

<https://regex101.com/>

Your new best friend <3

# Why use [regex101.com](#)?

The screenshot shows the regex101.com web application interface. At the top, there's a blue header bar with the 'regex101' logo on the left and several social media and sharing icons on the right. Below the header, the main interface is divided into sections: 'REGULAR EXPRESSION' on the left and 'TEST STRING' on the right. In the 'REGULAR EXPRESSION' section, a regular expression pattern is displayed: `:/ ([a-z]{2})\w+`. To the right of the pattern are two buttons: one for options ('/ gm') and one for copy ('copy'). A green box indicates '6 matches, 30 steps (~0ms)'. In the 'TEST STRING' section, the input string is: `regex101: build, test, and debug regex`. The word 'build' is highlighted in green, while 'test', 'and', 'debug', and 'regex' are highlighted in blue. The entire screenshot has a light gray background.

REGULAR EXPRESSION

6 matches, 30 steps (~0ms)

`:/ ([a-z]{2})\w+` / gm copy

TEST STRING

regex101: build, test, and debug regex

# RegEx101 - Basics

- First, lets try to match the word cat in this string:

rat bat cat sat fat cats eat tat cat mat CAT

# RegEx101 - Basics

The screenshot shows the regex101.com interface. The URL bar at the top displays 'regex101.com'. The main area is titled 'regular expressions 101'. On the left, there's a sidebar with 'SAVE & SHARE' options (Save Regex, copy), 'FLAVOR' dropdowns for various engines (PCRE2, PCRE, ECMAScript, Python, Golang, Java 8, .NET), and 'FUNCTION' dropdowns for Match, Substitution, List, and Unit Tests. A 'SPONSORS' section features 'DOPPLER' and 'Authentic'. The central 'REGULAR EXPRESSION' field contains ':r" cat' with a 'gm' flag. The 'TEST STRING' field contains 'rat•bat•cat•sat•fat•cats•eat•tat•cat•mat•CAT'. The 'EXPLANATION' panel details the match: 'cat' matches the characters 'cat' literally (case sensitive). It also explains global pattern flags: 'g modifier: global. All matches (don't return after first match)' and 'm modifier: multi line. Causes ^ and \$ to match the begin/end of each line'. The 'MATCH INFORMATION' panel lists three matches: Match 1 at index 8-11 (cat), Match 2 at index 20-23 (cat), and Match 3 at index 33-36 (cat). The 'QUICK REFERENCE' panel provides a search bar and a list of common regex concepts: All Tokens, Common Tokens (selected), General Tokens, Anchors, and Meta Sequences.

regular expressions 101

regex101.com

regular expressions 101

REGULAR EXPRESSION

:r" cat " gm

TEST STRING

rat•bat•cat•sat•fat•cats•eat•tat•cat•mat•CAT

EXPLANATION

" cat " gm

- cat matches the characters cat literally (case sensitive)

Global pattern flags

- g modifier: global. All matches (don't return after first match)
- m modifier: multi line. Causes ^ and \$ to match the begin/end of each line

MATCH INFORMATION

Match 1 8-11 cat

Match 2 20-23 cat

Match 3 33-36 cat

QUICK REFERENCE

Search reference

- All Tokens
- Common Tokens (selected)
- General Tokens
- Anchors
- Meta Sequences

DOPPLER

All your environment variables, in one place

Authentic

Your new development career awaits. Check out the latest listings.

ADS VIA CARBON

<https://www.sitepoint.com/learn-regex/>

# RegEx101 - Flag

- As we can see, this is case sensitive and will only match the words that are identical.
- We can make it insensitive by changing the flag.
- This will make it possible to match all the cats 😊

# RegEx101 - Flag

The screenshot shows the regex101.com web application interface. The URL in the address bar is `https://regex101.com`. The main area displays a regular expression `:|r| cat` with the flag `" gm`, resulting in `3 matches (12 steps, 0.0ms)`. The test string is `rat•bat•cat•sat•fat•cats•eat•tat•cat•mat•CAT`, with the word `cat` highlighted in blue across all three matches.

