

# Introduction to regular expressions

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON



**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 → Find all web links in a document
- Allow us to match patterns in other strings → Parse email addresses
- Applications of regular expressions: → Remove/replace unwanted characters

```
import re  
re.match('abc', 'abcdef')
```

```
<_sre.SRE_Match object; span=(0, 3), match='abc'>
```

```
word_regex = '\w+'  
re.match(word_regex,  
         'hi there!')
```

```
<_sre.SRE_Match object; span=(0, 2), match='hi'>
```

# Common regex patterns

pattern	matches	example
<code>\w+</code>	word	'Magic'

# Common regex patterns (2)

pattern	matches	example
<code>\w+</code>	word	'Magic'
<code>\d</code>	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	<b>not</b> 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	<b>not</b> 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

```
re.split('\s+', 'Split on spaces.')
```

```
['Split', 'on', 'spaces.']
```

# Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

# Introduction to tokenization

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON



**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

```
from nlTK.tokenize import word_tokenize  
word_tokenize("Hi there!")
```

```
['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()`

```
import re
re.match('abc', 'abcde')
```

```
<_sre.SRE_Match object; span=(0, 3), match='abc'>
```

```
re.search('abc', 'abcde')
```

```
<_sre.SRE_Match object; span=(0, 3), match='abc'>
```

```
re.match('cd', 'abcde')
re.search('cd', 'abcde')
```

```
<_sre.SRE_Match object; span=(2, 4), match='cd'>
```

# Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON

# Advanced tokenization with regex

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON



**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 []

```
import re  
match_digits_and_words = ('(\d+|\w+)')  
re.findall(match_digits_and_words, 'He has 11 cats.')
```

```
['He', 'has', '11', 'cats']
```

# Regex ranges and groups

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

# Character range with `re.match()`

```
import re
my_str = 'match lowercase spaces nums like 12, but no commas'
re.match('[a-z0-9 ]+', my_str)
```

```
<_sre.SRE_Match object;
      span=(0, 42), match='match lowercase spaces nums like 12'>
```

# Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON



# Charting word length with nltk

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON



**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!

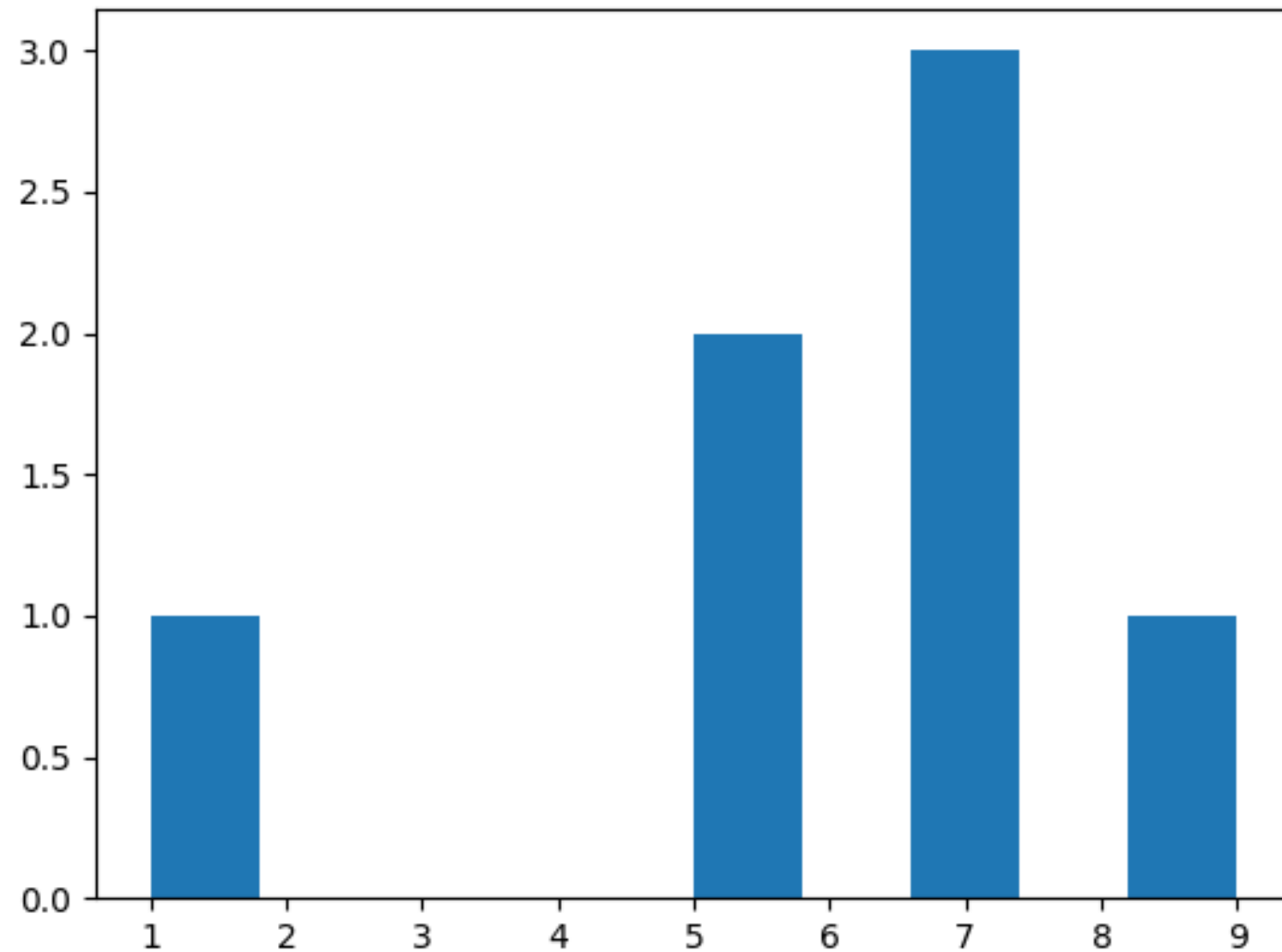
# Plotting a histogram with matplotlib

