МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Лабораторна робота № 5

ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ. ВИКОРИСТАННЯ РЕГУЛЯРНИХ ВИРАЗІВ ДЛЯ ОБРОБКИ ТЕКСТУ

Виконала:
студентка групи ПРЛс-11
Іваськів М.Є.

Прийняв:
Дупак Б.П.

Львів 2015

МЕТА РОБОТИ

Вивчення основ програмування на мові Python. Використання регулярних виразів для обробки текстів.

КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Синтаксис регулярних висловів залежить від інтерпретатора, що використовується для їх обробки. Пошук слів із закінченням ed можна здійснити використовуючи регулярний вираз «ed$». Потрібно використати функцію re.search(p, s), яка перевіряє чи може зразок p бути знайдений у будь-якому місці стрічки s. Потрібно визначити символи, які шукаємо та використати символ долара , який в регулярних виразах позначає кінець слова.

Символ ”.” універсальний символ , якому відповідає будь-який один символ. Нехай потрібно знайти слова з восьми літер, де j – третя літера та t –шоста літера. При створенні регулярного виразу у місцях де може бути будь-який символ вказується крапка. Символ ”^” вказує на початок стрічки.

Символ “ ? “ вказує на те що попередній символ не є обовязковим. Вираз «^e-?mail$» відповідає двом стрічкам email та e-mail. Можна знайти загальну кількість таких стрічок (врахувавши різні способи їх запису) у будь-якому тексті скориставшись sum(1 for w in text if re.search('^e-?mail$', w)).

Вираз re.search(regexp, w) дозволяє знаходити слова w , які відповідають регулярному виразу regexp . Регулярні вирази також можна використовувати для виявлення фрагментів слів, або для модифікації слів різними способами.

Метод re.findall() ("знайти все") дозволяє знайти всі відповідності даному регулярному виразу.

re.findall() може знаходити тільки суфікс, хоча регулярний вираз відповідає всьому слову. Це стається тому, що круглі дужки задають не тільки область дії оператора диз’юнкції але і виконують функцію вибору підстрічки яку потрібно вилучити. Коли потрібно в регулярному виразі використовувати круглі дужки для вказання області дії оператор,а але не потрібно здійснювати вилучення в регулярний вираз потрібно додати ?: ,.

Спеціальний тип регулярних виразів може використовуватися для пошуку серед слів у тексті (текст – послідовність окремих слів). Наприклад, за допомогою виразу "<a> <man>" можна знайти всі випадки вживання a man в тексті. Кутові дужки використовуються для позначення меж і всі пробіли мід цими дужками ігноруються (індивідуальна особливість NLTK's findall() методу для тексту). В наступному прикладі включено <.*> #1 для виявлення всіх окремих слів, а круглі дужки дозволять вибрати ці слова окремо від словосполучень (a monied man).

Для полегшення розробки токенізаторів існує можливість доступу до текстів, які токенізовані вручну. Порівнюючи результати токенізації з таким «gold-standard» токенізатором можна оцінити якість роботи програми. Набір корпусівNLTK включає приклади Penn Treebank , а саме тексти Wall Street Journal (`nltk.corpus.treebank_raw.raw()`) та їх токенізовану версію (`nltk.corpus.treebank.words()`).

ВИКОНАННЯ ЗАВДАНЬ

Варіант 4.

Завдання 1. Описати, які класи стрічок відповідають наступному регулярному виразу. [a-zA-Z]+. Результати перевірити використовуючи nltk.re_show().

Програма виводить усі слова, у яких є лише малі або великі літери (розділові знаки або цифри відсутні).

```
from __future__ import division
import nltk, re, pprint
wordlist = [w for w in nltk.corpus.words.words('en')]
str =[w for w in wordlist[:10] if re.search('[a-zA-Z]+', w)]
print str[:10]
show = [w for w in wordlist[:10] if nltk.re_show('[a-zA-Z]+', w)]
print show[:10]
>>>
['A', 'a', 'aa', 'aal', 'aalii', 'aam', 'Aani', 'aardvark', 'aardwolf', 'Aaron']
{A}
{a}
{aa}
{aal}
{aalii}
{aam}
{Aani}
{aardvark}
{aardwolf}
{Aaron}
[]
```

Завдання 2. Описати, які класи стрічок відповідають наступному регулярному виразу. [A-Z][a-z]*. Результати перевірити використовуючи nltk.re_show().

