THE UNIVERSITY OF WESTERN ONTARIO

Epic Vim Tutorial

January 19, 2012

Author:[1] Jeff Shantz

[1]Heavily adapted from Vim Tutor 1.7, written by Michael C. Pierce, Robert K. Ware, Charles Smith, and Bram Moolenaar

Contents

1 Introduction to Vim

Vim is a very powerful editor that has many commands – too many to explain in a single lab. This lab is designed to describe enough of the commands that you will be able to easily use Vim as an all-purpose editor.

It is important to remember that this lab is set up to teach by use. That means that you need to execute the commands to learn them properly. If you only read the text, you will forget the commands!

Prerequisites

Before beginning this lab, please perform the following tasks:

1. SSH into `obelix`
2. Create the directory displayed below **without** running `mkdir` more than once. To do this, pass the `-p` option to `mkdir` to create parent directories, as needed. See the manpage for `mkdir` for more details

   ```
   ~/courses/shared
   ```

3. Change to the previous directory by typing the following. The argument `!$` is a shortcut that references the last argument in the previous command typed

   ```
   cd !$
   ```

4. Using `git`, clone the Epic Vim Tutorial repository to download the exercise files:

   ```
   git clone git://github.com/jsuwo/Epic-Vim-Tutorial.git
   ```

5. List the contents of the directory. You should now have an `Epic-Vim-Tutorial` directory. Change to this directory

   ```
   ls
   cd Epic-Vim-Tutorial
   ```

Lesson 1.1 Moving the Cursor

In this lesson, you'll learn how to move around in vim. There are two ways to do so. In some cases, you may be able to use the arrow keys. However, if you are using the older `vi` editor, the arrow keys may not work. Hence, it is useful to know the other way to move around:

```
      ^
      k            Hint:  The h key is at the left and moves left.
< h     l >               The l key is at the right and moves right.
      j                  The j key looks like a down arrow.
      v                  The k key is, well, just the k key, and moves up.
```

Many vi/vim users claim that by using `hjkl`, you will be able to move around much faster, once you get used to it. Really!

---

1. Open the file `lesson1` in Vim by typing:

   ```
   vim lesson1
   ```

2. Without using the arrow keys, move the cursor to the start of the text that reads `MOVE HERE FIRST`
3. Without using the arrow keys, move the cursor to the start of the text that reads `MOVE HERE NEXT`
4. Without using the arrow keys, move the cursor to the start of the text that reads `THEN MOVE HERE`
5. Press `i` to enter *insert mode*. Try using the `hjkl` keys to move around. Notice that they don't work. This is because these keys are only used in *normal mode*. Press `Esc` to return to *normal mode*.
6. Try using the `hjkl` keys again to move around. Notice that they once again work
7. Press `i` to enter *insert mode*. Try using the arrow keys to move around. Notice again that they don't work
8. Type a word and try using the backspace key. Notice that it doesn't work properly. We will see how to fix these problems shortly

Lesson 1.2 Exiting vim

Type `:q` to quit.
Type `:q!` to quit without saving changes.

1. Press `Esc` to return to *normal mode*. To enter a command in vim, you must always be in normal mode
2. To quit vim, type `:q`
3. Observe that you are prevented from exiting, since there are unsaved changes. To save changes, you can use `:wq`, which means *write and quit*. To quit without saving your changes, you can use `:q!`, which tells vim to exit immediately
4. Exit vim without saving your changes
5. Open the file `lesson1` again in vim
6. Press `i` to enter *insert mode*. Type any word of your choosing
7. Exit vim, but this time save your changes

Lesson 1.3 Fixing vim

vim is a powerful editor, but it's not very useful that we can't use the arrow keys in insert mode. It would also be nice to see some colour when we're editing code, and line numbers would be great, too. Fortunately, your faithful TA has created a shell script that will fix up your vim editor on GAUL so that your arrow, PgUp/PgDn, and Backspace keys work properly, and your code is syntax-highlighted with numbered lines. We won't concern ourselves with how this script works – instead, we'll just run it and enjoy the result.

