Exercise 1:

To find the language of a string, you have to iterate through a sample list of words from that language and compare them against every word in the inputted text, every time a word is matched increment the counter for that language by one, at the end of the phrase compare each counter to work out which one has the highest value, the counter with the highest value determines what language the text is.

```
>>> def findLanguage(stext):
        stext = stext.split()
        engList = [ 'the', 'a', 'and', 'of', 'be', 'that', 'have', 'it', 'for', 'not' ]
        frList = [ 'le', 'la', 'de', 'ne', 'et', 'un', 'pas', 'vous', 'etre', 'les', 'en' ]
        espList = ['El', 'la', 'de', 'voluntad', 'y', 'a', 'no', 'tú', 'ser', 'la', 'es']
        engCount = 0
        frCount = 0
        espCount = 0
        for c in stext:
            if c in engList:
                engCount = engCount + 1
            if c in frList:
                frCount = frCount + 1
            if c in espList:
                espCount = espCount + 1

        if engCount > frCount:
            if engCount > espCount:
                print 'the language is english'
        elif frCount > espCount:
            print 'the language is french'
        else:
            print 'the language is spanish'
```

```
>>> findLanguage("And they said, Come, let us build ourselves a city, and a tower whose top is in the heavens; let us make a name for ourselves, lest we be scattered abroad over the face of the whole earth.")
```

the language is english

>>> findLanguage("Et ils ont dit: Venez, bâtissons-nous une ville et une tour dont le sommet est dans les cieux;. Faisons un nom pour nous-mêmes, de peur que nous ne soyons dispersés sur la face de toute la terre.")

the language is french

>>> findLanguage("Y dijeron: Vamos, edifiquémonos una ciudad y una torre cuya cúspide llegue al cielo;. Vamos a hacer un nombre para nosotros mismos, para que no seamos esparcidos sobre la faz de toda la tierra")

the language is Spanish

Exercise 2:

>>> sampleTexts = [["the lion the witch and the wardrobe"],["le lion , la sorcière et l'armoire magique"],["el león la bruja y el armario"], ["this is a sample sentence using the english language"], ["Ce est une phrase de l'échantillon en utilisant la langue anglaise"], ["esta es una frase de ejemplo utilizando el idioma Inglés"], ["a set of words that is complete in itself, typically containing a subject and predicate, conveying a statement, question, exclamation, or command, and consisting of a main clause and sometimes one or more subordinate clauses."], ["un ensemble de mots qui est complet en soi, contenant typiquement un sujet et le prédicat, véhiculant une déclaration, question, exclamation, ou commande, et composé d'une clause principale et parfois une ou plusieurs propositions subordonnées."],["un conjunto de palabras que es completo en sí mismo, por lo general contiene un sujeto y predicado, transmitiendo una declaración, pregunta, exclamación, o un comando, y que consiste en una oración principal y, a veces una o más cláusulas subordinadas."],["una gran serpiente de consistencia pesada no venenosa que ocurre en los trópicos del Viejo Mundo, matar a sus presas por constricción y asfixia."]]

>>> for i in sampleTexts:

```
    stext = i

    text = text.join(stext)

    print text

    findLanguage(text)
```

the lion the witch and the wardrobe

the language is english

le lion , la sorcière et l'armoire magique

the language is french

el león la bruja y el armario

the language is spanish

this is a sample sentence using the english language

the language is english

Ce est une phrase de l'échantillon en utilisant la langue anglaise

the language is french

esta es una frase de ejemplo utilizando el idioma Inglés

the language is spanish

a set of words that is complete in itself, typically containing a subject and predicate, conveying a statement, question, exclamation, or command, and consisting of a main clause and sometimes one or more subordinate clauses.

the language is english

un ensemble de mots qui est complet en soi, contenant typiquement un sujet et le prédicat, véhiculant une déclaration, question, exclamation, ou commande, et composé d'une clause principale et parfois une ou plusieurs propositions subordonnées.

the language is french

un conjunto de palabras que es completo en sí mismo, por lo general contiene un sujeto y predicado, transmitiendo una declaración, pregunta, exclamación, o un comando, y que consiste en una oración principal y, a veces una o más cláusulas subordinadas.

the language is french

una gran serpiente de consistencia pesada no venenosa que ocurre en los trópicos del Viejo Mundo, matar a sus presas por constricción y asfixia.

the language is Spanish

Out of ten tests my program managed to successfully identity nine languages correctly which gives my program an accuracy of 90%.

