1 General

Nearly all commands can be preceded by a number for a repeat count. eg. 5dd delete 5 lines <Esc> gets you out of any mode and back to command mode Commands preceded by : are executed on the command line at the bottom of the screen :help help with any command

2 Navigation

2.1 By words:

- w next word (by punctuation); W next Word (by spaces)
- b back word (by punctuation); B back Word (by spaces)
- e end word (by punctuation); E end Word (by spaces)

2.2 By line:

- 0 start of line; ^first non-whitespace
- $ end of line

2.3 By Sentence

Parenthesis!
(: to the end of the next sentence
): to the beginning of the next sentence.

2.4 By paragraph:

- { previous blank line;
- } next blank line

2.5 By file:

- gg start of file; G end of file
- 123G go to specific line number

2.6 By marker:

- mx set mark x; 'x go to mark x
- '. go to position of last edit
- ' ' go back to last point before jump

2.7 return to last position

^o to return to last location

2.8 Scrolling:

- ^F forward full screen; ^B backward full screen
- ^D down half screen; ^U up half screen
- ^E scroll one line up; ^Y scroll one line down
- zz centre cursor line

3 Text Objects

In Vim, editing commands have the following structure:

```
<number><command><text object or motion>
```

Objects will be listed here, commands in their own sections.

- aw/iw (a word/inner word)
- as/is (a sentence/inner sentence)
- ap/ip (a paragraph/inner paragraph)
- a“/i” (a double quoted string/inside double quoted string)
- a’/i’
- a‘/i‘
- a(/i(
- a{/i{
- a[/i[

3.1 vim-indent-object

Text Objects from the package vim-indent-object:

- ai/ii (This indent level and the line above/this indent level excluding the line above)

3.2 vim-pythonsense

- “ac” : Outer class text object. This includes the entire class, including the header (class name declaration) and decorators, the class body, as well as a blank line if this is given after the class definition.
- “ic” : Inner class text object. This is the class body only, thus excluding the class name declaration line and any blank lines after the class definition.
- “af” : Outer function text object. This includes the entire function, including the header (function name declaration) and decorators, the function body, as well as a blank line if this is given after the function definition.
- “if” : Inner function text object. This is the function body only, thus excluding the function name declaration line and any blank lines after the function definition.
- “ad” : Outer docstring text object.
- “id” : Inner docstring text object.

4 Editing

- u undo; ^R redo
- . repeat last editing command

5 Inserting

All insertion commands are terminated with <Esc> to return to command mode.

- i insert text at cursor; I insert text at start of line
- a append text after cursor; A append text after end of line
- o open new line below; O open new line above

6 Changing

All change commands except r and R are terminated with <Esc> to return to command mode.

- r replace single character; R replace multiple characters
- s change single character
- cw change word; C change to end of line; cc change whole line

- c<motion> changes text in the direction of the motion
- ci(change inside parentheses (see text object selection for more examples)

7 Deleting

- x delete char
- dw delete word; D delete to end of line; dd delete whole line
- d<motion> deletes in the direction of the motion

8 Cut and paste

- yy copy line into paste buffer; dd cut line into paste buffer
- p paste buffer below cursor line; P paste buffer above cursor line
- xp swap two characters (x to delete one character, then p to put it back after the cursor position)

9 Commenting

This is from vim-commentary:
Comment stuff out. Use gcc to comment out a line (takes a count), gc to comment out the target of a motion (for example, gcap to comment out a paragraph), gc in visual mode to comment out the selection, and gc in operator pending mode to target a comment. You can also use it as a command, either with a range like :7,17Commentary, or as part of a :global invocation like with :g/TODO/Commentary.

9.1 Re-flowing text

gq reflows text

10 ArgWrap

ArgWrap, switches between horizontally and vertically formatted arguments

11 Blocks

- v visual block stream; V visual block line; ^V visual block column
  most motion commands extend the block to the new cursor position o moves the cursor to the other end of the block

- d or x cut block into paste buffer
- y copy block into paste buffer
- > indent block; < unindent block
- gv reselect last visual block

12 Global

:%s/foo/bar/g substitute all occurrences of “foo” to “bar”
% is a range that indicates every line in the file
/g is a flag that changes all occurrences on a line instead of just the first one

13 Searching

- / search forward; ? search backward
- * search forward for word under cursor; # search backward for word under cursor
- n next match in same direction; N next match in opposite direction
- fx forward to next character x; Fx backward to previous character x
- ; move again to same character in same direction; , move again to same character in opposite direction

14 Files

- :w write file to disk
- :w name write file to disk as name
- ZZ write file to disk and quit
- :n edit a new file; :n! edit a new file without saving current changes
- :q quit editing a file; :q! quit editing without saving changes
- :e edit same file again (if changed outside vim)
- :e . directory explorer

15 Windows

- ^Wn new window
- ^Wj down to next window; ^Wk up to previous window
- ^W_ maximise current window; ^W= make all windows equal size
- ^W+ increase window size; ^W- decrease window size

16 Source Navigation

- % jump to matching parenthesis/bracket/brace, or language block if language module loaded
- gd go to definition of local symbol under cursor; ^O return to previous position
- ^] jump to definition of global symbol (requires tags file); ^T return to previous position (arbitrary stack of positions maintained)
- ^N (in insert mode) automatic word completion

17 Macros

To enter a macro, type:

```
q<letter><commands>q
```

To execute the macro <number> times (once by default), type:

```
<number>@<letter>
```

18 YouCompleteMe

- GoTo
- GoToReferences
- GetDoc