Emacs tutorial. See end for copying conditions.

Emacs commands generally involve the CONTROL key (sometimes labeled
CTRL or CTL) or the META key (sometimes labeled EDIT or ALT). Rather than
write that in full each time, we'll use the following abbreviations:

 C-<chr> means hold the CONTROL key while typing the character <chr>
     Thus, C-f would be: hold the CONTROL key and type f.
 M-<chr> means hold the META or EDIT or ALT key down while typing <chr>.
     If there is no META, EDIT or ALT key, instead press and release the
     ESC key and then type <chr>. We write <ESC> for the ESC key.

Important note: to end the Emacs session, type C-x C-c. (Two characters.)
To quit a partially entered command, type C-g.
To stop the tutorial, type C-x k, then <Return> at the prompt.
The characters ">>" at the left margin indicate directions for you to
try using a command. For instance:

[Middle of page left blank for didactic purposes.   Text continues below]

>> Now type C-v (View next screen) to scroll down in the tutorial.
    (go ahead, do it by holding down the CONTROL key while typing v).
    From now on, please do this whenever you reach the end of the screen.

Note that there is an overlap of two lines when you scroll a whole
screenful; this provides some continuity so you can continue reading
the text.

This is a copy of the Emacs tutorial text, customized slightly for
you. Later on we will instruct you to try various commands to alter
this text. Don't worry if you change this text before we tell you to;
that is called "editing" and that's what Emacs is for.

The first thing that you need to know is how to move around from place
to place in the text. You already know how to move forward one screen,
with C-v. To move backwards one screen, type M-v (hold down the META key
and type v, or type <ESC>v if you do not have a META, EDIT, or ALT key).

>> Try typing M-v and then C-v, a few times.

It is ok to scroll this text in other ways, if you know how.

* SUMMARY

The following commands are useful for viewing screenfuls:

```
C-v Move forward one screenful
M-v Move backward one screenful
C-l Clear screen and redisplay all the text,
      moving the text around the cursor
      to the center of the screen.
      (That's CONTROL-L, not CONTROL-1.)
```

>> Find the cursor, and note what text is near it. Then type C-l.
Find the cursor again and notice that the same text is still near
the cursor, but now it is in the center of the screen.
If you press C-l again, this piece of text will move to the top of
the screen. Press C-l again, and it moves to the bottom.

You can also use the PageUp and PageDn keys to move by screenfuls, if
your terminal has them, but you can edit more efficiently if you use
C-v and M-v.

* BASIC CURSOR CONTROL

Moving from screenful to screenful is useful, but how do you
move to a specific place within the text on the screen?

There are several ways you can do this. You can use the arrow keys,
but it's more efficient to keep your hands in the standard position
and use the commands C-p, C-b, C-f, and C-n. These characters
are equivalent to the four arrow keys, like this:

```
            Previous line, C-p
                 :
                 :
 Backward, C-b .... Current cursor position .... Forward, C-f
                 :
                 :
              Next line, C-n
```

>> Move the cursor to the line in the middle of that diagram
using C-n or C-p. Then type C-l to see the whole diagram
centered in the screen.

You'll find it easy to remember these letters by words they stand for:
P for previous, N for next, B for backward and F for forward. You
will be using these basic cursor positioning commands all the time.

>> Do a few C-n's to bring the cursor down to this line.

>> Move into the line with C-f's and then up with C-p's.
See what C-p does when the cursor is in the middle of the line.

Each line of text ends with a Newline character, which serves to
separate it from the following line. (Normally, the last line in
a file will have a Newline at the end, but Emacs does not require it.)

>> Try to C-b at the beginning of a line. It should move to
the end of the previous line. This is because it moves back
across the Newline character.

C-f can move across a Newline just like C-b.

```
>> Do a few more C-b's, so you get a feel for where the cursor is.
   Then do C-f's to return to the end of the line.
   Then do one more C-f to move to the following line.

When you move past the top or bottom of the screen, the text beyond
the edge shifts onto the screen.  This is called "scrolling".  It
enables Emacs to move the cursor to the specified place in the text
without moving it off the screen.

>> Try to move the cursor off the bottom of the screen with C-n, and
   see what happens.

If moving by characters is too slow, you can move by words.  M-f
(META-f) moves forward a word and M-b moves back a word.

>> Type a few M-f's and M-b's.

When you are in the middle of a word, M-f moves to the end of the word.
When you are in whitespace between words, M-f moves to the end of the
following word.  M-b works likewise in the opposite direction.

>> Type M-f and M-b a few times, interspersed with C-f's and C-b's
   so that you can observe the action of M-f and M-b from various
   places inside and between words.

Notice the parallel between C-f and C-b on the one hand, and M-f and
M-b on the other hand.  Very often Meta characters are used for
operations related to the units defined by language (words, sentences,
paragraphs), while Control characters operate on basic units that are
independent of what you are editing (characters, lines, etc).

This parallel applies between lines and sentences: C-a and C-e move to
the beginning or end of a line, and M-a and M-e move to the beginning
or end of a sentence.

>> Try a couple of C-a's, and then a couple of C-e's.
   Try a couple of M-a's, and then a couple of M-e's.

See how repeated C-a's do nothing, but repeated M-a's keep moving one
more sentence.  Although these are not quite analogous, each one seems
natural.

The location of the cursor in the text is also called "point".  To
paraphrase, the cursor shows on the screen where point is located in
the text.

Here is a summary of simple cursor-moving operations, including the
word and sentence moving commands:

    C-f Move forward a character
    C-b Move backward a character

    M-f Move forward a word
    M-b Move backward a word

    C-n Move to next line
    C-p Move to previous line

    C-a Move to beginning of line
    C-e Move to end of line

    M-a Move back to beginning of sentence
    M-e Move forward to end of sentence

>> Try all of these commands now a few times for practice.
   These are the most often used commands.
```

