Regex Cheat Sheet

Regex Syntax

Characters¶

Character	Matches
`a`	`a` character
`.`	Any character (except newline)
`\.`	`.` character
`\\`	`\` character
`\*`	`*` character

Character Classes¶

	Matches	Description
`[abcd]`	Any one of the letters `a` through `d`	Set of characters
`[^abcd]`	Any character but `a`, `b`, `c`, or `d`	Complement of a set of characters
`[a-d]`	Any one of the letters `a` through `d`	Range of characters
`[a-dz]`	Any of `a`, `b`, `c`, `d`, or `z`	Range of characters

Special Sequences¶

Type	Expression	Equivalent To	Description
Word Character	`\w`	`[a-zA-Z0-9_]`	Alphanumeric or underscore
Non-word Character	`\W`	`[^a-zA-Z0-9_]`	Anything but a word character
Digit Character	`\d`	`[0-9]`	Numeric
Non-digit Character	`\D`	`[^0-9]`	Non-numeric
Whitespace Character	`\s`	`[ \t\n\r\f\v]`	Whitespace
Non-whitespace Character	`\S`	`[^ \t\n\r\f\v]`	Anything but a whitespace characte

Anchors¶

Anchor	Matches
`^`	Start of the string
`$`	End of the string
`\b`	Boundary between word and non-word characters

Groups¶

Group Type	Expression
Capturing	`(` ... `)`
Non-capturing	`(?:` ... `)`

Quantifiers/Repetition¶

Quantifier	Modification
`{5}`	Match expression exactly 5 times
`{2,5}`	Match expression 2 to 5 times
`{2,}`	Match expression 2 or more times
`{,5}`	Match expression 0 to 5 times
`*`	Match expression 0 or more times
`{,}`	Match expression 0 or more times
`?`	Match expression 0 or 1 times
`{0,1}`	Match expression 0 or 1 times
`+`	Match expression 1 or more times
`{1,}`	Match expression 1 or more times

Non-greedy quantifiers¶

Quantifier	Modification
`{2,5}?`	Match 2 to 5 times (less preferred)
`{2,}?`	Match 2 or more times (less preferred)
`{,5}?`	Match 0 to 5 times (less preferred)
`*?`	Match 0 or more times (less preferred)
`{,}?`	Match 0 or more times (less preferred)

Quantifier	Modification
`??`	Match 0 or 1 times (less preferred)
`{0,1}?`	Match 0 or 1 times (less preferred)
`+?`	Match 1 or more times (less preferred)
`{1,}?`	Match 1 or more times (less preferred)

Alternators¶

Quantifier	Modification
`ABC|DEF`	Match string `ABC` or string `DEF`

Lookaround¶

Quantifier	Modification
`(?=abc)`	Zero-width match confirming `abc` will match upcoming chars
`(?!abc)`	Zero-width match confirming `abc` will **not** match upcoming chars

Python

functions¶

Function	Purpose	Usage
`re.search`	Return a match object if pattern found in string	`re.search(r'[pat`
`re.finditer`	Return an iterable of match objects (one for each match)	`re.finditer(r'[p`
`re.findall`	Return a list of all matched strings (different when capture groups)	`re.findall(r'[pa`
`re.split`	Split string by regex delimeter & return string list	`re.split(r'[ -]'`
`re.compile`	Compile a regular expression pattern for later use	`re.compile(r'[pa`

flags¶

Flag	Description
`re.IGNORECASE`	Match uppercase and lowercase characters interchangeably
`re.VERBOSE`	Ignore whitespace characters and allow `#` comments

Regex Cheatsheet

Please import **re** library to use regular expressions in Python:

```
import re
```

Basic RE Module Functions

1. **re.findall(x, y)** → Matches all instances of an expression **x** in a string **y** and returns them in a list.

2. **re.sub(x, y, z)** → Replace an expression **x** with another expression **y** in a string **z**.

Containers

1. **[a-k]** → Matches any alphabet from a to k.

Eg: `re.findall("[a-d0-3]", "cbr250r")`
`Output: ['c', 'b', '2', '0']`

2. **[dym]** → Matches either d, y, or m and not dym.

Eg: `re.findall("[c5]", "cbr250r")`
`Output: ['c', '5']`

3. **[^ab]** → Matches any character excluding a or b.

Eg: `re.findall("[^r]", "cbr250r")`
`Output: ['c', 'b', '2', '5', '0']`

Regular Expression Character Classes

1. **'\w'** → Matches an alphanumeric character, i.e., **a-z**, **A-Z**, and **0-9**. and underscore, '_'.

Eg: `re.findall("\w", "red_blue #8")`
`Output: ['r', 'e', 'd', '_', 'b', 'l', 'u', 'e', '8']`

2. '**\d**' → Matches a digit, **0-9**.

Eg: `re.findall("\d", "red_blue #8")`
`Output: ['8']`

3. **'\D' →** Matches a non-digit.

Eg: `re.findall("\D", "red_blue #8")`
`Output: ['r', 'e', 'd', '_', 'b', 'l', 'u', 'e', ' ', '#']`

4. '**\s**' → Matches one whitespace character.

Eg: `re.findall("\s", "red_blue \t #8")`
`Output: [' ', '\t', ' ']`

5. '**\S**' → Matches one non-whitespace character.

Eg: `re.findall("\S", "red_blue \t #8")`
`Output: ['r', 'e', 'd', '_', 'b', 'l', 'u', 'e', '#', '8']`

Special Characters

1. **' . '** → Matches any character except newline (\n).

Eg: `re.findall(".", "Hi guys\n")`
`Output: ['H', 'i', ' ', 'g', 'u', 'y', 's']`

2. **' ^ '** → Matches beginning of a string.

Eg: `re.findall("^[a-zA-Z]\w", "Hi guys!")`
`Output: ['Hi']`

`re.findall("^[a-zA-Z]\w", "5 star")`
`Output: [ ]`

3. **‘ $ ’** → Matches end of a string.

Eg:
```
re.findall("[a-zA-Z]$", "Hi guys")
Output: ['s']

re.findall("\w+[a-zA-Z]$", "Hi guys")
Output: ['guys']
```

4. **‘ \ ’** → Escapes special characters.

Eg:
```
re.findall("*", "5 star *")
Output: Error

re.findall("\*", "5 star *")
Output: ['*']
```

5. **‘ * ’** → Greedily matches the expression to its left 0 or more times (append ‘?’ for non-greedy).

Eg:
```
re.findall("H*", "Hi guys")
Output: ['H', '', '', '', '', '', '', '']
```

6. **‘ + ’** → Greedily matches the expression to its left 1 or more times (append ‘?’ for non-greedy).

Eg:
```
re.findall("H+", "Hi guys")
Output: ['H']
```

7. **‘ *? ’** or **‘ +? ’** → Non-greedy matching

Eg:
```
re.findall("a.*a", "Have a good day?")
Output: ['ave a good da']

re.findall("a.*?a", "Have a good day?")
Output: ['ave a']
```

8. **{m}** → Matches the expression to its left m times.

Eg:
```
re.findall("H.{2}", "Have a good day?")
Output: ['Hav']

re.findall("H.{5}", "Have a good day?")
Output: ['Have a']
```