Програма знаходить слова, що починаються з великої букви.

```
from __future__ import division
import nltk, re, pprint
wordlist = [w for w in nltk.corpus.words.words('en')]
str =[w for w in wordlist[:20] if re.search('[A-Z][a-z]*', w)]
print str
show = [w for w in wordlist[:20] if nltk.re_show('[A-Z][a-z]*', w)]
print show
>>>
['A', 'Aani', 'Aaron', 'Aaronic', 'Aaronical', 'Aaronite', 'Aaronitic', 'Aaru',
'Ab', 'Ababdeh', 'Ababua']
{A}
a
aa
aal
aalii
aam
{Aani}
aardvark
aardwolf
{Aaron}
{Aaronic}
{Aaronical}
{Aaronite}
{Aaronitic}
{Aaru}
{Ab}
aba
{Ababdeh}
{Ababua}
abac
[]
```

Завдання 3. Описати, які класи стрічок відповідають наступному регулярному виразу. \d+(\.\d+)?. Результати перевірити використовуючи nltk.re_show()

Програма шукає слова, які мають у своєму складі числа, та десяткові числа.

```
import nltk, re, pprint
mylist = ['55', '33 cats', 'g2g', 'l8ter', 'hello','u2']
words = [w for w in mylist if re.search('\d+(\.\d+)?', w)]
print words
show = [w for w in mylist if nltk.re_show('\d+(\.\d+)?', w)]
print show
>>>
['55', '33 cats', 'g2g', 'l8ter', 'u2']
{55}
{33} cats
g{2}g
l{8}ter
hello
u{2}
[]
```

Завдання 4. Описати, які класи стрічок відповідають наступному регулярному виразу. ([^aeiou][aeiou][^aeiou])*. Результати перевірити використовуючи nltk.re_show()

Програма шукає слова за шаблоном «не голосна-голосна-не голосна».

```
from __future__ import division
import nltk, re, pprint
wordlist = [w for w in nltk.corpus.words.words('en') if w.islower()]
str =[w for w in wordlist[:10] if re.search('([^aeiou][aeiou][^aeiou])*', w)]
print str[:10]
show = [w for w in wordlist[:10] if nltk.re_show('([^aeiou][aeiou][^aeiou])*', w
print show[:10]
>>>
['a', 'aa', 'aal', 'aalii', 'aam', 'aardvark', 'aardwolf', 'aba', 'abac', 'abaca
']
{}a{}
{}a{}a{}
{}a{}a{}l{}
{}a{}a{}l{}i{}i{}
{}a{}a{}m{}
{}a{}a{}r{}d{var}k{}
{}a{}a{}r{}d{wol}f{}
{}a{}b{}a{}
{}a{bac}
{}a{bac}a{}
[]
```

Завдання 5. Описати, які класи стрічок відповідають наступному регулярному виразу. \w+|[^\w\s]+.. Результати перевірити використовуючи nltk.re_show()

Програма показує усі цифри, сим лови та слова.

```
from __future__ import division
import nltk, re, pprint
wordlist = [w for w in nltk.corpus.words.words('en') if w.islower()]
str =[w for w in wordlist[:10] if re.search('\w+|[^\w\s]+', w)]
print str[5:10]
show = [w for w in wordlist[:10] if nltk.re_show('\w+|[^\w\s]+', w)]
print show[5:10]
>>>
['aardvark', 'aardwolf', 'aba', 'abac', 'abaca']
{a}
{aa}
{aal}
{aalii}
{aam}
{aardvark}
{aardwolf}
{aba}
{abac}
{abaca}
[]
```

Завдання 6. Описати, які класи стрічок відповідають наступному регулярному виразу. p[aeiou]{,2}t Результати перевірити використовуючи nltk.re_show()

Програма показує слова, у яких між буквами p та t можуть бути 0, 1 або 2 голосні.

```
from __future__ import division
import nltk, re, pprint
wordlist = [w for w in nltk.corpus.words.words('en') if w.islower()]
str =[w for w in wordlist if re.search('p[aeiou]{,2}t', w)]
print str[:15]
show = [w for w in wordlist[6550:6570] if nltk.re_show('p[aeiou]{,2}t', w)]
print show
```

```
['abaptiston', 'abepithymia', 'ableptical', 'ableptically', 'abrupt', 'abruptedl
y', 'abruption', 'abruptly', 'abruptness', 'absorpt', 'absorptance', 'absorptiom
eter', 'absorptiometric', 'absorption', 'absorptive']
anaplastic
anaplasty
anaplerosis
anaplerotic
anapnea
anapneic
anapnoeic
anapnograph
anapnoic
anapnometer
anapodeictic
anapophysial
anapophysis
anapsid
anapsidan
ana{pt}erygote
ana{pt}erygotism
ana{pt}erygotous
ana{pt}otic
ana{pt}ychus
```

Завдання 7. Написати регулярний вираз, який встановлює відповідність наступному класу стрічок: всі артиклі *(a, an, the*).

```
import nltk, re, pprint
from nltk.corpus import brown
wordlist = nltk.Text(brown.words(categories='mystery'))
article = wordlist.findall(r'<a>|<an>|<the>')
print article
>>>
```

```
the; the; a; the; the; the; the; a; the; a; the; the; the; the; the;
a; the; a; the; a; a; an; the; a; the; a; a; a; the; the; a; a; the;
a; a; a; an; a; the; a; an; a; a; a; a; the; a; a; a; a; the; a; an;
the; the; the; a; the; a; the; the; the; a; the; an; the; a; the; the;
the; the; the; the; the; the; the; a; a; a; a; the; a; the; a; a; a;
the; the; the; a; a; the; the; the; the; a; a; an; the; a; the;
the; the; the; a; a; the; a; the; the; the; the; the; the; the; the;
the; a; a; the; the; a; the; the; the; the; the; the; the; the; a;
the; the; the; the; the; the; the; the; the; a; the; the; the; a; the;
a; the; the; the; an; the; a; the; the; the; a; the; the; the; the;
the; a; a; an; the; an; the; a; a; the; an; the; the; a; the; the; a;
a; a; a; the; the; the; a; the; a; the; the; a; the; the; a; the; the;
the; the; a; a; the; the; the; a; the; the; the; the; the; the; the;
```

