vim notes

## movement and editing

* use w and e to move by words instead of characters. w moves to the beginning of the next word and e to the end of the next word
* moving the screen instead of the cursor can be useful.
  + ctrl+ E to go down
  + ctrl + Y to go up
* H M and L move the cursor to the top, the middle and the bottom of the window(not file, window)
* I and A moves to the beginning and end of the line and puts you into insert mode
* f to find a char and move the cursor to it, super useful when moving inside a line or something. F to do the same thing backwards.
* you can set marks with m[someletter] and then use that as part of motions like y`k will yank upto the mark k.
  + using ' instead of ` goes the first col in the marked line
* g; puts you at your last edited location
* ctrl + o and ctrl + i moves in and out of the vim editing stack use it to move forward and backwards in edits
* move down and up visual (wrapped) lines with gj and gk.
* =i{ (in normal mode) indents everything inside a {} to the same level
* :marks lists all marks inside a file
* you can create global bookmarks (across files) using upper case letters for marks
* { and } allow you to move up and down by blank lines
* :x is exactly the same as :wq
* zz puts current line in the middle
* use ctrl + o in insert mode to go into normal mode for one command only. Very useful to delete words etc.
* ctrl + t in insert mode indents the current line ctrl + d to unindent
* gu | gU | g~(toggle) with motions can be used to mess with the case
* use u,U or ~ to mess with the case of a visual selection
* use ctrl + x and ctrl + a to decrement | increment numbers under the cursor
* in insert mode simply press ctrl + r and then a register name to paste from that register
* :.! puts the result in the current window
* you can paste the last inserted text in insert mode using ctrl+a
* vim has builtin support for text expansion, just say :ab x y to define an abreiviation, x will be replaced with y as soon as you type it
* gq{motion} formats the selection according to the formatexpr, for insantce you can map autopep8 as formatexpr for python files and gq can be used to pep8-ify a seleciton.
* ctrl + c puts you in normal mode when in insert mode.
* always use set so=999. Keeps the cursorline centered at all times.
* always install the vim-repeat plugin by tpope. It supercharges the . repeat function.
* use K to summon the man page for the word under the cursor in normal mode.
* surround.vim can be used in visual mode. S followed by surround char will surround a visual selection.
* when in insert mode use shift + enter to enter a line above the current line and stay exactly where you are.
* use ctrl + x ctrl + f to enter file completion mode.
* use set cursorbind to move cursors in multiple splits at once.

## folding

* zf and zd to create delete code folds. Useful to fold out imports or some other code noise.
* zo and zc to open and close folds.
* set foldmethod=indent folds the entire document by indent levels.
* set foldmethod=syntax to fold my language syntax (this isn't always what i am looking for though)
* zM fold everything
* zR unfold everything (recursively)
* zm and zr fold / unfold one indnet level at a time

## searching

# /\* searches for the word the cursor is on (forwards/backwards), then you can n/N through the results

* once you do a f F t T you can use ; or , to repeat the last one (in forward and reverse direction respectively)
* to repeat the last search just hit n/N.
* use :g//# to show all lines containing the pattern along with their line numbers

## visual mode

* use o to move to the other end of the visual block
* use gv to reselect the last section

## misc

* in any bash type set -o vi to get vi style bindings
* :shell gets you an interactive shell instance from vim direclty. You can quit it by saying ctrl+d
* in command mode press ctrl + r followed by ctrl + w to paste the word that is below the cursor
* when opening large files switch to absolute line numberings since relative/hybrid line numbering can cause vim to lag,
* it is possible to give some starting commands to vim when launching it from the command line withe the -c flag. The example below will start vim with the list option enabled. vim -c ":set list"
* use g?{motion} to obscure the text of a document
* use ctrl+w [count] + or - to decrease or increase the size of the current window up and down and < or > for left and right
* use ctrl + w = to make all the windows the same size.
* in command mode type :py3 followed by a python snippet to execute it inline, good for small calculations and such
* use :w !diff % - to see changes since last save.
* when in bash/zsh use ctrl+x ctrl+e to edit the current command in vim
* when it comes to executing code from within vim, the vim-slime plugin seems to be the way to go. It is just slinging text back and forth between tmux panes, an easy to understand model that is very flexible and keeps the focus on vim.