NLP With Python:Semester-end Question Paper:Solutions

December 1, 2018

- [x] **Munendra Kumar**
- [x] **HOOMACL20170010**
- [x] **M.A. Computational Linguistics**
- [x] **Semester 3**
- [x] **LS176: NLP with Python**
- [x] **Course Instructor: Dr. Atreyee Sharma**

0.0.1 Section 1: Question 7: Define several variables containing lists of words, e.g., phrase1, phrase2, and so on. Join them together in various combinations (using the plus operator) to form whole sentences. What is the relationship between len(phrase1 + phrase2) and len(phrase1) + len(phrase2)?

0.0.2 Solution:

```
In [25]: phrase1=(['Hello','there!'])
         phrase2=(['Here','are','my','details','...'])
         phrase3=['Roll','No','---->','HOOMACL20170010']
         phrase4=[ 'Name' '---->' 'Munendra','.']
         phrase5=['Programme','---->','MA','Computational','Linguistics']
         phrase6=['Course','Code','---->','LS176']
         phrase7=['Course','Name','---->','NLP','with','Python']
         phrase8=['Course','Instructor','---->' ,'Dr.','Atreyee','Sharma']
```

Joining strings while ignoring separators.

```
In [26]: ''.join(phrase1+phrase2+phrase3+phrase4+phrase5+phrase6+phrase7+phrase8)

Out[26]: 'Hellothere!Herearemydetails...RollNo---->HOOMACL20170010Name---->Munendra.Programme--
```

Joining strings while maintaining whitespace separator.

```
In [27]: ' '.join(phrase1+phrase2+phrase3+phrase4+phrase5+phrase6+phrase7+phrase8)

Out[27]: 'Hello there! Here are my details ... Roll No ----> HOOMACL20170010 Name---->Munendra
```

Joining strings while maintaining any other separator eg. ',

```
In [28]: ','.join(phrase1+phrase2+phrase3+phrase4+phrase5+phrase6+phrase7+phrase8)

Out[28]: 'Hello,there!,Here,are,my,details,...,Roll,No,---->,HOOMACL20170010,Name---->Munendra

In [29]: #### Calculating length

In [30]: len(phrase1)+len(phrase2)+len(phrase3)+len(phrase4)+len(phrase5)+len(phrase6)+len(phra
          #calculating length of the phrases

Out[30]: 34

In [31]: len(phrase1+phrase2+phrase3+phrase4+phrase5+phrase6+phrase7+phrase8) #calculating leng

Out[31]: 34
```

The Function len(phrase1)+len(phrase2)+len(phrase3)... is redundant and repetitive instead function len(phrase1+phrase2+phrase3) is more convenient. It is like calling 'len' function just once instead of calling it 'n' times.

