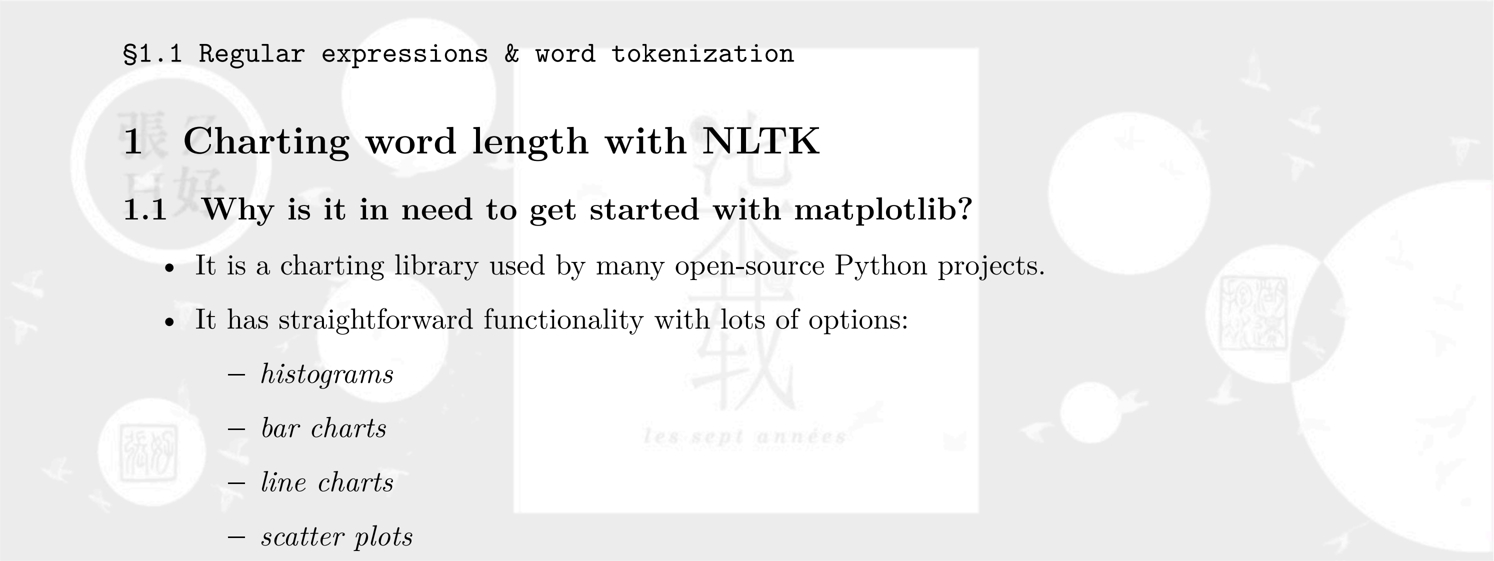
Charting word length with NLTK

Puteaux, Fall/Winter 2020-2021

```
#############################################
##                                         ##
##  Natural Language Processing in Python  ##
##                                         ##
#############################################
```



