

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Кафедра САПР

ЗВІТ

до лабораторної роботи № 6

на тему:

ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ
ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ.

ВИКОРИСТАННЯ РЕГУЛЯРНИХ ВИРАЗІВ ДЛЯ ОБРОБКИ ТЕКСТУ
з дисципліни “Комп’ютерна лінгвістика”

Виконала:

Студентка групи ПРЛм-12

Рибчак Х. В.

Перевірив:

Асистент кафедри САПР

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 обов'язковим. 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(regex, w)` дозволяє знаходити слова `w`, які відповідають регулярному виразу `regex`. 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 операторів, а але не потрібно здійснювати вилучення в регулярний вираз потрібно додати `?: , .`

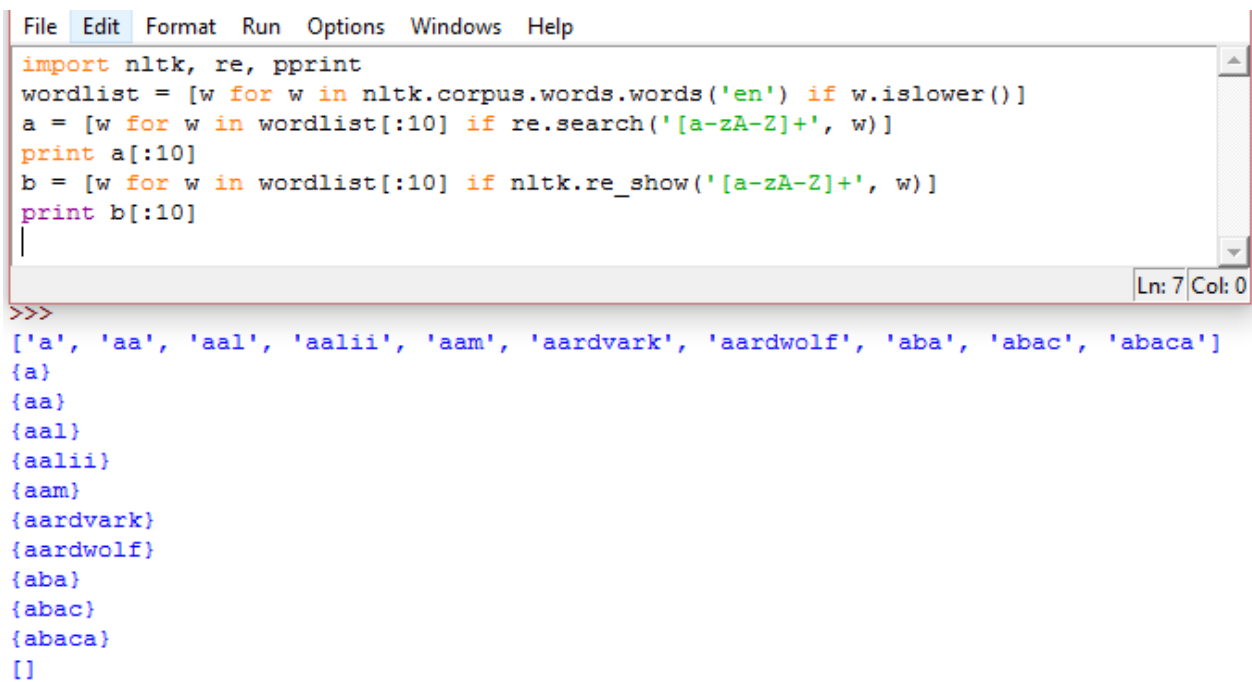
Спеціальний тип регулярних виразів може використовуватися для пошуку серед слів у тексті (текст – послідовність окремих слів). Наприклад, за допомогою виразу `"<a> <man>"` можна знайти всі випадки вживання `a` та `man` в тексті. Кутові дужки використовуються для позначення меж і всі пробіли між цими 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`).

