A regular expression (sometimes called a rational expression) is a sequence of characters that define a search pattern, mainly for use in pattern matching with strings, or string matching.

**Repeaters: \* , + and { } :**

These symbols act as repeaters and tell the computer that the preceding character is to be used for more than just one time.

**The asterisk symbol ( \* ):**

It tells the computer to match the preceding character (or set of characters) for 0 or more times (upto infinite).

**Example:** The regular expression ab\*c will give ac, abc, abbc,

abbbc….ans so on

**The Plus symbol ( + ):**

It tells the computer to repeat the preceding character (or set of characters) for at least one or more times (up to infinite).

Example: The regular expression ab+c will give abc, abbc,

abbc, … and so on.

**The curly braces {…}:**

It tells the computer to repeat the preceding character (or set of characters) for as many times as the value inside this bracket.

Example: {2} means that the preceding character is to be repeated 2

times, {min,} means the preceding character is matches min or more

times. {min,max} means that the preceding character is repeated at

least min & at most max times.

**Wildcard – ( . )**

The dot symbol can take place of any other symbol, that is why it

is called the wildcard character.

Example:

The Regular expression .\* will tell the computer that any character

can be used any number of times.

**Optional character – ( ? )**

This symbol tells the computer that the preceding character may

or may not be present in the string to be matched.

Example:

We may write the format for document file as – “docx?”

The ‘?’ tells the computer that x may or may not be

present in the name of file format.

**The caret ( ^ ) symbol**: Setting position for match :tells the computer that the match must start at the beginning of the string or line.

Example: ^\d{3} will match with patterns like "901" in "901-333-".

**The dollar ( $ ) symbol**

It tells the computer that the match must occur at the end of the string or before \n at the end of the line or string.

Example: -\d{3}$ will match with patterns like "-333" in "-901-333".

**Character Classes**

A character class matches any one of a set of characters. It is used to match the most basic element of a language like a letter, a digit, space, a symbol etc.

/s: matches any whitespace characters such as space and tab

/S: matches any non-whitespace characters

/d: matches any digit character

/D: matches any non-digit characters

/w: matches any word character (basically alpha-numeric)

/W: matches any non-word character

/b: matches any word boundary (this would include spaces, dashes, commas, semi-colons, etc)

[set\_of\_characters] – Matches any single character in set\_of\_characters. By default, the match is case-sensitive.

Example: [abc] will match characters a,b and c in any string.

[^set\_of\_characters] – Negation: Matches any single character that is not in set\_of\_characters. By default, the match is case sensitive.

Example: [^abc] will match any character except a,b,c .

[first-last] – Character range: Matches any single character in the range from first to last.

Example: [a-zA-z] will match any character from a to z or A to Z.

**The Escape Symbol:** \

If you want to match for the actual ‘+’, ‘.’ etc characters, add a backslash( \ ) before that character. This will tell the computer to treat the following character as a search character and consider it for matching pattern.

Example: \d+[\+-x\\*]\d+ will match patterns like "2+2"

and "3\*9" in "(2+2) \* 3\*9".

**Grouping Characters ( )**

A set of different symbols of a regular expression can be grouped together to act as a single unit and behave as a block, for this, you need to wrap the regular expression in the parenthesis( ).

Example: ([A-Z]\w+) contains two different elements of the regular

expression combined together. This expression will match any pattern

containing uppercase letter followed by any character.

**Vertical Bar ( | ) :**

Matches any one element separated by the vertical bar (|) character.

Example : th(e|is|at) will match words - the, this and that.

**\number :**

Backreference: allows a previously matched sub-expression(expression captured or enclosed within circular brackets ) to be identified subsequently in the same regular expression. \n means that group enclosed within the n-th bracket will be repeated at current position.

Example : ([a-z])\1 will match “ee” in Geek because the character

at second position is same as character at position 1 of the match.

**Comment : (?# comment) –**

Inline comment: The comment ends at the first closing parenthesis.

Example : \bA(?#This is an inline comment)\w+\b

# [to end of line] : X-mode comment. The comment starts at an unescaped # and continues to the end of the line.

Example : (?x)\bA\w+\b#Matches words starting with A