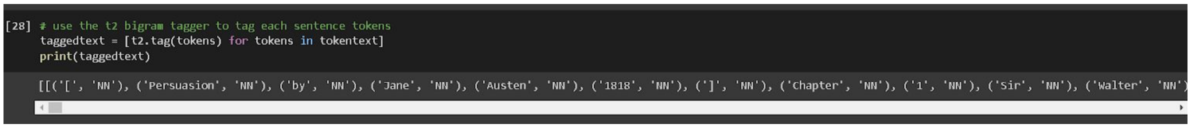
Book Name: Persuasion by Jane Austen

Outputs and Results:

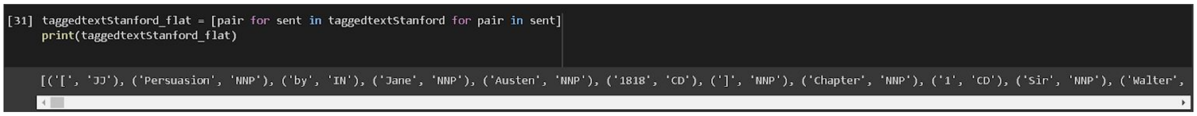
Using bigram tagger



```
[28] # use the t2 bigram tagger to tag each sentence tokens
     taggedtext = [t2.tag(tokens) for tokens in tokentext]
     print(taggedtext)

     [[('[', 'NN'), ('Persuasion', 'NN'), ('by', 'NN'), ('Jane', 'NN'), ('Austen', 'NN'), ('1818', 'NN'), (']', 'NN'), ('Chapter', 'NN'), ('1', 'NN'), ('Sir', 'NN'), ('Walter', 'NN')
```

Using Standford tagger



```
[31] taggedtextStanford_flat = [pair for sent in taggedtextStanford for pair in sent]
     print(taggedtextStanford_flat)

     [('[', 'JJ'), ('Persuasion', 'NNP'), ('by', 'IN'), ('Jane', 'NNP'), ('Austen', 'NNP'), ('1818', 'CD'), (']', 'NNP'), ('Chapter', 'NNP'), ('1', 'CD'), ('Sir', 'NNP'), ('Walter',
```

```
dict_keys(['JJ', 'NNP', 'IN', 'CD', ',', 'VBD', 'DT', 'NN', 'WP', 'PRP$',
'RB', 'RP', 'CC', ':', 'EX', 'PRP', 'NNS', 'VBN', 'VBG', 'JJS', 'MD',
'VB', 'WDT', '.', '``', '(', ')', 'VBZ', "''", 'POS', 'WRB', 'PDT', 'JJR',
'RBR', 'TO', 'WP$', 'RBS', 'VBP', 'NNPS', 'UH', 'FW'])
```

```
NN 10217
IN 10133
, 7024
DT 7012
PRP 7002
RB 6046
VBD 5337
JJ 5326
NNP 4968
VB 4151
CC 3799
. 3654
VBN 2805
TO 2781
PRP$ 2403
NNS 2333
MD 1897
VBG 1763
: 1591
VBP 1188
'' 941
```

```
VBZ 769
`` 652
WDT 492
CD 479
POS 471
WRB 429
WP 417
JJR 286
PDT 245
RP 238
JJS 231
EX 218
RBR 212
UH 106
RBS 84
( 70
) 70
WP$ 21
NNPS 20
FW 7
```

```
dict_keys(['J', 'N', 'I', 'C', ',', 'V', 'D', 'W', 'P', 'R', ':', 'E',
'M', '.', '`', '(', ')', "'", 'T', 'U', 'F'])
```

```
N 17538
V 16013
I 10133
P 10121
, 7024
D 7012
R 6580
J 5843
C 4278
. 3654
T 2781
M 1897
: 1591
W 1359
' 941
` 652
E 218
U 106
( 70
) 70
F 7
```

Observations:

- The backoff tagger capability of NLTK to create a combined tagger using Unigram tagger and Bigram tagger. The accuracy of the bigram was better than the unigram tagger.
- I was able to clearly understand how POS tagging works by training the sequence N-gram tagger with Penn Treebark.
- NN is the most common tag which is a noun.

Lesson Learned

1. The proper usage and utilization of Python tuples.
2. Part of speech tagging using an N-gram tagger and Stanford tagger.