Unit-IV
String Manipulation and Regular Expression
Lecture 5: Regular Expression

Regular Expression:

A regular expression commonly known as regex is a sequence of characters that forms a search pattern.

Regular expression is a compact way of describing a string pattern that matches a particular amount of text.

A regular expression can be a single character or a more complicated pattern. Regular expressions can be used to perform all types of text search and text replace operations.

Advantages and uses of Regular expressions;

- Regular expressions help in validation of text strings which are of programmer's interest.
- It offers a powerful tool for analyzing, searching a pattern and modifying the text data.
- It helps in searching specific string pattern and extracting matching results in a flexible manner.
- It helps in important user information validations like email address, phone numbers and IP address.
- Regexes are mostly used for browser detection, spam filtration, checking password strength and form validations.

How to create Regular Expressions:

Example:

Say we have created a customer feedback form, so we want to validate email address by encoding the standardized format of an email address in a regular expression.

We know Email addresses are of the form:

g0lu_4d3ven@gmail.com

i.e. the format includes:

- some alphanumeric or punctuation characters,
- followed by @ symbol,
- followed by a string of alphanumeric and hyphen characters, followed by more alphanumeric and hyphen characters

- and possibly more dots, up until the end of the string, which encodes as follows:

^[a-zA-Z0-9_\-\.]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9\-\.]+$

The sub expression:

^[a-zA-Z0-9_\-\.]+ means "start the string with at least one letter, number, underscore, hyphen or dot or some combination of those"

The @ symbol matches a literal @.

The sub expression **[a-zA-Z0-9\-]+** matches the first part of the host name including alphanumeric characters and hyphens. {you may observe that hyphen is slashed out **\-** because it's a special character inside square brackets.

The **\.** combination matches a literal .

The sub expression **[a-zA-Z0-9\-\.]+$** matches the rest of a domain name including letters, numbers, hyphens and more dots if required up until the end of the string.

Little bit more explanation:

Anchoring to the Beginning or End of a String:

In the above example you have noticed the **starting and end symbol i.e. ^ and $.**

You can specify if a particular sub-expression should appear at the start, the end or both.

The caret symbol (^) is used at the start of a regular expression to show that it must appear at the beginning of a searched string and

$ is used at the end of a regular expression to show that it must appear at the end.

e.g. ^golu → this matches golu at the start of a string.

com$ → matches com at the end of a string.

Matching Literal Special Characters:

If you want to match one of the special characters such as . , {, or $, **you must put a slash (\) in front of it**. If you want to represent a slash, you must replace it with two slashes \\.

List of Special Characters used in POSIX Regular Expressions outside square brackets:

Character	Meaning
\	Escape character
^	Match at start of string
$	Match at end of string
.	Match any character except newline (\n)
|	Start of alternative branch (read as OR)
(	Start subpattern
)	End subpattern
*	Repeat 0 or more times
+	Repeat 1 or more times
{	Start min/max quantifier
{	Start min/max quantifier

List of Special Characters used in POSIX regular expressions inside square brackets:

Character	Meaning
\	Escape character
^	NOT, only if used in initial position
-	Used to specify character ranges

Counted Subexpressions:

We can specify how many times something can be repeated by using a numerical expression in curly braces ({ }).

You can show:

- an exact number of repetitions ({3} means exactly 3 repetitions),
- a range of repetitions ({2,4} means from 2 to 4 repetitions), or
- an open ended range of repetitions ({ 2, } means at least two repetitions).

e.g. (very) {1,3}

matches 'very', 'very very' and 'very very very'

Character Sets and Classes:

Using character sets immediately gives regular expressions more power than exact matching expressions.

You can use the . character as a wildcard for any other single character except a new line (\n). e.g. the regular expression

.at

matches the strings

'cat', 'sat', and 'mat', among others.

If you want to specify a range of characters, then [a-z] i.e. between a and z.

Anything enclosed in the special square brace characters [and] is a character class i.e. a set of characters to which a matched character must belong.

e.g. [aeiou] means any vowel.

you can also describe range using hyphen lik [a-zA-Z] i.e. for any alphabetic character in upper or lowercase.

You can use sets to specify that a character cannot be a member of a set e.g.

[^a-z] matches any character that is not between a and z. The caret symbol means not when it is placed inside the square bracket.

Repetition:

Often you want to specify that there might be multiple occurrences of a particular string or class of character.

You can represent this using two special characters in your regular expression:

***** symbol means that the pattern can be repeated zero or more times and

+ symbol means that the pattern can be repeated one or more times.

The symbol should appear directly after the part of the expression that it applied to.