Regex 101

Einführung

- *Keine Library, keine Programmiersprache sondern eine Sequenz an Zeichen welche ein bestimmtes Suchmuster spezifiziert.*

- Ein Text kann aus so zeimlich allem bestehen: Buchstaben, Nummern, Leerzeichen, Sonderzeichen usw.
- Solange das Gesuchte irgend einem Muster folgt, kann es mit Regular Expressions gefunden werden
- Aus jeder Regular Expression kann auch ein deterministisch endlicher Automat erzeugt werden und anders herum

Übersicht

Characters:

- Escape character: \
- Any character: .
- Digit: \d
- Not a digit: \D
- Word character: \w
- Not a word character: \W
- Whitespace: \s
- Not whitespace: \S
- Word boundary: \b
- Not a word boundary: \B
- Beginning of a string: ^
- End of a string: $

Groupings

- Matches characters in brackets: [ ]
- Matches characters not in brackets: [^]
- Either or: |
- Capturing group: ( )

Quantifiers

- 0 or more: *
- 1 or more: +
- 0 or 1: ?
- An exact number of characters: { }
- Range of number of characters: {Minimum, Maximum}
- ? -> umschaltung zu lazy

POSIX	Description	ASCII	Unicode	Shorthand
[:alnum:]	Alphanumeric characters	[a-zA-Z0-9]	[\p{L}\p{Nl} \p{Nd}]	
[:alpha:]	Alphabetic characters	[a-zA-Z]	\p{L}\p{Nl}	
[:ascii:]	ASCII characters	[\x00-\x7F]	\p{InBasicLatin}	
[:blank:]	Space and tab	[\t]	[\p{Zs}\t]	\h
[:cntrl:]	Control characters	[\x00-\x1F\x7F]	\p{Cc}	
[:digit:]	Digits	[0-9]	\p{Nd}	\d
[:graph:]	Visible characters (anything except spaces and control characters)	[\x21-\x7E]	[^\p{Z}\p{C}]	
[:lower:]	Lowercase letters	[a-z]	\p{Ll}	\l
[:print:]	Visible characters and spaces (anything except control characters)	[\x20-\x7E]	\P{C}	
[:punct:]	Punctuation (and symbols).	[!"\#$%&'()*+, \-./:;<=>?@\[\\\]^_‘{|}~]	\p{P}	
[:space:]	All whitespace characters, including line breaks	[\t\r\n\v\f]	[\p{Z}\t\r\n\v\f]	\s
[:upper:]	Uppercase letters	[A-Z]	\p{Lu}	\u
[:word:]	Word characters (letters, numbers and underscores)	[A-Za-z0-9_]	[\p{L}\p{Nl} \p{Nd}\p{Pc}]	\w
[:xdigit:]	Hexadecimal digits	[A-Fa-f0-9]	[A-Fa-f0-9]	
POSIX	Description	ASCII	Unicode	Shorthand

Einfache Beispiele

- Zerlegen von Telefonnummern
- Zerlegen eines Datums
- Zerlegen von Namen
- Extrahieren von URLs
- Erkennen von eMail Adressen
- Zerlegen von Adressen
- Suchen in Arrays
- Greedy vs Lazy

Zerlegen von verschieden formatierten Telefonnummern

Daten:

- 9704443106
- (541) 741 3918
- (603)281-0308
- (814)-462-8074
- 9704443106

Zerlegen eines Datums

Daten:

- 20-02-2019
- 15/07/2020
- 14.09.2021

Zerlegen von Namen

Daten:

- Smith, Mr. John;
- Davis, Ms Nicole;
- Robinson, Mrs. Rebeccca
- Armstrong, Dr Sam;
- Downey, Mr. Robert;

Zerlegen von URLs

Daten:

- https://www.google.com/gmail,
- http://heise.de,
- https://twitter.com/home

Zerlegen von eMail Adresse

- email_pattern = "([a-zA-Z0-9_\\-\\.]+)@([a-zA-Z]+).(.+)"

- ([a-zA-Z0-9_\\-\\.]+) -> 1 or more lowercase letters, uppercase letters, digits, and special characters including underscore, hyphen, and full stop (first capture group i.e. username)
- @ -> at symbol
- ([a-zA-Z]+) -> 1 or more lowercase and uppercase letters (second capture group i.e. domain name)
- . -> a single full stop character
- (.+) -> 1 or more characters (third capture group i.e. domain)

Suchen in Arrays

Daten:

- mylist = ["Hund", "Katze",
  "Maus", "Wildkatze", "Seekuh",
  "Wollmaus", "Katzenfutter"]

- Findet alle Katzen!

Greedy vs Lazy

Daten:

- txt = "<body>\n<h1>test</h1>\n<hr />\n<h2>Text</h2>\n</body>"

- “<.+>” vs. “<.+?>”

Quellen

- https://towardsdatascience.com/regular-expressions-clearly-explained-with-examples-822d76b037b4
- https://regexr.com
- https://www.inf-schule.de/automaten-sprachen/sprachenundautomaten/spracherkennung/regulaeresprachen/theorie_regulaereausdrueckeendlicheautomaten
- http://www.hermann-gruber.com/data/format09-talk.pdf
- https://www.regular-expressions.info/posixbrackets.html#:~:text=POSIX%20bracket%20expressions%20are%20a,start%20negates%20the%20bracket%20expression.