Exercise 3:

```
>>> def npChunk(word):
        word = nltk.word_tokenize(word)
        word = nltk.pos_tag(word)
        grammar = "NP: {<DT>?<JJ>*<NN>}"
        cp = nltk.RegexpParser(grammar)
        result = cp.parse(word)
        print (result)
        result.draw()
```

```
>>> npChunk("And they said, Come, let us build ourselves a city, and a tower whose top is in the heavens; let us make a name for ourselves, lest we be scattered abroad over the face of the whole earth.")
(S
  And/CC
  they/PRP
  said/VBD
  ,/,
  Come/NNP
  ,/,
  let/VB
  us/PRP
  build/VB
  ourselves/NNS
  (NP a/DT city/NN)
  ,/,
  and/CC
  (NP a/DT tower/NN)
  whose/WP$
  top/JJ
  is/VBZ
  in/IN
  the/DT
  heavens/NNS
  ;/:
  let/VB
  us/PRP
  make/VB
  (NP a/DT name/NN)
  for/IN
  ourselves/NNS
  ,/,
```

lest/VBP

we/PRP

be/VB

scattered/VBN

abroad/RB

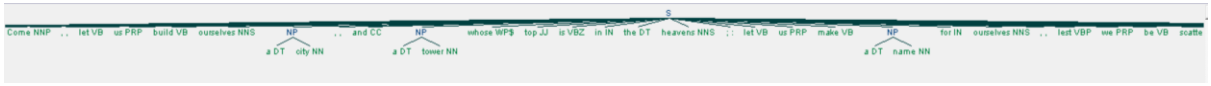
over/IN

(NP the/DT face/NN)

of/IN

(NP the/DT whole/JJ earth/NN)

./.)



npChunk("this is a sample sentence using the english language")

(S

this/DT

is/VBZ

(NP a/DT sample/NN)

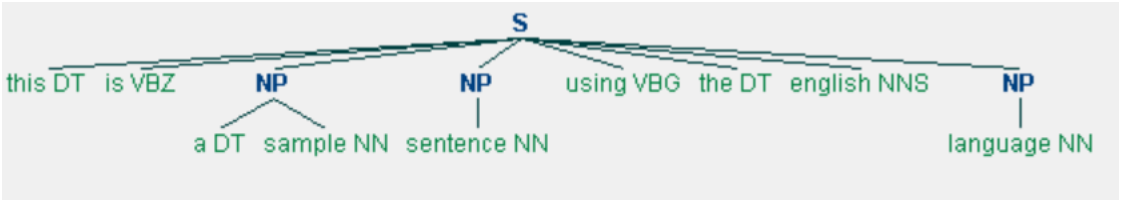
(NP sentence/NN)

using/VBG

the/DT

english/NNS

(NP language/NN))



```
>>> npChunk("a set of words that is complete in itself, typically containing a subject and predicate, conveying a statement, question, exclamation, or command, and consisting of a main clause and sometimes one or more subordinate clauses.")
(S
  (NP a/DT set/NN)
  of/IN
  words/NNS
  that/WDT
  is/VBZ
  complete/JJ
  in/IN
  itself/PRP
  ,/,
  typically/RB
  containing/VBG
  (NP a/DT subject/NN)
  and/CC
  (NP predicate/NN)
  ,/,
  conveying/VBG
  (NP a/DT statement/NN)
  ,/,
  (NP question/NN)
  ,/,
  (NP exclamation/NN)
  ,/,
  or/CC
  command/CC
  ,/,
  and/CC
  consisting/VBG
  of/IN
```

(NP a/DT main/JJ clause/NN)

and/CC

sometimes/RB

one/CD

or/CC

more/JJR

subordinate/JJ

clauses/NNS

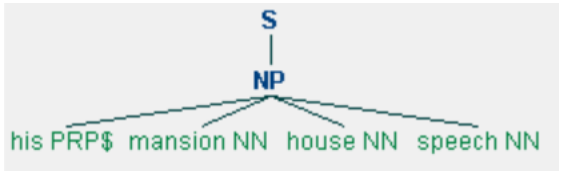
./.)