0.0.3 Section 2: Question 7: Write a function that finds the 50 most frequently occurring words of a text that are not stopwords.

```
In [32]: import nltk
         nltk.corpus.gutenberg.fileids() # accessing gutenberg corpus

Out[32]: ['austen-emma.txt',
          'austen-persuasion.txt',
          'austen-sense.txt',
          'bible-kjv.txt',
          'blake-poems.txt',
          'bryant-stories.txt',
          'burgess-busterbrown.txt',
          'carroll-alice.txt',
          'chesterton-ball.txt',
          'chesterton-brown.txt',
          'chesterton-thursday.txt',
          'edgeworth-parents.txt',
          'melville-moby_dick.txt',
          'milton-paradise.txt',
          'shakespeare-caesar.txt',
          'shakespeare-hamlet.txt',
          'shakespeare-macbeth.txt',
          'whitman-leaves.txt']

In [33]: from nltk.corpus import gutenberg #importing dependencies
         from nltk.corpus import stopwords
         from nltk.tokenize import sent_tokenize, word_tokenize
```

```
In [34]: text=nltk.data.load('/home/munendra/nltk_data/corpora/gutenberg/whitman-leaves.txt')
         #importing text from local disk

In [35]: print(text[:20])
         # printing range [0-20] of the corpus

[Leaves of Grass by


In [36]: tokens = word_tokenize(text)
         type(tokens)
         len(tokens)

Out[36]: 149203

In [37]: print(tokens[:10])

['[', 'Leaves', 'of', 'Grass', 'by', 'Walt', 'Whitman', '1855', ']', 'Come']


In [38]: import string
         tokens = [w.lower() for w in tokens]
         table = str.maketrans('', '', string.punctuation)
         stripped = [w.translate(table) for w in tokens]
         from nltk.corpus import stopwords
         words = [word for word in tokens if word.isalpha()]
         print(words[:10])
         stop_words = stopwords.words('english')
         my_stopwords=(['i','the','and','to','thy','you','in','of','thee','thou','us','these',
         words = [w for w in words if not w in stop_words]
         words = [w for w in words if not w in my_stopwords]
         print(words[:10])

         import string
         table = str.maketrans('', '', string.punctuation)
         stripped = [w.translate(table) for w in words]

['leaves', 'of', 'grass', 'by', 'walt', 'whitman', 'come', 'said', 'my', 'soul']
['leaves', 'grass', 'walt', 'whitman', 'come', 'said', 'soul', 'verses', 'body', 'let']


In [39]: import matplotlib.pyplot as plt # importing dependencies
         import pandas as pd
         from collections import Counter # importing counter
         Counter = Counter(words)
         most_occur = Counter.most_common()
         print(most_occur[:50]) # Printing 50 most occuring words

[('see', 432), ('one', 345), ('old', 275), ('shall', 263), ('yet', 262), ('love', 258), ('life
```

0.0.4 Section 3: Question 3: Take any string and perform all the operations and justify your answers by giving reasons? (check section 3.2 from the book)

First string input and calling string

```
In [40]: str1= 'Record 75% voting in Madhya Pradesh.'
         print (str1) #printing first string
```

```
Record 75% voting in Madhya Pradesh.
```

Second string input and calling string

```
In [41]: str2="A record 75 per cent voter turnout for the Madhya Pradesh's state Assembly polls
         print (str2)
         # printing second string
```

```
A record 75 per cent voter turnout for the Madhya Pradesh's state Assembly polls that was marre
```

Escaping a character i.e. intendending

```
In [42]: str2="A record 75 per cent voter turnout for the Madhya Pradesh\'s state Assembly poll
         print (str2)
         # printing a escaped character ''s' as in Madhya Pradesh's
```

```
A record 75 per cent voter turnout for the Madhya Pradesh's state Assembly polls that was marre
```

```
In [43]: str2="A record 75 per cent voter turnout for the Madhya Pradesh's "\
         "state Assembly polls that was marred by complaints "\
         "of faulty EVMs with the BJP looking for a fourth "\
         "straight term in a tough battle with a resurgent Congress "\
         "which is eyeing a comeback after 15 years."
         print (str2)
         # printing string in new line to avoid the problem above
         # where whole string is in one line.
         # but this doesn't give the output in the new line.
```

```
A record 75 per cent voter turnout for the Madhya Pradesh's state Assembly polls that was marre
```

```
In [44]: str2="""A record 75 per cent voter turnout for the Madhya Pradesh's
         state Assembly polls that was marred by complaints
         of faulty EVMs with the BJP looking for a fourth
         straight term in a tough battle with a resurgent Congress
         which is eyeing a comeback after 15 years."""
         print (str2)
         # printing string in the user formatted new line
         #using ('''or """) triple quoted strings
```

```
A record 75 per cent voter turnout for the Madhya Pradesh's
state Assembly polls that was marred by complaints
of faulty EVMs with the BJP looking for a fourth
straight term in a tough battle with a resurgent Congress
which is eyeing a comeback after 15 years.
```

```
In [45]: str2='''A record 75 per cent voter turnout for the Madhya Pradesh's
         state Assembly polls that was marred by complaints
         of faulty EVMs with the BJP looking for a fourth
         straight term in a tough battle with a resurgent Congress
         which is eyeing a comeback after 15 years.'''
         print (str2*2+str1*5)

         # printing combination of str2 twice
         # and combining str1 with str2 five times.
```

```
A record 75 per cent voter turnout for the Madhya Pradesh's
state Assembly polls that was marred by complaints
of faulty EVMs with the BJP looking for a fourth
straight term in a tough battle with a resurgent Congress
which is eyeing a comeback after 15 years.A record 75 per cent voter turnout for the Madhya Pra
state Assembly polls that was marred by complaints
of faulty EVMs with the BJP looking for a fourth
straight term in a tough battle with a resurgent Congress
which is eyeing a comeback after 15 years.Record 75% voting in Madhya Pradesh.Record 75% votin
```

```
In [46]: print(str1+str2)
         # printing combination of str1 and str2
```