**REGEX FLAGS** (shown expanded):

- global**: Don't return after first match (checked)
- multi line**: ^ and \$ match start/end of line (checked)
- insensitive**: Case insensitive match (checked)
- extended**: Ignore whitespace
- single line**: Dot matches newline
- unicode**: Match with full unicode
- ascii**: Make escape sequences perform ASCII-only matching

**EXPLANATION** (partial view):

- `" cat " gm`: `cat` matches the characters `cat` literally (case sensitive)
- Global pattern flags**:
  - g modifier**: global. All matches (don't return after first match)
  - m modifier**: multi line. Causes `^` and `$` to match the begin/end of each line

**MATCH INFORMATION** (partial view):

- Match 1: 8-11 | cat
- Match 2: 20-23 | cat
- Match 3: 33-36 | cat

**JICK REFERENCE** (partial view):

- All Tokens
- Common Tokens** (checked):
  - A si... [abc]
  - A ... [^abc]
  - A c... [a-z]
  - A ... [^a-z]
  - A [a-zA-Z]
  - Any sing... .
- General Tokens
- Anchors
- Meta Sequences

**SAVE & SHARE** (left sidebar):

- Save Regex `%s`
- FLAVOR:
  - PCRE2 (PHP >=7.3)
  - PCRE (PHP <7.3)
  - ECMAScript (JavaScript)
  - Python** (checked)
  - Golang
  - Java 8
  - .NET (C#)
- FUNCTION:
  - Match (checked)
  - Substitution
  - List
  - Unit Tests

**SPONSORS** (bottom left):

- DOPPLER**: All your environment variables, in one place
- Authentic**: Your new development career awaits. Check out the latest listings.

# RegEx101 - Flag

The screenshot shows the regex101.com web application interface. The URL in the address bar is `https://regex101.com`. The main area displays a regular expression search for the word "cat".

**REGULAR EXPRESSION:** `:r" cat`    `" gmi`

**TEST STRING:** `rat•bat•cat•sat•fat•cats•eat•tat•cat•mat•CAT`

**EXPLANATION:**

- Matched text: "cat" (case insensitive)
- Global pattern flags:**
  - g modifier:** global. All matches (don't return after first match)
  - m modifier:** multi line. Causes `\A` and `\Z` to match the begin/end of each line

