SUVRAJIT BHATTACHARJEE

SUID 866572756

QUESTION1:

Developing a CFG grammar to prase four sentences

First, I imported NLTK into my python environment. Then I defined the 4 given sentences into variables. text1,text2…. Respectively.

```
>>> import nltk

>>> from nltk import *

>>> text1="We had a nice party yesterday"

>>> text2="She came to visit me two days ago"

>>> text3="You may go now"

>>> text4="Their kids are not always naive"

>>> textsplit1 = nltk.sent_tokenize(text1)

>>> textsplit1

['We had a nice party yesterday']

>>> tokentext1 = [nltk.word_tokenize(sent) for sent in textsplit1]

>>> tokentext1

[['We', 'had', 'a', 'nice', 'party', 'yesterday']]
```

I first tokenized each sentence then further tokenized the words in the sentences. The sentences were tokenized twice.

What we need are the individual words. Hence we first tokenize the sentences the tokenize the words in that sentence.

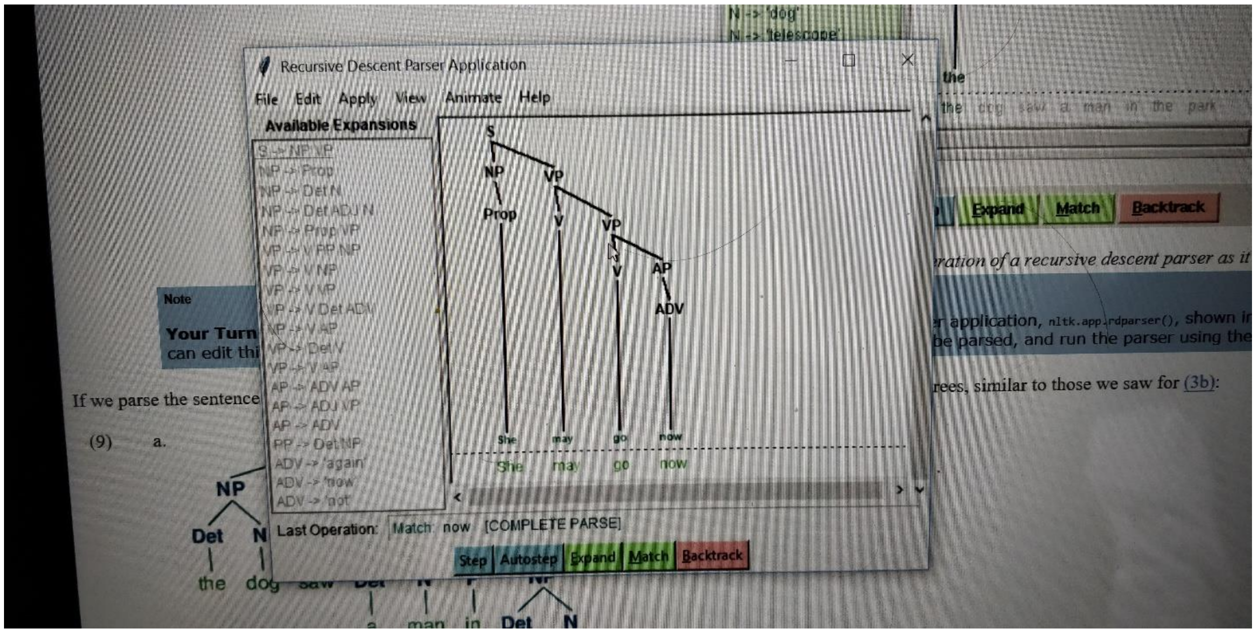
I did for all of the sentences:

```
>>> from nltk.corpus import treebank

>>> treebank_tagged = treebank.tagged_sents()

>>> size = int(len(treebank_tagged) * 0.9)

>>> treebank_train = treebank_tagged[:size]

>>> treebank_test = treebank_tagged[size:]

>>> t0 = nltk.DefaultTagger('NN')
>>> t1 = nltk.UnigramTagger(treebank_train, backoff=t0)

>>> t2 = nltk.BigramTagger(treebank_train, backoff=t1)

>>> taggedtext1 = [t2.tag(tokens) for tokens in tokentext1]

>>> taggedtext1

[[('We', 'PRP'), ('had', 'VBD'), ('a', 'DT'), ('nice', 'JJ'), ('party', 'NN'), ('yesterday', 'NN')]]

>>> textsplit3= nltk.sent_tokenize(text3)

>>> tokentext3= [nltk.word_tokenize(sent) for sent in textsplit3]
>>> taggedtext3= [t2.tag(tokens) for tokens in tokentext3]
>>> taggedtext3
[[('You', 'PRP'), ('may', 'MD'), ('go', 'VB'), ('now', 'RB')]]
```

```
>>> textsplit4= nltk.sent_tokenize(text4)
>>> tokentext4= [nltk.word_tokenize(sent) for sent in textsplit4]
>>> taggedtext4= [t2.tag(tokens) for tokens in tokentext4]
>>> taggedtext4
[[('Their', 'PRP$'), ('kids', 'NNS'), ('are', 'VBP'), ('not', 'RB'), ('always', 'RB'), ('naive',
'NN')]]
```

Then we proceed with the bottom up approach.