Two other important cursor motion commands are M-< (META Less-than), which moves to the beginning of the whole text, and M-> (META Greater-than), which moves to the end of the whole text.

On most terminals, the "<" is above the comma, so you must use the shift key to type it. On these terminals you must use the shift key to type M-< also; without the shift key, you would be typing M-comma.

>> Try M-< now, to move to the beginning of the tutorial.
   Then use C-v repeatedly to move back here.

>> Try M-> now, to move to the end of the tutorial.
   Then use M-v repeatedly to move back here.

You can also move the cursor with the arrow keys, if your terminal has arrow keys. We recommend learning C-b, C-f, C-n and C-p for three reasons. First, they work on all kinds of terminals. Second, once you gain practice at using Emacs, you will find that typing these Control characters is faster than typing the arrow keys (because you do not have to move your hands away from touch-typing position). Third, once you form the habit of using these Control character commands, you can easily learn to use other advanced cursor motion commands as well.

Most Emacs commands accept a numeric argument; for most commands, this serves as a repeat-count. The way you give a command a repeat count is by typing C-u and then the digits before you type the command. If you have a META (or EDIT or ALT) key, there is another, alternative way to enter a numeric argument: type the digits while holding down the META key. We recommend learning the C-u method because it works on any terminal. The numeric argument is also called a "prefix argument", because you type the argument before the command it applies to.

For instance, C-u 8 C-f moves forward eight characters.

>> Try using C-n or C-p with a numeric argument, to move the cursor
   to a line near this one with just one command.

Most commands use the numeric argument as a repeat count, but some commands use it in some other way. Several commands (but none of those you have learned so far) use it as a flag--the presence of a prefix argument, regardless of its value, makes the command do something different.

C-v and M-v are another kind of exception. When given an argument, they scroll the text up or down by that many lines, rather than by a screenful. For example, C-u 8 C-v scrolls by 8 lines.

>> Try typing C-u 8 C-v now.

This should have scrolled the text up by 8 lines. If you would like to scroll it down again, you can give an argument to M-v.

If you are using a graphical display, such as X or MS-Windows, there should be a tall rectangular area called a scroll bar on one side of the Emacs window. You can scroll the text by clicking the mouse in the scroll bar.

If your mouse has a wheel button, you can also use this to scroll.

* IF EMACS STOPS RESPONDING

If Emacs stops responding to your commands, you can stop it safely by typing C-g. You can use C-g to stop a command which is taking too

long to execute.

You can also use C-g to discard a numeric argument or the beginning of a command that you do not want to finish.

>> Type C-u 100 to make a numeric argument of 100, then type C-g.
   Now type C-f. It should move just one character, because you
   canceled the argument with C-g.

If you have typed an <ESC> by mistake, you can get rid of it with a C-g.

* DISABLED COMMANDS

Some Emacs commands are "disabled" so that beginning users cannot use them by accident.

If you type one of the disabled commands, Emacs displays a message saying what the command was, and asking you whether you want to go ahead and execute the command.

If you really want to try the command, type <SPC> (the Space bar) in answer to the question. Normally, if you do not want to execute the disabled command, answer the question with "n".

>> Type C-x C-l (which is a disabled command),
   then type n to answer the question.

* WINDOWS

Emacs can have several "windows", each displaying its own text. We will explain later on how to use multiple windows. Right now we want to explain how to get rid of extra windows and go back to basic one-window editing. It is simple:

   C-x 1 One window (i.e., kill all other windows).

That is CONTROL-x followed by the digit 1. C-x 1 expands the window which contains the cursor, to occupy the full screen. It deletes all other windows.

>> Move the cursor to this line and type C-u 0 C-l.
>> Type C-h k C-f.
   See how this window shrinks, while a new one appears
   to display documentation on the C-f command.

