Introduction to Regular Expressions

Christopher Ohge, Martin Steer
Riga Technical University, September 2019
Regular expressions

- Sequence of characters that define a search pattern

I watch three climb before it’s my turn. It’s a tough one. The guy before me tries twice. He falls twice. After the last one, he comes down. He’s finished for the day. It’s my turn. My buddy says “good luck!” to me. I noticed a bit of a problem. There’s an outcrop on this one. It’s about halfway up the wall. It’s not a

https://en.wikipedia.org/wiki/Regular_expression
Regular expressions

\[GC]\text{reeeting} \text{ matches } Greeting \text{ and } Creeting
Regular expressions

[0123456789] matches any number
Regular expressions

\[0-9\] matches any number
Regular expressions

\[ [0-9I] \] matches any number or /I/

So \[ [0-9I] \]^+ matches 1717 or /7I7/
Regular expressions

What does \[a-z]+ \] match?
Demo

https://regex101.com

http://txti.es/bleak-housexml
Examples

https://regex101.com

http://txti.es/bleak-housexml

Regular expressions to:

• Match all sentences
• Match all words
• Match all words at end of sentence
• Match all words start with ‘s’
• Match all words ending with ‘s’
• Match all Proper Nouns
Solutions

https://regex101.com

http://txti.es/bleak-housexml

Regular expressions to:

• Match all sentences
• Match all words
• Match all words at end of sentence
• Match all words start with ‘s’
• Match all words ending with ‘s’
• Match all Proper Nouns

. *? \b . *? \b \w* . \bs . *? \b [A-Za-z]+ s [A-Z][a-z]*? \b
Regex practice

• https://regexone.com/
Notes

- `ar` – search for string you want to find
- `a.l` - wildcards or character classes, the dot
- `a.l\.` – to find a fullstop (escape with backslash)
- `.{5}` - quantifiers – number of repetitions, not overlapping
- `!{2}`
- `.*` or `.+` - all or one or more
- `m[aeiou]` – any one character, say vowels, or digits [0-9]
- `m[0-9]+`
- `[A-Z][aeiou]`
- `\w` – any alphanumeric (or `\W` non alpha)
- `\d` – any digit (`\D` non digit)
- `\s` – any whitespace (space, tab, newline) (or `\S` non whitespace)
- `[a-z]+?\s\W`
- `t.*s` - all words begin with T and end with S, then introduce greedy ?
- `\b[Tt]\S+?\s\b` – word break, start with T, end with S, word break