Deleting Text

Goal:

The goal of this exercise is to practice deleting text in a file. You'll also practice the [count][operator]{motion} pattern.

Instructions:

Open the practicedeleting.txt file

First, start a command line session on your local machine. Next, use vim to open the "practicedeleting.txt" file that came in the course downloads. To do that, navigate to the location of the file. Remember this could be different for you depending on where you extracted the contents of the file. This example assumes the course download was saved into your Downloads folder and extracted from there.

```
cd Downloads
cd vimclass
vim practicedeleting.txt
```

Practice deleting individual characters

Move to the 3rd line of the file. Remember, you can do this in a few different ways. You can press the **j** key until you're there, you can use **`3gg`**, **`3G`**, or **`:3<ENTER>`**.

Remove the extra "k" in the word "mistakke" by positioning your cursor under one of the additional letters and hit the **x** key. Here is what the sentence looks like before your edit:

```
First, fix this spelling mistakke.
```

Here is what it looks like after:

```
First, fix this spelling mistake.
```

Move to the 4th line of the file. Remove the repeated letters from each of the words. Use **x** to delete the character under your cursor to do so. Here is what the sentence looks like before:

```
Fixx theese allso.
```

After you delete the additional "x", "e", and "l", it will look like this:

```
Fix these also.
```

Move down to the next line:

```
Delete this text with the X command.
```

Position your cursor at the end of the line. You can repeatedly press the **l** key, or if you want to be really efficient you can use the **$** key. Now delete all the text you can with **X** key. You'll find that only the period remains on the line:

```
.
```

To delete that character, press **x**. Now the line is empty.

Practice deleting motions

Move down to the next line:

```
Who let the dogs out? cats
```

Position your cursor one character right of the question mark. Your cursor will be in the space between "?" and "cats". Delete the remaining text on the line. You can use **d$**, or an even shorter version of the command **D**. After your edit, the line will look like the following:

```
Who let the dogs out?
```

Now position the cursor at the beginning of line 43. (Hint: **43gg** works.) Delete the first word of the line with the **d** operator. Remember the pattern of [count][operator]{motion}. To delete that first word you can use **dw** or even **dW**. Here is the line before the first word is deleted:

```
Far far away, behind the wild mountains, far from the countries Vokalia and
```

Here is the line after the first word is deleted:

```
far away, behind the wild mountains, far from the countries Vokalia and
```

Delete the second word, too. Now the line looks like so:

```
away, behind the wild mountains, far from the countries Vokalia and
```

Delete the text "away, " with two keystrokes. Remember that the **w** motion will stop at punctuation while the **W** motion ignores punctuation and stops the cursor after white space. So, to delete "away, " in two keystrokes you use **dW**. Now the line looks like this:

```
behind the wild mountains, far from the countries Vokalia and
```

Position your cursor at the beginning of the first occurrence of the word "the". Delete the first word in the sentence using an operator and a motion. To do that, type **db**. You could have also used **dB**. Here is how the line appears now:

```
the wild mountains, far from the countries Vokalia and
```

Now, delete words "the wild ". One way to do this is to use **2dw**.

```
mountains, far from the countries Vokalia and
```

Now delete "mountains, far ". A motion that moves your just past "mountains, far " is **2W**. So, use **d2W** to delete the text. This is what remains on the line, now:

```
from the countries Vokalia and
```

Practice deleting lines

Delete the line by using **dd**. Your cursor is now placed on this line:

```
Consonantia, there live the blind texts. Separated they live in Bookmarksgrove
```

To delete multiple lines use, **[count]dd**. Let's delete these lines:

```
Consonantia, there live the blind texts. Separated they live in Bookmarksgrove
right at the coast of the Semantics, a large language ocean.
```

Notice that there are 3 lines in total. One line begins with "Consonantia", the next with "right" and the third line is blank and doesn't have any text at all. To delete those 3 lines, use **3dd.** Now your cursor is placed on this line:

```
A small river named Duden flows by their place and supplies it with the
```

Delete the next three lines by using one keystroke. Simply type a period and the previous command will be repeated. When you press **.** the following three lines will be deleted:

```
A small river named Duden flows by their place and supplies it with the
necessary regelialia. It is a paradisematic country, in which roasted parts of
sentences fly into your mouth.
```

Save your work (or not!)

If you want to save your changes and keep vim running, you can use **:w<ENTER>**. To save your changes and immediately exit, type **:wq<ENTER>**. If you want to abandon your changes, use **:q!<ENTER>**. It's up to you.

Your turn

If you're up to it, have some fun deleting text in the file using what you know.

Practice Using the Vim Help System

Goal:

The goal of this exercise is to get experience using the built-in vim help system.

Instructions:

Open the help.txt file

First, start a command line session on your local machine. Next, use vim to open the "help.txt" file that came in the course downloads. To do that, navigate to the location of the file. Remember this could be different for you depending on where you extracted the contents of the file. This example assumes the course download was saved into your Downloads folder and extracted from there.

```
cd Downloads
cd vimclass
vim help.txt
```

Start the help system

Start the help system by typing **`:help<ENTER>`**. Read the text on your screen. Remember that this is a simple text file.