1. Open the file `lesson1-3.c` in vim
2. Notice that it is bland and dreary, lacking line numbers, syntax-highlighting, and, if you will, a certain... *je ne sais quoi*
3. Close vim. Let's fix things up and put some spunk into our text editing. Get excited. I know I am
4. Move the file fixvim.sh to your home directory

   ```
   mv fixvim.sh ~
   ```

5. Change to your home directory

   ```
   cd
   ```

6. List the contents of the directory. You should now see a script `fixvim.sh`
7. Make the script executable

   ```
   chmod +x fixvim.sh
   ```

8. Run the `fixvim.sh` script. Remember: get in the habit of typing `./fixvim.sh` instead of `fixvim.sh` since the current directory is not in the `$PATH` on most systems
9. Log out and SSH back into `obelix`
10. Change back to the `courses/shared` directory
11. Reopen the file `lesson1-3.c` in vim. Marvel at its beauty. Praise your instructor / TA. Consider buying them a Diet Pepsi, with lime
12. Press `i` to enter *insert mode*. Try using the arrow keys, PgUp/PgDown, and backspace keys. Marvel at their utility. Praise your instructor / TA. Consider buying them an Amazon.com gift card
13. Exit vim without saving any changes

Lesson 1.4 Text Editing - Deletion

Press `x` in normal mode to delete the character under the cursor.

1. Open the file `lesson1`
2. Move the cursor to the line marked

   ```
   LESSON 1.4 -->
   ```

3. To fix the first error, move the cursor until it is on top of the character to be deleted
4. Press the `x` key to delete the unwanted character.
5. Repeat until the sentence is correct

Lesson 1.5 Text Editing - Insertion

Press `i` in normal mode to insert text.

1. Move the cursor to the line marked

   ```
   LESSON 1.5 -->
   ```

2. To make the first line the same as the second, move the cursor on top of the first character AFTER where the text is to be inserted
3. Press `i` and type in the necessary additions
4. Repeat until the sentence is correct

Lesson 1.6 Text Editing - Appending

Press `a` to append text next to the current cursor position.
Press `A` to append text at the end of the line.

1. Move the cursor to the first line marked

   ```
   LESSON 1.6 -->
   ```

2. Move the cursor so that it sits upon the `e` in `te`
3. Press `a` and type in the necessary addition
4. Press `Esc` to return to *normal mode*. Move the cursor to the second line marked

   ```
   LESSON 1.6 -->
   ```

   It does not matter on what character the cursor is located.
5. Press `A` and type in the necessary additions
6. Press `Esc` to return to *normal mode*.

NOTE: `i`, `a`, and `A` all go to the same *insert mode*; the only difference is where the characters are inserted.

Lesson 1.7 Saving / Reopening

Press `:w filename` to save to a new file named `filename`.
Press `:qw filename` to save to a new file named `filename` and exit.
Press `:e filename` to open `filename`.
Press `:e! filename` to open `filename` without saving any current changes.
Press `:e` to reopen the current file.
Press `:e!` to reopen the current file without saving any current changes.

1. Save the current file as `lesson1.done` using the appropriate `:w` command above
2. Open the file `lesson1` using the appropriate `:e` command above
3. Press `i` to enter *insert mode*. Add a few words to the file
4. Press `Esc` to return to *normal mode*. Type `:e` to reload the current file
5. Notice that you are prevented from doing so, since there are unsaved changes. Use `:e!` to force the reload of the current file
6. Press `i` to enter *insert mode*. Add a few words to the file
7. Press `Esc` to return to *normal mode*. Type `:e lesson1.done` to load `lesson1.done`
8. Notice that you are prevented from doing so, since there are unsaved changes. Use `:e! lesson1.done` to force the loading of `lesson1.done`
9. Exit, saving the current file as `lesson1.final`

2 Moving Around and Deleting Things

In this section, you'll learn ways to move around more efficiently in vim, along with various methods for deleting characters, words, and lines. You'll also take a look at vim's undo and redo functionality.

Lesson 2.1 Deletion Commands

Type `dw` to delete a word.

1. Open the file `lesson2` in vim
2. Move the cursor to the line marked

   ```
   LESSON 2.1 -->
   ```

3. Move the cursor to the beginning of a word that needs to be deleted
4. Type `dw` to make the word disappear

   NOTE: The letter `d` will appear on the last line of the screen as you type it. Vim is waiting for you to type `w` . If you see another character than `d`, then you typed something wrong; press `Esc` and start over.

5. Repeat until the sentence is correct

Lesson 2.2 More Deletion Commands

Type `d$` to delete to the end of the line.

1. Move the cursor to the line marked

   ```
   LESSON 2.2 -->
   ```

2. Move the cursor to the end of the correct sentence (AFTER the first `.`)
3. Type `d$` to delete to the end of the line

We'll see what is happening here in the next lesson.

