МІНІСТЕРСТВО ОСВІТИ, НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Звіт
до лабораторної роботи №11 на тему:

«ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ. АВТОМАТИЧНИЙ СИНТАКСИЧНИЙ АНАЛІЗ (частина2)»

Виконала:
ст. групи ПРЛм-11
ІКНІ
Неїжмак О.А.
Прийняв:
Асистент кафедри САПР
Дупак Б. П.

Львів – 2015

МЕТА РОБОТА

- Вивчення основ програмування на мові *Python*.
- Ознайомлення з автоматичним синтаксичним аналізом в NLTK.

Варіант №10

2. В класі Tree реалізовано різноманітні корисні методи. Переглянути файл допомоги Tree з документації та описати основні з цих методів (import Tree, help(Tree).

Викликавши файл допомоги Tree, видно, що основним методом є chomsky_normal_form, оскільки його інтерпретації зустрічаються частіше, ніж інших методів.

```
>>> import nltk
>>> from nltk import Tree
>>> help(Tree)
```

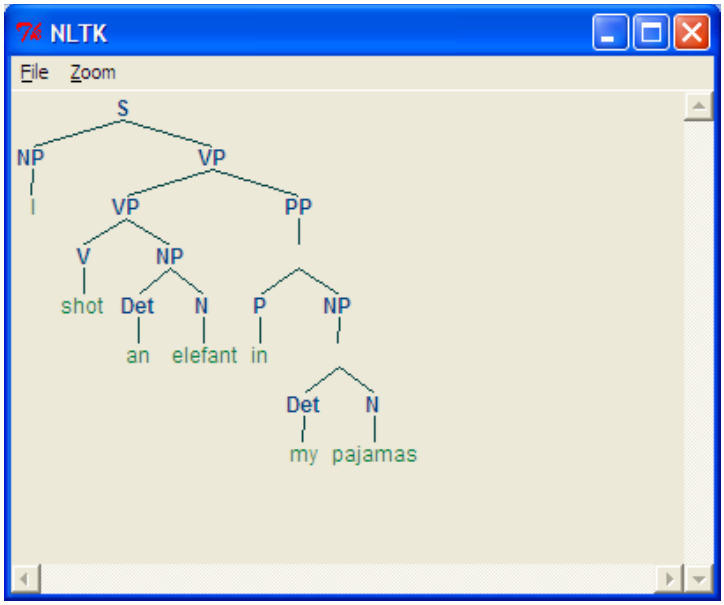
```
 |
 |  append(...)
 |      L.append(object) -- append object to end
 |
 |  count(...)
 |      L.count(value) -> integer -- return number of occurrences of value
 |
 |  extend(...)
 |      L.extend(iterable) -- extend list by appending elements from the iterabl
e
 |
 |  index(...)
 |      L.index(value, [start, [stop]]) -> integer -- return first index of valu
e.
 |      Raises ValueError if the value is not present.
 |
 |  insert(...)
 |      L.insert(index, object) -- insert object before index
 |
 |  pop(...)
 |      L.pop([index]) -> item -- remove and return item at index (default last)
.
 |      Raises IndexError if list is empty or index is out of range.
 |
 |  remove(...)
 |      L.remove(value) -- remove first occurrence of value.
 |      Raises ValueError if the value is not present.
 |
 |  reverse(...)
 |      L.reverse() -- reverse *IN PLACE*
 |
 |  sort(...)
 |      L.sort(cmp=None, key=None, reverse=False) -- stable sort *IN PLACE*;
 |      cmp(x, y) -> -1, 0, 1
 |
 |  ----------------------------------------------------------------------
```

4. Перетворити всі дерева, які зустрічаються в методичних вказівках зображені за допомогою дужок використовуючи nltk.Tree() .

Використовувати draw() для побудови графічного зображення дерева.

