

Regex		
(*) indicates that the preceding character can occur 0 or more times.	meo*w	mew, meow, meoooo, and meooooo- ooooooooow
? - character can appear either 0 or 1 time	humou? r	humour humor
. and it can match any single character (letter, number, symbol or whitespace) in a piece of text	any 9-character text
[] will match any of the characters included within the brackets	con[sc]e n[sc]us	consensus, concensus, consencus, and concencus
{ } contains the exact quantity	roa{3}r	roaaar
{ }n. the quantity range of characters to be matched	roa{3,6}r	roaaar, roaaaaar, roaaaaaar, or roaaaaaar
, allows for the matching of either of two subexpressions.	baboon- s g- orillas	will match the text baboons as well as the text gorillas.

Regex (cont)		
^ and dollar sign (\$) are used in regular expressions to match text at the start and end of a string, respectively.	^Monkeys: my mortal enemy\$	will completely match the text Monkeys: my mortal enemy but not match Spider Monkeys: my mortal enemy or Monkeys: my mortal enemy in the wild

[letter-letter] or [n-n]	a range of characters that can be matched	[A-Z]. : match any uppercase letter [a-z]. : match any lowercase letter [0-9]. : match any digit [A-Za-z] : match any uppercase or lowercase letter
--------------------------	---	---

Shorthand character classes simplify writing regular expressions	\w represents the regex range [A-Za-z0-9_], \d represents [0-9],	\W represents [^A-Za-z0-9_] matching any character not included by \w, \D represents [0-9] matching any character not included by \d
--	---	--

Regex (cont)		
Negated character set	[^cdh]are	will match the m in mare.
+ indicates that the preceding character can occur 1 or more times	meo+w	will match meow, meoooo, and meooooo- oo- ooooow, but not match mew

Text Preprocessing		
Noise removal	import re result = re.sub(r"[.\?-\!\\,;\ '"]", "", text)	Removes Punctuation
Tokenization is the text preprocessing task of breaking up text into smaller components of text	from nltk.tokenize import word_tokenize word_tokenize("This is a text to tokenize") tokenized = word_tokenize(text)	print(tokenized) # ["This", "-is", "a", "-text", "to", "tokenize"]
In natural language processing, normalization encompasses many text preprocessing tasks including	stemming, lemmatization,	upper or lowercase, and stopwords removal.



By **datamansam**

cheatography.com/datamansam/

Published 30th November, 2021.
Last updated 23rd November, 2021.
Page 1 of 2.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>

Text Preprocessing (cont)

```
Stemming In      from nltk.stem # ['So',
natural          import Porter- 'mani',
language         Stemmer    'squid',
processing,      tokenized =    'are',
stemming is     ["So", "many", "- 'jump']
the text        squids", "are", "-
preprocessing   jumping"]
normalization   stemmer =
task            PorterStemmer()
concerned      stemmed =
with bluntly    [stemmer.stem(t-
removing       oken) for token
word affixes   in tokenized]
(prefixes and  suffixes).
```

```
Lemmat-         from nltk.stem ['So',
ization In      import WordNe- 'many',
natural         tLemmatizer    'squid',
language        tokenized =    'be',
processing,     ["So", "many", "- 'jump']
lemmatization  squids", "are", "-
is the text    jumping"]
preprocessing  lemmatizer =
normalization  WordNetLemmat-
task           tizer()
concerned     lemmatized =
with bringing  [lemmatizer.lem-
words down    matize(token) for
to their root token in
forms.        tokenized]
```

Text Preprocessing (cont)

```
stopword        from nltk.c- # remove
removal is     orpus      stopwords
the process    import     from tokens
of removing    stopwords  in dataset
words from a   # define set statement-
string that    of English _no_stop =
don't provide stopwords [word for
any inform-   stop_words word in
ation about   = set(st-   word_tokens
the tone of a opwords.w- if word not in
statement.    ords('eng- stop_words]
               lish'))
```

```
parser.        Uses a set  {<DT|JJ> #
chunk.Reg-    of regular chunk
expParser     expression determiners
              patterns to and
              specify the  adjectives
              behavior of
              the parser
```

Token = Smaller Component of Text
 Stem = Remove prefix and suffix
 Lemmatization = Bring down to root
 Stopword = Remove meaningless

Lists and Strings

```
z = 'Natural    z.repl-    'Natural\nLa-
Language       ace(' ',   nguage\nProc-
Processing'    '\n')     essing'
```

```
list(z)        Split text into
               character
               tokens
```

```
set(z)         Unique tokens
```

```
x = ['Natural', x.inse-    ['Language',
'Language',     rt(0,     'Natural',
'Toolkit']     'Python') 'Python',
               'Toolkit']
```



By **datamansam**

cheatography.com/datamansam/

Published 30th November, 2021.

Last updated 23rd November, 2021.

Page 2 of 2.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>