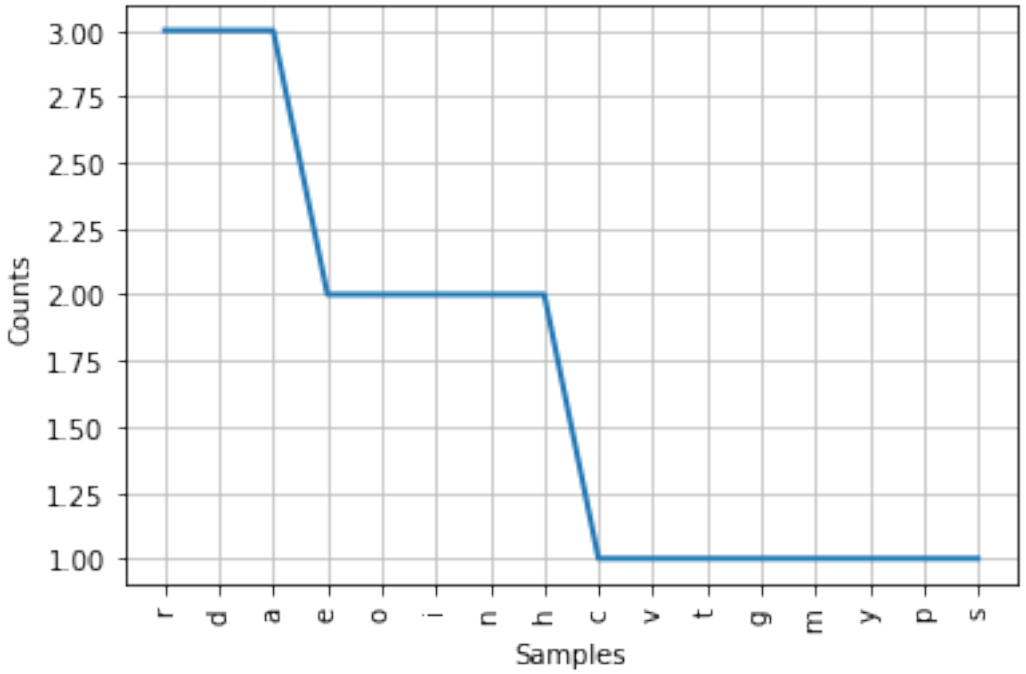
```
Record 75% voting in Madhya Pradesh.A record 75 per cent voter turnout for the Madhya Pradesh'
state Assembly polls that was marred by complaints
of faulty EVMs with the BJP looking for a fourth
straight term in a tough battle with a resurgent Congress
which is eyeing a comeback after 15 years.
```

```
In [47]: # Printing individual characters
         print(str1[0]) # printing character at index '0' in a string
         print(str1[1])# printing character at index '1' in a string
         print(str1[2])# printing character at index '2' in a string
         print(str1[3])# printing character at index '3' in a string
         print(str1[4])# printing character at index '4' in a string
         print(str1[5])# printing character at index '5' in a string

         # Printing a range of characters
         print(str1[0:2]) # printing range '0-2' characters in a string
         print(str1[0:10]) # printing range '0-10' characters in a string
```

```
        fdist = nltk.FreqDist(ch.lower() for ch in str1 if ch.isalpha()) #counter for charact
        print (fdist.keys()) # printing most frequent characters from a string
        fdist.plot() # visualising most frequent characters from string
```

```
R
e
c
o
r
d
Re
Record 75%
dict_keys(['r', 'e', 'c', 'o', 'd', 'v', 't', 'i', 'n', 'g', 'm', 'a', 'h', 'y', 'p', 's'])
```




```
In [48]: len (str1)
```

```
Out[48]: 36
```

```
In [49]: print(str1[-36:-1])
```

```
Record 75% voting in Madhya Pradesh
```

```
In [50]: for char in str1:
             print(char)

R
e
c
o
r
d

7
5
%

v
o
t
i
n
g

i
n

M
a
d
h
y
a

P
r
a
d
e
s
h
.


In [51]: str1.find('e')
         # finds index of first instance of string 'e' in str1

Out[51]: 1

In [52]: str1.rfind('e')
         # finds index of last instance of string 'e' in str1
```

```
Out[52]: 32

In [53]: str1.index('e')
         # finds index of first instance of string 'e' in str1

Out[53]: 1

In [54]: str1.rindex('e')
         # finds index of last instance of string 'e' in str1

Out[54]: 32

In [55]: str4=''
         for i in str1 :
             str4=str1.split()

         #storing string as a list

In [56]: print (str4) # printing list

['Record', '75%', 'voting', 'in', 'Madhya', 'Pradesh.']


In [57]: print(str2.splitlines()) # spliting lines

["A record 75 per cent voter turnout for the Madhya Pradesh's", 'state Assembly polls that was


In [58]: print(str2.lower())  # printing string in lower case

a record 75 per cent voter turnout for the madhya pradesh's