Завдання 9. Зберегти довільний текст у файлі corpus.txt. Визначити функцію для читання з цього файлу (назва файлу аргумент функції) і повертає стрічку, яка містить текст з файлу. Використовуючи nltk.regexp_tokenize() розробити токенізатор для токенізації різних типів пунктуації в цьому тексті. Використовувати багаторядковий запис регулярного виразу з коментарями та «verbose flag»

```
from __future__ import division
import nltk, re, pprint
def load(f):
    f = open(f)
    raw = f.read()
    return raw
```

```
>>> load("E:\Моє\Універ\Комп'ютерна лінгвістика\Python 2.7\corpus.txt")
"Egypt's prime minister said a technical fault was the most likely cause, dismis
sing claims from Islamic State militants that they were responsible.\nHowever, t
hree airlines - Emirates, Air France and Lufthansa - have decided not to fly ove
r the Sinai Peninsula until more information is available.\nRussia is observing
a day of mourning after its worst air disaster.\nThe plane's black boxes have be
en found and sent for analysis, officials said.\nThe BBC's Sally Nabil in Cairo
says the crash has been a major blow to Egypt's already struggling tourism indus
try, and the Egyptian authorities are trying very hard to accelerate the investi
gation process."
>>> text = "Egypt's prime minister said a technical fault was the most likely ca
use, dismissing claims from Islamic State militants that they were responsible.\
nHowever, three airlines - Emirates, Air France and Lufthansa - have decided not
 to fly over the Sinai Peninsula until more information is available.\nRussia is
 observing a day of mourning after its worst air disaster.\nThe plane's black bo
xes have been found and sent for analysis, officials said.\nThe BBC's Sally Nabi
l in Cairo says the crash has been a major blow to Egypt's already struggling to
urism industry, and the Egyptian authorities are trying very hard to accelerate
the investigation process."
>>> pattern = r'''(?x)          #regular expresion
            ([A-Z]\.)+          #abbreviations
            |\w+(-\w+)*         #words with hyphens (optional)
            |\$?\d+(\/\d+)?%?   #currency, percents
            |\.\.\.             #ellipsis
            |[][.,;"'?():-_]    #tokens
          '''
>>> nltk.regexp_tokenize(text, pattern)
['Egypt', "'", 's', 'prime', 'minister', 'said', 'a', 'technical', 'fault', 'was
', 'the', 'most', 'likely', 'cause', ',', 'dismissing', 'claims', 'from', 'Islam
ic', 'State', 'militants', 'that', 'they', 'were', 'responsible', '.', 'However'
, ',', 'three', 'airlines', 'Emirates', ',', 'Air', 'France', 'and', 'Lufthansa'
, 'have', 'decided', 'not', 'to', 'fly', 'over', 'the', 'Sinai', 'Peninsula', 'u
ntil', 'more', 'information', 'is', 'available', '.', 'Russia', 'is', 'observing
', 'a', 'day', 'of', 'mourning', 'after', 'its', 'worst', 'air', 'disaster', '.'
, 'The', 'plane', "'", 's', 'black', 'boxes', 'have', 'been', 'found', 'and', 's
ent', 'for', 'analysis', ',', 'officials', 'said', '.', 'The', 'BBC', "'", 's',
'Sally', 'Nabil', 'in', 'Cairo', 'says', 'the', 'crash', 'has', 'been', 'a', 'ma
jor', 'blow', 'to', 'Egypt', "'", 's', 'already', 'struggling', 'tourism', 'indu
stry', ',', 'and', 'the', 'Egyptian', 'authorities', 'are', 'trying', 'very', 'h
ard', 'to', 'accelerate', 'the', 'investigation', 'process', '.']
>>>
```

Завдання 11. Написати функцію unknown(), яка приймає інтернет адресу як аргумент і повертає не відомі слова, які зустрічаються в тексті. При розробці функції використовувати re.findall() для виявлення всіх підстрічок та корпус Words Corpus (nltk.corpus.words) для виявлення не відомих слів.

```
from __future__ import division
import nltk, re, pprint
import urllib
from urllib import urlopen
def unknown(url):
    web = urlopen(url).read()
    raw = nltk.clean_html(web)

    uknownw=nltk.corpus.words.words()
    if re.findall(r'[A-Za-z]*', raw) not in uknownw:
        return re.findall(r'[A-Za-z]*', raw)
>>> unknown("http://www.bbc.com/news/world-middle-east-34690936")
['Sinai', '', 'plane', '', 'crash', '', '', 'Egypt', '', 'dismisses', '', 'IS',
'', 'claim', '', '', '', 'BBC', '', 'News', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'Accessibility', '', 'links', '', '', 'Skip', '', 'to', '', 'content', '', '', '
Accessibility', '', 'Help', '', '', '', '', '', '', 'BBC', '', 'iD', '', '', '',
 '', '', '', '', '', '', 'BBC', '', 'navigation', '', '', '', 'News', '', '', ''
, 'News', '', '', '', 'Sport', '', '', '', 'Weather', '', '', '', 'Shop', '', ''
```

Завдання 15. Прочитати Додаток А. Дослідити явища описані у Додатку А використовуючи корпуси текстів та метод findall()для пошуку в токенізованому тексті.

```
import nltk, re
from nltk.corpus import brown
text = brown.raw()
words = text.split()
i = 0
slova = []
for word in words:
    i = word.find('/')
    word = word[0:i]
    slova.append(word)
text = nltk.Text(slova)
print 'as best __ can: ',
text.findall(r'<as best>(<.*>)<can>')
print 'as best as __ can: ',
text.findall(r'<as best as>(<.*>)<can>')
print 'the best __ can: ',
text.findall(r'<the best>(<.*>)<can>')
print 'the best that __ can: ',
text.findall(r'<the best that>(<.*>)<can>')
print 'best __ can: ',
text.findall(r'<best>(<.*>)<can>')
>>>
as best __ can:
as best as __ can:
the best __ can:
the best that __ can:
best __ can:  they
```