Lesson 2.3 On Operators and Motions

Many commands that change text are made from an operator and a motion. The format for a delete command with the `d` delete operator is as follows:

```
d motion
```

where:

- `d` is the delete operator
- `motion` is what the operator will operate on (listed below)

A short list of motions:

- `w` - until the start of the next word, EXCLUDING its first character
- `e` - to the end of the current word, INCLUDING the last character
- `$` - to the end of the line, INCLUDING the last character

Thus, typing `de` will delete from the cursor to the end of the word.

NOTE: Pressing just the motion while in normal mode without an operator will move the cursor as specified. So, pressing `e` will move to the end of the current word, while press `$` will move to the end of the line.

Lesson 2.4 Using a Count for a Motion

Typing a number before a motion repeats it that many times.

1. Move the cursor to the line marked
   ```
   LESSON 2.4 -->
   ```
2. Type `2w` to move the cursor two words forward
3. Type `3e` to move the cursor to the end of the third word forward
4. Type `0` (zero) to move to the start of the line
5. Repeat these steps with different numbers

Lesson 2.5 Using a Count to Delete More

Typing a number with an operator repeats it that many times.

In the combination of the delete operator and a motion mentioned above, you insert a count before typing `d` to delete more:

```
number d motion
```

1. Move the cursor to the line marked
   ```
   LESSON 2.5 -->
   ```
2. Move the cursor to the first UPPER CASE word in this line
3. Type `2dw` to delete the two UPPER CASE words
4. Repeat these steps with a different count to delete the consecutive UPPER CASE words with one command

---

Lesson 2.6 Operating on Lines

Type `dd` to delete a whole line.

Due to the frequency of whole line deletion, the designers of Vi decided it would be easier to simply type two `d`'s to delete a line.

1. Move the cursor to the line marked
   ```
   LESSON 2.6 -->
   ```
2. Move the cursor to the second phrase
3. Type `dd` to delete the line
4. Now move to the fourth phrase
5. Type `2dd` to delete two lines

Lesson 2.7 The Undo / Redo Commands

Press `u` to undo the last command.
Press `U` to fix a whole line.
Press `Ctrl+r` to redo the last command undone.

1. Move the cursor to the line marked
   ```
   LESSON 2.7 -->
   ```
2. Place the cursor on the first error
3. Type `x` to delete the first unwanted character
4. Now type `u` to undo the last command executed
5. This time fix all the errors on the line using the `x` command
6. Type `U` to return the line to its original state
7. Type `u` a few times to undo the `U` and preceding commands
8. Now type `Ctrl+r` (hold the `Ctrl` key while hitting `r`) a few times to redo the commands (undo the undo's)
9. Commit these commands to memory. You will use them all the time
10. Exit vim, without saving changes

---

3 Put, Replace, and Change

Prerequisites

1. Close your eyes
2. Picture an emacs or pico user in your head
3. Shake your head condescendingly
4. Extra points if you snort in derision – you're a Vi user now. Own it. In fact, *pwn*[2] it[3].

Lesson 3.1 The Put Command

Type `p` to put previously deleted text after the cursor.

1. Open the file `lesson3` in vim
2. Move the cursor to the line marked
   ```
   LESSON 3.1 -->
   ```
3. Type `dd` to delete the line and store it in a Vim register
4. Move the cursor to the c) line, ABOVE where the deleted line should go
5. Type `p` to put the line below the cursor
6. Repeat these steps to put all the lines in correct order

Lesson 3.2 The Replace Command

Type `rx` to replace the character at the cursor with `x`.

1. Move the cursor to the line marked
   ```
   LESSON 3.2 -->
   ```
2. Move the cursor so that it is on top of the first error
3. Type `r` and then the character which should be there
4. Repeat these steps until the first line matches the second

NOTE: Remember that you should be learning by doing, and not memorization.

[2] http://en.wikipedia.org/wiki/Pwn
[3] http://bit.ly/omgpwniesbib

Lesson 3.3 The Change Operator

To change until the end of a word, type `ce`.

1. Move the cursor to the line marked
   ```
   LESSON 3.3 -->
   ```
2. Place the cursor on the `u` in `lubw`
3. Type `ce` and the correct word (in this case, type `ine`)
4. Press `Esc` to return to *normal mode*. Move to the next character that needs to be changed
5. Repeat these steps until the first line matches the second

Notice that `ce` deletes the word starting from the character under the cursor, and places you in *insert mode*.

