RegExp

A regular expression is an object that describes a sequence of characters.

Regular expressions are used to perform pattern-matching and "search-and-replace" functions on text.

It is not a separate language but used in every programming language like C++, Java and Python.

A regular expression consists of a *pattern string* and, potentially, a *flag* specifying further detail on how the pattern should be matched.

Regular Expression Patterns

We generally construct RegEx patterns using the basic characters we wish to match (e.g., abc), or a combination of basic and special characters (e.g., ab*c or (\d+)\.\d*).

Flags

If specified, flags can have any combination of the following values:

- g: global match.
- i: ignore case.
- m: multiline. Treats beginning (^) and end ($) characters as working over multiple lines.
- u: unicode. Treat pattern as a sequence of unicode code points.
- y: sticky. Matches only from the index indicated by the lastIndex property of this regular expression in the target string.

Special Characters in Regular Expressions

- Character Classes
- Character Sets
- Alteration
- Boundaries
- Grouping and back references
- Quantifiers

- Assertions

Character Classes

- This is not a class in the traditional sense, but rather a term that refers to a set of one or more characters that can be used to match a single character from some input string. Here are the basic forms:
- Enclosed within square brackets. Specify the what you'd like your expression to match within square brackets; for example, [a-f] will match any lowercase a through f character.
- Predefined: These consist of a backslash character (\) followed by a letter. The table below shows some predefined character classes and the characters they match.

Character	Matches
.	The period matches any single character, except line terminators (e.g., a newline).
\d	A single digit character (i.e., [0-9]).
\D	A single non-digit character (i.e., [^0-9]).
\w	A single alphanumeric word character, including the underscore (i.e., [A-Za-z0-9_]).
\W	A single non-word character (i.e., [^A-Za-z0-9_]).
\s	A single whitespace character. This includes space (), tab (\t), form feed, line feed, and other Unicode spaces.
\S	A single non-whitespace character (i.e., [^\w]).

Character Sets

- *The character set [abcd] will match any one character from the set {a, b, c, d}. This is equivalent to [a-d].*
- *The character set [^abcd]: Matches anything other than the enclosed characters. This is equivalent to [^a-d].*

Alteration

- *We use the | symbol to match one thing or the other. For example, a|b matches either a or b.*

Boundaries

Character	Matches
^	Matches beginning of input. If the multiline flag is set to true, also matches immediately after a line break character.
$	Matches end of input. If the multiline flag is set to true, also matches immediately before a line break character.
\b	A zero-width word boundary, such as between a letter and a space.
\B	Matches a zero-width non-word boundary, such as between two letters or between two spaces.

Grouping and back references

- *(a)*: Matches a and remembers the match. These are called capturing groups.
- *(?:a)*: Matches a but does not remember the match. These are called non-capturing groups.
- *\n*: Here n is a positive integer. A back reference to the last substring matching the n parenthetical in the regular expression.

Quantifiers

- a**: Matches the preceding item a, 0 or more times.*
- a+*: Matches the preceding item a, 1 or more times.*
- a?*: Matches the preceding item a, 0 or 1 time.*
- a{n}*: Here n is a positive integer. Matches exactly n occurrences of the preceding item a.*
- a{n, }*: Here n is a positive integer. Matches at least n occurrences of the preceding item a.*
- a{n, m}*: Here n and m are positive integers. Matches at least n and at most m occurrences of the preceding item a.*

Assertions

- *a(?=b)*: Matches a only if a is followed by b.
- *a(?!b)*: Matches a only if a is not followed *by b.*