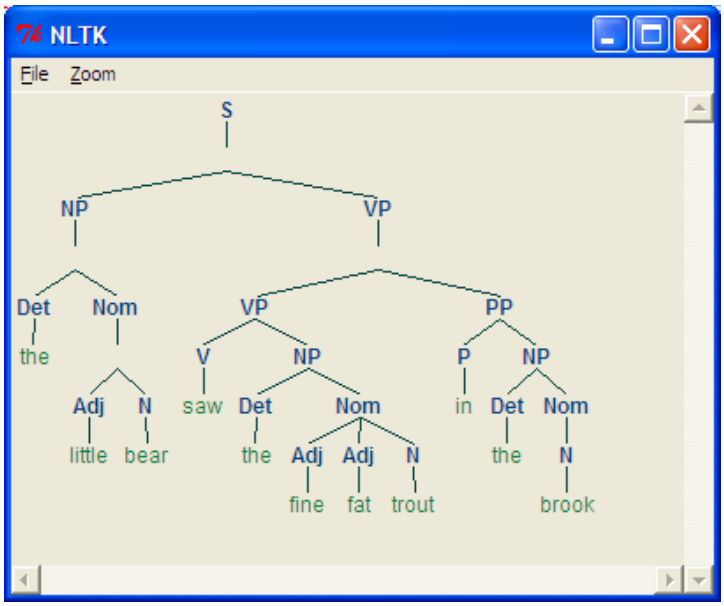
>>> t=nltk.Tree('(S(NP I)(VP(VP(V shot) (NP(Detan) (N elefant))) (PP((P in) (NP((Detmy) (N pajamas)))))))')

>>>t.draw()



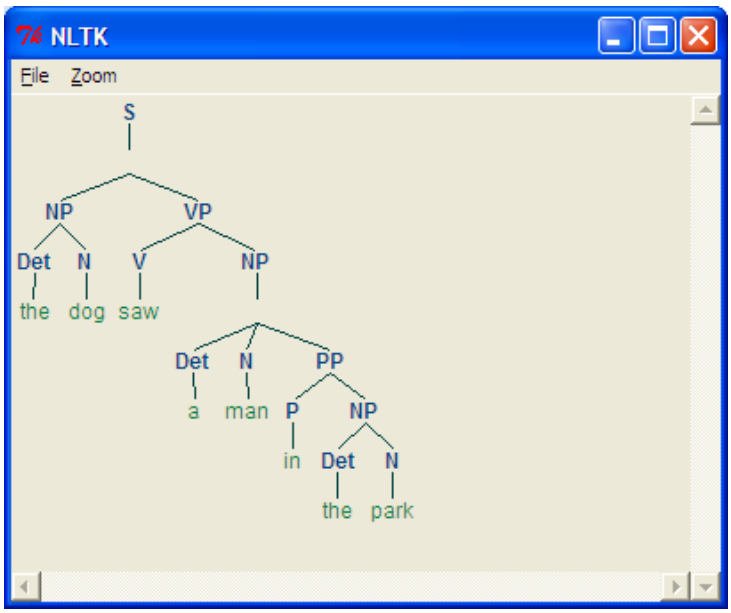
>>>t=nltk.Tree('(S((NP((Detthe)(Nom((Adjlittle)(N bear)))))(VP((VP(V saw)(NP(Detthe)(Nom(Adjfine)(Adjfat)(N trout))))(PP(P in)(NP(Detthe)(Nom(N brook))))))))')

>>>t.draw()



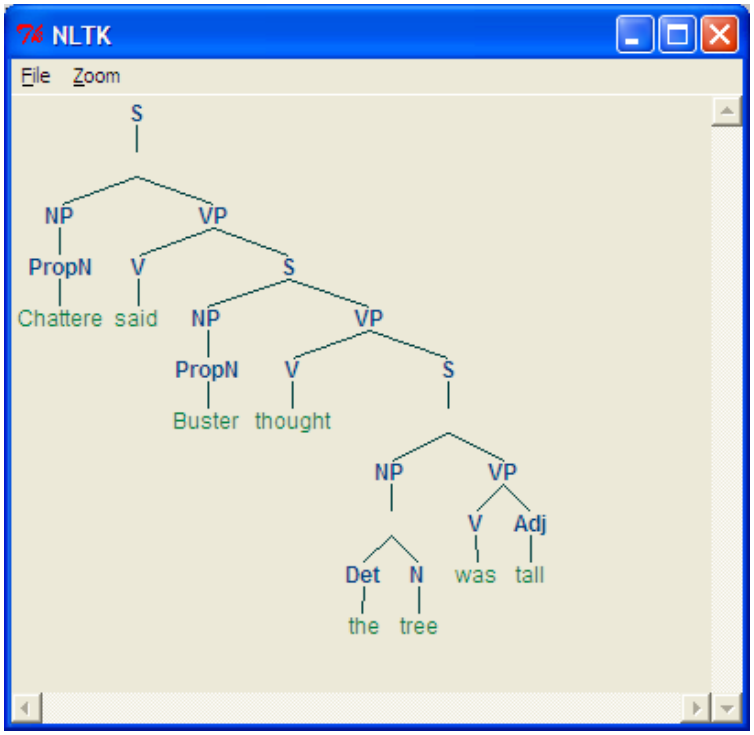
>>>t=nltk.Tree('(S((NP(Detthe)(N dog))(VP(V saw)(NP((Det a)(N man)(PP(P in)(NP(Detthe)(N park)))))))')

>>>t.draw()



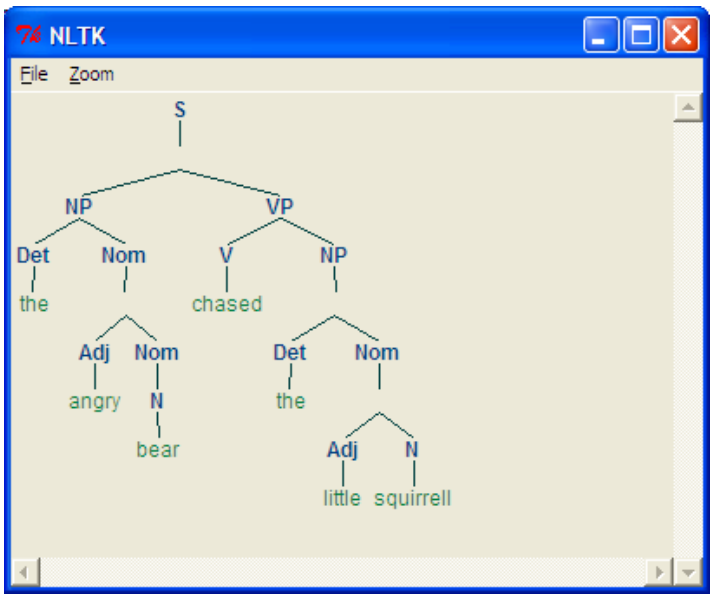
>>>t=nltk.Tree('(S((NP(PropNChattere))(VP(V said)(S(NP(PropNBuster))(VP(V thought)(S((NP((Detthe)(N tree)))(VP(V was)(Adjtall)))))))))')

>>>t.draw()



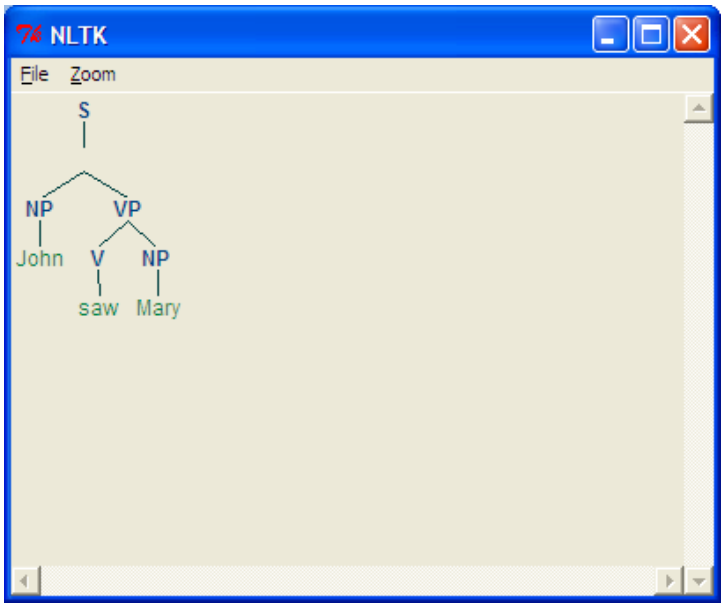
>>> t=nltk.Tree('(S((NP(Detthe)(Nom((Adjangry)(Nom(N bear)))))(VP(V chased)(NP((Detthe)(Nom((Adjlittle)(N squirrell))))))))')

>>>t.draw()



>>>t=nltk.Tree('(S((NP John)(VP(V saw)(NP Mary))))')

>>>t.draw()



5. Написати програму побудови дерев для речення Thewomansaw a manlastThursday.

```
import nltk
Buffalo_grammar = nltk.parse_cfg ("""
S -> NP VP
NP -> Det N | 'last' N
VP -> V NP NP
Det -> 'The' | 'a' | 'last'
N -> 'woman' | 'man' | 'Saturday'
V -> 'saw'
""")

sent = ['The', 'woman', 'saw', 'a', 'man', 'last', 'Saturday']
parser = nltk.ChartParser(Buffalo_grammar)
trees = parser.parse(sent)
for tree in trees:
    print (tree)
```

Ln: 2 Co

```
AttributeError: 'module' object has no attribute 'cfg'
>>> ================================ RESTART ================================
>>>
(NP (Det The) (N woman))
(VP (V saw) (NP (Det a) (N man)) (NP last (N Saturday)))
>>>
```

```
|
|  remove(...)
|      L.remove(value) -- remove first occurrence of value.
|      Raises ValueError if the value is not present.
|
|  reverse(...)
|      L.reverse() -- reverse *IN PLACE*
|
|  sort(...)
|      L.sort(cmp=None, key=None, reverse=False) -- stable sort *IN PLACE*;
|      cmp(x, y) -> -1, 0, 1
|
|  ----------------------------------------------------------------------
```

10. Здійснити аналіз послідовності слів: BuffalobuffaloBuffalobuffalobuffalo buffaloBuffalobuffalo. Оскільки, згідно з http://en.wikipedia.org/wiki/Buffalo_buffalo_Buffalo_buffalo_buffalo_buffalo_Buffalo_buffalo це граматично правильне речення, напишіть контексно-вільну граматику на основі дерева наведеного на цій сторінці з Інтернету. Здійсніть нормалізацію слів (lowercase), для моделювання ситуації коли слухач сприймає це речення на слух. Скільки дерев розбору може мати це дерево в такому випадку?

```
import nltk
Buffalo_grammar = nltk.parse_cfg("""
S -> NP VP
NP -> NP RC | PN N
VP -> V NP
RC -> NP V
PN -> 'Buffalo'
N -> 'buffalo'
V -> 'buffalo'
""")
buffalo_grammar = nltk.parse_cfg("""
S -> NP VP
NP -> NP RC | PN N
VP -> V NP
RC -> NP V
PN -> 'buffalo'
N -> 'buffalo'
V -> 'buffalo'
""")
sent = "Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo".split()
parser = nltk.ChartParser(Buffalo_grammar)
trees = parser.nbest_parse(sent)
for tree in trees:
    print tree
    tree.draw()
sent2 = "buffalo buffalo buffalo buffalo buffalo buffalo buffalo buffalo".split()
parser2 = nltk.ChartParser(buffalo_grammar)
trees2 = parser2.nbest_parse(sent2)
for tree in trees2:
    print tree
    tree.draw()
 |       L.sort(cmp=None, key=None, reverse=False) -- stable sort "IN PLACE";
 |       cmp(x, y) -> -1, 0, 1
 |
 |  ----------------------------------------------------------------------
```

