Python RegEX

Metacharacters

- ^ beginning of string
- $ end of string
- . any character except newline
- * match 0 or more times
- + match 1 or more times
- ? match 0 or 1 times; or: shortest match
- | alternative
- ( ) grouping; “storing”
- [ ] set of characters
- { } repetition modifier
- \ quote or special

Repetition / Quantifiers

- a* zero or more a’s
- a+ one or more a’s
- a?zero or one a’s (i.e., optional a)
- a{m} exactly m a’s
- a{m,} at least m a’s
- a{m,n} at least m but at most n a’s
- repetition? same as repetition but the shortest match is taken

Special notations with \

Single characters

- \t tab
- \n newline
- \r return (CR)
- \xhh character with hex. code hh

Zero-width assertions

- \b “word” boundary
- \B not a “word” boundary

Matching

- \w matches any single character classified as a “word” character (alphanumeric or “_”)
- \W matches any non-“word” character
- \s matches any whitespace character (space, tab, newline)
- \S matches any non-whitespace character
- \d matches any digit character, equiv. to [0-9]
- \D matches any non-digit character

Character sets: specialities inside [...]

Character sets

- [characters] matches any of the characters in the sequence
- [x-y] matches any of the characters from x to y (inclusively) in the ASCII code
- [\-] matches the hyphen character “-”
- [\n] matches the newline; other single character denotations with \ apply normally, too

Character sets

- [^something] matches any character except those that [something] denotes; that is, immediately after the leading “[”, the circumflex “^” means “not” applied to all of the rest

Examples

- abc abc (that exact character sequence, but anywhere in the string)
- ^abc abc at the beginning of the string
- abc$ abc at the end of the string
- a|b either of a and b
- ^abc|abc$ the string abc at the beginning or at the end of the string

Examples

- ab{2,4}c an a followed by two, three or four b's followed by a c
- ab{2,}c an a followed by at least two b's followed by a c
- ab*c an a followed by any number (zero or more) of b's followed by a c
- ab+c an a followed by one or more b's followed by a c

Examples

- a.c an a followed by any single character (not newline) followed by a c
- a\.c a.c exactly
- [abc] any one of a, b and c
- [Aa]bc either of Abc and abc
- [abc]+ any (nonempty) string of a’s, b’s and c’s (such as a, abba, acbabcacaa)

Examples

- [^abc]+ any (nonempty) string which does not contain any of a, b and c (such as defg)
- \d\d any two decimal digits, such as 42; same as \d{2}
- \w+ a “word”: a nonempty sequence of alphanumeric characters and low lines (underscores), such as foo and 12bar8 and foo_1
- 100\s*mk the strings 100 and mk optionally separated by any amount of white space (spaces, tabs, newlines)

Examples

- abc\b abc when followed by a word boundary (e.g. in abc! but not in abcd)
- perl\B perl when not followed by a word boundary (e.g. in perlert but not in perl stuff)