import re

27.02.2019

Write only code

1. Slow performance for bad regexps
2. Case sensitive: “Lol” vs “lol”
3. Sensitiveness to problem changes
4. For simple queries, built-in function are much more efficient
5. Difficult to read

```
(?:[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*|\"(?:[\x01-\x08\x0b\x0c\
x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*\")@(?:(?:[a-z0-9](?:[a-z0-
9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:25[0-5]|
2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?|[a-z0-9-]*[a-z0-
9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\])
```

Simple rules

Rule	Description	Example	Result
`.`	Any symbol except new line \n	`e.e`	eye, elephant, exe, prelevement
`\d`	Any digit	`\ds`	In the beginning of 60s
`\D`	Any symbol, except digits	`\D`	In the beginning of 60s
`\s`	Any whitespace symbol	`M \s S`	M Soyer, ISM series, GYM Section B
`\S`	Any non-whitespace symbol	`\S123`	K1234, 1 + 123, #123
`\w`	Any letter, digit or underscore	`\w\w\w\w`	Year, K1_0, cubics
`\W`	Any symbol except letters, digits and underscore	`quite\W`	quite!, ‘quite, please’, quietest

Simple rules

Rule	Description	Example	Result
`[..]`	Any symbol in brackets, or within the range specified	`[0-9][0-9A-Fa-f]`	12, 1F, 4S, E4
`[..]`	Any symbol in brackets, or within the range specified	`gr[ae]y`	Gray, grey
`[^..]`	Any symbol except specified in brackets	`[^_^o]`	2^4, k_pop
	Minus must be either in the end or in the beginning of matching letters	`[abc-], [-1]`	
	Only \ and] symbols should be escaped	`[*[(+\\\]\t]`	
`\b`	Word boundary	`\bcorn\b`	Pop corn, unicorn, corner
`\b`	Word boundary	`\bcorn`	Pop corn, unicorn, corner
`\B`	Not word boundary	`corn\B`	Pop corn, unicorns, corner

Quantifiers

Rule	Description	Example	Result
`{n}`	Exact n times	`\d{4}`	12,123,1234,12345
`{m,n}`	From m to n times	`\d{3,4}`	12,123,1234,12345
`{m,}`	At least m times	`\d{3,}`	12,123,1234,12345
`{,n}`	At most n times	`\d{,4}`	12,123,1234,12345
`?`	0 or 1 entry = {0,1}	`eggs?`	eggs, eggy, egg
`*`	0 or more times = {0,}	`Ex\d*`	Ex, Ex1, Ex12, Ex666
`+`	1 or more times = {1,}	`Ba+m`	Baaaaam, Bam, Bm

`*?`
`+?`
`??`
`{m,n}?`
	Quantifiers are greedy - accept the most possible number of symbols. Question mark make them lazy and take the least possible number of letters.	`.*?-`	Quantifiers are greedy - accept the most possible number of symbols.

Regular Expressions in Python

Function	Description	Example
`re.search(pattern, string)`	Find first match of `pattern` in `string`	`re.search(r'\W+', 'Words, words, words.')` - `<_sre.SRE_Match object; span=(5, 7), match=', '>`
`re.fullmatch(pattern, string)`	Check if the whole `string` matches the regular expression `pattern`	`re.fullmatch(r'bar', 'barcode')` - `None`
`re.split(pattern, string, maxsplit=0)`	Similar to `str.split()`. Splits `string` using `pattern` as separator	`re.split(r'\W+', 'Words, words, words.')` - `['Words', 'words', 'words', '']`
`re.findall(pattern, string)`	Find all non-overlapping `patterns` in `string`	`re.findall(r'\w+', 'Words, words, words.')` - `['Words', 'words', 'words']`
`re.finditer(pattern, string)`	Return an iterator with all non-overlapping matches of `pattern` in `string`. Empty matches are included in the result.	`m=re.finditer(r'\w+', 'Words, words, words.')` `print([m1[0] for m1 in m])`
`re.sub(pattern, repl, string, count=0)`	Substitute all non-overlapping matches of `pattern` for `repl` in `string`	`re.sub(r'\w+','dog' 'Words, words, words.')` - `dogs, dogs, dogs.`

Escape

- For regex `.^$*+?{}[]\|()` should be escaped with `\`.
- For Python string variable `\` should be escaped with `\` as well.