Рис.1 Виконанняпрограми №10

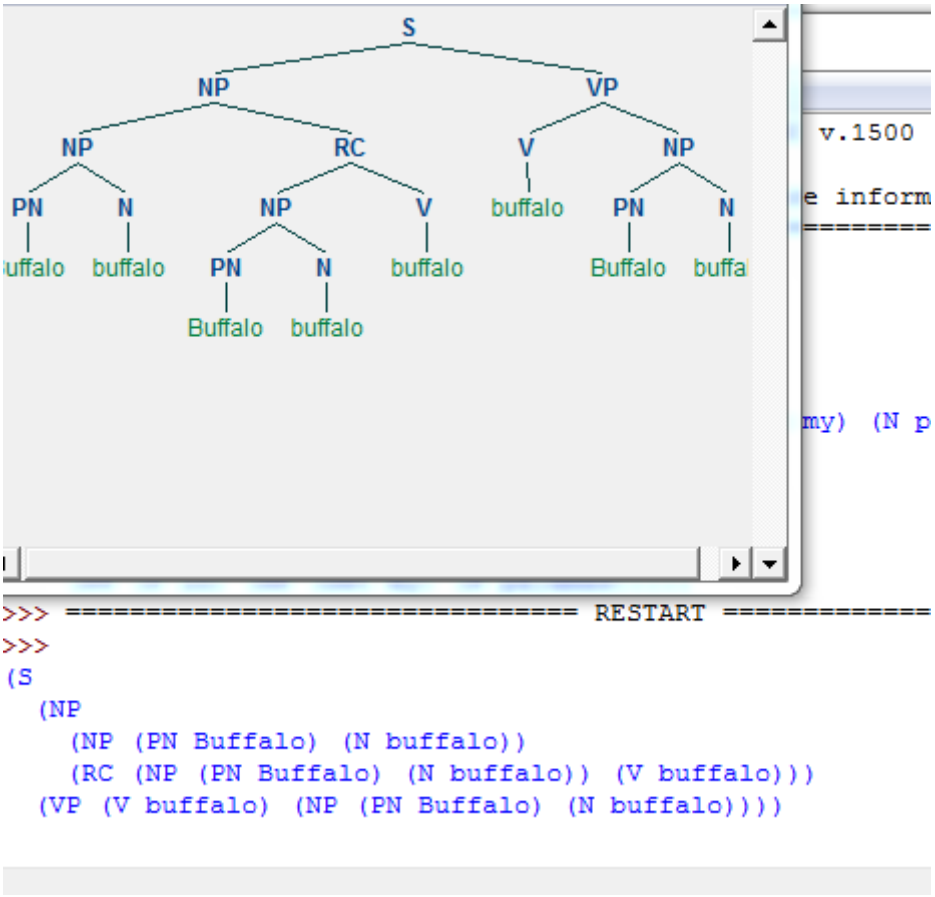
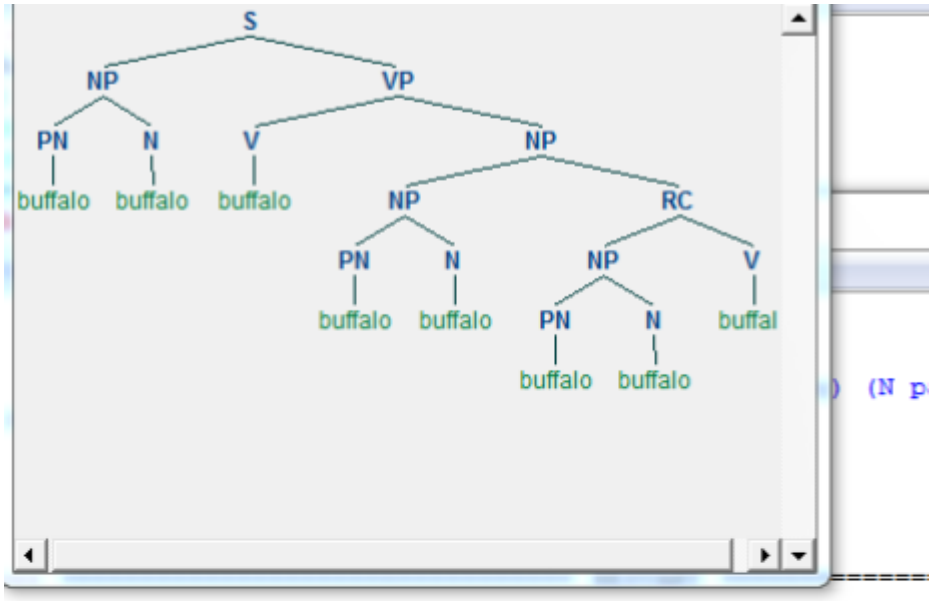