§1 Introduction to Natural Language Processing in Python

§1.1 Regular expressions & word tokenization

1 Charting word length with NLTK

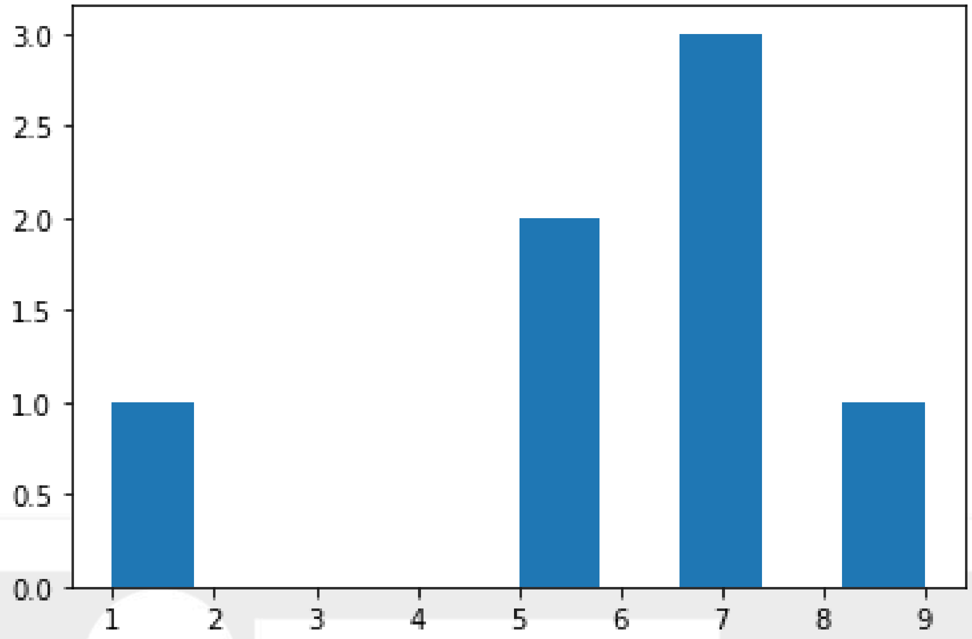
1.1 Why is it in need to get started with matplotlib?

- It is a charting library used by many open-source Python projects.
- It has straightforward functionality with lots of options:
  - *histograms*
  - *bar charts*
  - *line charts*
  - *scatter plots*
- And also, it has advanced functionality like 3D graphs and animations!

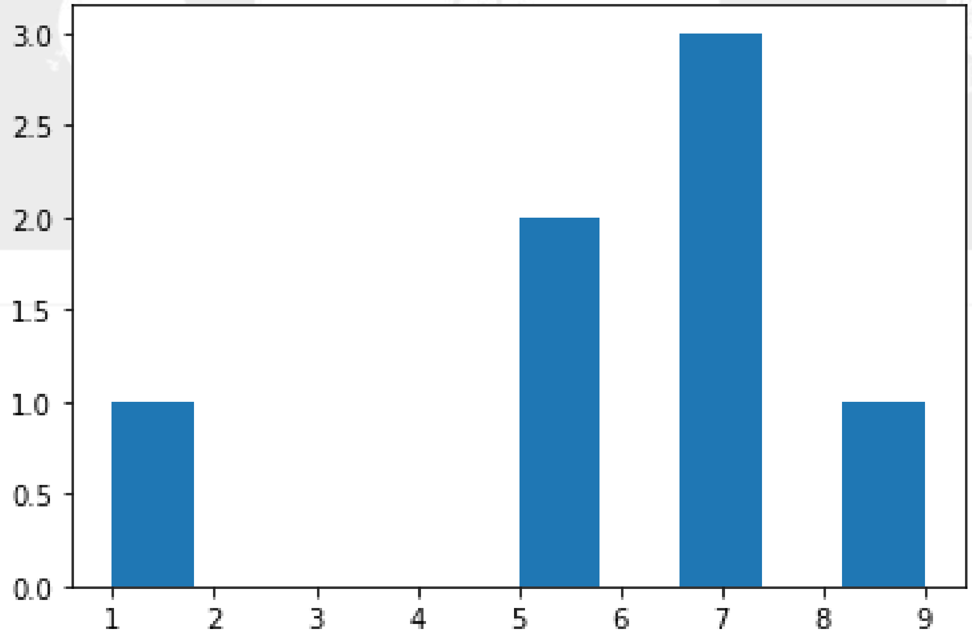
1.2 Code of plotting a histogram with matplotlib:

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

plt.hist([1, 5, 5, 7, 7, 7, 9])
```

[1]:
```
(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. ]),
 <BarContainer object of 10 artists>)
```



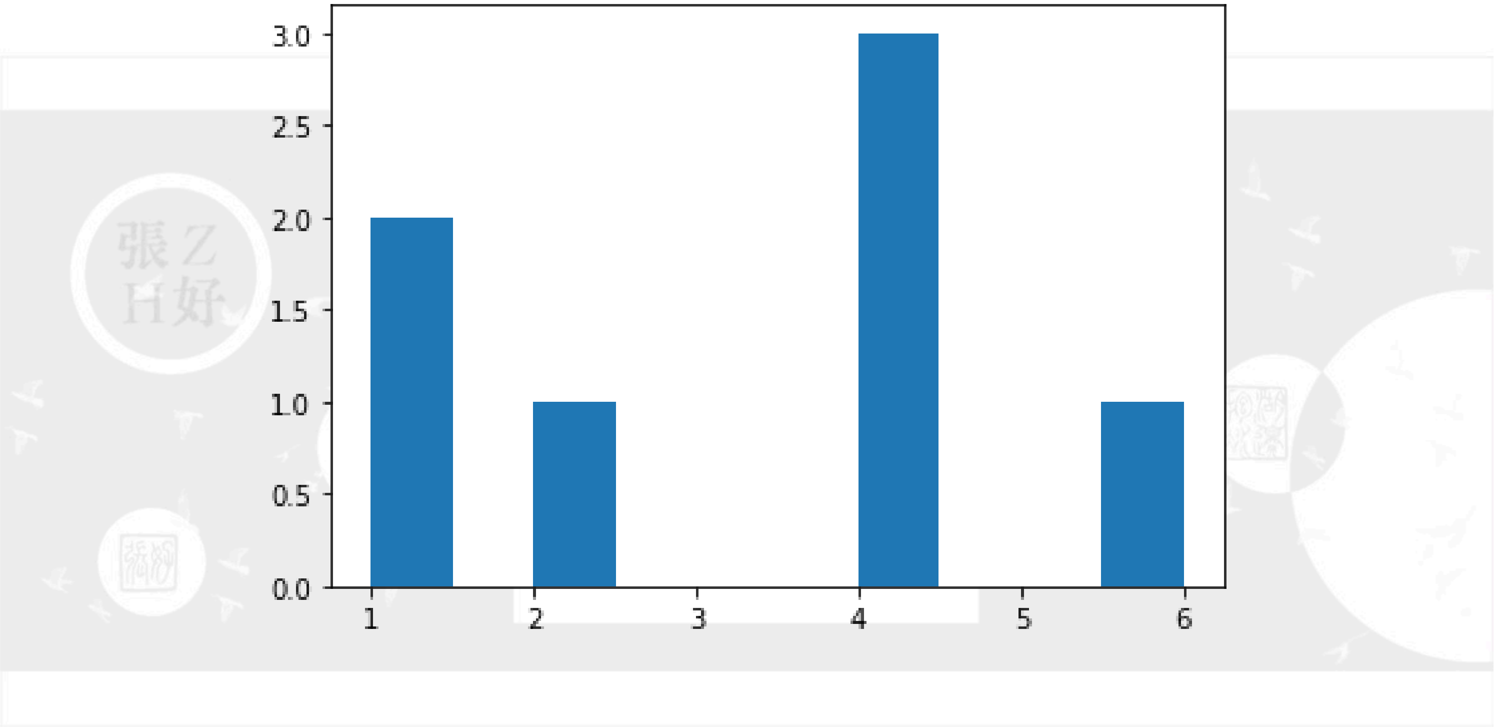
[2]:
```
plt.hist([1, 5, 5, 7, 7, 7, 9])
plt.show()
```



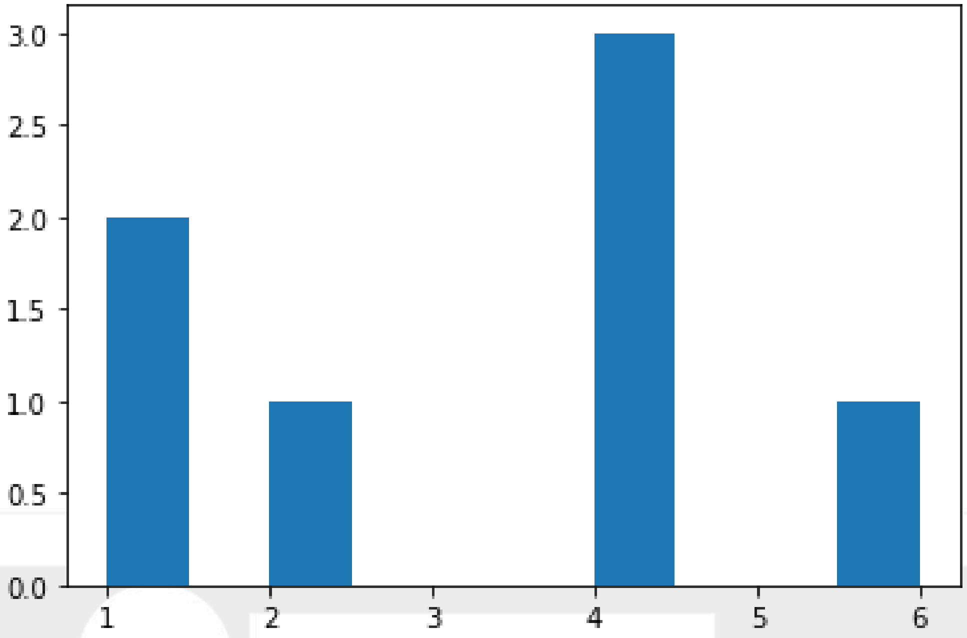
1.3 Code of combining NLP data extraction with plotting:

[3]:
```python
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)
```

[3]:
```
(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. ]),
 <BarContainer object of 10 artists>)
```



[4]:
```python
plt.hist(word_lengths)
plt.show()
```



1.4 Practice exercises for charting word length with NLTK:

▶ **Package pre-loading:**

[5]:
```
import re
from matplotlib import pyplot as plt