Lesson 3.4 More Changes Using `c`

The change operator is used with the same motions as delete. The format is:

```
[number] c motion
```

The motions are the same, such as `w` (word) and `$` (end of line).

1. Move the cursor to the line marked
   ```
   LESSON 3.4 -->
   ```
2. Move the cursor so that it rests on the `s` in `some`
3. Type `c$` and type the rest of the line like the second
4. Press `Esc` to return to *normal mode*.
5. Exit Vim without saving any changes

NOTE: You can use the Backspace key to correct mistakes while typing.

4 Finding and Replacing

In this section, you'll see how to go directly to specific lines. You'll also learn how to find and replace text.

Lesson 4.1 Cursor Location and File Status

Press `Ctrl+G` to show your location in the file and the file status.
Press `G` to move to a line in the file.

1. Open the file `lesson4` in vim
2. Notice the status bar at the bottom of the screen. This is caller the *ruler*
3. Move down a few lines with the arrow keys. Notice that the line number is updated in the ruler
4. Move to the right a few characters with the arrow keys. Notice that the column number is updated in the ruler
5. By default, the ruler is not turned on in vim. It was automatically enabled for you when you ran `fixvim.sh`. Turn off the ruler temporarily by typing `:set noruler`
6. Now vim appears as it would by default. To find out the location of the cursor, press `Ctrl+G`. A message will appear at the bottom of the screen with the filename and the position in the file. Remember the line number for a moment
7. Press `G` to move to the bottom of the file
8. Type `gg` to move to the start of the file
9. Type the number of the line you were on and then `G` . This will return you to the line you were on earlier
10. Exit vim without saving any changes

Lesson 4.2 Another Way to Move to Specific Lines

Type `:linenum` to move to line `linenum`.
Type `:1` to move to the first line.
Type `:$` to move to the last line.

1. Compile the file `hello-buggy.c`:

   ```
   gcc -o hello hello-buggy.c
   ```

   Observe the line number reported in the first error message output by `gcc`
2. Open the file `hello-buggy.c` in vim
3. Move to the buggy line by typing `:n`, where `n` is the line number on which the error was reported
4. Fix the bug in the code

5. Move to the top of the file by typing `:1`
6. Move to the bottom of the file by typing `:$`
7. Save the file and exit
8. Recompile the file `hello-buggy.c`
9. Run the resulting `hello` executable. Remember: get in the habit of typing `./hello` instead of `hello`

Lesson 4.3 The Search Command

Type `/` followed by a phrase to search for the phrase in the forward direction.
Type `?` followed by a phrase to search for the phrase in the backward direction.
Type `n` to repeat the search.
Type `N` to repeat the search in the opposite direction.

1. Open the file `lesson4` in vim
2. Type the `/` character. Notice that it and the cursor appear at the bottom of the screen as with the `:` command
3. Type `errroor` and press `Enter`. This is the word for which you want to search
4. Search for the same phrase again by typing `n`
5. Search for the same phrase in the opposite direction by typing `N`
6. To go back to where you came from, press `Ctrl+O`. Repeat to go back further. `Ctrl-I` goes forward.

NOTE: When the search reaches the end of the file, it will continue at the start, unless the `wrapscan` option has been reset.

Lesson 4.4 Matching Parentheses Search

Type `%` to find a matching `)`, `]`, or }

1. Move the cursor to the line marked

```
LESSON 4.4 -->
```

2. Place the cursor on any `(`, `[`, or { in the line
3. Type the `%` character. The cursor will move to the matching parenthesis, bracket, or brace.
4. Type `%` again to move the cursor to the other matching element
5. Move the cursor to another `(`, `)`, `[`, `]`, {, or } and see what `%` does

NOTE: This is very useful in debugging a program with unmatched parentheses!

Lesson 4.5 The Substitute Command

Type `:s/old/new/g` to substitute `new` for `old` on the current line.
Type `:%s/old/new/g` to substitute `new` for `old` on every line.
Type `:#,#s/old/new/g`, where #,# are numbers that specify a range of lines in which the substitution is to be performed.

