INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Introduction to regular expressions

Katharine Jarmul
Founder, kjamistan

What is Natural Language Processing?

- Field of study focused on making sense of language
  - Using statistics and computers
- You will learn the basics of NLP
  - Topic identification
  - Text classification
- NLP applications include:
  - Chatbots
  - Translation
  - Sentiment analysis
  - ... and many more!

What exactly are regular expressions?

- Strings with a special syntax
- Allow us to match patterns in other strings
- Applications of regular expressions:
  - Find all web links in a document
  - Parse email addresses, remove/replace unwanted characters

```
In [1]: import re

In [2]: re.match('abc', 'abcdef')
Out[2]: <_sre.SRE_Match object; span=(0, 3), match='abc'>

In [3]: word_regex = '\w+'

In [4]: re.match(word_regex, 'hi there!')
Out[4]: <_sre.SRE_Match object; span=(0, 2), match='hi'>
```

Common Regex Patterns

pattern	matches	example
\w+	word	'Magic'

Common Regex patterns (2)

pattern	matches	example
\w+	word	'Magic'
\d	digit	9

Common regex patterns (3)

pattern	matches	example
\w+	word	'Magic'
\d	digit	9
\s	space	' '

Common regex patterns (4)

pattern	matches	example
\w+	word	'Magic'
\d	digit	9
\s	space	' '
.*	wildcard	'username74'

Common regex patterns (5)

pattern	matches	example
\w+	word	'Magic'
\d	digit	9
\s	space	' '
.*	wildcard	'username74'
+ or *	greedy match	'aaaaaa'

Common regex patterns (6)

pattern	matches	example
\w+	word	'Magic'
\d	digit	9
\s	space	' '
.*	wildcard	'username74'
+ or *	greedy match	'aaaaaa'
\S	**not** space	'no_spaces'

Common regex patterns (7)

pattern	matches	example
\w+	word	'Magic'
\d	digit	9
\s	space	' '
.*	wildcard	'username74'
+ or *	greedy match	'aaaaaa'
\S	**not** space	'no_spaces'
[a-z]	lowercase group	'abcdefg'

Python's re Module

- `re` module
- `split`: split a string on regex
- `findall`: find all patterns in a string
- `search`: search for a pattern
- `match`: match an entire string or substring based on a pattern

- Pattern first, and the string second
- May return an iterator, string, or match object

```
In [5]: re.split('\s+', 'Split on spaces.')
Out[5]: ['Split', 'on', 'spaces.']
```

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Introduction to tokenization

Katharine Jarmul
Founder, kjamistan

What is tokenization?

- Turning a string or document into **tokens** (smaller chunks)
- One step in preparing a text for NLP
- Many different theories and rules
- You can create your own rules using regular expressions
- Some examples:
  - Breaking out words or sentences
  - Separating punctuation
  - Separating all hashtags in a tweet

nltk library

- `nltk`: natural language toolkit

```
In [1]: from nltk.tokenize import word_tokenize

In [2]: word_tokenize("Hi there!")
Out[2]: ['Hi', 'there', '!']
```

Why tokenize?

- Easier to map part of speech
- Matching common words
- Removing unwanted tokens
- "I don't like Sam's shoes."
- "I", "do", "n't", "like", "Sam", "'s", "shoes", "."

Other nltk tokenizers

- `sent_tokenize`: tokenize a document into sentences
- `regexp_tokenize`: tokenize a string or document based on a regular expression pattern
- `TweetTokenizer`: special class just for tweet tokenization, allowing you to separate hashtags, mentions and lots of exclamation points!!!

More regex practice

- Difference between `re.search()` and `re.match()`

```
In [1]: import re

In [2]: re.match('abc', 'abcde')
Out[2]: <_sre.SRE_Match object; span=(0, 3), match='abc'>

In [3]: re.search('abc', 'abcde')
Out[3]: <_sre.SRE_Match object; span=(0, 3), match='abc'>

In [4]: re.match('cd', 'abcde')

In [5]: re.search('cd', 'abcde')
Out[5]: <_sre.SRE_Match object; span=(2, 4), match='cd'>
```

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Advanced tokenization with regex

Katharine Jarmul
Founder, kjamistan

Regex groups using or "|"

- OR is represented using `|`
- You can define a group using `()`
- You can define explicit character ranges using `[]`

```
In [1]: import re

In [2]: match_digits_and_words = ('(\d+|\w+)')

In [3]: re.findall(match_digits_and_words, 'He has 11 cats.')
Out[3]: ['He', 'has', '11', 'cats']
```

Regex ranges and groups

pattern	matches	example
[A-Za-z]+	upper and lowercase English alphabet	'ABCDEFghijk'
[0-9]	numbers from 0 to 9	9
[A-Za-z\-\.]+	upper and lowercase English alphabet, - and .	'My-Website.com'
(a-z)	a, - and z	'a-z'
(\s+I,)	spaces or a comma	', '

Character range with re.match()

```
In [1]: import re

In [2]: my_str = 'match lowercase spaces nums like 12, but no commas'

In [3]: re.match('[a-z0-9 ]+', my_str)
Out[3]: <_sre.SRE_Match object;
          span=(0, 42), match='match lowercase spaces nums like 12'>
```

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Charting word length with nltk

