LOAL - Assignment: *Regular Expressions*

Objective

The goal of this assignment is to get familiar with *RegEx* to detect patterns in text. The exercises include typical real-world tasks you could face in your future.

Things To Learn

- Get familiar with *RegEx*

Materials

- Any *RegEx* testing tool, e.g.
  - regex101.com
  - regexpal.com
  - …

Submission Guidelines

A file (*PDF*, *Word*, *Markdown*…) containing your solutions and answers to the following tasks. For each task include: - Your *RegEx* pattern formatted in a *monospace* font. - A screenshot of your pattern applied on the test string.

Tasks

Course Abbreviations

Write a *RegEx* that matches our school's course abbreviations. Check with the strings below.

```
Should match:
0GGP
0SYP
0RK
0D
0BOBW
0AM
1NSVN
2NSVN
0TINF
1DBI
2DBI
0ETH
Must not match:
0
A
123
```

```
Obobw
LOAL
dbi12
PRO1
3NSVN
0PROGR
1DBI2
```

***Mayer*, *Mair*, *Meier*...**

Write a *RegEx* that matches variations of the surname *Mayer*, *Mair*.... Check with the strings below.

```
Mueller
Meyer //should match
Schmidt
Meier //should match
Schneider
Maier //should match
Fischer
Mayer //should match
Weber
Mayr //should match
Wagner
Mair //should match
Becker
Meir //should match
Schulz
```

Web Colors

Write a *RegEx* that matches web colors. Check with the strings below.

```
The following strings should match:
#1f1f1F
#AFAFAF
#1AFFa1
#222fff
#F00
#727
The following strings must not match:
123456  //Must start with a "#" symbol.
#afafah //"h" is not allowed, valid letters range from "a" to "f".
#123abce //Length must be either 3 or 6.
aFaE3f //Must start with a "#" symbol.
F00 //Must start with a "#" symbol.
#afaf //Length must be either 3 or 6.
```

```
#skv //Valid letters range from "a" to "f".
```

Detecting *HTML*-Tags

Write a regular expression that matches html tags.

```
<h4>Headline</h4>
<some completely weird line which is not allowed not match</>
<div id="unique">a block</div>
<ul class="menu">
<table id="raw_data" class="highlighted">
<H1>The title at the end</H1>

Here in between some ordinary text which, of course, must not match.
But here another line with <b>bold</b> letters, to be matched.

<ul id="main-navigation">
    <li id="about-tab"><a href="./index.html">About</a></li>
    <li id="diary-tab"><a href="./diary.html">Diary</a></li>
    <li id="recipes-tab"><a href="./recipes.html">Recipes</a></li>
    <li id="soccer-tab"><a href="./soccer.html">Soccer</a></li>
    <li id="music-tab"><a href="./music.html">Music</a></li>
</ul>
```

Dollars And Cents

Write a *RegEx* that matches US-American currency strings. Check with the strings below.

```
$149.5
$150
$ 150
This must not match
This makes $ 148.32
But $ 1.135 is too accurate and must not match too
$ 3
$3. is somehow stupid and must not match
```

HTTP-URLs

Write a *RegEx* that matches *HTTP-URLs*. Check with the strings below.

```
http://www.blauweiss-linz.at match.
http://www.htl-leonding.ac.at match.
https://regex101.com/ match.
htp://www.orf.at has a misspelled scheme name, must not match.
https://orf.at this matches though.
http://sport.orf.at even matches with a subdomain.
```

```
http:/widzew.com misses a slash and must not match.
http://www.widzew.com is fine though.
https://widzew.com/historia is also fine.
https://widzew.com/historia/index.html is fine, even with a specific page.
http://too-short is too short to be matched.
http://ww3.aec.at/ is fine, can also contain digits.
https://3ww.aec.at is fine as well, digits can be at the start.
https://-w3.aec.at does not match, hyphens at start are not allowed.
https://w3-.aec.at does not match, hyphens at end are not allowed.
```

Prolog Facts

Write a regular expression that matches facts written in *Prolog*. Check with the strings below.

```
food(barszcz). /*Should match.*/
food = potatoes; /*Is just weird, should not match. */
food(pizza). /*Should match.*/
food[asparagus]. /*Incorrect brackets, should not match. */
food(pierogi) /*Missing period, should not match.*/

likes(michal, barszcz). /*Multiple parameters match.*/
likes(Everyone, pizza). /*Also works with variables and constants.*/
likes(Nobody, asparagus,). /* Problem with commas, should not match.*/
likes(tom, pot8oes). /*Variables/constants can even contain numbers.*/
likes(Somebody, 7up). /*Not at the start though, must not match.*/
likes(guy_fieri, _). /*Variables/constants can contain underscores.
Anonymous variables are also allowed, especially in flavor town.*/
likes(9ine, 7even). /*Variables/constants can't start with numbers.*/
likes(,Everything). /* Another problem with commas, should not match.*/
likes(no commas). /* And another problem with commas, should not match.*/
likes(stinka, cat_food) /*Missing period, should not match.*/
likes(,). /* Empty parameters should not match.*/

alotoffood(barszcz, potatoes, pizza, pierogi). /*Also works with a lot of parameters.*/
```

European Style Dates

Write a regular expression that matches European written dates. They are in the form *dd.mm.yyyy*, where the separator . can also be - or a blank. The leading digit *d* or *m* can be omitted. Valid years are between *1900* and *2099*. But the leading two *yy* can also be omitted. Check with the strings below.

```
22.12.1992 Line 1 valid
35.05.2012 invalid since May 35th does not work
2.7.97 Line 2 valid
1.13.2000 invalid since only 12 months exist
```

```
30-6-1946 Line 3 valid
5.12.1867 Valid date but too old for us
30.2.1900 Line 4 valid though strictly speaking invalid but too tricky for us to fix
5-10.2019 Line 5 valid though strictly speaking invalid but we would need
backreferencing so we can't fix this
03 02 2020 Line 6 valid
```