**Regular Expressions are the bomb!**

A regular expression is a **search pattern** that is more general than a keyword search.

**Re or String or POS?** If you are looking for a specific word or name, it may be easier to use the **String** function than the re function. And don’t forget the NLTK taggers we covered earlier if (for example) your goal is to find proper names.

**What to do with regular expressions?**

**Identify** documents that have specific things (if statement that includes re.search).

**Extract** things from documents (re.findall).

**Substitute** things in documents (re.sub)

**Parse** documents (re.split)

The first step in python is to import the **re** package and skim the documentation. Then let the fun begin!

<https://docs.python.org/2/library/re.html>

**Metacharacters.** These characters mean something special. For example, a period means ‘any character.’ A \s means space. When you are looking for an actual period, you have to precede it with a backslash (\.) which says, don’t treat this as a metacharacter. When you are looking for an actual \s you have to precede it with a backslash ([\\s](file:///%5C%5Cs)). etc.

. ^ $ \* + ? { } [ ] \ | ( )

**Syntax.** The tutorial above provides complete information about the syntax options. Combining the different syntax expressions to capture a text pattern is what makes regular expressions so powerful.

**Methods**. Like other classes of objects, re has methods associated with it. For example the re.match method asks whether the search pattern occurs at the beginning of the string, whereas the re.search pattern asks whether it occurs anywhere in the string (and where).

**Raw**. The syntax r”your text here” says ‘there are no metacharacters between the quotes.’ When you specify raw text, you don’t have to use backslashes to prevent characters from being interpreted as metacharacters

**Beware of greedy re’s.** Regular expressions look for the longest possible case that meets the regular expression conditions. Suppose that you wanted to collect the first hashtag in this (contrived) example:

#blacklivesmatter #nojustice #whatever

A regular expression that said ‘collect what lies between a hash and another hash’ will collect everything between the first and last hash by default. That’s greedy. If you just wanted what was between the first and second, you need to say that (the metacharacter ? can be useful)

(continued)

**Class activity**

1. **Open regularexpressions.py** from syllabus and work through the examples. There are probably some helpful tools for your own project if you need to:
	1. find documents that contain particular words
	2. parse documents
	3. extract things
	4. count things
2. **Go to regexr.com**

Click on cheatsheet

Use the tool and the cheatsheet to write your own regular expressions to capture

different things in the text

For example, can you capture all words, then just the first word, then just email addresses, etc?