(http://www.pieriandata.com)

NLP Basics Project

For this project, I will be using the short story *An Occurrence at Owl Creek Bridge* (https://en.wikipedia.org/wiki/An_Occurrence_at_Owl_Creek_Bridge) by Ambrose Bierce (1890). The story is in the public domain; the text file was obtained from Project Gutenberg (https://www.gutenberg.org/ebooks/375.txt.utf-8).

In [1]:
```
# RUN THIS CELL to perform standard imports:
import spacy
nlp = spacy.load('en_core_web_sm')
```

1. Create a Doc object from the file `owlcreek.txt`

> HINT: Use `with open('../TextFiles/owlcreek.txt') as f:`

In [2]:
```
# Enter your code here:
with open('../TextFiles/owlcreek.txt') as f:
    doc = nlp(f.read())
```

In [3]:
```
#Run this cell to verify it worked:

doc[:36]
```

Out[3]:
```
AN OCCURRENCE AT OWL CREEK BRIDGE

by Ambrose Bierce

I

A man stood upon a railroad bridge in northern Alabama, looking down
into the swift water twenty feet below.
```

In [3]:
```
# Run this cell to verify it worked:

doc[:36]
```

Out[3]:
```
AN OCCURRENCE AT OWL CREEK BRIDGE

by Ambrose Bierce

I

A man stood upon a railroad bridge in northern Alabama, looking down
into the swift water twenty feet below.
```

2. How many tokens are contained in the file?

In [4]:
```

```

Out[4]:
```
4833
```

In [4]:
```
len(doc)
```

Out[4]:
```
4833
```

3. How many sentences are contained in the file?
HINT: You'll want to build a list first!

In [5]:
```

```

Out[5]:
```
211
```

In [5]:
```
sentences = []

for sentence in doc.sents:
    sentences.append(sentence)

len(sentences)
```

Out[5]: 211

4. Print the second sentence in the document
HINT: Indexing starts at zero, and the title counts as the first sentence.

In [6]:
```

```

```
A man stood upon a railroad bridge in northern Alabama, looking down
into the swift water twenty feet below.
```

In [27]:
```
print(sentences[1].text)
```

```
A man stood upon a railroad bridge in northern Alabama, looking down
into the swift water twenty feet below.
```

** 5. For each token in the sentence above, print its `text`, `POS` tag, `dep` tag and `lemma`
CHALLENGE: Have values line up in columns in the print output.**

In [7]:
```
# NORMAL SOLUTION:
```

```
A DET det a
man NOUN nsubj man
stood VERB ROOT stand
upon ADP prep upon
a DET det a
railroad NOUN compound railroad
bridge NOUN pobj bridge
in ADP prep in
northern ADJ amod northern
Alabama PROPN pobj alabama
, PUNCT punct ,
looking VERB advcl look
down PART prt down

 SPACE

into ADP prep into
the DET det the
swift ADJ amod swift
water NOUN pobj water
twenty NUM nummod twenty
feet NOUN npadvmod foot
below ADV advmod below
. PUNCT punct .
  SPACE
```

In [8]:
```
for token in sentences[1]:
    print(token.text, token.pos_, token.dep_, token.lemma_)
```

```
A DET det a
man NOUN nsubj man
stood VERB ROOT stand
upon ADP prep upon
a DET det a
railroad NOUN compound railroad
bridge NOUN pobj bridge
in ADP prep in
northern ADJ amod northern
Alabama PROPN pobj alabama
, PUNCT punct ,
looking VERB advcl look
down PART prt down

 SPACE

into ADP prep into
the DET det the
swift ADJ amod swift
water NOUN pobj water
twenty NUM nummod twenty
feet NOUN npadvmod foot
below ADV advmod below
. PUNCT punct .
  SPACE
```

In [8]:
```
# CHALLENGE SOLUTION:
```

```
A               DET   det         a
man             NOUN  nsubj       man
stood           VERB  ROOT        stand
upon            ADP   prep        upon
a               DET   det         a
railroad        NOUN  compound    railroad
bridge          NOUN  pobj        bridge
in              ADP   prep        in
northern        ADJ   amod        northern
Alabama         PROPN pobj        alabama
,               PUNCT punct       ,
looking         VERB  advcl       look
down            PART  prt         down

                SPACE

into            ADP   prep        into
the             DET   det         the
swift           ADJ   amod        swift
water           NOUN  pobj        water
twenty          NUM   nummod      twenty
feet            NOUN  npadvmod    foot
below           ADV   advmod      below
.               PUNCT punct       .
                SPACE
```

