

Regular expressions

Here are some examples of regular expressions. I only give the template here and explain what it means. Recall that to use these patterns in Perl, you want to enclose the template inside of forward slashes and, if necessary, use the binding operation to apply the regular expression to a variable that is not the default `$_` variable.

Some examples have been taken from *Teach Yourself Regular Expressions in 10 Minutes* by Ben Forta (SAMS Publishing).

1. `[a-z]` matches any lower case letter, so would match any string that has at least one lower case letter in it.

2. `colou?r`

Matches the word “color” and “colour” (as well as other words like “colors”, “colours”, etc). The question mark means that the preceding character appears zero times or one time. It wouldn’t match a misspelling like “colouur”.

3. `in[du]`

The square brackets matches a single character (one of those included in the square brackets). So this pattern will match words like “find”, “finding”, “finds”, “window”, “linux”, “Windows”, “bindle”, etc. It will not match a word like “Index” (recall that, unless you tell the regular expression engine, it pays attention to the (upper/lower) case of the expression).

4. `The(re|y|ir)?`

Matches the words “The”, “There”, “They”, and “Their” (as well as “They’re”, “Theater”, “Therapy”, etc). Again, the ? means that the entire block before it (in parentheses) occurs zero or one times. Without the ?, then the expression would not match the word “The” or “Theatre” as then something inside of the parentheses would need to be present to create a match on the string.

5. `ba{2,4}b`

Curly braces represents an *interval*. Like a ?, it applies to the character (or group) before it. In this case it means that the letter “a” must be included 2, 3, or 4 times. So this will match strings like “baab”, “baaab”, “baaaab”, as well as strings like “cabaabac”, “gcaabaaabgca” (as this includes the substring “baaab”), and an infinite number of other strings. It won’t match “bab” since this includes only one “a” in between the two “b”s.

6. `[A-Z]{1,2}\d`

Will match a string that consists of one or two capital letters followed by a digit. Something like `[A-Z]{2,}` will match a string that consists of two or more characters (nothing after the comma implies that there is no upper limit on the number

of characters), while an expression like `[A-Z]{2}` will match on exactly two capital letters.

7. `[A-Z]{1,2}\d[A-Z\d]? \d[ABD-HJLNP-UW-Z]{2}`

This template is used to match British post codes. Why is this?

A British post code consists of two parts, an *outcode* followed by an *incode*, with the two separated by a space. Examples of British post codes include

```
L69 3BX
CV32 7AL
SW1A 0AA
DH98 1BT
N1 9GU
```

The outcode consists one or two (upper case) alphabetic characters, followed by one or two digits, or one or two (upper case) characters followed by a digit and one (upper case) character. So to match the outcode, you can use a regular expression like `[A-Z]{1,2}\d[A-Z\d]?` to do this. (Convince yourself that this covers the cases mentioned for the incode).

The incode consists of a single digit followed by two (upper case) characters (excluding C, I, K, M, O, and V, in order to avoid confusion). So a regular expression of the form `\d[ABD-HJLNP-UW-Z]{2}` can be used to match the incode. Putting the two regular expressions together, separated by a space, gives the template mentioned to match the full post code.

If you wanted to be generous to someone who might have used more than one space to separate the outcode and incode, you could use a regular expression like

```
[A-Z]{1,2}\d[A-Z\d]? +\d[ABD-HJLNP-UW-Z]{2}
```

to do this. Note carefully the insertion of the “+” symbol following the space. This allows for one or more of the preceding character in the match (i.e. one or more spaces that separates the outcode and incode).

(Note: In reality, there are a few post codes that violate the rules stated above for UK post codes (especially if you include post codes for British overseas territories). So if you wanted a regular expression that would match all post codes, it would have to be more complicated than the one given. Also, the given regular expression will match expressions that *look like* a British post code, but aren’t an actual one, such as `ZZ1 1ZZ`. Again, if you wanted to exclude (some of) the non-valid post codes that match the format, you would need a much more complicated regular expression.)