Common measures of textual complexity are derived from simple counts of words, sentences and syllables. In this homework, you'll implement two of them: type-token ratio (a measure of vocabulary richness) and the Flesch-Kincaid Grade Level.

In [175…
```
import nltk, re
```

In [176…
```
# If you haven't downloaded the sentence segmentation model before, do so
nltk.download("punkt")
```

```
[nltk_data] Downloading package punkt to /Users/aayu/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```
Out[176…
```
True
```

Q1: Find two different texts you'd like to compare (from any source). For potential sources, see the The American Presidency Project for all state of the union addresses and Project Gutenberg for books in the public domain. Paste them in the `text1` and `text2` strings below. Ensure that both texts are a minimum of 500 words.

In [177…
```
text1="""The stranger came early in February, one wintry day, through a bit
Mrs. Hall lit the fire and left him there while she went to prepare him a r
“No,” he said without turning.
She was not sure she had heard him, and was about to repeat her question.
He turned his head and looked at her over his shoulder. “I prefer to keep t
“Very well, sir,” she said. “As you like. In a bit the room will be warmer
He made no answer, and had turned his face away from her again, and Mrs. Ha
“Thank you,” he said at the same time, and did not stir until she was closi
As she went behind the bar to the kitchen she heard a sound repeated at reg
She rapped and entered promptly. As she did so her visitor moved quickly, s
“Leave the hat,” said her visitor, in a muffled voice, and turning she saw
For a moment she stood gaping at him, too surprised to speak.
He held a white cloth—it was a serviette he had brought with him—over the l
```

In [178…
```
text2="""On an exceptionally hot evening early in July a young man came out
He had successfully avoided meeting his landlady on the staircase. His gari
This was not because he was cowardly and abject, quite the contrary; but fo
This evening, however, on coming out into the street, he became acutely awa
“I want to attempt a thing like that and am frightened by these trifles,” h
The heat in the street was terrible: and the airlessness, the bustle and th
He was so badly dressed that even a man accustomed to shabbiness would have
“I knew it,” he muttered in confusion, “I thought so! That’s the worst of a
He had not far to go; he knew indeed how many steps it was from the gate of
```

Q2: Use the `nltk.word_tokenize` method to implement the type-token ratio:

$$TTR = \frac{\text{number of distinct word types}}{\text{number of word tokens}}$$

TTR is dependent on text length (intuitively, the longer a text is, the greater chance you have of a word type repeating), so this number is only comparable between documents of identical lengths. Calculate this measure for the first 500 words of your two documents and report the results here. Exclude tokens that are exclusively punctuation from all counts.

In [179…
```
def type_token_ratio(text, num_words=500):
    # your answer here
    # get the first 500 words from raw text
    text = text.lower()
    tokens = text.split(' ')[0:num_words]
    # join them back into raw text
    text_shortened = ' '.join(tokens)
    tokens = nltk.word_tokenize(text_shortened)
    # regex to remove all words which are exclusively punctuation
    punct_regex = re.compile(r'[^a-z0-9 ]+')
    tokens = [i for i in tokens if not punct_regex.match(i)]
    # types: unique tokens
    types = set(tokens)
    # final TTR
    return len(types)/len(tokens)
```

In [180…
```
type_token_ratio(text1)
```

Out[180… `0.4871287128712871`

In [181…
```
type_token_ratio(text2)
```

Out[181… `0.49902152641878667`

Now we'll implement the Flesch-Kincaid Grade Level, which has the following formula:

$$0.39\left(\frac{\text{total words}}{\text{total sentences}}\right) + 11.8\left(\frac{\text{total syllables}}{\text{total words}}\right) - 15.59$$

Use `nltk.sent_tokenize` or spacy's `sents` function for counting the number of sentences, any word tokenization method we've covered for counting the number of words, and the `get_syllable_count` function below for counting the number of syllables in a word. Exclude tokens that are exclusively punctuation from word and syllable counts.

For calculating the syllables, we're going to use a number of resources: the CMU pronunciation dictionary, which lists the ARPABET pronunciation for a list of words, along with g2p, a neural model trained to predict the pronunciation for words (which we can use for words not in the CMU dictionary).

In [182…
```
arpabet = nltk.corpus.cmudict.dict()
```

In [183…
```
!pip install g2p_en
```

