DAY 3
RegEx (Regular Expressions)

Why RegEx?

- Helps programmers find specific patterns in text.
- Transcends languages, and is generally a CS skill that ought to be learned
- Needs problem solving abilities

Where can I run RegEx?

- RegEx can be run in a local interactive fashion using Atom’s and Sublime’s Ctrl+F put RegEx enabled along with Match Case settings.
- There are also a few website like regexer.com and regex101.com that have interactive UI.

Literal Search using RegEx

- Searching for a literal string can be done by just typing the string in the bar
- Meta Characters can’t be searched for as strings. Thus they need a special escape character, that is the backslash ‘\’. The backslash precedes the special character to make it seem recognizable as a string
    - Meta characters refer to characters that are innately used as means to write logic in RegEx itself.
    - Meta characters include .[]{}()\^$|?*+
    - E.g. Finding a period in a sentence
        - Using . selects every letter in the sentence
        - Using \. Selects every period in the sentence

RegEx Characters for Matching

Characters RegEx	**Description**
.	Any character except new line
\d	Digit (0-9)
\D	Not a digit (0-9)
\w	Word Character (a-z, A-Z,0-9,_)
\W	Not a Word Character
\s	Whitespace (space, tab, newline)
\S	Not a whitespace
Anchor RegEx (match positions)	**Description**

\b	Word Boundary
\B	Not a Word Boundary
^	Beginning of a string
$	End of a string
Grouping RegEx	**Description**
[]	Matches Characters in brackets
[^]	Matches Characters NOT in brackets
|	Either Or
( )	Group
Quantifiers	**Description**
*	0 or more
+	1 or more
?	0 or 1
{3}	Exact
{3,6}	Range (Min, Max)

RegEx Examples

- **Phone Number RegEx**
  - Let's assume 999-999-9999 and 999.999.9999 are valid phone numbers
  - [ ] creates a character set that can be used to make a decision of having multiple characters being accepted.
  - Note that meta characters don't *need* to be escaped in a character set
  - \d\d\d.\d\d\d.\d\d\d\d *works*, but it accepts 999_999_9999 as well.
  - Here we must use a character set [-.]
  - So the RegEx becomes \d\d\d[-.]\d\d\d[-.]\d\d\d\d
  - We can refine this using quantifiers
    - \d{3}[-.]\d{3}[-.]\d{4}

- **Phone Numbers that start with 800 or 900 RegEx**
  - Taking from the previous example we can use character sets to check if the number starts with 800 or 900
  - [89]00[-.]\d\d\d[-.]\d\d\d\d

- **Identifying names with the Mr title in front of their names**
  - ‘?’ means one or zero. We can use this to identify “Mr” even if “.” is present or absent.
  - Mr\.?\s[A-Z]\w* is the corresponding RegEx

- **Identifying names with the Mr., Ms., or Mrs. title in front of their names**
  - M[rs][s]?\.?\s[A-Z]\w* is the corresponding RegEx using **Character Sets**
  - M(r|s|rs)\.?\s[A-Z]\w* is the corresponding RegEx using **Groups**

- **Identifying Emails with specific format**
  - (\w|\-)+\w+\.?\w+@\w+\.\w+ is the corresponding RegEx

- **Identifying Website URL’s.**
  - Check Groups Related Example below
  - https?://(www\.)?(\w+)(\.\w+)
- **Identifying anchor tags in html documents**
  - Here if the regex transcends through lines, we must ensure our RegEx is capable of doing that. For this we use groups.
  - Also, when we do this, our RegEx starts to get greedy. Yes Greedy!
  - Thus we must convert it to make it non-greedy by employing “?”.
  - \<a\shref=\"(.|\n)*?<\/a> is the corresponding RegEx

Nuances in RegEx

- Character Set Related
  - The dash “ - “ , can be used as an actual meta literal, and also to specify a range of values when placed between two word characters of the same type.
    - [1-7] selects characters in a range of numbers inclusive of both 1 and 7
  - When ^ is used in a character set, the innate nature of the ^ to select beginning of strings, is disregarded and all the characters that aren’t present in the character set are selected.
    - [^a-z] selects all characters that aren’t lowercase alphabets.
- Groups Related
  - Groups can be used to capture leveled information
    - E.g. We are given a RegEx https?://(www\.)?(\w+)(\.\w+)
    - We can use $1, $2, $3 to signify the various groups selected.
    - For https://www.google.com
      - $1 will be www.
      - $2 will be google
      - $3 will be .com