Exercise 4:

```
>>> def npChunk(word):
        word = nltk.word_tokenize(word)
        word = nltk.pos_tag(word)
        grammar = "NP:
{<PRP\$|DT|LS|JJR>?<NN|JJR|JJS|IN>+<VBG|TO|CD>?<NNS|CD|NN>?<NN>?}"
        cp = nltk.RegexpParser(grammar)
        result = cp.parse(word)
        print (result)
        result.draw()
```

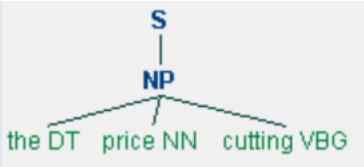
```
>>> npChunk("his mansion house speech")
```

(S (NP his/PRP$ mansion/NN house/NN speech/NN))



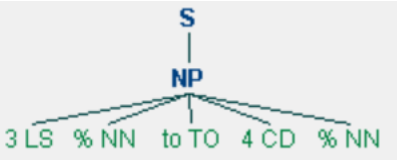
>>> npChunk("the price cutting")

(S (NP the/DT price/NN cutting/VBG))



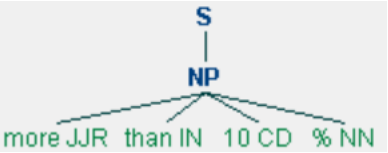
>>> npChunk("3% to 4%")

(S (NP 3/LS %/NN to/TO 4/CD %/NN))



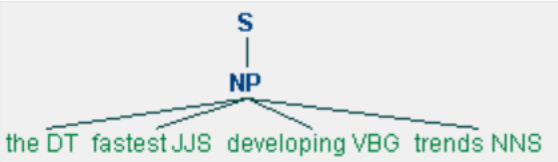
>>> npChunk("more than 10%")

(S (NP more/JJR than/IN 10/CD %/NN))



>>> npChunk("the fastest developing trends")

(S (NP the/DT fastest/JJS developing/VBG trends/NNS))



Exercise 5:

```
>>> def neTag(word):
        word = nltk.word_tokenize(word)
        word = nltk.pos_tag(word)
        print(nltk.ne_chunk(word, binary=True))
        print(nltk.ne_chunk(word))
```

```
>>> neTag("David Bascome started his First Division soccer career at the age of 16 with North Village Rams in Bermuda. At 20 years of age, he signed his first professional contract with the Harrisburg Heat of the National Professional Soccer League, USA.")
(S
  (NE David/NNP Bascome/NNP)
  started/VBD
  his/PRP$
  First/NNP
  Division/NNP
  soccer/NN
  career/NN
  at/IN
  the/DT
  age/NN
  of/IN
  16/CD
  with/IN
  (NE North/NNP Village/NNP Rams/NNP)
  in/IN
  Bermuda./NNP
  At/NNP
  20/CD
  years/NNS
  of/IN
  age/NN
  ,/,
  he/PRP
  signed/VBD
  his/PRP$
  first/JJ
  professional/JJ
```

contract/NN

with/IN

the/DT

(NE Harrisburg/NNP Heat/NNP)

of/IN

the/DT

(NE National/NNP Professional/NNP Soccer/NNP League/NNP)

,/,

(NE USA/NNP)

./.)

(S

(PERSON David/NNP)

(PERSON Bascome/NNP)

started/VBD

his/PRP$

First/NNP

Division/NNP

soccer/NN

career/NN

at/IN

the/DT

age/NN

of/IN

16/CD

with/IN

(PERSON North/NNP Village/NNP Rams/NNP)

in/IN

Bermuda./NNP

At/NNP

20/CD

years/NNS

of/IN

age/NN

,/,

he/PRP

signed/VBD

his/PRP$

first/JJ

professional/JJ

contract/NN

with/IN

the/DT

(ORGANIZATION Harrisburg/NNP Heat/NNP)

of/IN

the/DT

(ORGANIZATION National/NNP Professional/NNP Soccer/NNP League/NNP)

,/,

(ORGANIZATION USA/NNP)

./.)

