A Traveler’s Guide to Regex in the Wild

Megan Guiney

Introduction

The first regex I learned was Perl, in a workshop offered by the same organization where I learned most of my early skills. This was largely a result of the culture of the shop: it was incredibly old school, and mant of our core scripts were still perl. In any case, it was a bit of a shock, the first time I wrote a regex in python (just like i would have done in a bash script), and it just. didn't. work. I kept slipping into old habits, using perl regex when i should have been using the similiar- but-distinct python regex variant.

Eventually, in my case, I started working with a python regex reference pulled up in the background, so I decided to make a more unified reference pocketbook for my own use, as well as that of pretty much anyone who wants -it2.- This is also super handy to have around if you're just getting started with one of these regex variants, as a reference for building regexes, until you have the syntax more or less memorized.

Happy hacking, y'all!

Basic Symbols

Wat do?	How Perl do?	How Python do?
Custom character class	[...]	[...]
Negated custom character class	[^...]	[^...]
Ranges	[a-z] (with '-' escaped if it comes last)	[a-z] (with '-' escaped if it comes last)
Alternation ("or")	|	|

ing (or the bored) —> https://callumacrae.github.io/regex-tuesday/

- Most Crazy Regexes —> https://stackoverflow.com/questions/800813/what-is-the-most-difficult-challenging-regular-expression-you-have-ever-written
- Regex Humor: because regex humor is the universal language —> http://www.rexegg.com/regex-humor.html

Conclusion

((Coming soon))

Zero-width assertions

Wat do?	How Perl do?	How Python do?
Word boundary	\b	\b
Anywhere but word boundary	\B	\B
Beginning of line/string	^ / \A	^ / \A
End of line/string	$ / \Z	$ / \Z

Captures and Groups

Wat do?	How Perl do?	How Python do?
Capturing group	(...) or (?<name>...)	(...) or (?P<name>...)
Non-capturing group	(?:...)	(?:...)
Backreference to a specific group	\1, \g1	\1
Named backreference	\k<name>	(?P=name)

More learning resources

- Regex One: an interactive tutorial for teaching regex from the ground up —> https://regexone.com/
- Regex adventure: an educational workshop —> https://github.com/workshopper/regex-adventure
- Regex Crossword: a site offering a series of games allowing you to test your regex chops using old-school brainteasers —> https://regexcrossword.com/
- Redoku: regex sudoku/puzzle —> http://padolsey.github.io/redoku/
- Regex Tuesday - Challenges: regex challenges for the dar-

Which type of regex does $LINUX_UTIL use?

*nix util	Regex variant	Additional notes
awk	ERE	may depend on implementation
grep	BRE	grep -P switches to PCRE
egrep	ERE	N/A
less	ERE	usually ERE, the regex variant is supplied by the system
screen	plaintext	N/A
sed	BRE	Using the -E flag switches to ERE

Character Classes

Wat do?	How Perl do?	How Python do?
Any character (except newline)	.	.
Match a non-"word" character	\W	\W
Match a "word" character	\w or [[:word:]]	\w
Case	[[:upper:]] or [[:lower:]]	N/A
Whitespace (not including newlines)	N/A	N/A

Wat do?	How Perl do?	How Python do?
Whitespace (not including newlines)	N/A	N/A
Whitespace (including newline)	\s or [[:space:]]	\s
Match a non-whitespace character	\S	\S
Match a digit character	\d or [[:digit:]]	\d
Match a non-digit character	\D	\D
Any hex-adecimal digit	[[:xdigit:]]	N/A
Any octal digit	N/A	N/A

Other basic regex characters

Wat do?	How Perl do?	How Python do?
Independent non-backtracking pattern	(?>...)	N/A
Anywhere but word boundary	(?i) or (?-i)	(?i) or (?-i)

Examples

((Coming soon))

Multiplicity

Wat do?	How Perl do?	How Python do?
0 or 1	?	?
0 or 1, non-greedy	??	??
0 or 1, don't give back on backtrack	?+	N/A
0 or more	*	*
0 or more, non-greedy	*?	*?
0 or more, don't give back on backtrack	*+	N/A
1 or more	+	+
1 or more, non-greedy	*?	*?
1 or more, don't give back on backtrack	++	N/A
Specific number	{n} or {n,m} or{n,	{n} or {n,m} or{n,

Wat do?	How Perl do?	How Python do?
Any graphical character excluding "word" characters	[[:punct:]]	N/A
Any alphabetical character	[[:alpha:]]	N/A
Any alphanumerical character	[[:alnum:]]	N/A
ASCII character	[[:ascii:]]	N/A

Lookarounds

Wat do?	How Perl do?	How Python do?
Positive lookahead	(?=...)	(?=...)
Negative lookahead	(?!...)	(?!...)
Positive lookbehind	(?<=...)	(?<=...)
Negative lookbehind	(?<!..)	(?<!..)

Lookaheads assert that the character or series of characters immediately following the current position can be represented by the given expression (here represented by '...'), while lookbehinds assert that the expression is representative of the character immediately preceeding the current position.

Positive lookarounds suggest the presence of a match, while negative lookarounds assert the absense of an expression match.