

REGULÆRE UTTRYKK

En sekvens av karakterer som danner et søkbart mønster.
Dette kan brukes til å ekstrahere ut det man ønsker i en tekst.

Matcher et sett
av angitte
karakterer

Spesialtegn:
Matcher mellomrom

[A-Z][a-z]+\s[A-Z][a-z]+

Matcher forrige token mellom
en og ubegrensede ganger

QUIZ!

`[A-Z][a-z]+\s[A-Z][a-z]+`

New York



Ola Nordmann



Jonas Gahr Støre



new york



Norge



Frogner stadion



SETT VS. GRUPPE

[hallo!]

«hallo! Hvordan går det?»

[«h» «a» «l» «l» «o» «!» «h» «o» «a»]

(hallo!)

«hallo! Hvordan går det?»

[«hallo!»]

SPECIALTEGN

`[aA-zZ0-9_]+@[aA-zZ]+\.[aA-zZ]{2}\b`

`liljacs@uio.no` 
`liljacs@ifi.uio.no` 

Eller ..

`\w+@\w+\.\w{2}\b`

`\w` matcher alle bokstaver inkludert æ, ø, å, é, ä, ñ, o.l. (ikke case sensitive),
tall mellom 0-9, og ‘_’

MYE Å HUSKE?

Side for gruppe I

Eller google «regex cheat sheet» →

Anchors	Assertions	Groups and Ranges
<code>^</code> Start of string, or start of line in multi-line pattern	<code>?=</code> Lookahead assertion	<code>.</code> Any character except new line (<code>\n</code>)
<code>\A</code> Start of string	<code>?!</code> Negative lookahead	<code>(a b)</code> a or b
<code>\$</code> End of string, or end of line in multi-line pattern	<code>?<=</code> Lookbehind assertion	<code>(...)</code> Group
<code>\Z</code> End of string	<code>?!= or ?<!</code> Negative lookbehind	<code>(?:...)</code> Passive (non-capturing) group
<code>\b</code> Word boundary	<code>?></code> Once-only Subexpression	<code>[abc]</code> Range (a or b or c)
<code>\B</code> Not word boundary	<code>?()</code> Condition [if then]	<code>[^abc]</code> Not (a or b or c)
<code>\<</code> Start of word	<code>?() </code> Condition [if then else]	<code>[a-q]</code> Lower case letter from a to q
<code>\></code> End of word	<code>?#</code> Comment	<code>[A-Q]</code> Upper case letter from A to Q
Character Classes	Quantifiers	<code>[0-7]</code> Digit from 0 to 7
<code>\c</code> Control character	<code>*</code> 0 or more {3} Exactly 3	<code>\x</code> Group/subpattern number "x"
<code>\s</code> White space	<code>+</code> 1 or more {3,} 3 or more	Ranges are inclusive.
<code>\S</code> Not white space	<code>?</code> 0 or 1 {3,5} 3, 4 or 5	Pattern Modifiers
<code>\d</code> Digit	Add a <code>?</code> to a quantifier to make it ungreedy.	<code>g</code> Global match
<code>\D</code> Not digit	Escape Sequences	<code>i *</code> Case-insensitive
<code>\w</code> Word	<code>\</code> Escape following character	<code>m *</code> Multiple lines
<code>\W</code> Not word	<code>\Q</code> Begin literal sequence	<code>s *</code> Treat string as single line
<code>\x</code> Hexadecimal digit	<code>\E</code> End literal sequence	<code>x *</code> Allow comments and whitespace in pattern
<code>\O</code> Octal digit	"Escaping" is a way of treating characters which have a special meaning in regular expressions literally, rather than as special characters.	<code>e *</code> Evaluate replacement
POSIX	Common Metacharacters	<code>U *</code> Ungreedy pattern
<code>[upper:]</code> Upper case letters	<code>^</code> [. \$	* PCRE modifier
<code>[lower:]</code> Lower case letters	{ * (\	String Replacement
<code>[alpha:]</code> All letters	+ \ ?	<code>\$n</code> nth non-passive group
<code>[alnum:]</code> Digits and letters		<code>\$2</code> "xyz" in <code>/^(abc(xyz))\$/</code>
		<code>\$1</code> "xyz" in <code>/^(?:abc)(xyz)\$/</code>

REGULÆRE UTTRYKK I PYTHON

```
import re

tekst = "Jeg er på tur i New York"
uttrykk = r"[A-Z][a-z]+\s[A-Z][a-z]+"
tokens = re.findall(uttrykk, tekst)

print(tokens)
```

```
>>> [«New York»]
```

REGEX101

Et genialt verktøy! 🤖

The screenshot displays the regex101.com website interface. The top navigation bar includes links for 'social', 'donate', and 'info'. The main content area is divided into several sections:

- SAVE & SHARE:** Includes a 'Save new Regex' button and an 'Add to Community Libr...' link.
- FLAVOR:** A dropdown menu showing various programming languages and their respective regex engines: PCRE2 (PHP >=7.3), PCRE (PHP <7.3), ECMAScript (JavaScript), Python (checked), Golang, Java 8, .NET (C#), and Rust.
- REGULAR EXPRESSION:** The input field contains the regex `r"^\w+\s\w+\s\w+!$"` with flags `gm`. A status bar indicates '2 matches (27 steps, 0.1ms)'.
- TEST STRING:** The input text is `hei på deg!
hei på deg! Hvordan går det?
Velkommen til IN1140!`. The first line is highlighted in blue, and the second line is highlighted in yellow.
- EXPLANATION:** A detailed breakdown of the regex components:
 - `^` asserts position at start of a line.
 - `\w` matches any word character (equivalent to `[a-zA-Z0-9_]`).
 - `+` matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy).
 - `\s` matches any whitespace character (equivalent to `[\t\n\r\f]`).
- MATCH INFORMATION:** A table showing the matches found in the test string:

Match	Index	Text
Match 1	0-11	hei på deg!
Match 2	41-62	Velkommen til IN1140!

regex101.com