In [30]:
```
# CHALLENGE SOLUTION:

for token in sentences[1]:
    print(f'{token.text:{15}}, {token.pos_:{5}}, {token.dep_:{10}}, {tok
```

```
A               , DET  , det        , a
man             , NOUN , nsubj      , man
stood           , VERB , ROOT       , stand
upon            , ADP  , prep       , upon
a               , DET  , det        , a
railroad        , NOUN , compound   , railroad
bridge          , NOUN , pobj       , bridge
in              , ADP  , prep       , in
northern        , ADJ  , amod       , northern
Alabama         , PROPN, pobj       , alabama
,               , PUNCT, punct      , ,
looking         , VERB , advcl      , look
down            , PART , prt        , down

               , SPACE,            ,

into            , ADP  , prep       , into
the             , DET  , det        , the
swift           , ADJ  , amod       , swift
water           , NOUN , pobj       , water
twenty          , NUM  , nummod     , twenty
feet            , NOUN , npadvmod   , foot
below           , ADV  , advmod     , below
.               , PUNCT, punct      , .
                , SPACE,            ,
```

6. Write a matcher called 'Swimming' that finds both occurrences of the phrase "swimming vigorously" in the text

HINT: You should include an `'IS_SPACE': True` pattern between the two words!

In [12]:
```
# Import the Matcher library:

from spacy.matcher import Matcher
matcher = Matcher(nlp.vocab)
```

In [31]:
```
# Create a pattern and add it to matcher:

pattern1 = [{'LOWER': 'swimming'}, {'IS_SPACE': True, 'OP': '*'}, {'LOWE

matcher.add('Swimming', None, pattern1)
```

In [11]:
```
# Create a list of matches called "found_matches" and print the list:
```

```
[(12881893835109366681, 1274, 1277), (12881893835109366681, 3607, 3610
)]
```

In [32]:
```
found_matches = matcher(doc)

print(found_matches)
```

```
[(12881893835109366681, 1274, 1277), (12881893835109366681, 1274, 1277
), (12881893835109366681, 3607, 3610), (12881893835109366681, 3607, 36
10)]
```

7. Print the text surrounding each found match

In [12]:
```

```

```
By diving I could evade the bullets and, swimming
vigorously, reach the bank, take to the woods and get away home
```

In [20]:
```
match_id, start, end = found_matches[0]
span = doc[start - 10: end + 13]
print(span.text)
```

```
 By diving I could evade the bullets and, swimming
vigorously, reach the bank, take to the woods and get away home
```

In [13]:
```

```

```
over his shoulder; he was now swimming
vigorously with the current.
```

In [23]:
```
match_id, start, end = found_matches[1]
span = doc[start - 7: end + 4]
print(span.text)
```

```
over his shoulder; he was now swimming
vigorously with the current.
```

EXTRA CREDIT:
Print the *sentence* that contains each found match

In [18]:
```
```

```
By diving I could evade the bullets and, swimming
vigorously, reach the bank, take to the woods and get away home.
```

In [24]:
```
match_id, start, end = found_matches[0]
span = doc[start - 10: end + 14]
print(span.text)
```

```
 By diving I could evade the bullets and, swimming
vigorously, reach the bank, take to the woods and get away home.
```

In [34]:
```
for sentence in sentences:
    if found_matches[0][1] < sentence.end:
        print(sentence)
        break
```

```
By diving I could evade the bullets and, swimming
vigorously, reach the bank, take to the woods and get away home.
```

In [19]:
```
```

```
The hunted man saw all this over his shoulder; he was now swimming
vigorously with the current.
```

In [26]:
```
match_id, start, end = found_matches[1]
span = doc[start - 13: end + 4]
print(span.text)
```

```
The hunted man saw all this over his shoulder; he was now swimming
vigorously with the current.
```

In [ ]:
```
found
```

In [38]:
```
for sentence in sentences:
    if found_matches[1][1] < sentence.end:
        print(sentence)
        break
```

```
By diving I could evade the bullets and, swimming
vigorously, reach the bank, take to the woods and get away home.
```