>> Type C-x 1 and see the documentation listing window disappear.

There is a whole series of commands that start with CONTROL-x; many of them have to do with windows, files, buffers, and related things. These commands are two, three or four characters long.

* INSERTING AND DELETING

If you want to insert text, just type the text. Ordinary characters, like A, 7, *, etc., are inserted as you type them. To insert a Newline character, type <Return> (this is the key on the keyboard which is sometimes labeled "Enter").

To delete the character immediately before the current cursor position, type <DEL>. This is the key on the keyboard usually labeled

"Backspace"--the same one you normally use, outside Emacs, to delete
the last character typed.

There may also be another key on your keyboard labeled <Delete>, but
that's not the one we refer to as <DEL>.

>> Do this now--type a few characters, then delete them by
   typing <DEL> a few times. Don't worry about this file
   being changed; you will not alter the master tutorial.
   This is your personal copy of it.

When a line of text gets too big for one line on the screen, the line
of text is "continued" onto a second screen line. If you're using a
graphical display, little curved arrows appear in the narrow spaces on
each side of the text area (the left and right "fringes"), to indicate
where a line has been continued. If you're using a text terminal, the
continued line is indicated by a backslash ('\') on the rightmost
screen column.

>> Insert text until you reach the right margin, and keep on inserting.
   You'll see a continuation line appear.

>> Use <DEL>s to delete the text until the line fits on one screen
   line again. The continuation line goes away.

You can delete a Newline character just like any other character.
Deleting the Newline character between two lines merges them into
one line. If the resulting combined line is too long to fit in the
screen width, it will be displayed with a continuation line.

>> Move the cursor to the beginning of a line and type <DEL>. This
   merges that line with the previous line.

>> Type <Return> to reinsert the Newline you deleted.

The <Return> key is special, in that pressing it may do more than
just insert a Newline character. Depending on the surrounding text,
it may insert whitespace after the Newline character, so that when
you start typing on the newly created line, the text lines up with
that on the previous line. We call this behavior (where pressing a
key does more than simply inserting the relevant character) "electric".

>> Here is an example of <Return> being electric.
   Type <Return> at the end of this line.

You should see that after inserting the Newline, Emacs inserts spaces
so that the cursor moves under the "T" of "Type".

Remember that most Emacs commands can be given a repeat count;
this includes text characters. Repeating a text character inserts
it several times.

>> Try that now -- type C-u 8 * to insert ********.

You've now learned the most basic way of typing something in
Emacs and correcting errors. You can delete by words or lines
as well. Here is a summary of the delete operations:

<DEL>	Delete the character just before the cursor
C-d	Delete the next character after the cursor
M-<DEL>	Kill the word immediately before the cursor
M-d	Kill the next word after the cursor
C-k	Kill from the cursor position to end of line
M-k	Kill to the end of the current sentence

Notice that <DEL> and C-d vs M-<DEL> and M-d extend the parallel
started by C-f and M-f (well, <DEL> is not really a control character,
but let's not worry about that). C-k and M-k are like C-e and M-e,
sort of, in that lines are paired with sentences.

You can also kill a segment of text with one uniform method. Move to
one end of that part, and type C-<SPC>. (<SPC> is the Space bar.)
Next, move the cursor to the other end of the text you intend to kill.
As you do this, Emacs highlights the text between the cursor and the
position where you typed C-<SPC>. Finally, type C-w. This kills all
the text between the two positions.

>> Move the cursor to the Y at the start of the previous paragraph.
>> Type C-<SPC>. Emacs should display a message "Mark set"
   at the bottom of the screen.
>> Move the cursor to the n in "end", on the second line of the
   paragraph.
>> Type C-w. This will kill the text starting from the Y,
   and ending just before the n.

The difference between "killing" and "deleting" is that "killed" text
can be reinserted (at any position), whereas "deleted" things cannot
be reinserted in this way (you can, however, undo a deletion--see
below). Reinsertion of killed text is called "yanking". (Think of it
as yanking back, or pulling back, some text that was taken away.)
Generally, the commands that can remove a lot of text kill the text
(they are set up so that you can yank the text), while the commands
that remove just one character, or only remove blank lines and spaces,
do deletion (so you cannot yank that text). <DEL> and C-d do deletion
in the simplest case, with no argument. When given an argument, they
kill instead.

>> Move the cursor to the beginning of a line which is not empty.
   Then type C-k to kill the text on that line.
>> Type C-k a second time. You'll see that it kills the Newline
   which follows that line.

Note that a single C-k kills the contents of the line, and a second
C-k kills the line itself, and makes all the other lines move up. C-k
treats a numeric argument specially: it kills that many lines AND
their contents. This is not mere repetition. C-u 2 C-k kills two
lines and their Newlines; typing C-k twice would not do that.

You can yank the killed text either at the same place where it was
killed, or at some other place in the text you are editing, or even in
a different file. You can yank the same text several times; that
makes multiple copies of it. Some other editors call killing and
yanking "cutting" and "pasting" (see the Glossary in the Emacs
manual).

The command for yanking is C-y. It reinserts the last killed text,
at the current cursor position.

>> Try it; type C-y to yank the text back.

If you do several C-k's in a row, all of the killed text is saved
together, so that one C-y will yank all of the lines at once.

>> Do this now, type C-k several times.

Now to retrieve that killed text:

>> Type C-y. Then move the cursor down a few lines and type C-y
   again. You now see how to copy some text.

What do you do if you have some text you want to yank back, and then you kill something else? C-y would yank the more recent kill. But the previous text is not lost. You can get back to it using the M-y command. After you have done C-y to get the most recent kill, typing M-y replaces that yanked text with the previous kill. Typing M-y again and again brings in earlier and earlier kills. When you have reached the text you are looking for, you do not have to do anything to keep it. Just go on with your editing, leaving the yanked text where it is.

If you M-y enough times, you come back to the starting point (the most recent kill).

>> Kill a line, move around, kill another line.
   Then do C-y to get back the second killed line.
   Then do M-y and it will be replaced by the first killed line.
   Do more M-y's and see what you get. Keep doing them until
   the second kill line comes back, and then a few more.
   If you like, you can try giving M-y positive and negative
   arguments.

* UNDO

If you make a change to the text, and then decide that it was a mistake, you can undo the change with the undo command, C-/.

Normally, C-/ undoes the changes made by one command; if you repeat C-/ several times in a row, each repetition undoes one more command.

But there are two exceptions: commands that do not change the text don't count (this includes cursor motion commands and scrolling commands), and self-inserting characters are usually handled in groups of up to 20. (This is to reduce the number of C-/'s you have to type to undo insertion of text.)

>> Kill this line with C-k, then type C-/ and it should reappear.

C-_ is an alternative undo command; it works exactly the same as C-/. On some text terminals, typing C-/ actually sends C-_ to Emacs. Alternatively, C-x u also works exactly like C-/, but is a little less convenient to type.

A numeric argument to C-/, C-_, or C-x u acts as a repeat count.

You can undo deletion of text just as you can undo killing of text. The distinction between killing something and deleting it affects whether you can yank it with C-y; it makes no difference for undo.

* FILES

In order to make the text you edit permanent, you must put it in a file. Otherwise, it will go away when you exit Emacs. In order to put your text in a file, you must "find" the file before you enter the text. (This is also called "visiting" the file.)

Finding a file means that you see the contents of the file within Emacs. In many ways, it is as if you were editing the file itself. However, the changes you make using Emacs do not become permanent until you "save" the file. This is so you can avoid leaving a half-changed file on the system when you do not want to. Even when you save, Emacs leaves the original file under a changed name in case you later decide that your changes were a mistake.

If you look near the bottom of the screen you will see a line that
begins with dashes, and starts with " -:--- TUTORIAL" or something
like that. This part of the screen normally shows the name of the
file that you are visiting. Right now, you are visiting your personal
copy of the Emacs tutorial, which is called "TUTORIAL". When you find
a file with Emacs, that file's name will appear in that precise spot.

One special thing about the command for finding a file is that you
have to say what file name you want. We say the command "reads an
argument" (in this case, the argument is the name of the file). After
you type the command

   C-x C-f Find a file

Emacs asks you to type the file name. The file name you type appears
on the bottom line of the screen. The bottom line is called the
minibuffer when it is used for this sort of input. You can use
ordinary Emacs editing commands to edit the file name.

While you are entering the file name (or any minibuffer input),
you can cancel the command with C-g.

>> Type C-x C-f, then type C-g. This cancels the minibuffer,
   and also cancels the C-x C-f command that was using the
   minibuffer. So you do not find any file.

When you have finished entering the file name, type <Return> to
terminate it. The minibuffer disappears, and the C-x C-f command goes
to work to find the file you chose.

The file contents now appear on the screen, and you can edit the
contents. When you wish to make your changes permanent, type the
command

   C-x C-s Save the file

This copies the text within Emacs into the file. The first time you
do this, Emacs renames the original file to a new name so that it is
not lost. The new name is made by adding "~" to the end of the
original file's name. When saving is finished, Emacs displays the
name of the file written.

>> Type C-x C-s TUTORIAL <Return>.
   This should save this tutorial to a file named TUTORIAL, and show
   "Wrote ...TUTORIAL" at the bottom of the screen.

