NLP Sheet 1

1) Write a regular expression that matches any element in the set of

a) All Alphabetic Strings

→ Means any sequence of alphabetic characters

→ Clearly

/[a-zA-Z]+/ } one or more alphabetic characters

b) All lowercase alphabetic Strings ending in b

→

/[a-z]*b/

[a-z]* : a sequence of alphabetic char. (can be nothing)

b : ends with b

• Notice that given abcb → The * matches the largest String it can

c) All Strings from the alphabet {a, b} such that each a is immediately Preceded by and Followed by b)

→ This means that such sequence is only allowed

① to start with b (bbb... is a general begining) (bbb... : one or more)

→ If an a ever wants to appear then what

② must come after is any ≥1 no. of bs

(bbbbabbb...)

This scenario can repeat anytime later (e.g. →)

③ For any no. of times

Combining ①, ②, ③

$/ b + (ab+) * /$

① ② ③ · applies on ()

d) All binary strings with at least 4 1s

→ Should be a sequence of 0s and 1s

→ 1 should appear 4 times or at least

* Consider the 1st 4 or 1s in a random binary string. They must be seperated by 0s from each other. ① (any no. of)

* The rest of the string can be any binary string ②

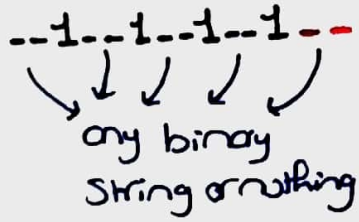
$/ \underbrace{0*10*10*10*1}_{①} \underbrace{[01]*}_{②} /$

This is clearly equivalent to the expression

$/ \underbrace{(0*1)\{4\}}_{①} \underbrace{[01]*}_{②} /$

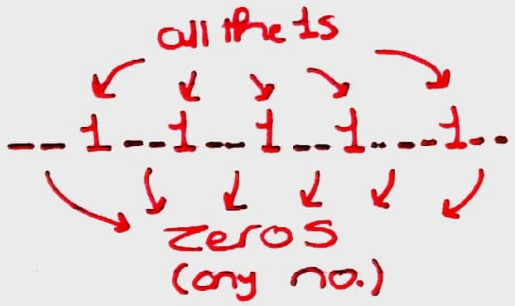
* Another way to think about is that if we spot 4 1s anywhere in a binary string then what seperates them is an arbitrary binary string (of any length)

/([01]*1){4}[01]*/



* Another way to think about it is that all the ones in a random binary string are Pair-wise seperated by any no. of zeros.

i.e. /(0*1){4,}0*/



* which can be also written as

/(0*10*){4,}/

. we used 1+ in tut.

"In a random binary string, every single 1 you find will be seperated from the next by zeros. (any no.) In our case, this must happen 4 times" (or more)

. This hopefully covers the answers we considered in the tutorial

e) All binary strings where the no. of zeros is a multiple of 3.

→ Need 0, 3, 6, 9, ... Zeros

→ They are seperated from each other by 1s

/(1*01*01*01*)+/

Don't want to match ""

. This assumes no zeros isn't allowed

• We can write the previous as /((1* 0 1*){3})+/

→ To allow that there can be no zeros in the expression (just 1s)

/((1*01*){3})+| 1 +/

⇒ Notice that we never resorted to bordaries because we were never constrained to do so (bordaries help guarantee the matched words are surronded by the right characters)

2) Define a word as
→ Sequence of alphabetic characters
→ Seperated from other words through white space, any relevant punct., line breaks,...

Does \b still work as a seperator here?

• In regex a word boundary is a position where a non-regex word occurs
↳ [a-zA-Z0-9_]
(anything that isn't a letter, no. or _ can be there)
↳ includes white space, relevant punct. and more that wasn't strictly disallowed
Thus, also the word's definition has changed, \b still works as a seperator.

* Write Regex to find elements belonging to the set of

a) All strings with two consecutive repeated words in the same case

capturing group

/\b([a-zA-Z]+)\S+(\1)\b/

This also assumes that the two words can be only seperate by spaces (and not punc. marks).
- assume any amount of space between them is okay (examples show 1)

Perhaps, [\W]+ instead of \S+ can work for that

b) All strings that start the beginning of the line with with an integer and that end the end of the line with a word.

/^\d+.*[a-zA-Z]+$/ (1)

Start with an int of 1 or more digits

.* ?!

ends with a word

but Notice
- → its sufficient to check that it starts with one integer (rest in .*) } Drop 1st +
- → integers can be -ve } add (-)?
- → to be a word (1) must be preceded by a word boundary but .* allows anything (even nos)

Hence, need to rewrite as

/^(-)?\d.*\b[a-zA-Z]+$/

- can drop ()

→ last char. in .* must be a seperator.

c) All strings that have both the word grotto and the word raven in them (neither should be a substring of another word as you'd expect)

/.*\bgrotto\b.*\braven\b.*/

- This clearly only catches them if grotto appears before raven
  → Need to 'OR' the opposite order. (Pipe)

(bonus)

» Can we generalize. what if we're checking for existence of more words (too many permutations to OR)

^.*(?=\bone\b).*(?=\btwo\b).*(?=\bthree\b).*$

- existence of 'one', 'two', 'three' checked regardless to order

- Thanks Saad <3

3) Write a Single Regex that matches all of

Colours
Colors
} optional u ①

they're colours
they're colors
} optional they're ②

they are colours
they are colours
} instead of they're, they are is also accepted ③

① /colo(u)?rs/

② /(they're)?colo(u)?rs/

③ /(they('| a)re)?colo(u)?rs/

* in the tutorial we used \s for the two spaces.

Thanks <3