Рис.2 Результативиконанняпрограми №10(а)




```
>>>
(S
  (NP
    (NP (PN Buffalo) (N buffalo))
    (RC (NP (PN Buffalo) (N buffalo)) (V buffalo)))
  (VP (V buffalo) (NP (PN Buffalo) (N buffalo))))
(S
  (NP (PN buffalo) (N buffalo))
  (VP
    (V buffalo)
    (NP
      (NP (PN buffalo) (N buffalo))
      (RC (NP (PN buffalo) (N buffalo)) (V buffalo)))))
```

Рис.3 Результативиконанняпрограми №10(б)

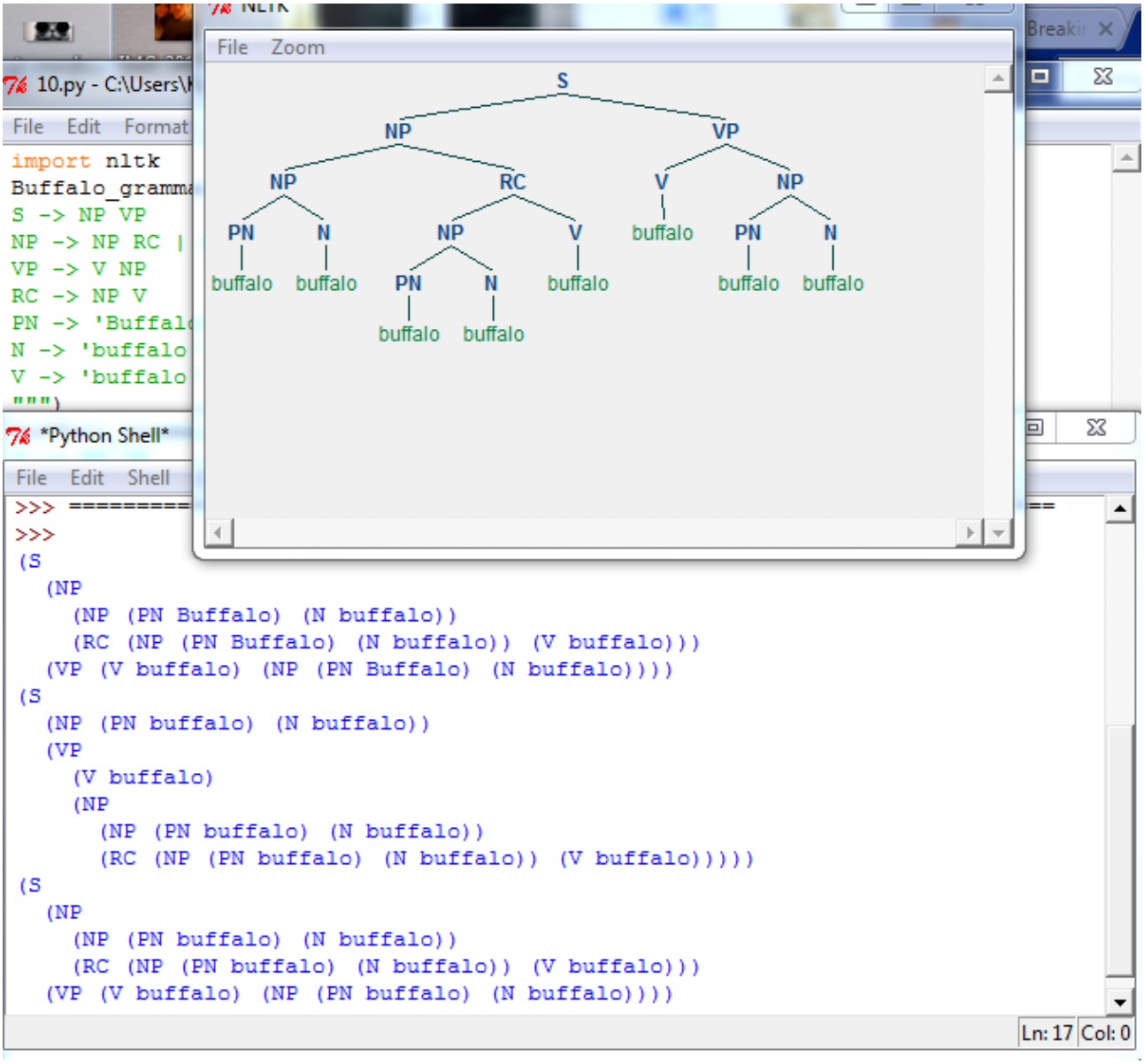


Рис.4Результативиконанняпрограми №10(в)