You can find an existing file, to view it or edit it. You can also
find a file which does not already exist. This is the way to create a
file with Emacs: find the file, which starts out empty, and then begin
inserting the text for the file. When you ask to "save" the file,
Emacs actually creates the file with the text that you have inserted.
From then on, you can consider yourself to be editing an already
existing file.

* BUFFERS

If you find a second file with C-x C-f, the first file remains
inside Emacs. You can switch back to it by finding it again with
C-x C-f. This way you can get quite a number of files inside Emacs.

Emacs stores each file's text inside an object called a "buffer".
Finding a file makes a new buffer inside Emacs. To see a list of the
buffers that currently exist, type

C-x C-b List buffers

>> Try C-x C-b now.

See how each buffer has a name, and it may also have a file name for
the file whose contents it holds. ANY text you see in an Emacs window
is always part of some buffer.

>> Type C-x 1 to get rid of the buffer list.

When you have several buffers, only one of them is "current" at any
time. That buffer is the one you edit. If you want to edit another
buffer, you need to "switch" to it. If you want to switch to a buffer
that corresponds to a file, you can do it by visiting the file again
with C-x C-f. But there is an easier way: use the C-x b command.
In that command, you have to type the buffer's name.

>> Create a file named "foo" by typing C-x C-f foo <Return>.
   Then type C-x b TUTORIAL <Return> to come back to this tutorial.

Most of the time, the buffer's name is the same as the file name
(without the file directory part). However, this is not always true.
The buffer list you make with C-x C-b shows you both the buffer name
and the file name of every buffer.

Some buffers do not correspond to files. The buffer named
"*Buffer List*", which contains the buffer list that you made with
C-x C-b, does not have any file. This TUTORIAL buffer initially did
not have a file, but now it does, because in the previous section you
typed C-x C-s and saved it to a file.

The buffer named "*Messages*" also does not correspond to any file.
This buffer contains the messages that have appeared on the bottom
line during your Emacs session.

>> Type C-x b *Messages* <Return> to look at the buffer of messages.
   Then type C-x b TUTORIAL <Return> to come back to this tutorial.

If you make changes to the text of one file, then find another file,
this does not save the first file. Its changes remain inside Emacs,
in that file's buffer. The creation or editing of the second file's
buffer has no effect on the first file's buffer. This is very useful,
but it also means that you need a convenient way to save the first
file's buffer. Having to switch back to that buffer, in order to save
it with C-x C-s, would be a nuisance. So we have

   C-x s   Save some buffers

C-x s asks you about each buffer which contains changes that you have
not saved. It asks you, for each such buffer, whether to save the
buffer.

>> Insert a line of text, then type C-x s.
   It should ask you whether to save the buffer named TUTORIAL.
   Answer yes to the question by typing "y".

* EXTENDING THE COMMAND SET

There are many, many more Emacs commands than could possibly be put
on all the control and meta characters. Emacs gets around this with
the X (eXtend) command. This comes in two flavors:

   C-x Character eXtend. Followed by one character.

M-x Named command eXtend. Followed by a long name.

These are commands that are generally useful but are used less than the commands you have already learned about. You have already seen a few of them: the file commands C-x C-f to Find and C-x C-s to Save, for example. Another example is the command to end the Emacs session--this is the command C-x C-c. (Do not worry about losing changes you have made; C-x C-c offers to save each changed file before it kills Emacs.)

If you are using a graphical display, you don't need any special command to move from Emacs to another application. You can do this with the mouse or with window manager commands. However, if you're using a text terminal which can only show one application at a time, you need to "suspend" Emacs to move to any other application.

C-z is the command to exit Emacs *temporarily*--so that you can go back to the same Emacs session afterward. When Emacs is running on a text terminal, C-z "suspends" Emacs; that is, it returns to the shell but does not destroy the Emacs job. In the most common shells, you can resume Emacs with the "fg" command or with "%emacs".

The time to use C-x C-c is when you are about to log out. It's also the right thing to use to exit an Emacs invoked for a quick edit, such as by a mail handling utility.

There are many C-x commands. Here is a list of the ones you have learned:

```
  C-x C-f     Find file
  C-x C-s     Save file
  C-x s       Save some buffers
  C-x C-b     List buffers
  C-x b       Switch buffer
  C-x C-c     Quit Emacs
  C-x 1       Delete all but one window
  C-x u       Undo
```

Named eXtended commands are commands which are used even less frequently, or commands which are used only in certain modes. An example is the command replace-string, which replaces one string with another in the buffer. When you type M-x, Emacs prompts you at the bottom of the screen with M-x and you should type the name of the command; in this case, "replace-string". Just type "repl s<TAB>" and Emacs will complete the name. (<TAB> is the Tab key, usually found above the CapsLock or Shift key near the left edge of the keyboard.) Submit the command name with <Return>.

The replace-string command requires two arguments--the string to be replaced, and the string to replace it with. You must end each argument with <Return>.