Here is an example for the sentence “She may go now”.



Last Operation: Match: now [COMPLETE PARSE]

A similar process was used for all 4 sentences to come up with the following grammar:

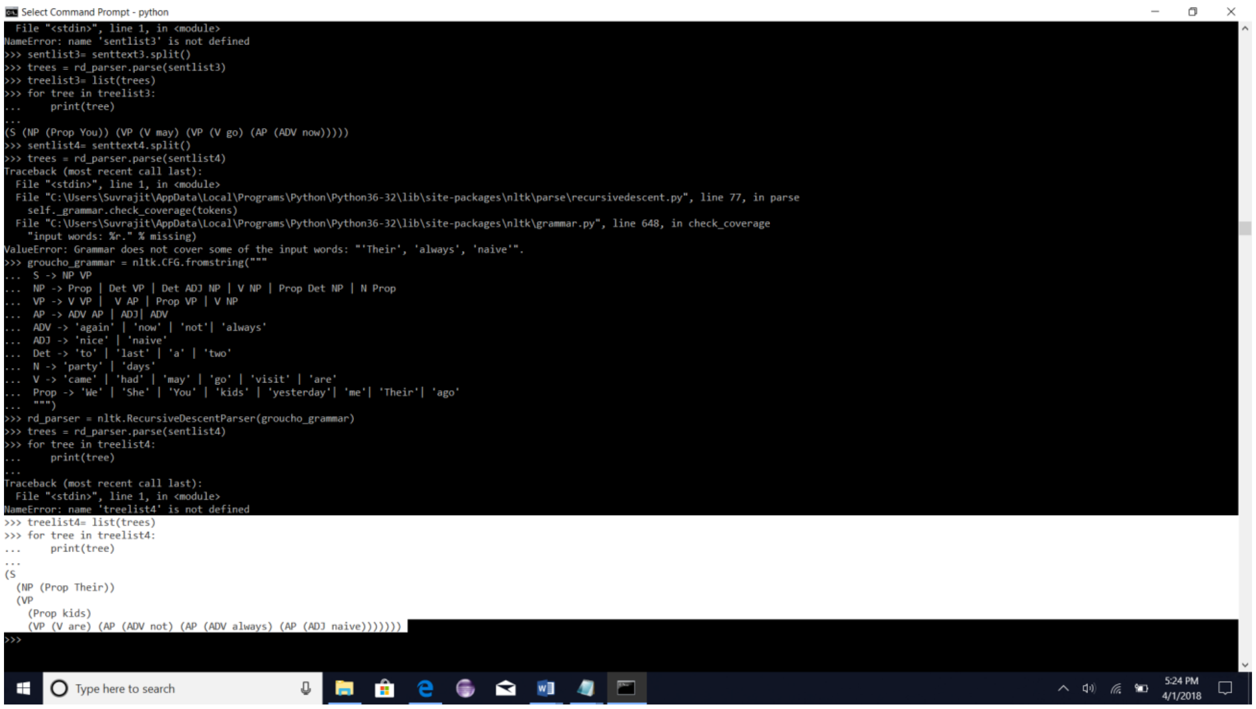
```
groucho_grammar = nltk.CFG.fromstring("""
...  S -> NP VP
...  NP -> Prop | Det VP | Det ADJ NP | V NP | Prop Det NP | N Prop
...  VP -> V VP |  V AP | Prop VP | V NP
...  AP -> ADV AP | ADJ| ADV
...  ADV -> ' 'now' | 'not'| 'always'
...  ADJ -> 'nice' | 'naive'
...  Det -> 'to' | 'last' | 'a' | 'two'
...  N -> 'party' | 'days'
...  V -> 'came' | 'had' | 'may' | 'go' | 'visit' | 'are'
...  Prop -> 'We' | 'She' | 'You' | 'kids' | 'yesterday'| 'me'| 'Their'| 'ago'
...  """)
```

