I completed the 3 reading chapters of the assignment, installed the NLP related packages and NLTK packages on my Python distribution. I then proceeded to run the Jupyter notebooks and tinker with the code. I encountered an issue while importing spacy.en and I resolved by correcting it to spacy.lang.en.

Chapter 1 elucidated primarily on points discussed in class (slides) in a more in-depth fashion. It discusses about textual structure, its components, and tokens.As I am new to NLP, it allowed me to grasp what we discussed in class much better.
The Jupyter notebooks contained Constituency Parsing, Dependency parsing and Part of Speech Tagging (POS Tagging) which were discussed in Chapter 1. Playing around with this code and more importantly, with different sentences allowed me to understand and get more insight into these parsings.

Chapter 2 was an introductory chapter to Python from a NLP perspective. This chapter was easy to digest as I have worked with Python before. However, there was some interesting new information towards the end of the chapter regarding functional programming, various encodings, and Regular expressions. Finally, it introduced some important packages that I promptly installed. I played with gensim from some tutorials to see Google's word2vec in action (Based on this tutorial:
https://machinelearningmastery.com/develop-word-embeddings-python-gensim/).

Chapter 3 deals with text processing and normalizing. The first topic is tokenization which is used in almost all the notebooks. While running the code gave me an idea of tokenization, the specific variations like splitting 'couldn't' into 'could' and ' n't ' etc were explained quite well in the reading material. The Text Analytics for Python notebook dealt with the implementation of these codes and showcased the various tokenizations. The NLP for hackers was an introductory notebook for NLTK and introduced something called NLP Pyramid.

A large part of text normalization involves "cleaning the text" which involves removing special characters, punctuations, stopwords, expanding contractions, correcting words, stemming (where we discussed various stemmers), lemmatization and case conversions. The repeating characters section in correcting words was particularly interesting as I had worked with this on a previous project involving real-time video OCR of printed text.
The Text Normalization notebook gave implementations of the various 'text cleaning'. Some like stemming and lemmatization were built into nltk and other were implemented as functions with the help of some library methods.
The 'Correcting spellings' notebook was interesting as it showed how simple functions like suggest() can return suggestion for error correction. It also worked on edit distances which is something that I had previously encountered while studying Algorithms (Dynamic Programming). Corrective suggestions seemed to be based on statistical probability and edit distances.

Chapter 3 also revisits constituency parsing, dependency parsing, POS tagging and introduces Shallow parsing. It focuses on which is the recommended methods while also showing how to build our own versions of such parsers and taggers through training them.
In POS tagging, the explanation of the various POS tags helped me understand both the result of the code and the slides better.
The POS tagging notebook allowed us to build various POS taggers using RegEx Taggers, N-Gram taggers, combined taggers etc. and training them. We finally get a accuracy of 93% with ClassifierBasedPOSTagger and NaiveBayesClassifier and 98.4% with MaxentClassifier.
The constituency parsing and dependency parsing notebooks is a direct implementation of what is discussed in Chapter 3. It shows the recommended parsers and how to build our own parsers.
The shallow parsing notebook enables us to light parse or chunk a sentence and visualize it in many ways. There is also code to build our own shallow parsers with RegEx, combined taggers etc. We are able to evaluate and get upto 99.6% accuracy for some treebank chunk data.