^u delete line before cursor

^rr insert the contents of register r

^r= use the expression register (try ^r=5+10)

^t increase line indent by shiftwidth

^d decrease line indent by shiftwidth

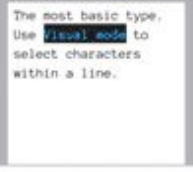
^x^l line completion

^n find next completion suggestion according to complete

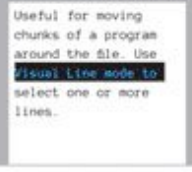
:h cmdline.txt

ENTERING VISUAL (SELECT) MODE

v



V



^v

switch cursor to start/end o
:h v_o

re-select previous area gv
:h gv

prepend to each Visual block line I
:h v_b_I

jump to start of prior area '<
:h '<

COMMAND-LINE MODE ONLY

edit using Normal mode ^f
cmdwin

insert word under cursor ^r^w
cmdline-editing

completion suggestions ^d
cmdline-completion

Put cnoremap %% <C-R>=expand('%:h').'/'<CR> in your .vimrc so you can type %% in Command-line mode to refer to the directory of the current file, regardless of pwd.

ZZ Write current file, if modified, and quit

ZQ Quit without checking for changes (like :q!)

:write Write current file

:wq Write current file and quit

Use :scriptnames to list all files sourced during initialization.

:syntax Enable and configure syntax highlighting
Use :sy sync fromstart to redraw broken highlights

:h quickfix

:make Run a compiler and enter quickfix mode

:! Execute external shell command

! Filter motion with shell command

Use :earlier and :later to quickly jump backward and forward in a file's history

Supply % as a range to the :substitute command to run it on every line in the file.

:%s/Scribbl/Design/ "Scribbled" -> "Designed"

Specify the "g" flag to apply the substitution to *every* match on a line.

:s/[dla]//g "badly" -> "by"

:h s_flags, :h /[

Vim supports many regular expression features.

:s/..k/ax/ "Mook" -> "Max"

:h usr_27, :h /

Use _. instead of . if you want to search across multiple lines.

:%s/heat_.*Bungle/anto/ "Cheatsheet\nBungler" -> "Cantor"

:h /_

Special escapes can be used to change the case of substitutions.

:s_\(f..\)_\U\1\E_ "foobar" -> "FOObar"

:h sub-replace-specia

Use :global to perform a command on matching lines.

:g/foobar/delete Delete all lines containing "foobar"

If your pattern contains slashes, just use a different character as your delimiter.

Use `:earlier` and `:later` to quickly jump backward and forward in a file's history.

:read

Read external program output into current file

```
:s_Data/Lore_Brent Spiner_
```

Use `\=` to evaluate expressions with replacement groups.

```
:s_\d_\=submatch(0) + 1_g      "10 25" -> "21 36"      :h sub-replace-\
```