While the approach to derive the grammar was bottom-up , I used the RecursiveDescentParser, which is a top-bottom parser.

```
 rd_parser = nltk.RecursiveDescentParser(groucho_grammar)
trees = rd_parser.parse(sentlist1)
 treelist= list(trees)
```

```
treelist
[Tree('S', [Tree('NP', [Tree('Prop', ['We'])]), Tree('VP', [Tree('V', ['had']), Tree('NP', [Tree('Det', ['a']),
Tree('ADJ', ['nice']), Tree('NP', [Tree('N', ['party']), Tree('Prop', ['yesterday'])])])])])]
>>> for tree in treelist:
...     print(tree)
...
(S
  (NP (Prop We))
  (VP
    (V had)
    (NP (Det a) (ADJ nice) (NP (N party) (Prop yesterday)))))
>>> trees = rd_parser.parse(sentlist2)
>>> treelist2= list(trees2)
>>> treelist= list(trees)
>>> for tree in treelist2:
...     print(tree)
...
(S
  (NP (Prop She))
  (VP
    (V came)
    (NP
      (Det to)
      (VP
        (V visit)
        (NP (Prop me) (Det two) (NP (N days) (Prop ago)))))))
>>> trees = rd_parser.parse(sentlist3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sentlist3' is not defined
```

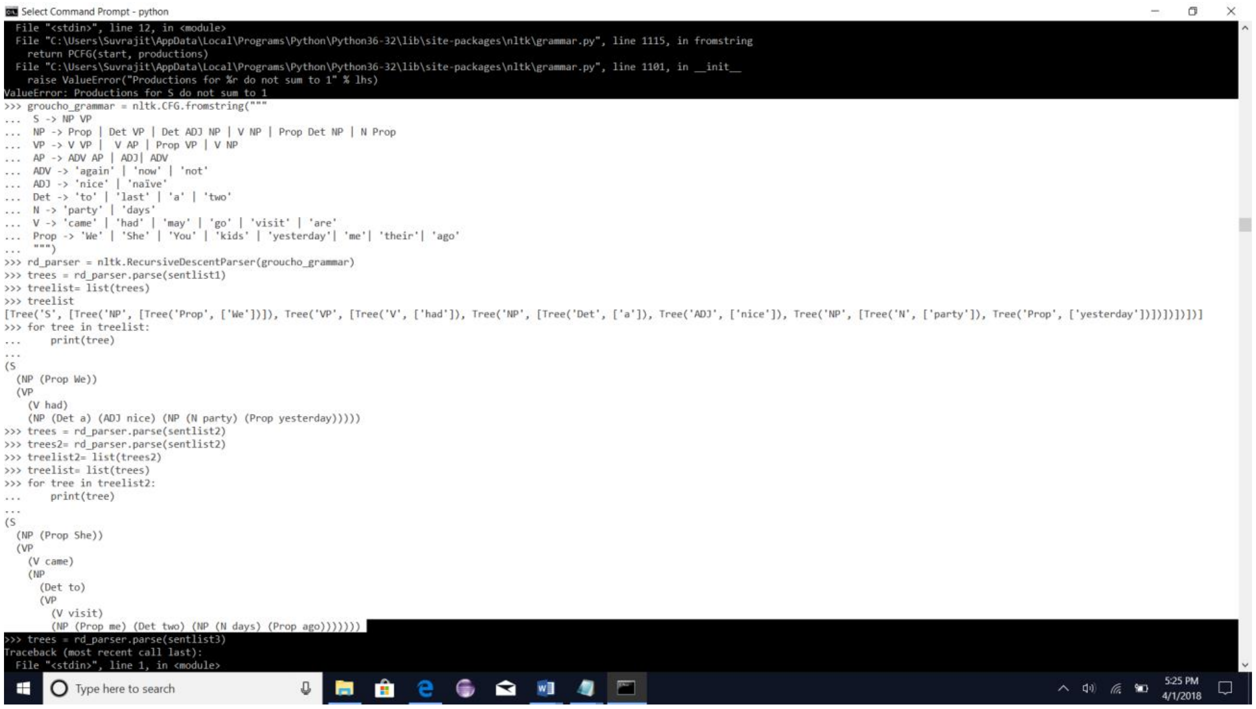
```
>>> sentlist3= senttext3.split()

>>> trees = rd_parser.parse(sentlist3)

>>> treelist3= list(trees)

>>> for tree in treelist3:

...     print(tree)

...

(S (NP (Prop You)) (VP (V may) (VP (V go) (AP (ADV now)))))


>>> treelist4= list(trees)

>>> for tree in treelist4:

...     print(tree)

...

(S

  (NP (Prop Their))

  (VP

    (Prop kids)

    (VP (V are) (AP (ADV not) (AP (ADV always) (AP (ADJ naive)))))))
```