ТЕКСТИ ПРОГРАМ НА МОВІ PYTHON

ВАРІАНТ №8

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

Програма знаходить всі слова, які містять малі або великі літери, але не містять цифр чи розділових знаків.



```
File Edit Format Run Options Windows Help
import nltk, re, pprint
wordlist = [w for w in nltk.corpus.words.words('en') if w.islower()]
a = [w for w in wordlist[:10] if re.search('[a-zA-Z]^+', w)]
print a[:10]
b = [w for w in wordlist[:10] if nltk.re_show('[a-zA-Z]^+', w)]
print b[:10]
|

>>>
['a', 'aa', 'aal', 'aalii', 'aam', 'aardvark', 'aardwolf', 'aba', 'abac', 'abaca']
{a}
{aa}
{aal}
{aalii}
{aam}
{aardvark}
{aardwolf}
{aba}
{abac}
{abaca}
[]
```

Рис. 1. Текст програми №1.

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

Програма знаходить всі слова, які починаються з великої літери.

```
2.py - D:/5 курс/Комп'ютерна лінгвістика/Звіти/Лаба6/2.py
File Edit Format Run Options Windows Help

import nltk, re, pprint
wordlist = [w for w in nltk.corpus.words.words('en')]
a = [w for w in wordlist[:15] if re.search('[A-Z][a-z]*', w)]
print a
b = [w for w in wordlist[:15] if nltk.re_show('[A-Z][a-z]*', w)]
print b

>>>
['A', 'Aani', 'Aaron', 'Aaronic', 'Aaronical', 'Aaronite', 'Aaronitic', 'Aaru']
{A}
a
aa
aal
aalii
aam
{Aani}
aardvark
aardwolf
{Aaron}
{Aaronic}
{Aaronical}
{Aaronite}
{Aaronitic}
{Aaru}
[]
~>>>
```

Рис. 2. Текст програми №2.

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

Програма шукає слова, що містять послідовність цифр, десяткові числа.

```
3.py - D:/5 курс/Комп'ютерна лінгвістика/Звіти/Лаба6/3.py
File Edit Format Run Options Windows Help

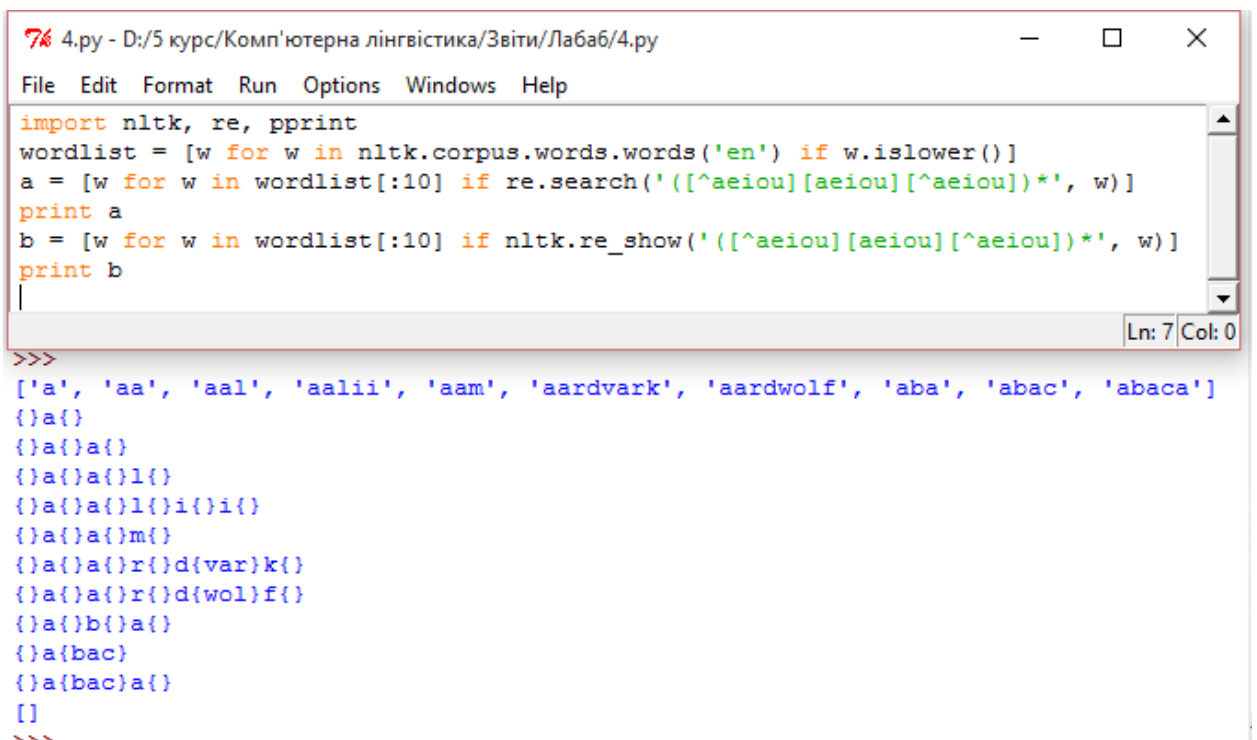
import nltk, re, pprint
lst = ['10', 'cat', '10.5', '0.5', 'a2', 'friend', 'you 2']
a = [w for w in lst if re.search('\d+(\.\d+)?', w)]
print a
b = [w for w in lst if nltk.re_show('\d+(\.\d+)?', w)]
print b

>>>
['10', '10.5', '0.5', 'a2', 'you 2']
{10}
cat
{10.5}
{0.5}
a{2}
friend
you {2}
[]
>>>
```