state assembly polls that was marred by complaints
of faulty evms with the bjp looking for a fourth
straight term in a tough battle with a resurgent congress
which is eyeing a comeback after 15 years.


In [59]: print(str2.upper())
         # printing string in upper case

A RECORD 75 PER CENT VOTER TURNOUT FOR THE MADHYA PRADESH'S
STATE ASSEMBLY POLLS THAT WAS MARRED BY COMPLAINTS
OF FAULTY EVMS WITH THE BJP LOOKING FOR A FOURTH
STRAIGHT TERM IN A TOUGH BATTLE WITH A RESURGENT CONGRESS
WHICH IS EYEING A COMEBACK AFTER 15 YEARS.


In [60]: print(str2.title())
         # printing string in title case
```

```
A Record 75 Per Cent Voter Turnout For The Madhya Pradesh'S
State Assembly Polls That Was Marred By Complaints
Of Faulty Evms With The Bjp Looking For A Fourth
Straight Term In A Tough Battle With A Resurgent Congress
Which Is Eyeing A Comeback After 15 Years.
```

```
In [61]: print(str2.strip())
         #removing extra spaces if any using strip function
```

```
A record 75 per cent voter turnout for the Madhya Pradesh's
state Assembly polls that was marred by complaints
of faulty EVMs with the BJP looking for a fourth
straight term in a tough battle with a resurgent Congress
which is eyeing a comeback after 15 years.
```

```
In [62]: print(str2.replace('record','huge record of'))
         # using replace function to replace 'record' with 'huge record of'
```

```
A huge record of 75 per cent voter turnout for the Madhya Pradesh's
state Assembly polls that was marred by complaints
of faulty EVMs with the BJP looking for a fourth
straight term in a tough battle with a resurgent Congress
which is eyeing a comeback after 15 years.
```

Difference between strings and Lists

```
In [63]: print (str1) # this is a string
         print (str4) # this is a list
```

```
Record 75% voting in Madhya Pradesh.
['Record', '75%', 'voting', 'in', 'Madhya', 'Pradesh.']
```

```
In [64]: print(str1[0]) # at index 0 string output
         print(str4[0]) # at index 0 list output
```

```
R
Record
```

```
In [65]: print(str1[:6]) # from range 0-6 string output
         print(str4[:6]) # from range 0-6 list output
```

```
Record
['Record', '75%', 'voting', 'in', 'Madhya', 'Pradesh.']
```

```
In [66]: print(len(str1)) # length of the string
         print(len(str4)) # length of the list
```

```
36
6
```

So it is clear from above that strings store every single character including whitespaces as a string. While lists store every word excluding whitespaces