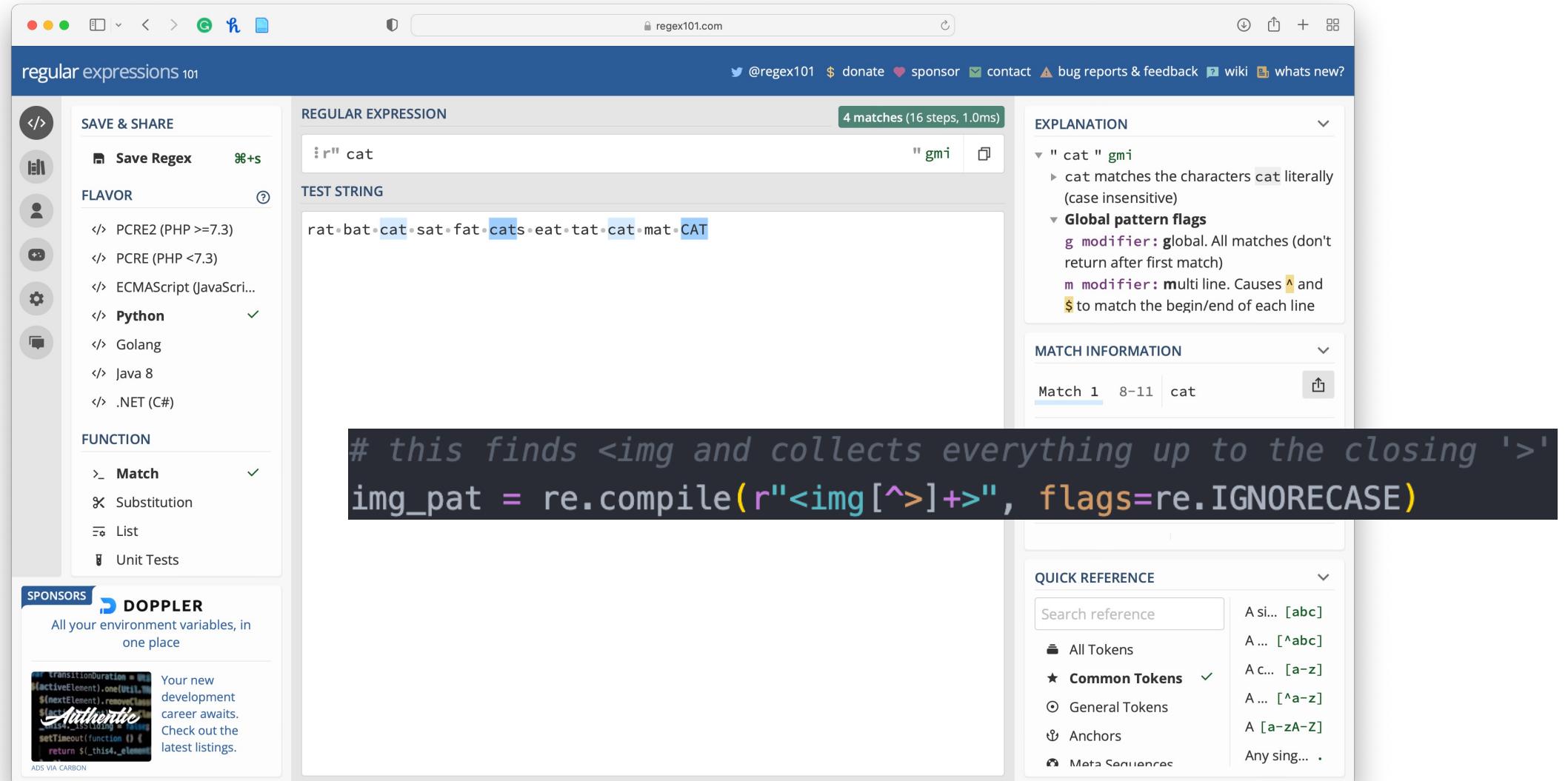
**MATCH INFORMATION:**

- Match 1: 8-11 | cat
- Match 2: 20-23 | cat
- Match 3: 33-36 | cat

**QUICK REFERENCE:**

- Search reference: `A si... [abc]`
- All Tokens: `A ... [^abc]`
- Common Tokens: `A c... [a-z]`
- General Tokens: `A ... [^a-z]`
- Anchors: `A [a-zA-Z]`
- Meta Sequences: `Any sing... .`

# RegEx101 – Flag (example from code)



regular expressions 101

SAVE & SHARE

Save Regex `⌘+S`

FLAVOR

- ✓ PCRE2 (PHP >=7.3)
- ✓ PCRE (PHP <7.3)
- ✓ ECMAScript (JavaScript)
- Python**
- ✓ Golang
- ✓ Java 8
- ✓ .NET (C#)

FUNCTION

- Match
- Substitution
- List
- Unit Tests

SPONSORS

DOPPLER

All your environment variables, in one place

Your new development career awaits. Check out the latest listings.

Authentic

ADS VIA CARBON

REGULAR EXPRESSION

`:r" cat` " gmi

4 matches (16 steps, 1.0ms)

TEST STRING

rat•bat•cat•sat•fat•cats•eat•tat•cat•mat•CAT

EXPLANATION

- " cat " gmi
  - cat matches the characters cat literally (case insensitive)
- Global pattern flags**
  - g modifier:** global. All matches (don't return after first match)
  - m modifier:** multi line. Causes `\A` and `\$` to match the begin/end of each line

MATCH INFORMATION

Match 1 8-11 cat

QUICK REFERENCE

Search reference

- All Tokens
- Common Tokens**
- General Tokens
- Anchors
- Meta Sequences
- Any sing... .

# this finds <img and collects everything up to the closing '>'  
img\_pat = re.compile(r"<img [^>]+>", flags=re.IGNORECASE)

# RegEx101 - Character Sets

- If we wanted to match “bat”, “cat”, and “fat”, can we do this by using character sets, denoted with [] (brackets).
- Basically, you put in multiple characters that you want to get matched.
- For example, [bcf]at will match multiple strings as follows:

# RegEx101 - Character Sets

The screenshot shows the regex101.com interface. The regular expression input field contains `:r" [bcf]at`. The test string is `rat•bat•cat•sat•fat•cats•eat•tat•cat•mat•CAT`. The results section shows 5 matches (45 steps, 0.0ms) with flags `" gm`. The explanation panel details the match for `[bcf]` as a single character present in the list, with `bcf` matching `bat`, `cat`, and `CAT`. The match information panel lists three matches: `Match 1 4-7 bat`, `Match 2 8-11 cat`, and `Match 3 16-19 fat`. The quick reference panel includes a search bar and a list of common tokens like `All Tokens`, `Common Tokens`, `General Tokens`, `Anchors`, and `Meta Sequences`.