Close the help system

Exit out of help by typing **`:q<ENTER>`**.

Start the help system again

Start the help system by typing **`:h<ENTER>`**. Many times there are shortened or abbreviated version of commands. In this example, **`:h`** is the same as **`:help`**.

Read the help on some of the commands you already know

You already know several vim commands. Use the help system to get information on the following commands. Read the related documentation to refresh your memory about each command.

`:h i`
`:h :wq`
`:h :q`

```
:h Ctrl-f
:h ^f          # Note, using a caret symbol is the same as "Ctrl".
          # So, ^f and Ctrl-f are the same.
:h ^b
:h w
```

Practicing jumping through the help system

Look up the documentation on the **W** command by typing **:h W<ENTER>**. Position your cursor under the world "exclusive". Type **Ctrl-]** to jump to the documentation for "exclusive." Once you've done reading, jump back to your previous location in the help system by typing **Ctrl-o**.

Now move your cursor below the word "count". Type **Ctrl-]** to jump to the count documentation. When you are finished reading about count, type **Ctrl-o** to return to your previous position.

Determine the equivalent command for Ctrl-G

Look at the documentation for Ctrl-G by typing **:h ^g<ENTER>**. Can you tell what other command or commands are the same as Ctrl-G? That's right, **:f** and **:file**. Try it now by typing **:f<ENTER>**. Notice the line of text that appears at the bottom of your screen. It shows you the name of the help file you're viewing.

Switch back to the file you're editing

Type **Ctrl-w-w**. (Hold down the **Ctrl** key and type **w** twice.) Your cursor is now in the bottom window that contains the `help.txt` file you opened at the beginning of this exercise. Confirm this by typing **Ctrl-g** or by using the **:f** or **:file** commands.

Your turn

Switch back to the help window by typing **Ctrl-w-w**. Think of some commands you've learned in the course. Use the help system to look up the documentation for those commands. Feel free to explore as much or as little as you like.

Finishing up

When you're done, exit out of the help system by typing **:q<ENTER>**. To stop editing the help.txt file and exit the vim editor use **:q!<ENTER>**.

Practice Using the Vim Help System

Goal:

The goal of this exercise is to get experience using the built-in vim help system.

Instructions:

Open the help.txt file

First, start a command line session on your local machine. Next, use vim to open the "help.txt" file that came in the course downloads. To do that, navigate to the location of the file. Remember this could be different for you depending on where you extracted the contents of the file. This example assumes the course download was saved into your Downloads folder and extracted from there.

```
cd Downloads
cd vimclass
vim help.txt
```

Start the help system

Start the help system by typing **`:help<ENTER>`**. Read the text on your screen. Remember that this is a simple text file.

Close the help system

Exit out of help by typing **`:q<ENTER>`**.

Start the help system again

Start the help system by typing **`:h<ENTER>`**. Many times there are shortened or abbreviated version of commands. In this example, **`:h`** is the same as **`:help`**.

Read the help on some of the commands you already know

You already know several vim commands. Use the help system to get information on the following commands. Read the related documentation to refresh your memory about each command.

`:h i`
`:h :wq`
`:h :q`

```
:h Ctrl-f
:h ^f          # Note, using a caret symbol is the same as "Ctrl".
          # So, ^f and Ctrl-f are the same.
:h ^b
:h w
```

Practicing jumping through the help system

Look up the documentation on the **W** command by typing **:h W<ENTER>**. Position your cursor under the world "exclusive". Type **Ctrl-]** to jump to the documentation for "exclusive." Once you've done reading, jump back to your previous location in the help system by typing **Ctrl-o**.

Now move your cursor below the word "count". Type **Ctrl-]** to jump to the count documentation. When you are finished reading about count, type **Ctrl-o** to return to your previous position.

Determine the equivalent command for Ctrl-G

Look at the documentation for Ctrl-G by typing **:h ^g<ENTER>**. Can you tell what other command or commands are the same as Ctrl-G? That's right, **:f** and **:file**. Try it now by typing **:f<ENTER>**. Notice the line of text that appears at the bottom of your screen. It shows you the name of the help file you're viewing.

Switch back to the file you're editing

Type **Ctrl-w-w**. (Hold down the **Ctrl** key and type **w** twice.) Your cursor is now in the bottom window that contains the `help.txt` file you opened at the beginning of this exercise. Confirm this by typing **Ctrl-g** or by using the **:f** or **:file** commands.

Your turn

Switch back to the help window by typing **Ctrl-w-w**. Think of some commands you've learned in the course. Use the help system to look up the documentation for those commands. Feel free to explore as much or as little as you like.

Finishing up

When you're done, exit out of the help system by typing **:q<ENTER>**. To stop editing the help.txt file and exit the vim editor use **:q!<ENTER>**.

Cut, Copy, and Paste (Delete, Yank, and Put)

Goal:

The goal of this exercise is to give you practice with cut, copy, and paste operations. Additionally you'll get a chance to work with registers.

