ЛАБОРАТОРНАЯ РАБОТА 1

«Введение в Python»

Python - популярный язык программирования общего назначения, которые используют для научных вычислений и анализа данных.

На данные момент активно используются две «ветки» языка: Python 2 и Python 3. Это связано с тем, что около 2008 года авторы языка решили очистить Python 2 от ошибочных архитектурных решений. К сожалению, для этого пришлось отказаться от обратной совместимости, то есть во многих случаях код на Python 2 нужно модифицировать, чтобы запустить его на **Python 3**.

Версия Python 2 считается устаревшей, и больше не дорабатывается. На данный момент большинство популярных пакетов (включая NumPy, SciPy и scikit-learn) перенесены на Python 3, и как правило лучше использовать Python 3.

Самый простой способ – воспользоваться дистрибутивом Python, который содержит интерпретатор и большое число модулей. Для выполнения лабораторных работ рекомендуется бесплатный дистрибутив **Anaconda**. Среда разработки – **Spyder** (входит в Anaconda).

Цель лабораторной работы – это приобрести начальные навыки работы с языком программирования Python, методами работы с текстом и со строками.

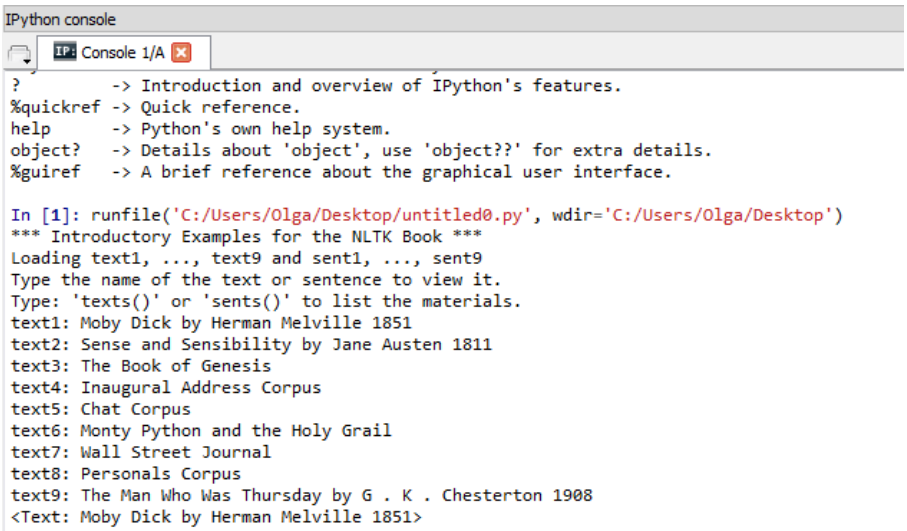
Использование Python для автоматизированной обработки естественного языка

Ход работы:

1. Загрузим тексты (книги) для работы из библиотеки *nltk.* И выведем список текстов (книг).

   *from nltk.book import **

   print text1



2. Поиск в текстах. Найдем тексты, в которых встречается слово «monstrous».
   print text1.concordance("monstrous")

```
In [2]: runfile('C:/Users/Olga/Desktop/untitled0.py', wdir='C:/Users/Olga/Desktop')
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some were thick
d as you gazed , and wondered what monstrous cannibal and savage could ever hav
that has survived the flood ; most monstrous and most mountainous ! That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse and more de
th of Radney .'" CHAPTER 55 Of the Monstrous Pictures of Whales . I shall ere l
ing Scenes . In connexion with the monstrous pictures of whales , I am strongly
ere to enter upon those still more monstrous stories of them which are to be fo
ght have been rummaged out of this monstrous cabinet there is no telling . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast up dead u
None
```

3. Определим, какие еще слова встречаются в одном контексте вместе со словом *monstrous* с помощью функции *similar.* С помощью функции *common_contexts* можно исследовать контексты, в которых встречаются оба эти слова.

   print text1
   print text2
   *print "**********"*
   print text1.similar("monstrous")
   *print "**********"*
   print text2.similar("monstrous")
   *print "**********"*
   print text2.common_contexts(["monstrous", "very"])