>> Move the cursor to the blank line two lines below this one.
   Then type M-x repl s<Return>changed<Return>altered<Return>.

   Notice how this line has changed: you've replaced the word
   "changed" with "altered" wherever it occurred, after the
   initial position of the cursor.

* AUTO SAVE

When you have made changes in a file, but you have not saved them yet, they could be lost if your computer crashes. To protect you from this, Emacs periodically writes an "auto save" file for each file that you are editing. The auto save file name has a # at the beginning and

the end; for example, if your file is named "hello.c", its auto save file's name is "#hello.c#". When you save the file in the normal way, Emacs deletes its auto save file.

If the computer crashes, you can recover your auto-saved editing by finding the file normally (the file you were editing, not the auto save file) and then typing M-x recover-this-file <Return>. When it asks for confirmation, type yes<Return> to go ahead and recover the auto-save data.

* ECHO AREA

If Emacs sees that you are typing multicharacter commands slowly, it shows them to you at the bottom of the screen in an area called the "echo area". The echo area contains the bottom line of the screen.

* MODE LINE

The line immediately above the echo area is called the "mode line". The mode line says something like this:

```
-:**-  TUTORIAL       63% L749    (Fundamental)
```

This line gives useful information about the status of Emacs and the text you are editing.

You already know what the filename means--it is the file you have found. NN% indicates your current position in the buffer text; it means that NN percent of the buffer is above the top of the screen. If the top of the buffer is on the screen, it will say "Top" instead of " 0%". If the bottom of the buffer is on the screen, it will say "Bot". If you are looking at a buffer so small that all of it fits on the screen, the mode line says "All".

The L and digits indicate position in another way: they give the current line number of point.

The stars near the front mean that you have made changes to the text. Right after you visit or save a file, that part of the mode line shows no stars, just dashes.

The part of the mode line inside the parentheses is to tell you what editing modes you are in. The default mode is Fundamental which is what you are using now. It is an example of a "major mode".

Emacs has many different major modes. Some of them are meant for editing different languages and/or kinds of text, such as Lisp mode, Text mode, etc. At any time one and only one major mode is active, and its name can always be found in the mode line just where "Fundamental" is now.

Each major mode makes a few commands behave differently. For example, there are commands for creating comments in a program, and since each programming language has a different idea of what a comment should look like, each major mode has to insert comments differently. Each major mode is the name of an extended command, which is how you can switch to that mode. For example, M-x fundamental-mode is a command to switch to Fundamental mode.

If you are going to be editing human-language text, such as this file, you should probably use Text Mode.

>> Type M-x text-mode <Return>.

Don't worry, none of the Emacs commands you have learned changes in
any great way. But you can observe that M-f and M-b now treat
apostrophes as part of words. Previously, in Fundamental mode,
M-f and M-b treated apostrophes as word-separators.

Major modes usually make subtle changes like that one: most commands
do "the same job" in each major mode, but they work a little bit
differently.

To view documentation on your current major mode, type C-h m.

>> Move the cursor to the line following this line.
>> Type C-l C-l to bring this line to the top of screen.
>> Type C-h m, to see how Text mode differs from Fundamental mode.
>> Type C-x 1 to remove the documentation from the screen.

Major modes are called major because there are also minor modes.
Minor modes are not alternatives to the major modes, just minor
modifications of them. Each minor mode can be turned on or off by
itself, independent of all other minor modes, and independent of your
major mode. So you can use no minor modes, or one minor mode, or any
combination of several minor modes.

One minor mode which is very useful, especially for editing
human-language text, is Auto Fill mode. When this mode is on, Emacs
breaks the line in between words automatically whenever you insert
text and make a line that is too wide.

You can turn Auto Fill mode on by doing M-x auto-fill-mode <Return>.
When the mode is on, you can turn it off again by doing
M-x auto-fill-mode <Return>. If the mode is off, this command turns
it on, and if the mode is on, this command turns it off. We say that
the command "toggles the mode".

>> Type M-x auto-fill-mode <Return> now. Then insert a line of "asdf "
   over again until you see it divide into two lines. You must put in
   spaces between them because Auto Fill breaks lines only at spaces.

The margin is usually set at 70 characters, but you can change it
with the C-x f command. You should give the margin setting you want
as a numeric argument.

>> Type C-x f with an argument of 20. (C-u 2 0 C-x f).
   Then type in some text and see Emacs fill lines of 20
   characters with it. Then set the margin back to 70 using
   C-x f again.

If you make changes in the middle of a paragraph, Auto Fill mode
does not re-fill it for you.
To re-fill the paragraph, type M-q (META-q) with the cursor inside
that paragraph.

>> Move the cursor into the previous paragraph and type M-q.

