QuickRef.ME

RegEX cheatsheet

# Regex examples

Characters

`ring`	Match ring springboard etc.
`.`	Match a, 9, + etc.
`h.o`	Match hoo, h2o, h/o etc.
`ring\?`	Match ring?
`\(quiet\)`	Match (quiet)
`c:\\windows`	Match c:\windows

Use \ to search for these special characters:
`[ \ ^ $ . | ? * + ( ) { }`

Alternatives

`cat|dog`	Match cat or dog
`id|identity`	Match id or identity
`identity|id`	Match id or identity

Order longer to shorter when alternatives overlap

Character classes

`[aeiou]`	Match any vowel
`[^aeiou]`	Match a NON vowel
`r[iau]ng`	Match ring, wrangle, sprung, etc.
`gr[ae]y`	Match gray or grey
`[a-zA-Z0-9]`	Match any letter or digit
`[\u3a00-\ufa99]`	Match any Unicode Hàn (中文)

In `[ ]` always escape `. \ ]` and sometimes `^ - .`

Shorthand classes

`\w`	"Word" character (letter, digit, or underscore)
`\d`	Digit
`\s`	Whitespace (space, tab, vtab, newline)
`\W, \D, or \S`	Not word, digit, or whitespace
`[\D\S]`	Means not digit or whitespace, both match
`[^\d\s]`	Disallow digit and whitespace

Occurrences

`colou?r`	Match color or colour
`[BW]ill[ieamy's]*`	Match Bill, Willy, William's etc.
`[a-zA-Z]+`	Match 1 or more letters
`\d{3}-\d{2}-\d{4}`	Match a SSN
`[a-z]\w{1,7}`	Match a UW NetID

Greedy versus lazy

`* + {n,}` greedy	Match as much as possible
`<.+>`	Finds 1 big match in <b>bold</b>
`*? +? {n,}?` lazy	Match as little as possible
`<.+?>`	Finds 2 matches in <b>bold</b>

Scope

`\b`	"Word" edge (next to non "word" character)
`\bring`	Word starts with "ring", ex ringtone
`ring\b`	Word ends with "ring", ex spring
`\b9\b`	Match single digit 9, not 19, 91, 99, etc..
`\b[a-zA-Z]{6}\b`	Match 6-letter words
`\B`	Not word edge
`\Bring\B`	Match springs and wringer
`^\d*$`	Entire string must be digits

Modifiers

`(?i)[a-z]*(?-i)`	Ignore case ON / OFF
`(?s).*(?-s)`	Match multiple lines (causes . to match newline)
`(?m)^.*;$(?-m)`	^ & $ match lines not whole string
`(?x)`	#free-spacing mode, this EOL comment ignored
`(?-x)`	free-spacing mode OFF
`/regex/ismx`	Modify mode for entire string

`^[a-zA-Z]{4,20}$`	String must have 4-20 letters
`^[A-Z]`	String must begin with capital letter
`[\.!?"')]$`	String must end with terminal puncutation

Groups

`(in|out)put`	Match input or output
`\d{5}(-\d{4})?`	US zip code ("+ 4" optional)

Parser tries EACH alternative if match fails after group.
Can lead to catastrophic backtracking.

Back references

`(to) (be) or not \1 \2`	Match to be or not to be
`([^\s])\1{2}`	Match non-space, then same twice more aaa, ...
`\b(\w+)\s+\1\b`	Match doubled words

Non-capturing group

`on(?:click|load)`	Faster than: `on(click|load)`

Use non-capturing or atomic groups when possible

Atomic groups

`(?>red|green|blue)`	Faster than non-capturing
`(?>id|identity)\b`	Match id, but not identity

"id" matches, but `\b` fails after atomic group, parser doesn't backtrack into group to retry 'identity'

If alternatives overlap, order longer to shorter.

Lookaround

`(?= )`	Lookahead, if you can find ahead
`(?! )`	Lookahead,if you can not find ahead
`(?<= )`	Lookbehind, if you can find behind
`(?<! )`	Lookbehind, if you can NOT find behind
`\b\w+?(?=ing\b)`	Match warbling, string, fishing, ...
`\b(?!\w+ing\b)\w+\b`	Words NOT ending in "ing"
`(?<=\bpre).*?\b`	Match pretend, present, prefix, ...
`\b\w{3}(?<!pre)\w*?\b`	Words NOT starting with "pre"

If-then-else

Match "Mr." or "Ms." if word "her" is later in string

```
M(?(?=.*?\bher\b)s|r)\.
```

```
\b\w+(?<!ing)\b
```

Match words NOT ending in "ing"

# RegEx in Python

Getting started

Import the regular expressions module

```python
import re
```

Functions

`re.findall`	Returns a list containing all matches
`re.finditer`	Return an iterable of match objects (one for each match)
`re.search`	Returns a Match object if there is a match anywhere in the string
`re.split`	Returns a list where the string has been split at each match
`re.sub`	Replaces one or many matches with a string

Examples

re.search()

```python
>>> sentence = 'This is a sample string'
>>> bool(re.search(r'this', sentence, flags=re.I))
True
>>> bool(re.search(r'xyz', sentence))
False
```

re.findall()

```python
>>> re.findall(r'\bs?pare?\b', 'par spar apparent spare part pare')
['par', 'spar', 'spare', 'pare']
>>> re.findall(r'\b0*[1-9]\d{2,}\b', '0501 035 154 12 26 98234')
['0501', '154', '98234']
```

re.finditer()

```python
>>> m_iter = re.finditer(r'[0-9]+', '45 349 651 593 4 204')
>>> [m[0] for m in m_iter if int(m[0]) < 350]
['45', '349', '4', '204']
```

re.split()

re.compile	Compile a regular expression pattern for later use

Flags

re.I	re.IGNORECASE	Ignore case
re.M	re.MULTILINE	Multiline
re.L	re.LOCALE	Make \w,\b,\s locale dependent
re.S	re.DOTALL	Dot matches all (including newline)
re.U	re.UNICODE	Make \w,\b,\d,\s unicode dependent
re.X	re.VERBOSE	Readable style

```
>>> re.split(r'\d+', 'Sample123string42with777numbers')
['Sample', 'string', 'with', 'numbers']
```

re.sub()

```
>>> ip_lines = "catapults\nconcatenate\ncat"
>>> print(re.sub(r'^', r'* ', ip_lines, flags=re.M))
* catapults
* concatenate
* cat
```

re.compile()

```
>>> pet = re.compile(r'dog')
>>> type(pet)
<class '_sre.SRE_Pattern'>
>>> bool(pet.search('They bought a dog'))
True
>>> bool(pet.search('A cat crossed their path'))
False
```