1. Move the cursor to the line marked

   ```
   LESSON 4.5 -->
   ```

2. Type `:s/thee/the` and press `Enter`. Note that this command only changes the first occurrence of `thee` in the line.
3. Now type `:s/thee/the/g` and press `Enter`. Adding the `g` flag means to substitute *globally* in the line, change all occurrences of `thee` in the line.
4. Type `:%s/errroor/errors/g` and press `Enter` to change every instance of `errroor` to `errors`
5. Type `:%s/errors/error/gc` and press `Enter` to change every instance of `errors` to `error`, confirming each substitution
6. Change all instances of `e` to `a` on all lines between lines 3 and 7 (inclusive). To do this, specify a range to the `:s` command, as shown above
7. Press `u` to undo your changes
8. Press `Ctrl+r` to redo your changes
9. Exit vim without saving any changes

5 Shell Commands, Selecting, and Merging

Lesson 5.1 Executing an External Command

Type `:!` followed by an external command to execute that command.

1. Open the file `lesson5` in vim
2. Type the familiar command `:` to set the cursor at the bottom of the screen. This allows you to enter a command-line command
3. Now type the `!` character. This allows you to execute any external shell command
4. As an example, type `ls` after the `!` and press `Enter`. This will show you a listing of your directory, just as if you were at the shell prompt.
5. Press `Enter` to return to vim

Notes:

- It is possible to execute any external command this way. You can also specify arguments, if needed, such as `:!ls -la`
- All `:` commands must be terminated by pressing `Enter`. Henceforth, this will be assumed and not explicitly mentioned

Lesson 5.2 Dropping to the Shell

Press `Ctrl+z` to pause vim and drop to the shell.
Type `fg` to resume vim (bring it back to the foreground).

1. Sometimes, while editing a file, it is useful to drop to the shell to perform a task. To do so, press `Ctrl+z`
2. Notice that vim is now stopped (paused in the background). You can now use the shell as you please
3. When ready to return to vim, type `fg` and press `Enter`. This puts vim back in the foreground

Lesson 5.3 Running a Command on a Selection

Press `v` to enter visual mode.
Press `V` to enter visual line mode.
Type `:!cmd` while a block is selected to execute `cmd` on the block.

1. Move the cursor to the top of the file by typing `:1`
2. Press `v`. This takes you into *visual mode*
3. Using the arrow keys, move down to the bottom of the first list so that the entirety of the first list is selected. Be sure to select the entirety of the last line by using the right arrow key

4. Press `:`. At the bottom of the screen, you will see `:'<,'>`. Do not delete this text. It specifies the range on which the command we will execute will operate
5. Type `!sort`. This executes the external `sort` command, passing to it the block of text that you selected. The output from the `sort` command then replaces the originally selected list
6. Notice that only the first list is sorted, since that was the list that was selected. The second list was not selected, so it was not passed to the `sort` command
7. You can run any shell commands that you wish on a selected block of text. Notice that the second list has duplicates in it. Move the cursor to the first line in the second list
8. Press `V` to enter *visual line mode*. Using the arrow keys, select the entirety of the second list. Notice that, in this mode, each line is selected when you use the arrows keys. This is contrast to *visual mode* in which individual characters are selected
9. Type `:!sort -r | uniq` to sort the list in descending lexicographical order, and remove all duplicates

Lesson 5.4 Retrieving and Merging Files

Type `:r FILE` to insert the contents of `FILE`

1. Type `:!ls`. You should see the file `numbers` in the directory listing
2. Place your cursor on the blank line between the two lists
3. Retrieve the file `numbers` by typing `:r numbers`. The contents of the file are placed below the line on which the cursor is resting
4. You can also retrieve the output of an external command. Move to the bottom of the file by typing `:$`
5. Type `:r !ls -la` to run the `ls` command and insert its output below the line on which the cursor is resting
6. Exit Vim without saving any changes

6 More Advanced Editing

Lesson 6.1 The Open Command

Press `o` to open up in insert mode on the line below the cursor
Press `O` to open up in insert mode on the line above the cursor

1. Open the file `lesson6` in vim
2. Move the cursor to the line marked

   ```
   LESSON 6.1 -->
   ```

3. Type `o` to open up a line BELOW the cursor and place you in *insert mode*
4. Type some text. Press `Esc` to return to *normal mode*.
5. Type `O` to open up a line ABOVE the cursor and place you in *insert mode*.
6. Type some text. Press `Esc` to return to *normal mode*.
7. You can add a line at the end of a file using `o`. Type `:$` to move to the last line in the file
8. Type `o` to add a new line at the end of the file. Press `Esc` to return to *normal mode*.
9. Similarly, you can add a line at the top of a file using `O`. Type `:1` to move to the first line in the file
10. Type `O` to add a new line at the top of the file. Press `Esc` to return to *normal mode*.

Lesson 6.2 The Join Command

Press `J` to join two lines

1. Move the cursor to the line marked

   ```
   LESSON 6.2 -->
   ```

2. Type `J` to join the line below with the current line
3. Type `3J` to join the two lines below with the current line
4. Type `5J` to join the four lines below with the current line
5. We've joined too many lines. Press `u` to undo the last join

Lesson 6.3 Replace Mode

Type `rx` to replace the character at the cursor with `x`.
Type `R` to enter *replace mode*, allowing you to replace more than one character

1. Move the cursor to the line marked

   ```
   LESSON 6.3 -->
   ```

2. Move the cursor to the beginning of the first `xxx` by pressing `w` several times
3. Type `r4` to replace the `x` under the cursor with `4`. Observe that using `r`, we can only replace one character at a time
4. Type `R` to enter *replace mode*, and then type `456`. Observe that, in *replace mode*, we can replace as many characters as we like
5. Press `Esc` to return to *normal mode*. Repeat the above step to replace the remaining `xxx`

NOTE: *Replace mode* is like *insert mode*, but every typed character deletes an existing character. It is often called *overwrite mode* in word processing software.

Lesson 6.4 Copying and Pasting

Use the `y` operator to copy (yank) text.
Use the `p` operator to paste (put) yanked text after the cursor.
Use the `P` operator to paste (put) yanked text before the cursor.

1. By searching for `6.4` using the `/` command, move the cursor to the line marked

   ```
   LESSON 6.4 -->
   ```

2. Place the cursor after `a)`
3. Enter *visual mode* by pressing `v`. Move the cursor to just before `first`
4. Press `y` to yank the highlighted text
5. Move the cursor to the end of the next line by typing `j$` (remember: `j` moves down, while `$` moves to the end of the current line
6. Press `p` to put the yanked text after the cursor
7. Press `a` to append text, and type `second.` Be sure to include the period.
8. Press `Esc` to return to *normal mode*.
9. Move the cursor to the space right after `first`
10. Press `v` to enter *visual mode* and select the text `item` (including the space before the word, but not including the period after)

11. Press `y` to yank the text
12. Move the cursor to the period at the end of the next line
13. To paste the yanked text before the period, press `P`

Lesson 6.5 More on Copying and Pasting

Type `yy` to copy the current line.
Type `5yy` to copy the next five lines (including the current line).
Type `yw` to copy the current word.

1. Move the cursor to the line marked

   ```
   LESSON 6.5 -->
   ```

2. Copy the entire line by typing `yy`
3. Paste the yanked line below by pressing `p`
4. Move back up one line
5. Copy two lines (including the current) by typing `2yy`
6. Press `P` to paste them above the current line
7. Press `w` a few times until the cursor is at the start of the word `a`
8. Press `2yw` to yank the next two words (including `a`)
9. Open up a new line in insert mode below the current line by pressing `o`
10. Press `Esc` to return to *normal mode*.
11. Press `p` to paste the yanked words
12. Exit vim without saving changes

7 Vim – The Programmer's Editor

Lesson 7.1 Indenting

Press `>>` to shift right
Press `<<` to shift left
Press `>` in visual mode to shift right
Press `<` in visual mode to shift left
Press `=` in visual mode to auto-indent

1. Open the file `lesson7.c` in vim
2. Move to the first `printf` statement in the `say_hello` function
3. Type `>>` to shift the line to the right
4. Press `.` to repeat the last command
5. Type `<<` to shift the line to the left
6. Move to the line containing the `if` statement in the `fact` function
7. Press `v` to enter *visual mode*
8. Move down to the `return` statement in the same function so that the entire body of the function is selected
9. Press `>` to shift the selected text to the right
10. Indent the two return statements with `>>`
11. Select the body of the `main` function (excluding the braces)
12. Press `=` to auto-indent the function

Lesson 7.2 Folding

Type `:fold` in visual mode to fold the selected text.
Type `za` to toggle a fold open and closed.
Type `zR` to open all folds.
Type `zM` to close all folds.

