How to match “anything up until this sequence of characters” in a regular expression?

405

128

Take this regular expression: `/^[^abc]/` . This will match any single character at the beginning of a string, except a, b, or c.

If you add a `*` after it – `/^[^abc]*/` – the regular expression will continue to add each subsequent character to the result, until it meets either an `a` , **or** `b` , **or** `c` .

For example, with the source string `"qwerty qwerty whatever abc hello"` , the expression will match up to `"qwerty qwerty wh"` .

But what if I wanted the matching string to be `"qwerty qwerty whatever "`

...In other words, how can I match everything up to (but not including) the *exact sequence* `"abc"` ?

regex

asked Aug 19 '11 at 16:45
callum
8,154 20 68 123

What do you mean by `match but not including` ? – Toto Aug 19 '11 at 16:53

4 I mean I want to match `"qwerty qwerty whatever "` – not including the "abc". In other words, I **don't** want the resulting match to be `"qwerty qwerty whatever abc"` . – callum Aug 19 '11 at 17:03

In javascript you can just `do string.split('abc')[0]` . Certainly not an official answer to this problem, but I find it more straightforward than regex. – Wylliam Judd May 23 '18 at 17:45

10 Answers

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

```
/.+?(?=abc)/
```

How it works

The `.+?` part is the un-greedy version of `.+` (one or more of anything). When we use `.+`, the engine will basically match everything. Then, if there is something else in the regex it will go back in steps trying to match the following part. This is the **greedy** behavior, meaning **as much as possible to satisfy**.

When using `.+?`, instead of matching all at once and going back for other conditions (if any), the engine will match the next characters by step until the subsequent part of the regex is matched (again if any). This is the **un-greedy**, meaning match **the fewest possible to satisfy**.

```
/.+X/  ~ "abcXabcXabcX"        /.+/  ~ "abcXabcXabcX"
          ^^^^^^^^^^^^                   ^^^^^^^^^^^^

/.+?X/ ~ "abcXabcXabcX"        /.+?/ ~ "abcXabcXabcX"
          ^^^^                           ^
```

Following that we have `(?= {contents} )`, a *zero width assertion*, a *look around*. This grouped construction matches its contents, but does not count as characters matched (**zero width**). It only returns if it is a match or not (**assertion**).

Thus, in other terms the regex `/.+?(?=abc)/` means:

> Match any characters as few as possible until a "abc" is found, without counting the "abc".

edited Jun 7 '18 at 14:12

answered Aug 19 '11 at 17:03
sidyll
43.1k 11 83 133

9 This will probably not work with line breaks, if they are supposed to be captured. – einord Oct 13 '16 at 14:42

30 Outstanding description of the code functionality. – staypuftman Dec 21 '16 at 17:19

3 What's the difference between `.+?` and `.*` ? – robbie0630 Apr 5 '17 at 1:21

96

If you're looking to capture everything up to "abc":

```
/^(.*?)abc/
```

Explanation:

`( )` capture the expression inside the parentheses for access using `$1` , `$2` , etc.

`^` match start of line

`.*` match anything, `?` non-greedily (match the minimum number of characters required) - [1]

[1] The reason why this is needed is that otherwise, in the following string:

```
whatever whatever something abc something abc
```

by default, regexes are *greedy*, meaning it will match as much as possible. Therefore `/^.*abc/` would match "whatever whatever something abc something ". Adding the non-greedy quantifier `?` makes the regex only match "whatever whatever something ".

edited Aug 19 '11 at 17:02

answered Aug 19 '11 at 16:48
Jared Ng
3,359 1 13 17

2 Thanks, but your one *does* include the abc in the match. In other words the resulting match is "whatever whatever something abc". – callum Aug 19 '11 at 17:05