regular expressions 101

regex101.com

SAVE & SHARE

Save Regex `⌘+S`

FLAVOR

PCRE2 (PHP >=7.3) `</>`

PCRE (PHP <7.3) `</>`

ECMAScript (JavaScript) `</>`

Python `</> ✓`

Golang `</>`

Java 8 `</>`

.NET (C#) `</>`

FUNCTION

Match `>_ ✓`

Substitution `⌘K`

List `≡`

Unit Tests `U`

SPONSORS

.ENV Sync your .env files, quickly & securely

Authentic Your new development career awaits. Check out the latest listings.

ADS VIA CARBON

REGULAR EXPRESSION

`:r" [bcf]at` `" gm`

TEST STRING

rat•bat•cat•sat•fat•cats•eat•tat•cat•mat•CAT

5 matches (45 steps, 0.0ms)

EXPLANATION

Match a single character present in the list below `[bcf]`

- `bcf` matches a single character in the list `bcf` (case sensitive)
- `at` matches the characters `at` literally (case sensitive)

Global pattern flags

MATCH INFORMATION

Match 1 4-7 bat

Match 2 8-11 cat

Match 3 16-19 fat

QUICK REFERENCE

Search reference

- All Tokens
- Common Tokens `★ ✓`
- General Tokens
- Anchors
- Meta Sequences
- Any sing... .
- A si... [abc]
- A ... [^abc]
- A c... [a-z]
- A ... [^a-z]
- A [a-zA-Z]

# RegEx101 - Ranges

- Let's assume we want to match all words that end with at.
- We could supply the full alphabet inside the character set, but that would be tedious.
- The solution is to use ranges like this [ a - z ]at:

# RegEx101 -Ranges

The screenshot shows the regex101.com web application interface. The URL in the address bar is `https://regex101.com`. The main area displays a regular expression `:r" [a-z]at` with the "gm" flag, resulting in 10 matches (51 steps, 1.0ms). The test string is `rat•bat•cat•sat•fat•cats•eat•tat•cat•mat•CAT`, where the matches are highlighted in blue. The "EXPLANATION" panel details the pattern: it matches a single character from a to z followed by "at". The "MATCH INFORMATION" panel lists three matches: Match 1 at index 0-3 (value "rat"), Match 2 at index 4-7 (value "bat"), and Match 3 at index 8-11 (value "cat"). The "QUICK REFERENCE" panel provides links to common regex concepts like tokens, anchors, and meta sequences.

regular expressions 101

REGULAR EXPRESSION

`:r" [a-z]at` " gm

TEST STRING

rat•bat•cat•sat•fat•cats•eat•tat•cat•mat•CAT

EXPLANATION

Match a single character present in the list below [a-z]  
a-z matches a single character in the range between a (index 97) and z (index 122) (case sensitive)  
at matches the characters at literally (case sensitive)

MATCH INFORMATION

Match 1 0-3 rat

Match 2 4-7 bat

Match 3 8-11 cat

SPONSORS

.env Sync your .env files, quickly & securely

Authentic Your new development career awaits. Check out the latest listings.

ADS VIA CARBON

SEARCH reference

All Tokens

Common Tokens

General Tokens

Anchors

Meta Sequences

A si... [abc]

A ... [^abc]

A c... [a-z]

A ... [^a-z]

A [a-zA-Z]

Any sing...

# RegEx101 - Ranges

- **Partial range:** selections such as [ a - f ] or [ g - p ].
- **Capitalized range:** [ A - Z ].
- **Digit range:** [ 0 - 9 ].
- **Symbol range:** for example, [ # \$ % & @ ].
- **Mixed range:** for example, [ a - z A - Z 0 - 9 ] includes all digits, lower and upper case letters.
- Do note that a range only specifies multiple alternatives for a single character in a pattern.

# RegEx101 - Repeating Characters

- Let's say you'd like to match all three-letter words. You'd probably do it like this: [ a - z ] [ a - z ] [ a - z ]
- This would match all three-letter words.
- But what if you want to match a five- or eight-character word?
- There's a better way to express such a pattern using the { } curly braces notation.
- All you have to do is specify the number of repeating characters.

