Computersystemen en -architectuur

Oplossingen Reguliere Expressies

1 Ba INF
2022–2023

Kasper Engelen
kasper.engelen@uantwerpen.be

In dit document vind je de oplossingen van de oefeningen over reguliere expressies. De reguliere expressies die het antwoord zijn op de oefening staan in het grijze kader:

```
reguliere expressie
```

Voor sommige oefeningen staat de output die je moet krijgen erbij in een blauw kader:

```
resultaat
```

1 Opdracht 1

Merk op dat de characters `a b c d e f g` allemaal op elkaar volgen, we kunnen dus gewoon de regex gebruiken die alle letters tussen `a` en `g` matcht:

```
[a-g]
```

2 Opdracht 2

We hebben gezien dat we met `^` het begin van een lijn kunnen aangeven en dat we met `[0-9]` cijfers kunnen specifieren. Met de `+` kunnen we meerdere karakters tegelijk vinden. Als we dit combineren krijgen we het volgende:

```
^[0-9]+
```

3 Opdracht 3

Een `.py` filename is gewoon een stuk tekst dat eindigd op `.py`. We kunnen dus één of meerdere karakters vinden met `[a-zA-Z_]+` en daarna `.py` vinden met `\.py`. In de oefeningen hebben we gezien dat om een `.` te vinden, we er een backslash voor moeten zetten. We zetten `\b` voor en na de regex zodat er geen andere karakters voor en na de filename voorkomen. Tenslotte combineren we alles:

```
\b[a-zA-Z_]+\.py\b
```

4 Opdracht 4

We willen hier alle woorden van 4 karakters matchen. Indien we 4 niet-whitespace karakters willen vinden doen we dit met `\S{4}`. Als alternatief kunnen we ook `[a-zA-Z]{4}` gebruiken indien we enkel letters willen. Tenslotte willen we niet gewoon 4 karakters matchen maar specifiek woorden van 4 karakters. Daarom zetten we een `\b` voor en na de reguliere expressie om het begin en het einde van het woord aan te duiden. Alles samen krijgen we één van de twee onderstaande reguliere expressies:

```
\b\S{4}\b
```

```
\b[a-zA-Z]{4}\b
```

5 Opdracht 5

Het eerste cijfer van een getal is altijd 1 tot en met 9, wat overeenkomt met `[1-9]`. Dit cijfer wordt dan eventueel gevolgd door één of twee cijfers van 0 tot 9. We gebruiken daarvoor de character set `[0-9]` en vermelden dat we één of twee karakters willen matchen met `{1,2}`. Het feit dat deze twee laatste cijfers ook weggelaten mogen worden specificeren we met de `?` operator. Tenslotte willen we enkel de getallen 1-999, er mogen dus geen cijfers voor of achter dit getal staan. Dit doen we met `\b`. Alles samen krijgen we:

```
\b[1-9]([0-9]{1,2})?\b
```

6 Opdracht 6

We nemen als voorbeeld de volgende twee adressen:

- `https://www.wikipedia.co.uk/`
- `http://www.google.co.uk/`

Deze adressen komen overeen met de volgende regex:

```
(https?://)?(www\.)?[a-zA-Z.]+\.co\.uk/?
```

Deze reguliere expressie bestaat uit onderstaande delen:

- `(https?://)?` Dit betekent `http` eventueel gevolgd door `s`, gevolgd door `://` en tenslotte een `?` omdat dit deel mag worden weggelaten.
- `(www\.)?` Een `www` gevolgd door een `.` en tenslotte een `?` operator omdat dit deel mag worden weggelaten.
- `[a-zA-Z.]+` Dit is de domeinnaam, die bestaat uit één of meerdere karakters en ook enkele punten.
- `\.co\.uk` Dit is de landcode.
- `/?` Aan het einde van een url mag er eventueel een spatie staan.

