

The Little Regular Expressionist

Vilja Hulden

August 2016
v0.1b

This little pamphlet, which is inspired by *The Little Schemer* by Daniel Friedman and Matthias Felleisen, aims to serve as a gentle introduction to regular expressions.

You may want to cover the right half of the page, and only move the cover down answer by answer; that way you give yourself time to digest the question and maybe even come up with the answer (sometimes you will have enough information to at least take a guess, though not always).

This pamphlet is by no means exhaustive; there is much more to regular expressions. It's a good idea to test and play; <http://regexr.com/> has both a tester and reference resources, and a good comprehensive cheat sheet can be found at <https://www.cheatography.com/davechild/cheat-sheets/regular-expressions/>.

1 Basics

Is a a regular expression?	Yes, it matches string a .
Is ba*c a regular expression?	Yes.
Does ba*c match bac ?	Yes.
Does ba*c match baaaac ?	Yes.
Does ba*c match bc ?	Yes.
Hmm. So do you mean that ba*c can be read as “ b followed by zero or more a ’s and then a c ?”	I do indeed.

So Mo*re would match both More and Moore ?	Yes, but it would of course also match Mre .
Oh. What if I don't want it to match Mre ?	Use the plus (+) instead of the star (*).
Aha! So ba+c matches bac but not bc ?	Exactly.
OK, so does ba+c match baaaac ?	Yes, because + means "one or more."
And it also matches baac and baaaaac and baaaaaaac and ...	Yes, you've got the idea.
So ba+c can be read as "b followed by one or more a's and then a c?"	Quite so.
OK, let me check a few more. Does b*a match ba ?	Yes.
Does b*a match a ?	Yes, because the star means you don't have to have a b.
Does b+a+ match aaaa ?	No, because the plus means that you have to have at least one b.
Does b+a match bbbba ?	Yes, because you can have as many bs as you like, as long as you have at least one.
Does b*a match bbbb ?	No. The a is not followed by a star, so it has to be there.
Does ba*c match BAC ?	No, because the matches are case-sensitive.

Does <code>b*a</code> match <code>bbBBa</code> ?	No, because the matches are case-sensitive! A lowercase <code>b</code> only matches a lowercase <code>b</code> , not an uppercase <code>B</code> .
Oh, OK. What if I want to match both?	We'll get to that, don't worry.
Fine. What if I want to match any character?	Use <code>.</code> (the period).
Like this: <code>b.c</code> to match <code>bac</code> ?	Yes.
Or the same thing, <code>b.c</code> , to match <code>bxc</code> ?	Yes.
Does <code>b.c</code> match <code>baaaac</code> too?	No.
Does it match <code>bxxxxc</code> ?	No. The period only matches a single character.
Oh. So does <code>b.*c</code> match <code>baaaac</code> ?	Yes!
Does <code>b.*c</code> match <code>bbbbbaaaac</code> ?	Yes, because <code>b</code> is also a character.
Does <code>b.*k</code> match <code>bark</code> ?	Yes, because both <code>a</code> and <code>r</code> are characters.
Does <code>B.*K</code> match <code>bark</code> ?	No, because the matches are case-sensitive. <code>B</code> only matches an uppercase <code>B</code> , not a lowercase <code>b</code> .
OK. But does <code>c.*p</code> match <code>co-op</code> ?	Yes, because <code>-</code> (the hyphen) is a character too.
So <code>c.*p</code> would match <code>c&?\$#e\$p</code> ?	Yes, those are all characters.

Does <code>a.*p</code> match <code>b635%#p</code> ?	No, because there's no <code>a</code> .
Does <code>a.*p</code> match <code>a&4>p</code> ?	Yes.
What if I have a word that looks like this: <code>ab</code> <code>cd</code> Will <code>a.*d</code> match it?	No, the one character that <code>.</code> does not match is the line break.
Oh, so it will only match a string if the string is all on one line?	Yes.
So let's say I have a line like this: <code>Cats beat dogs</code> I want to match the <code>Cats</code> in that line. I write <code>C.*s</code> to do that, right?	Actually, no.
What??	<code>C.*s</code> will match the whole line.
Why??	Because regular expressions are greedy . They take everything they can.
Oh, so <code>C.*s</code> will actually match the whole string <code>Cats beat dogs</code> because that string ends with <code>s</code> too?	Exactly.
So, let me make sure I've got all this right.	Please do.
What's the minimum number of <code>a</code> 's a string has to have for the expression <code>ba*c</code> to match?	Zero.
And what's the minimum number of <code>a</code> 's a string has to have for the expression <code>ba+</code> to match?	One.