Рис. 3. Текст програми №3.

4. Описати, які класи стрічок відповідають наступному регулярному виразу. $([\text{aeiou}][\text{aeiou}][\text{aeiou}])^*$. Результати перевірити використовуючи `nlk.re_show()`.

Програма шукає слова, що містять послідовність символів «не голосна — голосна — не голосна». Завдяки методу `nlk.re_show()` зрозуміло, що програма повертає всі слова через те, що така послідовність може зустрічатися 0 або більше разів (*).



```
74 4.py - D:/5 курс/Комп'ютерна лінгвістика/Звіти/Лаба6/4.py
File Edit Format Run Options Windows Help

import nltk, re, pprint
wordlist = [w for w in nltk.corpus.words.words('en') if w.islower()]
a = [w for w in wordlist[:10] if re.search('([\text{aeiou}][\text{aeiou}][\text{aeiou}])^*', w)]
print a
b = [w for w in wordlist[:10] if nlk.re_show('([\text{aeiou}][\text{aeiou}][\text{aeiou}])^*', w)]
print b
|

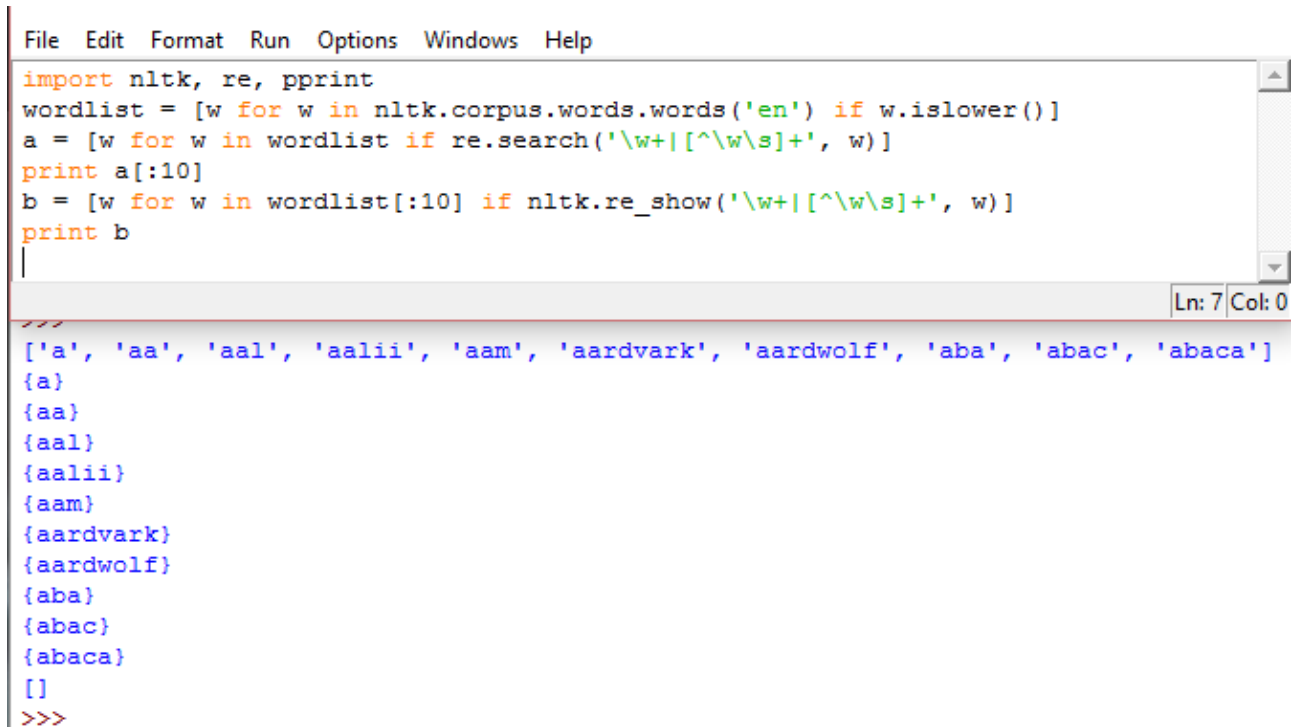
Ln: 7 Col: 0

>>>
['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{}
[]
~~~
```

Рис. 4. Текст програми №4.

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

Програма повертає всі слова, символи та цифри.



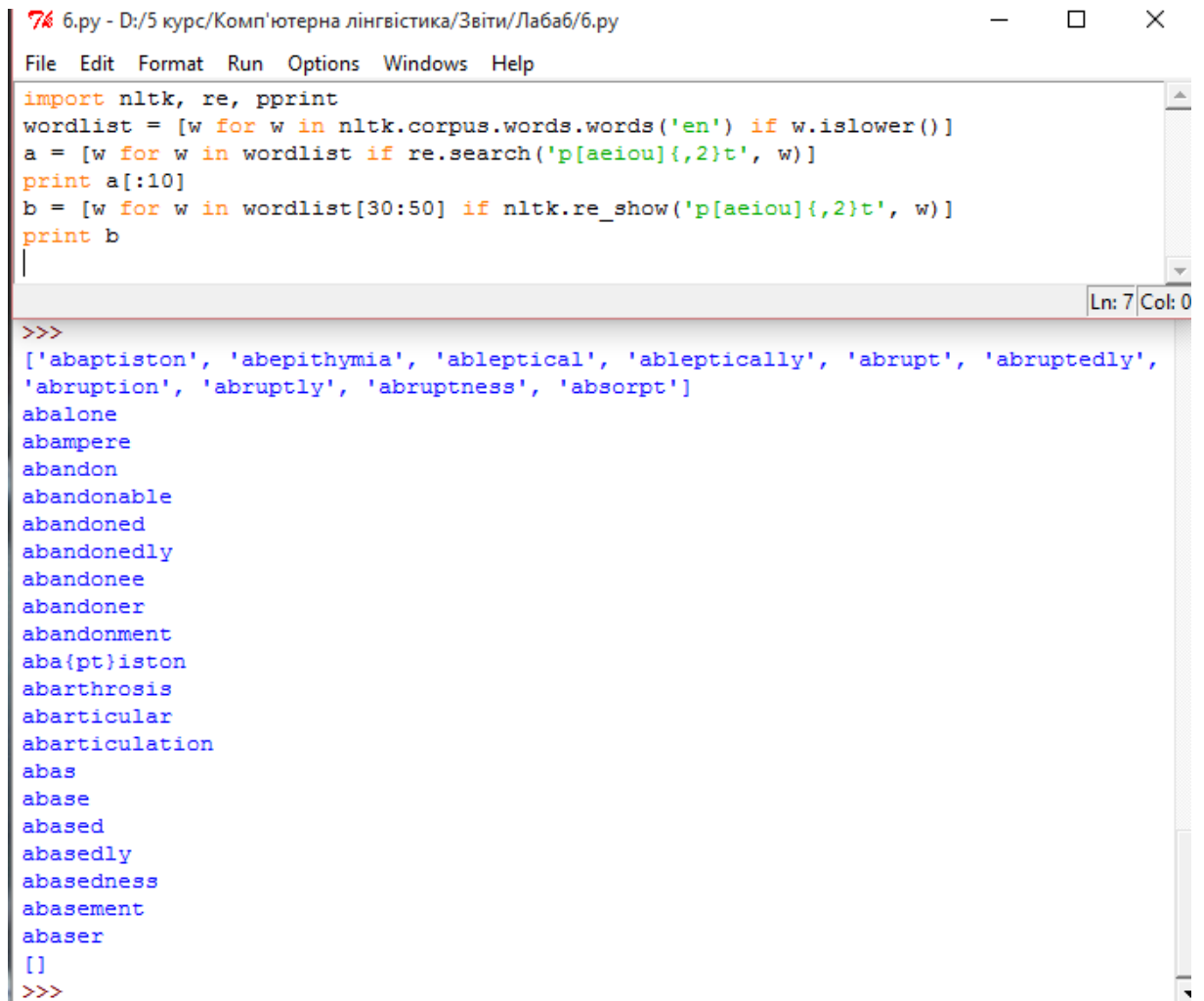
```
File Edit Format Run Options Windows Help
import nltk, re, pprint
wordlist = [w for w in nltk.corpus.words.words('en') if w.islower()]
a = [w for w in wordlist if re.search('\w+|[\^\w\s]+', w)]
print a[:10]
b = [w for w in wordlist[:10] if nltk.re_show('\w+|[\^\w\s]+', w)]
print b
|
Ln: 7 Col: 0

