to use regular expressions just import re

compile your regular expression: re.compile

then we can do a search on the pattern!

search: finds a left most match

Hence, it won’t return all matches!!

findall: finds ALL the matches!!

(note only returns left most match for overlaps: eee => will only return left most one!)

findall: we can pass in an index from which to start looking, stop looking etc!

we can also specify a function to do the replacement!!

(this allows referring o the matched group!!)

grouping: extract the match from each pair of parentheses

why r? then, we don’t need as many escape characters!! (it’s a good practice)

\w => matches alphanumeric characters

\W => matches NON-alpha

(common pattern: capitalize to invert!)

ex: you know the common \s and \d but \S could be useful as well!!

Quantifiers:

we can simply do word matches instead of our weird space shit with html:

\w\*r+\w\*

“exactly one a regex”

boundaries section vey useful!!

\b\w+\b is not necessarily good! it strips out the punctuation!!

\b word boundaries

\B non word boundaries!!

we won’t match something like eeexp

\b => might be able to handle some special things?!

{X} used to match X number of occurrences

\b[re]\w\*\b -- mathces any words that begin with r or e!!

we can extract captialized words…!

or functions:

(?:|)

-- sometimes you absolutely need this!!

(has to do with matching the entire regex vs just a single group)

lookahead and lookbehinds!!

assumptions we can make in assignment:

instead of <.\*> [this wont work due to greediness!)

we match anything that is NOT the closing bracje

tokens with only lower case letters:

a wack expression!

matching words with 2 numbers:

his will work with 7f7

\d\S\*\d

use the final re to get a regex that works for every case!!

lookahead/lookbehind-- a way of matching without consuming!!