Is there an upper limit to how many consecutive a 's there can be in a string for the expressions ba* or ba+ to match?	No.
What does . match?	Any character except line breaks.
And, a regular expression is greedy.	Yes!!
Can I make it “ungreedy?” Can I match the Cats in Cats beat dogs somehow?	Yes, you can add ? to the quantifier (the star or plus).
Like this: C.*?s?	Exactly.

Cheat sheet	
.	any character
*	zero or more
+	one or more
*?	zero or more, ungreedy
+?	one or more, ungreedy

2 Alternatives

Does <code>a b</code> match <code>a</code> ?	Yes.
Does <code>a b</code> match <code>b</code> ?	Yes.
Does <code>a b</code> match <code>c</code> ?	No, there's nothing in the expression that could match <code>c</code> .
Does <code>a b</code> match <code>ab</code> ?	No.
So <code>a b</code> means "a or b"?	Yes!
Does <code>a b+</code> match <code>abbbb</code> ?	No, it only matches one or the other side of the <code> </code> , not both at the same time.
OK. So does <code>a b+</code> match <code>bbbb</code> ?	Yes.
And does <code>a b+</code> match <code>a</code> ?	Yes.
Does <code>a b+</code> match <code>aaa</code> ?	No, because there's only one <code>a</code> in the expression.
Does <code>a+ b+</code> match <code>aaabbb</code> ?	No, it still only matches one or the other side of the pipe!
Oh, right. So does <code>a+ b+</code> match <code>aaa</code> ?	Yes.
Does <code>a+ b+</code> match <code>bbb</code> ?	Yes.
So <code>dog cat</code> matches <code>dog</code> ?	Yes.
But does <code>dog cat</code> match <code>dogs</code> ?	No, there's no <code>s</code> in the expression.

Oh yeah, that's true. But does <code>dog cat</code> match the dog in <code>dogs</code> ?	Yes!
Does <code>dog cat</code> match <code>docat</code> ?	No, because the whole expression on one side of the <code> </code> (the pipe) has to match.
Ah, right. So does <code>dog cat</code> match the <code>cat</code> in <code>docat</code> ?	Yes!
What if I want to match both <code>dogat</code> and <code>docat</code> ?	Group the <code>g c</code> with parentheses.
Like this: <code>do(g c)at</code> ?	Yes. That matches <code>dogat</code> as well as <code>docat</code> .
Does <code>dog cats</code> match <code>dogs</code> ?	No, you can't combine the two sides of the pipe.
Does <code>dog cats</code> match <code>cats</code> ?	Yes, because <code>cats</code> is all on one side of the pipe.
Does <code>(dog cat)s</code> match <code>dogs</code> ?	Yes, because now you've grouped the alternatives.
Does <code>(dog cat)s</code> match <code>cats</code> ?	Yes, because you've grouped the alternatives.
Does <code>(dog cat)s</code> match <code>cat</code> ?	No, because the <code>s</code> has to be there.
Does <code>(dog cat)s*</code> match <code>cat</code> ?	Yes, because now you've made the <code>s</code> optional.

3 Character classes

Does <code>[abc]</code> match <code>a</code> ?	Yes, because <code>a</code> is included in the class.
Does <code>[abc]</code> match <code>b</code> ?	Yes, because <code>b</code> is included in the class.
Does <code>[abc]</code> match <code>d</code> ?	No, because <code>d</code> is not included in the class.
Does <code>[abc]</code> match <code>ab</code> ?	No, because the class only represents one of its members at a time.
Does <code>[abc][abc]</code> match <code>ab</code> ?	Yes, because now there are two classes in a row.
Does <code>[abc][abc]</code> match <code>cb</code> ?	Yes, because the class can represent any one of its members.
Does <code>[abc]+</code> match <code>ab</code> ?	Yes! Very good.
Does <code>[abc]+</code> match <code>ba</code> ?	Yes.
Does <code>b[abc]*</code> match <code>ba</code> ?	Yes.
Does <code>[abc]+</code> match <code>bacab</code> ?	Yes.
So if a class is followed by <code>*</code> or <code>+</code> , any number of the members of that class can be strung together in any order?	Yes (well, to be precise, zero or more for <code>*</code> and one or more for <code>+</code>).
If a class contains all letters between <code>a</code> and <code>h</code> , do I have to list them all, like this: <code>[abcdefgh]</code> ?	No, you can define the range like this: <code>[a-h]</code> .
So, <code>[a-z]</code> matches <code>a</code> ?	Yes.