['a', 'aa', 'aal', 'aalii', 'aam', 'aardvark', 'aardwolf', 'aba', 'abac', 'abaca']
{a}
{aa}
{aal}
{aalii}
{aam}
{aardvark}
{aardwolf}
{aba}
{abac}
{abaca}
[]
>>>
```

Рис. 5. Текст програми №5.

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

Програма повертає слова, які містять послідовність літер 'p' та 't', між якими може бути 0, 1 або 2 голосні.



```
7% 6.py - D:/5 курс/Комп'ютерна лінгвістика/Звіти/Лаба6/6.py
File Edit Format Run Options Windows Help

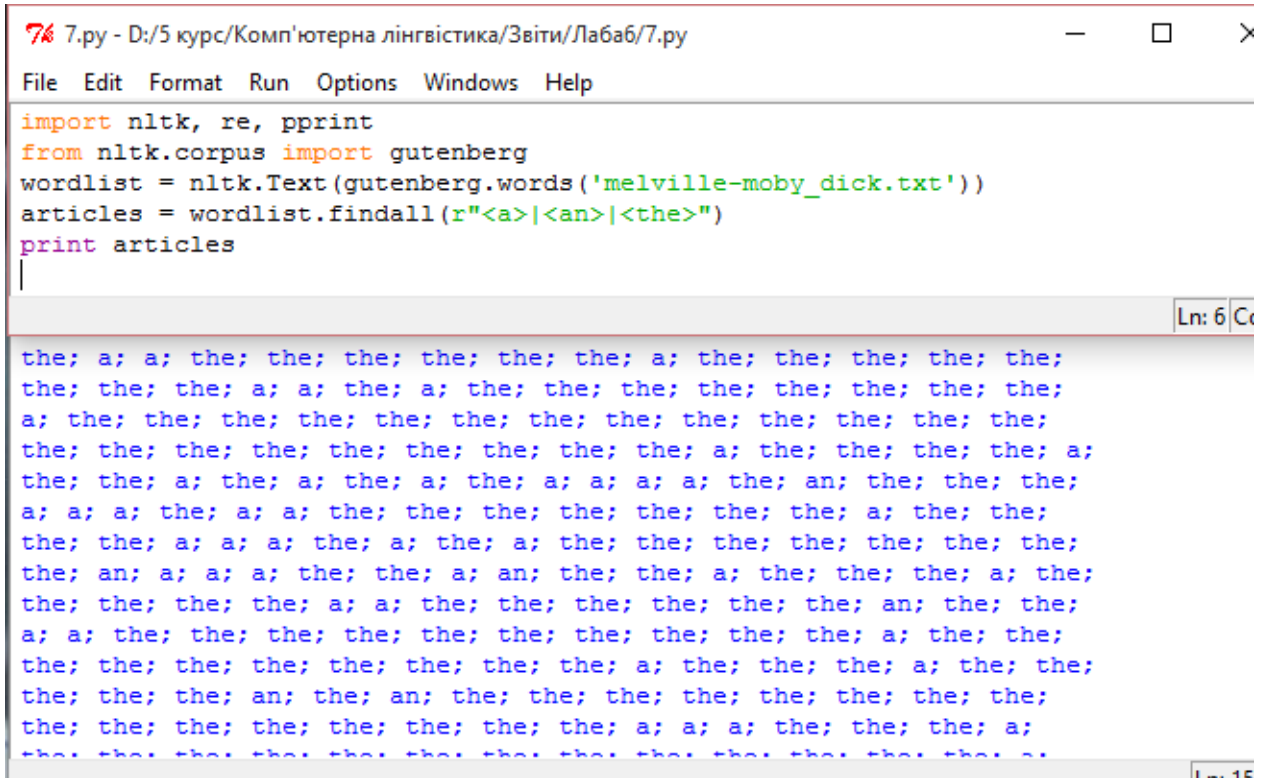
import nltk, re, pprint
wordlist = [w for w in nltk.corpus.words.words('en') if w.islower()]
a = [w for w in wordlist if re.search('p[aeiou]{,2}t', w)]
print a[:10]
b = [w for w in wordlist[30:50] if nltk.re_show('p[aeiou]{,2}t', w)]
print b
|