```
from matplotlib import pyplot as plt  
plt.hist([1, 5, 5, 7, 7, 7, 9])
```

```
(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>)
```

```
plt.show()
```

# Generated histogram



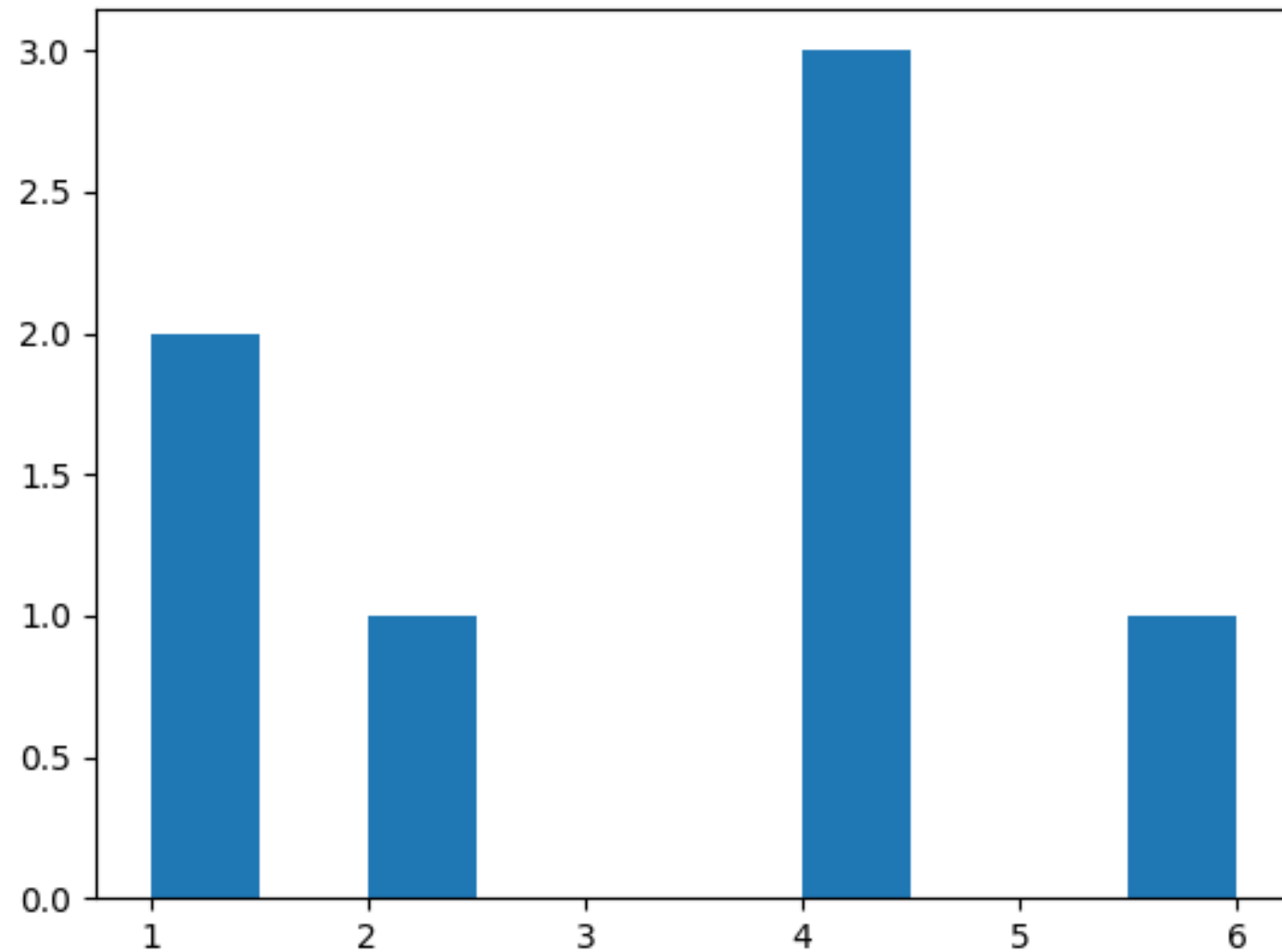
# Combining NLP data extraction with plotting

```
from matplotlib import pyplot as plt
from nltk.tokenize import word_tokenize
words = word_tokenize("This is a pretty cool tool!")
word_lengths = [len(w) for w in words]
plt.hist(word_lengths)
```

```
(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>)
```

```
plt.show()
```

# Word length histogram



# Let's practice!

INTRODUCTION TO NATURAL LANGUAGE PROCESSING IN PYTHON