Select Command Prompt - python

```
  File "<stdin>", line 1, in <module>
NameError: name 'sentlist3' is not defined
>>> sentlist3= senttext3.split()
>>> trees = rd_parser.parse(sentlist3)
>>> treelist3= list(trees)
>>> for tree in treelist3:
...     print(tree)
...
(S (NP (Prop You)) (VP (V may) (VP (V go) (AP (ADV now)))))
>>> sentlist4= senttext4.split()
>>> trees = rd_parser.parse(sentlist4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "C:\Users\Suvrajit\AppData\Local\Programs\Python\Python36-32\lib\site-packages\nltk\parse\recursivedescent.py", line 77, in parse
    self._grammar.check_coverage(tokens)
  File "C:\Users\Suvrajit\AppData\Local\Programs\Python\Python36-32\lib\site-packages\nltk\grammar.py", line 648, in check_coverage
    "input words: %r." % missing)
ValueError: Grammar does not cover some of the input words: "'Their', 'always', 'naive'".
>>> groucho_grammar = nltk.CFG.fromstring("""
...  S -> NP VP
...  NP -> Prop | Det VP | Det ADJ NP | V NP | Prop Det NP | N Prop
...  VP -> V VP |  V AP | Prop VP | V NP
...  AP -> ADV AP | ADJ| ADV
...  ADV -> 'again' | 'now' | 'not'| 'always'
...  ADJ -> 'nice' | 'naive'
...  Det -> 'to' | 'last' | 'a' | 'two'
...  N -> 'party' | 'days'
...  V -> 'came' | 'had' | 'may' | 'go' | 'visit' | 'are'
...  Prop -> 'We' | 'She' | 'You' | 'kids' | 'yesterday'| 'me'| 'Their'| 'ago'
...  """)
>>> rd_parser = nltk.RecursiveDescentParser(groucho_grammar)
>>> trees = rd_parser.parse(sentlist4)
>>> for tree in treelist4:
...     print(tree)
...
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'treelist4' is not defined
>>> treelist4= list(trees)
>>> for tree in treelist4:
...     print(tree)
...
(S
  (NP (Prop Their))
  (VP
    (Prop kids)
    (VP (V are) (AP (ADV not) (AP (ADV always) (AP (ADJ naive)))))))
>>>
```

