Hands on:
Zipf’s Law and $n$-grams

10 February 2022
Testing in Python

spaCy

Exercise: *Emma*, frequently

Assignment 3
Testing in Python
When you’re working on the assignments, you’re not required to write test cases, but it’s a good idea to convince yourself each function does what it should.
$ pip3 install pytest
def add_three(x):
    return x + 3

def test_add_three():
    assert add_three(0) == 3
    assert add_three(-3) == 0
    assert add_three(3) == 6
$ pytest testing.py

================================ test session starts ===============================
platform darwin -- Python 3.9.10, pytest-7.0.0, pluggy-1.0.0
rootdir: /Users/jgordon/Teach/366/Class/4 Probability, n-grams, and smoothing
collected 1 item

testing.py .                                                                   [100%]

================================= 1 passed in 0.01s ===========================
Sometimes the assignment gives you example output. If so, use these as test cases!

If not, read it carefully to be sure you understand the input and output types and what the function should do, and then write your own test cases.
spaCy
To install spaCy in the CS Department:

```bash
$ pip3 install -U pip setuptools wheel
$ pip3 install -U spacy
$ python3 -m spacy download en_core_web_sm
```
import spacy

nlp = spacy.blank("en")

This is a pipeline for processing natural language, including performing tokenization.
import spacy

nlp = spacy.blank("en")

doc = nlp("Hello world!")

for token in doc:
    print(token.text)

Hello
world!
import spacy

nlp = spacy.blank("en")

doc = nlp("Hello world!")

token = doc[1]
print(token.text)
import spacy

nlp = spacy.blank("en")

doc = nlp("Hello world!")

span = doc[1:3]
print(span.text)
In addition to `.text`, Tokens have the attributes

- `i`
- `is_alpha`
- `is_punct`
- `like_num`
There's much more we can do with spaCy, which we'll see as the semester continues!
Emma, frequently
Let’s use what we know about working with text in Python – including what we just saw about spaCy – to investigate the frequency of words (unigrams) and collocations (bigrams) in Jane Austen’s novel *Emma*. 
You can get the text from

/data/366/gutenberg/austen-emma.txt
To start, let’s build three objects:

- `etokens` – a list of word tokens, all in lowercase
- `etypes` – an alphabetically sorted word type list
- `efreq` – a word frequency distribution (Counter)
Code: https://www.cs.vassar.edu/~cs366/class/2022-02-10.py
Assignment 3
Acknowledgments

This class incorporates material from:

Na-Rae Han, University of Pittsburgh
Ines Montani, Advanced NLP with spaCy