from nltk.tokenize import regexp_tokenize
```

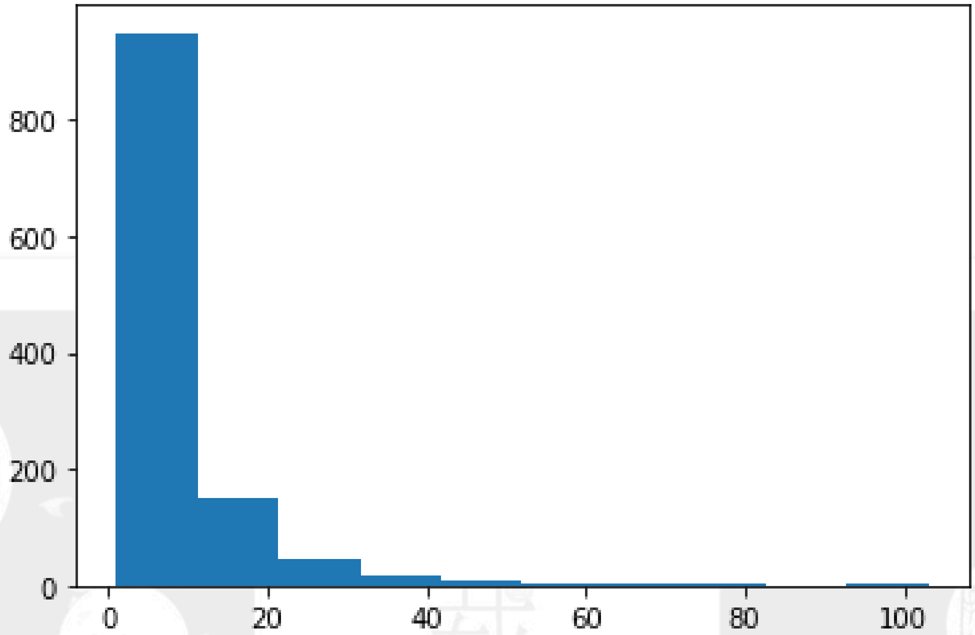
▶ **Data pre-loading:**

[6]:
```
holy_grail = open("ref2. Monty Python and the Holy Grail.txt").read()
```

▶ **Charting practice:**

[7]:
```
# Split the script into lines: lines
lines = holy_grail.split('\n')

# Replace all script lines for speaker
pattern = "[A-Z]{2,}(\s)?(#\d)?([A-Z]{2,})?:"
lines = [re.sub(pattern, '', l) for l in lines]

# Tokenize each line: tokenized_lines
tokenized_lines = [regexp_tokenize(s, '\w+') for s in lines]

# Make a frequency list of lengths: line_num_words
line_num_words = [len(t_line) for t_line in tokenized_lines]
```

```
# Plot a histogram of the line lengths
plt.hist(line_num_words)

# Show the plot
plt.show()
```


800
600
400
200
0
0
20
40
60
80
100