Type here to search

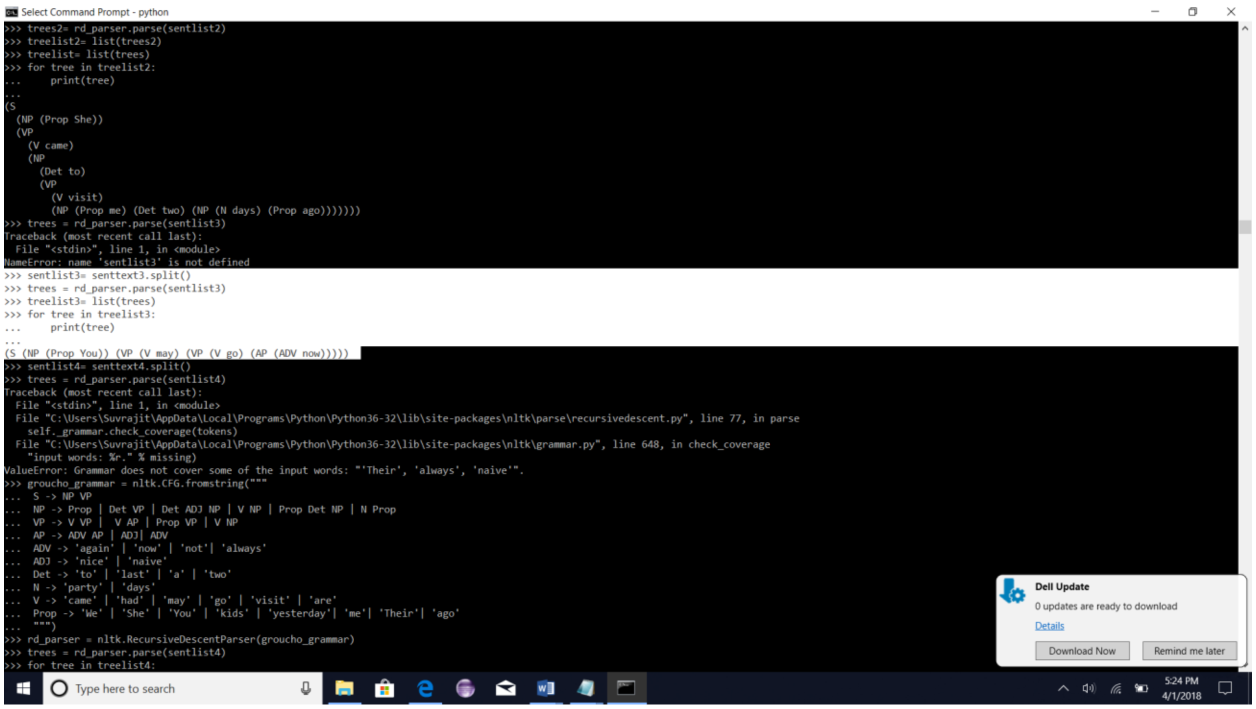
5:24 PM
4/1/2018


Select Command Prompt - python

```
  File "<stdin>", line 12, in <module>
  File "C:\Users\Suvrajit\AppData\Local\Programs\Python\Python36-32\lib\site-packages\nltk\grammar.py", line 1115, in fromstring
    return PCFG(start, productions)
  File "C:\Users\Suvrajit\AppData\Local\Programs\Python\Python36-32\lib\site-packages\nltk\grammar.py", line 1101, in __init__
    raise ValueError("Productions for %r do not sum to 1" % lhs)
ValueError: Productions for S do not sum to 1
>>> groucho_grammar = nltk.CFG.fromstring("""
...  S -> NP VP
...  NP -> Prop | Det VP | Det ADJ NP | V NP | Prop Det NP | N Prop
...  VP -> V VP |  V AP | Prop VP | V NP
...  AP -> ADV AP | ADJ| ADV
...  ADV -> 'again' | 'now' | 'not'
...  ADJ -> 'nice' | 'naïve'
...  Det -> 'to' | 'last' | 'a' | 'two'
...  N -> 'party' | 'days'
...  V -> 'came' | 'had' | 'may' | 'go' | 'visit' | 'are'
...  Prop -> 'We' | 'She' | 'You' | 'kids' | 'yesterday'| 'me'| 'their'| 'ago'
...  """)
>>> rd_parser = nltk.RecursiveDescentParser(groucho_grammar)
>>> trees = rd_parser.parse(sentlist1)
>>> treelist= list(trees)
>>> treelist
[Tree('S', [Tree('NP', [Tree('Prop', ['We'])]), Tree('VP', [Tree('V', ['had']), Tree('NP', [Tree('Det', ['a']), Tree('ADJ', ['nice']), Tree('NP', [Tree('N', ['party']), Tree('Prop', ['yesterday'])])])])])]
>>> for tree in treelist:
...     print(tree)
...
(S
  (NP (Prop We))
  (VP
    (V had)
    (NP (Det a) (ADJ nice) (NP (N party) (Prop yesterday)))))
>>> trees = rd_parser.parse(sentlist2)
>>> trees2= rd_parser.parse(sentlist2)
>>> treelist2= list(trees2)
>>> treelist= list(trees)
>>> for tree in treelist2:
...     print(tree)
...
(S
  (NP (Prop She))
  (VP
    (V came)
    (NP
      (Det to)
      (VP
        (V visit)
        (NP (Prop me) (Det two) (NP (N days) (Prop ago)))))))
>>> trees = rd_parser.parse(sentlist3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
```

Type here to search

5:25 PM
4/1/2018



```
>>> trees2= rd_parser.parse(sentlist2)
>>> treelist2= list(trees2)
>>> treelist= list(trees)
>>> for tree in treelist2:
...     print(tree)
...
(S
  (NP (Prop She))
  (VP
    (V came)
    (NP
      (Det to)
      (VP
        (V visit)
        (NP (Prop me) (Det two) (NP (N days) (Prop ago)))))))
>>> trees = rd_parser.parse(sentlist3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sentlist3' is not defined
>>> sentlist3= senttext3.split()
>>> trees = rd_parser.parse(sentlist3)
>>> treelist3= list(trees)
>>> for tree in treelist3:
...     print(tree)
...
(S (NP (Prop You)) (VP (V may) (VP (V go) (AP (ADV now)))))
>>> sentlist4= senttext4.split()
>>> trees = rd_parser.parse(sentlist4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "C:\Users\Suvrajit\AppData\Local\Programs\Python\Python36-32\lib\site-packages\nltk\parse\recursivedescent.py", line 77, in parse
    self._grammar.check_coverage(tokens)
  File "C:\Users\Suvrajit\AppData\Local\Programs\Python\Python36-32\lib\site-packages\nltk\grammar.py", line 648, in check_coverage
    "input words: %r." % missing)
ValueError: Grammar does not cover some of the input words: "'Their', 'always', 'naive'".
>>> groucho_grammar = nltk.CFG.fromstring("""
...  S -> NP VP
...  NP -> Prop | Det VP | Det ADJ NP | V NP | Prop Det NP | N Prop
...  VP -> V VP |  V AP | Prop VP | V NP
...  AP -> ADV AP | ADJ| ADV
...  ADV -> 'again' | 'now' | 'not'| 'always'
...  ADJ -> 'nice' | 'naive'
...  Det -> 'to' | 'last' | 'a' | 'two'
...  N -> 'party' | 'days'
...  V -> 'came' | 'had' | 'may' | 'go' | 'visit' | 'are'
...  Prop -> 'We' | 'She' | 'You' | 'kids' | 'yesterday'| 'me'| 'Their'| 'ago'
...  """)
>>> rd_parser = nltk.RecursiveDescentParser(groucho_grammar)
>>> trees = rd_parser.parse(sentlist4)
>>> for tree in treelist4:
```