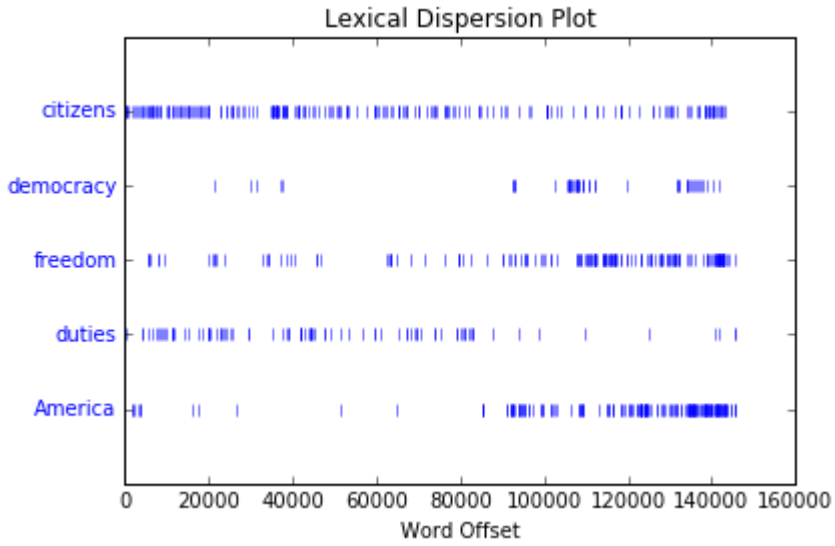
```
In [7]: runfile('C:/Users/Olga/Desktop/untitled0.py', wdir='C:/Users/Olga/Desktop')
<Text: Moby Dick by Herman Melville 1851>
<Text: Sense and Sensibility by Jane Austen 1811>
**********
imperial subtly impalpable pitiable curious abundant perilous
trustworthy untoward singular lamentable few determined maddens
horrible tyrannical lazy mystifying christian exasperate
None
**********
very exceedingly so heartily a great good amazingly as sweet
remarkably extremely vast
None
**********
a_pretty is_pretty a_lucky am_glad be_glad
None
```

4. С помощью метода dispersion_plot

   text4.dispersion_plot(["citizens", "democracy", "freedom", "duties", "America"])

можно узнать как часто то или иное слово появлялось в этом корпусе. Каждая вертикальная линия на графике представляет случайные слова и каждый горизонтальный ряд представляют весь текст. На рисунке ниже представлены слова за прошлые 220 лет

(на примере искусственного текста, построенного на основе Inaugural Address Corpus end-to-end, иннаугурационной речи).



5. С помощью метода *len* можно узнать длину текста от начала до конца, включая слова и пунктуацию.

   print len(text3)
   44764

6. Для того, чтобы получить все токены и затем отсортировать их используются следующие команды:

   print set(text3)
   t3 = sorted(set(text3))
   print t3
   print len(set(text3))

   Заметьте токены выводятся без повторений!
   Таким образом, мы можем посчитать среднее появление слов в этом тексте:

   print len(text3) / len(set(text3))
   16.0501972033

7. Для подсчета, сколько раз слово встретилось в тексте, используется функция *count*.

   print text3.count("smote")

Подсчитайте для своего текста, с какой вероятностью это слово встретилось в тексте.
Посчитайте это 2 способами с использованием функции и без нее.

```
def lexical_diversity(text):
... return len(text) / len(set(text))
...
 def percentage(count, total):
... return 100 * count / total
```

Тоже самое можно получить используя встроенные функции. Это такие функции как *lexical_diversity(), percentage().* Примеры:

```
lexical_diversity(text3)
        16.050197203298673
lexical_diversity(text5)
        7.4200461589185629
percentage(4, 5)
        80.0
percentage(text4.count('a'), len(text4))
        1.4643016433938312
```

Во многих лингвистических моделях текст рассматривается как набор слов. Рассмотрим, какие возможности предоставляет Python для этого.

```
 print sent1 # sent1 # это первое предложение в тексте Моби Дик
        ['Call', 'me', 'Ishmael', '.']
 print len(sent1)
        4
lexical_diversity(sent1)
        1.0
```

Мы можем соединить наборы слово с помощью оператора +:

```
str1 = [‘holly’, ‘xmas’, ‘bell’, ‘tree’]
str2 = [‘egg’, ‘rabbit’, ‘Easter’, ‘chocolate’]
print str1+str2
print sent4 + sent1
sent1.append("Some")
print sent1
```

С помощью индекса мы можем обратиться к любому слову нашей коллекции. Например:

```
print text4[173]
        'awaken'
print text4.index('awaken')
        173
```

Мы можем также выделить диапазон из коллекции. Например:

```
print text5[16715:16735]
        ['U86', 'thats', 'why', 'something', 'like', 'gamefly', 'is', 'so', 'good',
        'because', 'you', 'can', 'actually', 'play', 'a', 'full', 'game', 'without',
        'buying', 'it']

print text6[1600:1625]
        ['We', "'", 're', 'an', 'anarcho', '-', 'syndicalist', 'commune', '.', 'We',
        'take', 'it', 'in', 'turns', 'to', 'act', 'as', 'a', 'sort', 'of', 'executive',
        'officer', 'for', 'the', 'week']
```

Для того, чтобы отсортировать нашу коллекцию необходимо использовать метод *sorted().* Например: *print sorted(text6)*

Необходимо помнить. Что слова написанные с большой буквы будет идти первыми, чем слова с маленькой буквы.