* SEARCHING

Emacs can do searches for strings (a "string" is a group of contiguous
characters) either forward through the text or backward through it.
Searching for a string is a cursor motion command; it moves the cursor
to the next place where that string appears.

The Emacs search command is "incremental". This means that the

search happens while you type in the string to search for.

The command to initiate a search is C-s for forward search, and C-r
for reverse search. BUT WAIT! Don't try them now.

When you type C-s you'll notice that the string "I-search" appears as
a prompt in the echo area. This tells you that Emacs is in what is
called an incremental search waiting for you to type the thing that
you want to search for. <Return> terminates a search.

>> Now type C-s to start a search. SLOWLY, one letter at a time,
   type the word "cursor", pausing after you type each
   character to notice what happens to the cursor.
   Now you have searched for "cursor", once.
>> Type C-s again, to search for the next occurrence of "cursor".
>> Now type <DEL> four times and see how the cursor moves.
>> Type <Return> to terminate the search.

Did you see what happened? Emacs, in an incremental search, tries to
go to the occurrence of the string that you've typed out so far. To
go to the next occurrence of "cursor" just type C-s again. If no such
occurrence exists, Emacs beeps and tells you the search is currently
"failing". C-g would also terminate the search.

If you are in the middle of an incremental search and type <DEL>, the
search "retreats" to an earlier location. If you type <DEL> just
after you had typed C-s to advance to the next occurrence of a search
string, the <DEL> moves the cursor back to an earlier occurrence. If
there are no earlier occurrences, the <DEL> erases the last character
in the search string. For instance, suppose you have typed "c", to
search for the first occurrence of "c". Now if you type "u", the
cursor will move to the first occurrence of "cu". Now type <DEL>.
This erases the "u" from the search string, and the cursor moves back
to the first occurrence of "c".

If you are in the middle of a search and type a control or meta
character (with a few exceptions--characters that are special in a
search, such as C-s and C-r), the search is terminated.

C-s starts a search that looks for any occurrence of the search string
AFTER the current cursor position. If you want to search for
something earlier in the text, type C-r instead. Everything that we
have said about C-s also applies to C-r, except that the direction of
the search is reversed.

* MULTIPLE WINDOWS

One of the nice features of Emacs is that you can display more than
one window on the screen at the same time. (Note that Emacs uses the
term "frames"--described in the next section--for what some other
applications call "windows". The Emacs manual contains a Glossary of
Emacs terms.)

>> Move the cursor to this line and type C-l C-l.

>> Now type C-x 2 which splits the screen into two windows.
   Both windows display this tutorial. The editing cursor stays in
   the top window.

>> Type C-M-v to scroll the bottom window.
   (If you do not have a real META key, type <ESC> C-v.)

>> Type C-x o ("o" for "other") to move the cursor to the bottom window.
>> Use C-v and M-v in the bottom window to scroll it.

Keep reading these directions in the top window.

>> Type C-x o again to move the cursor back to the top window.
The cursor in the top window is just where it was before.

You can keep using C-x o to switch between the windows. The "selected window", where most editing takes place, is the one with a prominent cursor which blinks when you are not typing. The other windows have their own cursor positions; if you are running Emacs in a graphical display, those cursors are drawn as unblinking hollow boxes.

The command C-M-v is very useful when you are editing text in one window and using the other window just for reference. Without leaving the selected window, you can scroll the text in the other window with C-M-v.

C-M-v is an example of a CONTROL-META character. If you have a META (or Alt) key, you can type C-M-v by holding down both CONTROL and META while typing v. It does not matter whether CONTROL or META "comes first," as both of these keys act by modifying the characters you type.

If you do not have a META key, and you use <ESC> instead, the order does matter: you must type <ESC> followed by CONTROL-v, because CONTROL-<ESC> v will not work. This is because <ESC> is a character in its own right, not a modifier key.

>> Type C-x 1 (in the top window) to get rid of the bottom window.

(If you had typed C-x 1 in the bottom window, that would get rid of the top one. Think of this command as "Keep just one window--the window I am already in.")

You do not have to display the same buffer in both windows. If you use C-x C-f to find a file in one window, the other window does not change. You can find a file in each window independently.

Here is another way to use two windows to display two different things:

>> Type C-x 4 C-f followed by the name of one of your files.
End with <Return>. See the specified file appear in the bottom window. The cursor goes there, too.

>> Type C-x o to go back to the top window, and C-x 1 to delete the bottom window.

* MULTIPLE FRAMES

Emacs can also create multiple "frames". A frame is what we call one collection of windows, together with its menus, scroll bars, echo area, etc. On graphical displays, what Emacs calls a "frame" is what most other applications call a "window". Multiple graphical frames can be shown on the screen at the same time. On a text terminal, only one frame can be shown at a time.

>> Type C-x 5 2.
See a new frame appear on your screen.

You can do everything you did in the original frame in the new frame. There is nothing special about the first frame.