THREE OTHER SENTENCES THAT CAN BE PARSED BY THIS GRAMMAR ARE:

1. She may go now.

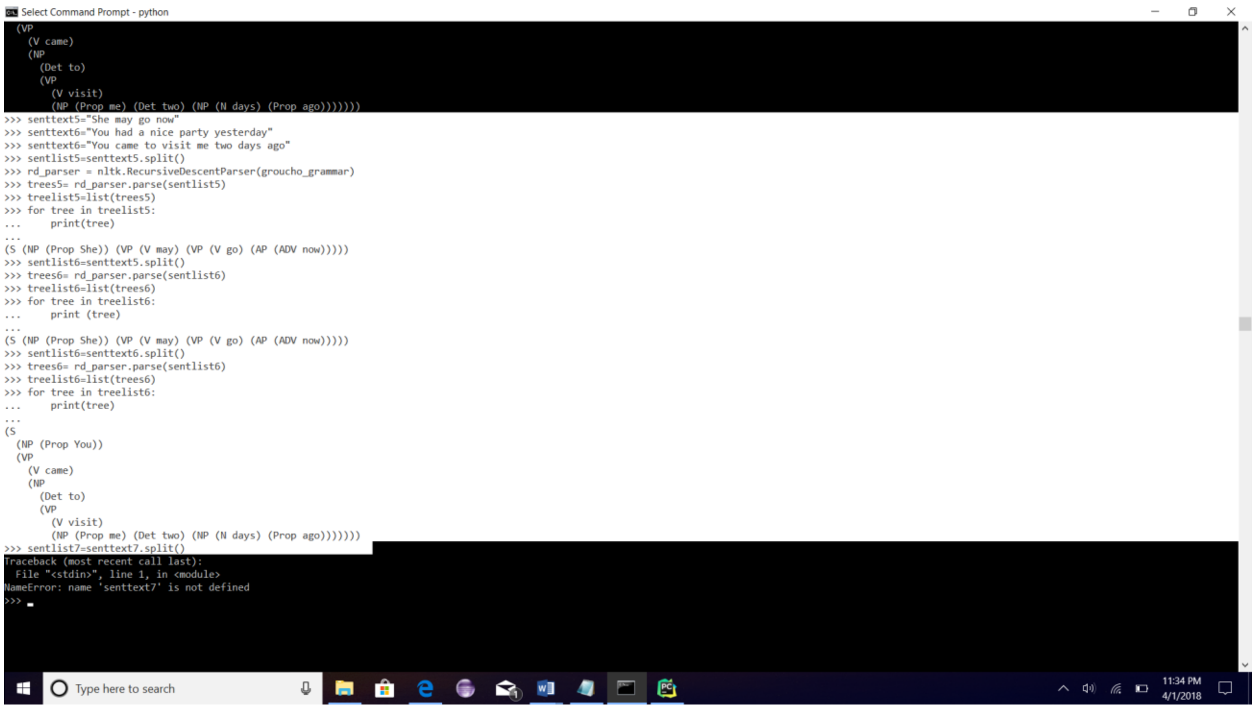
2.You had a nice party yesterday.