And <code>[a-z]</code> matches <code>g</code> ?	Yes.
And <code>[a-z]</code> match <code>k</code> ?	Yes. We could go on.
Let's not. But does <code>[a-z]</code> match <code>A</code> ?	No, the class is case sensitive.
Does <code>[a-z]</code> match <code>aa</code> ?	No, because the class only represents one of its members at a time.
Does <code>[a-z]+</code> match <code>abc</code> ?	Yes, of course.
Does <code>[a-z]+</code> match <code>bye</code> ?	Yes, of course.
Does <code>[a-z]</code> match <code>a-z</code> ?	No, the expression defines a class, not a string.
Does <code>[a-k]</code> match <code>k</code> ?	Yes.
Does <code>[a-k]</code> match <code>x</code> ?	No, <code>x</code> is not included in the range <code>a-k</code> .
Does <code>[a-z]</code> match <code>2</code> ?	No, no digits are included in the range <code>a-z</code> .
Does <code>[a-z]</code> match <code>&</code> ?	No, no punctuation marks are included in the range <code>a-z</code> .
Does <code>[0-5]</code> match <code>2</code> ?	Yes.
Does <code>[0-5]</code> match <code>7</code> ?	No, <code>7</code> is not included in the range <code>0-5</code> .
Does <code>[0-5]</code> match <code>b</code> ?	No, no letters are included in the range <code>0-5</code> .

Does <code>[a-z0-9]</code> match <code>b</code> ?	Yes.
Does <code>[a-z0-9]</code> match <code>5</code> ?	Yes.
Does <code>[a-z0-9]</code> match <code>b5</code> ?	No, because the class only represents one of its members at a time.
Does <code>[a-z0-9]+</code> match <code>b5</code> ?	Yes, because both <code>b</code> and <code>5</code> are members of the class and <code>+</code> means one or more.
Does <code>[a-z0-9]+</code> match <code>good4you</code> ?	Yes.
Does <code>[a-z0-9]+</code> match <code>good4you!</code> ?	No, because the exclamation mark is not a member of the class.
Does <code>[a-z0-9!]+</code> match <code>good4you!</code> ?	Yes, because now you've added the exclamation mark to the class.
So if I put any characters inside square brackets, those characters become members of a class?	Yes.
Hey, couldn't I use this to match both uppercase and lowercase letters, like I wanted to do earlier (on page 1)?	Yes!
Like this: <code>[Bb]ob</code> to match both "bob" and "Bob"?	Yes!
Or different spellings, like <code>gr[ae]y</code> to match both <code>grey</code> and <code>gray</code> ?	Absolutely!
OK, so by saying <code>[0-9]</code> I can match anything that's a number.	That's right.

But what if I want to match anything *except* a number? Do I have to make a class that lists everything that's not a number?

No, silly, of course not.

So how do I match anything except a number?

You negate the number class with `^` (a caret).

Oh, like this: `[^0-9]`?

Exactly.

Cheat sheet

<code>a b</code>	a or b
<code>[abc]</code>	a or b or c
<code>[a-z]</code>	any lowercase letter in the range a-z
<code>[A-Z]</code>	any uppercase letter in the range a-z
<code>[Bb]</code>	uppercase or lowercase b
<code>[0-9]</code>	any number in the range 0-9
<code>[0-4]</code>	any number in the range 0-4
<code>[^246]</code>	not 2, 4, or 6
<code>[^0-9]</code>	not a number

