Regular expressions can be used for pattern matching.
We use re module for regular expressions.

pattern: the regular expression pattern that you want to match.
string: the string which you want to search for the pattern.
flags (optional argument): a more advanced modifier that allows you to customize the behaviors of the function

Special Regex Characters:

These characters have special meaning in regex (to be discussed below):
., +, *, ?, ^, $, (, ), [, ], {, }, |, \.

Metacharacters:

| |
|---|
| . (dot): any one character except newline. Same as [^\n] |
| \d, \D: any one digit/non-digit character. Digits are [0-9] |
| \w, \W: any one word/non-word character. For ASCII, word characters are [a-zA-Z0-9_] |
| \s, \S: any one space/non-space character. For ASCII, whitespace characters are [ \n\r\t\f] |

## Occurrence Indicators (or Repetition Operators):

| |
|---|
| +: one or more (1+), e.g., [0-9]+ matches one or more digits such as '123', '000'. |
| *: zero or more (0+), e.g., [0-9]* matches zero or more digits.<br>It accepts all those in [0-9]+ plus the empty string. |
| ?: zero or one (optional), e.g., [+-]? matches an optional "+", "-", or an empty string. |
| {m,n}: m to n (both inclusive) |
| {m}: exactly m times |
| {m,}: m or more (m+) |

## Character class (or Bracket List)

| |
|---|
| [...]: Accept any one of the character within the square bracket,<br>e.g., [aeiou] matches "a", "e", "i", "o" or "u". |
| [.-.] (Range Expression): Accept any one of the character in the range,<br>e.g., [0-9] matches any digit; [A-Za-z] matches any uppercase or lowercase letters. |
| [^...]: NOT ONE of the character, e.g., [^0-9] matches any non-digit. |
| Only these four characters require escape sequence inside the bracket list: ^, -, ], \ |

## Escape Sequences (\char)

To match a character having special meaning in regex, you need to use a escape sequence prefix with a backslash (\).
E.g., \. matches "."; regex \+ matches "+"; and regex \( matches "(".

You also need to use regex \\ to match "\" (back-slash).

Regex recognizes common escape sequences such as \n for newline, \t for tab, \r for carriage-return, \nnn for a up to 3-digit octal number, \xhh for a two-digit hex code, \uhhhh for a 4-digit Unicode, \uhhhhhhhh for a 8-digit Unicode.

## OR Operator (|):

E.g., the regex four|4 accepts strings "four" or "4".

## Position Anchors

^, $: start-of-line and end-of-line respectively. E.g., ^[0-9]$ matches a numeric string.

\b: boundary of word, i.e., start-of-word or end-of-word. E.g., \bcat\b matches the word "cat" in the input string.

\B: Inverse of \b, i.e., non-start-of-word or non-end-of-word.

\<, \>: start-of-word and end-of-word respectively, similar to \b. E.g., \<cat\> matches the word "cat" in the input string.

\A, \Z: start-of-input and end-of-input respectively.

Parenthesized Back References:

| |
|---|
| Use parentheses ( ) to create a back reference. |
| Use $1, $2, ... (Java, Perl, JavaScript) or \1, \2, ... (Python) to retreive the back references in sequential order. |

Laziness (Curb Greediness for Repetition Operators)

| |
|---|
| *?, +?, ??, {m,n}?, {m,}? |

https://www3.ntu.edu.sg/home/ehchua/programming/howto/Regexe.html