# RegEx101 - Repeating Characters

Here are examples:

- `a { 5 }` will match “aaaaa”.
- `n { 3 }` will match “nnn”.
- `[ a - z ] { 4 }` will match any four-letter word such as “door”, “room” or “book”.
- `[ a - z ] { 6 , }` will match any word with six or more letters.
- `[ a - z ] { 8 , 11 }` will match any word between eight and 11 letters.
- `[ 0 - 9 ] { 11 }` will match an 11-digit number.

# RegEx101 - Metacharacters

- Metacharacters allow you to write regular expression patterns that are even more compact!
- \d matches any digit that is the same as [ 0 - 9 ]
- \w matches any letter, digit and underscore character
- \s matches a whitespace character — that is, a space or tab
- \t matches a tab character only

# RegEx101 - Metacharacters

- Here are some examples:
- `\w{5}` matches any five-letter word or a five-digit number
- `\d{11}` matches an 11-digit number

# RegEx101 - Special Characters

- Special characters take us a step further into writing more advanced pattern expressions:
  - + : One or more quantifiers (preceding character must exist and can be optionally duplicated).

For example, the expression `c+at` will match “cat”, “ccat” and “cccccccat”. You can repeat the preceding character as many times as you like and you’ll still get a match.
  - ? : Zero or one quantifier (preceding character is optional).

For example, the expression `c?at` will only match “cat” or “at”.

# RegEx101 - Special Characters

\* : Zero or more quantifier (preceding character is optional and can be optionally duplicated).

For example, the expression c\*at will match “at”, “cat” and “ccccccat”.

It’s like the combination of + and ?.

\ : this “escape character” is used when we want to use a special character literally.

For example, c\\* will exactly match “c\*” and not “cccccc”.

# RegEx101 - Special Characters

[ ^ ]: this “negate” notation is used to indicate a character that should not be matched within a range.

For example, the expression b[ ^a - c ]ld will not match “bald” or “bbld” because the second letters a to c are negative.

However, the pattern will match “beld”, “bild”, “bold” and so forth.

. : this “do” notation will match any digit, letter or symbol except newline.

For example, . { 8 } will match a an eight-character password consisting of letters, numbers and symbols. for example, “password” and “P@ssw0rd” will both match.

# RegEx101 - Special Characters

- From what we've learned so far, we can create an interesting variety of compact but powerful regular expressions.
- For example:
  - . + matches one or an unlimited number of characters. For example, "c" , "cc" and "bcd#.670" will all match.
  - [ a - z ]+ will match all lowercase letter words irrespective of length, as long as they contain at least one letter. For example, "book" and "boardroom" will both match.

# RegEx101 - Groups

- All the special characters we just mentioned only affect a single character or a range set.
- What if we wanted the effect to apply to a section of the expression?
- We can do this by creating groups using round brackets — ( ).
- For example, the pattern book( .com ) ?
  - will match both “book” and “book.com”, since we’ve made the “.com” part optional.
- Here’s a more complex example that would be used in a realistic scenario such as email validation:

# RegEx101 - Groups

The screenshot shows the regex101.com interface. The URL bar at the top contains "regex101.com". The main area displays a regular expression pattern and its matches against a test string.

**REGULAR EXPRESSION:** `:r"@\w+\.\w{2,3}(\.\w{2,3})?"` (with "gm" flag)

**TEST STRING:** abc.com  
abc@mail  
@mail.com  
@mail.co.ke

**EXPLANATION:**

- `@\w+` matches the character @ with index 64<sub>10</sub> (40<sub>16</sub> or 100<sub>8</sub>) literally (case sensitive)
- `\w` matches any word character (equivalent to [a-zA-Z0-9\_])
- `.` matches the character . with index 46<sub>10</sub> (2E<sub>16</sub> or 56<sub>8</sub>) literally (case sensitive)
- `\w` matches any word character (equivalent to [a-zA-Z0-9\_])

**MATCH INFORMATION:**

- Match 1: 19-28 @mail.com
- Match 2: 30-41 @mail.co.ke
- Group 1: 38-41 .ke

**FUNCTION:** Match

**SPONSORS:** DOPPLER

All your environment variables, in one place

Your new development career awaits. Check out the latest listings.

