+ getting help +  ctrl+g I'll do more You will actually see where this file is located on your Local system

:help ↵
:q ↵ // quits help

:help dd ↵ // the help for the dd command.

:help count ↵ // [count] An optional number that may
// precede the command to multiply or
// iterate the command.

Ctrl-O ↵ // go back to the previous section you were at
// Like the back button in web browser

Ctrl-i ↵ // Jump forward
// Like the forwod button in web browser

:help ; linewise ↵ // linewise. characterwise the operation either affects
// whole lines, or the characters between the start and
// end position.

:help :q ↵ // :q[uit] short to :quit

:help :help ↵ // :h[elp] short :h // shortcut
// open a window and display the help file in read-only
// mode

when you start the help system vim opens up a new window if you want to keep the help window open while editing & navigating through your file type ctrl+w ctrl+w
↳ the cursor will jump to the window at the bottom of your screen which is the file that you opened at beginning of this lesson. to go back to help ctrl w & ctrl w

ctrl = w + w

Command line completion and to do that will type colon h
Followed by the start of the cmd
:h :q ctrl d => much the cmd we deal with
=> :q vim will will all cmd start with :q
:qa
~~:quit~~
Also tab do the same

---

'wildmenu' 'wmnu' boolean (default off, set in defaults.vim)
'wildmenu' 'wmnu' :nowildmenu 'nowmnu'
:set 'wildmenu'

---

In Vim buffer = file

CTRL-F <Page Down> CTRL-F
:h ^f
ctrl
:h ^b <Page Up> CTRL-B
ctrl

~~:bw~~

CTRL [ => to jump to doc for what under cursor
close-bracket
go back
CTRL O

CTRL G
:h ^g

CTRL-G
:f [!]

CTRL-G :f :fi :fil
Print the current file name (as typed, unless ":cd" was used), the cursor position (unless the 'ruler' option is set) --

# Cut, Copy & Paste

- Move text around a file.
- Duplicat text
- Make an in-file backup
- Reuse the same text

** You've already been cutting
- d and x cut text, not just delete it.
- Cut = delete and save into a register
- Register is a clipboard-like storage location.

Cut = delete ⇒ deleted and save into reusable place in memory all at the same time

---

cut the line with dd CMD
↳ this place has the cut text in what vim calls the unnamed register = default register

press lower case p ⇒ that text is placed on the line below where your cursor is
↳ vim refers to the p CMD as the put CMD
PASTE CMD

---

let's SWAP the next two lines that contain text in the file

DD // cut
p // past
} ⇒ ddp // swap CMD

p, P
p ⇒ put text after the cursor
P ⇒ put text before your cursor

⇒ Even after your past the text it stays in the unnamed register until it is overwritten by another operation

dd o shift + p i esp esc

Now let's say that you just want to copy text and not cut it to do that
⇒ use the y operator which in vim terminology stands for Yank
↳ more accurately text is being yanked into a register.

Standard vs Vim
Cut = delete
copy = yank
past = put

So you're already familiar with the operator motion pattern using it to delete words and so on.
↳ You can use the same pattern with y operator so to yank or copy a word you would use yw and to yank to the end of the line → use y$

2yw shift + p

yy yank entire line
dd deletes ~~cut~~ an entire line

yyp

Yank multiple lines
4 yy yank 4 line into the register
shift + p

=

undo and ~~redo~~ repeat

undo ⇒ u CMD ↵ ⟺ Totally undo the last CMD no matter how many lines or character it affects
redo ⇒ ctrl + r ↵

++ Register types ++
- Unnamed - default = ""
- Numbered = "0 "1 ... "9
- Named
- The black hole = "_

registers preceded with "" the unnamed register is actually double quote the double quote

The ~~numbered~~

→ the unnamed register contains the last bit of text from an operation like delete or yank as you've already seen

Registers

"" holds text from d, c, s, x and y operations
"0 holds last text yanked (y)
"1 holds last text deleted (d) or changed (c).

with each successive deletion or change vim shifts the previous contents of register one into register two and register 2 to 3 and so forth losing the previous contents of register 9 it falls off

to look at the registers we can type
:reg ↵