```
Requirement already satisfied: g2p_en in /opt/anaconda3/envs/anlp/lib/pytho
n3.8/site-packages (2.1.0)
Requirement already satisfied: numpy>=1.13.1 in /opt/anaconda3/envs/anlp/li
b/python3.8/site-packages (from g2p_en) (1.20.3)
Requirement already satisfied: distance>=0.1.3 in /opt/anaconda3/envs/anlp/
lib/python3.8/site-packages (from g2p_en) (0.1.3)
Requirement already satisfied: nltk>=3.2.4 in /opt/anaconda3/envs/anlp/lib/
python3.8/site-packages (from g2p_en) (3.6.2)
Requirement already satisfied: inflect>=0.3.1 in /opt/anaconda3/envs/anlp/l
ib/python3.8/site-packages (from g2p_en) (5.3.0)
Requirement already satisfied: tqdm in /opt/anaconda3/envs/anlp/lib/python3
.8/site-packages (from nltk>=3.2.4->g2p_en) (4.62.1)
Requirement already satisfied: joblib in /opt/anaconda3/envs/anlp/lib/pytho
n3.8/site-packages (from nltk>=3.2.4->g2p_en) (1.0.1)
Requirement already satisfied: regex in /opt/anaconda3/envs/anlp/lib/python
3.8/site-packages (from nltk>=3.2.4->g2p_en) (2021.8.3)
Requirement already satisfied: click in /opt/anaconda3/envs/anlp/lib/python
3.8/site-packages (from nltk>=3.2.4->g2p_en) (7.1.2)
```

In [184…
```
from g2p_en import G2p
g2p = G2p()
```

In [185…
```
def get_pronunciation(word):
    if word in arpabet:
        # pick the first pronunciation
        return arpabet[word][0]

    else:
        return g2p(word)

def get_syllable_count(word):
    pronunciation=get_pronunciation(word)
    sylls=0
    for phon in pronunciation:
        # vowels in arpabet end in digits (indicating stress)
        if re.search("\d$", phon) is not None:
            sylls+=1
    return sylls
```

In [186…
```
get_syllable_count("Bamman")
```

Out[186… 2

In [187…
```
get_syllable_count("27")
```

Out[187… 4

Q3. Implement Flesch-Kincaid Grade Level and report its results for your two texts. Flesch-Kincaid relies on an implicit definition of a "word" and a "sentence", and different definitions will yield different grade level estimates. (In the problem definition above, we've already ruled out punctuation as constituing stand-alone words, and other assumptions lurk with every tokenization method). State your assumptions for the definition of "word" you have implemented and why they are reasonable.

In [188…
```
def flesch_kincaid_grade_level(text):
    # your answer here
    sent_count = len(nltk.sent_tokenize(text))
    tokens = nltk.word_tokenize(text)
    # regex to remove all words which are exclusively punctuation
    punct_regex = re.compile(r'[^a-z0-9 ]+')
    tokens = [i for i in tokens if not punct_regex.match(i)]
    word_count = len(tokens)
    # calculate Flesch-Kincaid Grade Level
    fkg_level = 0.39*(word_count/sent_count) + 11.8*(get_syllable_count(te
    return fkg_level
```

In [189…
```
# Should be 11.265
print(flesch_kincaid_grade_level("The Australian platypus is seemingly a, l
```

```
11.264615384615386
```

In [190…
```
flesch_kincaid_grade_level(text1.lower())
```

Out[190… 8.050391463649806

In [191…
```
flesch_kincaid_grade_level(text2.lower())
```

Out[191… 10.007810877564392

Q3 "word" assumptions:

Some assumptions being made in the definition of "word" are:

1) The delimiter between words (what marks the end of a word) is a single space character. This is reasonable because spaces are used to differentiate between words in the English language. While this may not hold true for some other languages, it's the general rule of thumb for English.

2) Hyphenated words are counted as one word, which increases the Flesch-Kincaid Grade Level value for texts which have them. This is automatically biased towards books which have lesser/no hyphenated words, and will show lower scores (hence more complexity) for books with lesser hyphens.

In [ ]: