Regex Guide

Complete Revolution In programming
For Text Detection…

What is
Regular Expression

- In computing, a **regular expression** is a specific pattern that provides concise and flexible means to "match" (specify and recognize) strings of text, such as particular characters, words, or patterns of characters. Common abbreviations for "regular expression" include **regex** and **regexp**.

Where Regex Can be used !!

- Regular expressions are used by many text editors, utilities, and programming languages to search and manipulate text based on patterns. Some of these languages, including Perl, Ruby, integrate regular expressions into the syntax of the core language itself. Other programming languages like .NET languages, Java, and Python instead provide regular expressions through standard libraries. For yet other languages, such as Object Pascal (Delphi) and C and C++, non-core libraries are available (however, version C++11 provides regular expressions in its Standard Libraries).

Let's Have Some !dea !!

- If You Search For Car ,
  The sequence of characters "car" appearing consecutively, such as in "car", "cartoon", or "bicarbonate"

Exact Matches For Regex

Regex → CAR

I AM DRIVING THE CAR

Matches …

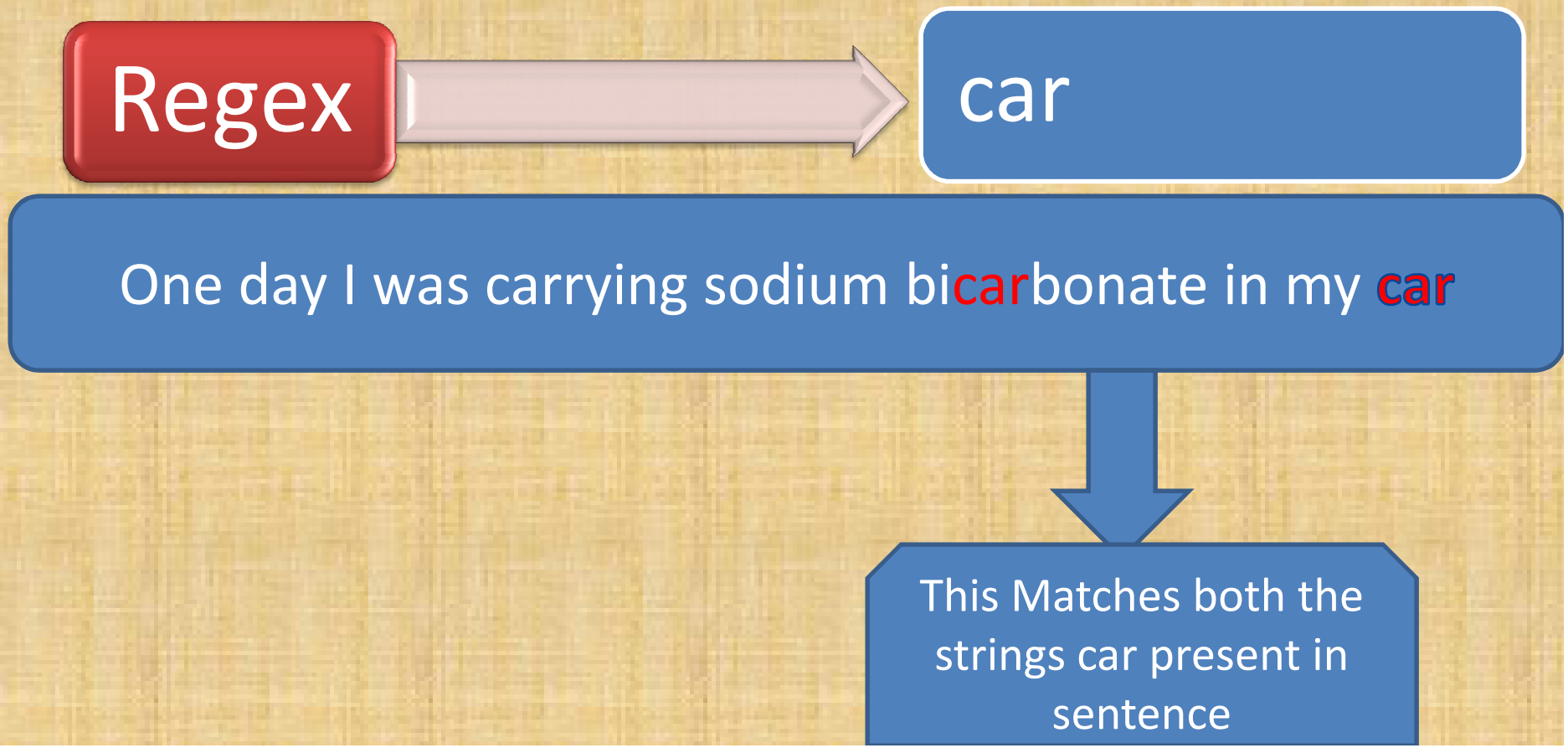
I AM WATCHING THE CARTOON

Matches …

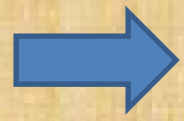
I AM DRIVING BIKE

Multiple Matches For Regex

- Regex Matches Strings at more than one place.