3.You came to visit me two days ago.

```
>>> senttext5="She may go now"

>>> senttext7="You had a nice party yesterday"

>>> senttext6="You came to visit me two days ago"

>>> sentlist5=senttext5.split()

>>> rd_parser = nltk.RecursiveDescentParser(groucho_grammar)

>>> trees5= rd_parser.parse(sentlist5)

>>> treelist5=list(trees5)

>>> for tree in treelist5:

...     print(tree)

...

(S (NP (Prop She)) (VP (V may) (VP (V go) (AP (ADV now)))))
```

```
>>> sentlist6=senttext5.split()
>>> trees6= rd_parser.parse(sentlist6)
>>> treelist6=list(trees6)
>>> for tree in treelist6:
...     print (tree)
...
(S (NP (Prop She)) (VP (V may) (VP (V go) (AP (ADV now)))))
>>> sentlist6=senttext6.split()
>>> trees6= rd_parser.parse(sentlist6)
>>> treelist6=list(trees6)
>>> for tree in treelist6:
...     print(tree)
...
(S
  (NP (Prop You))
  (VP
    (V came)
    (NP
      (Det to)
      (VP
        (V visit)
        (NP (Prop me) (Det two) (NP (N days) (Prop ago)))))))
```

This grammar is very limited. Because of that are only very few sentences that can be made. I tried making sentences that didn't make sense, but they could not be passed by this grammar. I think one of the reasons is that this grammar is very strict. Like mentioned above there are very set rules and the terminal options are very restricted.



```
  (VP
    (V came)
    (NP
      (Det to)
      (VP
        (V visit)
        (NP (Prop me) (Det two) (NP (N days) (Prop ago)))))))
>>> senttext5="She may go now"
>>> senttext6="You had a nice party yesterday"
>>> senttext6="You came to visit me two days ago"
>>> sentlist5=senttext5.split()
>>> rd_parser = nltk.RecursiveDescentParser(groucho_grammar)
>>> trees5= rd_parser.parse(sentlist5)
>>> treelist5=list(trees5)
>>> for tree in treelist5:
...     print(tree)
...
(S (NP (Prop She)) (VP (V may) (VP (V go) (AP (ADV now)))))
>>> sentlist6=senttext5.split()
>>> trees6= rd_parser.parse(sentlist6)
>>> treelist6=list(trees6)
>>> for tree in treelist6:
...     print (tree)
...
(S (NP (Prop She)) (VP (V may) (VP (V go) (AP (ADV now)))))
>>> sentlist6=senttext6.split()
>>> trees6= rd_parser.parse(sentlist6)
>>> treelist6=list(trees6)
>>> for tree in treelist6:
...     print(tree)
...
(S
  (NP (Prop You))
  (VP
    (V came)
    (NP
      (Det to)
      (VP
        (V visit)
        (NP (Prop me) (Det two) (NP (N days) (Prop ago)))))))
>>> sentlist7=senttext7.split()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'senttext7' is not defined
>>> _
```

Question3

Assuming that the above four sentences are your mini-mini training corpus, probabilistic context-free grammar is:

grammar = PCFG.fromstring("""

S -> NP VP [1.0]

NP -> Prop [0.45] | Det VP[0.11] | Det ADJ NP[0.11] | Prop Det NP[0.11] | N Prop[0.22]

VP -> V VP [0.15]| V AP [0.28]| Prop VP [0.15] | V NP [0.42]

AP -> ADV AP [0.34] | ADJ [0.33] | ADV [0.33]

ADV -> 'not' [0.34] | 'always' [0.33] | 'now' [0.33]

