**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**
**ІНСТИТУТ КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

Кафедра "Системи автоматизованого проектування"

Звіт

до лабораторної роботи №12

на тему: ВИВЧЕННЯ БІБЛІОТЕКИ ПРИКЛАДНИХ ПРОГРАМ NLTK, ДЛЯ ОПРАЦЮВАННЯ ТЕКСТІВ ПРИРОДНОЮ МОВОЮ. АВТОМАТИЧНИЙ СИНТАКСИЧНИЙ АНАЛІЗ (частина2).

з дисципліни "Комп'ютерна лінгвістика"

Виконала:
студентка групи ПРЛм-11
Гарбуз Л.В.
Прийняв:
викладач
Дупак Б.П.

Львів 2015

**Мета роботи:** вивчення основ програмування на мові Python. Ознайомлення з автоматичним синтаксичним аналізом в NLTK.

**Тексти програм на мові *Python*.**

Варіант – 3

1. Написати рекурсивну функцію для перегляду дерева, яка визначає його глибину. Дерево з одного вузла має глибину рівну нулю. (глибина піддерева це максимальна глибина його дітей плюс один)

```
import nltk
t = nltk.Tree('(S (NP Marry) (VP chased (NP the cat)))')
def traverse(t):
    try:
        t.node
    except AttributeError:
            print t,
    else:
            print '(', t.node,
            for child in t:
                traverse(child)
            print ')',
print t.height()
print traverse(t)

>>>
4
( S ( NP Marry ) ( VP chased ( NP the cat ) ) ) None
>>>
```

4. Розширити граматику grammar2 з попередньої лабораторної роботи правилами які розділяють прийменники як перехідні, неперехідні та такі що вимагають PP доповнення. На основі цих правил побудуйте дерево розбору для речення Lee ran away home, використовуючи аналізатор рекурсивного спуску.

```
import nltk.corpus
grammar2 = nltk.CFG.fromstring("""
S  -> NP VP
NP -> Det Nom | PropN| N
Nom -> Adj Nom | N
VP -> V Adj | V NP | V S | V NP PP|V PP
PP -> P NP
PropN -> 'Ann' | 'Katie' | 'Vitaliy'| 'He'
Det -> 'the' | 'a'
N -> 'sunrise' | 'puppy' | 'tree' | 'fish' | 'life'|'Lee'|'home'|'hill'
Adj  -> 'devious' | 'frightened' |  'strong' | 'tall'
V -> 'ran'| 'like'  | 'kiss' | 'said' | 'cheered' | 'was' | 'put'|'dance'
P -> 'on'|'away'|'up'
""")
rd_parser = nltk.RecursiveDescentParser(grammar2)
sent = "Lee ran away home".split()
for tree in rd_parser.parse(sent):
    print (tree)
|
grammar2 = nltk.CFG.fromstring("""
S  -> NP VP
NP -> Det Nom | PropN | N
TP -> P NP
IP -> P ADJ | P | P ADV
PC -> P PP
DP -> NP P NP
Nom -> Adj Nom | N
VP -> V Adj | V NP | V S | V NP PP| V PP
PP -> P NP
PropN -> 'Ann' | 'Katie' | 'Vitaliy'| 'He'
Det -> 'the' | 'a'
N -> 'sunrise' | 'puppy' | 'tree' | 'fish' | 'life'|'Lee'|'home'|'hill'
Adj  -> 'devious' | 'frightened' |  'strong' | 'tall'|'full'
ADV -> 'heavily'|'soon'|'already'
V -> 'ran'| 'walk'  | 'hug' | 'said' | 'love' | 'was' | 'put'
P -> 'on'|'away'|'up with'|'up'
""")
rd_parser = nltk.RecursiveDescentParser(grammar2)
sent = "Lee ran away home".split()
for tree in rd_parser.parse(sent):
```

```
>>>
(S (NP (N Lee)) (VP (V ran) (PP (P away) (NP (N home)))))
(S (NP (N Lee)) (VP (V ran) (PP (P away) (NP (N home)))))
>>>
```

5. Вибрати декілька (2) загальних дієслова та напишіть програми для вирішення наступних задач:

Пошук дієслів в корпусі Prepositional Phrase Attachment Corpus nltk.corpus.ppattach. Пошук всіх випадків вживання дієслова з двома різними PP в яких перший іменник, або другий іменник або прийменник залишаються незмінними. Розробити правила CFG граматики для врахування цих випадків.

```
import nltk.corpus
from nltk import Tree
entries = nltk.corpus.ppattach.attachments('training')
table = nltk.defaultdict(lambda: nltk.defaultdict(set))
for entry in entries:
    key = entry.noun1 + '-' + entry.prep + '-' + entry.noun2
    table[key][entry.attachment].add(entry.verb)
for key in sorted(table):
    if len(table[key]) > 1:
        print (key, 'N:', sorted(table[key]['N']), 'V:', sorted(table[key]['V']))
groucho_grammar = nltk.CFG.fromstring("""
VP -> V NP PP | V NP
NP -> N | N PP
PP -> P N
N -> 'authority'|'administration'|'inventors'|'microchips'
V -> 'gives'
P -> 'to'|'of'
""")

sent = ['gives', 'authority', 'to', 'administration']
sent2= ['gives', 'inventors', 'of', 'microchips']
parser = nltk.ChartParser(groucho_grammar)
trees = parser.parse(sent)
trees2 = parser.parse(sent2)

for tree in trees:
    print(tree)

for tree in trees2:
    print(tree)
```

```
NP -> NNP NNP NNP NNP
NP -> NP PP
NP-PRD -> NP PP
NP-SBJ -> -NONE-
NP-SBJ-1 -> NP , UCP ,
S -> NP-SBJ NP-PRD
S -> NP-SBJ-1 VP .
UCP -> ADJP CC NP
VBD -> 'was'
VBN -> 'named'
```

```
projections-for-year N: ['slashed'] V: ['exceed']
provisions-for-loans N: ['taken'] V: ['made']
rates-to-% N: ['boosting'] V: ['increase', 'pushed', 'raised']
reporter-in-bureau N: ['is'] V: ['is']
restrictions-on-use N: ['waiving'] V: ['impose']
revenue-for-year N: ['projected'] V: ['had']
revenue-in-quarter N: ['expects'] V: ['had']
sales-in-excess N: ['combined'] V: ['had']
sales-in-quarter N: ['had'] V: ['increasing']
sales-of-million N: ['expected', 'had', 'has', 'have', 'posted'] V: ['had']
salvo-in-battle N: ['marking'] V: ['marking']
services-for-customers N: ['offering'] V: ['provide']
shareholder-in-bank N: ['become'] V: ['become']
stake-in-Airlines N: ['acquiring', 'buy', 'raise'] V: ['buy']
stake-in-Mixte N: ['bring'] V: ['boost']
stake-in-Rally N: ['hold'] V: ['had']
stake-in-company N: ['bought', 'building', 'built', 'buying', 'give', 'hold',
btaining', 'own', 'owns', 'raised', 'take'] V: ['accumulating', 'had', 'has',
olds', 'own']
stake-in-concern N: ['acquires', 'lowered'] V: ['retaining']
stake-in-unit N: ['sell'] V: ['acquire']
stake-in-venture N: ['has', 'hold', 'holds'] V: ['held']
suit-against-Keating N: ['press'] V: ['brought']
swings-in-market N: ['cause', 'create'] V: ['cause']
system-for-city N: ['design'] V: ['design']
system-in-Pakistan N: ['operate'] V: ['operate']
time-for-Congress N: ['is'] V: ['buy', 'buys']
venture-with-Co. N: ['started'] V: ['started']
ventures-with-companies N: ['established'] V: ['form']
verdict-in-case N: ['is', 'won'] V: ['won']
volatility-in-stocks N: ['ignoring'] V: ['see']
vote-on-issue N: ['allow'] V: ['prevent']
way-for-declines N: ['open'] V: ['pave']
writer-in-York N: ['is'] V: ['is']
yen-to-yen N: ['shed'] V: ['advanced', 'fell', 'gained', 'lost', 'rose']
(VP (V gives) (NP (N authority)) (PP (P to) (N administration)))
(VP (V gives) (NP (N authority) (PP (P to) (N administration))))
(VP (V gives) (NP (N inventors)) (PP (P of) (N microchips)))
(VP (V gives) (NP (N inventors) (PP (P of) (N microchips))))
>>>
```

8. Здійснити аналіз корпуса Prepositional Phrase Attachment Corpus та спробувати знайти фактори, які впливають на місце приєднання PP.

```
>>> import nltk
>>> entries = nltk.corpus.ppattach.attachments('training')
>>> table = nltk.defaultdict(lambda: nltk.defaultdict(set))
>>> for entry in entries:
        key = entry.verb + '-' + entry.prep + '-' + entry.noun1
        table[key][entry.attachment].add(entry.verb)


>>> for key in sorted(table):
        if len(table[key]) > 1:
            print key, 'V:', sorted(table[key]['N']), 'V:',
        sorted(table[key]['V'])


[]
[]
["'re"]
["'re"]
[]
[]
[]
["'re"]
[]
[]
["'s"]
[]
["'s"]

['earned']
earned-on-million V: ['earned'] V:
['earned']
['earned']
['earning']
[]
['earns']
[]
[]
[]
['ease']
['ease']
['ease']
['ease']
[]
[]
ease-on-restrictions V: ['ease'] V:
['ease']
[]
['ease']
[]
['ease']
['eased']
['eased']
[]
[]
```

12. Розробити програму обробки дерев корпуса Treebank nltk.corpus.treebank, яка вилучить всі правила з кожного з дерев за допомогою Tree.productions(). Правилами, які зустрічаються тільки один раз можна знехтувати. Правила з

однаковими лівими частинами  та подібними правими частинами об’єднати для отримання еквівалентного але більш компактного набору правил.

```
import nltk
from nltk.corpus import treebank
t = treebank.parsed_sents('wsj_0002.mrg')[0]
print t
p= t.productions()
print p
Fdist=nltk.FreqDist(p)
fd=Fdist.keys()
for i in Fdist.keys():
    if Fdist[i]>0:
        print i
```

```
PP -> IN NP
IN -> 'of'
NP -> NNP NNP NNP NNP
NNP -> 'Consolidated'
NNP -> 'Gold'
NNP -> 'Fields'
NNP -> 'PLC'
, -> ','
VP -> VBD VP
VBD -> 'was'
VP -> VBN S
VBN -> 'named'
S -> NP-SBJ NP-PRD
NP-SBJ -> -NONE-
-NONE- -> '*-1'
NP-PRD -> NP PP
NP -> DT JJ NN
DT -> 'a'
JJ -> 'nonexecutive'
NN -> 'director'
PP -> IN NP
IN -> 'of'
NP -> DT JJ JJ NN
DT -> 'this'
JJ -> 'British'
JJ -> 'industrial'
NN -> 'conglomerate'
. -> '.'
>>>
```

**Висновок:** на цій лабораторній роботі я продовжила ознайомлення з автоматичним синтаксичним аналізом в NLTK.