>>> neTag("Avenida Caracas, sometimes called Carrera 14, is an arterial road in Bogotá, Colombia that runs through the city from north to south.")

(S

(NE Avenida/NNP Caracas/NNP)

,/,

sometimes/RB

called/VBN

Carrera/NNP

14/CD

,/,

is/VBZ

an/DT
arterial/JJ
road/NN
in/IN
Bogotá/NNP
,/,
(NE Colombia/NNP)
that/IN
runs/VBZ
through/IN
the/DT
city/NN
from/IN
north/JJ
to/TO
south/NN
./.)
(S
(PERSON Avenida/NNP)
(ORGANIZATION Caracas/NNP)
,/,
sometimes/RB
called/VBN
Carrera/NNP
14/CD
,/,
is/VBZ
an/DT
arterial/JJ
road/NN
in/IN

(GPE Bogotá/NNP)

,/,

(GPE Colombia/NNP)

that/IN

runs/VBZ

through/IN

the/DT

city/NN

from/IN

north/JJ

to/TO

south/NN

./.)

>>> neTag("Johann Ludwig Dammert (March 21, 1788 – January 25, 1855), was First Mayor and President of the Senate (head of state and head of government) of the sovereign city-state of Hamburg in 1843.")

(S

(NE Johann/NNP Ludwig/NNP Dammert/NNP)

(/NNP

March/NNP

21/CD

,/,

1788/CD

–/:

January/NNP

25/CD

,/,

1855/CD

)/:

,/,

was/VBD
First/NNP
Mayor/NNP
and/CC
President/NNP
of/IN
the/DT
(NE Senate/NNP)
(/NNP
head/NN
of/IN
state/NN
and/CC
head/NN
of/IN
government/NN
)/:
of/IN
the/DT
sovereign/NN
city-state/JJ
of/IN
(NE Hamburg/NNP)
in/IN
1843/CD
./.)
(S
(PERSON Johann/NNP)
(PERSON Ludwig/NNP Dammert/NNP)
(/NNP
March/NNP

21/CD

,/,

1788/CD

–/:

January/NNP

25/CD

,/,

1855/CD

)/:

,/,

was/VBD

(PERSON First/NNP Mayor/NNP)

and/CC

President/NNP

of/IN

the/DT

(ORGANIZATION Senate/NNP)

(/NNP

head/NN

of/IN

state/NN

and/CC

head/NN

of/IN

government/NN

)/:

of/IN

the/DT

sovereign/NN

city-state/JJ

of/IN

(GPE Hamburg/NNP)

in/IN

1843/CD

./.)

Exercise 6:

```
>>> def relExtract(word):
        word = nltk.word_tokenize(word)
        word = nltk.pos_tag(word)
        word = nltk.ne_chunk(word)
        IN = re.compile(r'.*\bin\b(?!\b.+ing)')
        for rel in nltk.sem.extract_rels('ORG', 'LOC', word,
                corpus='ace', pattern = IN):
                print nltk.sem.relextract.show_raw_rtuple(rel)
```

>>> relExtract("Mama Qucha, a figure of the Inca mythology, hispanicized spelling Mamacocha) is a lake in Peru.")

[ORG: ‘Mama Qucha’] ‘in’ [LOC: ‘Peru’]

>>> relExtract("Shilo Inns is a mid-priced hotel chain operating 43 hotels predominantly on the west coast of the United States, with a large concentration of locations in Oregon.")

[ORG: ‘Shilo Inns’] ‘with a large concentration of locations in’ [LOC: Oregon]

>>> relExtract("Przewalski's steppe lemming (Eolagurus przewalskii) is a species of rodents in the family Cricetidae. It is found in China.")

[ORG: ‘Cricetidae’] ‘in’ [LOC: ‘China’]

>>> relExtract("Wedgwood was born in Burslem")

[ORG: ‘Wedgwood’] ‘in’ [LOC: ‘Burslem’]

>>> relExtract("John Joseph McGee (August 6, 1845 – April 10, 1927) was Clerk of the Privy Council of Canada from May 20, 1882 to May 5, 1907 and is the longest-serving occupant of the position, born in Wexford")

[ORG: ‘Privy Council of Canada’] ‘in’ [LOC: ‘Wexford’]