Regular Expressions

Besant Technologies

Regular Expression (Regex):

- A sequence of characters that defines a search pattern.
- Used for searching strings.
- Ex: ^a...s$ - any five letter strings starting with ‘a’ and ending with ‘s’
-

Expression	String	Matched?
^a...s$	abs	No match
	alias	Match
	abyss	Match
	Alias	No match
	An abacus	No match

Metacharacters:

- Characters that are interpreted in a special way.
- List: [], ., ^, $, *, +, ?, {}, (), \, |
-

[]:

- Specifies a set of characters to match.
- Ex: [abc] - looks for any of these characters to match.
- Can specify a range of characters using -
- Ex: [a-z], [0-9] etc.,
- Can complement the character set using ^
- Ex: [^a-t], [^1-5] etc.,

Expression	String	Matched?
[abc]	a	1 match
	ac	2 matches
	Hey Jude	No match
	abc de ca	5 matches

- Matches any single character except new line.
-

Expression	String	Matched?
..	a	No match
	ac	1 match
	acd	1 match
	acde	2 matches (contains 4 characters)

^ - Caret

- Used to check if a string **starts** with certain character.
-

Expression	String	Matched?
`^a`	`a`	1 match
	`abc`	1 match
	`bac`	No match
`^ab`	`abc`	1 match
	`acb`	No match (starts with `a` but not followed by `b`)

$ - dollar

- Used to check if any string ends with certain character/pattern.

Expression	String	Matched?
a$	a	1 match
	formula	1 match
	cab	No match

*

- Matches zero or more occurrences of the pattern left to it.

Expression	String	Matched?
ma*n	mn	1 match
	man	1 match
	maaan	1 match
	main	No match (a is not followed by n)
	woman	1 match

+

- Matches one or more occurences of the pattern left to it.
-

Expression	String	Matched?
ma+n	mn	No match (no a character)
	man	1 match
	maaan	1 match
	main	No match (a is not followed by n)
	woman	1 match

?

- Matches zero or one occurences of the pattern left to it.

Expression	String	Matched?
`ma?n`	`mn`	1 match
	`man`	1 match
	`maaan`	No match (more than one `a` character)
	`main`	No match (a is not followed by n)
	`woman`	1 match

{}

- {n,m} - means atleast n and atmost m repetitions of the pattern left to it.
-

Expression	String	Matched?
a{2,3}	abc dat	No match
	abc daat	1 match (at daat)
	aabc daaat	2 matches (at aabc and daaat)
	aabc daaaat	2 matches (at aabc and daaaat)

Expression	String	Matched?
[0-9]{2,4}	ab123csde	1 match (match at ab123csde)
	12 and 345673	2 matches (at 12 and 345673)
	1 and 2	No match

| - or operator

Vertical bar `|` is used for alternation (`or` operator).

Expression	String	Matched?
`a|b`	`cde`	No match
	`ade`	1 match (match at `ade`)
	`acdbea`	3 matches (at `acdbea`)

Here, `a|b` match any string that contains either `a` or `b`

()

- Used to group sub patterns.

Parentheses `()` is used to group sub-patterns. For example, `(a|b|c)xz` match any string that matches either `a` or `b` or `c` followed by `xz`

Expression	String	Matched?
`(a|b|c)xz`	`ab xz`	No match
	`abxz`	1 match (match at `abxz`)
	`axz cabxz`	2 matches (at `axzbc cabxz`)

\ - excape character

Backlash `\` is used to escape various characters including all metacharacters. For example,

`\$a` match if a string contains `$` followed by `a`. Here, `$` is not interpreted by a RegEx engine in a special way.

If you are unsure if a character has special meaning or not, you can put `\` in front of it. This makes sure the character is not treated in a special way.

Special Sequences:

- Most commonly used patterns.
- ‘\A’, ‘\b’, ‘\B’, ‘\d’, ‘\D’, ‘\s’, ‘\S’, ‘\w’, ‘\W’, ‘\Z’

\A:

`\A` - Matches if the specified characters are at the start of a string.

Expression	String	Matched?
`\Athe`	`the sun`	Match
	`In the sun`	No match

\Z:

`\Z` - Matches if the specified characters are at the end of a string.

Expression	String	Matched?
`\ZPython`	`I like Python`	1 match
	`I like Python`	No match
	`Python is fun.`	No match

\b:

`\b` - Matches if the specified characters are at the beginning or end of a word.

Expression	String	Matched?
`\bfoo`	`football`	Match
	`a football`	Match
	`afootball`	No match
`foo\b`	`the foo`	Match
	`the afoo test`	Match
	`the afootest`	No match

\B:

`\B` - Opposite of `\b`. Matches if the specified characters are **not** at the beginning or end of a word.

Expression	String	Matched?
`\Bfoo`	`football`	No match
	`a football`	No match
	`afootball`	Match
`foo\B`	`the foo`	No match
	`the afoo test`	No match
	`the afootest`	Match

\d:

`\d` - Matches any decimal digit. Equivalent to `[0-9]`

Expression	String	Matched?
`\d`	`12abc3`	3 matches (at `12abc3`)
	`Python`	No match

\D:

`\D` - Matches any non-decimal digit. Equivalent to `[^0-9]`

Expression	String	Matched?
`\D`	`1ab34"50`	3 matches (at `1ab34"50`)
	`1345`	No match

\s:

`\s` - Matches where a string contains any whitespace character. Equivalent to `[ \t\n\r\f\v]`.

Expression	String	Matched?
`\s`	`Python RegEx`	1 match
	`PythonRegEx`	No match

\S:

`\S` - Matches where a string contains any non-whitespace character. Equivalent to `[^ \t\n\r\f\v]`.

Expression	String	Matched?
`\S`	`a b`	2 matches (at `a b`)
		No match

\w:

`\w` - Matches any alphanumeric character (digits and alphabets). Equivalent to `[a-zA-Z0-9_]`. By the way, underscore `_` is also considered an alphanumeric character.

Expression	String	Matched?
`\w`	`12&": ;c`	3 matches (at `12&": ;c`)
	`%"> !`	No match

\W:

`\W` - Matches any non-alphanumeric character. Equivalent to `[^a-zA-Z0-9_]`

Expression	String	Matched?
`\W`	`1a2%c`	1 match (at `1a2%c`)
	`Python`	No match

re module:

- ‘Re’ module to work with regular expressions.
- https://regex101.com/ - practice.
- Methods:
  - findall()
  - split()
  - sub()
  - subn()
  - match()
  - search()