12. Написати програму порівняння швидкодії всіх аналізаторів, які згадувалися в методичних. Використовувати timeit функцію для визначення часу синтаксичного аналізу одного і того самого речення різними аналізаторами.

```
import nltk
import timeit
def timeit(s ,parser):
    import time
    start = time.clock()
    print parser.parse(s)
    return time.clock()-start
grammar2 = nltk.parse_cfg("""
S  -> NP VP
NP -> Det Nom | PropN
Nom -> Adj Nom | N
VP -> V Adj | V NP | V S | V NP PP
PP -> P NP
PropN -> 'Buster' | 'Chatterer' | 'Joe'
Det -> 'the' | 'a'
N -> 'bear' | 'squirrel' | 'tree' | 'fish' | 'log'
Adj  -> 'angry' | 'frightened' |  'little' | 'tall'
V ->  'chased'  | 'saw' | 'said' | 'thought' | 'was' | 'put'
P -> 'on'
""")
parser = nltk.ChartParser(grammar2)
rd_parser = nltk.RecursiveDescentParser(grammar2)
sr_parser = nltk.ShiftReduceParser(grammar2)
sent = "the angry bear chased the frightened little squirrel".split()
ChartParser = timeit(sent, parser)
print 'ChartParser =', ChartParser
print
RecursiveDescentParser = timeit(sent, rd_parser)
print 'RecursiveDescentParser =', RecursiveDescentParser
print
ShiftReduceParser = timeit(sent, sr_parser)
print 'ShiftReduceParser =', ShiftReduceParser
```

