

Solutions, Week 1, Day 3

With all of these regular expression problems, there are probably many solutions, depending on how you interpret the parameters of the problem, and how strictly you want the pattern to match.

1. Write a single regular expression that will match all of the following - `figure3a`, `figure4x`, `figure9b` - but not the following - `figure3A`, `figureA3`, `figurex7`.

```
figure\d[a-z]
```

2. Write a single regular expression that will match all of the following - `Message34`, `message3a`, `message5x` - but not the following - `MessageA5`, `messagexx`, `Message3A`.

```
[mM]essage\d([a-z]|\d)
```

3. An earlier exercise asked you to write a function to clean up molecular sequences, trimming flanking gaps, changing it all to the same case. Do this with regular expressions.

```
def cleanUpSeq (seq):
    import re
    theStartGaps = re.compile (r'^~+')
    theEndGaps = re.compile (r'\-+$')
    theNewSeq = theStartGaps.sub (seq, '')
    theNewSeq = theEndGaps.sub (theNewSeq, '')
    # case conversion is easier with string methods
    theNewSeq = theNewSeq.lower()
    return theNewSeq
```

4. Develop a regular expression that matches lines more than 70 characters long.

This matches a line in the middle of a file or at the beginning or at the end but what if you have a singleline file?:

```
(\n(.{70,})\n)|(^(.{70,})\n)|(\n(.{70,})$)
```

5. Develop a regular expression to search for headers in HTML and LaTeX. (In HTML header lines start with `<h1>`, `<h2>`, ... `<h6>`. In LaTeX the following headers exist: `chapter`, `section`, `subsection`, `subsubsection`, and so on).

Either of the following would work, but the 2nd is stricter:

```
<[hH]\d>
<[hH][1-6]>
```

But what if you wanted to match any HTML tag? The following pattern will greedily match from the first angle opening bracket to the last closing bracket in strings like `<h6>` `<bold>`:

```
<.*>
```

Often it's better to instead try and match what something *isn't*. This pattern matches pairing brackets, as in `<h6>` and `<bold>`:

<[^>]*>

Latex tag matching could work like:

```
\\(chapter|section|subsection|subsubsection)\\b
```

6. Write a regular expression that matches a nonnegative floating point number without an exponent, like 6.25. Ensure that your script does not match 6:25!

The first allows numbers like “098.7”:

```
\\d+\\.\\d+
```

This ensure that numbers are “0.7” or start with a non-zero:

```
(([1-9]\\d*)|0)\\.\\d+
```

7. Extend your regular expression to match any floating point number without an exponent, like 6.25, 7, or -3.1415926535.

Prefix the previous pattern with something that allows ‘+’, ‘-’ or nothing:

Inline literal start-string without end-string.

```
(\\-|\\+)?\\d+\\.\\d+
```