**QUICK REFERENCE:**

- Search reference
- All Tokens
- Common Tokens
- General Tokens
- Anchors
- Meta Sequences

<https://www.sitepoint.com/learn-regex/>

# RegEx101 - Alternate Characters

- We can specify alternate characters using the “pipe” symbol: |
- This is different from the special characters we showed earlier, as it affects all the characters on each side of the pipe symbol.
- For example, the pattern sat|sit will match both “sat” and “sit” strings.
- We can rewrite the pattern as s(a|i)t to match the same strings.

# RegEx101 - Starting and Ending Patterns

- You may have noticed that some positive matches are a result of partial matching.
- For example, if I wrote a pattern to match the string “boo”, the string “book” will get a positive match as well, despite not being an exact match.
- To remedy this, we’ll use the following notations:

# RegEx101 - Starting and Ending Patterns

^ : placed at the start, this character matches a pattern at the start of a string.

\$ : placed at the end, this character matches a pattern at the end of the string.

# RegEx101 - Starting and Ending Patterns

- To fix the above situation, we can write our pattern as `boo$`.
- This will ensure that the last three characters match the pattern.
- However, there's one problem we haven't considered yet, as the following image shows:

# RegEx101 - Starting and Ending Patterns

The screenshot shows the regex101.com web application interface. The URL in the address bar is `https://regex101.com`. The main search input field contains the regular expression `^ boo$`. To the right of the input field, there is a green button labeled `3 matches (23 steps, 1.0ms)`. Below the input field is a section titled **TEST STRING** containing the following text:  
boo  
booboo  
sboo  
book

On the left side of the interface, there is a sidebar with several sections:

- SAVE & SHARE**: Includes a "Save Regex" button and a "⌘+s" keyboard shortcut.
- FLAVOR**: A dropdown menu set to "Python". Other options include PCRE2 (PHP >=7.3), PCRE (PHP <7.3), ECMAScript (JavaScript), Python, Golang, Java 8, and .NET (C#).
- FUNCTION**: A dropdown menu set to "Match". Other options include Substitution, List, and Unit Tests.
- SPONSORS**: Advertisements for DOPPLER and Authentic.

The main results area on the right includes the following sections:

- EXPLANATION**: Shows the breakdown of the regex: `^` matches the start of the string, `boo` matches the characters `boo` literally (case sensitive), and `$` asserts position at the end of a line. It also details global and multi-line flags.
- MATCH INFORMATION**: Lists three matches:
  - Match 1: 0-3 | boo
  - Match 2: 7-10 | boo
  - Match 3: 12-15 | boo
- QUICK REFERENCE**: A sidebar with a search bar and links to common tokens, general tokens, anchors, and meta sequences.

# RegEx101 - Starting and Ending Patterns

- The string “sboo” and “booboo” get a match because they still fulfill the current pattern-matching requirements.
- To fix this, we can update the pattern as follows: `^boo$`
- This will strictly match the word “boo”.
- If you use both of them, both rules are enforced.
- For example, `^[a-z]{5}$` strictly matches a five-letter word.
  - If the string has more than five letters, the pattern doesn’t match.

# RegEx101 - Starting and Ending Patterns

The screenshot shows the regex101.com web application interface. The URL bar at the top displays `regex101.com`. The main area is titled "regular expressions 101". On the left, there's a sidebar with "SAVE & SHARE" options (Save Regex, Share, Copy, Print, Embed), "FLAVOR" dropdowns for various programming languages (Python selected), and "FUNCTION" dropdowns for Match (selected), Substitution, List, and Unit Tests. A "SPONSORS" section features logos for Doppler and Authentic. The central workspace shows a "REGULAR EXPRESSION" input field containing `^boo$` with the "gmi" flag applied, resulting in "1 match (19 steps, 1.0ms)". Below it is a "TEST STRING" input field containing the words "boo", "booboo", "sboo", and "book". To the right, the "EXPLANATION" panel details the regex components: `^` asserts position at start of a line, `boo` matches the characters `boo` literally (case insensitive), and `$` asserts position at the end of a line. It also mentions "Global pattern flags" and the "g" modifier. The "MATCH INFORMATION" panel shows Match 1 from index 0 to 3 with the word "boo". The "QUICK REFERENCE" panel provides a search bar and lists common tokens, anchors, and meta sequences.