Putting Regular Expressions into Practice: Sublime Text Editor

To put regular expressions into practice in an environment in which you might use them, lets turn to a popular text editor,
Sublime. Open Sublime from the Applications on your laptop, then from the File menu, Open the file `example_data_leaders.txt`
that you downloaded in the github repository folder.

The file contains some information about former presidents in Peru and Ecuador. It's in an easy-to-read format for humans,
but what if we'd like to convert it to a .csv file so that a computer can easily tell where each piece of information is?
A .csv file (comma separated values) is just like a .txt file, except that we want one line per row (separated by new line
characters), and for each row, we want values separated by commas (which are the columns or cells of each row).

To find and replace in Sublime, go to the **Find** menu and select **Replace...** You can also use the shortcut **COMMAND-OPTION-F**.
A search and replace field will pop up at the bottom of the window. At the far left, there are a few options you can click.
Select the one that looks like `.*` (rather than A*a*) to use regular expressions in the find and replace fields.

Let's start with the first piece of information for each leader: we have a country and a name separated by a colon and space
`:`. Let's replace the two characters `: ` with a single comma `,`. But we want to make sure we only do this right after the
country at the start of a line, in case there are colons in other parts of the text. We can search for both Peru and Ecuador
at the same time, put them in parentheses to capture the matching text as a group, and use the character for the first group
captured $1 in our replace expression.

Enter the following into the **Find What** and **Replace With** fields and click the lower right button **Replace All**:

Find What: (make sure you put a space after the colon)

```
(Peru|Ecuador):
```

Replace With: (no space after the comma)

```
$1,
```

Now let's add the dates, which appear in the next line, to the same line as the country and leader names. We'll need to replace
a newline character with a comma, and we'll also want to separate the two dates into their own cells by replacing the -
between them with another comma. We want to do this for any pair of dates, so let's create a regular expression that will
match a date in the form mm/dd/yyyy. (Note that some of our dates only have one digit for the

month or day). We'll want to capture
both dates as separate groups, so that we can put a comma between them. But we don't want to capture and retain the new line
character before the first one, or the - between them, so lets leave those characters outside the parentheses.

Find What:

```
\n([0-9]{1,2}/[0-9]{1,2}/[0-9]{4}) - ([0-9]{1,2}/[0-9]{1,2}/[0-9]{4})
```

Replace With:

```
,$1,$2
```

How did this work? It worked for most of the entries, but there are a couple that didn't change, because they're the two leaders
still in office (so the second date is "Present"). Let's add a logical OR to the end date, still within the parentheses
for group 2, so that instead of a date format, it might match the string "Present".

Find What:

```
\n([0-9]{1,2}/[0-9]{1,2}/[0-9]{4}) - ([0-9]{1,2}/[0-9]{1,2}/[0-9]{4}|Present)
```

Replace With:

```
,$1,$2
```

Good. Now we've got one more line, which contains each leader's prior experience. There are already commas in that line, but
we might want to keep those in one cell. We can do that in a .csv file by adding double quotes " around the whole string.
This last line doesn't follow a clear format, so how do we find it? Let's use the date from the previous line, since we know
the prior experience line needs to come right after a date. We can copy the group for the end date from our last find/replace
(leaving it in parentheses since we'll want to retain that date), then add a newline character afterwards. We'll also want
to capture the entire next line as a second group. To capture any characters in one line of text, we can use the exclusion
group [^\n]+ that means any character except a newline, with a + meaning we're looking for one or more characters in a row,
and we'll put all of those characters in a second group with parentheses. Then we'll add one more newline at the end, to make
sure the regular expression will match everything up to the next line. We won't keep that second newline (leave it out of
the parentheses) because that will condense our lines so there are no blank lines between leaders. Notice that in the
replacement expression, we're using both groups (the preceding date and the string with prior experience), and we're putting
quotes around the entire second string.

Find What:

```
([0-9]{1,2}/[0-9]{1,2}/[0-9]{4}|Present)\n([^\n]+)\n
```

Replace With:

```
$1,"$2"
```

And there you have it, the file now looks like a .csv file! We can open this in Excel or read it into a programming language
to manipulate the rows and cells. For those familiar with Python or R, we'll demonstrate that next. For others who wish
to keep practicing regular expressions in Sublime, continue on to the following challenges on your own.