Ln: 7 Col: 0

>>>
['abaptiston', 'abepithymia', 'ableptical', 'ableptically', 'abrupt', 'abruptedly',
'abruption', 'abruptly', 'abruptness', 'absorpt']
abalone
abampere
abandon
abandonable
abandoned
abandonedly
abandonee
abandoner
abandonment
aba{pt}iston
abarthrosis
abarticular
abarticulation
abas
abase
abased
abasedly
abasedness
abasement
abaser
[]
>>>
```

Рис. 6. Текст програми №6.

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



```
7.py - D:/5 курс/Комп'ютерна лінгвістика/Звіти/Лаба6/7.py
File Edit Format Run Options Windows Help

import nltk, re, pprint
from nltk.corpus import gutenberg
wordlist = nltk.Text(gutenberg.words('melville-moby_dick.txt'))
articles = wordlist.findall(r"<a>|<an>|<the>")
print articles
```

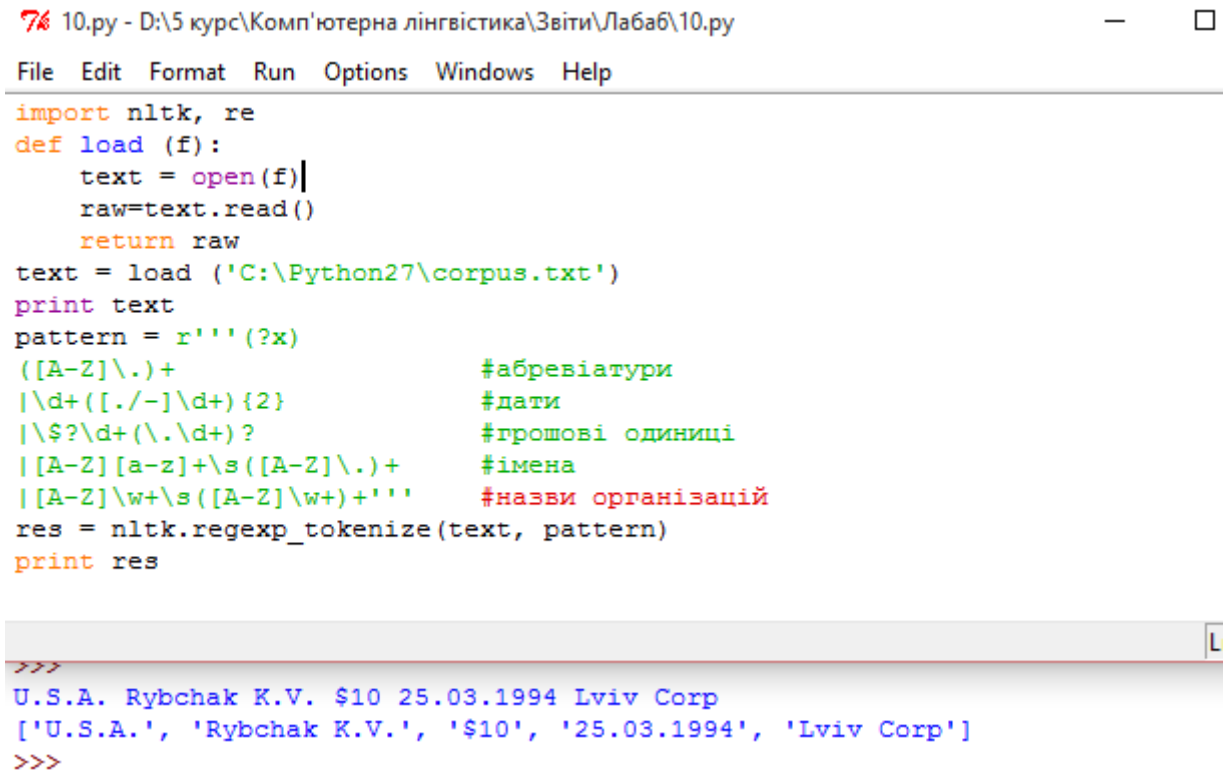
Ln: 6 Co

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

Ln: 15 Co

Рис. 7. Текст програми №7.

10. 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.regex_tokenize() розробити токенизатор для токенизації різних типів виразів: грошові одиниці, дати, імена людей та організацій. Використовувати багаторядковий запис регулярного виразу з коментарями та «verbose flag».