Katharine Jarmul
Founder, kjamistan

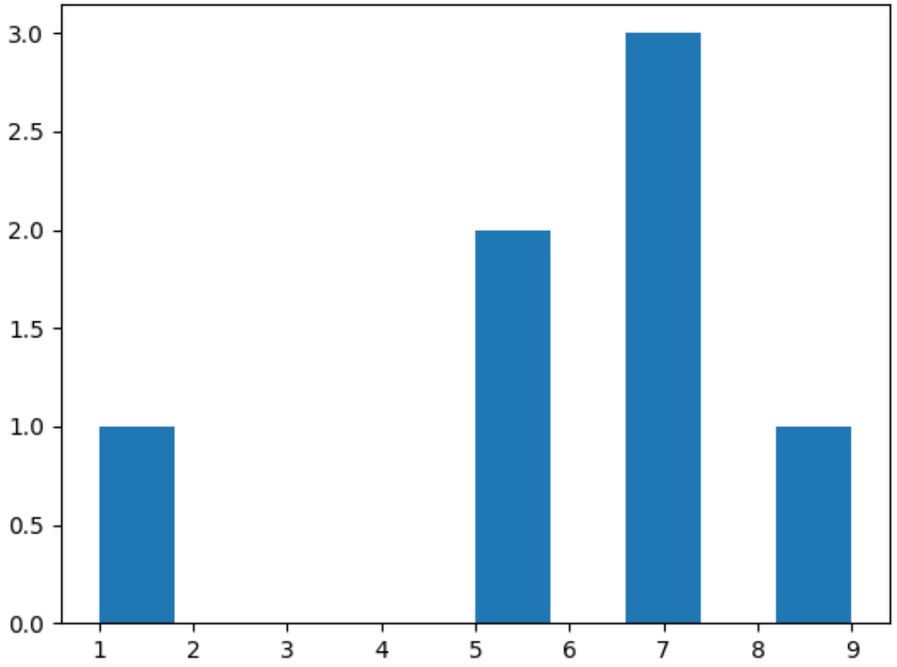
Getting started with matplotlib

- Charting library used by many open source Python projects
- Straightforward functionality with lots of options
  - Histograms
  - Bar charts
  - Line charts
  - Scatter plots
- ... and also advanced functionality like 3D graphs and animations!

Plotting a histogram with matplotlib

```
In [1]: from matplotlib import pyplot as plt

In [2]: plt.hist([1, 5, 5, 7, 7, 7, 9])
Out[2]: (array([ 1.,  0.,  0.,  0.,  0.,  2.,  0.,  3.,  0.,  1.]),
         array([ 1. ,  1.8,  2.6,  3.4,  4.2,  5. ,  5.8,  6.6,
                                                7.4,  8.2,  9. ]),
         <a list of 10 Patch objects>)

In [3]: plt.show()
```

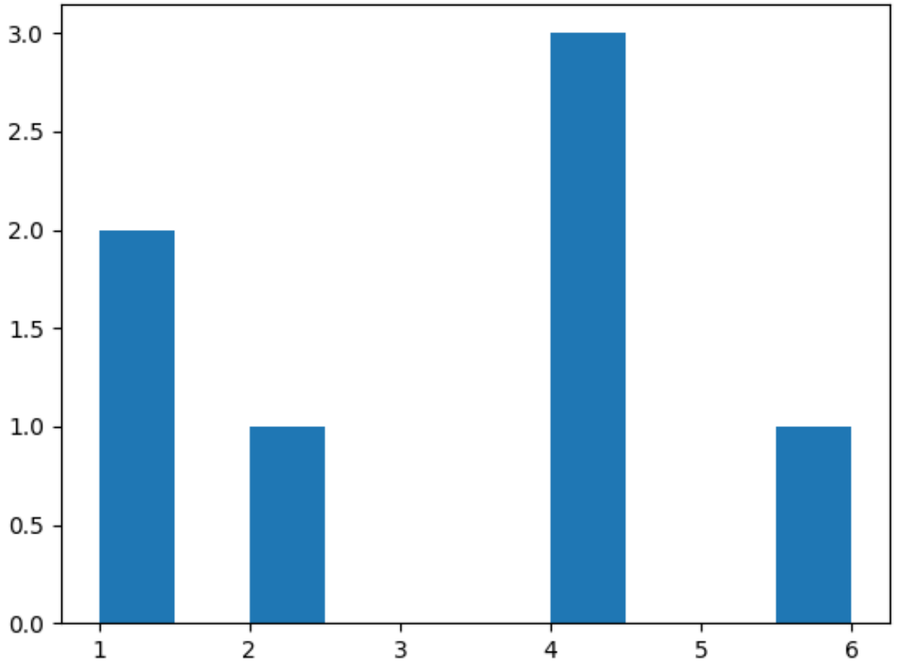
Generated Histogram



Combining NLP data extraction with plotting

```
In [1]: from matplotlib import pyplot as plt

In [2]: from nltk.tokenize import word_tokenize

In [3]: words = word_tokenize("This is a pretty cool tool!")

In [4]: word_lengths = [len(w) for w in words]

In [5]: plt.hist(word_lengths)
Out[5]: (array([ 2.,  0.,  1.,  0.,  0.,  0.,  3.,  0.,  0.,  1.]),
         array([ 1. ,  1.5,  2. ,  2.5,  3. ,  3.5,  4. ,  4.5,  5. ,  5.5,
                 6. ]),
         <a list of 10 Patch objects>)

In [6]: plt.show()
```

Word length histogram



INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

Let's practice!