Regular Expressions

Common functions

Pattern Matching

`a`, `X`, `9` - Ordinary characters that match themselves

`.` - Matches any single character except newline character

`\w` - Matches any single letter, digit or underscore

`\W` - Matches any character not part of `\w`

`\s` - Matches a single whitespace character like: space, newline, tab, return

`\S` - Matches any character not part of `\s`

`\t`, `\n`, `\r` - Matches tab, newline, return respectively

Pattern Matching Continued

`\d` - Matches decimal digit 0-9

`^` - Start of the string (and start of the line in-case of multiline string)

`$` - End of the string (and newline character in-case of multiline string)

`\` - Inhibit the specialness of a character

`[abc]` - Matches `a` or `b` or `c`

`[a-zA-Z0-9]` - Matches any letter from (`a` to `z`) or (`A` to `Z`) or (`0` to `9`)

`\A` - Matches only at the start of the string even in MULTILINE mode

`\Z` - Matches only at the end of the string even in MULTILINE mode

`\b` - Matches only the beginning or end of the word

Dealing with repetition

`+` - one or more characters

`*` - zero or more characters

`?` - zero or one character

`{n}` - repeat exactly n times

`{n,}` - repeat atleast n times or more

`{m, n}` - repease atleast m times but no more than n times

Chain commands

.search() and .findall() and .compile()

`findall()` matches all occurrences of a pattern in a string, but `search()` finds only the first occurrence of the pattern within the string while traversing from left-to-right.

`compile()` compiles the regular expression into a regular expression object that can be used later with `.search()`, `.findall()` or `.match()`. If you are using the same regex repeatedly, then it is much efficient to compile the regex first and then apply it on strings.

Example:

```
>>> pat = re.compile('^foo')
>>> pat.findall('foobar')
['foo']
```