```
7% 10.py - D:\5 курс\Комп'ютерна лінгвістика\Звіти\Лаба6\10.py
File Edit Format Run Options Windows Help
import nltk, re
def load (f):
    text = open(f)
    raw=text.read()
    return raw
text = load ('C:\Python27\corpus.txt')
print text
pattern = r'''(?x)
([A-Z]\.)+           #аббревіатури
|\d+([\./-]\d+){2}    #дати
|\$?\d+(\.\d+)?       #грошові одиниці
|[A-Z][a-z]+\s([A-Z]\.)+ #імена
|[A-Z]\w+\s([A-Z]\w+)+''' #назви організацій
res = nltk.regex_tokenize(text, pattern)
print res

>>>
U.S.A. Rybchak K.V. $10 25.03.1994 Lviv Corp
['U.S.A.', 'Rybchak K.V.', '$10', '25.03.1994', 'Lviv Corp']
>>>
```

Рис. 8. Текст програми №10.

13. Напишіть програму, яка конвертує текст в Pig Latin. String->ingstray, idle->idleay. (Конвертація відбувається переміщенням приголосної або групи приголосних на кінець слова та додаванням до слова 'ay').



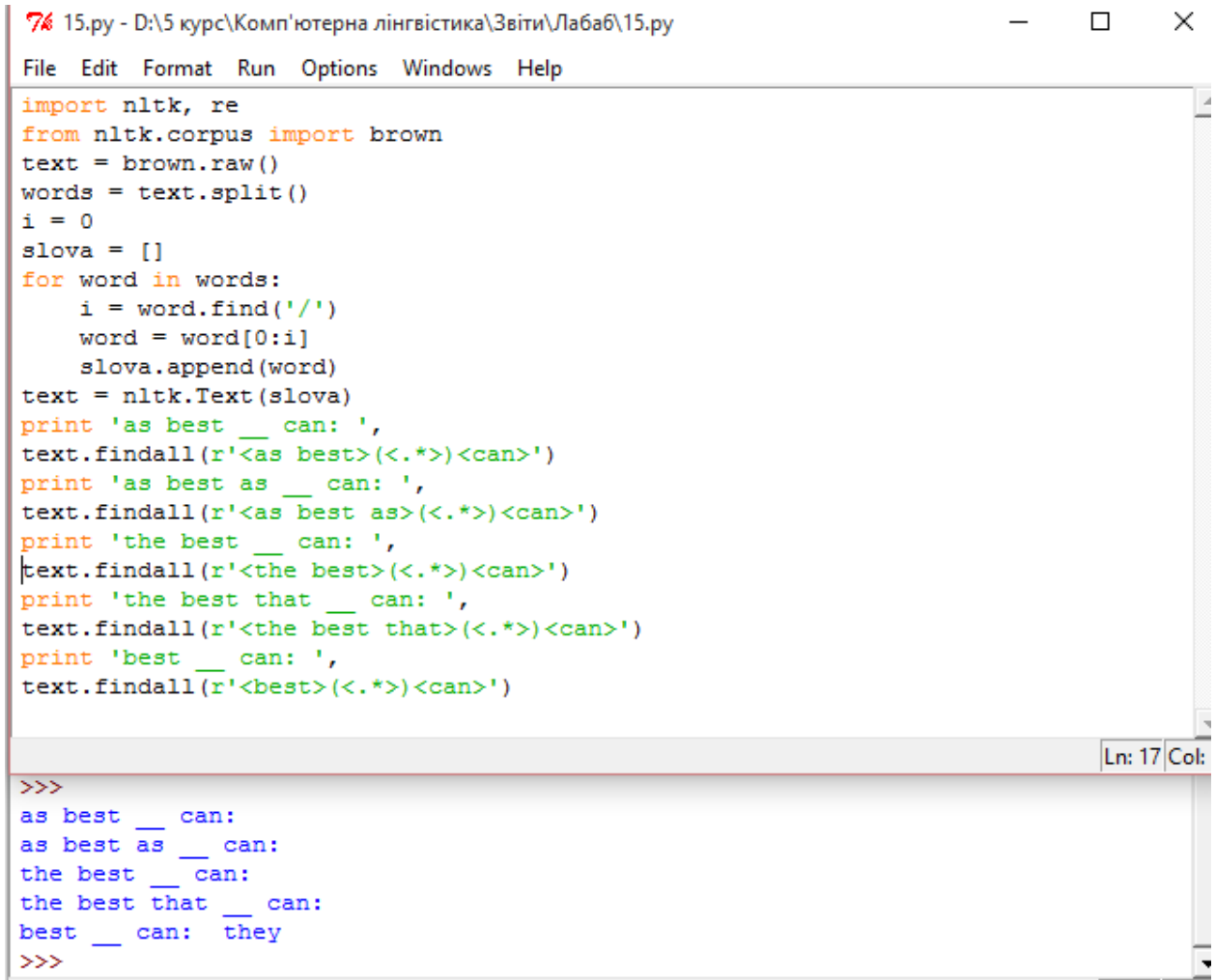
```
7% 13.py - D:\5 курс\Комп'ютерна лінгвістика\Звіти\Лаба6\13.py
File Edit Format Run Options Windows Help

import nltk, re
from nltk.corpus import gutenberg
def latin(w):
    piece = re.findall(regex, w)