>> Type C-x 5 0.
This removes the selected frame.

You can also remove a frame by using the normal method provided by the graphical system (often clicking a button with an "X" at a top corner of the frame). If you remove the Emacs job's last frame this way, that exits Emacs.

* RECURSIVE EDITING LEVELS

Sometimes you will get into what is called a "recursive editing level". This is indicated by square brackets in the mode line, surrounding the parentheses around the major mode name. For example, you might see [(Fundamental)] instead of (Fundamental).

To get out of the recursive editing level, type <ESC> <ESC> <ESC>. That is an all-purpose "get out" command. You can also use it for eliminating extra windows, and getting out of the minibuffer.

>> Type M-x to get into a minibuffer; then type <ESC> <ESC> <ESC> to get out.

You cannot use C-g to get out of a recursive editing level. This is because C-g is used for canceling commands and arguments WITHIN the recursive editing level.

* GETTING MORE HELP

In this tutorial we have tried to supply just enough information to get you started using Emacs. There is so much available in Emacs that it would be impossible to explain it all here. However, you may want to learn more about Emacs since it has many other useful features. Emacs provides commands for reading documentation about Emacs commands. These "help" commands all start with the character CONTROL-h, which is called "the Help character".

To use the Help features, type the C-h character, and then a character saying what kind of help you want. If you are REALLY lost, type C-h ? and Emacs will tell you what kinds of help it can give. If you have typed C-h and decide you do not want any help, just type C-g to cancel it.

(If C-h does not display a message about help at the bottom of the screen, try typing the F1 key or M-x help <Return> instead.)

The most basic HELP feature is C-h c. Type C-h, the character c, and a command character or sequence; then Emacs displays a very brief description of the command.

>> Type C-h c C-p.

The message should be something like this:

```
C-p runs the command previous-line
```

This tells you the "name of the function". Since function names are chosen to indicate what the command does, they can serve as very brief documentation--sufficient to remind you of commands you have already learned.

Multi-character commands such as C-x C-s and <ESC>v (instead of M-v, if you have no META or EDIT or ALT key) are also allowed after C-h c.

To get more information about a command, use C-h k instead of C-h c.

>> Type C-h k C-p.

This displays the documentation of the function, as well as its name,
in an Emacs window. When you are finished reading the output, type
C-x 1 to get rid of that window. You do not have to do this right
away. You can do some editing while referring to the help text, and
then type C-x 1.

Here are some other useful C-h options:

   C-h f Describe a function. You type in the name of the
          function.

>> Try typing C-h f previous-line <Return>.
   This displays all the information Emacs has about the
   function which implements the C-p command.

A similar command C-h v displays the documentation of variables,
including those whose values you can set to customize Emacs behavior.
You need to type in the name of the variable when Emacs prompts for it.

   C-h a Command Apropos. Type in a keyword and Emacs will list
          all the commands whose names contain that keyword.
          These commands can all be invoked with META-x.
          For some commands, Command Apropos will also list a
          sequence of one or more characters which runs the same
          command.

>> Type C-h a file <Return>.

This displays in another window a list of all M-x commands with "file"
in their names. You will see character-commands listed beside the
corresponding command names (such as C-x C-f beside find-file).

>> Type C-M-v to scroll the help window. Do this a few times.

>> Type C-x 1 to delete the help window.

   C-h i Read included Manuals (a.k.a. Info). This command puts
          you into a special buffer called "*info*" where you
          can read manuals for the packages installed on your system.
          Type m emacs <Return> to read the Emacs manual.
          If you have never before used Info, type h and Emacs
          will take you on a guided tour of Info mode facilities.
          Once you are through with this tutorial, you should
          consult the Emacs Info manual as your primary documentation.

* MORE FEATURES

You can learn more about Emacs by reading its manual, either as a
printed book, or inside Emacs (use the Help menu or type C-h r).
Two features that you may like especially are completion, which saves
typing, and dired, which simplifies file handling.

Completion is a way to avoid unnecessary typing. For instance, if you
want to switch to the *Messages* buffer, you can type C-x b *M<Tab>
and Emacs will fill in the rest of the buffer name as far as it can
determine from what you have already typed. Completion also works for
command names and file names. Completion is described in the Emacs
manual in the node called "Completion".

Dired enables you to list files in a directory (and optionally its
subdirectories), move around that list, visit, rename, delete and
otherwise operate on the files. Dired is described in the Emacs

manual in the node called "Dired".

The manual also describes many other Emacs features.

* CONCLUSION

To exit Emacs use C-x C-c.

This tutorial is meant to be understandable to all new users, so if
you found something unclear, don't sit and blame yourself - complain!

* COPYING

This tutorial descends from a long line of Emacs tutorials
starting with the one written by Stuart Cracraft for the original Emacs.