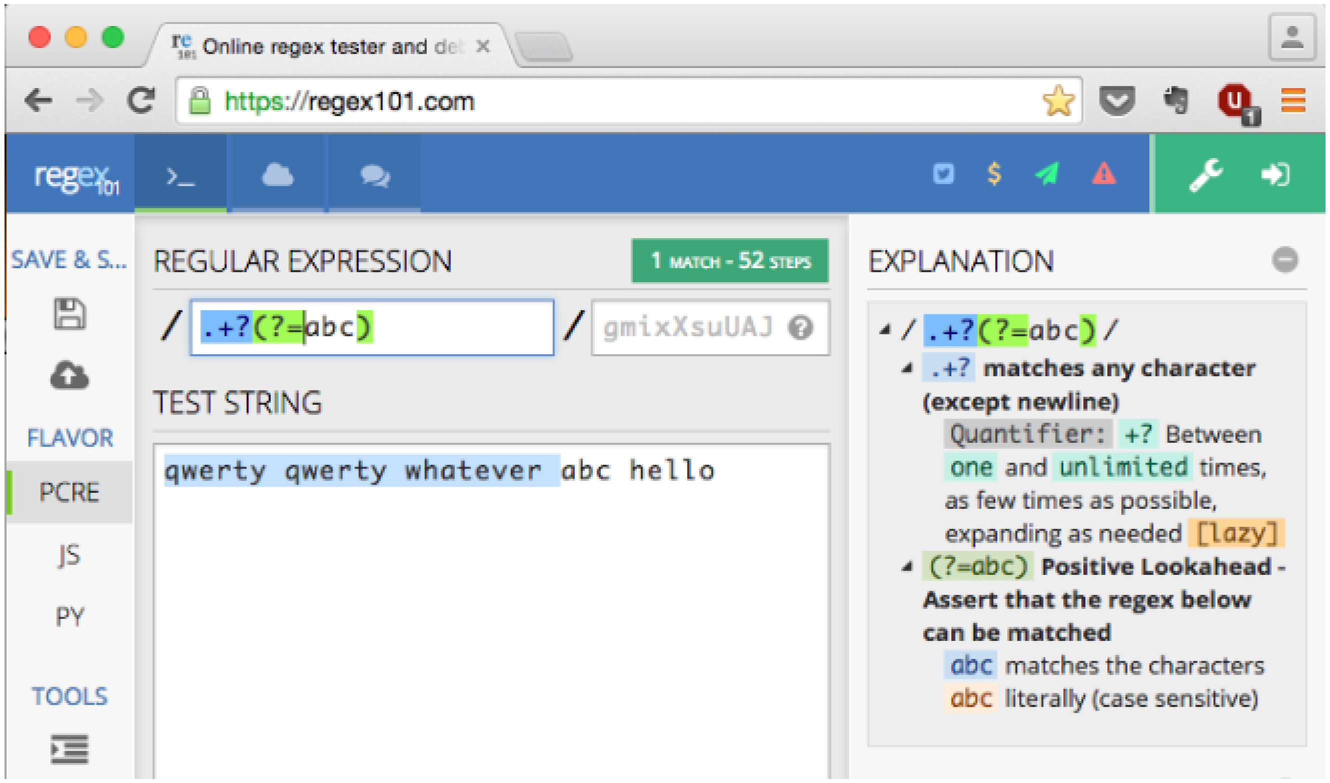
1 Could you explain what you're ultimately trying to do? If your scenario is: (A) You want to get everything leading up to "abc" -- just use parentheses around what you want to capture. (B) You want to match the string up to the "abc" -- you have to check the abc anyway, so it needs to be part of the regex regardless. How else can you check that it's there? – Jared Ng Aug 19 '11 at 17:09

`sed` doesn't seem to support non-greedy matching, nor does it support look-around (`(?=...)`). What else can I do? Example command: `echo "ONE: two,three, FOUR FIVE, six,seven" | sed -n -r "s/^ONE: (.+?), .*/\1/p"` returns `two,three, FOUR FIVE` , but I expect `two,three` ... – CoDEmanX Aug 23 '15 at 14:52

35

As @Jared Ng and @Issun pointed out, the key to solve this kind of RegEx like "matching everything up to a certain word or substring" or "matching everything after a certain word or substring" is called "lookaround" zero-length assertions. [Read more about them here.](#)

In your particular case, it can be solved by a positive look ahead. A picture is worth a thousand words. See the detail explanation in the screenshot.