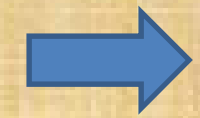
Dot(.) Character

- The Dot(.) Character Is used to match any string..
  Suppose, You have Regex  C .
- This Will Match..

  Bicarbonate was carried in car…

[ ] – Matches A set of Character

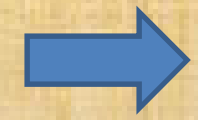
- [ ] Provides the feature to match set of character included in brackets..

- Suppose You have Regex  C[ael]r
- This Matches,

I take care of my car.

clrscr is used to clear screen.

^ Neglect Character

- ^ is used to neglect character .
- Suppose You have Regex  C[^ael]r
- This means the Regex will not match :

  car , cer and clr
- This will Match,

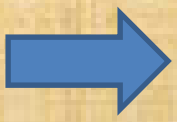
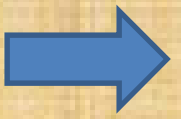
  core of the surface is solid..

  curriculum has been dispatched.

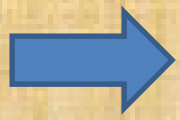
Other examples

- [aeiou] will match any character a, e, I, o or u..
  [cC]ode will match code / Code.
- Ranges :
  you can write [1-9] instead of [123456789].
  similarly [a-e] instead of [abcde].
  [abcde123456789] is equal to [a-e1-9].
  [-123] will match - , 1, 2, or 3. (here is no any class..
  you can use [a-zA-z0-9] for match all digits and characters.

Anchors in Regex.

- Anchors are used to match strings in beginning/end…
- ^ - is for beginning of line.
- $ - is for end of line.
- Regex..  ^c[ael]r
- care of my car
- Regex:  c[ael]r$
- care of my car

Repetitions in Regex

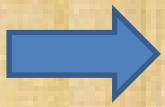
- The * is used to match repetitions..
- Suppose Regex :  Ya*ho
- This will match y(multiple times a)ho …

I.e. Yaaaaaaaaaaaaaaaahooooo..

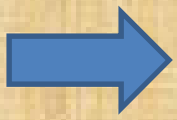
If there is 0 times a then,it will match.

I.e yhooooooooo

Repetitions Ranges(sub expressions)

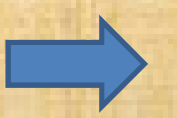
- Suppose I want to detect abcabc..
  Here abc is two times repeated.
- Hence,
- Regex :  (abc){2}
- This will detect abcabc…
- Here,
- {n} = exactly n times matches.
- {n,m}=least n times matches but no more than m.
- (abc){2,3} = abcabc ,, abcabcabc….
- .{0,} = .*
- a{2,} = aaa*

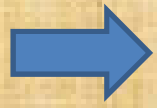
Search For Special Characters in Regex

- There are special characters in regex like,
  ^ , . , *, +, ?, | etc.
- To Detect the stings including this chars,
  We will Use backslash(\) followed by char.
- I.e. for search a*b* in any string ,
  Regex :  a *b *

This will match : a*b* is the output…

| (Pipe symbol in Regex)

- Pipe Symbol (|) is used for or operation..
- Regex :  c(a|u)t

  This will match.. cut and cat ..
- Regex :  A(ppll|x)e

  This will match… apple and axe ..

Quantifiers in Regex

- *, +, and ? Are quantifiers in Regex..
- * is for 0 or multiple matches..
- i.e. `abc*` will match ab , abc ,abcc..
- + is for 1 or multiple matches..
- i.e. `abc+` Will match abc, abcc, abccc
- ? Is for 0 or 1 matches..
- i.e. `abc?` Will match ab , abc .

Regex Metacharacters

\b Matches Word Boundary
\B Matches a nonword boundary.
\d Matches a digit character. Equivalent to [0-9].
\D Matches a nondigit character. Equivalent to [^0-9].
\f Matches a form-feed character.
\n Matches a newline character.
\r Matches a carriage return character.
\s Matches any white space including space, tab, form-feed, etc. Equivalent to [\f\n\r\t\v].
\S Matches any nonwhite space character. Equivalent to [^ \f\n\r\t\v].
\t Matches a tab character.
\v Matches a vertical tab character.
\w Matches any word character including underscore. Equivalent to [A-Za-z0-9_].
\W Matches any nonword character. Equivalent to [^A-Za-z0-9_].

End..

- Thank You..
- Credits to,

  My Mom , My Dad , Ashish Mistry…