    pos = len(piece[0])
    new = w[pos:] + piece[0] + 'ay'
    return new
raw = gutenberg.raw('austen-sense.txt')
text = re.split(r'[\W\d]+', raw)
print text[:20]
regex = r'^[AEIOUaeiou]*'
pig_latin = []
for word in text:
    pig_latin.append(latin(word))
print pig_latin[:20]
```

```
>>>
['', 'Sense', 'and', 'Sensibility', 'by', 'Jane', 'Austen', 'CHAPTER', 'The', 'f
amily', 'of', 'Dashwood', 'had', 'long', 'been', 'settled', 'in', 'Sussex', 'The
ir', 'estate']
['ay', 'enseSay', 'anday', 'ensibilitySay', 'byay', 'aneJay', 'Austenay', 'APTER
CHay', 'eThay', 'amilyfay', 'ofay', 'ashwoodDay', 'adhay', 'onglay', 'eenbay', '
ettledsay', 'inay', 'ussexSay', 'eirThay', 'estateay']
```

Рис. 9. Текст програми №13.

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



```
15.py - D:\5 курс\Комп'ютерна лінгвістика\Звіти\Лаба6\15.py
File Edit Format Run Options Windows Help

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
>>>
```

Рис. 10. Текст програми №15.

ВИСНОВОК

У цій лабораторній роботі я вивчила основи програмування на Python а саме використання регулярних виразів для обробки текстів.