REGULAR EXPRESSION 1 MATCH - 52 STEPS

```
/ .+?(?=abc) / gmixXsuUAJ
```

TEST STRING

```
qwerty qwerty whatever abc hello
```

EXPLANATION

/ .+?(?=abc) /
- .+? matches any character (except newline)
  - Quantifier: +? Between one and unlimited times, as few times as possible, expanding as needed [lazy]
- (?=abc) Positive Lookahead - Assert that the regex below can be matched
  - abc matches the characters abc literally (case sensitive)

edited Jun 2 '17 at 17:30
Michael
3,883 3 43 63

answered Sep 21 '15 at 19:21
Devy
5,413 3 40 46

8

See: [Lookahead and Lookbehind Zero-Length Assertions]()

Be aware that `[abc]` isn't the same as `abc` . Inside brackets it's not a string - each character is just one of the possibilities. Outside the brackets it becomes the string.

edited Dec 21 '15 at 21:09
kenorb
76.1k 33 435 453

answered Aug 19 '11 at 17:22
aevanko
12.7k 3 44 52

3

For regex in Java, and I believe also in most regex engines, if you want to include the last part this will work:

```
.+?(abc)
```

For example, in this line:

```
I have this very nice senabctence
```

select all characters until "abc" and also include abc

using our regex, the result will be: `I have this very nice senabc`

Test this out: [https://regex101.com/r/mX51ru/1]()

answered Nov 30 '16 at 8:17
Dadan
413 10 27

3

This will make sense about regex.

1. The exact word can be get from the following regex command:

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH Google Facebook

This is the example for "double quoted" words

then we will get "double quoted" from that sentence.

answered May 25 '17 at 6:57
Ponmurugan Mohanraj
41 4

Welcome to StackOverflow and thanks for your attempt to help. I find it however hard to see how this helps the goal stated in the question. Can you elaborate? Can you apply it to the given examples? You seem to focus on handling of `"` , which to me seems irrelevant for the question. – Yunnosch May 25 '17 at 7:07

Hi, I have explained how to get the word or sentences in between the special characters. Here our question is also "anything until the sequence of special characters". so I tried with double quotes and explained it here. Thanks. – Ponmurugan Mohanraj May 25 '17 at 9:08

0

I believe you need subexpressions. If I remember right you can use the normal `()` brackets for subexpressions.

This part is From grep manual:

```
Back References and Subexpressions
      The back-reference \n, where n is a single digit, matches the substring
      previously matched  by  the  nth  parenthesized  subexpression  of  the
      regular expression.
```