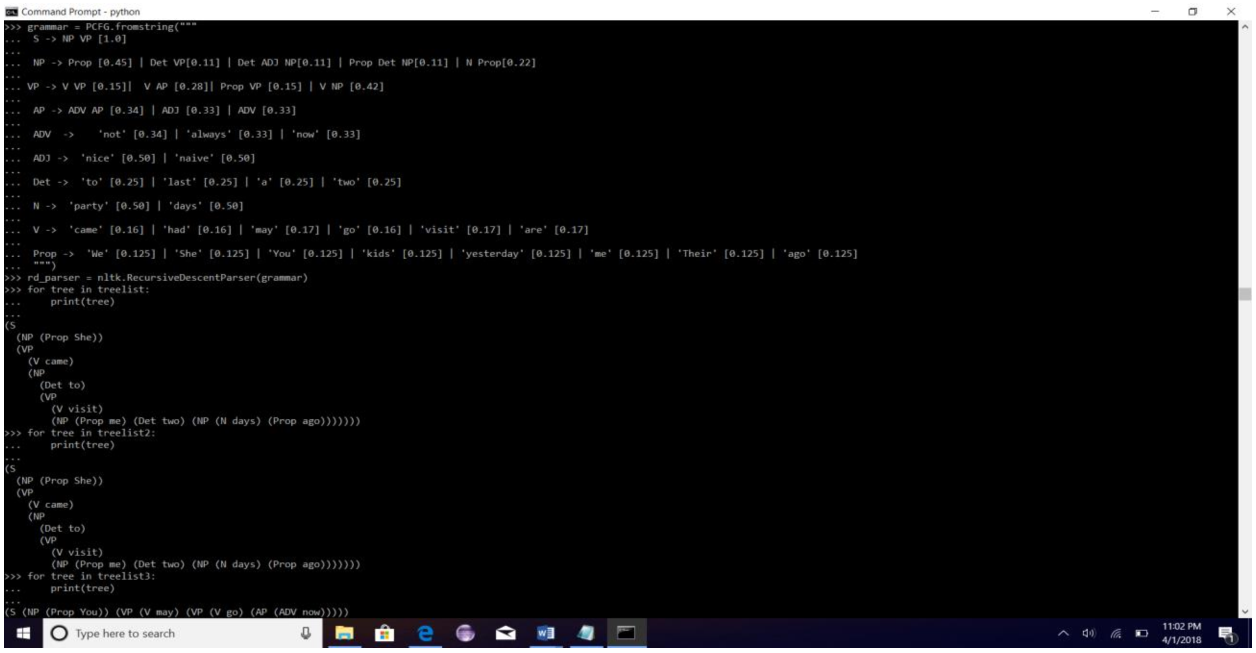
ADJ -> 'nice' [0.50] | 'naive' [0.50]

```
 Det ->  'to' [0.25] | 'last' [0.25] | 'a' [0.25] | 'two' [0.25]


 N ->  'party' [0.50] | 'days' [0.50]


 V ->  'came' [0.16] | 'had' [0.16] | 'may' [0.17] | 'go' [0.16] | 'visit' [0.17] | 'are' [0.17]


 Prop ->  'We' [0.125] | 'She' [0.125] | 'You' [0.125] | 'kids' [0.125] | 'yesterday' [0.125] | 'me' [0.125] |
'Their' [0.125] | 'ago' [0.125]
 """)
>>> rd_parser = nltk.RecursiveDescentParser(grammar)
```

TESTING IS LIKE CFG but here we use RecursiveDescentParser

```
>>> for tree in treelist2:
...     print(tree)
...
(S
  (NP (Prop She))
  (VP
    (V came)
    (NP
      (Det to)
      (VP
        (V visit)
        (NP (Prop me) (Det two) (NP (N days) (Prop ago)))))))
>>> for tree in treelist3:
...     print(tree)
...
(S (NP (Prop You)) (VP (V may) (VP (V go) (AP (ADV now)))))


>>> for tree in treelist4:
...     print(tree)
```

```
...
(S
  (NP (Prop Their))
  (VP
    (Prop kids)
    (VP (V are) (AP (ADV not) (AP (ADV always) (AP (ADJ naive))))))
```



Command Prompt - python

```
>>> grammar = PCFG.fromstring("""
...  S -> NP VP [1.0]
...
...  NP -> Prop [0.45] | Det VP[0.11] | Det ADJ NP[0.11] | Prop Det NP[0.11] | N Prop[0.22]
...
... VP -> V VP [0.15]|  V AP [0.28]| Prop VP [0.15] | V NP [0.42]
...
...  AP -> ADV AP [0.34] | ADJ [0.33] | ADV [0.33]
...
...  ADV  ->    'not' [0.34] | 'always' [0.33] | 'now' [0.33]
...
...  ADJ ->  'nice' [0.50] | 'naive' [0.50]
...
...  Det ->  'to' [0.25] | 'last' [0.25] | 'a' [0.25] | 'two' [0.25]
...
...  N ->  'party' [0.50] | 'days' [0.50]
...
...  V ->  'came' [0.16] | 'had' [0.16] | 'may' [0.17] | 'go' [0.16] | 'visit' [0.17] | 'are' [0.17]
...
...  Prop ->  'We' [0.125] | 'She' [0.125] | 'You' [0.125] | 'kids' [0.125] | 'yesterday' [0.125] | 'me' [0.125] | 'Their' [0.125] | 'ago' [0.125]
...  """)
>>> rd_parser = nltk.RecursiveDescentParser(grammar)
>>> for tree in treelist:
...     print(tree)
...
(S
  (NP (Prop She))
  (VP
    (V came)
    (NP
      (Det to)
      (VP
        (V visit)
        (NP (Prop me) (Det two) (NP (N days) (Prop ago)))))))
>>> for tree in treelist2:
...     print(tree)
...
(S
  (NP (Prop She))
  (VP
    (V came)
    (NP
      (Det to)
      (VP
        (V visit)
        (NP (Prop me) (Det two) (NP (N days) (Prop ago)))))))
>>> for tree in treelist3:
...     print(tree)
...
(S (NP (Prop You)) (VP (V may) (VP (V go) (AP (ADV now)))))
```