`re.search('\\\\par','\\\\par')`:

1. `\\\\par` in Python is read as `\\par`, since first `\` escapes second `\`.
2. `\\par` received by regex is read as `\par`, since first `\` escapes second `\`.
3. Therefore `\\\\par` corresponds to `\par` in the end.

Instead, use `re.search(r'\\par','\\\\par')` where `r` tells Python not to treat `\` as escape symbol (except the case of opening quotes).

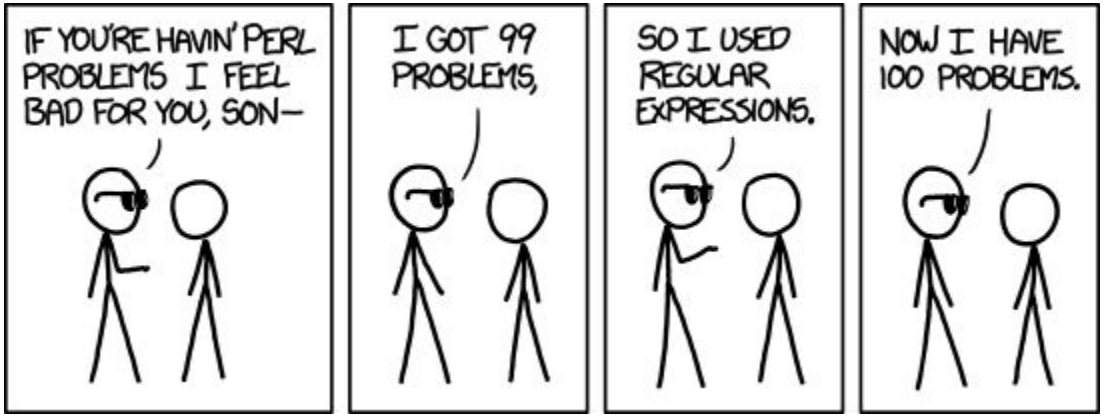
Grouping and Enumerating (or operation)

Rule	Description	Example	Result
`|`	Logical `Or`	`cat|dog|l[io]on`	I have pets - 1 cat, 2 dogs, small loon; and I want to get a lion.
`(..)`	Grouping the pattern	`((\d)(\d))((\d)(\d))`	123456789 -> 1234 -> 12 ->1,2 34 ->3,4
`(?:..){x}`	Grouping the pattern to be able to repeat it	`(?:[0-9A-Fa-f]{2}[:-]){5}[0-9A-Fa-f]{2}`	01:23:45:67:89:ab

Examples

Example	Result
`(?:\w\w\d\d)+`	I saw ISE14a, MUI86, but one told me about ISE13MUI76.
`(?:\w+\d+)+`	I saw ISE14a, MUI86, but one told me about ISE13MUI76.
`(?:\+33|07|06)(?:\d{2,3}){4}`	+33 761 61 24 22, 07 61 61 24 22, 06 61 61 24 22
`(?:[Hh][aoeiy]+)+`	Hahahahahaa, hahahihihihi, hyperactive
`\b(?:[Hh][aoeiy]+)+\b`	Hahahahahaa, hahahihihihi, hyperactive
`re.sub(r'(\d\d)/(\d\d)/(\d{4})', r'\2.\1.\3', text)`	text: "We arrive on 03/25/2018. So you are welcome after 04/01/2018." Out: "We arrive on 25.03.2018. So you are welcome after 01.04.2018."

LABS TIME


IF YOU'RE HAVIN' PERL PROBLEMS I FEEL BAD FOR YOU, SON—
I GOT 99 PROBLEMS,
SO I USED REGULAR EXPRESSIONS.
NOW I HAVE 100 PROBLEMS.