4 Special characters and shorthands

So if a period is short for “any character,” then how do I match a period and nothing else?	Good question! You have to “escape” it with a backslash, like this: <code>\.</code>
So <code>\.org</code> would only match <code>.org</code> , not, say, <code>borg</code> ?	That’s right.
Is it the same for <code>*</code> and <code>+</code> ?	Yes. The period, star, and plus are all special characters with a special meaning. You have to escape them to make them represent the literal character.
So writing <code>*borg\.org\+</code> would only match <code>*borg.org+</code> and nothing else?	That’s right.
Hey, could I also match <code>.org</code> by saying <code>[.]org</code> ?	Yes! Inside a character class, only <code>-</code> (the hyphen) and <code>^</code> the caret are special characters.
What if I want to include the hyphen in a character class?	You put it first, since then it can’t define a range.
So <code>[-a-z]+</code> would match <code>bye-bye</code> ?	Yes.
And the caret?	You put it anywhere but first, since it only negates if it’s the first character.
So <code>[a-z^]+</code> would match <code>o^o</code> ?	Yep.
What about if I want to match all whitespace? Do I make a character class?	You could, or you can just say <code>\s</code> to cover spaces, tabs, and the various flavors of newlines.
So <code>kitty\s*cat</code> would match both <code>kittycat</code> and <code>kitty cat</code> ?	Exactly.

Are there more shorthands like that?	You bet. Too many to list here.
Is there a shorthand for “all digits”?	Yes! It’s <code>\d</code> .
So <code>\d</code> matches the same thing as <code>[0-9]</code>	Exactly.
Is there another way to say <code>^[0-9]</code> , too?	Yes, <code>\D</code> .
Oh! So is <code>\S</code> then “any character except whitespace”?	It is!
Speaking of special characters, does the caret mean anything outside a character class?	Yes, it means “beginning of line.”
So <code>^dog</code> would find all lines beginning with <code>dog</code> ?	Exactly.
Is there a character for “end of line,” too?	Yes, <code>\$</code> .

5 Grouping and substitution

Say I want to match <code>kittykittykitty</code> as well as <code>kittykitty</code> . Can I do that with a plus, like I can match <code>aa</code> and <code>aaa</code> with <code>a+</code> ?	Yes! You can make <code>kitty</code> a single group by putting it in parentheses.
Like this: <code>(kitty)+?</code>	Exactly.
So if I replace <code>(kitty)+</code> with <code>doggie</code> , do I get <code>doggiedoggiedoggie</code> ?	No, you get <code>doggie</code> , because <code>(kitty)+</code> matches the whole string, no matter how many times <code>kitty</code> it has.

Oh. So actually to replace <code>kittykittykitty</code> with <code>doggiedoggiedoggie</code> , I should just replace <code>kitty</code> with <code>doggie</code> ?	Yes.
Can I also say <code>(kitty doggie)+</code> and match <i>either</i> <code>kittykittykitty</code> and <code>doggiedoggiedoggie</code> ?	Yes, of course you can. And that will of course also match <code>kitty</code> and <code>doggiedoggie</code> and so on.
Right. So what if I wanted to replace <code>kittykitty</code> with <code>here-kittykitty</code> and <code>doggiedoggiedoggie</code> with <code>here-doggiedoggiedoggie</code> ? Can I do that with one expression?	Yes; you can use a backreference. Any grouped expression is saved and numbered and can be accessed by <code>\$1</code> , <code>\$2</code> , and so on.
So replacing <code>That's a cute (kitty)</code> with <code>I like that \$1</code> would produce <code>I like that kitty</code> ?	Absolutely.
So then can I say replace <code>(kitty doggie)+</code> with <code>here-\$1</code> to get <code>here-doggiedoggie</code> and so on?	No, because now the grouped part only has <code>kitty</code> or <code>doggie</code> once.
Oh, so I'll always get <code>here-kitty</code> or <code>here-doggie</code> and never <code>here-doggiedoggie</code> .	Yes; you have to include the plus in a group.
Would this work: <code>((kitty doggie)+)</code> ?	It would.

Cheat sheet	
<code>\</code>	escape character
<code>\.</code>	period (literal)
<code>\s</code>	any whitespace character (space, tab, newline)
<code>\d</code>	any digit
<code>\S</code>	any non-whitespace character
<code>\D</code>	any non-digit character
<code>(ab)*</code>	ab zero or more times
<code>\$1</code>	cat in <code>kitty(cat)</code>
<code>\$2</code>	s in <code>kitty(cat)(s)</code>
<code>\$1</code>	kittycat in <code>(kitty(cat))</code>
<code>\$2</code>	cat in <code>(kitty(cat))</code>