Ending a sentence with a preposition is something up with which I will not put.

W Churchill

Some people, when confronted with a problem, think "I know, I'll use regular expressions." Now they have two problems.

Jamie Zawinski
Example

How can you search a file for sentences that end with a preposition?

It seems we need to determine two things:

- what are prepositions?
- what characters might mark the end of a sentence?

The second question seems to be fairly easy: . ! ?

Some sentences end with a double-quotations mark, but that will probably be preceded by one of the marks above. And some end with an ellipsis…

This suggests:

```
[ . ? ! ] \ . \ . \ .
```
So, what are prepositions? A preposition relates a noun or pronoun to another word in a sentence.

One source says there are 150 of them and gives the following partial list:

<table>
<thead>
<tr>
<th>aboard</th>
<th>about</th>
<th>above</th>
<th>across</th>
<th>after</th>
<th>against</th>
</tr>
</thead>
<tbody>
<tr>
<td>along</td>
<td>amid</td>
<td>among</td>
<td>anti</td>
<td>around</td>
<td>as</td>
</tr>
<tr>
<td>at</td>
<td>before</td>
<td>behind</td>
<td>below</td>
<td>by</td>
<td>beside</td>
</tr>
<tr>
<td>besides</td>
<td>between</td>
<td>beyond</td>
<td>but</td>
<td>concerning</td>
<td></td>
</tr>
<tr>
<td>considering</td>
<td>despite</td>
<td>down</td>
<td>during</td>
<td>except</td>
<td>excepting</td>
</tr>
<tr>
<td>excluding</td>
<td>following</td>
<td>for</td>
<td>from</td>
<td>in</td>
<td>inside</td>
</tr>
<tr>
<td>into</td>
<td>like</td>
<td>for minus</td>
<td>near</td>
<td>of</td>
<td>off</td>
</tr>
<tr>
<td>on</td>
<td>onto</td>
<td>opposite</td>
<td>outside</td>
<td>over</td>
<td>past</td>
</tr>
<tr>
<td>per</td>
<td>plus</td>
<td>regarding</td>
<td>round</td>
<td>save</td>
<td>since</td>
</tr>
<tr>
<td>than</td>
<td>through</td>
<td>to</td>
<td>toward</td>
<td>towards</td>
<td>under</td>
</tr>
<tr>
<td>underneath</td>
<td>unlike</td>
<td>until</td>
<td>up</td>
<td>upon</td>
<td>versus</td>
</tr>
<tr>
<td>via</td>
<td>with</td>
<td>within</td>
<td>without</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Allegedly, the most common ones are:

to, of, in, for, on, with, at, by, from, up, about, into, over, after
This suggests the regular expression used below:

```
$ grep -E '(\<to\>|\<of\>|\<in\>|\<for\>|\<on\>|\<with\>|\<at\>|\<by\>|\<from\>|\<up\>|\<about\>|\<into\>|\<over\>|\<after\>)
  (?!\.|\.|\.)' MobyDick.txt
```

once a whale in Spitzbergen that was white all over." --A VOYAGE TO
up a pair of as pretty rainbows as a Christian would wish to look at.
as they possibly can without falling in. And there they stand--miles of
penny that I ever heard of. On the contrary, passengers themselves must
one lodges in.
as a looker on.
the tidiest, certainly none of the finest. I began to twitch all over.
leaving a little interval between, for my back to settle down in. But I
till spoken to. Holding a light in one hand, and that identical New
out a sort of tomahawk, and a seal-skin wallet with the hair on. Placing
he never would have dreamt of getting under the bed to put them on. At
be sure there is more in that man than you perhaps think for.
night previous, and whom I had not as yet had a good look at. They were
to. Then the Captain knows that Jonah is a fugitive; but at the same
an adventurous whaleman to embark from. He at once resolved to accompany
whom I now companied with.
...
The POSIX definition of extended regular expressions includes definitions of some classes of characters, including:

<table>
<thead>
<tr>
<th>POSIX</th>
<th>ASCII</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:alnum:]</td>
<td>[A-Za-z0-9]</td>
<td>alphanumeric characters</td>
</tr>
<tr>
<td>[:alpha:]</td>
<td>[A-Za-z]</td>
<td>alphabetic characters</td>
</tr>
<tr>
<td>[:blank:]</td>
<td>[ \t]</td>
<td>space and tab</td>
</tr>
<tr>
<td>[:digit:]</td>
<td>[0-9]</td>
<td>digits</td>
</tr>
<tr>
<td>[:graph:]</td>
<td>[\x21-\x7E]</td>
<td>visible characters</td>
</tr>
<tr>
<td>[:print:]</td>
<td>[\x20-\x7E]</td>
<td>visible characters and space</td>
</tr>
<tr>
<td>[:lower:]</td>
<td>[a-z]</td>
<td>lower-case letters</td>
</tr>
<tr>
<td>[:upper:]</td>
<td>[A-Z]</td>
<td>upper-case letters</td>
</tr>
<tr>
<td>[:space:]</td>
<td>[ \t\r\n\v\f]</td>
<td>whitespace characters</td>
</tr>
<tr>
<td>[:punct:]</td>
<td>[!&quot;#$%&amp;'()*+,.:/;&lt;=?&gt;@^_`{</td>
<td>}~]-]</td>
</tr>
</tbody>
</table>
Let's use a character class to look for digits in a file (note the syntax):

```
$ grep -E [:digit:] MobyDick.txt
```

In chapters 24, 89, and 90, we substituted a capital L for the symbol:

"Several whales have come in upon this coast (Fife) Anno 1652, one informed), besides a vast quantity of oil, did afford 500 weight of Spermacetti Whales, but did not take of this one whale, amounted altogether to 10,440 yards or nearly six

October 13. "There she blows," was sung out from the mast-head.

--J. ROSS BROWNE'S ETCHINGS OF A WHALING CRUIZE. 1846.

...
Example

Let's use character classes to look for strings that consist of one or more alphabetic characters followed immediately by one or more digits:

```
grep -E '[:alpha:]+[:digit:]+' MobyDick.txt
```

upwards of L1,000,000? And lastly, how comes it that we whalemen of Savesoul's income of L100,000 seized from the scant bread and cheese without any of Savesoul's help) what is that globular L100,000 but a fish high and dry, promising themselves a good L150 from the precious PROVIDED IN PARAGRAPH F3. YOU AGREE THAT THE FOUNDATION, THE
Suppose you need to use a regular expression for a search on a system that does not use ASCII encoding for characters?

The order in which character codes are assigned to characters may not be compatible with ASCII.

So, it could be that A-Z doesn't define a valid range that includes all capital letters and nothing else.

Now, you might be able to figure out a workable range specification…

… but you wouldn't have a portable solution.

The POSIX classes give us a way to manage these issues in a portable manner.

Fortunately, GNU grep does support the POSIX classes described earlier.
Examples

What do you think the following searches will find?

$ grep -E '\<the\>\<Pequod\>' MobyDick.txt

$ grep -E '\<[Cc]aptain\>\<Ahab\>' MobyDick.txt

$ grep -E '\<[Cc]aptain\> \<Ahab\>' MobyDick.txt

$ grep -E '\<better\> \<than\> \<nothing\>' MobyDick.txt

$ grep -E 'better than nothing' MobyDick.txt
-i, --ignore-case
  Ignore case distinctions in both the PATTERN and the input files.

-v, --invert-match
  Invert the sense of matching, to select non-matching lines.

-w, --word-regexp
  Select only those lines containing matches that form whole words. The test is that the matching substring must either be at the beginning of the line, or preceded by a non-word constituent character. Similarly, it must be either at the end of the line or followed by a non-word constituent character. Word-constituent characters are letters, digits, and the underscore.

-x, --line-regexp
  Select only those matches that exactly match the whole line.

-c, --count
  Suppress normal output; instead print a count of matching lines for each input file. With the -v, --invert-match option (see below), count non-matching lines.

-o, --only-matching
  Print only the matched (non-empty) parts of a matching line, with each such part on a separate output line.
More Regular Expressions

-m NUM, --max-count=NUM
Stop reading a file after NUM matching lines. If the input is standard input from a regular file, and NUM matching lines are output, grep ensures that the standard input is positioned to just after the last matching line before exiting, regardless of the presence of trailing context lines. This enables a calling process to resume a search. When grep stops after NUM matching lines, it outputs any trailing context lines. When the -c or --count option is also used, grep does not output a count greater than NUM. When the -v or --invert-match option is also used, grep stops after outputting NUM non-matching lines.

-n, --line-number
Prefix each line of output with the 1-based line number within its input file.

-A NUM, --after-context=NUM
Print NUM lines of trailing context after matching lines. Places a line containing a group separator (--) between contiguous groups of matches. With the -o or --only-matching option, this has no effect and a warning is given.

-B NUM, --before-context=NUM
Print NUM lines of leading context before matching lines. Places a line containing a group separator (--) between contiguous groups of matches. With the -o or --only-matching option, this has no effect and a warning is given.