ReGex Patterns

🔤 Pattern	📌 Meaning	🧠 Example Match
.	Any character except newline	a, 1, *
^	Start of string	^Hello → matches "Hello world"
$	End of string	world$ → matches "Hello world"
*	0 or more of previous token	go*gle → ggle, google
+	1 or more of previous token	go+gle → gogle, google
?	0 or 1 of previous token	colou?r → color, colour
{n}	Exactly n repetitions	\d{4} → 2025
{n,}	At least n repetitions	\d{2,} → 1234, 56
{n,m}	Between n and m repetitions	\d{2,4} → 99, 123, 2024
[abc]	Match any one of these	a, b, or c
[^abc]	Not a, b, or c	any character except a/b/c
[a-z]	Range: lowercase a to z	h in [a-z]
[A-Z]	Range: uppercase A to Z	F in [A-Z]
[0-9]	Range: 0 to 9	5
\d	Digit (0–9)	1, 5, 9
\D	Non-digit	a, !, _

\w	Word char: [a-zA-Z0-9_]	a, 5, _
\W	Non-word char	@, #, !
\s	Whitespace: space, tab, newline	' ', \t
\S	Non-whitespace	a, 1, *
\b	Word boundary	\bword\b → matches full word
\B	Not a word boundary	\Bword\B
(abc)	Grouping	(\d+)-(\w+) → 2023-July
(?:abc)	Non-capturing group	Used for grouping without capturing
(?=abc)	Positive lookahead	\w+(?=@) → hello@example.com
(?!abc)	Negative lookahead	\d(?!\d) → last digit only
(?<=abc)	Positive lookbehind	(?<=@)\w+ → example from @example
(?<!abc)	Negative lookbehind	(?<!\d)\w+ → word not after digit
`	`	OR
\\	Escape character	\. matches a literal dot
(?P<name>...)	Named capturing group (Python-style)	Match named groups for readability
(?#comment)	Inline comment in RegEx	Not matched — used to document RegEx
(?i)	Case-insensitive mode	(?i)hello → matches Hello, HELLO

(?m)	Multiline mode	^ and $ match start/ end of each line