Если вам необходимо список слов вновь склеить в единую строку, то для этого используется команда *join().* Например:

```
print s =' '.join(['Monty', 'Python'])
        'Monty Python'
print s.split()
        ['Monty', 'Python']
```

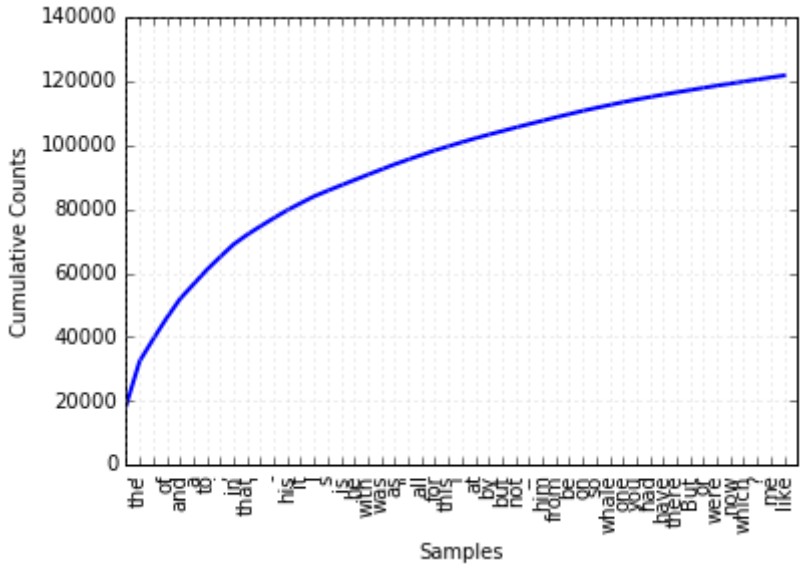
С помощью функции FreqDist() можно найти самые частотный слова для документа. Например:

```
fdist1 = FreqDist(text1)
print fdist1
        <FreqDist with 260819 outcomes>
vocabulary1 = fdist1.keys()
print vocabulary1[:50]
        [',', 'the', '.', 'of', 'and', 'a', 'to', ';', 'in', 'that', "'", '-',
        'his', 'it', 'I', 's', 'is', 'he', 'with', 'was', 'as', '"', 'all', 'for',
        'this', '!', 'at', 'by', 'but', 'not', '--', 'him', 'from', 'be', 'on',
        'so', 'whale', 'one', 'you', 'had', 'have', 'there', 'But', 'or', 'were',
        'now', 'which', '?', 'me', 'like']
print fdist1['whale']
```

906

Построим график для первых 50 самых частотных слов:

print fdist1.plot(50, cumulative=True)



Метод *hapaxes()* возвращает слова, которые встретились в тексте только 1 раз.

Если нас интересуют слова, длина которых больше 15 символов, то для этого необходимо написать следующий код:

```
V = set(text1)
long_words = [w for w in V if len(w) > 15]
sorted(long_words)
    ['CIRCUMNAVIGATION', 'Physiognomically', 'apprehensiveness',
    'cannibalistically', 'characteristically', 'circumnavigating',
    'circumnavigation', 'circumnavigations', 'comprehensiveness',
    'hermaphroditical', 'indiscriminately', 'indispensableness', 'irresistibleness',
    'physiognomically', 'preternaturalness', 'responsibilities',
    'simultaneousness', 'subterraneousness', 'supernaturalness',
    'superstitiousness', 'uncomfortableness', 'uncompromisedness',
    'undiscriminating', 'uninterpenetratingly']
```

Коллокацией называется словосочетание, имеющее признаки синтаксически и семантически целостной единицы, в котором выбор одного из компонентов осуществляется по смыслу, а выбор второго зависит от выбора первого (например, ставить условия — выбор глагола ставить определяется традицией и зависит от существительного условия, при слове предложение будет другой глагол — вносить). (Википедия)

Для того, чтобы найти коллокации мы вначале попробуем найти так называемые биграммы с помощью функции *bigrams():*

```
print  bigrams(['more', 'is', 'said', 'than', 'done'])
    [('more', 'is'), ('is', 'said'), ('said', 'than'), ('than', 'done')]
```

Однако, обычно хочется извлекать не просто биграммы, а именно коллокации – частотные слова, которые в тексте обычно встречаются вместе. Для этого в Python удобно использовать функцию *collocations():*

```
print text4.collocations()
    Building collocations list
    United States; fellow citizens; years ago; Federal Government; General
    Government; American people; Vice President; Almighty God; Fellow
    citizens; Chief Magistrate; Chief Justice; God bless; Indian tribes; public
    debt; foreign nations; political parties; State governments; National
    Government; United Nations; public money
print text8.collocations()
    Building collocations list
    medium build; social drinker; quiet nights; long term; age open; financially
    secure; fun times; similar interests; Age open; possr ship; single mum;
    permanent relationship; slim build; seeks lady; Late 30s; Photo pls; Vibrant
    personality; European background; ASIAN LADY; country drives
```

Задание на лабораторную работу

1) Реализовать все примеры указные в лабораторной работе на своих произвольных текстах. Тексты можно брать из библиотеки Python.