7 Opdracht 7

Voor de datums gebruiken we de onderstaande reguliere expressie:

```
\b(3[01]|[1-2][0-9]|0?[1-9])[./\- ](1[0-2]|0?[1-9])[./\- ][12][0-9]{3}\b
```

- `(3[01]|[1-2][0-9]|0?[1-9])` Dit staat voor de getallen 1 tot 31. De verschillende onderdelen zorgen ervoor dat bijv een `3` enkel gevolgd kan worden door `0` of `1`, dat `1` of `2` gevolgd wordt door een cijfer `0` tot `9`, en dat het anders één enkel getal `1` tot `9` is.
- `[./\- ]` De onderdelen van de datum worden gescheiden door een `'.'`, `'/'`, `'-'` of spatie.
- `(1[0-2]|0?[1-9])` De maanden van `1` tot `12`. De `0?` zorgt ervoor dat bijv april zowel als `4` en `04` kan worden geschreven.
- `[./\- ]` De onderdelen van de datum worden gescheiden door een `'.'`, `'/'`, `'-'` of spatie.
- `[12][0-9]{3}` Een jaartal tussen 1000-2999 is een `1` of `2`, gevolgd door drie cijfers `0` tot `9`.
- Voor en na de regex staat er `\b` omdat er voor en na de datum geen andere cijfers en letters mogen staan.

8 Opdracht 8

Een IP adres bestaat uit vier keren een getal tussen 1 en 255 met een punt ertussen. De reguliere expressie voor getallen tussen 1 en 255 is

```
(2[0-4][0-9]|25[0-9]|1[0-9]{2}|[1-9][0-9]|[0-9])
```

Deze bestaat uit onderstaande onderdelen:

- `2[0-4][0-9]` Voor de getallen 200 t.e.m. 249
- `25[0-9]` 250 t.e.m. 255
- `1[0-9]{2}` 100 t.e.m. 199
- `[1-9][0-9]` 10 t.e.m. 99
- `[0-9]` 0 t.e.m. 9

Merk op dat de volgorde van deze onderdelen belangrijk is, anders zullen bepaalde getallen niet herkend worden. We willen tenslotte deze expressie 4 keer herhalen, telkens met een punt ertussen. We plaatsen een `\b` voor en na de regex want voor en na het IP adres mogen er geen andere cijfers en letters staan. Als we dit combineren krijgen we het volgende:

```
\b((2[0-4][0-9]|25[0-9]|1[0-9]{2}|[1-9][0-9]|[0-9])\.){3}
(2[0-4][0-9]|25[0-9]|1[0-9]{2}|[1-9][0-9]|[0-9])\b
```

9 Opdracht 9

We downloaden de file `test.html` van de MSDL-website met daarin de onderstaande HTML code:

```
Dit is <b>vetgedrukte tekst</b> en dit is <strong>belangrijke
tekst</strong>.
```

We gebruiken dan onderstaand commando, en we zien dat de HTML tags eruit gefilterd zijn:

```
cat test.html | sed -E 's:<([^>]+)>([^<]*)</\1>:\2:g'
```

```
Dit is vetgedrukte tekst en dit is belangrijke tekst.
```

Wanneer we de gebruikte reguliere expressie bekijken, zien we dat deze bestaat uit verschillende onderdelen:

- `<([^>]+)>` De regex zal zoeken naar een < gevolgd door tekens die geen > zijn, gevolgd door één >. Dit is om te voorkomen dat er dubbele > voorkomen. De capture tussen de haakjes zal de naam van de HTML tag onthouden.
- `([^<]*)` Dit is de tekst tussen twee html tags. De regex zal zoeken naar eenderwelk teken dat geen < is. Dit is om dubbele < te voorkomen. De capture zal de tekst tussen de tags onthouden.
- `</\1>` Deze regex zal opzoek gaan naar een HTML tag met dezelfde naam als de eerste capture. Hierdoor worden enkel twee HTML tags gevonden als ze dezelfde naam hebben.
- `\2` Deze zorgt ervoor dat de HTML tags verwijderd worden terwijl de tekst ertussen (dit is de tweede capture) behouden blijft.