1. Go to line 3 by typing `:3`
2. Press `V` to enter *visual line mode*. There is no particular reason to use *visual line mode* over *visual mode* here – it is just used to give you practice using both modes
3. Select the entire `say_hello` function, including closing brace
4. Type `:`. You will see `:'<,'>` appear in the status bar at the bottom of the screen. Do not delete this text

5. Type `fold` and press `Enter`. This will fold the `say_hello` function, which is quite useful when you have a lot of functions and want to avoid distraction
6. Follow the steps above to fold the `fact` function
7. Return to line 3
8. Open the fold by typing `za`
9. Close the fold by typing `za` again
10. Open all folds by typing `zR`
11. Close all folds by typing `zM`
12. Exit vim without saving changes

Lesson 7.3 Visual Block Mode

Press `Ctrl+v` to enter *visual block mode*

1. Open the file `users` in vim
2. Move to the start of the first email address in the file
3. Press `v` to enter *visual mode*
4. Try selecting only the email address column, without selecting the names or provinces/states
5. Observe that you can't do it, since visual mode selects the entire line when you press the down arrow key. Press `Esc` to exit visual mode
6. Move back to the start of the first email address in the file
7. Press `Ctrl+v` to enter visual block mode
8. Using the arrow keys, select all email addresses in the file, without selecting the names or provinces/states
9. Once all email addresses are selected, press `y` to yank all the email addresses

We will use the yanked email addresses in the next lesson.

Lesson 7.4 Working in Split Screen

Type `:sp` to open the current file in a horizontal split
Type `:vsp` to open the current file in a vertical split
Type `:sp FILE` to open `FILE` in a horizontal split
Type `:vsp FILE` to open `FILE` in a vertical split
Press `Ctrl+w, arrow` to move between splits

1. Type `:sp email` to open a new file `email` in a split screen

2. Your cursor will now be in the `email` screen. Press `p` to paste the email addresses yanked in the previous lesson
3. Press `Ctrl+w` and then press the down arrow key to move back to the `users` screen
4. Move to the start of the first line
5. Enter *visual block mode* and select all names in the file, without selecting any email addresses or provinces/states
6. Press `y` to yank the selected names
7. Type `:vsp names` to open a new file `names` in a vertical split screen
8. Press `p` to paste the yanked names
9. Using `Ctrl+w` and the arrow keys, move back to the `email` screen
10. Press `:wq` to save and close the `email` screen
11. If necessary, using `Ctrl+w` and the arrow keys, move back to the `names` screen
12. Press `:wq` to save and close the `names` screen
13. Exit vim without saving any changes

Lesson 7.5 Auto-Completion

Press `Ctrl+n` to scroll forward through auto-completions for words in the current document
Press `Ctrl+p` to scroll backward through auto-completions for words in the current document

1. Open the file `Lesson7.java`
2. Move to the line that contains
   ```
   // Insert code below this line
   ```
3. Press `o` to open up in insert mode on the next line
4. Type `throw new Ex`
5. Press `Ctrl+n`. Notice that the first auto-completion – `Example` – appears
6. Press `Ctrl+n` again. This time, `Exception` should appear
7. Press `Ctrl+n` again. You should be back to the original `Ex`
8. Press `Ctrl+p` to move back to `Exception`
9. Type `();` to finish the line of code

Lesson 7.6 Incrementing / Decrementing Numbers

Press `Ctrl+a` to increment the next number on the current line.
Press `Ctrl+x` to decrement the next number on the current line.

1. Press `Enter` to move to the next line in `Example7.java`
2. Type the following code:

```
int i = 0;  long j = 1;
```

3. Press `Esc` to return to *normal mode*.
4. Ensure the cursor is on the line you just typed, and press `0` to move to the start of the line
5. Press `Ctrl+a` to increment the value of variable `i` by 1
6. Type `4` and then press `Ctrl+a` to increment the value of variable `i` by 4. The variable should now be assigned the value `5`
7. Press `w` to move past the value `5` to the semi-colon
8. Type `3` and then press `Ctrl+x` to decrement the value of variable `j` by 3. The variable should now be assigned the value `-2`

8 Where to Go From Here

Lesson 8.1 Getting Help

Vim has a comprehensive online help system. To get started, simply type `:help` and press `Enter`.

- Read the text in the help window to find out how the help works
- Type `Ctrl+w` `Ctrl+w` to jump from one window to another
- Type `:q` and press `Enter` to close the help window

You can find help on just about any subject, by giving an argument to the `:help` command. Try these (don't forget to press `Enter`):

- `:help w`
- `:help c_CTRL-D`
- `:help insert-index`
- `:help user-manual`