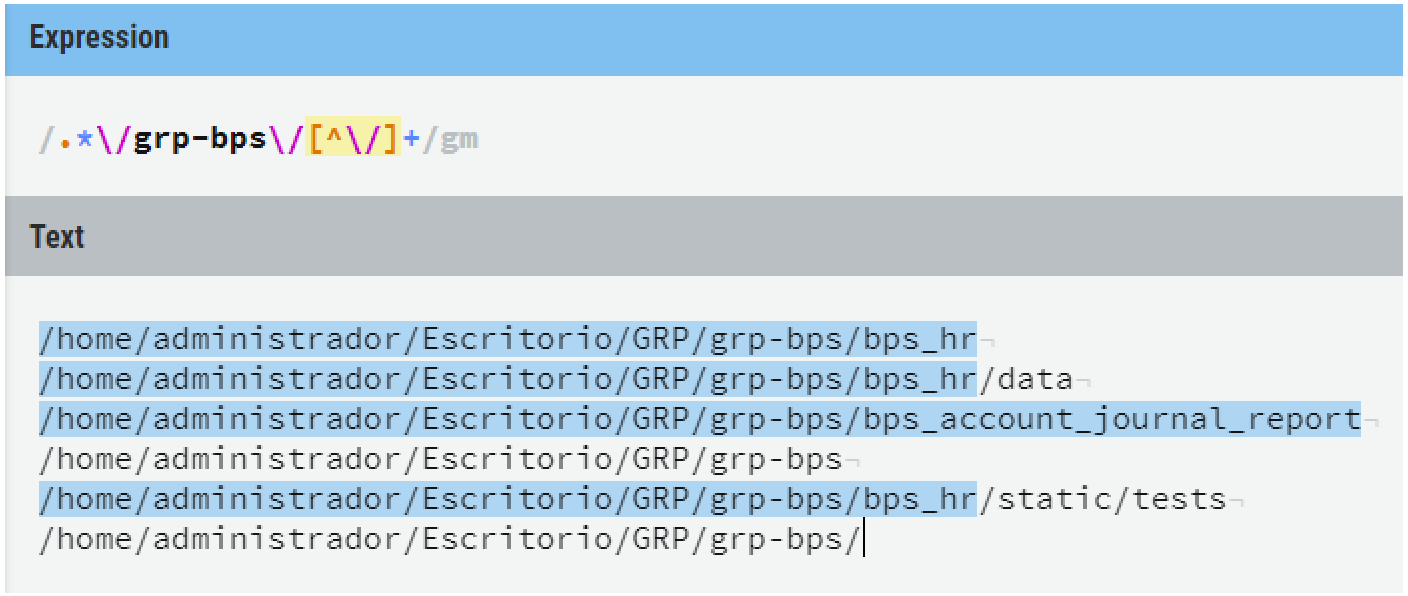
Do something like `^[^(abc)]` should do the trick.

edited Dec 21 '15 at 21:10
kenorb
76.1k 33 435 453

answered Aug 19 '11 at 16:52
Software Mechanic
394 4 19

Sorry, that doesn't work. Putting the abc in parentheses doesn't seem to make any difference. They are still treated as "a OR b OR c". – callum Aug 19 '11 at 17:04

0



```
/.*\/grp-bps\/[^\/]+/gm
```

```
/home/administrador/Escritorio/GRP/grp-bps/bps_hr
/home/administrador/Escritorio/GRP/grp-bps/bps_hr/data
/home/administrador/Escritorio/GRP/grp-bps/bps_account_journal_report
/home/administrador/Escritorio/GRP/grp-bps
/home/administrador/Escritorio/GRP/grp-bps/bps_hr/static/tests
/home/administrador/Escritorio/GRP/grp-bps/
```

As you can see, I needed up to one folder ahead of "grp-bps" folder, without including last dash. And it was required to have at least one folder after "grp-bps" folder.

answered Nov 20 '18 at 18:48
Loaderon
1,090 11 26

3 No text version? 🙄 – kiradotee Feb 18 at 1:49

-1

The `$` marks the end of a string, so something like this should work: `[[^abc]*]$` where you're looking for anything NOT ENDING in any iteration of `abc`, but it would have to be at the end

Also if you're using a scripting language with regex (like php or js), they have a search function that stops when it first encounters a pattern (and you can specify start from the left or start from the right, or with php, you can do an implode to mirror the string).

-3

try this

```
.+?efg
```

Query :

```
select REGEXP_REPLACE ('abcdefghijklmn','.+?efg', '') FROM dual;
```

output :

```
hijklmn
```

edited Oct 28 '16 at 19:42
Mohammad
16.4k 12 38 66

answered Oct 28 '16 at 12:51
Balakrishna Gondesi
120 1 6