Рис.5 Текст програми №12

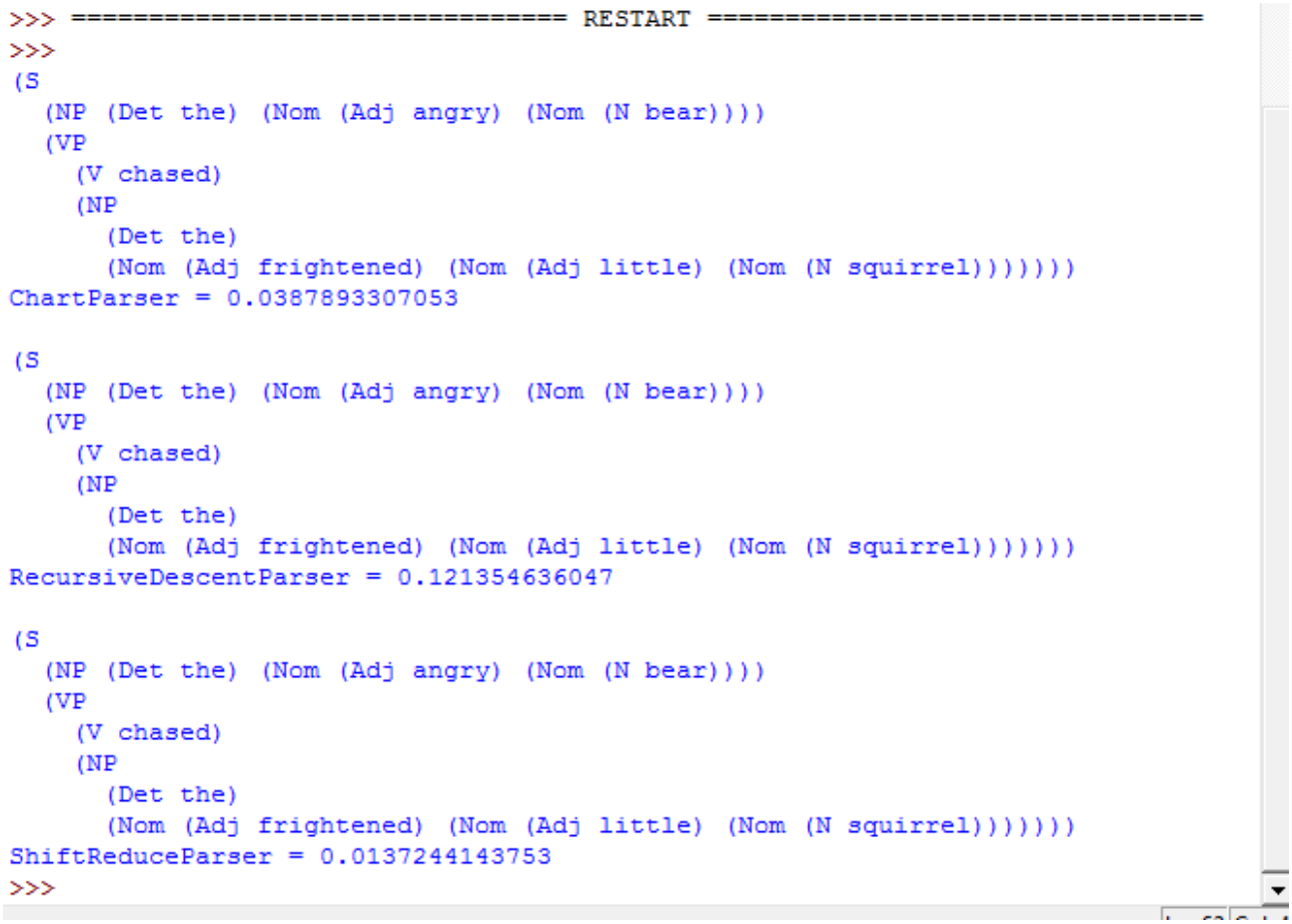


Рис.6 Результативиконанняпрограми №12

13.Прочитати про "gardenpath" речення http://en.wikipedia.org/wiki/Garden_path_sentence. Оцінити обчислювальну складність аналізу таких речень в порівнянні з труднощами аналізу таких речень людиною?

"Gardenpathsentences" - це граматично правильні речення, які починаються в такий спосіб, що в читачів тлумачення буде неправильним; вони здійснюють неправильний граматичний аналіз, який заводить їх в тупік. Ці речення використовуються в психолінгвістиці."Gardenpath" відноситься до висловлювання - "управляти садовою доріжкою" тобто "вводити в оману". Згідно з існуючою теорією в психолінгвістиці, коли читач читає "gardenpathsentences" то він будує структуру значення одного слова в даний момент часу. В певний момент читачеві стає очевидно, що наступне слово чи фраза не може бути вставлено в структуру збудовану до цього часу: це не сумісно з доріжкою якою він управляє. Ці речення є поширені в аналітичній

мові, де порядок в значній мірі залежить від встановлення граматичного відмінку в реченні.

Приклади "garden path sentence":

1) The author wrote the novel was likely to be a best-seller.

The author composed the novel...

The author wrote that the novel was likely to be a best-seller.

2) The man returned to his house was happy.

he man came back to his house...

Returned to his house, the man was happy.

3) The government plans to raise taxes were defeated.

The government is planning to raise taxes...

The government's plans to raise taxes were defeated.

Синтаксичний аналіз визначає як діляться фрази в "gardenpathsentences". Синтаксичний аналіз є загальним терміном, який використовується в психолінгвістиці описуючи розуміння мови. Людина скоріше(правильніше), ніж комп'ютер аналізує речення (визначає частини мови, синтаксичні зв'язки і т.д.)

Синтаксичний аналізатор, так як і людина визначає як поділити речення , але до одного і того ж речення можливо приписувати різні граматичні структури. Наприклад "Heatethecookiesonthecouch," це може означати, що він їсть печиво яке є на кушетці, або, що він сидить на кушетці в той час коли їсть печиво. Отже, як і людина так і машина може припуститися помилки, хоча людина правильніше аналізує речення.

Висновок: на цій лабораторній роботі я ознайомилася з автоматичним синтаксичним аналізом в NLTK, алгоритмами рекурсивного спуску(зверху-вниз) та переміщення-згортання (знизу-вверх), навчилися будувати дерева.