```
###  Lab Week 3 - Regular Expressions and Tokenization   ###
# This file has small examples that are meant to be run individually
#   in the Python shell

import nltk

# get the book Emma from the Gutenberg collection and keep as raw text
file0 = nltk.corpus.gutenberg.fileids( ) [0]
emmatext = nltk.corpus.gutenberg.raw(file0)
print(type(emmatext))
print(len(emmatext))
# display the first 150 characters of the str emmatext
emmatext[:150]

# print the first 150 characters in the str emmatext as one string
print(emmatext[:150])


# print the first 20 characters in emmatext by iterating over the characters
for c in emmatext[:20]:
  print(c)

## Review of strings and string operations
# replace end-of-line character with a space
newemmatext = emmatext.replace('\n', ' ')
newemmatext[:150]


### Development of regular expressions for tokenizing text
import re
# pattern to match words, i.e. anything with a sequence of word characters, ignores special chars
shorttext = 'That book is interesting.'
pword = re.compile('\w+')
print(re.findall(pword, shorttext))

specialtext = 'That U.S.A. poster-print costs $12.40, but with 10% off.'
print(re.findall(pword, specialtext))

# pattern to match words with internal hyphens
ptoken = re.compile('(\w+(-\w+)*)')
print(re.findall(ptoken, specialtext))
print(re.findall(ptoken, 'end-of-line character'))

# ignore the group of the inner parentheses
ptoken = re.compile('(\w+(?:-\w+)*)')
print(re.findall(ptoken, specialtext))
print(re.findall(ptoken, 'end-of-line character'))


# abbreviations like U.S.A.
pabbrev = re.compile('((?:[A-Z]\.)+)')
print(re.findall(pabbrev, specialtext))

# combine this pattern with the words to make more general tokens
ptoken = re.compile('(\w+(?:-\w+)*|(?:[A-Z]\.)+)')
print(re.findall(ptoken, specialtext))

# switch the order of the patterns to first match abbreviations and then other words
ptoken = re.compile('((?:[A-Z]\.)+|\w+(?:-\w+)*)')
print(re.findall(ptoken, specialtext))

# add expression for currency
ptoken = re.compile('((?:[A-Z]\.)+|\w+(?:-\w+)*|\$?\d+(?:\.\d+)?)')
print(re.findall(ptoken, specialtext))

# this is an equivalent regular expression except that it has extra parentheses
ptoken = re.compile(r'''((?:[A-Z]\.)+) # abbreviations, e.g. U.S.A.
   | (\w+(?:-\w+)*) # words with internal hyphens
   | (\$?\d+(?:\.\d+)?) # currency, like $12.40
```

```
   ''', re.X) # verbose flag

print(re.findall(ptoken, specialtext))

## More about findall()
# using the findall() function to find 2 parts of each match
email_text = "For more information, send a request to info@ischool.syr.edu. Or you can directly contact our
information staff at HelpfulHenry@syr.edu and SageSue@syr.edu."

# re with two parentheses to match username and domain in every email address
pemail = re.compile('([a-zA-Z]+)@([a-z.]+)')
matches = re.findall(pemail, email_text)
for m in matches:
    # format function puts each argument into the output string where the {} is
    email = 'User: {}, Domain:{}'.format(m[0],m[1])
    print(email)


### using NLTK's regular expression tokenizer
# first define a multi-line string that is a regular expression
pattern = r''' (?x)     # set flag to allow verbose regexps
        (?:[A-Z]\.)+    # abbreviations, e.g. U.S.A.
        | \$?\d+(?:\.\d+)?%?    # currency and percentages, $12.40, 50%
        | \w+(?:-\w+)*  # words with internal hyphens
        | \.\.\.        # ellipsis
        | [][.,;"'?():-_%#']    # separate tokens
        '''

# the nltk regular expression tokenizer compiles the re pattern, applies it to the text
#  and uses the matching groups to return a list of only the matched tokens
print(nltk.regexp_tokenize(shorttext, pattern))
print(nltk.regexp_tokenize(specialtext, pattern))

# compare with built-in word tokenizer
print(nltk.word_tokenize(specialtext))

# Tokenizer for Twitter derived tweetmotif from the ARK, developed at CMU
tweetPattern = r''' (?x)        # set flag to allow verbose regexps
      (?:https?://|www)\S+      # simple URLs
      | (?::-\)|;-\))           # small list of emoticons
      | &(?:amp|lt|gt|quot);    # XML or HTML entity
      | \#\w+                 # hashtags
      | @\w+                  # mentions
      | \d+:\d+               # timelike pattern
      | \d+\.\d+              # number with a decimal
      | (?:\d+,)+?\d{3}(?=(?:[^,]|$))   # number with a comma
      | (?:[A-Z]\.)+                    # simple abbreviations
      | (?:--+)               # multiple dashes
      | \w+(?:-\w+)*          # words with internal hyphens or apostrophes
      | ['\".?!,:;/]+         # special characters
      '''

tweet1 = "@natalieohayre I agree #hc09 needs reform- but not by crooked politicians who r clueless about
healthcare! #tcot #fishy NO GOV'T TAKEOVER!"

tweet2 = "To Sen. Roland Burris: Affordable, quality health insurance can't wait http://bit.ly/j63je #hc09 #IL
#60660"

tweet3 = "RT @karoli: RT @Seriou: .@whitehouse I will stand w/ Obama on #healthcare,  I trust him. #p2 #tlot"

print(nltk.regexp_tokenize(tweet1,tweetPattern))
print(nltk.regexp_tokenize(tweet2,tweetPattern))
print(nltk.regexp_tokenize(tweet3,tweetPattern))

# NLTK built-in tokenizer (more detailed version from TweetMotif)
from nltk.tokenize import TweetTokenizer
ttokenizer = TweetTokenizer()
print(ttokenizer.tokenize(tweet1))

# sentence example for the question
```

```
sent = "Mr. Black and Mrs. Brown attended the lecture by Dr. Gray, but Gov. White wasn't there."
print(nltk.regexp_tokenize(sent, pattern))
```