Instructions:

Open the dyp.txt file

First, start a command line session on your local machine. Next, use vim to open the "dyp.txt" file that came in the course downloads. To do that, navigate to the location of the file. Remember this could be different for you depending on where you extracted the contents of the file. This example assumes the course download was saved into your Downloads folder and extracted from there.

```
cd Downloads
cd vimclass
vim dyp.txt
```

Swap the first two lines of the file

First, delete the first line of the file by typing **dd**. This places the line into the default register. Now put the line below the new first line in the file with the **p** command.

Here is what the first two lines of the file look like before your edit:

```
This was originally the first line in the file.
This was originally the second line in the file.
```

Here is what they look like after:

```
This was originally the second line in the file.
This was originally the first line in the file.
```

Put the first line in another place in the file

Remember that you can reuse the contents of the default register. Place the original first line of the file just below this line:

```
What was the first line in the file originally?  Place it below:
```

To do that, position your cursor on that line and use the put command by typing **p**.

Practice putting lines above your cursor position

Put the original first line in the file just above this line:

```
What was the first line in the file originally?  Place it above:
```

To do that, place your cursor on that line and type **P**.

Fix spelling/typing mistakes by swapping characters

Position your cursor under the "e" on this line:

```
teh
```

Swap the "e" and "h" so that the word reads "the". To do that, type **x** to delete the "e" and **p** to paste it after your current cursor position.

Use the same process to fix the spelling or typing mistakes on the following four lines:

```
psell = spell
vmi = vim
wrod = word
taht = that
```

Swap words

Change this line from:

```
second, First, third.
```

To:

```
First, second, third.
```

To do that position your cursor at the beginning of the line under the "s" in the word second. Delete it along with the comma that follows with the **dW** command. (Notice the capital W). Next, position

your cursor at the beginning of the word "third" by typing **`W`**. Now put the text in the default register before your cursor position with the **`P`** command.

Duplicate a line

Duplicate the following line, placing another copy of it just below it:

```
Duplicate this line.
```

Position your cursor on the line and yank it into the default register with **`yy`**. Next, paste it below the line with the **`p`** command.

Duplicate a word

Duplicate the words "really, really," on the following line:

```
I really, really, love vim!
```

Place your cursor under the "r" in the first occurrence of the word "really". Next yank those two words, including the punctuation, into the default register with **`y2W`**. Next, paste them before your current cursor position with **`P`**. The line should now read:

```
I really, really, really, really, love vim!
```

Use the numbered registers

Put the text "TODO" above any lines in the file that start with "Fix this". Also, delete any lines that read "Delete this". Start at your current cursor position and work your way down through the file mixing your deletes and puts.

First, yank the line that reads "TODO" with **`yy`**. Next delete the two lines that read "Delete this" with **`2dd`**. Position your cursor on the line that reads "Fix this" and paste "TODO" above it. To do that you'll need to use the `"0` register which contains the most recent yanked text. Type **`"0P`**. Repeat this process until all the "Fix" lines have "TODO" above them and all the "Delete" lines are deleted.

Use named registers

Place the following line in the `"j` register:

```
Yank this line into the "j register.
```

Position your cursory anywhere on the line and type **"jyy**.

Now place the following line in the "f register.

```
Yank this line into the "f register.
```

Position your cursory anywhere on the line and type **"fyy**.

Next put the contents of the "j register below this line with **"jp**:

```
Put the contents of the "j register below:
```

Next put the contents of the "f register below this line with **"fp**:

```
Put the contents of the "f register below:
```

Append the following line to the "j register:

```
Append this line to the "j register.
```

Place your cursor anywhere on the line and type **"Jyy**. (Notice the capital J).

Append the following line to the "f register:

```
Append this line to the "f register.
```

Place your cursor anywhere on the line and type **"Fyy**. (Notice the capital F).

Look at the contents of all the registers with **:reg<ENTER>**. If you only want to view the "j and "f registers, use **:reg jf<ENTER>**

Now put the contents of the "j register below this line with **"jp**:

```
Put the contents of the "j register below:
```

Next put the contents of the "f register below this line with **"fp**:

```
Put the contents of the "f register below:
```

Undo and Redo

Delete the following three lines:

```
ONE)
TWO)
THREE)
```

You can do that by positioning your cursor on the first line and typing **3dd**. Undo your deletion by typing **u**. Notice how all three of the lines return.

Redo your command with **Ctrl-r**. Now the three lines are removed again.

Insert a new line into the file. First enter insert mode with **i**. Now type any sentence into vim. For example, you could use this sentence:

```
Vim is fun!
```

Press **<Escape>** to return to normal mode. To undo your text insertion type **u**. Now redo your text insertion with **Ctrl-r**.

Your turn

I encourage you to experiment and come up with some of your own practice exercises. Better yet… Do you have a file that needs editing? Open it up in vim and try using some of your delete, yank, and put skills!

Exit out of vim

If you want to abandon your changes so you can try this practice exercise again, use **:q!<ENTER>**.